



K.S.INSTITUTE OF TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

COURSE FILE

NAME OF THE STAFF :DEEPA .S.R
SUBJECT CODE/NAME :18CS34/COMPUTER ORGANIZATION
SEMESTER/YEAR : III Sem/II Year A Section
ACADEMIC YEAR : 2020-21
BRANCH :COMPUTER SCIENCE & ENGINEERING

Deepa

FACULTY IN-CHARGE

Dr. Venkatesh

HOD

Head of the Department
Dept. of Computer Science & Engg.
K.S. Institute of Technology
Bengaluru -560 109



K.S. INSTITUTE OF TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

Vision of the Institute

To impart quality technical education with ethical values, employable skills and research to achieve excellence

Mission of the Institute

- To attract and retain highly qualified, experienced & committed faculty.
- To create relevant infrastructure.
- Network with industry & premier institutions to encourage emergence of new ideas by providing research & development facilities to strive for academic excellence.
- To inculcate the professional & ethical values among young students with employable skills & knowledge acquired to transform the society.

Vision of the Department

To create competent professionals in Computer Science and Engineering with adequate skills to drive the IT industry

Mission of the Department

- Impart sound technical knowledge and quest for continuous learning.
- To equip students to furnish Computer Applications for the society through experiential learning and research with professional ethics.
- Encourage team work through inter-disciplinary project and evolve as leaders with social concerns.



Head of the Department
Dept. of Computer Science & Engg.
K.S. Institute of Technology
Bengaluru -560 109



K.S. INSTITUTE OF TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

Program Educational Objectives

- PEO1:** Excel in professional career by acquiring knowledge in cutting edge technology and contribute to the society as an excellent employee or as an entrepreneur in the field of Computer Science & Engineering.
- PEO2:** Continuously enhance their knowledge on par with the development in IT industry and pursue higher studies in Computer Science & Engineering.
- PEO3:** Exhibit professionalism, cultural awareness, team work, ethics, and effective communication skills with their knowledge in solving social and environmental problems by applying computer technology.

Program Specific Outcomes (PSO)

- PSO1:** Ability to understand, analyze problems and implement solutions in programming languages, as well to apply concepts in core areas of Computer Science in association with professional bodies and clubs.
- PSO2:** Ability to use computational skills and apply software knowledge to develop effective solutions and data to address real world challenges.

D. Venkatesh

Head of the Department
Dept. of Computer Science & Engg.
K.S. Institute of Technology
Bengaluru - 560 100



K.S. INSTITUTE OF TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

Program Outcomes

- PO1: Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
- PO2: Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
- PO3: Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
- PO4: Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
- PO5: Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
- PO6: The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
- PO7: Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

- PO8: Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
- PO9: Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
- PO10: Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
- PO11: Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
- PO12: Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.



Head of the Department
Dept. of Computer Science & Engg.
K.S. Institute of Technology
Bengaluru - 560 109



K. S. INSTITUTE OF TECHNOLOGY

#14, Raghuvanahalli, Kanakapura Main Road, Bengaluru-5600109

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Course: COMPUTER ORGANIZATION			
Type: Core		Course Code:18CS34	Academic Year:2020-21
Name: Deepa .S.R			Sem & Sec: III A
No of Hours per week			
Theory (Lecture Class)	Practical/Field Work/Allied Activities	Total/Week	Total teaching hours
4	0	4	40
Marks			
Internal Assessment	Examination	Total	Credits
40	60	100	3
<u>Aim/Objective of the Course:</u>			
This course enables students to:			
<ul style="list-style-type: none">• Explain the basic sub systems of a computer, their organization, structure and operation. Illustrate the concept of programs as sequences of machine instructions.• Demonstrate different ways of communicating with I/O devices and standard I/O interfaces. Describe memory hierarchy and concept of virtual memory.• Describe arithmetic and logical operations with integer and floating-point			
Course Learning Outcomes:			
After completing the course, the students will be able to,			
CO#	COURSE OUTCOMES		K-LEVEL
18CS34.1	Construct architecture & organization of a computer system, machine instruction formats and addressing modes.		Applying (K3)
18CS34.2	Build techniques for I/O communication with standard bus interfaces and interrupt service routines.		Applying (K3)
18CS34.3	Identify different memories and memory mapping techniques.		Applying (K3)
18CS34.4	Design different Arithmetic operational units.		Applying (K3)

		(K3)
18CS34.5	Derive control sequences for hardwired and micro-program control units for both single and multi bus processors.	Applying (K3)
Syllabus Content:		
Module 1 Basic Structure of Computers: Basic Operational Concepts, Bus Structures, Performance – Processor Clock, Basic Performance Equation, Clock Rate, Performance Measurement. Machine Instructions and Programs: Memory Location and Addresses, Memory Operations, Instructions and Instruction Sequencing, Addressing Modes, Assembly Language, Basic Input and Output Operations, Stacks and Queues, Subroutines, Additional Instructions, Encoding of Machine Instructions LO: At the end of this session the student will be able to, <ol style="list-style-type: none"> 1. Explain the structure of computer, organizational structure and operation 2. Illustrate the concept of programs as sequences of machine instructions and basic I/O operations 3. Understand addressing modes, stacks and queues. 		C01 8 hrs PO1-3 PO2-2 PO3-2 PS01-3 PS02-2
Module 2 Input/Output Organization: Accessing I/O Devices, Interrupts – Interrupt Hardware, Direct Memory Access, Buses, Interface Circuits, Standard I/O Interfaces – PCI Bus, SCSI Bus, USB. LO: At the end of this session the student will be able to, <ol style="list-style-type: none"> 1. Demonstrate different ways of communicating with I/O devices 2. Understand standard I/O Interfaces 		C02 8 hrs. PO1-3 PO2-2 PO3-2 PS01-3 PS02-2
Module 3: Memory System: Basic Concepts, Semiconductor RAM Memories, Read Only Memories, Speed, Size and Cost, Cache Memories – Mapping Functions and Performance Considerations. LO: At the end of this session the student will be able to, <ol style="list-style-type: none"> 1. Describe memory hierarchy 2. Compare different types of memories 3. Explain the concept of virtual memory 		C03 8 hrs PO1-3 PO2-2 PO3-3 PS01-3 PS02-2
Module 4: Arithmetic: Numbers, Arithmetic Operations and Characters, Addition and Subtraction of Signed Numbers, Design of Fast Adders, Multiplication of Positive Numbers, Signed Operand Multiplication, Fast Multiplication, Integer Division.		C04 8 hrs PO1-3

<p>LO: At the end of this session the student will be able to,</p> <ol style="list-style-type: none"> 1. Describe operations with integer operands 	<p>P02-2 P03-3 PS01-3 PS02-2</p>
<p><u>Module 5:</u></p> <p>Basic Processing Unit: Some Fundamental Concepts, Execution of a Complete Instruction, Multiple Bus Organization, Hard-wired Control, Micro programmed Control. Pipelining: Basic concepts of pipelining,</p> <p>LO: At the end of this session the student will be able to,</p> <ol style="list-style-type: none"> 1. Illustrate organization of a simple processor 2. Understand basic concept of pipelining 	<p>C05 8 hrs</p> <p>P01-3 P02-2 P03-2 PS01-3 PS02-2</p>
<p>Text Books: -</p> <p>1. Carl Hamacher, Zvonko Vranesic, Safwat Zaky, Computer Organization, 5th Edition, Tata McGraw Hill, 2002.</p>	
<p>Reference Books:</p> <p>1. William Stallings: Computer Organization & Architecture, 9th edition, Pearson, 2015</p>	
<p>Useful Journals/ Links</p> <p>http://nptel.vtu.ac.in/econtent/courses/CSE/06CS46/index.php</p>	
<p>Teaching and Learning Methods:</p> <ol style="list-style-type: none"> 1. Lecture class: 60 hrs. 2. Self-study: 3. Field visits/Group Discussions/Seminars: 	
<p>Assessment:</p> <p>Type of test/examination: Written examination</p> <p>Continuous Internal Evaluation(CIE) : 40 marks (Average three of tests will be considered)</p> <p>Semester End Exam(SEE) : 60 marks (students have to answer all main questions)</p> <p>Test duration: 1 :30 hr</p> <p>Examination duration: 3 hrs</p>	

PS01:Ability to understand, analyze problems and implement solutions in programming languages, as well as to apply concepts in core areas of computer Science in association with professional bodies and clubs.

PS02: Ability to use computational skills and apply software knowledge to develop effective solutions and data to address real world challenges

CO - PO MAPPING

P01: Science and engineering Knowledge
P02: Problem Analysis
P03: Design & Development
P04:Investigations of Complex Problems
P05: Modern Tool Usage
P06: Engineer & Society

P07:Environment and Society
P08:Ethics
P09:Individual & Team Work
P010: Communication
P011:Project Mngmt & Finance
P012:Life long Learning

Pedagogy quiz is planned in module 1 and 2 for mapping CO1 and CO2 to PO9


CO	P O1	PO 2	P O 3	P O 4	P O5	P O6	P O7	P O8	P O9	PO 10	PO 11	PO 12
18CS34.1	3	2	2						2			
18CS34.2	3	2	2						2			
18CS34.3	3	2	3									
18CS34.4	3	2	3									
18CS34.5	3	2	2									
18CS34	3	2	2.4						2			

CO	PSO1	PSO2
18CS34.1	3	2
18CS34.2	3	2
18CS34.3	3	2
18CS34.4	3	2
18CS34.5	3	2
18CS34	3	2

3	Substantial (High) Correlation
2	Moderate (Medium) Correlation
1	Slight (Low) Correlation
-	No correlation.


Signature of Course in-Charge


Signature of Module Coordinator


Signature of HOD
Head of the Department
Dept. of Computer Science & Engg.
K.S. Institute of Technology
Bengaluru -560 109



K. S INSTITUTE OF TECHNOLOGY, BENGALURU-560109

TENTATIVE CALENDAR OF EVENTS: ODD SEMESTER (2020-2021)

SESSION: SEP 2020 - DEC 2020

Week No.	Month	Day						Days	Activities
		Mon	Tue	Wed	Thu	Fri	Sat		
1	SEP		1*	2	3	4	5	5	1*-Commencement of Higher Semester
2	SEP	7	8	9	10	11	12	6	
3	SEP	14	15	16	17 DH	18	19	5	17- Mahalaya Amavasya
4	SEP	21	22	23	24	25	26TA	6	
5	SEP / OCT	28 T1	29 T1	30 T1	1	2 DH	3	5	2- Mahatma Gandhi Jayanthi
6	OCT	5	6BV	7ASD	8	9	10	6	5-10 First Feed Back
7	OCT	12	13	14	15	16	17 H	6	
8	OCT	19	20	21	22	23	24	6	24 - Monday Time Table
9	OCT	26 DH	27	28	29TA	30 DH	31 DH	3	26- Vijayadashami 30- Eid-Milad 31- Maharishi Valmiki Jayanti
10	NOV	2	3	4	5	6	7	6	7 - Wednesday Time Table
11	NOV	9 T2	10 T2	11 T2	12	13	14 H	6	
12	NOV	16 DH	17 BV	18 ASD	19	20	21	5	16 - Balipadyami Deepavalli 18 - 21 Second Feed Back 21 - Friday Time Table
13	NOV	23	24	25	26	27	28 H	6	
14	NOV /DEC	30	1	2	3 DH	4	5TA	5	3- Kanakadasa Jayanti 5 - Monday Time Table
15	DEC	7	8	9 LT	10 LT	11 LT	12 H	6	
16	DEC	14 T3	15 T3	16 T3	17 * BV			4	17* -Last Working Day
Total No of Working Days : 82									

Total Number of working days (Excluding holidays and Tests)=70

H	Holiday
BV	Blue Book Verification
T1,T2, T3	Tests 1,2, 3
ASD	Attendance & Sessional Display
DH	Declared Holiday
LT	Lab Test
TA	Test attendance

Monday	12
Tuesday	13
Wednesday	13
Thursday	13
Friday	13
Saturday	6
Total	70

Principal
K.S. INSTITUTE OF TECHNOLOGY
BENGALURU - 560 109



K. S INSTITUTE OF TECHNOLOGY, BENGALURU-560109

TENTATIVE CALENDAR OF EVENTS: ODD SEMESTER (2020-2021)

SESSION: SEP 2020 - JAN 2021

Week No.	Month	Day						Days	Activities
		Mon	Tue	Wed	Thu	Fri	Sat		
1	SEP		1*	2	3	4	5	5	1*-Commencement of Higher Semester
2	SEP	7	8	9	10	11	12	6	
3	SEP	14	15	16	17 H	18	19	5	17- Mahalaya Amavasya
4	SEP	21	22	23	24	25	26TA	6	
5	SEP / OCT	28 T1	29 T1	30 T1	1	2 H	3	5	2- Mahatma Gandhi Jayanthi
6	OCT	5	6BV	7ASD	8	9	10	6	5-10 First Feed Back
7	OCT	12	13	14	15	16	17 DH	5	
8	OCT	19	20	21	22	23	24	6	24 - Monday Time Table
9	OCT	26 H	27	28	29	30 H	31 H	3	26- Vijayadashami 30- Eid-Milad 31- Maharishi Valmiki Jayanti
10	NOV	2	3	4	5	6	7 TA	6	7 - Wednesday Time Table
11	NOV	9	10	11	12	13	14 DH	5	
12	NOV	16 H	17 T2	18 T2	19 T2	20	21	5	16 - Balipadyami Deepavalli 18 - 21 Second Feed Back 21 - Friday Time Table
13	NOV	23	24BV	25ASD	26	27	28 DH	5	
14	NOV / DEC	30	1	2	3 H	4	5	5	3- Kanakadasa Jayanti 5 - Monday Time Table
15	DEC	7	8	9	10	11	12 DH	5	
16	DEC	14	15	16	17	18	19	6	19- Monday Time Table
17	DEC	21	22	23	24	25H	26DH	4	25-Christmas
18	DEC/ JAN	28	29	30	31	1 TA	2	6	2Thursday Time Table
19	JAN	4 LT	5 LT	6 LT	7 LT	8	9DH	5	
20	JAN	11 T3	12 T3	13 T3	14H	15	16 *	5	14- Makara sankaranthi 16* -Last Working Day
Total No of Working Days : 106									

Total Number of working days (Excluding holidays and Tests)=87

H	Holiday
BV	Blue Book Verification
T1, T2, T3	Tests 1,2, 3
ASD	Attendance & Sessional Display
DH	Declared Holiday
LT	Lab Test
TA	Test attendance

Monday	16
Tuesday	16
Wednesday	16
Thursday	16
Friday	17
Saturday	6
Total	87

PRINCIPAL
K.S. INSTITUTE OF TECHNOLOGY
BENGALURU - 560 109.

**K. S INSTITUTE OF TECHNOLOGY, BENGALURU-560109**

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CALENDAR OF EVENTS: ODD SEMESTER (2020-2021)

SESSION: SEP 2020 - JAN 2021

KSIT

Week No.	Month	Day						Days	Activities	Department Activities
		Mon	Tue	Wed	Thu	Fri	Sat			
1	SEP		1*	2	3	4	5	5	1*-Commencement of Higher Semester	
2	SEP	7	8	9	10	11	12	6		
3	SEP	14	15	16	17 H	18	19	5	17- Mahalaya	
4	SEP	21	22	23	24	25	26TA	6		
5	SEP / OCT	28 T1	29 T1	30 T1	1	2 H	3	5	2- Mahatma Gandhi Jayanthi	
6	OCT	5	6BV	7ASD	8	9	10	6	5-10 First Feed Back	
7	OCT	12	13	14	15	16	17 DH	5		
8	OCT	19	20	21	22	23	24	6	24 - Monday Time Table	
9	OCT	26 H	27	28	29	30 H	31 H	3	26- Vijayadashami 30- Eid-Milad 31- Maharishi Valmiki Jayanti	Project Zeroth Review Presentation
10	NOV	2	3	4	5	6	7 TA	6	7 - Wednesday Time Table	
11	NOV	9	10	11	12	13	14 DH	5		13-11-2020 Webinar on An Insight into Web Application Development
12	NOV	16 H	17 T2	18 T2	19 T2	20	21	5	16 - Balipadyami Deepavalli 18 - 21 Second Feed Back 21 - Friday Time Table	
13	NOV	23	24BV	25ASD	26	27	28 DH	5		
14	NOV / DEC	30	1	2	3 H	4	5	5	3- Kanakadasa Jayanti 5 - Monday Time Table	Project Zeroth Phase Re-Presentation
15	DEC	7	8	9	10	11	12 DH	5		
16	DEC	14	15	16	17	18	19	6	19- Monday Time	
17	DEC	21	22	23	24	25H	26DH	4	25-Christmas	22-12-2020 & 23-12-2020 34th CSI Karnataka State Student Convention
18	DEC / JAN	28	29	30	31	1 TA	2	6	2Thursday Time Table	Project Phase - 1 Review Presentation
19	JAN	4 LT	5 LT	6 LT	7 LT	8	9 DH	5		
20	JAN	11 T3	12 T3	13 T3	14 H	15	16 *	5	14- Makara sankaranthi	
Total No of Working Days : 106										

Total Number of working days (Excluding holidays and Tests)=87

II	Holiday
BV	Blue Book
T1,T2, T3	Tests 1,2, 3
ASD	Attendance &
DH	Declared Holiday
LT	Lab Test
TA	Test attendance

Monday	16
Tuesday	16
Wednesday	16
Thursday	16
Friday	17
Saturday	6
Total	87

D. Narayana
24/8/2020
Head of the Department
Dept. of Computer Science & Engg.
K.S. Institute of Technology
Bengaluru -560 109

K. .S. INSTITUTE OF TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
III SEM A SECTION STUDENT DETAILS:2020-21

Sl. No	USN	Student Name	Email ID	Mobile Number
1	1KS19CS001	AAKRITI	aaki96798nov@gmail.com	8102100946
2	1KS19CS002	ABHISHEK B	abhibgowda2001@gmail.com	8296520760
3	1KS19CS003	ABHISHEK YADAV	abhishekyadav1998@gmail.com	6394788936
4	1KS19CS004	ADITH KARTHIK RAJU	adithkarthik10@gmail.com	9164875273
5	1KS19CS005	AJAY S KALBURGI	kalburgiajay@gmail.com	7090663930
6	1KS19CS006	AKASH A S	akashlisp@gmail.com	8088125905
7	1KS19CS007	AMAN KUSHWAHA	amankushwaha365@gmail.com	8081239899
8	1KS19CS008	AMILINENI HARINI	amilinenivenkatrshulu924@gmail.com	8919756866
9	1KS19CS009	AMIT K B	baluamit@gmail.com	9483354792
10	1KS19CS010	AMOGHA H S	amogha.hs@gmail.com	9845006222
11	1KS19CS011	AMRUTHA K H	amruthaharshan8347@gmail.com	7204219550
12	1KS19CS012	ANAGHA A HEBBAR	anaghaah01@gmail.com	9148571180
13	1KS19CS014	ANUSHA B	vbabuvbabu222@gmail.com	9342582372
14	1KS19CS015	AQSA AQEEL	aqsa.aqeel17@gmail.com	8707566256
15	1KS19CS016	ASHIKA H N	ashikananjegowda869@gmail.com	7975007345
16	1KS19CS017	BHOOMIKA A M	mohan_iist@yahoo.co.in	9980150041
17	1KS19CS018	BHOOMIKA K	bhoomikak17@gmail.com	9986598304
18	1KS19CS019	BHUMIKA M	ravikumar19721736@gmail.com	9964151736
19	1KS19CS020	C N SHREYAS	sheryascn611@gmail.com	6366306846
20	1KS19CS021	CHAITANYA SHIVARAJU	chaitanyashivaraju@gmail.com	8197730165
21	1KS19CS022	CHANDAN B V	chandanbv9123@gmail.com	7338568147
22	1KS19CS023	DEEKSHA NAIDU R	deeksha12082001@gmail.com	8095122655
23	1KS19CS024	DEEPA G	deepa1012gandhi@gmail.com	9148483322
24	1KS19CS025	DEEPTHI.N K	deepthink400@gmail.com	9148487058
25	1KS19CS026	DEVI PRASAD N	deviprasad4556@gmail.com	7639444564
26	1KS19CS028	DHEEMANTH G	dheemanthgirish03@gmail.com	9108250916
27	1KS19CS029	DINESH M	denishdinu1431@gmail.com	6366055811
28	1KS19CS030	G PRERITHA	prerithagowtham@gmail.com	8197458531
29	1KS19CS031	GAGAN REDDY S	gagan19200@gmail.com	8277884917
30	1KS19CS032	GAGANDEEP K	gagandeepk6549@gmail.com	7619129339
31	1KS19CS033	GEETHANJALI B K PRASAD	geethanjali9840@gmail.com	9632806412
32	1KS19CS034	GULSHAN KUMAR S	kumargulshan2702@gmail.com	8050991314
33	1KS19CS035	HARSHITHA J	harshithaj2306@gmail.com	6360781608
34	1KS19CS036	INDRAJITH H M	indrajithlikith@gmail.com	9972503057
35	1KS19CS037	JEEVAN T O	jjeevanto1227@gmail.com	9353380159
36	1KS19CS038	K KISHAN	kishanuk2002@gmail.com	9686480814
37	1KS19CS039	K R VAGEESH	vageesh2001@gmail.com	9742578725
38	1KS19CS040	KALYAN CHOWDHARY B	munirathnam931@gmail.com	9008838499
39	1KS19CS041	KARTHIK S MORAJKAR	karthik.morajkar26@gmail.com	9113629181
40	1KS19CS042	KAVYASHREE.S.L	kavyashrees1599@gmail.com	9113274206
41	1KS19CS043	KEERTHAN GOWDA S	keerthagowdas222@gmail.com	9845871477
42	1KS19CS044	KOTHAPALLI SREEJA	sreeja16112002@gmail.com	9491465760
43	1KS19CS045	KOTTALA SAIVENKATASUCHITH	suchithkottala@gmail.com	9059089926
44	1KS19CS046	KRISHNA K R	krkrishna2626@gmail.com	7358478255
45	1KS19CS047	KUMAR S	chittihoney225@gmail.com	9591084850
46	1KS19CS048	LIKITH G	likithgk01@gmail.com	9448116636
47	1KS19CS049	LISHA C	lishachoudhary2612@gmail.com	9036831198
48	1KS19CS050	MAHAK SHREE	shreemahak01@gmail.com	9973580148
49	1KS19CS051	MALLIPALLI SPURTHI REDDY	mspurthi49@gmail.com	9440090436
50	1KS19CS052	MANASA G L	manasagl@gmail.com	9901599887
51	1KS19CS053	MOHAMMED NOOR AMAN	mohammednooraman569@gmail.com	9880672867
52	1KS19CS054	MUKESH KUMAR	mukeshkeshri114@gmail.com	7991448599
53	1KS19CS055	MYTHREYI U	shobha.hande8@gmail.com	9741052966
54	1KS19CS056	N ASHOK	ashoknashok132000@gmail.com	8073952496
55	1KS19CS057	N BHAVYA	bhavyanarra56@gmail.com	8790762175
56	1KS19CS058	N P SHASHANKA RAO	shashanknp02@gmail.com	9606573173
57	1KS18CS011	BHARATH R	barathrahul100@gmail.com	7795844959
58	1KS20CS400	AKIF DELVI	-	9845038115


HOD-CSE

Head of the Department
 Dept. of Computer Science & Engg.
 K.S. Institute of Technology
 Bengaluru -560 109



K.S. INSTITUTE OF TECHNOLOGY, BENGALURU-109

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

III SEMESTER ONLINE CLASS TIME TABLE FOR THE YEAR 2020-21 (ODD SEMESTER)(TENTATIVE)

W.E.F: 01-09-2020

CLASS TEACHER: Mr. Sanjoy Das

SEC: 'A'

PERIOD	1	2		3	4		5	6
TIME DAY	09:00 AM-10:00 AM	10:00 AM-11:00 AM	11:00 AM-11:15 AM	11:15 AM-12:15 PM	12:15 PM-01:15 PM	01:15 PM-01:45 PM	01:45 PM-02:45 PM	02:45 PM-03:45 PM
MON	CO	ADE	BREAK	TCFS & NT	DMS	LUNCH BREAK	DS LAB	
TUE	DMS	SE		TCFS & NT	DSA		ADE LAB	
WED	TCFS & NT	ADE		DSA	SE		KANNADA	
THUR	SE	DMS		CO	ADE		PLACEMENT ACTIVITIES	
FRI	CO	TCFS & NT		DMS	DSA		TUTORIAL / PEDAGOGY ACTIVITIES	ADDITIONAL MATHEMATICS - I
SAT	ADE	CO		SE	DSA		TUTORIAL / PEDAGOGY ACTIVITIES	KANNADA (Non Kannadigas)


Subject Code	Subject Name	Faculty Name
18MAT31	TRANSFORM CALCULUS, FOURIER SERIES AND NUMERICAL TECHNIQUES	Mr. Chowdappa M R
18CS32	DATA STRUCTURES AND APPLICATIONS	Mrs. Vijayalaxmi M
18CS33	ANALOG AND DIGITAL ELECTRONICS	Mr. Sanjoy Das
18CS34	COMPUTER ORGANIZATION	Mrs. Deepa S R
18CS35	SOFTWARE ENGINEERING	Dr. Dayananda R B
18CS36	DISCRETE MATHEMATICAL STRUCTURES	Mr. Kushal Kumar B N
18CSL37	ANALOG AND DIGITAL ELECTRONICS LABORATORY	Mr. Sanjoy Das & Mrs. Deepa S R
18CSL38	DATA STRUCTURES LABORATORY	Mrs. Vijayalaxmi M & Mrs. Ranjitha K N & Mrs. Beena K
18KVK39/18KAK39	VYAVAHARIKA KANNADA (KANNADA FOR COMMUNICATION) AADALITHA KANNADA (KANNADA FOR ADMINISTRATION)	Mr. Thrimurthy

TIME TABLE INCHARGE

HOD
Head of the Department
Dept. of Computer Science & Engg.
K.S. Institute of Technology

COMPUTER ORGANIZATION (Effective from the academic year 2018 -2019) SEMESTER – III			
Course Code	18CS34	CIE Marks	40
Number of Contact Hours/Week	3:0:0	SEE Marks	60
Total Number of Contact Hours	40	Exam Hours	03
CREDITS –3			
Course Learning Objectives: This course (18CS34) will enable students to:			
<ul style="list-style-type: none"> • Explain the basic sub systems of a computer, their organization, structure and operation. • Illustrate the concept of programs as sequences of machine instructions. • Demonstrate different ways of communicating with I/O devices and standard I/O interfaces. • Describe memory hierarchy and concept of virtual memory. • Describe arithmetic and logical operations with integer and floating-point operands. • Illustrate organization of a simple processor, pipelined processor and other computing systems. 			
Module 1			Contact Hours
Basic Structure of Computers: Basic Operational Concepts, Bus Structures, Performance – Processor Clock, Basic Performance Equation, Clock Rate, Performance Measurement. Machine Instructions and Programs: Memory Location and Addresses, Memory Operations, Instructions and Instruction Sequencing, Addressing Modes, Assembly Language, Basic Input and Output Operations, Stacks and Queues, Subroutines, Additional Instructions, Encoding of Machine Instructions Text book 1: Chapter1 – 1.3, 1.4, 1.6 (1.6.1-1.6.4, 1.6.7), Chapter2 – 2.2 to 2.10 RBT: L1, L2, L3			08
Module 2			
Input/Output Organization: Accessing I/O Devices, Interrupts – Interrupt Hardware, Direct Memory Access, Buses, Interface Circuits, Standard I/O Interfaces – PCI Bus, SCSI Bus, USB. Text book 1: Chapter4 – 4.1, 4.2, 4.4, 4.5, 4.6, 4.7 RBT: L1, L2, L3			08
Module 3			
Memory System: Basic Concepts, Semiconductor RAM Memories, Read Only Memories, Speed, Size, and Cost, Cache Memories – Mapping Functions, Replacement Algorithms, Performance Considerations. Text book 1: Chapter5 – 5.1 to 5.4, 5.5 (5.5.1, 5.5.2), 5.6 RBT: L1, L2, L3			08
Module 4			
Arithmetic: Numbers, Arithmetic Operations and Characters, Addition and Subtraction of Signed Numbers, Design of Fast Adders, Multiplication of Positive Numbers, Signed Operand Multiplication, Fast Multiplication, Integer Division. Text book 1: Chapter2-2.1, Chapter6 – 6.1 to 6.6 RBT: L1, L2, L3			08
Module 5			
Basic Processing Unit: Some Fundamental Concepts, Execution of a Complete Instruction, Multiple Bus Organization, Hard-wired Control, Micro programmed Control. Pipelining: Basic concepts of pipelining, Text book 1: Chapter7, Chapter8 – 8.1 RBT: L1, L2, L3			08
Course Outcomes: The student will be able to :			
<ul style="list-style-type: none"> • Explain the basic organization of a computer system. 			

<ul style="list-style-type: none"> • Demonstrate functioning of different sub systems, such as processor, Input/output, and memory. • Illustrate hardwired control and micro programmed control, pipelining, embedded and other computing systems. • Design and analyse simple arithmetic and logical units.
Question Paper Pattern:
<ul style="list-style-type: none"> • The question paper will have ten questions. • Each full Question consisting of 20 marks • There will be 2 full questions (with a maximum of four sub questions) from each module. • Each full question will have sub questions covering all the topics under a module. • The students will have to answer 5 full questions, selecting one full question from each module.
Textbooks:
1. Carl Hamacher, Zvonko Vranesic, Safwat Zaky, Computer Organization, 5th Edition, Tata McGraw Hill, 2002. (Listed topics only from Chapters 1, 2, 4, 5, 6, 7, 8, 9 and 12)
Reference Books:
1. William Stallings: Computer Organization & Architecture, 9 th Edition, Pearson, 2015.


 Head of the Department
 Dept. of Computer Science & Engg.
 K.S. Institute of Technology
 Bengaluru -560 109



KS INSTITUTE OF TECHNOLOGY BANGALORE

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

NAME OF THE STAFF : DEEPA .S.R

SUBJECT CODE/NAME : 18CS34 / Computer Organization

SEMESTER/YEAR : III / 'A'

ACADEMIC YEAR : 2020-2021

Sl. No.	Topic to be covered	Mode of Delivery	Teaching Aid	No. of Periods	Cumulative No. of Periods	Proposed Date
MODULE 1: Basic Structure of Computers						
1	Introduction to CO	L+D	PPT using Zoom Platform	1	1	3/9/2020
2	Basic Operational Concepts	L+D	PPT using Zoom Platform	1	2	4/9/2020
3	Bus Structures, Performance – Processor Clock,	L+D	PPT using Zoom Platform	1	3	5/9/2020
4	Basic Performance Equation, Clock Rate	L+ D	PPT using Zoom Platform	1	4	7/9/2020
5	Performance Measurement	L+ D	PPT using Zoom Platform	1	5	10/9/2020
6	Machine Instructions and Programs: Memory Location and Addresses	L+ D	PPT using Zoom Platform	1	6	11/9/2020
7	Memory Operations, Instructions and Instruction Sequencing,	L+D	PPT using Zoom Platform	1	7	12/9/2020
8	Addressing Modes, Assembly Language	L+D	PPT using Zoom Platform	1	8	14/9/2020
9	Basic Input and Output Operations	L+D	PPT using Zoom Platform	1	9	18/9/2020

10	Stacks and Queues, Subroutines	L+D	PPT using Zoom Platform	1	10	19/9/2020
11	Additional Instructions, Encoding of Machine Instructions	L+D	PPT using Zoom Platform	1	11	21/9/2020
12	Pedagogy activity		PPT using Zoom Platform	1	12	24/9/2020
MODULE 2:Input/Output Organization						
13	Introduction to Input/Output Organization	L+ D	PPT using Zoom Platform	1	13	25/9/2020
14	Accessing Input Devices	L+D	PPT using Zoom Platform	1	14	26/9/2020
15	1 st Internal Assessment		Zoom Platform		15	29/9/2020
16	Accessing Output Devices	L+D	PPT using Zoom Platform	1	16	1/10/2020
17	Interrupts – Interrupt Hardware	L+D	PPT using Zoom Platform	1	17	3/10/202
18	Direct Memory Access	L+D	PPT using Zoom Platform	1	18	5/10/202
19	Buses	L+D	PPT using Zoom Platform	1	19	8/10/202
20	Interface Circuits	L+D	PPT using Zoom Platform	1	20	9/10/202
21	Standard I/O Interfaces – PCI Bus	L+D	PPT using Zoom Platform	1	21	10/10/202
22	SCSI Bus	L+D	PPT using Zoom Platform	1	22	12/10/202
23	USB.	L+D	PPT using Zoom Platform	1	23	15/10/202
24	Pedagogy activity		PPT using Zoom Platform	1	24	16/10/202
MODULE 3:Memory System						
25	Introduction to Memory System	L+D	PPT using Zoom Platform	1	25	19/10/202
26	Basic Concepts	L+D	PPT using Zoom Platform	1	26	22/10/202
27	Semiconductor RAM Memories: Internal organization	L+D	PPT using	1	27	23/10/202

	of Memory chips		Zoom Platform			
28	Static Memories, Asynchronous DRAMs and Synchronous DRAMs	L+D	PPT using Zoom Platform	1	28	24/10/202
29	Structure of Larger memories, memory system considerations	L+D	PPT using Zoom Platform	1	29	29/10/202
30	Read Only Memories	L+D	PPT using Zoom Platform	1	30	2/11/2020
31	Speed, Size, and Cost	L+D	PPT using Zoom Platform	1	31	5/11/2020
32	Cache Memories – Mapping Functions	L+D,PS(Tx)	PPT using Zoom Platform	1	32	6/11/2020
33	Replacement Algorithms	L+D,PS(Tx)	PPT using Zoom Platform	1	33	12/11/2020
34	Performance Consideration	L+D,PS(Tx)	PPT using Zoom Platform	1	34	13/11/2020
35	2 nd Internal Assessment		Zoom Platform		35	18/11/2020
36	Interleaving	L+I	PPT using Zoom Platform	1		19/11/2020
37	Hit rate and Miss penalty	L+D	PPT using Zoom Platform	1		20/11/2020
MODULE 4:Arithmetic						
38	Introduction to Arithmetic:	L+D,PS(Tx)	PPT using Zoom Platform	2	38	21/11/2020
39	Numbers, Arithmetic Operations and Characters,	L+D,PS(Tx)	PPT using Zoom Platform	1	39	23/11/2020
40	Addition and Subtraction of Signed Numbers	L+D,PS(Tx)	PPT using Zoom Platform	2	40	26/11/2020
41	Design of Fast Adders	L+D,PS(Tx)	PPT using Zoom Platform	1	41	27/11/2020
42	Multiplication of Positive Numbers,	L+D,PS(Tx)	PPT using Zoom Platform	1	42	30/11/2020
43	Signed Operand Multiplication	L+D,PS(Tx)	PPT using Zoom Platform	1	43	4/12/2020
44	Fast Multiplication	L+D,PS(Tx)	PPT using Zoom Platform	1	44	5/12/2020
45	Integer Division.	L+D,PS(Tx)	PPT using	1	45	7/12/2020

			Zoom Platform			
MODULE 5: Basic Processing Unit						
46	Introduction to Basic Processing Unit	L+D	PPT using Zoom Platform	1	46	10/12/2020
47	Some Fundamental Concepts	L+D	PPT using Zoom Platform	1	47	11/12/2020
48	Execution of a Complete Instruction	L+D	PPT using Zoom Platform	1	48	14/12/2020
49	Multiple Bus Organization	L+D	PPT using Zoom Platform	1	49	17/12/2020
50	Hard-wired Control – A Complete Processor	L+D	PPT using Zoom Platform	1	50	18/12/2020
51	Micro programmed Control.	L+D	PPT using Zoom Platform	1	51	19/12/2020
52	Micro instructions	L+D	PPT using Zoom Platform	1	52	20/12/2020
53	Micro program sequencing	L+D	PPT using Zoom Platform	1	53	24/12/2020
54	Micro instructions with next address field	L+D	PPT using Zoom Platform	1	54	28/12/2020
55	Pipelining: Basic concepts of pipelining	L+D	PPT using Zoom Platform	1	55	31/12/2020
56	Revision	L+D	PPT using Zoom Platform	1	56	1/1/2021
57	3 rd Internal Assessment		Zoom Platform		57	12/1/2021
58	Revision	L+D	PPT using Zoom Platform	1	58	2/1/2021
59	Revision	L+D	PPT using Zoom Platform	1	59	15/1/2021
60	Revision	L+D	PPT using Zoom Platform	1	60	16/1/2021

Text Books

1. Carl Hamacher, Zvonko Vranesic, Safwat Zaky, Computer Organization, 5th Edition, Tata McGraw Hill, 2002.

Reference Books

1. William Stallings: Computer Organization & Architecture, 9th edition, Pearson, 2015

Web Material:

<http://nptel.vtu.ac.in/econtent/courses/CSE/06CS46/index.php>

Details of Teaching Aids:

Power Point Presentations(PPT) using Zoom Platform

Signature of Course in-charge

Signature of Module Coordinator

Signature of HOD-CSE

*Head of the Department
Dept. of Computer Science & Engg.
K.S. Institute of Technology
Bengaluru -560 109*



K.S.Institute of Technology, Bangalore

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

ASSIGNMENT QUESTIONS

Academic Year	2020-2021		
Batch	2019-2023		
Year/Semester/section	II/III 'A' & 'B'		
Course Code-Title	18CS34/ Computer Organization		
Name of the Instructors	Deepa .S.R Vijaylaxmi Mekali	Dept	CSE

Assignment No: 1		Total marks:10		
Date of Issue: 29-9-2020		Date of Submission: 6-10-2020		
Sl.No	Assignment Questions	K Level	CO	Marks
1.	a. Identify performance measurement with SPEC rating for Computer by comparing running time for bench mark computer & computer to be tested.	Applying (K3)	1	1
	b. Experiment with following addressing Modes –Immediate addressing, Register addressing, Absolute addressing, Indirect addressing, Index addressing			1
	c. Identify the difference between Big Endian and Little Endian methods of byte addressing with proper example			1
	d. Make use of different examples to illustrate conditional code flags			1
	e. Model memory mapped IO with suitable example			1
	f. Pedagogy quiz			1
2.	a. Identify the different hardware components involved Program controlled IO with diagram	Applying (K3)	2	1
	b. Experiment with concepts of interrupt			1
	c. Model memory IO mapped IO with suitable example			1
	d. Utilize I/O interface to connect I/O device through the bus to the processor			1

deepa

Course in charge

deepa

Module Coordinator

m. natarajan

HOD

Head of the Department
Dept. of Computer Science & Engg.
K.S. Institute of Technology
Bangalore -560 109



K.S. INSTITUTE OF TECHNOLOGY, BANGALORE - 560109
I ASSIGNMENT 2020 – 21ODD SEMESTER

SCHEME AND SOLUTION

Degree	: B.E	Semester	: III A &B
Branch	: Computer Science & Engineering	Course Code	: 18CS34
Course Title	: Computer Organization	Max Marks	: 10

Q.N O.	POINTS	MAR KS
1(a)	<p>Identify performance measurement with SPEC rating for Computer by comparing running time for benchmark computer& computer to be tested.</p> <p style="text-align: center;">Processor Execution Time is given by</p> $T = N * S / R$ <p>This equation is called as Basic Performance Equation. Performance of a computer can also be measured by using benchmark programs. SPEC (System Performance Evaluation Corporation) is an organization, that measures performance of computer using SPEC rating.</p> <p><i>SPEC =</i> <i>Running time of referenceComputer</i> <i>Running time of computer undertest</i></p>	1
b)	<p>Experiment with following addressing Modes –Immediate addressing, Register addressing, Absolute addressing, Indirect addressing, Index addressing</p> <p>IMMEDIATEADDRESSING</p> <p>In this Addressing Mode operands are directly specified in the instruction. The source field is used to represent the operands. The operands are represented by # (hash) sign.</p> <p>Ex: MOVE #23, R0</p> <p>REGISTER ADDRESSING:</p> <p>In this mode operands are stored in the registers of CPU. The name of the register is directly specified in the instruction.</p> <p>Ex: MOVE R₁,R₂ Where R₁ and R₂ are the Source and Destination registers respectively.</p>	1

	<p>memory and higher byte of data is assigned to lower address of the memory. In this technique lower byte of data is assigned to lower address of the memory and higher byte of data is assigned to higher address of the memory.</p>					
d)	<p>Make use of different examples to illustrate conditional code flags</p> <table border="1" style="margin: 10px auto;"><tr><td>C</td><td>V</td><td>Z</td><td>N</td></tr></table> <p>1 N (NEGATIVE)Flag: It is designed to differentiate between positive and negative result. It is set 1 if the result is negative, and set to 0 if result is positive.</p> <p>2 Z (ZERO)Flag: It is set to 1 when the result of an ALU operation is found to zero, otherwise it is cleared.</p> <p>3 V (OVER FLOW)Flag: In case of 2^s Complement number system n-bit number is capable of representing a range of numbers and is given by -2^{n-1} to $+2^{n-1}$. The Over-Flow flag is set to 1 if the result is found to be out of this range.</p> <p>4 C (CARRY) Flag: This flag is set to 1 if there is a carry from addition or borrow from subtraction, otherwise it is cleared.</p>	C	V	Z	N	1
C	V	Z	N			
e)	<p>Model memory mapped IO with suitable example</p> <p>Memory-Mapped I/O Memory-Mapped I/O – some memory address values are used to refer to peripheral device buffer registers. No special instructions are needed. Also use device status registers.</p> <p>READWAIT Testbit #3, INSTATUS Branch=0 READWAIT MoveByte DATAIN, R1 WRITEWAIT Testbit #3, OUTSTATUS Branch=0 WRITEWAIT MoveByte R1, DATAOUT</p>	1				
2a)	<p>Identify the different hardware components involved in Program controlled IO with diagram</p> <p>Program-controlled I/O:</p> <ul style="list-style-type: none">• Processor repeatedly monitors a status flag to achieve the necessary synchronization.• Processor polls the I/O device.	1				

DIRECT ADDRESSING

It is also called as Absolute Addressing Mode. In this addressing mode operands are stored in the memory locations. The name of the memory location is directly specified in the instruction.

Ex: MOVE LOCA, R₁ : Where LOCA is the memory location and R₁ is the Register.

INDIRECT ADDRESSING

In this Addressing Mode effective address of an operand is stored in the memory location or General Purpose Register.

Ex: ADD (R₁), R₀ ; Where R₁ and R₀ are GPR^s

Ex: ADD (X), R₀

INDEX ADDRESSING MODE

In this addressing mode, the effective address of an operand is computed by adding constant value with the contents of Index Register and any one of the General Purpose Register namely R₀ to R_{n-1} can be used as the Index Register. The constant value is directly specified in the instruction.

The symbolic representations of this mode are as follows

X (R_i) where X is the Constant value and

R_j is the GPR. It can be represented as

EA of an operand = X + (R_i)

- c) Identify the difference between Big Endian and Little Endian methods of byte addressing with proper example

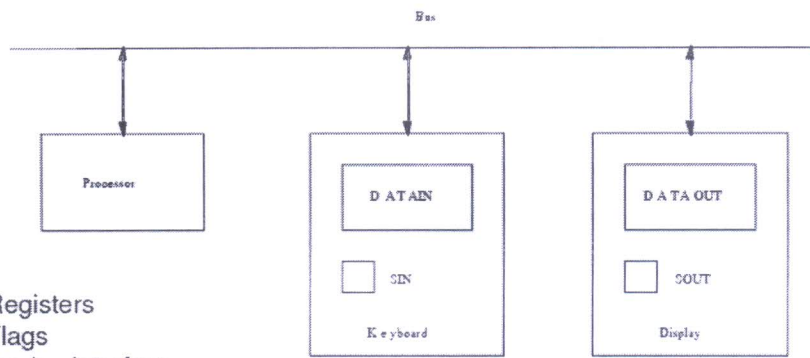
Word address	Byte address			
0	0	1	2	3
4	4	5	6	7
⋮				
2 ^k -4	2 ^k -4	2 ^k -3	2 ^k -2	2 ^k -1

BIG ENDIAN ASSIGNMENT

Byte address				
0	3	2	1	0
4	7	6	5	4
$2^k - 4$	$2^k - 1$	$2^k - 2$	$2^k - 3$	$2^k - 4$

LITTLE ENDIAN ASSIGNMENT

In this technique lower byte of data is assigned to higher address of the

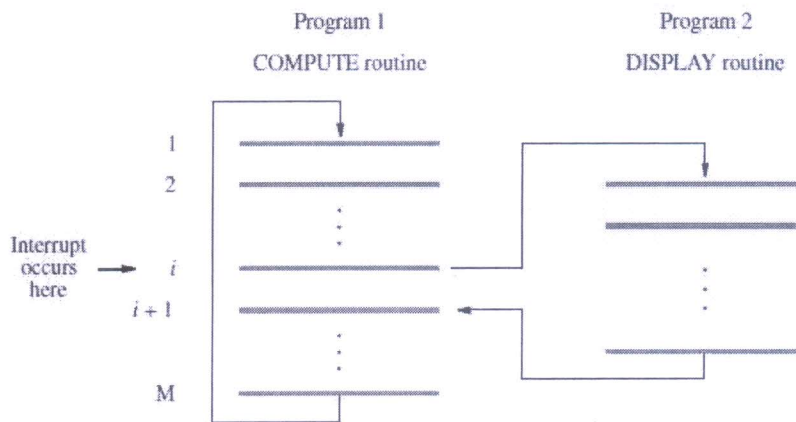


- Registers
- Flags
- Device interface

b) a. Experiment with concepts of interrupt

It is an event which suspends the execution of one program and begins the execution of another program.

The arrival of interrupt causes the processor to transfer the execution control from main program to subprogram



Explain Steps

c) Model memory IO mapped IO with suitable example
I/O Mapped I/O:

- In this technique CPU separates address space for memory and I/O devices.
- Hence two sets of instruction are used for data transfer.
- One set for memory operations and another set for I/O operations.
 - IN AL, DX

This instruction reads one byte of data from the I/P register whose address is store in DX register into AL register.

- The O/P operation can be implemented as, OUT DX,AL

This instruction transfer the contents of AL register into the

deepa

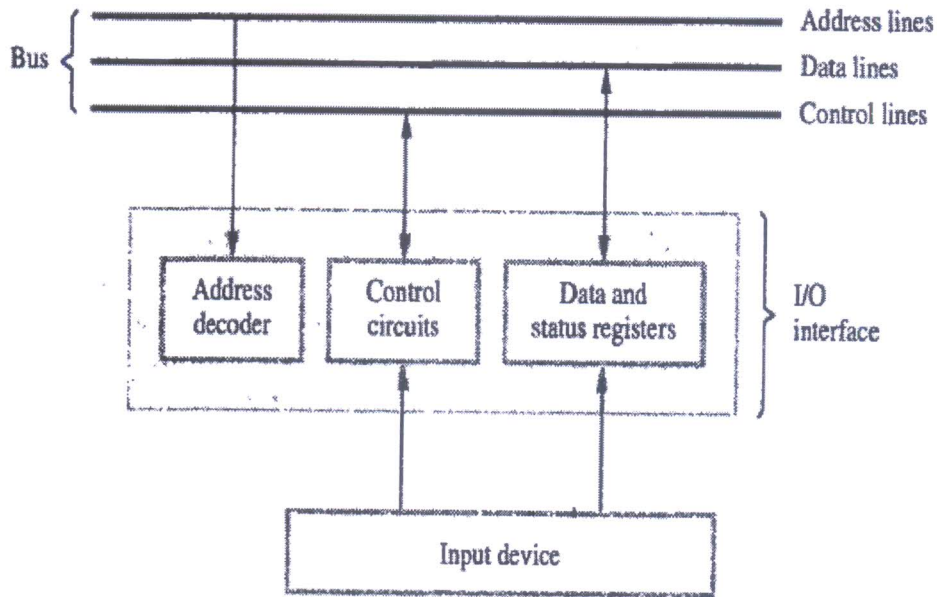
deepa

Head of the Department
Dept. of Computer Science & Engg.
K.S. Institute of Technology
Bengaluru -560 109

O/P register (DATA OUT) whose address is stored in DX register.

- d) Utilize I/O interface to connect I/O device through the bus to the processor

1



Explain

depa

Course Incharge

depa

Module Coordinator

Devaraj

HOD-CSE

Head of the Department
Dept. of Computer Science & Engg.
K.S. Institute of Technology
Bengaluru -560 109

Head of the Department
Dept. of Computer Science & Engg.
K.S. Institute of Technology
Bengaluru -560 109



K.S. INSTITUTE OF TECHNOLOGY, BANGALORE - 560109
II ASSIGNMENT 2020 – 21ODD SEMESTER

SCHEME AND SOLUTION

Degree : B.E
Branch : Computer Science & Engineering
Course Title : Computer Organization

Semester : III A & B
Course Code : 18CS34
Max Marks : 10

Q.NO.	POINTS	MARKS
1(i)	<p>Identify the followings with respect to USB.</p> <p>a. USB architecture</p> <div><div><input type="checkbox"/> A serial transmission format has been chosen for the USB because a serial bus satisfies the low- cost and flexibility requirements</div><div><input type="checkbox"/> Clock and data information are encoded together and transmitted as a single signal . Hence, there are no limitations on clock frequency or distance arising from dataskew</div><div><input type="checkbox"/> To accommodate a large number of devices that can be added or removed at any time, the USB has the tree structure</div></div> <p>Each node of the tree has a device called a hub, which acts as an intermediate controlpoint between the host and the I/O device . At the root of the tree, a root hub connects the entire tree to the host computer</p> <p>diagram</p> <p>b. USB addressing</p> <div><div>• Each device on the USB is assigned a 7-bitaddress.</div></div> <p>Explain</p> <p>c. USB protocols</p> <div><div><div><div><div>PID₀</div><div>PID₁</div><div>PID₂</div><div>PID₃</div><div>PID₄</div><div>PID₅</div><div>PID₆</div><div>PID₇</div></div><div>(a) Packet identifier field</div><div><div>Bits</div><div><div>8</div><div>7</div><div>4</div><div>5</div></div></div></div><div><div><div>PID</div><div>ADDR</div><div>ENDP</div><div>CRC16</div></div><div>(b) Token packet, IN or OUT</div><div><div>Bits</div><div><div>8</div><div>0 to 8192</div><div>16</div></div></div></div><div><div><div>PID</div><div>DATA</div><div>CRC16</div></div><div>(c) Data packet</div><div>USB packet formats.</div></div></div></div>	1
<p>Construct DMA circuit and explain its working</p>		

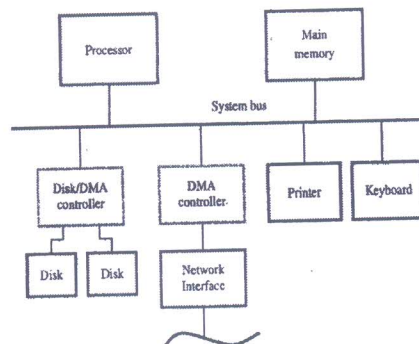
- It is the process of transferring the block of data at high speed in between main memory and external device (I/O devices) intervention of CPU is called as without continuous DMA.
- The DMA operation is performed by one control circuit and is part of the I/O interface.
- This control circuit is called DMA controller. Hence DMA transfer operation is performed by DMA controller.
- To initiate Directed data transfer between main memory and external devices DMA controller needs parameters from the CPU.
- These 3 Parameters are:

1) Starting address of the memory block.

2) No of words to be transferred.

3) Type of operation (Read or Write).

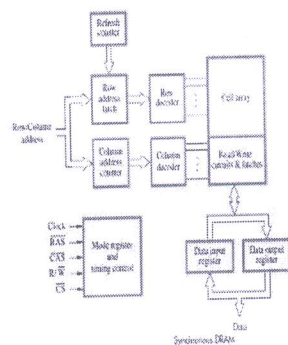
After receiving these 3 parameters from CPU, DMA controller establishes directed data transfer operation between main memory and external devices without the involvement of CPU.



2 i)

Obtain the main features of SDRAM with a neat diagram.

1

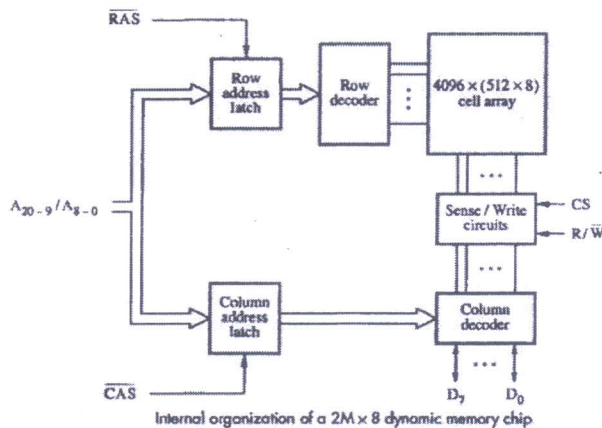


- The address and data connections are buffered by means of registers.
- The output of each sense amplifier is connected to a latch.

- A Read-operation causes the contents of all cells in the selected row to be loaded in these latches.
- Data held in latches that correspond to selected columns are transferred into data-output register.
- Thus, data becoming available on the data-output pins.
- The memory typically takes 2 or 3 clock cycles to activate the selected row.
- Then, the column-address is latched under the control of CAS signal.
- After a delay of one clock cycle, the first set of data bits is placed on the data-lines.
- SDRAM automatically increments column-address to access next 3 sets of bits in the selected row

ii.

Build the internal organization of a 2M x 8 asynchronous DRAM chip

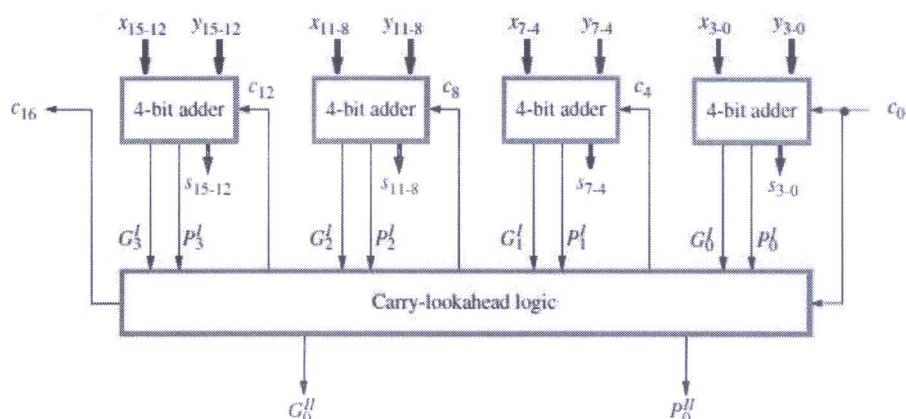


1

- The 4 bit cells in each row are divided into 512 groups of 8 as shown in the Figure.
- 21 bit address is needed to access a byte in the memory. 21 bit is divided as follows:
 - 1) 12 address bits are needed to select a row.
i.e. A8-0 → specifies row-address of a byte.
 - 2) 9 bits are needed to specify a group of 8 bits in the selected row.
i.e. A20-9 → specifies column-address of a byte.
- During Read/Write-operation,
 - row-address is applied first.
 - row-address is loaded into row-latch in response to a signal pulse on **RAS'** input of chip. (RAS = Row-address Strobe CAS = Column-address Strobe)
- When a Read-operation is initiated, all cells on the selected row are read and refreshed.
- Shortly after the row-address is loaded, the column-address is
 - applied to the address pins &
 - loaded into **CAS'**.
- The information in the latch is decoded.

<p>iii.</p>	<ul style="list-style-type: none"> The appropriate group of 8 Sense/Write circuits is selected. R/W'=1(read-operation) Output values of selected circuits are transferred to data-lines D0-D7. R/W'=0(write-operation) Information on D0-D7 are transferred to the selected circuits. <p>Obtain the main features of different types Read Only Memory</p> <p><u>TYPES OF ROM</u></p> <ul style="list-style-type: none"> Different types of non-volatile memory are <ol style="list-style-type: none"> PROM EPROM EEPROM & Flash Memory (Flash Cards & Flash Drives) <p><u>PROM (PROGRAMMABLE ROM)</u></p> <ul style="list-style-type: none"> PROM allows the data to be loaded by the user. Programmability is achieved by inserting a „fuse“ at point Pin a ROM cell. Before PROM is programmed, the memory contains all 0's. User can insert 1's at required location by burning-out fuse using high current-pulse. This process is irreversible. <p><u>EPROM (ERASABLE REPROGRAMMABLE ROM)</u></p> <ul style="list-style-type: none"> EPROM allows <ul style="list-style-type: none"> → stored data to be erased and → new data to be loaded. In cell, a connection to ground is always made at „P“ and a special transistor is used. The transistor has the ability to function as <ul style="list-style-type: none"> → a normal transistor or → a disabled transistor that is always turned „off“. Transistor can be programmed to behave as a permanently open switch, by injecting charge into it. Erase requires dissipating the charges trapped in the transistor of memory-cells. This can be done by exposing the chip to ultra-violet light. <p><u>EEPROM (ELECTRICALLY ERASABLE ROM)</u></p> <ul style="list-style-type: none"> Advantages: <ol style="list-style-type: none"> It can be both programmed and erased electrically. It allows the erasing of all cell contents selectively. Disadvantage: It requires different voltage for erasing, writing and reading the stored data. <p><u>FLASH MEMORY</u></p> <ul style="list-style-type: none"> In EEPROM, it is possible to read & write the contents of a single cell. In Flash device, it is possible to read contents of a single cell & 	<p>1</p>
-------------	---	----------

iv.	<p>write entire contents of a block.</p> <ul style="list-style-type: none"> • Prior to writing, the previous contents of the block are erased. Eg. In MP3 player, the flash memory stores the data that represents sound. • Single flash chips cannot provide sufficient storage capacity for embedded-system. <p>Identify the details of : i) memory latency ii) memory bandwidth</p> <ul style="list-style-type: none"> • A good indication of performance is given by 2 parameters: 1) Latency 2) Bandwidth. <p>Latency</p> <ul style="list-style-type: none"> • It refers to the amount of time it takes to transfer a word of data to or from the memory. • For a transfer of single word, the latency provides the complete indication of memory performance. • For a block transfer, the latency denotes the time it takes to transfer the first word of data. <p>Bandwidth</p> <ul style="list-style-type: none"> • It is defined as the number of bits or bytes that can be transferred in one second. • Bandwidth mainly depends on <ul style="list-style-type: none"> 1) The speed of access to the stored data & 2) The number of bits that can be accessed in parallel 	1
3 i)	<p>Make use of 16 bit carry look ahead adder and prove that the delay in generation of carry and sum are less than cascaded eight 4 bit adders</p> <p>16-bit adder can be built from four 4-bit adder blocks.</p> <ul style="list-style-type: none"> • These blocks provide new output functions defined as G_k and P_k, where $k=0$ for the first 4-bit block, $k=1$ for the second 4-bit block and so on. • In the first block, $P_0 = P_3 P_2 P_1 P_0$ & $G_0 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0$ • The first-level G_i and P_i functions determine whether bit stage i generates or propagates a carry, and the second level G_k and P_k functions determine whether block k generates or propagates a carry. • Carry c_{16} is formed by one of the carry-lookahead circuits as $c_{16} = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 c_0$ • Conclusion: All carries are available 5 gate delays after X, Y and c_0 are applied as inputs. Sum is obtained after 8 gate delays. 	1



In cascaded eight 4 bit adders, c_{16} is available after 9 gate delays and S_{15} is available after 10 gate delays.

ii.

Solve addition and subtraction for

- +4 and -6
- 5 and -2

Obtain the sum

Obtain the difference

1

deefa
Course Incharge

deefa
Module Coordinator

Murugan
HOD-CSE

Head of the Department
Dept. of Computer Science & Engg.
K.S. Institute of Technology
Bengaluru -560 109



K.S.Institute of Technology, Bangalore

DEPARTMENT OF DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

ASSIGNMENT QUESTIONS

Academic Year	2020-2021		
Batch	2019-2023		
Year/Semester/section	II/III 'A' & 'B'		
Course Code-Title	18CS34-Computer Organization		
Name of the Instructor	Deepa .S.R Vijaylaxmi Mekali	Dept	CSE

Assignment No: 3 Date of Issue: 28/12/2020		Total marks:10 Date of Submission: 7/1/2021		
Sl.No	Assignment Questions	K Level	CO	Marks
1.	a)Construct multiplication for i)+13 and -6 ii)+14 and -8 using Booth's algorithm	Applying(K3)	4	1
	b) Make use of an example and prove the efficiency in terms of Summands in Bit pair recording Vs Booth recording scheme			1
	c)Build division operation for X=100, Y=10010 and X=0101, Y=11111 using i)Restoring methods ii)Non restoring methods			1
	d) Identify the circuit arrangements for binary division.			1
2.	a) Build a single bus organization of the data path inside a processor b) Build the hardware implementation for one bit of register Ri c) Construct an Input Output gating for the register d) Identify the three steps to execute an instruction using IR and PC Registers e) Construct hardwired control unit. Show the generation Zin and END control signals. f) Identify pipeline stall with a suitable example	Applying(K3)	5	1 1 1 1 1 1

deepa
Course In-charge

deepa
Module Coordinator

Deepa
HOD

Head of the Department
Dept. of Computer Science & E
K.S. Institute of Technology
Bangaluru -560 109



K.S. INSTITUTE OF TECHNOLOGY, BANGALORE - 560109
ASSIGNMENT III 2020 – 21ODD SEMESTER

SCHEME AND SOLUTION

Degree : B.E
Branch : Computer Science & Engineering
Course Title : Computer Organization

Semester : III A
Course Code : 18CS34
Max Marks : 10

Q.NO	POINTS	MARKS
1 a	<p>a)Construct multiplication for i)+13 and -6 ii)+14 and -8 using Booth's algorithm i)</p> <div style="display: flex; align-items: center; justify-content: center;"> <div style="text-align: right;"> $\begin{array}{r} 01101 \quad (+13) \\ \times 11010 \quad (-6) \\ \hline \end{array}$ </div> <div style="margin: 0 20px; font-size: 2em;">→</div> <div style="text-align: left;"> $\begin{array}{r} 01101 \\ 0-1+1-10 \\ \hline 00000000000 \\ 111110011 \\ 00001101 \\ 1110011 \\ 000000 \\ \hline 1110110010 \quad (-78) \end{array}$ </div> </div> <p>ii) Similarly solve for +14* -8</p>	1
b	<p>Make use of an example and prove the efficiency in terms of Summands in Bit pair recording Vs Booth recording scheme</p> <div style="display: flex; align-items: center; justify-content: center;"> <div style="text-align: right;"> $\begin{array}{r} 01101 \quad (+13) \\ \times 11010 \quad (-6) \\ \hline \end{array}$ </div> <div style="margin: 0 20px; font-size: 2em;">→</div> <div style="text-align: left;"> $\begin{array}{r} 01101 \\ 0-1+1-10 \\ \hline 00000000000 \\ 111110011 \\ 00001101 \\ 1110011 \\ 000000 \\ \hline 1110110010 \quad (-78) \end{array}$ </div> </div>	1

$$\begin{array}{r}
 \begin{array}{ccccccc}
 & & & 0 & 1 & 1 & 0 & 1 \\
 & & & 0 & -1 & -2 \\
 \hline
 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\
 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & & \\
 0 & 0 & 0 & 0 & 0 & 0 & & & & \\
 \hline
 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0
 \end{array}
 \end{array}$$

c

Build division operation for X=1001, Y=10010 and X=0101, Y=11111 using

- i) Restoring methods
- ii) Non restoring methods

i) Build division operation for X=1001, Y=10010 using restoring method

Initially 0000010010

01001

Shift 000010010

subtract 10111

set q₀ 11000

restore 0100100100

0000100100

Shift 000100100

subtract 10111

set q₀ 10001

restore 0100101000

0001001000

shift 001001000

subtract 10111

set q₀ 110110

restore 010011000

00100

shift 010010000

subtract 10111

set q₀ 00000010000

shift 00000001

subtract 10111

set q₀ 1011100010

restore 01001

ii) Non restoring

Perform the operation of division using Restoring and non restoring method on the following of numbers using, where X is the divisor Y is the dividend $X=0101$, $Y=1111$

Restoring

	A	Q
	000000	1111
LS	000001	1111 □
add M	111011	
	111100	1111 0
add M	000101	
	000001	1111 0
S	000011	1110 □
add M	111011	
QM	111110	1110 0
add M	000101	
	000011	1110 0
S	000111	1100 □
	111011	
	000010	1100 1
LS	000101	1001 □
	111011	
	000000	1001 1
LS	000001	0011 □
add M	001011	
QM	111100	0011 0
	000101	
add M	000001	0011 0
	<u>000001</u>	<u>00110</u>
	Remainder	Quotient

d

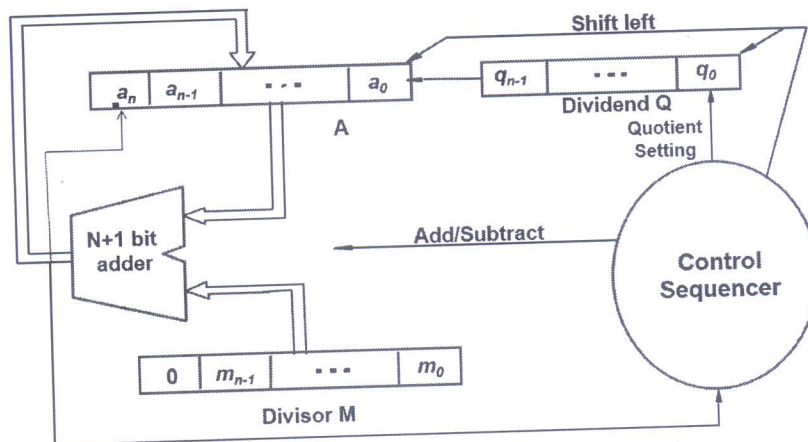
Identify the circuit arrangements for binary division.

1

 $M \leftarrow \text{divisor}, Q \leftarrow \text{dividend}, A \leftarrow 0$

- Shift A and Q left one binary position
- Subtract M from A, and place the answer back in A
- If the sign of A is 1, set q_0 to 0 and add M back to A (restore A); otherwise (If the sign of A is 0), set q_0 to 1

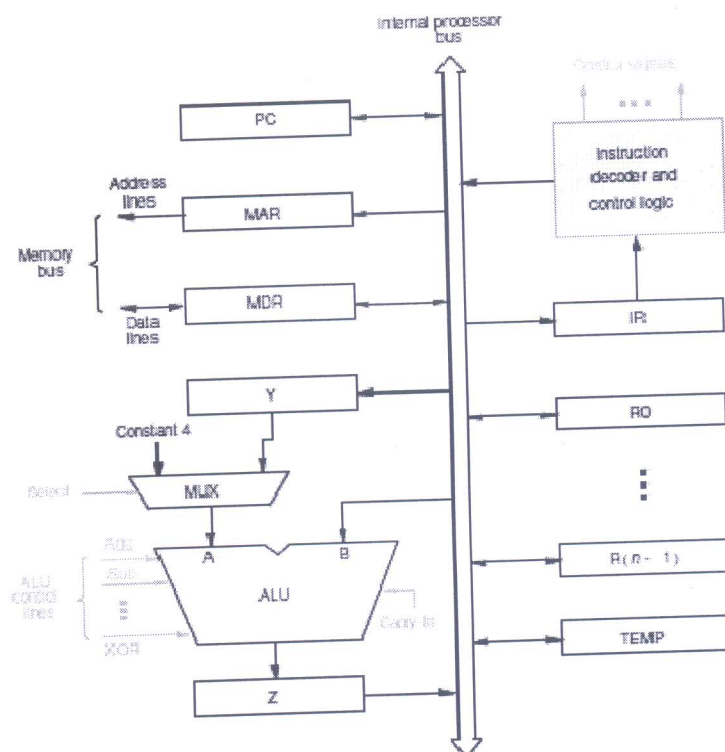
Repeat these steps n times



2 a

Build a single bus organization of the data path inside a processor

1

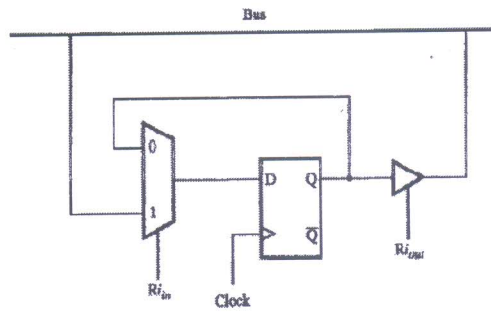


Explain

b

Build the hardware implementation for one bit of register R_i

1

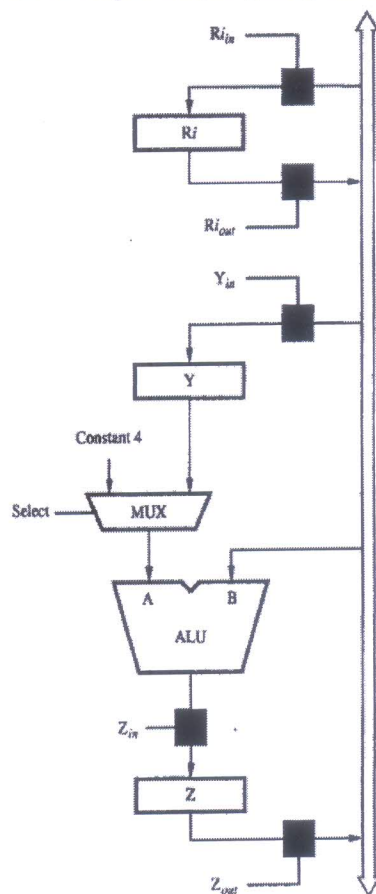


Explain

c

Construct an Input Output gating for the register

1



Explain

d

Identify the three steps to execute an instruction using IR and PC

Registers

Fetch the contents of the memory location pointed to by the PC. The contents of this location are loaded into the IR (fetch phase).

$IR \leftarrow [PC]$

2. Assuming that the memory is byte addressable, increment the contents of the PC by 4 (fetch phase).

$PC \leftarrow [PC] + 4$

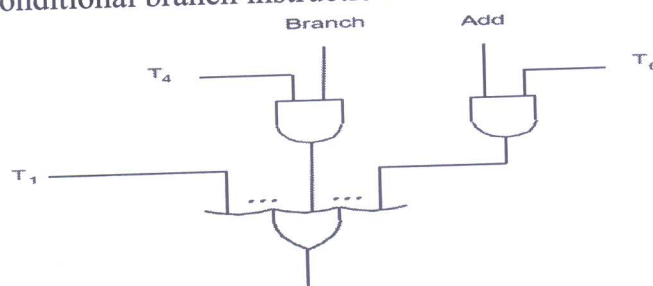
3. Carry out the actions specified by the instruction in the IR (execution phase).

1

e

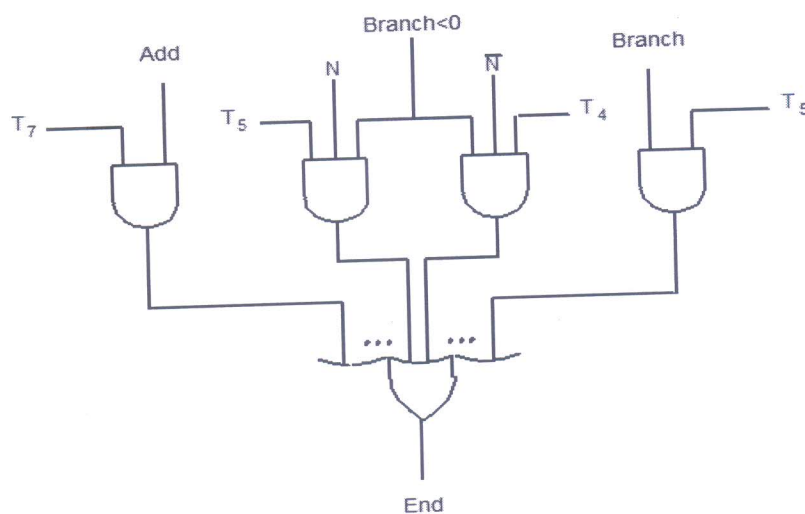
Construct hardwired control unit. Show the generation Z_{in} and END control signals.

Example of how the encoder Generates the Z_{in} control signal for the processor organization is given as. This signal is asserted during time slot T1 for all instructions, during T6 for an Add instruction, during T4 for an unconditional branch instruction



Generation of End control signal

$$\text{End} = T7 \cdot \text{ADD} + T5 \cdot \text{BR} + (T5 \cdot N + T4 \cdot N) \cdot \text{BRN} + \dots$$



f

Identify pipeline stall with a suitable example

For a variety of reasons one of pipeline stage may not be able to complete its task for a given instruction in the allocated time.

Ex: stage E in the four-stage pipeline as shown in figure is responsible for arithmetic and logic operations and one clock cycle is assigned for this task. But divide operations needs more number of clock cycle to complete the execution.

1

In figure 8.3 instruction I3 needs 3 clock cycles to complete cycle 4 to 6 and write stage is not suppose to perform is operation.

Thus information in buffer B2 must be preserved unit completion of execution phase.

Thus stage 2 and stage 1 are blocked in accepting new instruction because the information in buffer B1 cannot be over written. Thus steps D4 and F5 are postponed

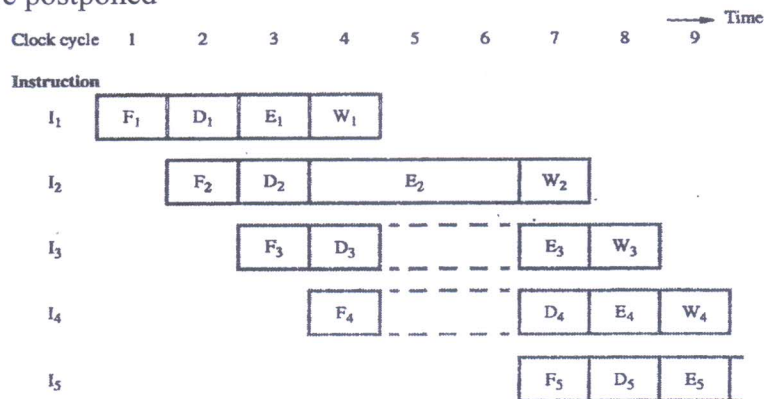


Figure 8.3 Effect of an execution operation taking more than one clock cycle.

Thus pipeline operation is said to stalled for two clock cycle (stall-means stage of pipeline not performs its specified task on designated clock cycle. Any condition that causes the stalls is called Hazard.

deefa

Course Incharge

deefa

Module Coordinator

W. Murthy

HOD-CSE

Head of the Department
Dept. of Computer Science & Engg.
K.S. Institute of Technology
Bengaluru -560 109



K.S. INSTITUTE OF TECHNOLOGY, BANGALORE - 560109
I SESSIONAL TEST QUESTION PAPER 2020 – 21 ODD SEMESTER

USN

Degree : B.E
Branch : Computer Science & Engg
Course Title : COMPUTER ORGANIZATION
Duration : 90 Minutes

Semester: III A& B
Course Code: 18CS34
Date: 6-10-2020
Max Marks : 30

Note: Answer ONE full question from each part.

Q No.	Question	Marks	CO mapping	K-Level
PART-A				
1(a)	Experiment with the following addressing Modes –Immediate addressing, Indirect addressing, Index addressing.	6	CO1	Applying (K3)
(b)	Identify performance measurement with SPEC rating for Computer by comparing running time for bench mark computer & computer to be tested.	6	CO1	Applying (K3)
(c)	Identify the difference between Big Endian and Little Endian methods of byte addressing with proper example	6	CO1	Applying (K3)
OR				
2(a)	Identify different condition code flags	6	CO1	Applying (K3)
(b)	Model memory mapped IO with suitable example	6	CO1	Applying (K3)
(c)	Identify the basic instruction types	6	CO1	Applying (K3)
PART-B				
3(a)	Identify the different hardware components involved Program controlled IO with diagram	6	CO2	Applying (K3)
(b)	Experiment with concepts of interrupt	6	CO2	Applying (K3)
OR				
4(a)	Utilize I/O interface to connect I/O device through the bus to the processor	6	CO2	Applying (K3)
(b)	Model memory IO mapped IO with suitable example	6	CO2	Applying (K3)

Email Id:

Sec-A deepasr@ksit.edu.in

Sec-B vijayalaxmimekali@ksit.edu.in

deepa
Course in-charge

deepa
Module Coordinator

W. Narayan
HOD
Head of the Department
Dept. of Computer Science & Engg.
K.S. Institute of Technology
Bengaluru -560 109



K.S. INSTITUTE OF TECHNOLOGY, BANGALORE - 560109
I SESSIONAL TEST QUESTION PAPER 2020 – 21 ODD SEMESTER

SCHEME AND SOLUTION

SET-A

Degree : B.E
Branch : Computer Science and Engineering
Course Title : Computer Organization

Semester : III
Course Code : 18CS34
Max Marks : 30

Q.NO.	POINTS	MARKS
1 (a)	<p>Addressing Modes –Immediate addressing, Indirect addressing, Index addressing.</p> <p>Ans:</p> <p>IMMEDIATE ADDRESSING</p> <p>In this Addressing Mode operands are directly specified in the instruction. The source field is used to represent the operands. The operands are represented by # (hash) sign.</p> <p>Ex: MOVE #23, R0</p> <p>INDIRECT ADDRESSING</p> <p>In this Addressing Mode effective address of an operand is stored in the memory location or General Purpose Register.</p> <p>Ex: ADD (R1), R0; Where R1 and R0 are GPRs</p> <p>Ex: ADD (X), R0</p> <p>INDEX ADDRESSING MODE</p> <p>In this addressing mode, the effective address of an operand is computed by adding constant value with the contents of Index Register and any one of the General Purpose Register namely R0 to Rn-1 can be used as the Index Register. The constant value is directly specified in the instruction.</p> <p>The symbolic representations of this mode are as follows</p> <p>X (Ri) where X is the Constant value and Ri is the GPR. It can be represented as EA of an operand = X + (Ri)</p>	2M+2M+2M
1 (b)	<p>Identify performance measurement with SPEC rating for Computer by comparing running time for bench mark computer & computer to be tested.</p> <p>Ans:</p> <p>Processor Execution Time is given by</p> $T = N * S / R$ <p>This equation is called as Basic Performance Equation. Performance of a computer can also be measured by using benchmark programs.</p>	3M+3M

POINTS

Q.NO.

SPEC (System Performance Evaluation Corporation) is an organization, that measures performance of computer using SPEC rating.

$$SPEC = \frac{\text{Running time of reference Computer}}{\text{Running time of computer under test}}$$

1 (c)

Identify the difference between Big Endian and Little Endian methods of byte addressing with proper example

Ans:

Word address	Byte address			
0	0	1	2	3
4	4	5	6	7
2^k-4	...			
	2^k-4	2^k-3	2^k-2	2^k-1

Big Endian assignment

In this technique lower byte of data is assigned to higher address of the memory and higher byte of data is assigned to lower address of the memory.

In this technique lower byte of data is assigned to lower address of the memory and higher byte of data is assigned to higher address of the memory.

Word address	Byte address			
	3	2	1	0
0	3	2	1	0
4	7	6	5	4
2^k-4	...			
	2^k-1	2^k-2	2^k-3	2^k-4

Little Endian assignment

2(a)

Identify different conditional code flags

Ans:

C	V	Z	N
---	---	---	---

1. **N (NEGATIVE) Flag:**

It is designed to differentiate between positive and negative result. It is set 1 if the result is negative, and set to 0 if result is positive.

2. **Z (ZERO) Flag:**

It is set to 1 when the result of an ALU operation is found to zero, otherwise it is cleared.

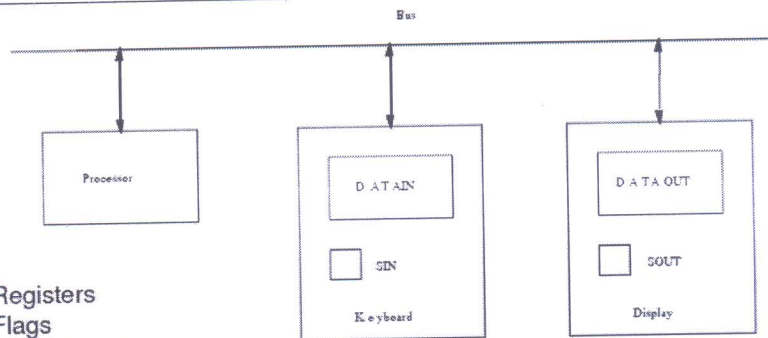
3. **V (OVER FLOW) Flag:**

In case of 2^s Complement number system n-bit number is capable of representing a range of numbers and is given by -2^{n-1} to $+2^{n-1}$. The Over-Flow flag is set to 1 if the result is found to be out of this range.

1M+2M+2M

6M

Q.NO.	POINTS	MARKS
2(b)	<p>4. C (CARRY) Flag : This flag is set to 1 if there is a carry from addition or borrow from subtraction, otherwise it is cleared.</p> <p>Model memory mapped IO with suitable example Ans:</p> <p>Memory-Mapped I/O Memory-Mapped I/O – some memory address values are used to refer to peripheral device buffer registers. No special instructions are needed. Also use device status registers.</p> <p>READWAIT Testbit #3, INSTATUS Branch=0 READWAIT MoveByte DATAIN, R1 WRITEWAIT Testbit #3, OUTSTATUS Branch=0 WRITEWAIT MoveByte R1, DATAOUT</p>	6M
2(c)	<p>Identify basic Instructions types Ans:</p> <ul style="list-style-type: none"> • Data Transfer Instructions • Data Manipulation Instructions <ul style="list-style-type: none"> ➤ Arithmetic ➤ Logical & Bit Manipulation ➤ Shift • Program Control Instructions • Conditional Branch Instructions 	6M
3(a)	<p>Identify the different hardware components involved in Program controlled IO with diagram Ans:</p> <p>Program-controlled I/O:</p> <ul style="list-style-type: none"> • Processor repeatedly monitors a status flag to achieve the necessary synchronization. • Processor polls the I/O device. 	4M+2M



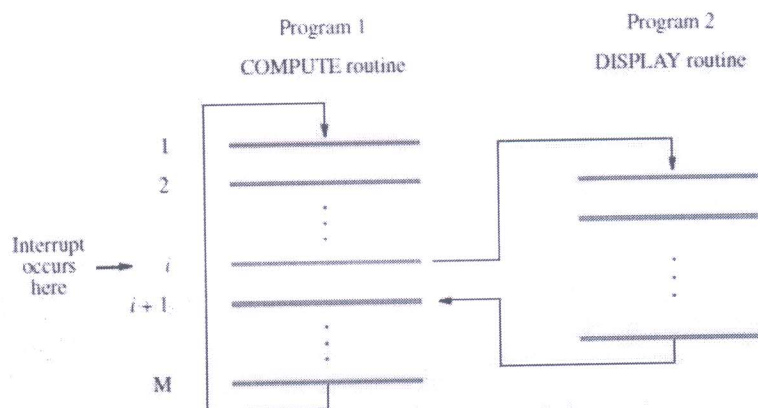
- Registers
- Flags
- Device interface

3(b)

Experiment with concepts of interrupt
Ans:

4M+2M

It is an event which suspends the execution of one program and begins the execution of another program.
 The arrival of interrupt causes the processor to transfer the execution control from main program to sub program

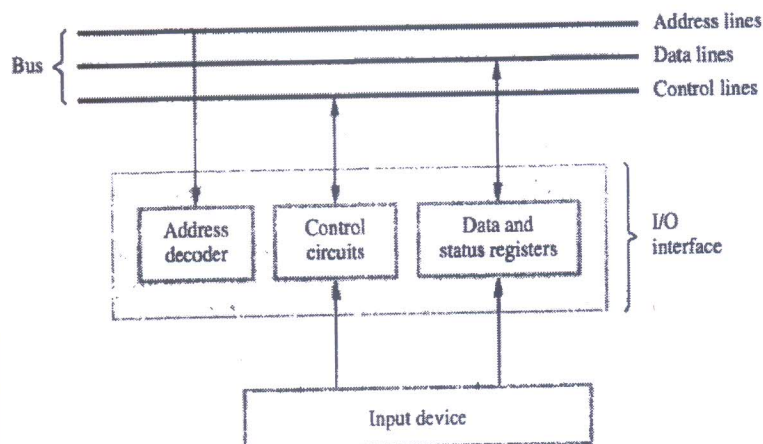


Explanation of interrupt concepts

4(a)

Utilize I/O interface to connect I/O device through the bus to the processor

2M + 4M



Explanation of concepts

Q.No	Points	MARKS
4(b)	<p>Model IO mapped IO with suitable example</p> <p>Ans:</p> <p>I/O Mapped I/O:</p> <ul style="list-style-type: none"> • In this technique CPU separates address space for memory and I/O devices. • Hence two sets of instruction are used for data transfer. • One set for memory operations and another set for I/O operations. • This instruction reads one byte of data from the I/P register whose address is store in DX register into AL register. • The O/P operation can be implemented as, OUT DX, AL • This instruction transfer the contents of AL register into the O/P register (DATA OUT) whose address is stored in DX register. 	6M

deeka
Course In-charge

deeka
Module Coordinator

murugan
HOD-CSE
Head of the Department
Dept. of Computer Science & Engg.
K.S. Institute of Technology
Bengaluru -560 109



K.S. INSTITUTE OF TECHNOLOGY, BANGALORE - 560109
I SESSIONAL TEST QUESTION PAPER 2020 – 21 ODD SEMESTER

SET B

USN

Degree : B.E
Branch : Computer Science & Engg
Course Title : COMPUTER ORGANIZATION
Duration : 90 Minutes

Semester: III A&B
Course Code: 18CS34
Date: 6-10-2020
Max Marks : 30

Note: Answer ONE full question from each part.

Q No.	Question	Marks	CO mapping	K-Level
PART-A				
1(a)	Experiment with the following addressing Modes –Immediate addressing, Indirect addressing, Index addressing.	6	CO1	Applying (K3)
(b)	Identify the difference between Big Endian and Little Endian methods of byte addressing with proper example	6	CO1	Applying (K3)
(c)	Identify performance measurement with SPEC rating for Computer by comparing running time for bench mark computer & computer to be tested.	6	CO1	Applying (K3)
OR				
2(a)	Identify the basic instruction types	6	CO1	Applying (K3)
(b)	Model memory mapped IO with suitable example	6	CO1	Applying (K3)
(c)	Identify different condition code flags	6	CO1	Applying (K3)
PART-B				
3(a)	Identify the different hardware components involved Program controlled IO with diagram.	6	CO2	Applying (K3)
(b)	Experiment with IO mapped IO with suitable example.	6	CO2	Applying (K3)
OR				
4(a)	Utilize I/O interface to connect I/O device through the bus to the processor	6	CO2	Applying (K3)
(b)	Model the concepts of interrupt	6	CO2	Applying (K3)


Course in charge

deefa
Module coordinator


HOD

Head of the Department
Dept. of Computer Science & Engg.
K.S. Institute of Technology
Bengaluru -560 109



K.S. INSTITUTE OF TECHNOLOGY, BANGALORE - 560109
I SESSIONAL TEST QUESTION PAPER 2020 – 21 ODD SEMESTER

SCHEME AND SOLUTION

SET-B

Degree : B.E
Branch : Computer Science
and Engineering
Course Title : Computer Organization

Semester : III
Course Code : 18CS34
Max Marks : 30

Q.NO.	POINTS	MARKS
1 (a)	<p>Addressing Modes –Immediate addressing, Indirect addressing, Index addressing.</p> <p>Ans:</p> <p>IMMEDIATE ADDRESSING</p> <p>In this Addressing Mode operands are directly specified in the instruction. The source field is used to represent the operands. The operands are represented by # (hash) sign.</p> <p>Ex: MOVE #23, R0</p> <p>INDIRECT ADDRESSING</p> <p>In this Addressing Mode effective address of an operand is stored in the memory location or General Purpose Register.</p> <p>Ex: ADD (R1), R0; Where R1 and R0 are GPRs</p> <p>Ex: ADD (X), R0</p> <p>INDEX ADDRESSING MODE</p> <p>In this addressing mode, the effective address of an operand is computed by adding constant value with the contents of Index Register and any one of the General Purpose Register namely R0 to Rn-1 can be used as the Index Register. The constant value is directly specified in the instruction.</p> <p>The symbolic representations of this mode are as follows</p> <p>$X(R_i)$ where X is the Constant value and R_j is the GPR. It can be represented as EA of an operand = $X + (R_i)$</p> <p>Identify the difference between Big Endian and Little Endian methods of byte addressing with proper example</p>	2M+2M+2M
1 (b)	<p>Ans:</p> <p>In this technique lower byte of data is assigned to higher address of the memory and higher byte of data is assigned to lower address of the memory.</p> <p>In this technique lower byte of data is assigned to lower address of the memory and higher byte of data is assigned to higher address of the memory.</p>	3M+3M

Q.NO.	POINTS	MARKS																																												
1 (c)	<div><div><p>Word address</p><table><tr><th colspan="4">Byte address</th></tr><tr><th>0</th><th>1</th><th>2</th><th>3</th></tr><tr><td>0</td><td>4</td><td>5</td><td>6</td></tr><tr><td>7</td><td>8</td><td>9</td><td>10</td></tr><tr><td colspan="4">⋮</td></tr><tr><td>2^k-4</td><td>2^k-3</td><td>2^k-2</td><td>2^k-1</td></tr></table><p>Big Endian assignment</p></div><div><p>Byte address</p><table><tr><th>3</th><th>2</th><th>1</th><th>0</th></tr><tr><td>0</td><td>4</td><td>5</td><td>6</td></tr><tr><td>7</td><td>8</td><td>9</td><td>10</td></tr><tr><td colspan="4">⋮</td></tr><tr><td>2^k-1</td><td>2^k-2</td><td>2^k-3</td><td>2^k-4</td></tr></table><p>Little Endian assignment</p></div></div> <p>Identify performance measurement with SPEC rating for Computer by comparing running time for bench mark computer & computer to be tested.</p> <p>Ans:</p> <p>Processor Execution Time is given by</p> $T = N * S / R$ <p>This equation is called as Basic Performance Equation.</p> <p>Performance of a computer can also be measured by using benchmark programs.</p> <p>SPEC (System Performance Evaluation Corporation) is an organization, that measures performance of computer using SPEC rating.</p> <p><i>SPEC =</i></p> $\frac{\text{Running time of reference Computer}}{\text{Running time of computer under test}}$ <p>Identify the basic instruction types</p> <p>Ans:</p> <ul style="list-style-type: none">• Data Transfer Instructions• Data Manipulation Instructions<ul style="list-style-type: none">➤ Arithmetic➤ Logical & Bit Manipulation➤ Shift• Program Control Instructions• Conditional Branch Instructions <p>Model memory mapped IO with suitable example</p> <p>Ans:</p> <p>Memory-Mapped I/O</p> <p>Memory-Mapped I/O – some memory address values are used to refer to peripheral device buffer registers. No special instructions are needed. Also use device status registers.</p> <p>READWAIT Testbit #3, INSTATUS</p> <p>Branch=0 READWAIT</p>	Byte address				0	1	2	3	0	4	5	6	7	8	9	10	⋮				2 ^k -4	2 ^k -3	2 ^k -2	2 ^k -1	3	2	1	0	0	4	5	6	7	8	9	10	⋮				2 ^k -1	2 ^k -2	2 ^k -3	2 ^k -4	<p>1M+2M+2M</p> <p>6M</p> <p>6M</p>
Byte address																																														
0	1	2	3																																											
0	4	5	6																																											
7	8	9	10																																											
⋮																																														
2 ^k -4	2 ^k -3	2 ^k -2	2 ^k -1																																											
3	2	1	0																																											
0	4	5	6																																											
7	8	9	10																																											
⋮																																														
2 ^k -1	2 ^k -2	2 ^k -3	2 ^k -4																																											

Q.NO.	POINTS	MARKS				
2(c)	<p>MoveByte DATAIN, R1 WRITEWAIT Testbit #3, OUTSTATUS Branch=0 WRITEWAIT MoveByte R1, DATAOUT</p> <p>Identify different condition code flags</p> <table border="1"><tr><td>C</td><td>V</td><td>Z</td><td>N</td></tr></table> <p>1. N (NEGATIVE) Flag: It is designed to differentiate between positive and negative result. It is set 1 if the result is negative, and set to 0 if result is positive.</p> <p>2. Z (ZERO) Flag: It is set to 1 when the result of an ALU operation is found to zero, otherwise it is cleared.</p> <p>3. V (OVER FLOW) Flag: In case of 2^s Complement number system n-bit number is capable of representing a range of numbers and is given by -2^{n-1} to $+2^{n-1}$. The Over-Flow flag is set to 1 if the result is found to be out of this range.</p> <p>4. C (CARRY) Flag : This flag is set to 1 if there is a carry from addition or borrow from subtraction, otherwise it is cleared.</p>	C	V	Z	N	6M
C	V	Z	N			
3(a)	<p>Identify the different hardware components involved in Program controlled IO with diagram</p> <p>Ans:</p> <p>Program-controlled I/O:</p> <ul style="list-style-type: none">__ Processor repeatedly monitors a status flag to achieve the necessary synchronization.__ Processor polls the I/O device. <div><div><div>Processor</div></div><div><div>DATA IN</div><div>SIN</div><div>Keyboard</div></div><div><div>DATA OUT</div><div>SOUT</div><div>Display</div></div></div> <p>- Registers - Flags - Device interface</p>	6M				

<p>3(b)</p>	<p>Model IO mapped IO with suitable example Ans:</p> <p>I/O Mapped I/O:</p> <ul style="list-style-type: none"> • In this technique CPU separates address space for memory and I/O devices. • Hence two sets of instruction are used for data transfer. • One set for memory operations and another set for I/O operations. • This instruction reads one byte of data from the I/P register whose address is store in DX register into AL register. • The O/P operation can be implemented as, OUT DX, AL • This instruction transfer the contents of AL register into the O/P register (DATA OUT) whose address is stored in DX register. 	<p>6M</p>
<p>4(a)</p>	<p>Utilize I/O interface to connect I/O device through the bus to the processor</p> <div data-bbox="368 853 1115 1258" data-label="Diagram"> </div> <ul style="list-style-type: none"> • I/O devices and the memory may have different address spaces: <ul style="list-style-type: none"> • Special instructions like IN and OUT are used to transfer data to and from I/O devices. • I/O devices may have to deal with fewer address lines. • I/O address lines need not be physically separate from memory address lines. • In fact, address lines may be shared between I/O devices and memory, with a control signal to indicate whether it is a memory address or an I/O address. • The hardware required to connect input-output devices to the bus is known as input-output device interface circuit. • I/O device is connected to the bus using an I/O interface circuit which has: <ul style="list-style-type: none"> • - Address decoder, control circuit, and data and status registers. • Address decoder decodes the address placed on the address lines thus enabling the device to recognize its address. If the address matches with the address of the device connected to it, then addressed device responds to the processor request. • Data register holds the data being transferred to or from the processor. 	<p>4M+2M</p> <p>2M + 4M</p>

QNO	Points	MARKS
4(b)	<p>Model the concepts of interrupt</p> <div data-bbox="298 405 1055 804" data-label="Diagram"> </div> <ul style="list-style-type: none"> • Status register holds information necessary for the operation of the I/O device. • Data and status registers are connected to the data lines, and have unique addresses. • I/O interface circuit coordinates I/O transfers. <ul style="list-style-type: none"> • Processor can perform other useful tasks while it is waiting for the device to be ready. • The routine executed in response to an interrupt request is called Interrupt Service Routine • Processor is executing the instruction located at address i when an interrupt occurs. • Routine executed in response to an interrupt request is called the interrupt-service routine. • When an interrupt occurs, control must be transferred to the interrupt service routine. • But before transferring control, the current contents of the PC ($i+1$), must be saved in a known location. • This will enable the return-from-interrupt instruction to resume execution at $i+1$. • Return address, or the contents of the PC are usually stored on the processor stack. 	6M

DeePa

Course Incharge

DeePa

Module Coordinator

Dr. M. Ramesh

HOD-CSE

Head of the Department
Dept. of Computer Science
K.S. Institute of Technology
Bengaluru -560 109



K.S. INSTITUTE OF TECHNOLOGY, BANGALORE - 560109

II SESSIONAL TEST QUESTION PAPER 2020 – 21 ODD SEMESTER

SET A

USN									
-----	--	--	--	--	--	--	--	--	--

Degree	: B.E	Semester	: III A& B
Branch	: COMPUTER SCIENCE AND ENGINEERING	Course Code	: 18CS34
Course Title	: COMPUTER ORGANIZATION	Date	: 18-11-2020
Duration	: 90 Minutes	Max Marks	: 30

Note: Answer ONE full question from each part.

Q No.	Question	Marks	CO mapping	K-Level
PART-A				
1(a)	Obtain the main features of different types Read Only Memory	6	CO3	Applying (K3)
(b)	Identify the main features of SDRAM with a neat diagram.	6	CO3	Applying (K3)
(C)	Build the internal organization of a 2M x 8 asynchronous DRAM chip.	6	CO3	Applying (K3)
OR				
2(a)	Identify the details of : i)memory latency ii) memory bandwidth	6	CO3	Applying (K3)
(b)	Identify the role of memory controller with a diagram	6	CO3	Applying (K3)
(C)	Construct Static RAM cell and explain its working	6	CO3	Applying (K3)
PART-B				
3(a)	Identify the followings with respect to USB. a. USB architecture b. USB addressing c. USB protocols	6	CO2	Applying (K3)
(b)	Solve addition and subtraction for a. +4 and -6 b. -5 and -2	6	CO4	Applying (K3)
OR				
4(a)	Construct DMA circuit and explain its working	6	CO2	Applying (K3)
(b)	Make use of 16 bit carry look ahead adder and prove that the delay in generation of carry and sum are less than cascaded eight 4 bit adders	6	CO4	Applying (K3)

Semester	Section	Faculty Name	Email ID
III	A	Deepa .S.R	deepasr@ksit.edu.in
III	B	Vijayalaxmi Mekali	vijayalaxmimekali@ksit.edu.in

Deepa
Course in charge

deepa
Module Coordinator

W. S. Aravind
HOD

Head of the Department
Dept. of Computer Science & Eng
K.S. Institute of Technology
Bangalore -560 109

K.S. INSTITUTE OF TECHNOLOGY, BANGALORE - 560109
II INTERNAL ASSESSMENT 2020 – 21 ODD SEMESTER

SET-A

SCHEME AND SOLUTION

Degree : B.E
Branch : Computer Science & Engineering
Course Title : Computer Organization

Semester : III A & B
Course Code : 18CS34
Max Marks : 30

Q.NO	POINTS	MARKS
1a)	<p>Obtain the main features of different types Read Only Memory</p> <p><u>TYPES OF ROM</u></p> <ul style="list-style-type: none"> Different types of non-volatile memory are <ol style="list-style-type: none"> PROM EPROM EEPROM & Flash Memory (Flash Cards & Flash Drives) <p><u>PROM (PROGRAMMABLE ROM)</u></p> <ul style="list-style-type: none"> PROM allows the data to be loaded by the user. Programmability is achieved by inserting a „fuse“ at point P in a ROM cell. Before PROM is programmed, the memory contains all 0“s. User can insert 1“s at required location by burning-out fuse using highcurrent-pulse. <p><u>EPROM (ERASABLE REPROGRAMMABLE ROM)</u></p> <ul style="list-style-type: none"> EPROM allows <ul style="list-style-type: none"> → stored data to be erased and → new data to be loaded. In cell, a connection to ground is always made at „P“ and a special transistor is used. <p><u>EEPROM (ELECTRICALLY ERASABLE ROM)</u></p> <ul style="list-style-type: none"> Advantages: <ol style="list-style-type: none"> It can be both programmed and erased electrically. It allows the erasing of all cell contents selectively. <p><u>FLASH MEMORY</u></p> <ul style="list-style-type: none"> In EEPROM, it is possible to read & write the contents of a single cell. In Flash device, it is possible to read contents of a single cell & write entire contents of a block. Prior to writing, the previous contents of the block are erased. <p>Identify the main features of SDRAM with a neat diagram</p>	<p>2</p> <p>2</p> <p>1</p> <p>1</p>
b	<p>Identify the main features of SDRAM with a neat diagram</p> <ul style="list-style-type: none"> The address and data connections are buffered by means of registers. The output of each sense amplifier is connected to a latch. A Read-operation causes the contents of all cells in the selected row to be loaded in these latches. Data held in latches that correspond to selected columns are transferred 	3

	<p>into data-output register.</p> <ul style="list-style-type: none"> • Thus, data becoming available on the data-output pins. • The memory typically takes 2 or 3 clock cycles to activate the selected row. <p>Diagram</p> <p>Build the internal organization of a 2M x 8 asynchronous DRAM chip.</p> <p>Internal organization of a 2M x 8 dynamic memory chip</p>	3
C		3
	<p>Explain</p> <p>Identify the details of : i) memory latency ii) memory bandwidth</p> <p>Latency</p> <ul style="list-style-type: none"> • It refers to the amount of time it takes to transfer a word of data to or from the memory. • For a transfer of single word, the latency provides the complete indication of memory performance. <p>Bandwidth</p> <ul style="list-style-type: none"> • It is defined as the number of bits or bytes that can be transferred in one second. • Bandwidth mainly depends on <ol style="list-style-type: none"> 1) The speed of access to the stored data & 2) The number of bits that can be accessed in parallel 	3
2a)		3
	<p>Identify the role of memory controller with a diagram</p> <p>To reduce the number of pins, the dynamic memory chips use multiplexed address inputs.</p> <ul style="list-style-type: none"> ◦ The address is divided into two parts ◦ Higher order bits that selects the a row in cell array ◦ Are provided first and latched into the memory chip under control of RAS signal ◦ Lower order bits are provided on same address pins to select the column and latched using to CAS signal ◦ Processor issue the all bits of address at a same time <p>diagram.</p>	3
b)		3
	<p>Construct Static RAM cell and explain its working</p>	

c)	<ul style="list-style-type: none"> • Memory that consists of circuit capable of retaining their state as long as power is applied are known as Static memories (SRAM). • Two transistor inverters are cross connected to implement a basic flip-flop. • The cell is connected to one word line and two bits lines by transistors T1 and 2. • These transistors acts as switches that can be opened or closed under control of word line <p>Diagram</p>	3
3a)	<p>Identify the followings with respect to USB.</p> <p>a. USB architecture</p> <p>b. USB addressing</p> <p>c. USB protocols</p> <p>USB architecture</p> <ul style="list-style-type: none"> <input type="checkbox"/> A serial transmission format has been chosen for the USB because a serial bus satisfies the low- cost and flexibility requirements <input type="checkbox"/> Clock and data information are encoded together and transmitted as a single signal . Hence, there are no limitations on clock frequency or distance arising from dataskew <p>USB addressing</p> <ul style="list-style-type: none"> • Each device on the USB is assigned a 7-bit address. <p>Explain</p> <p>c. USB protocols</p>	3
b)	<p>Solve addition and subtraction for</p> <p>a. +4 and -6</p> <p>b. -5 and -2</p> <p>Obtain the sum</p> <p>Obtain the difference</p>	2
4a)	<p>Construct DMA circuit and explain its working</p> <ul style="list-style-type: none"> • It is the process of transferring the block of data at high speed in between main memory and external device (I/O devices) intervention of CPU is called as without continuous DMA. • The DMA operation is performed by one control circuit and is part of the I/O interface. • This control circuit is called DMA controller. Hence DMA transfer operation is performed by DMA controller. • To initiate Directed data transfer between main memory and external devices DMA controller needs parameters from the CPU. <p>Circuit</p> <p>Make use of 16 bit carry look ahead adder and prove that the delay in generation of carry and sum are less than cascaded eight 4 bit adders</p>	3



K.S. INSTITUTE OF TECHNOLOGY, BANGALORE - 560109

SET B

II SESSIONAL TEST QUESTION PAPER 2020 – 21 ODD SEMESTER

USN									
-----	--	--	--	--	--	--	--	--	--

Degree : B.E	Semester : III A& B
Branch : COMPUTER SCIENCE AND ENGINEERING	Course Code : 18CS34
Course Title : COMPUTER ORGANIZATION	Date : 18-11-2020
Duration : 90 Minutes	Max Marks : 30

Note: Answer ONE full question from each part.

Q No.	Question	Marks	CO mapping	K-Level
PART-A				
1(a)	Identify the main features of Synchronous Dynamic RAM chip with a neat diagram.	6	CO3	Applying (K3)
(b)	Construct Static RAM cell and explain its working	6	CO3	Applying (K3)
(c)	Build the 2M x 32 memory module using 1M x 8 chips.	6	CO3	Applying (K3)
OR				
2(a)	Identify the details of i. miss rate ii. miss penalty iii. Hit rate iv. Memory bandwidth v. Memory latency	6	CO3	Applying (K3)
(b)	Identify the role of memory controller with a diagram	6	CO3	Applying (K3)
(c)	Obtain the main features of different types Read Only Memory.	6	CO3	Applying (K3)
PART-B				
3(a)	Identify the followings with respect to USB. a. USB architecture b. USB addressing c. USB protocols	6	CO2	Applying (K3)
(b)	Solve addition a. 7 and -3 b. -8 and -2	6	CO4	Applying (K3)
OR				
4(a)	Construct the DMA and identify how the data is transferred between the memory and I/O devices using DMA controller.	6	CO2	Applying (K3)
(b)	Make use of 16 bit carry look ahead adder and prove that the delay in generation of carry and sum are less than cascaded eight 4 bit adders.	6	CO4	Applying (K3)

Semester	Section	Faculty Name	Email ID
III	A	Deepa .S.R	deepasr@ksit.edu.in
III	B	Vijayalaxmi Mekali	vijayalaxmimekali@ksit.edu.in

Course In-Charge

Module Coordinator

HOD-CSE

Head of the Department
Dept. of Computer Science & Engg.
K.S. Institute of Technology
Bengaluru -560 109



K.S. INSTITUTE OF TECHNOLOGY, BANGALORE - 560109
II SESSIONAL TEST QUESTION PAPER 2020 – 21 ODD SEMESTER

SCHEME AND SOLUTION

SET- B

Degree : B.E
Branch : Computer Science & Engineering
Course Title : Computer Organization

Semester : III A & B
Course Code : 18CS34
Max Marks : 10

Q.NO.	POINTS	MARKS
1(a)	<p>Obtain the main features of SDRAM with a neat diagram.</p> <ul style="list-style-type: none"> • The address and data connections are buffered by means of registers. • The output of each sense amplifier is connected to a latch. • A Read-operation causes the contents of all cells in the selected row to be loaded in these latches. • Data held in latches that correspond to selected columns are transferred into data-output register. • Thus, data becoming available on the data-output pins. • The memory typically takes 2 or 3 clock cycles to activate the selected row. • Then, the column-address is latched under the control of CAS" signal. • After a delay of one clock cycle, the first set of data bits is placed on the data-lines. • SDRAM automatically increments column-address to access next 3 sets of bits in the selected row. 	6
(b)	<p>Construct Static RAM cell and explain its working</p> <ul style="list-style-type: none"> • Memory that consists of circuit capable of retaining their state as long as power is applied are known as Static memories (SRAM). • Two transistor inverters are cross connected to implement a basic flip-flop. • The cell is connected to one word line and two bits lines by transistors T1 and 2. • These transistors acts as switches that can be opened or closed under control of word line. • When word line is at ground level, the transistors are turned off and the latch retains its state • Ex: Let us assume that the cells in the state 1 if the logic value at point X is 1 and at point Y it is 0. This state is maintained as long as the signal on the word line is at ground. 	4+2

Read operation:

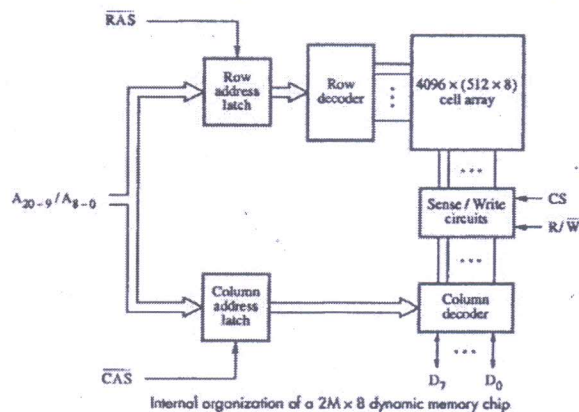
- In order to read state of SRAM cell, the word line is activated to close switches T1 and T2. Signal on the bit line b is high and the signal on the bit line b' is low. The opposite is true for state 0.
- Thus b and b' are complement each other.
- Sense/Write circuits at the bottom monitor the state of b and b' and the sets the output according to that.

Write operation:

State of the cell is set by placing the appropriate value on bit line b and its complement on b' and activating the word line. This forces the cell into corresponding state. The required signals on the bit lines are generated by the sense/write circuit.

(c) Build the internal organization of a 2M x 8 asynchronous DRAM chip

ii



4+2

- The 4 bit cells in each row are divided into 512 groups of 8 as shown in the Figure.
- 21 bit address is needed to access a byte in the memory. 21 bit is divided as follows:
 - 12 address bits are needed to select a row.
 - i.e. A₈₋₀ → specifies row-address of a byte.
 - 9 bits are needed to specify a group of 8 bits in the selected row.
 - i.e. A₂₀₋₉ → specifies column-address of a byte.
- During Read/Write-operation,
 - row-address is applied first.
 - row-address is loaded into row-latch in response to a signal pulse on RAS' input of chip. (RAS = Row-address Strobe CAS = Column-address Strobe)
- When a Read-operation is initiated, all cells on the selected row are read and refreshed.
- Shortly after the row-address is loaded, the column-address is
 - applied to the address pins &
 - loaded into CAS'.
- The information in the latch is decoded.
- The appropriate group of 8 Sense/Write circuits is selected.
 - R/W'=1(read-operation) Output values of selected circuits are transferred to data-lines D₀-D₇.
 - R/W'=0(write-operation) Information on D₀-D₇ are transferred to the selected circuits.

2 (a)

Identify the details of i. miss rate ii. miss penalty iii. Hit rate iv. Memory bandwidth v. Memory latency

- **Miss rate:** The fraction or percentage of accesses that result in a miss is called the miss rate.
- **Miss penalty:** The difference between lower level access time and cache access time is called the miss penalty.
- **Hit rate:** The fraction or percentage of accesses that result in a hit is called the hit rate
- **Memory latency:** It is the time it takes to transfer a word of data to or from memory
- **Memory bandwidth:** It is the number of bits or bytes that can be transferred in one second

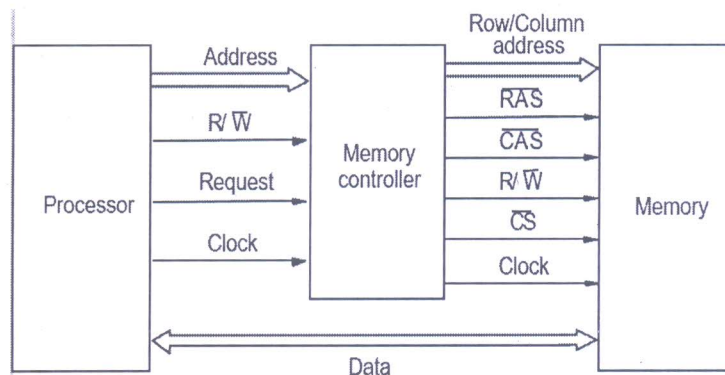
6

(b)

Identify the role of memory controller with a diagram

- To reduce the number of pins, the dynamic memory chips use multiplexed address inputs.
- The address is divided into two parts
- Higher order bits that selects the a row in cell array
- Are provided first and latched into the memory chip under control of RAS signal
- Lower order bits are provided on same address pins to select the column and latched using to CAS signal
- Processor issue the all bits of address at a same time.
- The required multiplexing of address bits is usually performed by memory controller unit.
- Controller unit is appears between processor and DRAM.
- The controller accepts the complete address and R/W signal from the processor under a control of Request signal which indicates that a memory access operation is needed.
- The controller then forwards the row and column portion of address to the memory and generates RAS and CAS signals.
- Controller provides RAS-CAS timing, in addition to its address multiplexing function.
- It also sends R/W and CS signals to memory.
- Data lines are connected directly to between the processor and memory.
-

4+2



(c)

Obtain the main features of different types Read Only Memory

6

TYPES OF ROM

- Different types of non-volatile memory are
- PROM
- EPROM
- EEPROM &
- Flash Memory (Flash Cards & Flash Drives).

PROM (PROGRAMMABLE ROM)

- PROM allows the data to be loaded by the user.
- Programmability is achieved by inserting a „fuse“ at point P in a ROM cell.
- Before PROM is programmed, the memory contains all 0“s.
- User can insert 1“s at required location by burning-out fuse using high current-pulse.
- This process is irreversible.

EPROM (ERASABLE REPROGRAMMABLE ROM)

- EPROM allows
 - stored data to be erased and
 - new data to be loaded.
- In cell, a connection to ground is always made at „P“ and a special transistor is used.
- The transistor has the ability to function as
 - a normal transistor or
 - a disabled transistor that is always turned „off“.
- Transistor can be programmed to behave as a permanently open switch, by injecting charge into it.
- Erasure requires dissipating the charges trapped in the transistor of memory-cells. This can be done by exposing the chip to ultra-violet light.

EEPROM (ELECTRICALLY ERASABLE ROM)

- **Advantages:**
- It can be both programmed and erased electrically.
- It allows the erasing of all cell contents selectively.
- **Disadvantage:** It requires different voltage for erasing, writing and reading the stored data.

1

FLASH MEMORY

- In EEPROM, it is possible to read & write the contents of a single cell.
- In Flash device, it is possible to read contents of a single cell & write entire contents of a block.
- Prior to writing, the previous contents of the block are erased.
 - Eg. In MP3 player, the flash memory stores the data that represents sound.
- Single flash chips cannot provide sufficient storage capacity for embedded-system.

3(a)

Identify the followings with respect to USB.

a. USB architecture

- A serial transmission format has been chosen for the USB because a serial bus satisfies the low- cost and flexibility requirements
- Clock and data information are encoded together and transmitted as a single signal . Hence, there are no limitations on clock frequency or distance arising from data skew
- To accommodate a large number of devices that can be added or removed at any time, the USB has the tree structure

Each node of the tree has a device called a hub, which acts as an intermediate control point between the host and the I/O device . At the root of the tree, a root hub connects the entire tree to the host computer

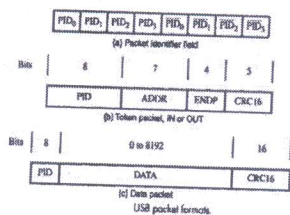
diagram

b. USB addressing

- Each device on the USB is assigned a 7-bit address.

Explain

c. USB protocols



(b)

Solve addition and subtraction for

- 7 and -3
- 8 and -2

1. Add 7 and -3

Solution

7 → 0 1 1 1

-3 → 1 1 0 1

0	1	1	1
+	1	1	0
	1	1	
1	0	1	0

Carry out ignore it

When we add 7 and -3 ans is 4

2+2+2

3+3

Solution

-8 → 1 0 0 0

-2 → 1 1 1 0

	1	0	0	0
+	1	1	1	0
1	0	1	1	0 (MSB is 0 so it is + number)

(overflow occurs as the sign of both the summands is **minus** and **sign of resultant is plus** ie sign of summands and results differs.

4+2

4(a)

Construct the DMA and identify how the data is transferred between the memory and I/O devices using DMA controller.

- It is the process of transferring the block of data at high speed in between main memory and external device (I/O devices) intervention of CPU is called as without continuous DMA.
- The DMA operation is performed by one control circuit and is part of the I/O interface.
- This control circuit is called DMA controller. Hence DMA transfer operation is performed by DMA controller.
- To initiate Directed data transfer between main memory and external devices DMA controller needs parameters from the CPU.
- These 3 Parameters are:

• **Starting address of the memory block.**

• **No of words to be transferred.**

• **Type of operation (Read or Write).**

After receiving these 3 parameters from CPU, DMA controller establishes directed data transfer operation between main memory and external devices without the involvement of CPU.

4+2

Make use of 16 bit carry look ahead adder and prove that the delay in generation of carry and sum are less than cascaded eight 4 bit adders

4(b)

16-bit adder can be built from four 4-bit adder blocks.

- These blocks provide new output functions defined as G_k and P_k , where $k=0$ for the first 4-bit block, $k=1$ for the second 4-bit block and so on.

• In the first block,

$$P_0 = P_3 P_2 P_1 P_0$$

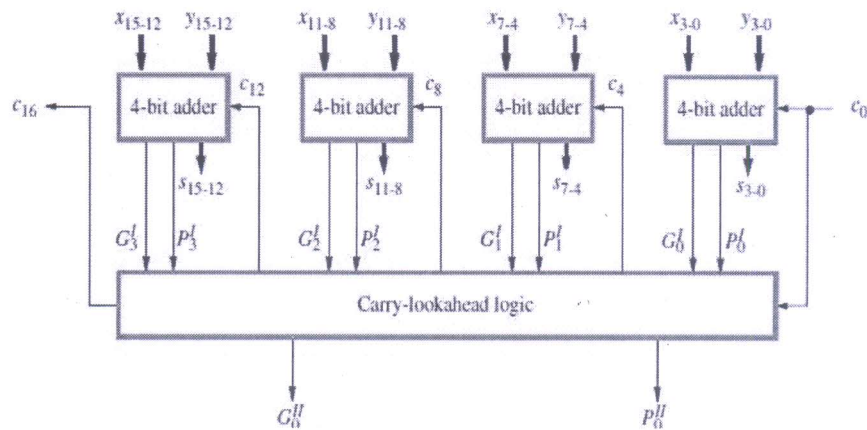
&

$$G_0 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0$$

- The first-level G_i and P_i functions determine whether bit stage i generates or propagates a carry, and the second level G_k and P_k functions determine whether block k generates or propagates a carry.

- Carry c_{16} is formed by one of the carry-lookahead circuits as

$$c_{16} = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0 + P_3P_2P_1P_0c_0$$
- Conclusion: All carries are available 5 gate delays after X , Y and c_0 are applied as inputs. Sum is obtained after 8 gate delays.




Course Incharge


Module Coordinator


HOD-CSE

Head of the Department
 Dept. of Computer Science & Engg.
 K.S. Institute of Technology
 Bengaluru -560 109



K.S. INSTITUTE OF TECHNOLOGY, BANGALORE - 560109
III SESSIONAL TEST QUESTION PAPER 2020 – 21ODD SEMESTER

USN

SET – A

Degree : B.E Semester: 3rd A & B
Branch : Computer Science & Engineering
Course Title : Computer Organization
Duration : 90 Minutes

Subject Code: 18CS34
Date: 7 - 1 - 2021
Max Marks: 30

Note: Answer ONE full question from each part.

Q No.	Questions	Marks	CO mapping	K-Level
PART-A				
1(a)	Make use of multiplication of +13 and -6 and prove the efficiency in terms of Summands in Bit pair recording Vs Booth recording scheme	6	CO4	Applying (K3)
(b)	Construct multiplication for i) +15 and -6 ii) 110011 and 101100 using Booth's algorithm	6	CO4	Applying (K3)
(c)	Build the hardware implementation for one bit of register Ri	6	CO5	Applying (K3)
OR				
2(a)	Build division operation for X=1001, Y=10010 using Restoring method	6	CO4	Applying (K3)
(b)	Identify the circuit arrangements for binary division.	6	CO4	Applying (K3)
(c)	Construct hardwired control unit. Show the generation Zin and END control signals.	6	CO5	Applying (K3)
PART-B				
3(a)	Identify pipeline stall with a suitable example	6	CO5	Applying (K3)
(b)	Identify the three steps to execute an instruction using IR and PC Registers	6	CO5	Applying (K3)
OR				
4(a)	Build a single bus organization of the data path inside a processor	6	CO5	Applying (K3)
(b)	Construct the control sequence for the instruction ADD R4,R5,R6 for three bus organization.	6	CO5	Applying (K3)

deepa
Course In-charge

deepa
Module Coordinator

Dr. M. Narayana
HOD-CSE
 Head of the Department
 Dept. of Computer Science & Engg.
 K.S. Institute of Technology
 Bengaluru - 560 109



K.S. INSTITUTE OF TECHNOLOGY, BANGALORE - 560109
III INTERNAL ASSESSMENT 2020 – 21 ODD SEMESTER

SET A

SCHEME AND SOLUTION

Degree : B.E
Branch : Computer Science & Engineering
Course Title : Computer Organization

Semester : III A & B
Course Code : 18CS34
Max Marks : 30

Q.NO	POINTS	MARKS
1(a)	<p>Make use of multiplication of +13 and -6 and prove the efficiency in terms of Summands in Bit pair recording Vs Booth recording scheme</p> <div style="display: flex; align-items: center; justify-content: center;"> <div style="text-align: right;"> $\begin{array}{r} 01101 \quad (+13) \\ \times 11010 \quad (-6) \\ \hline \end{array}$ </div> <div style="font-size: 2em; margin: 0 10px;">→</div> <div style="text-align: left;"> $\begin{array}{r} 01101 \\ 0-1+1-10 \\ \hline \end{array}$ </div> </div> <div style="text-align: center;"> $\begin{array}{r} 0000000000 \\ 111110011 \\ 000001101 \\ 1110011 \\ 0000000 \\ \hline 11100110010 \quad (-78) \end{array}$ </div> <div style="text-align: center;"> $\begin{array}{r} 01101 \\ 0-1-2 \\ \hline 1111100110 \\ 11110011 \\ 000000 \\ \hline 11100110010 \end{array}$ </div>	3
b)	<p>Construct multiplication for</p> <p>i) +15 and -6</p> <p>ii) 110011 and 101100 using Booth's algorithm</p>	3

i)

					0	1	1	1	1
				X	0	-1	+1	-1	0
0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	0	0	0	1	X
0	0	0	0	1	1	1	1	X	X
1	1	1	0	0	0	1	X	X	X
0	0	0	0	0	0	X	X	X	X
1	1	1	0	1	0	0	1	1	0

3

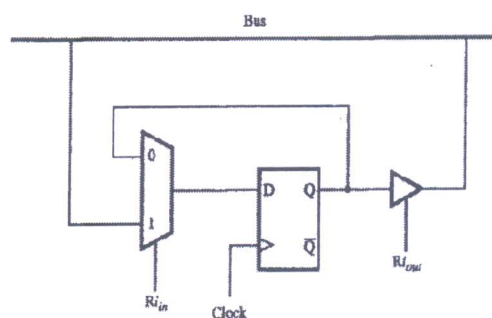
ii)

						1	1	0	0	1	1
						-1	+1	0	-1	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	X
0	0	0	0	0	0	1	1	0	1	X	X
0	0	0	0	0	0	0	0	0	X	X	X
1	1	1	1	0	0	1	1	X	X	X	X
0	0	0	1	1	0	1	X	X	X	X	X
0	0	0	1	0	0	0	0	0	1	0	0

3

c)

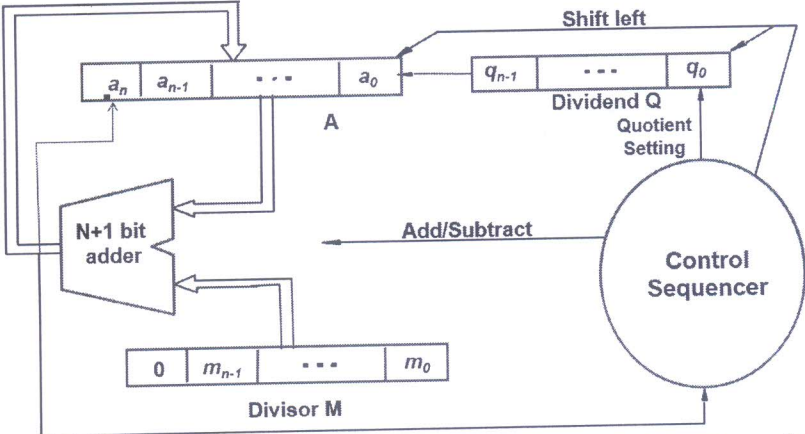
Build the hardware implementation for one bit of register Ri



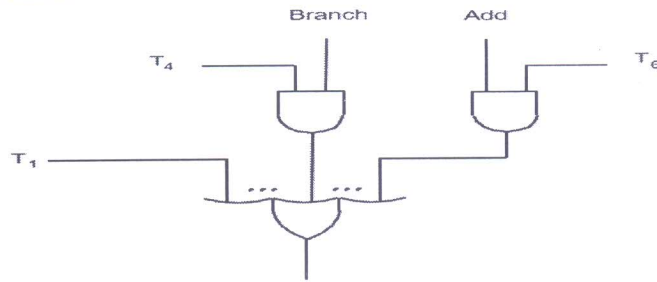
3

Explain

3

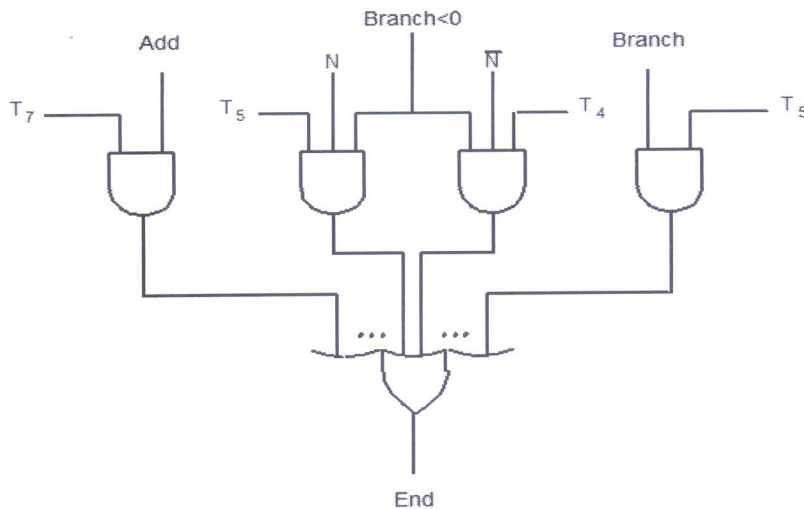
2)a)	<p>Build division operation for $X=1001$, $Y=10010$ using restoring method</p> <p>Initially 0000010010 01001 Shift 000010010 subtract 10111 set q_0 11000 restore 0100100100 0000100100 Shift 000100100 subtract 10111 set q_0 10001 restore 0100101000 0001001000 shift 001001000 subtract 10111 set q_0 110110 restore 010011000 00100 shift 010010000 subtract 10111 set q_0 00000010000 shift 00000001 subtract 10111 set q_0 1011100010 restore 01001</p>	<p>Initialize 1 mark, Each stage 1 mark</p>
2b)	<p>Identify the circuit arrangements for binary division.</p> <p>$M \leftarrow \text{divisor}$, $Q \leftarrow \text{dividend}$, $A \leftarrow 0$</p> <ul style="list-style-type: none"> Shift A and Q left one binary position Subtract M from A, and place the answer back in A If the sign of A is 1, set q_0 to 0 and add M back to A (restore A); otherwise (If the sign of A is 0), set q_0 to 1 <p>Repeat these steps n times</p> 	<p>2</p> <p>4</p>
2c)	<p>Construct hardwired control unit. Show the generation Z_{in} and END control signals.</p> <p>Example of how the encoder Generates the Z_{in} control signal for the processor organization is given as. This signal is asserted during time slot T1 for all instructions, during T6 for an Add instruction, during T4 for an</p>	<p>3</p>

unconditional branch instruction



Generation of End control signal

$$\text{End} = T7 \cdot \text{ADD} + T5 \cdot \text{BR} + (T5 \cdot N + T4 \cdot N) \cdot \text{BRN} + \dots$$



3

3a)

Identify pipeline stall with a suitable example

For a variety of reasons one of pipeline stage may not be able to complete its task for a given instruction in the allocated time.

Ex: stage E in the four-stage pipeline as shown in figure is responsible for arithmetic and logic operations and one clock cycle is assigned for this task. But divide operations needs more number of clock cycle to complete the execution.

In figure 8.3 instruction I3 needs 3 clock cycles to complete cycle 4 to 6 and write stage is not suppose to perform is operation.

Thus information in buffer B2 must be preserved unit completion of execution phase.

Thus stage 2 and stage 1 are blocked in accepting new instruction because the information in buffer B1 cannot be over written. Thus steps D4 and F5 are postponed

3

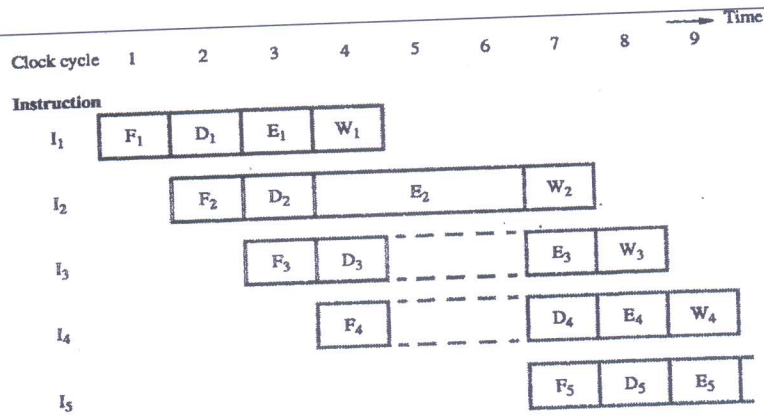
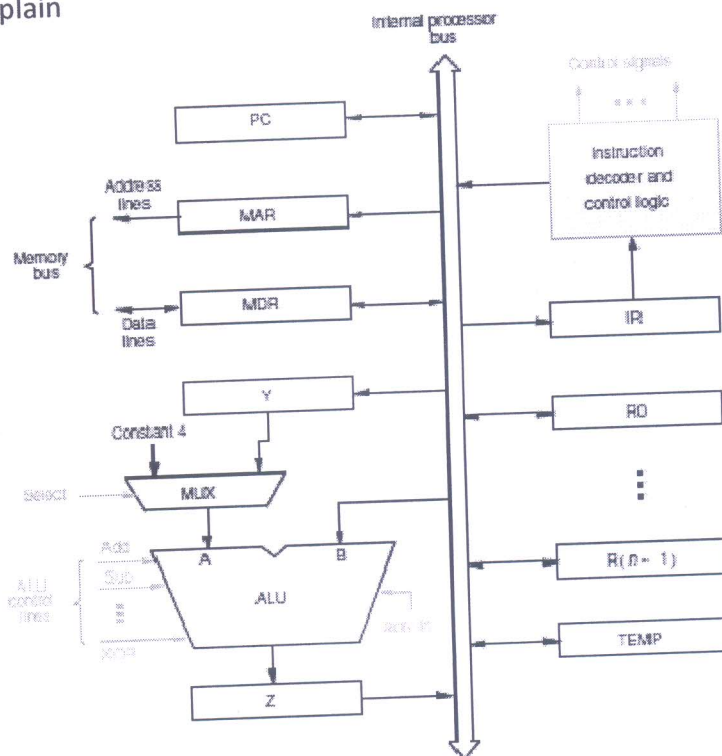


Figure 8.3 Effect of an execution operation taking more than one clock cycle.

Thus pipeline operation is said to stalled for two clock cycle (stall-means stage of pipeline not performs its specified task on designated clock cycle. Any condition that causes the stalls is called Hazard.

- b Identify the three steps to execute an instruction using IR and PC Registers
- Fetch the contents of the memory location pointed to by the PC. The contents of this location are loaded into the IR (fetch phase).
- $IR \leftarrow [PC]$
2. Assuming that the memory is byte addressable, increment the contents of the PC by 4 (fetch phase).
- $PC \leftarrow [PC] + 4$
3. Carry out the actions specified by the instruction in the IR (execution phase).

- 4a) Build a single bus organization of the data path inside a processor Explain



b)	<p>Construct the control sequence for the instruction ADD R4,R5,R6 for three bus organization</p> <p>Step Action</p> <p>1 $PC_{out}, R=B, MAR_{in}, \text{Read, IncPC}$</p> <p>2 WMFC</p> <p>3 $MDR_{outB}, R=B, IR_{in}$</p> <p>4 $R4_{outA}, R5_{outB}, \text{SelectA, Add, } R6_{in}, \text{End}$</p> <p>Explain</p>	<p>4</p> <p>2</p>
----	--	-------------------

deefa
Course Incharge

deefa
Module Coordinator

Om meenapen
HOD-CSE
Head of the Department
Dept. of Computer Science & Engg.
K.S. Institute of Technology
Bengaluru -560 109



K.S. INSTITUTE OF TECHNOLOGY, BANGALORE - 560109
III SESSIONAL TEST QUESTION PAPER 2020 – 21ODD SEMESTER

USN

SET – B

Degree : B.E
Branch : Computer Science & Engineering
CourseTitle : Computer Organization
Duration : 90Minutes


Semester: 3rd A & B
SubjectCode: 18CS34
Date: 7-1-2021
MaxMarks: 30

Note: Answer ONE full question from each part.

Q.No.	Questions	Marks	CO mapping	K-Level
PART-A				
1(a)	Solve multiplication for i) -12 and -8 ii) 0 1 1 1 0 and 0 1 1 0 0 using Booth's algorithm	6	CO4	Applying (K3)
(b)	Make use of multiplication of +13 and -6 and prove the efficiency in terms of Summands in Bit pair recording Vs Booth recording scheme	6	CO4	Applying (K3)
(c)	Develop the hardware implementation for one bit of register Ri	6	CO5	Applying (K3)
OR				
2(a)	Build division operation for X=1100, Y=101 X=0101, Y=11111 using Restoring method	6	CO4	Applying (K3)
(b)	Construct hardwired control unit. Show the generation Zin and END control signals.	6	CO4	Applying (K3)
(c)	Identify the circuit arrangements for binary division.	6	CO5	Applying (K3)
PART-B				
3(a)	Choose the three steps to execute an instruction using IR and PC Registers	6	CO5	Applying (K3)
(b)	Identify pipeline stall with a suitable example	6	CO5	Applying (K3)
OR				
4(a)	Build a three bus organization of the data path inside a processor	6	CO5	Applying (K3)
(b)	Construct the control sequence for the instruction ADD R4, R5, R6 for single bus organization.	6	CO5	Applying (K3)


Course In charge


Module Coordinator


HOD-CSE
 Head of the Department
 Dept. of Computer Science & Engg
 K.S. Institute of Technology
 Bengaluru -560 109

SCHEME AND SOLUTION

SET-B

Degree : **B.E**
Branch : **Computer Science and Engineering**
Course Title : **Computer Organization**

Semester : III
Course Code : 18CS34

Max Marks : 30

[illegible]

				0	0	1	1	1	0		
				1	1	1					
Product				0	0	1	0	1	0	1	0
Product is 168											

1 (b)

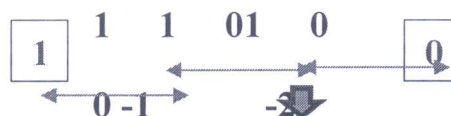
Booths algorithm

6M

									0	1	1	0	1
								X	0	-1	+1	-1	0
				0	0	0	0	0	0	0	0	0	0
				1	1	1	1	1	0	0	1	1	
				0	0	0	0	1	1	0	1		
				1	1	1	0	0	1	1			
				0	0	0	0	0	0				
				1	1	1	1	1	1	1			
Product is negative number				1	1	1	0	1	1	0	0	1	0
To find the what exactly a number is find its 2's complement, put the -sign Ans is -78													

Find the append 0 at right, shortage of bit at left side to make group of three bits apply sign extension of 1 bit at left.

6M



Booth bit pair recording of multiplier 0 -1 -2

Find the 2's complement of Multiplicand

$$\begin{array}{r}
 01101 \\
 10010 \\
 \hline
 1
 \end{array}$$

1 0 0 1 1 (2's complement of multiplicand is negative)

									0	1	1	0	1
							X		0		-1		-2
				1	1	1	1	1	0	0	1	1	0
				1	1	1	1	0	0	1	1		
				0	0	0	0	0	0				
				1	1	1			1	1			
Product is				1	1	1	0	1	1	0	0	1	0
				0	0	0	1	0	0	1	1	0	1
													1
				0	0	0	1	0	0	1	1	1	0
							64			8	4	2	0

1(c)

6M

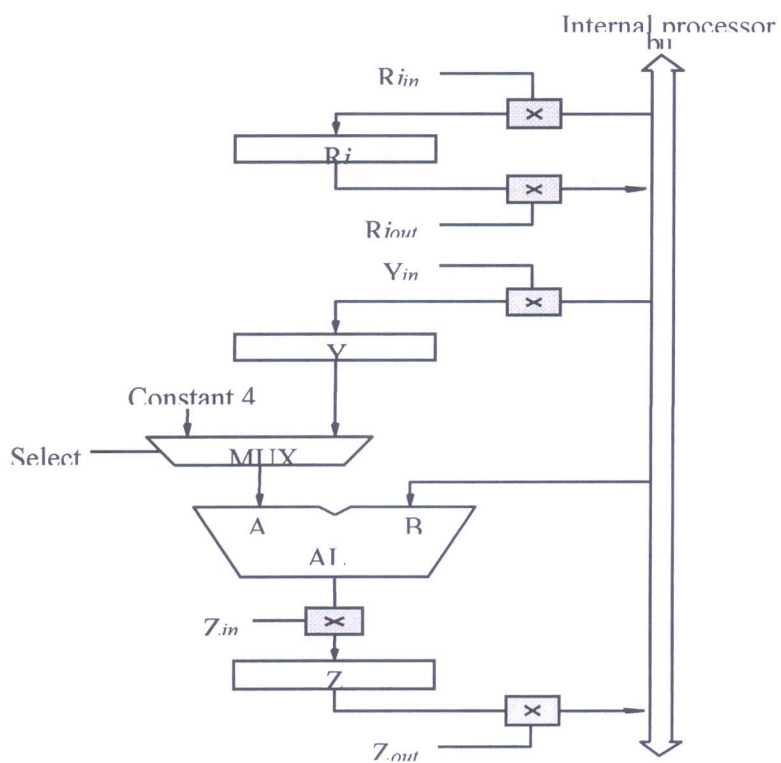


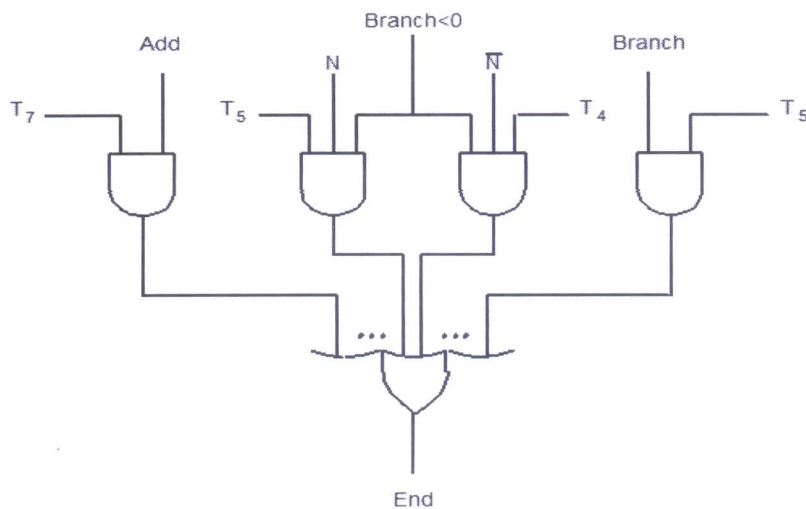
Figure 7.2. Input and output paths for the registers in Figure.

Q.NO.	POINTS				MARKS			
2(a)	Build division operation for X=1100, Y=101 Restoring method Ans:				3M			
	n	M	A	Q		Operation		
	4	00101	00000	1100		Initialization		
				00001		100?	SL of A and Q	
				00001 11011 1100		100?	A=A-M	
				00001		1000	Set $q_0 \leftarrow 0$, restore M	
	3			00011		000?	SL of A and Q ₂	
				00011 11011 1110		000?	A=A-M	
				00011		0000	Set $q_0 \leftarrow 0$, restore M	
				00110		000?	SL of A and Q ₃	
	2			00110 11011 00001		000?	A=A-M	
				00001		0001	Set $q_0 \leftarrow 1$,	
		1				00010	001?	SL of A and Q
						00010 11011 11101	001?	A=A-M
				00010		0010	Set $q_0 \leftarrow 1$,	
	0		Reminder is 0010	Quetiont is 00010				
2(b)	X=0101, Y=11111 using Restoring method Ans:				3M			
	Example of how the encoder Generates the Z_{in} control signal for the processor organization is given as. This signal is asserted during time slot T1 for all instructions, during T6 for an Add instruction, during T4 for an unconditional branch instruction				3M			
<div><div>Branch</div><div>Add</div><div><div>T₄</div><div>T₆</div><div>T₁</div><div>...</div><div>...</div></div></div>								

Generation of End control signal

$$\text{End} = T_7 \cdot \text{ADD} + T_5 \cdot \text{BR} + (T_5 \cdot N + T_4 \cdot \bar{N}) \cdot \text{BRN} + \dots$$

3M

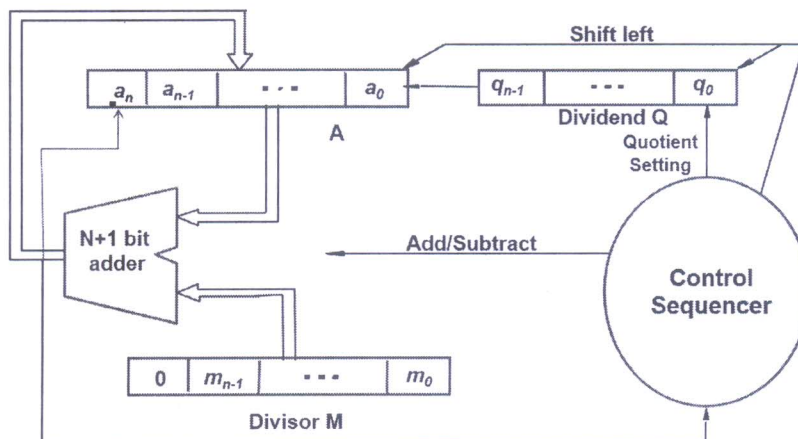


$M \leftarrow \text{divisor}, Q \leftarrow \text{dividend}, A \leftarrow 0$

2(c)

- Shift A and Q left one binary position
- Subtract M from A, and place the answer back in A
- If the sign of A is 1, set q_0 to 0 and add M back to A (restore A); otherwise (If the sign of A is 0), set q_0 to 1
- Repeat these steps n times

2M+4M



3(a)

- Instruction execution proceeds as follows.
- In step 1, the instruction fetch operation is initiated by loading the contents of the PC into MAR and sending the Read request to the memory.
- Select signal is set to Select4 which causes the MUX to select the constant 4. This value is added to the operand at input B, which is content of the PC and result is stored in register Z.
- The updated value is added to PC in step 2.
- In step 3 the word is fetched from memory is loaded into the register IR.
- Step 1 to 3 constitutes instruction fetch.
- At the beginning of step 4 instruction decoding is done by instruction decoder.
- Step 4 to 7 instruction execution
- The content of Register R3 loaded into MAR to read the content of memory location pointed by register R3, memory read operation is initiated.
- Content of R1 are transferred to register Y in step 5 to prepare the addition operation.
- When the read operation is completed, the memory operand is available in

6M

- register MDR and addition operation is performed in step 6.
- The content of MDR are gated to the bus and provided to ALU as B input.
- Register Y is selected as second input. Addition is performed

This can be achieved when instruction stages completes their operation without any interruption throughout their execution.

For a variety of reasons one of pipeline stage may not be able to complete its task for a given instruction in the allocated time.

Ex: stage E in the four-stage pipeline as shown in figure is responsible for arithmetic and logic operations and one clock cycle is assigned for this task. But divide operations needs more number of clock cycle to complete the execution.

In figure 8.3 instruction I3 needs 3 clock cycles to complete cycle 4 to 6 and write stage is not suppose to perform is operation.

Thus information in buffer B2 must be preserved unit completion of execution phase.

Thus stage 2 and stage 1 are blocked in accepting new instruction because the information in buffer B1 cannot be over written. Thus steps D4 and F5 are postponed

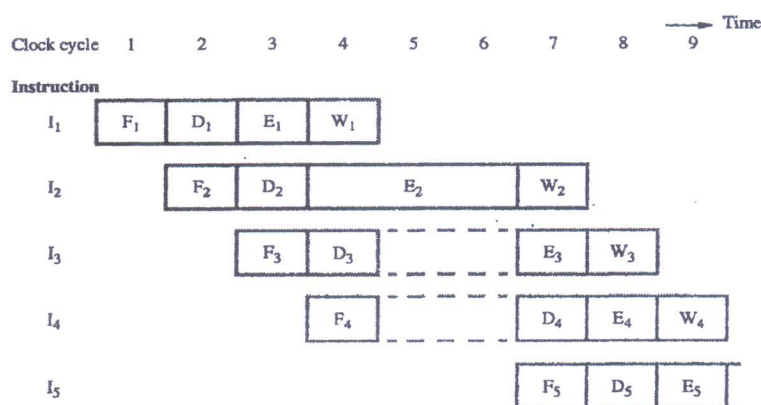


Figure 8.3 Effect of an execution operation taking more than one clock cycle.

Thus pipeline operation is said to stalled for two clock cycle (stall-means stage of pipeline not performs its specified task on designated clock cycle).

Any condition that causes the stalls is called Hazard.

Processor with only one bus, only one data can be transferred in one clock cycle. With this performance of system reduces.

By this number of elementary operations are more. To reduce the number of steps for instruction execution, most commercial processor uses multiple buses.

4(a) Fig A. shows the processor with three internal buses used to connect the registers, and ALU of processor. All general purpose registers (GPRs) are combined into single block called register file.

Register file have three ports (three buses).

There are two outputs ports, allowing the content of two registers to be accessed simultaneously and their contents are placed over the buses A and B.

The third port allows the data on bus C to be loaded into a third register during the same clock.

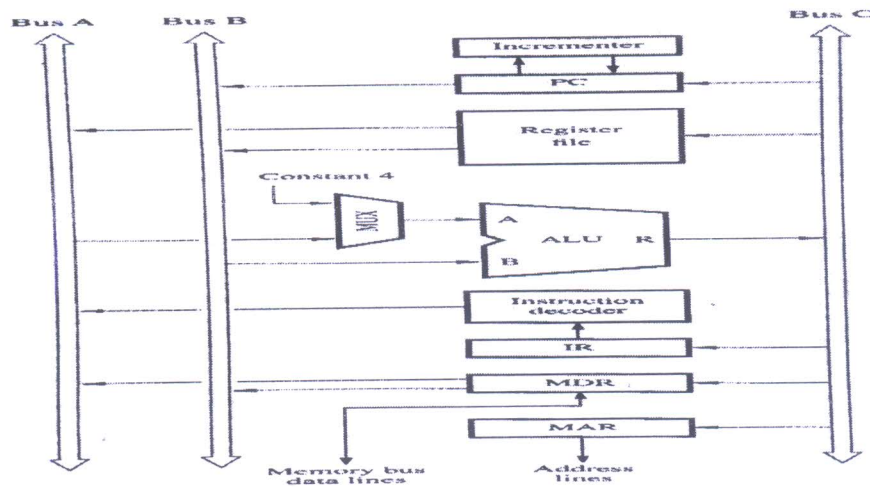
Buses A and B are used to transfer the source operands to the A and B inputs of ALU, where an arithmetic logical operations must be performed.

The result is transferred to the destination over the bus C

While the steps still have to be performed by sequentially changing from one step to another, the number of independent steps can be clubbed into single step in a multiple

bus organization.

Improvement in performance in a multiple bus organization



4(b)

- Instruction execution proceeds as follows.
- In step 1, the instruction fetch operation is initiated by loading the contents of the PC into MAR and sending the Read request to the memory.
- Select signal is set to Select4 which causes the MUX to select the constant 4. This value is added to the operand at input B, which is content of the PC and result is stored in register Z.
- The updated value is added to PC in step 2.
- In step 3 the word is fetched from memory is loaded into the register IR.
- Step 1 to 3 constitutes instruction fetch.
- At the beginning of step 4 instruction decoding is done by instruction decoder.
- Step 4 to 7 instruction execution
- The content of Register R3 loaded into MAR to read the content of memory location pointed by register R3, memory read operation is initiated.
- Content of R1 are transferred to register Yin step5 to prepare the addition operation.
- When the read operation is completed, the memory operand is available in register MDR and addition operation is performed in step 6.
- The content of MDR are gated to the bus and provided to ALU as B input.
- Register Y is selected as second input. Addition is performed

4M+2M

Step Action

1	PC _{out} , MAR _{in} , Read, Select4, Add, Z _{in}
2	Z _{out} , PC _{in} , Y _{in} , WMFC
3	MDR _{out} , IR _{in}
4	R3 _{out} , MAR _{in} , Read
5	R1 _{out} , Y _{in} , WMFC
6	MDR _{out} , SelectY, Add, Z _{in}
7	Z _{out} , R1 _{in} , End

Course In-charge

Module Coordinator

HOD-CSE

Head of the Department
Dept. of Computer Science
K.S. Institute of Technology
Bengaluru -560 109

K.S. INSTITUTE OF TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE & ENGG.
UPDATED LIST OF STUDENTS STUDYING IN III A SEC
FOR THE ACADEMIC YEAR 2020-2021

Computer Organization-18CS34

SI No.	USN	NAME OF THE STUDENT	IA1	IA2	IA3	IA Avg	Assi gnment	Total	Signature
1	1KS19CS001	AAKRITI	25	28	30	28	10	38	Aakriti
2	1KS19CS002	ABHISHEK B	30	30	17	26	10	36	Abhishek B
3	1KS19CS003	ABHISHEK YADAV	30	30	11	24	10	34	Abhishek Y
4	1KS19CS004	ADITH KARTHIK RAJU	30	6	24	20	10	30	Adith
5	1KS19CS005	AJAY S KALBURGI	30	30	22	28	10	38	Ajay S. K
6	1KS19CS006	AKASH A S	30	26	30	29	10	39	Akash A S
7	1KS19CS007	AMAN KUSHWAHA	30	30	19	27	10	37	Aman
8	1KS19CS008	AMILINENI HARINI	0	0					
9	1KS19CS009	AMIT K B	30	30	24	28	10	38	Amit
10	1KS19CS010	AMOGHA H S	30	27	18	25	10	35	Amogha
11	1KS19CS011	AMRUTHA K H	29	30	30	30	10	40	Amrutha
12	1KS19CS012	ANAGHA A HEBBAR	29	30	30	30	10	40	Anagha
13	1KS19CS014	ANUSHA B	28	30	25	28	10	38	Anusha
14	1KS19CS015	AQSA AQEEL	25	30	28	28	10	38	Aqsa
15	1KS19CS016	ASHIKA H N	30	30	29	30	10	40	Ashika
16	1KS19CS017	BHOOMIKA A M	29	30	29	30	10	40	Bhoomika A.M
17	1KS19CS018	BHOOMIKA K	30	30	29	30	10	40	Bmo
18	1KS19CS019	BHUMIKA M	30	30	24	28	10	38	Bhumika
19	1KS19CS020	C N SHREYAS	30	30	24	28	10	38	C.N.Shreyas
20	1KS19CS021	CHAITANYA	30	30	30	30	10	40	Chaitanya
21	1KS19CS022	CHANDAN B V	30	30	10	24	10	34	Chandana
22	1KS19CS023	DEEKSHA NAIDU R	29	30	29	30	10	40	Deeksha
23	1KS19CS024	DEEPA G	30	30	26	29	10	39	Deepa
24	1KS19CS025	DEEPTHI.N.K	20	30	28	30	10	40	Deepthi.Nk
25	1KS19CS026	DEVI PRASAD N	28	24	18	24	10	34	Devi
26	1KS19CS028	DHEEMANTH G	30	27	9	22	10	32	Dheemanth
27	1KS19CS029	DINESH M	30	23	22	25	10	35	Dinesh
28	1KS19CS030	G PRERITHA	30	28	25	28	10	38	Gpreritha
29	1KS19CS031	GAGAN REDDY S	30	28	15	25	10	35	Gagan Reddy
30	1KS19CS032	GAGANDEEP K	29	30	9	23	10	33	Gagan
31	1KS19CS033	GEETHANJALI B K	30	30	26	29	10	39	Geetha
32	1KS19CS034	GULSHAN KUMAR S	30	29	29	30	10	40	Gulshan
33	1KS19CS035	HARSHITHA J	30	30	22	28	10	38	Harshitha
34	1KS19CS036	INDRAJITH H M	30	30	28	30	10	40	Indrajith
35	1KS19CS037	JEEVAN T O	30	30					Jeevan
36	1KS19CS038	K KISHAN	30	30	15	26	10	36	K.Kishan

37	1KS19CS039	K R VAGEESH	30	30	30	30	40	40	Vageesh
38	1KS19CS040	KALYAN CHOWDHARY	27	30	11	23	10	33	Kalyan
39	1KS19CS041	KARTHIK S MORAJKAR	30	30	30	30	40	40	Karthik
40	1KS19CS042	KAVYASHREE.S.L	30	30	30	30	10	40	Kavya
41	1KS19CS043	KEERTHAN GOWDA S	30	29	30	30	10	40	Keerth
42	1KS19CS044	KOTHAPALLI SREEJA	21	30	24	25	10	35	K. Sreeja
43	1KS19CS045	KOTTALA SAIYENKATA SUCHITHA	25	30	10	22	10	32	K. Suchitha
44	1KS19CS046	KRISHNA K R	25	30	05	20	10	30	K. R. K.
45	1KS19CS047	KUMAR S	29	30	0	20	10	30	Kumar
46	1KS19CS048	LIKITH G	16	30	14	20	10	30	Likith
47	1KS19CS049	LISHA C	30	29	23	38	10	38	Lisha C.
48	1KS19CS050	MAHAK SHREE	30	30	30	30	10	40	Mahak
49	1KS19CS051	MALLIPALLI SPURTHI	30	30	29	30	10	40	Spurthi
50	1KS19CS052	MANASA G L	29	30	23	28	10	38	Manasa
51	1KS19CS053	MOHAMMED NOOR	30	30	05	22	10	32	Amma
52	1KS19CS054	MUKESH KUMAR	30	30	14	25	10	35	Mukesh
53	1KS19CS055	MYTHREYI U	30	30	30	30	10	40	Mythreya
54	1KS19CS056	N ASHOK	30	30	26	29	10	39	N. Ashok
55	1KS19CS057	N.BHAVYA	26	30	22	26	10	36	N. Bhavya
56	1KS19CS058	N P SHASHANKA RAO	25	30	17	24	10	34	N. P. Shashanka
57	1KS18CS011	BHARATH R	30	27	18	25	10	35	Bharath
58	1KS19CS116	VRATHIKA BILLAVA	29	19	20	23	10	33	Vrathika
59	1KS20CS400	AKIF DELVI	30	30	20	27	10	37	Akif
60	1KS20CS404	PRANAV CHANDRAN P	18	18	12	16	10	26	Pranav
61	1KS20CS402	KEERTHI KUMAR V	21	24	13	20	10	30	Keerthi

deefa

W. Murthy
Head of the Department
Dept. of Computer Science & Engg.
K.S. Institute of Technology
Bengaluru -560 109



K.S. INSTITUTE OF TECHNOLOGY, BANGALORE
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

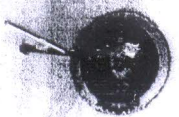
YEAR / SEMESTER	II / III
COURSE TITLE	COMPUTER ORGANIZATION
COURSE CODE	18CS34
ACADEMIC YEAR	2020-2021
INTERNAL	II

SI.NO.	USN	NAME	Q.N. 1A	Q.N. 1B	Q.N. 1C	Q.N. 2A	Q.N. 2B	Q.N. 2C	Q.N. 3A	Q.N. 3B	Q.N. 4A	Q.N. 4B	IA TOT	CO'S			Assignmen	Total
			CO3	CO3	CO3	CO3	CO3	CO3	CO2	CO4	CO2	CO4		CO2	CO3	CO4		
			6 Marks	6 Marks	6 Marks	6 Marks	6 Marks	6 Marks	6 Marks	6 Marks	6 Marks	6 Marks	30 Marks	6 Marks	18 Marks	6 Marks	10 marks	40 Marks
1	1KS19CS001	AAKRITI	6	6	6				5	5			28	6	18	5	10	38
2	1KS19CS002	ABHISHEK B	6	6	6				6	6			30	6	18	6	10	40
3	1KS19CS003	ABHISHEK YADAV	6	6	6				6	6			30	6	18	6	10	40
4	1KS19CS004	ADITH KARTHIK RAJU	0	0	0				0	6			6	6	0	6	10	16
5	1KS19CS005	AJAY S KALBURGI	6	6	6						6	6	30	6	18	6	10	40
6	1KS19CS006	AKASH A S	6	6	6				2	6			26	2	18	6	10	36
7	1KS19CS007	AMAN KUSHWAHA	6	6	6				6	6			30	6	18	6	10	40
8	1KS19CS009	AMIT K B	6	6	6						6	6	30	6	18	6	10	40
9	1KS19CS010	AMOGHA H S	6	6	6				6	3			27	6	18	3	10	37
10	1KS19CS011	AMRUTHA K H				6	6	6	6	6			30	6	18	6	10	40
11	1KS19CS012	ANAGHA A HEBBAR	6	6	6				6	6			30	6	18	6	10	40
12	1KS19CS014	ANUSHA B	6	6	6						6	6	30	6	18	6	10	40
13	1KS19CS015	AQSA AQEEL	6	6	6				6	6			30	6	18	6	10	40
14	1KS19CS016	ASHIKA H N	6	6	6				6	6			30	6	18	6	10	40
15	1KS19CS017	BHOOMIKA A M	6	6	6						6	6	30	6	18	6	10	40
16	1KS19CS018	BHOOMIKA K	6	6	6				6	6			30	6	18	6	10	40
17	1KS19CS019	BHUMIKA M	6	6	6				6	6			30	6	18	6	10	40
18	1KS19CS020	C N SHREYAS	6	6	6				6	6			30	6	18	6	10	40
19	1KS19CS021	CHAITANYA SHIVARAJU	6	6	6				6	6			30	6	18	6	10	40
20	1KS19CS022	CHANDAN B.V	6	6	6				6	6			30	6	18	6	10	40
21	1KS19CS023	DEEKSHA NAIDU R	6	6	6						6	6	30	6	18	6	10	40
22	1KS19CS024	DEEPA G	6	6	6						6	6	30	6	18	6	10	40

23	1KS19CS025	DEEPTI.N.K	6	6	6					6	6	30	6	18	6	10	40	
24	1KS19CS026	DEVI PRASAD N	6	6	6					6		24	6	18	0	10	34	
25	1KS19CS028	DHEEMANTH G	6	6	6			6	3			27	6	18	3	10	37	
26	1KS19CS029	DINESH M	5	6	6			6				23	6	17	0	10	33	
27	1KS19CS030	G PRERITHA	6	6	6			4	6			28	4	18	6	10	38	
28	1KS19CS031	GAGAN REDDY S	6	6	6			4	6			28	4	18	6	10	38	
29	1KS19CS032	GAGANDEEP K	6	6	6			6	6			30	6	18	6	10	40	
30	1KS19CS033	GEETHANJALI B K PRASAD	6	6	6			6	6			30	6	18	6	10	40	
31	1KS19CS034	GULSHAN KUMAR S	5	6	6					6	6	29	6	17	6	10	39	
32	1KS19CS035	HARSHITHA J	6	6	6			6	6			30	6	18	6	10	40	
33	1KS19CS036	INDRAJITH H M	6	6	6			6	6			30	6	18	6	10	40	
34	1KS19CS038	K KISHAN	6	6	6			6	6			30	6	18	6	10	40	
35	1KS19CS039	K R VAGEESH	6	6	6			6	6			30	6	18	6	10	40	
36	1KS19CS040	KALYAN CHOWDHARY B	6	6	6			6	6			30	6	18	6	10	40	
37	1KS19CS041	KARTHIK S MORAJKAR	6	6	6			6	6			30	6	18	6	10	40	
38	1KS19CS042	KAVYASHREE.S.L				6	6	6			6	6	30	6	18	6	10	40
39	1KS19CS043	KEERTHAN GOWDA S				6	6	6	6	5		29	6	18	5	10	39	
40	1KS19CS044	KOTHAPALLI SREEJA	6	6	6			6	6			30	6	18	6	10	40	
41	1KS19CS045	KOTTALA SAIVENKATASUCHITH	6	6	6			6	6			30	6	18	6	10	40	
42	1KS19CS046	KRISHNA K R	6	6	6			6	6			30	6	18	6	10	40	
43	1KS19CS047	KUMAR S	6	6	6					6	6	30	6	18	6	10	40	
44	1KS19CS048	LIKITH G	6	6	6			6	6			30	6	18	6	10	40	
45	1KS19CS049	LISHA C	5	6	6					6	6	29	6	17	6	10	39	
46	1KS19CS050	MAHAK SHREE	6	6	6			6	6			30	6	18	6	10	40	
47	1KS19CS051	MALLIPALLI SPURTHI REDDY	6	6	6			6	6			30	6	18	6	10	40	
48	1KS19CS052	MANASA G L	6	6	6			6	6			30	6	18	6	10	40	
49	1KS19CS053	MOHAMMED NOOR AMAN				6	6	6	6	6		30	6	18	6	10	40	
50	1KS19CS054	MUKESH KUMAR	6	6	6			6	6			30	6	18	6	10	40	
51	1KS19CS055	MYTHREYI U	6	6	6					6	6	30	6	18	6	10	40	
52	1KS19CS056	N ASHOK	6	6	6			6	6			30	6	18	6	10	40	
53	1KS19CS057	N BHAVYA	6	6	6			6	6			30	6	18	6	10	40	
54	1KS19CS058	N P SHASHANKA RAO	6	6	6			6	6			30	6	18	6	10	40	
55	1KS18CS011	BHARATH R	6	6	6			6	3			27	6	18	3	10	37	
56	1KS19CS116	VRATHIKA BILLAVA				6		1	6	6		19	6	7	6	10	29	
57	1KS20CS400	AKIF DELVI	6	6	6				6	6		30	6	18	6	10	40	
58	1KS20CS404	PRANAV CHANDRAN	6	6		6					6	18	6	12	0	10	28	
59	1KS20CS402	KEERTHI KUMAR				6	6				6	6	24	6	12	6	10	34

deepa

Deepa
Head of the Department
Dept. of Computer Science & Engg.
K.S. Institute of Technology
Bengaluru -560 109



Visvesvaraya Technological University

K.S. INSTITUTE OF TECHNOLOGY, BANGALORE

Branch : CS

Scheme : 2018

Semester : 3

Sl NO.	USN	18CS32	18CS33	18CS34	18CS35	18CS36	18CSL37	18CSL38	18KAK39	18KVK39	18MAT31	18MATDIP31	STUDENT SIGNATURE
1	1KS18CS011	26	32	35	29	27	36	32	82	-	25	-	
2	1KS18CS021	29	38	33	32	22	29	36	-	100	32	-	
3	1KS18CS068	36	34	29	33	29	31	39	50	-	30	-	
4	1KS19CS001	38	39	38	37	32	34	39	-	75	40	-	
5	1KS19CS002	34	35	36	30	35	35	36	89	-	33	-	
6	1KS19CS003	30	34	34	33	33	30	30	-	65	36	-	
7	1KS19CS004	36	35	30	30	33	31	36	89	-	33	-	
8	1KS19CS005	37	37	38	32	35	32	38	74	-	39	-	
9	1KS19CS006	38	40	39	37	34	33	39	99	-	35	-	
10	1KS19CS007	32	25	37	26	36	35	36	-	67	30	-	
11	1KS19CS009	38	34	38	33	32	35	37	99	-	39	-	
12	1KS19CS010	37	37	35	34	37	32	37	88	-	39	-	
13	1KS19CS011	38	40	40	35	38	36	36	-	75	40	-	
14	1KS19CS012	37	40	40	37	35	39	40	-	93	39	-	
15	1KS19CS014	35	35	38	38	37	31	38	98	-	39	-	
16	1KS19CS015	39	35	38	27	37	35	38	-	90	37	-	
17	1KS19CS016	39	39	40	32	34	37	35	89	-	35	-	
18	1KS19CS017	39	40	40	38	37	40	40	100	-	40	-	
19	1KS19CS018	39	40	40	37	39	40	40	100	-	40	-	
20	1KS19CS019	39	40	38	38	37	37	40	98	-	40	-	
21	1KS19CS020	37	39	38	34	34	38	34	97	-	40	-	
22	1KS19CS021	39	40	40	39	39	39	40	98	-	40	-	
23	1KS19CS022	34	26	34	29	34	33	34	82	-	31	-	
24	1KS19CS023	37	40	40	36	36	37	36	-	98	37	-	
25	1KS19CS024	38	39	39	35	34	37	39	97	-	39	-	
26	1KS19CS025	37	39	40	35	35	38	40	93	-	40	-	
27	1KS19CS026	35	37	34	30	31	35	32	-	74	38	-	
28	1KS19CS028	34	36	32	33	34	33	34	85	-	39	-	
29	1KS19CS029	32	33	35	33	27	30	32	81	-	30	-	
30	1KS19CS030	38	38	38	35	36	33	32	-	95	37	-	
31	1KS19CS031	25	29	35	29	21	30	32	84	-	23	-	
32	1KS19CS032	32	31	33	31	33	33	34	69	-	27	-	
33	1KS19CS033	39	40	39	36	37	38	40	93	-	38	-	
34	1KS19CS034	39	34	40	35	37	37	40	-	92	40	-	
35	1KS19CS035	39	40	38	36	39	38	40	99	-	40	-	
36	1KS19CS036	40	40	40	37	39	40	40	100	-	39	-	
37	1KS19CS038	34	36	36	35	33	39	35	96	-	30	-	
38	1KS19CS039	39	40	40	37	37	40	40	99	-	40	-	
39	1KS19CS040	37	34	33	32	27	37	32	-	92	33	-	
40	1KS19CS041	39	40	40	34	38	39	40	93	-	40	-	
41	1KS19CS042	40	39	40	37	38	39	39	96	-	40	-	
42	1KS19CS043	40	39	40	35	39	40	40	100	-	40	-	
43	1KS19CS044	37	37	35	36	37	39	39	-	87	38	-	
44	1KS19CS045	37	38	32	30	33	35	33	-	92	33	-	
45	1KS19CS046	33	33	30	30	34	35	35	-	93	30	-	
46	1KS19CS047	22	26	30	27	21	26	25	70	-	23	-	
47	1KS19CS048	29	31	30	29	24	30	35	86	-	32	-	
48	1KS19CS049	36	37	38	33	31	39	35	95	-	37	-	

SI NO.	USN	18CS32	18CS33	18CS34	18CS35	18CS36	18CSL37	18CSL38	18KAK39	18KVK39	18MAT31	18MATDIP31	SIGN
51	1KS19CS052	36	38	38	36	36	39	38	96	-	36	-	
52	1KS19CS053	31	38	32	29	29	29	25	77	-	34	-	
53	1KS19CS054	31	33	35	33	28	31	33	-	89	31	-	
54	1KS19CS055	39	40	40	38	40	40	40	97	-	40	-	
55	1KS19CS056	36	40	39	37	33	35	33	91	-	39	-	
56	1KS19CS057	33	40	36	39	37	39	39	-	95	40	-	
57	1KS19CS058	38	37	34	33	39	38	40	-	98	33	-	
58	1KS19CS059	38	40	35	33	35	35	39	94	-	35	-	
59	1KS19CS060	38	40	36	35	38	36	39	99	-	40	-	
60	1KS19CS061	39	40	37	31	37	33	40	-	100	40	-	
61	1KS19CS062	36	37	34	32	30	31	36	82	-	38	-	
62	1KS19CS063	40	40	38	36	34	38	39	98	-	40	-	
63	1KS19CS064	37	39	36	31	35	38	35	85	-	36	-	
64	1KS19CS065	37	38	36	34	34	32	36	-	93	38	-	
65	1KS19CS066	38	38	36	33	31	34	37	97	-	37	-	
66	1KS19CS067	38	38	39	38	39	38	39	92	-	40	-	
67	1KS19CS068	34	34	35	38	30	30	40	-	95	36	-	
68	1KS19CS069	21	31	23	26	22	25	32	-	99	33	-	
69	1KS19CS070	40	39	38	39	40	39	40	98	-	39	-	
70	1KS19CS071	38	40	38	38	38	37	40	-	97	40	-	
71	1KS19CS072	38	40	38	37	37	33	40	94	-	40	-	
72	1KS19CS073	37	35	37	35	33	25	36	-	97	39	-	
73	1KS19CS074	38	36	36	32	36	31	40	92	-	36	-	
74	1KS19CS075	36	38	32	34	39	38	40	61	-	39	-	
75	1KS19CS076	35	28	33	28	30	29	32	-	77	38	-	
76	1KS19CS077	38	38	34	33	37	35	35	99	-	39	-	
77	1KS19CS078	36	40	35	35	36	34	36	97	-	40	-	
78	1KS19CS079	37	38	38	37	33	36	36	94	-	37	-	
79	1KS19CS080	35	37	36	36	33	31	34	100	-	40	-	
80	1KS19CS081	35	37	32	31	32	27	39	-	95	36	-	
81	1KS19CS082	26	24	21	33	29	28	35	-	80	31	-	
82	1KS19CS083	38	39	29	31	31	34	39	93	-	35	-	
83	1KS19CS084	37	38	33	39	37	26	35	98	-	40	-	
84	1KS19CS085	40	39	38	36	38	36	40	99	-	40	-	
85	1KS19CS086	39	40	35	38	35	35	38	98	-	40	-	
86	1KS19CS087	39	39	37	38	35	35	39	97	-	40	-	
87	1KS19CS088	40	40	38	36	37	35	40	97	-	38	-	
88	1KS19CS089	39	37	34	38	35	34	38	-	96	40	-	
89	1KS19CS090	27	26	26	34	25	27	34	90	-	25	-	
90	1KS19CS091	35	36	35	38	34	29	37	88	-	38	-	
91	1KS19CS092	35	34	31	33	33	29	37	96	-	34	-	
92	1KS19CS093	35	36	32	33	33	25	36	73	-	35	-	
93	1KS19CS094	39	37	38	37	36	33	40	74	-	39	-	
94	1KS19CS096	28	35	29	29	30	25	32	99	-	33	-	
95	1KS19CS097	35	39	34	36	34	31	37	99	-	38	-	
96	1KS19CS098	39	40	35	34	35	35	36	100	-	40	-	
97	1KS19CS099	33	34	27	30	30	25	40	-	97	34	-	
98	1KS19CS100	35	34	27	29	34	25	35	88	-	32	-	
99	1KS19CS101	37	35	32	31	36	29	36	74	-	37	-	
100	1KS19CS102	33	39	35	34	36	26	36	73	-	37	-	
101	1KS19CS103	28	33	28	31	28	26	30	66	-	30	-	
102	1KS19CS104	36	40	38	38	35	32	40	97	-	40	-	
103	1KS19CS105	36	36	29	20	28	26	32	96	-	35	-	
104	1KS19CS106	37	38	38	31	34	26	34	66	-	34	-	
105	1KS19CS107	37	39	34	36	34	34	40	99	-	39	-	
106	1KS19CS108	21	20	22	20	22	25	25	-	86	23	-	
107	1KS19CS109	39	40	32	30	29	25	36	74	-	34	-	
108	1KS19CS110	38	40	39	38	34	38	40	75	-	40	-	
109	1KS19CS111	34	33	29	34	30	26	40	-	99	33	-	

	USN	18CS32	18CS33	18CS34	18CS35	18CS36	18CSL37	18CSL38	18KAK39	18KVK39	18MAT31	18MATDIP31	STUDENT SIGNATURE
111	1KS19CS113	40	40	38	36	40	39	40	74	-	40	-	
112	1KS19CS115	40	40	40	32	40	40	40	90	-	40	-	
113	1KS19CS116	39	34	33	34	39	37	37	99	-	40	-	
114	1KS20CS400	36	36	37	33	36	38	35	79	-	33	26	
115	1KS20CS401	30	33	29	33	30	25	35	94	-	33	26	
116	1KS20CS402	33	33	30	29	34	36	37	94	-	33	31	
117	1KS20CS403	34	31	30	27	30	28	36	94	-	28	26	
118	1KS20CS404	25	24	26	29	21	30	35	89	-	23	23	
-x-	Faculty Signature	<i>[Signature]</i>	<i>[Signature]</i>	<i>[Signature]</i>	<i>[Signature]</i>	<i>[Signature]</i>	<i>[Signature]</i>	<i>[Signature]</i>	<i>[Signature]</i>	<i>[Signature]</i>	<i>[Signature]</i>	<i>[Signature]</i>	-----XXXXXXXX-----

* - values are either optional subjects or the faculty has not yet entered the marks

[Signature]
HOD

Seal and Signature

Head of the Department
Dept. of Computer Science & Engg.
K.S. Institute of Technology
Bengaluru -560 109

[Signature]
PRINCIPAL

Seal and Signature

PRINCIPAL
K.S. INSTITUTE OF TECHNOLOGY
BENGALURU - 560 109



K S Institute of Technology, Bangalore-109.

Computer Science & Engineering

COMPUTER ORGANIZATION -18CS34

III Sem -Computer Science and Engineering

Sl.No	USN	Name of the Student	18CS34		
			IA	EX	TOT
1	1KS19CS001	AAKRITI	38	38	76
2	1KS19CS002	ABHISHEK B	36	22	58
3	1KS19CS003	ABHISHEK YADAV	34	21	55
4	1KS19CS004	ADITH KARTHIK RAJU	30	21	51
5	1KS19CS005	AJAY S KALBURGI	38	33	71
6	1KS19CS006	AKASH A S	39	30	69
7	1KS19CS007	AMAN KUSHWAHA	37	23	60
8	1KS19CS009	AMIT K B	38	33	71
9	1KS19CS010	AMOGHA H S	35	21	56
10	1KS19CS011	AMRUTHA K H	40	45	85
11	1KS19CS012	ANAGHA A HEBBAR	40	22	62
12	1KS19CS014	ANUSHA B	38	21	59
13	1KS19CS015	AQSA AQEEL	38	37	75
14	1KS19CS016	ASHIKA H N	40	26	66
15	1KS19CS017	BHOOMIKA A M	40	32	72
16	1KS19CS018	BHOOMIKA K	40	34	74
17	1KS19CS019	BHUMIKA M	38	35	73
18	1KS19CS020	C N SHREYAS	38	29	67
19	1KS19CS021	CHAITANYA SHIVARAJU	40	38	78
20	1KS19CS022	CHANDAN B V	34	22	56
21	1KS19CS023	DEEKSHA NAIDU R	40	11	51
22	1KS19CS024	DEEPA G	39	29	68
23	1KS19CS025	DEEPTHI.N.K	40	30	70
24	1KS19CS026	DEVI PRASAD N	34	29	63
25	1KS19CS028	DHEEMANTH G	32	14	46
26	1KS19CS029	DINESH M	35	24	59
27	1KS19CS030	G PRERITHA	38	24	62
28	1KS19CS031	GAGAN REDDY S	35	11	46
29	1KS19CS032	GAGANDEEP K	33	21	54
30	1KS19CS033	GEETHANJALI B K PRASAD	39	27	66
31	1KS19CS034	GULSHAN KUMAR S	40	29	69
32	1KS19CS035	HARSHITHA J	38	34	72
33	1KS19CS036	INDRAJITH H M	40	29	69
34	1KS19CS038	K KISHAN	36	26	62
35	1KS19CS039	K R VAGEESH	40	31	71
36	1KS19CS040	KALYAN CHOWDHARY B	33	21	54
37	1KS19CS041	KARTHIK S MORAJKAR	40	26	66
38	1KS19CS042	KAVYASHREE.S.L	40	34	74
39	1KS19CS043	KEERTHAN GOWDA S	40	38	78
40	1KS19CS044	KOTHAPALLI SREEJA	35	24	59

41	1KS19CS045	KOTTALA SAIVENKATASUCHITH	32	18	50
42	1KS19CS046	KRISHNA K R	30	21	51
43	1KS19CS047	KUMAR S	30	4	34
44	1KS19CS048	LIKITH G	30	21	51
45	1KS19CS049	LISHA C	38	32	70
46	1KS19CS050	MAHAK SHREE	40	42	82
47	1KS19CS051	MALLIPALLI SPURTHI REDDY	40	26	66
48	1KS19CS052	MANASA G L	38	29	67
49	1KS19CS053	MOHAMMED NOOR AMAN	32	21	53
50	1KS19CS054	MUKESH KUMAR	35	21	56
51	1KS19CS055	MYTHREYI U	40	30	70
52	1KS19CS056	N ASHOK	39	31	70
53	1KS19CS057	N BHAVYA	36	25	61
54	1KS19CS058	N P SHASHANKA RAO	34	21	55
55	1KS18CS011	BHARATH R	35	6	41
56	1KS19CS116	VRATHIKA BILLAVA	33	33	66
57	1KS20CS400	AKIF DELVI	37	22	59
58	1KS20CS402	KEERTHI KUMAR V	30	14	44
59	1KS20CS404	PRANAV CHANDRAN P	26	21	47

IA-Internal Assesment

Ex-VTU End Sem Examination

Total Students=59

Pass=53

Fail=6

Percentage=89.83

Deeba

Pranav

YEAR / SEMESTER	II / III			
COURSE TITLE	COMPUTER ORGANIZATION			
COURSE CODE	18CS34			
ACADEMIC YEAR	2020-2021			
INTERNALS	I			

Sl.NO.	USN	NAME	Q.N.1A	Q.N.1B	Q.N.1C	Q.N.2A	Q.N.2B	Q.N.2C	Q.N.3A	Q.N.3B	Q.N.4A	Q.N.4B	IA TOTAL	CO'S		Assignm ent	Total
			CO1	CO1	CO1	CO1	CO1	CO1	CO2	CO2	CO2	CO2		CO1	CO2		
			6 Marks	6 Marks	6 Marks	6 Marks	6 Marks	6 Marks	6 Marks	6 Marks	6 Marks	6 Marks	30 Marks	18 Marks	12 Marks	10 marks	40 Marks
1	1KS19CS001	AAKRITI	6	6	6				1	6			25	18	7	10	35
2	1KS19CS002	ABHISHEK B	6	6	6				6	6			30	18	12	10	40
3	1KS19CS003	ABHISHEK YADAV	6	6	6						6	6	30	18	12	10	40
4	1KS19CS004	ADITH KARTHIK RAJU	6	6	6						6	6	30	18	12	10	40
5	1KS19CS005	AJAY S KALBURGI	6	6	6						6	6	30	18	12	10	40
6	1KS19CS006	AKASH A S	6	6	6				6	6			30	18	12	10	40
7	1KS19CS007	AMAN KUSHWAHA	6	6	6						6	6	30	18	12	10	40
8	1KS19CS009	AMIT K B	6	6	6						6	6	30	18	12	10	40
9	1KS19CS010	AMOGHA H S	6	6	6						6	6	30	18	12	10	40
10	1KS19CS011	AMRUTHA K H	5	6	6				6	6			29	17	12	10	39
11	1KS19CS012	ANAGHA A HEBBAR	6	6	6						5	6	29	18	11	10	39
12	1KS19CS014	ANUSHA B	6	6	6						6	4	28	18	10	10	38
13	1KS19CS015	AQSA AQEEL	5	6	5				3	6			25	16	9	10	35
14	1KS19CS016	ASHIKA H N	6	6	6				6	6			30	18	12	10	40
15	1KS19CS017	BHOOMIKA A M	6	6	6						5	6	29	18	11	10	39
16	1KS19CS018	BHOOMIKA K	6	6	6				6	6			30	18	12	10	40
17	1KS19CS019	BHUMIKA M	6	6	6						6	6	30	18	12	10	40
18	1KS19CS020	C N SHREYAS	6	6	6				6	6			30	18	12	10	40
19	1KS19CS021	CHAITANYA SHIVARAJU	6	6	6						6	6	30	18	12	10	40
20	1KS19CS022	CHANDAN .B.V	6	6	6						6	6	30	18	12	10	40
21	1KS19CS023	DEEKSHA NAIDU R	6	6	6						5	6	29	18	11	10	39
22	1KS19CS024	DEEPA G	6	6	6				6	6			30	18	12	10	40

23	1KS19CS025	DEEPTHI.N.K	6	6	6				6	6			30	18	12	10	40
24	1KS19CS026	DEVI PRASAD N	4	6	6						6	6	28	16	12	10	38
25	1KS19CS028	DHEEMANTH G	6	6	6						6	6	30	18	12	10	40
26	1KS19CS029	DINESH M	6	6	6						6	6	30	18	12	10	40
27	1KS19CS030	G PRERITHA	6	6	6						6	6	30	18	12	10	40
28	1KS19CS031	GAGAN REDDY S	6	6	6						6	6	30	18	12	10	40
29	1KS19CS032	GAGANDEEP K	6	6	5				6	6			29	17	12	10	39
30	1KS19CS033	GEETHANJALI B K PRASAD	6	6	6				6	6			30	18	12	10	40
31	1KS19CS034	GULSHAN KUMAR S	6	6	6						6	6	30	18	12	10	40
32	1KS19CS035	HARSHITHA J	6	6	6				6	6			30	18	12	10	40
33	1KS19CS036	INDRAJITH H M	6	6	6						6	6	30	18	12	10	40
34	1KS19CS038	K KISHAN	6	6	6				6	6	6	6	30	18	12	10	40
35	1KS19CS039	K R VAGEESH	6	6	6				6	6			30	18	12	10	40
36	1KS19CS040	KALYAN CHOWDHARY B	5	6	5						5	6	27	16	11	10	37
37	1KS19CS041	KARTHIK S MORAJKAR	6	6	6						6	6	30	18	12	10	40
38	1KS19CS042	KAVYASHREE.S.L				6	6	6			6	6	30	18	12	10	40
39	1KS19CS043	KEERTHAN GOWDA S	6	6	6				6	6	6	6	30	18	12	10	40
40	1KS19CS044	KOTHAPALLI SREEJA	5	6	6						2	2	21	17	4	10	31
41	1KS19CS045	KOTTALA SAIVENKATASUCHITH	5	6	6				2	6			25	17	8	10	35
42	1KS19CS046	KRISHNA K R	5	6	6				2	6			25	17	8	10	35
43	1KS19CS047	KUMAR S	6	6	6				5	6			29	18	11	10	39
44	1KS19CS048	LIKITH G	4	4	6				2				16	14	2	10	26
45	1KS19CS049	LISHA C	6	6	6						6	6	30	18	12	10	40
46	1KS19CS050	MAHAK SHREE	6	6	6						6	6	30	18	12	10	40
47	1KS19CS051	MALLIPALLI SPURTHI REDDY	6	6	6				6	6			30	18	12	10	40
48	1KS19CS052	MANASA G L	5	6	6						6	6	29	17	12	10	39
49	1KS19CS053	MOHAMMED NOOR AMAN	6	6	6						6	6	30	18	12	10	40
50	1KS19CS054	MUKESH KUMAR	6	6	6				6	6			30	18	12	10	40
51	1KS19CS055	MYTHREYI U	6	6	6				6	6			30	18	12	10	40
52	1KS19CS056	N ASHOK	6	6	6						6	6	30	18	12	10	40
53	1KS19CS057	N BHAVYA	6	6	6				2	6			26	18	8	10	36
54	1KS19CS058	N P SHASHANKA RAO	5	6	6				2	6			25	17	8	10	35
55	1KS18CS011	BHARATH R	6	6	6						6	6	30	18	12	10	40
56	1KS19CS116	VRATHIKA BILLAVA	6	6	5				6	6			29	17	12	10	39
57	1KS20CS400	AKIF DELVI	6	6	6				6	6			30	18	12	10	40
58	1KS20CS404	PRANAV CHANDRAN	6	6	4				4		4		18	14	4	10	28
59	1KS20CS402	KEERTHI KUMAR	6	5					5	5			21	11	10	10	31

deefa

Murthy
Head of the Department
Dept. of Computer Science & Engg.
K.S. Institute of Technology
Bengaluru -560 109



K.S. INSTITUTE OF TECHNOLOGY, BANGALORE - 560109
DEPARTMENT OF COMPUTER SCIENCES & ENGINEERING
TEACHING AND LEARNING
PEDAGOGY REPORT

Academic Year	2020-21
Name of the Faculty	Deepa .S.R
Course Name /Code	Computer Organization-18CS34
Semester/Section	III A
Activity Name	Quiz
Topic Covered	Module-1 Basic Structure of Computers
Date	--2020- 28-9-2020
No. of Participants	55
Objectives/Goals	To analyse the understanding of the students regarding Basic Structure of Computers
ICT Used	Google Classroom (Forms)
Appropriate Method/Instructional materials/Exam Questions 1. Where R1 is Source Operand-1, R2 is the Source Operand-2 and R3 is the Destination. This instruction adds the contents of Register R1 with the contents of R2 and the result is placed in R3 Register. The symbolic representation of this instruction is 2. _____ stores the address of the Memory Location 3. • _____ is a special purpose register used to hold the address of the next instruction to be executed 4. _____ holds the instruction to be executed 5. It performs arithmetic and logical operations on given data 6. Processor Execution Time is given by _____ 7. In _____ technique lower byte of data is assigned to higher address of the memory and higher byte of data is assigned to lower address of the memory. 8. ADD A, B, Z is _____ instruction 9. The _____ flag is set to 1 if the result is found to be out of this range. 10. MOVE R1,R2 is an example of _____ - addressing mode 11. ADD (R1), R0 is an example of _____ - addressing mode 12. ADD 20(R1), R2 Effective address is _____ 13. EA of an operand = X + (PC) is an example of _____ addressing 14. MOVE (R1)+, R2 for 32 bit machine R1 increments by _____ 15. 2. When SIN = _____, the processor reads the ASCII value of a character from DATAIN register into the Processor register 16. _____ is used to keep of the address of the element that is top of stack at any time 17. LshiftL #2, R0 shifts the data 0110....011 to obtain the result _____ 18. AShiftR #2, R0 shifts the data 10011....010 to obtain the result _____1 19. RotateLC #2, R0 on data 01110....011 gives the result _____ 20. RotateRC #2, R0 on data 01110....011 gives the result _____	
Significance of Results/Outcomes	Got to know how much students have understood module-1
Reflective Critique	

Proofs (Photographs/Videos/Reports/Charts/Models)

Google form link

<https://docs.google.com/forms/d/e/1FAIpQLSe-SonuApW08CYNQwxPilG3INEv84jHBVrDWofG7sE6q3kK2g/viewform>

Deefa
2/10/20

Signature of Course In charge

Deefa
2/10/20

Signature of HOD CSE

Head of the Department
Dept. of Computer Science & Engg.
K.S. Institute of Technology
Bengaluru -560 109

Pedaggogy-1 Assessment

SL. No	Name of student	USN	Assignment Marks Q1f
1	AMOGHA H S	1KS19CS010	2
2	ABHISHEK B	1KS19CS002	2
3	ABHISHEK YADAV	1KS19CS003	2
4	ADITH KARTHIK RAJU	1KS19CS004	2
5	AJAY S KALBURGI	1KS19CS005	2
6	AKASH A S	1KS19CS006	2
7	AMAN KUSHWAHA	1KS19CS007	2
8	AMILINENI HARINI	1KS19CS008	2
9	AMIT K B	1KS19CS009	2
10	AAKRITI	1KS19CS001	2
11	AMRUTHA K H	1KS19CS011	2
12	ANAGHA A HEBBAR	1KS19CS012	2
13	ANUSHA B	1KS19CS014	2
14	AQSA AQEEL	1KS19CS015	2
15	ASHIKA H N	1KS19CS016	2
16	BHOOMIKA A M	1KS19CS017	2
17	BHOOMIKA K	1KS19CS018	2
18	BHUMIKA M	1KS19CS019	2
19	C N SHREYAS	1KS19CS020	2
20	CHAITANYA SHIVARAJU	1KS19CS021	2
21	CHANDAN B.V	1KS19CS022	2
22	DEEKSHA NAIDU R	1KS19CS023	2
23	DEEPA G	1KS19CS024	2
24	DEEPTHI.N.K	1KS19CS025	2
25	DEVI PRASAD N	1KS19CS026	2
26	DHEEMANTH G	1KS19CS028	2
27	DINESH M	1KS19CS029	2
28	G PRERITHA	1KS19CS030	2

29	GAGAN REDDY S	1KS19CS031	2
30	GAGANDEEP K	1KS19CS032	2
31	GEETHANJALI B K PRASAD	1KS19CS033	2
32	GULSHAN KUMAR S	1KS19CS034	2
33	HARSHITHA J	1KS19CS035	2
34	INDRAJITH H M	1KS19CS036	2
35	JEEVAN T O	1KS19CS037	2
36	K KISHAN	1KS19CS038	2
37	K R VAGEESH	1KS19CS039	2
38	KALYAN CHOWDHARY B	1KS19CS040	2
39	KARTHIK S MORAJKAR	1KS19CS041	2
40	KAVYASHREE.S.L	1KS19CS042	2
41	KEERTHAN GOWDA S	1KS19CS043	2
42	KOTHAPALLI SREEJA	1KS19CS044	2
43	KOTTALA SAIVENKATASUCHITH	1KS19CS045	2
44	KRISHNA K R	1KS19CS046	2
45	KUMAR S	1KS19CS047	2
46	LIKITH G	1KS19CS048	2
47	LISHA C	1KS19CS049	2
48	MAHAK SHREE	1KS19CS050	2
49	MALLIPALLI SPURTHI REDDY	1KS19CS051	2
50	MANASA G L	1KS19CS052	2
51	MOHAMMED NOOR AMAN	1KS19CS053	2
52	MUKESH KUMAR	1KS19CS054	2
53	MYTHREYI U	1KS19CS055	2
54	N ASHOK	1KS19CS056	2
55	N BHAVYA	1KS19CS057	2
56	N P SHASHANKA RAO	1KS19CS058	2
57	BHARATH R	1KS18CS011	2
56	VRATHIKA BILLAVA	1KS19CS116	2
57	AKIF DELVI	1KS20CS400	2
58	PRANAV CHANDRAN	1KS20CS404	2
59	KEERTHI KUMAR	1KS20CS402	2

deber

Drucapm



K.S. INSTITUTE OF TECHNOLOGY, BANGALORE - 560109
DEPARTMENT OF COMPUTER SCIENCES & ENGINEERING
TEACHING AND LEARNING
PEDAGOGY REPORT

Academic Year	2020-21
Name of the Faculty	Deepa .S.R
Course Name /Code	Computer Organization-18CS34
Semester/Section	III A
Activity Name	Quiz
Topic Covered	Module-2 Input/Output Organization
Date	--2020 29 October 2020
No. of Participants	55
Objectives/Goals	To analyse the understanding of the students regarding Input/Output Organization of computer
ICT Used	Google Classroom (Forms)
Appropriate Method/Instructional materials/Exam Questions 1. _____ Bus is unidirectional 2. Input-buffer associated with keyboard _____ 3. _____ refers to any event that causes an interruption 4. Transferring the block of data at high speed in between main memory and external device (I/O devices) without continuous intervention of CPU is called as _____ 5. Number of Registers a DMA Controller has _____ 6. The DMA controller requests the CPU to transfer new block of data from source to destination by activating this bit _____ 7. The DMA controller takes over the memory cycles from the CPU 8. In _____ technique CPU acts as a bus-master or any control unit connected to bus can be acts as a bus master 9. In case of _____ bus all the devices derive the timing information from common bus line 10. _____ Port transfers data in the form of a number of bits (8 or 16) simultaneously to or from the device 11. _____ defines an expansion bus on the motherboard	
Significance of Results/Outcomes	Got to know how much students have understood module-2
Reflective Critique	

Proofs (Photographs/Videos/Reports/Charts/Models)

Google form link

<https://docs.google.com/forms/d/e/1FAIpQLScSyusisZJap2JNqia89N9vcIKK4xU0dmkLGu2BU-LTEeEuRw/viewform>

Deeka
9/11/20

Signature of Course In charge

Manu ar 9/11/20.

Signature of HOD CSE

*Head of the Department
Dept. of Computer Science & Engg.
K.S. Institute of Technology
Bengaluru -560 109*

Pedagogy-2 Assessment

SL. No	Name of student	USN	Assignment Marks Q2v
1	AMOGHA H S	1KS19CS010	2
2	ABHISHEK B	1KS19CS002	2
3	ABHISHEK YADAV	1KS19CS003	2
4	ADITH KARTHIK RAJU	1KS19CS004	2
5	AJAY S KALBURGI	1KS19CS005	2
6	AKASH A S	1KS19CS006	2
7	AMAN KUSHWAHA	1KS19CS007	2
8	AMILINENI HARINI	1KS19CS008	2
9	AMIT K B	1KS19CS009	2
10	AAKRITI	1KS19CS001	2
11	AMRUTHA K H	1KS19CS011	2
12	ANAGHA A HEBBAR	1KS19CS012	2
13	ANUSHA B	1KS19CS014	2
14	AQSA AQEEL	1KS19CS015	2
15	ASHIKA H N	1KS19CS016	2
16	BHOOMIKA A M	1KS19CS017	2
17	BHOOMIKA K	1KS19CS018	2
18	BHUMIKA M	1KS19CS019	2
19	C N SHREYAS	1KS19CS020	2
20	CHAITANYA SHIVARAJU	1KS19CS021	2
21	CHANDAN B.V	1KS19CS022	2
22	DEEKSHA NAIDU R	1KS19CS023	2
23	DEEPA G	1KS19CS024	2
24	DEEPTHI.N.K	1KS19CS025	2
25	DEVI PRASAD N	1KS19CS026	2
26	DHEEMANTH G	1KS19CS028	2
27	DINESH M	1KS19CS029	2
28	G PRERITHA	1KS19CS030	2
29	GAGAN REDDY S	1KS19CS031	2
30	GAGANDEEP K	1KS19CS032	2
31	GEETHANJALI B K PRASAD	1KS19CS033	2
32	GULSHAN KUMAR S	1KS19CS034	2
33	HARSHITHA J	1KS19CS035	2
34	INDRAJITH H M	1KS19CS036	2
35	JEEVAN T O	1KS19CS037	2
36	K KISHAN	1KS19CS038	2
37	K R VAGEESH	1KS19CS039	2
38	KALYAN CHOWDHARY B	1KS19CS040	2
39	KARTHIK S MORAJKAR	1KS19CS041	2
40	KAVYASHREE.S.L	1KS19CS042	2
41	KEERTHAN GOWDA S	1KS19CS043	2

42	KOTHAPALLI SREEJA	1KS19CS044	2
43	KOTTALA SAIVENKATASUCHITH	1KS19CS045	2
44	KRISHNA K R	1KS19CS046	2
45	KUMAR S	1KS19CS047	2
46	LIKITH G	1KS19CS048	2
47	LISHA C	1KS19CS049	2
48	MAHAK SHREE	1KS19CS050	2
49	MALLIPALLI SPURTHI REDDY	1KS19CS051	2
50	MANASA G L	1KS19CS052	2
51	MOHAMMED NOOR AMAN	1KS19CS053	2
52	MUKESH KUMAR	1KS19CS054	2
53	MYTHREYI U	1KS19CS055	2
54	N ASHOK	1KS19CS056	2
55	N BHAVYA	1KS19CS057	2
56	N P SHASHANKA RAO	1KS19CS058	2
57	BHARATH R	1KS18CS011	2
56	VRATHIKA BILLAVA	1KS19CS116	2
57	AKIF DELVI	1KS20CS400	2
58	PRANAV CHANDRAN	1KS20CS404	2
59	KEERTHI KUMAR	1KS20CS402	2

deefa

D. Nagaraju
Head of the Department
Dept. of Computer Science & Engg.
K.S. Institute of Technology
Bengaluru -560 109



K.S. INSTITUTE OF TECHNOLOGY, BANGALORE - 560109

Course Title : COMPUTER ORGANIZATION Course Code: 18CS34

Faculty Incharge: Dr. Deepa S R and Mrs. Vijayalaxmi Mekali

Question Bank

Module-I

Sl. No	Question
1	Explain the basic operational concepts of computer with neat diagram
2	What is performance measurement? Explain overall SPEC RATING for the computer in program suite. OR Write the basic performance equation. Explain the role of each of the parameters in the equation of the performance of the computer
3	Explain the following i. Byte Addressing ii. Big-Endian assignment iii. Little Endian assignment
4	Show how below expression will be executed in one address, two address and three address processors in an accumulator organization $X = A \times B + C \times D$
5	What is the effective address of the source operand in each of the following instructions when the register R1 and R2 of computer contains the decimal value 1200 and 4600? i. Load 20(R1), R5 ii. Move #3000, R5 iii. Store R5, 30(R1, R2) iv. Add -(R2), R5 v. Subtract (R1)+, R5
6	Interpret the Subroutine Stack Frame with example OR Explain the concept of stack frames, when subroutines are nested.
7	Define the functions following processor registers i. MAR ii. MDR iii. IP iv. IR
8	Compute the contents of 8 bit register namely R1 and R2 containing the value of $-17_{(10)}$ and $+98_{(10)}$ with a initial carry bit as 1 after performing following shift or rotate operations by 2 times i. SHR R1, 2 ii. SAR R1, 2 [Arithmetic shift] iii. ROR R2, 2 iv. RCR R2, 2 [Rotate right with carry]
9	What is the need of processor stack? Explain a commonly used layout for information in the subroutine stack frame
10	With a relevant examples briefly explain about any 2 encoding types of machine instructions
11	With a memory layout starting at address 'i' represent how "ABCD" data is stored in big endian and little endian assignment scheme in a system of word length 16 bits
12	Explain with a neat diagram, the connection between the processor and the computer memory
13	Explain the basic instruction types with example.
14	Define the addressing mode, Explain the various addressing modes with example
15	Write the assembly program that reads the line of characters and display it.
16	What are assembly drivers? Explain various drivers with example.

17	Point out various shift and rotate instructions with example with a neat diagram and example.
18	Explain the connection between the computer and processor memory with a neat diagram and show how to add A+B to form C with the help of the sane diagram
19	What is subroutine? How to pass the parameters to subroutines? Illustrate with example.
20	How to encode assembly instructions into 32 bits words? Explain with examples.
21	Explain the methods to improve the performance of computer
22	Write a use of Rotate and Shift instructions with examples.
Module-II	
1	Illustrate the program that reads one line from the keyboard , stores it in memory buffer , and echos it back to the display in an I/O interfaces
2	What is an interrerrup? Whar are interrupt service routines and what are vectored interrupts? Explain with example OR What is an interrupt? With example illustrate concepts of interrupts
3	Demonstrate the DMA and its implemnetation and show how the data is transferred between the memory and I/O devices using DMA controller.
4	With a neat diagram explain the general 8-bit parallel interface circuit
5	Explain PCI bis data transfer in a compueter system
6	Explain how simultaneous interrupts requests from several I/O devices can be handled by a processor through a single INTR line
7	What is bus arbitration? With a neat diagram explain about distributed arbistration process. OR What is bus arbitration? With a neat diagram explain centralized and distributed arbistration process.
8	With a neat diagram, explain about how data is read in asynchronous bus scheme
9	Explain with a neat diagram, the hardware components needed for connecting a keyboard to a processor.
10	With a neat sequence diagram explain the process of how the output operation is carried between processor and output device connected to host through USB hub
11	What are interrupts? Point out and explain different ways of eanbling and disabling interrupts.
12	What are exceptions? Point out and explain different kind of exeptions.
13	What is interrupt nesting, explain with neat diagram the implementation of interrept priority , using individual interrupt request and acknowledge lines.
14	Explain Universal Serial Bus (USB) tree structure and split bus operation with neat diagram
15	Write a short note on Daisy chain ii) Subroutine iii) Interrupt hardware iv) Exceptions Explain the following methods of handling interrupts from multiple devices Interrupt nesting/ priority structure ii) Daisy chain i) OR i)


16	Explain how DMA (with register) is taking place in a system with necessary diagram OR Write a note on registers in DMA interface OR With a neat diagram explain DMA controller
17	Give the comparison between memory mapped I/O AND I/O mapped I/O
	Draw a neat timing diagram and explain i)
	Multicycle synchronous bus transfer for read operation ii)
18	Asynchronous bus transfer for write operation
	Explain the following with respect to USB i)
19	USB architecture ii) USB addressing iii) USB protocol
20	With a help of timing diagram briefly discuss the main phases of SCSI bus involved in its operations.
21	With a neat diagram explain how to interface printer to the processor
22	What are priority interrupts? Explain any one priority scheme
23	Draw single bus structure. Discuss about memory mapped I/O.
24	Explain PCI bus.
25	List SCSI bus signals with their functionalities
	With supporting diagram, explain the following with respect to interrupts i.
26	Vectored interrupt ii. Interrupt nesting iii. Simultaneous request
27	With a neat timing diagram, illustrate the asynchronous bus data transfer during an input operation. Use handshake scheme.
Module-III	
1	Explain the organization of 1K x 1 memory chip
2	With a neat figure explain the direct mapped cache in the mapping functions.
3	What is memory interleaving? Explain
4	With a neat diagram briefly explain the internal organization of 2M x 8 Dynamic memory chip
5	Illustrate any two cache mapping techniques
6	Calculate the average access time experienced by a processor, if a cache hit rate is 0.88, miss penalty is 0.015 milliseconds and cache access time is 10 microseconds
7	With a neat diagram briefly explain the design of 2M x 32 memory module using 1M x 8 chips.
8	Explain the process of address translation with a neat diagram.
9	With a neat diagram discuss about organization of magnetic disk
10	Calculate the average access time experienced by a processor, if a miss rate is 10%, miss penalty is 17 clock cycles and milliseconds and cache access time is 1 clock cycle.
11	Explain synchronous DRAM with the block diagram.
12	Define ROM, point out and explain various types of ROMs
13	Define cache memory, explain various types of it with a neat diagram
14	Define i. miss rate ii. miss penalty iii. Hit rate iv. Memory bandwidth v. Memory latency

15	With a neat figure explain the internal organization of a 128 X 8 memory chip.
16	With a neat figure explain the SRAM (Static RAM) and DRAM (Asynchronous Dynamic RAM) chip, explain the read and write operations on each of them.
17	Explain the different types of mapping techniques to (between memory to cache memory) with diagram.
18	Calculate the average access time experienced by a processor, if a cache hit rate (n) is 0.5, miss penalty (M) is 100ns and cache access time (c) is 100 nanoseconds.
19	Distiguish between SRAM and DRAM
20	Describe any two mapping functions in cache
21	Describe the principles of magnetic disk.
22	With a neat diagram briefly explain the organization 2M x 32 memory using 512M x 8 memory chips
23	Explain in detail the working of set associative mapped cache with two blocks per set with relevant diagram.
24	Define the following with respect to cache memory i. Valid bit ii. Dirty data iii. Stale data iv. Flush the cache
Module-IV	
1	Perform the addition and subtraction of signed numbers 1. +4 and -6 i. -5 and -2 iii. +7 and -3 iv +2 and +3
2	Explain the 4-bit carry-lookahead adder with neat diagram.
3	Perform bit pair recording for (+13) and (-6)
4	Perform the Booths algorithm for signed numbers (-13) and (+11) OR Multiply +15 and -6 using Booths algorithm
5	Show and perform non restoring division for 3 and 8
6	Design and Explain the working of 16-bit carry-lookahead adder built from 8-bit carry-lookahead adder. Compare the its performance with 16 bit ripple carry adder
7	Calculate the product of $-2_{(10)} \times +4_{(10)}$ using bit pair recording multiplier method. Why bit pair method is better than the Booths algorithm?
8	Perform non restoring division for the given binary numbers where dividend is $1011_{(2)}$ and divisor is $0101_{(2)}$ with all cycles. OR Perform the non-restoring division for $8/3$ by showing all steps
9	Perform the multiplication for signed numbers (-13) and (+9) with Booths algorithm and explain the Booths algorithm
10	Explain with a neat diagram the circuit arrangement for binary division.
11	Write down the steps of Booths multiplication algoorthm. OR Explain Booths algorithm
12	Perform Booths multiplication for (+13) and (-6)
13	Explain generation and propagation functions used in Carry Lookahead Adder
14	Explain Bit Pair Recording / Fast Multiplication with example.
15	Explain the steps of restoring divisiom algorithm. Apply Restoring algorithm on 1000/11
16	Conver the following pairs of decimal numbers to 5-bit signed 2's complement binary numbers and add them. State whether overflow has occurred. i. -5 and 7 ii. -10 and -13 iii. -14 and 11

17	Explain the concept of carry save addition for the multiplication of operation $M \times Q = P$ for 4-bit operands, with diagram and suitable example
18	Perform the operations on 5-bit signed numbers using 2's complement system. Also indicate the whether overflow has occurred i. $(-10) + (-13)$ ii. $(-10) - (-13)$ iii. $(-2) + (+9)$
19	Perform multiplication of $(+13)$ and (-6) using Booths algorithm and bit-pair recording method.
20	Perform restoring division for 8 by 3 by showing all the steps
21	Perform the operations on 5-bit signed numbers using 2's complement system. Also indicate the whether overflow has occurred i. $(-9) + (-7)$ ii. $(+7) - (-8)$
22	Multiply the following signed 2's complement numbers using Booth's algorithm multiplicand $= (010111)_2$ and multiplier $= (110110)_2$
23	Perform division operation on unsigned numbers using restoring method where dividend $= 10101_2$ and divisor $= 00100_2$ with all cycles.
24	Perform Booths multiplication for (-13) and $(+09)$
25	Design the logic circuit to perform addition/subtraction of 'n' bit number X and Y
26	With a figure explain the circuit arrangement for binary division
27	Covert the following pairs of decimal numbers to 5-bit signed 2's complement binary numbers and add them. State whether or not overflow occurs in each case i. 5 and 10 ii. -14 and 11 iii. -5 and 7 iv. -10 and -13
28	With a example explain the Booths algorithm to multiply two signed numbers. Multiply each of the following pairs of signed 2's complement number using Booth algorithm (A is multiplicand and B is multiplier) i. $A = 010111$ and $B = 110110$ ii. $A = 110011$ and $B = 101100$ iii. $A = 110101$ and $B = 011011$ iv. $A = 001111$ and $B = 001111$
Module-V	
1	Illustrate the sequence of operations required to execute the following instructions Add (R3), R1 OR Write and discuss about micro-routine for complete execution of instruction Add (R3), R1 in single bus organization.
2	Explain the three bus organization of data path with a neat diagram or Explain the multiple bus organization of data path with a neat diagram. And also Write control sequence for the instruction Add R4, R5, R6 for multiple bus organization.
3	Compare and contrast the following i. Hard-wired control Microprogrammed control ii.

4	What is pipeline? Explain the 4 stage pipeline with its instruction execution steps and hardware organization
5	Write a control sequence for instruction Add R4, R5, R6 for 3 bus organization
6	Briefly explain the different ways of implementing multiprocessor system with supportive diagram
7	What do you mean by micro-instructions? Design the basic organization of a microprogrammed control unit with diagram
8	Describe the microcontroller with diagram. Also mention parallel and serial I/O port on brief.
9	With a figure explain single bus organization of datapath inside a processor
10	Write a sequence of control steps required for single bus organization for each of the following instructions i. Add the content of memory location NUM to register R1 ii. Add the contents of memory locations whose address is at memory location NUM to register R1

deeba


Head of the Department
Dept. of Computer Science & Engg.
K.S. Institute of Technology
Bengaluru -560 109



K. S. Institute of Technology, Bangalore – 109
Department of Computer Science and Engineering

COURSE END SURVEY

ACADEMIC YEAR 2020-2021 (ODD SEMESTER)

Subject: Computer Organization

Faculty Name: Mrs. Deepa .S.R

Sem and Sec:III A

1. Your ability to identify machine instruction formats and addressing modes								
2. Are you able to build techniques for I/O communication with standard bus interfaces and interrupt service routines.								
3. Your ability to Identify different memories and memory mapping techniques.								
4 Are you able to design different Arithmetic (adder, division etc.) units.								
5. our ability to derive control sequences for hardwired and micro-program control units for both single and multi bus processor								
SL. No	Name of student	USN	Name of the Faculty	Q1	Q2	Q3	Q4	Q5
1	AMOGHA H S	1KS19CS010	Deepa .S.R	Good	Satisfactory	Good	Excellent	Good
2	ABHISHEK B	1KS19CS002	Deepa .S.R	Good	Good	Good	Satisfactory	Satisfactory
3	ABHISHEK YADAV	1KS19CS003	Deepa .S.R	Very good	Very good	Very good	Very good	Very good
4	ADITH KARTHIK RAJU	1KS19CS004	Deepa .S.R	Excellent	Excellent	Good	Good	Good
5	AJAY S KALBURGI	1KS19CS005	Deepa .S.R	Excellent	Excellent	Excellent	Excellent	Excellent
6	AKASH A S	1KS19CS006	Deepa .S.R	Good	Good	Excellent	Excellent	Good
7	AMAN KUSHWAHA	1KS19CS007	Deepa .S.R	Good	Satisfactory	Good	Good	Satisfactory
8	AMIT K B	1KS19CS009	Deepa .S.R	Good	Good	Good	Excellent	Good
9	AAKRITI	1KS19CS001	Deepa .S.R	Good	Very good	Good	Good	Good
10	AMRUTHA K H	1KS19CS011	Deepa .S.R	Good	Good	Good	Good	Good

11	ANAGHA A HEBBAR	1KS19CS012	Deepa .S.R	Excellent	Excellent	Excellent	Excellent	Excellent
12	ANUSHA B	1KS19CS014	Deepa .S.R	Excellent	Excellent	Good	Excellent	Excellent
13	AQSA AQEEL	1KS19CS015	Deepa .S.R	Satisfactory	Satisfactory	Satisfactory	Satisfactory	Satisfactory
14	ASHIKA H N	1KS19CS016	Deepa .S.R	Excellent	Excellent	Excellent	Excellent	Excellent
15	BHOOMIKA A M	1KS19CS017	Deepa .S.R	Good	Good	Good	Excellent	Good
16	BHOOMIKA K	1KS19CS018	Deepa .S.R	Good	Good	Good	Good	Good
17	BHUMIKA M	1KS19CS019	Deepa .S.R	Excellent	Excellent	Excellent	Excellent	Excellent
18	C N SHREYAS	1KS19CS020	Deepa .S.R	Excellent	Excellent	Excellent	Excellent	Excellent
19	CHAITANYA SHIVARAJU	1KS19CS021	Deepa .S.R	Good	Good	Good	Good	Good
20	CHANDAN B.V	1KS19CS022	Deepa .S.R	Good	Satisfactory	Good	Good	Satisfactory
21	DEEKSHA NAIDU R	1KS19CS023	Deepa .S.R	Good	Good	Good	Good	Good
22	DEEPA G	1KS19CS024	Deepa .S.R	Good	Good	Good	Good	Good
23	DEEPTHI.N.K	1KS19CS025	Deepa .S.R	Good	Good	Good	Good	Good
24	DEVI PRASAD N	1KS19CS026	Deepa .S.R	Good	Good	Good	Good	Good
25	DHEEMANTH G	1KS19CS028	Deepa .S.R	Good	Good	Good	Good	Good
26	DINESH M	1KS19CS029	Deepa .S.R	Good	Good	Good	Satisfactory	Good
27	G PRERITHA	1KS19CS030	Deepa .S.R	Satisfactory	Satisfactory	Satisfactory	Satisfactory	Satisfactory
28	GAGAN REDDY S	1KS19CS031	Deepa .S.R	Excellent	Excellent	Excellent	Good	Good
29	GAGANDEEP K	1KS19CS032	Deepa .S.R	Excellent	Good	Good	Excellent	Excellent

30	GEETHANJALI B K PRASAD	1KS19CS033	Deepa .S.R	Good	Excellent	Good	Excellent	Good
31	GULSHAN KUMAR S	1KS19CS034	Deepa .S.R	Good	Good	Good	Good	Good
32	HARSHITHA J	1KS19CS035	Deepa .S.R	Good	Good	Good	Good	Good
33	INDRAJITH H M	1KS19CS036	Deepa .S.R	Good	Good	Good	Good	Good
34	K KISHAN	1KS19CS038	Deepa .S.R	Good	Good	Good	Good	Good
35	K R VAGEESH	1KS19CS039	Deepa .S.R	Very good	Very good	Very good	Very good	Very good
36	KALYAN CHOWDHARY B	1KS19CS040	Deepa .S.R	Excellent	Excellent	Excellent	Excellent	Excellent
37	KARTHIK S MORAJKAR	1KS19CS041	Deepa .S.R	Excellent	Excellent	Excellent	Excellent	Excellent
38	KAVYASHREE.S.L	1KS19CS042	Deepa .S.R	Excellent	Excellent	Excellent	Excellent	Excellent
39	KEERTHAN GOWDA S	1KS19CS043	Deepa .S.R	Excellent	Good	Good	Good	Excellent
40	KOTHAPALLI SREEJA	1KS19CS044	Deepa .S.R	Good	Good	Good	Good	Good
41	KOTTALA SAIVENKATASUC HITH	1KS19CS045	Deepa .S.R	Very good	Very good	Very good	Very good	Very good
42	KRISHNA K R	1KS19CS046	Deepa .S.R	Very good	Very good	Good	Good	Good
43	KUMAR S	1KS19CS047	Deepa .S.R	Very good	Very good	Very good	Very good	Very good
44	LIKITH G	1KS19CS048	Deepa .S.R	Good	Good	Good	Good	Very good
45	LISHA C	1KS19CS049	Deepa .S.R	Good	Good	Good	Very good	Good
46	MAHAK SHREE	1KS19CS050	Deepa .S.R	Good	Good	Good	Excellent	Excellent
47	MALLIPALLI SPURTHI REDDY	1KS19CS051	Deepa .S.R	Very good	Excellent	Excellent	Excellent	Excellent
48	MANASA G L	1KS19CS052	Deepa .S.R	Very good	Good	Very good	Very good	Excellent

49	MOHAMMED NOOR AMAN	1KS19CS053	Deepa .S.R	Very good	Very good	Excellent	Very good	Excellent
50	MUKESH KUMAR	1KS19CS054	Deepa .S.R	Very good	Very good	Very good	Very good	Excellent
51	MYTHREYI U	1KS19CS055	Deepa .S.R	Excellent	Excellent	Very good	Excellent	Excellent
52	N ASHOK	1KS19CS056	Deepa .S.R	Excellent	Excellent	Excellent	Excellent	Excellent
53	N BHAVYA	1KS19CS057	Deepa .S.R	Excellent	Excellent	Excellent	Excellent	Very good
54	N P SHASHANKA RAO	1KS19CS058	Deepa .S.R	Excellent	Excellent	Excellent	Excellent	Excellent
55	BHARATH R	1KS18CS011	Deepa .S.R	Very good	Very good	Very good	Very good	Very good
56	AKIF DELVI	DIP	Deepa .S.R	Very good	Good	Very good	Very good	Very good

Deepa

Course Incharge

Deepa
HOD

Head of the Department
Dept. of Computer Science & Engg.
K.S. Institute of Technology
Bengaluru -560 109



K.S.INSTITUTE OF TECHNOLOGY, BANGALORE
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
INDIRECT ATTAINMENT COURSE ENDS SURVEY

YEAR/SEMESTER	II/III
COURSE TITLE	Computer Organization
COURSE CODE	18CS34
ACADEMIC YEAR	2020-21

	Q1	Q2	Q3	Q4	Q5	
EXCELLENT	17	17	15	21	19	93.56
VERY GOOD	11	9	9	10	8	
GOOD	26	25	30	21	24	
SATISFACTORY	2	5	2	4	5	
STUDENTS RESPONSE (GOOD & ABOVE)	96.42	91.07	96.42	92.85	91.07	

deefa

STAFF SIGNATURE

Dr. Venkatesh

HOD

Head of the Department
Dept. of Computer Science & Engg.
K.S. Institute of Technology
Bengaluru - 560 109

CBCS SCHEME

15CS34

USN

--	--	--	--	--	--	--	--	--	--

Third Semester B.E. Degree Examination, June/July 2018

Computer Organization

Time: 3 hrs.

Max. Marks: 80

Note: Answer any FIVE full questions, choosing one full question from each module.

Module-1

- 1 a. Define Addressing Mode. Give the details of different addressing modes. (08 Marks)
b. Describe the basic operational concepts between the processor and memory. (08 Marks)

OR

- 2 a. What is Subroutine? How to pass parameters to subroutines? Illustrate with an example. (08 Marks)
b. How to encode assembly instructions into 32-bit words? Explain with examples. (08 Marks)

Module-2

- 3 a. Define Bus Arbitration. With diagrams, explain the centralized bus arbitration mechanism. (08 Marks)
b. With the help of timing diagram, briefly discuss the main phases of SCSI bus involved in its operation. (08 Marks)

OR

- 4 a. With neat diagrams, explain how to interface printer to the processor. (08 Marks)
b. Explain the following methods of handling interrupts from multiple devices.
i) Interrupt nesting/priority structure ii) Daisy chain method. (08 Marks)

Module-3

- 5 a. Describe how to translate virtual address into physical address with diagram. (08 Marks)
b. Draw and explain the internal organisation of $2M \times 8$ asynchronous DRAM chip. (08 Marks)

OR

- 6 a. Describe any two mapping functions in cache. (08 Marks)
b. Describe the principles of magnetic disk. (08 Marks)

Module-4

- 7 a. Perform the operations on 5-bit signed numbers using 2's complement system. Also indicate whether overflow has occurred. (06 Marks)
i) $(-10) + (-13)$ ii) $(-10) - (-13)$ iii) $(-2) + (-9)$.
b. Perform the multiplication of 13 and -6 using Booth algorithm and Bit-pair recoding method. (10 Marks)

OR

- 8 a. Perform the restoring division for $8 \div 3$ by showing all the steps. (06 Marks)
b. Explain the logic diagram of 4-bit carry look ahead adder and its operations. (10 Marks)

Module-5

- 9 a. Draw and explain multiple bus organization along with its advantages. (10 Marks)
b. Write down the control sequence for the instruction Add (R_1), R_1 for single bus organization. (06 Marks)

OR

- 10 a. With block diagram, explain the general requirements and working of digital camera. (10 Marks)
b. Write the control sequence for an unconditional branch instruction. (06 Marks)

Third Semester B.E. Degree Examination, Dec.2017/Jan.2018

Computer Organization

Time: 3 hrs.

Max. Marks: 80

Note: Answer any FIVE full questions, choosing one full question from each module.

Module-1

- 1 a. List the steps needed to execute the machine instruction Add LOCA, RO in terms of data transfers between the processor and the memory along with some simple control commands. Assume that the instruction itself is stored in the memory at location INSTR and that the address is initially in register PC. The first two steps might be expressed as:

- Transfer the contents of Register PC to register MAR.
- Issue a Read command to the memory and then wait until it has transferred the requested word into register MDR.

Remember to include the steps needed to update the contents of PC from INSTR to INSTR+1 so that the next instruction can be fetched. (08 Marks)

- b. What is performance measurement? Explain the overall SPEC rating for the computer program suite. (08 Marks)

OR

- 2 a. With relevant figure define the little Endian and big Endian assignments. (04 Marks)
- b. Consider a computer that has a byte addressable memory organized in 32 bit words according to the big Endian scheme. A program reads ASCII characters entered at the keyboard and store them in successive byte location starting at location 1000. Show the contents of the two memory words at locations 1000 and 1004 after the name "Johnson" has been entered. (ASCII codes J = 4 AH, o = 6 FH, n = 6 EH, S = 73 H) (04 Marks)
- c. Write about shift and rotate instruction with neat diagram and example of each. (08 Marks)

Module-2

- 3 a. With supporting diagram, explain the following with respect to interrupts:
- Vectored interrupts
 - Interrupt Nesting
 - Simultaneous requests. (06 Marks)
- b. Three devices A, B and C are connected to the bus of a computer. I/O transfers for all the devices use interrupt control. Interrupt nesting for devices A and B is not allowed, interrupt requests from C may be accepted while either A or B is being serviced. Suggest different ways in which this can be accomplished in each of the following cases:
- The computer has one interrupt request line.
 - Two interrupt request line, INTR1 and INTR2 are available with INTR1 having high priority. Specify when and how interrupts are enabled and disabled in each case. (06 Marks)

- c. Illustrate the tree structure of USB with diagram. (04 Marks)

OR

- 4 a. With a neat diagram, explain the centralized arbitration and distributed bus arbitration scheme. (08 Marks)
- b. With neat timing diagram illustrate the asynchronous bus data transfer during an input operation. Use handshake scheme. (08 Marks)

Module-3

- Draw a diagram and explain the working of 16 Megabit DRAM chip configured as $2M \times 32$ memory. (08 Ma)
- Describe organization of an $2M \times 32$ memory using $512K \times 8$ memory chips. (08 Ma)

OR

- Discuss in detail the working of set associative mapped cache with two blocks per set. Draw a relevant diagram. (08 Ma)
- Define the following with respect to cache memory: (i) Valid bit, (ii) Dirty bit, (iii) Stale data, (iv) Flush the cache. (04 Ma)
- A block-set associative cache consists of a total of 64 blocks divided into 4-blocks sets. Main memory contains 4096 blocks, each consisting of 128 words.
- How many bits are there in a main memory address?
 - How many bits are there in each of the TAG, SET and WORD fields? (04 Ma)

Module-4

- Convert the following pairs of decimal numbers to 5-bit signed 2's complement binary numbers and add them. State whether or not overflow occurs in each case.
- 5 and 10
 - 14 and 11
 - 5 and 7
 - 10 and -13 (04 Ma)
- Design the 16 bit carry look ahead adder using 4-bit adder. Also unite the expression for C_{i+1} . (08 Ma)
- Draw the two n-bit number x and y to perform addition/subtraction. (04 Ma)

OR

- With an example explain the Booths algorithm to multiply two signed operands. (08 Ma)
- Multiply each of the following pairs of signed 2's complement number using the Booth algorithm. (A = multiplicand and B = multiplier).
- A = 010111 and B = 110110
 - A = 110011 and B = 101100
 - A = 110101 and B = 011011
 - A = 001111 and B = 001111 (08 Ma)

Module-5

- Discuss with neat diagram, the single bus organization of the data path inside a processor. (08 Ma)
- Write the sequence of control steps required for single bus structure for each of the following instructions.
- Add the contents of memory location NUM to register R1.
 - Add the contents of memory location whose address is at memory location NUM to register R1. (08 Ma)

OR

- Discuss the microwave oven with neat block diagram. (08 Ma)
- Discuss the digital camera with neat block diagram. (08 Ma)

SN

--	--	--	--	--	--	--	--	--	--

15CS34

Third Semester B.E. Degree Examination, Dec.2016/Jan.2017
Computer Organization

Time: 3 hrs.

Max. Marks: 80

Note: Answer any FIVE full questions, choosing one full question from each module.

Module-1

- 1 a. With a neat diagram, explain basic operational concept of computer. (06 Marks)
- b. What is performance measurement? Explain overall SPEC rating for computer. (04 Marks)
- c. Draw single bus structure, discuss about memory mapped I/O. (06 Marks)

OR

- 2 a. What is an addressing mode? Explain any three addressing modes with example. (10 Marks)
- b. Explain BIG-ENDIAN and LITTLE-ENDIAN methods of byte addressing with proper example. (06 Marks)

Module-2

- 3 a. What is an Interrupt? With example illustrate concept of interrupt. (06 Marks)
- b. Define Exception. Explain 2 kinds of exception. (04 Marks)
- c. With a neat diagram explain DMA controller. (06 Marks)

OR

- 4 a. Explain PCI bus. (05 Marks)
- b. List SCSI bus signal with their functionalities. (05 Marks)
- c. Explain the tree structure of USB with split bus operation. (06 Marks)

Module-3

- 5 a. Briefly explain any two mapping function used in cache memory. (08 Marks)
- b. With a neat diagram explain the internal organization of memory chip (2M×8 and dynamic memory chip). (08 Marks)

OR

- 6 a. Explain the following:
 - i) Hit Rate and Miss penalty
 - ii) Virtual memory organization.
 (08 Marks)
- b. With diagram explain how virtual memory translation take place. (08 Marks)

Module-4

- 7 a. Draw 4-bit carry-look ahead adder and explain. (06 Marks)
- b. Perform multiplication for -13 and +09 using Booth's Algorithm. (06 Marks)
- c. Design a logic circuit to perform addition/subtraction of 'n' bit number X and Y. (04 Marks)

OR

- 8 a. Explain IEEE standard for floating point number. (06 Marks)
- b. With figure explain circuit arrangement for binary division. (10 Marks)

Module-5

- 9 a. With a figure explain single bus organization of datapath inside a processor. (08 Marks)
- b. What are the actions required to Execute a complete instruction Add (R3), R₁. (02 Marks)
- c. Give the control sequence for execution of instruction ADD (R3), R₁. (06 Marks)

OR

- 10 a. Briefly explain the block diagram of camera. (08 Marks)
- b. Explain multiprocessors. Justify how time is reduced. (08 Marks)

Code No: 134AK**JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD****B.Tech II Year II Semester Examinations, April - 2018****COMPUTER ORGANIZATION****(Common to CSE, IT)****Time: 3 Hours****Max. Marks: 75****Note:** This question paper contains two parts A and B.

Part A is compulsory which carries 25 marks. Answer all questions in Part A.

Part B consists of 5 Units. Answer any one full question from each unit. Each question carries 10 marks and may have a, b, c as sub questions.

PART- A**(25 Marks)**

- 1.a) Explain RTL and its control function. [2]
- b) Compare horizontal and vertical organization. [3]
- c) Differentiate jump and loop instructions. [2]
- d) Briefly explain special processor activities. [3]
- e) What is an assembler? [2]
- f) Explain the machine code for: LES DI,[0600H] and NEG 50[BP]. [3]
- g) Explain overflow and underflow. [2]
- h) Differentiate isolated I/O and memory mapped I/O. [3]
- i) Explain the cache incoherence. [2]
- j) Explain the locality of reference. [3]

PART-B**(50 Marks)**

- 2.a) List and explain different performance measures used to represent a computer system performance.
 - b) Elucidate the functioning of a Micro program sequencer. [5+5]
- OR**
- 3.a) Elucidate common bus system.
 - b) Formulate a mapping procedure that provides eight consecutive micro instructions for each routine. The operation code has 7 bits and control memory has 4096 words. [5+5]
- 4.a) Explain the register organization in 8086.
 - b) Elucidate machine language instruction formats. [5+5]
- OR**
- 5.a) Explain the pin configuration details of 8086.
 - b) Explain the assembler directives with examples. [5+5]
- 6.a) Explain the steps involved in writing a program using an assembler.
 - b) Write a program to find out the number of positive numbers and negative numbers from a given series of signed numbers. [5+5]
- OR**
- 7.a) Add the contents of the memory location 4000H:0600H to contents of 5000H:0700H and store the result in 8000H:0900H
 - b) Write a program for addition of two numbers. [5+5]

- 8.a) Draw a flow chart for Floating point Add/subtract operations.
b) Illustrate asynchronous communication interface in detail.

[5+5]

OR

- 9.a) Explain in detail with neat sketch Booth Multiplication Algorithm.
b) Explain different types of modes of control.

[5+5]

- 10.a) Explain arithmetic pipeline with example.
b) Elucidate Inter processor communication.

[5+5]

OR

- 11.a) Elucidate array processor in detail.
b) Explain various Interconnection Structures.

[5+5]

---ooOoo---

Code No: 134AK**JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD****B.Tech II Year II Semester Examinations, May - 2019****COMPUTER ORGANIZATION****(Common to CSE, IT)****Time: 3 Hours****Max. Marks: 75****Note:** This question paper contains two parts A and B.

Part A is compulsory which carries 25 marks. Answer all questions in Part A.

Part B consists of 5 Units. Answer any one full question from each unit. Each question carries 10 marks and may have a, b as sub questions.

PART – A**(25 Marks)**

- 1.a) What is the purpose of BUN instruction? [2]
- b) Define computer organization, computer architecture. [3]
- c) Contrast 8086 minimum mode with maximum mode. [2]
- d) How an address is latched in 8086? [3]
- e) What is the need of a linker? [2]
- f) What is the difference between a macro and a procedure? [3]
- g) What is the disadvantage of strobe method? [2]
- h) Provide the hardware for signed-2's complement addition and subtraction. [3]
- i) Define miss penalty for cache memory. [2]
- j) Draw the system bus structure for multiprocessors. [3]

PART – B**(50 Marks)**

2. List the registers for the basic computer and give their functionality in program execution. [10]
- OR**
3. Describe the micro programmed control organization and compare its advantages over hardwired control. [10]
 4. Evaluate the following arithmetic statement using zero, one, two and three address instructions. Use the conventional symbols and instructions.
 $X = (A+B) * (C+D).$ [10]
- OR**
5. Does 8086 support instruction pipelining? Justify your answer with relevant example instructions. [10]
 6. Develop an assembly language program to find out numbers odd and even numbers in a given series of 16-bit hexa decimal numbers. [10]
- OR**
7. Elaborate on the techniques used to pass parameters to procedures in assembly language program. [10]

8. Show the step-by-step multiplication process using Booth algorithm when the following binary numbers are multiplied. $(+33) \times (-12)$. [10]

OR

9. Design a circuit for a 4×4 First In First Out Buffer and explain its functionality. [10]
10. A digital computer has a memory unit of $64K \times 16$ and a cache memory of 1K words. The cache uses direct mapping with a block size of 4 words.
- (a) How many bits are there in the tag, index, block and word fields of the address format?
- (b) How many bits are there in each word of cache and how are they divided into function? Include a valid bit. [10]

OR

11. Does pipelining get affected by data dependencies among the instruction? Justify your answer with lucid examples. [10]

---ooOoo---



K. S. INSTITUTE OF TECHNOLOGY

#14, Raghuvanahalli, Kanakapura Main Road, Bengaluru-5600109

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Course Name: Computer Organization

Course code: 18CS34

Semester : 3rd Year: 2nd Section: A

Staff: Deepa S.R

Challenging Questions with Answers (Module 1 – Module 5)

1. What are the Different types of Interrupts in a Microprocessor System, Explain?

In the normal execution of a program there are three types of interrupts that can cause a break:

External Interrupts: These types of interrupts generally come from external input / output devices which are connected externally to the processor. They are generally independent and oblivious of any programming that is currently running on the processor.

Internal Interrupts: They are also known as traps and their causes could be due to some illegal operation or the erroneous use of data. Instead of being triggered by an external event they are usually triggered due to any exception that has been caused by the program itself. Some of the causes of these types of interrupts can be due to attempting a division by zero or an invalid opcode etc.

Software interrupts: These types of interrupts can occur only during the execution of an instruction. They can be used by a programmer to cause interrupts if need be. The primary purpose of such interrupts is to switch from user mode to supervisor mode.

2. What is Virtual Memory in Computer?

Virtual memory is that when the available RAM memory is not sufficient for the system to run the current applications it will take some memory from hard disk. This memory is termed as Virtual memory.

3. State some of the common rules of assembly language?

Some of the common rules of assembly level language are as follows:

- In assembly language the label field can be either empty or may specify a symbolic address.
- Instruction fields can specify pseudo or machine instructions.
- Comment fields can be left empty or can be commented with.
- Up to 4 characters are only allowed in the case of symbolic addresses.
- The symbolic addresses field are terminated by a comma whereas the comment field begins with a forward slash.

4. Explain types of memory in computer architecture?

Computer have different type of memory like primary memory , Auxiliary memory , buffer memory , Cache memory , virtual memory ,the work of all memory heterogeneously primary memory is directly communicate with the CPU . Auxiliary memory are used for storing the data for long time . Buffer memory are mainly used for storing the intermediate data between the travel . cache memory are usedfor storing the those data that currently required at process time for increase the speed of the data .virtual memory are put in between the two memory for increase the speed of data or instruction it means it put between HDD and RAM .

5. Consider a case study when the CPU is busy but you want to stop and do some other task. How do you do it?

- rise a non maskable interrupt.
- Then give jump instruction to required subroutine.

6. What do you understand by virtualization? mention the pros and cons?

In a way virtualization appears similar to emulation but actually it shares hardware resources from the host OS.

- This method is slower as compared to partition method but is faster than emulation.
- Virtualization had also vast support considering it can also provide with 3d support.
- With the help of virtualization it enable users to create virtual clusters.
- But virtualization systems require a lot of memory in form of ram.
- For virtualization it is mandatory that the virtualized platform has the same architecture as the host pc otherwise due to incompatibilities it is not possible.

7. What are the major difficulties of pipeline conflicts in processors supporting pipe lining?

The following are the main reasons for pipe line conflicts in the processor:

- When the same resource is accessed at the same time by two different segments it results in resource conflicts. The only way to resolve this problem is to use separate data memories.
- In case an instruction's execution depends on the result of a previous instruction and that result is unavailable it leads to data dependency conflicts.
- Instructions that change the count of the PC can cause a lot of problems. This is prevalent particularly in the case of Branch instructions. A method to resolve this issue is known as delayed load where certain instruction are made to execute in a delayed manner to avoid conflicts.

deefa

Nurcatapu



K S INSTITUTE OF TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

YEAR / SEMESTER	II / III
COURSE TITLE	COMPUTER ORGANIZATION
COURSE CODE	18CS34
ACADEMIC YEAR	2020-2021

Sl. No	USN	Name of the Student	18CS34														SEE
			IA1		A1		IA2			A2			IA3		A3		
			CO1	CO2	CO1	CO2	CO2	CO3	CO4	CO2	CO3	CO4	CO4	CO5	CO4	CO5	
			18	12	6	4	6	18	6	2	6	2	12	18	4	6	
1	IKS19CS001	AAKRITI	18	7	6	4	6	18	5	2	6	2	12	18	4	6	38
2	IKS19CS002	ABHISHEK B	18	12	6	4	6	18	6	2	6	2	6	11	4	6	22
3	IKS19CS003	ABHISHEK YADAV	18	12	6	4	6	18	6	2	6	2	8	3	4	6	21
4	IKS19CS004	ADITH KARTHIK RAJU	18	12	6	4	6	NA	6	2	6	2	7	17	4	6	21
5	IKS19CS005	AJAY S KALBURGI	18	12	6	4	6	18	6	2	6	2	11	11	4	6	33
6	IKS19CS006	AKASH A S	18	12	6	4	2	18	6	2	6	2	12	18	4	6	30
7	IKS19CS007	AMAN KUSHWAHA	18	12	6	4	6	18	6	2	6	2	7	11	4	6	23
8	IKS19CS009	AMIT K B	18	12	6	4	6	18	6	2	6	2	12	12	4	6	33
9	IKS19CS010	AMOGHA H S	18	12	6	4	6	18	3	2	6	2	11	7	4	6	21
10	IKS19CS011	AMRUTHA K H	17	12	6	4	6	18	6	2	6	2	12	18	4	6	45
11	IKS19CS012	ANAGHA A HEBBAR	18	11	6	4	6	18	6	2	6	2	12	18	4	6	22
12	IKS19CS014	ANUSHA B	18	10	6	4	6	18	6	2	6	2	8	17	4	6	21
13	IKS19CS015	AQSA AQEEL	16	9	6	4	6	18	6	2	6	2	11	17	4	6	37
14	IKS19CS016	ASHIKA H N	18	12	6	4	6	18	6	2	6	2	12	17	4	6	26
15	IKS19CS017	BHOOMIKA A M	18	11	6	4	6	18	6	2	6	2	11	18	4	6	32
16	IKS19CS018	BHOOMIKA K	18	12	6	4	6	18	6	2	6	2	11	18	4	6	34
17	IKS19CS019	BHUMIKA M	18	12	6	4	6	18	6	2	6	2	12	12	4	6	35
18	IKS19CS020	C N SHREYAS	18	12	6	4	6	18	6	2	6	2	12	12	4	6	29
19	IKS19CS021	CHAITANYA SHIVARAJU	18	12	6	4	6	18	6	2	6	2	12	18	4	6	38
20	IKS19CS022	CHANDAN B V	18	12	6	4	6	18	6	2	6	2	10	NA	4	6	22
21	IKS19CS023	DEEKSHA NAIDU R	18	11	6	4	6	18	6	2	6	2	11	18	4	6	11
22	IKS19CS024	DEEPA G	18	12	6	4	6	18	6	2	6	2	10	16	4	6	29
23	IKS19CS025	DEEPTHI.N.K	18	12	6	4	6	18	6	2	6	2	11	17	4	6	30
24	IKS19CS026	DEVI PRASAD N	16	12	6	4	6	18	NA	2	6	2	6	12	4	6	29
25	IKS19CS028	DHEEMANTH G	18	12	6	4	6	18	3	2	6	2	9	NA	4	6	14
26	IKS19CS029	DINESH M	18	12	6	4	6	17	NA	2	6	2	10	12	4	6	24
27	IKS19CS030	G PRERITHA	18	12	6	4	4	18	6	2	6	2	7	18	4	6	24
28	IKS19CS031	GAGAN REDDY S	18	12	6	4	4	18	6	2	6	2	9	6	4	6	11

29	IKS19CS032	GAGANDEEP K	17	12	6	4	6	18	6	2	6	2	NA	9	4	6	21
30	IKS19CS033	GEETHANJALI B K PRASAD	18	12	6	4	6	18	6	2	6	2	8	18	4	6	27
31	IKS19CS034	GULSHAN KUMAR S	18	12	6	4	6	17	6	2	6	2	11	18	4	6	29
32	IKS19CS035	HARSHITHA J	18	12	6	4	6	18	6	2	6	2	12	NA	4	6	34
33	IKS19CS036	INDRAJITH H M	18	12	6	4	6	18	6	2	6	2	12	16	4	6	29
34	IKS19CS038	K KISHAN	18	12	6	4	6	18	6	2	6	2	NA	18	4	6	26
35	IKS19CS039	K R VAGEESH	18	12	6	4	6	18	6	2	6	2	12	18	4	6	31
36	IKS19CS040	KALYAN CHOWDHARY B	16	11	6	4	6	18	6	2	6	2	5	6	4	6	21
37	IKS19CS041	KARTHIK S MORAJKAR	18	12	6	4	6	18	6	2	6	2	12	18	4	6	26
38	IKS19CS042	KAVYASHREE.S.L	18	12	6	4	6	18	6	2	6	2	12	18	4	6	34
39	IKS19CS043	KEERTHAN GOWDA S	18	12	6	4	6	18	5	2	6	2	12	18	4	6	38
40	IKS19CS044	KOTHAPALLI SREEJA	17	4	6	4	6	18	6	2	6	2	9	15	4	6	24
41	IKS19CS045	KOTTALA SAIVENKATASUCHITH	17	8	6	4	6	18	6	2	6	2	4	6	4	6	18
42	IKS19CS046	KRISHNA K R	17	8	6	4	6	18	6	2	6	2	2	3	4	6	21
43	IKS19CS047	KUMAR S	18	11	6	4	6	18	6	2	6	2	NA	NA	4	6	4
44	IKS19CS048	LIKITH G	14	2	6	4	6	18	6	2	6	2	8	6	4	6	21
45	IKS19CS049	LISHA C	18	12	6	4	6	17	6	2	6	2	11	12	4	6	32
46	IKS19CS050	MAHAK SHREE	18	12	6	4	6	18	6	2	6	2	12	18	4	6	42
47	IKS19CS051	MALLIPALLI SPURTHI REDDY	18	12	6	4	6	18	6	2	6	2	11	18	4	6	26
48	IKS19CS052	MANASA G L	17	12	6	4	6	18	6	2	6	2	8	15	4	6	29
49	IKS19CS053	MOHAMMED NOOR AMAN	18	12	6	4	6	18	6	2	6	2	4	1	4	6	21
50	IKS19CS054	MUKESH KUMAR	18	12	6	4	6	18	6	2	6	2	3	11	4	6	21
51	IKS19CS055	MYTHREYI U	18	12	6	4	6	18	6	2	6	2	12	18	4	6	30
52	IKS19CS056	N ASHOK	18	12	6	4	6	18	6	2	6	2	11	15	4	6	31
53	IKS19CS057	N BHAVYA	18	8	6	4	6	18	6	2	6	2	9	13	4	6	25
54	IKS19CS058	N P SHASHANKA RAO	17	8	6	4	6	18	6	2	6	2	9	8	4	6	21
55	IKS19CS059	NAVEEN K M	17	10	6	4	6	18	5	2	6	2	8	9	4	6	32
56	IKS19CS060	NEELAMMA SALI	18	12	6	4	6	18	5	2	6	2	10	9	4	6	23
57	IKS19CS061	NETHRA R	18	9	6	4	6	18	6	2	6	2	12	12	4	6	34
58	IKS19CS062	NIKHIL CHOWDARY V R	17	10	6	4	6	17	6	2	6	2	5	11	4	6	33
59	IKS19CS063	NITHYA S N	18	10	6	4	6	17	5	2	6	2	11	15	4	6	35
60	IKS19CS064	PAVAN B INDRESH	16	11	6	4	6	17	6	2	6	2	9	13	4	6	9
61	IKS19CS065	PAVAN L	18	11	6	4	6	17	6	2	6	2	4	15	4	6	15
62	IKS19CS066	PAVITHRA S P	17	12	6	4	6	16	5	2	6	2	6	14	4	6	27
63	IKS19CS067	POOJA J	18	11	6	4	6	17	6	2	6	2	10	17	4	6	36
64	IKS19CS068	POTHURU SAI PRIYANKA	17	10	6	4	6	17	6	2	6	2	3	16	4	6	21
65	IKS19CS069	PRAJWAL G	16	3	6	4	4	15		2	6	2	1	NA	4	6	21
66	IKS19CS070	PRAJWAL KULKARNI	18	11	6	4	6	18	3	2	6	2	12	14	4	6	24
67	IKS19CS071	PREETHI K P	17	12	6	4	6	18	4	2	6	2	12	15	4	6	32
68	IKS19CS072	PRIYA E	18	12	6	4	6	18	5	2	6	2	12	12	4	6	44
69	IKS19CS073	PRIYANKA R	17	12	6	4	6	18	5	2	6	2	10	13	4	6	24
70	IKS19CS074	PUVAN KUMAR V	18	11	6	4	6	16	5	2	6	2	10	11	4	6	26
71	IKS19CS075	R V YASHVANTH	18	8	6	4	6	16	5	2	6	2	8	5	4	6	21
72	IKS19CS076	RAJ KUMAR	17	11	6	4	6	13	4	2	6	2	11	7	4	6	10

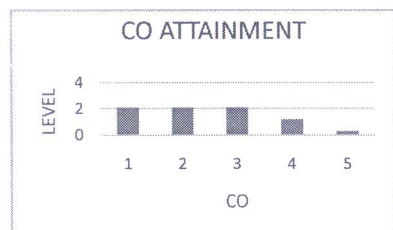
73	1KS19CS077	RAKESH M J	18	12	6	4	6	16	5	2	6	2	9	6	4	6	33
74	1KS19CS078	RAKSHA T M	17	12	6	4	6	17	5	2	6	2	3	13	4	6	21
75	1KS19CS079	RANJITHA D V	17	11	6	4	6	18	5	2	6	2	10	15	4	6	21
76	1KS19CS080	SAHANA S	16	12	6	4	6	16	5	2	6	2	5	17	4	6	29
77	1KS19CS081	SAI SHIRISHA S B	17	11	6	4	6	16	5	2	6	2	3	6	4	6	26
78	1KS19CS082	SANGHARSH KUMAR RAI	NA	NA	6	4	5	14	5	2	6	2	3	6	4	6	12
79	1KS19CS083	SANJEEV MYSORE	15	7	6	4	6	14	5	2	6	2	7	2	4	6	21
80	1KS19CS084	SANKALP KESTI	18	10	6	4	6	13	5	2	6	2	11	5	4	6	26
81	1KS19CS085	SATISH V	18	11	6	4	6	18	5	2	6	2	7	17	4	6	28
82	1KS19CS086	SHANVI B P	17	12	6	4	6	18	5	2	6	2	10	6	4	6	22
83	1KS19CS087	SHIVANI G K	17	12	6	4	6	18	5	2	6	2	6	15	4	6	33
84	1KS19CS088	SHREESHA S	17	12	6	4	5	17	5	2	6	2	9	18	4	6	50
85	1KS19CS089	SHREYA R KIRAN	17	12	6	4	6	18	5	2	6	2	6	8	4	6	36
86	1KS19CS090	SHRI HARSHA S	17	11	6	4	NA	NA	NA	2	6	2	4	14	4	6	12
87	1KS19CS091	SINDHU S	17	12	6	4	6	17	6	2	6	2	5	10	4	6	21
88	1KS19CS092	SMRITHI SHEKAR	16	12	6	4	5	16	6	2	6	2	4	3	4	6	9
89	1KS19CS093	SRUJAN K	17	12	6	4	6	17	5	2	6	2	4	3	4	6	21
90	1KS19CS094	SRUSTI S GOWDA	16	12	6	4	6	18	5	2	6	2	11	16	4	6	41
91	1KS19CS096	SUHAS S	16	8	6	4	4	17	5	2	6	2	2	6	4	6	15
92	1KS19CS097	SUSHMITHA K	13	8	6	4	6	18	3	2	6	2	5	9	4	6	30
93	1KS19CS098	SWETHA M KULKARNI	18	12	6	4	6	16	5	2	6	2	8	8	4	6	30
94	1KS19CS099	SYED ZAINUL ABIDIN	18	12	6	4	6	14	5	2	6	2	2	1	4	6	1
95	1KS19CS100	TALLURU MAURYA	16	7	6	4	2	17	5	2	6	2	7	NA	4	6	3
96	1KS19CS101	TANUSHREE R	13	7	6	4	6	18	5	2	6	2	6	1	4	6	21
97	1KS19CS102	TEJAS N	17	12	6	4	6	17	5	2	6	2	6	13	4	6	7
98	1KS19CS103	TEJAS P	18	8	6	4	6	17	5	2	6	2	NA	13	4	6	21
99	1KS19CS104	TEJASWINI NAYAKA S	13	8	6	4	6	17	6	2	6	2	11	13	4	6	38
100	1KS19CS105	THANUSHREE S	17	12	6	4	6	17	5	2	6	2	2	1	4	6	21
101	1KS19CS106	THIRUMAL R	15	10	6	4	6	18	5	2	6	2	12	14	4	6	24
102	1KS19CS107	THRIVENI U	17	11	6	4	6	17	5	2	6	2	10	8	4	6	23
103	1KS19CS108	UDHAY KUMAR G	17	9	6	4	NA	NA	NA	2	6	2	7	9	4	6	13
104	1KS19CS109	VAISHNAVI G	16	6	6	4	6	18	5	2	6	2	2	4	4	6	21
105	1KS19CS110	VARSHA BAI R	18	12	6	4	6	16	6	2	6	2	12	18	4	6	42
106	1KS19CS111	VARUN KAMBALI	18	10	6	4	6	17	5	2	6	2	4	2	4	6	7
107	1KS19CS112	VISHAL GUPTA	15	8	6	4	6	18	5	2	6	2	11	11	4	6	27
108	1KS19CS113	YASHWANTH S R	18	11	6	4	6	16	5	2	6	2	10	17	4	6	42
109	1KS19CS115	AKSHAY R	18	11	6	4	NA	NA	NA	2	6	2	12	18	4	6	36
110	1KS19CS116	VRATHIKA BILLAVA	17	12	6	4	6	7	6	2	6	2	10	10	4	6	33
111	1KS20CS400	AKIF DEL VI	18	12	6	4	6	18	6	2	6	2	8	12	4	6	22
112	1KS20CS401	ANUSHA A R	12	11	6	4	6	6	6	2	6	2	NA	NA	4	6	29
113	1KS20CS402	KEERTHI KUMAR V	11	10	6	4	6	12	6	2	6	2	4	9	4	6	14
114	1KS20CS403	NITHIN S	12	11	6	4	6	8	6	2	6	2	10	6	4	6	23
115	1KS20CS404	PRANAV CHANDRAN P	14	4	6	4	6	12	0	2	6	2	NA	12	4	6	21
60% of Maximum marks (X)			11	07	04	02	04	11	04	01	04	01	07	11	02	04	36

No. of students above X	114	105	115	115	110	108	104	115	115	115	71	71	115	115	15
Total number of students (Y)	114	114	115	115	112	111	109	115	115	115	109	108	115	115	115
CO Percentage	100.00	92.11	100.00	100.00	98.21	97.30	95.41	100.00	100.00	100.00	65.14	65.74	100.00	100.00	13.04
	CO1	CO2	CO1	CO2	CO2	CO3	CO4	CO2	CO3	CO4	CO4	CO5	CO4	CO5	SEE

CO	CIE	SEE	DIRECT ATTAINMENT	Level	COURSE END SURVEY	LEVEL	ATTAINMENT
CO1	100.00	13.04	56.52	2.00	94.40	3.00	2.1
CO2	97.58	13.04	55.31	2.00	94.40	3.00	2.1
CO3	98.65	13.04	55.85	2.00	94.40	3.00	2.1
CO4	90.14	13.04	51.59	1.00	94.40	3.00	1.2
CO5	82.87	13.04	47.96	0.00	94.40	3.00	0.3
AVERAGE							1.56

	IA1	A1	IA2	A2	IA3	A3	AVG
CO1	100.00	100					100.00
CO2	92.11	100	98.21	100.00			97.58
CO3			97.30	100			98.65
CO4			95.41	100.00	65.14	100.00	90.14
CO5					65.74	100	82.87

CO Attainment Level	Significance	
Level 3	60% and above students should have scored $\geq 60\%$ of Total marks	For Direct attainment, 50% of CIE and 50% of SEE marks are considered.
Level 2	55% to 59% of students should have scored $\geq 60\%$ of Total marks	For indirect attainment, Course end survey is considered.
Level 1	50% to 54% of students should have scored $\geq 60\%$ of Total marks	CO attainment is 90% of direct attainment + 10% of Indirect attainment.
		PO attainment = CO-PO mapping strength/3 * CO attainment.



Co-Po Mapping Table														
CO'S	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PS01	PS02
CO1	3	2	2	-	-	-	-	-	2	-	-	-	3	2
CO2	3	2	2	-	-	-	-	-	2	-	-	-	3	2
CO3	3	2	3	-	-	-	-	-	-	-	-	-	3	2
CO4	3	2	3	-	-	-	-	-	-	-	-	-	3	2
CO5	3	2	2	-	-	-	-	-	-	-	-	-	3	2
AVG	3.00	2.00	2.40	-	-	-	-	-	2.00	-	-	-	3	2

PO ATTAINMENT TABLE																
CO'S	CO Attainment in %	CO RESULT	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PS01	PS02
CO1	2.10	Y	2.10	1.40	1.40	-	-	-	-	-	1.4	-	-	-	2.10	1.40
CO2	2.10	Y	2.10	1.40	1.40	-	-	-	-	-	1.40	-	-	-	2.10	1.40
CO3	2.10	Y	2.10	1.40	2.10	-	-	-	-	-	-	-	-	-	2.10	1.40
CO4	1.20	N	1.20	0.80	1.20	-	-	-	-	-	-	-	-	-	1.20	0.80
CO5	0.30	N	0.30	0.20	0.20	-	-	-	-	-	-	-	-	-	0.30	0.20
Average			1.56	1.04	1.26	-	-	-	-	-	1.40	-	-	-	1.56	1.04

deba
Course Incharge

Shravan
HOD
Head of the Department
Dept. of Computer Science & Engg.
K.S. Institute of Technology
Bengaluru -560 109

MODULE– 1

Basic Structure of Computers & Machine Instructions and Programs

1. BASIC OPERATIONAL CONCEPTS

The program to be executed is stored in memory. Instructions are accessed from memory to the processor one by one and executed.

STEPS FOR INSTRUCTION EXECUTION

Consider the following instruction

Ex: 1 Add LOCA, R₀

This instruction is in the form of the following instruction format

OpcodeSource, Destination

Where Add is the *operation code*, LOCA is the Memory operand and R₀ is Register operand. This instruction adds the contents of memory location LOCA with the contents of Register R₀ and the result is stored in R₀ Register.

The symbolic representation of this instruction is

$$[LOCA] + [R_0] \rightarrow R_0$$

The contents of memory location LOCA and Register R₀ before and after the execution of this instruction is as follows

Before instruction execution

LOCA = 23H

R₀ = 22H

After instruction execution

LOCA = 23H

R₀ = 45H

The steps for instruction execution are as follows

1. Fetch the instruction from memory into the IR (instruction register in CPU).
2. Decode the instruction
3. Access the Memory Operand
4. Access the Register Operand
5. Perform the operation according to the Operation Code.
6. Store the result into the Destination Memory location or Destination Register.

Ex:2 Add R₁, R₂, R₃

This instruction is in the form of the following instruction format

Opcode, Source-1, Source-2, Destination

Where R1 is Source Operand-1, R2 is the Source Operand-2 and R3 is the Destination. This instruction adds the contents of Register R1 with the contents of R2 and the result is placed in R3 Register.

The symbolic representation of this instruction is

$$[R1] + [R2] \rightarrow R3$$

The contents of Registers R1,R2,R3 before and after the execution of this instruction is as follows.

Before instruction execution

R1 = 24H

R2 = 34H

R3 = 38H

After instruction execution

R1 = 24H

R2 = 34H

R3 = 58H

The steps for instruction execution is as follows

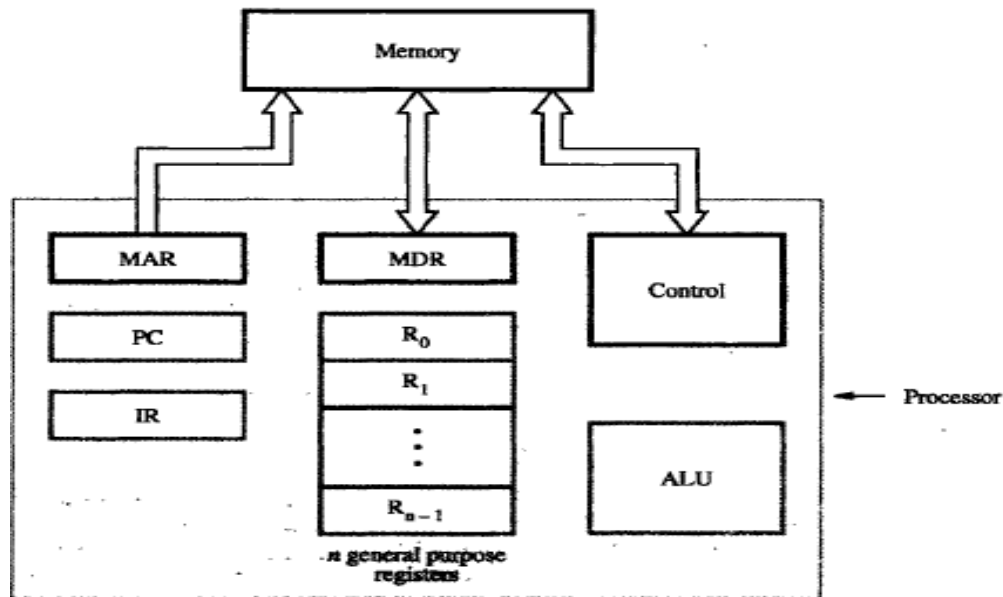
1. Fetch the instruction from memory into the IR.
2. Decode the instruction
3. Access the First Register Operand R1
4. Access the Second Register Operand R2
5. Perform the operation according to the Operation Code.
6. Store the result into the Destination Register R3.

CONNECTION BETWEEN MEMORY AND PROCESSOR

The connection between Memory and Processor is as shown in the figure.

The Processor consists of different types of registers.

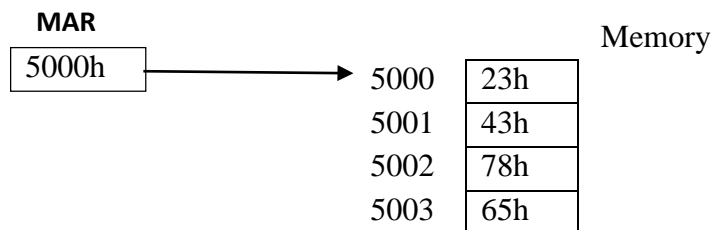
1. MAR (Memory Address Register)
2. MDR (Memory Data Register)
3. Control Unit
4. PC (Program Counter)
5. General Purpose Registers
6. IR (Instruction Register)
7. ALU (Arithmetic and Logic Unit)



The functions of these registers are as follows

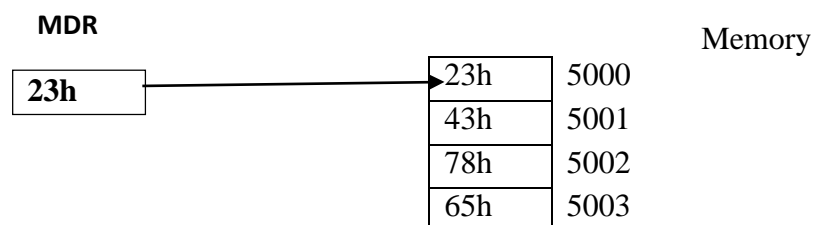
MAR

- It establishes communication between Memory and Processor
- It stores the address of the Memory Location as shown in the figure.



MDR

- It also establishes communication between Memory and the Processor.
- It stores the **contents** of the memory location (data or operand), written into or read from memory as shown in the figure.



CONTROL UNIT

- It controls the data transfer operations between memory and the processor.
- It controls the data transfer operations between I/O and processor.
- It generates control signals for Memory and I/O devices.

PC (PROGRAM COUNTER)

- It is a special purpose register used to hold the address of the next instruction to be executed.
- The contents of PC are incremented by 1 or 2 or 4, during the execution of current instruction.
- The contents of PC are incremented by 1 for 8 bit CPU, 2 for 16 bit CPU and for 4 for 32 bit CPU.

GENERAL PURPOSE REGISTER / REGISTER ARRAY

The structure of register file is as shown in the figure

R₀
R₁
R₂
.
R_{n-1}

It consists of set of registers.

- A register is defined as group of flip flops. Each flip flop is designed to store 1 bit of data.
- It is a storage element.
- It is used to store the data temporarily during the execution of the program(eg: result).
- It can be used as a pointer to Memory.
- The Register size depends on the processing speed of the CPU
- EX: Register size = 8 bits for 8 bit CPU

IR (INSTRUCTION REGISTER)

It holds the instruction to be executed. It notifies the control unit, which generates timing signals that controls various operations in the execution of that instruction.

ALU (ARITHMETIC and LOGIC UNIT)

- It performs arithmetic and logical operations on given data.

Steps for reading the instruction.

PC contents are transferred to MAR and read signal is sent to memory by control unit.

The data from memory location is read and sent to MDR.

The content of MDR is moved to IR.

[PC] → MAR → Memory → MDR → IR
CU (read signal)

2. BUS STRUCTURE

Bus is defined as set of parallel wires used for data communication. Each wire carries 1 bit of data. There are 3 of buses, namely

1. Address bus
2. Data bus and
3. Control bus1.

1. *Addressbus :*

- It is unidirectional.
- The processor (CPU) sends the address of an I/O device or Memory device by means of this bus.

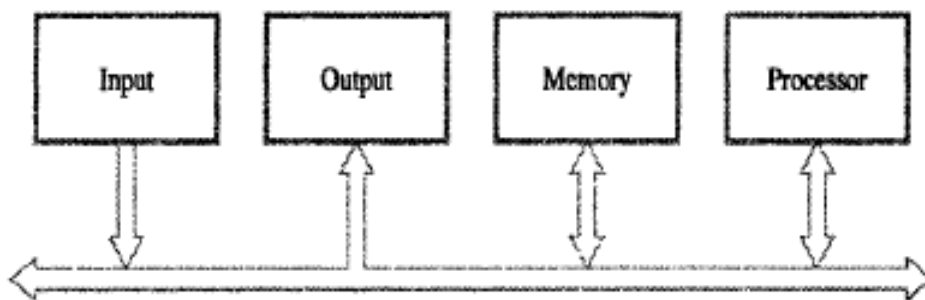
2. *Data bus*

- It is a bidirectional bus.
- The CPU sends data from Memory to CPU and vice versa as well as from I/O to CPU and vice versa by means of this bus.

3. *Control bus:*

- This bus carries control signals for Memory and I/O devices. It generates control signals for Memory namely MEMRD and MEMWR and control signals for I/O devices namely IORD and IOWR.
- It also generates special control signal to differentiate between Memory and I/O device. This control signal is called as M/IO.
- $M/IO = 1$ for Memory operations and $M/IO = 0$ for I/O operations

The structure of single bus organization is as shown in the figure.



- The I/O devices, Memory and CPU are connected to this bus is as shown in the figure.
- It establishes communication between two devices.

Features of Single bus organization are

- Less Expensive
- Flexible to connect I/O devices.
- Poor performance due to single bus.

There is a variation in the devices connected to this bus in terms of speed of operation. Few devices like keyboard, are very slow. So to provide the synchronization between two

devices, a buffer register is attached to each device. It holds the data temporarily during the data transfer between two devices.

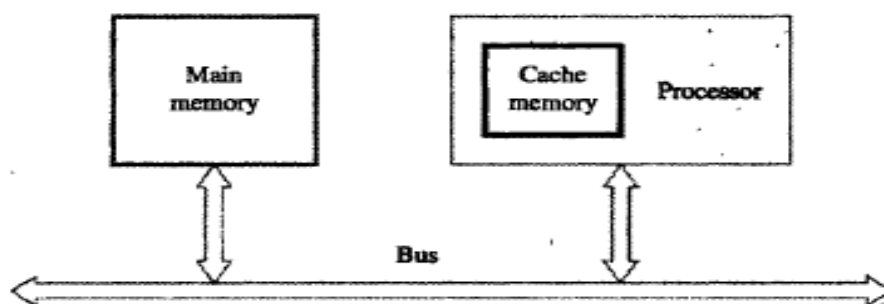
3. PERFORMANCE

- The performance of a Computer System is based on hardware design of the processor and the instruction set of the processors.
- To obtain high performance of computer system it is necessary to reduce the execution time of the processor.
- Execution time: It is defined as total time required executing one complete program.
- The processing time of a program includes time taken to read inputs, display outputs, system services, execution time etc.
- The performance of the processor is inversely proportional to execution time of the processor.
- More performance = Less Execution time.
- Less Performance = More Execution time.

The Performance of the Computer System is based on the following factors

1. *Cache Memory*
2. *Processor clock*
3. *Basic Performance Equation*
4. *Pipelining and Super Scalar operation*
5. *Instruction set*
6. *Compiler*

CACHE MEMORY: It is defined as a *fast access memory* located in between CPU and Memory. It is part of the processor as shown in the fig



The processor needs more time to read the data and instructions from main memory because main memory is away from the processor as shown in the figure. Hence it slows down the performance of the system.

The processor needs less time to read the data and instructions from Cache Memory because it is part of the processor. Hence it improves the performance of the system.

PROCESSOR CLOCK: The processor circuits are controlled by timing signals called as Clock. It defines constant time intervals and are called as Clock Cycles. To execute one instruction there are 3 basic steps namely

1. Fetch
2. Decode
3. Execute.

The processor uses one clock cycle to perform one operation as shown in the figure

Clock Cycle	→	T1	T2	T3
Instruction	→	Fetch	Decode	Execute

The performance of the processor depends on the length of the clock cycle. To obtain high performance reduce the length of the clock cycle. Let 'P' be the number of clock cycles generated by the Processor and 'R' be the Clock rate.

The Clock rate is inversely proportional to the number of clock cycles.

i.e $R = 1/P$.

Cycles/second is measured in Hertz (Hz). Eg: 500MHz, 1.25GHz.

Two ways to increase the clock rate –

- Improve the IC technology by making the logical circuit work faster, so that the time taken for the basic steps reduces.
- Reduce the clock period, P.

BASIC PERFORMANCE EQUATION

Let 'T' be *total time* required to execute the program.

Let 'N' be the *number of instructions* contained in the program.

Let 'S' be the *average number of steps* required to one instruction.

Let 'R' be number of clock cycles per second generated by the processor to execute one program.

Processor Execution Time is given by

$$T = N * S / R$$

This equation is called as Basic Performance Equation.

For the programmer the value of T is important. To obtain high performance it is necessary to reduce the values of N & S and increase the value of R

Performance of a computer can also be measured by using **benchmark** programs.

SPEC (System Performance Evaluation Corporation) is an organization, that measures performance of computer using SPEC rating.

$$SPEC = \frac{\text{Running time of reference Computer}}{\text{Running time of computer under test}}$$

DIFFERENCES MULTIPROCESSOR AND MULTICOMPUTER

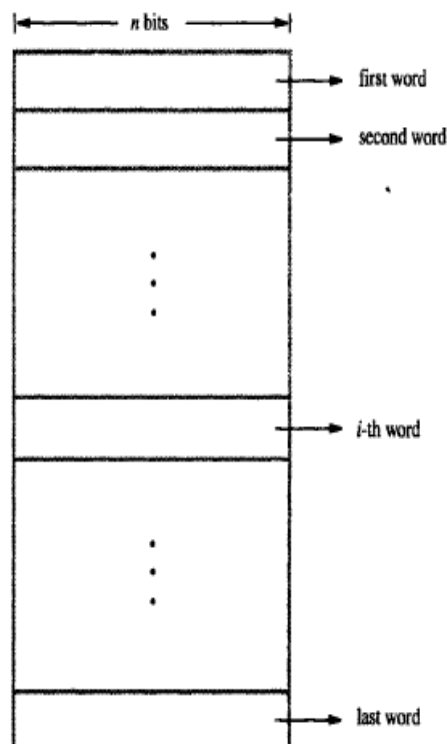
MULTIPROCESSOR	MULTICOMPUTER
1. It is a process of interconnection of two or more processors by means of system bus.	It is a process of interconnection of two or more computers by means of system bus.
2. The diagrammatic representation of multiprocessor is as shown in the figure	The diagrammatic representation of multicomputer is as shown in the figure
3. It uses common memory to hold the data and instructions.	It has its own memory to store data and instructions.
4. Complexity in hardware design.	Not much complexity in hardware design.
5. Difficult to program for multiprocessor system.	Easy to program for multiprocessor system

4. MEMORY LOCATIONS AND ADDRESSES

1. Memory is a storage device. It is used to store character operands, data operands and instructions.
2. It consists of number of semiconductor cells and each cell holds 1 bit of information. A group of 8 bits is called as byte and a group of 16 or 32 or 64 bits is called as word.

Word length = 16 for 16 bit CPU and Word length = 32 for 32 bit CPU. Word length is defined as number of bits in a word.

- Memory is organized in terms of bytes or words.
- The organization of memory for 32 bit processor is as shown in the fig.



The contents of memory location can be accessed for read and write operation. The memory is accessed either by specifying address of the memory location or by name of the memory location.

- **Address space :** It is defined as number of bytes accessible to CPU and it depends on the number of address lines.

5. BYTE ADDRESSABILITY

Each byte of the memory are addressed, this addressing used in most computers are called byte addressability. Hence Byte Addressability is the process of assignment of address to successive bytes of the memory. The successive bytes have the addresses 1, 2, 3, 4..... 2^n-1 . The memory is accessed in words.

In a 32 bit machine, each word is 32 bit and the successive addresses are 0,4,8,12,... and so on.

Address	32 – bit word			
0000	0 th byte	1 st byte	2 nd byte	3 rd byte
0004	4 th byte	5 th byte	6 th byte	7 th byte
0008	8 th byte	9 th byte	10 th byte	11 th byte
0012	12 th byte	13 th byte	14 th byte	15 th byte
.....
n-3	n-3 th byte	n-2 th byte	n-1 th byte	n th byte

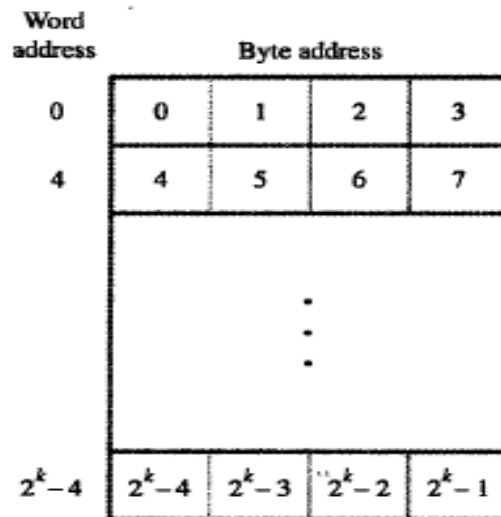
BIG ENDIAN and LITTLE ENDIAN ASSIGNMENT

Two ways in which byte addresses can be assigned in a word.

Or

Two ways in which a word is stored in memory.

1. Big endian
2. Little endian

BIG ENDIAN ASSIGNMENT

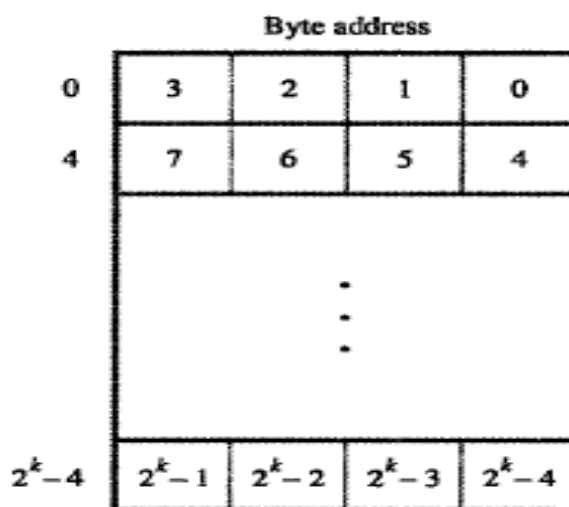
In this technique lower byte of data is assigned to higher address of the memory and higher byte of data is assigned to lower address of the memory.

The structure of memory to represent 32 bit number for big endian assignment is as shown in the above figure.

LITTLE ENDIAN ASSIGNMENT

In this technique lower byte of data is assigned to lower address of the memory and higher byte of data is assigned to higher address of the memory.

The structure of memory to represent 32 bit number for little endian assignment is as shown in the fig.



Eg – store a word “JOHNSENA” in memory starting from word 1000, using Big Endian and Little endian.

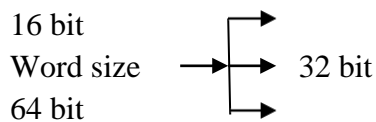
Bigendian -

1000	J	O	H	N
1004	S	E	N	A

Little endian -

1000	N	H	O	J
1004	A	N	E	S

WORD ALIGNMENT



The structure of memory for 16 bit CPU, 32 bit CPU and 64 bit CPU are as shown in the figures 1,2 and 3 respectively

For 16 bit CPU

5000	34H
5002	65H
5004	86H
5006	93H
5008	45H

For 32 bit CPU

5000	34H
5004	65H
5008	86H
5012	93H
5016	45H

For 64 bit CPU

5000	34H
5008	65H
5016	86H
5024	93H
5032	45H

It is process of assignment of addresses of two successive words and this address is the number of bytes in the word is called as Word alignment.

ACCESSING CHARACTERS AND NUMBERS

The character occupies 1 byte of memory and hence byte address for memory.

The numbers occupies 2 bytes of memory and hence word address for numbers.

6. MEMORY OPERATION

Both program instructions and operands are in memory. To execute each instruction has to be read from memory and after execution the results must be written to memory.

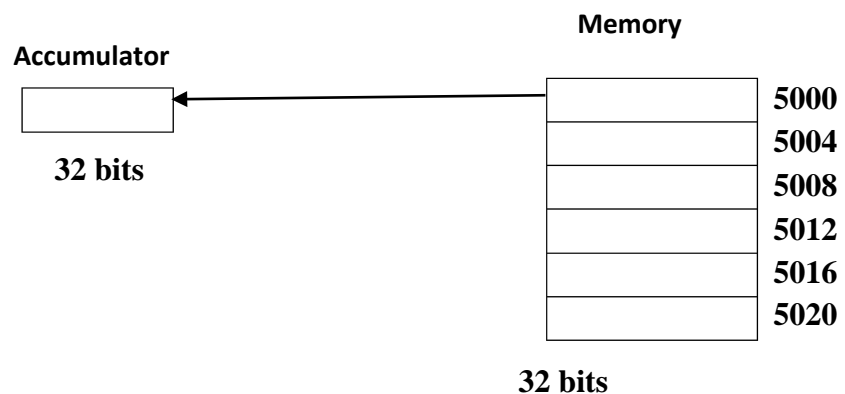
There are two types of memory operations namely 1. Memory read and 2. Memory write

Memory read operation [Load/ Read / Fetch]

Memory write operation [Store/ write]

1. **MEMORY READ OPERATION:**

- ✓ It is the process of transferring of 1 word of data from memory into Accumulator (GPR).
- ✓ It is also called as Memory fetch operation.
- ✓ The Memory read operation can be implemented by means of LOAD instruction.
- ✓ The LOAD instruction transfers 1 word of data(1 word = 32 bits) from Memory into the Accumulator as shown in the fig.



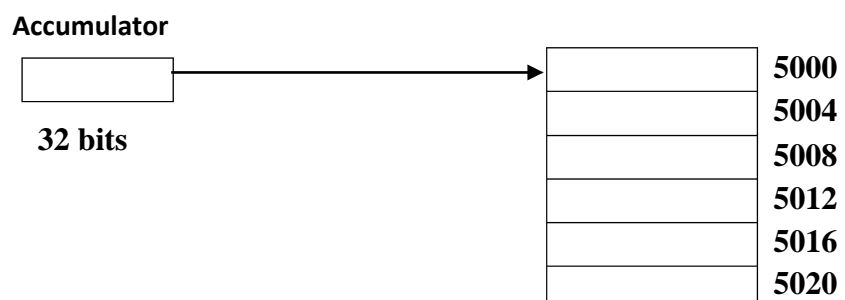
Steps for Memory Read Operation

- (1) The processor loads MAR(Memory Address Register)with the address of the memory location.
- (2) The Control unit of processor issues memory read control signal to enable the memory component for read operation.
- (3) The processor reads the data from memory into the Accumulator by means of bi-directional data bus.

[MAR] → Memory → Accumulator

MEMORY WRITE OPERATION

- It is the process of transferring the 1 word of data from Accumulator into the Memory.
- The Memory write operation can be implemented by means of STORE instruction. The STORE instruction transfers 1 word of data from Accumulator into the Memory location as shown in the fig.



32 bits

Steps for Memory Write Operation

- The processor loads MAR with the address of the Memory location.
- The Control Unit issues the Memory Write control signal.
- The processor transfers 1 word of data from the Accumulator into the Memory location by means of bi-directional data bus.

7. COMPUTER OPERATIONS (OR) INSTRUCTIONS AND INSTRUCTION EXECUTION

The Computer is designed to perform 4 types of operations, namely

- Data transfer operations
- ALU Operations
- Program sequencing and control.
- I/O Operations.

1. Data Transfer Operations

a) Data transfer between two registers.

Format: Opcode Source1 , Destination
representation of this instruction is $R1 \rightarrow R2$.

Ex : MOV R₁, R₂ : R₁ and R₂ are the registers.

Where MOV is the operation code, R₁ is the source operand and R₂ is the destination operand. This instruction transfers the contents of R₁ to R₂.

EX: Before the execution of MOV R₁ The processor uses MOV instruction to perform data transfer operation between two registers

The mathematical

,R₂, the contents of R₁ and R₂ are as follows R₁ = 34h and R₂ = 65h

After the execution of MOV R₁, R₂, the contents of R₁ and R₂ are as follows

R₁ = 34H and R₂ = 34H

b) Data transfer from memory to register

The processor uses **LOAD** instruction to perform data transfer operation from memory to register. The mathematical representation of this instruction is

[LOCA] \rightarrow ACC. Where ACC is the Accumulator.

Format :opcode operand

Ex: LOAD LOCA

For this instruction Memory Location is the source and Accumulator is the destination.

c) Data transfer from Accumulator register to memory

The processor uses **STORE** instruction to perform data transfer operation from Accumulator register to memory location. The mathematical representation of this instruction is

[ACC] \rightarrow LOCA. Where, ACC is the Accumulator.

Format:opcode operand

Ex: STORE LOCA

For this instruction accumulator is the source and memory location is the destination.

2. ALU Operations

The instructions are designed to perform arithmetic operations such as Addition, Subtraction, Multiplication and Division as well as logical operations such as AND, OR and NOT operations.

Ex1: ADD R₀, R₁

The mathematical representation of this instruction is as follows:

$R_1 \leftarrow [R_0] + [R_1]$; Adds the content of R₀ with the content of R₁ and result is placed in R₁.

Ex2: SUB R₀, R₁

The mathematical representation of this instruction is as follows:

$R_1 \leftarrow [R_0] - [R_1]$; Subtracts the content of R₀ from the content of R₁ and result is placed in R₁.

EX3: AND R₀, R₁; It Logically multiplies the content of R₀ with the content of R₁ and result is stored in R₁. ($R_1 = R_0 \text{ AND } R_1$)

Ex4: NOT R₀; It performs the function of complementation.

a) Input Operation: It is a process of transferring one WORD of data from DATA IN register to processor register.

Ex: MOV DATAIN, R₀

The mathematical representation of this instruction is as follows,

$R_0 \leftarrow [\text{DATAIN}]$

1. I/O Operations: The instructions are designed to perform INPUT and OUTPUT operations. The processor uses MOV instruction to perform I/O operations.

The input Device consists of one temporary register called as DATAIN register and output register consists of one temporary register called as DATAOUT register.

a) Output Operation: It is a process of transferring one WORD of data from processor register to DATAOUT register.

Ex: MOV R₀, DATAOUT

The mathematical representation of this instruction is as follows,

$[R_0] \rightarrow \text{DATAOUT}$

REGISTER TRANSFER NOTATION

- There are 3 locations to store the operands during the execution of the program namely 1. Register 2. Memory location 3. I/O Port. Location is the storage space used to store the data.
- The instructions are designed to transfer data from one location to another location. Consider the first statement to transfer data from one location to another location
- “Transfer the contents of Memory location whose symbolic name is given by AMOUNT into processor register R₀.”
- The mathematical representation of this statement is given by
 $R_0 \leftarrow [\text{AMOUNT}]$
 Consider the second statement to add data between two registers
- “Add the contents of R₀ with the contents of R₁ and result is stored in R₂”

- The mathematical representation of this statement is given by

$$R_2 \leftarrow [R_0] + [R_1].$$

Such a notation is called as “Register Transfer Notation”.

It uses two symbols

1. A pair of square brackets [] to indicate the contents of Memory location and
2. \leftarrow to indicate the data transfer operation.

ASSEMBLY LANGUAGE NOTATION

Consider the first statement to transfer data from one location to another location

- “Transfer the contents of Memory location whose symbolic name is given by AMOUNT into processor register R_0 .”

- The assembly language notation of this statement is given by

MOV AMOUNT, R_0

Opcode Source Destination

This instruction transfers 1 word of data from Memory location whose symbolic name is given by AMOUNT into the processor register R_0 .

- The mathematical representation of this statement is given by

$$R_0 \leftarrow [AMOUNT]$$

Consider the second statement to add data between two registers

- “Add the contents of R_0 with the contents of R_1 and result is stored in R_2 ”
- The assembly language notation of this statement is given by

ADD R_0 , R_1 , R_2

Opcode source1, Source2, Destination

This instruction adds the contents of R_0 with the contents of R_1 and result is stored in R_2 .

- The mathematical representation of this statement is given by

$$R_2 \leftarrow [R_0] + [R_1].$$

Such a notations are called as “Assembly Language Notations”

BASIC INSTRUCTION TYPES

There are 3 types basic instructions namely

1. Three address instruction format
2. Two address instruction format
3. One address instruction format

Consider the arithmetic expression $Z = A + B$, Where A,B,Z are the Memory locations.

Steps for evaluation

1. Access the first memory operand whose symbolic name is given by A.
2. Access the second memory operand whose symbolic name is given by B.
3. Perform the addition operation between two memory operands.
4. Store the result into the 3rd memory location Z.
5. The mathematical representation is $Z \leftarrow [A] + [B]$.

- a) Three address instruction format : Its format is as follows

opcode	Source-1	Source-2	destination
--------	----------	----------	-------------

Destination \leftarrow [source-1] + [source-2]

Ex: ADD A, B, Z

$Z \leftarrow [A] + [B]$

- a) Two address instruction format : Its format is as follows

opcode	Source	destination
--------	--------	-------------

Destination \leftarrow [source] + [destination]

The sequence of two address m/c instructions to evaluate the arithmetic expression

$Z \leftarrow A + B$ are as follows

MOV A, R₀

MOV B, R₁

ADD R₀, R₁

MOV R₁, Z

- b) One address instruction format : Its format is as follows

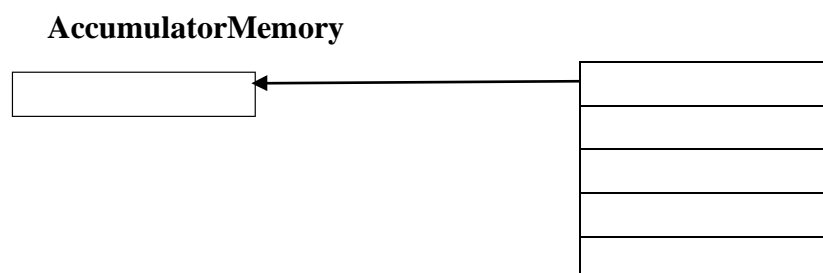
opcode	operand
--------	---------

Ex1: LOAD B

This instruction copies the contents of memory location whose symbolic name is given by 'B' into the Accumulator as shown in the figure.

The mathematical representation of this instruction is as follows

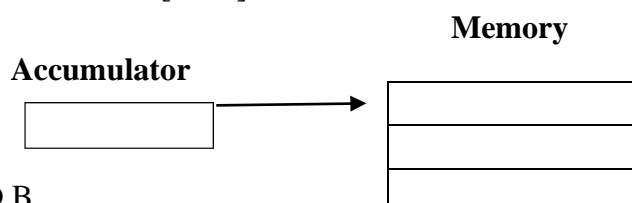
$ACC \leftarrow [B]$



Ex2: STORE B

This instruction copies the contents of Accumulator into memory location whose symbolic name is given by 'B' as shown in the figure. The mathematical representation is as follows

$B \leftarrow [ACC]$.



Ex3: ADD B

- This instruction adds the contents of Accumulator with the contents of Memory location 'B' and result is stored in Accumulator.
- The mathematical representation of this instruction is as follows

$$ACC \leftarrow [ACC] + [B]$$

STRAIGHT LINE SEQUENCING AND INSTRUCTION EXECUTION

Consider the arithmetic expression

$C = A + B$, Where A,B,C are the memory operands.

The mathematical representation of this instruction is

$C = [A] + [B]$.

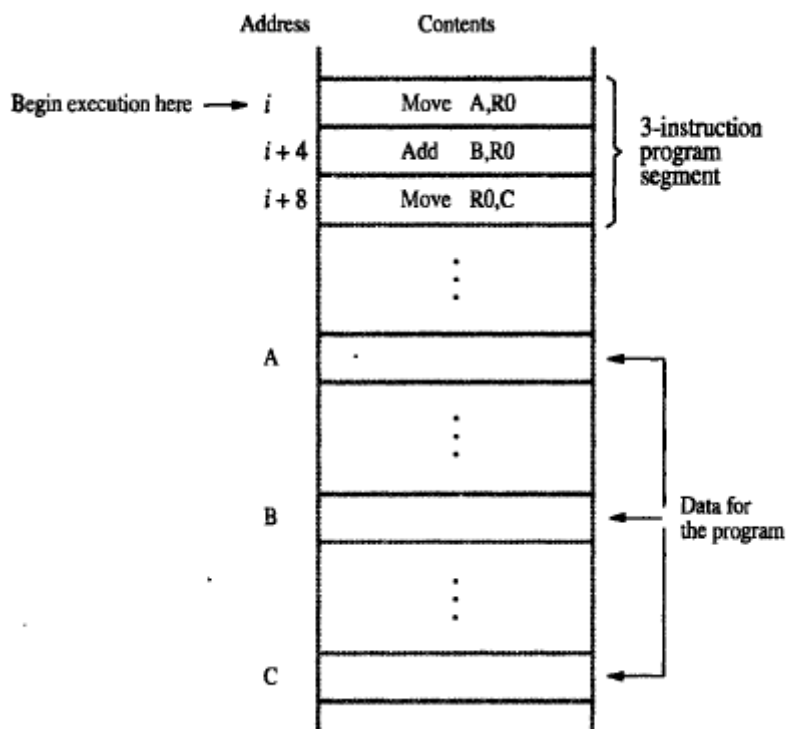
The sequence of instructions using two address instruction format are as follows

```
MOV  A,  R0
ADD  B,  R0
MOV  R0, C
```

Such a program is called as 3 instruction program.

NOTE: The size of each instruction is 32 bits.

- The 3 instruction program is stored in the successive memory locations of the processor is as shown in the fig.
- The system bus consists of uni-directional address bus, bi-directional data bus and control



bus

"It is the process of accessing the 1st instruction from memory whose address is stored in program counter into Instruction Register (IR) by means of bi-directional data bus and at the

same time after instruction access the contents of PC are incremented by 4 in order to access the next instruction. Such a process is called as “Straight Line Sequencing”.

INSTRUCTION EXECUTION

There are 4 steps for instruction execution

- 1 Fetch the instruction from memory into the Instruction Register (IR) whose address is stored in PC.

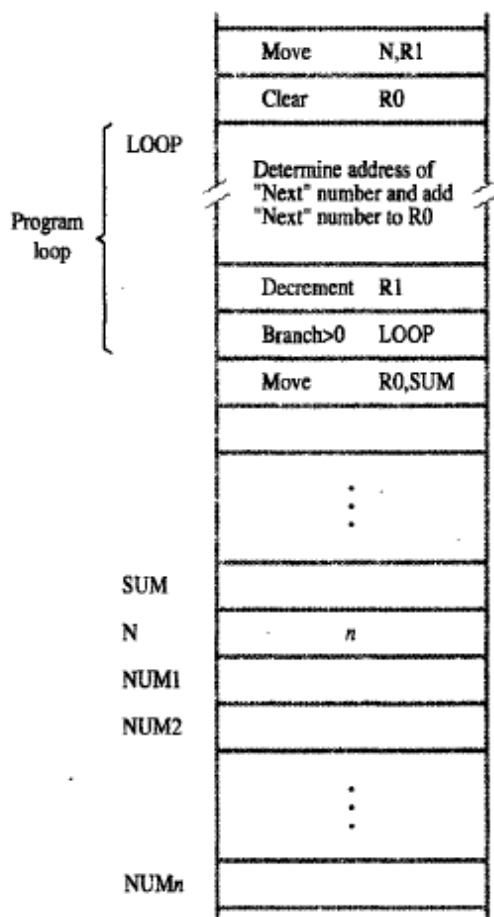
$$IR \leftarrow [PC]$$
- 2 Decode the instruction.
- 3 Perform the operation according to the opcode of an instruction
- 4 Load the result into the destination.
- 5 During this process, Increment the contents of PC to point to next instruction (In 32 bit machine increment by 4 address)

$$PC \leftarrow [PC] + 4.$$
- 6 The next instruction is fetched, from the address pointed by PC.

BRANCHING

Suppose a list of ‘N’ numbers have to be added. Instead of adding one after the other, the add statement can be put in a loop. The loop is a straight-line of instructions executed as many times as needed.

The ‘N’ value is copied to R1 and R1 is decremented by 1 each time in loop. In the loop find the value of next element and add it with R0.



In conditional branch instruction, the loop continues by coming out of sequence only if the condition is true. Here the PC value is set to 'LOOP' if the condition is true.

Branch > 0 LOOP // if >0 go to LOOP

The PC value is set to LOOP, if the previous statement value is >0 ie. after decrementing R1 value is greater than 0.

If R1 value is not greater than 0, the PC value is incremented in a normal sequential way and the next instruction is executed.

CONDITION CODE BITS

- The processor consists of series of flip-flops to store the status information after ALU operation.
- It keeps track of the results of various operations, for subsequent usage.
- The series of flip-flop-flops used to store the status and control information of the processor is called as "Condition Code Register". It defines 4 flags. The format of condition code register is as follows.

C	V	Z	N
---	---	---	---

- 1 N (NEGATIVE) Flag:

It is designed to differentiate between positive and negative result.

It is set 1 if the result is negative, and set to 0 if result is positive.

- 2 Z (ZERO) Flag:

It is set to 1 when the result of an ALU operation is found to zero, otherwise it is cleared.

- 3 V (OVER FLOW) Flag:

In case of 2^s Complement number system n-bit number is capable of representing a range of numbers and is given by -2^{n-1} to $+2^{n-1}$. The Over-Flow flag is set to 1 if the result is found to be out of this range.

- 4 C (CARRY) Flag :

This flag is set to 1 if there is a carry from addition or borrow from subtraction, otherwise it is cleared.

8. Addressing Modes

The various formats of representing operand in an instruction or location of an operand is called as "Addressing Mode". The different types of Addressing Modes are

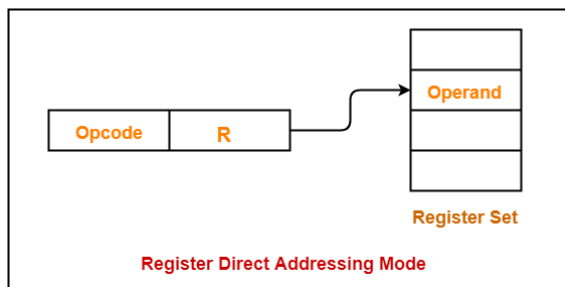
- a) Register Addressing
- b) Direct Addressing
- c) Immediate Addressing
- d) Indirect Addressing

- e) Index Addressing
- f) Relative Addressing
- g) Auto Increment Addressing
- h) Auto Decrement Addressing

a. REGISTER ADDRESSING:

In this mode operands are stored in the registers of CPU. The name of the register is directly specified in the instruction.

Ex: MOVE R₁,R₂ Where R₁ and R₂ are the Source and Destination registers respectively. This



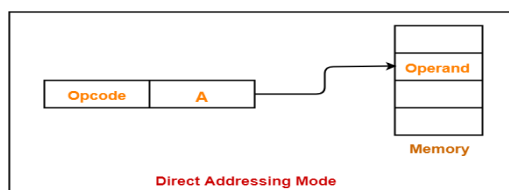
instruction transfers 32 bits of data from R₁ register into R₂ register. This instruction does not refer memory for operands. The operands are directly available in the registers.

b. DIRECT ADDRESSING

It is also called as Absolute Addressing Mode. In this addressing mode operands are stored in the memory locations. The name of the memory location is directly specified in the instruction.

Ex: MOVE LOCA, R₁ : Where LOCA is the memory location and R₁ is the Register.

This instruction transfers 32 bits of data from memory location X into the General Purpose

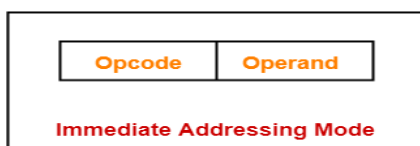


Register R₁.

c. IMMEDIATE ADDRESSING

In this Addressing Mode operands are directly specified in the instruction. The source field is used to represent the operands. The operands are represented by # (hash) sign.

Ex: MOVE #23, R₀



d. INDIRECT ADDRESSING

In this Addressing Mode effective address of an operand is stored in the memory location or General Purpose Register.

The memory locations or GPR^S are used as the memory pointers.

Memory pointer: It stores the address of the memory location.

There are two types Indirect Addressing

- i) Indirect through GPR^S
- ii) Indirect through memory location

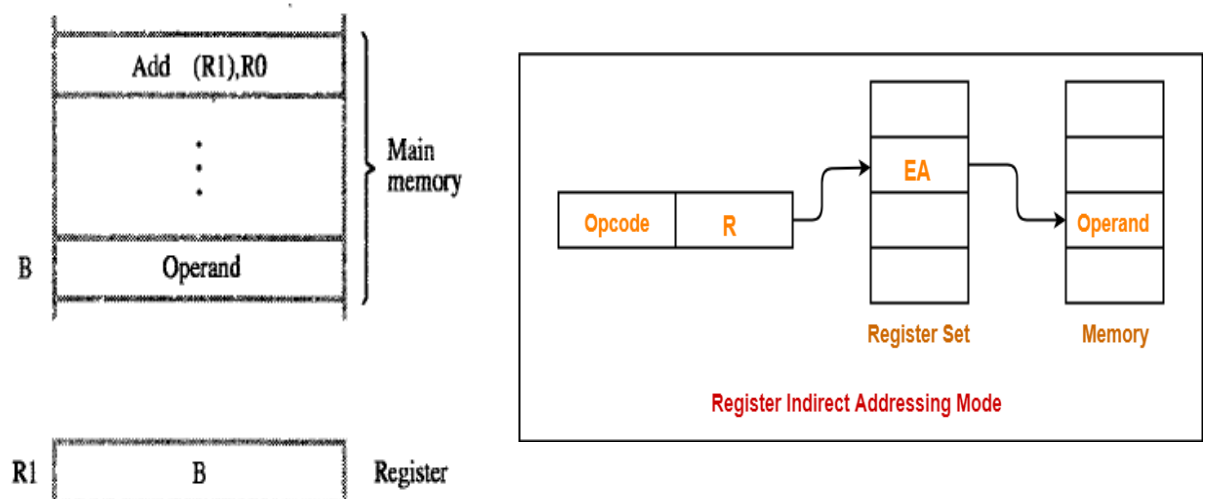
i) Indirect Addressing Mode through GPR^S.

In this Addressing Mode the effective address of an operand is stored in the one of the General Purpose Register of the CPU.

Ex: ADD (R₁), R₀ ; Where R₁ and R₀ are GPR^S.

This instruction adds the data from the memory location whose address is stored in R₁ with the contents of R₀ Register and the result is stored in R₀ register as shown in the fig.

The diagrammatic representation of this addressing mode is as shown in the fig.



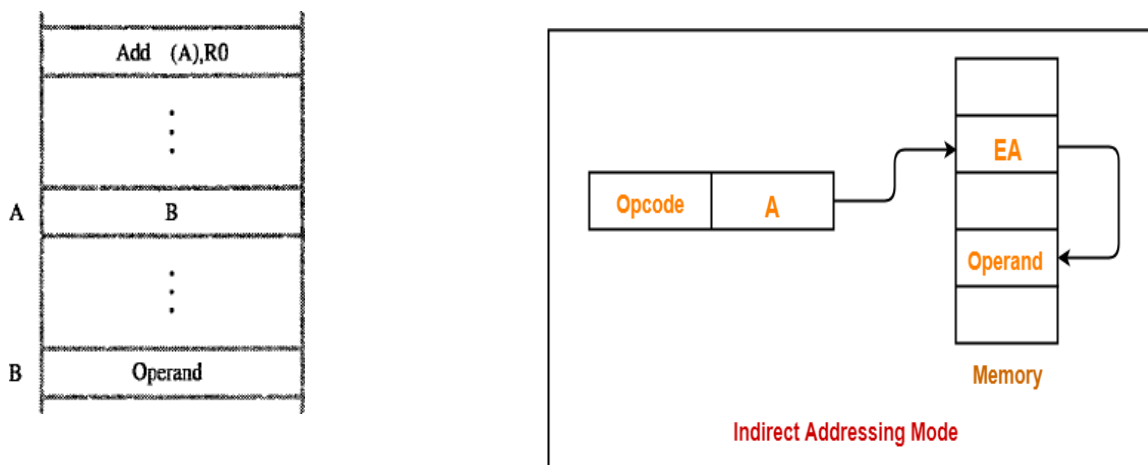
ii) Indirect Addressing Mode through Memory Location.

In this Addressing Mode, effective address of an operand is stored in the memory location.

Ex: ADD (X), R₀

This instruction adds the data from the memory location whose address is stored in 'X' memory location with the contents of R₀ and result is stored in R₀ register.

The diagrammatic representation of this addressing mode is as shown in the fig.

**e. INDEX ADDRESSING MODE**

In this addressing mode, the effective address of an operand is computed by adding constant value with the contents of Index Register and any one of the General Purpose Register namely R₀ to R_{n-1} can be used as the Index Register. The constant value is directly specified in the instruction.

The symbolic representations of this mode are as follows

1. X (R_i) where X is the Constant value and R_j is the GPR.

It can be represented as

$$EA \text{ of an operand} = X + (R_i)$$

2. (R_i, R_j) Where R_i and R_j are the General Purpose Registers used to store addresses of an operand and constant value respectively. It can be represented as

The EA of an operand is given by

$$EA = (R_i) + (R_j)$$

3. X (R_i, R_j) Where X is the constant value and R_i and R_j are the General Purpose Registers used to store the addresses of the operands. It can be represented as

The EA of an operand is given by

$$EA = (R_i) + (R_j) + X$$

There are two types of Index Addressing Modes

- i) **Offset is given as constant.**
- ii) **Offset is in Index Register.**

Note : Offset : It is the difference between the starting effective address of the memory location and the effective address of the operand fetched from memory.

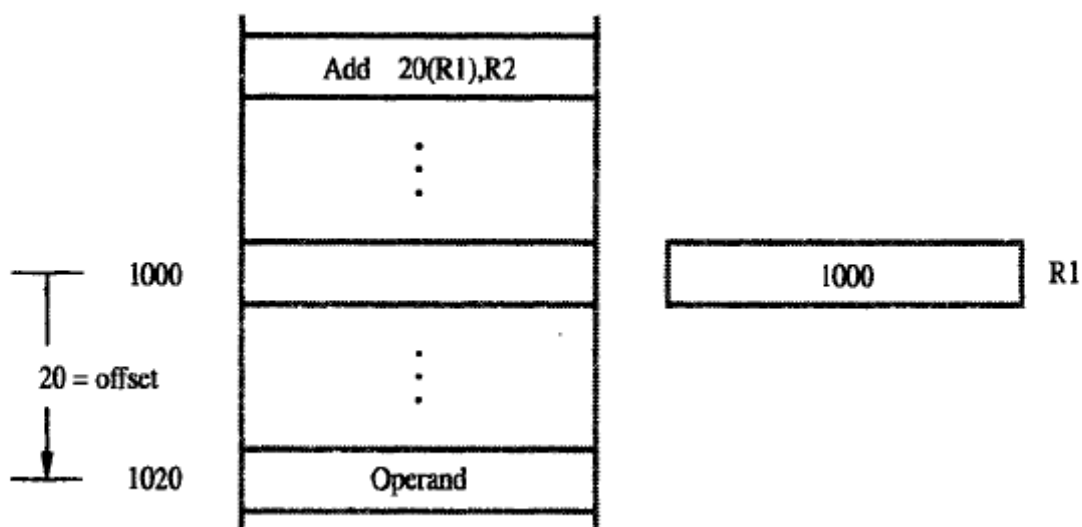
i) Offset is given as constant

Ex: ADD 20(R₁), R₂

The EA of an operand is given by

$$EA = 20 + [R_1]$$

This instruction adds the contents of memory location whose EA is the sum of contents of R₁ with 20 and with the contents of R₂ and result is placed in R₂ register. The diagrammatic representation of this mode is as shown in the fig.



ii) Offset is in Index Register

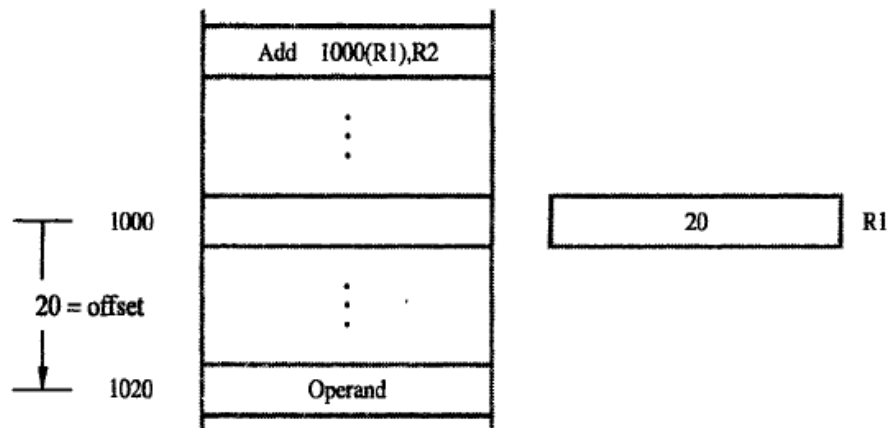
Ex: ADD 1000(R₁), R₂ R₁ holds the offset address of an operand.

The EA of an operand is given by

$$EA = 1000 + [R_1]$$

This instruction adds the data from the memory location whose address is given by [1000 + [R₁]] with the contents of R₂ and result is placed in R₂ register.

The diagrammatic representation of this mode is as shown in the fig.



f. RELATIVE ADDRESSING MODE:

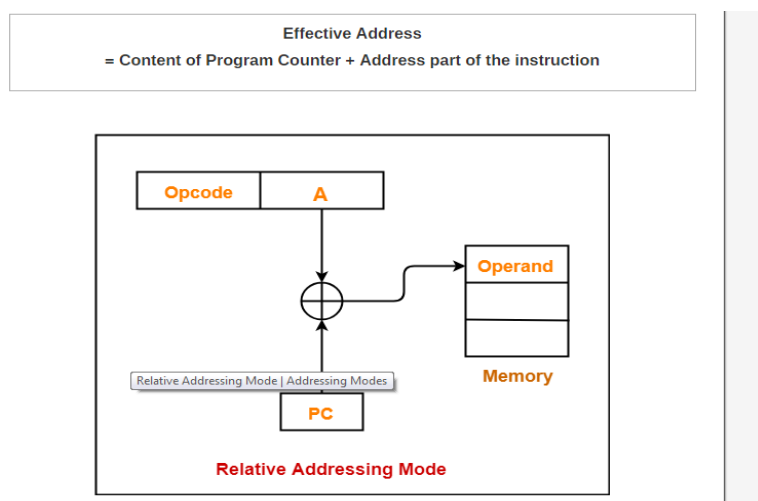
In this Addressing Mode EA of an operand is computed by the Index Addressing Mode. This Addressing Mode uses PC (Program Counter) to store the EA of the next instruction instead of GPR.

The symbolic representation of this mode is $X(PC)$. Where X is the offset value and PC is the Program Counter to store the address of the next instruction to be executed.

It can be represented as

$EA \text{ of an operand} = X + (PC)$.

This Addressing Mode is useful to calculate the EA of the target memory location.



g. AUTO INCREMENT ADDRESSING MODE

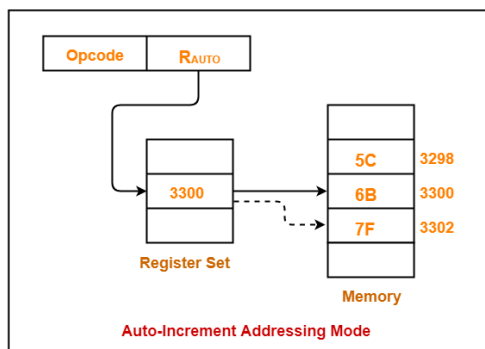
In this Addressing Mode , EA of an operand is stored in the one of the GPR^s of the CPU. This Addressing Mode increment the contents of memory register by 4 memory locations after operand access.

The symbolic representation is

$(R_i)+$ Where R_i is the one of the GPR.

Ex: MOVE $(R_1)+$, R_2

This instruction transfer's data from the memory location whose address is stored in R_1 into R_3 register and then it increments the contents of R_1 by 4 memory locations.

**h. AUTO DECREMENT ADDRESSING MODE**

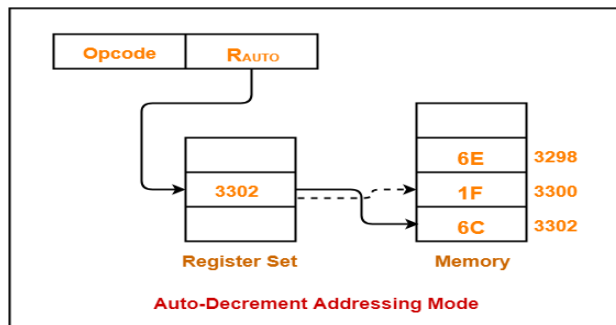
In this Addressing Mode , EA of an operand is stored in the one of the GPR^s of the CPU. This Addressing Mode decrements the contents of memory register by 4 memory locations and then transfers the data to destination.

The symbolic representation is

$-(R_i)$ Where R_i is the one of the GPR.

Ex: MOVE $-(R_1)$, R_2

This instruction first decrements the contents of R_1 by 4 memory locations and then transfer's data of that location to destination register.



9. ASSEMBLY LANGUAGE

- The Assembly language uses Symbolic names to represent opcodes, memory locations and registers.
- The Assembler converts Assembly language programs into machine level language programs.
- The Assembly language is called as “Source program” and m/c language program is called as “Object program”.
- The Assembler converts source program into object program.
- A set of symbolic names and set of rules for their use forms a programming language and is called as “Assembly Language”.

Ex: MOV, ADD, LOAD → Opcodes.

X, Y, AMOUNT → Memory locations.

Where R0 to R7 are the registers.

- The examples for Assembly Language instructions are as follows

MOV R₀, R₁ ; Register Addressing.

MOV #23, R₀ ; Immediate Addressing.

9.1 DIRECTIVES

There are two types of instructions namely i) Processor Instructions and ii) Assembler Instructions.

- The Processor instructions are converted into m/c instructions by means of Assembler. Hence Assembler generates m/c code for processor instructions.
- The Assembler instructions are not converted into m/c instructions and hence Assembler does not generate the m/c code for Assembler instructions

“A set of commands given to Assembler while converting source program into object program is called as Assembler Directive”.

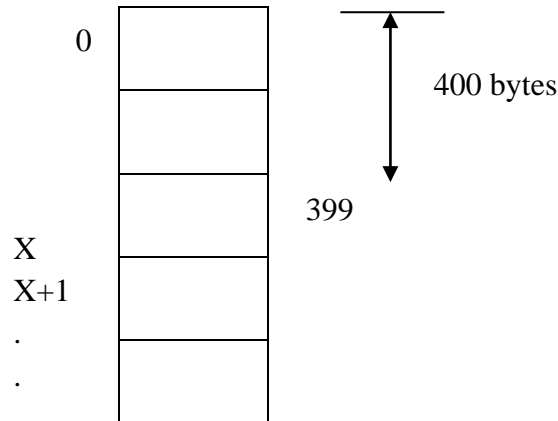
TYPES OF ASSEMBLER DIRECTIVES

1. RESERVE

This directive is used to allocate a block of memory. This block of memory is used only for data.

```
X    RESERVE    400
```

This directive reserves 400 bytes of memory whose symbolic name is X as shown in the fig.



2.EQU directive

This directive is used to assign numerical values to symbolic names during the execution of the assembly language program.

The following code describes the use of EQU directive.

```
X      EQU      32
MOVE   #23,     R0
MOVE   X,       R1
ADD    R0,      R1
```

3. DATA WORD

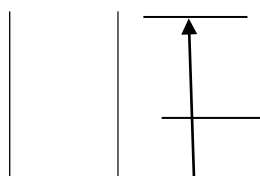
```
Y      23
Y+1
Y+2    45
Y+3
```

4 bytes

This directive is used to allocate 4 bytes of memory.

```
Y      DATAWORD 23456789
```

The memory representation of this directive is as shown in the fig.



67
89

4. ORIGIN DIRECTIVE

This directive converts source program into object program. The object program is loaded into memory for execution by means of loader. The Origin directive is used to assign the successive addresses for sequence of instructions or operands.

5. END DIRECTIVE

This Directive is used to terminate the Assembly level language program. The Assembler ignores the execution of instructions after the execution of this directive.

9.2 NUMBER REPRESENTATION

There are 3 types of numbers namely

- Decimal Numbers (0 to 9)
- Binary Numbers (0 or 1)
- Hexadecimal Numbers (0 to 9 and A to F)

The 32 bit processor is capable of performing ALU operations on three types of operands namely decimal, binary and hexadecimal

The representation of all these operands is as follows

For decimal `MOVE #23, R0`

For binary `MOVE %# 01100011, R2`

For hexadecimal `MOVE #$24, R4`

10. BASIC INPUT AND OUTPUT OPERATIONS

The simple arrangement of connecting i/p and o/p devices into the processor is as shown in the fig

The Processor performs two operations with respect to i/o device namely

i) Input operation and

ii) Output operation

i) Input operation: It is the process of reading the data or instructions from the input device.

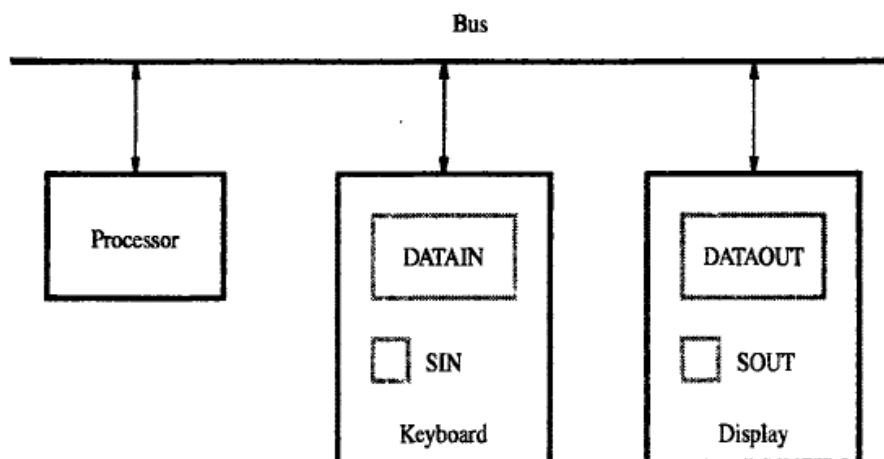
The I/O subsystem consists of block of instructions to perform i/p operation.

Output operation: It is the process of writing the data or instructions into the output device.

The I/O subsystem consists of block of instructions to perform o/p operation

Consider a problem of transferring 1 byte of data from i/p device to o/p device. The i/p device transfers few characters/sec. The data transfer rate of i/p device is expressed in terms of

few characters/sec. Similarly o/p device transfers thousands of characters to o/p device



for display. The processor is capable of executing millions of instructions per second.

From the above analysis it is clear that the processing and transfer speed varies in different devices. So the devices must be synchronized.

To provide the synchronization between Processor, i/p device and o/p device it is necessary to follow the several steps are as follows

Steps to provide synchronization between Processor and i/p device

1. When a character is pressed on the keyboard, the ASCII value of a character is stored in DATAIN register and hence SIN flag is set to 1.
2. When SIN = 1, the processor reads the ASCII value of a character from DATAIN register into the Processor register.
3. After reading, the SIN flag is reset to 0.

READWAIT	if SIN = 0	
	Branch to READWAIT	//No data to read
	Input data from DATAIN to R1	//Data is read

Steps to provide synchronization between Processor and O/p device

1. When the o/p device is ready to display the character ,the Processor transfers the character code from the processor register into the DATAOUT register.
2. The SOUT flag is set to 0 when DATAOUT register holds the character code.
3. The SOUT flag is cleared when the character code is transferred to o/p device

WRITEWAIT	if SOUT = 0	
	Branch to WRITEWAIT	//No data to read
	Output data from R1 to DATAIN	//Data is read

The i/p operation can be implemented as follows

Let R₀ be the Processor register and DATAIN be the internal register of the i/p device.

MOVE DATAIN, R0

The o/p operation can be implemented as follows

Let R₀ be the Processor register and DATAOUT be the internal register of the O/p device.

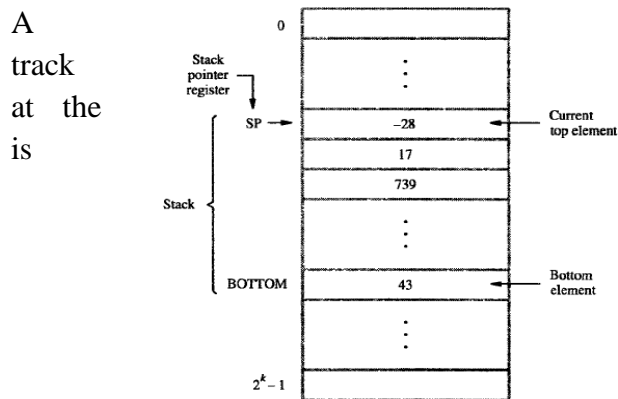
MOVE R₀, DATAOUT

11. STACKS AND QUEUES

STACK - A stack is a Data Structure, in which the accessing is restricted at only one end of the stack. It is similar to a bottle, in which elements can be added and removed from the same end. The end of the stack, from which elements can be added or removed is called the top of the stack and the other end is called the bottom of the stack.

It works on the principle of LIFO (Last In First Out), the last item placed on the stack is the first to be removed. The term ‘push’ and ‘pop’ are used to describe the placing a new item on stack and removing the top item from the stack.

Assume that the first element is placed in the location BOTTOM, and when new elements are pushed to the stack, they are placed in successive lower addresses.

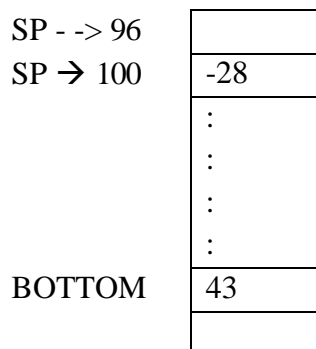


processor register is used to keep of the address of the element that is top of stack at any time. This register called Stack Pointer (**SP**).

In the above figure, the SP pointer is currently pointing to the topmost value -28. To add a new element, the SP will decrement its value by 1 address, so as to point at next location and add the new value.

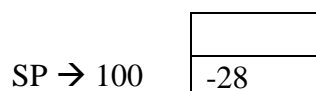
PUSH Operation – Subtract #4, SP
MOV NEWITEM, (SP)

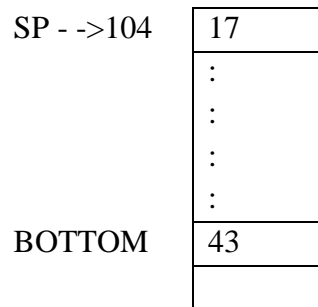
The subtract instruction subtracts the SP value by 4, now SP points to the next lower address. The MOV instruction moves the new element to the address location stored in SP.



POP Operation - MOV (SP), ITEM
ADD #4, SP

The MOV instruction moves the element at the location pointed by SP to ITEM and the SP pointer is moved to the next higher address, so that it points to the new top element.





QUEUE – A queue is a Data Structure that works on the principal of FIFO (First In First Out)ie., data that are stored first are retrieved first on FIFO basis.

The elements are added at one end (IN) and retrieved from other end (OUT). In stack, one end is fixed where as in queue both ends are pointed by pointers and both end changes its location. One end is used to add items and other end is used to delete items.

12. SUBROUTINE

Subfunctions in a program necessary to perform a particular subtask is called a subroutine.

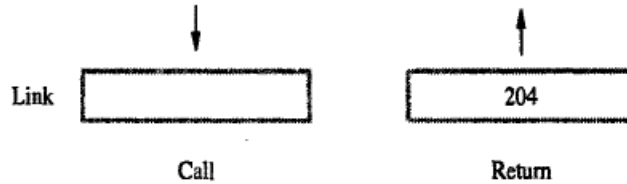
Eg: Subroutine to sort a list of numbers,

Subroutine to add the given numbers etc.

In a program, the subroutines can be called from different locations and different functions. When a program branches to a subroutine, then it is calling the subroutine. The instruction that perform this branch operation is called **call instruction**.

Whenever the subroutine is called, the execution starts from the starting address of subroutine. After its execution, the execution of calling function is resumed from the location where it called the subroutine. Hence the content of PC is stored before moving to the subroutine.

The way in which a computer calls and returns from a subroutine is called subroutine linkage method. The return address is stored in link register. After the execution of subroutine, the return instruction returns to the calling program by using the link register.



Subroutine linkage using a link register.

The Call instruction is a special

branch instruction –

- It stores the content of PC in link register
- Stores the specified subroutine address in PC and branch to that address.

The Return instruction of subroutine is a special branch instruction –

- It branches to the address contained in link register.

13. ADDITIONAL INSTRUCTIONS

There are 3 types

- a. Logical instructions – NOT, AND , OR
- b. Shift instructions – Logical Shift Left, Logical Shift Right, Arithmetic Shift right
- c. Rotate instructions – Rotate Left with carry, Rotate Left without carry,
Rotate Right with carry, Rotate Right without carry.

a. Logical instructions

The processors are designed to perform logical operations such as AND,OR and NOT operations.

i) NOT instruction

Format: opcode destination

The opcode specifies operation to be performed and destination specifies the operand. The operand can be register operand or memory operand.

Ex 1: NOT R0

It performs the function of complementation. It is the process of converting binary bit 0 into binary bit 1 and vice versa.

The illustration of this instruction is as follows.

Before instruction execution

R0 = 10101100

After instruction execution

R0 = 01010011.

ii) AND instruction

It performs the function of logical AND operation.

Format: opcode source, destination

Ex:1 AND R3, R0

This instruction logically ANDs the contents of R3 with the contents of R0 and result is stored in R0 register.

iii) OR instruction

It performs the function of logical OR operation.

Format: opcode source, destination

Ex: 1 OR R3, R0

This instruction logically OR^s the contents of R3 with the contents of R0 and result is stored in R0 register.

b. Shift instructions

The shift instructions are designed to shift the contents of processor register or memory location to left or right according to the number of bits specified in the count.

There are 2 types of shift instructions.

1. Logical Shift Left.
2. Logical Shift Right.
3. Arithmetic Shift Right

1. Logical Shift Left

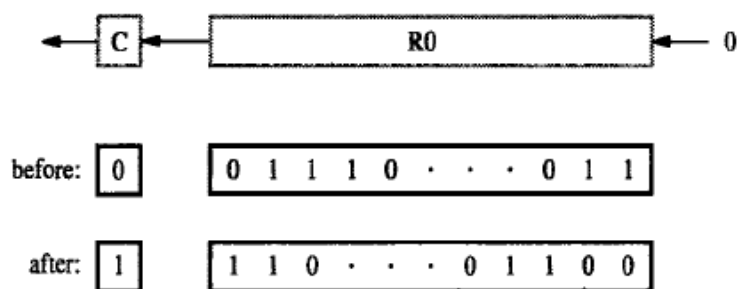
Format: opcode count, destination

The opcode indicates operation to be performed. The count can be either immediate operand or the contents of processor register. The destination can be either register operand or memory operand.

Ex: LshiftL #2, R0

This instruction shifts the contents of register R0 to left through carry by 2 bits. The count value directly specified in the instruction as an immediate operand.

The contents of R0 before and after the execution of this instruction are as shown in the fig. The **shifted positions are filled with zeros from right side** as shown in the fig.



2. Logical Shift Right

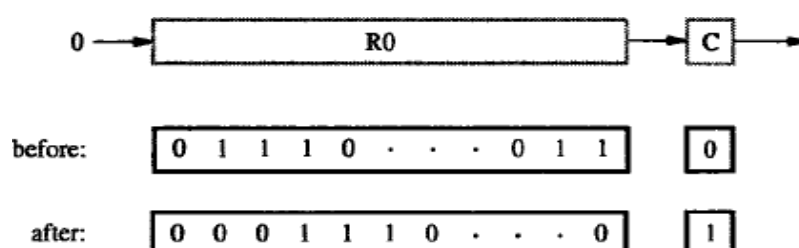
Format: opcode count, destination

The opcode indicates operation to be performed. The count can be either immediate operand or the contents of processor register. The destination can be either register operand or memory operand.

Ex: LshiftR #2, R0

This instruction shifts the contents of register R0 to right through carry by 2 bits. The count value directly specified in the instruction as an immediate operand.

The contents of R0 before and after the execution of this instruction are as shown in the fig. The **shifted positions are filled with zeros from left side** as shown in the fig.



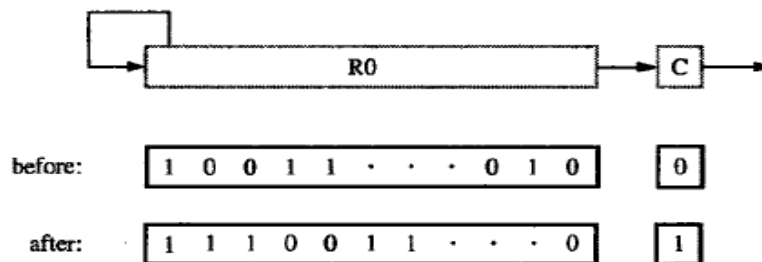
3.Arithmetic Shift Right

Format: opcode count, destination

The opcode indicates operation to be performed. The count can be either immediate operand or the contents of processor register. The destination can be either register operand or memory operand

Ex: AShiftR #2, R0

This instruction is designed **to preserve the sign bit**. This instruction shifts the contents of register or memory location to **right through carry**, by number of bits specified in the count. After each shift it **copies leftmost bit to Most Significant Bit**. The contents of R0 before and after the execution of this instruction are as shown in the fig.



c.Rotate instructions

The Rotate instructions are designed to rotate the contents of register or memory location to left or right according to the number of bits specified in the count.

There are 4 types of Rotate instructions.

1. Rotate left without carry
2. Rotate left with carry
3. Rotate right without carry
4. Rotate right with carry

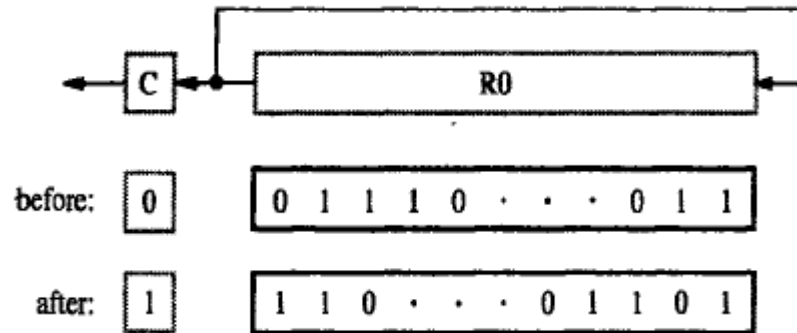
1. Rotate left without carry

Format: opcode count, destination

The opcode indicates operation to be performed. The count can be either immediate operand or the contents of processor register. The destination can be either register operand or memory operand.

Ex: RotateL #2, R0

This instruction rotates the contents of register R0 **to left without carry** by 2 bits as shown in the fig. The Most Significant Bits are transferred to Least Significant Bits as shown in the fig. The contents of register R0 before and after the execution of this instruction is as shown in the fig.



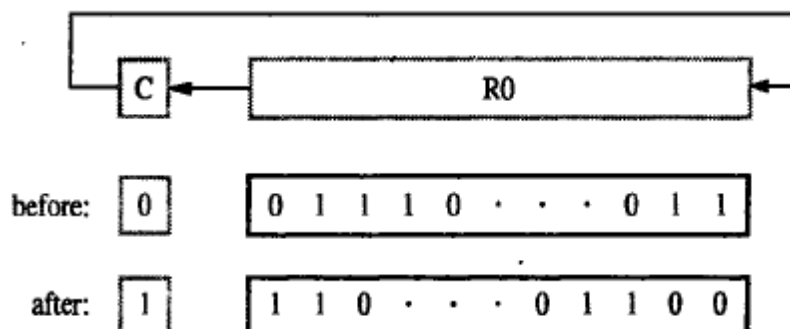
2. Rotate left with carry

Format: opcode count, destination

The opcode indicates operation to be performed. The count can be either immediate operand or the contents of processor register. The destination can be either register operand or memory operand.

Ex: RotateLC #2, R0

This instruction rotates the contents of register R0 **to left with carry** by 2 bits as shown in the fig. The Most Significant Bits are transferred to carry and then transferred to Least Significant Bits as shown in the fig. The contents of register R0 before and after the execution of this instruction is as shown in the fig.



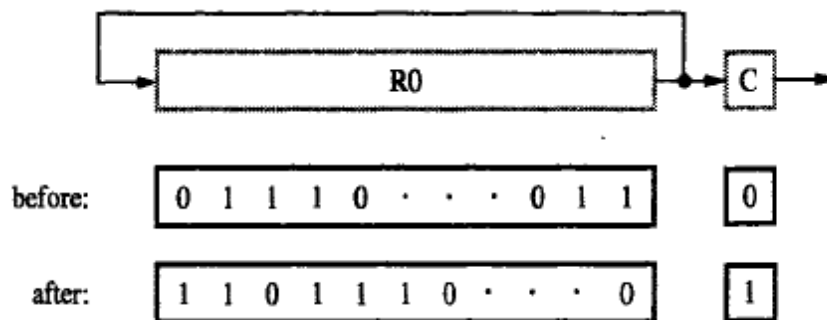
3. Rotate right without carry

Format: opcode count, destination

The opcode indicates operation to be performed. The count can be either immediate operand or the contents of processor register. The destination can be either register operand or memory operand.

Ex: RotateR #2, R0

This instruction rotates the contents of register R0 to right without carry by 2 bits as shown in the fig. **The Least Significant Bits are transferred to Most Significant Bits** are as shown in the fig. The contents of register R0 before and after the execution of this instruction is as shown in the fig.



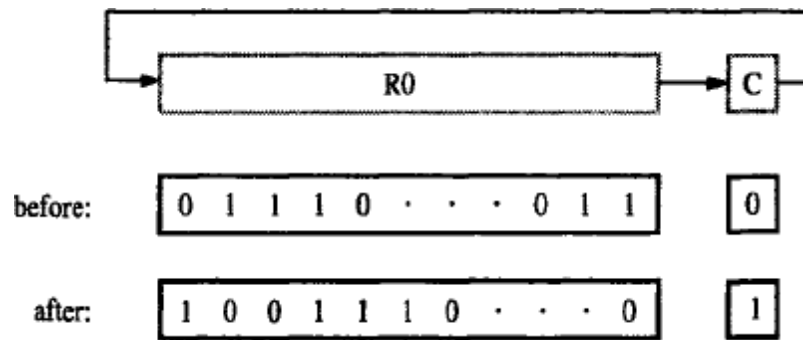
4. Rotate right with carry

Format: opcode count, destination

The opcode indicates operation to be performed. The count can be either immediate operand or the contents of processor register. The destination can be either register operand or memory operand.

Ex: RotateRC #2, R0

This instruction rotates the contents of register R0 to **right with carry** by 2 bits as shown in the fig. **The Least Significant Bits are transferred to carry and then transferred to Most Significant Bits** are as shown in the fig. The contents of register R0 before and after the execution of this instruction is as shown in the fig.



14. ENCODING OF MACHINE INSTRUCTIONS.

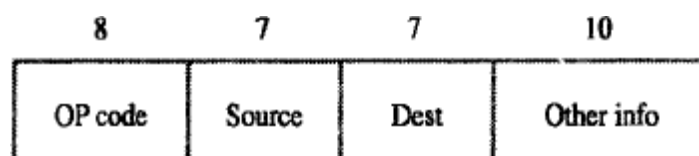
A list of instructions is called as program. To execute a program in processor instructions must be encoded into a compact binary form. Such encoded instructions are called as Machine instructions. The instructions that use symbolic names are called as “Assembly Language “.The Assembler converts Assembly Language instructions into Machine Language instructions.

Consider few instructions to perform several operations such as add, sub, multiply, shift, branch. These instructions may use operands of different size such as 32 bit , 8 bit or 16 bit. The type of operation to be performed and type of operand may be specified by encoded binary pattern and is called as “ opcode “.

There are 3 types of instruction formats.

- One word instruction format.
 - Two word instruction format.
 - Three operand instruction format
- a) **One word instruction format**

The one word instruction format is as shown in the fig



Ex:1 ADD R0, R1

This instruction is an example for Register Addressing mode.

This instruction transfers 32 bit data from R0 to R1. Where R0 is the Source Register and R1 is the Destination Register. The encoding of this instruction according to the above instruction format is as follows.

8 bits → opcode
 4 bits → Source Register.
 3 bits → Source Register addressing mode.
 4 bits → Destination Register.
 3 bits → Destination Register addressing mode.
 10 bits → Index value or immediate operand.

Ex:2 MOVE 24(R0), R5

This instruction is an example for Register Indirect Addressing mode.

This instruction transfers 32 bit data from Memory to Register. The effective address of the operand is stored in the Register R0.

The EA of an operand is given by

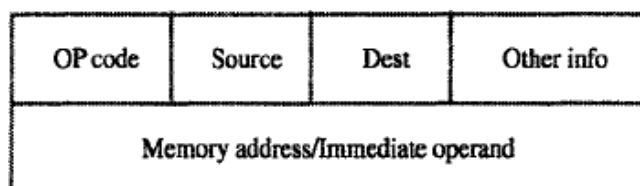
$$EA = [R0] + 24.$$

The encoding of this instruction according to the above instruction format is as follows.

8 bits → opcode
 4 bits → Source Register.
 3 bits → Source Register addressing mode.
 4 bits → Destination Register.
 3 bits → Destination Register addressing mode.
 10 bits → Index value or immediate operand.

b) Two word instruction format

The two instruction format is as shown in the fig.



Ex :1 MOVE R2, LOCA

This instruction is an example for Direct Addressing Mode.

This instruction transfers 32 bit data from Register R2 to Memory location whose symbolic name is given by LOCA.

The encoding of this instruction according to above instruction format is as follows

- This instruction format consists of 2 words.
- The 1st word is used to specify the opcode, Source register, Addressing Mode for Source, Destination Register, Addressing Mode for Destination and index value or immediate operand as follows

8 bits → opcode

4 bits → Source Register.

3 bits → Source Register addressing mode.

4 bits → Destination Memory location.

3 bits → Destination Memory location addressing mode.

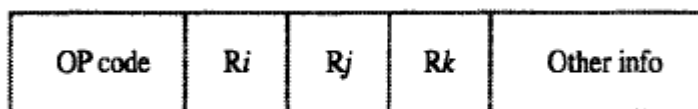
10 bits → Index value or immediate operand.

- The 2nd word is used to specify the 32 bit Memory address or 32 bit operand.

c) Three operand instruction format

The three operand instruction format is as shown in the fig.

This instruction is an example for Three operand Register Addressing Mode.



Ex: ADD R1, R2, R3

This instruction adds the contents of Register R1 with the contents of R2 and result is placed in R3 Register.

The mathematical representation of this is as follows

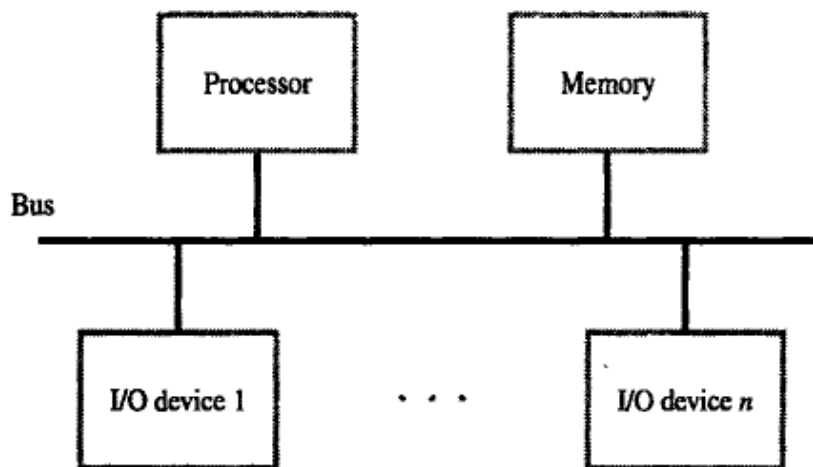
$$[R1] + [R2] \rightarrow [R3].$$

MODULE 2

INPUT/OUTPUT ORGANIZATION

ACCESSING I/O-DEVICES

A **single bus-structure** can be used for connecting I/O-devices to a computer. The simple arrangement of connecting set of I/O devices to memory and processor by means of system bus is as shown in the figure. Such a arrangement is called as Single Bus Organization.



- The single bus organization consists of
 - Memory
 - Processor
 - System bus
 - I/O device
- The system bus consists of 3 types of buses:
 - Address bus (Unidirectional)
 - Data bus (Bidirectional)

- Control bus (Bidirectional)
- The system bus enables all the devices connected to it to involve in the data transfer operation.
- The system bus establishes data communication between I/O device and processor.
- Each I/O device is assigned a unique set of address.
- When processor places an address on address-lines, the intended-device responds to the command.
- The processor requests either a read or write-operation.
- The requested-data are transferred over the data-lines

Steps for input operation:

- The address bus of system bus holds the address of the input device.
- The control unit of CPU generates \overline{IOR} Control signal.
- When this control signal is activated the processor reads the data from the input device (DATAIN) into the CPU register.

Steps for output operation:

- The address bus of system bus holds the address of the output device.
- The control unit of CPU generates \overline{IOWR} control signal.
- When this control signal is enabled CPU transfers the data from processor register to output device(DATAOUT)

There are 2 schemes available to connect I/O devices to CPU

1.Memory mapped I/O:

- In this technique both memory and I/O devices can share the common memory to store the instruction as well as the operands.
- Memory related instructions are used for data transfer between I/O and processor.

- In case of memory mapped I/O input operation can be implemented as,

```

MOVE  DATAIN ,    R0
      ↓           ↓
      Source      destination
  
```

This instruction sends the contents of location DATAIN to register R0.

- Similarly output can be implemented as,

```

MOVE  R0, DATAOUT
      ↓           ↓
      Source      destination
  
```

2. I/O Mapped I/O:

- In this technique CPU separates address space for memory and I/O devices.
- Hence two sets of instruction are used for data transfer.
- One set for memory operations and another set for I/O operations.
- The I/O operation can be implemented as,

```
IN AL, DX
```

This instruction reads one byte of data from the I/P register whose address is store in DX register into AL register.

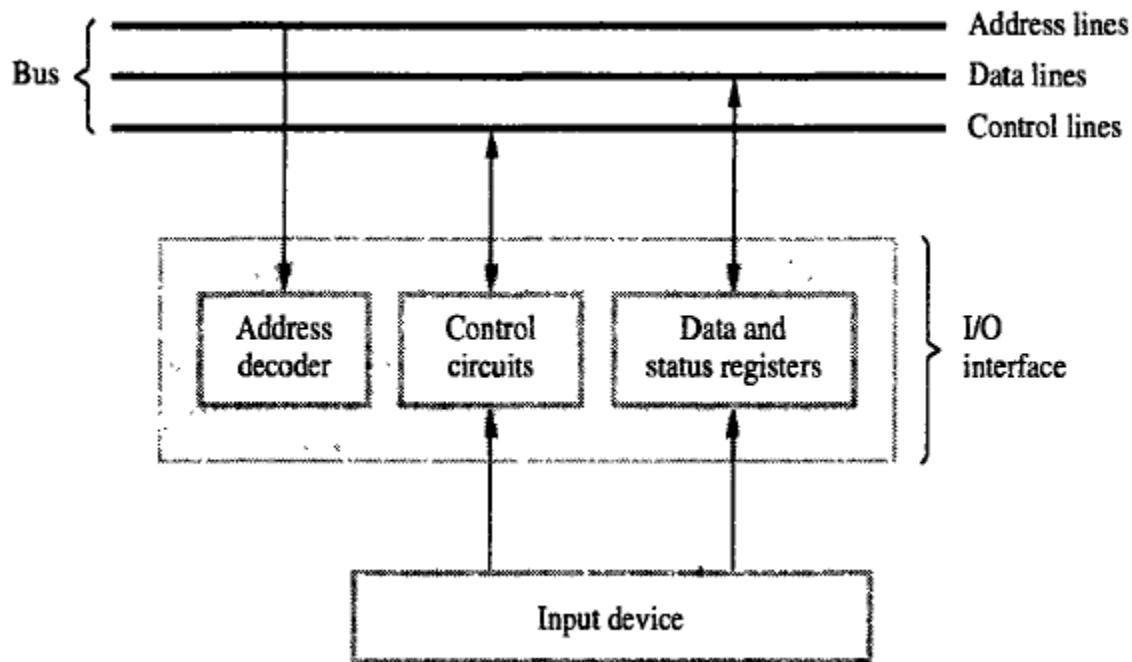
- The O/P operation can be implemented as,
OUT DX, AL

This instruction transfer the contents of AL register into the O/P register (DATA OUT) whose address is stored in DX register.

Memory Mapped I/O	I/O Mapped I/O
Memory & I/O share the entire address range of processor	Processor provides separate address range for memory & I/O
Processor provides more address lines for accessing memory	Less address lines for accessing I/O
More Decoding is required	Less decoding is required
Memory control signals used to control Read & Write I/O operations	I/O control signals are used to control Read & Write I/O operations

I/O INTERFACE FOR AN INPUT DEVICE

The hardware arrangement of connecting I/P device to the system bus is as shown in the fig.



This hardware arrangement is called as I/O interface. The I/O interface consists of 3 functional devices namely:

1) **Address Decoder:**

- Its function is to decode the address in-order to recognize the input device whose address is available on the unidirectional address bus.
- The unidirectional address bus of system bus is connected to input of the address decoder as shown in figure

2) **Control Circuit:**

- The control bus of system bus is connected to control circuit as shown in the fig.
- It controls the read write operations with respect to I/O device.

3) **Status & Data register:**

- It specifies type of operation (either read or write operation) to be performed on I/O device.

4) **Data Register:**

- It stores the data to be read from input device to or it holds the data to be written into output device. There are 2 types:

DATAIN - Input-buffer associated with keyboard.

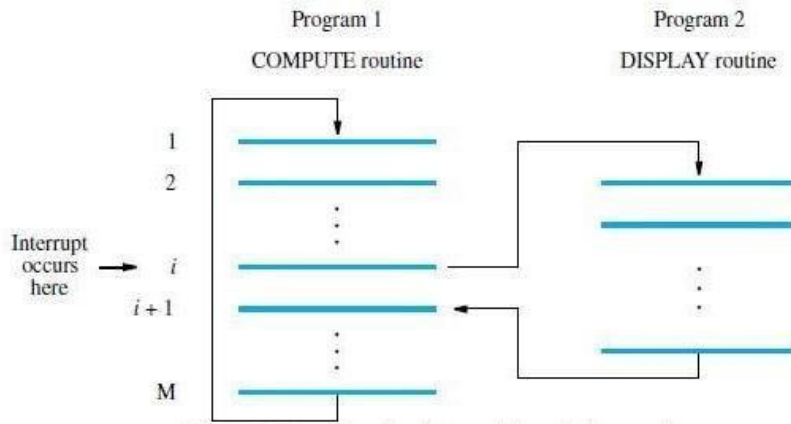
DATAOUT -Output data buffer of display/printer.

Program Controlled I/O

- To explain the concept of program controlled I/O consider two operations namely input operation and output operation.
 - a) Input operation:
 - It is the process of transferring ASCII value of a character from DATA IN register to CPU register.
 - b) Output operation:
 - It is the process of transferring ASCII value of a character from CPU register to DATA OUT register.
- Design 2 sets of instruction for input and output operations respectively,
- These 2 sets of instructions are stored in memory of the CPU. Now allow the CPU to execute this program in-order to control the input and output operation.
- “It is the process of controlling the input and output operations by executing 2 sets of instruction, one set for input operation and the next set for output operation.”

Interrupt

- It is an event which suspends the execution of one program and begins the execution of another program.
- The arrival of interrupt causes the processor to transfer the execution control from main program to sub program.
- To explain the concept of interrupt, consider the following two programs namely:



The following steps takes place when the interrupt related instruction is executed:

- It suspends the execution of current instruction i.
- Transfer the execution control to sub program from main program.
- Increments the content of PC by 4 memory location.
- It decrements SP by 4 memory locations.
- Pushes the contents of PC into the stack segment memory whose address is stored in SP.
- It loads PC with the address of the first instruction of the sub program.

The following steps takes place when return instruction is executed

- It transfers the execution control from sub program to main program.
- It retrieves the content of stack memory location whose address is stored in SP into the PC.
- After retrieving the return address from stack memory location into the PC it increments the Content of SP by 4 memory location.

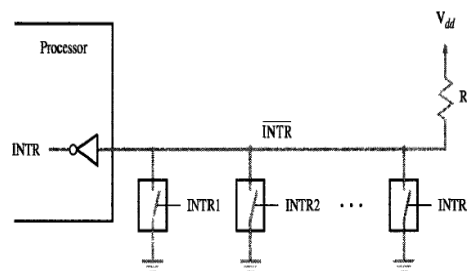
Interrupt Latency is a delay between

- time an interrupt-request is received and
- start of the execution of the ISR.

- Generally, the long interrupt latency in unacceptable.

INTERRUPT HARDWARE

- The external device (I/O device) request the processor by activating one bus line and this bus line is called as interrupt request line.
- The one end of this interrupt request line is connected to input power supply by means of pull up register is as shown in the fig.
- The another end of interrupt request line is connected to INTR (Interrupt request) signal of processor as shown in the fig.



- The I/O device is connected to interrupt request line by means of switch as shown in the fig.
- When all the switches are open the voltage drop on interrupt request line is equal to the V_{DD} .
- This state is called as in-active state of the interrupt request line.
- The I/O device interrupts the processor by closing its switch.
- When switch is closed the voltage drop on the interrupt request line is found to be zero.

Therefore $INTR=0$ and $INTR=1$.

- The signal on the interrupt request line is logical OR of requests from the several I/O devices.

Therefore, $INTR = \underline{INTR1} + \underline{INTR2} + \dots + \underline{INTRn}$

ENABLING AND DISABLING THE INTERRUPTS

- All computers fundamentally should be able to enable and disable interruptions as desired.
- The problem of infinite loop occurs due to successive interruptions of active INTR signals.
- There are 3 mechanisms to solve problem of infinite loop:

- 1) Processor should ignore the interrupts until execution of first instruction of the ISR.
- 2) Processor should automatically disable interrupts before starting the execution of the ISR.
- 3) Processor has a special INTR line for which the interrupt-handling circuit.

Interrupt-circuit responds only to leading edge of signal. Such line is called edge-triggered.

- Sequence of events involved in handling an interrupt-request:

- 1) The device raises an interrupt-request.
- 2) The processor interrupts the program currently being executed.
- 3) Interrupts are disabled by changing the control bits in the processor status register (PS).
- 4) The device is informed that its request has been recognized.

In response, the device deactivates the interrupt-request signal.

- 5) The action requested by the interrupt is performed by the interrupt-service routine.
- 6) Interrupts are enabled and execution of the interrupted program is resumed.

HANDLING MULTIPLE DEVICES

While handling multiple devices, the issues concerned are:

- How can the processor recognize the device requesting an interrupt?
- How can the processor obtain the starting address of the appropriate ISR?
- Should a device be allowed to interrupt the processor while another interrupt is being serviced?
- How should 2 or more simultaneous interrupt-requests be handled?

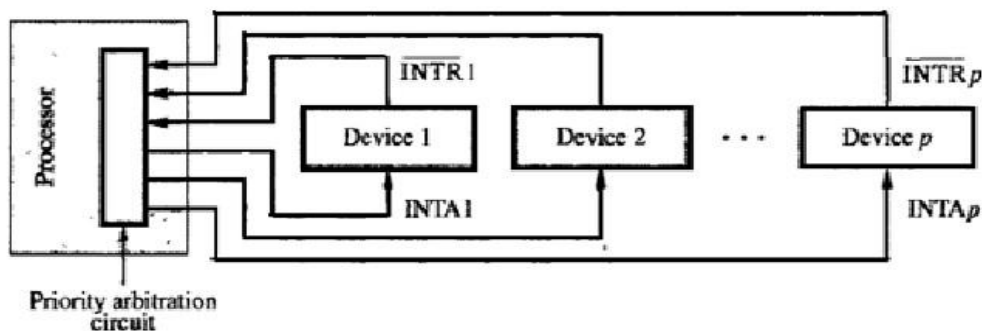
VECTORED INTERRUPT

- A device requesting an interrupt identifies itself by sending a special-code to processor over bus.
- Then, the processor starts executing the ISR.
- The special-code indicates starting-address of ISR.
- The special-code length ranges from 4 to 8 bits.

- The location pointed to by the interrupting-device is used to store the starting address to ISR.
- The starting address to ISR is called the **interrupt vector**.
- Processor
 - loads interrupt-vector into PC &
 - executes appropriate ISR.
- When processor is ready to receive interrupt-vector code, it activates INTA line.
- Then, I/O-device responds by sending its interrupt-vector code & turning off the INTR signal.
- The interrupt vector also includes a new value for the Processor Status Register

INTERRUPT NESTING

- A multiple-priority scheme is implemented by using separate INTR & INTA lines for each device
- Each INTR line is assigned a different priority-level as shown in Figure.



- Priority-level of processor is the priority of program that is currently being executed.
- Processor accepts interrupts only from devices that have higher-priority than its own.
- At the time of execution of ISR for some device, priority of processor is raised to that of the device.
- Thus, interrupts from devices at the same level of priority or lower are disabled.

Privileged Instruction

- Processor's priority is encoded in a few bits of PS word. (PS = Processor-Status).
- Encoded-bits can be changed by **Privileged Instructions** that write into PS.
- Privileged-instructions can be executed only while processor is running in **Supervisor Mode**.

- Processor is in supervisor-mode only when executing operating-system routines.

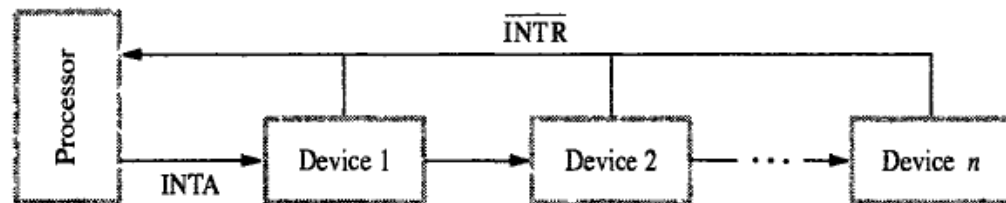
Privileged Exception

- User program cannot
 - accidentally or intentionally change the priority of the processor &
 - disrupt the system-operation.
- An attempt to execute a privileged-instruction while in user-mode leads to a **Privileged Exception**.

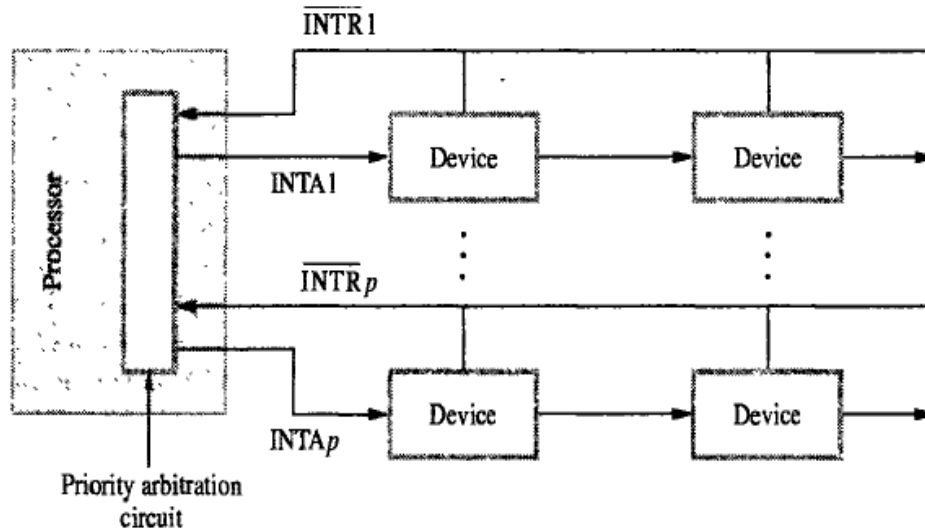
SIMULTANEOUS REQUESTS

DAISY CHAIN

- The daisy chain with multiple priority levels is as shown in the figure.



- The interrupt request line INTR is common to all devices as shown in the fig.
- The interrupt acknowledge line is connected in a daisy fashion as shown in the figure.
- This signal propagates serially from one device to another device.
- The several devices raise an interrupt by activating INTR signal. In response to the signal, processor transfers its device by activating INTA signal.
- This signal is received by device 1. The device-1 blocks the propagation of INTA signal to device-2, when it needs processor service.
- The device-1 transfers the INTA signal to next device when it does not require the processor service.
- In daisy chain arrangement device-1 has the highest priority.
- **Advantage:** It requires fewer wires than the individual connections.

ARRANGEMENT OF PRIORITY GROUPS

- In this technique, devices are organized in a group and each group is connected to the processor at a different priority level.
- Within a group devices are connected in a daisy chain fashion as shown in the figure.

EXCEPTIONS

- **Exception** refers to any event that causes an interruption.

For ex: I/O interrupts.

- Types of Exception

1. Recovery from Errors

- These are techniques to ensure that all hardware components are operating properly.
For ex: Many computers include an ECC in memory which allows detection of errors in stored-data. (ECC = Error Checking Code, ESR= Exception Service Routine).
- If an error occurs, control-hardware
 - detects the errors &
 - informs processor by raising an interrupt.

- When exception processing is initiated (as a result of errors), processor.
 - suspends program being executed &
 - starts an ESR. This routine takes appropriate action to recover from the error.

2. Debugging

Debugger is used to find errors in a program and uses exceptions to provide 2 important facilities:

i) Trace & ii) Breakpoints

i) Trace

- When a processor is operating in trace-mode, an exception occurs after execution of every instruction (using debugging-program as ESR).
- Debugging-program enables user to examine contents of registers, memory-locations and so on.
- On return from debugging-program, next instruction in program being debugged is executed, then debugging-program is activated again.
- The trace exception is disabled during the execution of the debugging-program.

ii) Breakpoints

- Here, the program being debugged is interrupted only at specific points selected by user.
- An instruction called Trap (or Software interrupt) is usually provided for this purpose.
- When program is executed & reaches breakpoint, the user can examine memory & register contents.

3. Privilege Exception

- To protect OS from being corrupted by user-programs, **Privileged Instructions** are executed only while processor is in supervisor-mode.
- For e.g. When processor runs in user-mode, it will not execute instruction that change priority of processor.
- An attempt to execute privileged-instruction will produce a **Privilege Exception**.
- As a result, processor switches to supervisor-mode & begins to execute an appropriate routine in OS.

Direct Memory Address (DMA)

- It is the process of transferring the block of data at high speed in between main memory and external device (I/O devices) without continuous intervention of CPU is called as DMA.
- The DMA operation is performed by one control circuit and is part of the I/O interface.

- This control circuit is called DMA controller. Hence DMA transfer operation is performed by DMA controller.
- To initiate Directed data transfer between main memory and external devices DMA controller needs parameters from the CPU.
- These 3 Parameters are:

1) Starting address of the memory block.

2) No of words to be transferred.

3) Type of operation (Read or Write).

After receiving these 3 parameters from CPU, DMA controller establishes directed data transfer operation between main memory and external devices without the involvement of CPU.

- **Register of DMA Controller:**

It consists of 3 type of register:

Starting address register:

The format of starting address register is as shown in the fig. It is used to store the starting address of the memory block.



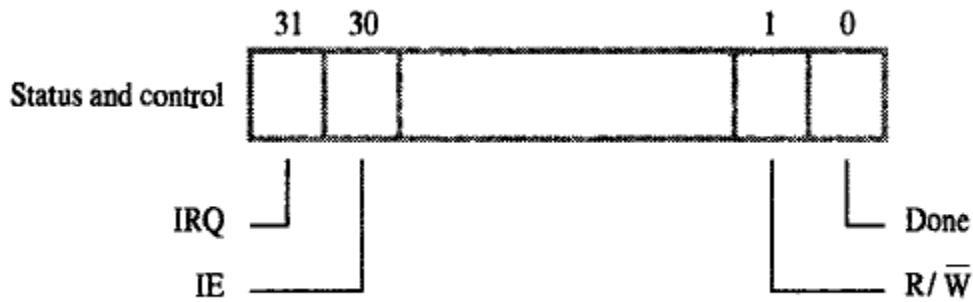
Word-Count register:

The format of word count register is as shown in fig. It is used to store the no of words to be transferred from main memory to external devices and vice versa.



Status and Controller register:

The format of status and controller register is as shown in fig.



a) DONE bit:

- The DMA controller sets this bit to 1 when it completes the direct data transfer between main memory and external devices.
- This information is informed to CPU by means of DONE bit.

b) R/W (Read or Write):

- This bit is used to differentiate between memory read or memory write operation.
- The $R/\bar{W} = 1$ for read operation and $= 0$ for write operation.
- When this bit is set to 1, DMA controller transfers the one block of data from external device to main memory.
- When this bit is set to 0, DMA controller transfers the one block of data from main memory to external device.

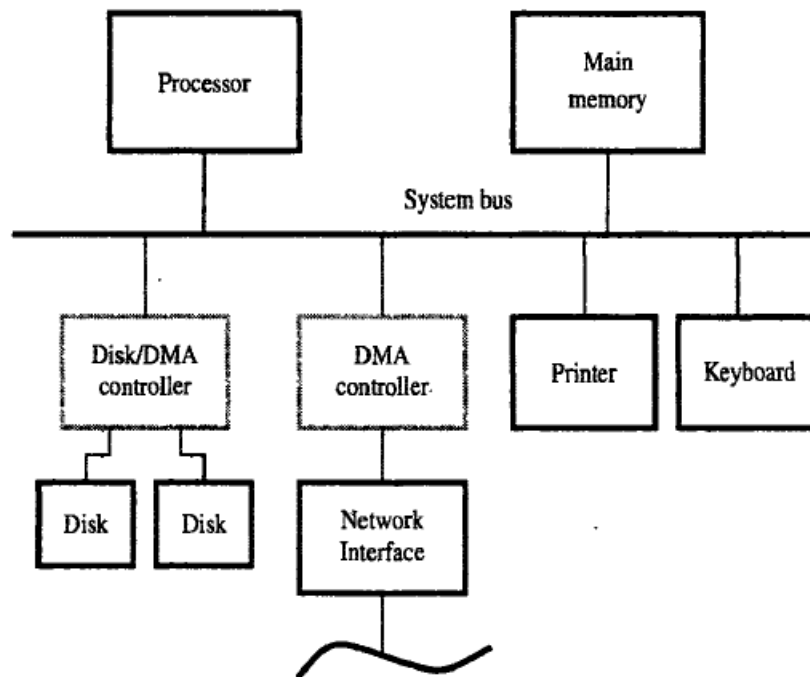
c) IE (Interrupt enable) bit:

- The DMA controller enables the interrupt enable bit after the completion of DMA operation

d) Interrupt request (IRQ):

- The DMA controller requests the CPU to transfer new block of data from source to destination by activating this bit.

The computer with DMA controller is as shown in the fig.:



- The DMA controller connects two external devices namely disk 1 and disk 2 to system bus as shown in the above fig.
- The DMA controller also interconnects high speed network devices to system bus as shown in the above fig.
- Let us consider direct data transfer operation by means of DMA controller without the involvement of CPU in between main memory and disk 1 as indicated by dotted lines (in the fig.).
- To establish direct data transfer operation between main memory and disk 1. DMA controller request the processor to obtain 3 parameters namely:

- 1) Starting address of the memory block.
- 2) No of words to be transferred.
- 3) Type of operation (Read or Write).

- After receiving these 3 parameters from processor, DMA controller directly transfers block of data main memory and external devices (disk 1).
 - This information is informed to CPU by setting respective bits in the status and controller register of DMA controller.
- These are 2 types of request with respect to system bus
- 1). CPU request.

2). DMA request.

Highest priority will be given to DMA request.

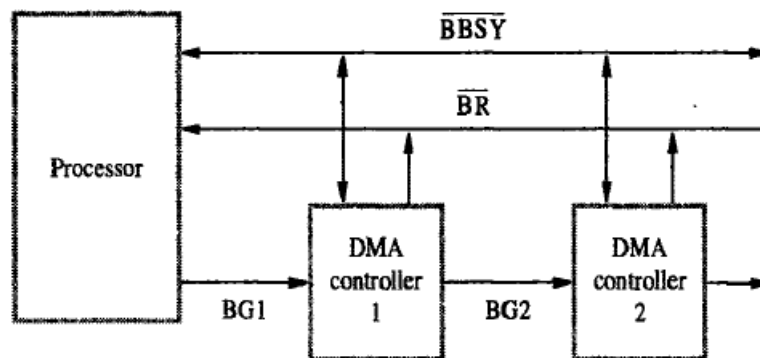
- Actually the CPU generates memory cycles to perform read and write operations. The DMA controller steals memory cycles from the CPU to perform read and write operations. This approach is called as “**Cycle stealing**”.
- An exclusive option will be given for DMA controller to transfer block of data from external devices to main memory and from main memory to external devices. This technique is called as “**Burst mode of operation.**”

BUS ARBITRATION

- Any device which initiates data transfer operation on bus at any instant of time is called as Bus-Master.
- When the bus mastership is transferred from one device to another device, the next device is ready to obtain the bus mastership.
- The bus-mastership is transferred from one device to another device based on the principle of priority system. There are two types of bus-arbitration technique:

a)Centralized bus arbitration:

In this technique CPU acts as a bus-master or any control unit connected to bus can be acts as a bus master.



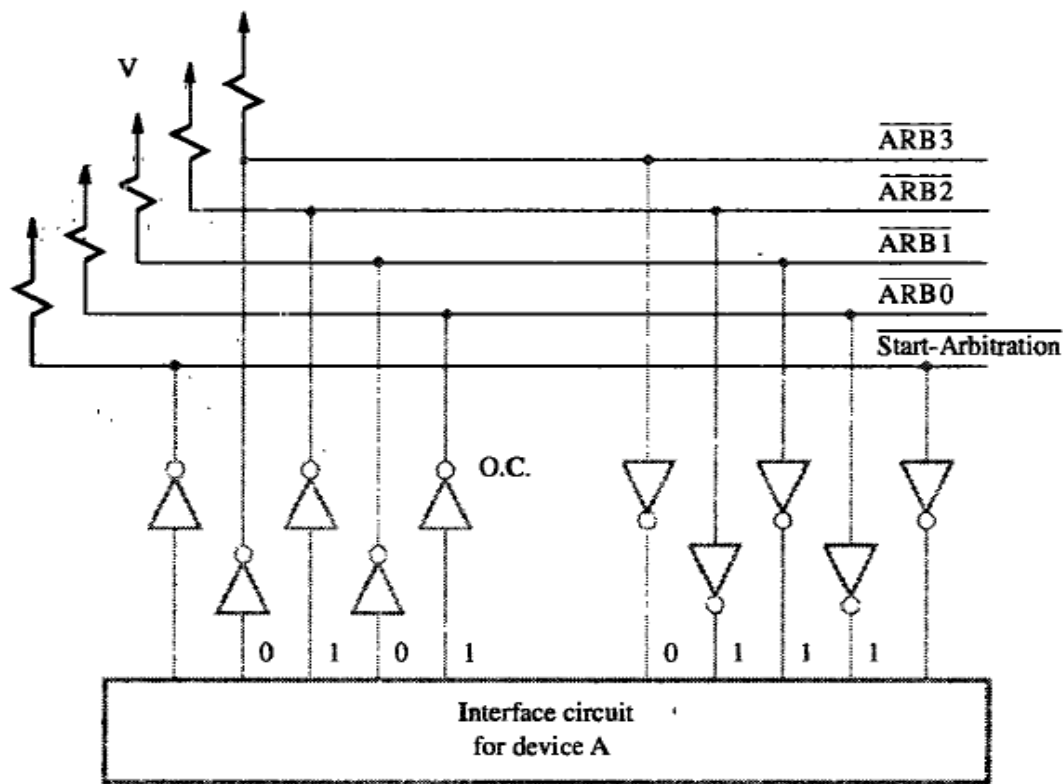
The schematic diagram of centralized bus arbitration is as shown in the fig.:

The following steps are necessary to transfer the bus mastership from CPU to one of the DMA controller:

- The DMA controller request the processor to obtain the bus mastership by activating \overline{BR} (Bus request) signal
- In response to this signal the CPU transfers the bus mastership to requested devices DMA controller1 in the form of BG (Bus grant).
- When the bus mastership is obtained from CPU the DMA controller1 blocks the propagation of bus grant signal from one device to another device.
- The \overline{BG} signal is connected to DMA controller2 from DMA controller1 in as daisy fashion style is as shown in the figure.
- When the DMA controller1 transfers the bus mastership to DMA controller2 by unblocking bus grant signal.
- When the DMA controller1 receives the bus grant signal it enables BBSY signal. When BBSY signal is set to 1 the set of devices connected to system bus doesn't have any rights to obtain the bus mastership from the CPU.

b)Distributed bus arbitration:

- In this technique 2 or more devices trying to access system bus at the same time may participate in bus arbitration process.
- The schematic diagram of distributed bus arbitration is as shown in the figure:



- The external device requests the processor to obtain bus mastership by enabling start arbitration signal.
- In this technique 4 bit code is assigned to each device to request the CPU in order to obtain bus mastership.
- Two or more devices request the bus by placing 4 bit code over the system bus.
- The signals on the bus interpret the 4 bit code and produces winner as a result from the CPU.
- When the input to the one driver = 1, and input to the another driver = 0, on the same bus line, this state is called as “Low level voltage state of bus”.
- Consider 2 devices namely A & B trying to access bus mastership at the same time. Let assigned code for devices A & B are 5 (0101) & 6 (0110) respectively.
- The device A sends the pattern (0101) and device B sends its pattern (0110) to master. The signals on the system bus interpret the 4 bit code for devices A & B produces device B as a winner.
- The device B can obtain the bus mastership to initiate direct data transfer between external devices and main memory.

BUSES

- The primary function of the bus is to inter connect 3 functional device namely CPU, memory and I/O devices.
- It is defined as set of similar wires used to establish data transfer operation between CPU and memory as well as CPU and I/O devices.
- It consists of 3 types:
 - a)Uni-directional address line.
 - b)Bi-directional data lines.
 - c)Control lines.
- The address bus of system bus is used to carry either the address of I/O device or the address of memory.
- The bi-directional data bus is used to carry data to be returned into I/O device or read from I/O device.
- The control bus of system bus is used to carry control signals as well as timing information. It is designed to carry control signals as RD, WR, and R/W.
The $R/W = 1$ for read operation.
 $= 0$ for write operation.

In addition to these 3 lines the bus consists of 2 special lines reserved for interrupts and bus arbitration.

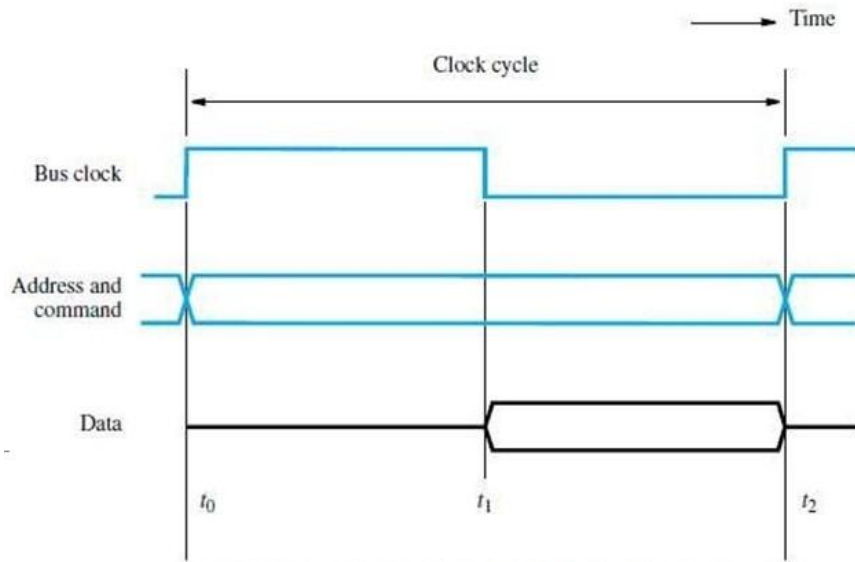
- There are 2 types of buses:

Synchronous bus

- In case of Synchronous bus all the devices derive the timing information from common bus line.
- A equally placed pulses on this common bus line are called as timing intervals or timing signals.
- Two or more timing intervals are called as 'bus cycle'.

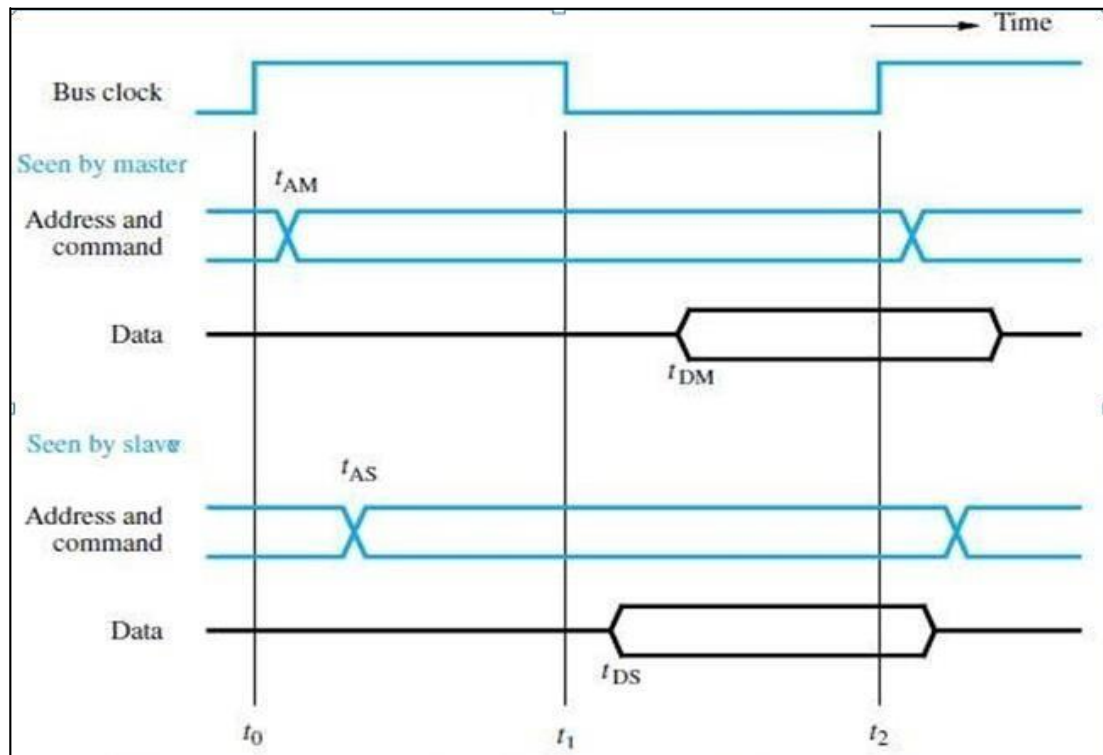
Timing Diagram for the Read-operation that shows a sequence of events during a read-operation

- At time t_0 , the master (processor)
 - places the device-address on address-lines &
 - sends an appropriate command on control-lines as shown in Figure.



- The command will
 - indicate an input operation &
 - specify the length of the operand to be read.
- Information travels over bus at a speed determined by physical & electrical characteristics.
- Clock pulse width($t_1 - t_0$) must be longer than max. propagation-delay b/w devices connected to bus.
- The clock pulse width should be long to allow the devices to decode the address & control signals.
- The slaves take no action or place any data on the bus before t_1 .
- Information on bus is unreliable during the period t_0 to t_1 because signals are changing state.
- Slave places requested input-data on data-lines at time t_1 .
- At end of clock cycle (at time t_2), master strobes (captures) data on data-lines into its input-buffer
- For data to be loaded correctly into a storage device, data must be available at input of that device for a period greater than setup-time of device

A Detailed Timing Diagram for the Read-operation

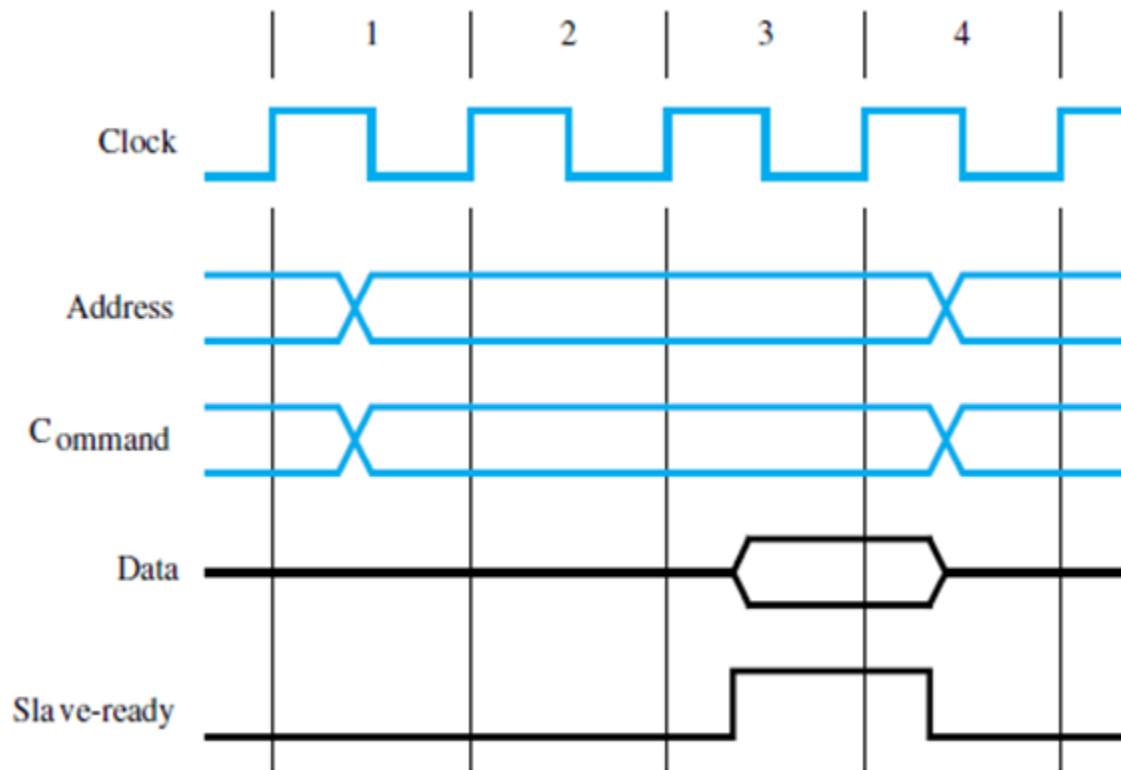


- The Figure shows two views of the signal.
- One view shows the signal seen by the master & the other is seen by the slave.
- Master sends the address & command signals on the rising edge at the beginning of clock period (t_0).
- These signals do not actually appear on the bus until t_{AM} .
- Sometimes later, at t_{AS} the signals reach the slave.
- The slave decodes the address.
- At t_1 , the slave sends the requested-data.
- At t_2 , the master loads the data into its input-buffer.
- Hence the period t_2 , t_{DM} is the setup time for the masters input-buffer.
- The data must be continued to be valid after t_2 , for a period equal to the hold time of that buffers.

Disadvantages

- The device does not respond.
- The error will not be detected.

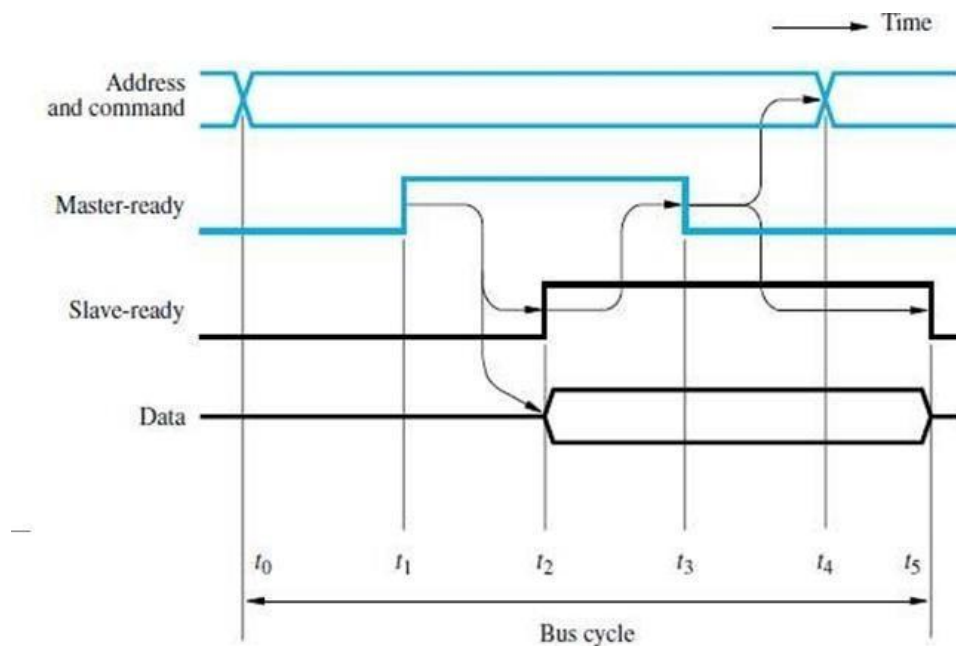
Multiple Cycle Transfer for Read-operation



- During, clock cycle-1, master sends address/command info to the bus requesting a “read” operation.
- The slave receives & decodes address/command information.
- At the active edge of the clock i.e. the beginning of clock cycle-2, it makes access to respond immediately.
- The data becomes ready & are placed in the bus at clock cycle-3.
- At the same time, the slave asserts a control signal called **slave-ready**.
- The master strobes the data to its input-buffer at the end of clock cycle-3.
- The bus transfer operation is now complete.
- And the master sends a new address to start a new transfer in clock cycle-4.
- The slave-ready signal is an acknowledgement from the slave to the master.

ASYNCHRONOUS BUS

- This method uses handshake-signals between master and slave for coordinating data-transfers.
- There are 2 control-lines:
 - 1) **Master-Ready (MR)** is used to indicate that master is ready for a transaction.
 - 2) **Slave-Ready (SR)** is used to indicate that slave is ready for a transaction.



The Read Operation proceeds as follows:

- At t_0 , master places address/command information on bus.
- At t_1 , master sets MR-signal to 1 to inform all devices that the address/command-info is ready.
 - MR-signal = 1 causes all devices on the bus to decode the address.
 - The delay $t_1 - t_0$ is intended to allow for any skew that may occurs on the bus.

- Skew occurs when 2 signals transmitted from 1 source arrive at destination at different time
- Therefore, the delay $t_1 - t_0$ should be larger than the maximum possible bus skew.
- At t_2 , slave
 - performs required input-operation &
 - sets SR signal to 1 to inform all devices that it is ready (Figure 7.6).
- At t_3 , SR signal arrives at master indicating that the input-data are available on bus.
- At t_4 , master removes address/command information from bus.
- At t_5 , when the device-interface receives the 1-to-0 transition of MR signal, it removes data and SR signal from the bus. This completes the input transfer.

INTERFACE CIRCUITS

- An **I/O Interface** consists of the circuitry required to connect an I/O device to a computer-bus.
- On one side of the interface, we have bus signals.
- On the other side, we have a data path with its associated controls to transfer data between the interface and the I/O device known as **port**.
- Two types are:
 1. **Parallel Port** transfers data in the form of a number of bits (8 or 16) simultaneously to or from the device.
 2. **Serial Port** transmits and receives data one bit at a time.
- Communication with the bus is the same for both formats.
- The conversion from the parallel to the serial format, and vice versa, takes place inside the interface- circuit.
- In parallel-port, the connection between the device and the computer uses
 - a multiple-pin connector and
 - a cable with as many wires.
- This arrangement is suitable for devices that are physically close to the computer.
- In serial port, it is much more convenient and cost-effective where longer cables are needed.

Functions of I/O Interface

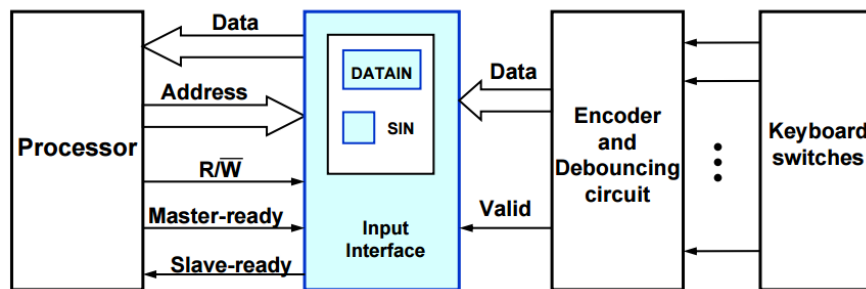
- 1) Provides a storage buffer for at least one word of data.
- 2) Contains status-flags that can be accessed by the processor to determine whether the buffer is full or empty.
- 3) Contains address-decoding circuitry to determine when it is being

addressed by the processor.

- 4) Generates the appropriate timing signals required by the bus control scheme.
- 5) Performs any format conversion that may be necessary to transfer data between the bus and the I/O device (such as parallel-serial conversion in the case of a serial port).

Parallel Port

The hardware components needed for connecting a keyboard to a processor.



- Keyboard is connected to a processor using a parallel-port.
- Processor uses
 - memory-mapped I/O and
 - asynchronous bus protocol.
- On the processor-side of the interface, we have:
 - Data-lines
 - Address-lines
 - Control or R/W line
 - Master-Ready signal and
 - Slave-Ready signal.
- The output of the encoder consists of
 - bits representing the encoded character and
 - one signal called **valid**, which indicates the key is pressed.

The information is sent to the interface-circuits

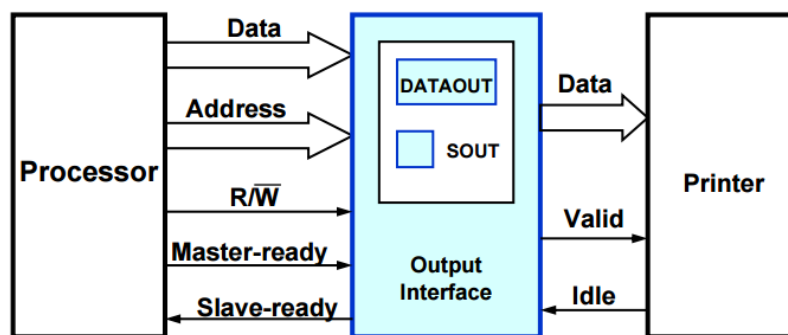
- Interface-circuits contain
 - 1) Data register DATAIN &
 - 2) Status-flag SIN.
- When a key is pressed, the Valid signal changes from 0 to 1.

Then, $SIN=1$, when ASCII code is loaded into DATAIN.

$SIN = 0$, when processor reads the contents of the DATAIN.

- The interface-circuit is connected to the asynchronous bus.
- Data transfers on the bus are controlled using the handshake signals:
 - 1) Master ready &
 - 2) Slave ready.

The hardware components needed for connecting a Printer to a processor.



- Processor uses
 - memory-mapped I/O and
 - asynchronous bus protocol.
- On the processor-side of the interface, we have:
 - Data-lines
 - Address-lines
 - Control or R/W line
 - Master-Ready signal and
 - Slave-Ready signal.

The circuit of output interface,

- Slave-ready
- $R/\sim W$
- Master-ready
- Address decoder

- Handshake control

The input and output interfaces can be combined into a single interface.

The general purpose parallel interface circuit that can be configured in a variety of ways.

For increased flexibility, the circuit makes it possible for some lines to serve as inputs and some lines to serve as outputs, under program control.

Serial Port

- A serial port is used to connect the processor to I/O devices that require transmission of data one bit at a time.
- The key feature of an interface circuit for a serial port is that it is capable of communicating in a bit-serial fashion on the device side and in a bit parallel fashion on the bus side
- The transformation between the parallel and serial formats is achieved with shift registers that have parallel access capability.

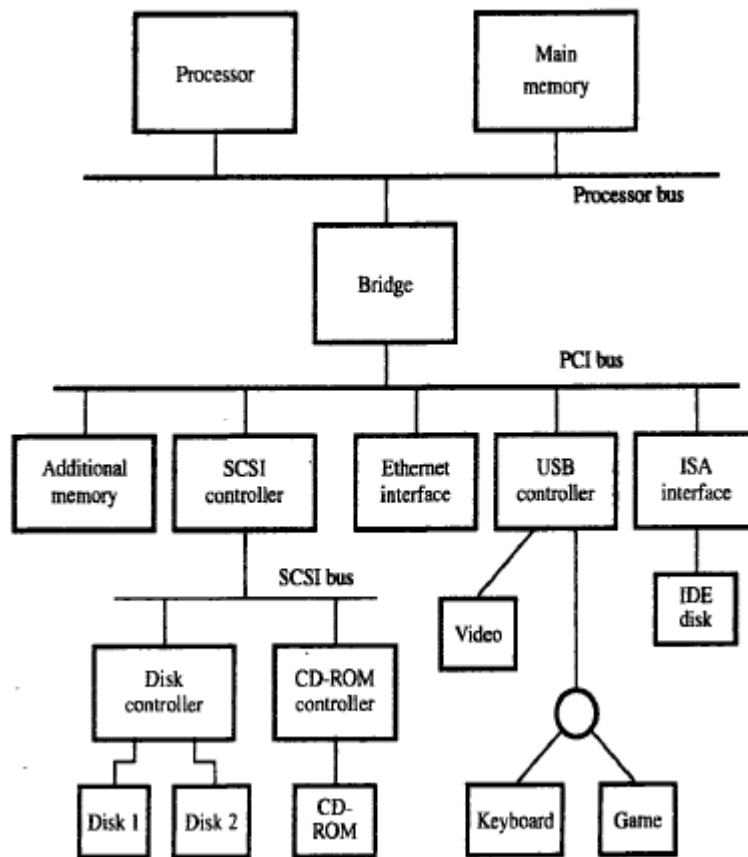
STANDARD I/O INTERFACES

Consider a computer system using different interface standards. The three major standard I/O interfaces discussed here are:

- PCI (Peripheral Component Interconnect)
- SCSI (Small Computer System Interface)
- USB (Universal Serial Bus)
- PCI defines an expansion bus on the motherboard.
- SCSI and USB are used for connecting additional devices both inside and outside the computer-box.

PCI (Peripheral Component Interconnect)

- PCI is developed as a low cost bus that is truly processor independent.
- PCI supports high speed disk, graphics and video devices.
- PCI has plug and play capability for connecting I/O devices.
- To connect new devices, the user simply connects the device interface board to the bus.
- Processor bus and Peripheral Component Interconnect (PCI) bus are interconnected by a circuit called **Bridge**.
- The bridge translates the signals and protocols of one bus into another.
- The bridge-circuit introduces a small delay in data transfer between processor and the devices.

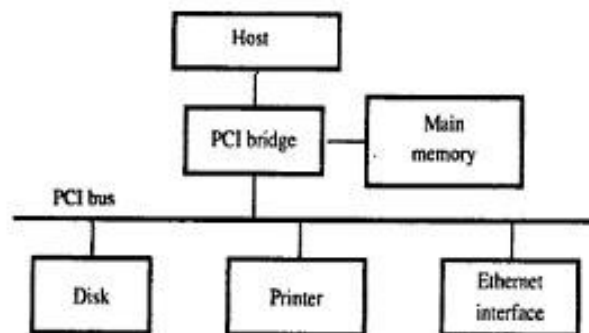


DATA TRANSFER IN PCI

- During **read-operation**,
 - When the processor specifies an address, the memory responds by sending a sequence of data-words from successive memory-locations.
- During **write-operation**,
 - When the processor sends an address, a sequence of data-words is written into successive memory-locations.
- PCI supports read and write-operation.
- A read/write-operation involving a single word is treated as a burst of length one.
- PCI has 3 address-spaces. They are

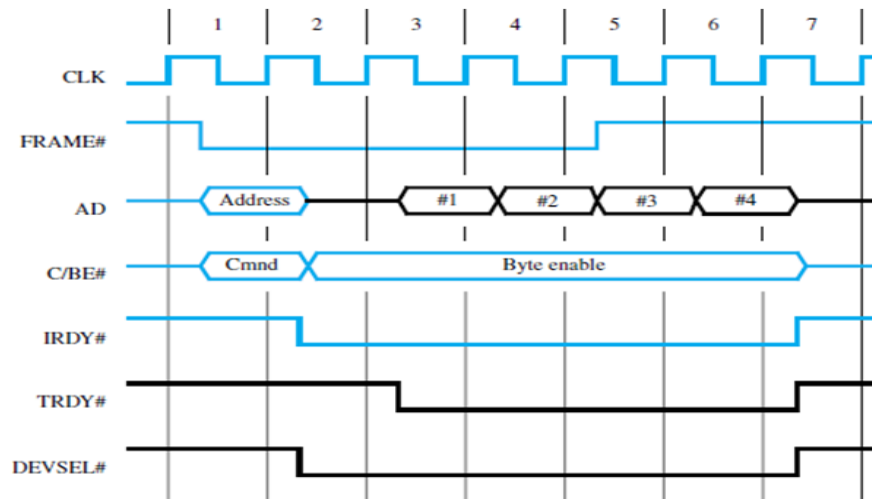
- 1) Memory address-space
- 2) I/O address-space &
- 3) Configuration address-space.

- I/O Address-space is intended for use with processor.
- Configuration space is intended to give PCI, its plug and play capability.
- **PCI Bridge** provides a separate physical connection to main-memory.



- The master maintains the address information on the bus until data-transfer is completed.
- At any time, only one device acts as **Bus-Master**.
- A master is called “initiator” which is either processor or DMA.
- The addressed-device that responds to read and write commands is called a **Target**.
 - A complete transfer operation on the bus, involving an address and burst of data is called a transaction.
- Individual word transfers are called “**phases**”.
- Data transfer signals on PCI bus

Data transfer signals on the PCI bus.	
Name	Function
CLK	A 33-MHz or 66-MHz clock
FRAME#	Sent by the initiator to indicate the duration of a transmission
AD	32 address/data lines, which may be optionally increased to 64
C/BE#	4 command/byte-enable lines (8 for a 64-bit bus)
IRDY#, TRDY#	Initiator-ready and Target-ready signals
DEVSEL#	A response from the device indicating that it has recognized its address and is ready for a data transfer transaction
IDSEL#	Initialization Device Select

Read Operation on PCI Bus

- During Clock cycle-1,
 - The processor
 - asserts FRAME# to indicate the beginning of a transaction;
 - sends the address on AD lines and command on C/BE# Lines.
- During Clock cycle-2,
 - The processor removes the address and disconnects its drivers from AD lines.
 - Selected target
 - enables its drivers on AD lines and
 - fetches the requested-data to be placed on bus.
 - Selected target
 - asserts DEVSEL# and
 - maintains it in asserted state until the end of the transaction.
 - C/BE# is
 - used to send a bus command and it is
 - used for different purpose during the rest of the transaction.
- During Clock cycle-3,
 - The initiator asserts IRDY# to indicate that it is ready to receive data.
 - If the target has data ready to send then it asserts TRDY#. In our eg, the target sends 3 more words of data in clock cycle 4 to 6.
- During Clock cycle-5
 - The initiator uses FRAME# to indicate the duration of the burst, since it read 4 words, the initiator negates FRAME# during clock cycle 5.

- During Clock cycle-7,
 - After sending 4th word, the target
 - disconnects its drivers and
 - negates DEVSEL# during clock cycle 7.

DEVICE CONFIGURATION OF PCI

- The PCI has a configuration ROM that stores information about that device.
- The configuration ROM's of all devices are accessible in the configuration address-space.
- The initialization software read these ROM's whenever the system is powered up or reset.
- In each case, it determines whether the device is a printer, keyboard or disk controller.
- Devices are assigned address during initialization process.
- Each device has an input signal called IDSEL# (Initialization device select) which has 21 address- lines (AD11 to AD31).
- During configuration operation,
 - The address is applied to AD input of the device and
 - The corresponding AD line is set to 1 and all other lines are set to 0. AD11 - AD31 ,**Upper** address-line
A0 - A10 , **Lower** address-line: Specify the type of the operation and to access the content of device configuration ROM.
- The configuration software scans all 21 locations. PCI bus has interrupt-request lines.
- Each device may requests an address in the I/O space or memory space

SCSI Bus

- SCSI stands for Small Computer System Interface.
- SCSI refers to the standard bus which is defined by ANSI (American National Standard Institute).
- SCSI bus the several options. It may be,

Narrow bus	It has 8 data-lines & transfers 1 byte at a time.
Wide bus	It has 16 data-lines & transfer 2 byte at a time.
Single-Ended Transmission	Each signal uses separate wire.
HVD (High Voltage Differential)	It was 5v (TTL cells)

LVD (Low Voltage Differential)	It uses 3.3v
--------------------------------	--------------

- Because of these various options, SCSI connector may have 50, 68 or 80 pins.
- The data transfer rate ranges from 5MB/s to 160MB/s 320Mb/s, 640MB/s. The transfer rate depends on,
 - 1) Length of the cable
 - 2) Number of devices connected.
- To achieve high transfer rate, the bus length should be 1.6m for SE signaling and 12m for LVD signaling.
- The SCSI bus is connected to the processor-bus through the SCSI controller.
- The data are stored on a disk in blocks called sectors. Each sector contains several hundreds of bytes. These data will not be stored in contiguous memory-location.
- SCSI protocol is designed to retrieve the data in the first sector or any other selected sectors.
- Using SCSI protocol, the burst of data are transferred at high speed.
- The controller connected to SCSI bus is of 2 types.

They are 1) Initiator

2) Target

1) Initiator

- It has the ability to select a particular target & to send commands specifying the operation to be performed.
- They are the controllers on the processor side.

2) Target

- The disk controller operates as a target.
- It carries out the commands it receives from the initiator.
- The initiator establishes a logical connection with the intended target.

Steps for Read-operation.

The processor sends a command to the SCSI controller, which causes the following sequence of events to take place:

- 1) The SCSI controller contends for control of the bus (initiator).
- 2) When the initiator wins the arbitration-process, the initiator
 - selects the target controller and
 - hands over control of the bus to it.
- 3) The target starts an output operation. The initiator sends a command specifying the required read-operation.

- 4) The target
 - sends a message to initiator indicating that it will temporarily suspend connection b/w them.
 - then releases the bus.
- 5) The target controller sends a command to the disk drive to move the read head to the first sector involved in the requested read-operation.
6. The target
 - transfers the contents of the data buffer to the initiator and
 - then suspends the connection again.
- 7) The target controller sends a command to the disk drive to perform another seek operation.
- 8) As the initiator controller receives the data, it stores them into the main-memory using the DMA approach.
- 9) The SCSI controller sends an interrupt to the processor indicating that the data are now available.

BUS SIGNALS OF SCSI

Table 4.4 The SCSI bus signals

Category	Name	Function
Data	–DB(0) to –DB(7)	Data lines: Carry one byte of information during the information transfer phase and identify device during arbitration, selection and reselection phases
	–DB(P)	Parity bit for the data bus
Phase	–BSY	Busy: Asserted when the bus is not free
	–SEL	Selection: Asserted during selection and reselection
Information type	–C/D	Control/Data: Asserted during transfer of control information (command, status or message)
	–MSG	Message: indicates that the information being transferred is a message
Handshake	–REQ	Request: Asserted by a target to request a data transfer cycle
	–ACK	Acknowledge: Asserted by the initiator when it has completed a data transfer operation
Direction of transfer	–I/O	Input/Output: Asserted to indicate an input operation (relative to the initiator)
Other	–ATN	Attention: Asserted by an initiator when it wishes to send a message to a target
	–RST	Reset: Causes all device controls to disconnect from the bus and assume their start-up state

PHASES IN SCSI BUS

- The phases in SCSI bus operation are:

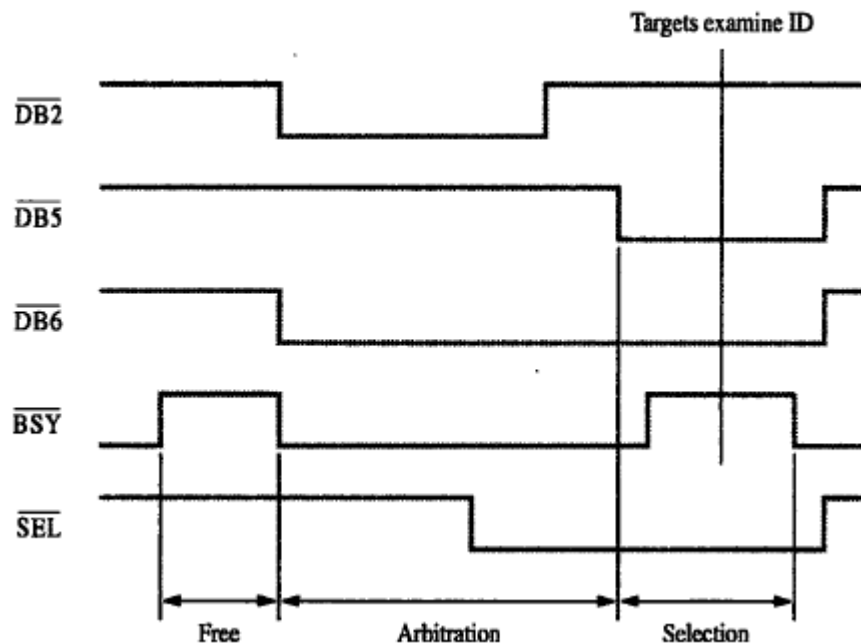
- 1) Arbitration
- 2) Selection
- 3) Information transfer
- 4) Reselection

1) Arbitration

- When the –BSY signal is in inactive state,
 - the bus will be free &
 - any controller can request the use of bus.
- SCSI uses distributed arbitration scheme because each controller may generate requests at the same time.
- Each controller on the bus is assigned a fixed priority.
- When BSY becomes active, all controllers that are requesting the bus
 - examines the data-lines &

→ determine whether highest priority device is requesting bus at the same time.

- The controller using the highest numbered line realizes that it has won the arbitration-process.
- At that time, all other controllers disconnect from the bus & wait for $\overline{\text{BSY}}$ to become inactive again.



2) Information Transfer

- The information transferred between two controllers may consist of
 - commands from the initiator to the target
 - status responses from the target to the initiator or
 - data-transferred to/from the I/O device.
- Handshake signaling is used to control information transfers, with the target controller taking the role of the bus-master.

3) Selection

- Here, Device
 - wins arbitration and
 - asserts $\overline{\text{BSY}}$ and $\overline{\text{DB6}}$ signals.
- The Select Target Controller responds by asserting $\overline{\text{BSY}}$.
- This informs that the connection that it requested is established.

4) Reselection

- The connection between the two controllers has been reestablished, with the target in

control of the bus as required for data transfer to proceed.

Universal Serial Bus (USB)

- USB stands for Universal Serial Bus.
- USB supports 3 speed of operation. They are,
 - 1) Low speed (1.5 Mbps)
 - 2) Full speed (12 mbps) &
 - 3) High speed (480 mbps).
- The USB has been designed to meet the key objectives. They are,
 - 1) Provide a simple, low-cost and easy to use interconnection system.
This overcomes difficulties due to the limited number of I/O ports available on a computer.
 - 2) Accommodate a wide range of data transfer characteristics for
I/O devices. For e.g. telephone and Internet connections
 - 3) Enhance user convenience through a “plug-and-play” mode of operation.
- **Advantage:** USB helps to add many devices to a computer system at any time without opening the computer-box.

Port Limitation

- Normally, the system has a few limited ports.
- To add new ports, the user must open the computer-box to gain access to the internal expansion bus & install a new interface card.
- The user may also need to know to configure the device & the s/w.

Plug & Play

- The main objective: USB provides a plug & play capability.
- The plug & play feature enhances the connection of new device at any time, while the system is operation.
- The system should
 - Detect the existence of the new device automatically.
 - Identify the appropriate device driver s/w.
 - Establish the appropriate addresses.
 - Establish the logical connection for communication.

DEVICE CHARACTERISTICS OF USB

- The kinds of devices that may be connected to a computer cover a wide range of functionality.
- The speed, volume & timing constraints associated with data transfer to & from devices varies significantly.

Eg: 1 Keyboard

- Since the event of pressing a key is not synchronized to any other event in a computer system, the data generated by keyboard are called asynchronous.
- The data generated from keyboard depends upon the speed of the human operator which is about 100 bytes/sec.

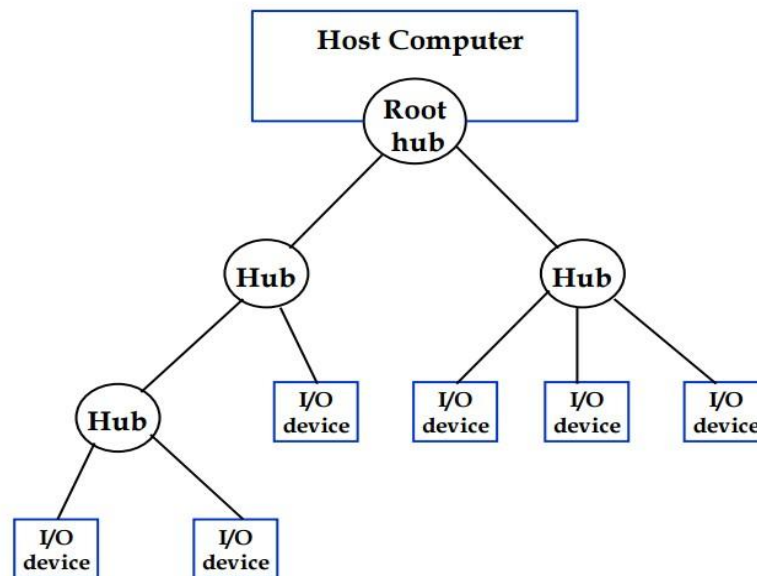
Eg: 2 Microphone attached in a computer system internally/externally

- The sound picked up by the microphone produces an analog electric signal, which must be converted into digital form before it can be handled by the computer.
- This is accomplished by sampling the analog signal periodically.
- The sampling process yields a continuous stream of digitized samples that arrive at regular intervals, synchronized with the sampling clock. Such a stream is called isochronous (i.e.) successive events are separated by equal period of time.
- If the sampling rate is s samples/sec then the maximum frequency captured by sampling process is $s/2$.
- A standard rate for digital sound is 44.1 KHz.

USB architecture

- A serial transmission format has been chosen for the USB because a serial bus satisfies the low-cost and flexibility requirements
- Clock and data information are encoded together and transmitted as a single signal. Hence, there are no limitations on clock frequency or distance arising from data skew
- To accommodate a large number of devices that can be added or removed at any time, the USB has the tree structure
- Each node of the tree has a device called a hub, which acts as an intermediate control point between the host and the I/O device. At the root of the tree, a root hub connects the entire tree to the host computer
- To accommodate a large number of devices that can be added or removed at any time, the USB has the tree structure. Each node has a device called a hub. Root hub, functions, split bus operations – high speed (HS) and Full/Low speed

- Tree structure of a USB is shown fig below



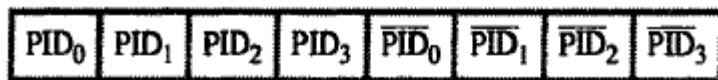
USB ADDRESSING

- Each device may be a hub or an I/O device.
- Each device on the USB is assigned a 7-bit address.
- This address
 - is local to the USB tree and
 - is not related in any way to the addresses used on the processor-bus.
- A hub may have any number of devices or other hubs connected to it, and addresses are assigned arbitrarily.
- When a device is first connected to a hub, or when it is powered-on, it has the address 0.
- The hardware of the hub detects the device that has been connected, and it records this fact as part of its own status information.
- Periodically, the host polls each hub to
 - collect status information and
 - learn about new devices that may have been added or disconnected.
- When the host is informed that a new device has been connected, it uses sequence of commands to

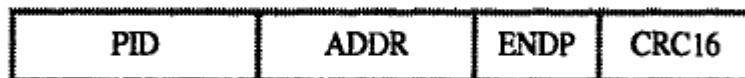
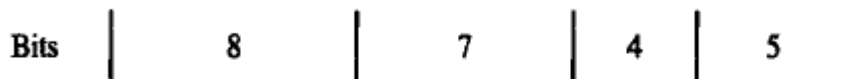
- send a reset signal on the corresponding hub port.
- read information from the device about its capabilities.
- send configuration information to the device, and
- assign the device a unique USB address.
- Once this sequence is completed, the device
 - begins normal operation and
 - responds only to the new address.

USB PROTOCOL

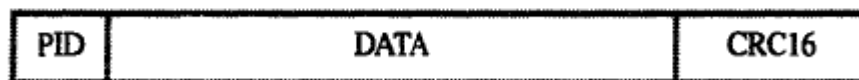
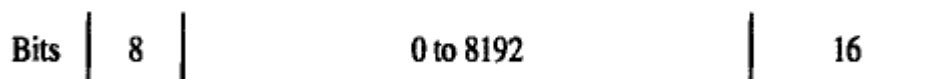
- All information transferred over the USB is organized in packets.
- A packet consists of one or more bytes of information.
- There are many types of packets that perform a variety of control functions.
- The information transferred on USB is divided into 2 broad categories: 1) Control and 2) Data.
- Control packets perform tasks such as
 - addressing a device to initiate data transfer.
 - acknowledging that data have been received correctly or
 - indicating an error.
- Data-packets carry information that is delivered to a device.
- A packet consists of one or more fields containing different kinds of information.
- The first field of any packet is called the **Packet Identifier (PID)** which identifies type of that packet.
- They are transmitted twice.
 - 1) The first time they are sent with their true values and
 - 2) The second time with each bit complemented.
- The four PID bits identify one of 16 different packet types.
- Some control packets, such as ACK (Acknowledge), consist only of the PID byte.
- Control packets used for controlling data transfer operations are called **Token Packets**.



(a) Packet identifier field



(b) Token packet, IN or OUT



(c) Data packet

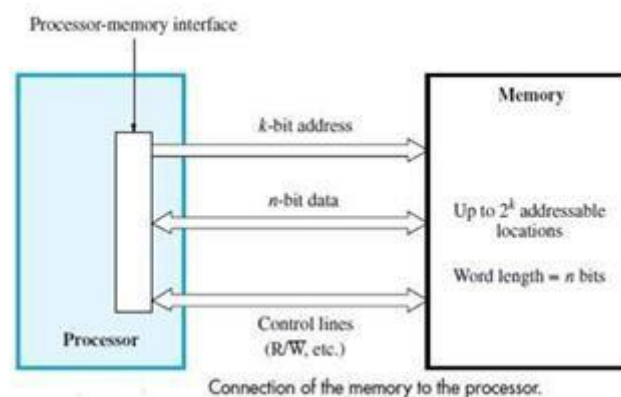
USB packet formats.

MODULE 3

MEMORY SYSTEM

BASIC CONCEPTS

- Maximum size of memory that can be used in any computer is determined by addressing mode.
- If MAR is k-bits long then



- memory may contain upto 2^K addressable-locations

Address	Memory Locations
16 Bit	$2^{16} = 64 \text{ K}$
32 Bit	$2^{32} = 4\text{G (Giga)}$
40 Bit	$2^{40} = 1\text{T (Tera)}$

- If MDR is n-bits long, then
 - n-bits of data are transferred between the memory and processor.
- The data-transfer takes place over the processor-bus (Figure 8.1).
- The processor-bus has
 - 1) Address-Line
 - 2) Data-line &
 - 3) Control-Line (R/W, MFC – Memory Function Completed).
- The Control-Line is used for coordinating data-transfer.
- The processor reads the data from the memory by
 - loading the address of the required memory-location into MAR and
 - setting the R/W line to 1.
- The memory responds by
 - placing the data from the addressed-location onto the data-lines and
 - confirms this action by asserting MFC signal.
- Upon receipt of MFC signal, the processor loads the data from the data-lines into MDR.
- The processor writes the data into the memory-location by
 - loading the address of this location into MAR &
 - setting the R/W line to 0.

- **Memory Access Time:** It is the time that elapses between
 - initiation of an operation &
 - completion of that operation.
- **Memory Cycle Time:** It is the minimum time delay that required between the initiation of the two successive memory-operations.

RAM (Random Access Memory)

- In RAM, any location can be accessed for a Read/Write-operation in fixed amount of time,

Cache Memory

- It is a small, fast memory that is inserted between
 - larger slower main-memory and
 - processor.
- It holds the currently active segments of a program and their data.

Virtual Memory

- The address generated by the processor is referred to as a **virtual/logical address**.
- The virtual-address-space is mapped onto the physical-memory where data are actually stored.
- The mapping-function is implemented by MMU. (MMU = memory management unit).
- Only the active portion of the address-space is mapped into locations in the physical-memory.
- The remaining virtual-addresses are mapped onto the bulk storage devices such as magnetic disk.
- As the active portion of the virtual-address-space changes during program execution, the

MMU

- changes the mapping-function &
- transfers the data between disk and memory.
- During every memory-cycle, MMU determines whether the addressed-page is in the memory. If the page is in the memory.
 - Then, the proper word is accessed and execution proceeds.
 - Otherwise, a page containing desired word is transferred from disk to memory.

- Memory can be classified as follows:

1) RAM which can be further classified as follows:

- i) Static RAM
- ii) Dynamic RAM (DRAM) which can be further classified as synchronous & asynchronous DRAM.

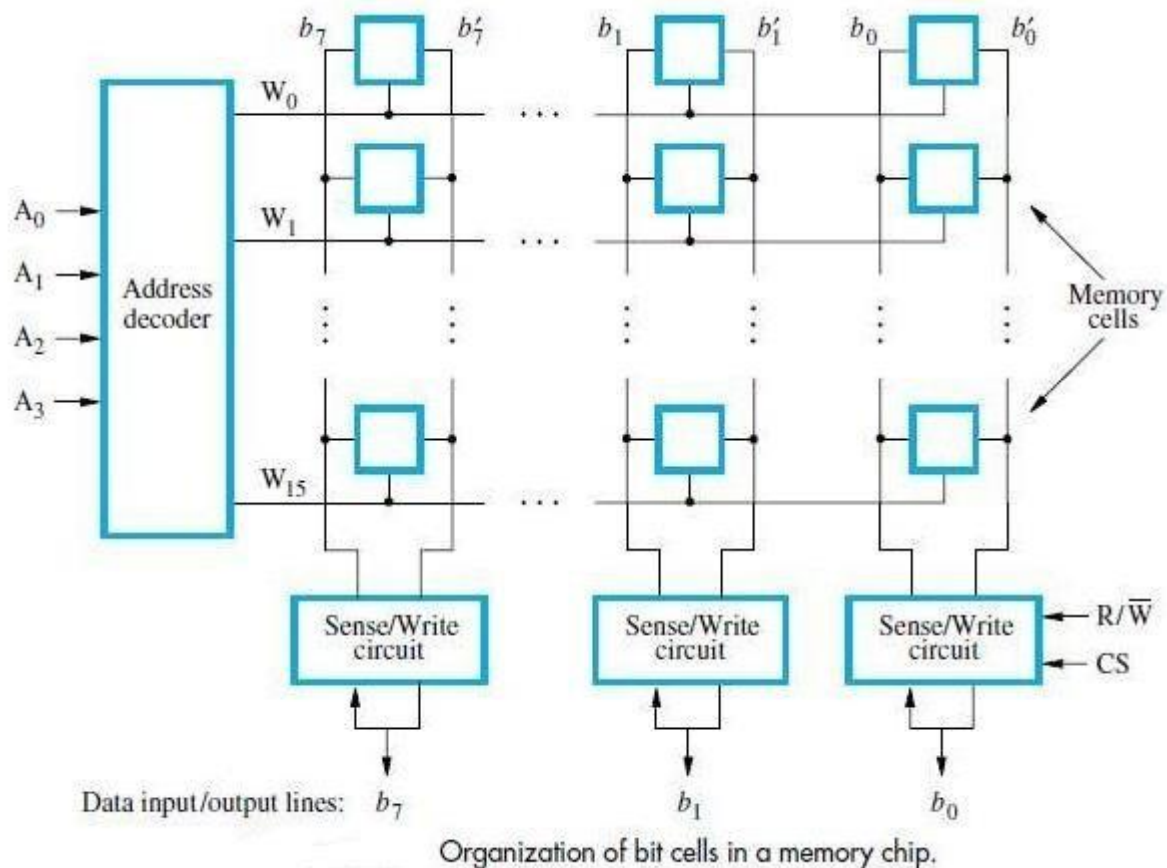
2) ROM which can be further classified as follows:

- i) PROM
- ii) EPROM
- iii) EEPROM &
- iv) Flash Memory which can be further classified as Flash Cards & Flash Drives.

SEMI CONDUCTOR RAM MEMORIES

INTERNAL ORGANIZATION OF MEMORY-CHIPS

- Memory-cells are organized in the form of array as shown in the figure.
- Each cell is capable of storing 1-bit of information.
- Each row of cells forms a memory-word.
- All cells of a row are connected to a common line called as **Word-Line**.
- The cells in each column are connected to **Sense/Write** circuit by 2-bit-lines.
- The Sense/Write circuits are connected to data-input or output lines of the chip.
- During a write-operation, the sense/write circuit
 - receive input information &
 - store input info in the cells of the selected word.

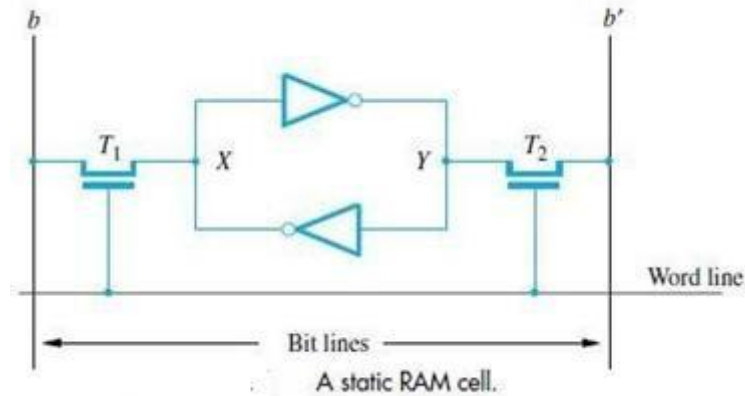


- The data-input and data-output of each Sense/Write circuit are connected to a single bidirectional data-line.
- Data-line can be connected to a data-bus of the computer.
- Following 2 control lines are also used:
 - 1) R/\bar{W} □ specifies the required operation.
 - 2) CS' □ Chip Select input selects a given chip in the multi-chip memory-system.

Bit Organization	Requirement of external connection for address, data and control lines
128 (16x8)	14
(1024) 128x8(1k)	19

STATIC RAM (OR MEMORY)

- Memories consist of circuits capable of retaining their state as long as power is applied are known.



- Two inverters are cross connected to form a latch as shown in the above Figure.
- The latch is connected to 2-bit-lines by transistors T1 and T2.
- The transistors act as switches that can be opened/closed under the control of the word-line.
- When the word-line is at ground level, the transistors are turned off and the latch retain its state.

Read Operation

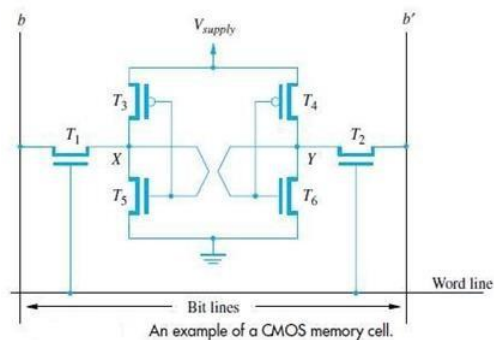
- To read the state of the cell, the word-line is activated to close switches T1 and T2.
- If the cell is in state 1, the signal on bit-line b is high and the signal on the bit-line b'' is low.
- Thus, b and b'' are complement of each other.
- Sense/Write circuit
 - monitors the state of b & b'' and
 - sets the output accordingly.

Write Operation

- The state of the cell is set by
 - placing the appropriate value on bit-line b and its complement on b'' and
 - then activating the word-line. This forces the cell into the corresponding state.
- The required signal on the bit-lines is generated by Sense/Write circuit.

CMOS Cell

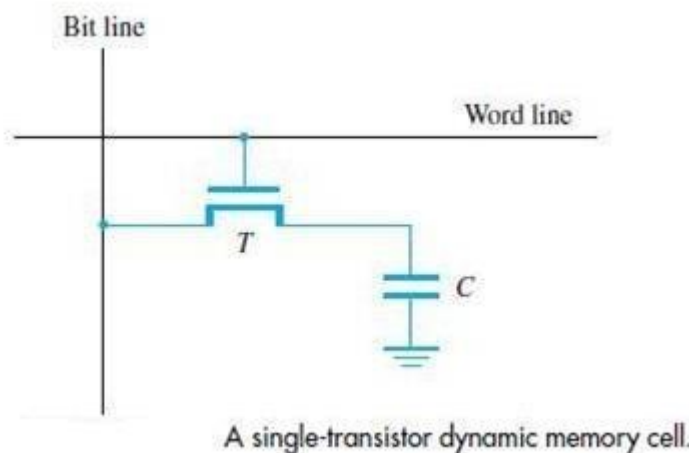
- Transistor pairs (T3, T5) and (T4, T6) form the inverters in the latch as shown in the Figure



- In state 1, the voltage at point X is high by having T5, T6 ON and T4, T5 are OFF.
- Thus, T1 and T2 returned ON (Closed), bit-line b and b'' will have high and low signals respectively.
- Advantages:**
 - 1) It has low power consumption „“ the current flows in the cell only when the cell is active.
 - 2) Static RAM"s can be accessed quickly. It access time is few nanoseconds.
- Disadvantage:** SRAMs are said to be volatile memories „“ their contents are lost when power is interrupted.

ASYNCHRONOUS DRAM

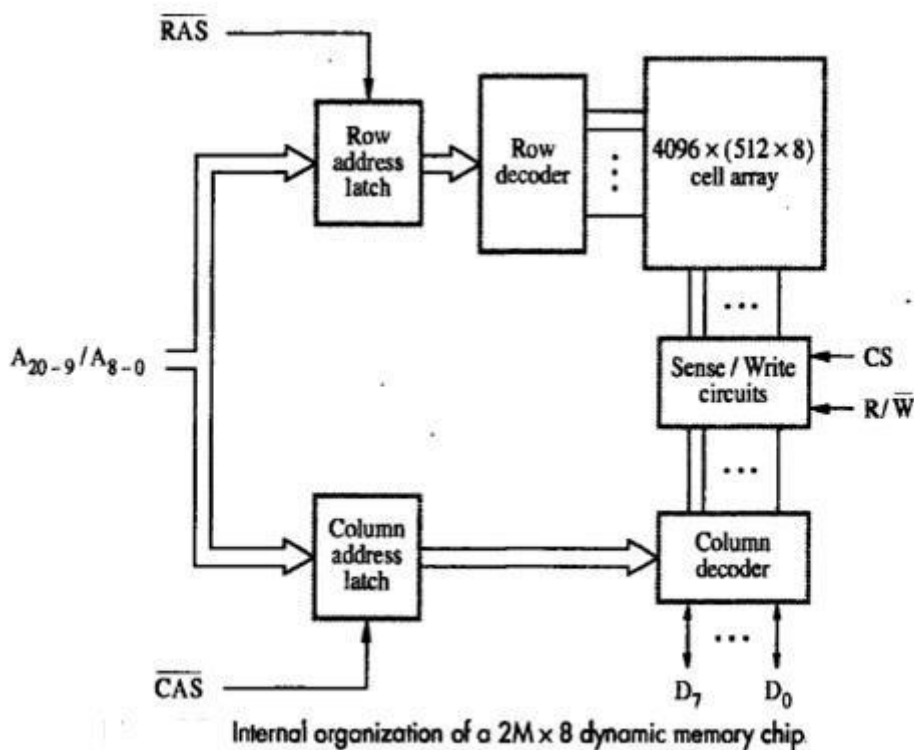
- Less expensive RAMs can be implemented if simple cells are used.
- Such cells cannot retain their state indefinitely. Hence they are called **Dynamic RAM (DRAM)**.
- The information stored in a dynamic memory-cell in the form of a charge on a capacitor.
- This charge can be maintained only for tens of milliseconds.
- The contents must be periodically refreshed by restoring this capacitor charge to its full value.



- In order to store information in the cell, the transistor T is turned “ON” as shown in the Figure .
- The appropriate voltage is applied to the bit-line which charges the capacitor.
- After the transistor is turned off, the capacitor begins to discharge.
- Hence, info. stored in cell can be retrieved correctly before threshold value of capacitor drops down.
- During a read-operation,
 - transistor is turned “ON”
 - a sense amplifier detects whether the charge on the capacitor is above the threshold value.
 - If (charge on capacitor) > (threshold value) □ Bit-line will have logic value “1”
 - If (charge on capacitor) < (threshold value) □ Bit-line will set to logic value “0”.

ASYNCHRONOUS DRAM DESCRIPTION

- The 4 bit cells in each row are divided into 512 groups of 8 as shown in the Figure .
- 21 bit address is needed to access a byte in the memory. 21 bit is divided as follows:
 - 1) 12 address bits are needed to select a row.
i.e. A₈₋₀ → specifies row-address of a byte.
 - 2) 9 bits are needed to specify a group of 8 bits in the selected row.
i.e. A₂₀₋₉ → specifies column-address of a byte.



- During Read/Write-operation,
 - row-address is applied first.
 - row-address is loaded into row-latch in response to a signal pulse on **RAS**’ input of chip.
(RAS = Row-address Strobe CAS = Column-address Strobe)
- When a Read-operation is initiated, all cells on the selected row are read and refreshed.
- Shortly after the row-address is loaded, the column-address is
 - applied to the address pins &
 - loaded into **CAS**’.

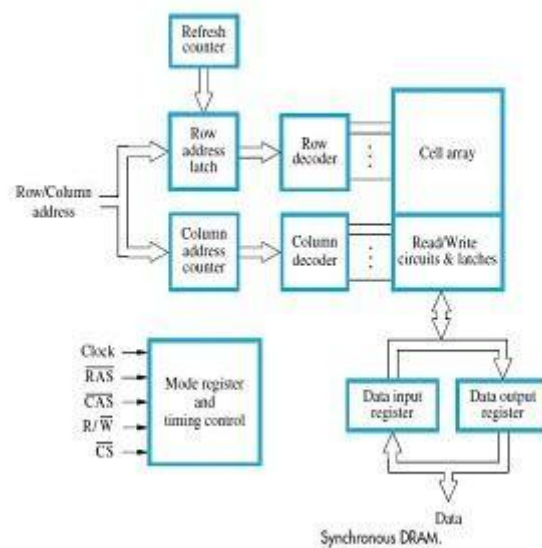
- The information in the latch is decoded.
- The appropriate group of 8 Sense/Write circuits is selected.
 $R/W=1$ (read-operation) □ Output values of selected circuits are transferred to data-lines D0-D7.
 $R/W=0$ (write-operation) □ Information on D0-D7 are transferred to the selected circuits.
- RAS & CAS are active-low so that they cause latching of address when they change from high to low.
- To ensure that the contents of DRAMs are maintained, each row of cells is accessed periodically.
- A special memory-circuit provides the necessary control signals RAS & CAS that govern the timing.
- The processor must take into account the delay in the response of the memory.

Fast Page Mode

- Transferring the bytes in sequential order is achieved by applying the consecutive sequence of column-address under the control of successive CAS signals.
- This scheme allows transferring a block of data at a faster rate.
- The block of transfer capability is called as *fast page mode*.

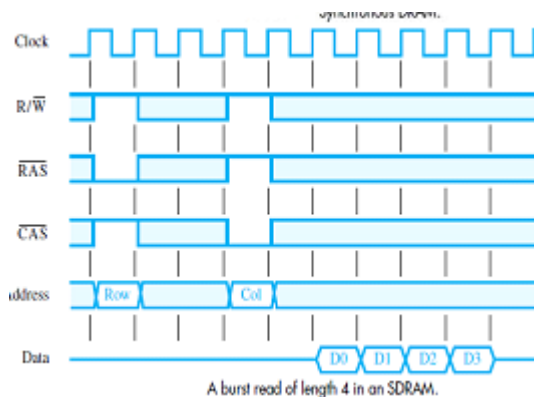
SYNCHRONOUS DRAM

- The operations are directly synchronized with clock signal as shown in the figure



- The address and data connections are buffered by means of registers.
- The output of each sense amplifier is connected to a latch.
- A Read-operation causes the contents of all cells in the selected row to be loaded in these latches.
- Data held in latches that correspond to selected columns are transferred into data-output register.
- Thus, data becoming available on the data-output pins.

- First, the row-address is latched under control of RAS" signal as shown in the Figure



- The memory typically takes 2 or 3 clock cycles to activate the selected row.
- Then, the column-address is latched under the control of CAS" signal.
- After a delay of one clock cycle, the first set of data bits is placed on the data-lines.
- SDRAM automatically increments column-address to access next 3 sets of bits in the selected row.

LATENCY & BANDWIDTH

- A good indication of performance is given by 2 parameters: 1) Latency 2) Bandwidth.

Latency

- It refers to the amount of time it takes to transfer a word of data to or from the memory.
- For a transfer of single word, the latency provides the complete indication of memory performance.
- For a block transfer, the latency denotes the time it takes to transfer the first word of data.

Bandwidth

- It is defined as the number of bits or bytes that can be transferred in one second.
- Bandwidth mainly depends on
 - 1) The speed of access to the stored data &
 - 2) The number of bits that can be accessed in parallel.

DOUBLE DATA RATE SDRAM (DDR-SDRAM)

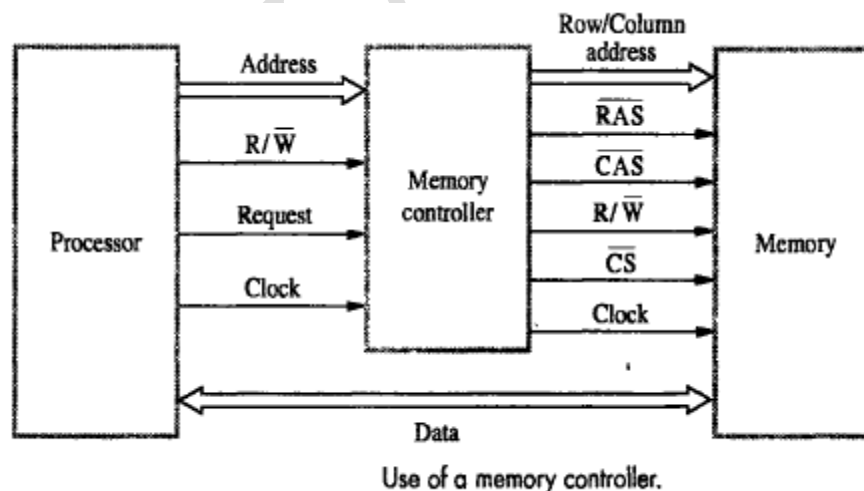
- The standard SDRAM performs all actions on the rising edge of the clock signal.
- The DDR-SDRAM transfer data on both the edges (leading edge, trailing edge).
- The Bandwidth of DDR-SDRAM is doubled for long burst transfer.
- To make it possible to access the data at high rate, the cell array is organized into two banks.
- Each bank can be accessed separately.
- Consecutive words of a given block are stored in different banks.
- Such interleaving of words allows simultaneous access to two words.
- The two words are transferred on successive edge of the clock.

STRUCTURE OF LARGER MEMORIES**Dynamic Memory System**

- The physical implementation is done in the form of memory-modules.
- If a large memory is built by placing DRAM chips directly on the Motherboard, then it will occupy large amount of space on the board.
- These packaging consideration have led to the development of larger memory units known as SIMM's & DIMM's.
 - 1) SIMM □ Single Inline memory-module
 - 2) DIMM □ Dual Inline memory-module
- SIMM/DIMM consists of many memory-chips on small board that plugs into a socket on motherboard.

MEMORY SYSTEM CONSIDERATION**MEMORY CONTROLLER**

- To reduce the number of pins, the dynamic memory-chips use multiplexed-address inputs.
- The address is divided into 2 parts:
 - 1) **High Order Address Bit**
 - Select a row in cell array.
 - It is provided first and latched into memory-chips under the control of RAS's signal.
 - 2) **Low Order Address Bit**
 - Selects a column.
 - They are provided on same address pins and latched using CAS' signals.
- The Multiplexing of address bit is usually done by **Memory Controller Circuit** as shown in the figure.



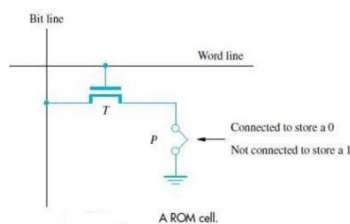
- The Controller accepts a complete address & R/W' signal from the processor.
- A Request signal indicates a memory access operation is needed.
- Then, the Controller
 - forwards the row & column portions of the address to the memory.
 - generates RAS' & CAS' signals &
 - sends R/W' & CS' signals to the memory.

RAMBUS MEMORY

- The usage of wide bus is expensive.
- Rambus developed the implementation of narrow bus.
- Rambus technology is a fast signaling method used to transfer information between chips.
- The signals consist of much smaller voltage swings around a reference voltage V_{ref} .
- The reference voltage is about 2V.
- The two logical values are represented by 0.3V swings above and below V_{ref} .
- This type of signaling is generally known as **Differential Signalling**.
- Rambus provides a complete specification for design of communication called as **Rambus Channel**.
- Rambus memory has a clock frequency of 400 MHz.
- The data are transmitted on both the edges of clock so that effective data-transfer rate is 800MHz.
- Circuitry needed to interface to Rambus channel is included on chip. Such chips are called **RDRAM**. (RDRAM = Rambus DRAMs).
- Rambus channel has:
 - 1) 9 Data-lines (1 -8 line -> Transfer the data, 9 line -> Parity checking).
 - 2) Control-Line &
 - 3) Power line.
- A two channel rambus has 18 data-lines which has no separate Address-Lines.
- Communication between processor and RDRAM modules is carried out by means of packets transmitted on the data-lines.
- There are 3 types of packets:
 - 1) Request
 - 2) Acknowledge &
 - 3) Data.

READ ONLY MEMORY (ROM)

- Both SRAM and DRAM chips are volatile, i.e. They lose the stored information if power is turned off.
- Many application requires non-volatile memory which retains the stored information if power is turned off.
- For ex:
 - OS software has to be loaded from disk to memory i.e. it requires non-volatile memory.
- Non-volatile memory is used in embedded system.
- Since the normal operation involves only reading of stored data, a memory of this type is called ROM.
 - **At Logic value '0'** Transistor(T) is connected to the ground point(P).
 - Transistor switch is closed & voltage on bit-line nearly drops to zero as shown in Figure
 - **At Logic value '1'** Transistor switch is open.
 - The bit-line remains at high voltage.



- To read the state of the cell, the word-line is activated.
- A Sense circuit at the end of the bit-line generates the proper output value.

TYPES OF ROM

- Different types of non-volatile memory are
 - 1) PROM
 - 2) EPROM
 - 3) EEPROM &
 - 4) Flash Memory (Flash Cards & Flash Drives)

PROM (PROGRAMMABLE ROM)

- PROM allows the data to be loaded by the user.
- Programmability is achieved by inserting a „fuse“ at point P in a ROM cell.
- Before PROM is programmed, the memory contains all 0's.
- User can insert 1's at required location by burning-out fuse using high current-pulse.
- This process is irreversible.
- **Advantages:**
 - 1) It provides flexibility.
 - 2) It is faster.
 - 3) It is less expensive because they can be programmed directly by the user.

EPROM (ERASABLE REPROGRAMMABLE ROM)

- EPROM allows
 - stored data to be erased and
 - new data to be loaded.
- In cell, a connection to ground is always made at „P“ and a special transistor is used.
- The transistor has the ability to function as
 - a normal transistor or
 - a disabled transistor that is always turned „off“.
- Transistor can be programmed to behave as a permanently open switch, by injecting charge into it.
- Erasure requires dissipating the charges trapped in the transistor of memory-cells. This can be done by exposing the chip to ultra-violet light.
- **Advantages:**
 - 1) It provides flexibility during the development-phase of digital-system.
 - 2) It is capable of retaining the stored information for a long time.
- **Disadvantages:**
 - 1) The chip must be physically removed from the circuit for reprogramming.
 - 2) The entire contents need to be erased by UV light.

EEPROM (ELECTRICALLY ERASABLE ROM)

- **Advantages:**
 - 1) It can be both programmed and erased electrically.
 - 2) It allows the erasing of all cell contents selectively.
 - **Disadvantage:** It requires different voltage for erasing, writing and reading the stored data.
-

FLASH MEMORY

- In EEPROM, it is possible to read & write the contents of a single cell.
- In Flash device, it is possible to read contents of a single cell & write entire contents of a block.
- Prior to writing, the previous contents of the block are erased.

Eg. In MP3 player, the flash memory stores the data that represents sound.

- Single flash chips cannot provide sufficient storage capacity for embedded-system.

- **Advantages:**

- 1) Flash drives have greater density which leads to higher capacity & low cost per bit.
- 2) It requires single power supply voltage & consumes less power.

- There are 2 methods for implementing larger memory: 1) Flash Cards & 2) Flash Drives

- 1) Flash Cards**

- One way of constructing larger module is to mount flash-chips on a small card.
 - Such flash-card have standard interface.
 - The card is simply plugged into a conveniently accessible slot.
 - Memory-size of the card can be 8, 32 or 64MB.
 - Eg: A minute of music can be stored in 1MB of memory. Hence 64MB flash cards can store an hour of music.

- 2) Flash Drives**

- Larger flash memory can be developed by replacing the hard disk-drive.
 - The flash drives are designed to fully emulate the hard disk.
 - The flash drives are solid state electronic devices that have no movable parts.

- Advantages:**

- 1) They have shorter seek & access time which results in faster response.
- 2) They have low power consumption. ∴ they are attractive for battery driven application.
- 3) They are insensitive to vibration.

- Disadvantages:**

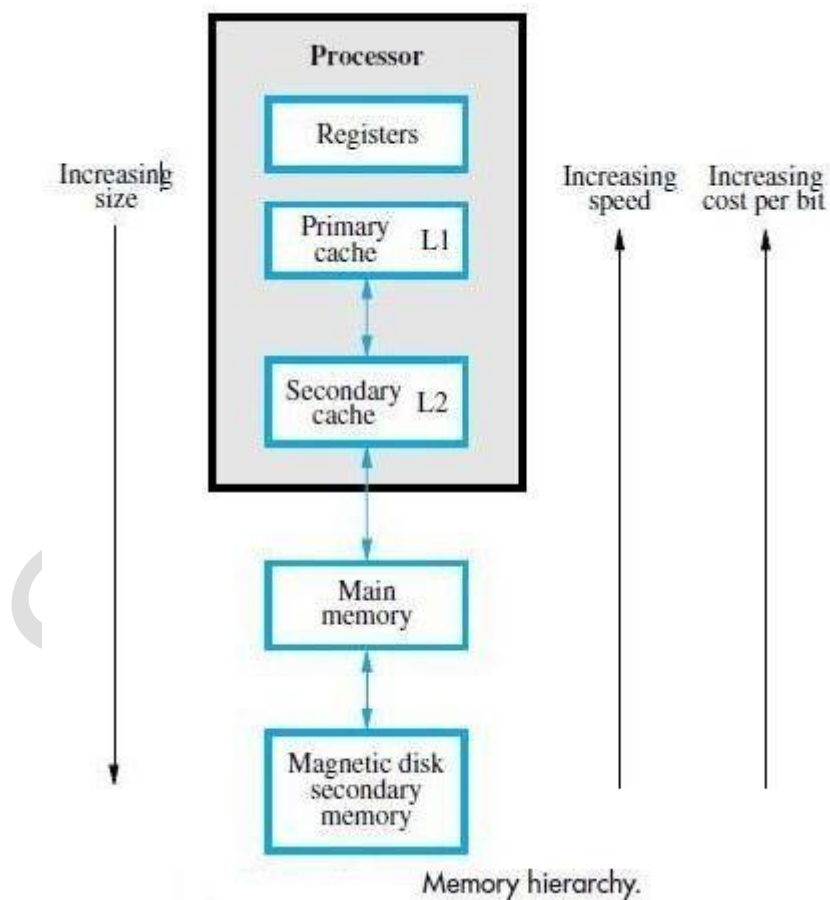
- 1) The capacity of flash drive (<1GB) is less than hard disk (>1GB).
- 2) It leads to higher cost per bit.
- 3) Flash memory will weaken after it has been written a number of times (typically at least 1 million times).

SPEED, SIZE COST

A big challenge in the design of a computer system is to provide a sufficiently large memory, with a reasonable speed at an affordable cost.

Characteristics	SRAM	DRAM	Magnetis Disk
Speed	Very Fast	Slower	Much slower than DRAM
Size	Large	Small	Small
Cost	Expensive	Less Expensive	Low price

- **Memory hierarchy** separates computer storage into a hierarchy based on response time as shown in the figure.



Level-0:

- At level-0, **registers** are present which are contained inside the CPU.
- Since they are present inside the CPU, they have least access time.
- They are most expensive and therefore smallest in size (in KB).
- Registers are implemented using **Flip-Flops**.

Level-1:

- At level-1, **Cache Memory** is present.
- It stores the segments of program that are frequently accessed by the processor.
- It is expensive and therefore smaller in size (in MB).
- Cache memory is implemented using static RAM.

Level-2:

- At level-2, **main memory** is present.
- It can communicate directly with the CPU and with auxiliary memory devices through an I/O processor.
- It is less expensive than cache memory and therefore larger in size (in few GB).
- Main memory is implemented using dynamic RAM.

Level-3:

- At level-3, secondary storage devices like **Magnetic Disk** are present.
- They are used as back up storage.
- They are cheaper than main memory and therefore much larger in size (in few TB).

Level-4:

- At level-4, tertiary storage devices like magnetic tape are present.
- They are used to store removable files.
- They are cheapest and largest in size (1-20 TB).

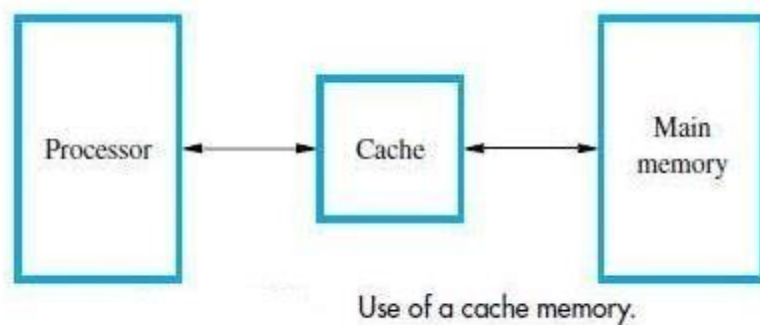
Memory	Speed	Size	Cost
Registers	Very high	Lower	Very Lower
Primary cache	High	Lower	Low
Secondary cache	Low	Low	Low
Main memory	Lower than Secondary cache	High	High
Secondary Memory	Very low	Very High	Very High

CACHE MEMORIES

- The effectiveness of cache mechanism is based on the property of “**Locality of Reference**”

Locality of Reference

- Many instructions in the localized areas of program are executed repeatedly during some time period
- Remainder of the program is accessed relatively infrequently as shown in the Figure .
- There are 2 types:
 - 1) Temporal**
 - The recently executed instructions are likely to be executed again very soon.
 - 2) Spatial**
 - Instructions in close proximity to recently executed instruction are also likely to be executed soon.
- If active segment of program is placed in cache-memory, then total execution time can be reduced.
- Block** refers to the set of contiguous address locations of some size.
- The cache-line is used to refer to the cache-block.



- The Cache-memory stores a reasonable number of blocks at a given time.
- This number of blocks is small compared to the total number of blocks available in main-memory.
- Correspondence b/w main-memory-block & cache-memory-block is specified by mapping-function.
- Cache control hardware decides which block should be removed to create space for the new block.
- The collection of rule for making this decision is called the **Replacement Algorithm**.
- The cache control-circuit determines whether the requested-word currently exists in the cache.
- The write-operation is done in 2 ways: 1) Write-through protocol & 2) Write-back protocol.
 - Write-Through Protocol**
 - Here the cache-location and the main-memory-locations are updated simultaneously.
 - Write-Back Protocol**
 - This technique is to
 - update only the cache-location &
 - mark the cache-location with associated flag bit called **Dirty/Modified Bit**.
 - The word in memory will be updated later, when the marked-block is removed from cache.

During Read-operation

- If the requested-word currently not exists in the cache, then **read-miss** will occur.
- To overcome the read miss, *Load-through/Early restart protocol* is used.
 - Load-Through Protocol**
 - The block of words that contains the requested-word is copied from the memory into cache.
 - After entire block is loaded into cache, the requested-word is forwarded to processor.

During Write-operation

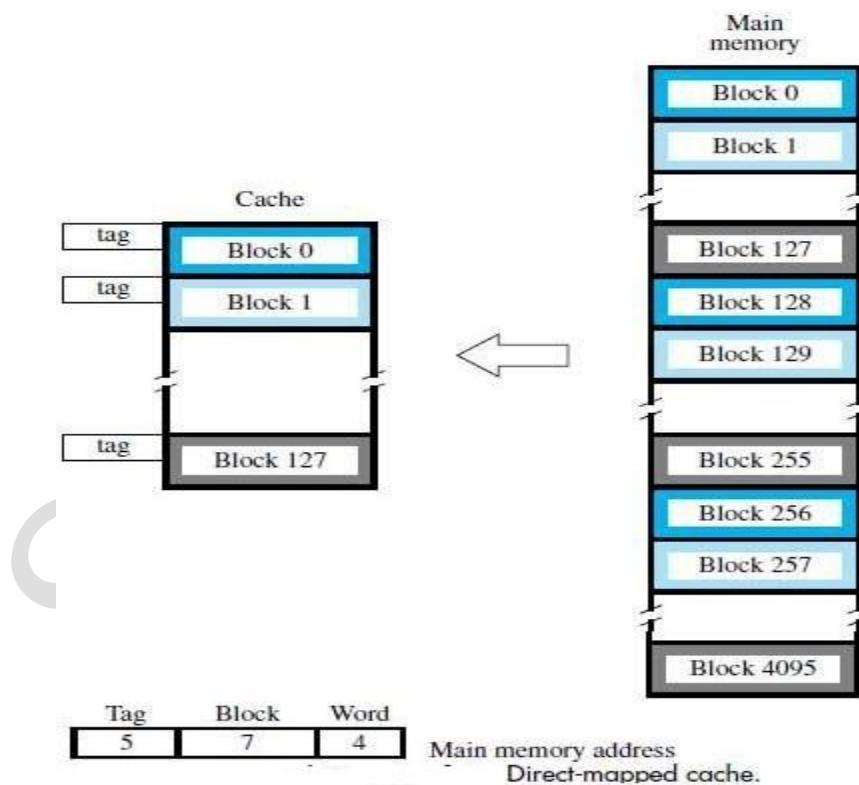
- If the requested-word not exists in the cache, then **write-miss** will occur.
 - If **Write Through Protocol** is used, the information is written directly into main-memory.
 - If **Write Back Protocol** is used,
 - then block containing the addressed word is first brought into the cache &
 - then the desired word in the cache is over-written with the new information.

MAPPING-FUNCTION

- Here we discuss about 3 different mapping-function:
 - Direct Mapping
 - Associative Mapping
 - Set-Associative Mapping

DIRECT MAPPING

- A direct-mapped cache is the simplest approach: each main memory address maps to exactly one cache block.
- The block-j of the main-memory maps onto block-j modulo-128 of the cache as shown in the Figure

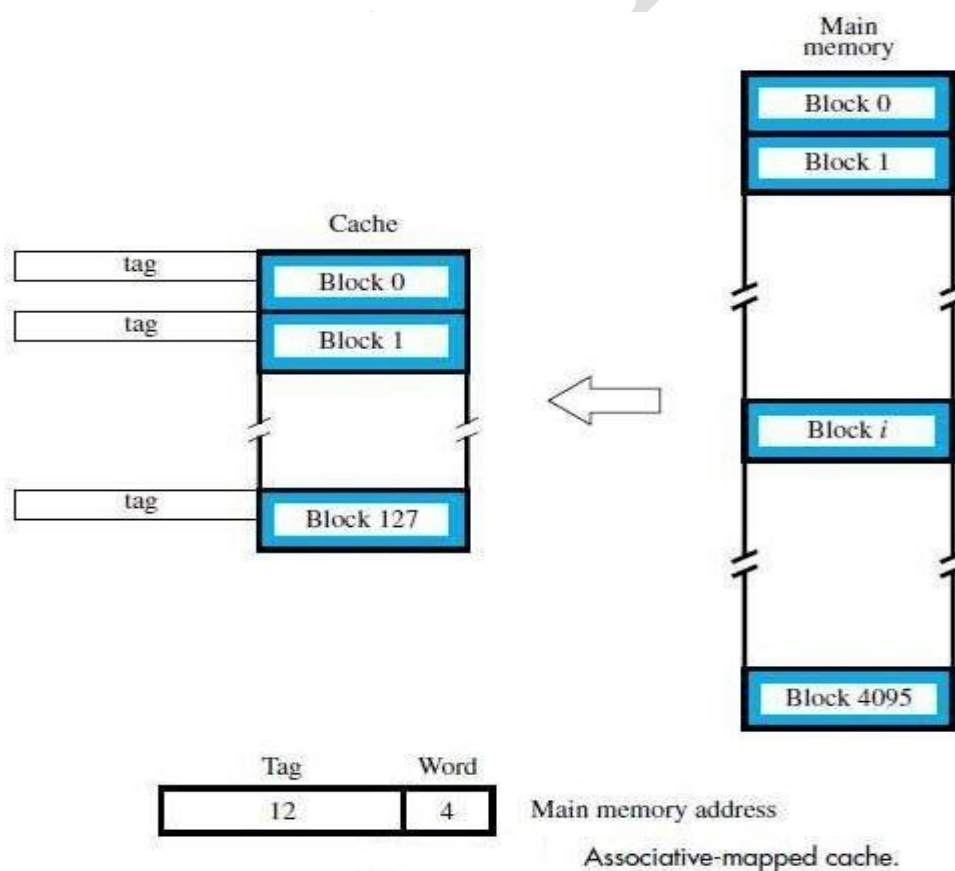


- When the memory-blocks 0, 128, & 256 are loaded into cache, the block is stored in cache-block 0. Similarly, memory-blocks 1, 129, 257 are stored in cache-block 1.
- The contention may arise when
 - When the cache is full.
 - When more than one memory-block is mapped onto a given cache-block position.

- The contention is resolved by allowing the new blocks to overwrite the currently resident-block.
- Memory-address determines placement of block in the cache.
- The memory-address is divided into 3 fields:
 - 1) Low Order 4 bit field**
 - Selects one of 16 words in a block.
 - 2) 7 bit cache-block field**
 - 7-bits determine the cache-position in which new block must be stored.
 - 3) 5 bit Tag field**
 - 5-bits memory-address of block is stored in 5 tag-bits associated with cache-location.
- As execution proceeds,
 - 5-bit tag field of memory-address is compared with tag-bits associated with cache-location.
 - If they match, then the desired word is in that block of the cache.
 - Otherwise, the block containing required word must be first read from the memory.
 - And then the word must be loaded into the cache.

ASSOCIATIVE MAPPING

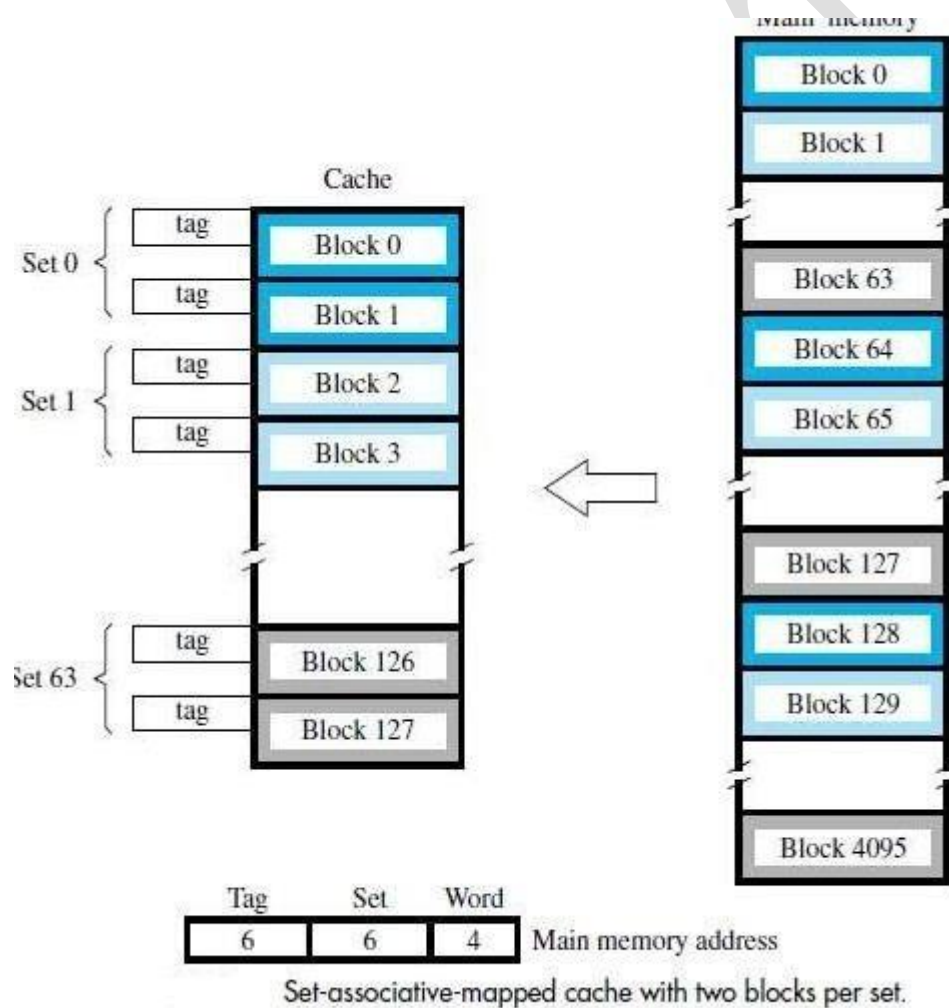
- The memory-block can be placed into any cache-block position as shown in the Figure .



- 12 tag-bits will identify a memory-block when it is resolved in the cache.
- Tag-bits of an address received from processor are compared to the tag-bits of each block of cache.
- This comparison is done to see if the desired block is present.
- It gives complete freedom in choosing the cache-location.
- A new block that has to be brought into the cache has to replace an existing block if the cache is full.
- The memory has to determine whether a given block is in the cache.
- **Advantage:** It is more flexible than direct mapping technique.
- **Disadvantage:** Its cost is high.

SET-ASSOCIATIVE MAPPING

- It is the combination of direct and associative mapping as shown in the figure



- The blocks of the cache are grouped into sets.
- The mapping allows a block of the main-memory to reside in any block of the specified set.
- The cache has 2 blocks per set, so the memory-blocks 0, 64, 128...4032 maps into cache set „0“.
- The cache can occupy either of the two block position within the set.

6 bit set field

- Determines which set of cache contains the desired block.

6 bit tag field

- The tag field of the address is compared to the tags of the two blocks of the set.
- This comparison is done to check if the desired block is present.

- The cache which contains 1 block per set is called **direct mapping**.
- A cache that has „k“ blocks per set is called as “**k-way set associative cache**”.
- Each block contains a control-bit called a **valid-bit**.
- The Valid-bit indicates that whether the block contains valid-data.
- The dirty bit indicates that whether the block has been modified during its cache residency.
- **Valid-bit=0** □ When power is initially applied to system.
- **Valid-bit=1** □ When the block is loaded from main-memory at first time.
- If the main-memory-block is updated by a source & if the block in the source is already exists in the cache, then the valid-bit will be cleared to “0”.
- If Processor & DMA uses the same copies of data then it is called as **Cache Coherence Problem**.
- **Advantages:**
 - 1) Contention problem of direct mapping is solved by having few choices for block placement.
 - 2) The hardware cost is decreased by reducing the size of associative search.

REPLACEMENT ALGORITHM

- In direct mapping method,
the position of each block is pre-determined and there is no need of replacement strategy.
- In associative & set associative method,
The block position is not pre-determined.
If the cache is full and if new blocks are brought into the cache,
then the cache-controller must decide which of the old blocks has to be replaced.
- When a block is to be overwritten, the block with longest time w/o being referenced is over-written.
- This block is called **Least recently Used (LRU) block** & the technique is called **LRU algorithm**.
- The cache-controller tracks the references to all blocks with the help of block-counter.
- **Advantage:** Performance of LRU is improved by randomness in deciding which block is to be over-written.

Eg:

Consider 4 blocks/set in set associative cache.

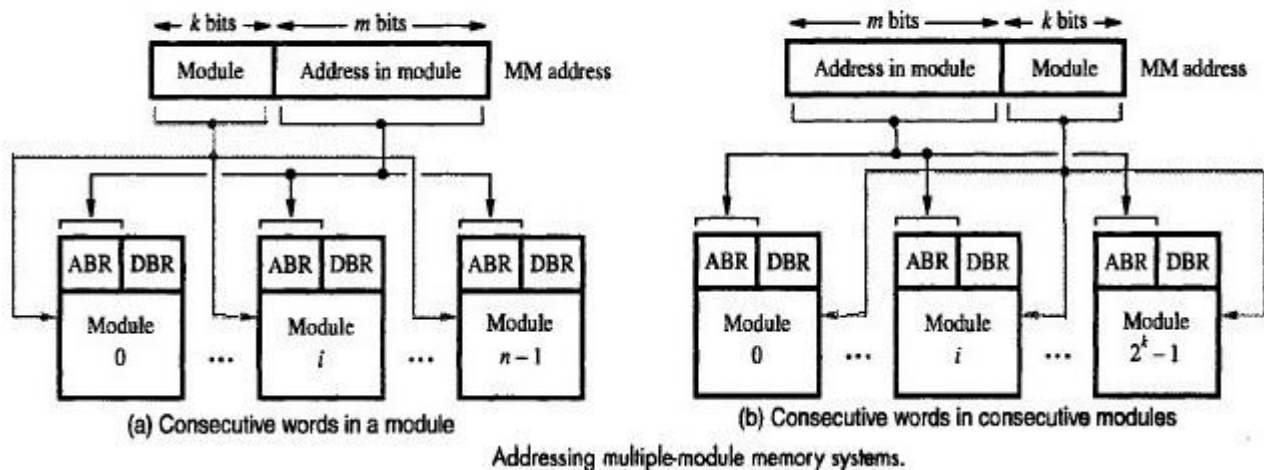
- 2 bit counter can be used for each block.
- When a ‘**hit**’ occurs, then block counter=0; The counter with values originally lower than the referenced one are incremented by 1 & all others remain unchanged.
- When a ‘**miss**’ occurs & if the set is full, the blocks with the counter value 3 is removed, the new block is put in its place & its counter is set to “0” and other block counters are incremented by 1.

PERFORMANCE CONSIDERATION

- Two key factors in the commercial success are 1) performance & 2) cost.
- In other words, the best possible performance at low cost.
- A common measure of success is called the **Price Performance ratio**.
- Performance depends on
 - how fast the machine instructions are brought to the processor &
 - how fast the machine instructions are executed.
- To achieve parallelism, *interleaving* is used.
- Parallelism means both the slow and fast units are accessed in the same manner.

INTERLEAVING

- The main-memory of a computer is structured as a collection of physically separate modules.
- Each module has its own
 - 1) ABR (address buffer register) &
 - 2) DBR (data buffer register).
- So, memory access operations may proceed in more than one module at the same time as shown in the figure



- Thus, the aggregate-rate of transmission of words to/from the main-memory can be increased.
- The low-order k -bits of the memory-address select a module.
 - While the high-order m -bits name a location within the module.
 - In this way, consecutive addresses are located in successive modules.
- Thus, any component of the system can keep several modules busy at any one time T .
- This results in both
 - faster access to a block of data and
 - higher average utilization of the memory-system as a whole.
- To implement the interleaved-structure, there must be 2^k modules;
 - Otherwise, there will be gaps of non-existent locations in the address-space.

Hit Rate & Miss Penalty

- The number of hits stated as a fraction of all attempted accesses is called the **Hit Rate**.
 - The extra time needed to bring the desired information into the cache is called the **Miss Penalty**.
 - High hit rates well over 0.9 are essential for high-performance computers.
 - Performance is adversely affected by the actions that need to be taken when a miss occurs.
 - A performance penalty is incurred because
 - of the extra time needed to bring a block of data from a slower unit to a faster unit.
 - During that period, the processor is stalled waiting for instructions or data.
 - We refer to the total access time seen by the processor when a miss occurs as the miss penalty.
- Let h be the hit rate, M the miss penalty, and C the time to access information in the cache. Thus, the average access time experienced by the processor is

$$t_{avg} = hC + (1 - h)M$$

- In high performance processors 2 levels of caches are normally used.
- Avg access time in a system with 2 levels of caches is

$$t_{ave} = h_1C_1 + (1 - h_1)h_2C_2 + (1 - h_1)(1 - h_2)M$$

where

h_1 is the hit rate in the L1 cache.

h_2 is the hit rate in the L2 cache.

C_1 is the time to access information in the L1 cache.

C_2 is the time to access information in the L2 cache.

M is the time to access information in the main memory.

3 Other Performance Enhancements

1. Write buffer

- **Write-through:**
 - Each write operation involves writing to the main memory.
 - If the processor has to wait for the write operation to be complete, it slows down the processor.
 - Processor does not depend on the results of the write operation.
 - Write buffer can be included for temporary storage of write requests.
 - Processor places each write request into the buffer and continues execution.
 - If a subsequent Read request references data which is still in the write buffer, then this data is referenced in the write buffer.
- **Write-back:**
 - Block is written back to the main memory when it is replaced.
 - If the processor waits for this write to complete, before reading the new block, it is slowed down.
 - Fast write buffer can hold the block to be written, and the new block can be read first.

2. Prefetching

- New data are brought into the processor when they are first needed.
- Processor has to wait before the data transfer is complete.
- Prefetch the data into the cache before they are actually needed, or a before a Read miss occurs.
- Prefetching can be accomplished through software by including a special instruction in the machine language of the processor.
 - Inclusion of prefetch instructions increases the length of the programs.
- Prefetching can also be accomplished using hardware:
 - Circuitry that attempts to discover patterns in memory references and then prefetches according to this pattern.

3. Lockup-Free Cache

- Prefetching scheme does not work if it stops other accesses to the cache until the prefetch is completed.
- A cache of this type is said to be “locked” while it services a miss.
- Cache structure which supports multiple outstanding misses is called a lockup free cache.
- Since only one miss can be serviced at a time, a lockup free cache must include circuits that keep track of all the outstanding misses.
- Special registers may hold the necessary information about these misses.

MODULE 4: ARITHMETIC

NUMBERS, ARITHMETIC OPERATIONS AND CHARACTERS

NUMBER REPRESENTATION

- Numbers can be represented in 3 formats:
 - 1) Sign and magnitude
 - 2) 1's complement
 - 3) 2's complement
- In all three formats, MSB=0 for +ve numbers & MSB=1 for -ve numbers.
- In **sign-and-magnitude system**,
 - negative value is obtained by changing the MSB from 0 to 1 of the corresponding positive value.
 - For ex, +5 is represented by 0101 & -5 is represented by 1101.
- In **1's complement system**,
 - negative values are obtained by complementing each bit of the corresponding positive number.
 - For ex, -5 is obtained by complementing each bit in 0101 to yield 1010.
 - (In other words, the operation of forming the 1's complement of a given number is equivalent to subtracting that number from 2^n-1).
- In **2's complement system**,
 - forming the 2's complement of a number is done by subtracting that number from 2^n .
 - For ex, -5 is obtained by complementing each bit in 0101 & then adding 1 to yield 1011.
 - (In other words, the 2's complement of a number is obtained by adding 1 to the 1's complement of that number).
- 2's complement system yields the most efficient way to carry out addition/subtraction operations.

B $b_3 b_2 b_1 b_0$	Values represented		
	Sign and magnitude	1's complement	2's complement
0 1 1 1	+7	+7	+7
0 1 1 0	+6	+6	+6
0 1 0 1	+5	+5	+5
0 1 0 0	+4	+4	+4
0 0 1 1	+3	+3	+3
0 0 1 0	+2	+2	+2
0 0 0 1	+1	+1	+1
0 0 0 0	+0	+0	+0
1 0 0 0	-0	-7	-8
1 0 0 1	-1	-6	-7
1 0 1 0	-2	-5	-6
1 0 1 1	-3	-4	-5
1 1 0 0	-4	-3	-4
1 1 0 1	-5	-2	-3
1 1 1 0	-6	-1	-2
1 1 1 1	-7	-0	-1

Figure 1.3 Binary, signed-integer representations.

ADDITION OF POSITIVE NUMBERS

- Consider adding two 1-bit numbers.
- The sum of 1 & 1 requires the 2-bit vector 10 to represent the value 2. We say that sum is 0 and the carry-out is 1.

$$\begin{array}{cccc}
 \begin{array}{r} 0 \\ + 0 \\ \hline 0 \end{array} &
 \begin{array}{r} 1 \\ + 0 \\ \hline 1 \end{array} &
 \begin{array}{r} 0 \\ + 1 \\ \hline 1 \end{array} &
 \begin{array}{r} 1 \\ + 1 \\ \hline 10 \end{array} \\
 & & & \uparrow \\
 & & & \text{Carry-out}
 \end{array}$$

Figure 2.2 Addition of 1-bit numbers.

COMPUTER ORGANIZATION

ADDITION & SUBTRACTION OF SIGNED NUMBERS

- Following are the two rules for addition and subtraction of n-bit signed numbers using the 2's complement representation system (Figure 1.6).

Rule 1:

- **To Add** two numbers, add their n-bits and ignore the carry-out signal from the MSB position.
- Result will be algebraically correct, if it lies in the range -2^{n-1} to $+2^{n-1}-1$.

Rule 2:

- **To Subtract** two numbers X and Y (that is to perform $X-Y$), take the 2's complement of Y and then add it to X as in rule 1.
- Result will be algebraically correct, if it lies in the range (-2^{n-1}) to $+(2^{n-1}-1)$.

- When the result of an arithmetic operation is outside the representable-range, an arithmetic overflow is said to occur.
- To represent a signed in 2's complement form using a larger number of bits, repeat the sign bit as many times as needed to the left. This operation is called **sign extension**.
- In 1's complement representation, the result obtained after an addition operation is not always correct. The carry-out(c_n) cannot be ignored. If $c_n=0$, the result obtained is correct. If $c_n=1$, then a 1 must be added to the result to make it correct.

OVERFLOW IN INTEGER ARITHMETIC

- When result of an arithmetic operation is outside the representable-range, an **arithmetic overflow** is said to occur.
- For example: If we add two numbers +7 and +4, then the output sum S is 1011($\leftarrow 0111+0100$), which is the code for -5, an incorrect result.
- An overflow occurs in following 2 cases
 - 1) Overflow can occur only when adding two numbers that have the same sign.
 - 2) The carry-out signal from the sign-bit position is not a sufficient indicator of overflow when adding signed numbers.

(a)	$\begin{array}{r} 0010 \\ + 0011 \\ \hline 0101 \end{array}$	$\begin{array}{r} (+2) \\ (+3) \\ \hline (+5) \end{array}$	(b)	$\begin{array}{r} 0100 \\ + 1010 \\ \hline 1110 \end{array}$	$\begin{array}{r} (+4) \\ (-6) \\ \hline (-2) \end{array}$
(c)	$\begin{array}{r} 1011 \\ + 1110 \\ \hline 1001 \end{array}$	$\begin{array}{r} (-5) \\ (-2) \\ \hline (-7) \end{array}$	(d)	$\begin{array}{r} 0111 \\ + 1101 \\ \hline 0100 \end{array}$	$\begin{array}{r} (+7) \\ (-3) \\ \hline (+4) \end{array}$
(e)	$\begin{array}{r} 1101 \\ - 1001 \\ \hline \end{array}$	$\begin{array}{r} (-3) \\ (-7) \\ \hline \end{array}$	\Rightarrow	$\begin{array}{r} 1101 \\ + 0111 \\ \hline 0100 \end{array}$	$\begin{array}{r} \\ \\ \hline (+4) \end{array}$
(f)	$\begin{array}{r} 0010 \\ - 0100 \\ \hline \end{array}$	$\begin{array}{r} (+2) \\ (+4) \\ \hline \end{array}$	\Rightarrow	$\begin{array}{r} 0010 \\ + 1100 \\ \hline 1110 \end{array}$	$\begin{array}{r} \\ \\ \hline (-2) \end{array}$
(g)	$\begin{array}{r} 0110 \\ - 0011 \\ \hline \end{array}$	$\begin{array}{r} (+6) \\ (+3) \\ \hline \end{array}$	\Rightarrow	$\begin{array}{r} 0110 \\ + 1101 \\ \hline 0011 \end{array}$	$\begin{array}{r} \\ \\ \hline (+3) \end{array}$
(h)	$\begin{array}{r} 1001 \\ - 1011 \\ \hline \end{array}$	$\begin{array}{r} (-7) \\ (-5) \\ \hline \end{array}$	\Rightarrow	$\begin{array}{r} 1001 \\ + 0101 \\ \hline 1110 \end{array}$	$\begin{array}{r} \\ \\ \hline (-2) \end{array}$
(i)	$\begin{array}{r} 1001 \\ - 0001 \\ \hline \end{array}$	$\begin{array}{r} (-7) \\ (+1) \\ \hline \end{array}$	\Rightarrow	$\begin{array}{r} 1001 \\ + 1111 \\ \hline 1000 \end{array}$	$\begin{array}{r} \\ \\ \hline (-8) \end{array}$
(j)	$\begin{array}{r} 0010 \\ - 1101 \\ \hline \end{array}$	$\begin{array}{r} (+2) \\ (-3) \\ \hline \end{array}$	\Rightarrow	$\begin{array}{r} 0010 \\ + 0011 \\ \hline 0101 \end{array}$	$\begin{array}{r} \\ \\ \hline (+5) \end{array}$

Figure 1.6 2's-complement Add and Subtract operations.

ADDITION & SUBTRACTION OF SIGNED NUMBERS

n-BIT RIPPLE CARRY ADDER

- A cascaded connection of n full-adder blocks can be used to add 2-bit numbers.
- Since carries must propagate (or ripple) through cascade, the configuration is called an n-bit ripple carry adder (Figure 9.1).

x_i	y_i	Carry-in c_i	Sum s_i	Carry-out c_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$s_i = \overline{x_i}\overline{y_i}c_i + \overline{x_i}y_i\overline{c_i} + x_i\overline{y_i}\overline{c_i} + x_iy_ic_i = x_i \oplus y_i \oplus c_i$$

$$c_{i+1} = y_ic_i + x_ic_i + x_iy_i$$

Example:

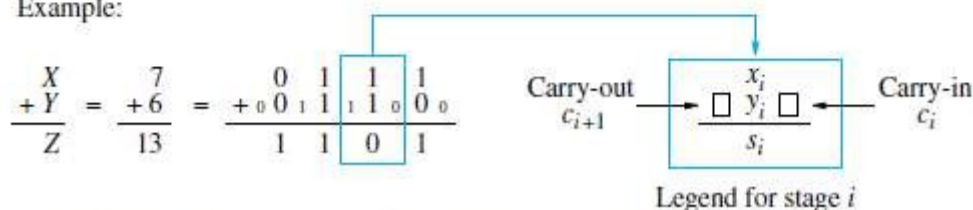


Figure 9.1 Logic specification for a stage of binary addition.

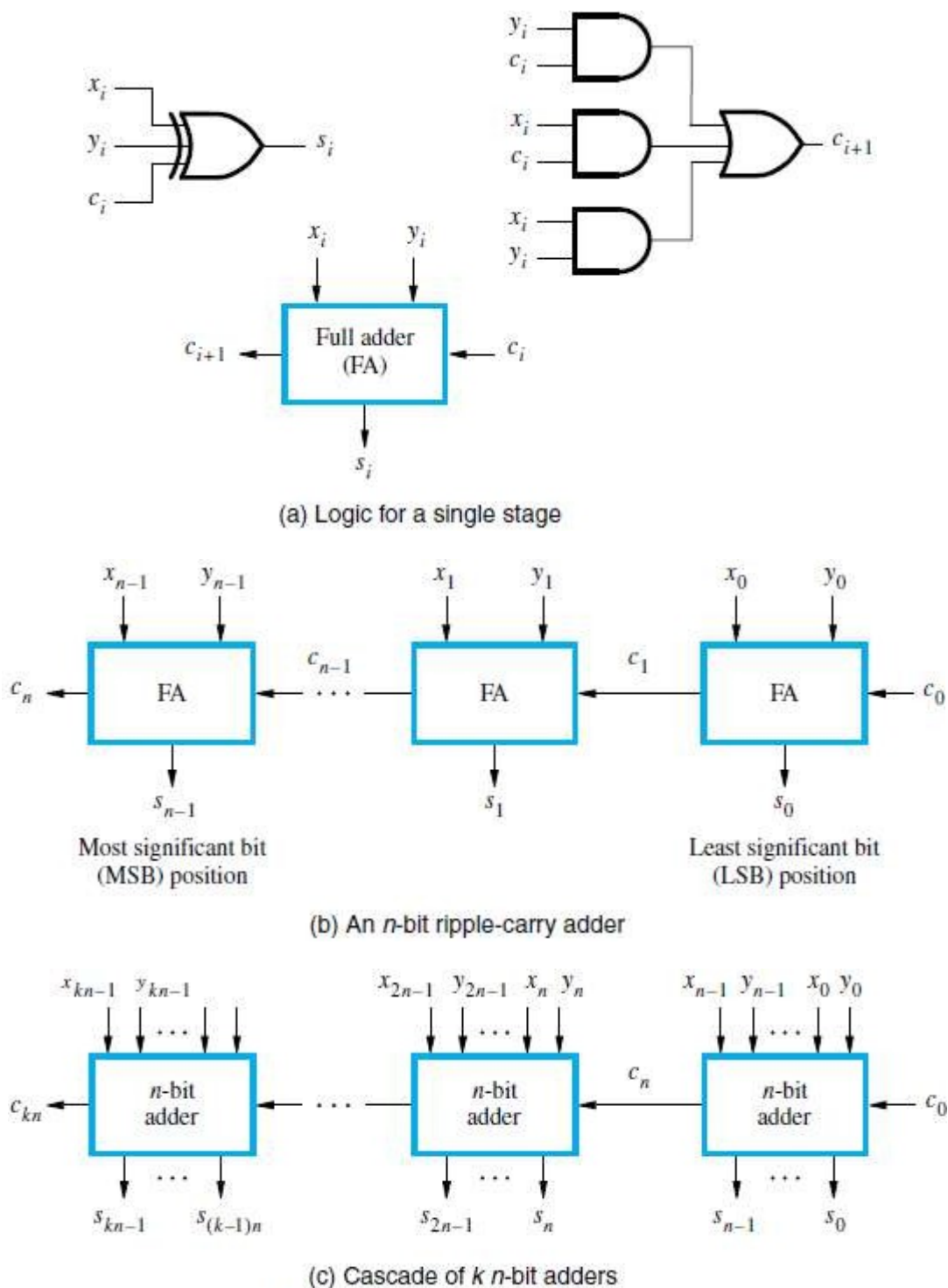


Figure 9.2 Logic for addition of binary numbers.

COMPUTER ORGANIZATION

ADDITION/SUBTRACTION LOGIC UNIT

- The n -bit adder can be used to add 2's complement numbers X and Y (Figure 9.3).
- **Overflow** can only occur when the signs of the 2 operands are the same.
- In order to perform the subtraction operation $X-Y$ on 2's complement numbers X and Y ; we form the 2's complement of Y and add it to X .
- Addition or subtraction operation is done based on value applied to the Add/Sub input control-line.
- Control-line=0 for addition, applying the Y vector unchanged to one of the adder inputs.
- Control-line=1 for subtraction, the Y vector is 2's complemented.

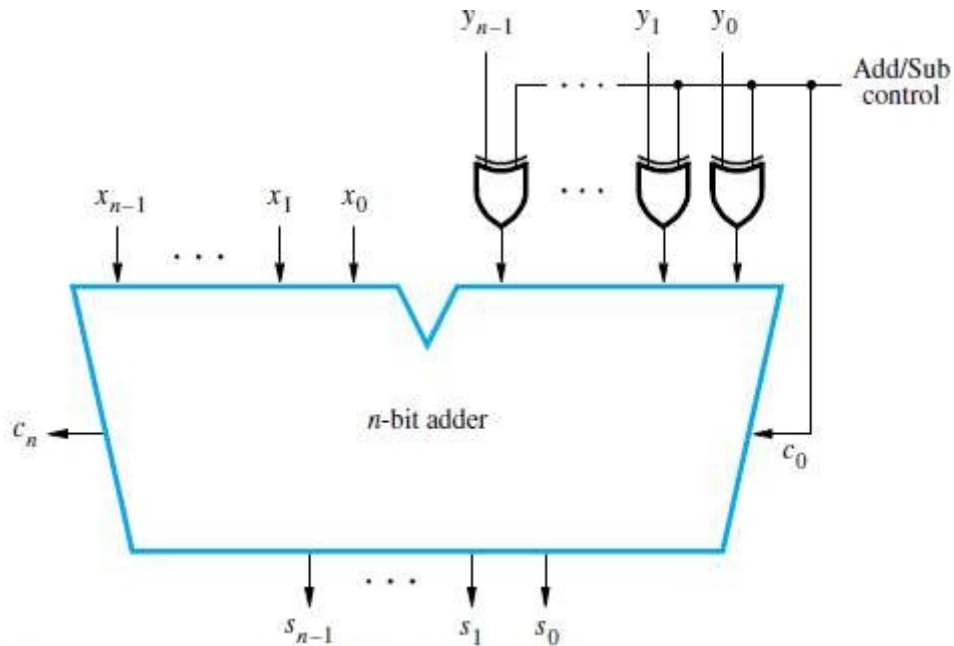


Figure 9.3 Binary addition/subtraction logic circuit.

DESIGN OF FAST ADDERS

- **Drawback of ripple carry adder:** If the adder is used to implement the addition/subtraction, all sum bits are available in $2n$ gate delays.
- Two approaches can be used to reduce delay in adders:
 - 1) Use the fastest possible electronic-technology in implementing the ripple-carry design.
 - 2) Use an augmented logic-gate network structure.

COMPUTER ORGANIZATION

CARRY-LOOKAHEAD ADDITIONS

- The logic expression for s_i (sum) and c_{i+1} (carry-out) of stage i are

$$s_i = x_i + y_i + c_i \quad \text{-----(1)}$$

$$c_{i+1} = x_i y_i + x_i c_i + y_i c_i \quad \text{-----(2)}$$

- Factoring (2) into

$$c_{i+1} = x_i y_i + (x_i + y_i) c_i$$

we can write

$$c_{i+1} = G_i + P_i c_i \quad \text{where } G_i = x_i y_i \text{ and } P_i = x_i + y_i$$

- The expressions G_i and P_i are called generate and propagate functions (Figure 9.4).
- If $G_i = 1$, then $c_{i+1} = 1$, independent of the input carry c_i . This occurs when both x_i and y_i are 1. Propagate function means that an input-carry will produce an output-carry when either $x_i = 1$ or $y_i = 1$.
- All G_i and P_i functions can be formed independently and in parallel in one logic-gate delay.
- Expanding c_i terms of $i-1$ subscripted variables and substituting into the c_{i+1} expression, we obtain

$$c_{i+1} = G_i + P_i G_{i-1} + P_i P_{i-1} G_{i-2} + \dots + P_i P_{i-1} \dots P_0 c_0$$
- Conclusion: Delay through the adder is 3 gate delays for all carry-bits & 4 gate delays for all sum-bits.
- Consider the design of a 4-bit adder. The carries can be implemented as

$$c_1 = G_0 + P_0 c_0$$

$$c_2 = G_1 + P_1 G_0 + P_1 P_0 c_0$$

$$c_3 = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 c_0$$

$$c_4 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 c_0$$

The carries are implemented in the block labeled carry-lookahead logic. An adder implemented in this form is called a **Carry-Lookahead Adder**.

- Limitation: If we try to extend the carry-lookahead adder for longer operands, we run into a problem of gate fan-in constraints.

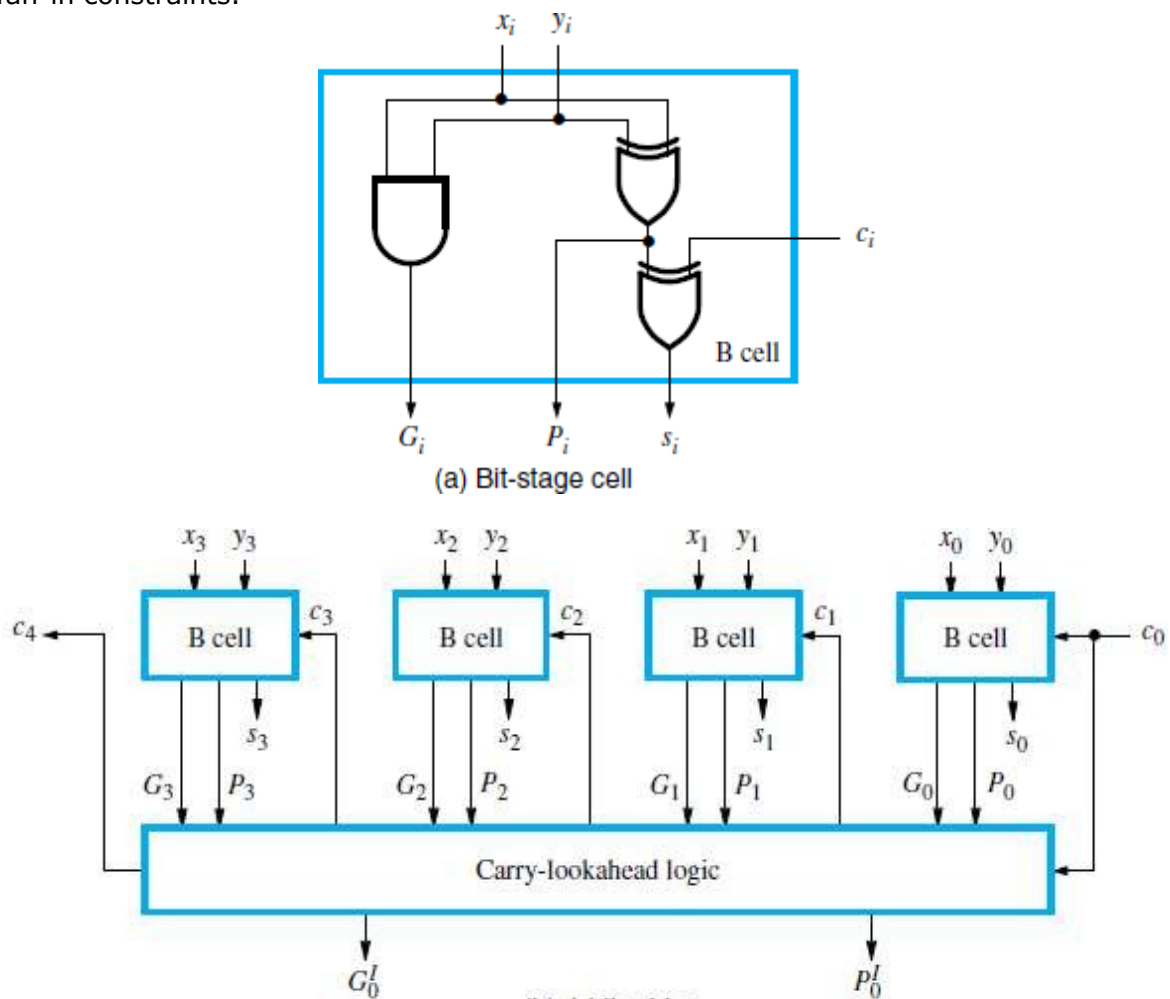


Figure 9.4 A 4-bit carry-lookahead adder.

COMPUTER ORGANIZATION

HIGHER-LEVEL GENERATE & PROPAGATE FUNCTIONS

- 16-bit adder can be built from four 4-bit adder blocks (Figure 9.5).
- These blocks provide new output functions defined as G_k and P_k , where $k=0$ for the first 4-bit block, $k=1$ for the second 4-bit block and so on.
- In the first block,
$$P_0 = P_3 P_2 P_1 P_0$$
$$\&$$
$$G_0 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0$$
- The first-level G_i and P_i functions determine whether bit stage i generates or propagates a carry, and the second level G_k and P_k functions determine whether block k generates or propagates a carry.
- Carry c_{16} is formed by one of the carry-lookahead circuits as
$$c_{16} = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 c_0$$
- Conclusion: All carries are available 5 gate delays after X , Y and c_0 are applied as inputs.

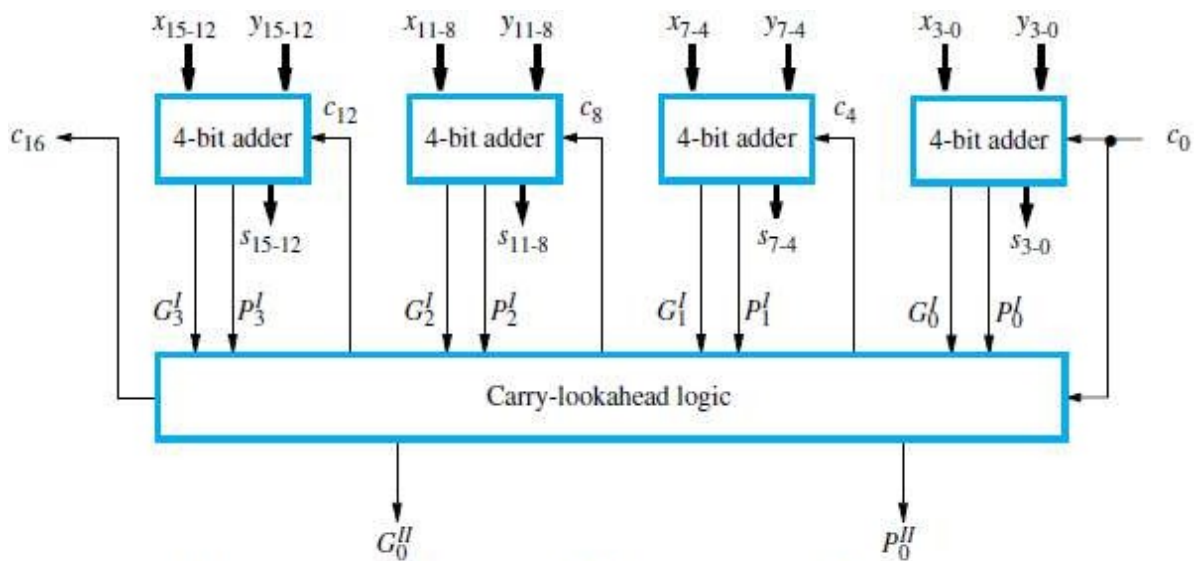


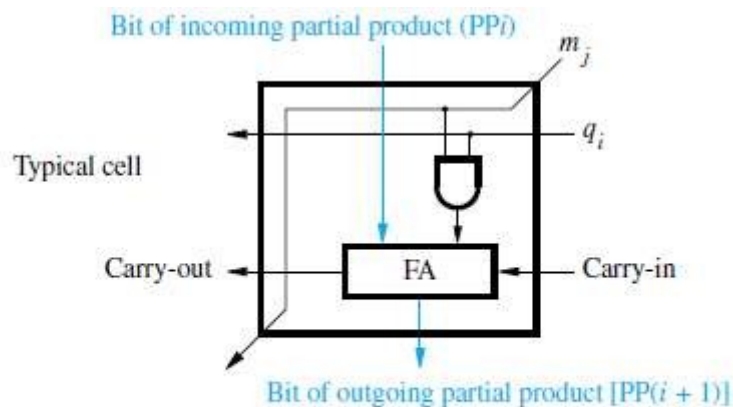
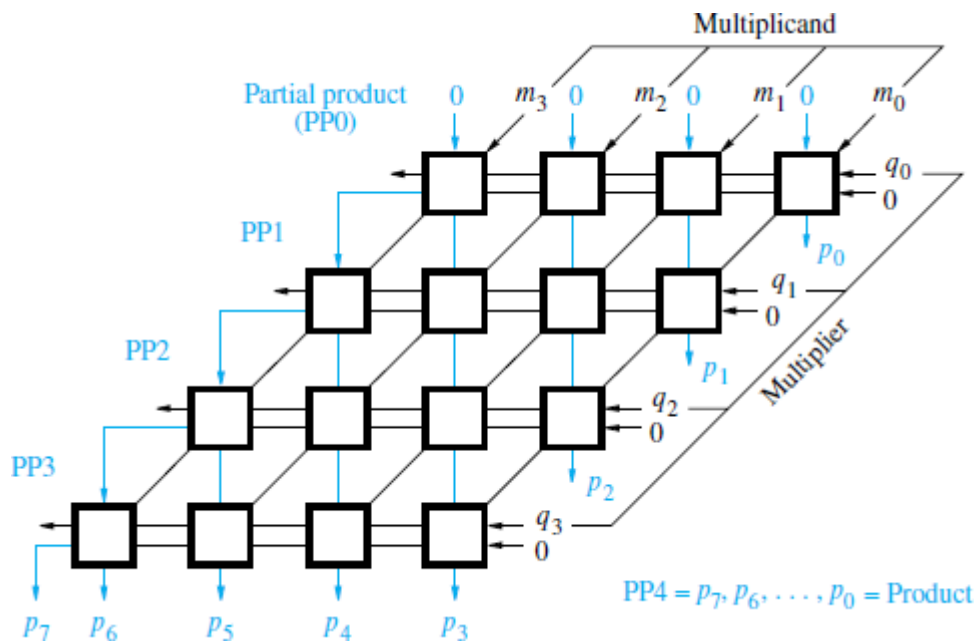
Figure 9.5 A 16-bit carry-lookahead adder built from 4-bit adders (see Figure 9.4b).

MULTIPLICATION OF POSITIVE NUMBERS

$$\begin{array}{r}
 1101 \\
 \times 1011 \\
 \hline
 1101 \\
 0000 \\
 1101 \\
 1000 \\
 \hline
 10001111
 \end{array}$$

(13) Multiplicand M
(11) Multiplier Q
(143) Product P

(a) Manual multiplication algorithm



(b) Array implementation

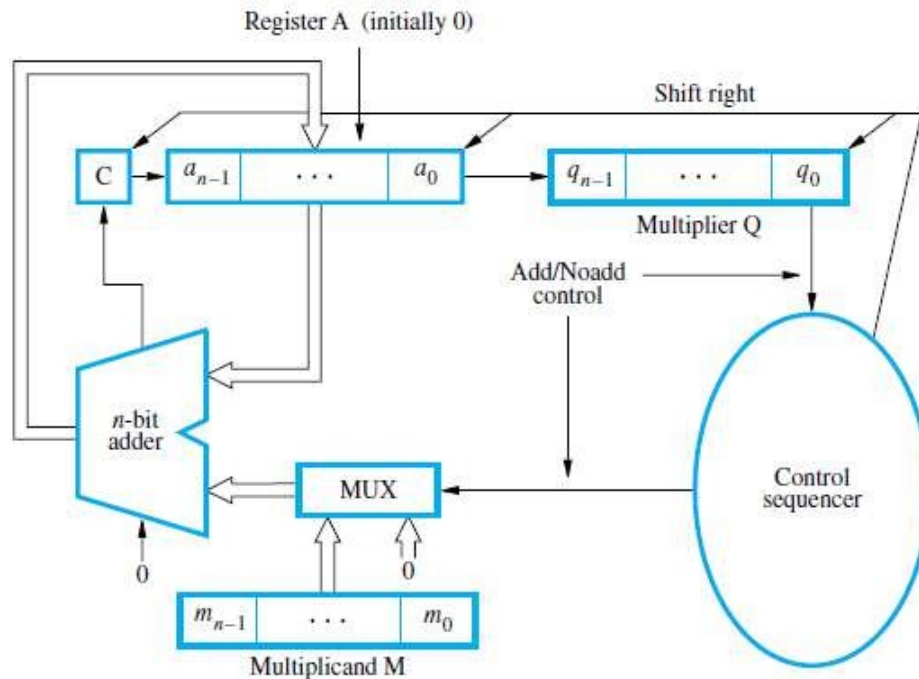
Figure 9.6 Array multiplication of unsigned binary operands.

ARRAY MULTIPLICATION

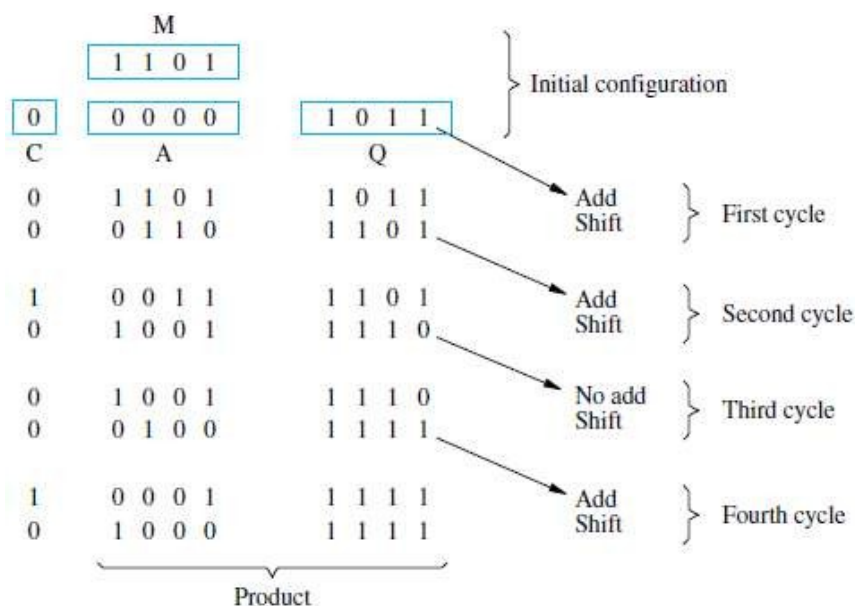
- The main component in each cell is a full adder(FA)..
- The AND gate in each cell determines whether a multiplicand bit m_j , is added to the incoming partial-product bit, based on the value of the multiplier bit q_i (Figure 9.6).

SEQUENTIAL CIRCUIT BINARY MULTIPLIER

- Registers A and Q combined hold PP (partial product) while the multiplier bit q_i generates the signal Add/Noadd.
- The carry-out from the adder is stored in flip-flop C (Figure 9.7).
- Procedure for multiplication:
 - Multiplier is loaded into register Q, Multiplicand is loaded into register M and C & A are cleared to 0.
 - If $q_0=1$, add M to A and store sum in A. Then C, A and Q are shifted right one bit-position. If $q_0=0$, no addition performed and C, A & Q are shifted right one bit-position.
 - After n cycles, the high-order half of the product is held in register A and the low-order half is held in register Q.



(a) Register configuration



(b) Multiplication example

Figure 9.7 Sequential circuit binary multiplier.

COMPUTER ORGANIZATION

SIGNED OPERAND MULTIPLICATION BOOTH ALGORITHM

- This algorithm
 - generates a 2n-bit product
 - treats both positive & negative 2's-complement n-bit operands uniformly (Figure 9.9-9.12).
- Attractive feature: This algorithm achieves some efficiency in the number of addition required when the multiplier has a few large blocks of 1s.
- This algorithm suggests that we can reduce the number of operations required for multiplication by representing multiplier as a difference between 2 numbers.

For e.g. multiplier (Q) 14(001110) can be represented as

010000 (16)

-000010 (2)

001110 (14)

- Therefore, product $P = M * Q$ can be computed by adding 2^4 times the M to the 2's complement of 2^1 times the M.

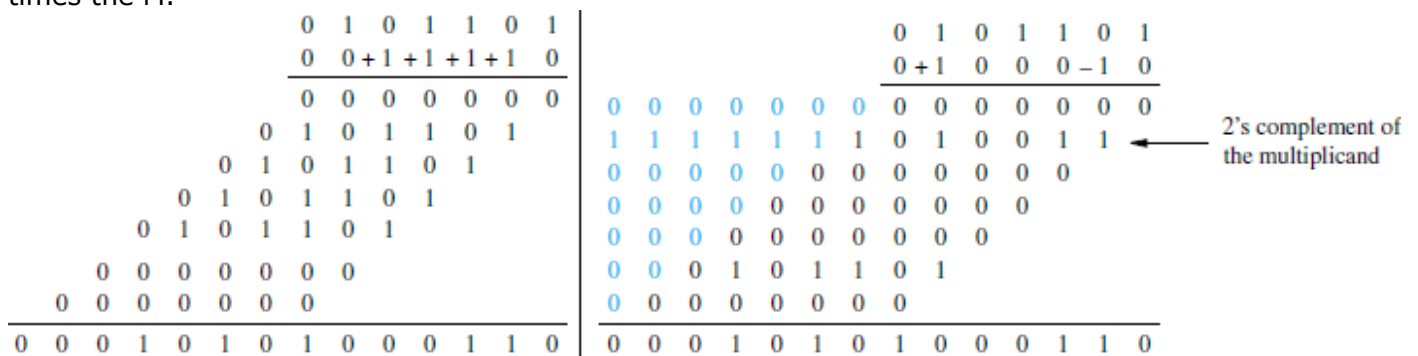


Figure 9.9 Normal and Booth multiplication schemes.

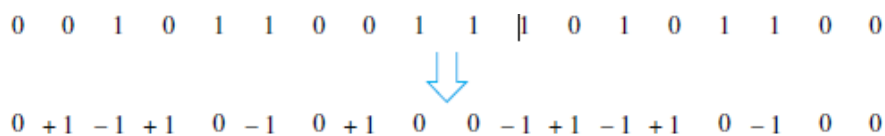


Figure 9.10 Booth recoding of a multiplier.

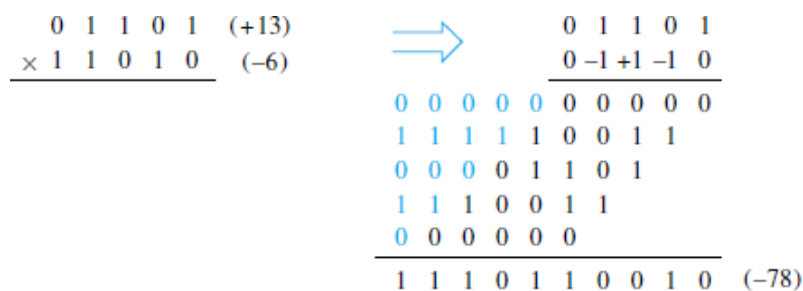


Figure 9.11 Booth multiplication with a negative multiplier.

Multiplier		Version of multiplicand selected by bit i
Bit i	Bit $i-1$	
0	0	$0 \times M$
0	1	$+1 \times M$
1	0	$-1 \times M$
1	1	$0 \times M$

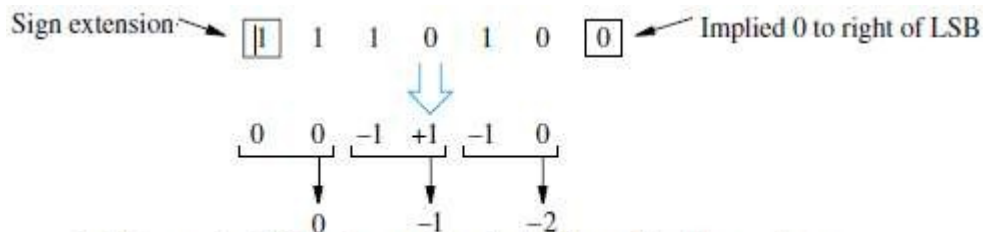
Figure 9.12 Booth multiplier recoding table.

COMPUTER ORGANIZATION

FAST MULTIPLICATION

BIT-PAIR RECODING OF MULTIPLIERS

- This method
 - derived from the booth algorithm
 - reduces the number of summands by a factor of 2
- Group the Booth-recoded multiplier bits in pairs. (Figure 9.14 & 9.15).
- The pair (+1 -1) is equivalent to the pair (0 +1).



(a) Example of bit-pair recoding derived from Booth recoding

Multiplier bit-pair		Multiplier bit on the right	Multiplicand selected at position i
$i+1$	i	$i-1$	
0	0	0	$0 \times M$
0	0	1	$+1 \times M$
0	1	0	$+1 \times M$
0	1	1	$+2 \times M$
1	0	0	$-2 \times M$
1	0	1	$-1 \times M$
1	1	0	$-1 \times M$
1	1	1	$0 \times M$

(b) Table of multiplicand selection decisions

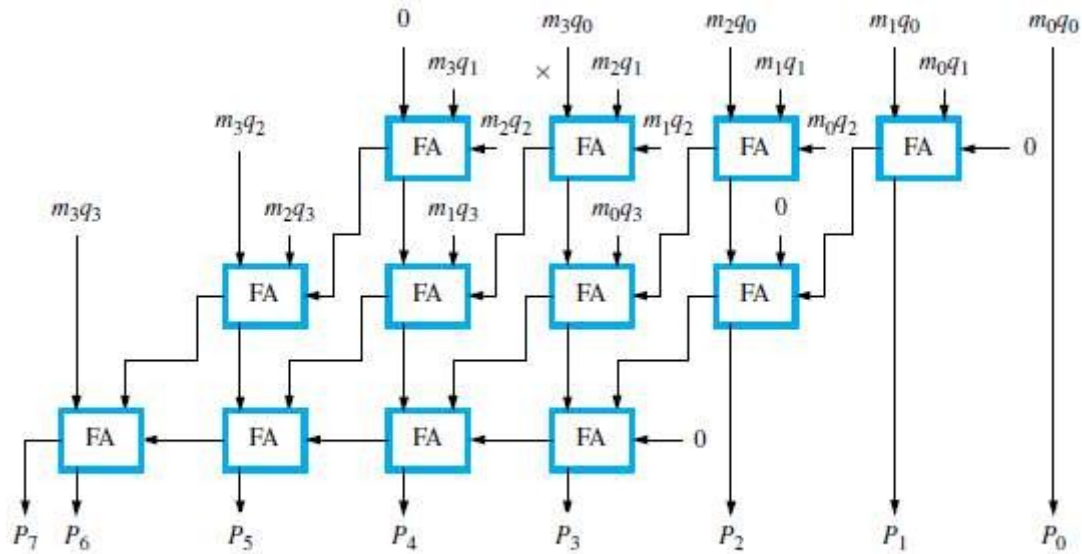
Figure 9.14 Multiplier bit-pair recoding.

$$\begin{array}{r}
 \begin{array}{cccccc} 0 & 1 & 1 & 0 & 1 & (+13) \\ \times & 1 & 1 & 0 & 1 & 0 & (-6) \\ \hline \end{array} \\
 \Downarrow \\
 \begin{array}{r}
 \begin{array}{cccccc} 0 & 1 & 1 & 0 & 1 \\ 0 & -1 & +1 & -1 & 0 \end{array} \\
 \hline
 \begin{array}{cccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 & & \\ 0 & 0 & 0 & 0 & 0 & 0 & & & \\ \hline 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & (-78) \end{array} \\
 \Downarrow \\
 \begin{array}{r}
 \begin{array}{cccccc} 0 & 1 & 1 & 0 & 1 \\ 0 & -1 & -2 \end{array} \\
 \hline
 \begin{array}{cccccccc} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & & \\ 0 & 0 & 0 & 0 & 0 & 0 & & & & \\ \hline 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \end{array}
 \end{array}$$

Figure 9.15 Multiplication requiring only $n/2$ summands.

CARRY-SAVE ADDITION OF SUMMANDS

- Consider the array for 4*4 multiplication. (Figure 9.16 & 9.18).
- Instead of letting the carries ripple along the rows, they can be "saved" and introduced into the next row, at the correct weighted positions.



(b) Carry-save array

Figure 9.16 carry-save arrays for a 4×4 multiplier.

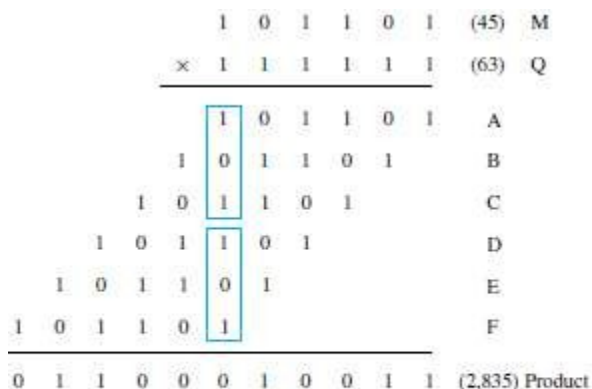


Figure 9.17 A multiplication example used to illustrate carry-save addition as shown in Figure 9.18.

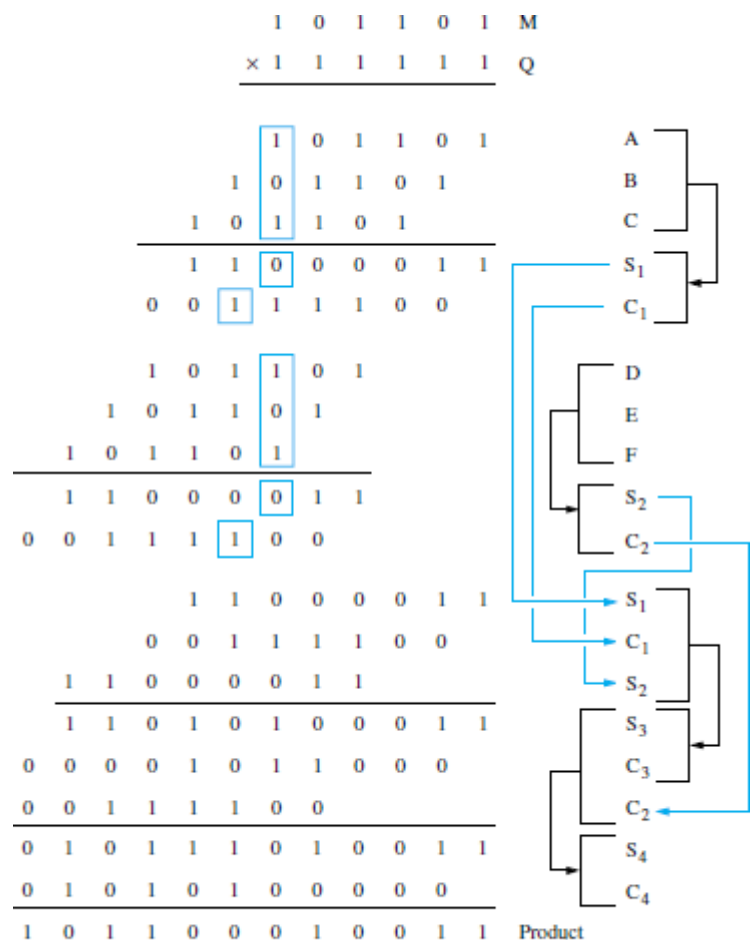


Figure 9.18 The multiplication example from Figure 9.17 performed using carry-save addition.

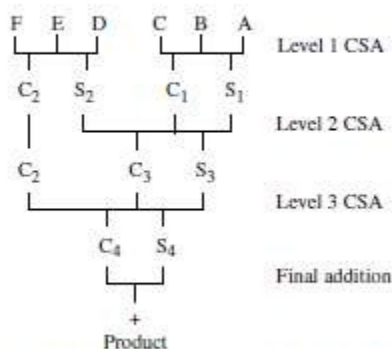


Figure 9.19 Schematic representation of the carry-save addition operations in Figure 9.18.

COMPUTER ORGANIZATION

- The full adder is input with three partial bit products in the first row.
- Multiplication requires the addition of several summands.
- CSA speeds up the addition process.
- Consider the array for 4x4 multiplication shown in fig 9.16.
- First row consisting of just the AND gates that implement the bit products m_3q_0 , m_2q_0 , m_1q_0 and m_0q_0 .
- The delay through the carry-save array is somewhat less than delay through the ripple-carry array. This is because the S and C vector outputs from each row are produced in parallel in one full-adder delay.
- Consider the addition of many summands in fig 9.18.
- Group the summands in threes and perform carry-save addition on each of these groups in parallel to generate a set of S and C vectors in one full-adder delay
- Group all of the S and C vectors into threes, and perform carry-save addition on them, generating a further set of S and C vectors in one more full-adder delay
- Continue with this process until there are only two vectors remaining
- They can be added in a RCA or CLA to produce the desired product.
- When the number of summands is large, the time saved is proportionally much greater.
- Delay: AND gate + 2 gate/CSA level + CLA gate delay, Eg., 6 bit number require 15 gate delay, array 6x6 require $6(n-1)-1 = 29$ gate Delay.
- In general, CSA takes $1.7 \log_2 k - 1.7$ levels of CSA to reduce k summands.

COMPUTER ORGANIZATION

INTEGER DIVISION

- An n -bit positive-divisor is loaded into register M.
An n -bit positive-dividend is loaded into register Q at the start of the operation.
Register A is set to 0 (Figure 9.21).
- After division operation, the n -bit quotient is in register Q, and the remainder is in register A.

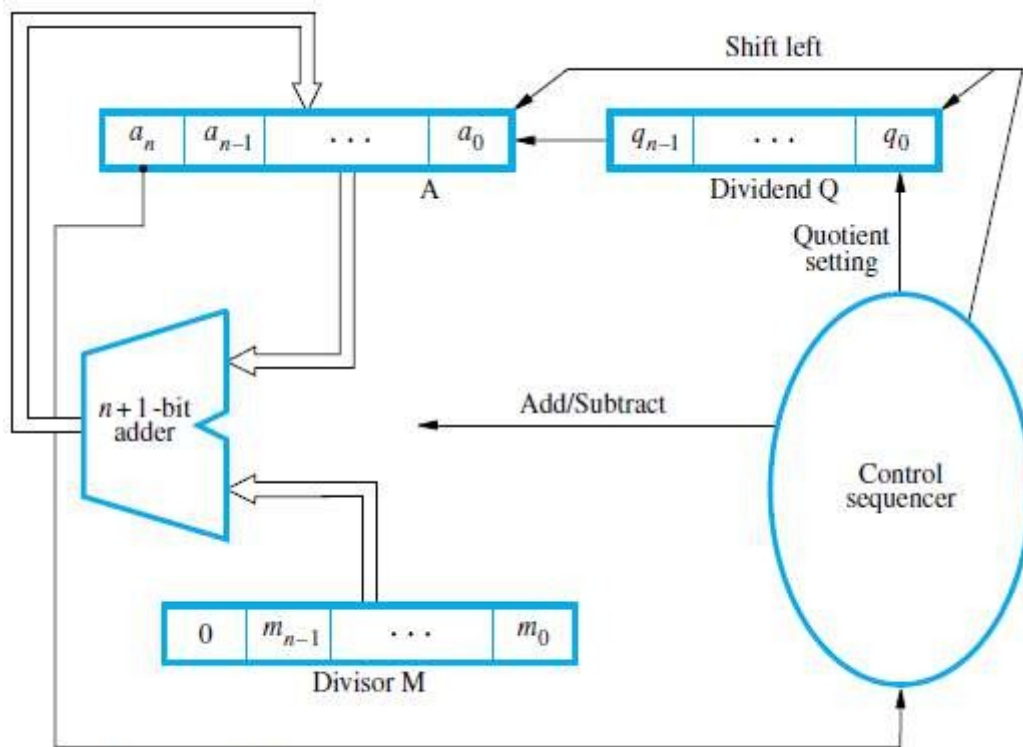


Figure 9.23 Circuit arrangement for binary division.

$$\begin{array}{r} 21 \\ 13 \overline{) 274} \\ \underline{26} \\ 14 \\ \underline{13} \\ 1 \end{array} \qquad \begin{array}{r} 10101 \\ 1101 \overline{) 100010010} \\ \underline{1101} \\ 10000 \\ \underline{1101} \\ 1110 \\ \underline{1101} \\ 1 \end{array}$$

Figure 9.22 Longhand division examples.

NON-RESTORING DIVISION

• Procedure:

Step 1: Do the following n times

- i) If the sign of A is 0, shift A and Q left one bit position and subtract M from A; otherwise, shift A and Q left and add M to A (Figure 9.23).
- ii) Now, if the sign of A is 0, set q_0 to 1; otherwise set q_0 to 0.

Step 2: If the sign of A is 1, add M to A (restore).

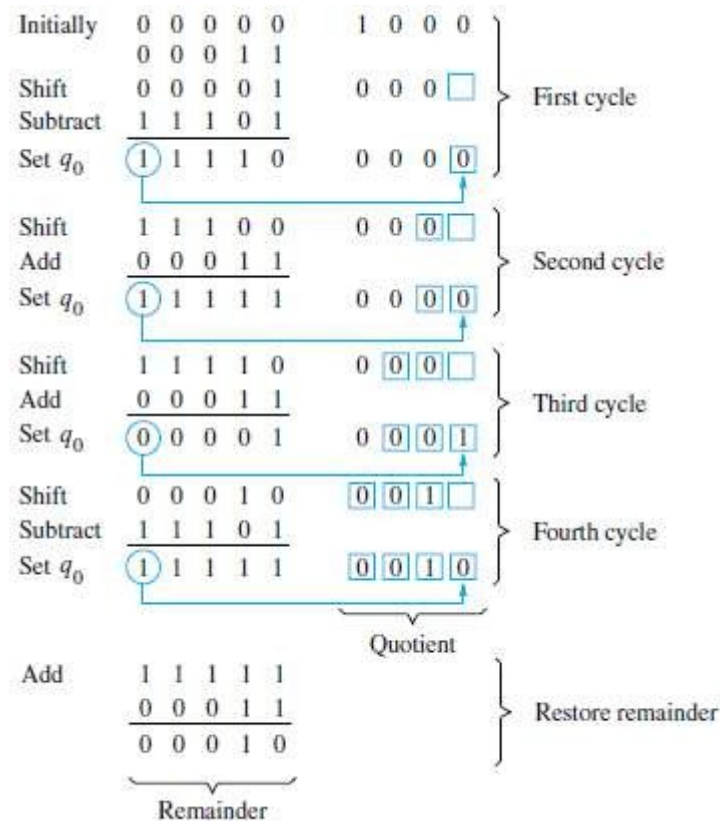


Figure 9.25 A non-restoring division example.

COMPUTER ORGANIZATION

RESTORING DIVISION

- Procedure: Do the following n times
 - 1) Shift A and Q left one binary position (Figure 9.22).
 - 2) Subtract M from A, and place the answer back in A
 - 3) If the sign of A is 1, set q_0 to 0 and add M back to A (restore A).
If the sign of A is 0, set q_0 to 1 and no restoring done.

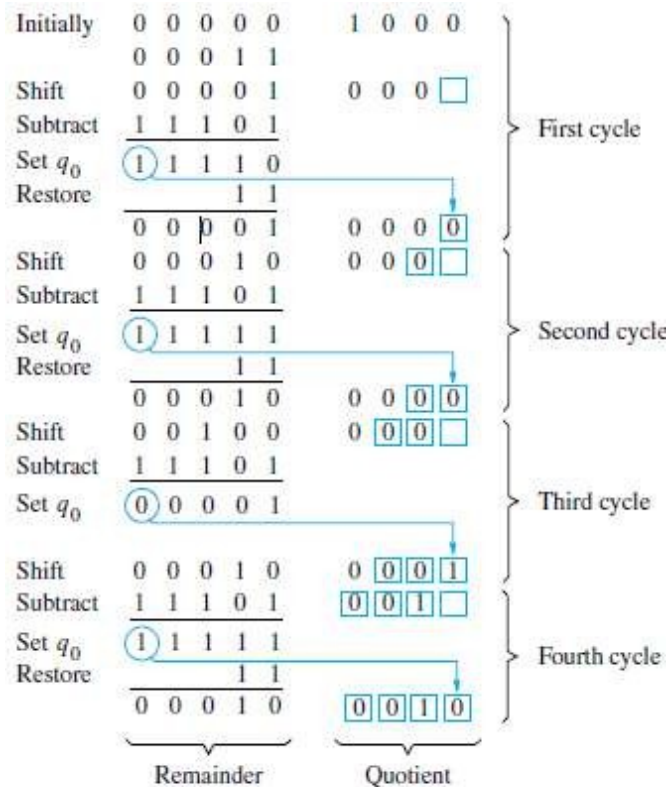


Figure 9.24 A restoring division example.

FLOATING-POINT NUMBERS & OPERATIONS

IEEE STANDARD FOR FLOATING POINT NUMBERS

- Single precision representation occupies a single 32-bit word.
The scale factor has a range of 2^{-126} to 2^{+127} (which is approximately equal to 10^{+38}).
- The 32 bit word is divided into 3 fields: sign(1 bit), exponent(8 bits) and mantissa(23 bits).
- Signed exponent= E .
Unsigned exponent $E'=E+127$. Thus, E' is in the range $0 < E' < 255$.
- The last 23 bits represent the mantissa. Since binary normalization is used, the MSB of the mantissa is always equal to 1. (M represents fractional-part).
- The 24-bit mantissa provides a precision equivalent to about 7 decimal-digits (Figure 9.24).
- Double precision representation occupies a single 64-bit word. And E' is in the range $1 < E' < 2046$.
- The 53-bit mantissa provides a precision equivalent to about 16 decimal-digits.

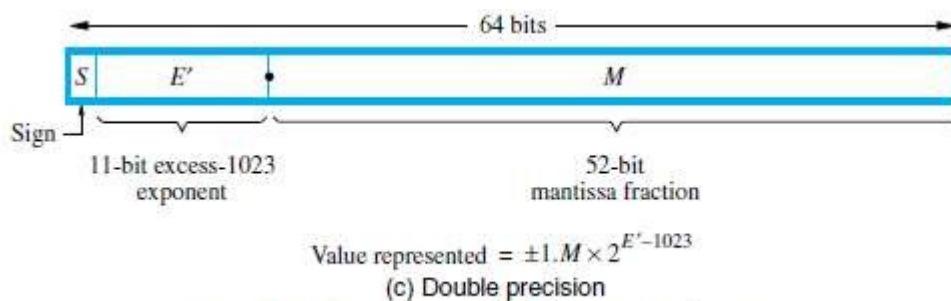
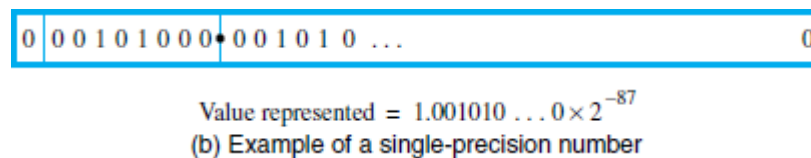
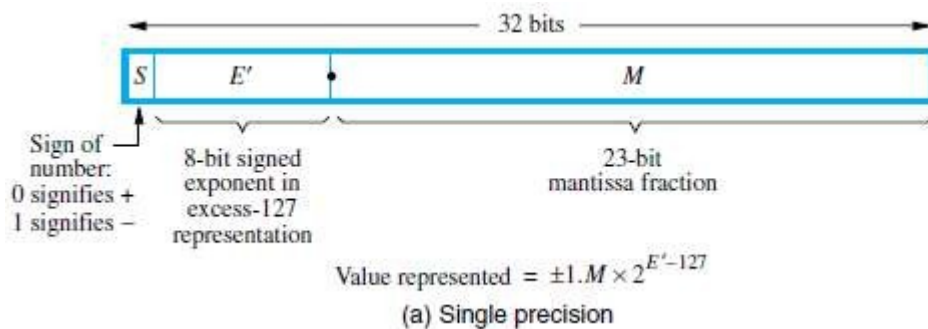


Figure 9.26 IEEE standard floating-point formats.

COMPUTER ORGANIZATION

NORMALIZATION

- When the decimal point is placed to the right of the first(non zero) significant digit, the number is said to be normalized.
- If a number is not normalized, it can always be put in normalized form by shifting the fraction and adjusting the exponent. As computations proceed, a number that does not fall in the representable range of normal numbers might be generated.
- In single precision, it requires an exponent less than -126 (underflow) or greater than +127 (overflow). Both are exceptions that need to be considered.

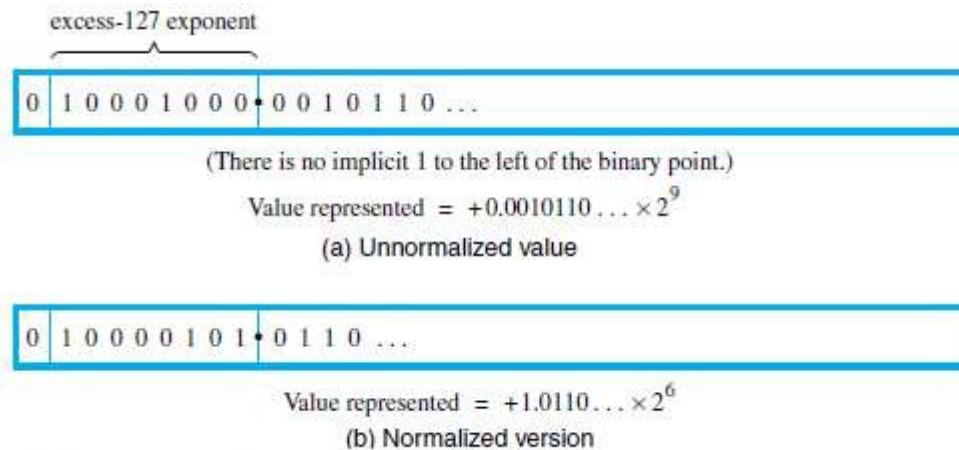


Figure 9.27 Floating-point normalization in IEEE single-precision format.

SPECIAL VALUES

- The end values 0 and 255 of the excess-127 exponent E' are used to represent special values.
- When $E'=0$ and the mantissa fraction m is zero, the value exact 0 is represented.
- When $E'=255$ and $M=0$, the value ∞ is represented, where ∞ is the result of dividing a normal number by zero.
- when $E'=0$ and $M \neq 0$, denormal numbers are represented. Their value is $\pm 0.M \times 2^{-126}$
- When $E'=255$ and $M \neq 0$, the value represented is called not a number (NaN). A NaN is the result of performing an invalid operation such as $0/0$ or $\sqrt{0}$.

ARITHMETIC OPERATIONS ON FLOATING-POINT NUMBERS

Multiply Rule

- 1) Add the exponents & subtract 127.
- 2) Multiply the mantissas & determine sign of the result.
- 3) Normalize the resulting value if necessary.

Divide Rule

- 1) Subtract the exponents & add 127.
- 2) Divide the mantissas & determine sign of the result.
- 3) Normalize the resulting value if necessary.

Add/Subtract Rule

- 1) Choose the number with the smaller exponent & shift its mantissa right a number of steps equal to the difference in exponents (n).
- 2) Set exponent of the result equal to larger exponent.
- 3) Perform addition/subtraction on the mantissas & determine sign of the result.
- 4) Normalize the resulting value if necessary.

COMPUTER ORGANIZATION

IMPLEMENTING FLOATING-POINT OPERATIONS

- First compare exponents to determine how far to shift the mantissa of the number with the smaller exponent.
- The shift-count value n
 - is determined by 8 bit subtractor &
 - is sent to SHIFTER unit.
- In step 1, sign is sent to SWAP network (Figure 9.26).
 - If $\text{sign}=0$, then $E_A > E_B$ and mantissas M_A & M_B are sent straight through SWAP network.
 - If $\text{sign}=1$, then $E_A < E_B$ and the mantissas are swapped before they are sent to SHIFTER.
- In step 2, 2:1 MUX is used. The exponent of result E is tentatively determined as E_A if $E_A > E_B$ or E_B if $E_A < E_B$.
- In step 3, CONTROL logic
 - determines whether mantissas are to be added or subtracted.
 - determines sign of the result.
- In step 4, result of step 3 is normalized. The number of leading zeros in M determines number of bit shifts(X) to be applied to M .

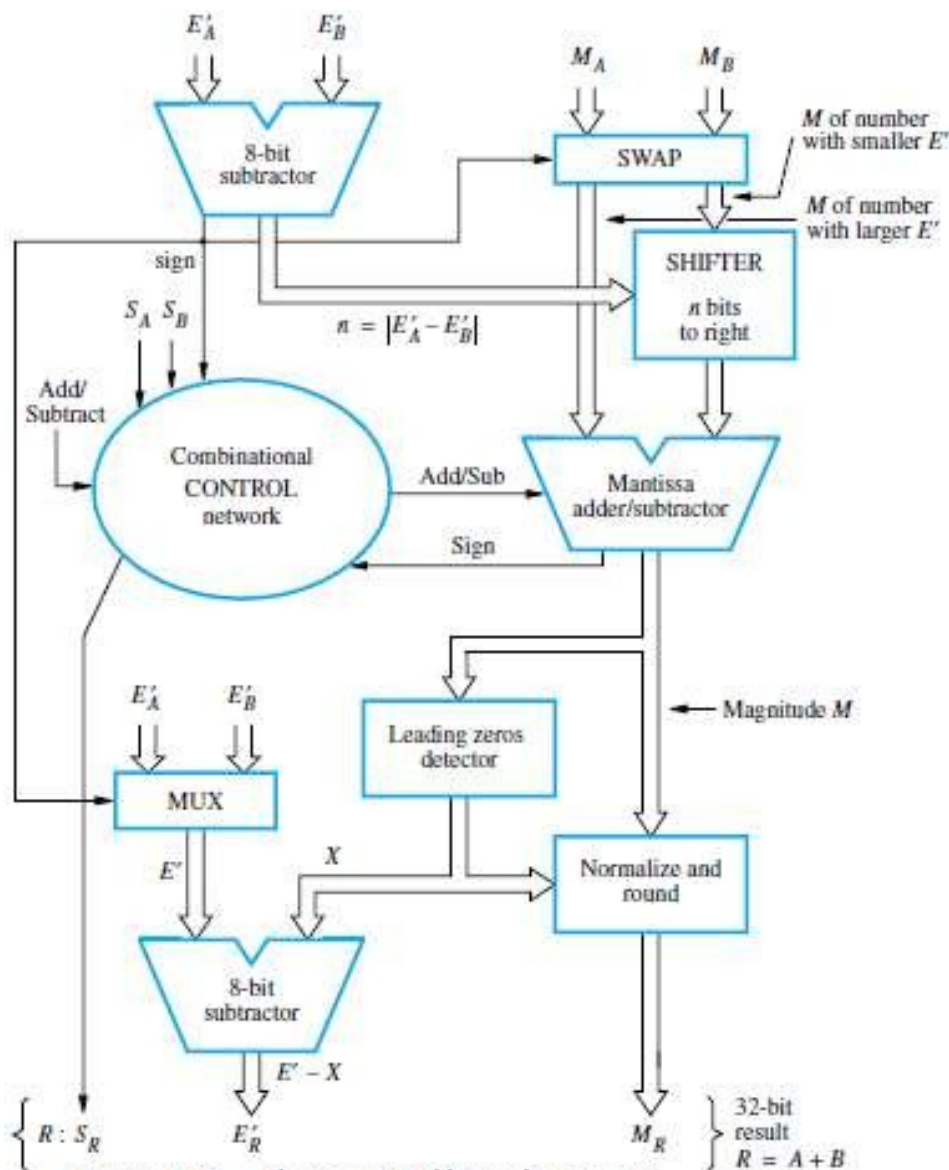


Figure 9.28 Floating-point addition-subtraction unit.

COMPUTER ORGANIZATION

Problem 1:

Represent the decimal values 5, -2, 14, -10, 26, -19, 51 and -43 as signed 7-bit numbers in the following binary formats:

- (a) sign-and-magnitude
- (b) 1's-complement
- (c) 2's-complement

Solution:

The three binary representations are given as:

Decimal values	Sign-and-magnitude representation	1's-complement representation	2's-complement representation
5	0000101	0000101	0000101
-2	1000010	1111101	1111110
14	0001110	0001110	0001110
-10	1001010	1110101	1110110
26	0011010	0011010	0011010
-19	1010011	1101100	1101101
51	0110011	0110011	0110011
-43	1101011	1010100	1010101

Problem 2:

(a) Convert the following pairs of decimal numbers to 5-bit 2's-complement numbers, then add them. State whether or not overflow occurs in each case.

- a) 5 and 10 b) 7 and 13
- c) -14 and 11 d) -5 and 7
- e) -3 and -8

(b) Repeat Problem 1.7 for the subtract operation, where the second number of each pair is to be subtracted from the first number. State whether or not overflow occurs in each case.

Solution:

(a)

(a)	00101	(b)	00111	(c)	10010
	+ 01010		+ 01101		+ 01011
	-----		-----		-----
	01111		10100		11101
	no overflow		overflow		no overflow

(d)	11011	(e)	11101	(f)	10110
	+ 00111		+ 11000		+ 10011
	-----		-----		-----
	00010		10101		01001
	no overflow		no overflow		overflow

(b) To subtract the second number, form its 2's-complement and add it to the first number.

(a)	00101	(b)	00111	(c)	10010
	+ 10110		+ 10011		+ 10101
	-----		-----		-----
	11011		11010		00111
	no overflow		no overflow		overflow
(d)	11011	(e)	11101	(f)	10110
	+ 11001		+ 01000		+ 01101
	-----		-----		-----
	10100		00101		00011
	no overflow		no overflow		no overflow

COMPUTER ORGANIZATION

Problem 3:

Perform following operations on the 6-bit signed numbers using 2's complement representation system. Also indicate whether overflow has occurred.

$$\begin{array}{r}
 010110 \quad 101011 \quad 111111 \\
 +001001 \quad +100101 \quad +000111 \\
 \hline
 011001 \quad 110111 \quad 010101 \\
 +010000 \quad +111001 \quad +101011 \\
 \hline
 010110 \quad 111110 \quad 100001 \\
 -011111 \quad -100101 \quad -011101 \\
 \hline
 111111 \quad 000111 \quad 011010 \\
 -000111 \quad -111000 \quad -100010 \\
 \hline
 \end{array}$$

Solution:

$ \begin{array}{r} 010110 \\ + 001001 \\ \hline 011111 \end{array} $	$ \begin{array}{r} (+22) \\ + (+9) \\ \hline (+31) \end{array} $	$ \begin{array}{r} 101011 \\ + 100101 \\ \hline 010000 \\ \text{overflow} \end{array} $	$ \begin{array}{r} (-21) \\ + (-27) \\ \hline (-48) \end{array} $	$ \begin{array}{r} 111111 \\ + 000111 \\ \hline 000110 \end{array} $	$ \begin{array}{r} (-1) \\ + (+7) \\ \hline (+6) \end{array} $
$ \begin{array}{r} 011001 \\ + 010000 \\ \hline 101001 \\ \text{overflow} \end{array} $	$ \begin{array}{r} (+25) \\ + (+16) \\ \hline (+41) \end{array} $	$ \begin{array}{r} 110111 \\ + 111001 \\ \hline 110000 \end{array} $	$ \begin{array}{r} (-9) \\ + (-7) \\ \hline (-16) \end{array} $	$ \begin{array}{r} 010101 \\ + 101011 \\ \hline 000000 \end{array} $	$ \begin{array}{r} (+21) \\ + (-21) \\ \hline (0) \end{array} $
$ \begin{array}{r} 010110 \\ - 011111 \\ \hline \end{array} $	$ \begin{array}{r} (+22) \\ - (+31) \\ \hline (-9) \end{array} $	$ \begin{array}{r} 010110 \\ + 100001 \\ \hline 110111 \end{array} $			
$ \begin{array}{r} 111110 \\ - 100101 \\ \hline \end{array} $	$ \begin{array}{r} (-2) \\ - (-27) \\ \hline (+25) \end{array} $	$ \begin{array}{r} 111110 \\ + 011011 \\ \hline 011001 \end{array} $			
$ \begin{array}{r} 100001 \\ - 011101 \\ \hline \end{array} $	$ \begin{array}{r} (-31) \\ - (+29) \\ \hline (-60) \end{array} $	$ \begin{array}{r} 100001 \\ + 100011 \\ \hline 000100 \\ \text{overflow} \end{array} $			
$ \begin{array}{r} 111111 \\ - 000111 \\ \hline \end{array} $	$ \begin{array}{r} (-1) \\ - (+7) \\ \hline (-8) \end{array} $	$ \begin{array}{r} 111111 \\ + 111001 \\ \hline 111000 \end{array} $			
$ \begin{array}{r} 000111 \\ - 111000 \\ \hline \end{array} $	$ \begin{array}{r} (+7) \\ - (-8) \\ \hline (+15) \end{array} $	$ \begin{array}{r} 000111 \\ + 001000 \\ \hline 001111 \end{array} $			
$ \begin{array}{r} 011010 \\ - 100010 \\ \hline \end{array} $	$ \begin{array}{r} (+26) \\ - (-30) \\ \hline (+56) \end{array} $	$ \begin{array}{r} 011010 \\ + 011110 \\ \hline 111000 \\ \text{overflow} \end{array} $			

COMPUTER ORGANIZATION

Problem 4:

Perform signed multiplication of following 2's complement numbers using Booth's algorithm.

- (a) A=010111 and B=110110 (b) A=110011 and B=101100
 (c) A=110101 and B=011011 (d) A=001111 and B=001111
 (e) A=10100 and B=10101 (f) A=01110 and B=11000

Solution:

$$\begin{array}{r}
 \begin{array}{r}
 010111 \\
 \times 110110 \\
 \hline
 \end{array}
 \quad
 \begin{array}{r}
 +23 \\
 \times -10 \\
 \hline
 -230
 \end{array}
 \end{array}$$

sign extension

$$\begin{array}{r}
 \begin{array}{r}
 010111 \\
 \times 110110 \\
 \hline
 \end{array}
 \quad
 \begin{array}{r}
 010111 \\
 \times 0-1+10-10 \\
 \hline
 0 \\
 111111010001 \\
 0000010111 \\
 1111021001 \\
 \hline
 111100011010
 \end{array}$$

$$\begin{array}{r}
 \begin{array}{r}
 110011 \\
 \times 101100 \\
 \hline
 \end{array}
 \quad
 \begin{array}{r}
 -13 \\
 \times -20 \\
 \hline
 260
 \end{array}
 \end{array}$$

sign extension

$$\begin{array}{r}
 \begin{array}{r}
 110011 \\
 \times 101100 \\
 \hline
 \end{array}
 \quad
 \begin{array}{r}
 110011 \\
 \times -1+10-100 \\
 \hline
 0 \\
 000001101 \\
 11110011 \\
 010111021 \\
 \hline
 000100000100
 \end{array}$$

$$\begin{array}{r}
 \begin{array}{r}
 110101 \\
 \times 011011 \\
 \hline
 \end{array}
 \quad
 \begin{array}{r}
 -11 \\
 \times 27 \\
 \hline
 -297
 \end{array}
 \end{array}$$

sign extension

$$\begin{array}{r}
 \begin{array}{r}
 110101 \\
 \times 011011 \\
 \hline
 \end{array}
 \quad
 \begin{array}{r}
 110101 \\
 \times +10-1+10-1 \\
 \hline
 0 \\
 000001011 \\
 1111110101 \\
 000001011 \\
 111101011 \\
 \hline
 111011010111
 \end{array}$$

$$\begin{array}{r}
 \begin{array}{r}
 001111 \\
 \times 001111 \\
 \hline
 \end{array}
 \quad
 \begin{array}{r}
 15 \\
 \times 15 \\
 \hline
 225
 \end{array}
 \end{array}$$

$$\begin{array}{r}
 \begin{array}{r}
 001111 \\
 \times 001111 \\
 \hline
 \end{array}
 \quad
 \begin{array}{r}
 001111 \\
 \times 0+1000-1 \\
 \hline
 1111111110001 \\
 00001111 \\
 \hline
 000011100001
 \end{array}$$

$$\begin{array}{r}
 \begin{array}{r}
 10100(-12) \\
 \times 10101(-11) \\
 \hline
 \end{array}
 \quad
 \rightarrow
 \begin{array}{r}
 \begin{array}{r}
 10100 \\
 -11-11-1-1 \\
 \hline
 \end{array}
 \quad
 \begin{array}{r}
 000001100 \\
 11110100 \\
 0001100 \\
 110100 \\
 01100 \\
 \hline
 010000100 \quad (+132)
 \end{array}
 \end{array}$$

(recoded multiplier)

$$\begin{array}{r}
 \begin{array}{r}
 01110(+14) \\
 \times 11000(-8) \\
 \hline
 \end{array}
 \quad
 \rightarrow
 \begin{array}{r}
 \begin{array}{r}
 01110 \\
 0-1000 \\
 \hline
 \end{array}
 \quad
 \begin{array}{r}
 0000000000 \\
 0000000000 \\
 0000000000 \\
 1110010 \\
 000000 \\
 \hline
 1110010000 \quad (-112)
 \end{array}
 \end{array}$$

(recoded multiplier)

COMPUTER ORGANIZATION

Problem 5:

Perform signed multiplication of following 2's complement numbers using bit-pair recoding method.

(a) A=010111 and B=110110

(b) A=110011 and B=101100

(c) A=110101 and B=011011

(d) A=001111 and B=001111

Solution:

$$\begin{array}{r}
 010111 \\
 \times 110110 \\
 \hline
 \end{array}
 \qquad
 \begin{array}{r}
 010111 \\
 -1 \quad +2 \quad -2 \\
 \hline
 111111 \quad 1010010 \\
 0000101110 \\
 111101011 \\
 \hline
 111100011010
 \end{array}$$

$$\begin{array}{r}
 110011 \\
 \times 101100 \\
 \hline
 \end{array}
 \qquad
 \begin{array}{r}
 110011 \\
 -1 \quad -1 \quad 0 \\
 \hline
 0000001101 \\
 0000111011 \\
 \hline
 000100000100
 \end{array}$$

$$\begin{array}{r}
 110101 \\
 \times 011011 \\
 \hline
 \end{array}
 \qquad
 \begin{array}{r}
 110101 \\
 +2 \quad -1 \quad -1 \\
 \hline
 0000001011 \\
 0000001011 \\
 11101011 \\
 \hline
 111011010111
 \end{array}$$

$$\begin{array}{r}
 001111 \\
 \times 001111 \\
 \hline
 \end{array}
 \qquad
 \begin{array}{r}
 001111 \\
 +1 \quad -1 \\
 \hline
 111111110001 \\
 00001111 \\
 \hline
 000011100001
 \end{array}$$

COMPUTER ORGANIZATION

Problem 6:

Given A=10101 and B=00100, perform A/B using restoring division algorithm.

Solution:

Initially	0 0 0 0 0 0 (A)	1 0 1 0 1 (Q)
	0 0 0 1 0 0 (M)	
Shift	0 0 0 0 0 1	0 1 0 1
Subtract	1 1 1 1 0 0	
Set q0	1 1 1 1 0 1	
Restore	1 0 0	
Shift	0 0 0 0 0 1	0 1 0 1 0
Subtract	0 0 0 0 1 0	1 0 1 0
Shift	0 0 0 0 1 0	
Subtract	1 1 1 1 0 0	
Set q0	1 1 1 1 1 0	
Restore	1 0 0	
Shift	0 0 0 0 1 0	1 0 1 0 0
Subtract	0 0 0 1 0 1	0 1 0 0
Shift	0 0 0 1 0 1	
Subtract	1 1 1 1 0 0	
Set q0	0 0 0 0 0 1	
No restore	0 0 0 0 0 0	
Shift	0 0 0 0 0 1	1 0 1 0 0
Subtract	0 0 0 0 0 1	0 1 0 0 1
Shift	0 0 0 0 0 1	
Subtract	1 1 1 1 0 0	
Set q0	1 1 1 1 1 0	
Restore	1 0 0	
Shift	0 0 0 0 1 0	1 0 0 1 0
Subtract	0 0 0 1 0 1	0 0 1 0
Shift	0 0 0 1 0 1	
Subtract	1 1 1 1 0 0	
Set q0	0 0 0 0 0 1	
No restore	0 0 0 0 0 0	
Shift	0 0 0 0 0 1	0 0 1 0 1
	remainder	quotient

Problem 7:

Given A=10101 and B=00101, perform A/B using non-restoring division algorithm.

Solution:

	000000	10101	
	A	Q	
	000101		Initial configuration
	M		
shift	000001	0 1 0 1 	
subtract	111011		1st cycle
	111100	0 1 0 1 0	
shift	111000	1 0 1 0 	
add	000101		2nd cycle
	111101	1 0 1 0 0	
shift	111011	0 1 0 0 	
add	000101		3rd cycle
	000000	0 1 0 0 1	
shift	000000	1 0 0 1 	
subtract	111011		4th cycle
	111011	1 0 0 1 0	
shift	110111	0 0 1 0 	
add	000101		5th cycle
	111100	0 0 1 0 0	
add	000101		
	000001		
		quotient	
	remainder		

COMPUTER ORGANIZATION

Problem 8:

Represent 1259.12510 in single precision and double precision formats

Solution:

Step 1: Convert decimal number to binary format

$$1259_{(10)} = 10011101011_{(2)}$$

Fractional Part

$$0.125_{(10)} = 0.001$$

$$\begin{aligned}\text{Binary number} &= 10011101011 + 0.001 \\ &= 10011101011.001\end{aligned}$$

Step 2: Normalize the number

$$10011101011.001 = 1.0011101011001 \times 2^{10}$$

Step 3: Single precision format:

For a given number $S=0$, $E=10$ and $M=0011101011001$

Bias for single precision format is = 127

$$\begin{aligned}E' &= E + 127 = 10 + 127 = 137_{(10)} \\ &= 10001001_{(2)}\end{aligned}$$

Number in single precision format is given as

$$\underbrace{0}_{\text{Sign}} \underbrace{10001001}_{\text{Exponent}} \underbrace{0011101011001\dots0}_{\text{Mantissa(23 bit)}}$$

Step 4: Double precision format:

For a given number $S=0$, $E=10$ and $M=0011101011001$

Bias for double precision format is = 1023

$$\begin{aligned}E' &= E + 1023 = 10 + 1023 = 1033_{(10)} \\ &= 10000001001_{(2)}\end{aligned}$$

Number in double precision format is given as

$$\underbrace{0}_{\text{Sign}} \underbrace{10001001}_{\text{Exponent}} \underbrace{0011101011001\dots0}_{\text{Mantissa(23 bit)}}$$

MODULE 5: BASIC PROCESSING UNIT

SOME FUNDAMENTAL CONCEPTS

- To execute an instruction, processor has to perform following 3 steps:
 - 1) Fetch contents of memory-location pointed to by PC. Content of this location is an instruction to be executed. The instructions are loaded into IR, Symbolically, this operation is written as:
 $IR \leftarrow [PC]$
 - 2) Increment PC by 4.
 $PC \leftarrow [PC] + 4$
 - 3) Carry out the actions specified by instruction (in the IR).
- The first 2 steps are referred to as **Fetch Phase**.
Step 3 is referred to as **Execution Phase**.
- The operation specified by an instruction can be carried out by performing one or more of the following actions:
 - 1) Read the contents of a given memory-location and load them into a register.
 - 2) Read data from one or more registers.
 - 3) Perform an arithmetic or logic operation and place the result into a register.
 - 4) Store data from a register into a given memory-location.
- The hardware-components needed to perform these actions are shown in Figure 5.1.

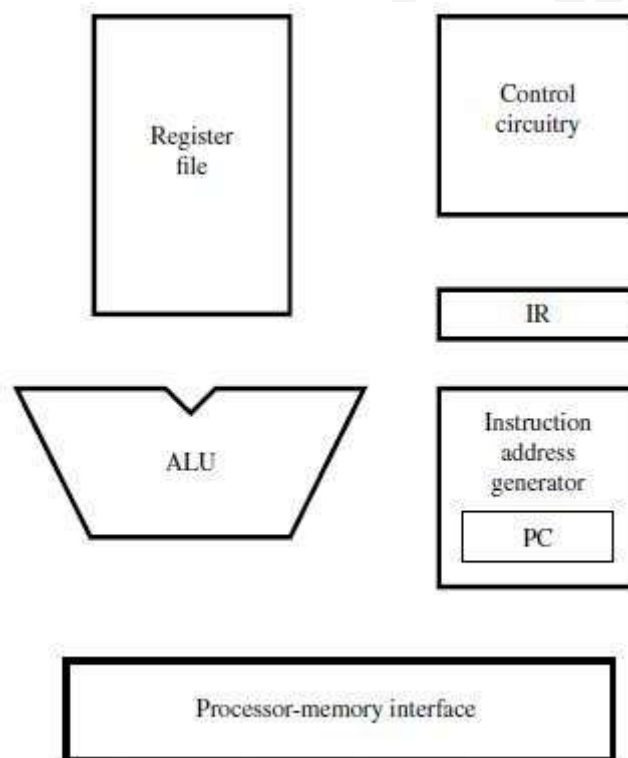


Figure 5.1 Main hardware components of a processor.

COMPUTER ORGANIZATION

SINGLE BUS ORGANIZATION

- ALU and all the registers are interconnected via a **Single Common Bus** (Figure 7.1).
- Data & address lines of the external memory-bus is connected to the internal processor-bus via MDR & MAR respectively. (MDR → Memory Data Register, MAR → Memory Address Register).
- **MDR** has 2 inputs and 2 outputs. Data may be loaded
 - into MDR either from memory-bus (external) or
 - from processor-bus (internal).
- **MAR**'s input is connected to internal-bus;
MAR's output is connected to external-bus.
- **Instruction Decoder & Control Unit** is responsible for
 - issuing the control-signals to all the units inside the processor.
 - implementing the actions specified by the instruction (loaded in the IR).
- Register R0 through R(n-1) are the **Processor Registers**.
The programmer can access these registers for general-purpose use.
- Only processor can access 3 registers **Y, Z & Temp** for temporary storage during program-execution.
The programmer cannot access these 3 registers.
- In **ALU**,
 - 1) „A“ input gets the operand from the output of the multiplexer (MUX).
 - 2) „B“ input gets the operand directly from the processor-bus.
- There are 2 options provided for „A“ input of the ALU.
- MUX is used to select one of the 2 inputs.
- **MUX** selects either
 - output of Y or
 - constant-value 4 (which is used to increment PC content).

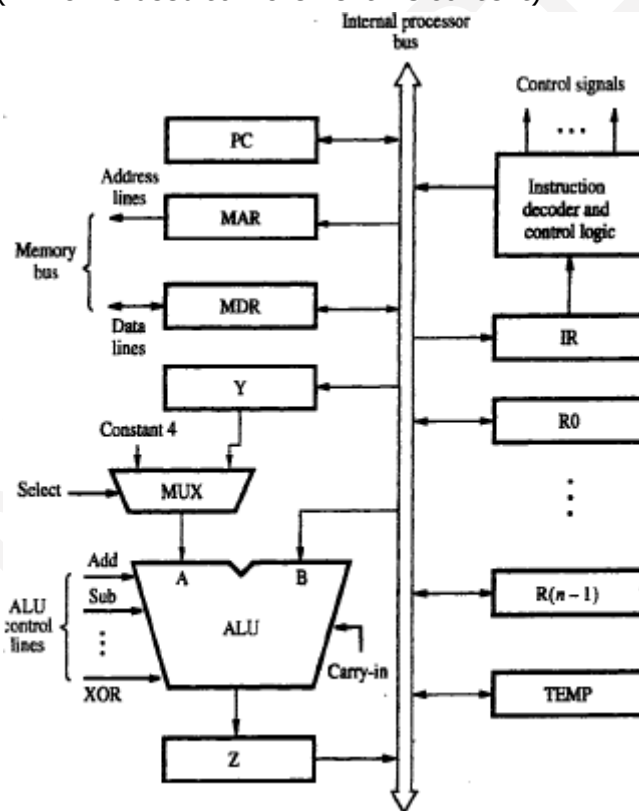


Figure 7.1 Single-bus organization of the datapath inside a processor.

- An instruction is executed by performing one or more of the following operations:
 - 1) Transfer a word of data from one register to another or to the ALU.
 - 2) Perform arithmetic or a logic operation and store the result in a register.
 - 3) Fetch the contents of a given memory-location and load them into a register.
 - 4) Store a word of data from a register into a given memory-location.

- **Disadvantage:** Only one data-word can be transferred over the bus in a clockcycle.

Solution: Provide multiple internal-paths. Multiple paths allow several data-transfers to take place in parallel.

COMPUTER ORGANIZATION

REGISTER TRANSFERS

- Instruction execution involves a sequence of steps in which data are transferred from one register to another.
- For each register, two control-signals are used: Ri_{in} & Ri_{out} . These are called **Gating Signals**.
- $Ri_{in}=1 \rightarrow$ data on bus is loaded into Ri .
 $Ri_{out}=1 \rightarrow$ content of Ri is placed on bus.
- $Ri_{out}=0$, \rightarrow bus can be used for transferring data from other registers.
- For example, *Move R1, R2*; This transfers the contents of register R1 to register R2. This can be accomplished as follows:
 - 1) Enable the output of registers R1 by setting $R1_{out}$ to 1 (Figure 7.2).
This places the contents of R1 on processor-bus.
 - 2) Enable the input of register R2 by setting $R2_{in}$ to 1.
This loads data from processor-bus into register R4.
- All operations and data transfers within the processor take place within time-periods defined by the **processor-clock**.
- The control-signals that govern a particular transfer are asserted at the start of the clock cycle.

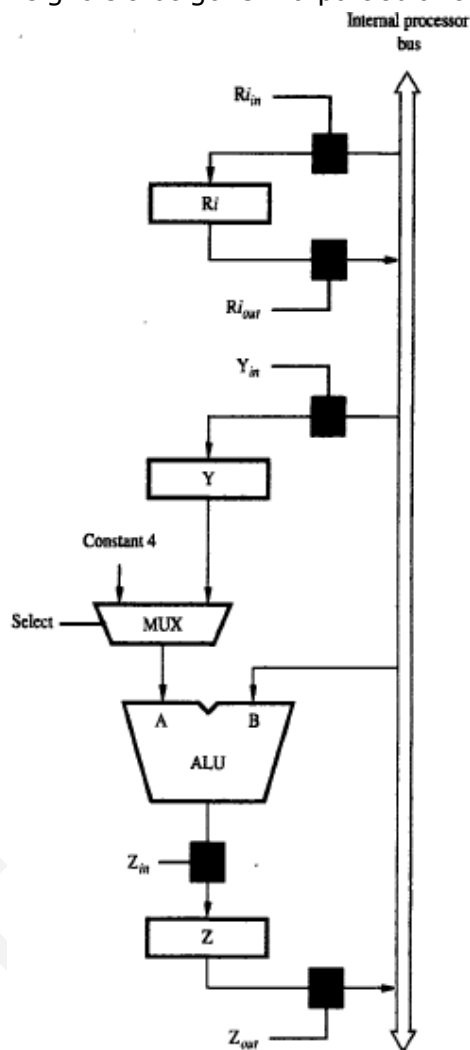


Figure 7.2 Input and output gating for the registers in Figure 7.1.

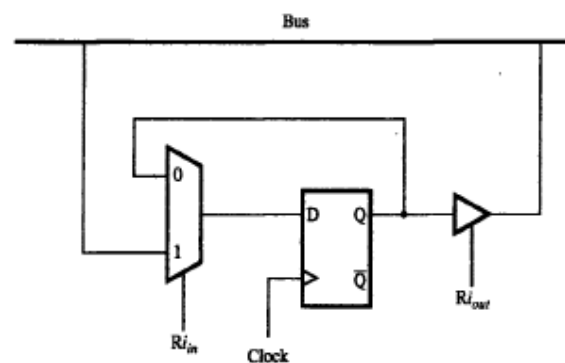


Figure 7.3 Input and output gating for one register bit.

Input & Output Gating for one Register Bit

- A 2-input multiplexer is used to select the data applied to the input of an edge-triggered D flip-flop.
- $Ri_{in}=1 \rightarrow$ mux selects data on bus. This data will be loaded into flip-flop at rising-edge of clock.
 $Ri_{in}=0 \rightarrow$ mux feeds back the value currently stored in flip-flop (Figure 7.3).
- Q output of flip-flop is connected to bus via a tri-state gate.
 $Ri_{out}=0 \rightarrow$ gate's output is in the high-impedance state.
 $Ri_{out}=1 \rightarrow$ the gate drives the bus to 0 or 1, depending on the value of Q.

COMPUTER ORGANIZATION

PERFORMING AN ARITHMETIC OR LOGIC OPERATION

- The ALU performs arithmetic operations on the 2 operands applied to its A and B inputs.
- One of the operands is output of MUX;
And, the other operand is obtained directly from processor-bus.
- The result (produced by the ALU) is stored temporarily in register Z.
- The sequence of operations for $[R3] \leftarrow [R1] + [R2]$ is as follows:
 - 1) $R1_{out} \rightarrow Y_{in}$
 - 2) $R2_{out} \rightarrow \text{SelectY, Add, } Z_{in}$
 - 3) $Z_{out} \rightarrow R3_{in}$
- Instruction execution proceeds as follows:
Step 1 --> Contents from register R1 are loaded into register Y.
Step2 --> Contents from Y and from register R2 are applied to the A and B inputs of ALU;
Addition is performed &
Result is stored in the Z register.
Step 3 --> The contents of Z register is stored in the R3 register.
- The signals are activated for the duration of the clock cycle corresponding to that step. All other signals are inactive.

CONTROL-SIGNALS OF MDR

- The MDR register has 4 control-signals (Figure 7.4):
 - 1) MDR_{in} & MDR_{out} control the connection to the internal processor data bus &
 - 2) MDR_{inE} & MDR_{outE} control the connection to the memory Data bus.
- MAR register has 2 control-signals.
 - 1) MAR_{in} controls the connection to the internal processor address bus &
 - 2) MAR_{out} controls the connection to the memory address bus.

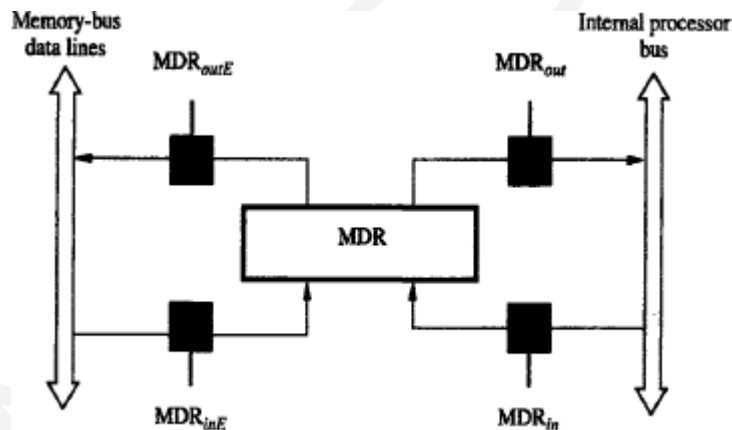


Figure 7.4 Connection and control signals for register MDR.

COMPUTER ORGANIZATION

FETCHING A WORD FROM MEMORY

- To fetch instruction/data from memory, processor transfers required address to MAR.
At the same time, processor issues Read signal on control-lines of memory-bus.
- When requested-data are received from memory, they are stored in MDR. From MDR, they are transferred to other registers.
- The response time of each memory access varies (based on cache miss, memory-mapped I/O). To accommodate this, MFC is used. (MFC → Memory Function Completed).
- MFC is a signal sent from addressed-device to the processor. MFC informs the processor that the requested operation has been completed by addressed-device.
- Consider the instruction Move (R1),R2. The sequence of steps is (Figure 7.5):
 - 1) $R1_{out}, MAR_{in}$, Read ;desired address is loaded into MAR & Read command is issued.
 - 2) $MDR_{inE}, WMFC$;load MDR from memory-bus & Wait for MFC response from memory.
 - 3) $MDR_{out}, R2_{in}$;load R2 from MDR.

where WMFC=control-signal that causes processor's control. circuitry to wait for arrival of MFC signal.

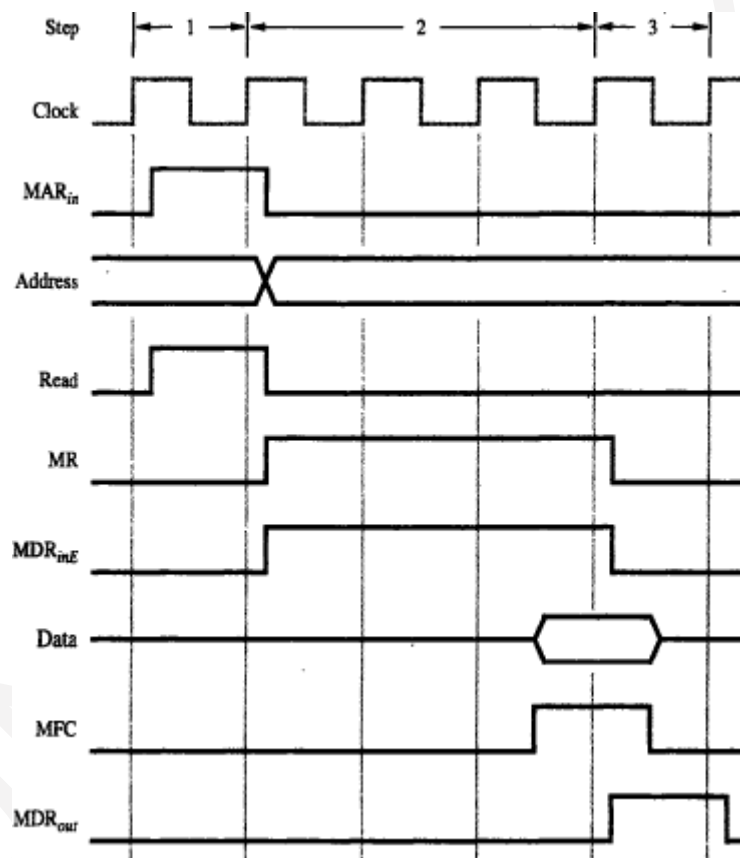


Figure 7.5 Timing of a memory Read operation.

Storing a Word in Memory

- Consider the instruction Move R2,(R1). This requires the following sequence:
 - 1) $R1_{out}, MAR_{in}$;desired address is loaded into MAR.
 - 2) $R2_{out}, MDR_{in}, Write$;data to be written are loaded into MDR & Write command is issued.
 - 3) $MDR_{outE}, WMFC$;load data into memory-location pointed by R1 from MDR.

COMPUTER ORGANIZATION

EXECUTION OF A COMPLETE INSTRUCTION

- Consider the instruction *Add (R3),R1* which adds the contents of a memory-location pointed by R3 to register R1. Executing this instruction requires the following actions:
 - 1) Fetch the instruction.
 - 2) Fetch the first operand.
 - 3) Perform the addition &
 - 4) Load the result into R1.

Step	Action
1	$PC_{out}, MAR_{in}, Read, Select4, Add, Z_{in}$
2	$Z_{out}, PC_{in}, Y_{in}, WMFC$
3	MDR_{out}, IR_{in}
4	$R3_{out}, MAR_{in}, Read$
5	$R1_{out}, Y_{in}, WMFC$
6	$MDR_{out}, SelectY, Add, Z_{in}$
7	$Z_{out}, R1_{in}, End$

Figure 7.6 Control sequence for execution of the instruction *Add (R3),R1*

- Instruction execution proceeds as follows:
 - Step1--> The instruction-fetch operation is initiated by
 - loading contents of PC into MAR &
 - sending a Read request to memory.The Select signal is set to Select4, which causes the Mux to select constant 4. This value is added to operand at input B (PC's content), and the result is stored in Z.
 - Step2--> Updated value in Z is moved to PC. This completes the PC increment operation and PC will now point to next instruction.
 - Step3--> Fetched instruction is moved into MDR and then to IR.
 - The step 1 through 3 constitutes the **Fetch Phase**.
 - At the beginning of step 4, the instruction decoder interprets the contents of the IR. This enables the control circuitry to activate the control-signals for steps 4 through 7.
 - The step 4 through 7 constitutes the **Execution Phase**.
 - Step4--> Contents of R3 are loaded into MAR & a memory read signal is issued.
 - Step5--> Contents of R1 are transferred to Y to prepare for addition.
 - Step6--> When Read operation is completed, memory-operand is available in MDR, and the addition is performed.
 - Step7--> Sum is stored in Z, then transferred to R1. The End signal causes a new instruction fetch cycle to begin by returning to step1.

COMPUTER ORGANIZATION

BRANCHING INSTRUCTIONS

- Control sequence for an **unconditional branch instruction** is as follows:

Step	Action
1	$PC_{out}, MAR_{in}, Read, Select4, Add, Z_{in}$
2	$Z_{out}, PC_{in}, Y_{in}, WMFC$
3	MDR_{out}, IR_{in}
4	$Offset\text{-}field\text{-}of\text{-}IR_{out}, Add, Z_{in}$
5	Z_{out}, PC_{in}, End

Figure 7.7 Control sequence for an unconditional Branch instruction.

- Instruction execution proceeds as follows:
 - Step 1-3--> The processing starts & the fetch phase ends in step3.
 - Step 4--> The offset-value is extracted from IR by instruction-decoding circuit.
Since the updated value of PC is already available in register Y, the offset X is gated onto the bus, and an addition operation is performed.
 - Step 5--> the result, which is the branch-address, is loaded into the PC.
- The branch instruction loads the branch target address in PC so that PC will fetch the next instruction from the branch target address.
- The branch target address is usually obtained by adding the offset in the contents of PC.
- The offset X is usually the difference between the branch target-address and the address immediately following the branch instruction.
- In case of **conditional branch**,
 - we have to check the status of the condition-codes before loading a new value into the PC.
e.g.: $Offset\text{-}field\text{-}of\text{-}IR_{out}, Add, Z_{in}$, If $N=0$ then End
If $N=0$, processor returns to step 1 immediately after step 4.
If $N=1$, step 5 is performed to load a new value into PC.

COMPUTER ORGANIZATION

MULTIPLE BUS ORGANIZATION

• **Disadvantage of Single-bus organization:** Only one data-word can be transferred over the bus in a clock cycle. This increases the steps required to complete the execution of the instruction

Solution: To reduce the number of steps, most processors provide multiple internal-paths. Multiple paths enable several transfers to take place in parallel.

• As shown in fig 7.8, three buses can be used to connect registers and the ALU of the processor.

• All general-purpose registers are grouped into a single block called the **Register File**.

• Register-file has 3 ports:

1) Two output-ports allow the contents of 2 different registers to be simultaneously placed on buses A & B.

2) Third input-port allows data on bus C to be loaded into a third register during the same clock-cycle.

• Buses A and B are used to transfer source-operands to A & B inputs of ALU.

• The result is transferred to destination over bus C.

• **Incrementer Unit** is used to increment PC by 4.

Step	Action
1	$PC_{out}, R=B, MAR_{in}, Read, IncPC$
2	WMFC
3	$MDR_{outB}, R=B, IR_{in}$
4	$R4_{outA}, R5_{outB}, SelectA, Add, R6_{in}, End$

Figure 7.9 Control sequence for the instruction Add R4,R5,R6

• Instruction execution proceeds as follows:

Step 1--> Contents of PC are

→ passed through ALU using $R=B$ control-signal &

→ loaded into MAR to start memory Read operation. At the same time, PC is incremented by 4.

Step 2--> Processor waits for MFC signal from memory.

Step 3--> Processor loads requested-data into MDR, and then transfers them to IR.

Step 4--> The instruction is decoded and add operation takes place in a single step.

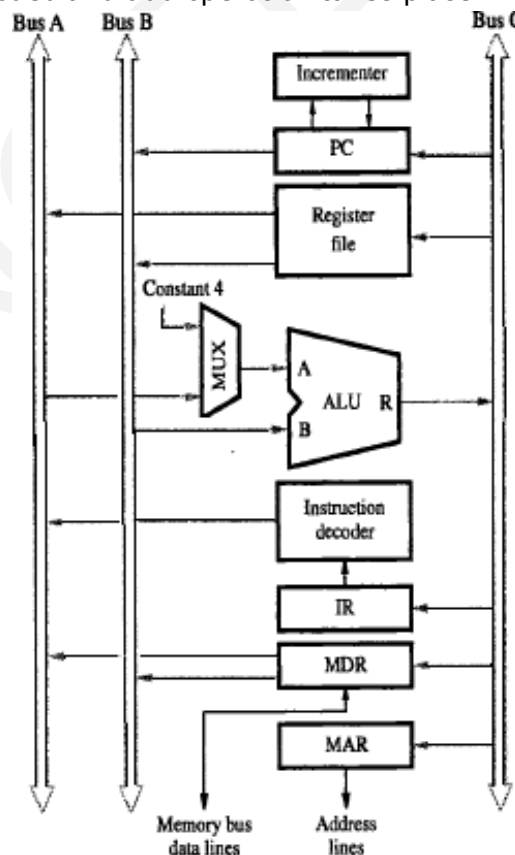


Figure 7.8 Three-bus organization of the datapath.

COMPUTER ORGANIZATION

COMPLETE PROCESSOR

- This has separate processing-units to deal with integer data and floating-point data.
 - Integer Unit** → To process integer data. (Figure 7.14).
 - Floating Unit** → To process floating -point data.
- **Data-Cache** is inserted between these processing-units & main-memory.
 - The integer and floating unit gets data from data cache.
- **Instruction-Unit** fetches instructions
 - from an instruction-cache or
 - from main-memory when desired instructions are not already in cache.
- Processor is connected to system-bus & hence to the rest of the computer by means of a **Bus Interface**.
- Using separate caches for instructions & data is common practice in many processors today.
- A processor may include several units of each type to increase the potential for concurrent operations.
- The 80486 processor has 8-kbytes single cache for both instruction and data.
 - Whereas the Pentium processor has two separate 8 kbytes caches for instruction and data.

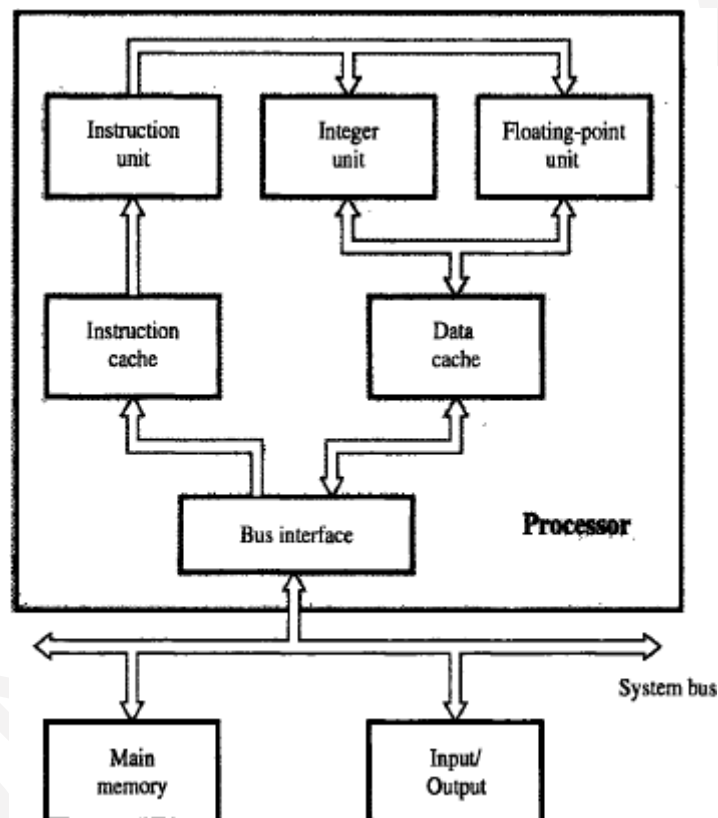


Figure 7.14 Block diagram of a complete processor.

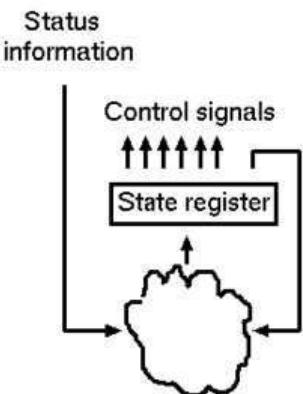
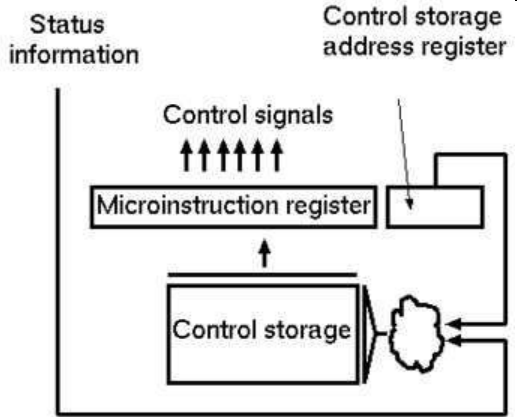
Note:

To execute instructions, the processor must have some means of generating the control-signals. There are two approaches for this purpose:

- 1) Hardwired control and 2) Microprogrammed control.

COMPUTER ORGANIZATION

HARDWIRED CONTROL VS MICROPROGRAMMED CONTROL

Attribute	Hardwired Control	Microprogrammed Control
Definition	Hardwired control is a control mechanism to generate control-signals by using gates, flip-flops, decoders, and other digital circuits.	Micro programmed control is a control mechanism to generate control-signals by using a memory called control store (CS), which contains the control-signals.
Speed	Fast	Slow
Control functions	Implemented in hardware.	Implemented in software.
Flexibility	Not flexible to accommodate new system specifications or new instructions.	More flexible, to accommodate new system specification or new instructions redesign is required.
Ability to handle large or complex instruction sets	Difficult.	Easier.
Ability to support operating systems & diagnostic features	Very difficult.	Easy.
Design process	Complicated.	Orderly and systematic.
Applications	Mostly RISC microprocessors.	Mainframes, some microprocessors.
Instructionset size	Usually under 100 instructions.	Usually over 100 instructions.
ROM size	-	2K to 10K by 20-400 bit microinstructions.
Chip area efficiency	Uses least area.	Uses more area.
Diagram		

COMPUTER ORGANIZATION

MICROPROGRAMMED CONTROL

- Microprogramming is a method of control unit design (Figure 7.16).
- Control-signals are generated by a program similar to machine language programs.
- **Control Word(CW)** is a word whose individual bits represent various control-signals (like Add, PC_{in}).
- Each of the control-steps in control sequence of an instruction defines a unique combination of 1s & 0s in CW.
- Individual control-words in microroutine are referred to as **microinstructions** (Figure 7.15).
- A sequence of CWs corresponding to control-sequence of a machine instruction constitutes the **microroutine**.
- The microroutines for all instructions in the instruction-set of a computer are stored in a special memory called the **Control Store (CS)**.
- Control-unit generates control-signals for any instruction by sequentially reading CWs of corresponding microroutine from CS.
- **μPC** is used to read CWs sequentially from CS. (μPC → Microprogram Counter).
- Every time new instruction is loaded into IR, o/p of **Starting Address Generator** is loaded into μPC.
- Then, μPC is automatically incremented by clock;
causing successive microinstructions to be read from CS.
Hence, control-signals are delivered to various parts of processor in correct sequence.

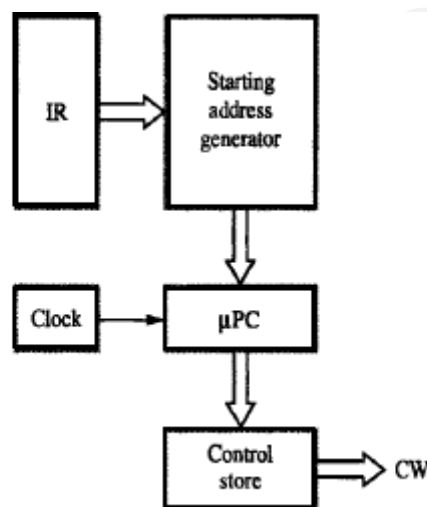


Figure 7.16 Basic organization of a microprogrammed control unit.

Micro - instruction	.	PC _{in}	PC _{out}	MAR _{in}	Read	MDR _{out}	IR _{in}	Y _{in}	Select	Add	Z _{in}	Z _{out}	R1 _{out}	R1 _{in}	R3 _{out}	WMFC	End	.
1		0	1	1	1	0	0	0	1	1	1	0	0	0	0	0	0	
2		1	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	
3		0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	

Figure 7.15 An example of microinstructions for Figure 7.6.

Advantages

- It simplifies the design of control unit. Thus it is both, cheaper and less error prone implement.
- Control functions are implemented in software rather than hardware.
- The design process is orderly and systematic.
- More flexible, can be changed to accommodate new system specifications or to correct the design errors quickly and cheaply.
- Complex function such as floating point arithmetic can be realized efficiently.

Disadvantages

- A microprogrammed control unit is somewhat slower than the hardwired control unit, because time is required to access the microinstructions from CM.
- The flexibility is achieved at some extra hardware cost due to the control memory and its access circuitry.

COMPUTER ORGANIZATION

ORGANIZATION OF MICROPROGRAMMED CONTROL UNIT TO SUPPORT CONDITIONAL BRANCHING

- **Drawback of previous Microprogram control:**

- It cannot handle the situation when the control unit is required to check the status of the condition codes or external inputs to choose between alternative courses of action.

- **Solution:**

- Use conditional branch microinstruction.

- In case of conditional branching, microinstructions specify which of the external inputs, condition-codes should be checked as a condition for branching to take place.

- **Starting and Branch Address Generator Block** loads a new address into μ PC when a microinstruction instructs it to do so (Figure 7.18).

- To allow implementation of a conditional branch, inputs to this block consist of
 - external inputs and condition-codes &
 - contents of IR.

- μ PC is incremented every time a new microinstruction is fetched from microprogram memory except in following situations:

- 1) When a new instruction is loaded into IR, μ PC is loaded with starting-address of microroutine for that instruction.
- 2) When a Branch microinstruction is encountered and branch condition is satisfied, μ PC is loaded with branch-address.
- 3) When an End microinstruction is encountered, μ PC is loaded with address of first CW in microroutine for instruction fetch cycle.

Address	Microinstruction
0	PC_{out} , MAR_{in} , Read, Select4, Add, Z_{in}
1	Z_{out} , PC_{in} , Y_{in} , WMFC
2	MDR_{out} , IR_{in}
3	Branch to starting address of appropriate microroutine
25	If $N=0$, then branch to microinstruction 0
26	Offset-field-of- IR_{out} , SelectY, Add, Z_{in}
27	Z_{out} , PC_{in} , End

Figure 7.17 Microroutine for the instruction Branch < 0.

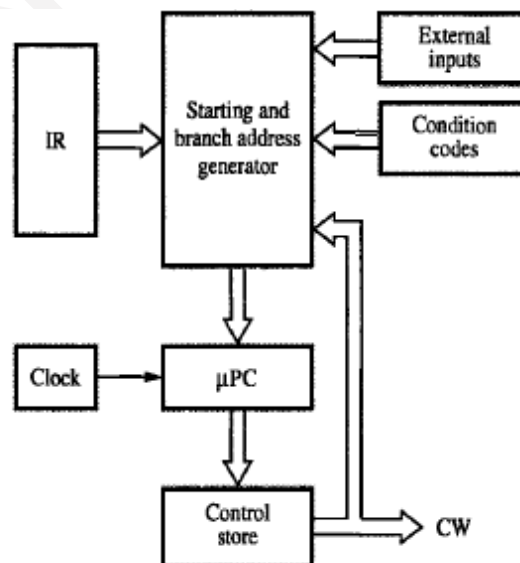


Figure 7.18 Organization of the control unit to allow conditional branching in the microprogram.

COMPUTER ORGANIZATION

MICROINSTRUCTIONS

- A simple way to structure microinstructions is to assign one bit position to each control-signal required in the CPU.
- There are 42 signals and hence each microinstruction will have 42 bits.
- **Drawbacks of microprogrammed control:**
 - 1) Assigning individual bits to each control-signal results in long microinstructions because the number of required signals is usually large.
 - 2) Available bit-space is poorly used because only a few bits are set to 1 in any given microinstruction.
- **Solution:** Signals can be grouped because
 - 1) Most signals are not needed simultaneously.
 - 2) Many signals are mutually exclusive. E.g. only 1 function of ALU can be activated at a time.

For ex: Gating signals: IN and OUT signals (Figure 7.19).
Control-signals: Read, Write.
ALU signals: Add, Sub, Mul, Div, Mod.
- Grouping control-signals into fields requires a little more hardware because decoding-circuits must be used to decode bit patterns of each field into individual control-signals.
- **Advantage:** This method results in a smaller control-store (only 20 bits are needed to store the patterns for the 42 signals).

Microinstruction				
F1	F2	F3	F4	F5
F1 (4 bits)	F2 (3 bits)	F3 (3 bits)	F4 (4 bits)	F5 (2 bits)
0000: No transfer	000: No transfer	000: No transfer	0000: Add	00: No action
0001: PC _{out}	001: PC _{in}	001: MAR _{in}	0001: Sub	01: Read
0010: MDR _{out}	010: IR _{in}	010: MDR _{in}	:	10: Write
0011: Z _{out}	011: Z _{in}	011: TEMP _{in}	:	
0100: R0 _{out}	100: R0 _{in}	100: Y _{in}	1111: XOR	
0101: R1 _{out}	101: R1 _{in}		16 ALU functions	
0110: R2 _{out}	110: R2 _{in}			
0111: R3 _{out}	111: R3 _{in}			
1010: TEMP _{out}				
1011: Offset _{out}				
F6	F7	F8	...	
F6 (1 bit)	F7 (1 bit)	F8 (1 bit)		
0: SelectY	0: No action	0: Continue		
1: Select4	1: WMFC	1: End		

Figure 7.19 An example of a partial format for field-encoded microinstructions.

COMPUTER ORGANIZATION

TECHNIQUES OF GROUPING OF CONTROL-SIGNALS

- The grouping of control-signal can be done either by using
 - 1) Vertical organization &
 - 2) Horizontal organisation.

Vertical Organization	Horizontal Organization
Highly encoded schemes that use compact codes to specify only a small number of control functions in each microinstruction are referred to as a vertical organization.	The minimally encoded scheme in which many resources can be controlled with a single microinstruction is called a horizontal organization.
Slower operating-speeds.	Useful when higher operating-speed is desired.
Short formats.	Long formats.
Limited ability to express parallel microoperations.	Ability to express a high degree of parallelism.
Considerable encoding of the control information.	Little encoding of the control information.

MICROPROGRAM SEQUENCING

- The task of microprogram sequencing is done by microprogram sequencer.
- Two important factors must be considered while designing the microprogram sequencer:
 - 1) The size of the microinstruction &
 - 2) The address generation time.
- The size of the microinstruction should be minimum so that the size of control memory required to store microinstructions is also less.
- This reduces the cost of control memory.
- With less address generation time, microinstruction can be executed in less time resulting better throughout.
- During execution of a microprogram the address of the next microinstruction to be executed has 3 sources:
 - 1) Determined by instruction register.
 - 2) Next sequential address &
 - 3) Branch.
- Microinstructions can be shared using microinstruction branching.
- **Disadvantage of microprogrammed branching:**
 - 1) Having a separate microroutine for each machine instruction results in a large total number of microinstructions and a large control-store.
 - 2) Execution time is longer because it takes more time to carry out the required branches.
- Consider the instruction *Add src, Rdst* ;which adds the source-operand to the contents of Rdst and places the sum in Rdst.
- Let source-operand can be specified in following addressing modes (Figure 7.20):
 - a) Indexed
 - b) Autoincrement
 - c) Autodecrement
 - d) Register indirect &
 - e) Register direct
- Each box in the chart corresponds to a microinstruction that controls the transfers and operations indicated within the box.
- The microinstruction is located at the address indicated by the octal number (001,002).

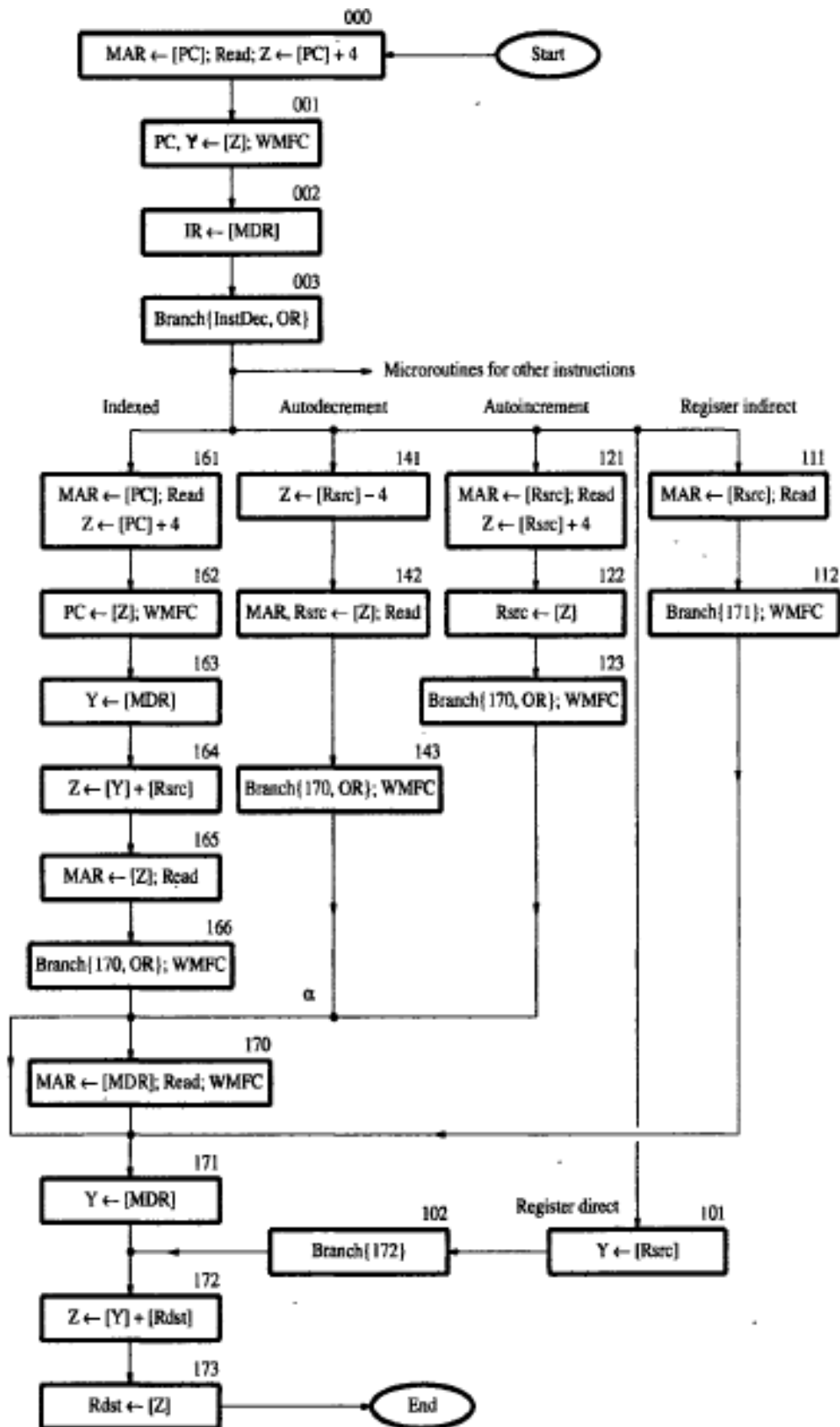


Figure 7.20 Flowchart of a microprogram for the `Add src, Rdst` instruction.

COMPUTER ORGANIZATION

BRANCH ADDRESS MODIFICATION USING BIT-ORING

- The branch address is determined by ORing particular bit or bits with the current address of microinstruction.
- **Eg:** If the current address is 170 and branch address is 171 then the branch address can be generated by ORing 01(bit 1), with the current address.
- Consider the point labeled α in the figure. At this point, it is necessary to choose between direct and indirect addressing modes.
- If indirect-mode is specified in the instruction, then the microinstruction in location 170 is performed to fetch the operand from the memory.
 - If direct-mode is specified, this fetch must be bypassed by branching immediately to location 171.
- The most efficient way to bypass microinstruction 170 is to have bit-ORing of
 - current address 170 &
 - branch address 171.

WIDE BRANCH ADDRESSING

- The instruction-decoder (InstDec) generates the starting-address of the microroutine that implements the instruction that has just been loaded into the IR.
- Here, register IR contains the Add instruction, for which the instruction decoder generates the microinstruction address 101. (However, this address cannot be loaded as is into the μ PC).
- The source-operand can be specified in any of several addressing-modes. The bit-ORing technique can be used to modify the starting-address generated by the instruction-decoder to reach the appropriate path.

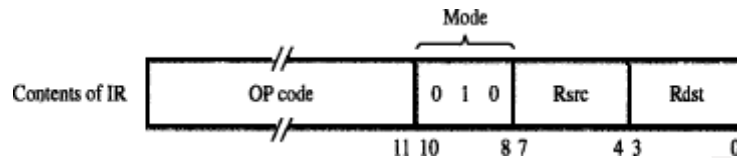
Use of WMFC

- WMFC signal is issued at location 112 which causes a branch to the microinstruction in location 171.
- WMFC signal means that the microinstruction may take several clock cycles to complete. If the branch is allowed to happen in the first clock cycle, the microinstruction at location 171 would be fetched and executed prematurely. To avoid this problem, WMFC signal must inhibit any change in the contents of the μ PC during the waiting-period.

COMPUTER ORGANIZATION

Detailed Examination of Add (Rsrc)+,Rdst

- Consider *Add (Rsrc)+,Rdst*; which adds Rsrc content to Rdst content, then stores the sum in Rdst and finally increments Rsrc by 4 (i.e. auto-increment mode).
- In bit 10 and 9, bit-patterns 11, 10, 01 and 00 denote indexed, auto-decrement, auto-increment and register modes respectively. For each of these modes, bit 8 is used to specify the indirect version.
- The processor has 16 registers that can be used for addressing purposes; each specified using a 4-bit-code (Figure 7.21).
- There are 2 stages of decoding:
 - 1) The microinstruction field must be decoded to determine that an Rsrc or Rdst register is involved.
 - 2) The decoded output is then used to gate the contents of the Rsrc or Rdst fields in the IR into a second decoder, which produces the gating-signals for the actual registers R0 to R15.



Address (octal)	Microinstruction
000	PC _{out} , MAR _{in} , Read, Select4, Add, Z _{in}
001	Z _{out} , PC _{in} , Y _{in} , WMFC
002	MDR _{out} , IR _{in}
003	μBranch {μPC ← 101 (from Instruction decoder); μPC _{5,4} ← [IR _{10,9}]; μPC ₃ ← [IR ₁₀] · [IR ₉] · [IR ₈]}
121	Rsrc _{out} , MAR _{in} , Read, Select4, Add, Z _{in}
122	Z _{out} , Rsrc _{in}
123	μBranch {μPC ← 170; μPC ₀ ← [IR ₈]}, WMFC
170	MDR _{out} , MAR _{in} , Read, WMFC
171	MDR _{out} , Y _{in}
172	Rdst _{out} , SelectY, Add, Z _{in}
173	Z _{out} , Rdst _{in} , End

Figure 7.21 Microinstruction for Add (Rsrc)+,Rdst.

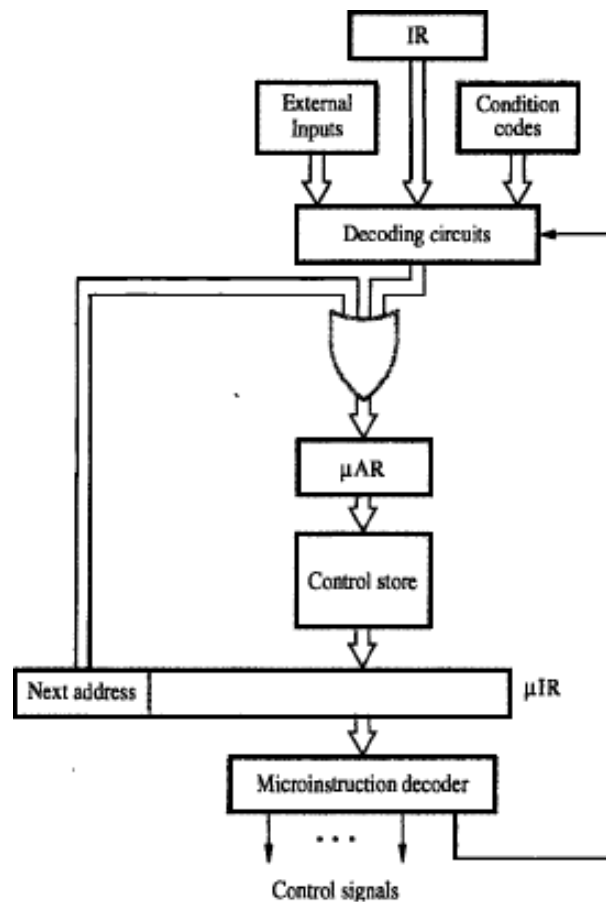


Figure 7.22 Microinstruction-sequencing organization.

- **Drawback of previous organization:**

- The microprogram requires several branch microinstructions which perform no useful operation. Thus, they detract from the operating-speed of the computer.

- **Solution:**

- Include an address-field as a part of every microinstruction to indicate the location of the next microinstruction to be fetched. (Thus, every microinstruction becomes a branch microinstruction).

- The flexibility of this approach comes at the expense of additional bits for the address-field (Fig 7.22).
- **Advantage:** Separate branch microinstructions are virtually eliminated. (Figure 7.23-24).
- **Disadvantage:** Additional bits for the address field (around 1/6).
- There is no need for a counter to keep track of sequential address. Hence, μPC is replaced with μAR .
- The next-address bits are fed through the OR gate to the μAR , so that the address can be modified on the basis of the data in the IR, external inputs and condition-codes.
- The decoding circuits generate the starting-address of a given microroutine on the basis of the opcode in the IR. ($\mu AR \rightarrow$ Microinstruction Address Register).

Microinstruction

F0	F1	F2	F3
F0 (8 bits)	F1 (3 bits)	F2 (3 bits)	F3 (3 bits)
Address of next microinstruction	000: No transfer 001: PC _{out} 010: MDR _{out} 011: Z _{out} 100: Rsrc _{out} 101: Rdst _{out} 110: TEMP _{out}	000: No transfer 001: PC _{in} 010: IR _{in} 011: Z _{in} 100: Rsrc _{in} 101: Rdst _{in}	000: No transfer 001: MAR _{in} 010: MDR _{in} 011: TEMP _{in} 100: Y _{in}

F4	F5	F6	F7
F4 (4 bits)	F5 (2 bits)	F6 (1 bit)	F7 (1 bit)
0000: Add 0001: Sub ⋮ 1111: XOR	00: No action 01: Read 10: Write	0: SelectY 1: Select4	0: No action 1: WMFC

F8	F9	F10
F8 (1 bit)	F9 (1 bit)	F10 (1 bit)
0: NextAdrs 1: InstDec	0: No action 1: OR _{mode}	0: No action 1: OR _{indsrc}

Figure 7.23 Format for microinstructions in the example of Section 7.5.3.

Octal address	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
000	00000001	001	011	001	0000	01	1	0	0	0	0
001	00000010	011	001	100	0000	00	0	1	0	0	0
002	00000011	010	010	000	0000	00	0	0	0	0	0
003	00000000	000	000	000	0000	00	0	0	1	1	0
121	01010010	100	011	001	0000	01	1	0	0	0	0
122	01111000	011	100	000	0000	00	0	1	0	0	1
170	01111001	010	000	001	0000	01	0	1	0	0	0
171	01111010	010	000	100	0000	00	0	0	0	0	0
172	01111011	101	011	000	0000	00	0	0	0	0	0
173	00000000	011	101	000	0000	00	0	0	0	0	0

Figure 7.24 Implementation of the microroutine of Figure 7.21 using a next-microinstruction address field. (See Figure 7.23 for encoded signals.)

COMPUTER ORGANIZATION

PREFETCHING MICROINSTRUCTIONS

• **Disadvantage of Microprogrammed Control:** Slower operating-speed because of the time it takes to fetch microinstructions from the control-store.

Solution: Faster operation is achieved if the next microinstruction is pre-fetched while the current one is being executed.

Emulation

- The main function of microprogrammed control is to provide a means for simple, flexible and relatively inexpensive execution of machine instruction.
- Its flexibility in using a machine's resources allows diverse classes of instructions to be implemented.
- Suppose we add to the instruction-repository of a given computer M1, an entirely new set of instructions that is in fact the instruction-set of a different computer M2.
- Programs written in the machine language of M2 can be then be run on computer M1 i.e. M1 emulates M2.
- Emulation allows us to replace obsolete equipment with more up-to-date machines.
- If the replacement computer fully emulates the original one, then no software changes have to be made to run existing programs.
- Emulation is easiest when the machines involved have similar architectures.

COMPUTER ORGANIZATION

Problem 1:

Why is the Wait-for-memory-function-completed step needed for reading from or writing to the main memory?

Solution:

The WMFC step is needed to synchronize the operation of the processor and the main memory.

Problem 2:

For the single bus organization, write the complete control sequence for the instruction: Move (R1), R1

Solution:

- 1) $PC_{out}, MAR_{in}, Read, Select4, Add, Z_{in}$
- 2) $Z_{out}, PC_{in}, Y_{in}, WMFC$
- 3) MDR_{out}, IR_{in}
- 4) $R1_{out}, MAR_{in}, Read$
- 5) $MDR_{inE}, WMFC$
- 6) $MDR_{out}, R2_{in}, End$

Problem 3:

Write the sequence of control steps required for the single bus organization in each of the following instructions:

- a) Add the immediate number NUM to register R1.
- b) Add the contents of memory-location NUM to register R1.
- c) Add the contents of the memory-location whose address is at memory-location NUM to register R1.

Assume that each instruction consists of two words. The first word specifies the operation and N the addressing mode, and the second word contains the number NUM

Solution:

- (a)
 1. $PC_{out}, MAR_{in}, Read, Select4, Add, Z_{in}$
 2. $Z_{out}, PC_{in}, Y_{in}, WMFC$
 3. MDR_{out}, IR_{in}
 4. $PC_{out}, MAR_{in}, Read, Select4, Add, Z_{in}$
 5. Z_{out}, PC_{in}, Y_{in}
 6. $R1_{out}, Y_{in}, WMFC$
 7. $MDR_{out}, SelectY, Add, Z_{in}$
 8. $Z_{out}, R1_{in}, End$
- (b)
 - 1-4. Same as in (a)
 5. $Z_{out}, PC_{in}, WMFC$
 6. $MDR_{out}, MAR_{in}, Read$
 7. $R1_{out}, Y_{in}, WMFC$
 8. MDR_{out}, Add, Z_{in}
 9. $Z_{out}, R1_{in}, End$
- (c)
 - 1-5. Same as in (b)
 6. $MDR_{out}, MAR_{in}, Read, WMFC$
 - 7-10. Same as 6-9 in (b)

Problem 4:

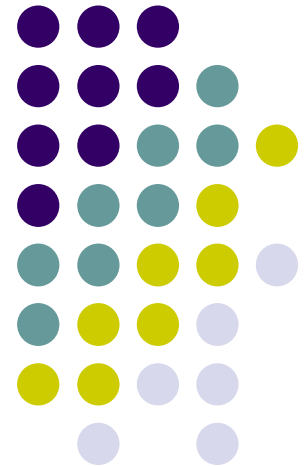
Show the control steps for the Branch on Negative instruction for a processor with three-bus organization of the data path

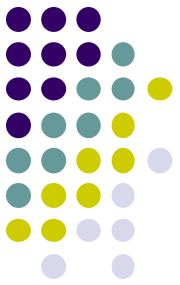
Solution:

- 1 $PC_{out}, R=B, MAR_{in}, Read, IncPC$
- 2 $WMFC$
- 3 $MDR_{outB}, R=B, IR_{in}$
4. $PC_{out}, Offset \text{ field of } IR_{out}, Add, \text{ If } N - 1 \text{ then } PC_{in}, End$

VTUNOTESBYSRI

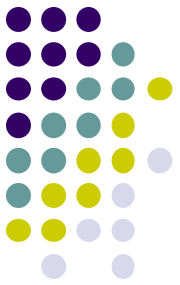
Module 1. Basic Structure of Computers





Outline

- Basic Operational Concepts
- Bus Structures
- Basic Performance Equation
- Performance Measurement
- Machine Instructions and Programs
- Memory Operations
- Memory Operations
- Addressing Modes
- Encoding of Machine Instructions



Learning Objectives

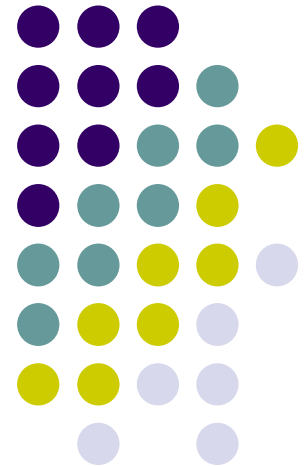
- Understand the basics of computer organization: structure and operation of computers and their peripherals
- Understand the concepts of programs as sequences or machine instructions.
- To analyse various addressing modes and machine instructions
- To study basic IO operations and Stack operations

Computer organization vs Computer Architecture



- Computer architecture refers to those attributes of a system visible to a programmer
- Computer organization refers to the operational units and their interconnections that realize the architectural specifications.

Functional Units



Functional Units

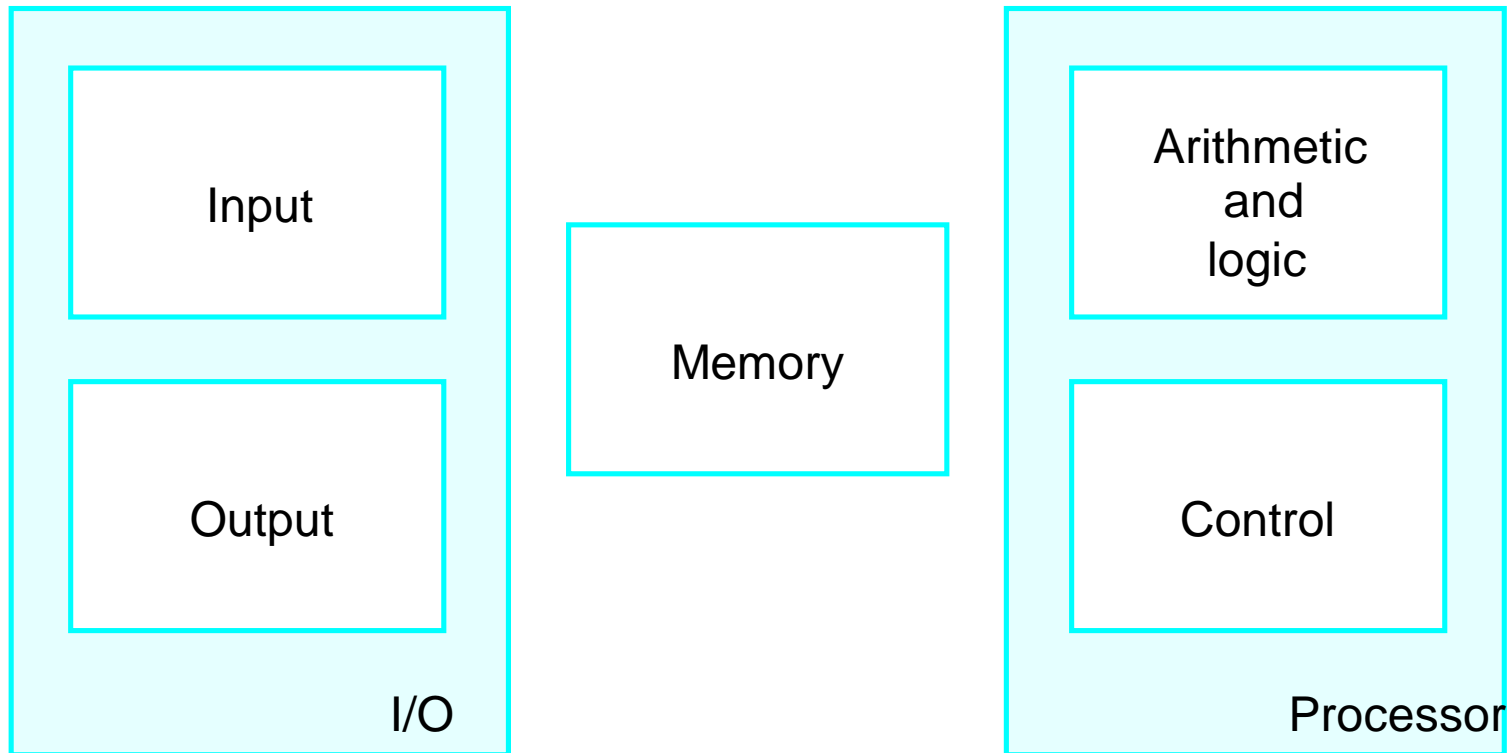
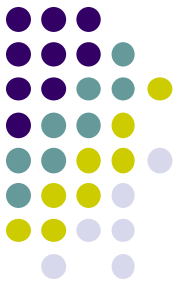
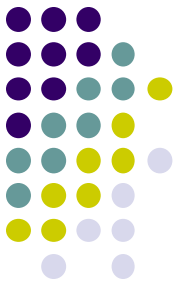
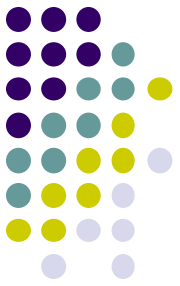


Figure 1.1. Basic functional units of a computer.

Information Handled by a Computer



- Instructions/machine instructions
 - Govern the transfer of information within a computer as well as between the computer and its I/O devices
 - Specify the arithmetic and logic operations to be performed
 - Program
- Data
 - Used as operands by the instructions
 - Source program
- Encoded in binary code – 0 and 1



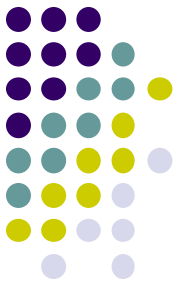
Memory Unit

- Store programs and data
- Two classes of storage
 - Primary storage
 - ❖ Fast
 - ❖ Programs must be stored in memory while they are being executed
 - ❖ Large number of semiconductor storage cells
 - ❖ Processed in words
 - ❖ Address
 - ❖ RAM and memory access time
 - ❖ Memory hierarchy – cache, main memory
 - Secondary storage – larger and cheaper

Arithmetic and Logic Unit (ALU)



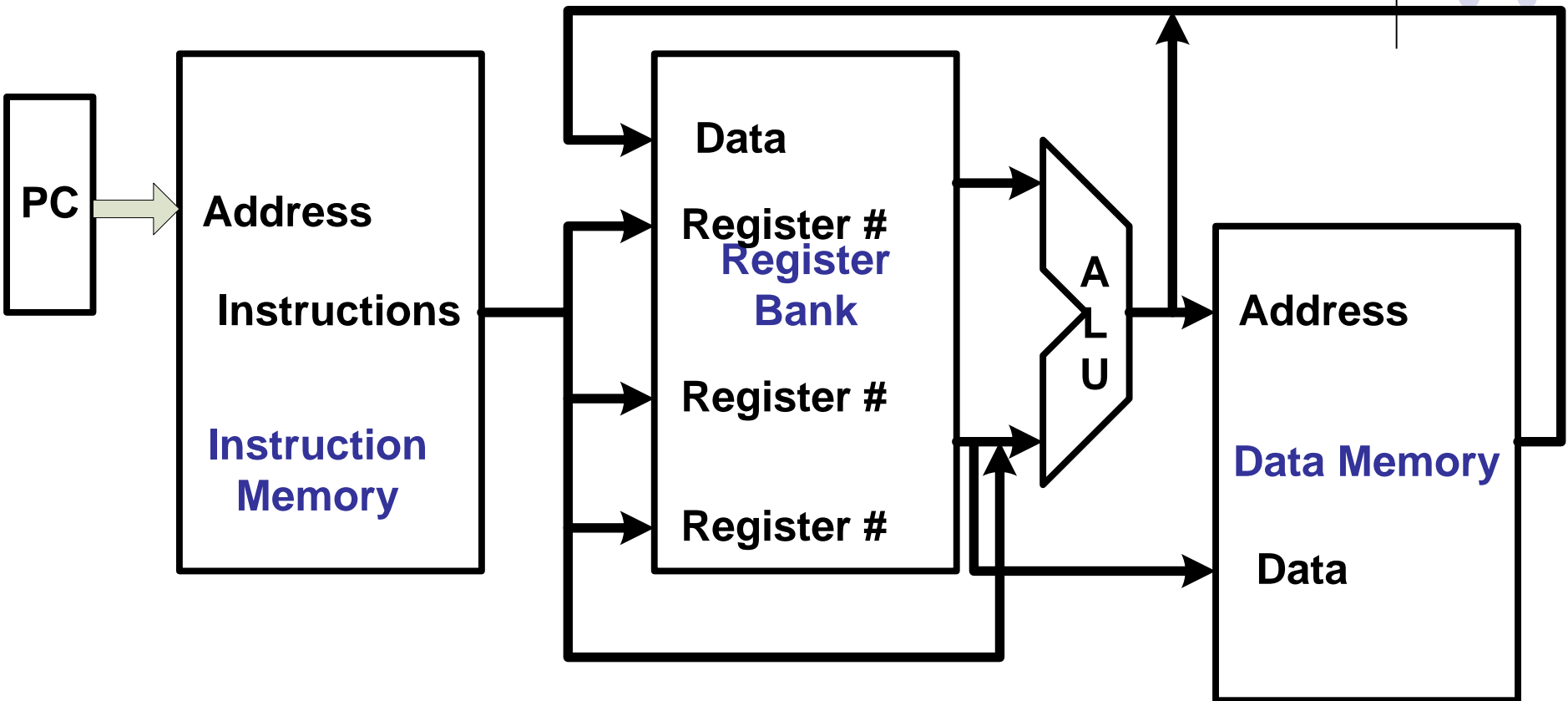
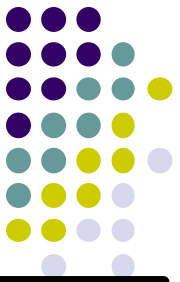
- Most computer operations are executed in ALU of the processor.
- Load the operands into memory – bring them to the processor – perform operation in ALU – store the result back to memory or retain in the processor.
- Registers
- Fast control of ALU



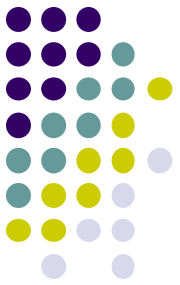
Control Unit

- All computer operations are controlled by the control unit.
- The timing signals that govern the I/O transfers are also generated by the control unit.
- Control unit is usually distributed throughout the machine instead of standing alone.
- Operations of a computer:
 - Accept information in the form of programs and data through an input unit and store it in the memory
 - Fetch the information stored in the memory, under program control, into an ALU, where the information is processed
 - Output the processed information through an output unit
 - Control all activities inside the machine through a control unit

The processor : Data Path and Control



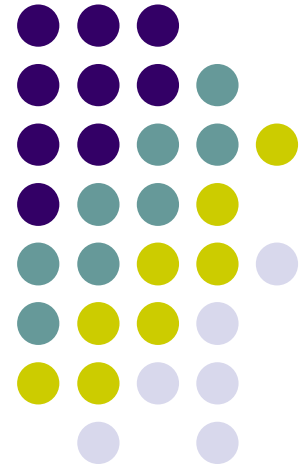
- Two types of functional units:
 - elements that operate on data values (combinational)
 - elements that contain state (state elements)

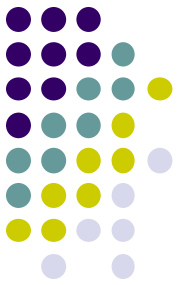


Five Execution Steps

Step name	Action for R-type instructions	Action for Memory-reference Instructions	Action for branches	Action for jumps
Instruction fetch	IR = MEM[PC] PC = PC + 4			
Instruction decode/ register fetch	A = Reg[IR[25-21]] B = Reg[IR[20-16]] ALUOut = PC + (sign extend (IR[15-0])<<2)			
Execution, address computation, branch/jump completion	ALUOut = A op B	ALUOut = A+sign extend(IR[15-0])	IF(A==B) Then PC=ALUOut	PC=PC[31-28] (IR[25-0]<<2)
Memory access or R-type completion	Reg[IR[15-11]] = ALUOut	Load:MDR =Mem[ALUOut] or Store:Mem[ALUOut] = B		
Memory read completion		Load: Reg[IR[20-16]] = MDR		

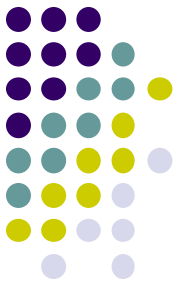
Basic Operational Concepts





Review

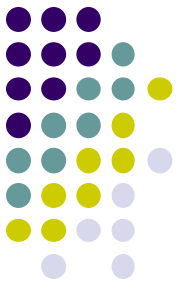
- Activity in a computer is governed by instructions.
- To perform a task, an appropriate program consisting of a list of instructions is stored in the memory.
- Individual instructions are brought from the memory into the processor, which executes the specified operations.
- Data to be used as operands are also stored in the memory.



A Typical Instruction

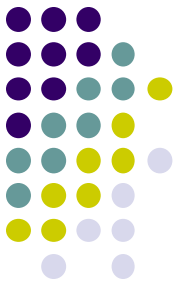
- **Add LOCA, R0**
- Add the operand at memory location LOCA to the operand in a register R0 in the processor.
- Place the sum into register R0.
- The original contents of LOCA are preserved.
- The original contents of R0 is overwritten.
- Instruction is fetched from the memory into the processor – the operand at LOCA is fetched and added to the contents of R0 – the resulting sum is stored in register R0.

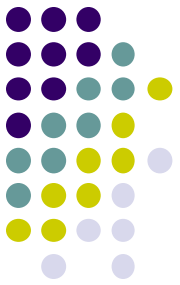
Separate Memory Access and ALU Operation



- Load LOCA, R1
- Add R1, R0
- Whose contents will be overwritten?

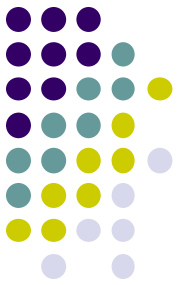
Connection Between the Processor and the Memory





Registers

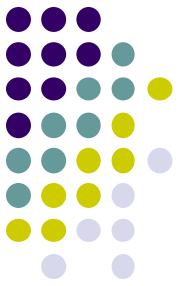
- Instruction register (IR)
- Program counter (PC)
- General-purpose register ($R_0 - R_{n-1}$)
- Memory address register (MAR)
- Memory data register (MDR)



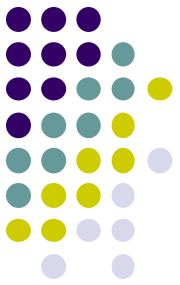
Typical Operating Steps

- Programs reside in the memory through input devices
- PC is set to point to the first instruction
- The contents of PC are transferred to MAR
- A Read signal is sent to the memory
- The first instruction is read out and loaded into MDR
- The contents of MDR are transferred to IR
- Decode and execute the instruction

Typical Operating Steps (Cont')

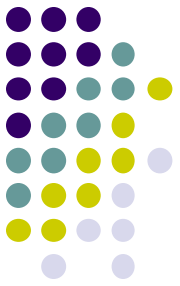


- Get operands for ALU
 - General-purpose register
 - Memory (address to MAR – Read – MDR to ALU)
- Perform operation in ALU
- Store the result back
 - To general-purpose register
 - To memory (address to MAR, result to MDR – Write)
- During the execution, PC is incremented to the next instruction



Interrupt

- Normal execution of programs may be preempted if some device requires urgent servicing.
- The normal execution of the current program must be interrupted – the device raises an *interrupt* signal.
- Interrupt-service routine
- Current system information backup and restore (PC, general-purpose registers, control information, specific information)

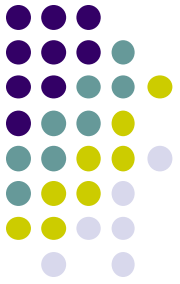


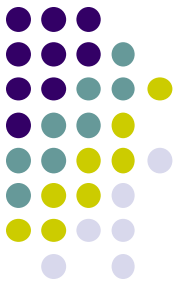
Bus Structures

- There are many ways to connect different parts inside a computer together.
- A group of lines that serves as a connecting path for several devices is called a *bus*.
- Address/data/control

Bus Structure

- Single-bus

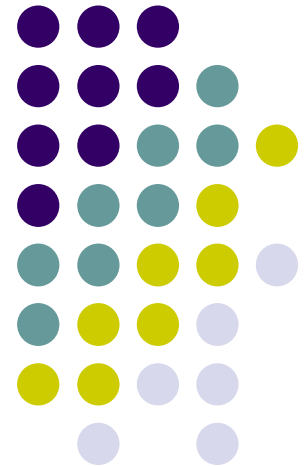


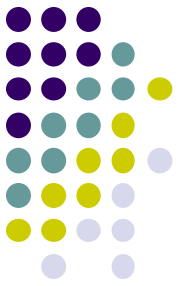


Speed Issue

- Different devices have different transfer/operate speed.
- If the speed of bus is bounded by the slowest device connected to it, the efficiency will be very low.
- How to solve this?
- A common approach – use buffers.

Performance

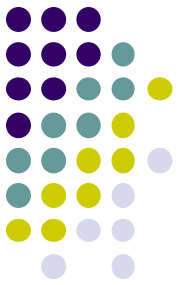




Performance

- The most important measure of a computer is how quickly it can execute programs.
- Three factors affect performance:
 - Hardware design
 - Instruction set
 - Compiler

Performance



- Processor time to execute a program depends on the hardware involved in the execution of individual machine instructions.

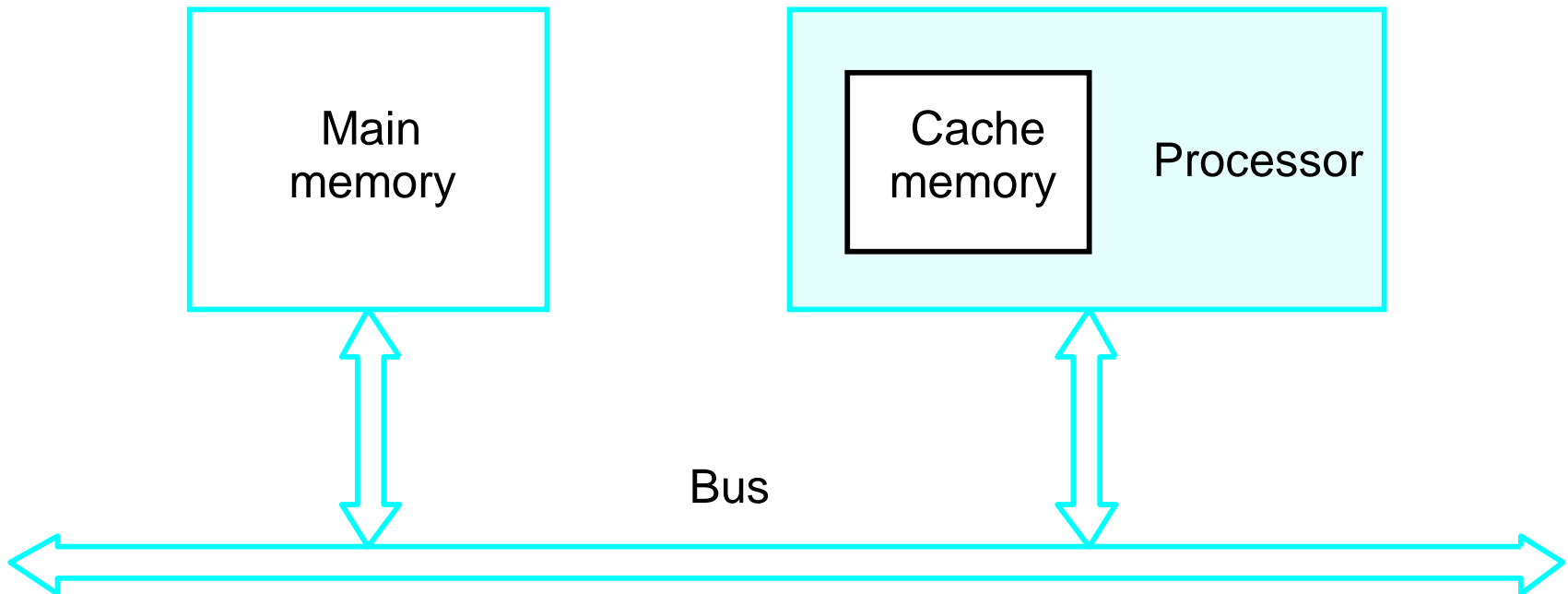
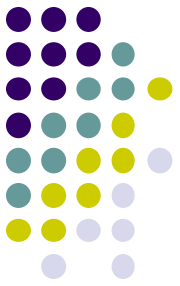
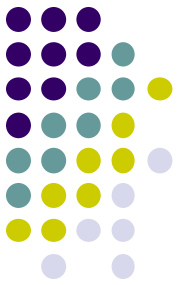


Figure 1.5. The processor cache.



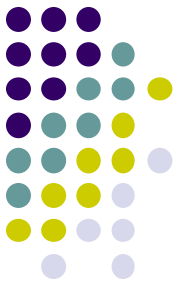
Performance

- The processor and a relatively small cache memory can be fabricated on a single integrated circuit chip.
- Speed
- Cost
- Memory management



Processor Clock

- Clock, clock cycle, and clock rate
- The execution of each instruction is divided into several steps, each of which completes in one clock cycle.
- Hertz – cycles per second

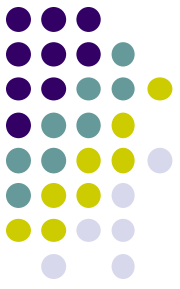


Basic Performance Equation

- T – processor time required to execute a program that has been prepared in high-level language
- N – number of actual machine language instructions needed to complete the execution (note: loop)
- S – average number of basic steps needed to execute one machine instruction. Each step completes in one clock cycle
- R – clock rate
- Note: these are not independent to each other

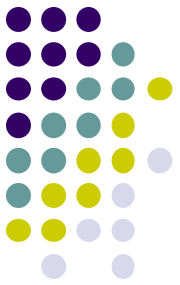
$$T = \frac{N \times S}{R}$$

How to improve T?



Pipeline

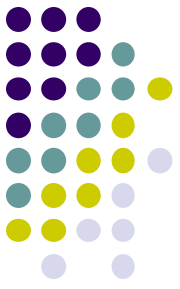
- Instructions are not necessarily executed one after another.
- The value of S doesn't have to be the number of clock cycles to execute one instruction.
- Pipelining – overlapping the execution of successive instructions.



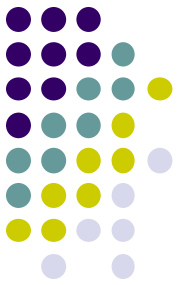
Clock Rate

- Increase clock rate
 - Improve the integrated-circuit (IC) technology to make the circuits faster
 - Reduce the amount of processing done in one basic step (however, this may increase the number of basic steps needed)
- Increases in R that are entirely caused by improvements in IC technology affect all aspects of the processor's operation equally except the time to access the main memory.

Compiler



- A compiler translates a high-level language program into a sequence of machine instructions.
- To reduce N , we need a suitable machine instruction set and a compiler that makes good use of it.
- Goal – reduce $N \times S$
- A compiler may not be designed for a specific processor; however, a high-quality compiler is usually designed for, and with, a specific processor.



Performance Measurement

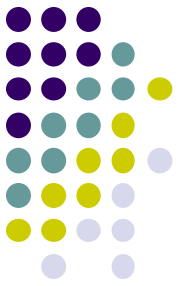
- T is difficult to compute.
- Measure computer performance using benchmark programs.
- System Performance Evaluation Corporation (SPEC) selects and publishes representative application programs for different application domains, together with test results for many commercially available computers.
- Compile and run (no simulation)
- Reference computer

$$SPEC\ rating = \frac{\text{Running time on the reference computer}}{\text{Running time on the computer under test}}$$

$$SPEC\ rating = \left(\prod_{i=1}^n SPEC_i \right)^{\frac{1}{n}}$$

Machine Instructions and Programs





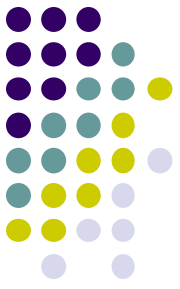
Objectives

- Machine instructions and program execution, including branching and subroutine call and return operations.
- Addressing methods for accessing register and memory operands.
- Assembly language for representing machine instructions, data, and programs.
- Program-controlled Input/Output operations.

Memory Locations, Addresses, and Operations



Memory Location, Addresses, and Operation



- Memory consists of many millions of storage cells, each of which can store 1 bit.
- Data is usually accessed in n -bit groups. n is called word length.

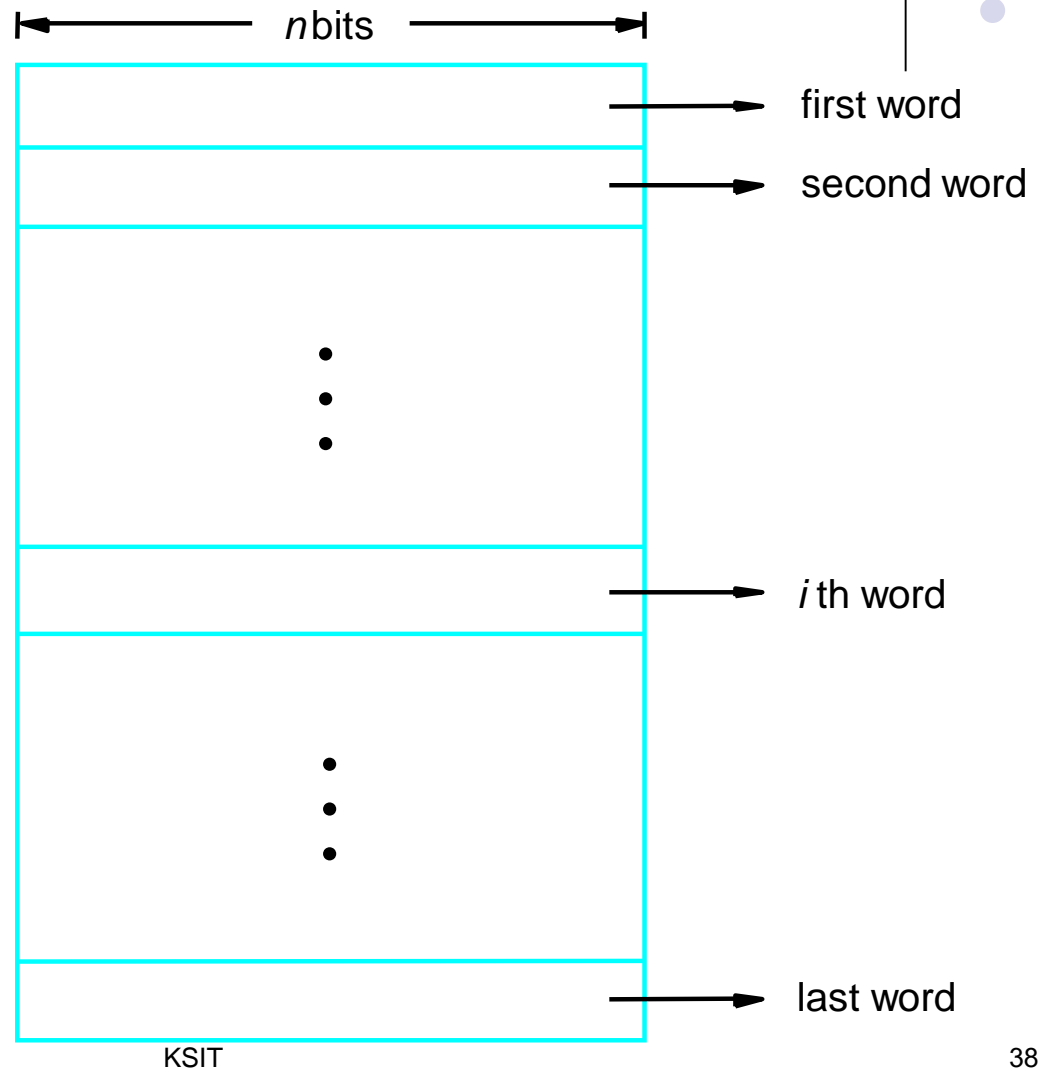
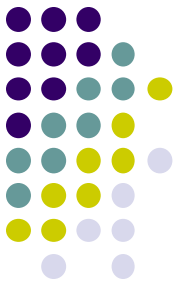
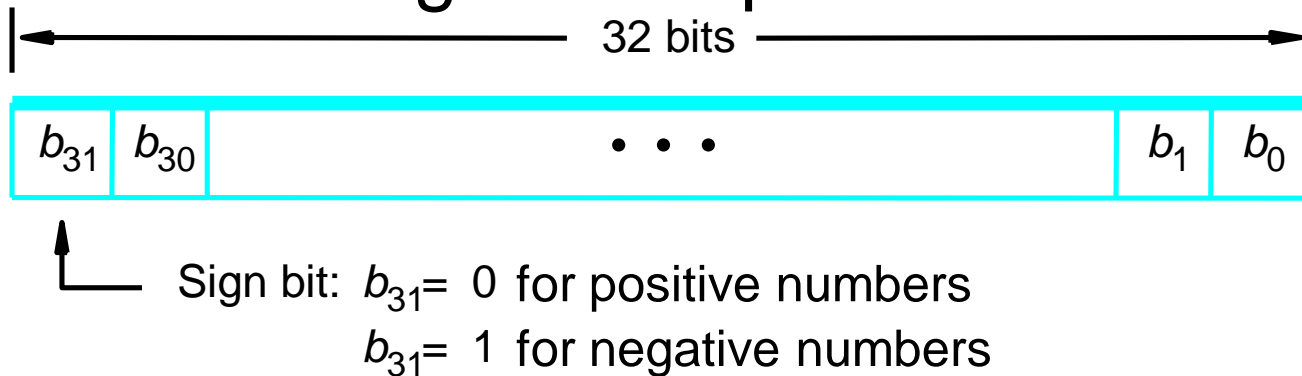


Figure 2.5. Memory words.

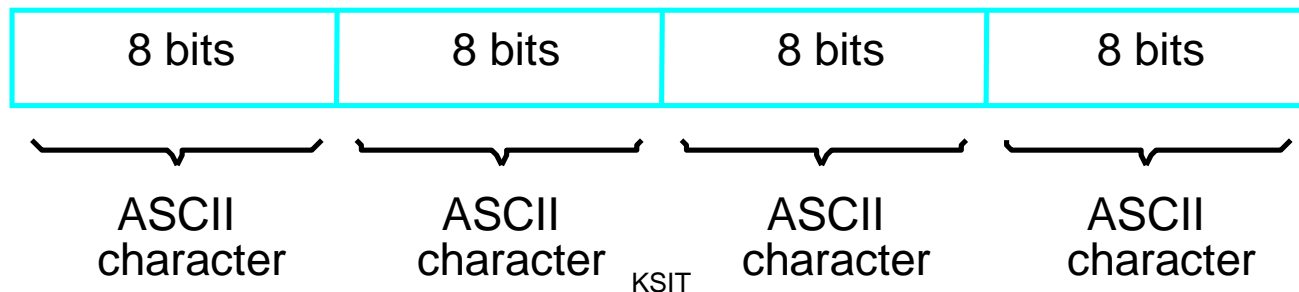
Memory Location, Addresses, and Operation



- 32-bit word length example



(a) A signed integer



(b) Four characters

Memory Location, Addresses, and Operation



- To retrieve information from memory, either for one word or one byte (8-bit), addresses for each location are needed.
- A k -bit address memory has 2^k memory locations, namely $0 - 2^k - 1$, called memory space.
- 24-bit memory: $2^{24} = 16,777,216 = 16\text{M}$ ($1\text{M} = 2^{20}$)
- 32-bit memory: $2^{32} = 4\text{G}$ ($1\text{G} = 2^{30}$)
- $1\text{K(kilo)} = 2^{10}$
- $1\text{T(tera)} = 2^{40}$

Memory Location, Addresses, and Operation



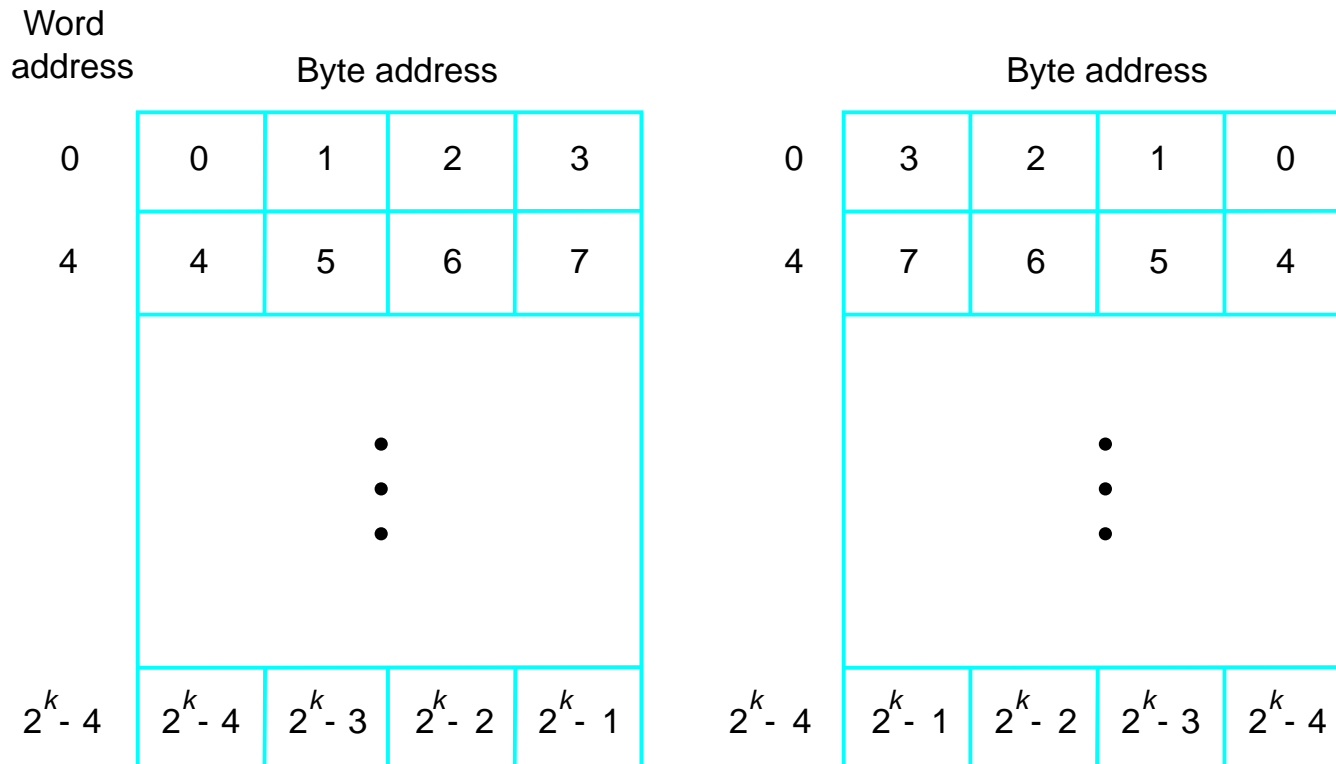
- It is impractical to assign distinct addresses to individual bit locations in the memory.
- The most practical assignment is to have successive addresses refer to successive byte locations in the memory – byte-addressable memory.
- Byte locations have addresses 0, 1, 2, ... If word length is 32 bits, they successive words are located at addresses 0, 4, 8,...

Big-Endian and Little-Endian Assignments



Big-Endian: lower byte addresses are used for the most significant bytes of the word

Little-Endian: opposite ordering. lower byte addresses are used for the less significant bytes of the word



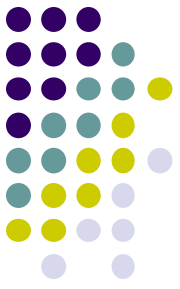
(a) Big-endian assignment

(b) Little-endian assignment

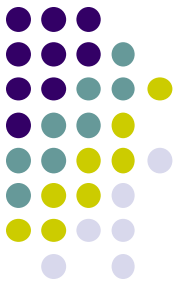
KSIT

Figure 2.7. Byte and word addressing.

Memory Location, Addresses, and Operation



- Address ordering of bytes
- Word alignment
 - Words are said to be aligned in memory if they begin at a byte addr. that is a multiple of the num of bytes in a word.
 - 16-bit word: word addresses: 0, 2, 4,....
 - 32-bit word: word addresses: 0, 4, 8,....
 - 64-bit word: word addresses: 0, 8,16,....
- Access numbers, characters, and character strings

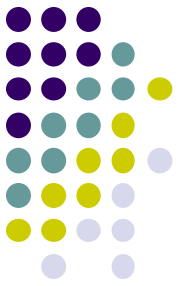


Memory Operation

- Load (or Read or Fetch)
 - Copy the content. The memory content doesn't change.
 - Address – Load
 - Registers can be used
- Store (or Write)
 - Overwrite the content in memory
 - Address and Data – Store
 - Registers can be used

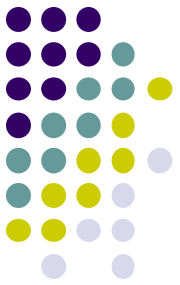
Instruction and Instruction Sequencing





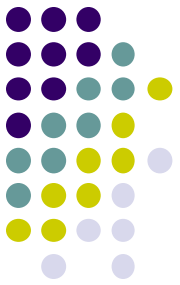
“Must-Perform” Operations

- Data transfers between the memory and the processor registers
- Arithmetic and logic operations on data
- Program sequencing and control
- I/O transfers



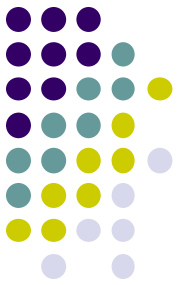
Register Transfer Notation

- Identify a location by a symbolic name standing for its hardware binary address (LOC, R0,...)
- Contents of a location are denoted by placing square brackets around the name of the location ($R1 \leftarrow [LOC]$, $R3 \leftarrow [R1] + [R2]$)
- Register Transfer Notation (RTN)



Assembly Language Notation

- Represent machine instructions and programs.
- Move LOC, $R1 = R1 \leftarrow [LOC]$
- Add R1, R2, $R3 = R3 \leftarrow [R1] + [R2]$



CPU Organization

- Single Accumulator
 - Result usually goes to the Accumulator
 - Accumulator has to be saved to memory quite often
- General Register
 - Registers hold operands thus reduce memory traffic
 - Register bookkeeping
- Stack
 - Operands and result are always in the stack

Instruction Formats



- Three-Address Instructions

- ADD R1, R2, R3 $R1 \leftarrow R2 + R3$

- Two-Address Instructions

- ADD R1, R2 $R1 \leftarrow R1 + R2$

- One-Address Instructions

- ADD M $AC \leftarrow AC + M[AR]$

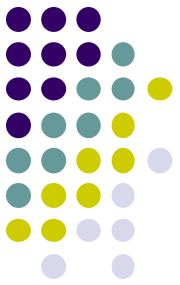
- Zero-Address Instructions

- ADD $TOS \leftarrow TOS + (TOS - 1)$

- RISC Instructions

- Lots of registers. Memory is restricted to Load & Store





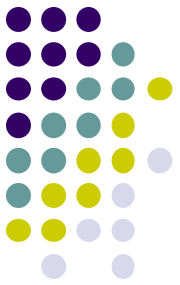
Instruction Formats

Example: Evaluate $(A+B) * (C+D)$

- Three-Address

- | | | | |
|----|-----|-----------|-------------------------------|
| 1. | ADD | R1, A, B | ; $R1 \leftarrow M[A] + M[B]$ |
| 2. | ADD | R2, C, D | ; $R2 \leftarrow M[C] + M[D]$ |
| 3. | MUL | X, R1, R2 | ; $M[X] \leftarrow R1 * R2$ |





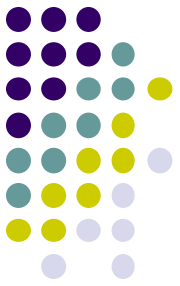
Instruction Formats

Example: Evaluate $(A+B) * (C+D)$

- Two-Address

- | | | | |
|----|-----|--------|-----------------------------|
| 1. | MOV | R1, A | ; $R1 \leftarrow M[A]$ |
| 2. | ADD | R1, B | ; $R1 \leftarrow R1 + M[B]$ |
| 3. | MOV | R2, C | ; $R2 \leftarrow M[C]$ |
| 4. | ADD | R2, D | ; $R2 \leftarrow R2 + M[D]$ |
| 5. | MUL | R1, R2 | ; $R1 \leftarrow R1 * R2$ |
| 6. | MOV | X, R1 | ; $M[X] \leftarrow R1$ |

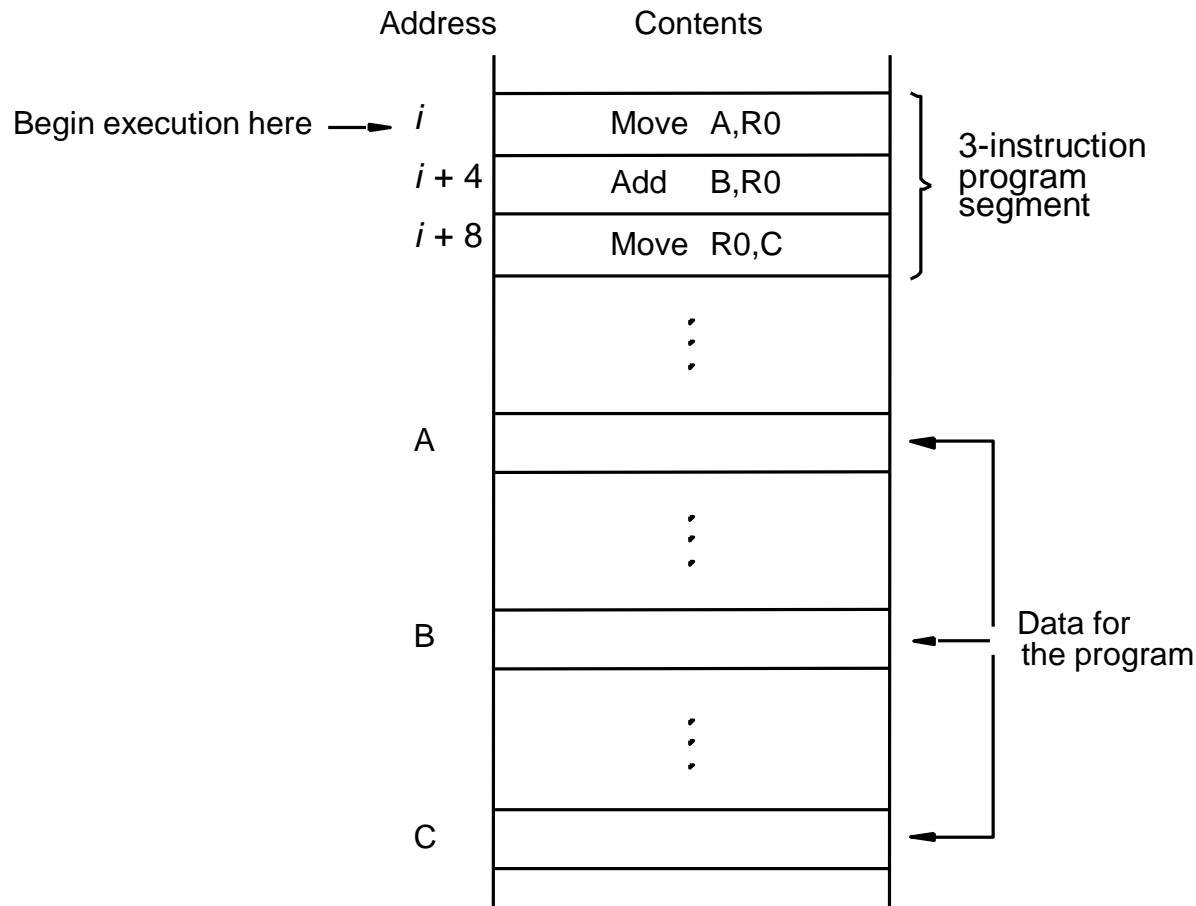
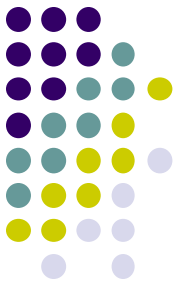




Using Registers

- Registers are faster
- Shorter instructions
 - The number of registers is smaller (e.g. 32 registers need 5 bits)
- Potential speedup
- Minimize the frequency with which data is moved back and forth between the memory and processor registers.

Instruction Execution and Straight-Line Sequencing



Assumptions:

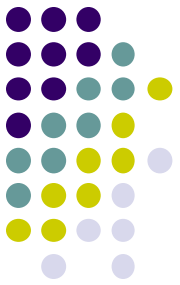
- One memory operand per instruction
- 32-bit word length
- Memory is byte addressable
- Full memory address can be directly specified in a single-word instruction

Two-phase procedure

- Instruction fetch
- Instruction execute

Page 43

Branching



i	Move	NUM1,R0
$i + 4$	Add	NUM2,R0
$i + 8$	Add	NUM3,R0
	⋮	
$i + 4n - 4$	Add	NUM n ,R0
$i + 4n$	Move	R0,SUM
	⋮	
SUM		
NUM1		
NUM2		
	⋮	
NUM n		

Figure 2.9. A straight-line program for adding n numbers.

Branching

Branch target

Conditional branch

Program loop

LOOP

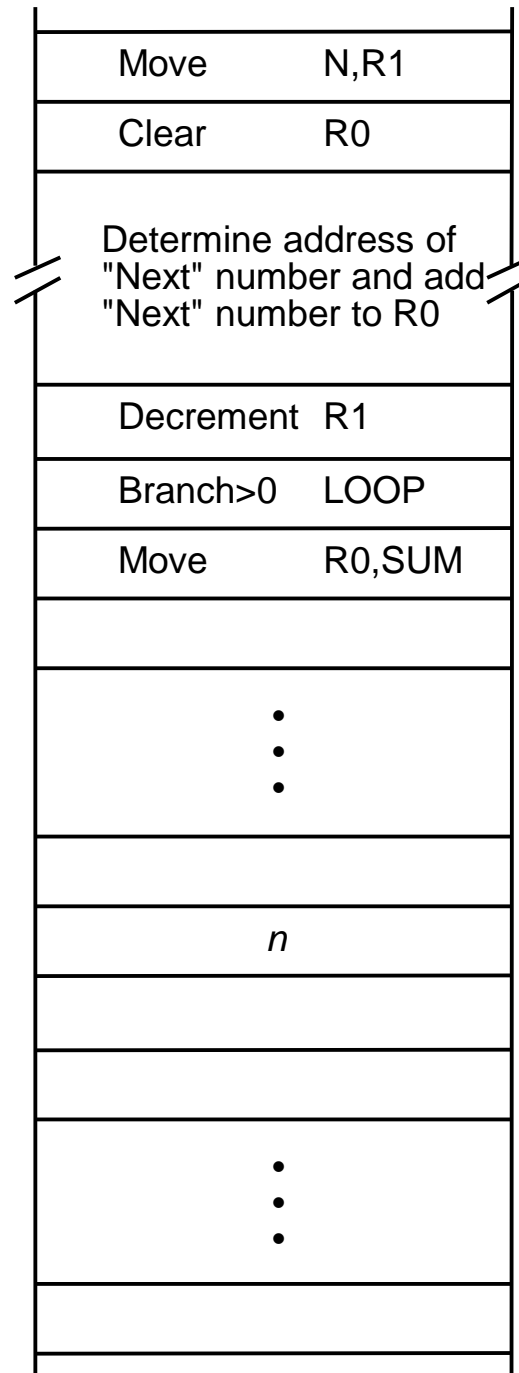
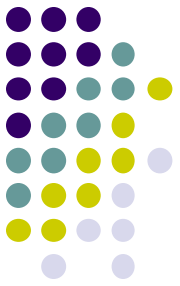


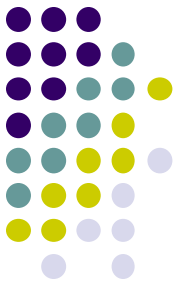
Figure 2.10. Using a loop to add n numbers.



Condition Codes

- Condition code flags
- Condition code register / status register
- N (negative)
- Z (zero)
- V (overflow)
- C (carry)
- Different instructions affect different flags

Conditional Branch Instructions



- Example:

- A: 1 1 1 1 0 0 0 0

- B: 0 0 0 1 0 1 0 0

A: 1 1 1 1 0 0 0 0

+(-B): 1 1 1 0 1 1 0 0

1 1 0 1 1 1 0 0

C = 1

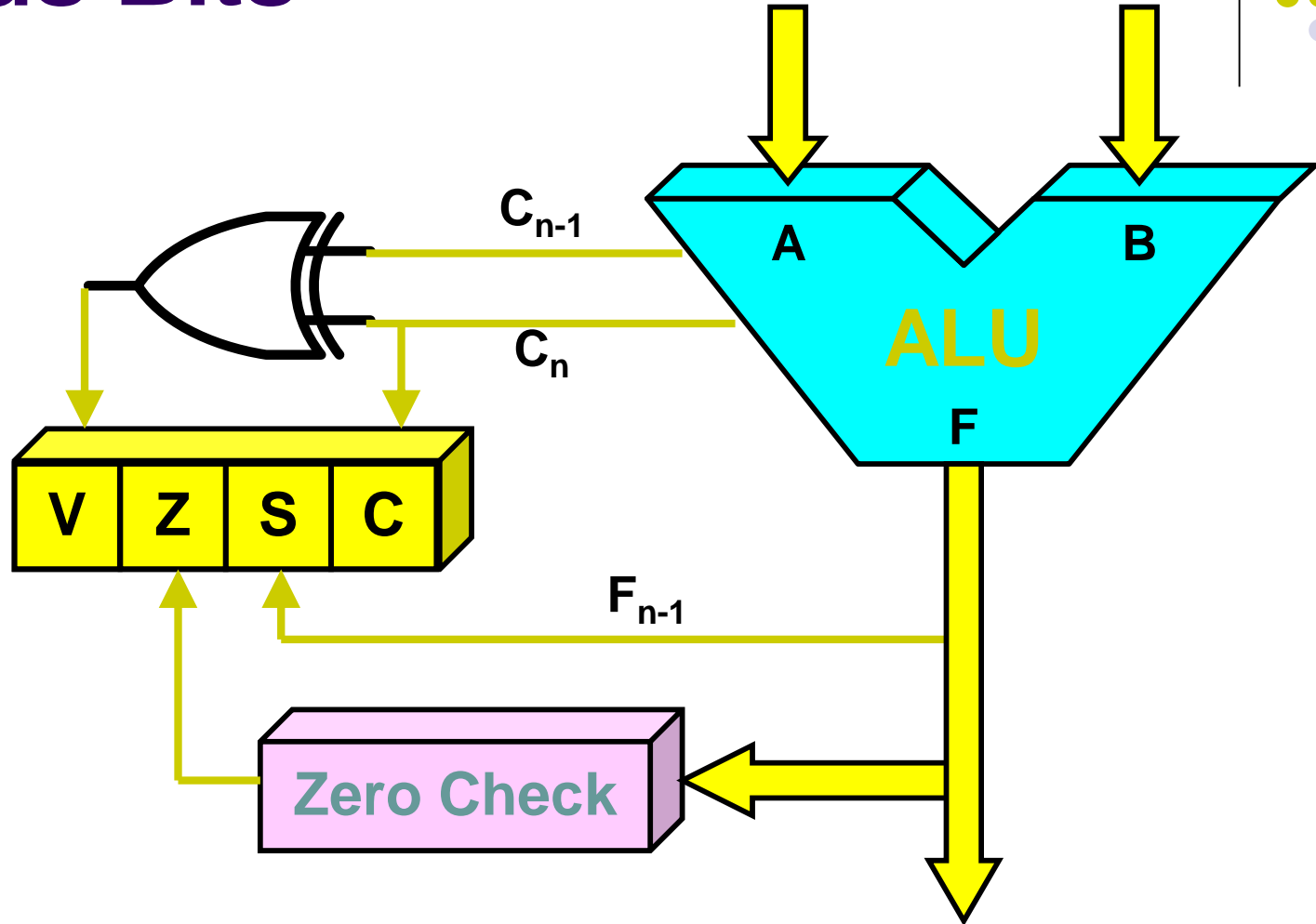
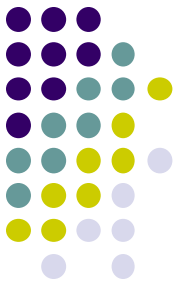
Z = 0

S = 1

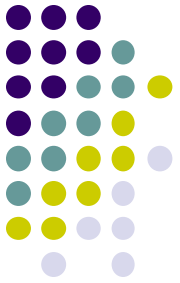
V = 0



Status Bits



Addressing Modes



Generating Memory Addresses



- How to specify the address of branch target?
- Can we give the memory operand address directly in a single Add instruction in the loop?
- Use a register to hold the address of NUM1; then increment by 4 on each pass through the loop.

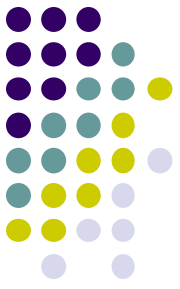
Addressing Modes



- Implementation of variables and constants
 1. **Register Addressing Mode** : the operand is the contents of a processor register.
 2. **Absolute or Direct Addressing Mode** : the operand is in a memory location; the address of this location is given explicitly in the instruction.
 3. **Immediate Addressing Mode** : the operand is given explicitly in the instruction.



Addressing Modes



- Pointers

1. **Register Indirect Addressing Mode** : one of the register in the instruction holds the address of the operand

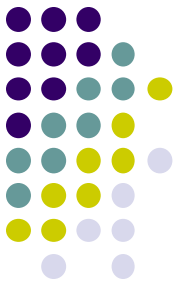
$EA=[Ri]$

2. **Indirect Addressing Mode** : the effective address of the operand is the memory location whose address appears in the instruction

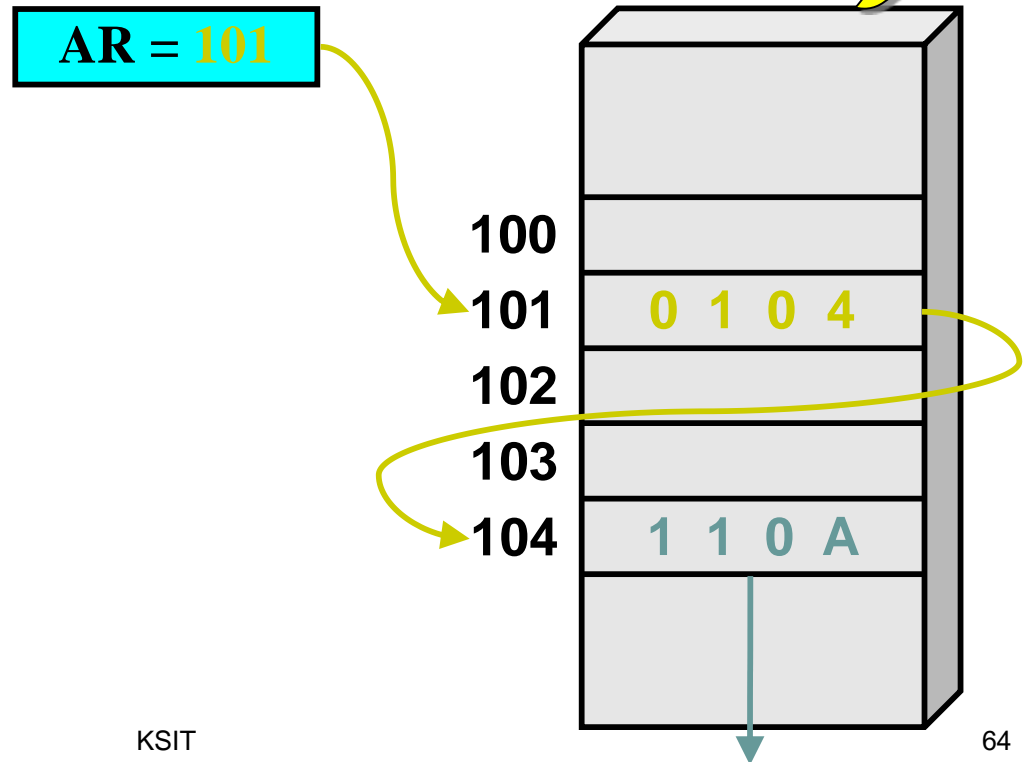
$EA=[LOC]$



Addressing Modes



- Indirect Address
 - Indicate the memory location that holds the address of the memory location that holds the data



Addressing Modes

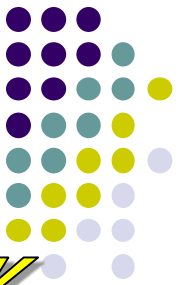


- Arrays

1. **Indexed Addressing Mode** : the EA of the operand is generated by adding a constant value to the contents of a register
$$X(R_i) \Rightarrow EA = [R_i] + X$$
2. **Base with Index** : $(R_i, R_j) \Rightarrow EA = [R_i] + [R_j]$
3. **Base with index and Offset** : $X(R_i, R_j) \Rightarrow EA = [R_i] + [R_j] + X$

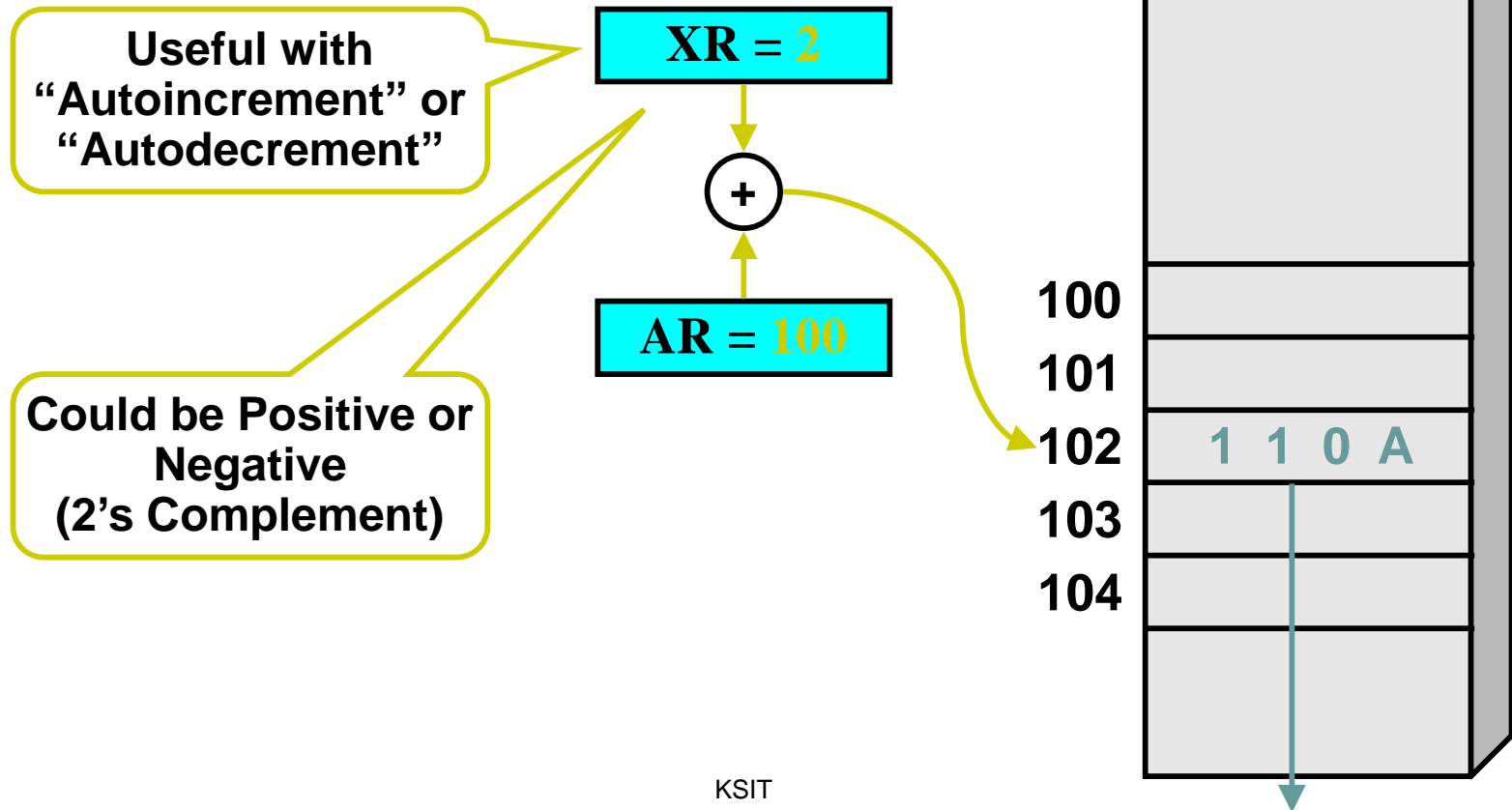


Addressing Modes



- Indexed

- $EA = \text{Index Register} + \text{Relative Addr}$



Addressing Modes

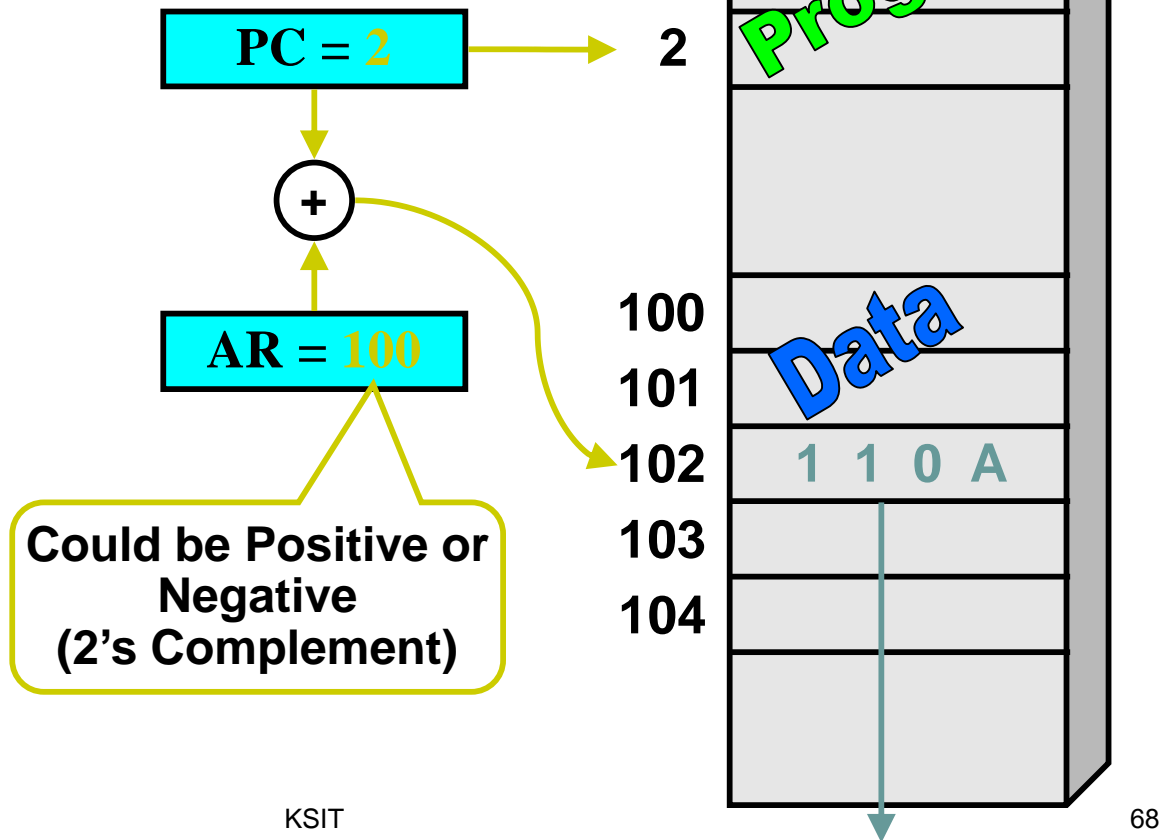


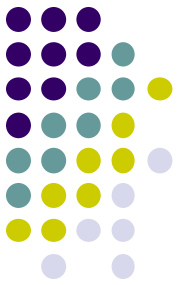
- Relative Addressing Mode
- The EA is determined by the index mode using the PC in place of the GPR Ri.
$$X(PC) \quad EA = [PC] + X$$
- Relative to the PC content
- Used to specify target address in branch instruction
Branch>0 LOOP
- This location is computed by specifying it as an offset from the current value of PC.
- Branch target may be either before or after the branch instruction, the offset is given as a signed num.



Addressing Modes

- Relative Address
 - $EA = PC + \text{Relative Addr}$

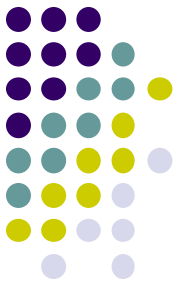




Addressing Modes

- The different ways in which the location of an operand is specified in an instruction are referred to as addressing modes.

Name	Assembler syntax	Addressingfunction
Immediate	#Value	Operand = Value
Register	R_i	$EA = R_i$
Absolute(Direct)	LOC	$EA = LOC$
Indirect	(R_i)	$EA = [R_i]$
	(LOC)	$EA = [LOC]$
Index	$X(R_i)$	$EA = [R_i] + X$
Basewith index	(R_i, R_j)	$EA = [R_i] + [R_j]$
Basewith index and offset	$X(R_i, R_j)$	$EA = [R_i] + [R_j] + X$
Relative	$X(PC)$	$EA = [PC] + X$
Autoincrement	$(R_i) +$	$EA = [R_i] ;$ Increment R_i
Autodecrement	$-(R_i)_{KSIT}$	Decrement $R_i ;$ $EA = [R_i]$



Additional Modes

- Autoincrement mode – the effective address of the operand is the contents of a register specified in the instruction. After accessing the operand, the contents of this register are automatically incremented to point to the next item in a list.
- $(R_i)+$. The increment is 1 for byte-sized operands, 2 for 16-bit operands, and 4 for 32-bit operands.
- Autodecrement mode: $-(R_i)$ – decrement first

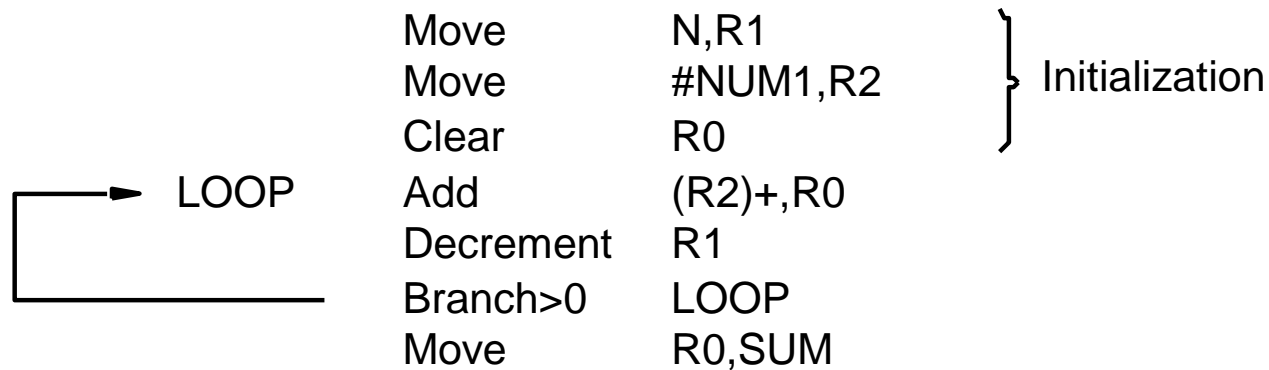
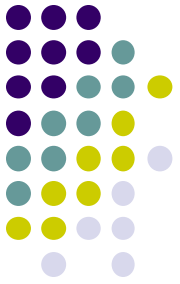


Figure 2.16. The Autoincrement addressing mode used in the program of Figure 2.12.

Assembly Language



Assembly Language

- Human understandable notation for machine level language
- Mnemonics – symbolic names
- Set of rules – syntax



Types of Instructions

- Data Transfer Instructions

Name	Mnemonic
Load	LD
Store	ST
Move	MOV
Exchange	XCH
Input	IN
Output	OUT
Push	PUSH
Pop	POP



Data value is not modified

Data Transfer Instructions



Mode	Assembly	Register Transfer
Direct address	LD ADR	$AC \leftarrow M[ADR]$
Indirect address	LD $@ADR$	$AC \leftarrow M[M[ADR]]$
Relative address	LD $\$ADR$	$AC \leftarrow M[PC+ADR]$
Immediate operand	LD $\#NBR$	$AC \leftarrow NBR$
Index addressing	LD $ADR(X)$	$AC \leftarrow M[ADR+XR]$
Register	LD $R1$	$AC \leftarrow R1$
Register indirect	LD $(R1)$	$AC \leftarrow M[R1]$
Autoincrement	LD $(R1)+$	$AC \leftarrow M[R1], R1 \leftarrow R1+1$



Data Manipulation Instructions



- Arithmetic
- Logical & Bit Manipulation
- Shift

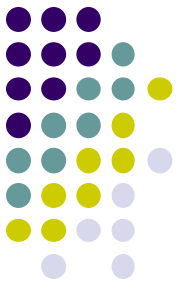
Name	Mnemonic
Clear	CLR
Complement	COM
AND	AND
OR	OR
Exclusive-OR	XOR
Clear carry	CLRC
Set carry	SETC
Complement carry	COMC
Enable interrupt	EI
Disable interrupt	DI

Name	Mnemonic
Increment	INC
Decrement	DEC
Add	ADD
Subtract	SUB
Multiply	MUL
Divide	DIV
Add with carry	ADDC
Subtract with borrow	SUBB
Negate	NEG

Name	Mnemonic
Logical shift right	SHR
Logical shift left	SHL
Arithmetic shift right	SHRA
Arithmetic shift left	SHLA
Rotate right	ROR
Rotate left	ROL
Rotate right through carry	RORC
Rotate left through carry	ROLC



Program Control Instructions



Name	Mnemonic
Branch	BR
Jump	JMP
Skip	SKP
Call	CALL
Return	RET
Compare (Subtract)	CMP
Test (AND)	TST

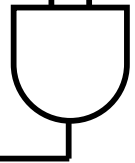
Subtract A – B but
don't store the result



1 0 1 1 0 0 0 1

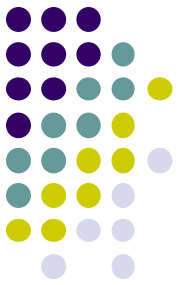
0 0 0 0 1 0 0 0

0 0 0 0 0 0 0 0



Mask

Conditional Branch Instructions



Mnemonic	Branch Condition	Tested Condition
BZ	Branch if zero	$Z = 1$
BNZ	Branch if not zero	$Z = 0$
BC	Branch if carry	$C = 1$
BNC	Branch if no carry	$C = 0$
BP	Branch if plus	$S = 0$
BM	Branch if minus	$S = 1$
BV	Branch if overflow	$V = 1$
BNV	Branch if no overflow	$V = 0$



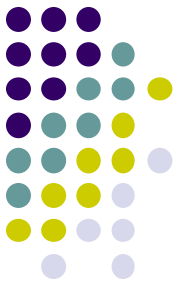
Assembler Directives



- It allows the programmer to specify other information needed to translate the source program
- `SUM EQU 200`
- It will not appear in the object code
- It simply informs assembler that the Name `SUM` should be replaced by the value 200

Basic Input/Output Operations



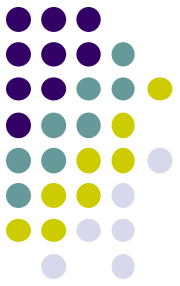


I/O

- The data on which the instructions operate are not necessarily already stored in memory.
- Data need to be transferred between processor and outside world (disk, keyboard, etc.)
- I/O operations are essential, the way they are performed can have a significant effect on the performance of the computer.

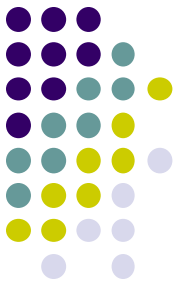
I/O

- Three methods
 1. Program controlled IO
 2. Interrupt IO
 3. Direct Memory Access DMA



Program-Controlled I/O

Example – keyboard Interfacing



- Read in character input from a keyboard and produce character output on a display screen.
 - Rate of data transfer (keyboard, display, processor)
 - Difference in speed between processor and I/O device creates the need for mechanisms to synchronize the transfer of data.
 - A solution: on output, the processor sends the first character and then waits for a signal from the display that the character has been received. It then sends the second character. Input is sent from the keyboard in a similar way.

Program-Controlled I/O Example

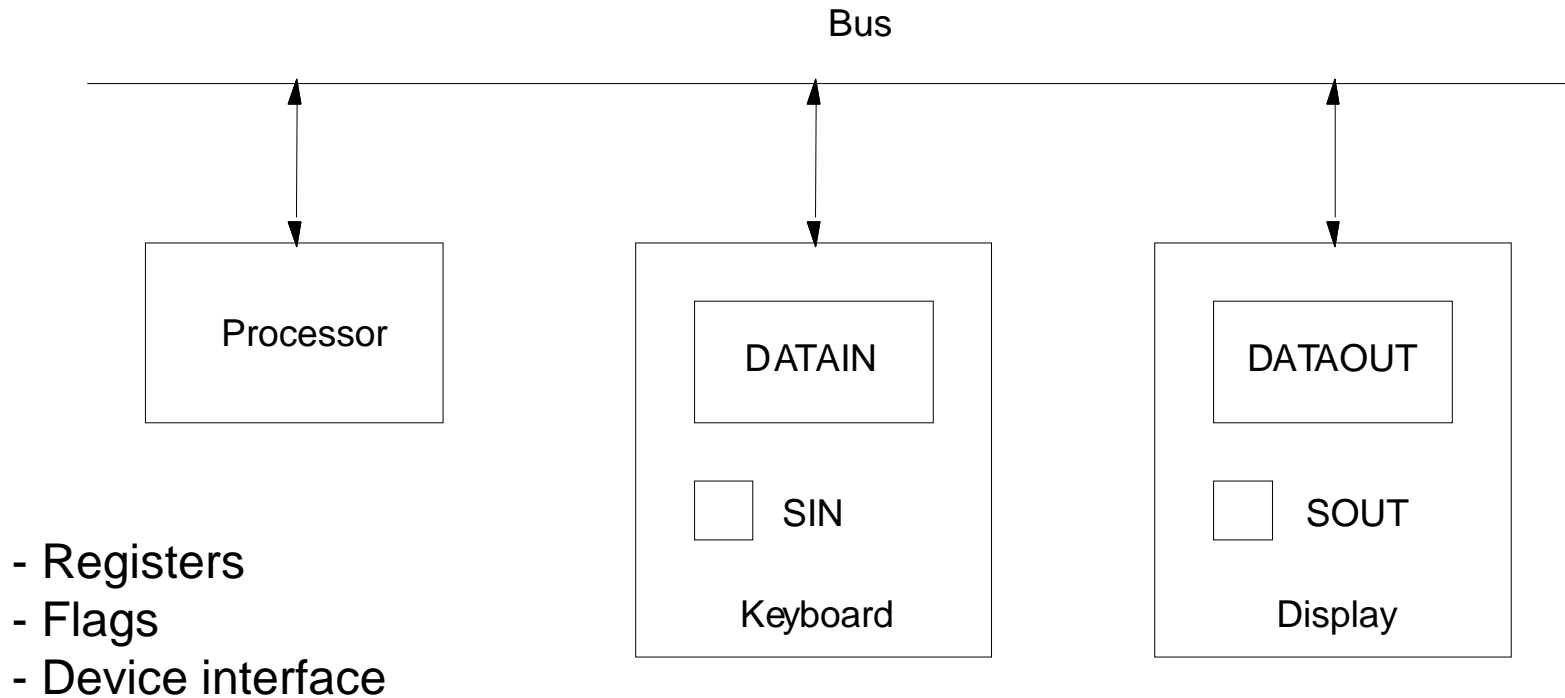
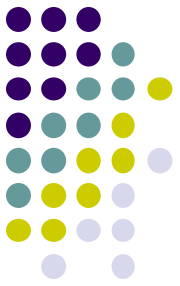
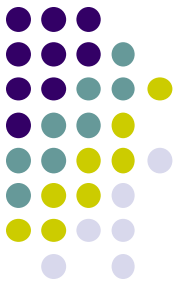


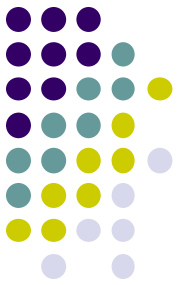
Figure 2.19 Bus connection for processor, keyboard, and display

Program-Controlled I/O Example



- Machine instructions that can check the state of the status flags and transfer data:
 READWAIT Branch to READWAIT if SIN = 0
 Input from DATAIN to R1

 WRITEWAIT Branch to WRITEWAIT if SOUT = 0
 Output from R1 to DATAOUT



Memory-Mapped I/O

- Memory-Mapped I/O – some memory address values are used to refer to peripheral device buffer registers. No special instructions are needed. Also use device status registers.

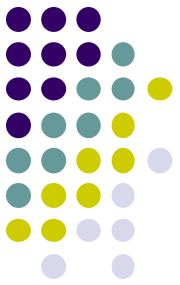
```
READWAIT Testbit #3, INSTATUS  
          Branch=0 READWAIT  
          MoveByte DATAIN, R1
```

```
WRITEWAIT Testbit #3, OUTSTATUS  
          Branch=0 WRITEWAIT  
          MoveByte R1, DATAOUT
```

Program-Controlled I/O Example



- Assumption – the initial state of SIN is 0 and the initial state of SOUT is 1.
- Any drawback of this mechanism in terms of efficiency?
 - Two wait loops → processor execution time is wasted
- Alternate solution?
 - Interrupt



Stacks

Stacks

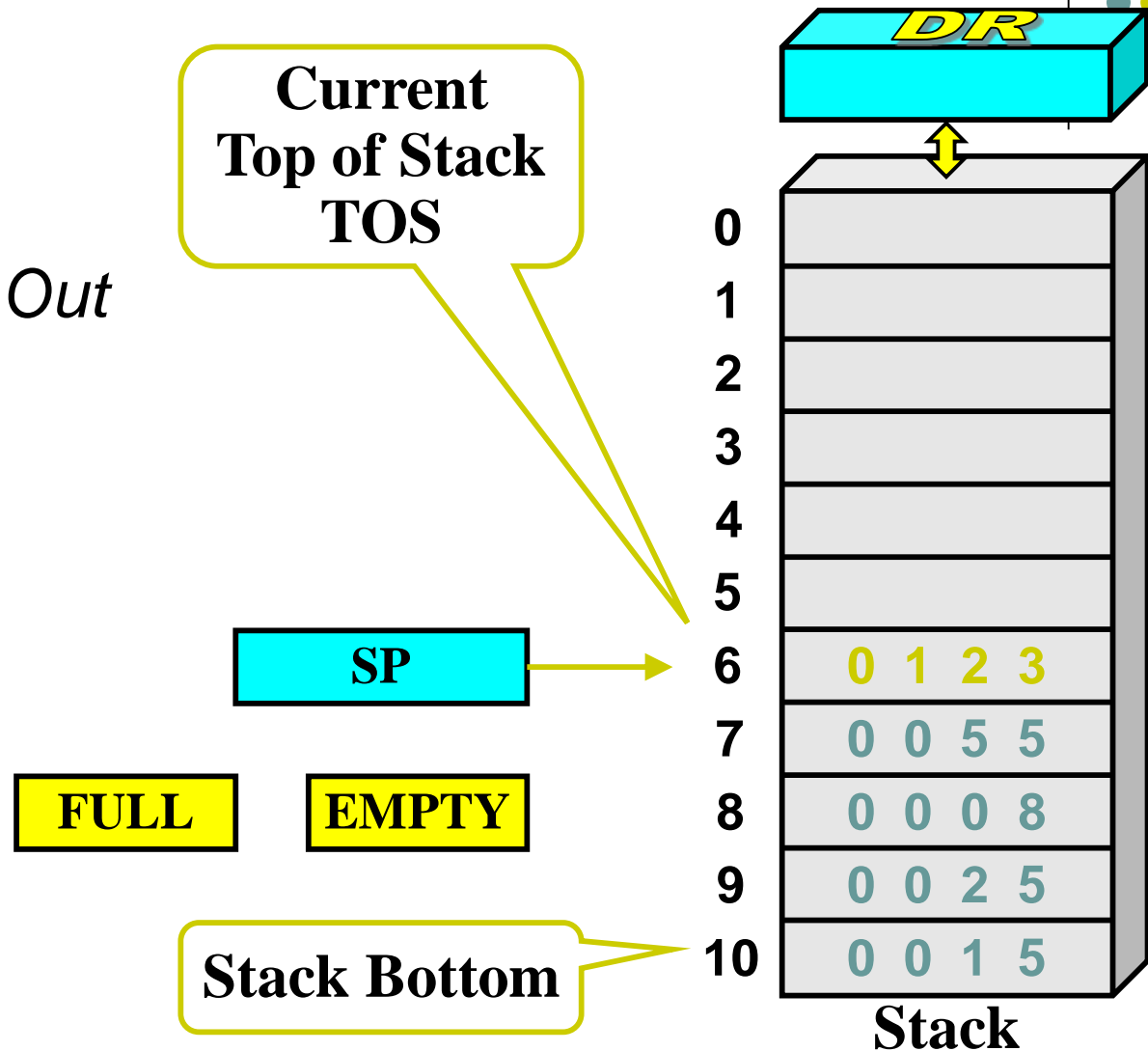


- Data Structure – list of data elements
- Access Restriction - Elements can be added or removed at one end of the list only
- LIFO Stack

Stack Organization

- LIFO

Last In First Out



Stack Organization

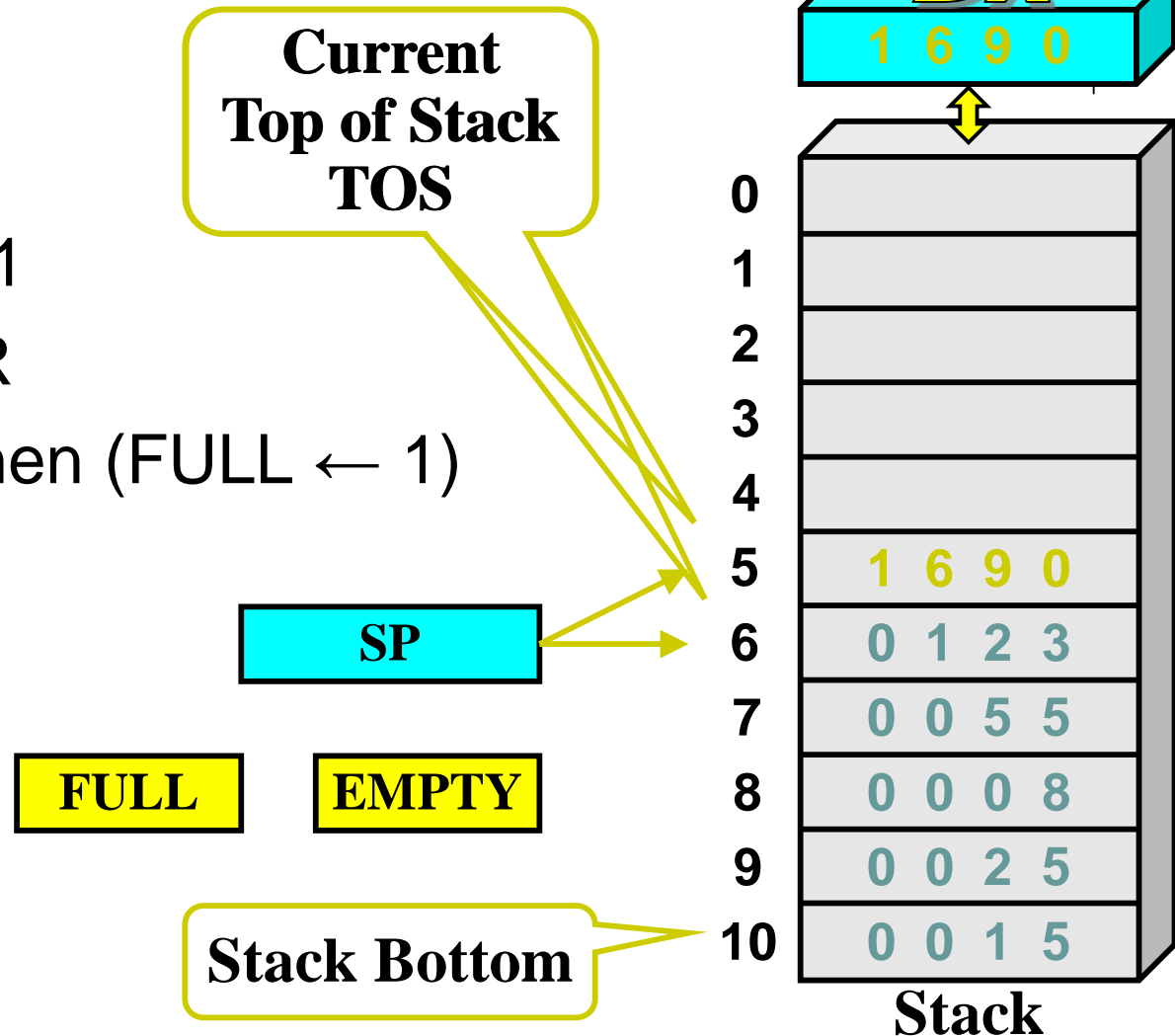
- PUSH

$SP \leftarrow SP - 1$

$M[SP] \leftarrow DR$

If $(SP = 0)$ then $(FULL \leftarrow 1)$

$EMPTY \leftarrow 0$



Stack Organization

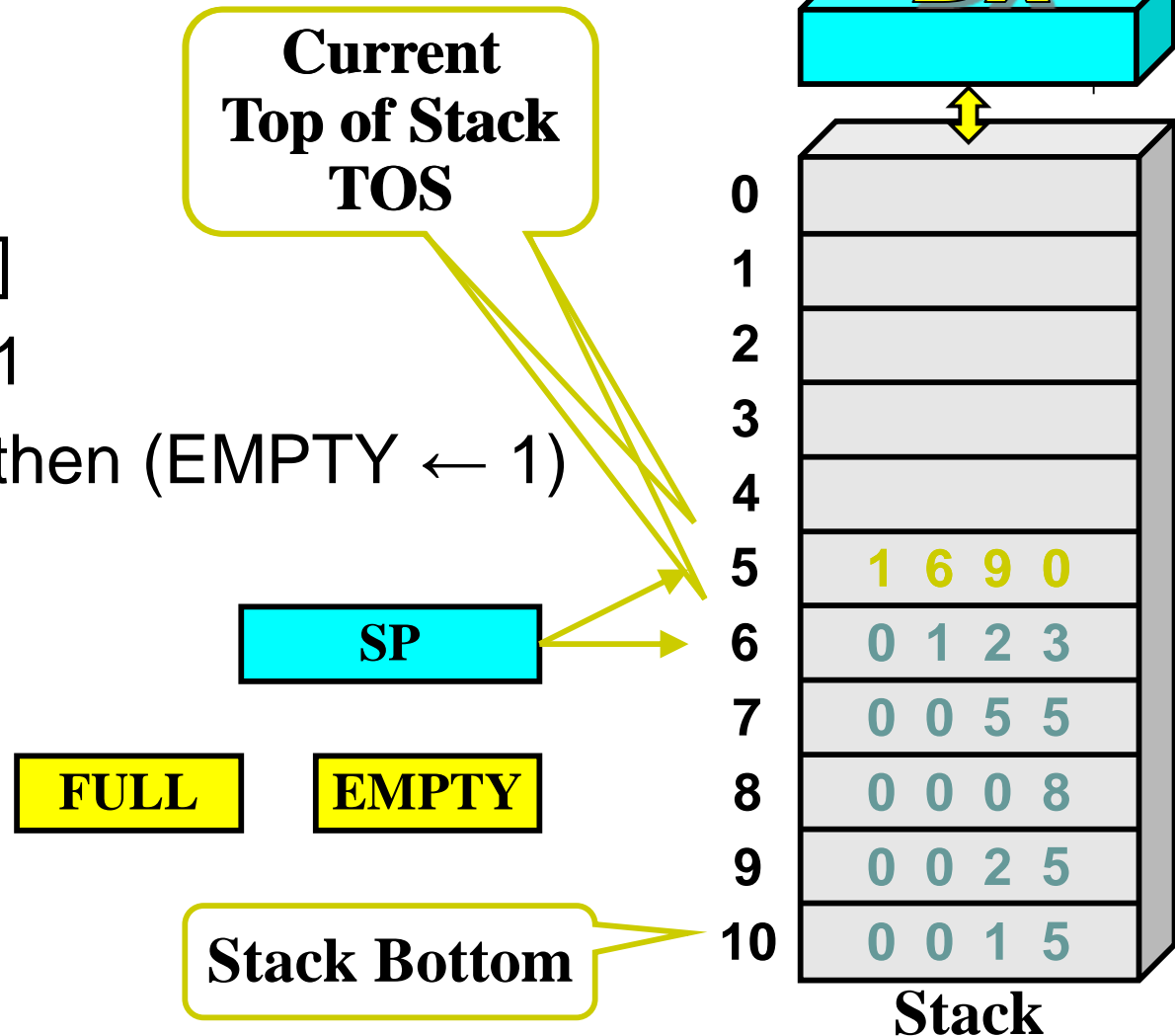
- POP

$DR \leftarrow M[SP]$

$SP \leftarrow SP + 1$

If $(SP = 11)$ then $(EMPTY \leftarrow 1)$

$FULL \leftarrow 0$



Stack Organization

- Memory Stack

- PUSH**

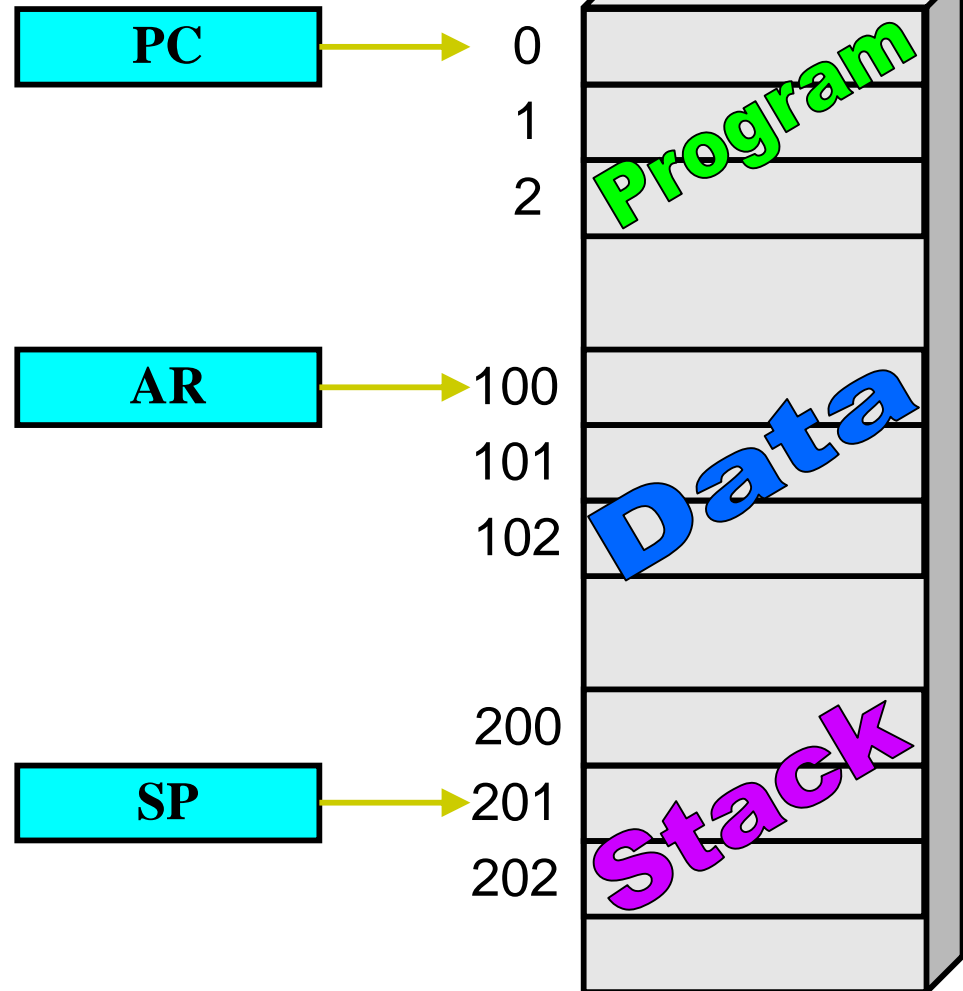
$SP \leftarrow SP - 1$

$M[SP] \leftarrow DR$

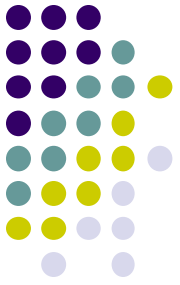
- POP**

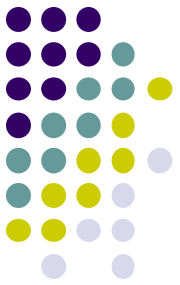
$DR \leftarrow M[SP]$

$SP \leftarrow SP + 1$



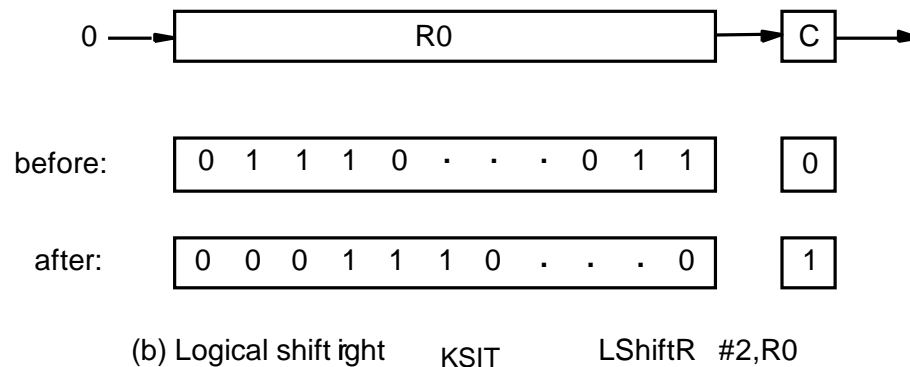
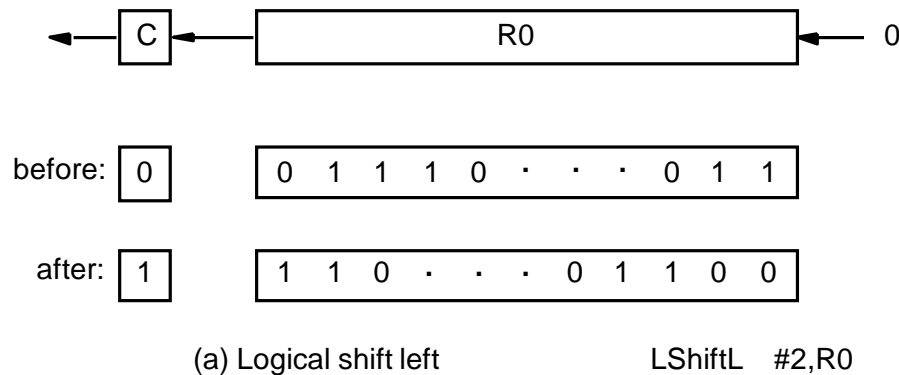
Additional Instructions



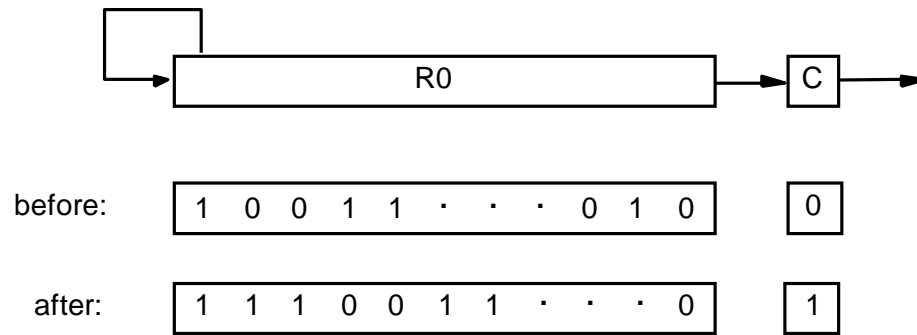
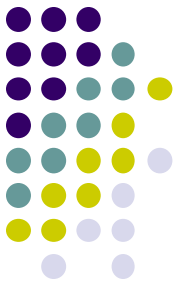


Logical Shifts

- Logical shift – shifting left (LShiftL) and shifting right (LShiftR)



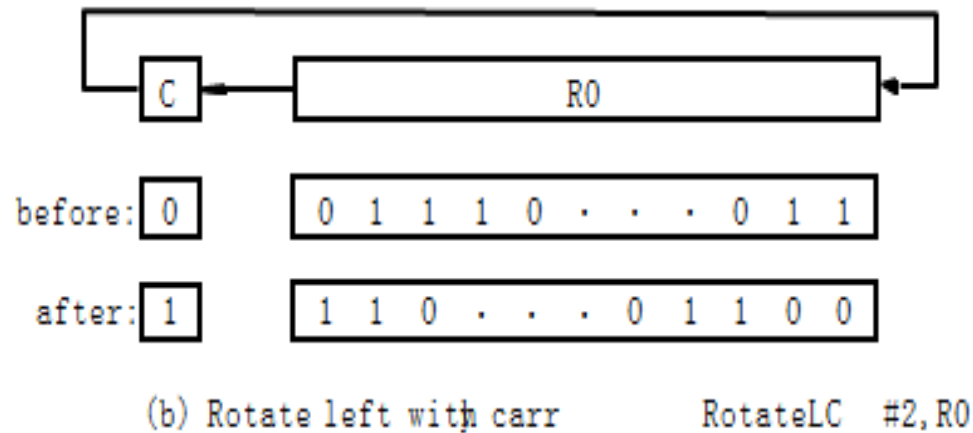
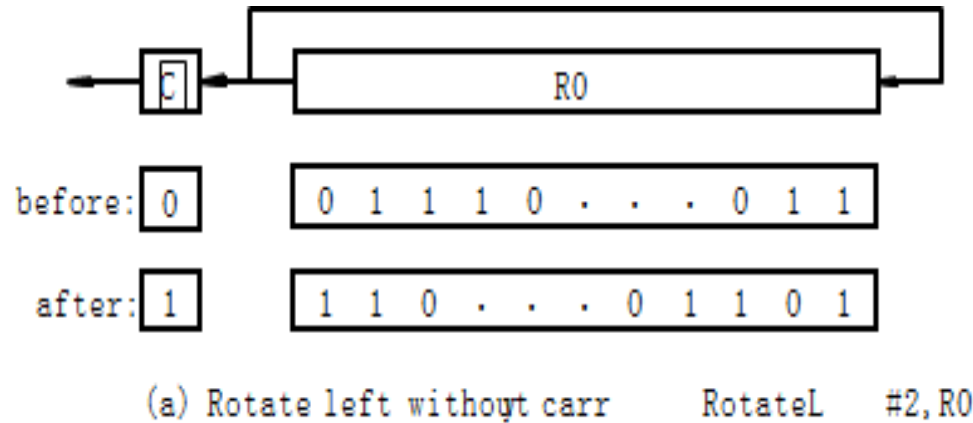
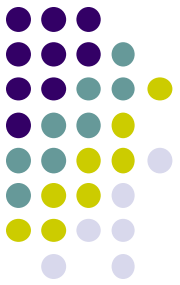
Arithmetic Shifts



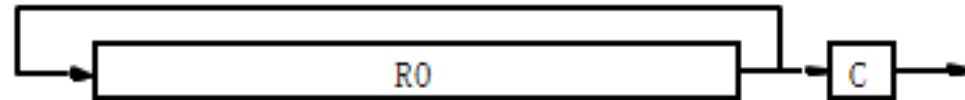
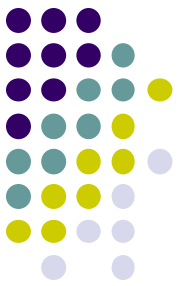
(c) Arithmetic shift right

AShiftr #2,R0

Rotate



Rotate



before:

0	1	1	1	0	.	.	.	0	1	1
---	---	---	---	---	---	---	---	---	---	---

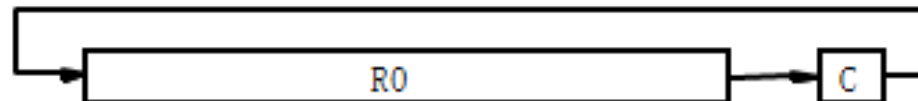
0

after:

1	1	0	1	1	1	0	.	.	.	0
---	---	---	---	---	---	---	---	---	---	---

1

(c) Rotate right without carr RotateR #2,R0



before:

0	1	1	1	0	.	.	.	0	1	1
---	---	---	---	---	---	---	---	---	---	---

0

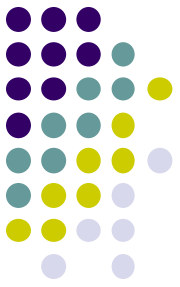
after:

1	0	0	1	1	1	0	.	.	.	0
---	---	---	---	---	---	---	---	---	---	---

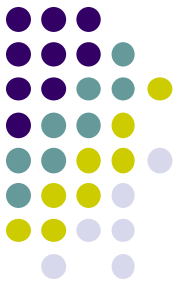
1

(d) Rotate right with carr RotateRC #2,R0

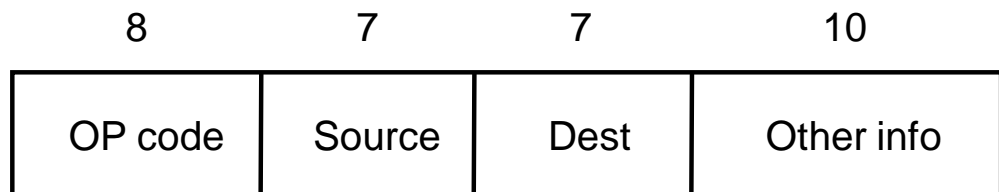
Encoding of Machine Instructions



Encoding of Machine Instructions



- Assembly language program needs to be converted into machine instructions. (ADD = 0100 in ARM instruction set)
- In the previous section, an assumption was made that all instructions are one word in length.
- OP code: the type of operation to be performed and the type of operands used may be specified using an encoded binary pattern
- Suppose 32-bit word length, 8-bit OP code (how many instructions can we have?), 16 registers in total (how many bits?), 3-bit addressing mode indicator.
- Add R1, R2
- Move 24(R0), R5
- LshiftR #2, R0
- Move #\$3A, R1
- Branch>0 LOOP

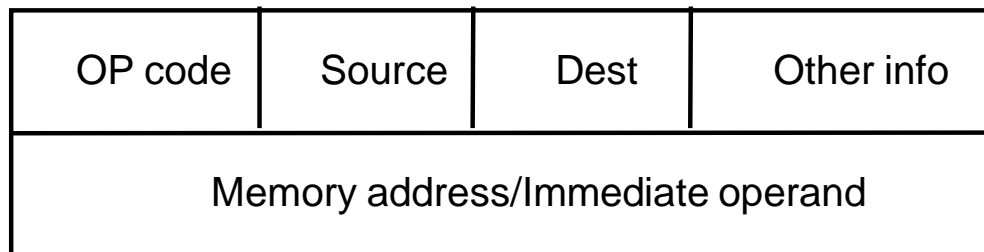


(a) One-word instruction

Encoding of Machine Instructions

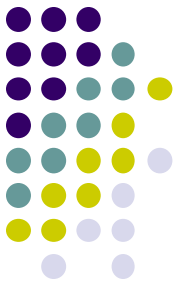


- What happens if we want to specify a memory operand using the Absolute addressing mode?
- Move R2, LOC
- 14-bit for LOC – insufficient
- Solution – use two words



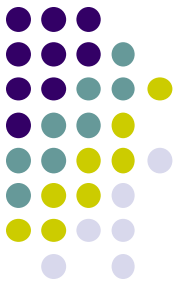
(b) Two-word instruction

Encoding of Machine Instructions

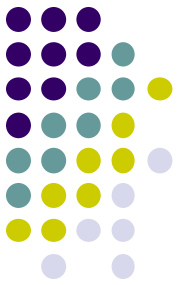


- Then what if an instruction in which two operands can be specified using the Absolute addressing mode?
- Move LOC1, LOC2
- Solution – use two additional words
- This approach results in instructions of variable length. Complex instructions can be implemented, closely resembling operations in high-level programming languages – Complex Instruction Set Computer (CISC)

Encoding of Machine Instructions



- If we insist that all instructions must fit into a single 32-bit word, it is not possible to provide a 32-bit address or a 32-bit immediate operand within the instruction.
- It is still possible to define a highly functional instruction set, which makes extensive use of the processor registers.
- Add R1, R2 ----- yes
- Add LOC, R2 ----- no
- Add (R3), R2 ----- yes




Subroutines

- It is a subtask
- CALL instruction
 - ❑ Store the contents of the PC in the link register
 - ❑ Branch to the target address specified by the instruction
- RETURN instruction
 - ❑ Branch to the address contained in the link register
- The way in which a computer makes it possible to call and return from subroutine – subroutine linkage

Subroutines

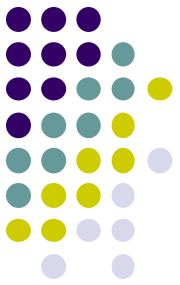


Memory Location	Calling Program	Memory Location	Subroutine
...	...		
200	CALL SUB 	1000	First instruction
204	Next instruction		...
208	...		Return

PC

New PC Value

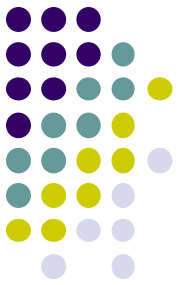
Link



Stack Frame

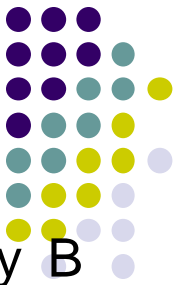
- Location constitute a private work space for the subroutine
- Created at the time the subroutine is entered and freed up when the subroutine returns
- Frame Pointer – to access the local variables of subroutine

Model Questions



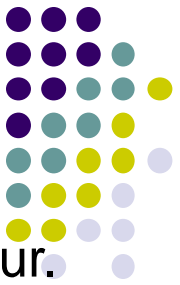
1. What is performance measurement? explain the overall SPEC rating for the computer in a program suite
2. Mention four types of operations to be performed by instructions in a computer. Explain with basic types of instruction formats to carry out $C = [A] + [B]$.
3. Define an addressing mode. Explain the following addressing modes with example: immediate, indirect, index, relative and auto increment
4. What is a stack frame? Explain a commonly used layout for information in a subroutine stack frame
5. Explain shift and rotate operations with example
6. Draw the connection between processor and memory and mention the functions of each component in the connection.
7. Write the difference between RISC and CISC processors.
8. A program contain 1000 instructions. Out of that 25% instructions require 4 clock cycles, 40% instructions require 5 cock cycles and remaining requires 3 clock cycles for execution. Find the total time required to execute the program running in a 1 GHz machine.

Model Questions



9. Explain different rotate instructions.
10. Write ALP program to copy N numbers from array A to array B using indirect addresses.
11. Explain with necessary block diagram the basic functional unit of a computer.
12. Big Endian and little Endian assignments, explain with necessary figure. Represent the number 64243848H in 32 bits big endian and little endian memory.
13. List the name, assembler syntax and addressing functions for the different addressing modes.
14. Draw the arrangement of a single bus structure and brief about memory mapped IO.
15. Explain I) Interrupt enabling, II) Interrupt disabling, III) Edge triggering with respect to interrupts.
16. Explain how to encode the instructions into 32 bit words.

Model Questions



17. With a neat diagram explain the different processor registers.
18. What are the factors that affect the performance? Explain any four.
19. With a neat block diagram, describe the IO operations.
20. Discuss briefly encoding of machine instructions.
21. Derive the basic performance equation. Discuss the measures to improve the performance.
22. What is subroutine linkage? Explain with an example subroutine linkage using linkage register.
23. Registers R1 and R2 of a computer contain the decimal values 1200 and 4600. what is EA of the memory opened in each of the following instructions?
 - I) Load 20(R1),R5 II) Move #3000,R5
 - III) Store R5, 30(R1,R2) IV) Add -(R2) R5 V) Subtract (R1)+ , R5

Module 2.

INPUT/OUTPUT ORGANIZATION

Outline

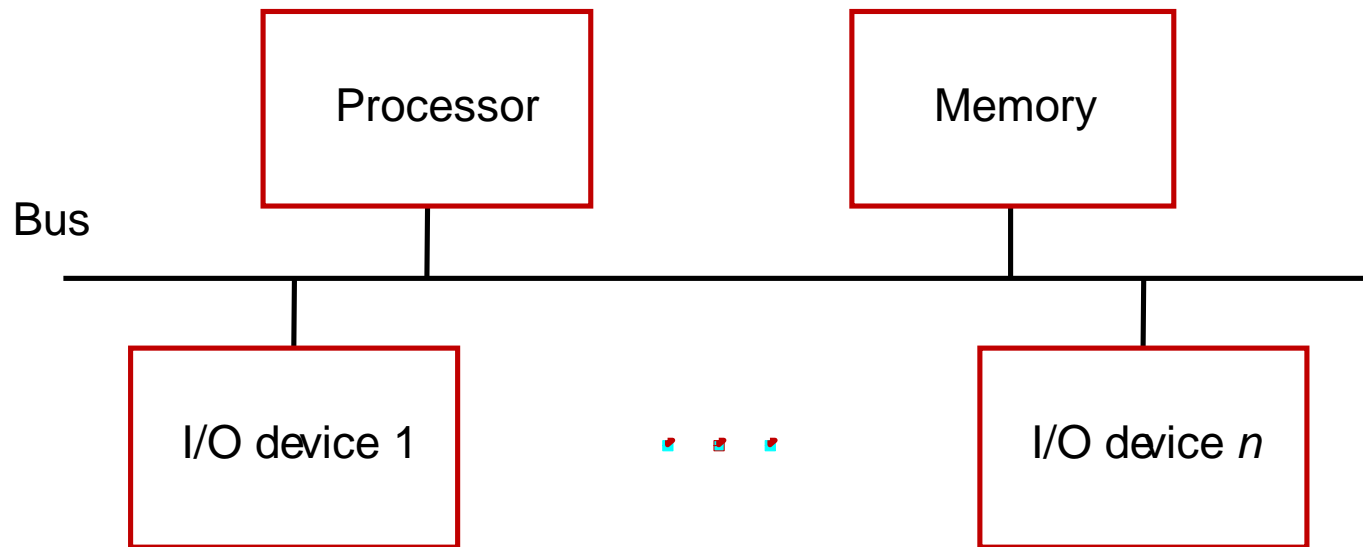
- Accessing I/O Devices
- Interrupts
- Direct Memory Access
- Buses Interface Circuits
- Standard I/O Interfaces

Learning Objectives

1. Expose different ways of communicating with I/O devices and standard I/O interfaces.
2. To analyze interrupts and handling multiple device requests , enabling and disabling interrupts.
3. To analyze the functionality of DMA architecture.
4. To analyze the various connecting buses and interface circuits.

Accessing I/O Devices

Accessing I/O devices

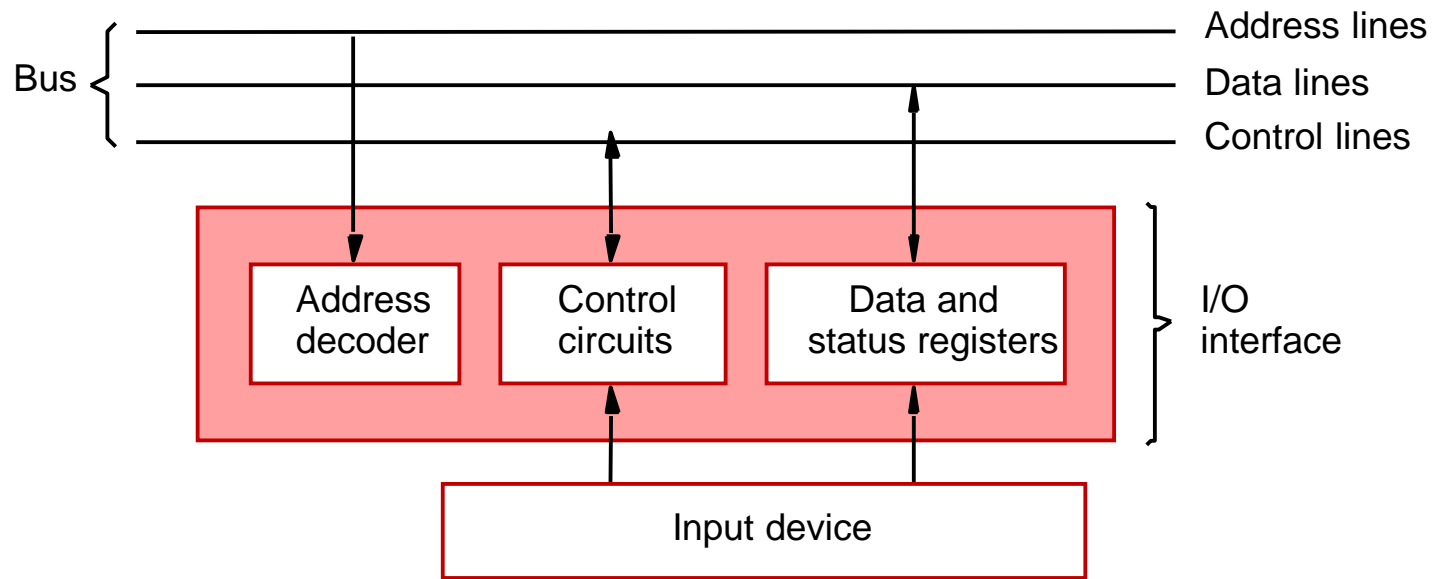


- Multiple I/O devices may be connected to the processor and the memory via a bus.
- Bus consists of three sets of lines to carry address, data and control signals.
- Each I/O device is assigned an unique address.
- To access an I/O device, the processor places the address on the address lines.
- The device recognizes the address, and responds to the control signals.

Accessing I/O devices (contd..)

- I/O devices and the memory may share the same address space:
 - Memory-mapped I/O.
 - Any machine instruction that can access memory can be used to transfer data to or from an I/O device.
 - Simpler software.
- I/O devices and the memory may have different address spaces:
 - Special instructions to transfer data to and from I/O devices.
 - I/O devices may have to deal with fewer address lines.
 - I/O address lines need not be physically separate from memory address lines.
 - In fact, address lines may be shared between I/O devices and memory, with a control signal to indicate whether it is a memory address or an I/O address.

Accessing I/O devices (contd..)



- *I/O device is connected to the bus using an I/O interface circuit which has:*
 - *Address decoder, control circuit, and data and status registers.*
- *Address decoder decodes the address placed on the address lines thus enabling the device to recognize its address.*
- *Data register holds the data being transferred to or from the processor.*
- *Status register holds information necessary for the operation of the I/O device.*
- *Data and status registers are connected to the data lines, and have unique addresses.*
- *I/O interface circuit coordinates I/O transfers.*

Accessing I/O devices (contd..)

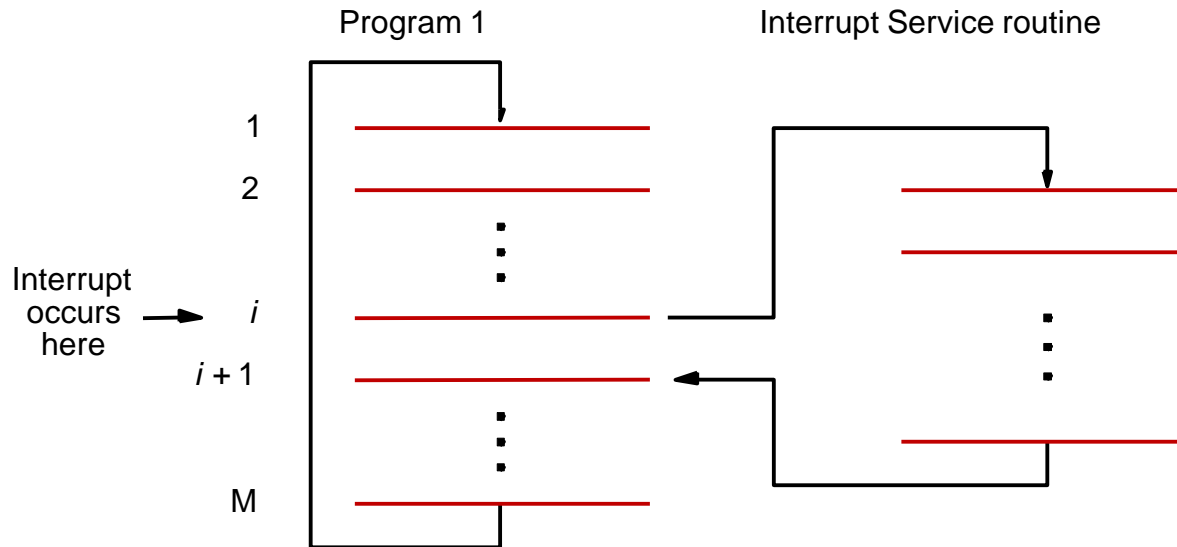
- The rate of transfer to and from I/O devices is slower than the speed of the processor. This creates the need for mechanisms to synchronize data transfers between them.
- **Program-controlled I/O:**
 - Processor repeatedly monitors a status flag to achieve the necessary synchronization.
 - Processor polls the I/O device.
- **Two other mechanisms used for synchronizing data transfers between the processor and memory:**
 - **Interrupts.**
 - **Direct Memory Access.**

Interrupts

Interrupts

- In program-controlled I/O, when the processor continuously monitors the status of the device, it does not perform any useful tasks.
- An alternate approach would be for the I/O device to alert the processor when it becomes ready.
 - Do so by sending a hardware signal called an interrupt to the processor.
 - At least one of the bus control lines, called an interrupt-request line is dedicated for this purpose.
- Processor can perform other useful tasks while it is waiting for the device to be ready.
- The routine executed in response to an interrupt request is called Interrupt Service Routine

Interrupts (contd..)



- Processor is executing the instruction located at address i when an interrupt occurs.
- Routine executed in response to an interrupt request is called the interrupt-service routine.
- When an interrupt occurs, control must be transferred to the interrupt service routine.
- But before transferring control, the current contents of the PC ($i+1$), must be saved in a known location.
- This will enable the return-from-interrupt instruction to resume execution at $i+1$.
- Return address, or the contents of the PC are usually stored on the processor stack.

Interrupts (contd..)

- Treatment of an interrupt-service routine is very similar to that of a subroutine.
- However there are significant differences:
 - A subroutine performs a task that is required by the calling program.
 - Interrupt-service routine may not have anything in common with the program it interrupts.
 - Interrupt-service routine and the program that it interrupts may belong to different users.
 - As a result, before branching to the interrupt-service routine, not only the PC, but other information such as condition code flags, and processor registers used by both the interrupted program and the interrupt service routine must be stored.
 - This will enable the interrupted program to resume execution upon return from interrupt service routine.

Interrupts (contd..)

- Saving and restoring information can be done automatically by the processor or explicitly by program instructions.
- Saving and restoring registers involves memory transfers:
 - Increases the total execution time.
 - Increases the delay between the time an interrupt request is received, and the start of execution of the interrupt-service routine. This delay is called interrupt latency.
- In order to reduce the interrupt latency, most processors save only the minimal amount of information:
 - This minimal amount of information includes Program Counter and processor status registers.
- Any additional information that must be saved, must be saved explicitly by the program instructions at the beginning of the interrupt service routine.

Interrupts (contd..)

- When a processor receives an interrupt-request, it must branch to the interrupt service routine.
- It must also inform the device that it has recognized the interrupt request.
- This can be accomplished in two ways:
 - Some processors have an explicit interrupt-acknowledge control signal for this purpose.
 - In other cases, the data transfer that takes place between the device and the processor can be used to inform the device.

Enabling & Disabling Interrupts

- Interrupt-requests interrupt the execution of a program, and may alter the intended sequence of events:
 - Sometimes such alterations may be undesirable, and must not be allowed.
 - For example, the processor may not want to be interrupted by the same device while executing its interrupt-service routine.
- Processors generally provide the ability to enable and disable such interruptions as desired.
- One simple way is to provide machine instructions such as *Interrupt-enable* and *Interrupt-disable* for this purpose.
- To avoid interruption by the same device during the execution of an interrupt service routine:
 - First instruction of an interrupt service routine can be Interrupt-disable.
 - Last instruction of an interrupt service routine can be Interrupt-enable.

Edge Triggered Interrupts

- Interrupt handling circuit responds only to the leading edge of the signal
- Processor receives only one request regardless of how long the line is activated
- No need to explicitly disable interrupt requests

Interrupts Handling

1. The device raises an interrupt request
2. The processor interrupts the program currently being executed
3. Interrupts are disabled
4. The device is informed that its request has been recognized
5. The action requested by the interrupt is performed by the ISR
6. Interrupts are enabled and execution of the interrupted program is resumed

Handling Multiple Devices

- Multiple I/O devices may be connected to the processor and the memory via a bus. Some or all of these devices may be capable of generating interrupt requests.
 - Each device operates independently, and hence no definite order can be imposed on how the devices generate interrupt requests?
- How does the processor know which device has generated an interrupt? (Status register)
- How does the processor know which interrupt service routine needs to be executed? (Polling)
- When the processor is executing an interrupt service routine for one device, can other device interrupt the processor?
- If two interrupt-requests are received simultaneously, then how to break the tie?

Handling Multiple Devices

- Consider a simple arrangement where all devices send their interrupt-requests over a single control line in the bus.
- When the processor receives an interrupt request over this control line, how does it know which device is requesting an interrupt?
- This information is available in the status register of the device requesting an interrupt:
 - The status register of each device has an *IRQ* bit which it sets to 1 when it requests an interrupt.
- Interrupt service routine can poll the I/O devices connected to the bus. The first device with *IRQ* equal to 1 is the one that is serviced.
- Polling mechanism is easy, but time consuming to query the status bits of all the I/O devices connected to the bus.

Vectored Interrupts

- The device requesting an interrupt may identify itself directly to the processor.
 - Device can do so by sending a special code (4 to 8 bits) the processor over the bus.
 - Code supplied by the device may represent a part of the starting address of the interrupt-service routine.
 - The remainder of the starting address is obtained by the processor based on other information such as the range of memory addresses where interrupt service routines are located.
- Usually the location pointed to by the interrupting device is used to store the starting address of the interrupt-service routine.
 - Processor reads this address (interrupt vector) and loads it into the PC

Interrupt Nesting

- Previously, before the processor started executing the interrupt service routine for a device, it disabled the interrupts from the device.
- In general, same arrangement is used when multiple devices can send interrupt requests to the processor.
 - During the execution of an interrupt service routine of device, the processor does not accept interrupt requests from any other device.
 - Since the interrupt service routines are usually short, the delay that this causes is generally acceptable.
- However, for certain devices this delay may not be acceptable.
 - Which devices can be allowed to interrupt a processor when it is executing an interrupt service routine of another device?

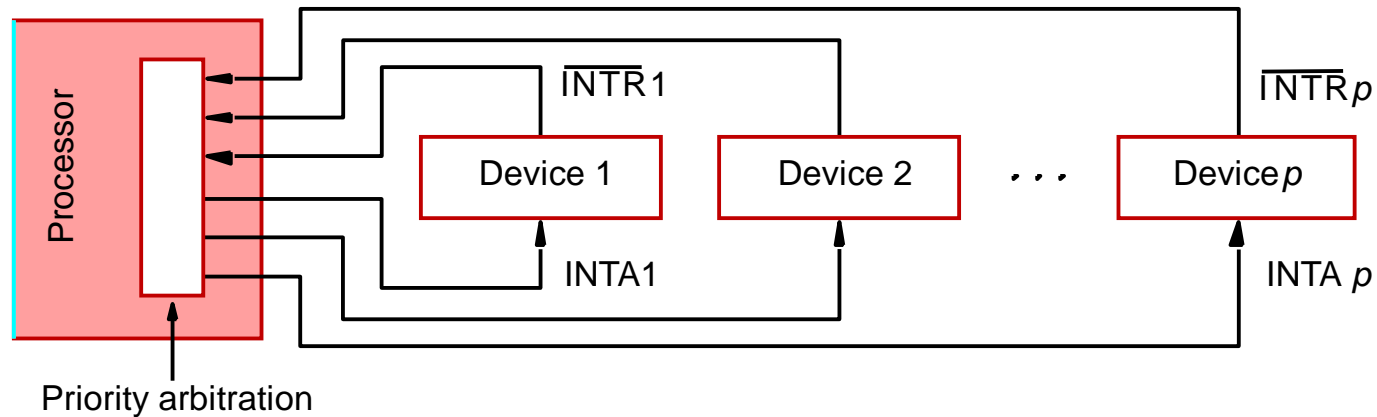
Interrupt Nesting

- I/O devices are organized in a priority structure:
 - An interrupt request from a high-priority device is accepted while the processor is executing the interrupt service routine of a low priority device.
- A priority level is assigned to a processor that can be changed under program control.
 - Priority level of a processor is the priority of the program that is currently being executed.
 - When the processor starts executing the interrupt service routine of a device, its priority is raised to that of the device.
 - If the device sending an interrupt request has a higher priority than the processor, the processor accepts the interrupt request.

Interrupt Nesting

- Processor's priority is encoded in a few bits of the processor status register.
 - Priority can be changed by instructions that write into the processor status register.
 - Usually, these are privileged instructions, or instructions that can be executed only in the supervisor mode.
 - Privileged instructions cannot be executed in the user mode.
 - Prevents a user program from accidentally or intentionally changing the priority of the processor.
- If there is an attempt to execute a privileged instruction in the user mode, it causes a special type of interrupt called as privilege exception.

Interrupt Nesting



- *Each device has a separate interrupt-request and interrupt-acknowledge line.*
- *Each interrupt-request line is assigned a different priority level.*
- *Interrupt requests received over these lines are sent to a priority arbitration circuit in the processor.*
- *If the interrupt request has a higher priority level than the priority of the processor, then the request is accepted.*

Interrupts – Simultaneous Requests

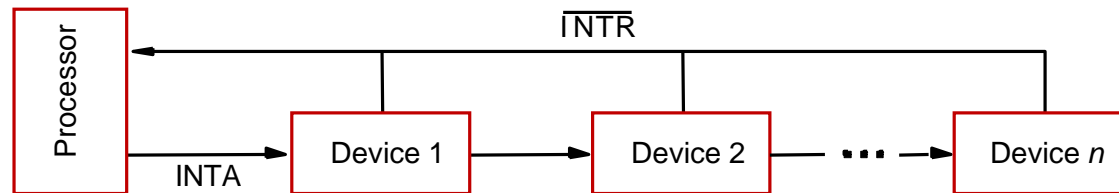
- Which interrupt request does the processor accept if it receives interrupt requests from two or more devices simultaneously?.
- If the I/O devices are organized in a priority structure, the processor accepts the interrupt request from a device with higher priority.
 - Each device has its own interrupt request and interrupt acknowledge line.
 - A different priority level is assigned to the interrupt request line of each device.
- However, if the devices share an interrupt request line, then how does the processor decide which interrupt request to accept?

Interrupts – Simultaneous Requests

Polling scheme:

- If the processor uses a polling mechanism to poll the status registers of I/O devices to determine which device is requesting an interrupt.
- In this case the priority is determined by the order in which the devices are polled.
- The first device with status bit set to 1 is the device whose interrupt request is accepted.

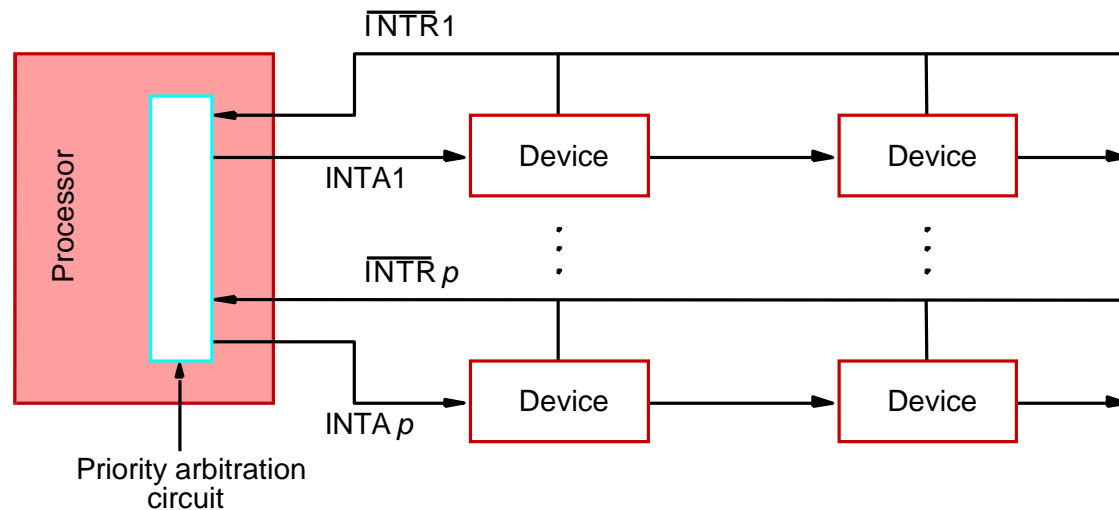
Daisy chain scheme:



- Devices are connected to form a daisy chain.
- Devices share the interrupt-request line, and interrupt-acknowledge line is connected to form a daisy chain.
- When devices raise an interrupt request, the interrupt-request line is activated.
- The processor in response activates interrupt-acknowledge.
- Received by device 1, if device 1 does not need service, it passes the signal to device 2.
- Device that is electrically closest to the processor has the highest priority.

Interrupts – Simultaneous Requests

- When I/O devices were organized into a priority structure, each device had its own interrupt-request and interrupt-acknowledge line.
- When I/O devices were organized in a daisy chain fashion, the devices shared an interrupt-request line, and the interrupt-acknowledge propagated through the devices.
- A combination of priority structure and daisy chain scheme can also be used.



- Devices are organized into groups.
- Each group is assigned a different priority level.
- All the devices within a single group share an interrupt-request line, and are connected to form a daisy chain.

Controlling Device Requests

- Only those devices that are being used in a program should be allowed to generate interrupt requests.
- To control which devices are allowed to generate interrupt requests, the interface circuit of each I/O device has an interrupt-enable bit.
 - If the interrupt-enable bit in the device interface is set to 1, then the device is allowed to generate an interrupt-request.
- Interrupt-enable bit in the device's interface circuit determines whether the device is allowed to generate an interrupt request.
- Interrupt-enable bit in the processor status register or the priority structure of the interrupts determines whether a given interrupt will be accepted.

Exceptions

- Interrupts caused by interrupt-requests sent by I/O devices.
- Interrupts could be used in many other situations where the execution of one program needs to be suspended and execution of another program needs to be started.
- In general, the term exception is used to refer to any event that causes an interruption.
 - Interrupt-requests from I/O devices is one type of an exception.
- Other types of exceptions are:
 - Recovery from errors
 - Debugging
 - Privilege Exception

Exceptions - Recovery from errors

- Many sources of errors in a processor. For example:
 - Error in the data stored.
 - Error during the execution of an instruction.
- When such errors are detected, exception processing is initiated.
 - Processor takes the same steps as in the case of I/O interrupt-request.
 - It suspends the execution of the current program, and starts executing an exception-service routine.
- Difference between handling I/O interrupt-request and handling exceptions due to errors:
 - In case of I/O interrupt-request, the processor usually completes the execution of an instruction in progress before branching to the interrupt-service routine.
 - In case of exception processing however, the execution of an instruction in progress usually cannot be completed.

Exceptions - Debugging

- Debugger uses exceptions to provide important features:
 - Trace,
 - Breakpoints.
- Trace mode:
 - Exception occurs after the execution of every instruction.
 - Debugging program is used as the exception-service routine.
- Breakpoints:
 - Exception occurs only at specific points selected by the user.
 - Debugging program is used as the exception-service routine.

Exceptions - Privilege Exception

- Certain instructions can be executed only when the processor is in the supervisor mode. These are called privileged instructions.
- If an attempt is made to execute a privileged instruction in the user mode, a privilege exception occurs.
- Privilege exception causes:
 - Processor to switch to the supervisor mode,
 - Execution of an appropriate exception-servicing routine.

Direct Memory Access

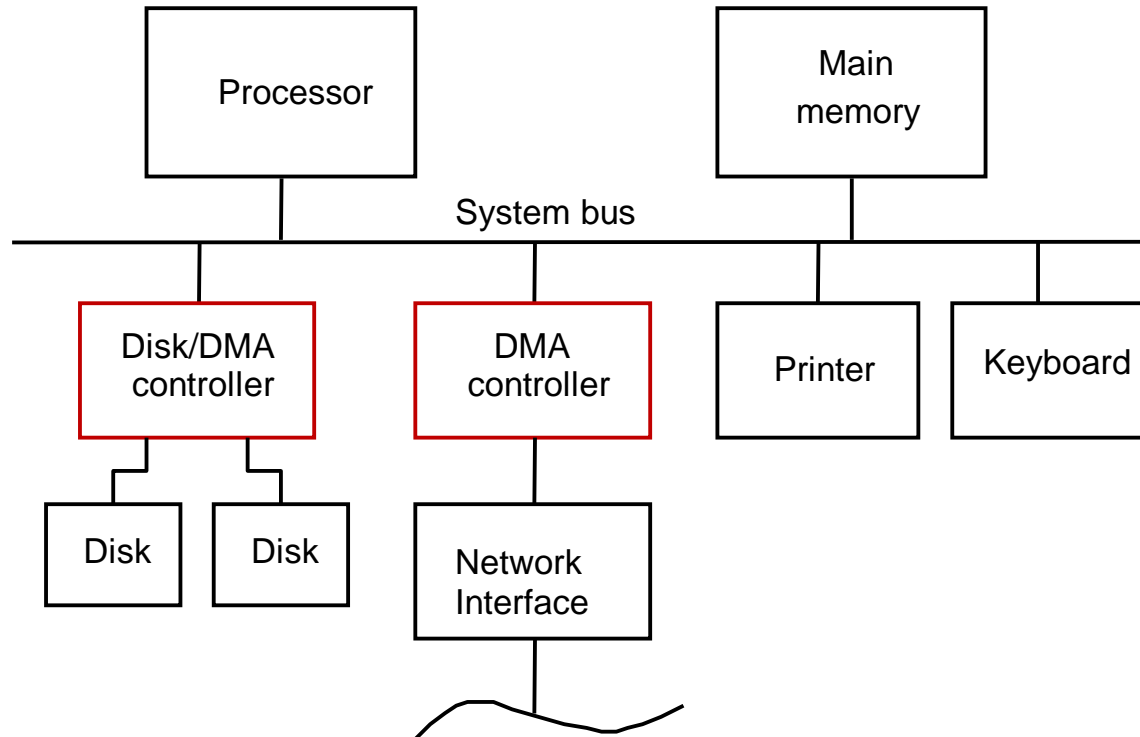
Direct Memory Access (contd..)

- **Direct Memory Access (DMA):**
 - A special control unit may be provided to transfer a block of data directly between an I/O device and the main memory, without continuous intervention by the processor.
- Control unit which performs these transfers is a part of the I/O device's interface circuit. This control unit is called as a DMA controller.
- DMA controller performs functions that would be normally carried out by the processor:
 - For each word, it provides the memory address and all the control signals.
 - To transfer a block of data, it increments the memory addresses and keeps track of the number of transfers.

Direct Memory Access (contd..)

- DMA controller can transfer a block of data from an external device to the processor, without any intervention from the processor.
 - However, the operation of the DMA controller must be under the control of a program executed by the processor. That is, the processor must initiate the DMA transfer.
- To initiate the DMA transfer, the processor informs the DMA controller of:
 - Starting address,
 - Number of words in the block.
 - Direction of transfer (I/O device to the memory, or memory to the I/O device).
- Once the DMA controller completes the DMA transfer, it informs the processor by raising an interrupt signal.

Direct Memory Access



- *DMA controller connects a high-speed network to the computer bus.*
- *Disk controller, which controls two disks also has DMA capability. It provides two DMA channels.*
- *It can perform two independent DMA operations, as if each disk has its own DMA controller. The registers to store the memory address, word count and status and control information are duplicated.*

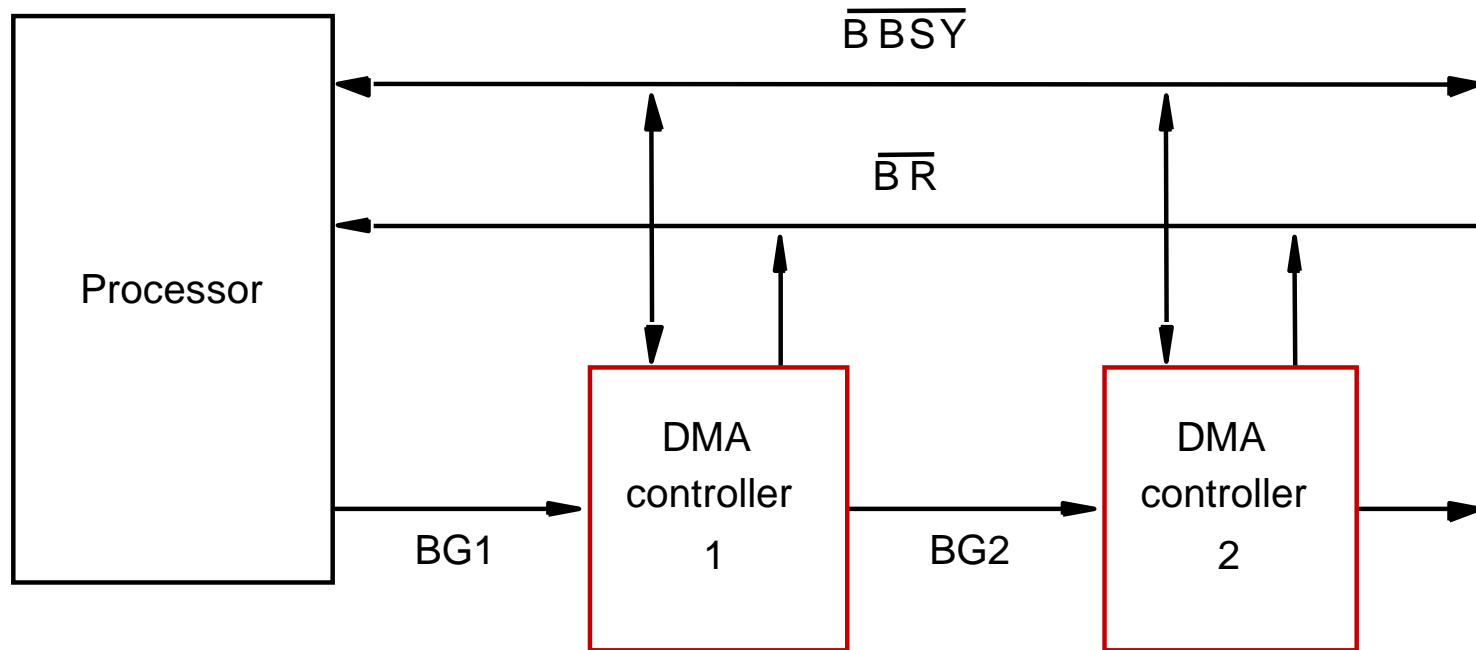
Direct Memory Access (contd..)

- Processor and DMA controllers have to use the bus in an interwoven fashion to access the memory.
 - DMA devices are given higher priority than the processor to access the bus.
 - Among different DMA devices, high priority is given to high-speed peripherals such as a disk or a graphics display device.
- Processor originates most memory access cycles on the bus.
 - DMA controller can be said to “steal” memory access cycles from the bus. This interweaving technique is called as “cycle stealing”.
- An alternate approach is to provide a DMA controller an exclusive capability to initiate transfers on the bus, and hence exclusive access to the main memory. This is known as the block or burst mode.

Bus Arbitration

- Processor and DMA controllers both need to initiate data transfers on the bus and access main memory.
- The device that is allowed to initiate transfers on the bus at any given time is called the bus master.
- When the current bus master relinquishes its status as the bus master, another device can acquire this status.
 - The process by which the next device to become the bus master is selected and bus mastership is transferred to it is called bus arbitration.
- Centralized arbitration:
 - A single bus arbiter performs the arbitration.
- Distributed arbitration:
 - All devices participate in the selection of the next bus master.

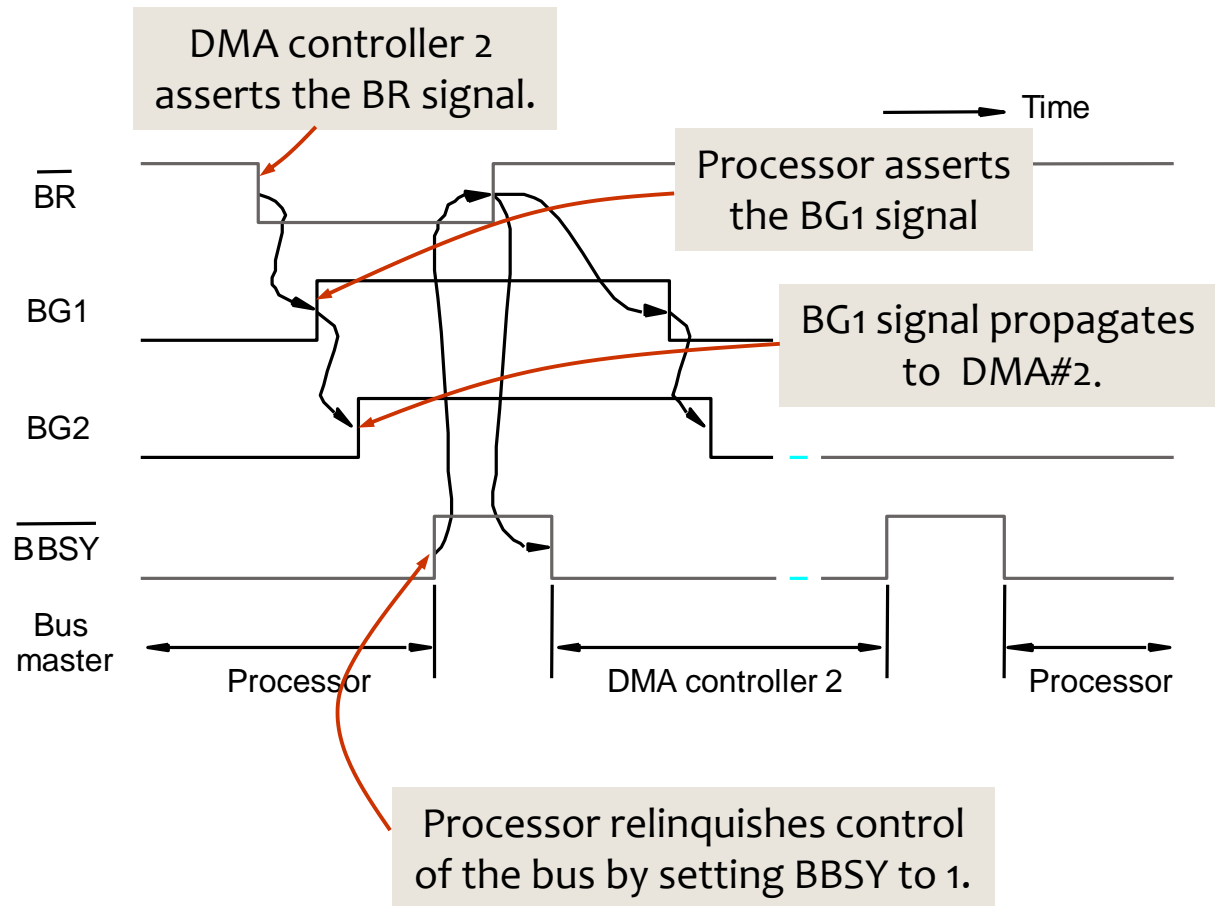
Centralized Bus Arbitration



Centralized Bus Arbitration(cont.,)

- *Bus arbiter may be the processor or a separate unit connected to the bus.*
- *Normally, the processor is the bus master, unless it grants bus membership to one of the DMA controllers.*
- *DMA controller requests the control of the bus by asserting the Bus Request (BR) line.*
- *In response, the processor activates the Bus-Grant1 (BG1) line, indicating that the controller may use the bus when it is free.*
- *BG1 signal is connected to all DMA controllers in a daisy chain fashion.*
- *BBSY signal is 0, it indicates that the bus is busy. When BBSY becomes 1, the DMA controller which asserted BR can acquire control of the bus.*

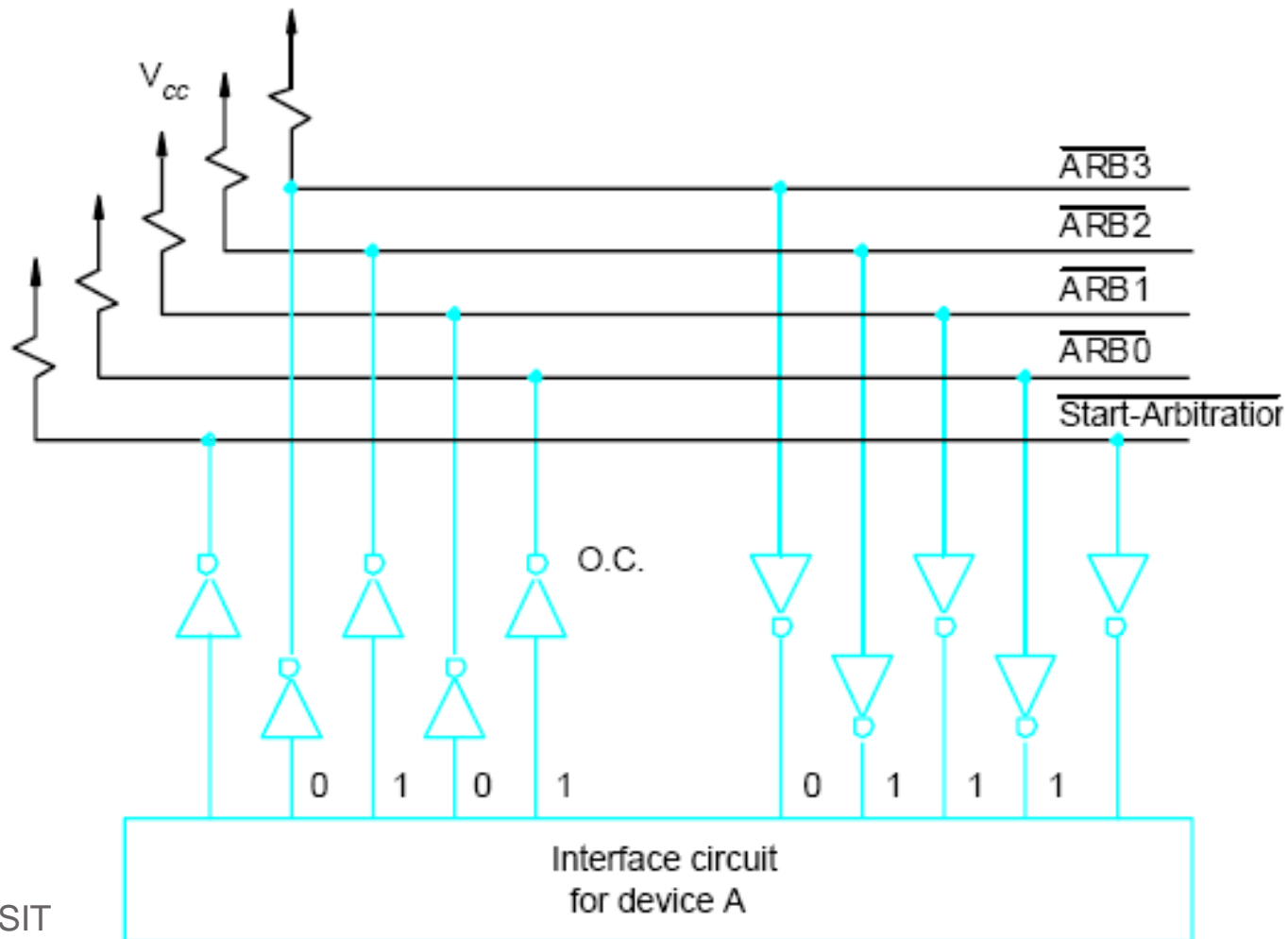
Centralized arbitration (contd..)



Distributed arbitration

- All devices waiting to use the bus share the responsibility of carrying out the arbitration process.
 - Arbitration process does not depend on a central arbiter and hence distributed arbitration has higher reliability.
- Each device is assigned a 4-bit ID number.
- All the devices are connected using 5 lines, 4 arbitration lines to transmit the ID, and one line for the Start-Arbitration signal.
- To request the bus a device:
 - Asserts the Start-Arbitration signal.
 - Places its 4-bit ID number on the arbitration lines.
- The pattern that appears on the arbitration lines is the logical-OR of all the 4-bit device IDs placed on the arbitration lines.

Distributed arbitration



Distributed arbitration(Contd.,)

- Arbitration process:
 - *Each device compares the pattern that appears on the arbitration lines to its own ID, starting with MSB.*
 - *If it detects a difference, it transmits 0s on the arbitration lines for that and all lower bit positions.*
 - *The pattern that appears on the arbitration lines is the logical-OR of all the 4-bit device IDs placed on the arbitration lines.*

Distributed arbitration (contd..)

- *Device A has the ID 5 and wants to request the bus:*
 - *Transmits the pattern 0101 on the arbitration lines.*
- *Device B has the ID 6 and wants to request the bus:*
 - *Transmits the pattern 0110 on the arbitration lines.*
- *Pattern that appears on the arbitration lines is the logical OR of the patterns:*
 - *Pattern 0111 appears on the arbitration lines.*

Arbitration process:

- *Each device compares the pattern that appears on the arbitration lines to its own ID, starting with MSB.*
- *If it detects a difference, it transmits 0s on the arbitration lines for that and all lower bit positions.*
- *Device A compares its ID 5 with a pattern 0101 to pattern 0111.*
- *It detects a difference at bit position 0, as a result, it transmits a pattern 0100 on the arbitration lines.*
- *The pattern that appears on the arbitration lines is the logical-OR of 0100 and 0110, which is 0110.*
- *This pattern is the same as the device ID of B, and hence B has won the arbitration.*

Buses

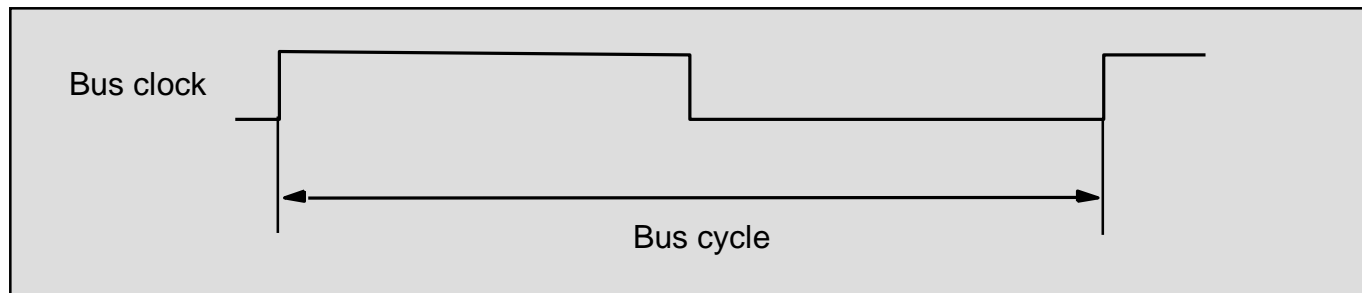
Buses

- Processor, main memory, and I/O devices are interconnected by means of a bus.
- Bus provides a communication path for the transfer of data.
 - Bus also includes lines to support interrupts and arbitration.
- A bus protocol is the set of rules that govern the behavior of various devices connected to the bus, as to when to place information on the bus, when to assert control signals, etc.

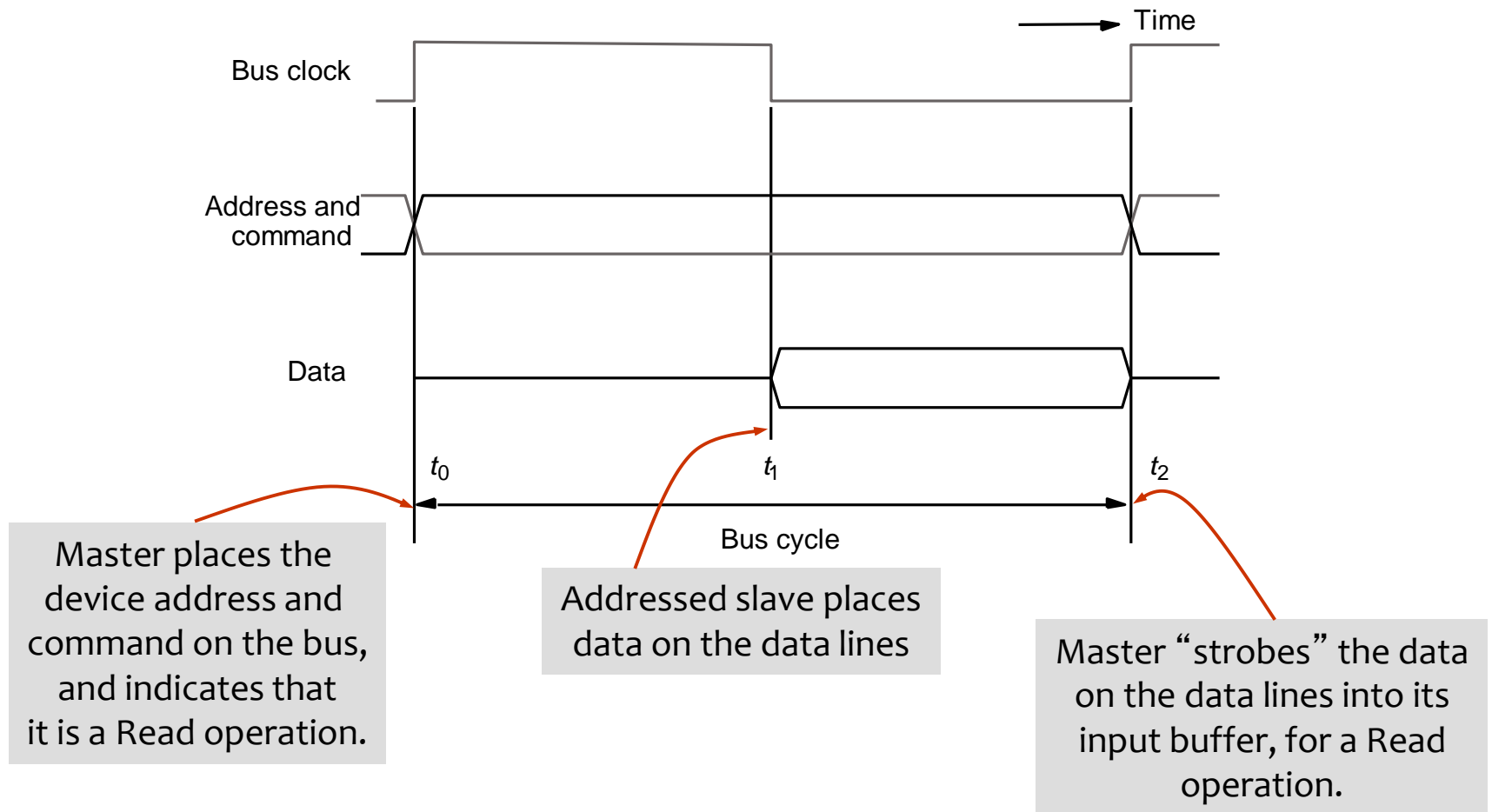
Buses (contd..)

- Bus lines may be grouped into three types:
 - Data
 - Address
 - Control
- Control signals specify:
 - Whether it is a read or a write operation.
 - Required size of the data, when several operand sizes (byte, word, long word) are possible.
 - Timing information to indicate when the processor and I/O devices may place data or receive data from the bus.
- Schemes for timing of data transfers over a bus can be classified into:
 - Synchronous,
 - Asynchronous.

Synchronous bus



Synchronous bus (contd..)



- *In case of a Write operation, the master places the data on the bus along with the address and commands at time t_0 .*
- *The slave strobes the data into its input buffer at time t_2 .*

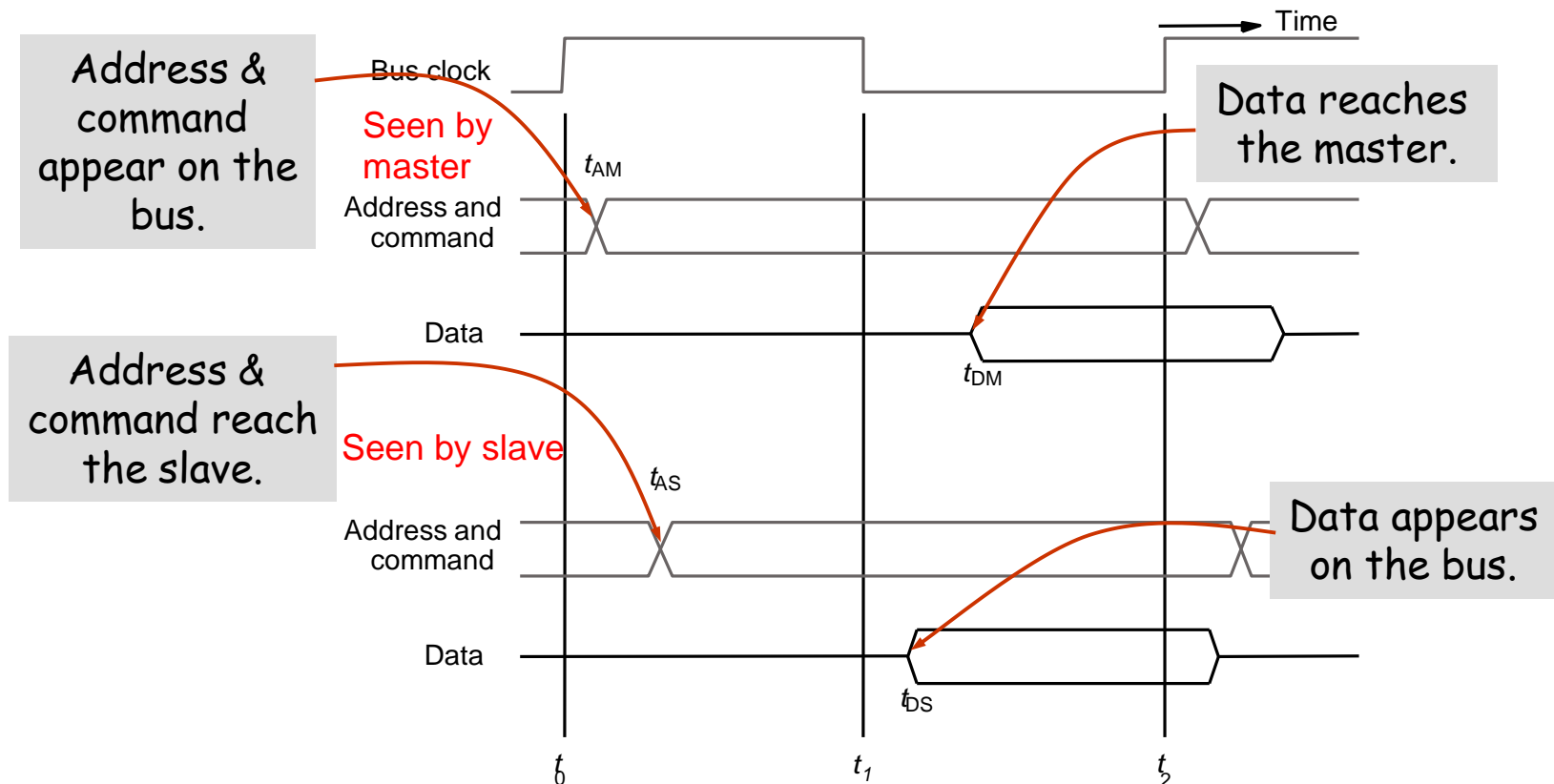
Synchronous bus (contd..)

- Once the master places the device address and command on the bus, it takes time for this information to propagate to the devices:
 - This time depends on the physical and electrical characteristics of the bus.
- Also, all the devices have to be given enough time to decode the address and control signals, so that the addressed slave can place data on the bus.
- Width of the pulse $t_1 - t_0$ depends on:
 - Maximum propagation delay between two devices connected to the bus.
 - Time taken by all the devices to decode the address and control signals, so that the addressed slave can respond at time t_1 .

Synchronous bus (contd..)

- At the end of the clock cycle, at time t_2 , the master strobesc the data on the data lines into its input buffer if it's a Read operation.
 - “Strobe” means to capture the values of the data and store them into a buffer.
- When data are to be loaded into a storage buffer register, the data should be available for a period longer than the setup time of the device.
- Width of the pulse $t_2 - t_1$ should be longer than:
 - Maximum propagation time of the bus plus
 - Set up time of the input buffer register of the master.

Synchronous bus (contd..)



- Signals do not appear on the bus as soon as they are placed on the bus, due to the propagation delay in the interface circuits.
- Signals reach the devices after a propagation delay which depends on the characteristics of the bus.
- Data must remain on the bus for some time after t_2 equal to the hold time of the buffer.

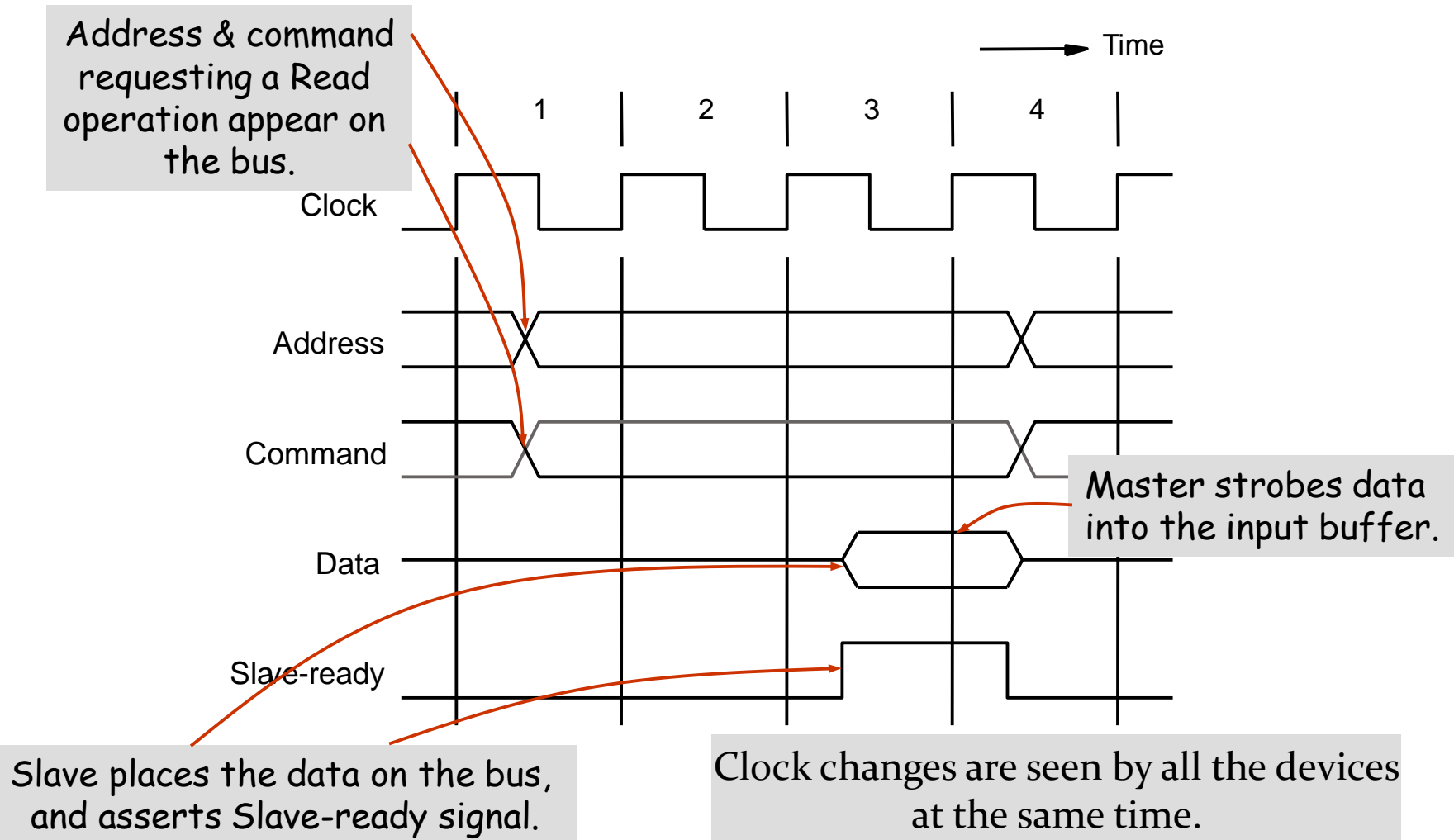
Synchronous bus (contd..)

- Data transfer has to be completed within one clock cycle.
 - Clock period $t_2 - t_0$ must be such that the longest propagation delay on the bus and the slowest device interface must be accommodated.
 - Forces all the devices to operate at the speed of the slowest device.
- Processor just assumes that the data are available at t_2 in case of a Read operation, or are read by the device in case of a Write operation.
 - What if the device is actually failed, and never really responded?

Synchronous bus (contd..)

- Most buses have control signals to represent a response from the slave.
- Control signals serve two purposes:
 - Inform the master that the slave has recognized the address, and is ready to participate in a data transfer operation.
 - Enable to adjust the duration of the data transfer operation based on the speed of the participating slaves.
- High-frequency bus clock is used:
 - Data transfer spans several clock cycles instead of just one clock cycle as in the earlier case.

Synchronous bus (contd..)



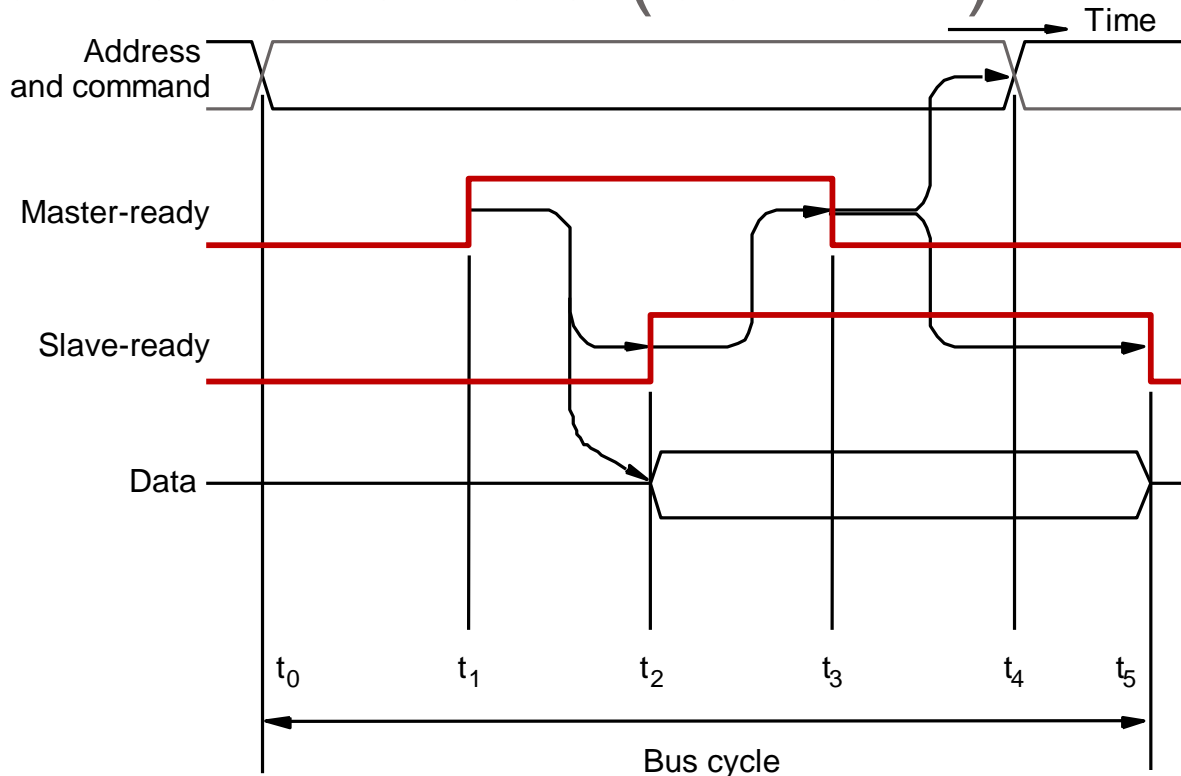
Asynchronous bus

- Data transfers on the bus is controlled by a handshake between the master and the slave.
- Common clock in the synchronous bus case is replaced by two timing control lines:
 - Master-ready,
 - Slave-ready.
- Master-ready signal is asserted by the master to indicate to the slave that it is ready to participate in a data transfer.
- Slave-ready signal is asserted by the slave in response to the master-ready from the master, and it indicates to the master that the slave is ready to participate in a data transfer.

Asynchronous bus (contd..)

- Data transfer using the handshake protocol:
 - Master places the address and command information on the bus.
 - Asserts the Master-ready signal to indicate to the slaves that the address and command information has been placed on the bus.
 - All devices on the bus decode the address.
 - Address slave performs the required operation, and informs the processor it has done so by asserting the Slave-ready signal.
 - Master removes all the signals from the bus, once Slave-ready is asserted.
 - If the operation is a Read operation, Master also strobes the data into its input buffer.

Asynchronous bus (contd..)



t_0 - Master places the address and command information on the bus.

t_1 - Master asserts the Master-ready signal. Master-ready signal is asserted at t_1 instead of t_0 .

t_2 - Addressed slave places the data on the bus and asserts the Slave-ready signal.

t_3 - Slave-ready signal arrives at the master.

t_4 - Master removes the address and command information.

t_5 - Slave receives the transition of the Master-ready signal from 1 to 0. It removes the data from the bus and the Slave-ready signal from the bus.

Asynchronous vs. Synchronous bus

- **Advantages of asynchronous bus:**
 - Eliminates the need for synchronization between the sender and the receiver.
 - Can accommodate varying delays automatically, using the Slave-ready signal.
- **Disadvantages of asynchronous bus:**
 - Data transfer rate with full handshake is limited by two-round trip delays.
 - Data transfers using a synchronous bus involves only one round trip delay, and hence a synchronous bus can achieve faster rates.

Interface Circuits

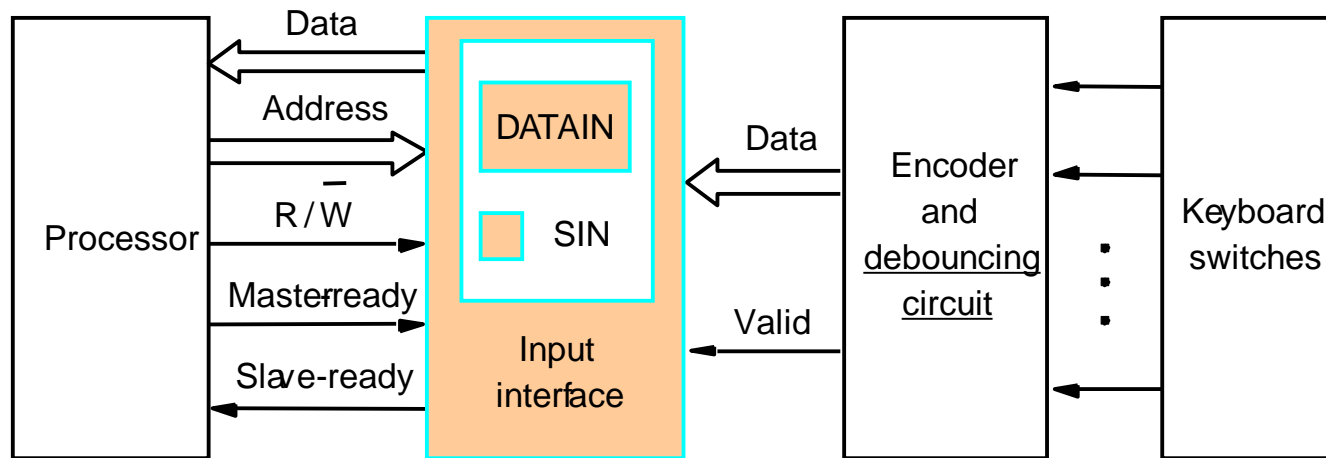
Interface circuits

- I/O interface consists of the circuitry required to connect an I/O device to a computer bus.
- Side of the interface which connects to the computer has bus signals for:
 - Address,
 - Data
 - Control
- Side of the interface which connects to the I/O device has:
 - Datapath and associated controls to transfer data between the interface and the I/O device.
 - This side is called as a “port”.
- Ports can be classified into two:
 - Parallel port,
 - Serial port.

Interface circuits (contd..)

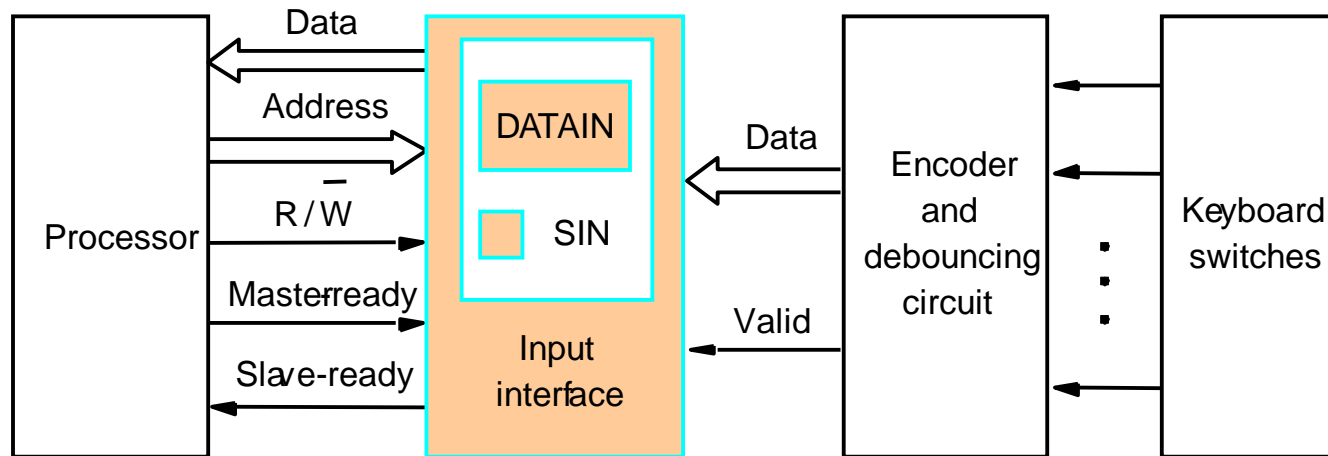
- Parallel port transfers data in the form of a number of bits, normally 8 or 16, to or from the device.
- Serial port transfers and receives data one bit at a time.
- Processor communicates with the bus in the same way, whether it is a parallel port or a serial port.
 - Conversion from the parallel to serial and vice versa takes place inside the interface circuit.

Parallel port



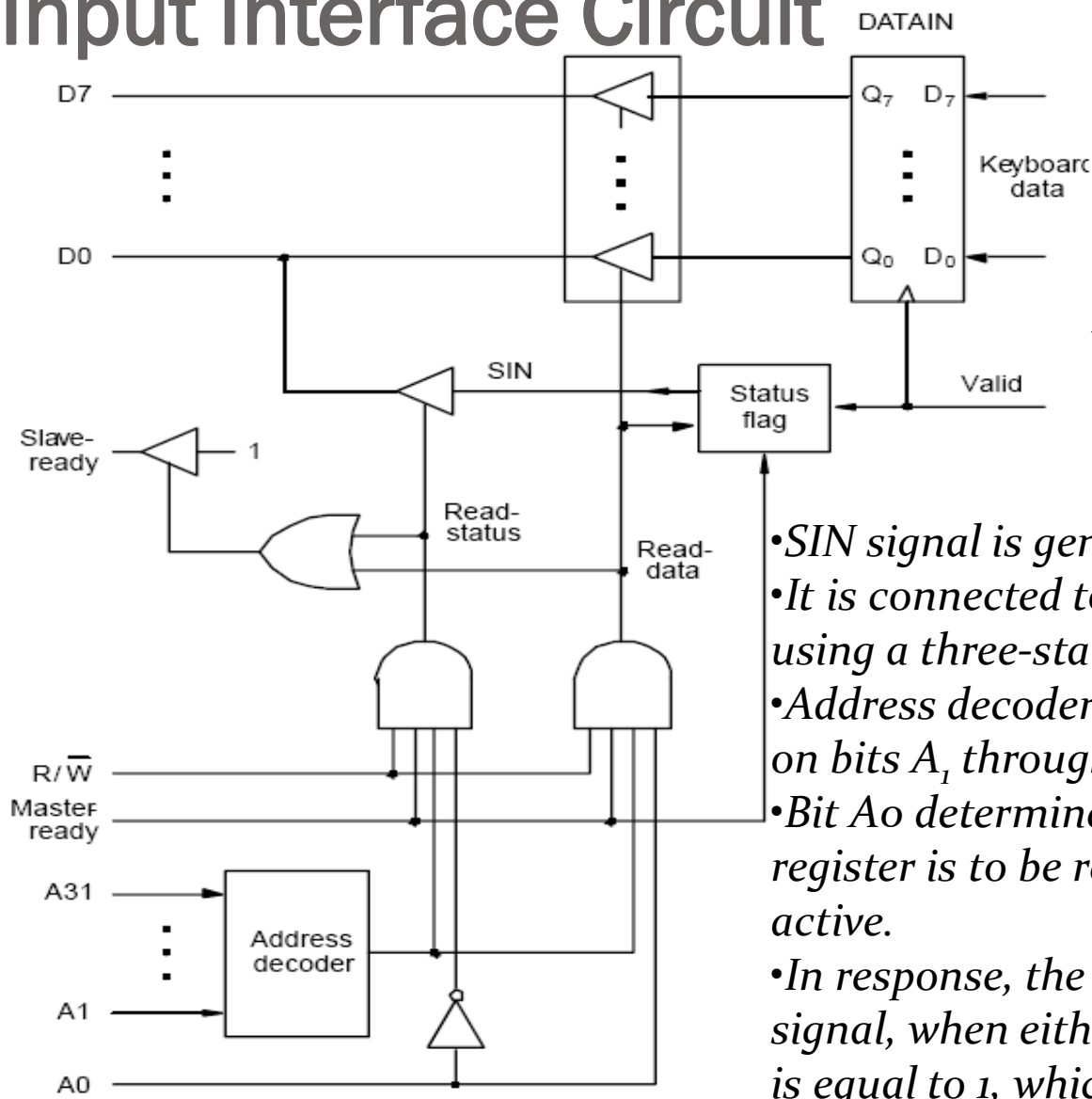
- *Keyboard is connected to a processor using a parallel port.*
- *Processor is 32-bits and uses memory-mapped I/O and the asynchronous bus protocol.*
- *On the processor side of the interface we have:*
 - *Data lines.*
 - *Address lines*
 - *Control or R/W line.*
 - *Master-ready signal and*
 - *Slave-ready signal.*

Parallel port (contd..)



- On the keyboard side of the interface:
 - Encoder circuit which generates a code for the key pressed.
 - Debouncing circuit which eliminates the effect of a key bounce (a single key stroke may appear as multiple events to a processor).
 - Data lines contain the code for the key.
 - Valid line changes from 0 to 1 when the key is pressed. This causes the code to be loaded into **DATAIN** and **SIN** to be set to 1.

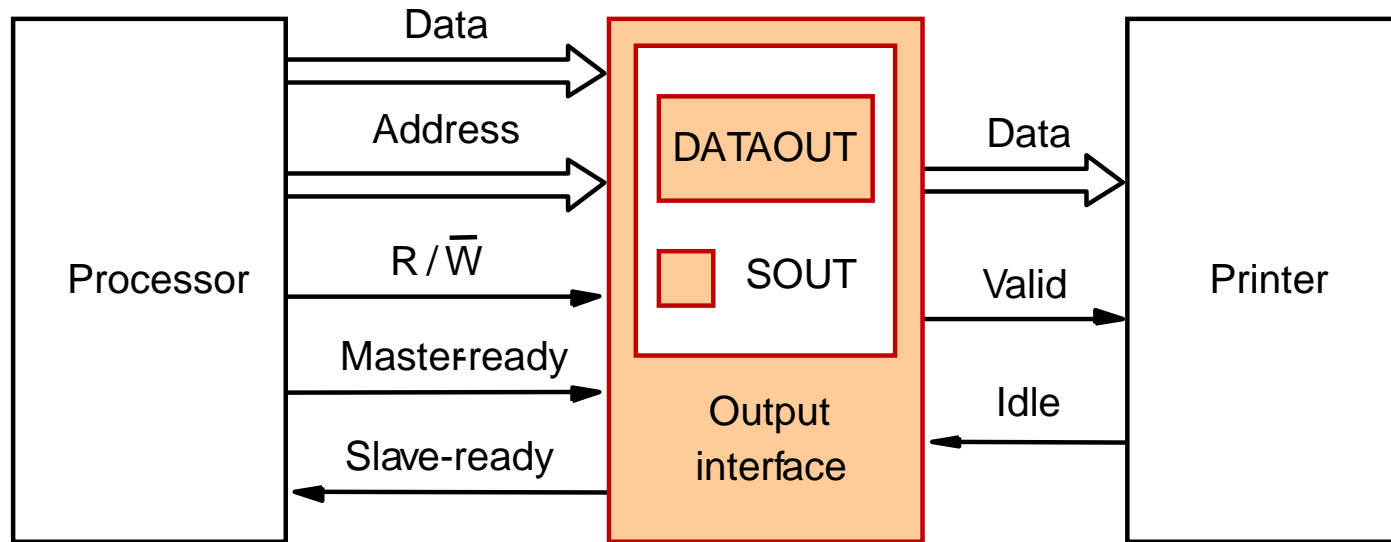
Input Interface Circuit



- Output lines of *DATAIN* are connected to the data lines of the bus by means of 3 state drivers
- Drivers are turned on when the processor issues a read signal and the address selects this register.

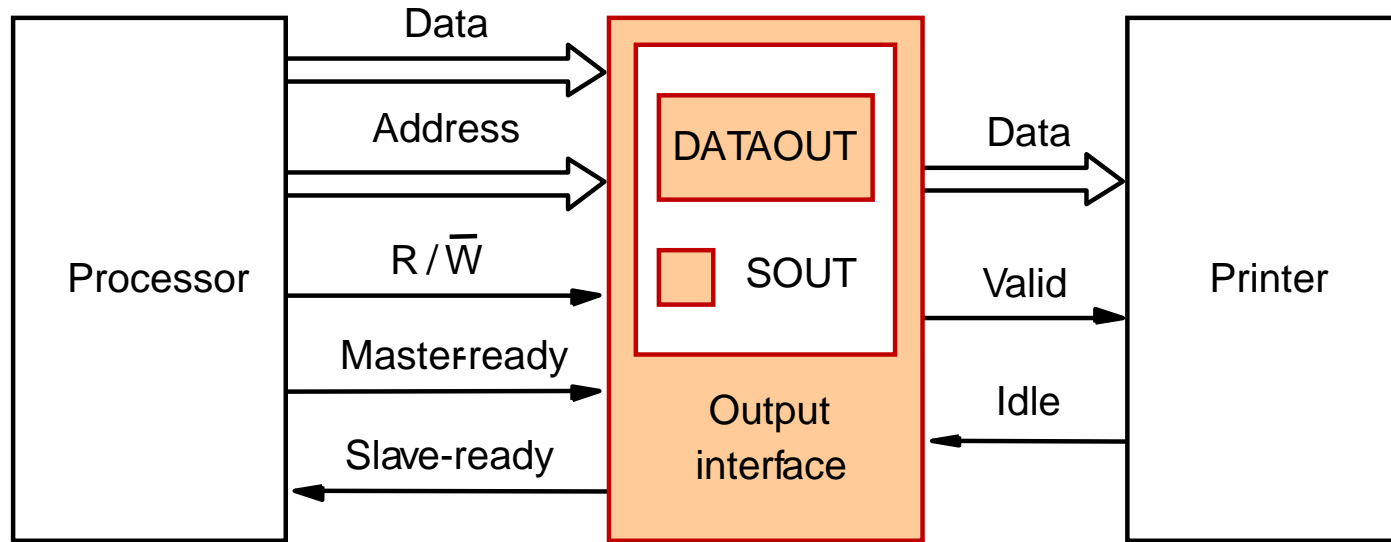
- *SIN signal is generated using a status flag circuit.*
- *It is connected to line D_o of the processor bus using a three-state driver.*
- *Address decoder selects the input interface based on bits A_1 through A_{3r} .*
- *Bit A_o determines whether the status or data register is to be read, when Master-ready is active.*
- *In response, the processor activates the Slave-ready signal, when either the Read-status or Read-data is equal to 1, which depends on line A_o .*

Parallel port (contd..)



- *Printer is connected to a processor using a parallel port.*
- *Processor is 32 bits, uses memory-mapped I/O and asynchronous bus protocol.*
- *On the processor side:*
 - *Data lines.*
 - *Address lines*
 - *Control or R/W line.*
 - *Master-ready signal and*
 - *Slave-ready signal.*

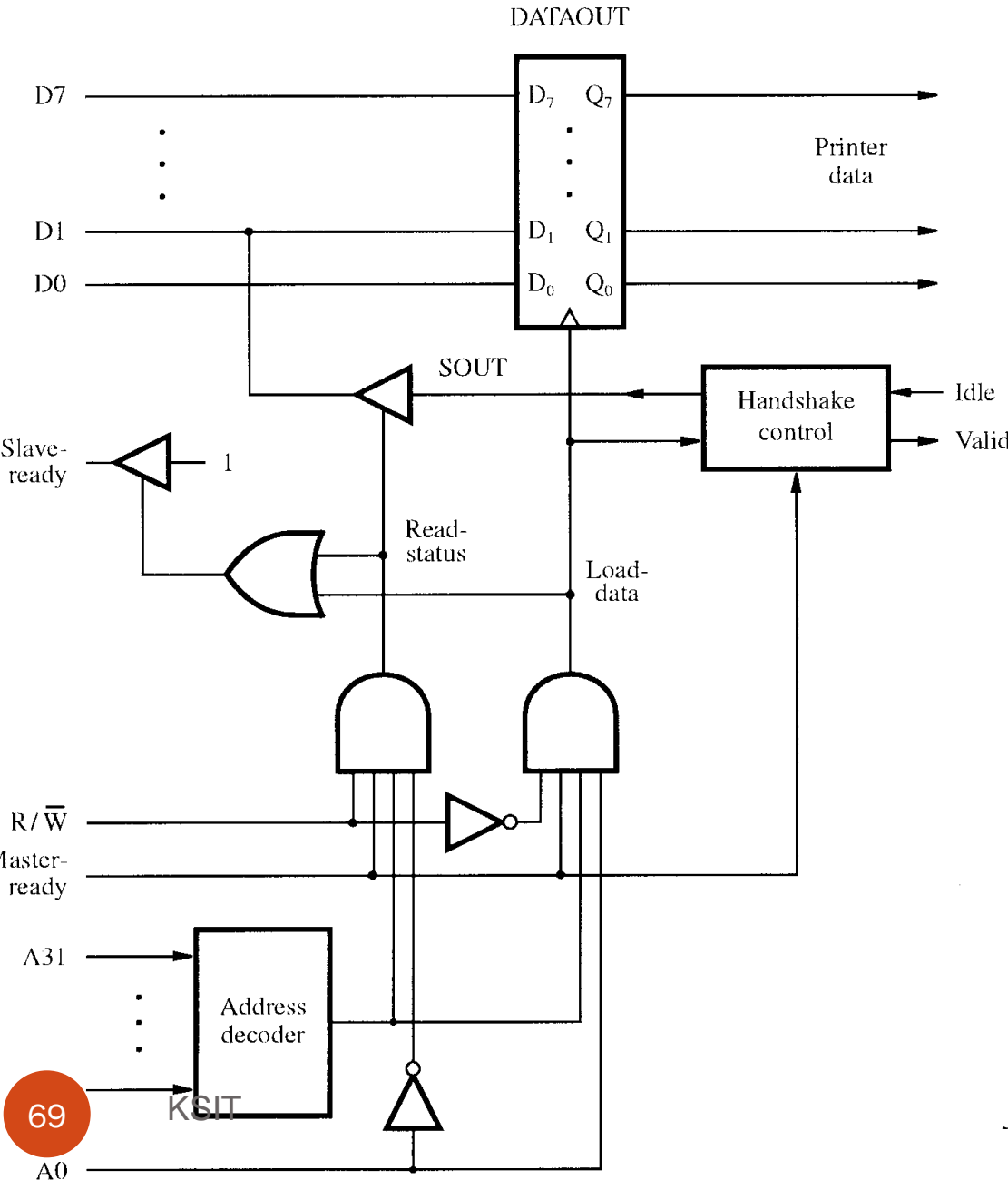
Parallel port (contd..)



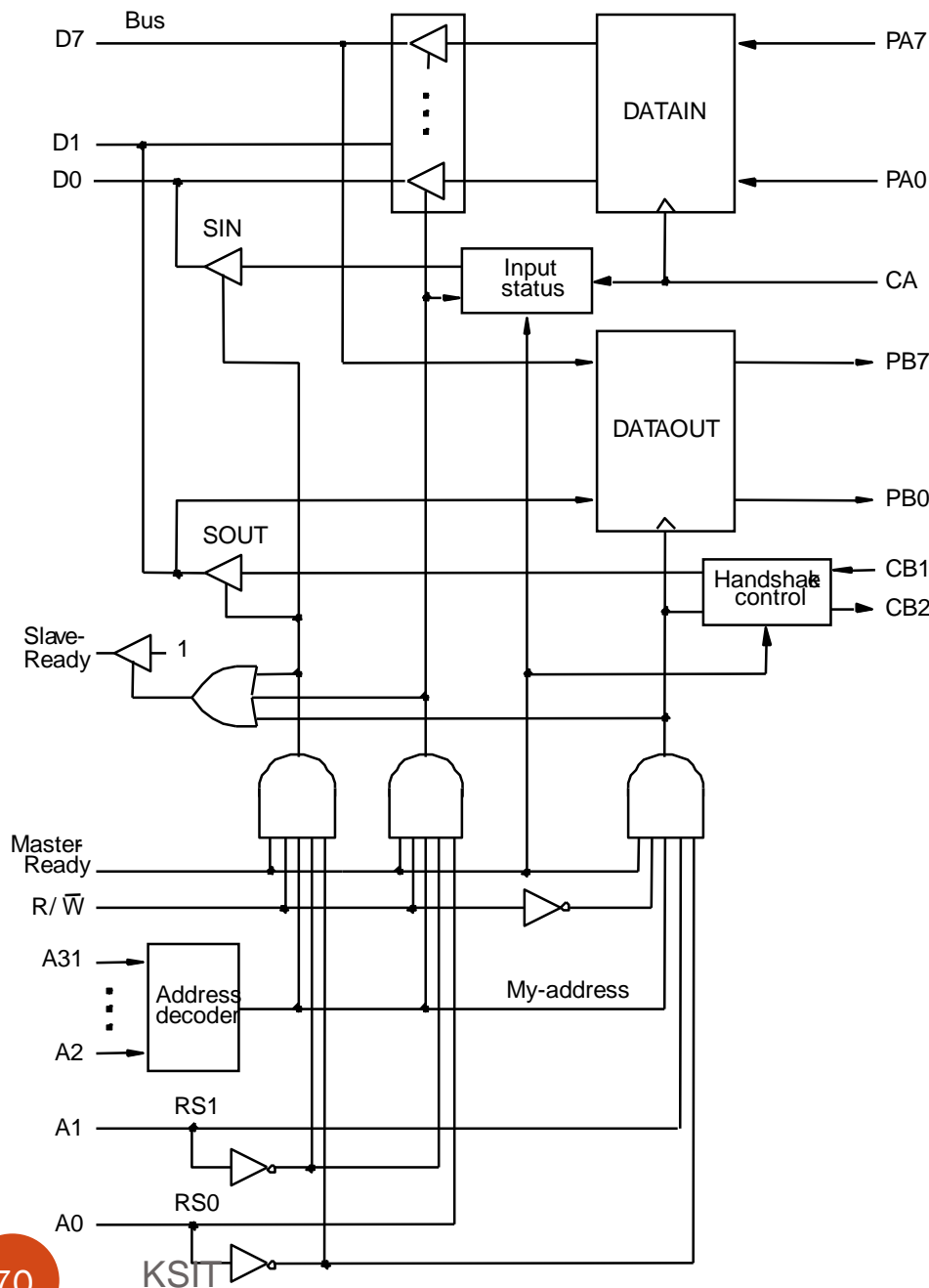
•On the printer side:

- Idle signal line which the printer asserts when it is ready to accept a character. This causes the SOUT flag to be set to 1.
- Processor places a new character into a DATAOUT register.
- Valid signal, asserted by the interface circuit when it places a new character on the data lines.

Output Interface Circuit



- Data lines of the processor bus are connected to the DATAOUT register of the interface.
- The status flag SOUT is connected to the data line D₁ using a three-state driver.
- The three-state driver is turned on, when the control Read-status line is 1.
- Address decoder selects the output interface using address lines A₁ through A₃₁.
- Address line A₀ determines whether the data is to be loaded into the DATAOUT register or status flag is to be read.
- If the Load-data line is 1, then the Valid line is set to 1.
- If the Idle line is 1, then the status flag SOUT is set to 1.



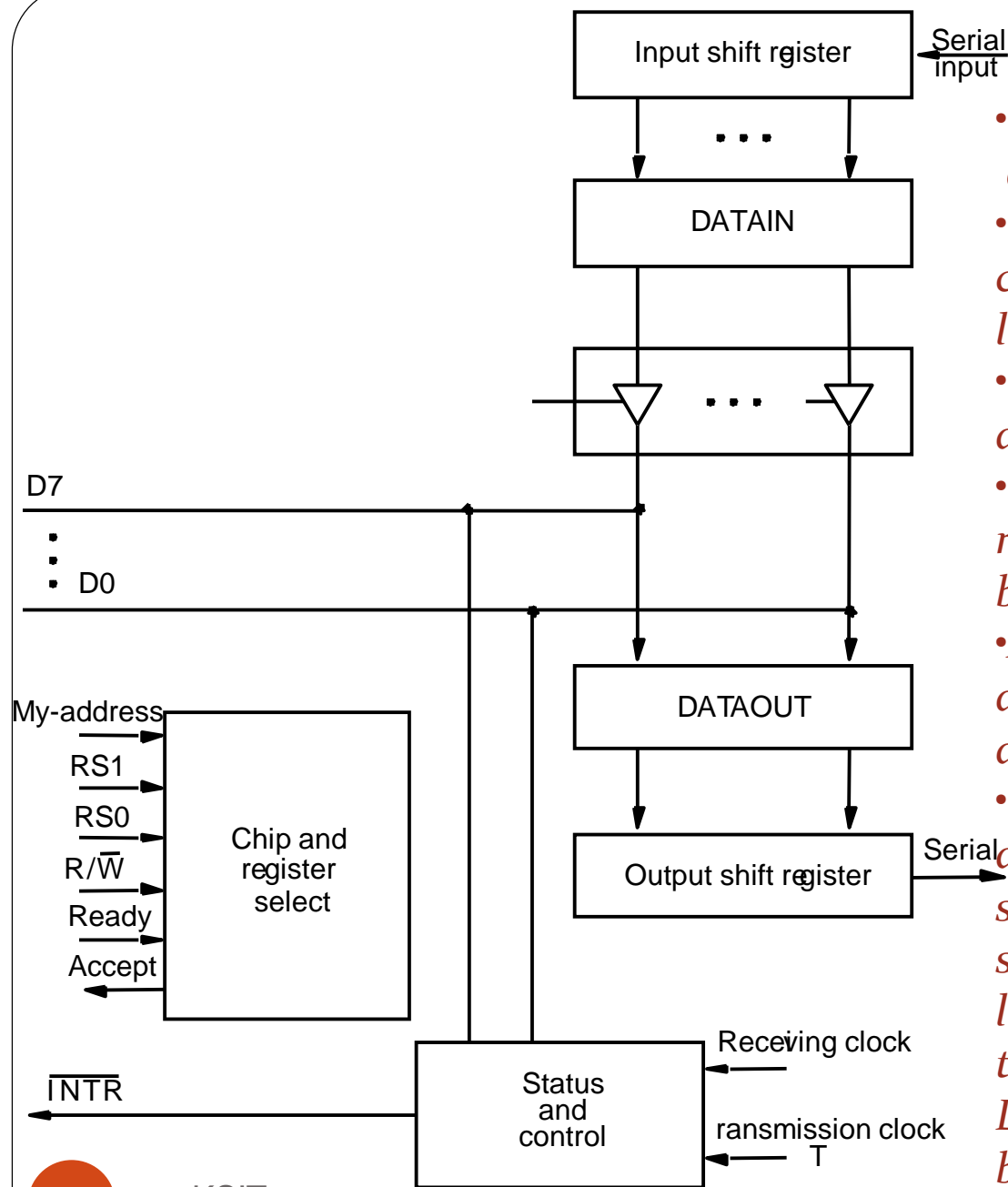
- Combined I/O interface circuit.
- Address bits A_2 through A_{31} , that is 30 bits are used to select the overall interface.
- Address bits A_1 through A_0 , that is, 2 bits select one of the three registers, namely, *DATAIN*, *DATAOUT*, and the status register.
- Status register contains the flags *SIN* and *SOUT* in bits 0 and 1.
- Data lines PA_0 through PA_7 connect the input device to the *DATAIN* register.
- *DATAOUT* register connects the data lines on the processor bus to lines PB_0 through PB_7 which connect to the output device.
- Separate input and output data lines for connection to an I/O device.



- 71

Serial port

- Serial port is used to connect the processor to I/O devices that require transmission of data one bit at a time.
- Serial port communicates in a bit-serial fashion on the device side and bit parallel fashion on the bus side.
 - Transformation between the parallel and serial formats is achieved with shift registers that have parallel access capability.



- Input shift register accepts input one bit at a time from the I/O device.
- Once all the 8 bits are received, the contents of the input shift register are loaded in parallel into DATAIN register.
- Output data in the DATAOUT register are loaded into the output shift register.
- Bits are shifted out of the output shift register and sent out to the I/O device one bit at a time.
- As soon as data from the input shift register are loaded into DATAIN, it can start accepting another 8 bits of data.
- Input shift register and DATAIN register are both used at input so that the input shift register can start receiving another set of 8 bits from the input device after loading the contents to DATAIN, before the processor reads the contents of DATAIN. This is called as double-buffering.

Serial port (contd..)

- Serial interfaces require fewer wires, and hence **serial** transmission is convenient for connecting devices that are physically distant from the computer.
- Speed of transmission of the data over a serial interface is known as the “bit rate”.
 - Bit rate depends on the nature of the devices connected.
- In order to accommodate devices with a range of speeds, a serial interface must be able to use a range of clock speeds.
- Several standard serial interfaces have been developed:
 - Universal Asynchronous Receiver Transmitter (UART) for low-speed serial devices.
 - RS-232-C for connection to communication links.

Standard I/O interfaces

- I/O device is connected to a computer using an interface circuit.
- Do we have to design a different interface for every combination of an I/O device and a computer?
- A practical approach is to develop standard interfaces and protocols.
- A personal computer has:
 - A motherboard which houses the processor chip, main memory and some I/O interfaces.
 - A few connectors into which additional interfaces can be plugged.
- Processor bus is defined by the signals on the processor chip.
 - Devices which require high-speed connection to the processor are connected directly to this bus.

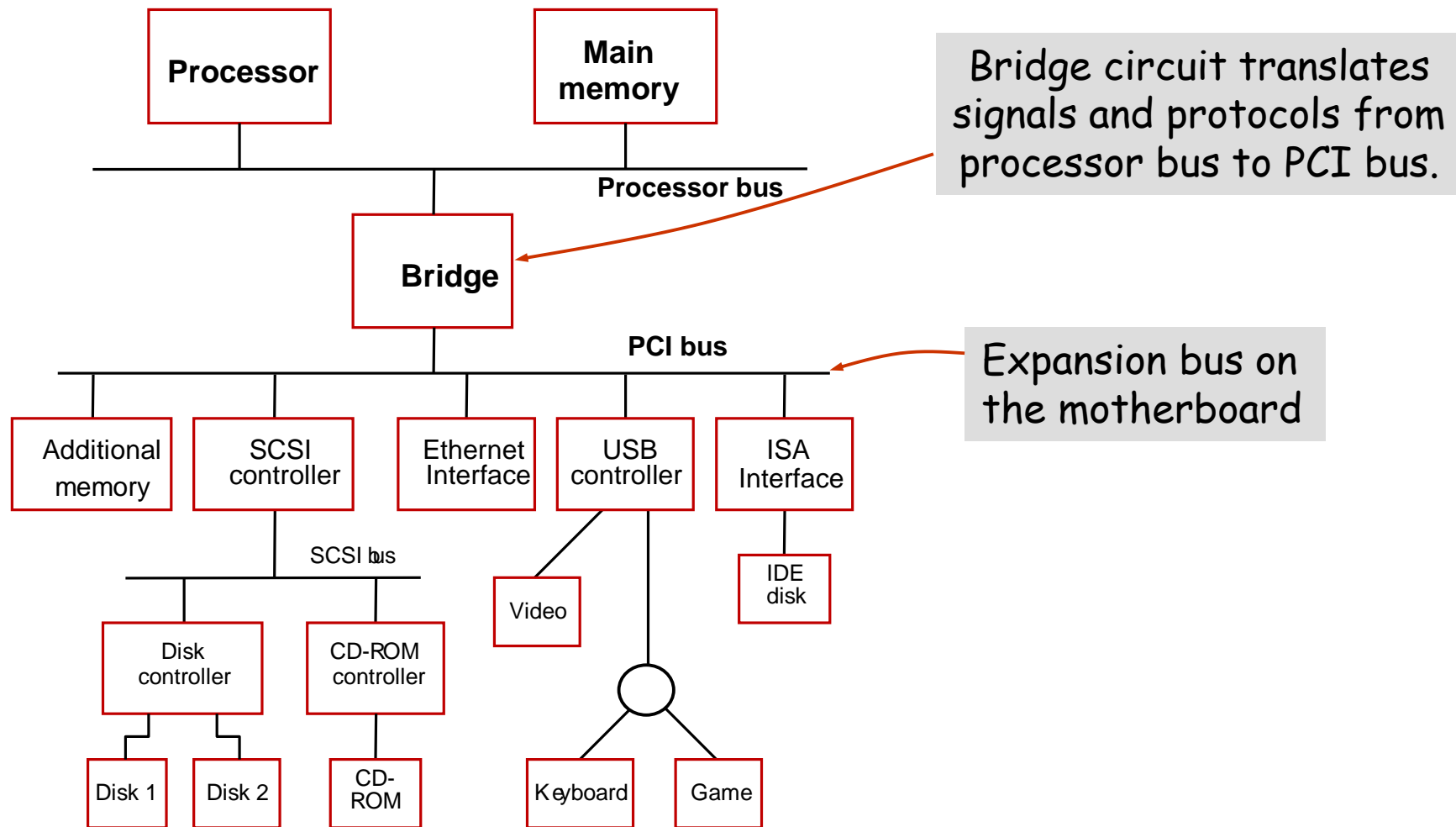
Standard I/O interfaces (contd..)

- Because of electrical reasons only a few devices can be connected directly to the processor bus.
- Motherboard usually provides another bus that can support more devices.
 - Processor bus and the other bus (called as expansion bus) are interconnected by a circuit called “bridge”.
 - Devices connected to the expansion bus experience a small delay in data transfers.
- Design of a processor bus is closely tied to the architecture of the processor.
 - No uniform standard can be defined.
- Expansion bus however can have uniform standard defined.

Standard I/O interfaces (contd..)

- A number of standards have been developed for the expansion bus.
 - Some have evolved by default.
 - For example, IBM's Industry Standard Architecture.
- Three widely used bus standards:
 - PCI (Peripheral Component Interconnect)
 - SCSI (Small Computer System Interface)
 - USB (Universal Serial Bus)

Standard I/O interfaces (contd..)

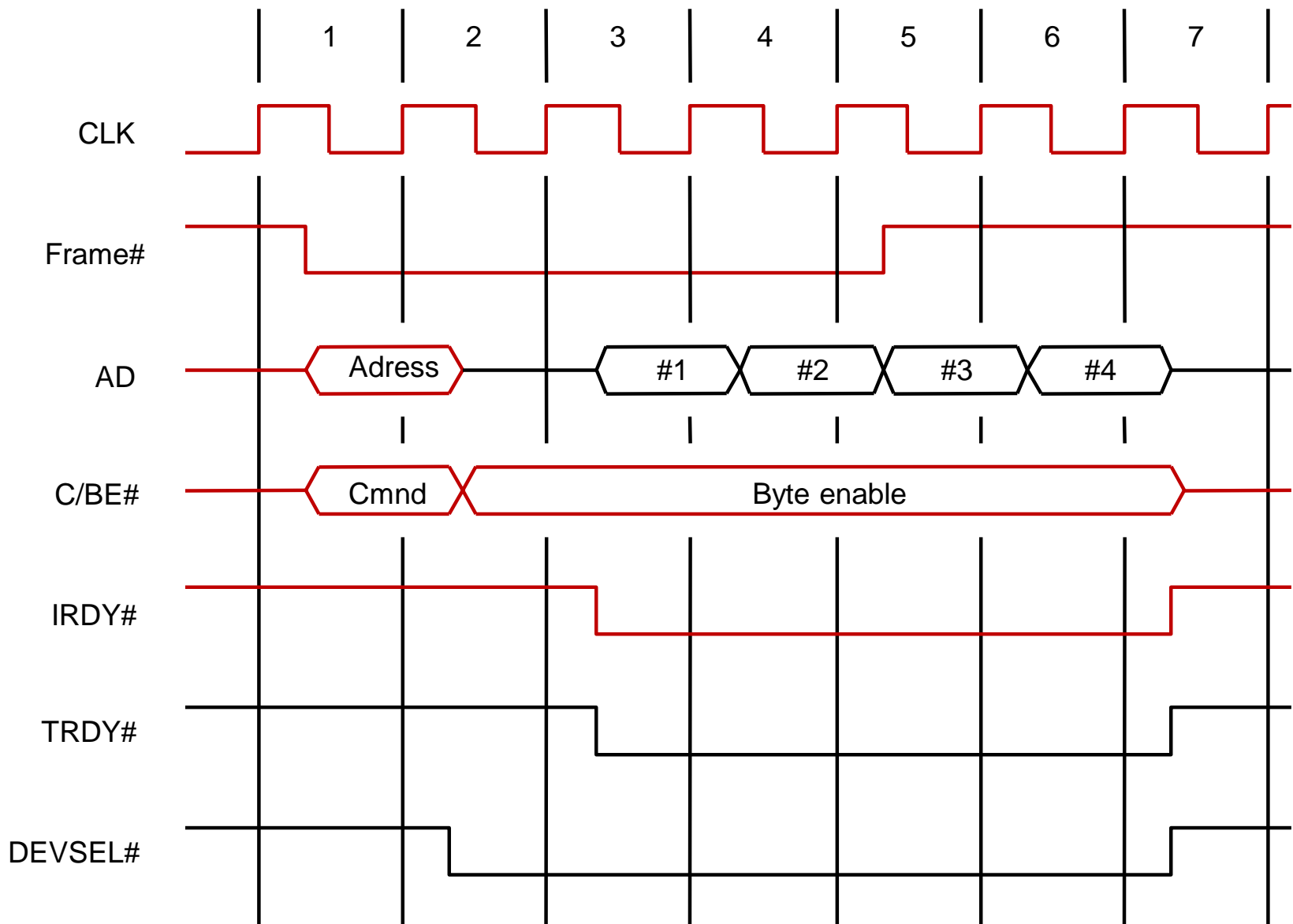


PCI Bus

- *Peripheral Component Interconnect*
- Introduced in 1992
- Low-cost bus
- Processor independent
- Plug-and-play capability
- In today's computers, most memory transfers involve a burst of data rather than just one word. The PCI is designed primarily to support this mode of operation.
- The bus supports three independent address spaces: memory, I/O, and configuration.
- we assumed that the master maintains the address information on the bus until data transfer is completed. But, the address is needed only long enough for the slave to be selected. Thus, the address is needed on the bus for one clock cycle only, freeing the address lines to be used for sending data in subsequent clock cycles. The result is a significant cost reduction.
- A master is called an initiator in PCI terminology. The addressed device that responds to read and write commands is called a target.

Data transfer signals on the PCI bus.

Name	Function
CLK	A 33-MHz or 66-MHz clock.
FRAME#	Sent by the initiator to indicate the duration of a transaction.
AD	32 address/data lines, which may be optionally increased to 64.
C/BE#	4 command/byte-enable lines (8 for a 64-bit bus).
IRD Y#, TRD Y#	Initiator-ready and Target-ready signals.
DEVSEL#	A response from the device indicating that it has recognized its address and is ready for a data transfer transaction.
IDSEL#	Initialization Device Select.



Device Configuration

- When an I/O device is connected to a computer, several actions are needed to configure both the device and the software that communicates with it.
- PCI incorporates in each I/O device interface a small configuration ROM memory that stores information about that device.
- The configuration ROMs of all devices are accessible in the configuration address space. The PCI initialization software reads these ROMs and determines whether the device is a printer, a keyboard, an Ethernet interface, or a disk controller. It can further learn about various device options and characteristics.
- Devices are assigned addresses during the initialization process.
- This means that during the bus configuration operation, devices cannot be accessed based on their address, as they have not yet been assigned one.
- Hence, the configuration address space uses a different mechanism. Each device has an input signal called Initialization Device Select, IDSEL#
- **Electrical characteristics:**
 - PCI bus has been defined for operation with either a 5 or 3.3 V power supply

SCSI Bus

- The acronym SCSI stands for Small Computer System Interface.
- It refers to a standard bus defined by the American National Standards Institute (ANSI) under the designation X3.131 .
- In the original specifications of the standard, devices such as disks are connected to a computer via a 50-wire cable, which can be up to 25 meters in length and can transfer data at rates up to 5 megabytes/s.
- The SCSI bus standard has undergone many revisions, and its data transfer capability has increased very rapidly, almost doubling every two years.
- SCSI-2 and SCSI-3 have been defined, and each has several options.
- Because of various options SCSI connector may have 50, 68 or 80 pins.

SCSI Bus (Contd.,)

- Devices connected to the SCSI bus are not part of the address space of the processor
- The SCSI bus is connected to the processor bus through a SCSI controller. This controller uses DMA to transfer data packets from the main memory to the device, or vice versa.
- A packet may contain a block of data, commands from the processor to the device, or status information about the device.
- A controller connected to a SCSI bus is one of two types – an initiator or a target.
- An initiator has the ability to select a particular target and to send commands specifying the operations to be performed. The disk controller operates as a target. It carries out the commands it receives from the initiator.
- The initiator establishes a logical connection with the intended target.
- Once this connection has been established, it can be suspended and restored as needed to transfer commands and bursts of data.
- While a particular connection is suspended, other device can use the bus to transfer information.
- This ability to overlap data transfer requests is one of the key features of the SCSI bus that leads to its high performance.

SCSI Bus (Contd.,)

- Data transfers on the SCSI bus are always controlled by the target controller.
- To send a command to a target, an initiator requests control of the bus and, after winning arbitration, selects the controller it wants to communicate with and hands control of the bus over to it.
- Then the controller starts a data transfer operation to receive a command from the initiator.

SCSI Bus (Contd.,)

- Assume that processor needs to read block of data from a disk drive and that data are stored in disk sectors that are not contiguous.
- The processor sends a command to the SCSI controller, which causes the following sequence of events to take place:
 1. The SCSI controller, acting as an initiator, contends for control of the bus.
 2. When the initiator wins the arbitration process, it selects the target controller and hands over control of the bus to it.
 3. The target starts an output operation (from initiator to target); in response to this, the initiator sends a command specifying the required read operation.
 4. The target, realizing that it first needs to perform a disk seek operation, sends a message to the initiator indicating that it will temporarily suspend the connection between them. Then it releases the bus.
 5. The target controller sends a command to the disk drive to move the read head to the first sector involved in the requested read operation. Then, it reads the data stored in that sector and stores them in a data buffer. When it is ready to begin transferring data to the initiator, the target requests control of the bus. After it wins arbitration, it reselects the initiator controller, thus restoring the suspended connection.

SCSI Bus (Contd.,)

6. The target transfers the contents of the data buffer to the initiator and then suspends the connection again
7. The target controller sends a command to the disk drive to perform another seek operation. Then, it transfers the contents of the second disk sector to the initiator as before. At the end of this transfers, the logical connection between the two controllers is terminated.
8. As the initiator controller receives the data, it stores them into the main memory using the DMA approach.
9. The SCSI controller sends as interrupt to the processor to inform it that the requested operation has been completed

Operation of SCSI bus from H/W point of view

Category	Name	Function
Data	– DB(0) to – DB(7)	Datalines: Carry one byte of information during the information transfer phase and identify device during arbitration, selection and reselection phases
	– DB(P)	Parity bit for the data bus
Phase	– BSY	Busy: Asserted when the bus is not free
	– SEL	Selection: Asserted during selection and reselection
Information type	– C/D	Control/Data: Asserted during transfer of control information (command, status or message)
	– MSG	Message: indicates that the information being transferred is a message

Table 4. The SCSI bus signals.

Table 4. The SCSI bus signals.(*cont.*)

Category	Name	Function
Handshake	– REQ	Request: Asserted by a target to request a data transfer cycle
	– ACK	Acknowledge: Asserted by the initiator when it has completed a data transfer operation
Direction of transfer	– I/O	Input/Output: Asserted to indicate an input operation (relative to the initiator)
Other	– ATN	Attention: Asserted by an initiator when it wishes to send a message to a target
	– RST	Reset: Causes all device controls to disconnect from the bus and assume their start-up state

Main Phases involved

- Arbitration
 - A controller requests the bus by asserting BSY and by asserting it's associated data line
 - When BSY becomes active, all controllers that are requesting bus examine data lines
- Selection
 - Controller that won arbitration selects target by asserting SEL and data line of target. After that initiator releases BSY line.
 - Target responds by asserting BSY line
 - Target controller will have control on the bus from then
- Information Transfer
 - Handshaking signals are used between initiator and target
 - At the end target releases BSY line
- Reselection

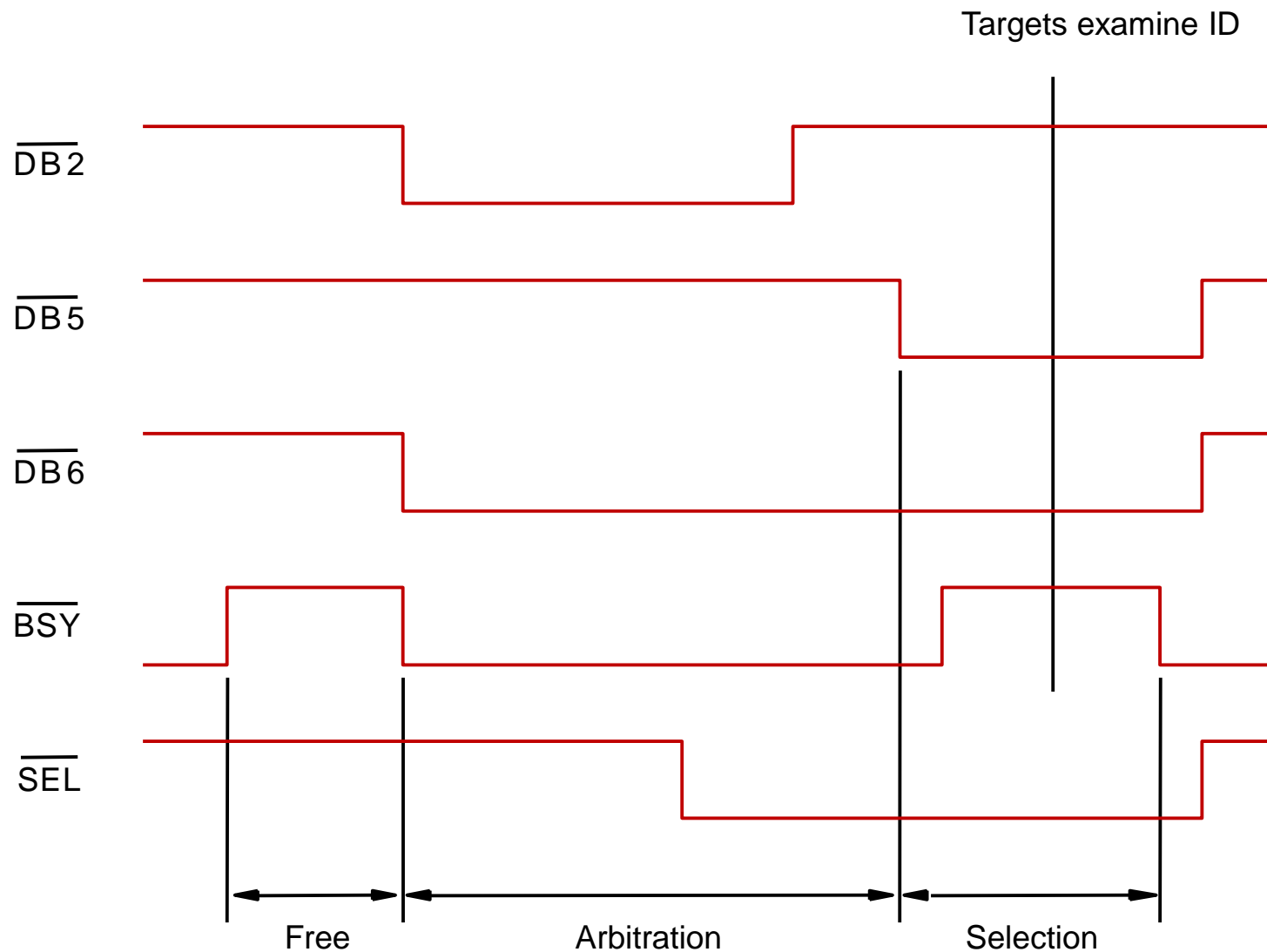
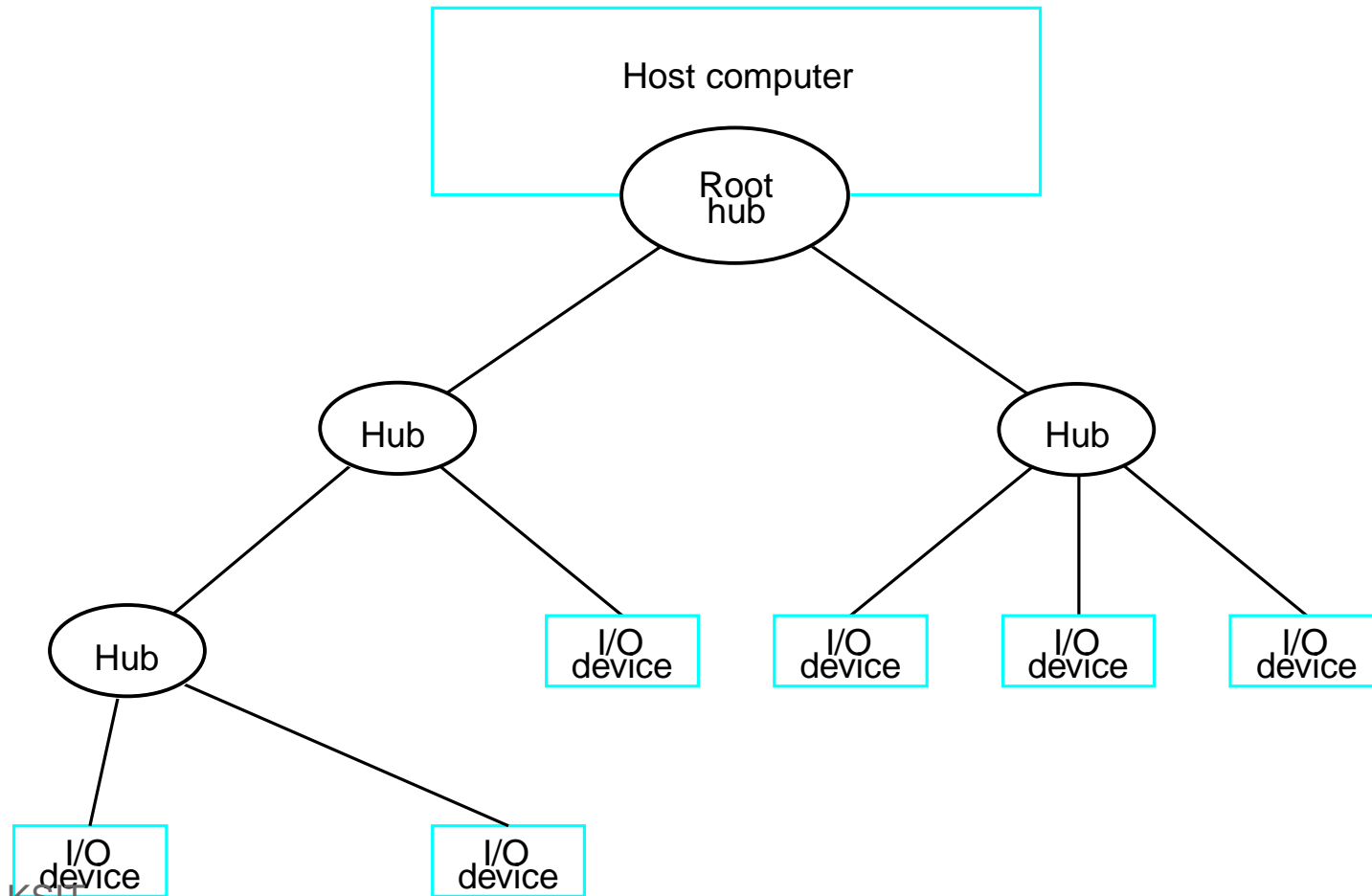


Figure 42. Arbitration and selection on the SCSI bus.
Device 6 wins arbitration and selects device 2.

USB

- Universal Serial Bus (USB) is an industry standard developed through a collaborative effort of several computer and communication companies, including Compaq, Hewlett-Packard, Intel, Lucent, Microsoft, Nortel Networks, and Philips.
- Speed
 - Low-speed(1.5 Mb/s)
 - Full-speed(12 Mb/s)
 - High-speed(480 Mb/s)
- Port Limitation
- Device Characteristics
- Plug-and-play

Universal Serial Bus tree structure



Universal Serial Bus tree structure

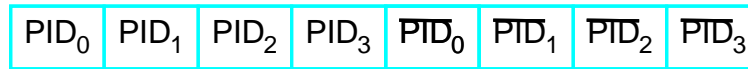
- To accommodate a large number of devices that can be added or removed at any time, the USB has the tree structure as shown in the figure.
- Each node of the tree has a device called a hub, which acts as an intermediate control point between the host and the I/O devices. At the root of the tree, a root hub connects the entire tree to the host computer. The leaves of the tree are the I/O devices being served (for example, keyboard, Internet connection, speaker, or digital TV)
- In normal operation, a hub copies a message that it receives from its upstream connection to all its downstream ports. As a result, a message sent by the host computer is broadcast to all I/O devices, but only the addressed device will respond to that message. However, a message from an I/O device is sent only upstream towards the root of the tree and is not seen by other devices. Hence, the USB enables the host to communicate with the I/O devices, but it does not enable these devices to communicate with each other.

Addressing

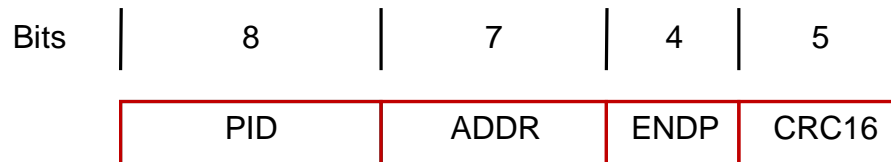
- When a USB is connected to a host computer, its root hub is attached to the processor bus, where it appears as a single device. The host software communicates with individual devices attached to the USB by sending packets of information, which the root hub forwards to the appropriate device in the USB tree.
- Each device on the USB, whether it is a hub or an I/O device, is assigned a 7-bit address. This address is local to the USB tree and is not related in any way to the addresses used on the processor bus.
- A hub may have any number of devices or other hubs connected to it, and addresses are assigned arbitrarily. When a device is first connected to a hub, or when it is powered on, it has the address 0. The hardware of the hub to which this device is connected is capable of detecting that the device has been connected, and it records this fact as part of its own status information. Periodically, the host polls each hub to collect status information and learn about new devices that may have been added or disconnected.
- When the host is informed that a new device has been connected, it uses a sequence of commands to send a reset signal on the corresponding hub port, read information from the device about its capabilities, send configuration information to the device, and assign the device a unique USB address. Once this sequence is completed the device begins normal operation and responds only to the new address.

USB Protocols

- All information transferred over the USB is organized in packets, where a packet consists of one or more bytes of information. There are many types of packets that perform a variety of control functions.
- The information transferred on the USB can be divided into two broad categories: control and data.
 - Control packets perform such tasks as addressing a device to initiate data transfer, acknowledging that data have been received correctly, or indicating an error.
 - Data packets carry information that is delivered to a device.
- A packet consists of one or more fields containing different kinds of information. The first field of any packet is called the packet identifier, PID, which identifies the type of that packet.
- They are transmitted twice. The first time they are sent with their true values, and the second time with each bit complemented
- The four PID bits identify one of 16 different packet types. Some control packets, such as ACK (Acknowledge), consist only of the PID byte.

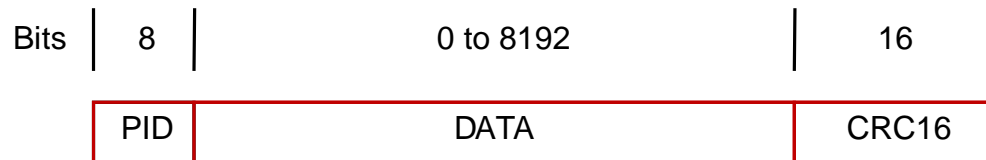


(a) Packet identifier field



Control packets used for controlling data transfer operations are called token packets.

(b) Token packet, IN or OUT



(c) Data packet

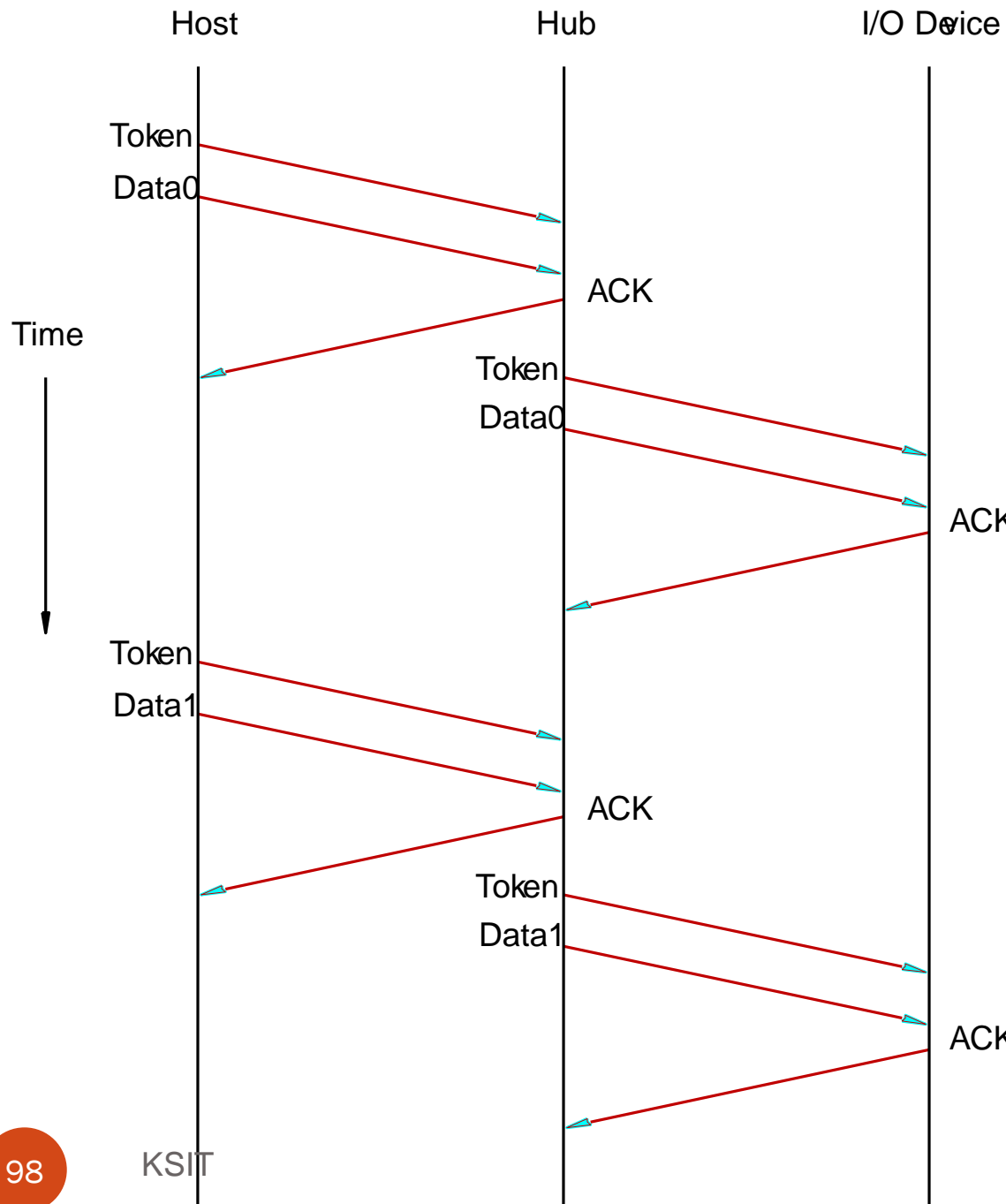


Figure: An output transfer

Isochronous Traffic on USB

- One of the key objectives of the USB is to support the transfer of isochronous data.
- Devices that generate or receive isochronous data require a time reference to control the sampling process.
- To provide this reference, transmission over the USB is divided into frames of equal length.
- A frame is 1 ms long for low- and full-speed data.
- The root hub generates a Start of Frame control packet (SOF) precisely once every 1 ms to mark the beginning of a new frame.
- The arrival of an SOF packet at any device constitutes a regular clock signal that the device can use for its own purposes.
- To assist devices that may need longer periods of time, the SOF packet carries an 11-bit frame number.
- Following each SOF packet, the host carries out input and output transfers for isochronous devices.
- This means that each device will have an opportunity for an input or output transfer once every 1 ms.

Electrical Characteristics

- The cables used for USB connections consist of four wires.
- Two are used to carry power, +5V and Ground.
 - Thus, a hub or an I/O device may be powered directly from the bus, or it may have its own external power connection.
- The other two wires are used to carry data.
- Different signaling schemes are used for different speeds of transmission.
 - At low speed, 1s and 0s are transmitted by sending a high voltage state (5V) on one or the other of the two signal wires. For high-speed links, differential transmission is used.

Model Questions


1. In a situation where multiple devices capable of initiating interrupts are connected to processor, explain the implementation of interrupt priority, using individual INTER and INTA and a common INTR line to all devices.
2. Define the terms 'cycle stealing' and 'block mode'.
3. What is bus arbitration ? Explain the different approaches to bus arbitration.
4. Briefly discuss the main phases involved in the operation of SCSI bus.
5. Explain the tree structure of USB with split bus operation.
6. Explain the following terms I) interrupt service routine II) interrupt latency III) interrupt disabling
7. With a diagram explain daisy chaining technique
8. With a block diagram explain how the printer is interfaced to processor
9. Define two types of SCSI controller.

Model Questions

10. Explain the use of PCI bus in a computer with necessary figure.
11. List the SCSI bus signals with their functions.
12. Define memory mapped IO and IO mapped IO with examples.
13. What are the different methods of DMA? Explain them in brief. Explain the registers in DMA.
14. Explain the serial port and serial interface.
15. What is an interrupt? with example illustrate the concept of interrupts. Explain polling and vectored interrupts.
16. Describe how a read operation is performed on a PCI bus.
17. List the sequence of events that takes place when a processor sends a commands to the SCSI controller.
18. Define exceptions. Explain two kinds of exceptions

Model Questions

19. Draw and explain the general 8 bit parallel processing.
20. Explain the following with respect to USB, I) USB architecture, II) USB addressing, III) USB protocols.
21. List out the functions of an IO interface.



Module 3.

The Memory System

Outline

- Basic Concepts
- Semiconductor RAM Memories
- Read Only Memories
- Speed, Size, and Cost
- Cache Memories – Mapping Functions
- Performance Considerations
- Virtual Memories
- Secondary Storage



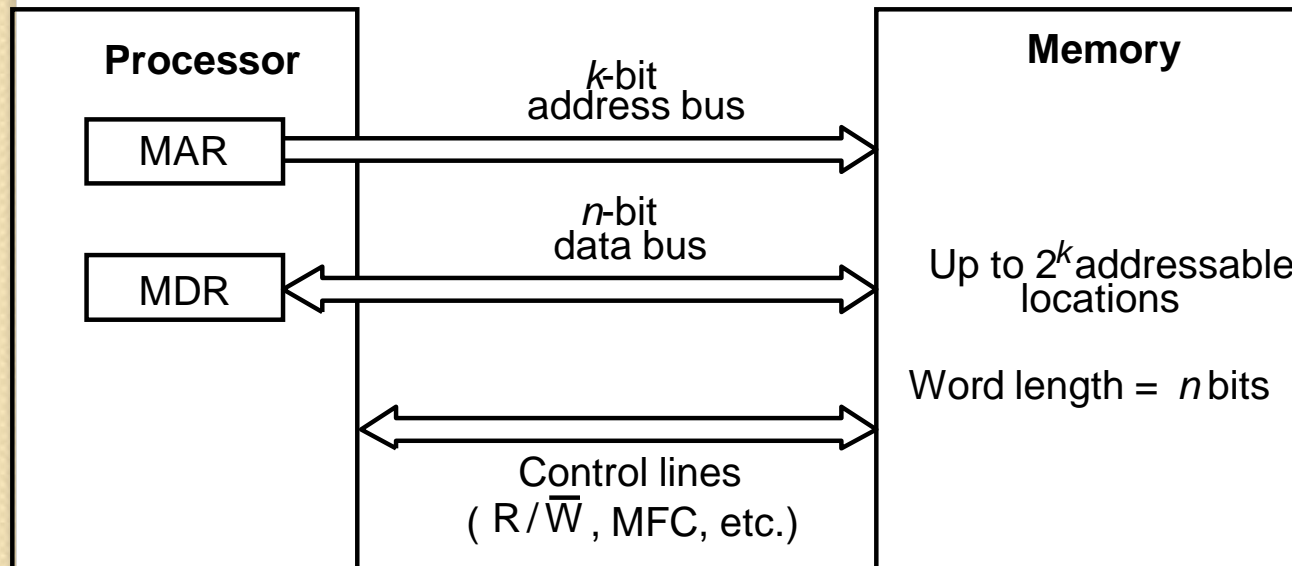
Learning Objectives

1. To learn basic memory circuits and organization of the main memory
2. To analyze cache memory concept, which shortens the effective memory access time
3. To gain knowledge on virtual memory mechanisms which increases the apparent size of the main memory
4. To analyze various secondary memory elements like magnetic disks, optical disks and magnetic tapes



Some basic concepts

- Maximum size of the Main Memory
- byte-addressable
- CPU-Main Memory Connection



Some basic concepts(Contd.,)

- Measures for the speed of a memory:
 - memory access time.
 - memory cycle time.
- An important design issue is to provide a computer system with as large and fast a memory as possible, within a given cost target.
- Several techniques to increase the effective size and speed of the memory:
 - Cache memory (to increase the effective speed).
 - Virtual memory (to increase the effective size).





The Memory System

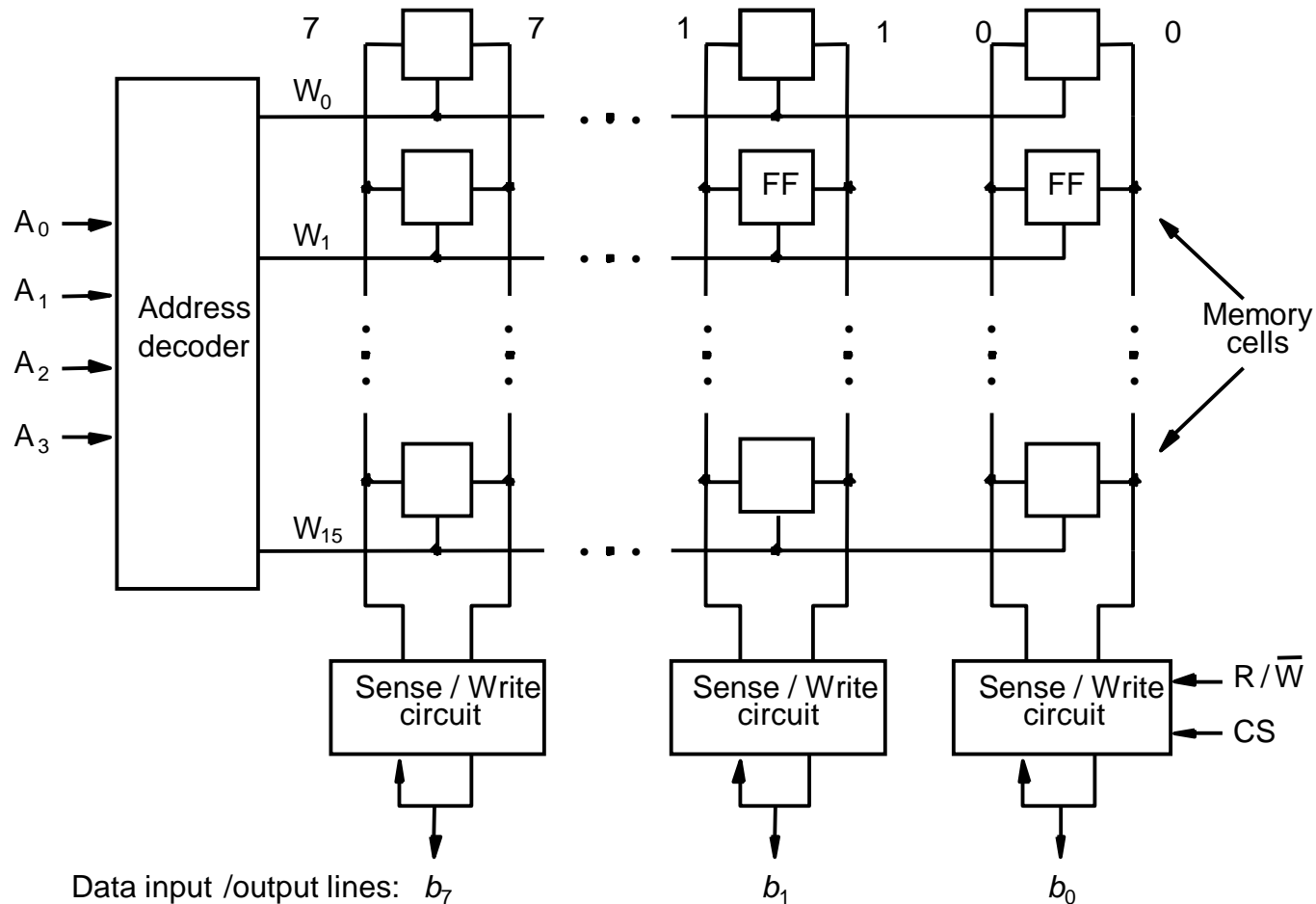
Semiconductor RAM memories

Internal organization of memory chips

- Each memory cell can hold one bit of information.
- Memory cells are organized in the form of an array.
- One row is one memory word.
- All cells of a row are connected to a common line, known as the “word line”.
- Word line is connected to the address decoder.
- Sense/write circuits are connected to the data input/output lines of the memory chip.

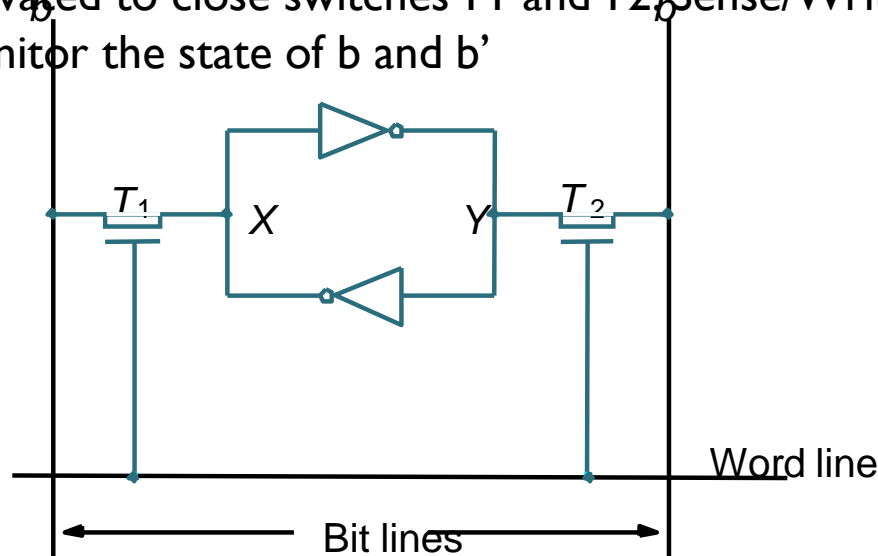


Internal organization of memory chips (Contd.,)



SRAM Cell

- Two transistor inverters are cross connected to implement a basic flip-flop.
- The cell is connected to one word line and two bits lines by transistors T_1 and T_2
- When word line is at ground level, the transistors are turned off and the latch retains its state
- Read operation: In order to read state of SRAM cell, the word line is activated to close switches T_1 and T_2 . Sense/Write circuits at the bottom monitor the state of b and b'



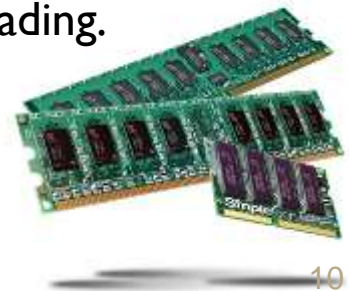
Asynchronous DRAMs

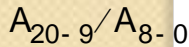
- **Static RAMs (SRAMs):**

- Consist of circuits that are capable of retaining their state as long as the power is applied.
- Volatile memories, because their contents are lost when power is interrupted.
- Access times of static RAMs are in the range of few nanoseconds.
- However, the cost is usually high.

- **Dynamic RAMs (DRAMs):**

- Do not retain their state indefinitely.
- Contents must be periodically refreshed.
- Contents may be refreshed while accessing them for reading.



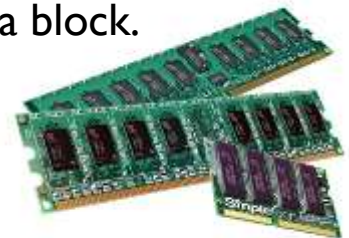


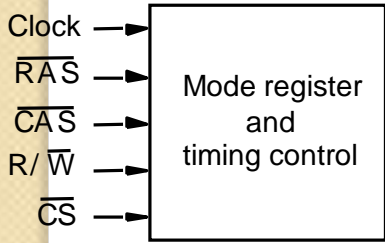
- R/VW



Fast Page Mode

- Suppose if we want to access the consecutive bytes in the selected row.
- This can be done without having to reselect the row.
 - Add a latch at the output of the sense circuits in each row.
 - All the latches are loaded when the row is selected.
 - Different column addresses can be applied to select and place different bytes on the data lines.
- Consecutive sequence of column addresses can be applied under the control signal CAS, without reselecting the row.
 - Allows a block of data to be transferred at a much faster rate than random accesses.
 - A small collection/group of bytes is usually referred to as a block.
- This transfer capability is referred to as the fast page mode feature.





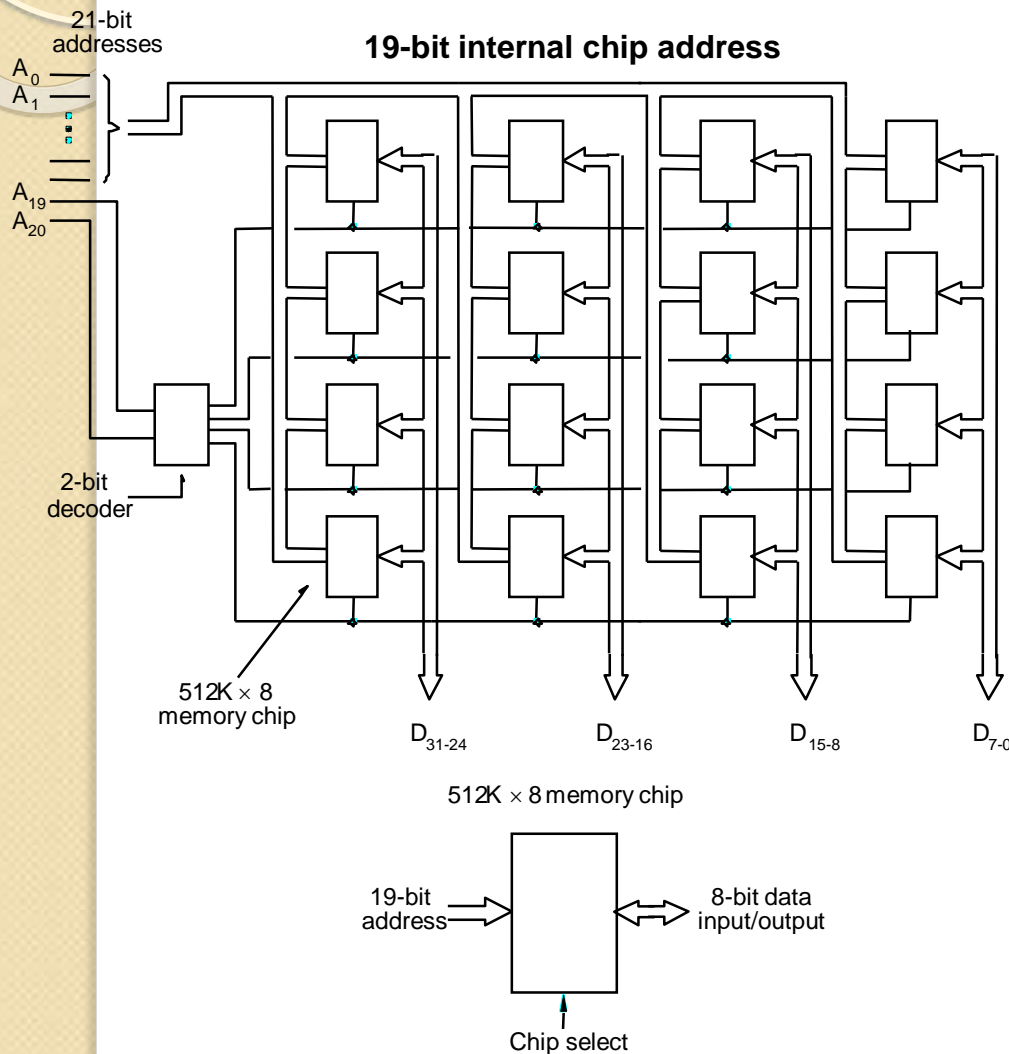
- Operation is directly synchronized with processor clock signal.
 - The outputs of the sense circuits are connected to a latch.
 - During a Read operation, the contents of the cells in a row are loaded onto the latches.
 - During a refresh operation, the contents of the cells are refreshed without changing the contents of the latches.
 - Data held in the latches correspond to the selected columns are transferred to the output.
 - For a burst mode of operation, successive columns are selected using column address counter and clock.
- CAS signal need not be generated externally. A new data is placed during raising edge of the clock

Latency, Bandwidth, and DDR SDRAMs

- Memory latency is the time it takes to transfer a word of data to or from memory
- Memory bandwidth is the number of bits or bytes that can be transferred in one second.
- DDRSDRAMs
 - Cell array is organized in two banks



Static memories



Implement a memory unit of 2M words of 32 bits each.

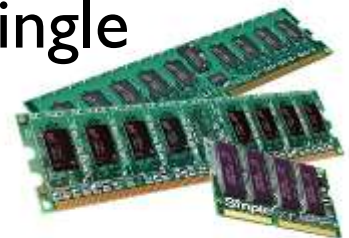
Use 512K × 8 static memory chips. Each column consists of 4 chips. Each chip implements one byte position.

A chip is selected by setting its chip select control line to 1. Selected chip places its data on the data output line, outputs of other chips are in high impedance state. 21 bits to address a 32-bit word. High order 2 bits are needed to select the row, by activating the four Chip Select signals.

19 bits are used to access specific byte locations inside the selected chip.

Dynamic memories

- Large dynamic memory systems can be implemented using DRAM chips in a similar way to static memory systems.
- Placing large memory systems directly on the motherboard will occupy a large amount of space.
 - Also, this arrangement is inflexible since the memory system cannot be expanded easily.
- Packaging considerations have led to the development of larger memory units known as SIMMs (Single In-line Memory Modules) and DIMMs (Dual In-line Memory Modules).
- Memory modules are an assembly of memory chips on a small board that plugs vertically onto a single socket on the motherboard.
 - Occupy less space on the motherboard.
 - Allows for easy expansion by replacement.

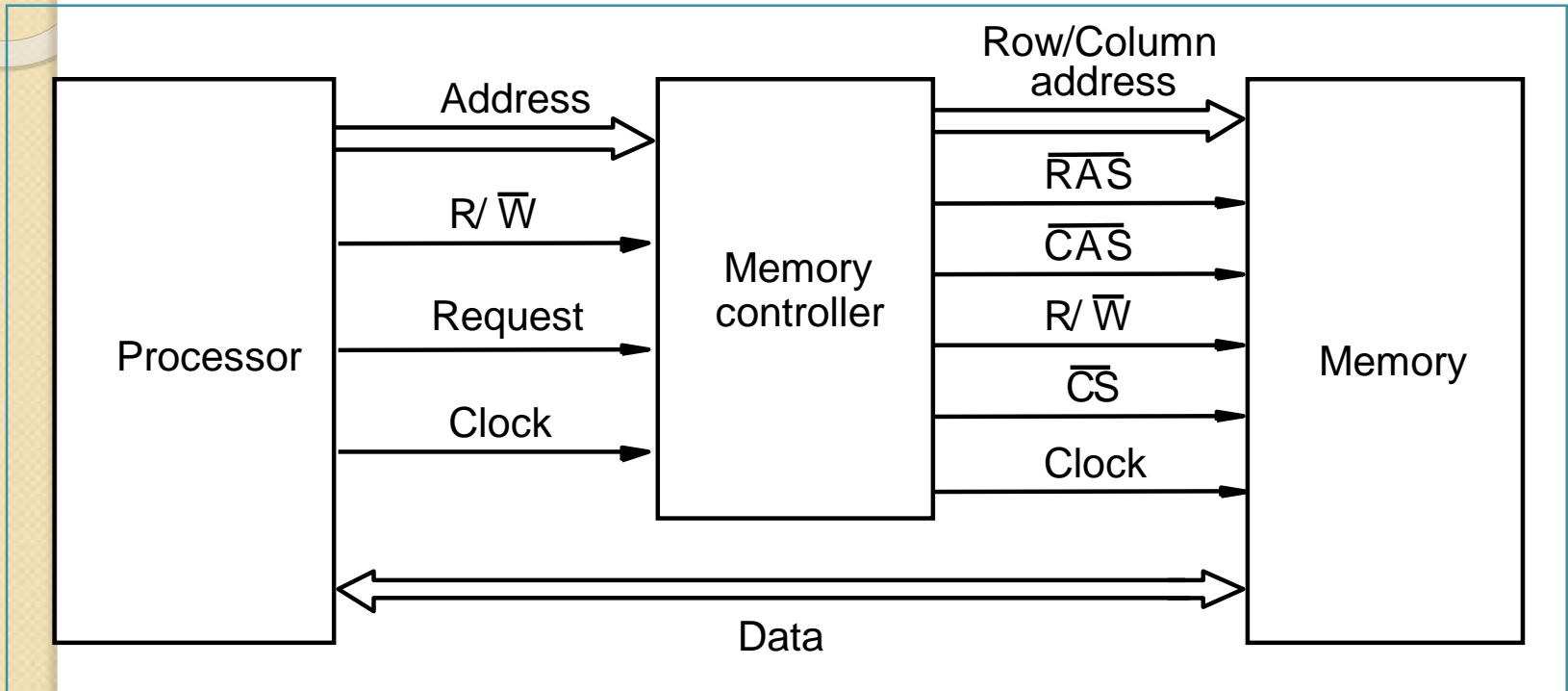


Memory controller

- Recall that in a dynamic memory chip, to reduce the number of pins, multiplexed addresses are used.
- Address is divided into two parts:
 - High-order address bits select a row in the array.
 - They are provided first, and latched using RAS signal.
 - Low-order address bits select a column in the row.
 - They are provided later, and latched using CAS signal.
- However, a processor issues all address bits at the same time.
- In order to achieve the multiplexing, memory controller circuit is inserted between the processor and memory.



Memory controller (contd..)





The Memory System

Read-Only Memories (ROMs)

Read-Only Memories (ROMs)

- SRAM and SDRAM chips are volatile:
 - Lose the contents when the power is turned off.
- Many applications need memory devices to retain contents after the power is turned off.
 - For example, computer is turned on, the operating system must be loaded from the disk into the memory.
 - Store instructions which would load the OS from the disk.
 - Need to store these instructions so that they will not be lost after the power is turned off.
 - We need to store the instructions into a non-volatile memory.
- Non-volatile memory is read in the same manner as volatile memory.
 - Separate writing process is needed to place information in this memory.
 - Normal operation involves only reading of data, this type of memory is called Read-Only memory (ROM).



Read-Only Memories (Contd.,)

■ Read-Only Memory:

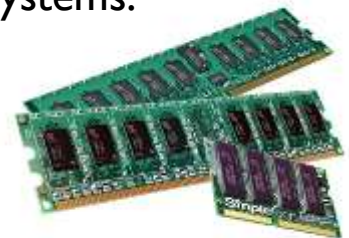
- Data are written into a ROM when it is manufactured.

■ Programmable Read-Only Memory (PROM):

- Allow the data to be loaded by a user.
- Process of inserting the data is irreversible.
- Storing information specific to a user in a ROM is expensive.
- Providing programming capability to a user may be better.

■ Erasable Programmable Read-Only Memory (EPROM):

- Stored data to be erased and new data to be loaded.
- Flexibility, useful during the development phase of digital systems.
- Erasable, reprogrammable ROM.
- Erasure requires exposing the ROM to UV light.



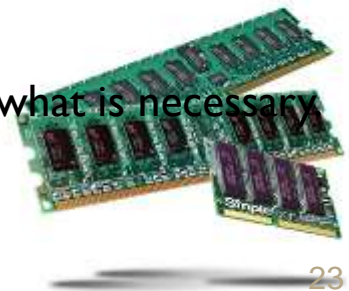
Read-Only Memories (Contd.,)

- **Electrically Erasable Programmable Read-Only Memory (EEPROM):**
 - To erase the contents of EPROMs, they have to be exposed to ultraviolet light.
 - Physically removed from the circuit.
 - EEPROMs the contents can be stored and erased electrically.
- **Flash memory:**
 - Has similar approach to EEPROM.
 - Read the contents of a single cell, but write the contents of an entire block of cells.
 - Flash devices have greater density.
 - Higher capacity and low storage cost per bit.
 - Power consumption of flash memory is very low, making it attractive for use in equipment that is battery-driven.
 - Single flash chips are not sufficiently large, so larger memory modules are implemented using flash cards and flash drives.

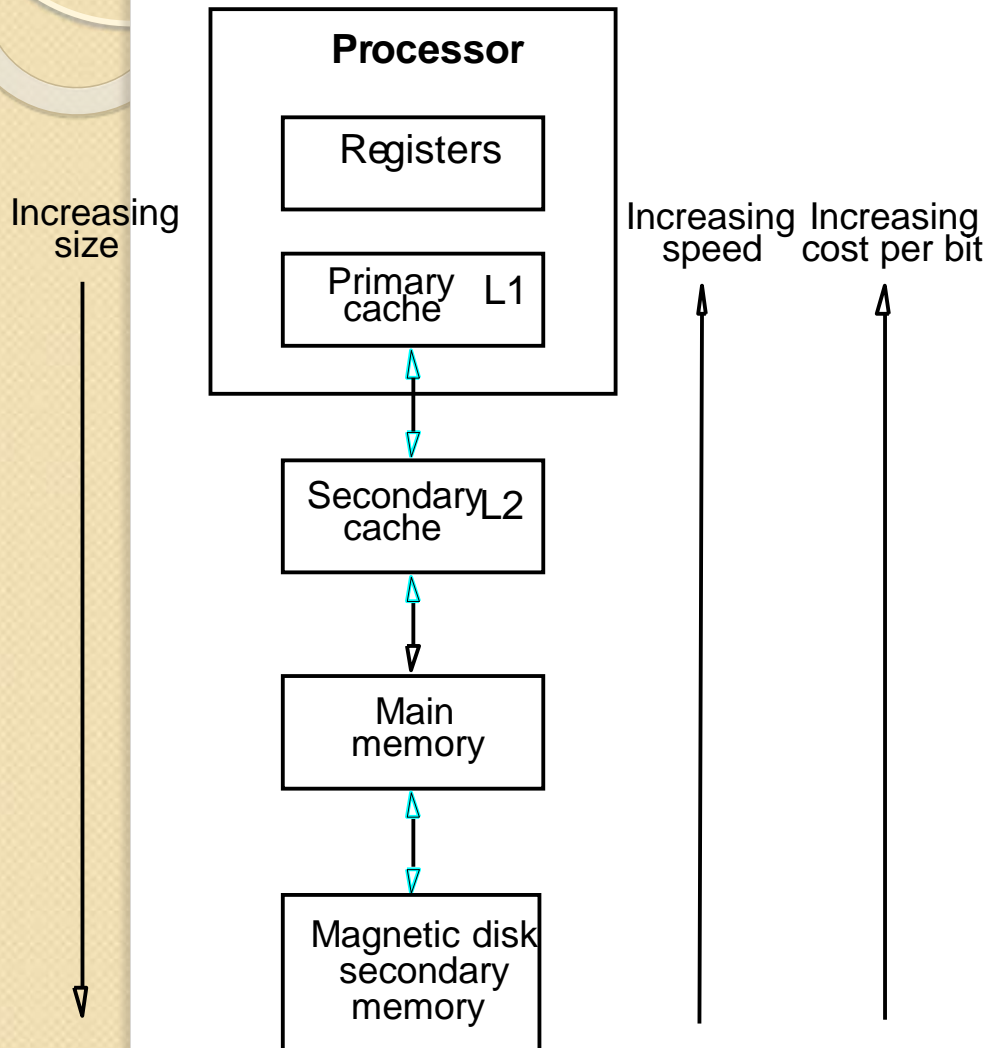


Speed, Size, and Cost

- A big challenge in the design of a computer system is to provide a sufficiently large memory, with a reasonable speed at an affordable cost.
- **Static RAM:**
 - Very fast, but expensive, because a basic SRAM cell has a complex circuit making it impossible to pack a large number of cells onto a single chip.
- **Dynamic RAM:**
 - Simpler basic cell circuit, hence are much less expensive, but significantly slower than SRAMs.
- **Magnetic disks:**
 - Storage provided by DRAMs is higher than SRAMs, but is still less than what is necessary
 - Secondary storage such as magnetic disks provide a large amount of storage, but is much slower than DRAMs.



Memory Hierarchy



- Fastest access is to the data held in processor registers. Registers are at the top of the memory hierarchy.
- Relatively small amount of memory that can be implemented on the processor chip. This is processor cache.
- Two levels of cache. Level 1 (L1) cache is on the processor chip. Level 2 (L2) cache is in between main memory and processor.
- Next level is main memory, implemented as SIMMs. Much larger, but much slower than cache memory.
- Next level is magnetic disks. Huge amount of inexpensive storage.
- Speed of memory access is critical, the idea is to bring instructions and data that will be used in the near future as close to the processor as possible.





The Memory System

Cache Memories

Cache Memories

- **Processor is much faster than the main memory.**
 - As a result, the processor has to spend much of its time waiting while instructions and data are being fetched from the main memory.
 - Major obstacle towards achieving good performance.
- **Speed of the main memory cannot be increased beyond a certain point.**
- **Cache memory is an architectural arrangement which makes the main memory appear faster to the processor than it really is.**
- **Cache memory is based on the property of computer programs known as “locality of reference”.**

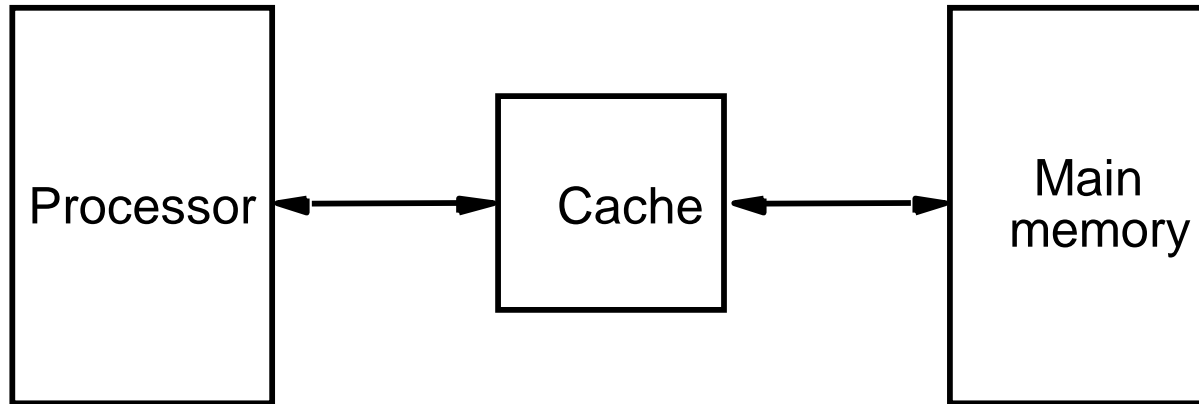


Locality of Reference

- Analysis of programs indicates that many instructions in localized areas of a program are executed repeatedly during some period of time, while the others are accessed relatively less frequently.
 - These instructions may be the ones in a loop, nested loop or few procedures calling each other repeatedly.
 - This is called “locality of reference”.
- **Temporal locality of reference:**
 - Recently executed instruction is likely to be executed again very soon.
- **Spatial locality of reference:**
 - Instructions with addresses close to a recently instruction are likely to be executed soon.



Cache memories



- Processor issues a Read request, a block of words is transferred from the main memory to the cache, one word at a time.
- Subsequent references to the data in this block of words are found in the cache.
- At any given time, only some blocks in the main memory are held in the cache. Which blocks in the main memory are in the cache is determined by a “mapping function”.
- When the cache is full, and a block of words needs to be transferred from the main memory, some block of words in the cache must be replaced. This is determined by a “replacement algorithm”.



Cache hit

- Existence of a cache is transparent to the processor. The processor issues Read and Write requests in the same manner.
- If the data is in the cache it is called a Read or Write hit.
- Read hit:
 - The data is obtained from the cache.
- Write hit:
 - Cache has a replica of the contents of the main memory.
 - Contents of the cache and the main memory may be updated simultaneously. This is the write-through protocol.
 - Update the contents of the cache, and mark it as updated by setting a bit known as the dirty bit or modified bit. The contents of the main memory are updated when this block is replaced. This is write-back or copy-back protocol.



Cache miss

- If the data is not present in the cache, then a Read miss or Write miss occurs.
- **Read miss:**
 - Block of words containing this requested word is transferred from the memory.
 - After the block is transferred, the desired word is forwarded to the processor.
 - The desired word may also be forwarded to the processor as soon as it is transferred without waiting for the entire block to be transferred. This is called load-through or early-restart.
- **Write-miss:**
 - Write-through protocol is used, then the contents of the main memory are updated directly.
 - If write-back protocol is used, the block containing the addressed word is first brought into the cache. The desired word is overwritten with new information.



Cache Coherence Problem

- A bit called as “valid bit” is provided for each block.
- If the block contains valid data, then the bit is set to 1, else it is 0.
- Valid bits are set to 0, when the power is just turned on.
- When a block is loaded into the cache for the first time, the valid bit is set to 1.
- Data transfers between main memory and disk occur directly bypassing the cache.
- When the data on a disk changes, the main memory block is also updated.
- However, if the data is also resident in the cache, then the valid bit is set to 0.
- What happens if the data in the disk and main memory changes and the write-back protocol is being used?
- In this case, the data in the cache may also have changed and is indicated by the dirty bit.
- The copies of the data in the cache, and the main memory are different. This is called the cache coherence problem.
- One option is to force a write-back before the main memory is updated from the disk.

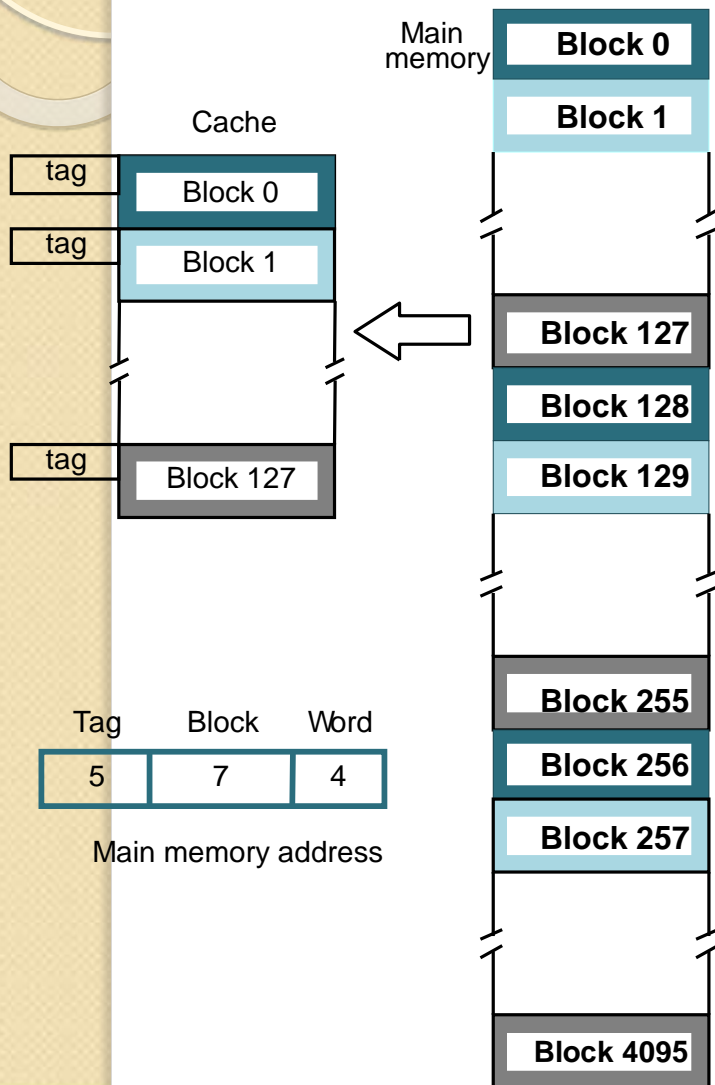


Mapping functions

- Mapping functions determine how memory blocks are placed in the cache.
- A simple processor example:
 - Cache consisting of 128 blocks of 16 words each.
 - Total size of cache is 2048 (2K) words.
 - Main memory is addressable by a 16-bit address.
 - Main memory has 64K words.
 - Main memory has 4K blocks of 16 words each.
- Three mapping functions:
 - Direct mapping
 - Associative mapping
 - Set-associative mapping.

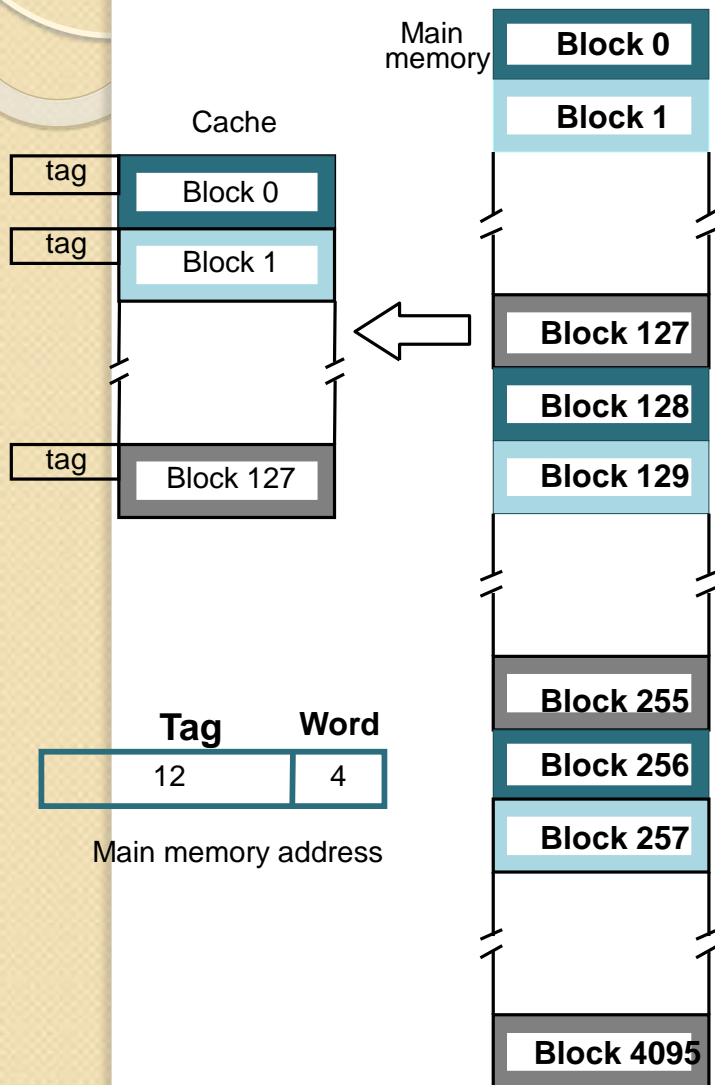


Direct mapping



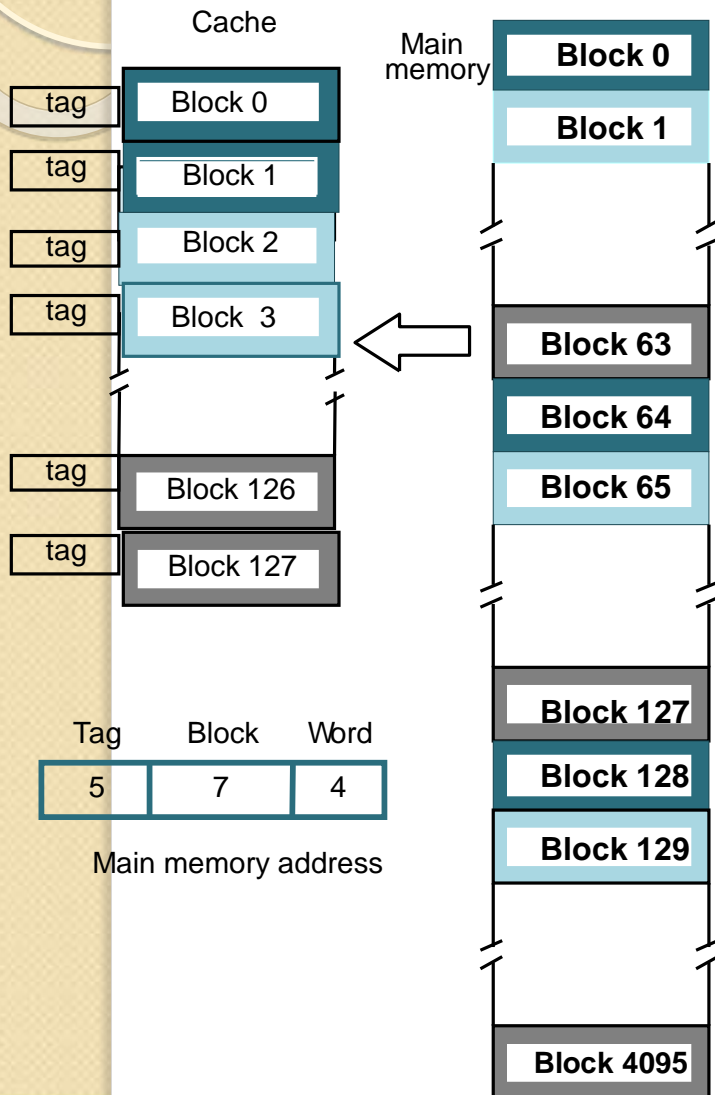
- Block j of the main memory maps to j modulo 128 of the cache. 0 maps to 0, 129 maps to 1.
- More than one memory block is mapped onto the same position in the cache.
- May lead to contention for cache blocks even if the cache is not full.
- Resolve the contention by allowing new block to replace the old block, leading to a trivial replacement algorithm.
- Memory address is divided into three fields:
 - Low order 4 bits determine one of the 16 words in a block.
 - When a new block is brought into the cache, the the next 7 bits determine which cache block this new block is placed in.
 - High order 5 bits determine which of the possible 32 blocks is currently present in the cache. These are tag bits.
- Simple to implement but not very flexible.

Associative mapping



- *Main memory block can be placed into any cache position.*
- *Memory address is divided into two fields:*
 - *Low order 4 bits identify the word within a block.*
 - *High order 12 bits or tag bits identify a memory block when it is resident in the cache.*
- *Flexible, and uses cache space efficiently.*
- *Replacement algorithms can be used to replace an existing block in the cache when the cache is full.*
- *Cost is higher than direct-mapped cache because of the need to search all 128 patterns to determine whether a given block is in the cache.*

Set-Associative mapping



Blocks of cache are grouped into sets.

Mapping function allows a block of the main memory to reside in any block of a specific set.

Divide the cache into 64 sets, with two blocks per set. Memory block 0, 64, 128 etc. map to block 0, and they can occupy either of the two positions.

Memory address is divided into three fields:

- 6 bit field determines the set number.
- High order 6 bit fields are compared to the tag fields of the two blocks in a set.

Set-associative mapping combination of direct and associative mapping.

Number of blocks per set is a design parameter.

- One extreme is to have all the blocks in one set, requiring no set bits (fully associative mapping).
- Other extreme is to have one block per set, is the same as direct mapping.

Replacement Algorithms

- In a direct mapped cache, the position of each block is predetermined – no replacement strategy exists.
- In associative and set associative caches there exists some flexibility.
- Least recently used blocks LRU – LRU replacement algorithm.



The Memory System

Performance considerations

Performance considerations

- A key design objective of a computer system is to achieve the best possible performance at the lowest possible cost.
 - Price/performance ratio is a common measure of success.
- Performance of a processor depends on:
 - How fast machine instructions can be brought into the processor for execution.
 - How fast the instructions can be executed.

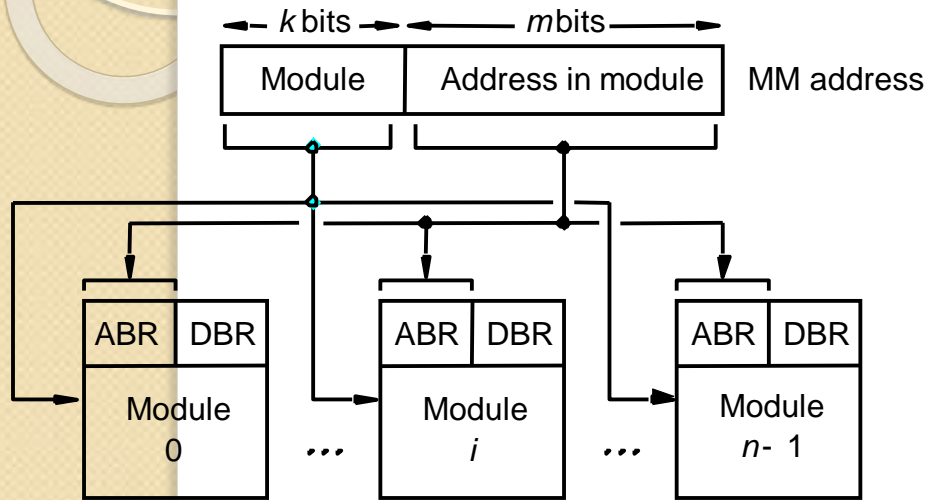


Interleaving

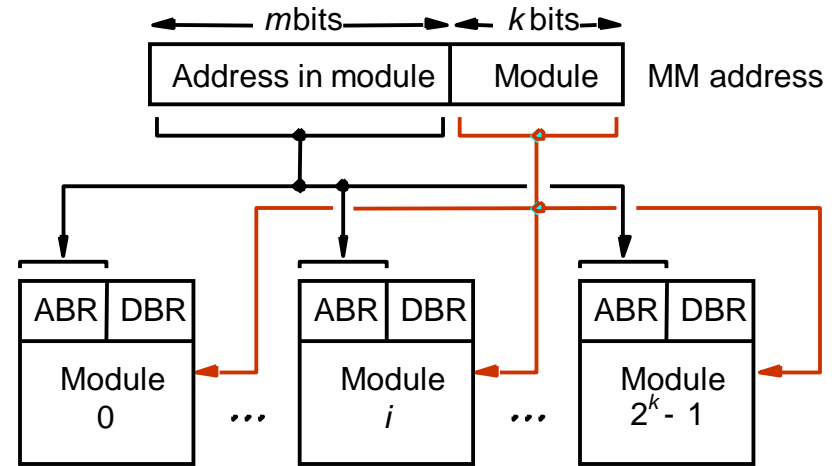
- Divides the memory system into a number of memory modules. Each module has its own address buffer register (ABR) and data buffer register (DBR).
- Arranges addressing so that successive words in the address space are placed in different modules.
- When requests for memory access involve consecutive addresses, the access will be to different modules.
- Since parallel access to these modules is possible, the average rate of fetching words from the Main Memory can be increased.



Methods of address layouts



- Consecutive words are placed in a module.
- High-order k bits of a memory address determine the module.
- Low-order m bits of a memory address determine the word within a module.
- When a block of words is transferred from main memory to cache, only one module is busy at a time.



- Consecutive words are located in consecutive modules.
- Consecutive addresses can be located in consecutive modules.
- While transferring a block of data, several memory modules can be kept busy at the same time.



Hit Rate and Miss Penalty

- Hit rate
- Miss penalty
- Hit rate can be improved by increasing block size, while keeping cache size constant
- Block sizes that are neither very small nor very large give best results.
- Miss penalty can be reduced if load-through approach is used when loading new blocks into cache.



Caches on the processor chip

- In high performance processors 2 levels of caches are normally used.
- Avg access time in a system with 2 levels of caches is

$$T_{ave} = h_1c_1 + (1-h_1)h_2c_2 + (1-h_1)(1-h_2)M$$



Other Performance Enhancements

Write buffer

■ Write-through:

- Each write operation involves writing to the main memory.
- If the processor has to wait for the write operation to be complete, it slows down the processor.
- Processor does not depend on the results of the write operation.
- Write buffer can be included for temporary storage of write requests.
- Processor places each write request into the buffer and continues execution.
- If a subsequent Read request references data which is still in the write buffer, then this data is referenced in the write buffer.

■ Write-back:

- Block is written back to the main memory when it is replaced.
- If the processor waits for this write to complete, before reading the new block, it is slowed down.
- Fast write buffer can hold the block to be written, and the new block can be read first.



Other Performance Enhancements (Contd.,)

Prefetching

- New data are brought into the processor when they are first needed.
- Processor has to wait before the data transfer is complete.
- Prefetch the data into the cache before they are actually needed, or a before a Read miss occurs.
- Prefetching can be accomplished through software by including a special instruction in the machine language of the processor.
 - **Inclusion of prefetch instructions increases the length of the programs.**
- Prefetching can also be accomplished using hardware:
 - **Circuitry that attempts to discover patterns in memory references and then prefetches according to this pattern.**



Other Performance Enhancements (Contd.,)

Lockup-Free Cache

- *Prefetching scheme does not work if it stops other accesses to the cache until the prefetch is completed.*
- *A cache of this type is said to be “locked” while it services a miss.*
- *Cache structure which supports multiple outstanding misses is called a lockup free cache.*
- *Since only one miss can be serviced at a time, a lockup free cache must include circuits that keep track of all the outstanding misses.*
- *Special registers may hold the necessary information about these misses.*





The Memory System

Virtual Memory

Virtual memories

- Recall that an important challenge in the design of a computer system is to provide a large, fast memory system at an affordable cost.
- Architectural solutions to increase the effective speed and size of the memory system.
- Cache memories were developed to increase the effective speed of the memory system.
- Virtual memory is an architectural solution to increase the effective size of the memory system.



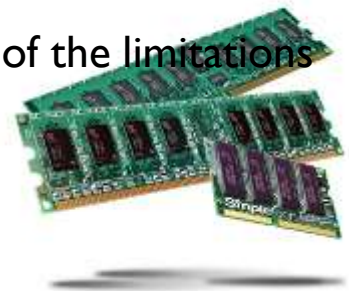
Virtual memories (contd..)

- Recall that the **addressable memory space** depends on the number of address bits in a computer.
 - For example, if a computer issues 32-bit addresses, the addressable memory space is 4G bytes.
- **Physical main memory** in a computer is generally not as large as the entire possible addressable space.
 - Physical memory typically ranges from a few hundred megabytes to 1 G bytes.
- **Large programs that cannot fit completely into the main memory** have their parts stored on **secondary storage devices** such as magnetic disks.
 - Pieces of programs must be transferred to the main memory from secondary storage before they can be executed.



Virtual memories (contd..)

- When a new piece of a program is to be transferred to the main memory, and the main memory is full, then some other piece in the main memory must be replaced.
 - Recall this is very similar to what we studied in case of cache memories.
- Operating system automatically transfers data between the main memory and secondary storage.
 - Application programmer need not be concerned with this transfer.
 - Also, application programmer does not need to be aware of the limitations imposed by the available physical memory.

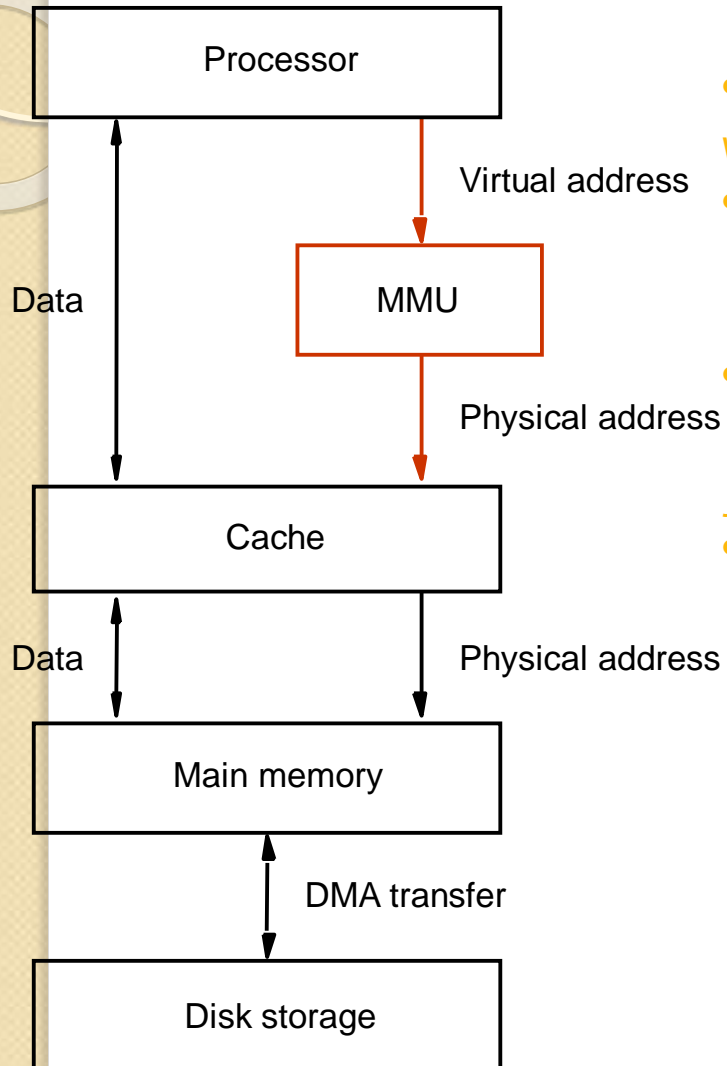


Virtual memories (contd..)

- Techniques that automatically move program and data between main memory and secondary storage when they are required for execution are called virtual-memory techniques.
- Programs and processors reference an instruction or data independent of the size of the main memory.
- Processor issues binary addresses for instructions and data.
 - These binary addresses are called logical or virtual addresses.
- **Virtual addresses are translated into physical addresses** by a combination of hardware and software subsystems.
 - If virtual address refers to a part of the program that is currently in the main memory, it is accessed immediately.
 - If the address refers to a part of the program that is not currently in the main memory, it is first transferred to the main memory before it can be used.



Virtual memory organization



- *Memory management unit (MMU) translates virtual addresses into physical addresses.*
- *If the desired data or instructions are in the main memory they are fetched as described previously.*
- *If the desired data or instructions are not in the main memory, they must be transferred from secondary storage to the main memory.*
- *MMU causes the operating system to bring the data from the secondary storage into the main memory.*

Address translation

- Assume that program and data are composed of fixed-length units called pages.
- A page consists of a block of words that occupy contiguous locations in the main memory.
- Page is a basic unit of information that is transferred between secondary storage and main memory.
- Size of a page commonly ranges from 2K to 16K bytes.
 - Pages should not be too small, because the access time of a secondary storage device is much larger than the main memory.
 - Pages should not be too large, else a large portion of the page may not be used, and it will occupy valuable space in the main memory.



Address translation (contd..)

- Concepts of virtual memory are similar to the concepts of cache memory.
- Cache memory:
 - Introduced to bridge the speed gap between the processor and the main memory.
 - Implemented in hardware.
- Virtual memory:
 - Introduced to bridge the speed gap between the main memory and secondary storage.
 - Implemented in part by software.



Address translation (contd..)

- Each virtual or logical address generated by a processor is interpreted as a virtual page number (high-order bits) plus an offset (low-order bits) that specifies the location of a particular byte within that page.
- Information about the main memory location of each page is kept in the page table.
 - Main memory address where the page is stored.
 - Current status of the page.
- Area of the main memory that can hold a page is called as page frame.
- Starting address of the page table is kept in a page table base register.

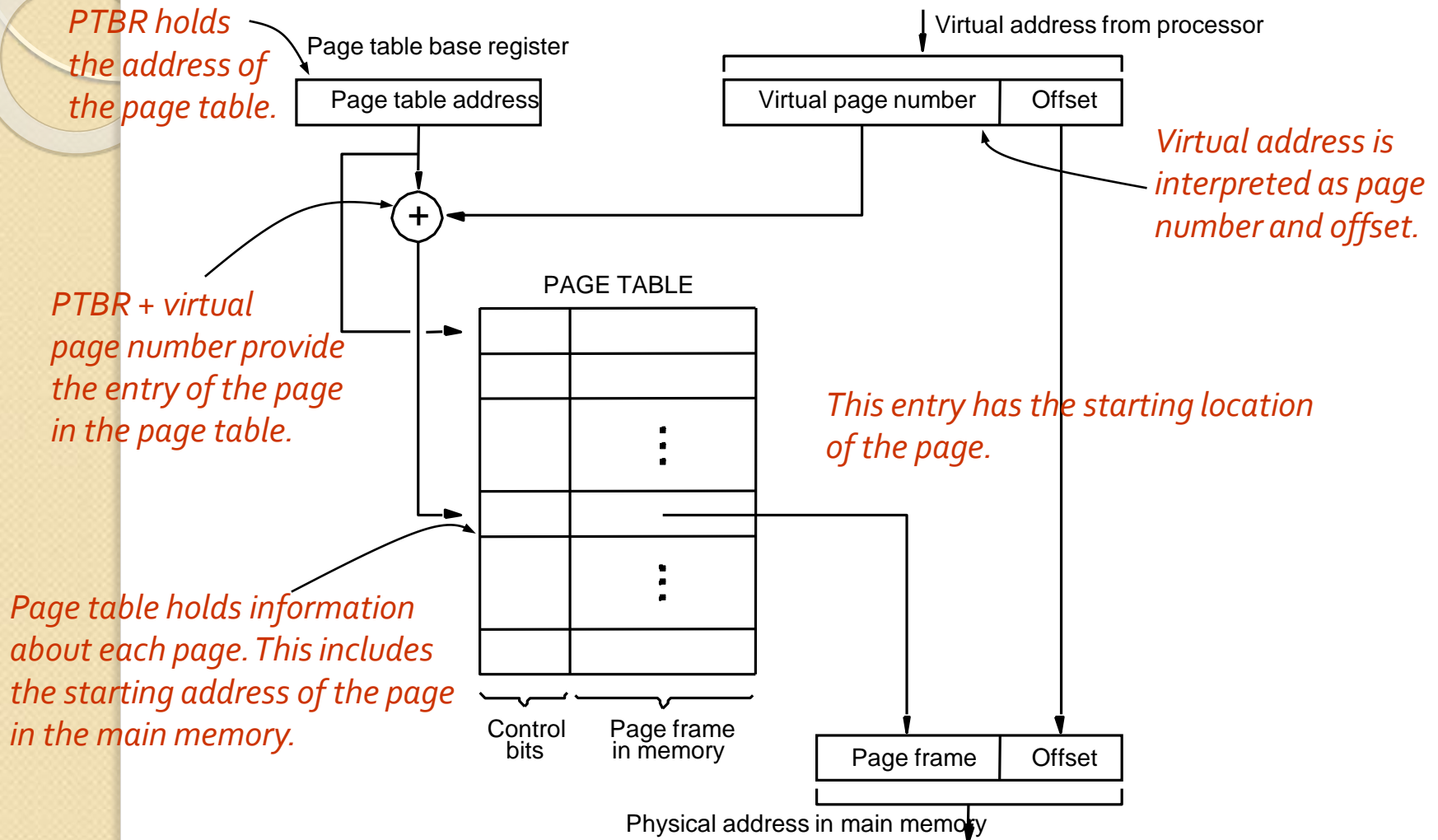


Address translation (contd..)

- Virtual page number generated by the processor is added to the contents of the page table base register.
 - This provides the address of the corresponding entry in the page table.
- The contents of this location in the page table give the starting address of the page if the page is currently in the main memory.



Address translation (contd..)



Address translation (contd..)

- Page table entry for a page also includes some control bits which describe the status of the page while it is in the main memory.
- One bit indicates the validity of the page.
 - Indicates whether the page is actually loaded into the main memory.
 - Allows the operating system to invalidate the page without actually removing it.
- One bit indicates whether the page has been modified during its residency in the main memory.
 - This bit determines whether the page should be written back to the disk when it is removed from the main memory.
 - Similar to the dirty or modified bit in case of cache memory.



Address translation (contd..)

- Other control bits for various other types of restrictions that may be imposed.
 - For example, a program may only have read permission for a page, but not write or modify permissions.



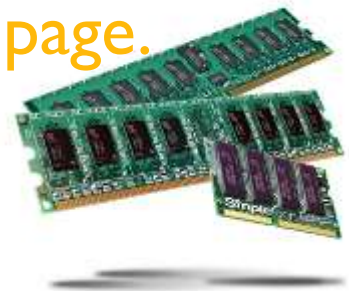
Address translation (contd..)

- Where should the page table be located?
- Recall that the page table is used by the MMU for every read and write access to the memory.
 - Ideal location for the page table is within the MMU.
- Page table is quite large.
- MMU is implemented as part of the processor chip.
- Impossible to include a complete page table on the chip.
- Page table is kept in the main memory.
- A copy of a small portion of the page table can be accommodated within the MMU.
 - Portion consists of page table entries that correspond to the most recently accessed pages.

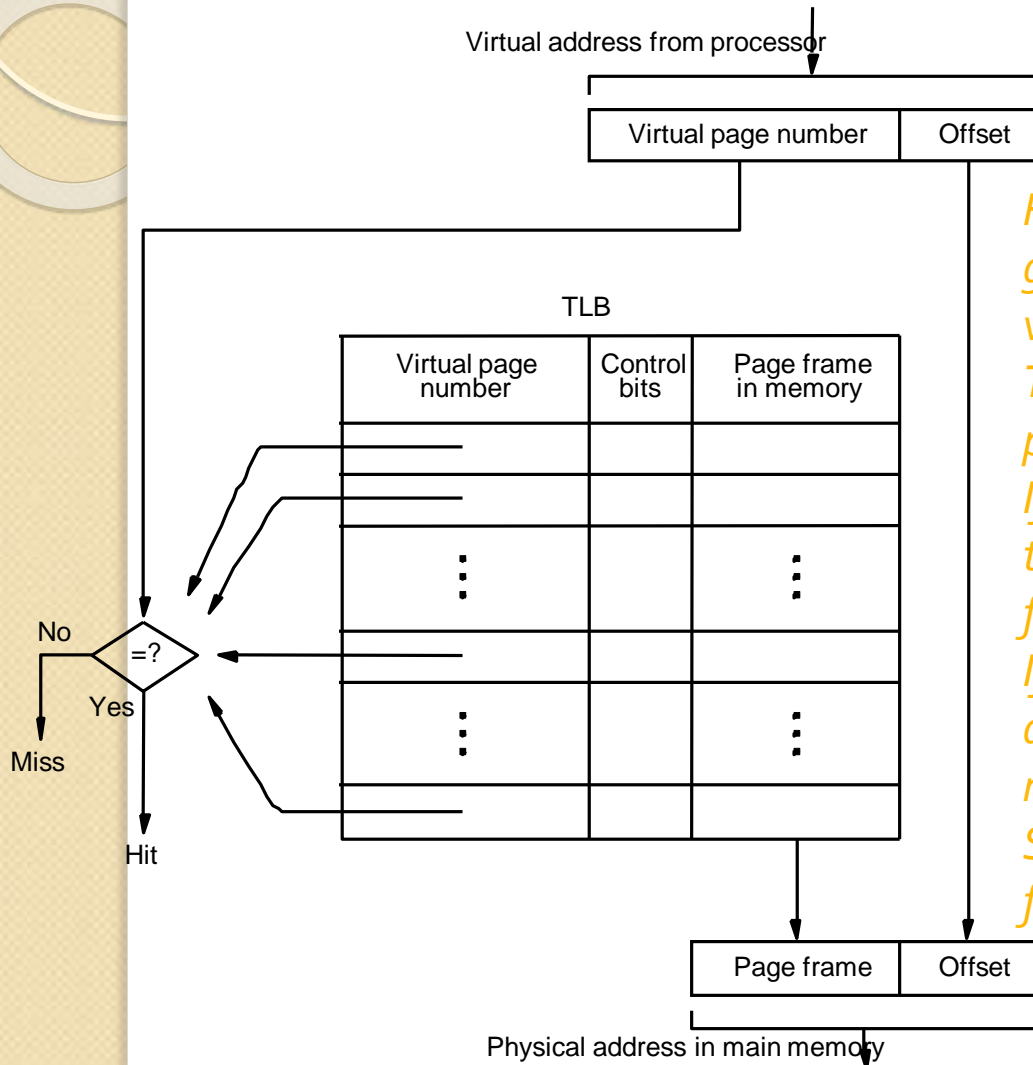


Address translation (contd..)

- A small cache called as Translation Lookaside Buffer (TLB) is included in the MMU.
 - TLB holds page table entries of the most recently accessed pages.
- Recall that cache memory holds most recently accessed blocks from the main memory.
 - Operation of the TLB and page table in the main memory is similar to the operation of the cache and main memory.
- Page table entry for a page includes:
 - Address of the page frame where the page resides in the main memory.
 - Some control bits.
- In addition to the above for each page, TLB must hold the virtual page number for each page.



Address translation (contd..)



Associative-mapped TLB

High-order bits of the virtual address generated by the processor select the virtual page.

These bits are compared to the virtual page numbers in the TLB.

If there is a match, a hit occurs and the corresponding address of the page frame is read.

If there is no match, a miss occurs and the page table within the main memory must be consulted.

Set-associative mapped TLBs are found in commercial processors.

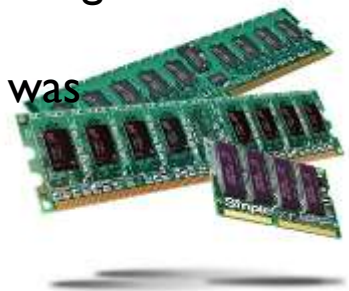
Address translation (contd..)

- How to keep the entries of the TLB coherent with the contents of the page table in the main memory?
- Operating system may change the contents of the page table in the main memory.
 - Simultaneously it must also invalidate the corresponding entries in the TLB.
- A control bit is provided in the TLB to invalidate an entry.
- If an entry is invalidated, then the TLB gets the information for that entry from the page table.
 - Follows the same process that it would follow if the entry is not found in the TLB or if a “miss” occurs.



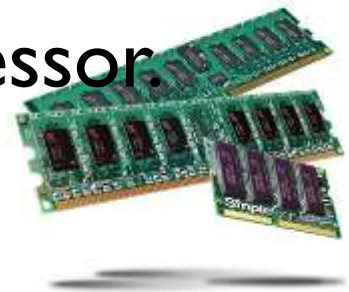
Address translation (contd..)

- What happens if a program generates an access to a page that is not in the main memory?
- In this case, a page fault is said to occur.
 - Whole page must be brought into the main memory from the disk, before the execution can proceed.
- Upon detecting a page fault by the MMU, following actions occur:
 - MMU asks the operating system to intervene by raising an exception.
 - Processing of the active task which caused the page fault is interrupted.
 - Control is transferred to the operating system.
 - Operating system copies the requested page from secondary storage to the main memory.
 - Once the page is copied, control is returned to the task which was interrupted.



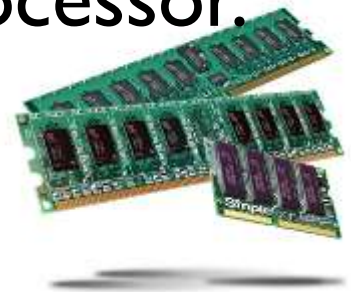
Address translation (contd..)

- Servicing of a page fault requires transferring the requested page from secondary storage to the main memory.
- This transfer may incur a long delay.
- While the page is being transferred the operating system may:
 - Suspend the execution of the task that caused the page fault.
 - Begin execution of another task whose pages are in the main memory.
- Enables efficient use of the processor.



Address translation (contd..)

- How to ensure that the interrupted task can continue correctly when it resumes execution?
- There are two possibilities:
 - Execution of the interrupted task must continue from the point where it was interrupted.
 - The instruction must be restarted.
- Which specific option is followed depends on the design of the processor.



Address translation (contd..)

- When a new page is to be brought into the main memory from secondary storage, the main memory may be full.
 - Some page from the main memory must be replaced with this new page.
- How to choose which page to replace?
 - This is similar to the replacement that occurs when the cache is full.
 - The principle of locality of reference (?) can also be applied here.
 - A replacement strategy similar to LRU can be applied.
- Since the size of the main memory is relatively larger compared to cache, a relatively large amount of programs and data can be held in the main memory.
 - Minimizes the frequency of transfers between secondary storage and main memory.



Address translation (contd..)

- A page may be modified during its residency in the main memory.
- When should the page be written back to the secondary storage?
- Recall that we encountered a similar problem in the context of cache and main memory:
 - Write-through protocol(?)
 - Write-back protocol(?)
- **Write-through protocol cannot be used, since it will incur a long delay each time a small amount of data is written to the disk.**

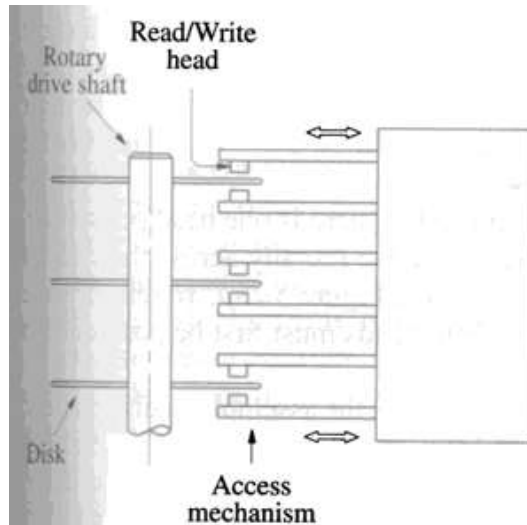




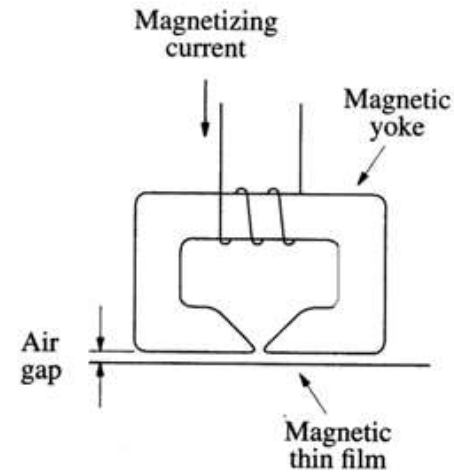
The Memory System

Secondary Storage

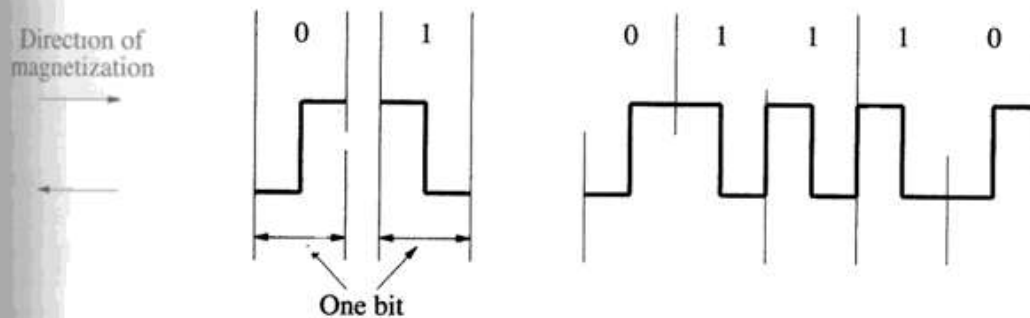
Magnetic Hard Disks



(a) Mechanical structure



(b) Read/Write head detail



(c) Bit representation by phase encoding

Disk

Disk drive

Disk controller



Organization of Data on a Disk

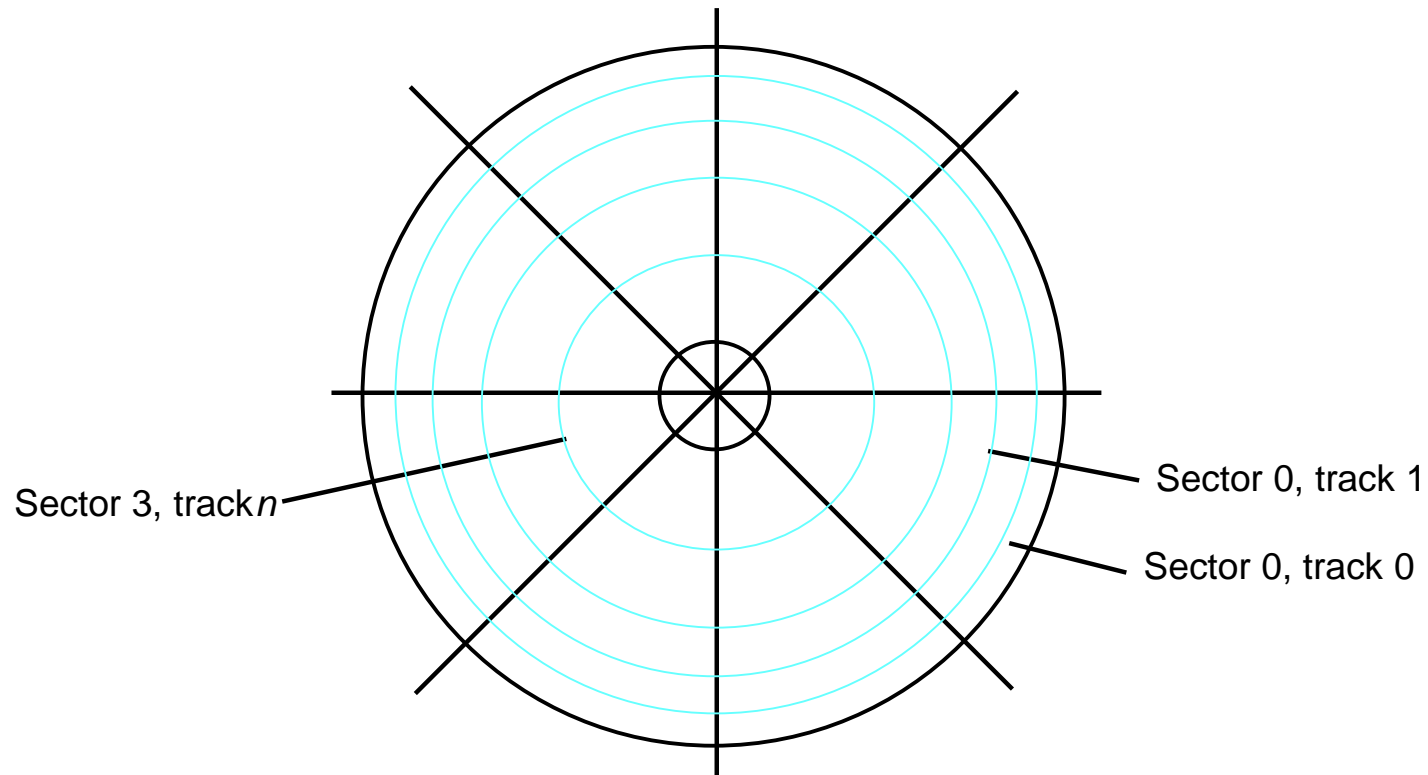


Figure 5.30. Organization of one surface of a disk.



Access Data on a Disk

- Sector header
- Following the data, there is an error-correction code (ECC).
- Formatting process
- Difference between inner tracks and outer tracks
- Access time – seek time / rotational delay (latency time)
- Data buffer/cache



Disk Controller

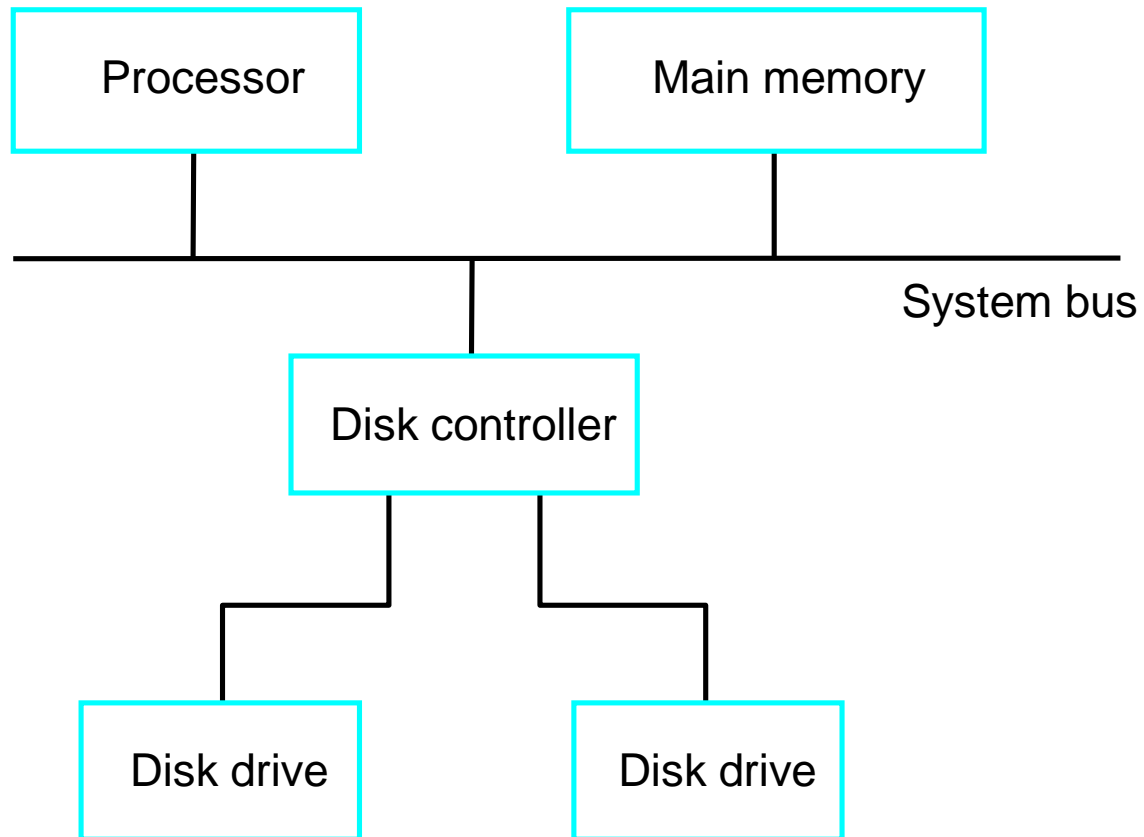
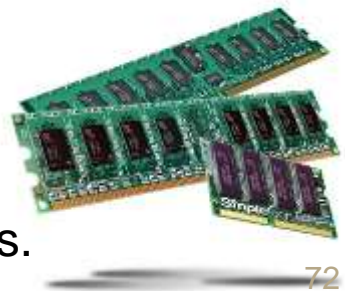


Figure 5.31. Disks connected to the system bus.



Disk Controller

- Seek
- Read
- Write
- Error checking

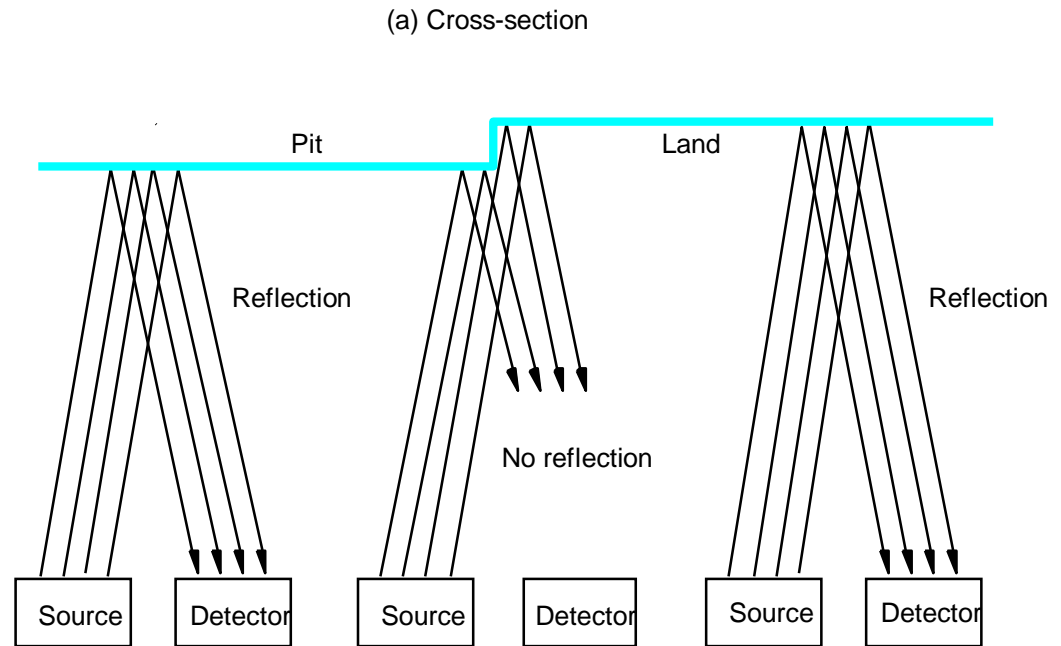


RAID Disk Arrays

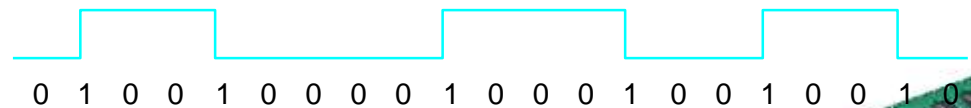
- Redundant Array of Inexpensive Disks
- Using multiple disks makes it cheaper for huge storage, and also possible to improve the reliability of the overall system.
- RAID0 – data striping
- RAID1 – identical copies of data on two disks
- RAID2, 3, 4 – increased reliability
- RAID5 – parity-based error-recovery



Optical Disks

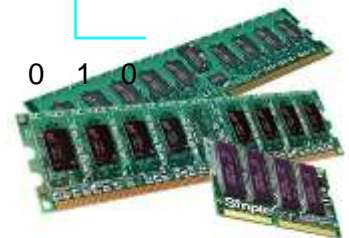


(b) Transition from pit to land



(c) Stored binary pattern

Figure 5.32. Optical disk.



Optical Disks

- CD-ROM
- CD-Recordable (CD-R)
- CD-ReWritable (CD-RW)
- DVD
- DVD-RAM



Magnetic Tape Systems

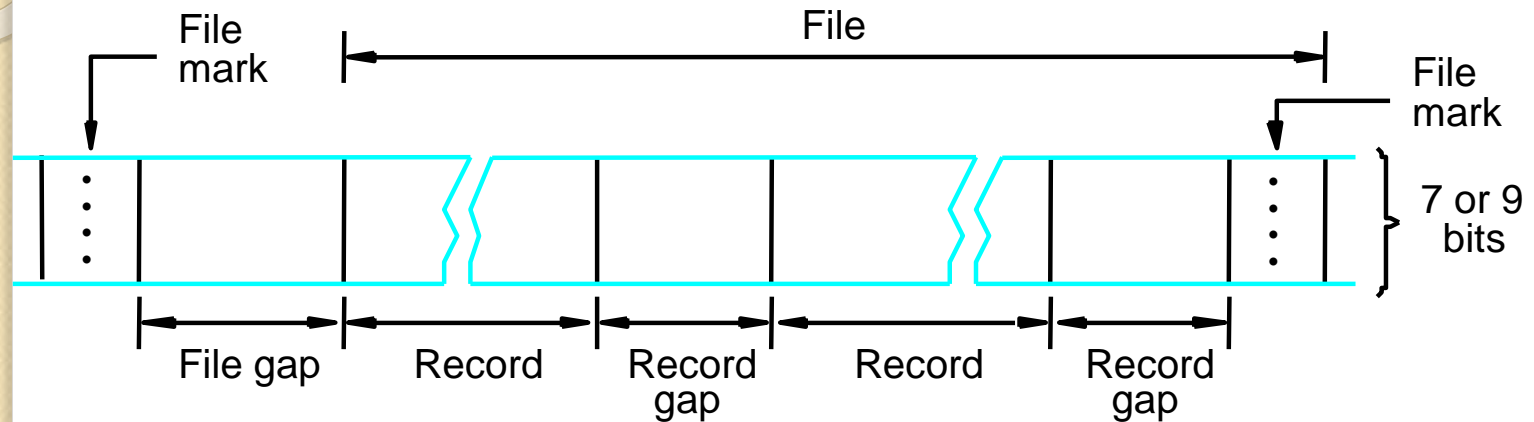


Figure 5.33. Organization of data on magnetic tape.



Model Questions

1. Explain the internal organization of a 16 megabit DRAM chip, configured as $2M \times 8$ cells. Also explain as at how can be made to work in fast page mode.
2. With a block diagram, explain the direct and set associative mapping between cache and main memory.
3. Describe the principles of magnetic disk.
4. What is virtual memory? With a diagram, explain how virtual memory address is translated.
5. Draw for $1K \times 1$ memory chip with neat figure.
6. Analyze with diagram the memory hierarchy with respect to speed, size and cost.
7. Briefly explain any four non - volatile memory concepts.
8. Discuss the internal organization of a $2M \times 8$ asynchronous DRAM chip.
9. Describe the different mapping functions in cache.
10. Define: i) memory latency ii) memory bandwidth iii) hit rate iv) miss penalty
11. Explain any one feature of memory design that leads to improved performance of computer.



Module 4.

Arithmetic

Outline

- Numbers - Arithmetic Operations and Characters
- Addition and Subtraction of Signed Numbers
- Design of Fast Adders
- Multiplication of Positive Numbers
- Signed Operand Multiplication
- Fast Multiplication
- Integer Division
- Floating-point Numbers and Operations

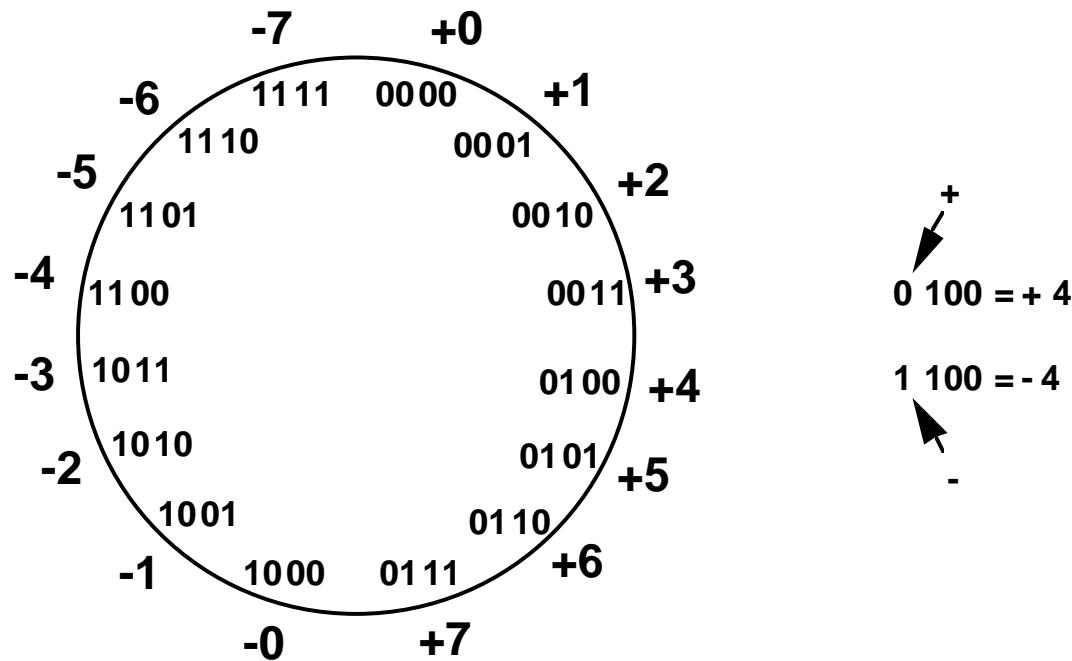
Learning Objectives

1. To understand number representation and addition/subtraction in the 2s complement form.
2. To analyze the booth algorithm used to determine how multiplicand summands are selected by the multiplier bit patterns.
3. To analyze high speed adders using carry lookahead logic to generate carry signal in parallel.
4. To gain extensive knowledge in representing floating point numbers in the IEEE standard format and perform basic arithmetic operation.

Number, Arithmetic Operations, and Characters

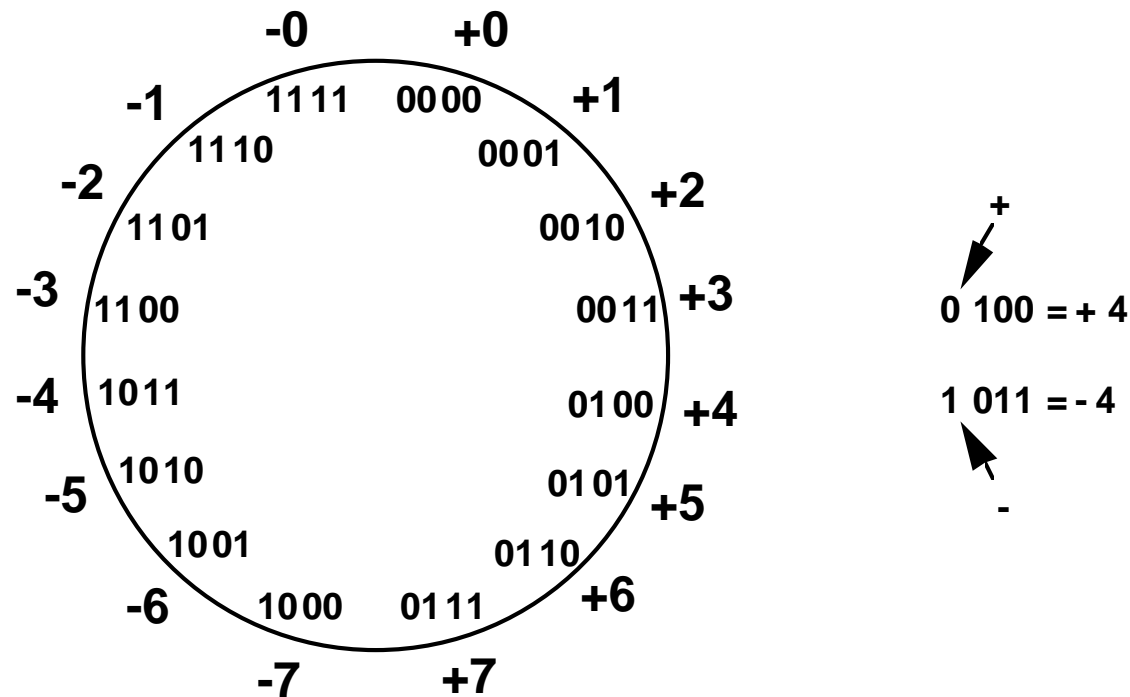
- Signed Integer
- 3 major representations:
 - Sign and magnitude
 - One's complement
 - Two's complement
- Assumptions:
 - 4-bit machine word
 - 16 different values can be represented
 - Roughly half are positive, half are negative

Sign and Magnitude Representation



High order bit is sign: 0 = positive (or zero), 1 = negative
Three low order bits is the magnitude: 0 (000) thru 7 (111)
Number range for n bits = $\pm 2^{n-1} - 1$
Two representations for 0

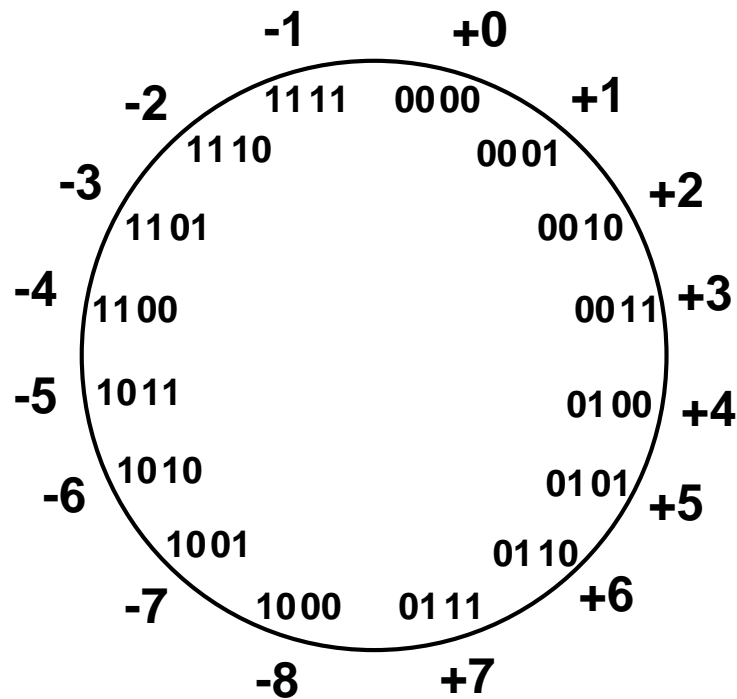
One's Complement Representation



- Subtraction implemented by addition & 1's complement
- Still two representations of 0! This causes some problems
- Some complexities in addition

Two's Complement Representation

*like 1's comp
except shifted
one position
clockwise*



$0\ 100 = +4$
 $1\ 100 = -4$

- Only one representation for 0
- One more negative number than positive number

Binary, Signed-Integer Representations

B	Values represented		
	$b_3 b_2 b_1 b_0$	Sign and magnitude	1's complement 2's complement
	0 1 1 1	+ 7	+ 7 + 7
	0 1 1 0	+ 6	+ 6 + 6
	0 1 0 1	+ 5	+ 5 + 5
	0 1 0 0	+ 4	+ 4 + 4
	0 0 1 1	+ 3	+ 3 + 3
	0 0 1 0	+ 2	+ 2 + 2
	0 0 0 1	+ 1	+ 1 + 1
	0 0 0 0	+ 0	+ 0 + 0
	1 0 0 0	- 0	- 7 - 8
	1 0 0 1	- 1	- 6 - 7
	1 0 1 0	- 2	- 5 - 6
	1 0 1 1	- 3	- 4 - 5
	1 1 0 0	- 4	- 3 - 4
	1 1 0 1	- 5	- 2 - 3
	1 1 1 0	- 6	- 1 - 2
	1 1 1 1	- 7	- 0 - 1

Figure 2.1. Binary, signed-integer representations.

Addition and Subtraction – 2's Complement

If carry-in to the high order bit = carry-out then ignore carry

if carry-in differs from carry-out then overflow

$$\begin{array}{r}
 4 \quad 0100 \\
 + 3 \quad 0011 \\
 \hline
 7 \quad 0111
 \end{array}$$

$$\begin{array}{r}
 -4 \quad 1100 \\
 + (-3) \quad 1101 \\
 \hline
 -7 \quad 11001
 \end{array}$$

$$\begin{array}{r}
 4 \quad 0100 \\
 - 3 \quad 1101 \\
 \hline
 1 \quad 10001
 \end{array}$$

$$\begin{array}{r}
 -4 \quad 1100 \\
 + 3 \quad 0011 \\
 \hline
 -1 \quad 1111
 \end{array}$$

Simpler addition scheme makes twos complement the most common choice for integer number systems within digital systems

Addition/subtraction of signed numbers

x_i	y_i	Carry-in c_i	Sums s_i	Carry-out c_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

At the i^{th} stage:

Input:

c_i is the carry-in

Output:

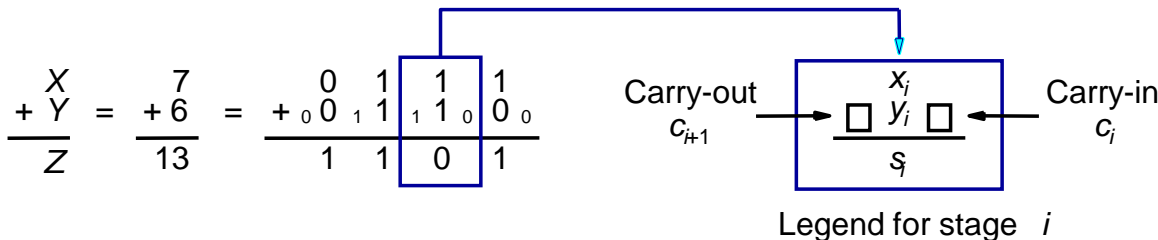
s_i is the sum

c_{i+1} carry-out to $(i+1)^{st}$ state

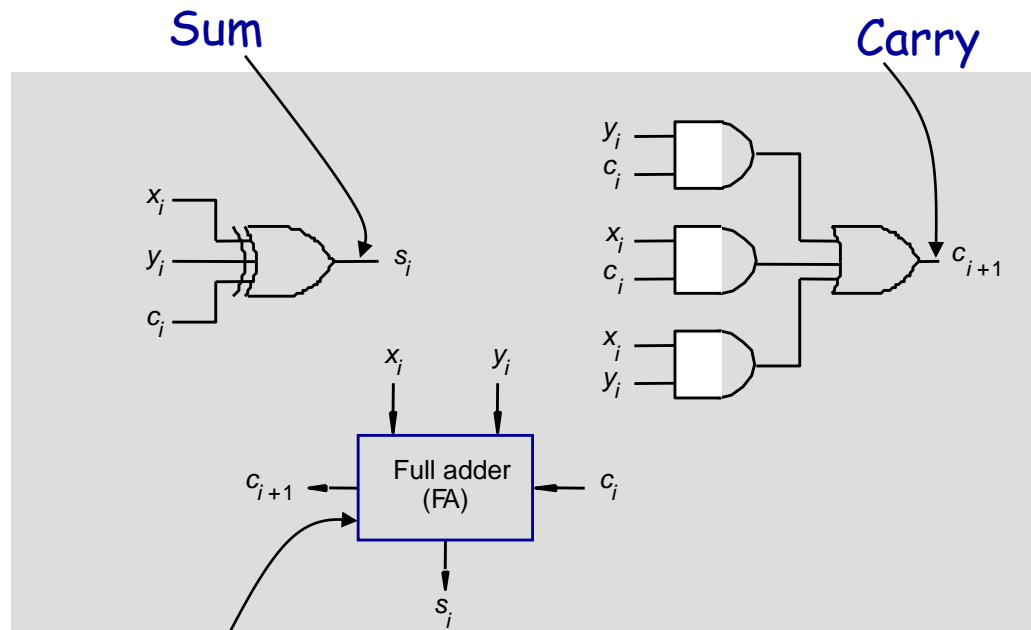
$$s_i = \bar{x}_i \bar{y}_i c_i + \bar{x}_i y_i \bar{c}_i + x_i \bar{y}_i \bar{c}_i + x_i y_i c_i = x_i \oplus y_i \oplus c_i$$

$$c_{i+1} = y_i c_i + x_i c_i + x_i y_i$$

Example:



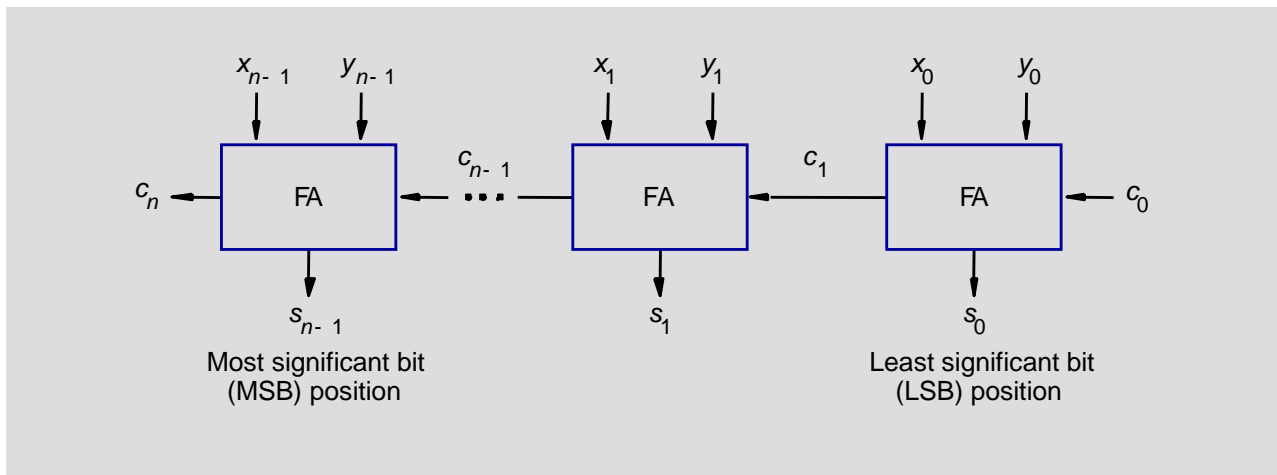
Addition logic for a single stage



Full Adder (FA): Symbol for the complete circuit for a single stage of addition.

n -bit adder

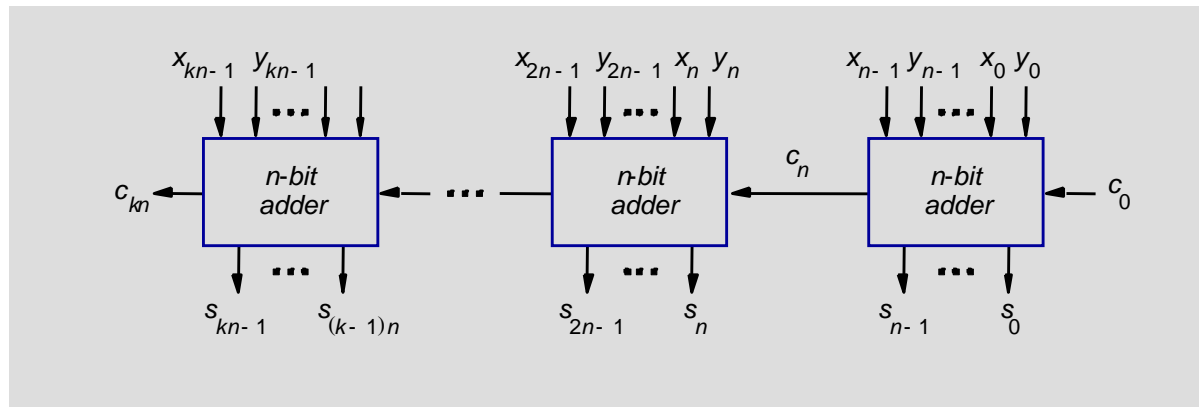
- Cascade n full adder (FA) blocks to form a n -bit adder.
- Carries propagate or ripple through this cascade, n -bit ripple carry adder.



Carry-in c_0 into the LSB position provides a convenient way to perform subtraction.

K n -bit adder

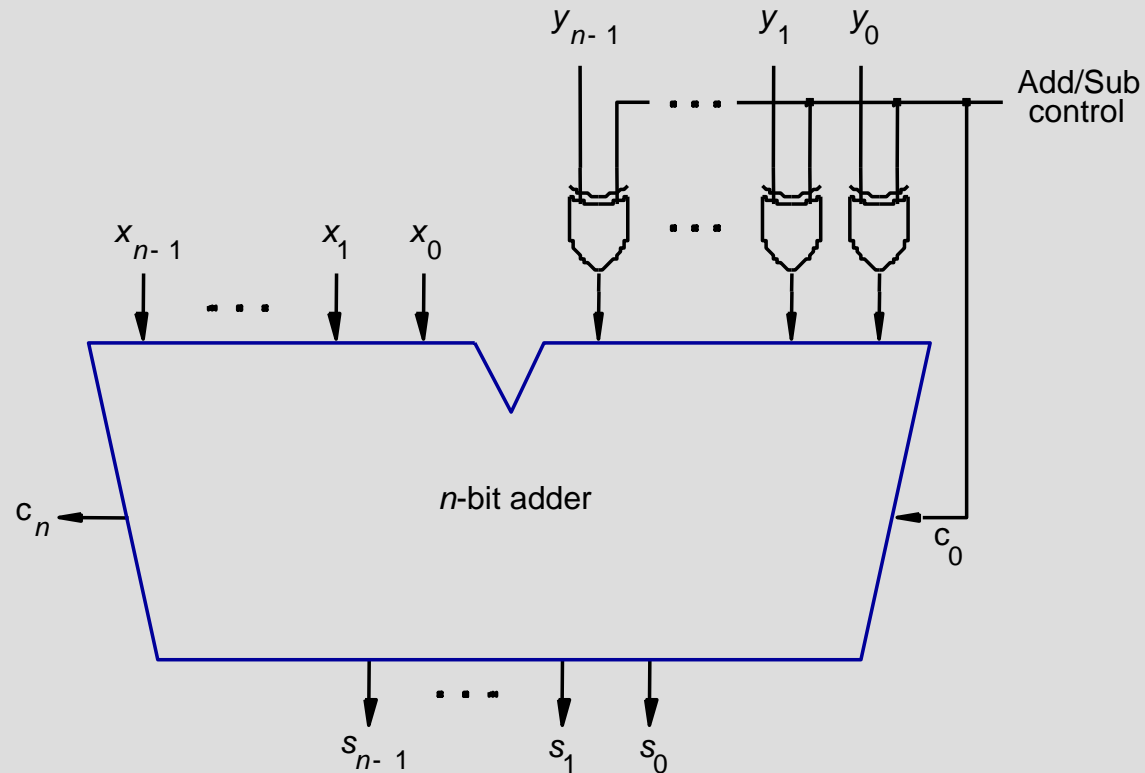
K n -bit numbers can be added by cascading k n -bit adders.



Each n -bit adder forms a block, so this is cascading of blocks.

Carries ripple or propagate through blocks, [Blocked Ripple Carry Adder](#)

n -bit adder/subtractor (contd..)



- Add/sub control = 0, addition.
- Add/sub control = 1, subtraction.

Fast addition

Recall the equations:

$$s_i = x_i \oplus y_i \oplus c_i$$

$$c_{i+1} = x_i y_i + x_i c_i + y_i c_i$$

Second equation can be written as:

$$c_{i+1} = x_i y_i + (x_i + y_i) c_i$$

We can write:

$$c_{i+1} = G_i + P_i c_i$$

$$\text{where } G_i = x_i y_i \text{ and } P_i = x_i + y_i$$

- G_i is called generate function and P_i is called propagate function
- G_i and P_i are computed only from x_i and y_i and not c_i , thus they can be computed in one gate delay after X and Y are applied to the inputs of an n -bit adder.

Carry lookahead

$$c_{i+1} = G_i + P_i c_i$$

$$c_i = G_{i-1} + P_{i-1} c_{i-1}$$

$$\Rightarrow c_{i+1} = G_i + P_i (G_{i-1} + P_{i-1} c_{i-1})$$

continuing

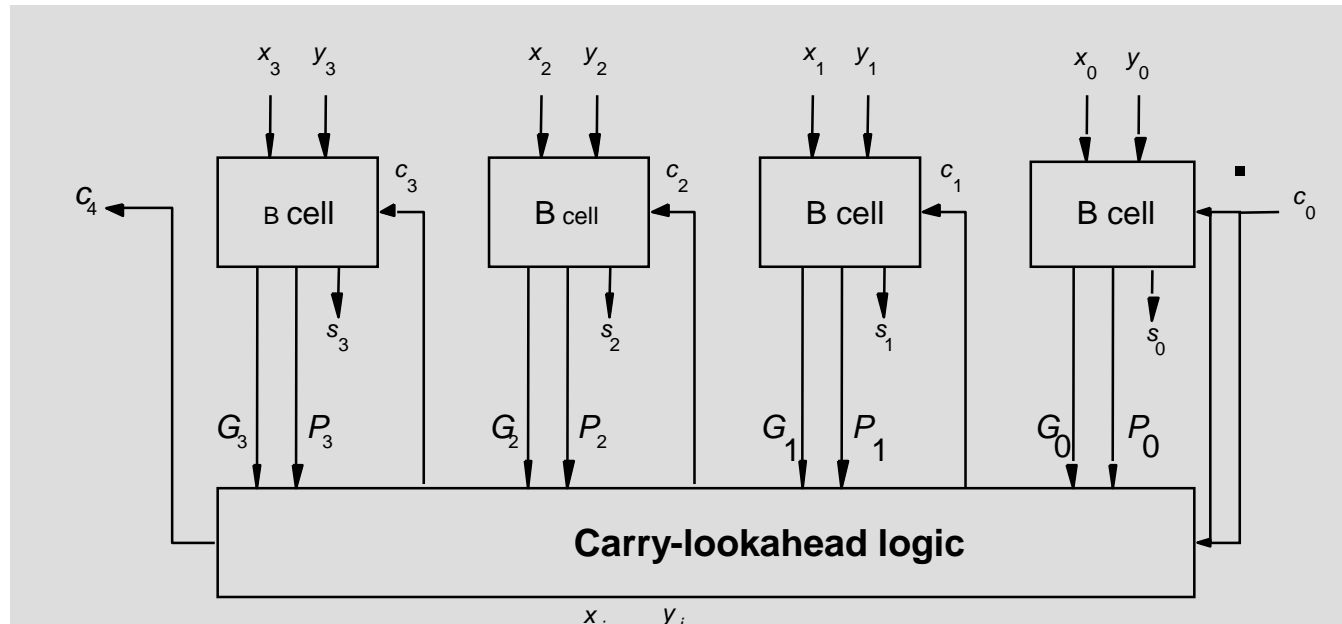
$$\Rightarrow c_{i+1} = G_i + P_i (G_{i-1} + P_{i-1} (G_{i-2} + P_{i-2} c_{i-2}))$$

until

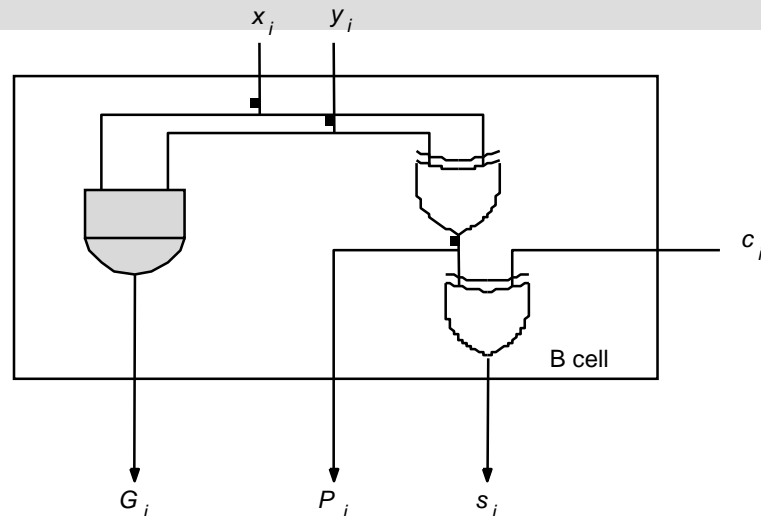
$$c_{i+1} = G_i + P_i G_{i-1} + P_i P_{i-1} G_{i-2} + \dots + P_i P_{i-1} \dots P_1 G_0 + P_i P_{i-1} \dots P_0 c_0$$

- All carries can be obtained 3 gate delays after X , Y and c_0 are applied.
 - One gate delay for P_i and G_i
 - Two gate delays in the AND-OR circuit for c_{i+1}
- All sums can be obtained 1 gate delay after the carries are computed.
- Independent of n , n -bit addition requires only 4 gate delays.
- This is called Carry Lookahead adder.

Carry-lookahead adder



**4-bit
carry-lookahead
adder**



B-cell for a single stage

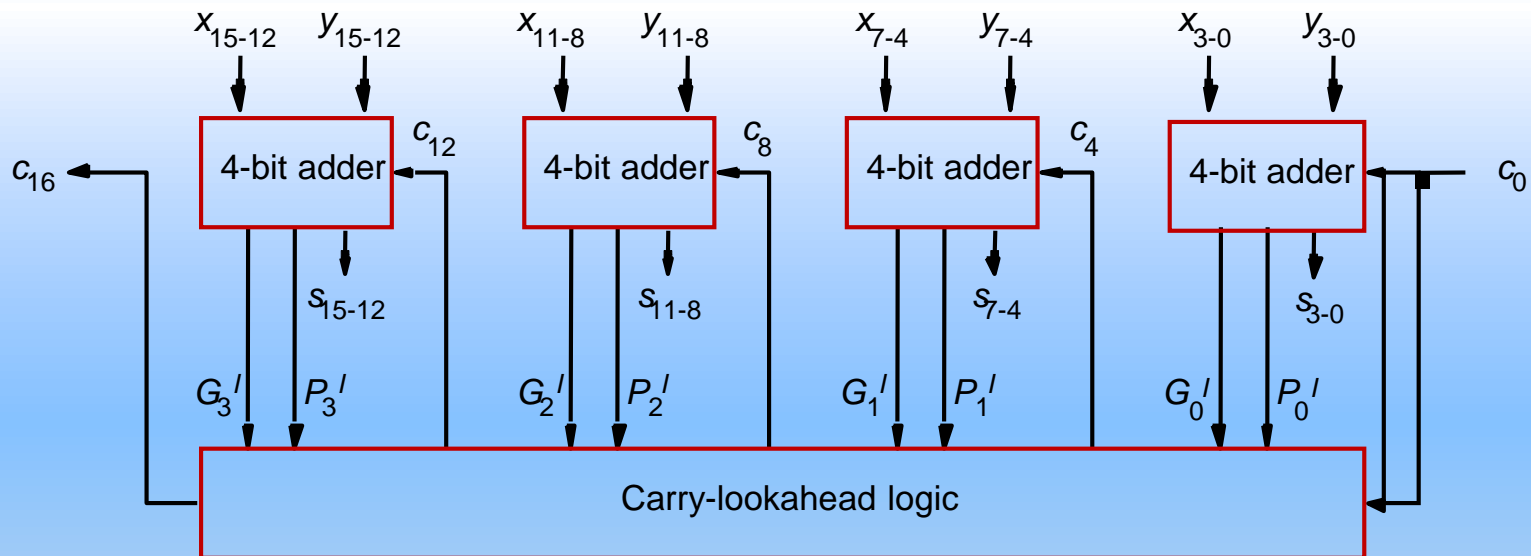
Carry lookahead adder (contd..)

- Performing n -bit addition in 4 gate delays independent of n is good only theoretically because of fan-in constraints.

$$c_{i+1} = G_i + P_i G_{i-1} + P_i P_{i-1} G_{i-2} + \dots + P_i P_{i-1} \dots P_1 G_0 + P_i P_{i-1} \dots P_0 c_0$$

- Last AND gate and OR gate require a fan-in of $(n+1)$ for a n -bit adder.
 - For a 4-bit adder ($n=4$) fan-in of 5 is required.
 - Practical limit for most gates.
- In order to add operands longer than 4 bits, we can cascade 4-bit Carry-Lookahead adders. Cascade of Carry-Lookahead adders is called Blocked Carry-Lookahead adder.

16 bit Carry-Lookahead adder



After x_i, y_i and c_0 are applied as inputs:

- G_i and P_i for each stage are available after 1 gate delay.
- P^I is available after 2 and G^I after 3 gate delays.
- All carries are available after 5 gate delays.
- c_{16} is available after 5 gate delays.
- s_{15} which depends on c_{12} is available after 8 (5+3) gate delays (Recall that for a 4-bit carry lookahead adder, the last sum bit is available 3 gate delays after all inputs are available)

Multiplication

Multiplication of unsigned numbers

$$\begin{array}{r} 1 1 0 1 \quad (13) \text{ Multiplicand M} \\ 1 0 1 1 \quad (11) \text{ Multiplier Q} \\ \hline 1 1 0 1 \\ 1 1 0 1 \\ 0 0 0 0 \\ 1 1 0 1 \\ \hline 1 0 0 0 1 1 1 1 \quad (143) \text{ Product P} \end{array}$$

Product of 2 n -bit numbers is at most a $2n$ -bit number.

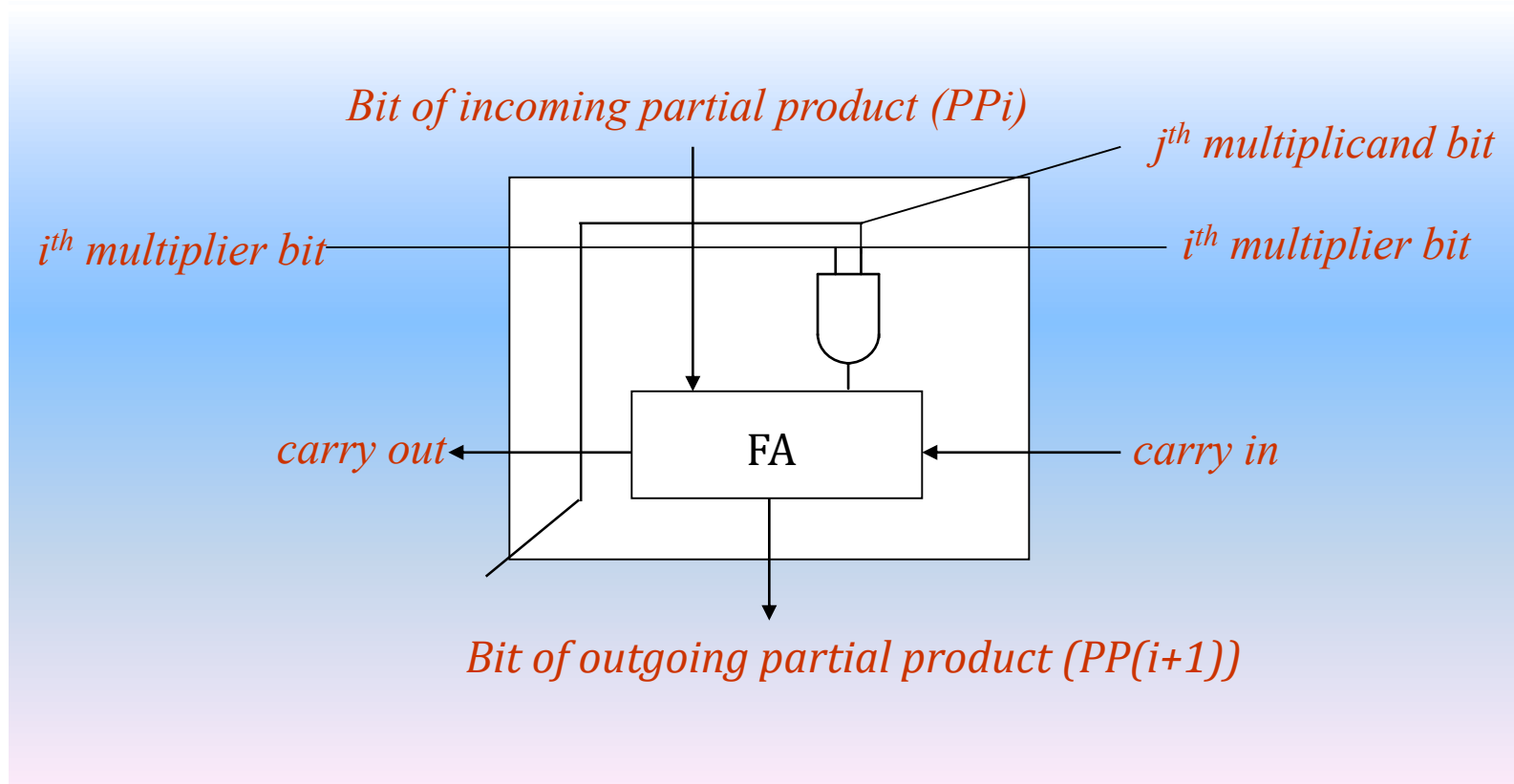
Unsigned multiplication can be viewed as addition of shifted versions of the multiplicand.

Multiplication of unsigned numbers (contd..)

- We added the partial products at end.
 - Alternative would be to add the partial products at each stage.
- Rules to implement multiplication are:
 - If the i^{th} bit of the multiplier is 1, shift the multiplicand and add the shifted multiplicand to the current value of the partial product.
 - Hand over the partial product to the next stage
 - Value of the partial product at the start stage is 0.

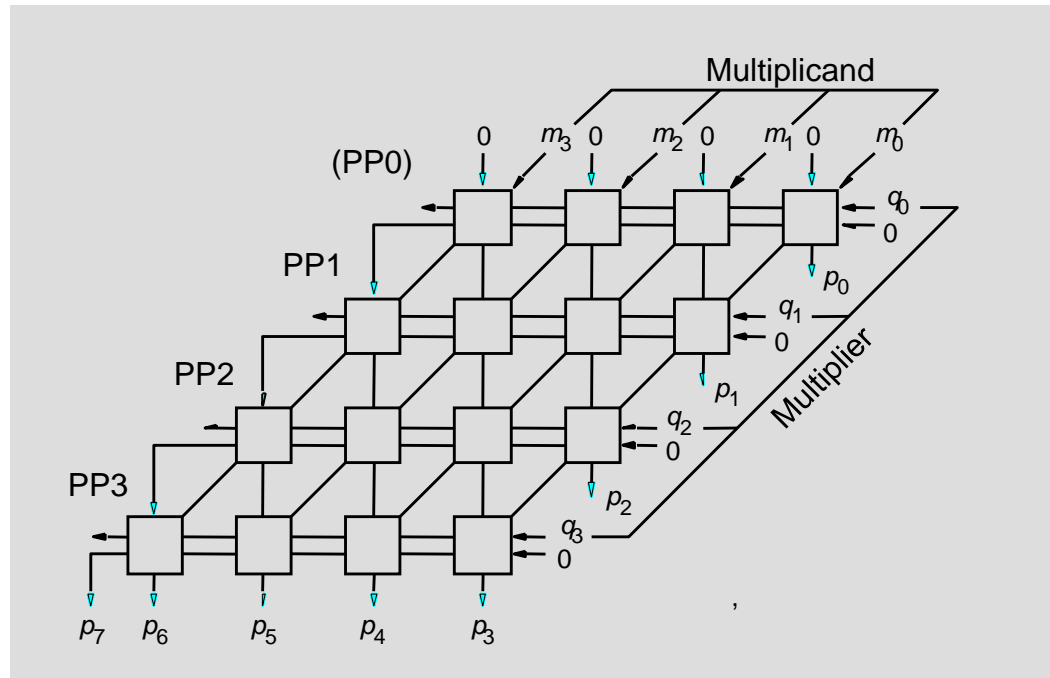
Multiplication of unsigned numbers

Typical multiplication cell



Combinatorial array multiplier

Combinatorial array multiplier



Product is: p_7, p_6, \dots, p_0

Multiplicand is shifted by displacing it through an array of adders.

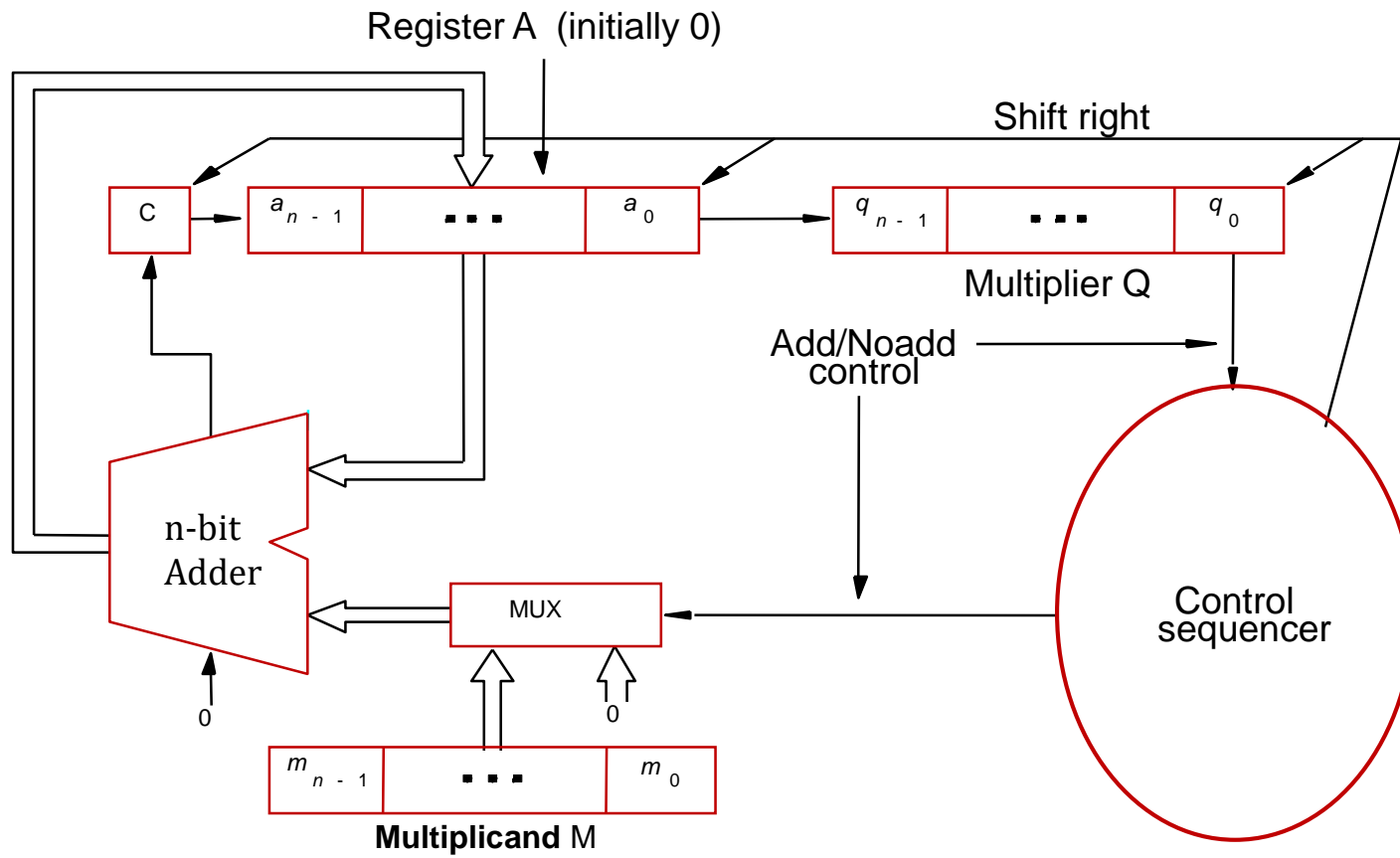
Combinatorial array multiplier (contd..)

- Combinatorial array multipliers are:
 - Extremely inefficient.
 - Have a high gate count for multiplying numbers of practical size such as 32-bit or 64-bit numbers.
 - Perform only one function, namely, unsigned integer product.
- Improve gate efficiency by using a mixture of combinatorial array techniques and sequential techniques requiring less combinational logic.

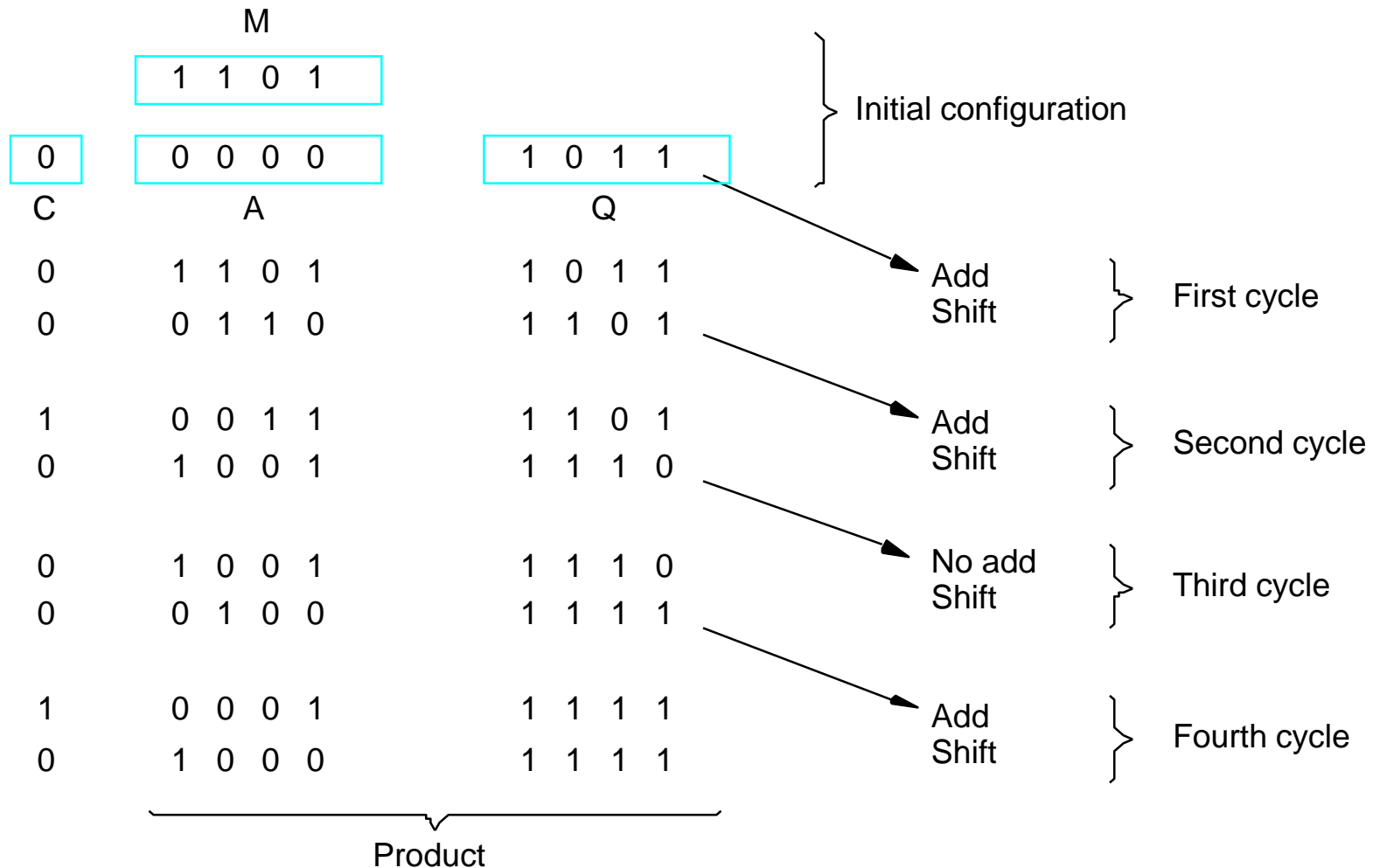
Sequential multiplication

- Recall the rule for generating partial products:
 - If the i th bit of the multiplier is 1, add the appropriately shifted multiplicand to the current partial product.
 - Multiplicand has been shifted left when added to the partial product.
- However, adding a left-shifted multiplicand to an unshifted partial product is equivalent to adding an unshifted multiplicand to a right-shifted partial product.

Sequential Circuit Multiplier



Sequential multiplication (contd..)



Signed Multiplication

Signed Multiplication

- Considering 2's-complement signed operands, what will happen to $(-13) \times (+11)$ if following the same method of unsigned multiplication?

Sign extension is shown in blue

						1	0	0	1	1	(- 13)
						0	1	0	1	1	(+11)
						<hr/>					
	1	1	1	1	1	1	0	0	1	1	
	1	1	1	1	1	0	0	1	1		
	0	0	0	0	0	0	0	0			
	1	1	1	0	0	1	1				
	0	0	0	0	0	0					
	<hr/>										
	1	1	0	1	1	1	0	0	0	1	(- 143)

Sign extension of negative multiplicand.

Signed Multiplication

- For a negative multiplier, a straightforward solution is to form the 2's-complement of both the multiplier and the multiplicand and proceed as in the case of a positive multiplier.
- This is possible because complementation of both operands does not change the value or the sign of the product.
- A technique that works equally well for both negative and positive multipliers – Booth algorithm.

Booth Algorithm

- Consider in a multiplication, the multiplier is positive 0011110, how many appropriately shifted versions of the multiplicand are added in a standard procedure?

$$\begin{array}{r}
 \begin{array}{cccccccc}
 & & & & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\
 & & & & 0 & 0 & +1 & +1 & +1 & +1 & 0 \\
 \hline
 & & & & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & & & 0 & 1 & 0 & 1 & 1 & 0 & 1 & \\
 & & 0 & 1 & 0 & 1 & 1 & 0 & 1 & & \\
 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & & & \\
 & & 0 & 1 & 0 & 1 & 1 & 0 & 1 & & \\
 & & & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\
 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & & & \\
 \hline
 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0
 \end{array}
 \end{array}$$

Booth Algorithm

- Since $0011110 = 0100000 - 0000010$, if we use the expression to the right, what will happen?

								0	1	0	1	1	0	1
								0	+1	0	0	0	-1	0
								0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	0	1	0	0	1	1	← 2's complement of the multiplicand
0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0	0	0	1	0	1	1	0	1						
0	0	0	0	0	0	0	0	0						
0	0	0	1	0	1	0	1	0	0	0	1	1	0	

Booth Algorithm

- In general, in the Booth scheme, -1 times the shifted multiplicand is selected when moving from 0 to 1, and +1 times the shifted multiplicand is selected when moving from 1 to 0, as the multiplier is scanned from right to left.

0	0	1	0	1	1	0	0	1	1	1	0	1	0	1	1	0	0
									↓								
0	+1	-1	+1	0	-1	0	+1	0	0	-1	+1	-1	+1	0	-1	0	0

Booth recoding of a multiplier.

Booth Algorithm

$\begin{array}{r} 01101 \\ \times 11010 \\ \hline \end{array}$	\Rightarrow	$\begin{array}{r} 01101 \\ 0-1+1-10 \\ \hline 00000 \\ 111110011 \\ 00001101 \\ 1110011 \\ 000000 \\ \hline 1110110010 \end{array}$
$(+13)$ (-6)		(-78)

Booth multiplication with a negative multiplier.

Booth Algorithm


Multiplier		Version of multiplicand selected by bit
Bit i	Bit $i-1$	
0	0	0 X M
0	1	+ 1 X M
1	0	- 1 X M
1	1	0 X M

Booth multiplier recoding table.


Booth Algorithm

- Best case — a long string of 1's (skipping over 1s)
- Worst case — 0's and 1's are alternating


Worst-case multiplier

0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
															
+1	-1	+1	-1	+1	-1	+1	-1	+1	-1	+1	-1	+1	-1	+1	-1

Ordinary multiplier

1	1	0	0	0	1	0	1	1	0	1	1	1	1	0	0
															
0	-1	0	0	+1	-1	+1	0	-1	+1	0	0	0	-1	0	0

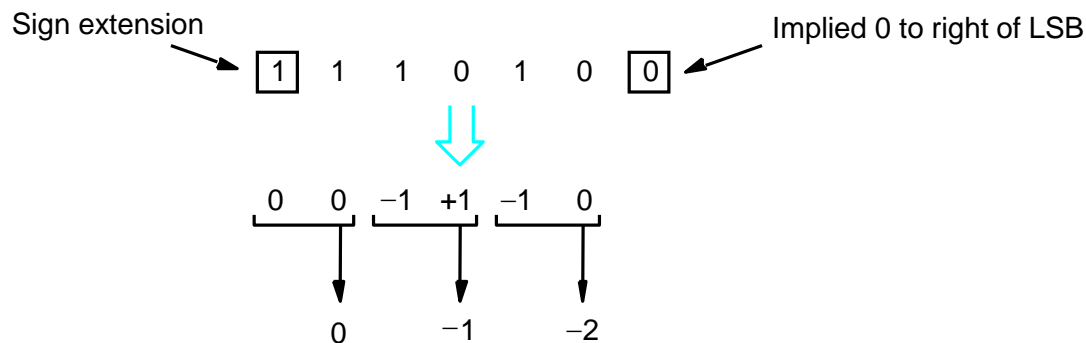
Good multiplier

0	0	0	0	1	1	1	1	1	0	0	0	0	1	1	1
															
0	0	0	+1	0	0	0	0	-1	0	0	0	+1	0	0	-1

Fast Multiplication

Bit-Pair Recoding of Multipliers

- Bit-pair recoding halves the maximum number of summands (versions of the multiplicand).



(a) Example of bit-pair recoding derived from Booth recoding

Bit-Pair Recoding of Multipliers

Multiplier bit-pair		Multiplier bit on the right $i-1$	Multiplicand selected at position i
$i+1$	i		
0	0	0	0 X M
0	0	1	+ 1 X M
0	1	0	+ 1 X M
0	1	1	+ 2 X M
1	0	0	- 2 X M
1	0	1	- 1 X M
1	1	0	- 1 X M
1	1	1	0 X M

(b) Table of multiplicand selection decisions

Bit-Pair Recoding of Multipliers

$$\begin{array}{r} 01101 \quad (+13) \\ 11010 \quad (-6) \\ \hline \end{array}$$



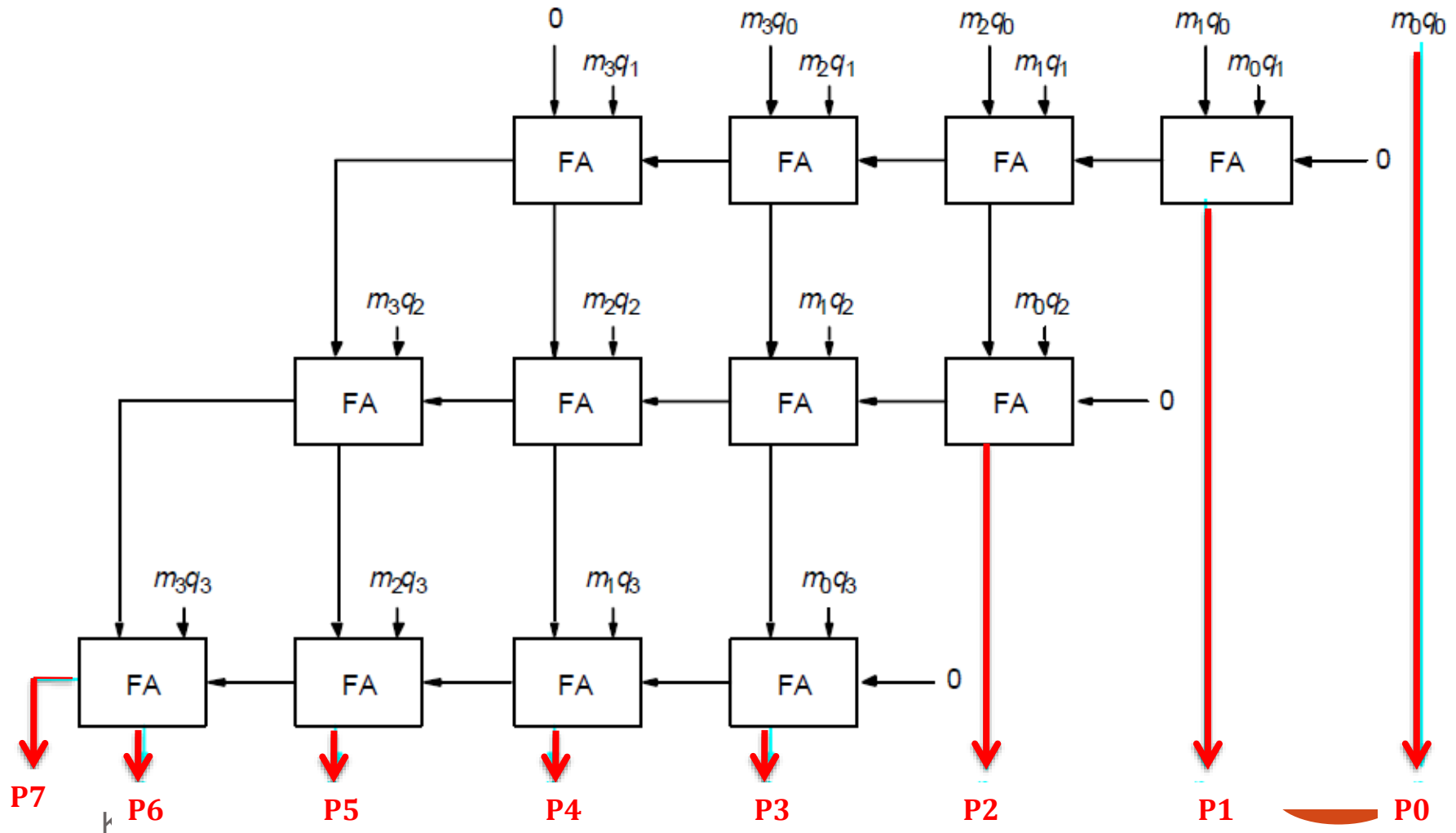
$$\begin{array}{r} 01101 \\ 0-1+1-10 \\ \hline 00000 \\ 11111 \\ 00001 \\ 11001 \\ 11001 \\ 00000 \\ \hline 1110110010 \quad (-78) \end{array}$$



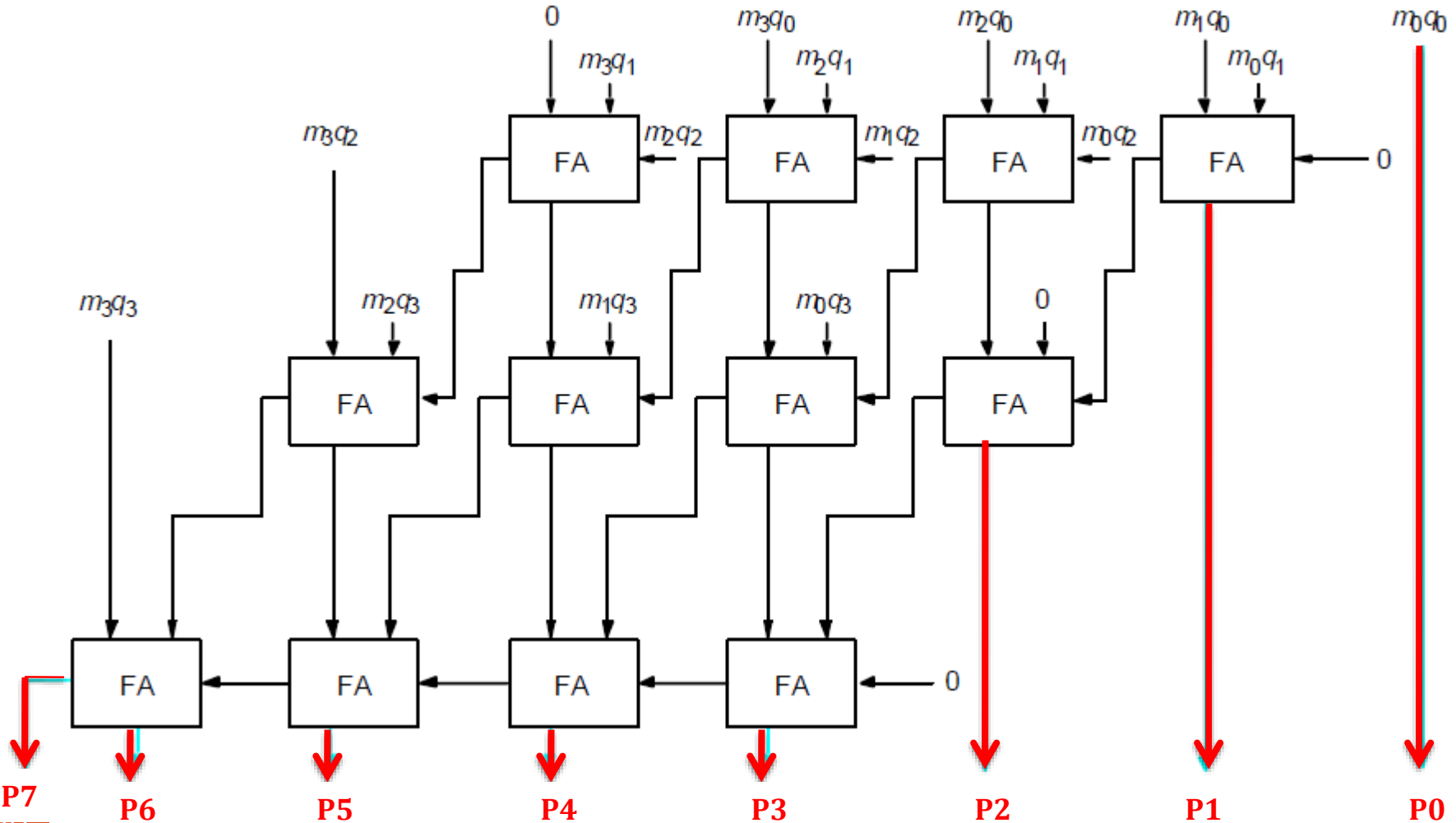
$$\begin{array}{r} 01101 \\ 0-1-2 \\ \hline 11111 \\ 11110 \\ 00000 \\ \hline 1110110010 \end{array}$$

Carry-Save Addition of Summands

- CSA speeds up the addition process.



Carry-Save Addition of Summands(Cont.,)



Carry-Save Addition of Summands(Cont.,)

- Consider the addition of many summands, we can:
 - Group the summands in threes and perform carry-save addition on each of these groups in parallel to generate a set of S and C vectors in one full-adder delay
 - Group all of the S and C vectors into threes, and perform carry-save addition on them, generating a further set of S and C vectors in one more full-adder delay
 - Continue with this process until there are only two vectors remaining
 - They can be added in a RCA or CLA to produce the desired product

Carry-Save Addition of Summands

						1	0	1	1	0	1	(45)	M
					x	1	1	1	1	1	1	(63)	Q
						1	0	1	1	0	1	A	
				1		0	1	1	0	1		B	
			1	0		1	1	0	1			C	
		1	0	1		1	0	1				D	
	1	0	1	1		0	1					E	
1	0	1	1	0		1						F	
1	0	1	1	0	0	0	0	1	0	0	1	(2,835)	Product

Figure 6.17. A multiplication example used to illustrate carry-save addition as shown in Figure 6.18.

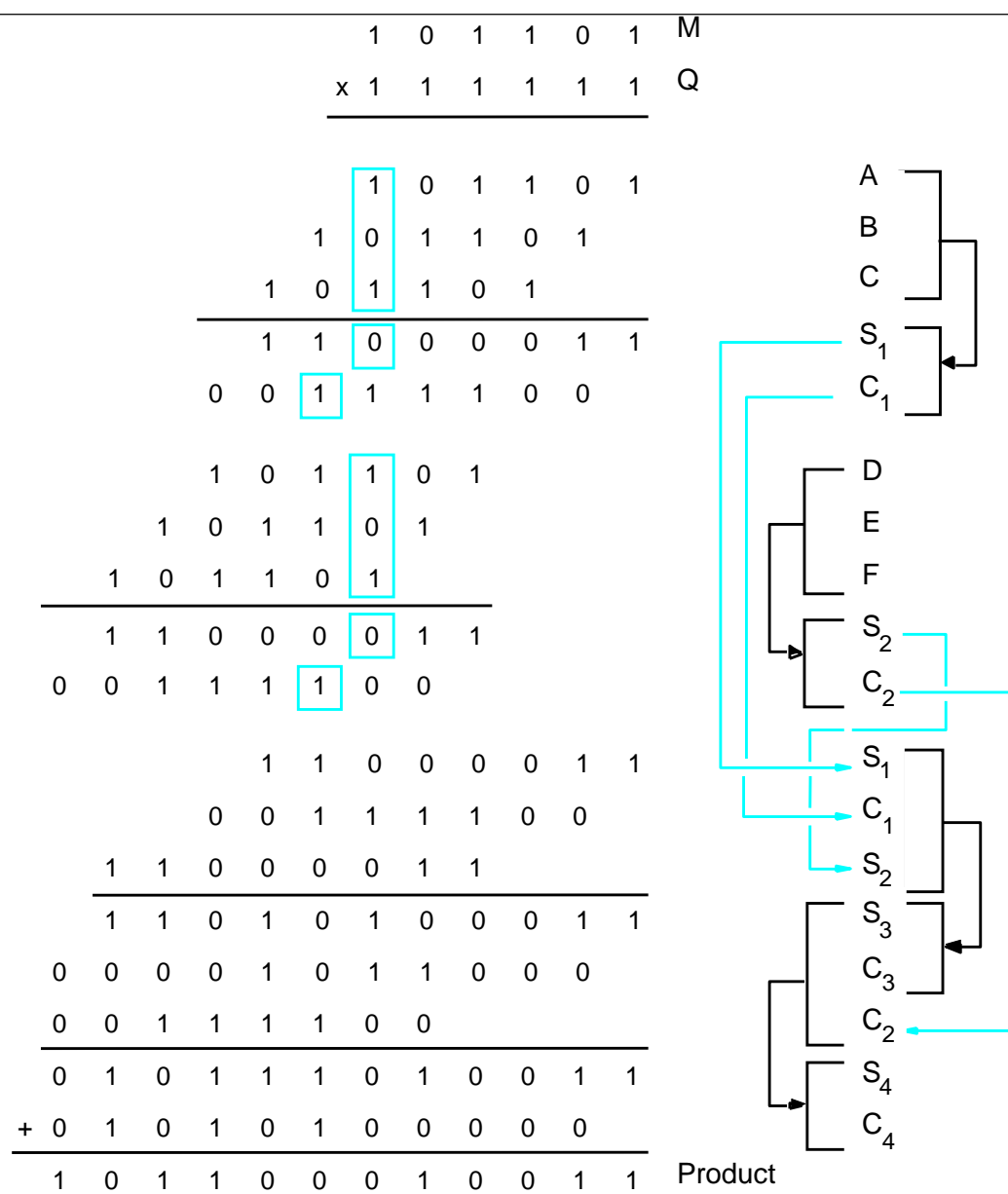


Figure 6.18. The multiplication example from Figure 6.17 performed using carry-save addition.

Integer Division

Manual Division

$$\begin{array}{r} 21 \\ 13 \overline{) 274} \\ \underline{26} \\ 14 \\ \underline{13} \\ 1 \end{array}$$

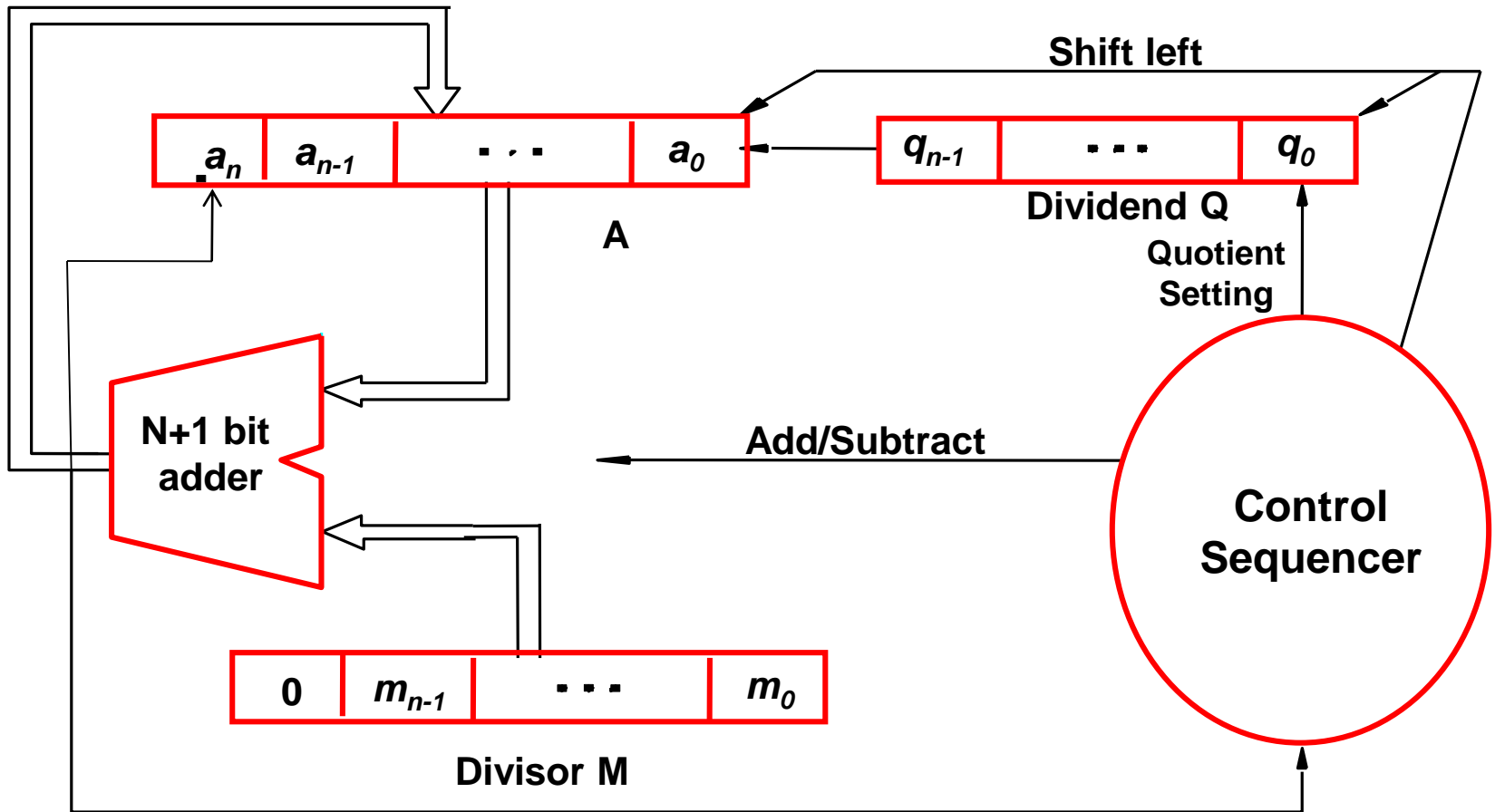
$$\begin{array}{r} 10101 \\ 1101 \overline{) 100010010} \\ \underline{1101} \\ 10000 \\ \underline{1101} \\ 1110 \\ \underline{1101} \\ 1 \end{array}$$

Longhand division examples.

Longhand Division Steps

- Position the divisor appropriately with respect to the dividend and performs a subtraction.
- If the remainder is zero or positive, a quotient bit of 1 is determined, the remainder is extended by another bit of the dividend, the divisor is repositioned, and another subtraction is performed.
- If the remainder is negative, a quotient bit of 0 is determined, the dividend is restored by adding back the divisor, and the divisor is repositioned for another subtraction.

Circuit Arrangement



Restoring Division

- Shift A and Q left one binary position
- Subtract M from A, and place the answer back in A
- If the sign of A is 1, set q_0 to 0 and add M back to A (restore A); otherwise, set q_0 to 1
- Repeat these steps n times

Examples

$$\begin{array}{r}
 10 \\
 11 \overline{) 1000} \\
 \underline{11} \\
 10
 \end{array}$$

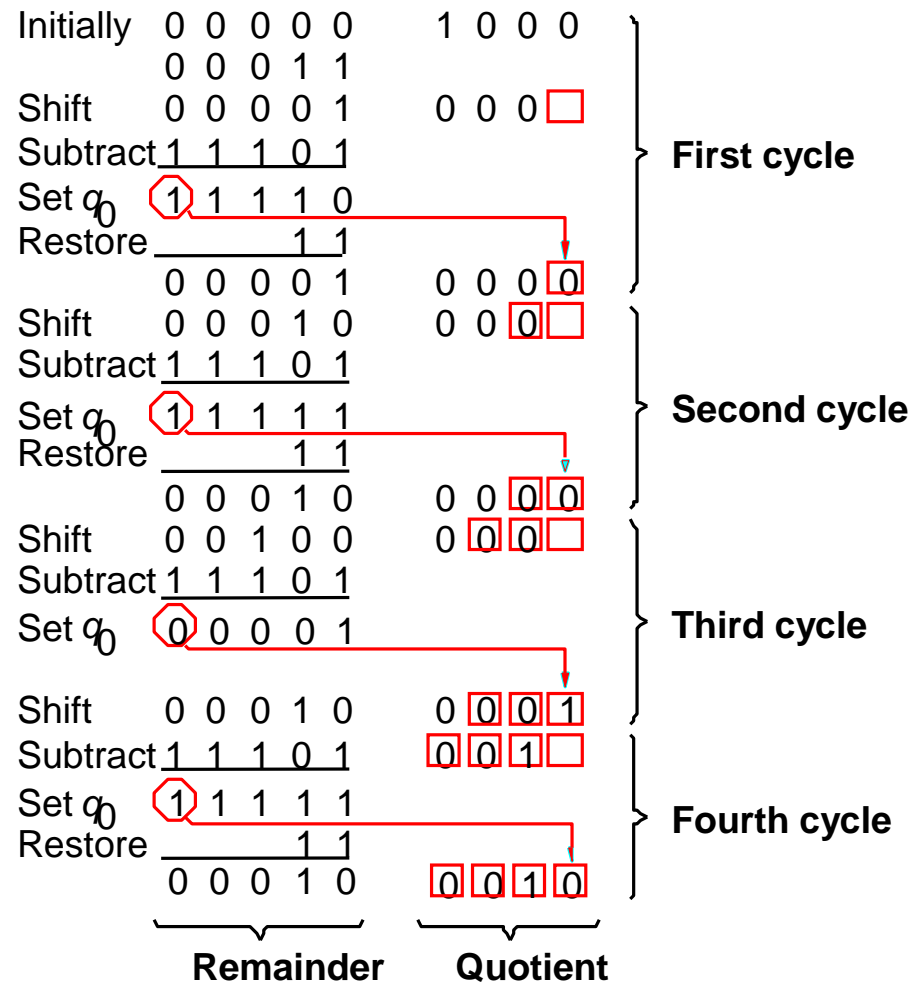


Figure 6.22. A restoring-division example.

Nonrestoring Division

- Avoid the need for restoring A after an unsuccessful subtraction.
- Any idea?
- Step 1: (Repeat n times)
 - If the sign of A is 0, shift A and Q left one bit position and subtract M from A ; otherwise, shift A and Q left and add M to A .
 - Now, if the sign of A is 0, set q_0 to 1; otherwise, set q_0 to 0.
- Step2: If the sign of A is 1, add M to A

Examples

$$\begin{array}{r}
 1\ 1\ 1\ 1\ 1 \\
 0\ 0\ 0\ 1\ 1 \\
 \hline
 \text{Add } 0\ 0\ 0\ 1\ 0 \\
 \hline
 \text{Remainder}
 \end{array}
 \quad \left. \vphantom{\begin{array}{r} 1\ 1\ 1\ 1\ 1 \\ 0\ 0\ 0\ 1\ 1 \\ \hline \text{Add } 0\ 0\ 0\ 1\ 0 \\ \hline \text{Remainder} \end{array}} \right\} \text{Restore remainder}$$

Initially	0 0 0 0 0	1 0 0 0	} First cycle
Shift	0 0 0 0 1	0 0 0 	
Subtract	1 1 1 0 1		
Set q_0	1 1 1 1 0	0 0 0 0	
Shift	1 1 1 0 0	0 0 0 	} Second cycle
Add	0 0 0 1 1		
Set q_0	1 1 1 1 1	0 0 0 0	
Shift	1 1 1 1 0	0 0 0 	} Third cycle
Add	0 0 0 1 1		
Set q_0	0 0 0 0 1	0 0 0 1	
Shift	0 0 0 1 0	0 0 1 	} Fourth cycle
Subtract	1 1 1 0 1		
Set q_0	1 1 1 1 1	0 0 1 0	

Quotient

Floating-Point Numbers

and
Operations

Fractions

If b is a binary vector, then we have seen that it can be interpreted as an unsigned integer by:

$$V(b) = b_{31} \cdot 2^{31} + b_{30} \cdot 2^{30} + b_{29} \cdot 2^{29} + \dots + b_1 \cdot 2^1 + b_0 \cdot 2^0$$

This vector has an implicit binary point to its immediate right:

$$b_{31}b_{30}b_{29}\dots\dots\dots b_1b_0 \quad \text{implicit binary point}$$

Suppose if the binary vector is interpreted with the implicit binary point is just left of the sign bit:

$$\text{implicit binary point} \quad .b_{31}b_{30}b_{29}\dots\dots\dots b_1b_0$$

The value of b is then given by:

$$V(b) = b_{31} \cdot 2^{-1} + b_{30} \cdot 2^{-2} + b_{29} \cdot 2^{-3} + \dots + b_1 \cdot 2^{-31} + b_0 \cdot 2^{-32}$$

Range of fractions

The value of the unsigned binary fraction is:

$$V(b) = b_{31} \cdot 2^{-1} + b_{30} \cdot 2^{-2} + b_{29} \cdot 2^{-3} + \dots + b_1 \cdot 2^{-31} + b_0 \cdot 2^{-32}$$

The range of the numbers represented in this format is:

$$0 \leq V(b) \leq 1 - 2^{-32} \approx 0.9999999998$$

In general for a n -bit binary fraction (a number with an assumed binary point at the immediate left of the vector), then the range of values is:

$$0 \leq V(b) \leq 1 - 2^{-n}$$

Scientific notation

- Previous representations have a fixed point. Either the point is to the immediate right or it is to the immediate left. This is called Fixed point representation.
- Fixed point representation suffers from a drawback that the representation can only represent a finite range (and quite small) range of numbers.

A more convenient representation is the scientific representation, where the numbers are represented in the form:

$$x = m_1.m_2m_3m_4 \times b^{\pm e}$$

Components of these numbers are:

Mantissa (m), implied base (b), and exponent (e)

Significant digits

A number such as the following is said to have 7 significant digits

$$x = \pm 0.m_1m_2m_3m_4m_5m_6m_7 \times b^{\pm e}$$

Fractions in the range 0.0 to 0.9999999 need about 24 bits of precision (in binary). For example the binary fraction with 24 1's:

$$111111111111111111111111 = 0.9999999404$$

Not every real number between 0 and 0.9999999404 can be represented by a 24-bit fractional number.

The smallest non-zero number that can be represented is:

$$00000000000000000000000001 = 5.96046 \times 10^{-8}$$

Every other non-zero number is constructed in increments of this value.

Sign and exponent digits

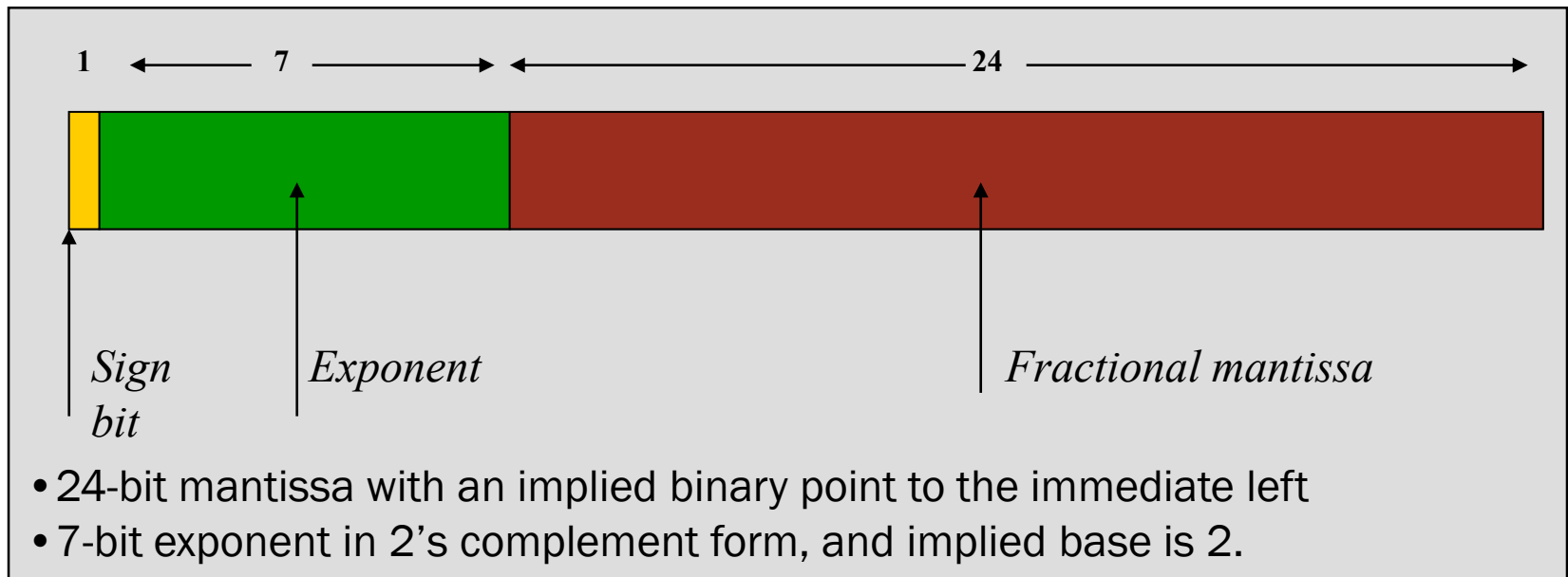
- In a 32-bit number, suppose we allocate 24 bits to represent a fractional mantissa.
- Assume that the mantissa is represented in sign and magnitude format, and we have allocated one bit to represent the sign.
- We allocate 7 bits to represent the exponent, and assume that the exponent is represented as a 2's complement integer.
- There are no bits allocated to represent the base, we assume that the base is implied for now, that is the base is 2.
- Since a 7-bit 2's complement number can represent values in the range -64 to 63, the range of numbers that can be represented is:

$$0.0000001 \times 2^{-64} \leq |x| \leq 0.9999999 \times 2^{63}$$

- In decimal representation this range is:

$$0.5421 \times 10^{-20} \leq |x| \leq 9.2237 \times 10^{18}$$

A sample representation



IEEE notation

IEEE Floating Point notation is the standard representation in use. There are two representations:

- Single precision.
- Double precision.

Both have an implied base of 2.

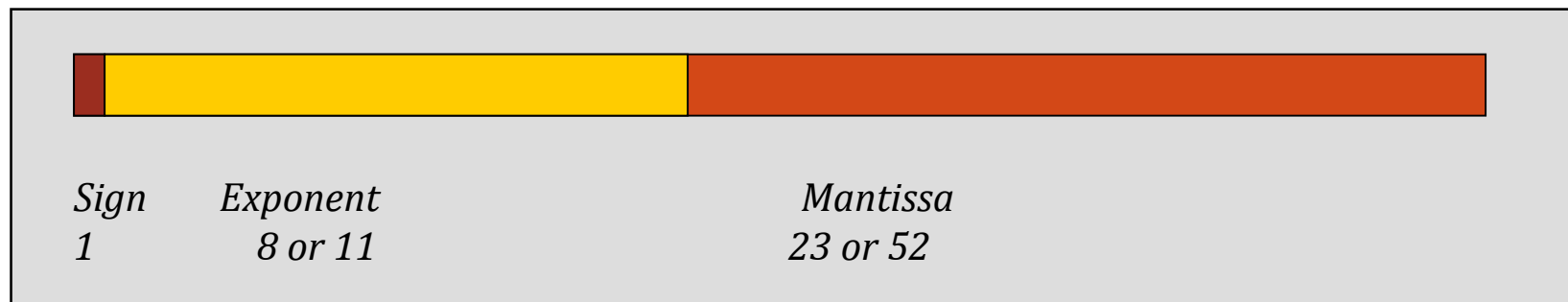
Single precision:

- 32 bits (23-bit mantissa, 8-bit exponent in excess-127 representation)

Double precision:

- 64 bits (52-bit mantissa, 11-bit exponent in excess-1023 representation)

Fractional mantissa, with an implied binary point at immediate left.



Exponent field

In the IEEE representation, the exponent is in excess-127 (excess-1023) notation.

The actual exponents represented are:

$$\begin{aligned} &-126 \leq E \leq 127 \quad \text{and} \quad -1022 \leq E \leq 1023 \\ &\text{not} \\ &-127 \leq E \leq 128 \quad \text{and} \quad -1023 \leq E \leq 1024 \end{aligned}$$

This is because the IEEE uses the exponents -127 and 128 (and -1023 and 1024), that is the actual values 0 and 255 to represent special conditions:

- Exact zero
- Infinity

Floating point arithmetic

Addition:

$$3.1415 \times 10^8 + 1.19 \times 10^6 = 3.1415 \times 10^8 + 0.0119 \times 10^8 = 3.1534 \times 10^8$$

Multiplication:

$$3.1415 \times 10^8 \times 1.19 \times 10^6 = (3.1415 \times 1.19) \times 10^{(8+6)}$$

Division:

$$3.1415 \times 10^8 / 1.19 \times 10^6 = (3.1415 / 1.19) \times 10^{(8-6)}$$

Biased exponent problem:

If a true exponent e is represented in excess- p notation, that is as $e+p$.
Then consider what happens under multiplication:

$$a. 10^{(x+p)} * b. 10^{(y+p)} = (a.b). 10^{(x+p+y+p)} = (a.b). 10^{(x+y+2p)}$$

Representing the result in excess- p notation implies that the exponent should be $x+y+p$. Instead it is $x+y+2p$.

Biases should be handled in floating point arithmetic.

Floating point arithmetic: ADD/SUB rule

- Choose the number with the smaller exponent.
- Shift its mantissa right until the exponents of both the numbers are equal.
- Add or subtract the mantissas.
- Determine the sign of the result.
- Normalize the result if necessary and truncate/round to the number of mantissa bits.

Note: This does not consider the possibility of overflow/underflow.

Floating point arithmetic: MUL rule

- Add the exponents.
- Subtract the bias.
- Multiply the mantissas and determine the sign of the result.
- Normalize the result (if necessary).
- Truncate/round the mantissa of the result.

Floating point arithmetic: DIV rule

- Subtract the exponents
- Add the bias.
- Divide the mantissas and determine the sign of the result.
- Normalize the result if necessary.
- Truncate/round the mantissa of the result.

Note: Multiplication and division does not require alignment of the mantissas the way addition and subtraction does.

Model Questions

1. Explain with figure the design and working of a 16 bit carry look ahead adder built from 4 bit adders.
2. Explain booth algorithm. Apply booth algorithm to multiply the signed numbers $+13 \times -6$ and $-13 \times +9$.
3. Write circuit arrangement for sequential binary multiplier, explain with example.
4. Differentiate between restoring and non - restoring division. Perform restoring division for the given binary numbers $1000/11$, show all cycles.
5. Design 4 bit carry look ahead logic and explain how it is faster than 4 bit ripple adder.
6. Multiply 14×-8 using booth's algorithm.
7. Explain normalization, excess exponent and special values with respect to IEEE floating point representation.
8. With figure explain circuit arrangements for binary division.
9. IEEE standard for floating point numbers, explain
10. Design a logic circuit to perform addition/ subtraction of two 'n' bit numbers X and Y.
11. Explain the different arithmetic operation on floating point numbers.



MODULE-5

BASIC PROCESSING UNIT

1

OBJECTIVE

- Basic Processing Unit:
 - Execution of instructions by a processor
 - The functional units of a processor and how they are interconnected
 - Hardware for generating control signals
 - Microprogrammed control
- Embedded Systems and Large Computer Systems
 - Embedded applications
 - Microcontrollers for embedded systems
 - Different structure for implementing multiprocessors

○ Text Books:

- Carl Hamacher, Zvonko Vranesic, Safwat Zaky: Computer Organization, 5th Edition, Tata McGraw Hill, 2002.
- Carl Hamacher, Zvonko Vranesic, Safwat Zaky, Naraig Manjikian : Computer Organization and Embedded Systems, 6th Edition, Tata McGraw Hill, 2012.

○ Reference Books:

- William Stallings: Computer Organization & Architecture, 9th Edition, Pearson, 2015.



BASIC PROCESSING UNIT

4

INTRODUCTION

- Instruction Set Processor (ISP)
- Central Processing Unit (CPU)
- A typical computing task consists of a series of steps specified by a sequence of machine instructions that constitute a program.
- An instruction is executed by carrying out a sequence of more rudimentary operations.



SOME FUNDAMENTAL CONCEPTS

6

FUNDAMENTAL CONCEPTS

- Processor fetches one instruction at a time and perform the operation specified.
- Instructions are fetched from successive memory locations until a branch or a jump instruction is encountered.
- Processor keeps track of the address of the memory location containing the next instruction to be fetched using Program Counter (PC).
- Instruction Register (IR) decodes the instruction and informs control unit to generate required signal.

EXECUTING AN INSTRUCTION

- **Fetch phase and Execution phase**
- Fetch the contents of the memory location pointed to by the PC. The contents of this location are loaded into the IR (**Fetch phase**).

$$IR \leftarrow [[PC]]$$

- Assuming that the memory is byte addressable, increment the contents of the PC by 4 (fetch phase).

$$PC \leftarrow [PC] + 4$$

- Carry out the actions specified by the instruction in the IR (**execution phase**).

PROCESSOR ORGANIZATION-1

- The registers, the ALU, and the interconnecting bus are collectively referred to as the *data path*.

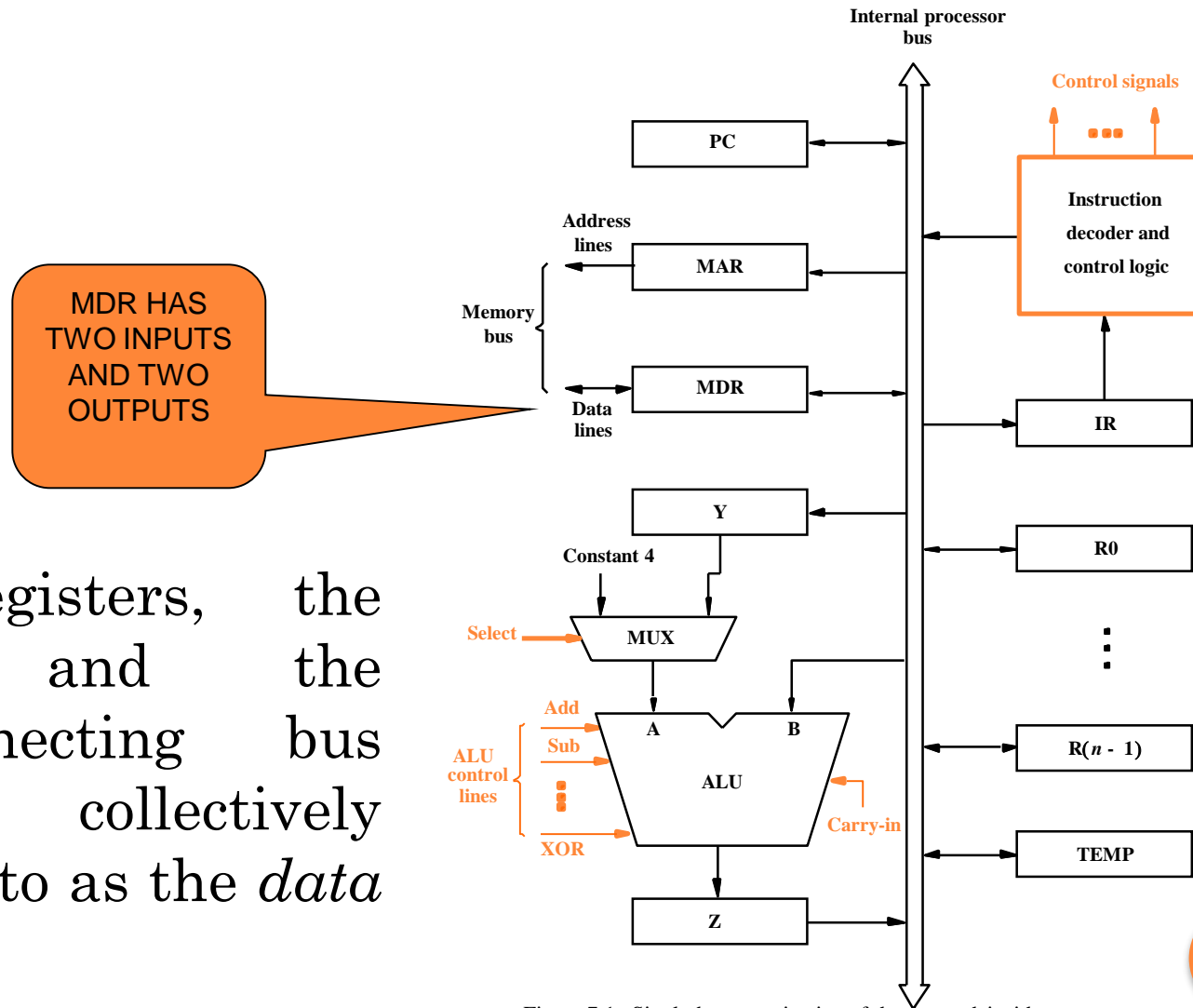


Figure 7.1. Single-bus organization of the datapath inside a processor.

EXECUTING AN INSTRUCTION

- Transfer a word of data from one processor register to another or to the ALU.
- Perform an arithmetic or a logic operation and store the result in a processor register.
- Fetch the contents of a given memory location and load them into a processor register.
- Store a word of data from a processor register into a given memory location.

REGISTER TRANSFERS

- The input and output of register R_i are connected to the bus via switches controlled by the signals R_{iin} and R_{iout} respectively.
- When R_{iin} is set to 1, the data on the bus are loaded into R_i .
- R_{iout} , is set to 1, the contents of register R_i are placed on the bus.
- While R_{iout} is equal to 0, the bus can be used for transferring data from other registers.
- All operation coordinate by processor clock. That can be rising or falling edge of the clock.
- E.g. Data transfer from R_1 to R_4

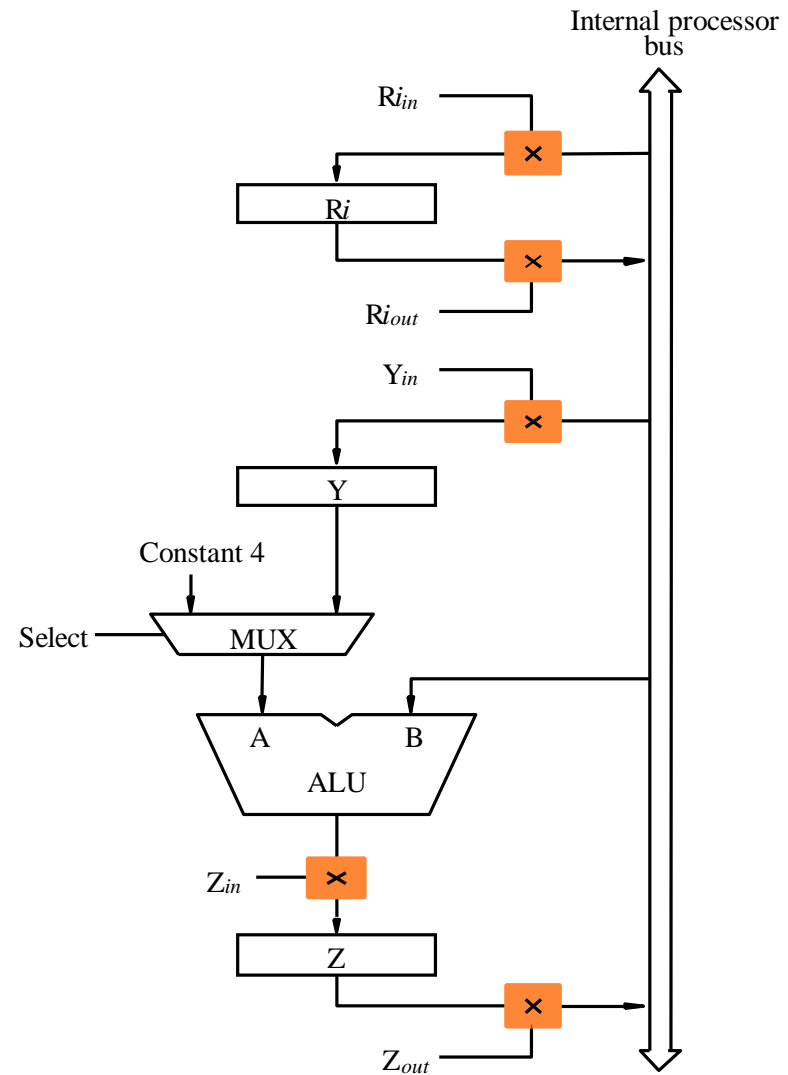


Figure 7.2. Input and output gating for the registers in Figure 7.1.

REGISTER TRANSFERS

- A two-input multiplexer is used to select the data applied to the input of an edge-triggered D flip-flop.
- When the control input R_{in} is equal to 1, the multiplexer selects the data on the bus. This data will be loaded into the flip-flop at the rising edge of the clock.
- When R_{in} is equal to 0, the multiplexer feeds back the value currently stored in the flip-flop. The Q output of the flip-flop is connected to the bus via a tri-state gate.
- When R_{out} is equal to 0, the gate's output is in the high-impedance (electrically disconnected) state. This corresponds to the open-circuit state of a switch.
- When $R_{out} = 1$, the gate drives the bus to 0 or 1, depending on the value of Q .

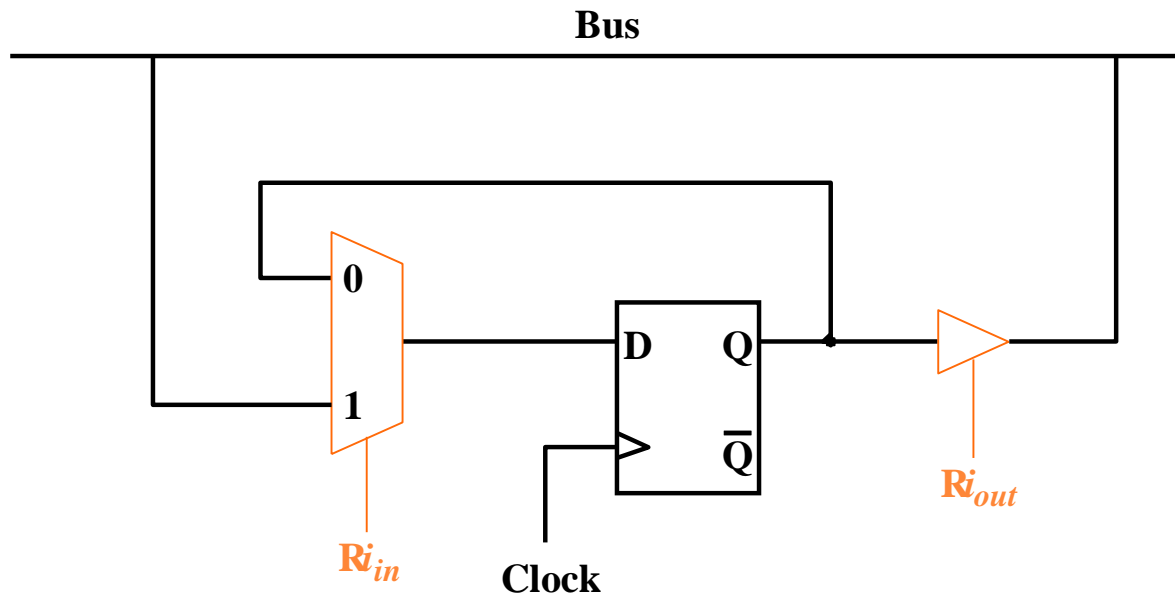


Figure 7.3. Input and output gating for one register bit.

PERFORMING AN ARITHMETIC OR LOGIC OPERATION

- The ALU is a combinational circuit that has no internal storage.
- ALU gets the two operands from MUX and bus. The result is temporarily stored in register Z.
- What is the sequence of operations to add the contents of register R1 to those of R2 and store the result in R3?
 1. R1out, Yin (1 Clock Cycle)
 2. R2out, SelectY, Add, Zin (2 Clock Cycle)
 3. Zout, R3in (3 Clock Cycle)

FETCHING A WORD FROM MEMORY

- Address into MAR; issue Read operation; data into MDR.

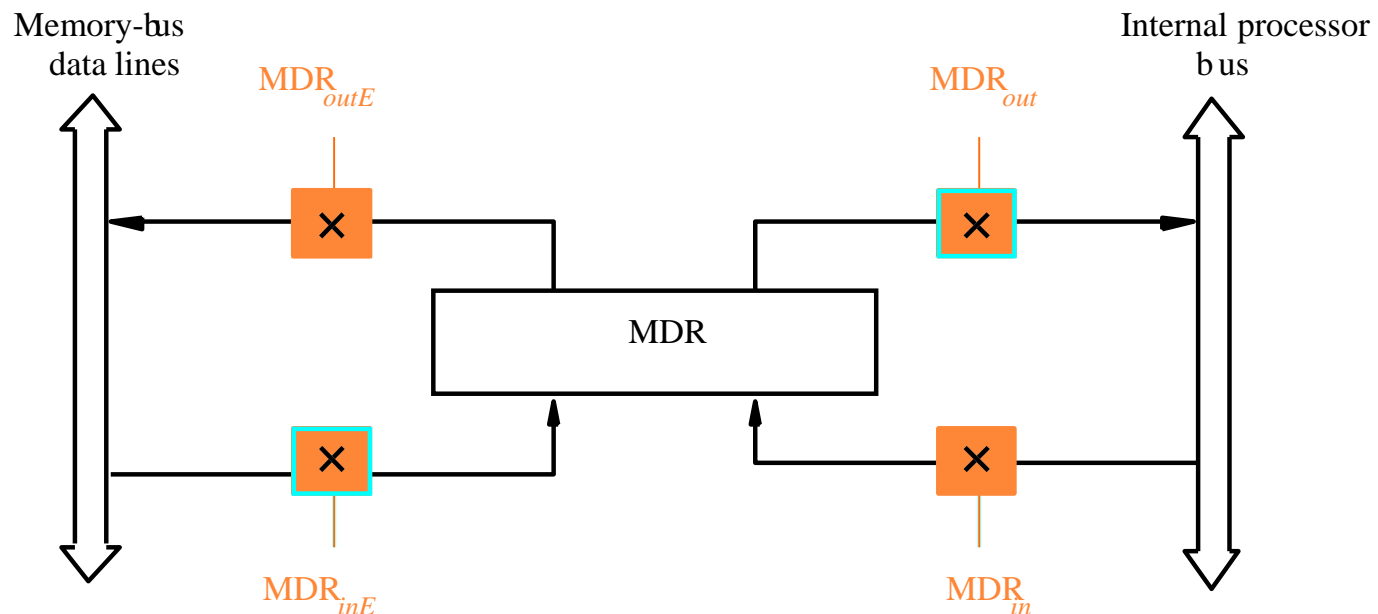


Figure 7.4. Connection and control signals for register MDR.

Fetching a Word from Memory

- The response time of each memory access varies (cache miss, memory-mapped I/O,...).
- To accommodate this, the processor waits until it receives an indication that the requested operation has been completed (Memory-Function-Completed, MFC).
- Move (R1), R2
 1. $MAR \leftarrow [R1]$
 2. Start a Read operation on the memory bus
 3. Wait for the MFC response from the memory
 4. Load MDR from the memory bus
 5. $R2 \leftarrow [MDR]$

TIMING

Assume MAR is always available on the address lines of the memory bus.

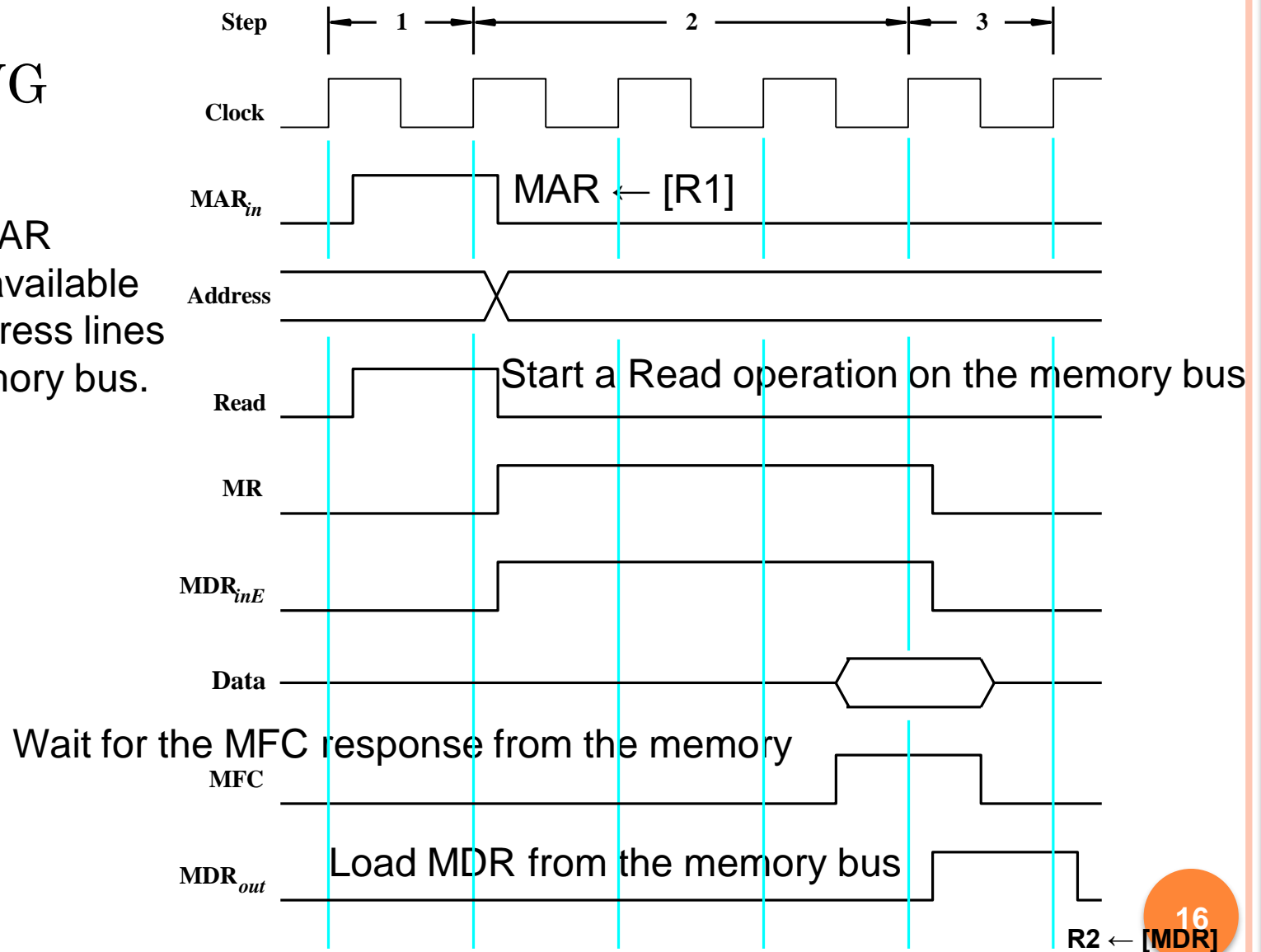


Figure 7.5. Timing of a memory Read operation.

FETCHING A WORD FROM MEMORY

- The memory read operation requires three steps described as follows:
 1. $R1_{out}$, MAR_{in} , Read
 2. MDR_{inE} , WMFC
 3. MDR_{out} , $R2_{in}$
- Similar steps are for Write data into memory (Store in memory)

EXECUTION OF A COMPLETE INSTRUCTION

- Add (R3), R1
 1. Fetch the instruction
 2. Fetch the first operand (the contents of the memory location pointed to by R3)
 3. Perform the addition
 4. Load the result into R1

ARCHITECTURE

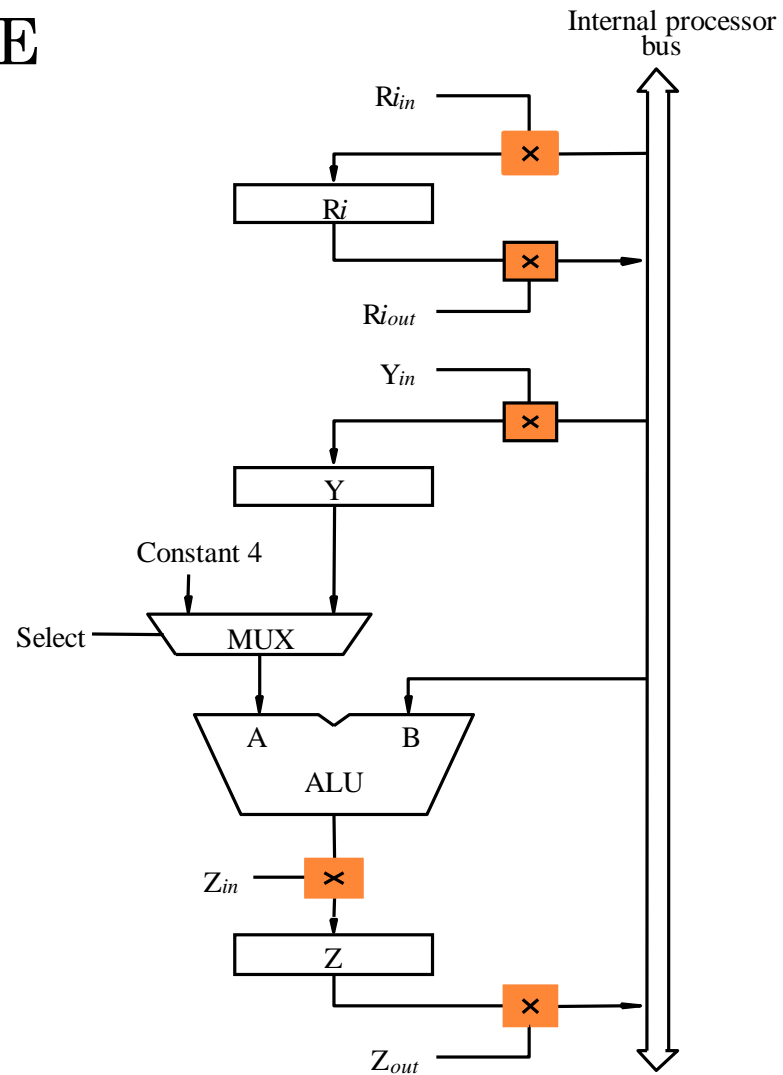
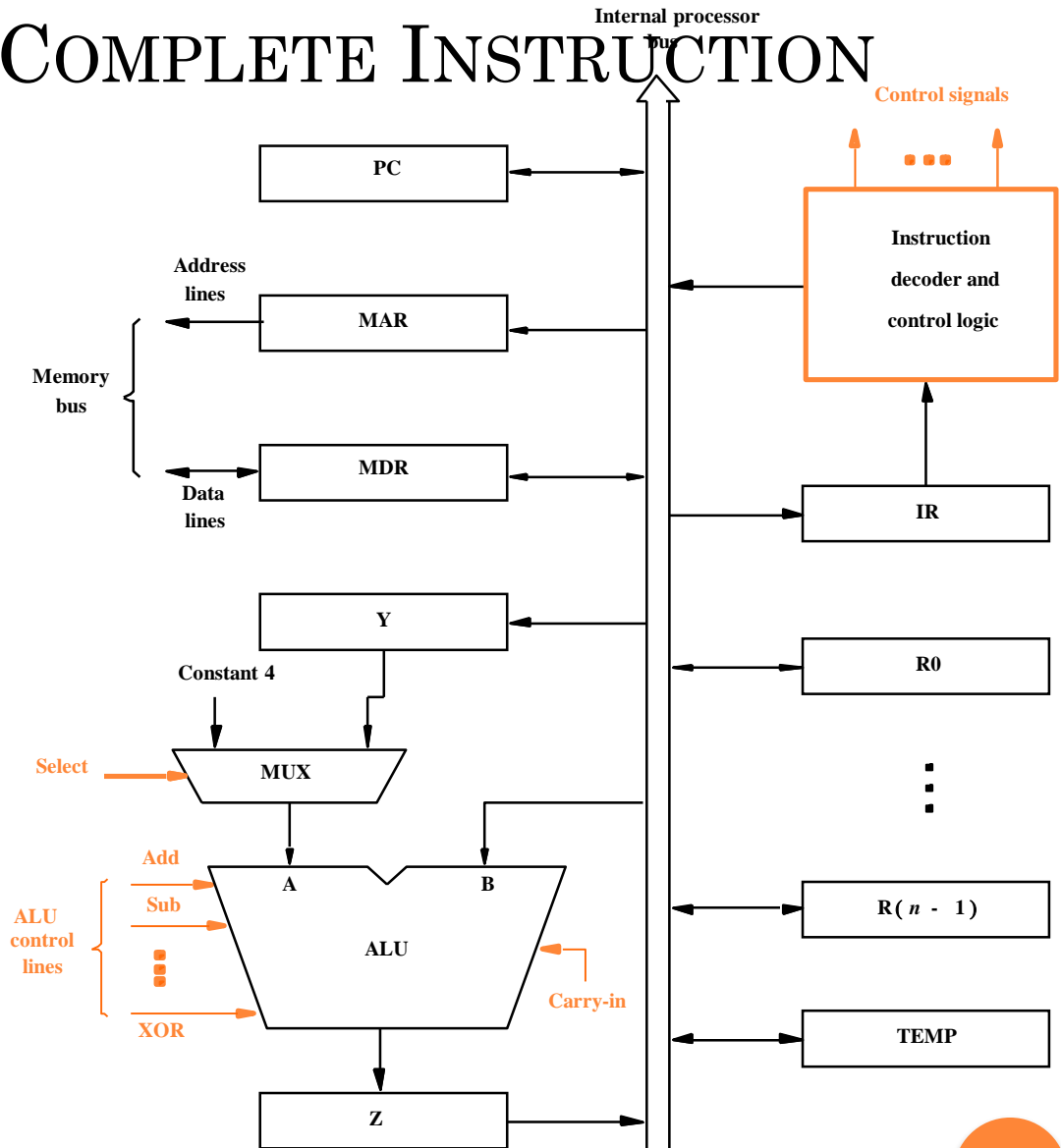


Figure 7.2. Input and output gating for the registers in Figure 7.1.

EXECUTION OF A COMPLETE INSTRUCTION

Add (R3), R1



Yin step 2. There is no need to copy the updated contents of PC into register Y when executing the Add instruction. But, in Branch instructions the updated value of the PC is needed to compute the Branch target address.

EXECUTION OF BRANCH INSTRUCTIONS

- A branch instruction replaces the contents of PC with the branch target address, which is usually obtained by adding an offset X given in the branch instruction.
- The offset X is usually the difference between the branch target address and the address immediately following the branch instruction.
- Conditional branch

EXECUTION OF BRANCH INSTRUCTIONS

Step Action

- 1 PC_{out} , MAR_{in} , Read, Select4, Add, Z_{in}
- 2 Z_{out} , PC_{in} , Y_{in} , WMFC
- 3 MDR_{out} , IR_{in}
- 4 Offset-field-of- IR_{out} , Add, Z_{in}
- 5 Z_{out} , PC_{in} , End

For Conditional branch Instruction

E.g.

Offset-field-of- IR_{out} , Add, Z_{in} , If N=0 then End

if N (Negative)=0 processor returns to step 1 immediately after step 4

if N (Negative)=1 step 5 is performed

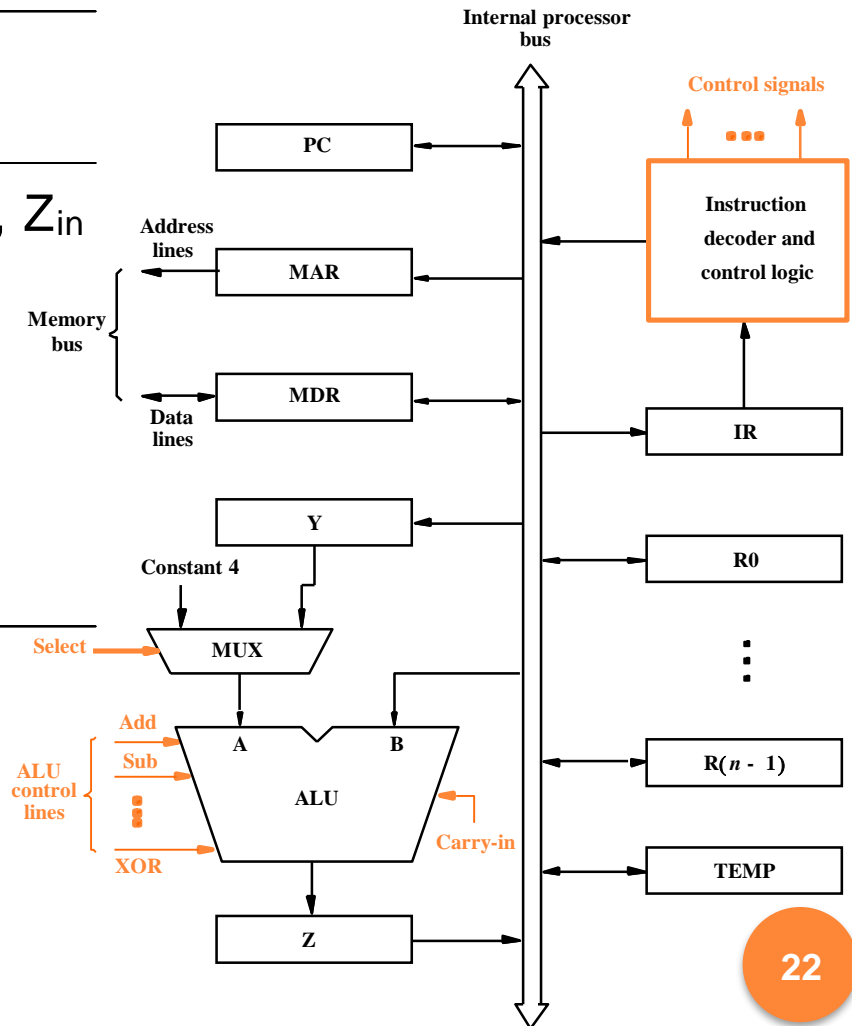


Figure 7.7. Control sequence for an unconditional branch instruction.

MULTIPLE-BUS ORGANIZATION

- The resulting control sequences shown are quite long because only one data item can be transferred over the bus in a clock cycle.
- To reduce the number of steps needed, most commercial processors provide multiple internal paths that enable several transfers to take place in parallel.

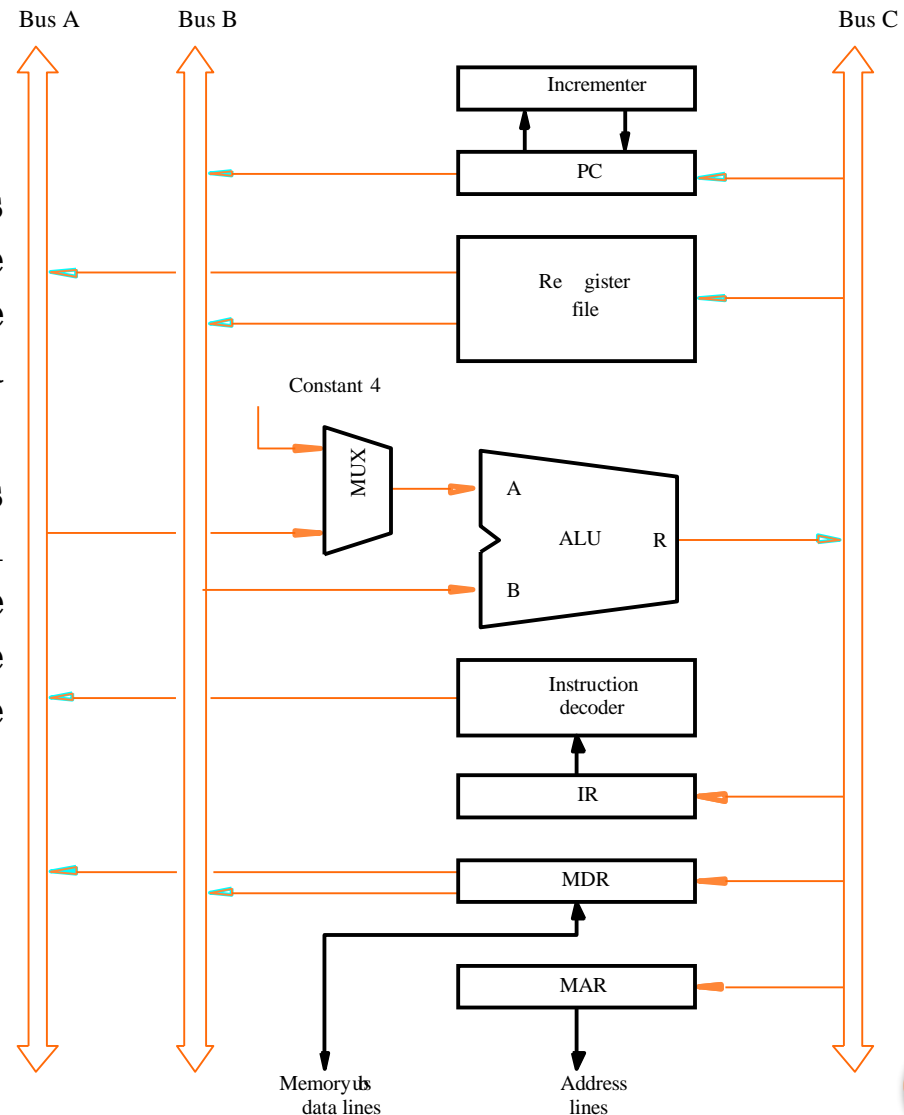


Figure 7.8. Three-bus organization of the datapath.

MULTIPLE-BUS ORGANIZATION

- Add R4, R5, R6

Step Action

1	PC_{out} , $R=B$, MAR_{in} , Read, IncPC
2	WMFC
3	MDR_{outB} , $R=B$, IR_{in}
4	$R4_{outA}$, $R5_{outB}$, SelectA, Add, $R6_{in}$, End

Control sequence for the instruction. Add R4,R5,R6,for the three-bus organization

Buses A and B are used to transfer the source operands to A and B inputs of the ALU. The result of ALU (R) is transferred to the destination C.

$R=B$ Bus B data transferred to the Bus C



HARDWIRED CONTROL

25

OVERVIEW

- To execute instructions, the processor must have some means of generating the control signals needed in the proper sequence.
- Two categories: hardwired control and microprogrammed control
- Hardwired system can operate at high speed; but with little flexibility.

CONTROL UNIT ORGANIZATION-1

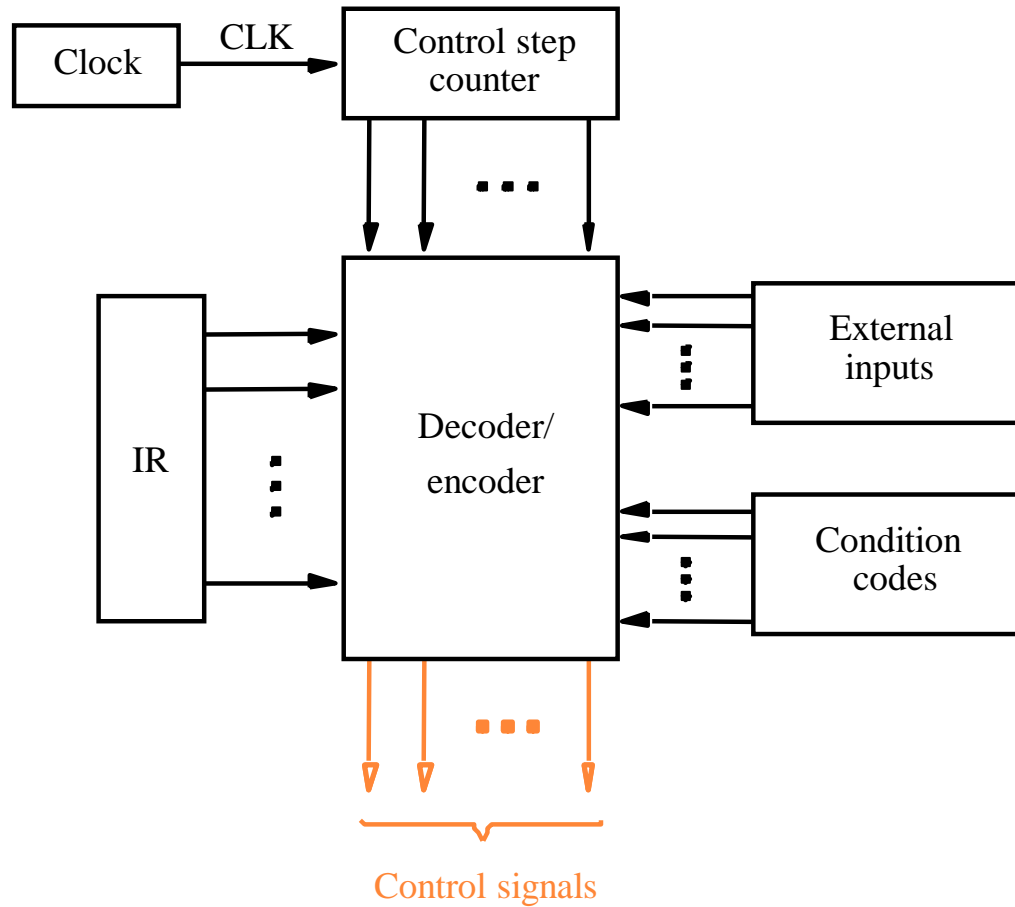


Figure 7.10. Control unit organization.

CONTROL UNIT ORGANIZATION

- The sequence of control signals step is complete in one clock period.
- A counter may be used to keep track of the control steps, as shown in previous slide. Each state, or count, of this counter corresponds to one control step.
- The required control signals are determined by the following information:
 1. Contents of the control step counter
 2. Contents of the instruction register
 3. Contents of the condition code flags
 4. External input signals, such as MFC and interrupt requests

DETAILED BLOCK DESCRIPTION

BY SEPARATING THE DECODING AND ENCODING FUNCTIONS, WILL OBTAIN THE MORE DETAILED BLOCK DIAGRAM

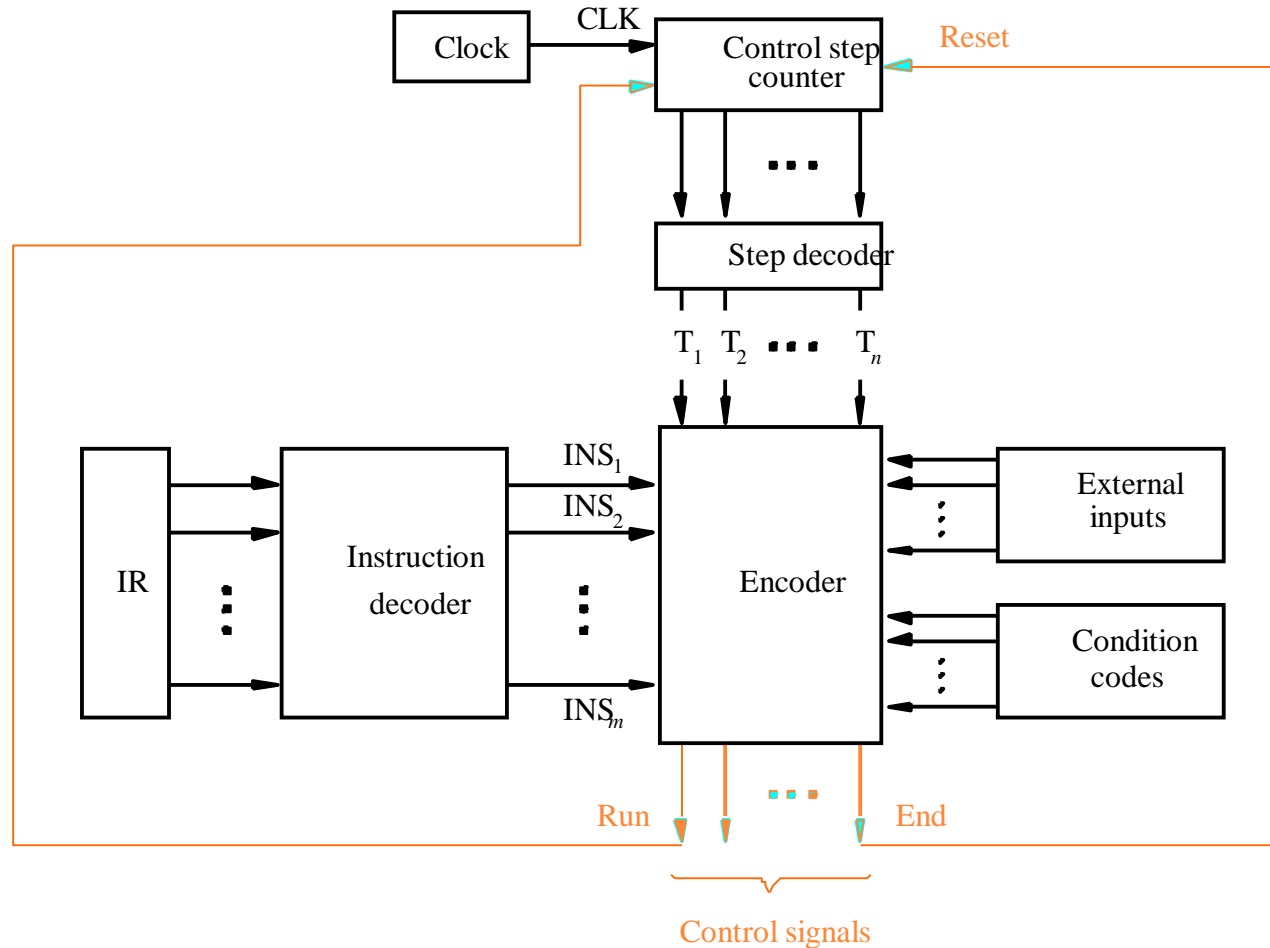


Figure 7.11. Separation of the decoding and encoding functions.

DETAILED BLOCK DESCRIPTION

- The step decoder provides a separate signal line for each step, or time slot, in the control sequence.
- Similarly, the output of the instruction decoder consists of a separate line for each machine instruction. For any instruction loaded in the IR, one of the output lines INS_1 through INS_m is set to 1, and all other lines are set to 0.
- The input signals to the encoder block in diagram combined to generate the individual control signals Y_{in} , PC_{out} , Add , End , and so on.

GENERATING Z_{IN}

- $Z_{in} = T_1 + T_6 \cdot ADD + T_4 \cdot BR + \dots$
- This signal is asserted during time slot T1 for all instructions, during T6 for an Add instruction, during T4 for an unconditional branch instruction, and so on. The logic function for Z_{in} is derived from the control sequences

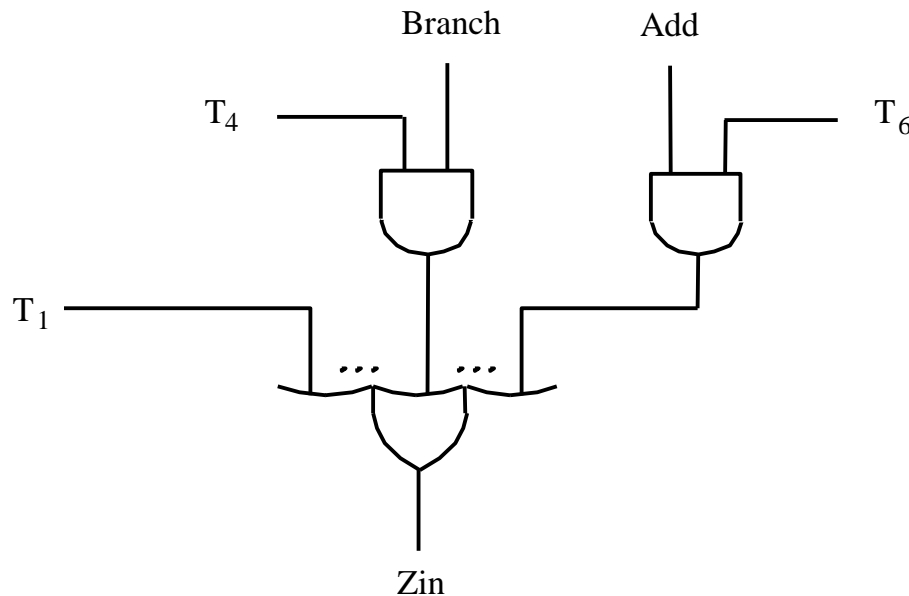


Figure 7.12. Generation of the Z_{in} control signal for the processor in Figure 7.1.

GENERATING END

- $\text{End} = T_7 \cdot \text{ADD} + T_5 \cdot \text{BR} + (T_5 \cdot N + T_4 \cdot \bar{N}) \cdot \text{BRN} + \dots$
- The End signal starts a new instruction fetch cycle by resetting the control step counter to its starting value.
- Set to 1, RUN causes the counter to be incremented by one at the end of every clock cycle. When RUN is equal to 0, the counter stops counting.
- This is needed whenever the WMFC signal is issued, to cause the processor to wait for the reply from the memory.

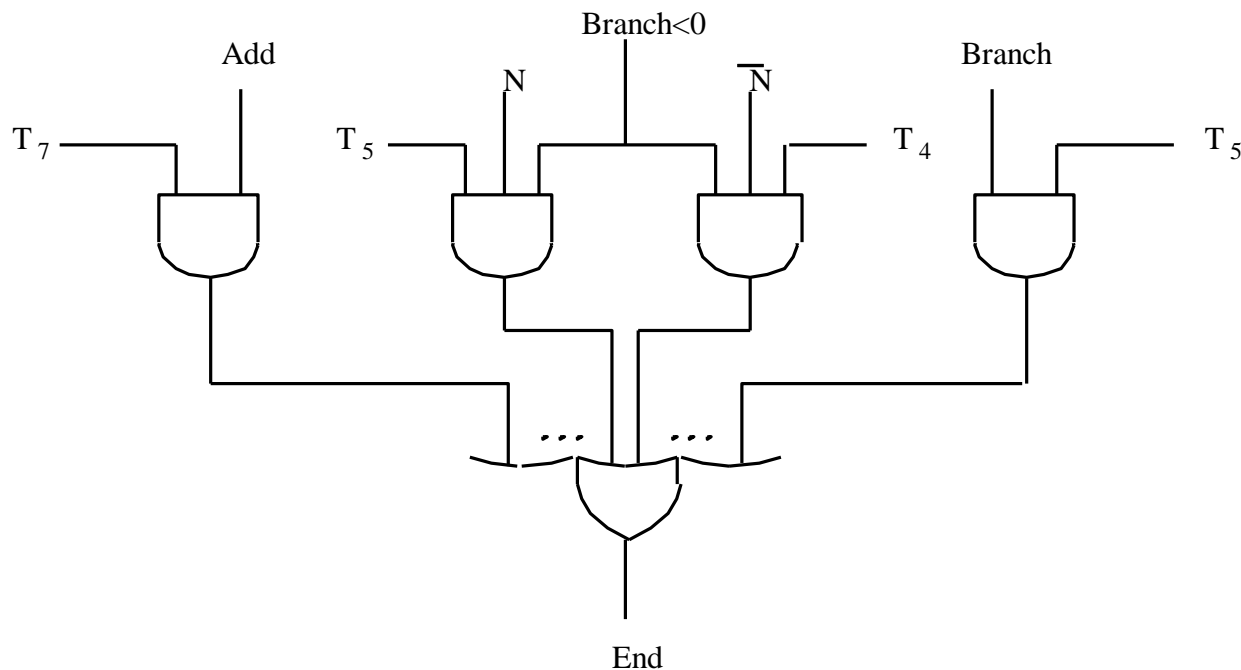
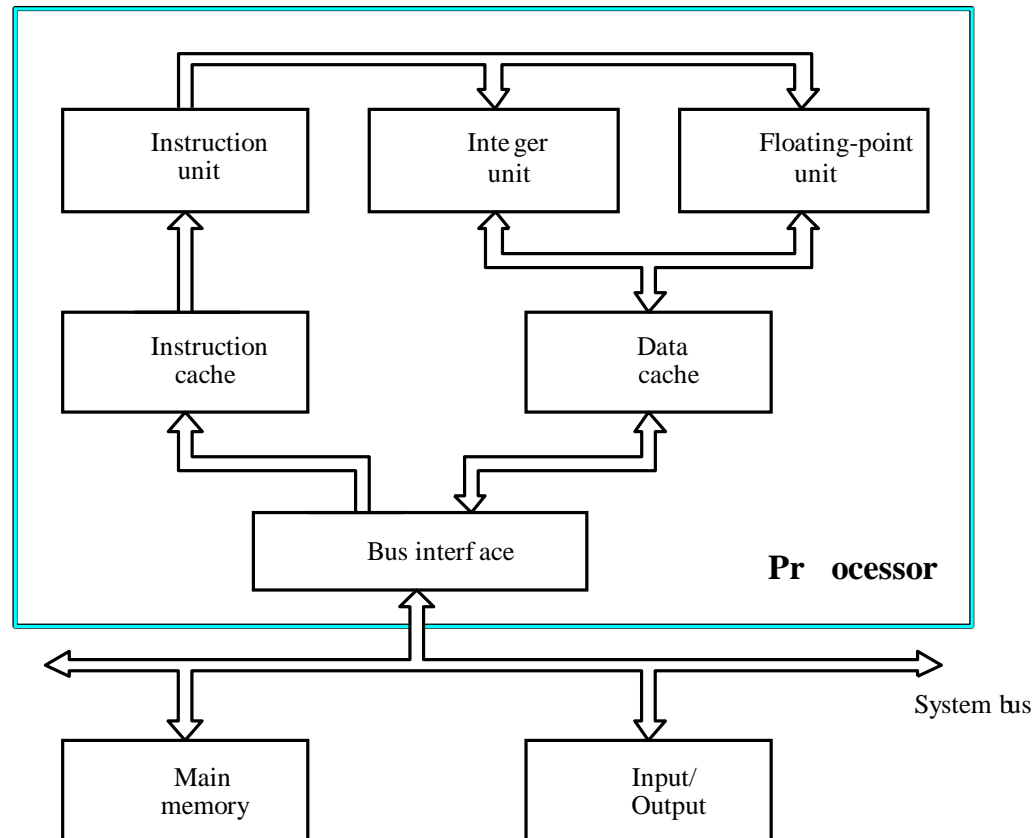


Figure 7.13. Generation of the End control signal.

A COMPLETE PROCESSOR



Block diagram of a complete processor



MICROPROGRAMMED CONTROL

34

OVERVIEW

OVERVIEW

- Control signals are generated by a program similar to machine language programs.
- Control Word (CW); microroutine; microinstruction

Micro - instruction	,,	PC _{in}	PC _{out}	MAR _{in}	Read	MDR _{out}	IR _{in}	Y _{in}	Select	Add	Z _{in}	Z _{out}	R1 _{out}	R1 _{in}	R3 _{out}	WMFC	End	,
1		0	1	1	1	0	0	0	1	1	1	0	0	0	0	0	0	
2		1	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	
3		0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	
4		0	0	1	1	0	0	0	0	0	0	0	0	0	1	0	0	
5		0	0	0	0	0	0	1	0	0	0	0	1	0	0	1	0	
6		0	0	0	0	1	0	0	0	1	1	0	0	0	0	0	0	
7		0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1	

Figure 7.15 An example of microinstructions for Figure 7.6.

OVERVIEW

- Control store

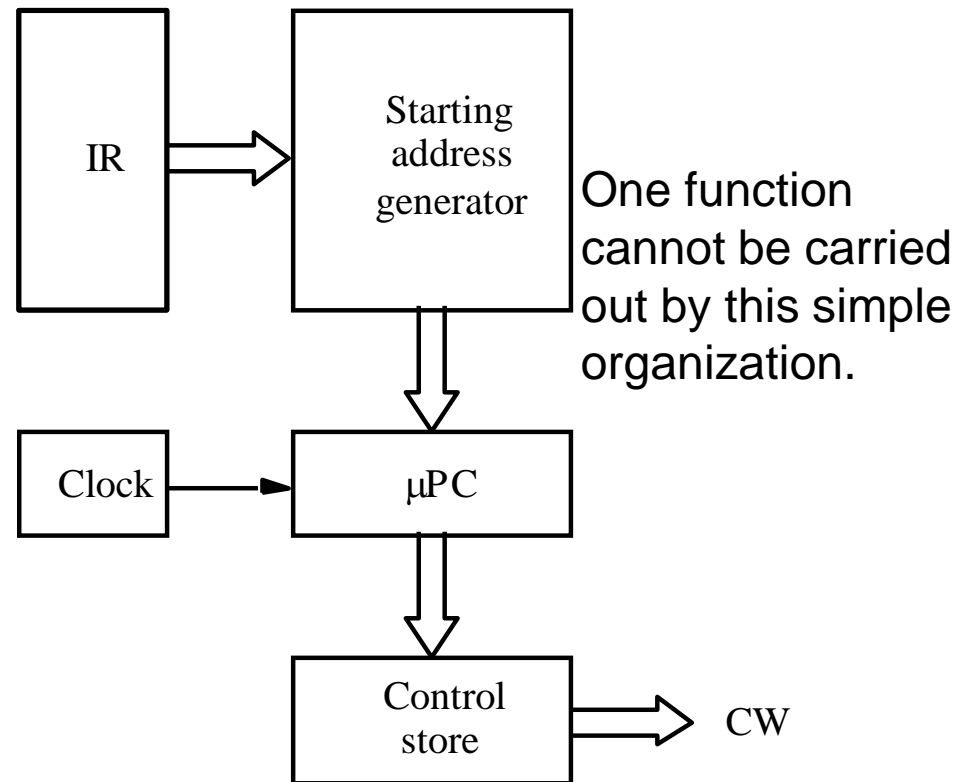


Figure 7.16. Basic organization of a microprogrammed control unit.

OVERVIEW

- A **control word (CW)** is a word whose individual bits represent the various control signals. Each of the control steps in the control sequence of an instruction defines a unique combination of 1s and 0s in the CW.
- A sequence of CWs corresponding to the control sequence of a machine instruction constitutes the **microroutine** for that instruction, and the individual control words in this micro routine are referred to as **microinstructions**.

OVERVIEW

- The micro routines for all instructions in the instruction set of a computer are stored in a special memory called the control store.
- The control unit can generate the control signals for any instruction by sequentially reading the CWs of the corresponding micro routine from the control store.
- To read the control words sequentially from the control store, a micro program counter (μ PC) is used. Every time a new instruction is loaded into the IR, the output of the block labeled "starting address generator" is loaded into the μ PC.
- The μ PC is then automatically incremented by the clock, causing successive microinstructions to be read from the control store.
- Hence, the control signals are delivered to various parts of the processor in the correct sequence.

OVERVIEW

- The previous organization cannot handle the situation when the control unit is required to check the status of the condition codes or external inputs to choose between alternative courses of action.
- Use conditional branch microinstruction.

AddressMicroinstruction	
0	PC_{out} , MAR_{in} , Read, Select4, Add, Z_{in}
1	Z_{out} , PC_{in} , Y_{in} , WMFC
2	MDR_{out} , IR_{in}
3	Branch to starting address of appropriate microroutine
.....	
25	If $N=0$, then branch to microinstruction 0
26	Offset-field-of- IR_{out} , SelectY, Add, Z_{in}
27	Z_{out} , PC_{in} , End

Figure 7.17. Microroutine for the instruction Branch<0.

OVERVIEW

- The instruction Branch <0 loading this instruction into IR, a branch microinstruction transfers control to the corresponding micro routine, which is assumed to start at location 25 in the control store. This address is the output of starting address generator block codes.
- If this bit is equal to 0, a branch takes place to location 0 to fetch a new machine instruction. Otherwise, the microinstruction at location 0 to fetch a new machine instruction.
- Otherwise the microinstruction at location 27 loads this address into the PC.

OVERVIEW

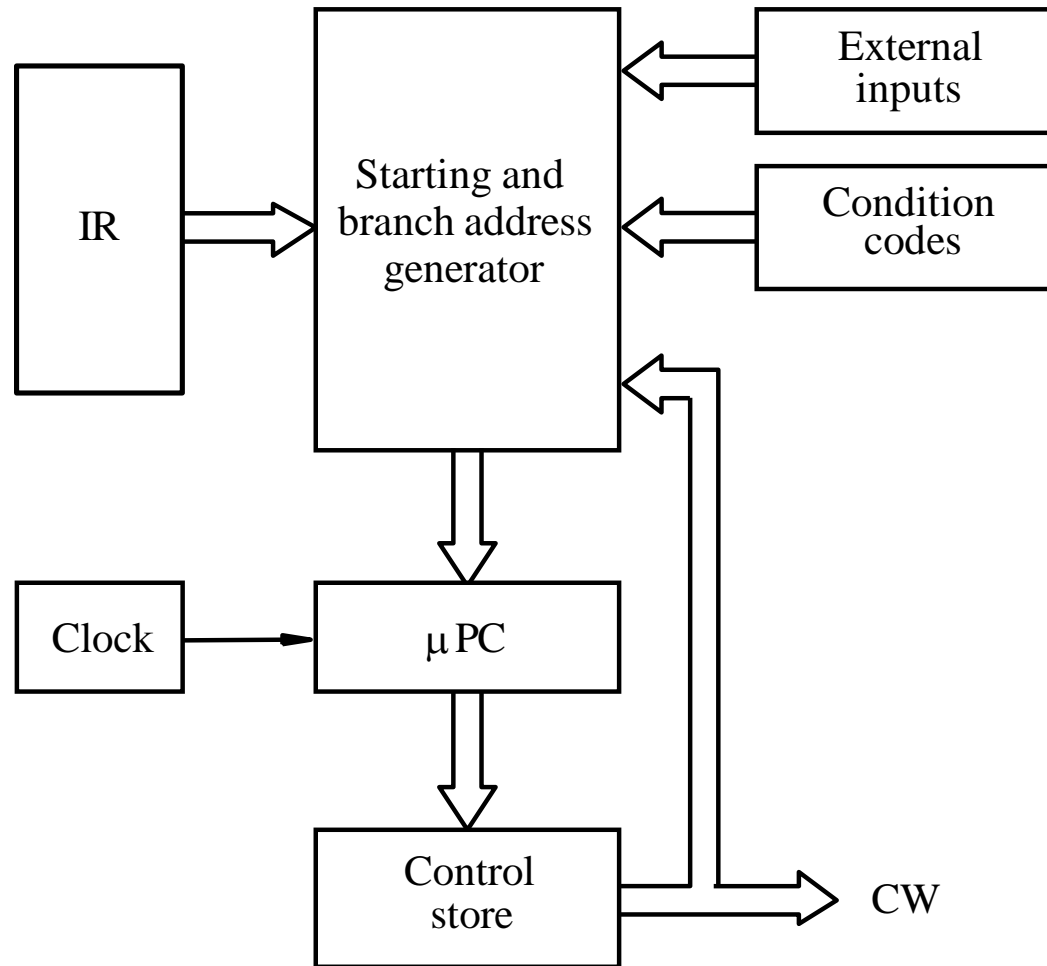


Figure 7.18. Organization of the control unit to allow conditional branching in the microprogram.

MICROINSTRUCTIONS

- A straightforward way to structure microinstructions is to assign one bit position to each control signal.
- However, this is very inefficient.
- The length can be reduced: most signals are not needed simultaneously, and many signals are mutually exclusive.
- All mutually exclusive signals are placed in the same group in binary coding.
- Assigning individual bits to each control signal results in long microinstructions because the number of required signals is usually large.
- Moreover, only a few bits are set to 1 (to be used for active gating) in any given microinstruction, which means the available bit space is poorly used.

PARTIAL FORMAT FOR THE MICROINSTRUCTIONS

Microinstruction

F1	F2	F3	F4	F5
F1 (4 bits)	F2 (3 bits)	F3 (3 bits)	F4 (4 bits)	F5 (2 bits)
0000: No transfer	000: No transfer	000: No transfer	0000: Add	00: No action
0001: PC _{out}	001: PC _{in}	001: MAR _{in}	0001: Sub	01: Read
0010: MDR _{out}	010: IR _{in}	010: MDR _{in}	⋮	10: Write
0011: Z _{out}	011: Z _{in}	011: TEMP _{in}	1111: XOR	
0100: R0 _{out}	100: R0 _{in}	100: Y _{in}	⏟	
0101: R1 _{out}	101: R1 _{in}		16 ALU functions	
0110: R2 _{out}	110: R2 _{in}			
0111: R3 _{out}	111: R3 _{in}			
1010: TEMP _{out}				
1011: Offset _{out}				
F6	F7	F8	...	
F6 (1 bit)	F7 (1 bit)	F8 (1 bit)		
0: SelectY	0: No action	0: Continue		
1: Select4	1: WMFC	1: End		

What is the price paid for this scheme?

Figure 7.19.

An example of a partial format for field-encoded microinstructions.

PARTIAL FORMAT FOR THE MICROINSTRUCTIONS

- The length of the microinstructions can be reduced easily.
- Most signals are not needed simultaneously, and many signals are mutually exclusive. For e.g., only one function of the ALU can be activated at a time.
- The source for a data transfer must be unique because it is not possible to gate the contents of two different registers onto the bus at the same time.
- Read and Write signals to the memory cannot be active simultaneously.
- This suggests that signals can be grouped so that all mutually exclusive signals are placed in the same group. Thus, at most one *micro operation* per group is specified in any microinstruction.

PARTIAL FORMAT FOR THE MICROINSTRUCTIONS

- Then it is possible to use a binary coding scheme to represent the signals within a group.
- For example, four bits suffice to represent the 16 available functions in the ALU. Register output control signals can be placed in a group consisting of PCout, MDRout, Zout, Offsetout, R0out, R1out, R2out, R3out, and TEMPout. Any one of these can be selected by a unique 4-bit code.

FURTHER IMPROVEMENT

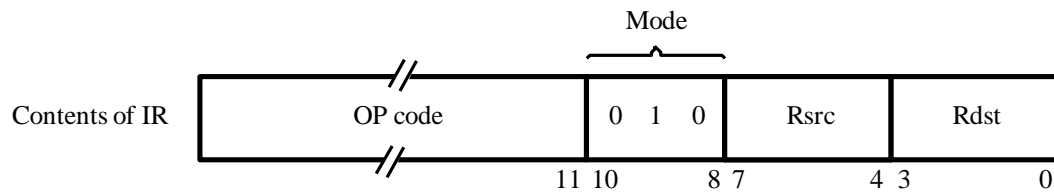
- Enumerate the patterns of required signals in all possible microinstructions. Each meaningful combination of active control signals can then be assigned a distinct code.
- Vertical organization
- Horizontal organization

FURTHER IMPROVEMENT

- Highly encoded schemes that use compact codes to specify only a small number of control functions in each microinstruction are referred to as a ***vertical organization***.
- On the other hand, the minimally encoded scheme, in which many resources can be controlled with a single microinstruction, is called a ***horizontal organization***.
- The horizontal approach is useful when a higher operating speed is desired and when the machine structure allows parallel use of resources.
- The vertical approach results in considerably slower operating speeds because more microinstructions are needed to perform the desired control functions. Although fewer bits are required for each microinstruction, this does not imply that the total number of bits in the control store is smaller. The significant factor is that less hardware is needed to handle the execution of microinstructions.
- Horizontal and vertical organizations represent the two organizational extremes in micro programmed control.

MICROPROGRAM SEQUENCING

- If all microprograms require only straightforward sequential execution of microinstructions except for branches, letting a μ PC governs the sequencing would be efficient.
- However, two disadvantages:
 - Having a separate microroutine for each machine instruction results in a large total number of microinstructions and a large control store.
 - Longer execution time because it takes more time to carry out the required branches.
- Example: Add src, Rdst
- Four addressing modes: register, autoincrement, autodecrement, and indexed (with indirect forms).



Address (octal)	Microinstruction
000	PC_{out}, MAR_{in} , Read, Select 4, Add, Z_{in}
001	Z_{out}, PC_{in}, Y_{in} , WMFC
002	MDR_{out}, IR_{in}
003	μ Branch { $\mu PC \leftarrow 101$ (from Instruction decoder); $\mu PC_{5,4} \leftarrow [IR_{10,9}]; \mu PC_3 \leftarrow [\overline{IR_{10}}] \times [\overline{IR_9}] \times [IR_8]$ }
121	$Rsrc_{out}, MAR_{in}$, Read, Select4, Add, Z_{in}
122	$Z_{out}, Rsrc_{in}$
123	μ Branch { $\mu PC \leftarrow 170; \mu PC_0 \leftarrow [\overline{IR_8}]$ }, WMFC
170	MDR_{out}, MAR_{in} , Read, WMFC
171	MDR_{out}, Y_{in}
172	$Rdst_{out}$, Select Y, Add, Z_{in}
173	$Z_{out}, Rdst_{in}$, End

Figure 7.21. Microinstruction for Add (Rsrc)+,Rdst.

Note: Microinstruction at location 170 is not executed for this addressing mode.

MICROINSTRUCTIONS WITH NEXT-ADDRESS FIELD

- The microprogram we discussed requires several branch microinstructions, which perform no useful operation in the datapath.
- A powerful alternative approach is to include an address field as a part of every microinstruction to indicate the location of the next microinstruction to be fetched.
- Pros: separate branch microinstructions are virtually eliminated; few limitations in assigning addresses to microinstructions.
- Cons: additional bits for the address field (around 1/6)

MICROINSTRUCTIONS WITH NEXT-ADDRESS FIELD

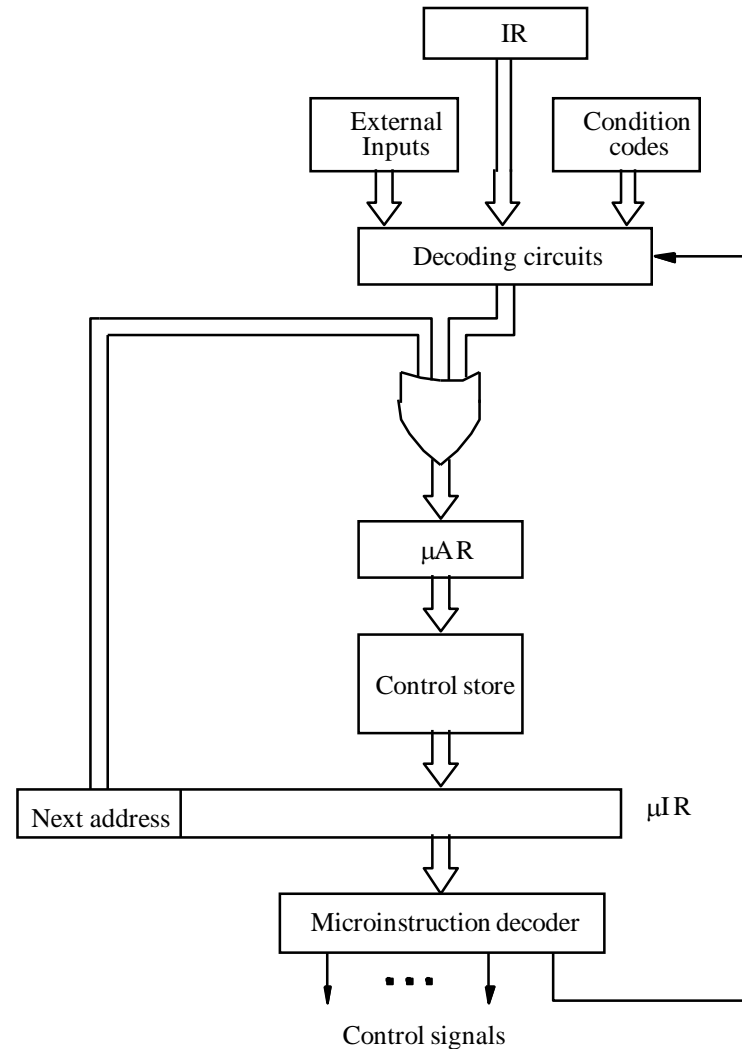


Figure 7.22. Microinstruction-sequencing organization.

Microinstruction

F0	F1	F2	
F0 (8 bits)	F1 (3 bits)	F2 (3 bits)	F3 (3 bits)
Address of next microinstruction	000: No transfer 001: PC _{out} 010: MDR _{out} 011: Z _{out} 100: Rsrc _{out} 101: Rdst _{out} 110: TEMP _{out}	000: No transfer 001: PC _{in} 010: IR _{in} 011: Z _{in} 100: Rsrc _{in} 101: Rdst _{in}	000: No transfer 001: MAR _{in} 010: MDR _{in} 011: TEMP _{in} 100: Y _{in}

	F5	F6	F7
F4 (4 bits)	F5 (2 bits)	F6 (1 bit)	F7 (1 bit)
0000: Add 0001: Sub ⋮ 1111: XOR	00: No action 01: Read 10: Write	0: SelectY 1: Select4	0: No action 1: WMFC

	F9	F10
F8 (1 bit)	F9 (1 bit)	F10 (1 bit)
0: NextAdrs 1: InstDec	0: No action 1: OR _{mode}	0: No action 1: OR _{indsrc}

Figure 7.23.

Format for microinstructions in the example of Section 7.5.3.

IMPLEMENTATION OF THE MICROROUTINE

Octal address	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
0 0 0	0 0 0 0 0 0 0 1	0 0 1	0 1 1	0 0 1	0 0 0 0	0 1	1	0	0	0	0
0 0 1	0 0 0 0 0 0 1 0	0 1 1	0 0 1	1 0 0	0 0 0 0	0 0	0	1	0	0	0
0 0 2	0 0 0 0 0 0 1 1	0 1 0	0 1 0	0 0 0	0 0 0 0	0 0	0	0	0	0	0
0 0 3	0 0 0 0 0 0 0 0	0 0 0	0 0 0	0 0 0	0 0 0 0	0 0	0	0	1	1	0
1 2 1	0 1 0 1 0 0 1 0	1 0 0	0 1 1	0 0 1	0 0 0 0	0 1	1	0	0	0	0
1 2 2	0 1 1 1 1 0 0 0	0 1 1	1 0 0	0 0 0	0 0 0 0	0 0	0	1	0	0	1
1 7 0	0 1 1 1 1 0 0 1	0 1 0	0 0 0	0 0 1	0 0 0 0	0 1	0	1	0	0	0
1 7 1	0 1 1 1 1 0 1 0	0 1 0	0 0 0	1 0 0	0 0 0 0	0 0	0	0	0	0	0
1 7 2	0 1 1 1 1 0 1 1	1 0 1	0 1 1	0 0 0	0 0 0 0	0 0	0	0	0	0	0
1 7 3	0 0 0 0 0 0 0 0	0 1 1	1 0 1	0 0 0	0 0 0 0	0 0	0	0	0	0	0

Figure 7.24. Implementation of the microroutine of Figure 7.21 using a next-microinstruction address field. (See Figure 7.23 for encoded signals.)

BIT-ORING

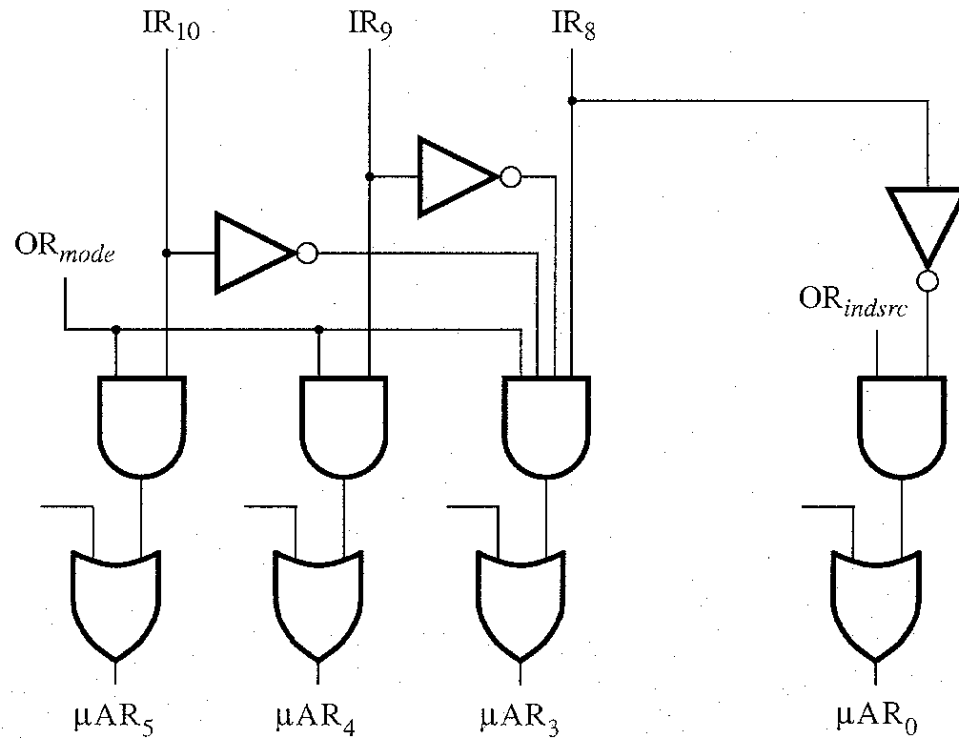


Figure 7.26. Control circuitry for bit-ORing
(part of the decoding circuits in Figure 7.25).

FURTHER DISCUSSIONS

○ Prefetching Microinstruction

- To make faster the operation

○ Emulation

- Instruction set of computer M2 is can run on Computer M2 i.e. M1 emulated M2.
- Emulate allows to replace outdated equipment with more up to data machines.
- If replacement computer fully emulate the original one then no software change have to be made to run existing program.