Contents

1. Cover Page

GEETHANJALI COLLEGE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

(Name of the Subject / Lab Course) COMPUTER ORGANIZATION

(JNTU CODE –A40506) **Programme : UG / PG**

Branch: **CSE A, B, C & D** **Version No : 2**

Year: **II** **Updated on : 26-11-2015**

Semester: **II** **No.of pages :** 154

Classification status (Unrestricted / Restricted)

Distribution List :

Prepared by :	**1) Name :**	**N.Radhika Amareshwari**	**1) Name : P. Gowtamee Radha**
	2) Sign :		**2) Sign :**
	3) Design :	**Asst.Prof**	**3) Design : Asst. Prof**
	4) Date :		**4) Date :**

Modified by :	**1) Name :**	**M.Raja Krishna Kumar**	**1) Name : K.Anusha**	**1) Name : Maninder**
	2) Sign :		**2) Sign :**	**2) Sign :**
	3) Design :	**Assoc.Prof**	**3) Design : Asst. Prof**	**3) Design : Asst. Prof**
	4) Date :		**4) Date :**	**4) Date :**

Verified by :	*** For Q.C Only.**
1) Name :M.Vamsi Krishna	**1) Name :**
2) Sign :	**2) Sign :**
3) Design : Asst. Prof	**3) Design :**
4) Date :	**4) Date :**

Approved by : (HOD) **1) Name :**

2) Sign :

3) Date :

2. Syllabus

Unit-1

Basic Computer Organization –Functions of CPU,I/O Units, Memory Instructions: Instruction Formats- One Address, two Address, Zero Address and Three Addresses and comparison; Addresing Modes with numerical examples: program Control-status bit conditions, conditional branch instructions, Program interrupts: Types of Interrupts.

Unit-II

Input-Output Organizations-I/O Interface,I/O Bus and Interface Modules:

I/O Vs memory Bus, Isolated Vs Memory -Mapped I/O, Asynchronous data transfer –Strobe Control, hand Shaking: Asynchronous Serial Transfer- Asynchronous Communication Interface, Modes of transfer-Programmed I/O, Interrupt Initiated I/O,DMA:DMA Controller, Transfer, IOP-CPU-IOP Communication, Intel 8089 IOP

Unit-III

Memory Organizations

Memory Hierarchy, Main Memory, RAM,ROM Chips, Memory Address Map, Memory Connection to CPU, Associate Memory , Cache Memory, Data Cache, Instruction Cache, Miss and Hit ratio, Access time, associative, set associative, mapping, waiting into cache, introduction to virtual memory

Unit-IV

8086 CPU Pin Diagram-Special functions of general purpose registers, segment register, concept of pipelining,8086Flag register, Addressing modes of 8086

Unit-V

8086 Instruction formats:

Assembly Language programs involving branch& Call instructions, sorting, evaluation of arithmetic expressions

3. Vision of the Department

To produce globally competent and socially responsible computer science engineers contributing to the advancement of engineering and technology which involves creativity and innovation by providing excellent learning environment with world class facilities.

4. Mission of the Department

1. To be a center of excellence in instruction, innovation in research and scholarship, and service to the stake holders, the profession, and the public.

2. To prepare graduates to enter a rapidly changing field as a competent computer science engineer.

3. To prepare graduate capable in all phases of software development, possess a firm understanding of hardware technologies, have the strong mathematical background necessary for scientific computing, and be sufficiently well versed in general theory and practice to allow growth within the discipline as it advances.

4. To prepare graduates to assume leadership roles by possessing good communication skills, the ability to work effectively as team members, and an appreciation for their social and ethical responsibility in a global setting.

5. PROGRAM EDUCATIONAL OBJECTIVES (PEOs) OF C.S.E. DEPARTMENT

1. To provide graduates with a good foundation in mathematics, sciences and engineering fundamentals required to solve engineering problems that will facilitate them to find employment in industry and / or to pursue postgraduate studies with an appreciation for lifelong learning.

2. To provide graduates with analytical and problem solving skills to design algorithms, other hardware / software systems, and inculcate professional ethics, inter-personal skills to work in a multi-cultural team.

3. To facilitate graduates to get familiarized with the art software / hardware tools, imbibing creativity and innovation that would enable them to develop cutting-edge technologies of multi-disciplinary nature for societal development.

PROGRAM OUTCOMES (CSE)

1. An ability to apply knowledge of mathematics, science and engineering to develop and analyze computing systems.
2. An ability to analyze a problem and identify and define the computing requirements appropriate for its solution under given constraints.
3. An ability to perform experiments to analyze and interpret data for different applications.
4. An ability to design, implement and evaluate computer-based systems, processes, components or programs to meet desired needs within realistic constraints of time and space.
5. An ability to use current techniques, skills and modern engineering tools necessary to practice as a CSE professional.
6. An ability to recognize the importance of professional, ethical, legal, security and social issues and addressing these issues as a professional.
7. An ability to analyze the local and global impact of systems /processes /applications/technologies on individuals, organizations, society and environment.
8. An ability to function in multidisciplinary teams.
9. An ability to communicate effectively with a range of audiences.
10. Demonstrate knowledge and understanding of the engineering, management and economic principles and apply them to manage projects as a member and leader in a team.
11. A recognition of the need for and an ability to engage in life-long learning and continuing professional development
12. Knowledge of contemporary issues.
13. An ability to apply design and development principles in producing software systems of varying complexity using various project management tools.
14. An ability to identify, formulate and solve innovative engineering problems.

6. Course Objectives

The objectives of this course are,

1. To facilitate the students learn the fundamentals of computer organization and its relevance to classical and modern problems of computer design.
2. To facilitate the students to be familiarized with the hardware components and concepts related to the input-output organization.
3. To facilitate the students to be familiarized with the hardware components and concepts related to the memory organization.
4. To facilitate the students to be familiarized with the concepts related to the 8086 micro controller like pin diagram, different types of registers and addressing modes.
5. To facilitate the students to be familiarized with the 8086 instruction formats by writing assembly language programs.

Course Outcomes

At the end of the course students will be able to

A40506.1: Describe the computer components in general and in Von Neumann Architecture in particular and their functionalities.

A40506.2: Evaluate different hardware components associated with the input-output organization of a computer.

A40506.3: Recommend instruction formats, addressing modes, interrupts, I/O & Memory bus, Isolated & Memory Mapped I/O.

A40506.4: Recommend asynchronous serial data transfer in different modes using an interface.

A40506.5 : Evaluate different hardware components associated with the memory organization of a computer.

A40506.6 : Recommend RAM, ROM, Cache memory, Virtual memory.

A40506.7 : Design and implement systems using 8086 micro controller with the knowledge of Pin Diagram, registers and instruction formats of 8086 micro controller by writing assembly language programs.

7. Mapping of Course to PEO's and PO's

Mapping of Course outcomes to Program Outcomes

Course	PEOs	POs
Computer Organization	PEO1,PEO3	PO1,PO2,PO3,PO4,PO5,PO6,PO7,PO11,PO12,PO13,PO14

S.No.	Course Outcome	POs
A40506.1:	Describe the computer components in general and in Von Neumann Architecture in particular and their functionalities.	PO1,PO2,PO4,PO7,PO11,PO12,PO14
A40506.2:	Evaluate different hardware components associated with the input-output organization of a computer.	PO1,PO2,PO3,PO4,PO5,PO7,PO11,PO12,PO14
A40506.3:	Recommend instruction formats, addressing modes, interrupts, I/O & Memory bus, Isolated & Memory Mapped I/O.	PO1,PO2,PO6,PO7,PO11,PO14
A40506.4:	Recommend asynchronous serial data transfer in different modes using an interface.	PO1,PO2,PO4,PO7,PO11,PO14
A40506.5 :	Evaluate different hardware components associated with the memory organization of a computer.	PO1,PO2,PO3,PO4,PO5,PO7,PO11,PO12,PO14
A40506.6 :	Recommend RAM, ROM, Cache memory, Virtual memory.	PO1,PO2,PO6,PO7,PO11,PO14
A40506.7 :	Design and implement systems using 8086 micro controller with the knowledge of Pin Diagram, registers and instruction formats of 8086 micro controller by writing assembly language programs.	PO1,PO2,PO3,PO4,PO5,PO6,PO7,PO11,PO13,PO14

NOTE:

- The outcomes PO08 & PO10 are obtained when the students take up a project (even though not compulsory) to build a system using 8086 micro controller by actually identifying the problem and providing the solution.
- The outcome PO09 is obtained when students give seminars, presentations about the concepts of this course or present their work (related to the course) to a range of audience.

Mapping of Course Outcomes with PO'S

Course Name COMPUTER ORGANIZATION	Program Outcomes													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
A40506.1: Describe the computer components in general and in Von Neumann Architecture in particular and their functionalities.	**2**	**2**		**1**			**2**				**2**	**1**		**2**
A40506.2: Evaluate different hardware components associated with the input-output organization of a computer.	**2**	**2**	**1**	**1**	**1**		**2**				**2**	**1**		**2**
A40506.3: Recommend instruction formats, addressing modes, interrupts, I/O & Memory bus, Isolated & Memory Mapped I/O.	**2**	**2**				**1**	**2**				**2**			**2**
A40506.4: Recommend asynchronous serial data transfer in different modes using an interface.	**2**	**2**		**1**			**2**				**2**			**2**
A40506.5: Evaluate different hardware components associated with the memory organization of a computer.	**2**	**2**		**1**	**1**		**2**				**2**	**1**		**2**
A40506.6 : Recommend RAM, ROM, Cache memory, Virtual memory.	**2**	**2**				**1**	**2**				**2**			**2**
A40506.7 : Design and implement systems using 8086 micro controller with the knowledge of Pin Diagram, registers and instruction formats of 8086 micro controller by writing assembly language programs.	**2**	**2**	**1**	**1**	**1**	**1**	**2**				**2**		**1**	**2**

8. The Importance of computer organization

Computer architecture is a specification detailing how a set of software and hardware technology standards interact to form a computer system or platform. computer architecture refers to how a computer system is designed and what technologies it is compatible with.
There are three categories of computer architecture:

- System Design: This includes all hardware components in the system, including data processors aside from the CPU, such as the graphics processing unit and direct memory access. It also includes memory controllers, data paths and miscellaneous things like multiprocessing and virtualization
- Instruction Set Architecture (ISA): This is the embedded programming language of the central processing unit. It defines the CPU's functions and capabilities based on what programming it can perform or process. This includes the word size, processor register types, memory addressing modes, data formats and the instruction set that programmers use.
- Micro architecture: Otherwise known as computer organization, this type of architecture defines the data paths, data processing and storage elements, as well as how they should be implemented in the ISA.

Course Description

Introduction to computer organization. Computer instruction set. Machine language. Data processing. Arithmetic unit: Carry look-ahead adders, Subtractors, and shifters. Logic unit. Combinational and sequential multipliers and dividers. Floating-point number representation and arithmetic. Data path design. Control unit design. Microprogramming. Pipelining. Memory Hierarchy. Memory organization, 8086 pin diagram, addressing modes and instruction formats.

What does the course offer?

- This course forms a strong foundation in the understanding and design of modern computing systems. this course explores techniques that go into designing a modern microprocessor.
- Fundamental understanding of computer architecture helps in hardware and processor design, and forms foundation for compilers, operating systems, and high performance programming.
- This course will explore how the computer architect can utilize the increasing number of transistors available to improve the performance of a processor. Focus will be given to architectures that can exploit different forms of parallelism,
- This course covers architectural techniques such as multi-issue superscalar processors, advanced caching, and multiprocessor systems.

9. Prerequisites

- Introduce students to lower level computer organization structures, e.g., to machine language structure, pipelined instruction execution as well as the implementation of processors and memory hierarchy.
- Digital Logic

- Digital Logic Design of Circuits and Systems.

10. Instructional learning outcomes

S.No	Unit	Contents	Outcomes
1	**I**	Basic Computer Organization- functions of CPU,I/O Units, Memory Instructions	• Identify Basic components of the computer and its functionality. • Analyze the Basic operations between the memory and processor. • Analyze Data transfer between the components. • Compute binary arithmetic operations. • Recognize Register transfer language notations for data transfer (basic assembly language). • Distinguish Instruction formats and different addressing modes. • Write Data transfer and manipulation notations . • Describe Instruction cycle and arithmetic and logical shift operations.
2	**II**	Input-Output organizations-I/O Interface, I/OBus and Interface modules	• Recall Basic concepts on different input and output devices . • Identify Types I/O interfaces . • Compare Different kinds of data transfers between the devices .
3	**III**	Memory Organization	• Summarize Basic concepts on memory and differences between those memories. • Apply different Cache memory mapping techniques .

			• Recall Virtual memory concepts . • Identify Different secondary storage devices . • Introduction of Virtual memory
4	**IV**	8086 CPU Pin Diagram	• Describe the circuitry of the 8086 • Describe the operation of an 8086 and observe various signals generated by the 8086 microprocessor • Introduce special functions of general purpose registers • Basic concepts on parallel and pipe line processing techniques . • Apply 8086 Flag Register • Determine different addressing modes
5	**V**	8086 Instruction Formats	• Demonstrate the ability to enter a program into the 8086 microprocessor • The architecture, I/O ports mode functions, transfer instructions, increment and decrement instructions, computer addition and subtraction, as well as arithmetic, logic, compare, jump, multiply, divide instructions are of the 8086 microprocessor are described.

11. Class Time Table

12. Individual TimeTable

13. Lesson Plan

S.No	No of periods	Topics to be covered	**Regular / Additional**	**Teaching aids used LCD/OHP/BB**
1.	2	Computer Types, Generations, Functional units	Regular	BB
2.	1	Basic operational concepts, Bus structure	Regular	BB
3.	3	Instruction Formats	Regular	BB
4.	1	Addressing modes with Examples	Regular	BB
5.	1	Program control	Regular	BB
6.	1	Status bit conditions	Regular	BB
7.	1	Conditional branch instructions	Regular	BB
8.	1	Program interrupts	Regular	BB
9.	1	Types of interrupts	Regular	BB
UNIT-II				
1.	1	I/O Interface	Regular	BB
2.	1	I/O bus and Interface modules	Regular	BB
3.	1	I/O vs Memory Bus	Regular	BB
4.	1	Isolated vs Memory-Mapped I/O	Regular	BB
5.	1	Asynchronous Data Transfer- Strobe control	Regular	BB
6.	1	Handshaking Method	Regular	BB/OHP
7.	1	Asynchronous Serial Transfer	Regular	BB
8.	1	Asynchronous Communication Interface	Regular	BB
9.	1	Modes Of Transfer-Programmed I/O	Regular	BB
10.	1	Interrupted Initiated	Regular	BB

		I/O		
11.	1	DMA-DMA Controller, Transfer	Regular	BB/OHP
12.	1	IOP-CPU-IOP Communication, intel 8089 IOP	Regular	BB/OHP
UNIT-III				
1.	1	Memory Hierarchy, Main Memory , RAM, ROM Chips	Regular	BB
2.	1	Memory Address map ,memory connection to CPU	Regular	BB
3.	2	Associative memory	Regular	BB/OHP
4.	2	Cache Memory	Regular	BB/OHP
5.	2	Types of Cache Mappings	Regular	BB/OHP
6.	3	Introduction to Virtual Memory	Regular	BB/OHP
UNIT-IV				
1.	2	8086 Pin Diagram	Regular	BB/OHP
2.	2	Special functions of General purpose registers	Regular	BB/OHP
3.	2	Concepts of Pipelining	Regular	BB/OHP
4.	2	8086 Flag Registers	Regular	BB/OHP
5.	3	Addressing modes of 8086	Regular	BB
UNIT-V				
1.	4	8086 Instruction Formats	Regular	BB/OHP
2.	4	Assembly Language Programs	Regular	BB/OHP
3.	3	Branch and Call Instructions	Regular	BB/OHP
4.	2	Sorting	Regular	BB/OHP
5.	3	Evaluation of Arithmetic Expressions	Regular	BB/OHP

Lesson Schedule For "A" Sec

S.No	No of periods	Topics to be covered	**Regular / Additional**	**Teaching aids used LCD/OHP/BB**	**Dates**
1.	2	Computer Types, Generations, Functional units	Regular	BB	
2.	1	Basic operational concepts, Bus structure	Regular	BB	
3.	3	Instruction Formats	Regular	BB	
4.	1	Addressing modes with Examples	Regular	BB	
5.	1	Program control	Regular	BB	
6.	1	Status bit conditions	Regular	BB	
7.	1	Conditional branch instructions	Regular	BB	
8.	1	Program interrupts	Regular	BB	
9.	1	Types of interrupts	Regular	BB	
UNIT-II					
1	1	I/O Interface	Regular	BB	
2.	1	I/O bus and Interface modules	Regular	BB	
3.	1	I/O vs Memory Bus	Regular	BB	
4.	1	Isolated vs Memory-Mapped I/O	Regular	BB	
5.	1	Asynchronous Data Transfer- Strobe control	Regular	BB	
6.	1	Handshaking Method	Regular	BB/OHP	
7.	1	Asynchronous Serial Transfer	Regular	BB	
8.	1	Asynchronous Communication Interface	Regular	BB	
9.	1	Modes Of Transfer-Programmed I/O	Regular	BB	
10.	1	Interrupted Initiated I/O	Regular	BB	
11.	1	DMA-DMA Controller, Transfer	Regular	BB/OHP	
12.	1	IOP-CPU-IOP Communication, intel	Regular	BB/OHP	

		8089 IOP			
UNIT-III					
1.	1	Memory Hierarchy, Main Memory , RAM, ROM Chips	Regular	BB	
2.	1	Memory Address map ,memory connection to CPU	Regular	BB	
3.	2	Associative memory	Regular	BB/OHP	
4.	2	Cache Memory	Regular	BB/OHP	
5.	2	Types of Cache Mappings	Regular	BB/OHP	
6.	3	Introduction to Virtual Memory	Regular	BB/OHP	
UNIT-IV					
6.	2	8086 Pin Diagram	Regular	BB/OHP	
7.	2	Special functions of General purpose registers	Regular	BB/OHP	
8.	2	Concepts of Pipelining	Regular	BB/OHP	
9.	2	8086 Flag Registers	Regular	BB/OHP	
10.	3	Addressing modes of 8086	Regular	BB	
UNIT-V					
1.	4	8086 Instruction Formats	Regular	BB/OHP	
2.	4	Assembly Language Programs	Regular	BB/OHP	
3.	3	Branch and Call Instructions	Regular	BB/OHP	
4.	2	Sorting	Regular	BB/OHP	
5.	3	Evaluation of Arithmetic Expressions	Regular	BB/OHP	

Total no. of Classes: 62

Lesson Schedule For "B" Sec

S.No	No of periods	Topics to be covered	**Regular / Additional**	**Teaching aids used LCD/OHP/BB**	**Dates**
1.	2	Computer Types, Generations, Functional units	Regular	BB	
2.	1	Basic operational concepts, Bus structure	Regular	BB	
3.	3	Instruction Formats	Regular	BB	
4.	1	Addressing modes with Examples	Regular	BB	
5.	1	Program control	Regular	BB	
6.	1	Status bit conditions	Regular	BB	
7.	1	Conditional branch instructions	Regular	BB	
8.	1	Program interrupts	Regular	BB	
9.	1	Types of interrupts	Regular	BB	
UNIT-II					
1.	1	I/O Interface	Regular	BB	
2.	1	I/O bus and Interface modules	Regular	BB	
3.	1	I/O vs Memory Bus	Regular	BB	
4.	1	Isolated vs Memory-Mapped I/O	Regular	BB	
5.	1	Asynchronous Data Transfer- Strobe control	Regular	BB	
6.	1	Handshaking Method	Regular	BB/OHP	
7.	1	Asynchronous Serial Transfer	Regular	BB	
8.	1	Asynchronous Communication Interface	Regular	BB	
9.	1	Modes Of Transfer-Programmed I/O	Regular	BB	
10.	1	Interrupted Initiated I/O	Regular	BB	
11.	1	DMA-DMA Controller, Transfer	Regular	BB/OHP	
12.	1	IOP-CPU-IOP Communication, intel	Regular	BB/OHP	

		8089 IOP			
UNIT-III					
1.	1	Memory Hierarchy, Main Memory , RAM, ROM Chips	Regular	BB	
2.	1	Memory Address map ,memory connection to CPU	Regular	BB	
3.	2	Associative memory	Regular	BB/OHP	
4.	2	Cache Memory	Regular	BB/OHP	
5.	2	Types of Cache Mappings	Regular	BB/OHP	
6.	3	Introduction to Virtual Memory	Regular	BB/OHP	
UNIT-IV					
1.	2	8086 Pin Diagram	Regular	BB/OHP	
2.	2	Special functions of General purpose registers	Regular	BB/OHP	
3.	2	Concepts of Pipelining	Regular	BB/OHP	
4.	2	8086 Flag Registers	Regular	BB/OHP	
5.	3	Addressing modes of 8086	Regular	BB	
UNIT-V					
1.	4	8086 Instruction Formats	Regular	BB/OHP	
2.	4	Assembly Language Programs	Regular	BB/OHP	
3.	3	Branch and Call Instructions	Regular	BB/OHP	
4.	2	Sorting	Regular	BB/OHP	
5.	3	Evaluation of Arithmetic Expressions	Regular	BB/OHP	

Total no. of Classes: 62

Lesson Schedule For "C" Sec

S.No	No of periods	Topics to be covered	**Regular / Additional**	**Teaching aids used LCD/OHP/BB**	**Dates**
1.	2	Computer Types, Generations, Functional units	Regular	BB	
2.	1	Basic operational concepts, Bus structure	Regular	BB	
3.	3	Instruction Formats	Regular	BB	
4.	1	Addressing modes with Examples	Regular	BB	
5.	1	Program control	Regular	BB	
6.	1	Status bit conditions	Regular	BB	
7.	1	Conditional branch instructions	Regular	BB	
8.	1	Program interrupts	Regular	BB	
9.	1	Types of interrupts	Regular	BB	
UNIT-II					
1	1	I/O Interface	Regular	BB	
2.	1	I/O bus and Interface modules	Regular	BB	
3.	1	I/O vs Memory Bus	Regular	BB	
4.	1	Isolated vs Memory-Mapped I/O	Regular	BB	
5.	1	Asynchronous Data Transfer- Strobe control	Regular	BB	
6.	1	Handshaking Method	Regular	BB/OHP	
7.	1	Asynchronous Serial Transfer	Regular	BB	
8.	1	Asynchronous Communication Interface	Regular	BB	
9.	1	Modes Of Transfer-Programmed I/O	Regular	BB	
10.	1	Interrupted Initiated I/O	Regular	BB	
11.	1	DMA-DMA Controller, Transfer	Regular	BB/OHP	
12.	1	IOP-CPU-IOP Communication, intel	Regular	BB/OHP	

		8089 IOP			
UNIT-III					
1.	1	Memory Hierarchy, Main Memory , RAM, ROM Chips	Regular	BB	
2.	1	Memory Address map ,memory connection to CPU	Regular	BB	
3.	2	Associative memory	Regular	BB/OHP	
4.	2	Cache Memory	Regular	BB/OHP	
5.	2	Types of Cache Mappings	Regular	BB/OHP	
6.	3	Introduction to Virtual Memory	Regular	BB/OHP	
UNIT-IV					
6.	2	8086 Pin Diagram	Regular	BB/OHP	
7.	2	Special functions of General purpose registers	Regular	BB/OHP	
8.	2	Concepts of Pipelining	Regular	BB/OHP	
9.	2	8086 Flag Registers	Regular	BB/OHP	
10.	3	Addressing modes of 8086	Regular	BB	
UNIT-V					
1.	4	8086 Instruction Formats	Regular	BB/OHP	
2.	4	Assembly Language Programs	Regular	BB/OHP	
3.	3	Branch and Call Instructions	Regular	BB/OHP	
4.	2	Sorting	Regular	BB/OHP	
5.	3	Evaluation of Arithmetic Expressions	Regular	BB/OHP	

Total no. of Classes: 62

Lesson Schedule For "D" Sec

S.No	No of periods	Topics to be covered	**Regular / Additional**	**Teaching aids used LCD/OHP/BB**	**Dates**
1.	2	Computer Types, Generations, Functional units	Regular	BB	
2.	1	Basic operational concepts, Bus structure	Regular	BB	
3.	3	Instruction Formats	Regular	BB	
4.	1	Addressing modes with Examples	Regular	BB	
5.	1	Program control	Regular	BB	
6.	1	Status bit conditions	Regular	BB	
7.	1	Conditional branch instructions	Regular	BB	
8.	1	Program interrupts	Regular	BB	
9.	1	Types of interrupts	Regular	BB	
UNIT-II					
1	1	I/O Interface	Regular	BB	
2.	1	I/O bus and Interface modules	Regular	BB	
3.	1	I/O vs Memory Bus	Regular	BB	
4.	1	Isolated vs Memory-Mapped I/O	Regular	BB	
5.	1	Asynchronous Data Transfer- Strobe control	Regular	BB	
6.	1	Handshaking Method	Regular	BB/OHP	
7.	1	Asynchronous Serial Transfer	Regular	BB	
8.	1	Asynchronous Communication Interface	Regular	BB	
9.	1	Modes Of Transfer-Programmed I/O	Regular	BB	
10.	1	Interrupted Initiated I/O	Regular	BB	
11.	1	DMA-DMA Controller, Transfer	Regular	BB/OHP	
12.	1	IOP-CPU-IOP Communication, intel	Regular	BB/OHP	

		8089 IOP			
UNIT-III					
1.	1	Memory Hierarchy, Main Memory , RAM, ROM Chips	Regular	BB	
2.	1	Memory Address map ,memory connection to CPU	Regular	BB	
3.	2	Associative memory	Regular	BB/OHP	
4.	2	Cache Memory	Regular	BB/OHP	
5.	2	Types of Cache Mappings	Regular	BB/OHP	
6.	3	Introduction to Virtual Memory	Regular	BB/OHP	
UNIT-IV					
11.	2	8086 Pin Diagram	Regular	BB/OHP	
12.	2	Special functions of General purpose registers	Regular	BB/OHP	
13.	2	Concepts of Pipelining	Regular	BB/OHP	
14.	2	8086 Flag Registers	Regular	BB/OHP	
15.	3	Addressing modes of 8086	Regular	BB	
UNIT-V					
1.	4	8086 Instruction Formats	Regular	BB/OHP	
2.	4	Assembly Language Programs	Regular	BB/OHP	
3.	3	Branch and Call Instructions	Regular	BB/OHP	
4.	2	Sorting	Regular	BB/OHP	
5.	3	Evaluation of Arithmetic Expressions	Regular	BB/OHP	

Total no. of Classes: 62

Detailed Notes

Unit-I

Basic Structure of Computers

Computer Architecture in general covers three aspects of computer design namely: Computer Hardware, Instruction set Architecture and Computer Organization.

Computer hardware consists of electronic circuits, displays, magnetic and optical storage media and communication facilities.

Instruction set Architecture is programmer visible machine interface such as instruction set, registers, memory organization and exception handling. Two main approaches are mainly CISC (Complex Instruction Set Computer) and RISC (Reduced Instruction Set Computer)

Computer Organization includes the high level aspects of a design, such as memory system, the bus structure and the design of the internal CPU.

Computer Types

Computer is a fast electronic calculating machine which accepts digital input, processes it according to the internally stored instructions (Programs) and produces the result on the output device. The internal operation of the computer can be as depicted in the figure below:

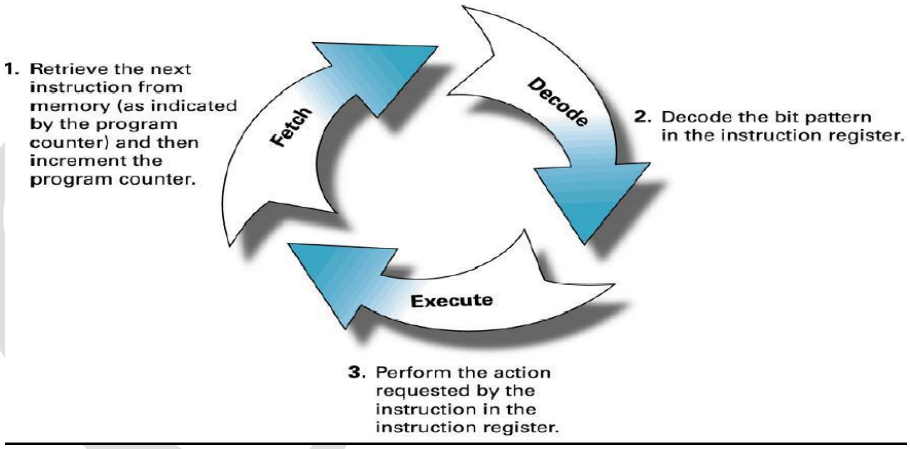


Figure 1: Fetch, Decode and Execute steps in a Computer System

The computers can be classified into various categories as given below:

- Micro Computer
- Laptop Computer
- Work Station
- Super Computer
- Main Frame
- Hand Held
- Multi core

Micro Computer: A personal computer; designed to meet the computer needs of an individual. Provides access to a wide variety of computing applications, such as word processing, photo editing, e-mail, and internet.

Laptop Computer: A portable, compact computer that can run on power supply or a battery unit. All components are integrated as one compact unit. It is generally more expensive than a comparable desktop. It is also called a Notebook.

Work Station: Powerful desktop computer designed for specialized tasks. Generally used for tasks that requires a lot of processing speed. Can also be an ordinary personal computer attached to a LAN (local area network).

Super Computer: A computer that is considered to be fastest in the world. Used to execute tasks that would take lot of time for other computers. For Ex: Modeling weather systems, genome sequence, etc (Refer site: http://www.top500.org/)

Main Frame: Large expensive computer capable of simultaneously processing data for hundreds or thousands of users. Used to store, manage, and process large amounts of data that need to be reliable, secure, and centralized.

Hand Held: It is also called a PDA (Personal Digital Assistant). A computer that fits into a pocket, runs on batteries, and is used while holding the unit in your hand. Typically used as an appointment book, address book, calculator and notepad.

Multi Core: Have Multiple Cores – parallel computing platforms. Many Cores or computing elements in a single chip. Typical Examples: Sony Play station, Core2Duo,i3,i7etc

Functional Unit

A computer in its simplest form comprises five functional units namely input unit, output unit memory unit, arithmetic & logic unit and control unit. Figure 2 depicts the functional units of a computer system.

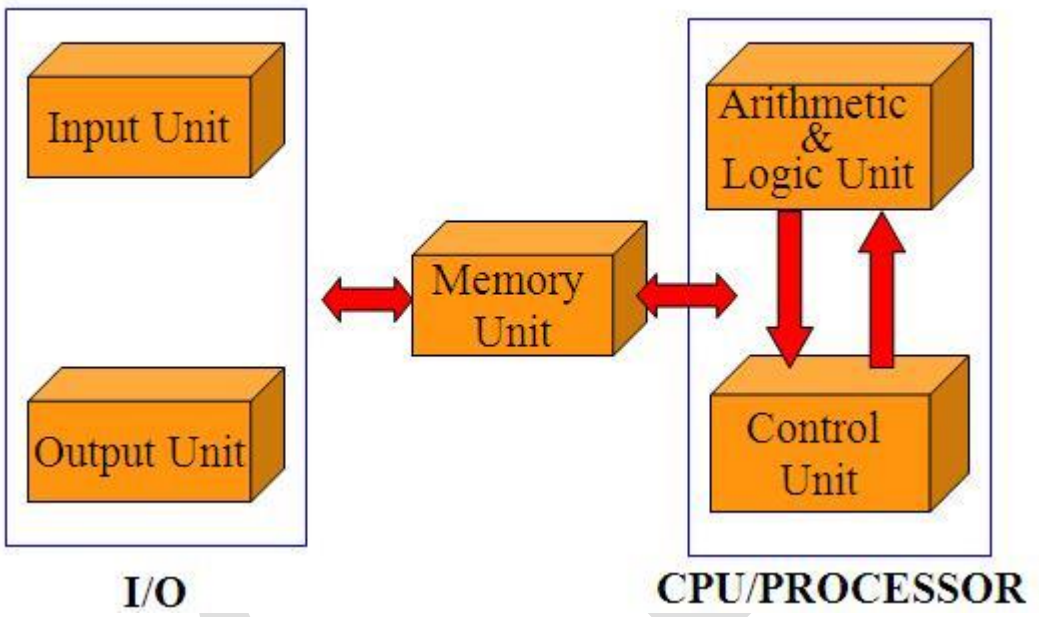


Figure 2: Basic functional units of a computer

Let us discuss about each of them in brief:

1. **Input Unit:** Computer accepts encoded information through input unit. The standard input device is a keyboard. Whenever a key is pressed, keyboard controller sends the code to CPU/Memory.

   Examples include Mouse, Joystick, Tracker ball, Light pen, Digitizer, Scanner etc.

2. **Memory Unit:** Memory unit stores the program instructions (Code), data and results of computations etc. Memory unit is classified as:

   - Primary /Main Memory
   - Secondary
   - Memory/Auxiliary

Primary memory is a semiconductor memory that provides access at high speed. Run time program instructions and operands are stored in the main memory. Main memory is classified again as ROM and RAM. ROM holds system programs and firmware routines such as BIOS, POST, I/O Drivers that are essential to manage the hardware of a computer. RAM is termed as Read/Write memory or user memory that holds run time program instruction and data. While primary storage is essential, it is volatile in nature and expensive. Additional requirement of memory could be supplied as auxiliary memory at cheaper cost. **Secondary memories** are non volatile in nature.

3. **Arithmetic and logic unit:** ALU consist of necessary logic circuits like adder, comparator etc., to perform operations of addition, multiplication, comparison of two numbers etc.

4. **Output Unit:** Computer after computation returns the computed results, error messages, etc. via output unit. The standard output device is a video monitor, LCD/TFT monitor. Other output devices are printers, plotters etc.

5. **Control Unit:** Control unit co-ordinates activities of all units by issuing control signals. Control signals issued by control unit govern the data transfers and then appropriate operations take place. Control unit interprets or decides the operation/action to be performed.

The operations of a computer can be summarized as follows:

1. A set of instructions called a program reside in the main memory of computer.

2. The CPU fetches those instructions sequentially one-by-one from the main memory, decodes them and performs the specified operation on associated data operands in ALU.

3. Processed data and results will be displayed on an output unit.

4. All activities pertaining to processing and data movement inside the computer machine are governed by control unit.

Basic Operational Concepts

An Instruction consists of two parts, an Operation code and operand/s as shown below:

OPCODE	OPERAND/s

Let us see a typical instruction

ADD LOCA, R0

This instruction is an addition operation. The following are the steps to execute the instruction:

Step 1: Fetch the instruction from main memory into the processor

Step 2: Fetch the operand at location LOCA from main memory into the processor

Step 3: Add the memory operand (i.e. fetched contents of LOCA) to the contents of register R0 Step 4: Store the result (sum) in R0.

The same instruction can be realized using two instructions as

Load LOCA, R1
Add R1, R0

The steps to execute the instructions can be enumerated as below:

Step 1: Fetch the instruction from main memory into the processor

Step 2: Fetch the operand at location LOCA from main memory into the processor Register R1

Step 3: Add the content of Register R1 and the contents of register R0

Step 4: Store the result (sum) in R0.

Figure 3 below shows how the memory and the processor are connected. As shown in the diagram, in addition to the ALU and the control circuitry, the processor contains a number of registers used for several different purposes. The instruction register holds the instruction that is currently being executed. The program counter keeps track of the execution of the program. It contains the memory address of the next instruction to be fetched and executed. There are n general purpose registers R0 to R_{n-1} which can be used by the programmers during writing programs.

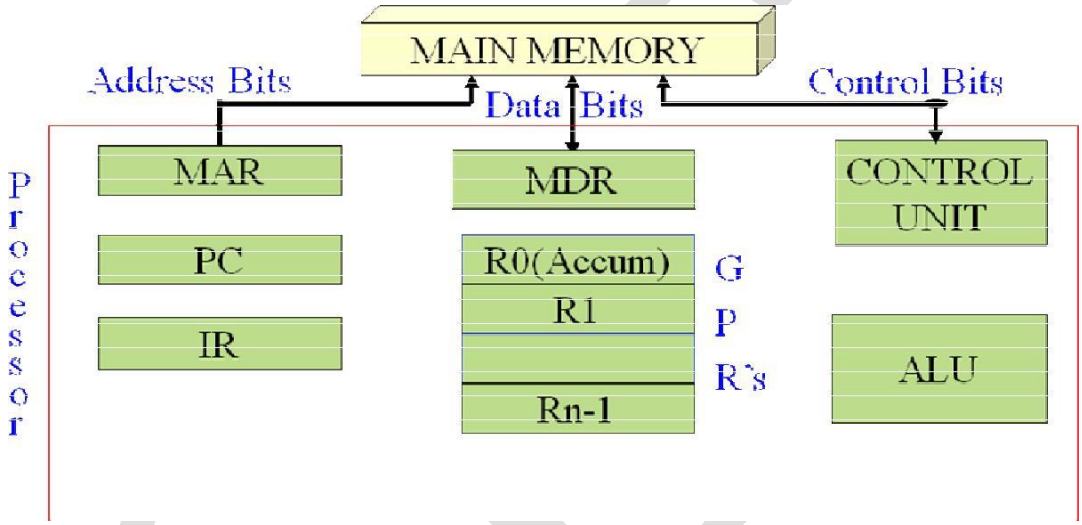


Figure 3: Connections between the processor and the memory

The interaction between the processor and the memory and the direction of flow of information is as shown in the diagram below:

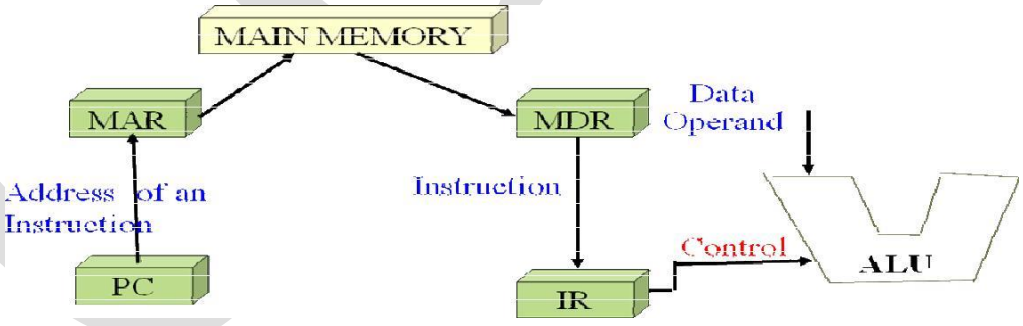


Figure 4: Interaction between the memory and

The most common fields found in instruction format are:-

(1) An operation code field that specified the operation to be performed
(2) An address field that designates a memory address or a processor registers.

(3) A mode field that specifies the way the operand or the effective address is determined.

Computers may have instructions of several different lengths containing varying number of addresses. The number of address field in the instruction format of a computer depends on the internal organization of its registers. Most computers fall into one of three types of CPU organization.

(1) Single Accumulator organization ADD X AC ® AC + M [×]
(2) General Register Organization ADD R1, R2, R3 R ® R2 + R3
(3) Stack Organization PUSH X

Three address Instruction

Computer with three addresses instruction format can use each address field to specify either processor register are memory operand.

ADD R1, A, B A1 ® M [A] + M [B]
ADD R2, C, D R2 ® M [C] + M [B] X = (A + B) * (C + A)
MUL X, R1, R2 M [X] R1 * R2

The advantage of the three address formats is that it results in short program when evaluating arithmetic expression. The disadvantage is that the binary-coded instructions require too many bits to specify three addresses.

Two Address Instruction

Most common in commercial computers. Each address field can specify either a processes register on a memory word.

MOV R1, A R1 ® M [A]
ADD R1, B R1 ® R1 + M [B]
MOV R2, C R2 ® M [C] X = (A + B) * (C + D)
ADD R2, D R2 ® R2 + M [D]
MUL R1, R2 R1 ® R1 * R2
MOV X1 R1 M [X] ® R1

One Address instruction

It used an implied accumulator (AC) register for all data manipulation. For multiplication/division, there is a need for a second register.

```
LOAD    A          AC ® M [A]
ADD     B          AC ® AC + M [B]
STORE T            M [T] ® AC          X = (A +B) × (C + A)
```

All operations are done between the AC register and a memory operand. It's the address of a temporary memory location required for storing the intermediate result.

```
 LOAD    C          AC ® M (C)
ADD      D          AC ® AC + M (D)
ML       T          AC ® AC + M (T)
STORE  X           M [×]® AC
```

Zero – Address Instruction

A stack organized computer does not use an address field for the instruction ADD and MUL. The PUSH & POP instruction, however, need an address field to specify the operand that communicates with the stack (TOS ® top of the stack)

```
PUSH      A      TOS ® A
PUSH      B      TOS ® B
ADD              TOS ®  (A + B)
PUSH      C      TOS ® C
PUSH      D      TOS ® D
ADD              TOS ® (C + D)
MUL              TOS ® (C + D) * (A + B)
POP       X      M [X] TOS
```

CISC Characteristics

A computer with large number of instructions is called complex instruction set computer or CISC. Complex instruction set computer is mostly used in scientific computing applications requiring lots of floating point arithmetic.

1. A large number of instructions - typically from 100 to 250 instructions.
2. Some instructions that perform specialized tasks and are used infrequently.
3. A large variety of addressing modes - typically 5 to 20 different modes.
4. Variable-length instruction formats
5. Instructions that manipulate operands in memory.

RISC Characteristics

A computer with few instructions and simple construction is called reduced instruction set computer or RISC. RISC architecture is simple and efficient. The major characteristics of RISC architecture are,

1. Relatively few instructions
2. Relatively few addressing modes
3. Memory access limited to load and store instructions
4. All operations are done within the registers of the CPU
5. Fixed-length and easily-decoded instruction format.
6. Single cycle instruction execution
7. Hardwired and micro programmed control

Addressing Modes

The operation field of an instruction specifies the operation to be performed. This operation must be executed on some data stored in computer register as memory words. The way the operands are chosen during program execution is dependent on the addressing mode of the instruction. The addressing mode specifies a rule for interpreting or modifying the address field of the instruction between the operand is activity referenced. Computer use addressing mode technique for the purpose of accommodating one or both of the following provisions.

(1) To give programming versatility to the uses by providing such facilities as pointer to memory, counters for top control, indexing of data, and program relocation.
(2) To reduce the number of bits in the addressing fields of the instruction.

The basic operation cycle of the computer

(1) Fetch the instruction from memory
(2) Decode the instruction
(3) Execute the instruction

Program Counter (PC) keeps track of the instruction in the program stored in memory. PC holds the address of the instruction to be executed next and in incremented each time an instruction is fetched from memory.

Addressing Modes: The most common addressing techniques are
• Immediate
• Direct
• Indirect
• Register
• Register Indirect
• Displacement
• Stack

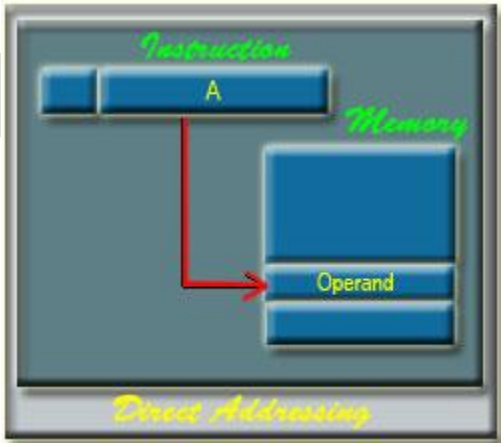
All computer architectures provide more than one of these addressing modes. The question arises as to how the control unit can determine which addressing mode

is being used in a particular instruction. Several approaches are used. Often, different opcodes will use different addressing modes. Also, one or more bits in the instruction format can be used as a mode field. The value of the mode field determines which addressing mode is to be used.
What is the interpretation of effective address. In a system without virtual memory, the effective address will be either a main memory address or a register. In a virtual memory system, the effective address is a virtual address or a register. The actual mapping to a physical address is a function of the paging mechanism and is invisible to the programmer.

Opcode	Mode	Address

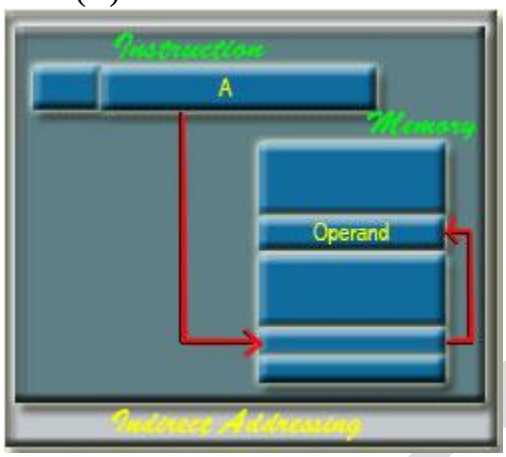
Immediate Addressing:
The simplest form of addressing is immediate addressing, in which the operand is actually present in the instruction:
OPERAND = A
This mode can be used to define and use constants or set initial values of variables. The advantage of immediate addressing is that no memory reference other than the instruction fetch is required to obtain the operand. The disadvantage is that the size of the number is restricted to the size of the address field, which, in most instruction sets, is small compared with the world length.

Direct Addressing:
A very simple form of addressing is direct addressing, in which the address field contains the effective address of the operand:
EA = A
It requires only one memory reference and no special calculation.

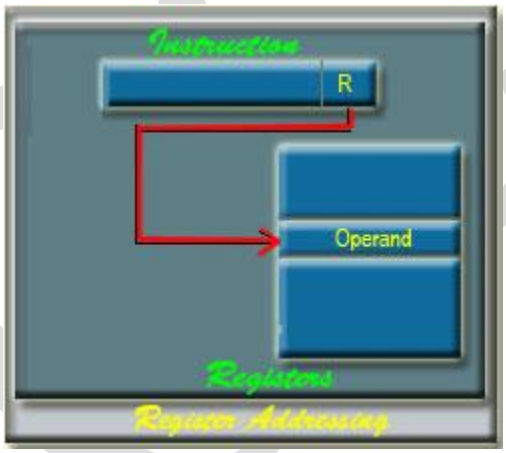


Indirect Addressing:

With direct addressing, the length of the address field is usually less than the word length, thus limiting the address range. One solution is to have the address field refer to the address of a word in memory, which in turn contains a full-length address of the operand. This is know as indirect addressing:
EA = (A)



Register Addressing:
Register addressing is similar to direct addressing. The only difference is that the address field refers to a register rather than a main memory address:
EA = R
The advantages of register addressing are that only a small address field is needed in the instruction and no memory reference is required.
The disadvantage of register addressing is that the address space is very limited.

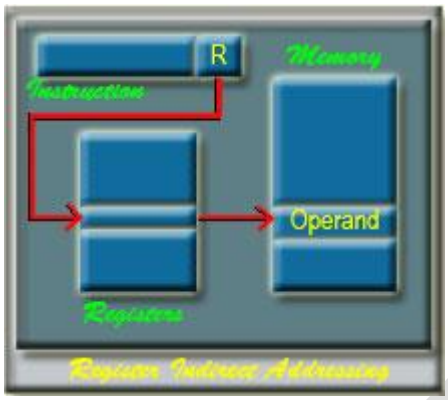


The exact register location of the operand in case of Register Addressing Mode is shown in the Figure 34.4. Here, 'R' indicates a register where the operand is present.

Register Indirect Addressing:
Register indirect addressing is similar to indirect addressing, except that the address field refers to a register instead of a memory location. It requires only one memory reference and no special calculation.

EA = (R)

Register indirect addressing uses one less memory reference than indirect addressing. Because, the first information is available in a register which is nothing but a memory address. From that memory location, we use to get the data or information. In general, register access is much more faster than the memory access.



Displacement Addressing:

A very powerful mode of addressing combines the capabilities of direct addressing and register indirect addressing, which is broadly categorized as displacement addressing:

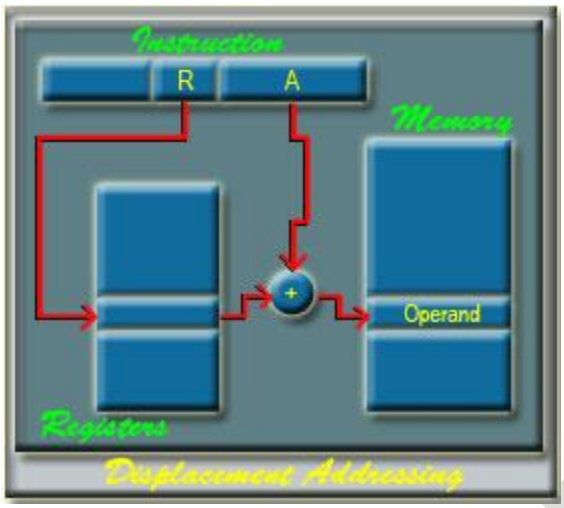
EA = A + (R)

Displacement addressing requires that the instruction have two address fields, at least one of which is explicit. The value contained in one address field (value = A) is used directly. The other address field, or an implicit reference based on opcode, refers to a register whose contents are added to A to produce the effective address.

The general format of Displacement Addressing is shown in the Figure 4.6.

Three of the most common use of displacement addressing are:

- Relative addressing
- Base-register addressing
- Indexing



Relative Addressing:
For relative addressing, the implicitly referenced register is the program counter (PC). That is, the current instruction address is added to the address field to produce the EA. Thus, the effective address is a displacement relative to the address of the instruction.

Base-Register Addressing:
The reference register contains a memory address, and the address field contains a displacement from that address. The register reference may be explicit or implicit. In some implementation, a single segment/base register is employed and is used implicitly. In others, the programmer may choose a register to hold the base address of a segment, and the instruction must reference it explicitly.

Indexing:
The address field references a main memory address, and the reference register contains a positive displacement from that address. In this case also the register reference is sometimes explicit and sometimes implicit.
Generally index register are used for iterative tasks, it is typical that there is a need to increment or decrement the index register after each reference to it. Becausethis is such a common operation, some system will automatically do this as part of the same instruction cycle.
This is known as auto-indexing. We may get two types of auto-indexing: -one is auto-incrementing and the other one is -auto-decrementing.
If certain registers are devoted exclusively to indexing, then auto-indexing can be invoked implicitly and automatically. If general purpose register are used, the auto index operation may need to be signaled by a bit in the instruction.
Auto-indexing using increment can be depicted as follows:
EA = A + (R)
R = (R) + 1
Auto-indexing using decrement can be depicted as follows:
EA = A + (R)
R = (R) - 1
In some machines, both indirect addressing and indexing are provided, and it

is possible to employ both in the same instruction. There are two possibilities: The indexing is performed either before or after the indirection.
If indexing is performed after the indirection, it is termed post indexing
EA = (A) + (R)
First, the contents of the address field are used to access a memory location containing an address. This address is then indexed by the register value.
With pre indexing, the indexing is performed before the indirection:
EA = (A + (R)
An address is calculated, the calculated address contains not the operand, but the address of the operand.

Stack Addressing:
A stack is a linear array or list of locations. It is sometimes referred to as a pushdown list or last-in-first-out queue. A stack is a reserved block of locations. Items are appended to the top of the stack so that, at any given time, the block is partially filled. Associated with the stack is a pointer whose value is the address of the top of the stack. The stack pointer is maintained in a register. Thus, references to stack locations in memory are in fact register indirect addresses.
The stack mode of addressing is a form of implied addressing. The machine instructions need not include a memory reference but implicitly operate on the top of the stack.

Introduction about Program Control:-

A program that enhances an operating system by creating an environment in which you can run other programs. Control programs generally provide a graphical interface and enable you to run several programs at once in different windows.

Control programs are also called *operating environments*.

The program control functions are used when a series of conditional or unconditional jump and return instruction are required. These instructions allow the program to execute only certain sections of the control logic if a fixed set of logic conditions are met. The most common instructions for the program control available in most controllers are described in this section.

Introduction About status bit register:-

A **status register**, **flag register**, or **condition code register** is a collection of status flag bits for a processor. An example is the FLAGS register of the computer architecture. The flags might be part of a larger register, such as a program status word (PSW) register.

The status register is a hardware register which contains information about the state of the processor. Individual bits are implicitly or explicitly read and/or written by the machine code instructions executing on the processor. The status register in a traditional processor design includes at least three central flags: Zero, Carry, and Overflow, which are set or cleared automatically as effects of arithmetic and bit manipulation operations. One or more of the flags may then be read by a subsequent conditional jump instruction (including conditional calls, returns, etc. in some machines) or by some arithmetic, shift/rotate or bitwise operation, typically using the carry flag as input in addition to any explicitly given operands. There are also processors where other classes of instructions may read or write the fundamental zero, carry or overflow flags, such as block-, string- or dedicated input/output instructions, for instance.

Some CPU architectures, such as the MIPS and Alpha, do not use a dedicated flag register. Others do not implicitly set and/or read flags. Such machines either do not pass *implicit* status information between instructions at all, or do they pass it in a explicitly selected general purpose register.

A status register may often have other fields as well, such as more specialized flags, interrupt enable bits, and similar types of information. During an interrupt, the status of the thread currently executing can be preserved (and later recalled) by storing the current value of the status register along with the program counter and other active registers into the machine stack or some other reserved area of memory.

Common flags:-

This is a list of the most common CPU status register flags, implemented in almost all modern processors.

Flag	Name	Description
Z	Zero flag	Indicates that the result of an arithmetic or logical operation (or, sometimes, a load) was zero.
C	Carry flag	Enables numbers larger than a single word to be added/subtracted by carrying a binary digit from a less significant word to the least significant bit of a more significant word as needed. It is also used to extend bit shifts and rotates in a similar manner on many processors (sometimes done via a dedicated **X** flag).
S / **N**	Sign flag Negative flag	Indicates that the result of a mathematical operation is negative. In some processors, the N and S flags are distinct with different meanings and usage: One indicates whether the last result was negative whereas the other indicates whether a subtraction or addition has taken place.
V / **O** / **W**	Overflow flag	Indicates that the signed result of an operation is too large to fit in the register width using twos complement representation.

Introduction about Conditional branch instruction:-

Conditional branch instruction:-

Conditional branch instruction is the branch instruction bit and BR instruction is the Program control instruction.

The conditional Branch Instructions are listed as Bellow:-

Mnemonics	Branch Instruction	Tested control
BZ	Branch if Zero	Z=1
BNZ	Branch if not Zero	Z=0
BC	Branch if Carry	C=1
BNC	Branch if not Carry	C=0
BP	Branch if Plus	S=0
BM	Branch if Minus	S=1
BV	Branch if Overflow	V=1
BNV	Branch if not Overflow	V=0

Unsigned Compare(A-B):-

Mnemonics	Branch Instruction	Tested control
BHI	Branch if Higher	A > B
BHE	Branch if Higher or Equal	A >= B
BLO	Branch if Lower	A < B
BLE	Branch if Lower or Equal	A <= B
BE	Branch if Equal	A=B
BNE	Branch if not Equal	A not = B

Signed Compare(A-B):-

Mnemonics	Branch Instruction	Tested control
BGT	Branch if Greater Than	A > B
BGE	Branch if Greater Than or Equal	A >= B
BLT	Branch if Less Than	A < B
BLE	Branch if Less Than or Equal	A <= B
BE	Branch if Equal	A=B
BNE	Branch if not Equal	A not = B

Conditional Branch instruction are represented with the help of mnemonics. Each Mnemonic is constructed with B (Branch) and abbreviation of condition name.

For Example:-

BC ----> Branch if Carry

If condition state is used for the Negative than N is inserted to define the Zero state i.e.

BNC----> Branch if Not Carry

If tested condition is true Program control is transfer to the address specified by instruction. If the tested condition is false than control continuous with instruction that follows.

Introduction about program interrupt:-

When a Process is executed by the CPU and when a user Request for another Process then this will create disturbance for the Running Process. This is also called as the **Interrupt**.

Interrupts can be generated by User, Some Error Conditions and also by Software's and the hardware's. But CPU will handle all the Interrupts very carefully because when Interrupts are generated then the CPU must handle all the Interrupts Very carefully means the CPU will also Provides Response to the Various Interrupts those are generated. So that When an interrupt has Occurred then the CPU will handle by using the Fetch, decode and Execute Operations.

Interrupts allow the operating system to take notice of an external event, such as a mouse click. Software interrupts, better known as exceptions, allow the OS to handle unusual events like divide-by-zero errors coming from code execution.

The sequence of events is usually like this:

1. Hardware signals an interrupt to the processor
2. The processor notices the interrupt and suspends the currently running software
3. The processor jumps to the matching interrupt handler function in the OS
4. The interrupt handler runs its course and returns from the interrupt
5. The processor resumes where it left off in the previously running software

The most important interrupt for the operating system is the timer tick interrupt. The timer tic interrupt allows the OS to periodically regain control from the currently running user process. The OS can then decide to schedule another process, return back to the same process, do housekeeping, etc. The timer tick interrupt provides the foundation for the concept of preemptive multitasking.

Types of Interrupts

Generally there are three types of Interrupts those are Occurred For Example

1) Internal Interrupt

2) External Interrupt.

3) Software Interrupt.

1. Internal Interrupt:-

• When the hardware detects that the program is doing something wrong, it will usually generate an interrupt usually generate an interrupt.
– Arithmetic error - Invalid Instruction
– Addressing error - Hardware malfunction
– Page fault - Debugging
• A Page Fault interrupt is not the result of a program error, but it does require the operating system to get control.
• Internal interrupts are sometimes called exceptions

The Internal Interrupts are those which are occurred due to Some Problem in the Execution For Example When a user performing any Operation which contains any Error and which contains any type of Error. So that Internal Interrupts are those which are occurred by the Some Operations or by Some Instructions and the Operations those are not Possible but a user is trying for that Operation. And The Software Interrupts are those which are made some call to the System for Example while we are Processing Some Instructions and when we want to Execute one more Application Programs.

2. External Interrupt:-

• I/O devices tell the CPU that an I/O request has completed by sending an interrupt signal to the processor.
• I/O errors may also generate an interrupt.
• Most computers have a timer which
interrupts the CPU every so many interrupts the CPU every so many milliseconds.

The External Interrupt occurs when any Input and Output Device request for any Operation and the CPU will Execute that instructions first For Example When a Program is executed and when we move the Mouse on the Screen then the CPU will handle this External interrupt first and after that he will resume with his Operation.

3.Software interrupts:-

These types if 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.

A software interrupt occurs when the processor executes an INT instruction. Written in the program, typically used to invoke a system service.

A processor interrupt is caused by an electrical signal on a processor pin. Typically used by devices to tell a driver that they require attention. The clock tick interrupt is very common, it wakes up the processor from a halt state and allows the scheduler to pick other work to perform.

A processor fault like access violation is triggered by the processor itself when it encounters a condition that prevents it from executing code. Typically when it tries to read or write from unmapped memory or encounters an invalid instruction.

Unit 2

Introduction about Input Output Organization:-

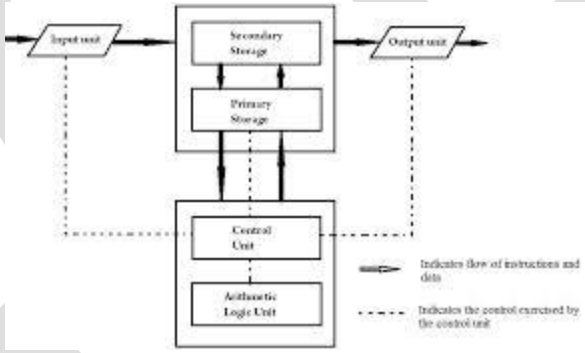
Input Output Organization:

I/O operations are accomplished through external devices that provide a means of exchanging data between external environment and computer. An external device attaches to the computer by a link to an I/O module. An external device linked to an I/O module is called peripheral device or peripheral. The figure below shows attachment of external devices through I/O module.

External Devices can be categorized as

1. Human readable: suitable for communicating with computer user. For example - video display terminals and printers.
2. Machine readable: suitable for communicating with equipment. For example - sensor, actuators used in robotics application.
3. Communication: suitable for communicating with remote devices. They may be human readable device such as terminal and machine readable device such as another computer.

Block diagram of external device is described below.



1. The interface to I/O module: The interface to I/O module is in the form of

a) Control Signal – determines the function that the device will perform. E.g. send data to I/O module (READ or INPUT), receive data from I/O module (WRITE or OUTPUT), report status or perform some control function such as position a disk head. b) Data Signal – send or receive the data from I/O module. c) Status Signal – it indicates the status of signal. E.g. READY/NOT READY

1. Control Logic: associated with the device controls on specific operation as directed from I/O module.

2. Transducer: converts the data from electrical to other form of energy during output and from other forms of electrical during input.
3. Buffer: is associated with transducer to temporarily hold data during data transmission from I/O module and external environment. Buffer size of 8 to 16 bits is common.

Introduction about input-output interface:-

An I/O interface is required whenever the I/O device is driven by the processor. The interface must have necessary logic to interpret the device address generated by the processor. Handshaking should be implemented by the interface using appropriate commands (like BUSY, READY, and WAIT), and the processor can communicate with an I/O device through the interface.

It would not be practical for every I/O device to be wired to the computer in a different way, so we must have a scheme where the hardware connections are fixed, and yet the communication with the device is flexible, so that the widely varying needs of devices can all be met.

An I/O device, from the viewpoint of the CPU, is a set of registers. The CPU communicates with and controls the I/O device by reading and writing these registers. For example, SPIM, the MIPS simulator, uses two registers to communicate with the keyboard.

- The keyboard data register contains the ASCII code of the last key pressed.
- The keyboard control register indicates when a new key has been pressed. If bit 0 is one, a key has been pressed since the last character was read. The keyboard controller sets this bit when a key is pressed. It clears this bit when the keyboard data register is read.

The CPU can find out whether a new character is available by reading the keyboard control register and testing bit 0. If bit 0 is 1, it then reads the keyboard data register to get the new key.

Accessing I/O devices at the hardware level is a lot like accessing memory. The registers in the I/O devices are connected to the CPU using buses. We need an address bus to specify which I/O device register is to be accessed. We need control lines to specify what kind of access is desired (read, write, reset, etc.) Finally, we need a data bus to transfer the data between the CPU and the device.

Each device has one or more control, status, and data registers at various I/O addresses. A hypothetical example:

Address	Register
ff00	keyboard status
ff01	keyboard data

ff02 display status
ff03 display data
ff04 disk status
ff05 disk block address
ff06 disk block size
ff07 disk data address
...

I/O read and write operations can be more complex than memory read and write operations, but the basic idea is the same. I/O control generally involves more than just read and write control lines. In a sense, memory can be viewed as a very simple, fast I/O device.

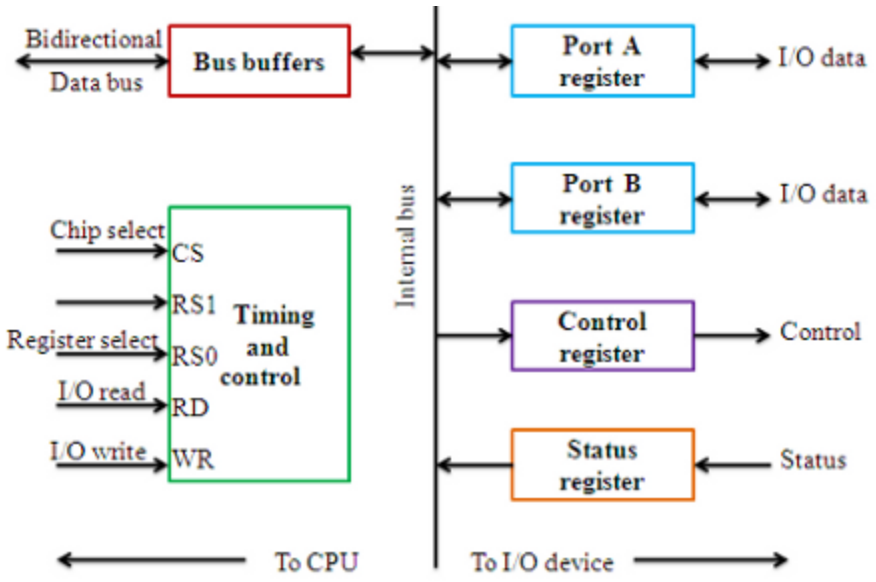
Whereas memory is just a large pool of slow, inexpensive registers for storing data, each I/O device register has a unique purpose in controlling a specific I/O device. This does not affect how the CPU accesses them at the hardware level, but it does affect how they are used by software.

Simple device control, such as stating whether an I/O register is to be read or written, can be done over the control lines. More complex devices are often controlled by sending special data blocks called *Peripheral Control Blocks (PCBs)* over the data lines. This is the primary method for communicating with disk drives, for example.

Since I/O devices are of a very different nature than CPU circuits, there must be interface hardware to connect each device to the CPU.

Example of I/O Interface

An example of an I/O interface unit is shown in figure. It consists of two data registers called ports, a control register, a status register, bus buffers and timing and control circuits.



The four registers communicate directly with the I/O device attached to the interface. The I/O data to and from the device can be transferred into either port A or port B. Port A may be defined as an input port and port B may be defined as an output port. The output device such as magnetic disk transfers data in both directions. So bidirectional data bus is used. CPU gives control information to control register. The bits in the status register are used for status conditions. It is also used for recording errors that may occur during the data transfer. The bus buffers use the bidirectional data bus to communicate with the CPU. A timing and control circuit is used to detect the address assigned to the bus buffers.

CS	RS1	RS0	Register selected
0	X	X	None: data bus in high-impedance
1	0	0	Port A register
1	0	1	Port B register
1	1	0	Control register
1	1	1	Status register

There are basically three type of input-output interfaces. These are as:-

1. I/O bus and interface modules,.

2. I/O versus memory bus.

3. isolated versus memory-mapped I/O.

Introduction About Input-Output Bus And Interface Module:-

The processor of computer is communicate with several peripheral devices such as keyboard, VDU, Printer, magnetic disk, magnetic tape, etc.

Each peripheral device has its own interface . Each interface communicate with i/o bus. The communication link between processor and peripherals is shown as below:-

Each interface decode addresses and control receive from input-output bus and interpret them for peripherals and provide signal for peripheral controller . It synchronize data flow at supervise the transfer between peripherals and CPU. Each peripheral has its own controller.

For example:- Printer controller control the paper motion , the printing time and selection of printing characters.

The input-output bus fro the processor is attached to all peripheral interfaces.

The input-output bus three lines:

1. Data line

2. Address line.

3. Control line.

1. Data line:-Data line of input-output bus carry the data to and from the peripherals.

1. Address line:-Address line contain the address of data and instructions.

1. Control line:-It contain control instructions in the form of function and input-output command. These command control instruction are of four types:-

1. Control Command

2. Status Command

3. Data output Command

4. Data input Command

1. Control Command:-A control command is issue to activate the peripheral and to inform it what to do.

2.Status Command:-A Status command is used to test the various status condition in the interface and the peripheral.

3.Data output Command:-A Data output command is responsible for transfering the data from the bus into peripherals.

3.Data output Command:-A Data output command is responsible for transfering the data from the peripherals into input-output bus.

Introduction About Asynchronous Data Transfer:-

Asynchronous Data Transfer

The internal operations in a digital system are synchronized by means of clock pulses supplied by a common pulse generator. Clock pulses are applied to all registers within a unit and all data transfers among internal registers occur simultaneously during the occurrence of a clock pulse. Two units, such as a CPU and an I/O interface, are designed independently of each other.

If the registers in the interface share a common clock with the CPU registers, the transfer between the two units is said to be synchronous. In most cases, the internal timing in each unit is independent from the other in that each uses its own private clock for internal registers.

In that case, the two units are said to be asynchronous to each other. This approach is widely used in most computer systems. Asynchronous data transfer between two independent units requires that control signals be transmitted between the communicating units to indicate the time at which data is being transmitted. One way of achieving this is by means of a strobe pulse supplied by one of the units to indicate to the other unit when the transfer has to occur.

Another method commonly used is to accompany each data item being transferred with a control signal that indicates the presence of data in the bus. The unit receiving the data item responds with another control signal to acknowledge receipt of the data. This type of agreement between two independent units is referred to as handshaking.

The strobe pulse method and the handshaking method of asynchronous data transfer are not restricted to I/O transfers. In fact, they are used extensively on numerous occasions requiring the transfer of data between two independent units. In the general case we consider the transmitting unit as the source and the receiving unit as the destination.

For example, the CPU is the source unit during an output or a write transfer and it is the destination unit during an input or a read transfer. It is customary to specify the asynchronous transfer between two independent units by means of a timing diagram that shows the timing relationship that must exist between the control signals and the data in the buses. The sequence of control during an asynchronous transfer depends on whether the transfer is initiated by the source or by the destination unit.

There are two types of asynchronous data transmittion methods:-

1. Strobe control

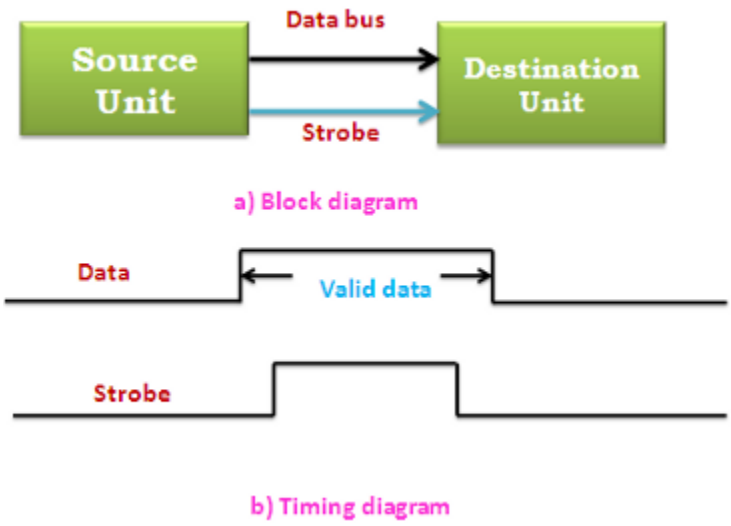
2. Handshaking.

Strobe Control

This method of asynchronous data transfer uses a single control line to time each transfer. The strobe may be activated by the source or the destination unit.

(i) Source Initiated Data Transfer:

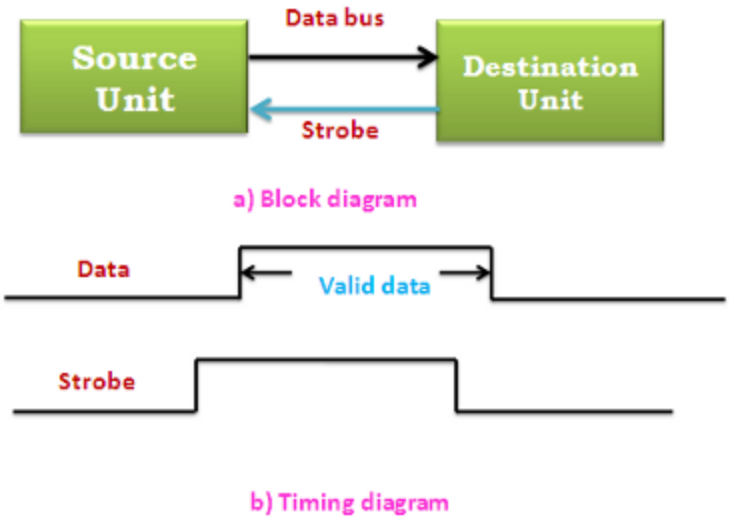
- The data bus carries the information from source to destination. The strobe is a single line. The signal on this line informs the destination unit when a data word is available in the bus.
- The strobe signal is given after a brief delay, after placing the data on the data bus. A brief period after the strobe pulse is disabled the source stops sending the data.



Source - initiated strobe for data transfer

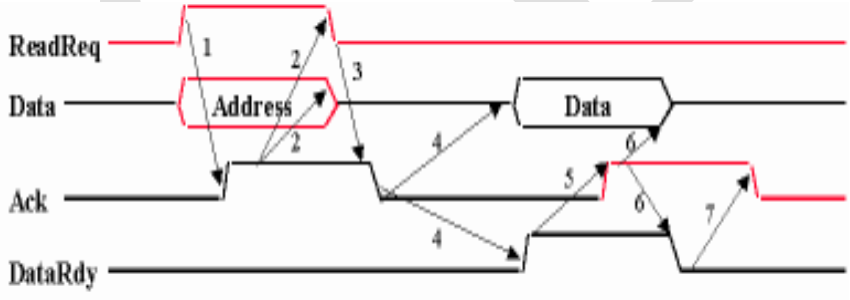
(ii) Destination Initiated Data Transfer:

- In this case the destination unit activates the strobe pulse informing the source to send data. The source places the data on the data bus. The transmission is stopped briefly after the strobe pulse is removed.
- The disadvantage of the strobe is that the source unit that initiates the transfer has no way of knowing whether the destination unit has received the data or not. Similarly if the destination initiates the transfer it has no way of knowing whether the source unit has placed data on the bus or not. This difficulty is solved by using hand shaking method of data transfer.



Destination - initiated strobe for data transfer

A Handshaking Protocol



- Three control lines
- ReadReq: indicate a read request for memory

  Address is put on the data lines at the same time

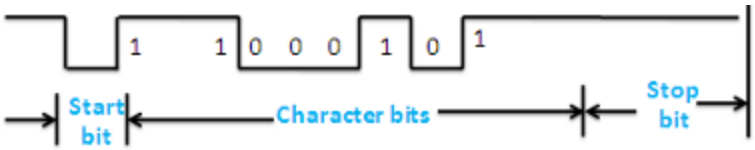
- DataRdy: indicate the data word is now ready on the data lines

  Data is put on the data lines at the same time

- Ack: acknowledge the ReadReq or the DataRdy of the other party

Asynchronous Serial Transfer

The transfer of data between tow units my be done in parallel or serial. in parallel data transmission, total message is transmitted at the same time. In serial data transmission, each bit in the message is sent in sequence one at a time. In asynchronous transmission, binary information is sent only when it is available and the line remains idle when there is no information to be transmitted.



Asynchronous serial transmission

Asynchronous serial transmission is character oriented. Each character transmitter consists of a start bit, character bits, and stop bits. The first bit is called the start bit. It is always a 0 and is used to indicate the beginning of a character. The last bit called the stop bit is always a 1.

-

Introduction About Mode of transfer:-

Mode of transfer are work in between CPU and peripherals. Input peripherals sends the data to memory which is computed by CPU. The computed data is further send back to the memory and further to output peripherals.

CPU merely execute the input-output instruction and may accept the data temporary but ultimate source and destination is the memory location.

Data transfer between CPU and input-output devices may be handled in variety of modes. these are:-

1. Programmed input-output.

2. Interrupt initiated input-output.

3. Direct Memory Access input-output.

Programmed I/O

- Programmed I/O operations are the result of I/O instructions written in computer program. Each data item transfer is initiated by an instruction in the program. The I/O device does not have direct access to memory. A transfer from an I/O device

to memory requires the execution of several instructions by the CPU. The data transfer can be synchronous or asynchronous depending upon the type and the speed of the I/O devices.

- If the speeds match then synchronous data transfer is used. When there is mismatch then asynchronous data transfer is used. The transfer is to and from a CPU register and peripheral. Other instructions are needed to transfer the data to and from CPU and memory. This method requires constant monitoring of the peripheral by the CPU. Once a data transfer is initiated the CPU is required to monitor
- The interface to see when a transfer can again be made. In this method the CPU stays in a loop till the I/O unit indicates that it is ready for data transfer. This is time consuming process which can be solved by using interrupt.

Interrupt initiated I/O

In the programmed I/O method, the CPU stays in a program loop until the I/O unit indicates that it is ready for data transfer. This is a time-consuming process since it keeps the processor busy needlessly.

It can be avoided by using an interrupt facility and special commands to inform the interface to issue an interrupt request signal when the data are available from the device.

In the meantime the CPU can proceed to execute another program. The interface meanwhile keeps monitoring the device. When the interface determines that the device is ready for data
transfer, it generates an interrupt request to the computer.

Upon detecting the external interrupt signal, the CPU momentarily stops the task it is processing, branches to a service program to process the I/O transfer, and then returns to the task it was originally performing.

Example of Interrupt initiated I/O:

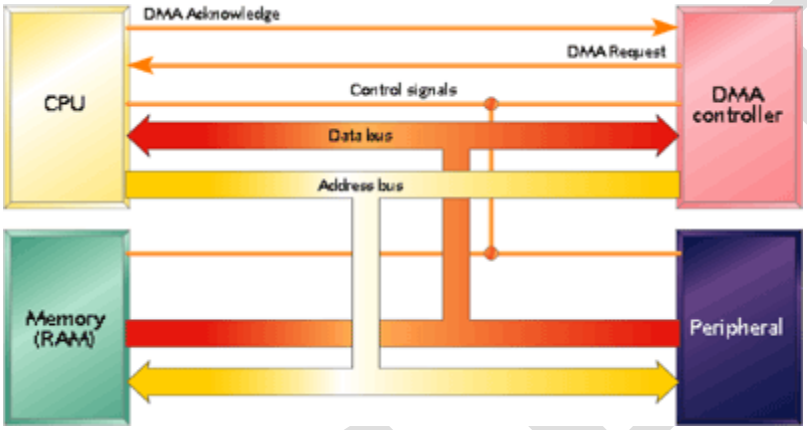
1. Vectored interrupt
2. Non vectored interrupt

Vectored interrupt :
In vectored interrupt, the source that interrupts supplies the branch information to the computer. This information is called the interrupt vector.

Non vectored interrupt
In a non vectored interrupt, the branch address is assigned to a fixed location in memory.

Direct Memory Access

DMA Short for direct memory access, a technique for transferring data from main memory to a device without passing it through the CPU. Computers that have DMA channels can transfer data to and from devices much more quickly than computers without a DMA channel can. This is useful for making quick backups and for real-time applications. Some expansion boards, such as CD-ROM cards, are capable of accessing the computer's DMA channel. When you install the board, you must specify which DMA channel is to be used, which sometimes involves setting a jumper or DIP switch.



Direct Memory Access interactions

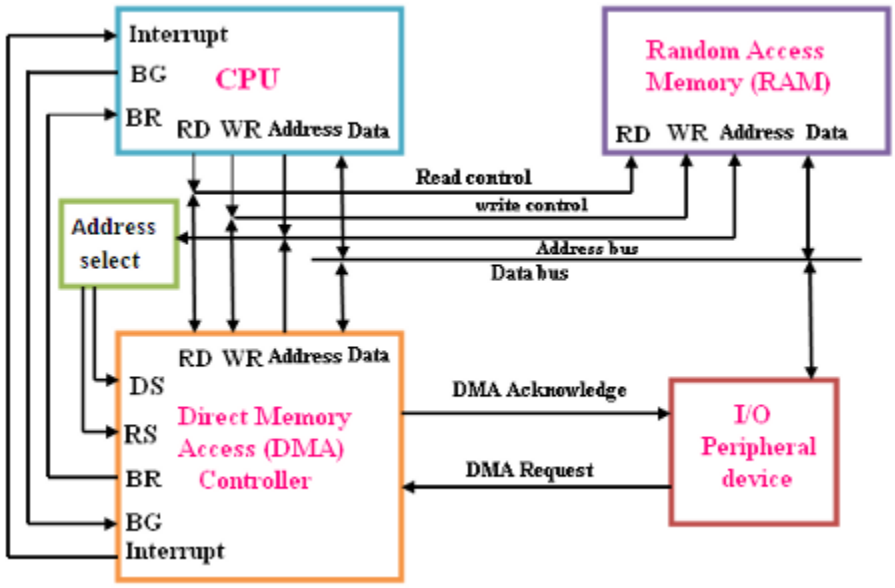
Direct Memory Access Controller

DMA controller is used to transfer the data between the memory and i/o device.

- The DMA controller needs the usual circuits to communicate with the CPU and i/o device.
- In addition to this, it needs an address register and address bus buffer.
- The address register contains an address of the desired location in memory.
- The word count register holds the number of words to be transferred. The control register specifies the mode of transfer.
- The DMA communicates with the i/o devices through the DMA request and DMA acknowledge line.
- The DMA communicates with the CPU through the data bus and control lines.
- The RD (Read) and WR (write) signals are bidirectional.
- When the BG (Bus Grant) signal are bidirectional.
- When the BG (Bus Grant) signal is 0, the CPU can communicate with the DMA registers through the data bus.
- When BG is 1, the CPU has relinquished the buses. The the DMA can communicate directly with the memory.

DMA Transfer

The connection between the DMA controller and other components in a computer system for DMA transfer is shown in figure.



DMA transfer in a computer system

- The DMA request line is used to request a DMA transfer.
- The bus request (BR) signal is used by the DMA controller to request the CPU to relinquish control of the buses.
- The CPU activates the bus grant (BG) output to inform the external DMA that its buses are in a high-impedance state (so that they can be used in the DMA transfer.)
- The address bus is used to address the DMA controller and memory at given location
- The Device select (DS) and register select (RS) lines are activated by addressing the DMA controller.
- The RD and WR lines are used to specify either a read (RD) or write (WR) operation on the given memory location.
- The DMA acknowledge line is set when the system is ready to initiate data transfer.
- The data bus is used to transfer data between the I/O device and memory.
- When the last word of data in the DMA transfer is transferred, the DMA controller informs the termination of the transfer to the CPU by means of the interrupt line.

I/O Programming

The IOP is similar to a CPU except that it is designed to handle the details of I/O processing. The IOP can fetch and execute its own instructions. IOP instructions are specifically designed to facilitate I/O transfers.

- The memory unit occupies a central position and can communicate with each processor by means of direct memory access. The CPU is responsible for processing data needed in the solution of computation tasks. The IOP provides a path for transfer of data between various peripheral devices and the memory unit.
- The CPU performs the task of initiating the I/O program. Then the IOP operates independent of the CPU and continues to transfer data. The data is transferred between the external devices and memory.
- The data formats of peripheral devices differ from memory and CPU data formats. The IOP must structure data words from many different sources. For example, it may be necessary to take four bytes from an input device and pack them into one 32-bit word before the transfer to memory.
- Data are gathered in the IOP at the device rate and bit capacity while the CPU is executing its own program. After the input data are assembled into a memory word, they are transferred from IOP directly into memory by "stealing" one memory cycle from the CPU.
- Similarly, an output word transferred from memory to the IOP is directed from the IOP to the output device at the device rate and bit capacity. The IOP responds to CPU request by placing its status word in memory location which is further examined by the CPU. Instructions that are nread from memory to IOP are called commands

CPU - IOP Communications

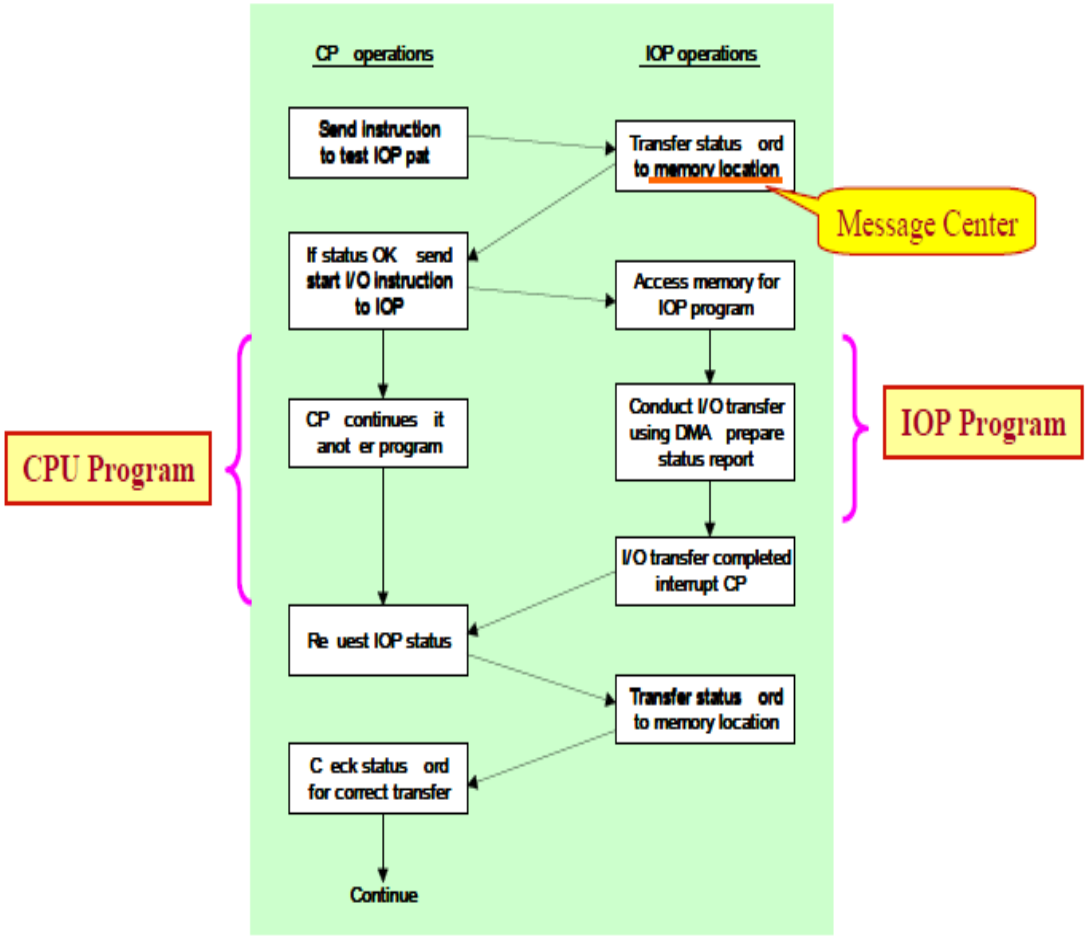
The communication between CPU and IOP may take different forms depending on the particular computer considered.

The CPU sends a test I/O instruction to IOP to test the IOP path.

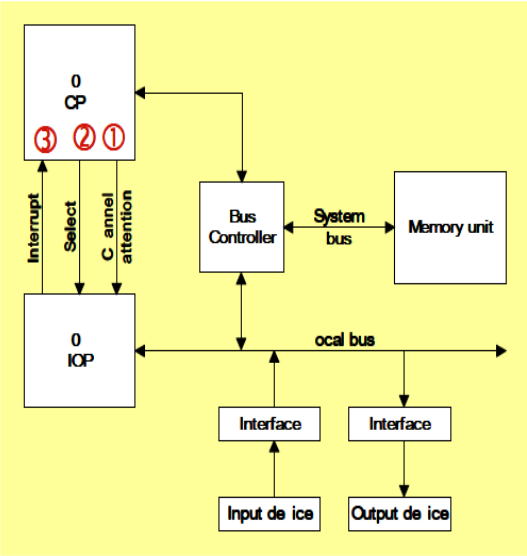
- The responds by inserting a status word in memory location.
- The CPU refers to the status word in memory. If everything is in order, the CPU sends the start I/O instruction to start the I/O transfer.
- The IOP accesses memory for IOP program.
- The CPU can now continue with another program while the IOP is busy with the program. Both programs refer to memory by means of DMA transfer.
- When the IOP terminates the execution of its program, it sends an interrupt request to the CPU.
- The CPU then issues a read I/O instruction to read the status from the IOP.
- The IOP transfers the status word to memory location.
- The status word indicates whether the transfer has been completed satisfactorily or if any error has occurred during the transfer.

♦ CPU - IOP Communication : *Fig. 11-20*

- Memory units acts as a message center : Information
  - » each processor leaves information for the other



◆ Intel 8089 IOP : *Fig. 11-23*

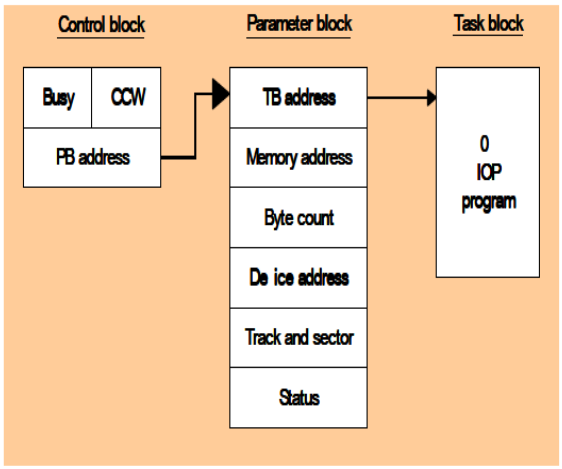


① CPU enables channel attention

② Select one of two channels of 8089

③ 8089 gets attention of the CPU by sending an interrupt request

◆ Location of Information : *Fig. 11-24*

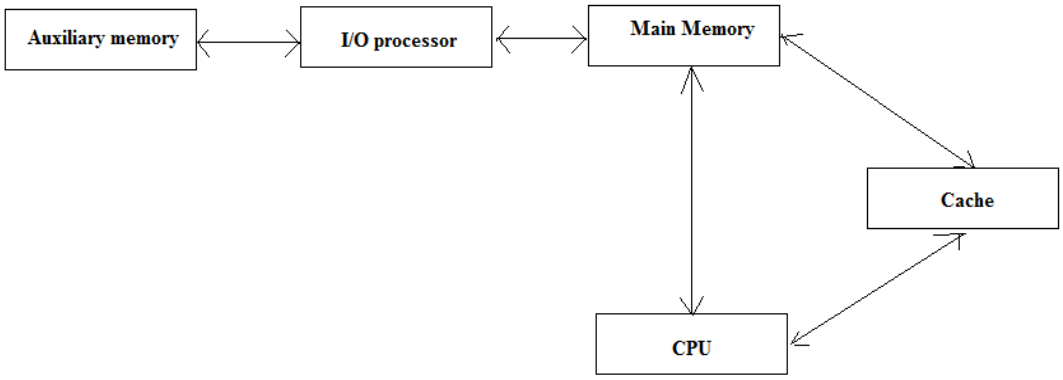


- Channel Command Word (CCW) : **message center**
  - » Start command
  - » Suspend command
  - » Resume command
  - » Halt command

Unit III

Memory Hierarchy

- The memory unit is an essential component in any digital computer since it is needed for storing programs and data
- Not all accumulated information is needed by the CPU at the same time
- Therefore, it is more economical to use low-cost storage devices to serve as a backup for storing the information that is not currently used by CPU
- The memory unit that directly communicate with CPU is called the *main memory*
- Devices that provide backup storage are called *auxiliary memory*
- The memory hierarchy system consists of all storage devices employed in a computer system from the slow by high-capacity auxiliary memory to a relatively faster main memory, to an even smaller and faster cache memory
- The main memory occupies a central position by being able to communicate directly with the CPU and with auxiliary memory devices through an I/O processor
- A special very-high-speed memory called **cache** is used to increase the speed of processing by making current programs and data available to the CPU at a rapid rate



CPU logic is usually faster than main memory access time, with the result that processing speed is limited primarily by the speed of main memory

- The cache is used for storing segments of programs currently being executed in the CPU and temporary data frequently needed in the present calculations
- The typical access time ratio between cache and main memory is about 1to7

Auxiliary memory access time is usually 1000 times that of main memory

Main Memory

- Most of the main memory in a general purpose computer is made up of RAM integrated circuits chips, but a portion of the memory may be constructed with ROM chips
- RAM– Random Access memory
  - In tegated RAM are available in two possible operating modes, S*tatic and Dynamic*
- ROM– Read Only memory

Random-Access Memory (RAM)

Static RAM (SRAM)

- Each cell stores bit with a six-transistor circuit.
- Retains value indefinitely, as long as it is kept powered.
- Relatively insensitive to disturbances such as electrical noise.
- Faster and more expensive than DRAM.

Dynamic RAM (DRAM)

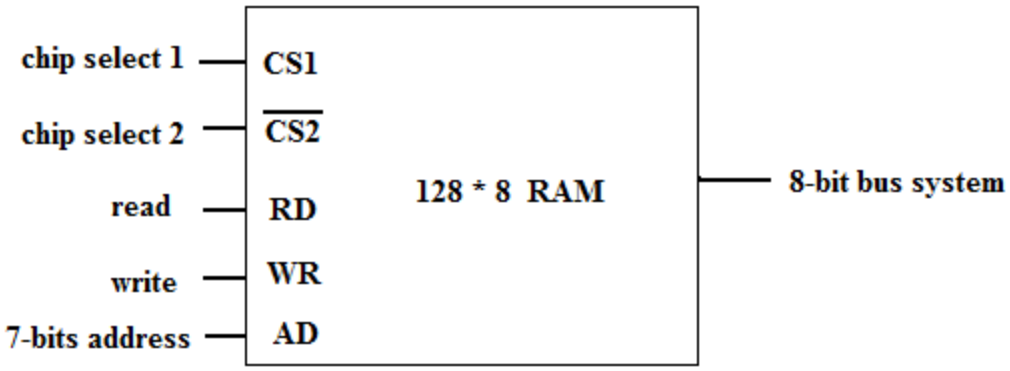
- Each cell stores bit with a capacitor and transistor.
- Value must be refreshed every 10-100 ms.
- Sensitive to disturbances.
- Slower and cheaper than SRAM.

ROM

- ROM is used for storing programs that are **PERMENTLY** resident in the computer and for tables of constants that do not change in value once the production of the computer is completed
- The ROM portion of main memory is needed for storing an initial program called *bootstrap loader,* witch is to start the computer software operating when power is turned off

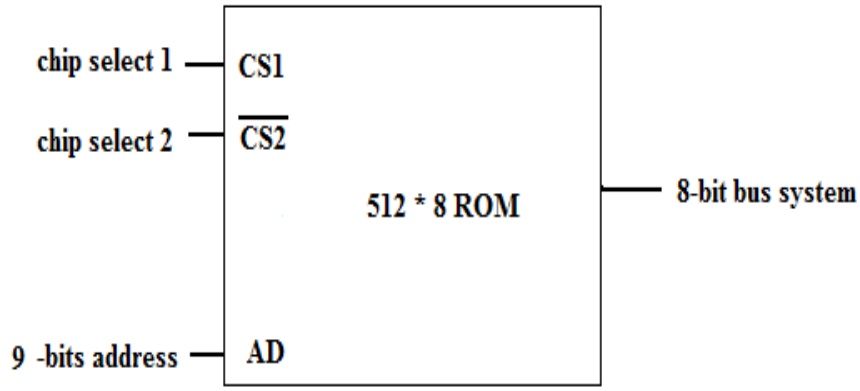
A RAM chip is better suited for communication with the CPU if it has one or more control inputs that select the chip when needed

The Block diagram of a RAM chip is shown next slide, the capacity of the memory is 128 words of 8 bits (one byte) per word



CS1	$\overline{CS2}$	RD	WD	Memory Function	State of data bus
0	0	*	*	Inhibit	High-impedance
0	1	*	*	Inhibit	High-impedance
1	0	0	0	Inhibit	High-impedance
1	0	0	1	Write	Input data to RAM
1	0	1	*	Read	Output data from RAM
1	1	*	*	Inhibit	High-impedance

ROM



Memory Address Map

- Memory Address Map is a pictorial representation of assigned address space for each chip in the system
- To demonstrate an example, assume that a computer system needs 512 bytes of RAM and 512 bytes of ROM
- The RAM have 128 byte and need seven address lines, where the ROM have 512 bytes and need 9 address lines

Component	Hexadecimal Address	10	9	8	7	6	5	4	3	2	1
RAM1	0000-007F	0	0	0	*	*	*	*	*	*	*
RAM2	0080-00FF	0	0	1	*	*	*	*	*	*	*
RAM3	0100-017F	0	1	0	*	*	*	*	*	*	*
RAM4	0180-01FF	0	1	1	*	*	*	*	*	*	*
ROM	0200-03FF	1	*	*	*	*	*	*	*	*	*

- The hexadecimal address assigns a range of hexadecimal equivalent address for each chip
- Line 8 and 9 represent four distinct binary combination to specify which RAM we chose
- When line 10 is 0, CPU selects a RAM. And when it's 1, it selects the ROM

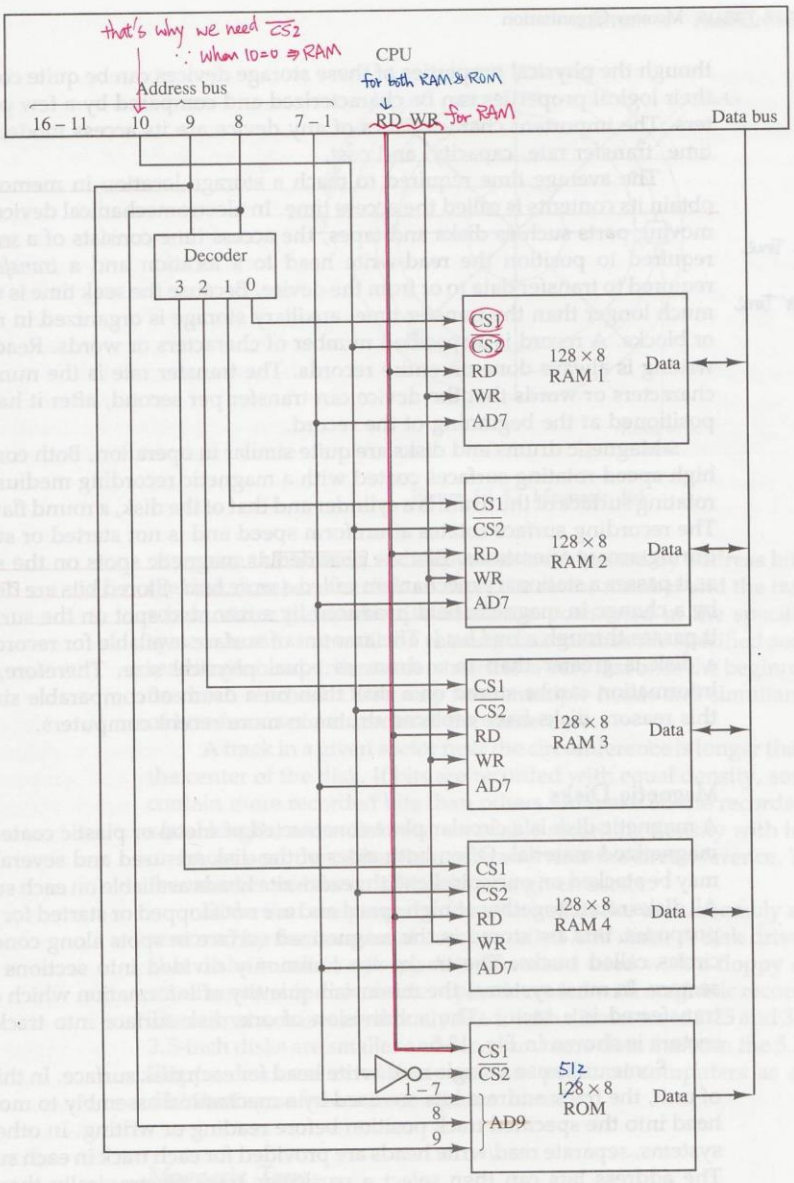


Figure 12-4 Memory connection to the CPU. RAM & ROM connection

Memory Interleaving concept (p324)

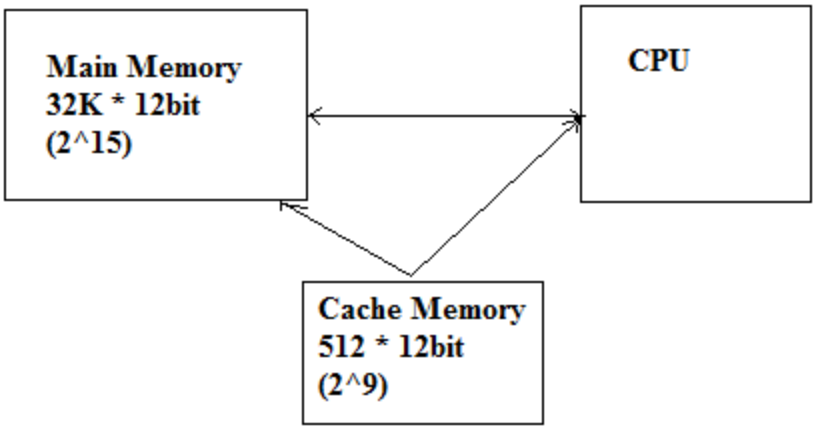
453

Cache memory

- If the active portions of the program and data are placed in a fast small memory, the average memory access time can be reduced,
- Thus reducing the total execution time of the program
- Such a fast small memory is referred to as cache memory

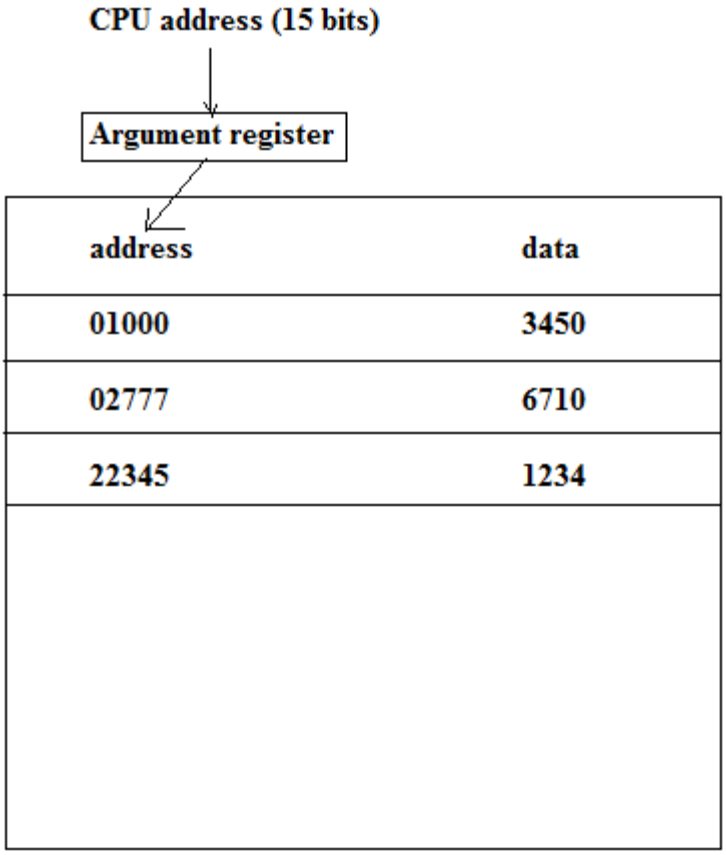
- The cache is the fastest component in the memory hierarchy and approaches the speed of CPU component
- When CPU needs to access memory, the cache is examined
- If the word is found in the cache, it is read from the fast memory
- If the word addressed by the CPU is not found in the cache, the main memory is accessed to read the word
- The performance of cache memory is frequently measured in terms of a quantity called **hit ratio**
- When the CPU refers to memory and finds the word in cache, it is said to produce a **hit**
- Otherwise, it is a **miss**
- **Hit ratio = hit / (hit+miss)**
- The basic characteristic of cache memory is its fast access time,
- Therefore, very little or no time must be wasted when searching the words in the cache
- The transformation of data from main memory to cache memory is referred to as a **mapping** process, there are three types of mapping:
    - Associative mapping
    - Direct mapping
    - Set-associative mapping

To help understand the mapping procedure, we have the following example:



Associative mapping

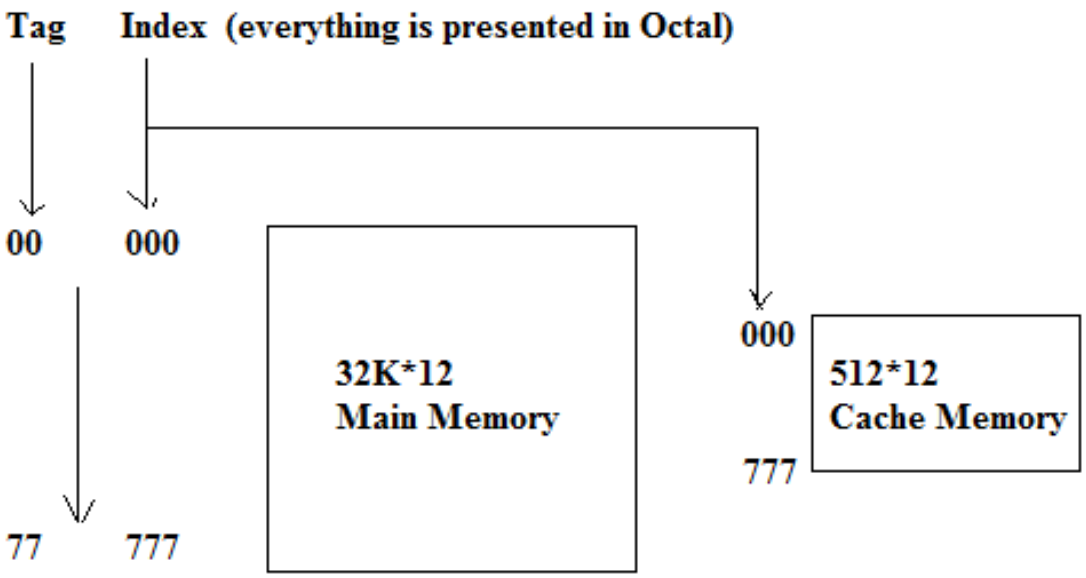
- The fastest and most flexible cache organization uses an associative memory
- The associative memory stores both the address and data of the memory word
- This permits any location in cache to store ant word from main memory
- The address value of 15 bits is shown as a five-digit **octal** number and its corresponding 12-bit word is shown as a four-digit octal number



- A CPU address of 15 bits is places in the argument register and the associative memory us searched for a matching address
- If the address is found, the corresponding 12-bits data is read and sent to the CPU
- If not, the main memory is accessed for the word
- If the cache is full, an address-data pair must be displaced to make room for a pair that is needed and not presently in the cache

Direct Mapping

- Associative memory is expensive compared to RAM
- In general case, there are 2^k words in cache memory and 2^n words in main memory (in our case, k=9, n=15)
- The n bit memory address is divided into two fields: k-bits for the index and n-k bits for the tag field



Memory Address	Memory Data
00000	1220
00777	2340
01000	3450
01111	2222
01777	4560
02000	5670
02777	6710

Index Address	Tag	Data
000	00	1220
111	01	2222
777	02	6710

Set-Associative Mapping

- The disadvantage of direct mapping is that two words with the same index in their address but with different tag values cannot reside in cache memory at the same time
- Set-Associative Mapping is an improvement over the direct-mapping in that each word of cache can store two or more word of memory under the same index address

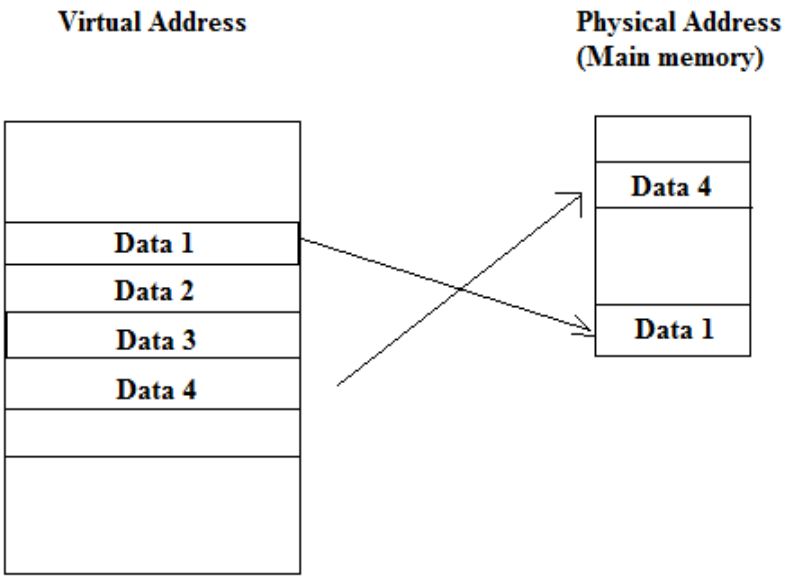
Memory Address	Memory Data
00000	1220
00777	2340
01000	3450
01111	2222
01777	4560
02000	5670
02777	6710

Index Address	Tag	Data	Tag	Data
000	01	3450	02	5670
111	01	2222		
777	02	6710	00	2340

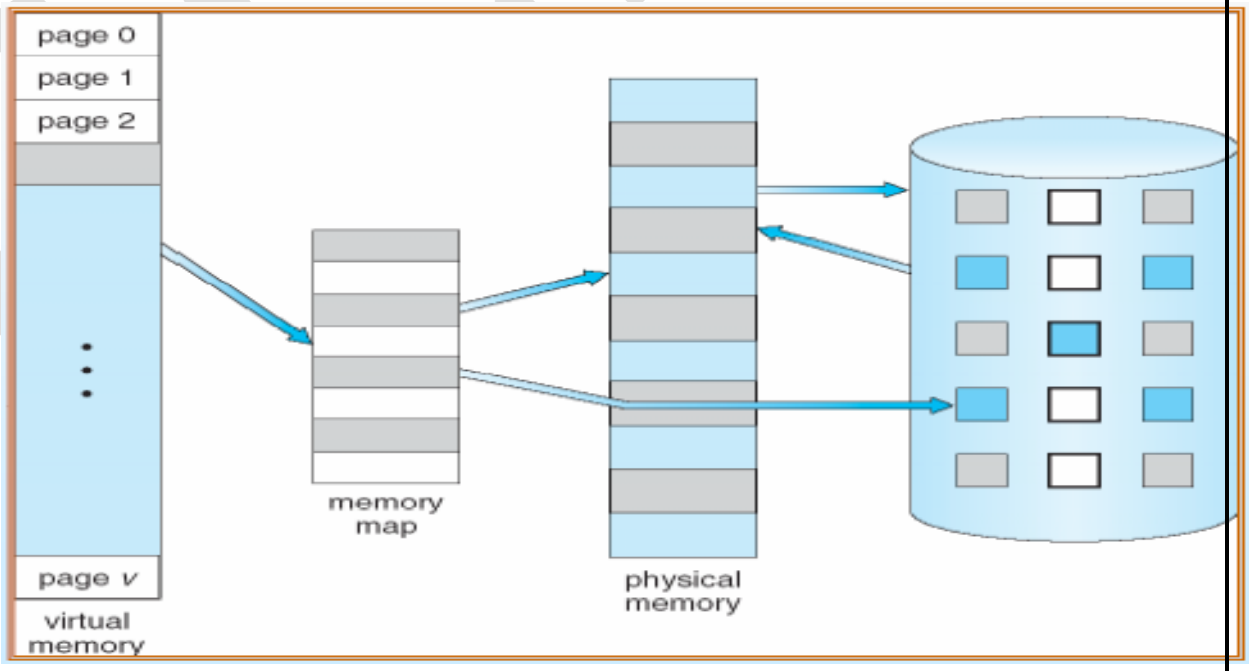
- In the slide, each index address refers to two data words and their associated tags
- Each tag requires six bits and each data word has 12 bits, so the word length is 2*(6+12) = 36 bits

Virtual Memory

- The address used by a programmer will be called a virtual address or logical address.
- An address in main memory is called a physical address



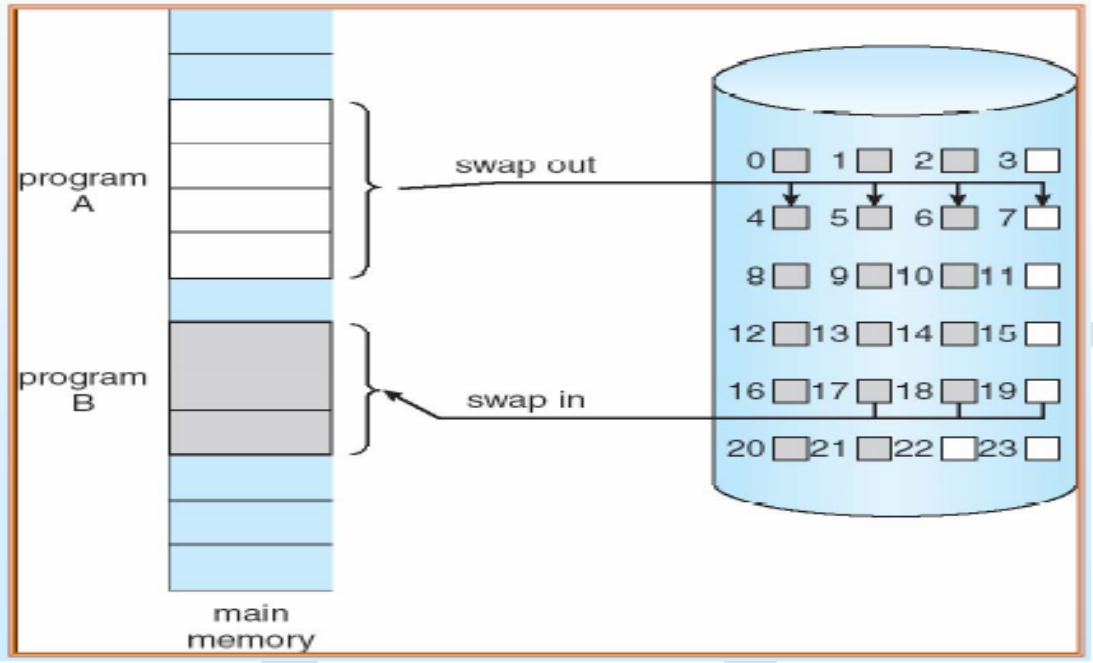
- The term **page** refers to groups of address space of the same size
- For example: if auxiliary memory contains 1024K and main memory contains 32K and page size equals to 1K, then auxiliary memory has 1024 pages and main memory has 32 pages
- Only part of the program needs to be in memory for execution
- Logical address space can therefore be much larger than physical address space
- Allows for more efficient process creation



Demand Paging

- In stead of loading whole program into memory, **demand paging** is an alternative strategy to initially load pages only as they are needed
- **Lazy Swapper:** Pages are only loaded when they are demanded during program execution

Transfer of a page memory to continuous disk space



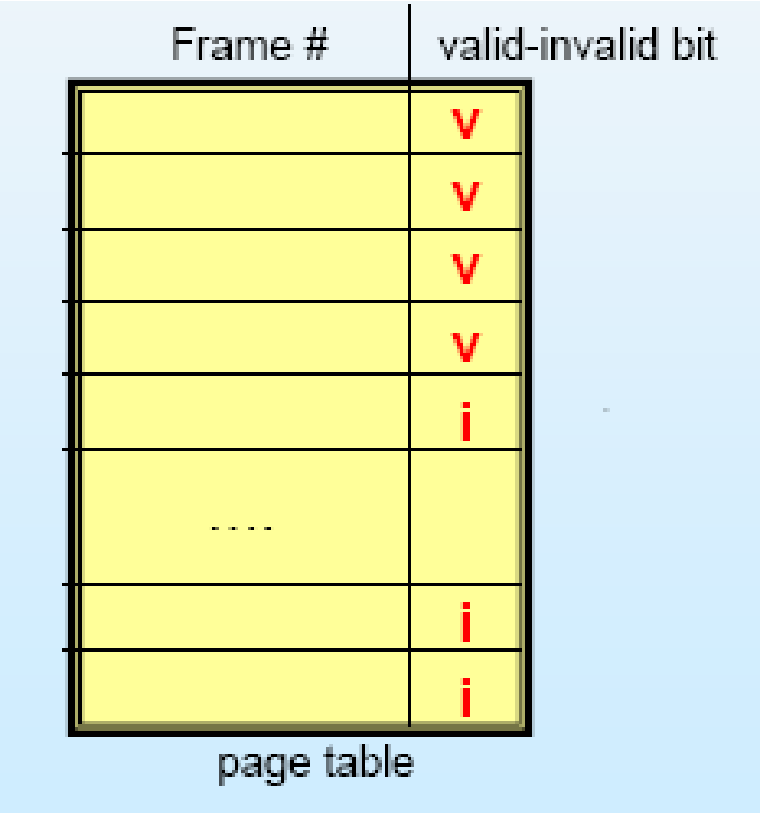
- When a process is to be swapped in, the pager guesses which pages will be used before the process is swapped out again.
- Instead of swapping in a whole process, the pager brings only those necessary pages into memory

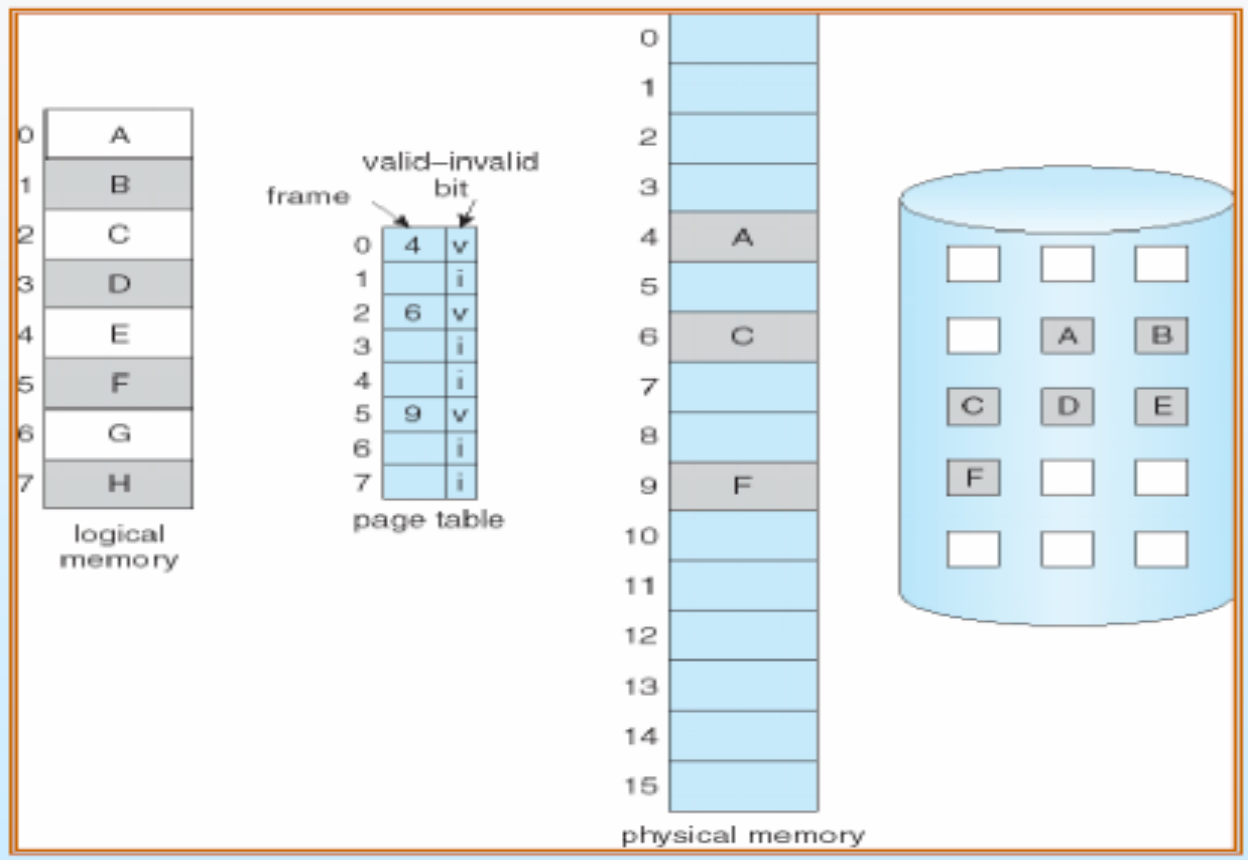
Valid-Invalid Bit

- With each page table entry a

valid–invalid bit is associated
(**v**=> in-memory , **i** =>not-in-memory)

- Initially valid–invalid bit is set to **i** on all entries
- During address translation, if valid–invalid bit in page table entry is **i** => page fault


Frame #
valid-invalid bit
v
v
v
v
i
. . . .
i
i
page table

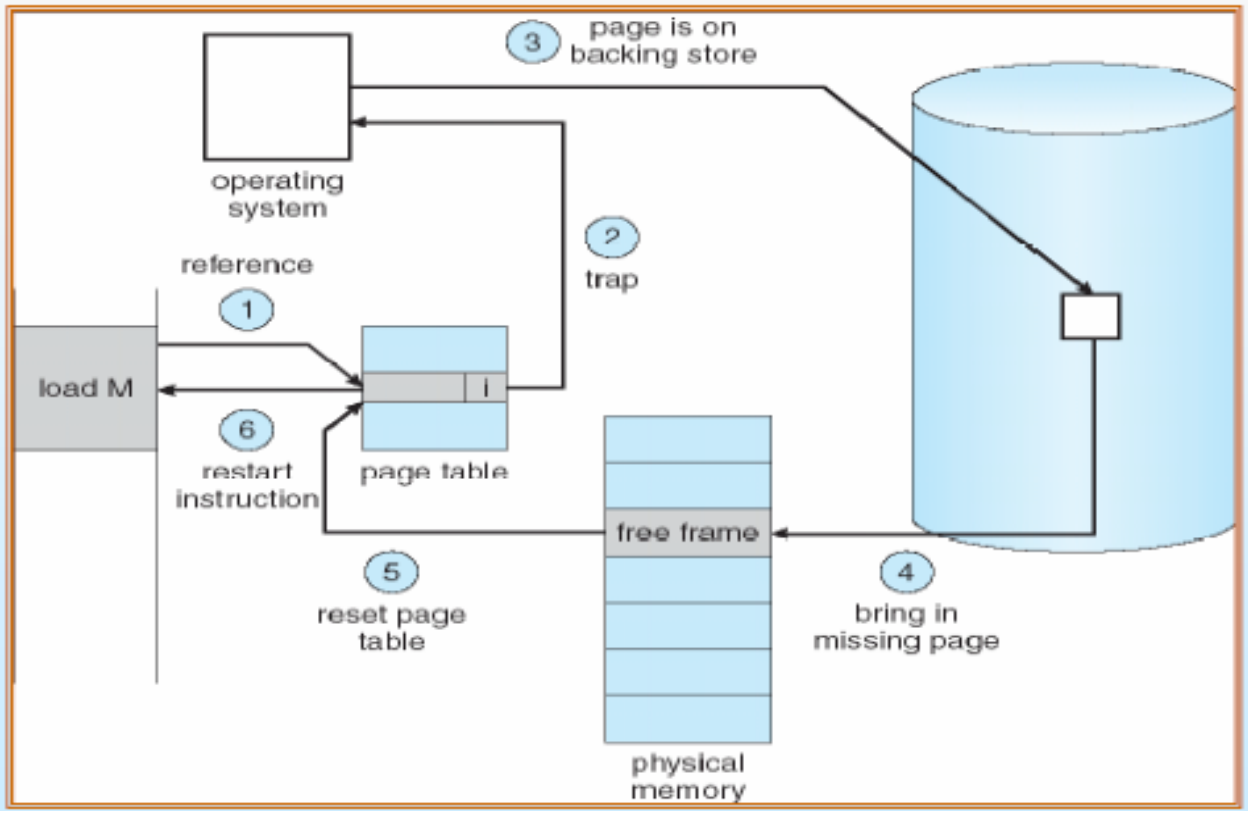

0 A
1 B
2 C
3 D
4 E
5 F
6 G
7 H
logical memory
valid–invalid bit
frame
0 4 v
1 i
2 6 v
3 i
4 i
5 9 v
6 i
7 i
page table
0
1
2
3
4 A
5
6 C
7
8
9 F
10
11
12
13
14
15
physical memory
A B
C D E
F

Page Fault

- If there is a reference to a page, first reference to that page will trap to operating system:

  page fault

1. Operating system looks at another table to decide:
   - Invalid reference ⇒ abort
   - Just not in memory
2. Get empty frame
3. Swap page into frame
4. Reset tables
5. Set validation bit = v
6. Restart the instruction that caused the page fault



Performance of Demand Paging

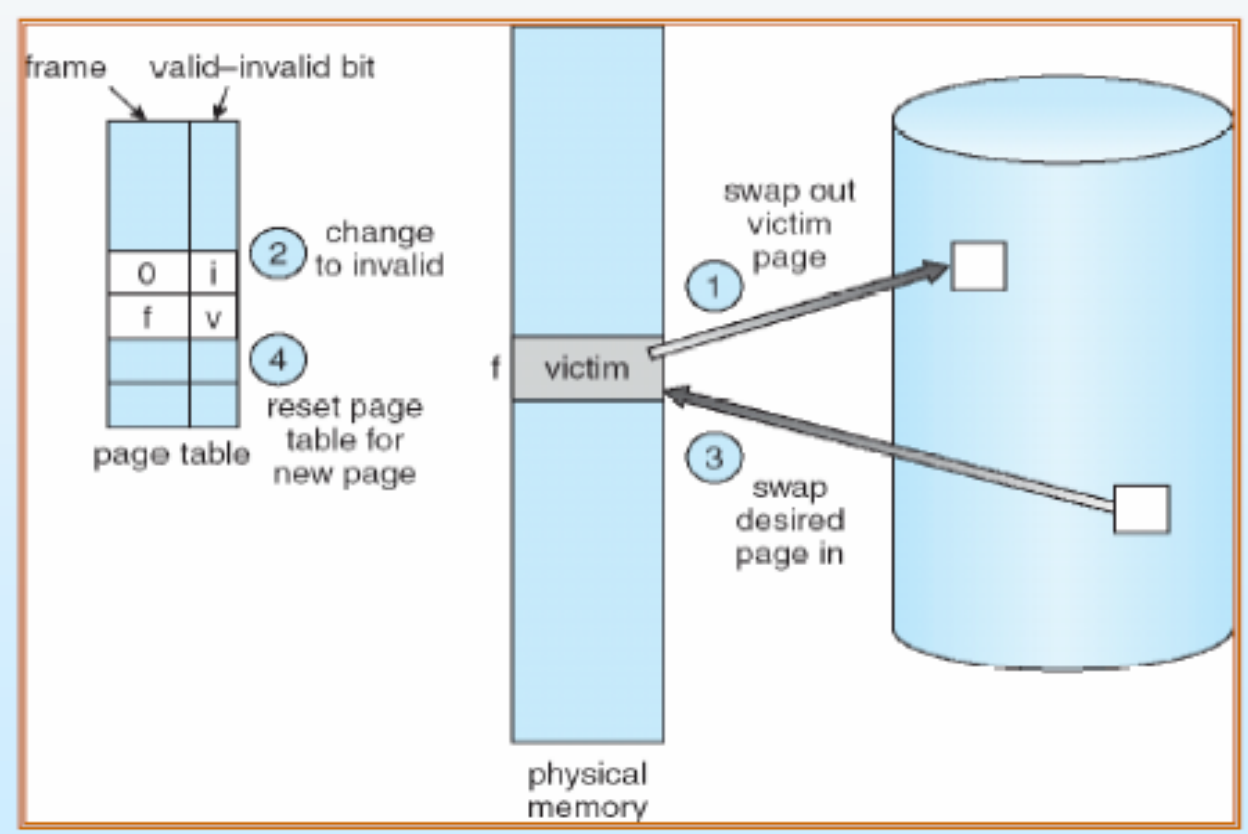
- Memory access time = 200 nanoseconds
- Average page-fault service time = 8 milliseconds
- EAT = (1 – p) x 200 + p (8 milliseconds)
  = (1 – p x 200 + p x 8,000,000
  = 200 + p x 7,999,800
- If we want performance degradation to be less than 10%, we need

220 > 200+7,999,800*p
20>7,999,800*p
P<0.0000025
It is important to keep the page-fault rate low in a demand-paging system

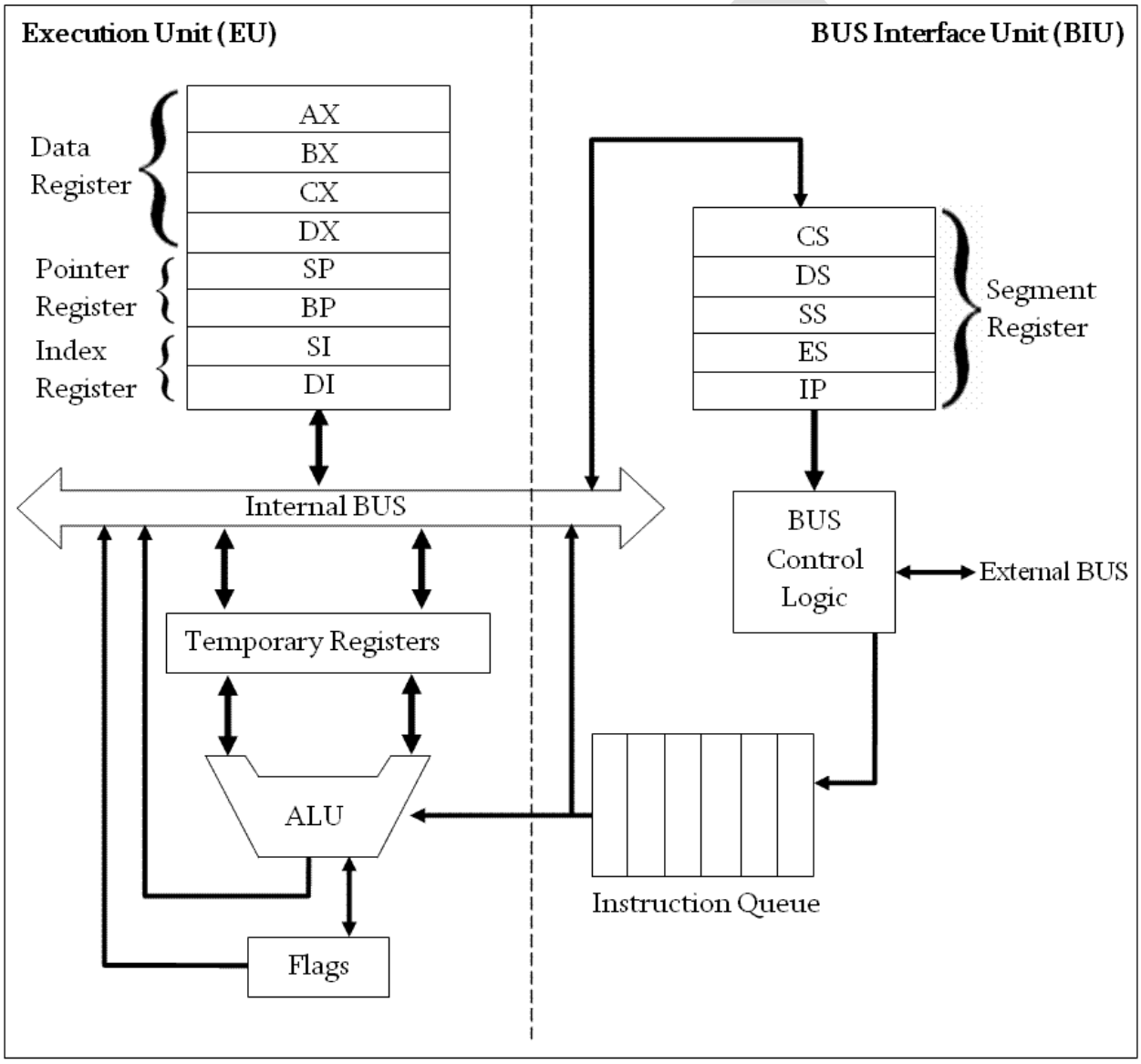
Page Replacement

- What if there is no free frame?
- Page replacement –find some page in memory, but not really in use, swap it out
- In this case, same page may be brought into memory several times

1. Find the location of the desired page on disk
2. Find a free frame:
   - If there is a free frame, use it
   - If there is no free frame, use a page replacement algorithm to select a **victim** frame
3. Bring the desired page into the (newly) free frame; update the page and frame tables
4. Restart the process


frame
valid–invalid bit
0 i
f v
page table
2
change to invalid
4
reset page table for new page
f
victim
1
swap out victim page
3
swap desired page in
physical memory

Unit IV
8086 Architecture



8086 microprocessor has two units; Execution Unit (EU) and Bus Interface Unit (BIU). They are dependent and get worked by each other. Below is a short description of these two units.

Execution Unit (EU):

Execution unit receives program instruction codes and data from the BIU, executes them and stores the results in the general registers. It can also store the data in a memory location or send them to an I/O device by passing the data back to the BIU. This unit, EU, has no connection with the system Buses. It receives and outputs all its data through BIU.

ALU (Arithmetic and Logic Unit) : The EU unit contains a circuit board called the Arithmetic and Logic Unit. The ALU can perform arithmetic, such as, +,-,×,/ and logic such as OR, AND, NOT operations.
Registers : A register is like a memory location where the exception is that these are denoted by name rather than numbers. It has 4 data registers, AX, BX, CX, DX and 2 pointer registers SP, BP and 2 index registers SI, DI and 1 temporary register and 1 status register FLAGS .
AX, BX, CX and DX registers has 2 8-bit registers to access the high and low byte data registers. The high byte of AX is called AH and the low byte is AL. Similarly, the high and low bytes of BX, CX, DX are BH and BL, CH and Cl, DH and DL respectively. All the data, pointer, index and status registers are of 16 bits. Else these, the temporary register holds the operands for the ALU and the individual bits of the FLAGS register reflect the result of a computation.

Bus Interface Unit:

As the EU has no connection with the system Busses, this job is done by BIU. BIU and EU are connected with an internal bus. BIU connects EU with the memory or I/O circuits. It is responsible for transmitting data, addresses and control signal on the busses.
Registers : BIU has 4 segment busses, CS, DS, SS, ES. These all 4 segment registers holds the addresses of instructions and data in memory. These values are used by the processor to access memory locations. It also contain 1 pointer register IP. IP contains the address of the next instruction to executed by the EU.
Instruction Queue : BIU also contain an instruction queue. When the EU executes instructions, the BIU gets up to 6 bytes of the next instruction and stores them in the instruction queue and this process is called instruction prefetch. This is a process to speed up the processor. Also when the EU needs to be connected with memory or peripherals, BIU suspends instruction prefetch and performs the needed operations.

ALU (Arithmetic & Logic Unit):

This unit can perform various arithmetic and logical operation, if required, based on the instruction to be executed. It can perform arithmetical operations, such as add, subtract, increment, decrement, convert byte/word and compare etc and logical operations, such as AND, OR, exclusive OR, shift/rotate and test etc.

Purpose of using Instruction Queue:

BIU contains an instruction queue. When the EU executes instructions, the BIU gets up to 6 bytes of the next instruction and stores them in the instruction queue and this process is called instruction prefetch. This is a process to speed up the processor. A subtle advantage of instruction queue is that, as next several instructions are usually in the queue, the BIU can access memory at a somewhat "leisurely" pace. This means that slow-memory parts can be used without affecting overall system performance.

Registers

Data Registers

- AX = Accumulator Register
- BX = Base Register
- DX = Data Register
- CX = Count Register

Index Registers

- SI = Source Index
- DI = Destination Index

Segment Registers

- DS = Data Segment
- SS = Stack Segment
- ES = Extra Segment
- CS = Code Segment

Pointer Registers

- IP = Instruction Pointer
- BP = Base Pointer
- SP = Stack Pointer

Segment Register:

CS (Code Segment) :

Code segment (CS) is a 16-bit register containing address of 64 KB segment with processor instructions. The processor uses CS segment for all accesses to instructions referenced by instruction pointer (IP) register. CS register cannot be changed directly. The CS register is automatically updated during far jump, far call and far return instructions.

Stack segment (SS)

---

is a 16-bit register containing address of 64KB segment with program stack. By default, the processor assumes that all data referenced by the stack pointer (SP) and base pointer (BP) registers is located in the stack segment. SS register can be changed directly using POP instruction.

Data segment (DS)

---

is a 16-bit register containing address of 64KB segment with program data. By default, the processor assumes that all data referenced by general registers (AX, BX, CX, DX) and index register (SI, DI) is located in the data segment. DS register can be changed directly using POP and LDS instructions.

Extra segment (ES)

---

is a 16-bit register containing address of 64KB segment, usually with program data. By default, the processor assumes that the DI register references the ES segment in string manipulation instructions. ES register can be changed directly using POP and LES instructions. It is possible to change default segments used by general and index registers by prefixing instructions with a CS, SS, DS or ES prefix.

IP (Instruction Pointer) :

To access instructions the 8086 uses the registers CS and IP. The CS register contains the segment number of the next instruction and the IP contains the offset. IP is updated each time an instruction is executed so that it will point to the next instruction. Unlike other registers the IP can't be directly manipulated by an instruction, that is, an instruction may not contain IP as its operand.

General Registers :

All general registers of the 8086 microprocessor can be used for arithmetic and logic operations. The general registers are:

AX (Accumulator):

This is accumulator register. It gets used in arithmetic, logic and data transfer instructions. In manipulation and division , one of the numbers involved must be in AX or AL.

BX (Base Register):

This is base register. BX register is an address register. It usually contain a data pointer used for based, based indexed or register indirect addressing.

CX (Count register):

This is Count register. This serves as a loop counter. Program loop constructions are facilitated by it. Count register can also be used as a counter in string manipulation and shift/rotate instruction.

DX (Data Register):

This is data register. Data register can be used as a port number in I/O operations. It is also used in multiplication and division.

SP (Stack Pointer):

This is stack pointer register pointing to program stack. It is used in conjunction with SS for accessing the stack segment.

BP (Base Pointer):

This is base pointer register pointing to data in stack segment. Unlike SP, we can use BP to access data in the other segments.

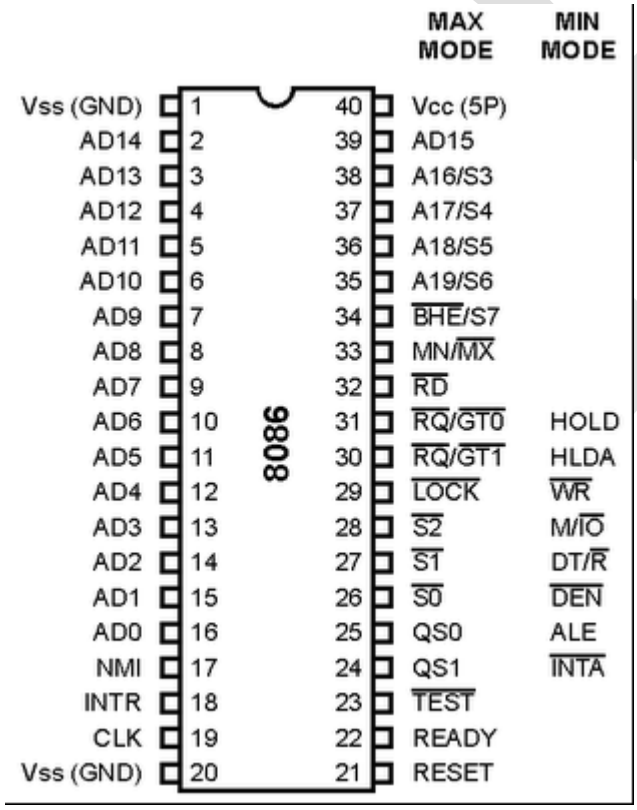
SI (Source Index):

This is source index register which is used to point to memory locations in the data segment addressed by DS. By incrementing the contents of SI one can easily access consecutive memory locations.

DI (Destination Index):

This is destination index register performs the same function as SI. There is a class of instructions called string operations, that use DI to access the memory locations addressed by ES.

Pin Diagram and Pin description of 8086



The following pin function descriptions are for the microprocessor 8086 in either minimum or maximum mode.

AD0 - AD15 (I/O): Address Data Bus

These lines constitute the time multiplexed memory/IO address during the first clock cycle (T1) and data during T2, T3 and T4 clock cycles. A0 is analogous to BHE for the lower byte of the data bus, pins D0-D7. A0 bit is Low during T1 state when a byte is to be transferred on the lower portion of the bus in memory or I/O operations. 8-bit oriented devices tied to the lower half would normally use A0 to condition chip select functions. These lines are active high and float to tri-state during interrupt acknowledge and local bus "Hold acknowledge".

A19/S6, A18/S5, A17/S4, A16/S3 (0): Address/Status

During T1 state these lines are the four most significant address lines for memory operations. During I/O operations these lines are low. During memory and I/O operations, status information is available on these lines during T2, T3, and T4 states.S5: The status of the interrupt enable flag bit is updated at the beginning of each cycle. The status of the flag is indicated through this bus.

S6:

When Low, it indicates that 8086 is in control of the bus. During a "Hold acknowledge" clock period, the 8086 tri-states the S6 pin and thus allows another bus master to take control of the status bus.

S3 & S4:

Lines are decoded as follows:

A17/S4	A16/S3	Function
0	0	Extra segment access
0	1	Stack segment access
1	0	Code segment access
1	1	Data segment access

After the first clock cycle of an instruction execution, the A17/S4 and A16/S3 pins specify which segment register generates the segment portion of the 8086 address. Thus by decoding these lines and using the decoder outputs as chip selects for memory chips, up to 4 Megabytes (one Mega per segment) of memory can be accesses. This feature also provides a degree of protection by preventing write operations to one segment from

erroneously overlapping into another segment and destroying information in that segment.

BHE /S7 (O): Bus High Enable/Status

During T1 state theBHE should be used to enable data onto the most significant half of the data bus, pins D15 - D8. Eight-bit oriented devices tied to the upper half of the bus would normally use BHE to control chip select functions. BHE is Low during T1 state of read, write and interrupt acknowledge cycles when a byte is to be transferred on the high portion of the bus.
The S7 status information is available during T2, T3 and T4 states. The signal is active Low and floats to 3-state during "hold" state. This pin is Low during T1 state for the first interrupt acknowledge cycle.

RD (O): READ

The Read strobe indicates that the processor is performing a memory or I/O read cycle. This signal is active low during T2 and T3 states and the Tw states of any read cycle. This signal floats to tri-state in "hold acknowledge cycle".

TEST (I)

TEST pin is examined by the "WAIT" instruction. If the TEST pin is Low, execution continues. Otherwise the processor waits in an "idle" state. This input is synchronized internally during each clock cycle on the leading edge of CLK.

INTR (I): Interrupt Request

It is a level triggered input which is sampled during the last clock cycle of each instruction to determine if the processor should enter into an interrupt acknowledge operation. A subroutine is vectored to via an interrupt vector look up table located in system memory. It can be internally masked by software resetting the interrupt enable bit INTR is internally synchronized. This signal is active HIGH.

NMI (I): Non-Muskable Interrupt

An edge triggered input, causes a type-2 interrupt. A subroutine is vectored to via the interrupt vector look up table located in system memory. NMI is not maskable internally by software. A transition from a LOW to HIGH on this pin initiates the interrupt at the end of the current instruction. This input is internally synchronized.

Reset (I)

Reset causes the processor to immediately terminate its present activity. To be recognised, the signal must be active high for at least four clock cycles, except after power-on which requires a 50 Micro Sec. pulse. It causes the 8086 to initialize registers DS, SS, ES, IP and flags to all zeros. It also initializes CS to FFFF H. Upon removal of the RESET signal from the RESET pin, the 8086 will fetch its next instruction from the 20 bit physical address FFFF0H. The reset signal to 8086 can be generated by the 8284. (Clock generation chip). To guarantee reset from power-up, the reset input must remain below 1.5 volts for 50 Micro sec. after Vcc has reached the minimum supply voltage of 4.5V.

Ready (I)

Ready is the acknowledgement from the addressed memory or I/O device that it will complete the data transfer. The READY signal from memory or I/O is synchronized by the 8284 clock generator to form READY. This signal is active HIGH. The 8086 READY input is not synchronized. Correct operation is not guaranteed if the setup and hold times are not met.

CLK (I): Clock

Clock provides the basic timing for the processor and bus controller. It is asymmetric with 33% duty cycle to provide optimized internal timing. Minimum frequency of 2 MHz is required, since the design of 8086 processors incorporates dynamic cells. The maximum clock frequencies of the 8086-4, 8086 and 8086-2 are4MHz, 5MHz and 8MHz respectively.
Since the 8086 does not have on-chip clock generation circuitry, and 8284 clock generator chip must be connected to the 8086 clock pin. The crystal connected to 8284 must have a frequency 3 times the 8086 internal frequency. The 8284 clock generation chip is used to generate READY, RESET and CLK.

MN/MX (I): Maximum / Minimum

This pin indicates what mode the processor is to operate in. In minimum mode, the 8086 itself generates all bus control signals. In maximum mode the three status signals are to be decoded to generate all the bus control signals.
Minimum Mode Pins The following 8 pins function descriptions are for the 8086 in

minimum mode; MN/ MX = 1. The corresponding 8 pins function descriptions for maximum mode is explained later.

M/IO (O): Status line

This pin is used to distinguish a memory access or an I/O accesses. When this pin is Low, it accesses I/O and when high it access memory. M / IO becomes valid in the T4 state preceding a bus cycle and remains valid until the final T4 of the cycle. M/IO floats to 3 - state OFF during local bus "hold acknowledge".

WR (O): Write

Indicates that the processor is performing a write memory or write IO cycle, depending on the state of the M /IOsignal. WR is active for T2, T3 and Tw of any write cycle. It is active LOW, and floats to 3-state OFF during local bus "hold acknowledge ".

INTA (O): Interrupt Acknowledge

It is used as a read strobe for interrupt acknowledge cycles. It is active LOW during T2, T3, and T4 of each interrupt acknowledge cycle.

ALE (O): Address Latch Enable

ALE is provided by the processor to latch the address into the 8282/8283 address latch. It is an active high pulse during T1 of any bus cycle. ALE signal is never floated.

DT/ R (O): DATA Transmit/Receive

In minimum mode, 8286/8287 transceiver is used for the data bus. DT/ R is used to control the direction of data flow through the transceiver. This signal floats to tri-state off during local bus "hold acknowledge".

DEN (O): Data Enable

It is provided as an output enable for the 8286/8287 in a minimum system which uses the transceiver. DEN is active LOW during each memory and IO access. It will be low beginning with T2 until the middle of T4, while for a write cycle, it is active from the beginning of T2 until the middle of T4. It floats to tri-state off during local bus "hold acknowledge".

HOLD & HLDA (I/O): Hold and Hold Acknowledge

Hold indicates that another master is requesting a local bus "HOLD". To be acknowledged, HOLD must be active HIGH. The processor receiving the "HOLD " request will issue HLDA (HIGH) as an acknowledgement in the middle of the T1-clock cycle. Simultaneous with the issue of HLDA, the processor will float the local bus and control lines. After "HOLD" is detected as being Low, the processor will lower the HLDA and when the processor needs to run another cycle, it will again drive the local bus and control lines.
Maximum Mode The following pins function descriptions are for the 8086/8088 systems in maximum mode (i.e.. MN/MX = 0). Only the pins which are unique to maximum mode are described below.

S2, S1, S0 (O): Status Pins

These pins are active during T4, T1 and T2 states and is returned to passive state (1,1,1 during T3 or Tw (when ready is inactive). These are used by the 8288 bus controller to generate all memory and I/O operation) access control signals. Any change by S2, S1, S0 during T4 is used to indicate the beginning of a bus cycle. These status lines are encoded as shown in table 3.

S2	S1	S0	Characteristics
0	0	0	Interrupt acknowledge
0	0	1	Read I/O port
0	1	0	Write I/O port
0	1	1	Halt
1	0	0	Code access1 0 1 Read memory
1	1	0	Write memory
1	1	1	Passive State

Table 3

QS0, QS1 (O): Queue – Status

Queue Status is valid during the clock cycle after which the queue operation is performed. QS0, QS1 provide status to allow external tracking of the internal 8086 instruction queue. The condition of queue status is shown in table 4.

Queue status allows external devices like In-circuit Emulators or special instruction set extension co-processors to track the CPU instruction execution. Since instructions are executed from the 8086 internal queue, the queue status is presented each CPU clock cycle and is not related to the bus cycle activity. This mechanism allows (1) A processor to detect execution of a ESCAPE instruction which directs the co- processor to perform a specific task and (2) An in-circuit Emulator to trap execution of a specific memory location.

QS1	QS1	Characteristics

0	0	No operation
0	1	First byte of opcode from queue
1	0	Empty the queue
1	1	Subsequent byte from queue

Table 4

LOCK (O)

It indicates to another system bus master, not to gain control of the system bus while LOCK is active Low. The LOCK signal is activated by the "LOCK" prefix instruction and remains active until the completion of the instruction. This signal is active Low and floats to tri-state OFF during 'hold acknowledge'. Example:

```
LOCK XCHG reg., Memory  ; Register is any register and memory GT0
                        ; is the address of the semaphore.
```

RQ/GT0 and RQ/GT1 (I/O): Request/Grant

These pins are used by other processors in a multi processor organization. Local bus masters of other processors force the processor to release the local bus at the end of the processors current bus cycle. Each pin is bi-directional and has an internal pull up resistors. Hence they may be left un-connected.

Flag Registers

The 8086 microprocessor has a 16 bit register for flag register. In this register 9 bits are active for flags. This register has 9 flags which are divided into two parts that are as follows

Conditional Flags

Conditional flags represent result of last arithmetic or logical instruction executed. Conditional flags are as follows:

1. **CF (Carry Flag)**

   This flag indicates an overflow condition for unsigned integer arithmetic. It is also used in multiple-precision arithmetic.

2. **AF (Auxiliary Flag)**

If an operation performed in ALU generates a carry/barrow from lower nibble (i.e. D0 – D3) to upper nibble (i.e. D4 – D7), the AF flag is set i.e. carry given by D3 bit to D4 is AF flag. This is not a general-purpose flag; it is used internally by the processor to perform Binary to BCD conversion.

3. **PF (Parity Flag)**

   This flag is used to indicate the parity of result. If lower order 8-bits of the result contains even number of 1's, the Parity Flag is set and for odd number of 1's, the Parity Flag is reset.

4. **ZF (Zero Flag)**

   It is set; if the result of arithmetic or logical operation is zero else it is reset.

5. **SF (Sign Flag)**

   In sign magnitude format the sign of number is indicated by MSB bit. If the result of operation is negative, sign flag is set.

6. **OF (Overflow Flag)**

   This stands for over flow flag. It occurs when signed numbers are added or subtracted. An OF indicates that the result has exceeded the capacity of machine. It becomes set if the sign result cannot express within the number of bites.

Control Flags

Control flags are set or reset deliberately to control the operations of the execution unit. Control flags are as follows:

1. **TF (Trap Flag):**

   It is used for single step control. It allows user to execute one instruction of a program at a time for debugging. When trap flag is set, program can be run in single step mode.

2. **IF (Interrupt Flag):**

   It is an interrupt enable/disable flag. This stands for interrupt flag. This flag is used to enable or disable the interrupt in a program. If it is set, the maskable interrupt of 8086 is enabled and if it is reset, the interrupt is disabled. It can be set by executing instruction sit and can be cleared by executing CLI instruction.

3. **DF (Direction Flag):**

This flag stands for direction flag and is used for the direction of strings. If it is set, string bytes are accessed from higher memory address to lower memory address. When it is reset, the string bytes are accessed from lower memory address to higher memory address

Unit-V

Instruction Set of 8086

The 8086 instructions are categorized into the following main types.

i. Data Copy / Transfer Instructions
ii. Arithmetic and Logical Instructions
iii. Branch Instructions
iv. Loop Instructions
v. Machine Control Instructions
vi. Flag Manipulation Instructions
vii. Shift and Rotate Instructions
viii. String Instructions

Data Copy / Transfer Instructions :

MOV :

This instruction copies a word or a byte of data from some source to a destination. The destination can be a register or a memory location. The source can be a register, a memory location, or an immediate number.

```
MOV  AX,BX
MOV  AX,5000H
MOV  AX,[SI]
MOV  AX,[2000H]
MOV  AX,50H[BX]
MOV  [734AH],BX
MOV  DS,CX
MOV  CL,[357AH]
```

Direct loading of the segment registers with immediate data is not permitted.

2

PUSH : Push to Stack

This instruction pushes the contents of the specified register/memory location on to the stack. The stack pointer is decremented by 2, after each execution of the instruction.

E.g. PUSH AX

- PUSH DS
- PUSH [5000H]

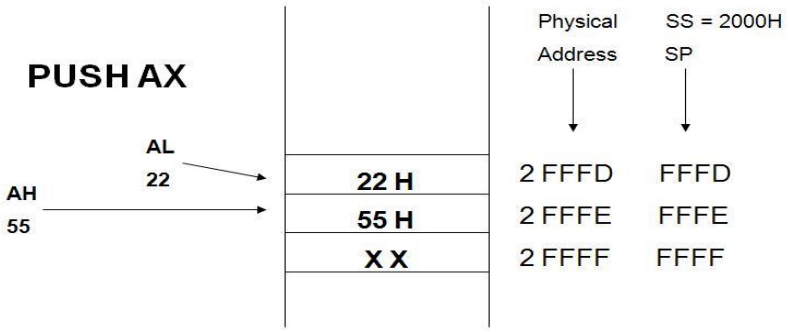


Fig. 2.2 Push Data to stack memory

POP : Pop from Sack

This instruction when executed, loads the specified register/memory location with the contents of the memory location of which the address is formed using the current stack segment and stack pointer.

The stack pointer is incremented by

2 Eg. POP AX

POP DS POP [5000H]

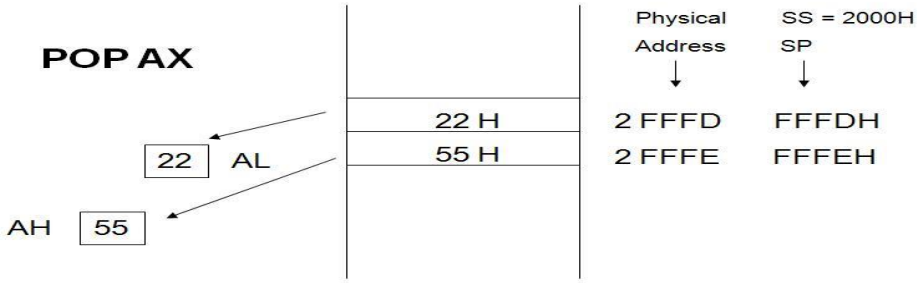


Fig 2.3 Popping Register Content from Stack Memory

XCHG : Exchange byte or word

This instruction exchange the contents of the specified source and destination operands

Eg. XCHG [5000H], AX

XCHG BX, AX

XLAT :

Translate byte using look-up table

Eg. LEA BX, TABLE1
MOV AL, 04H
XLAT

Simple input and output port transfer Instructions:

IN:

Copy a byte or word from specified port to accumulator.

Eg. IN AL,03H
IN AX,DX

OUT:

Copy a byte or word from accumulator specified port.

Eg. OUT 03H, AL
OUT DX, AX

LEA :

Load effective address of operand in specified register.
[reg] offset portion of address in DS

Eg. LEA reg, offset

LDS:

Load DS register and other specified register from memory.

[reg] [mem]
[DS] [mem + 2]

Eg. LDS reg, mem

LES:

Load ES register and other specified register from memory.

[reg] [mem]
[ES] [mem + 2]

Eg. LES reg, mem

Flag transfer instructions:

LAHF:

Load (copy to) AH with the low byte the flag register.
[AH] ← [Flags low byte]

Eg. LAHF

SAHF:

Store (copy) AH register to low byte of flag register.
[Flags low byte] ⟵ [AH]
Eg. SAHF

PUSHF:

Copy flag register to top of stack.
[SP] ⟵ [SP] – 2
[[SP]] ⟵ [Flags]
Eg. PUSHF

POPF :

Copy word at top of stack to flag register.
[Flags] ⟵ [[SP]]
[SP] ⟵ [SP] + 2

Arithmetic Instructions:

The 8086 provides many arithmetic operations: addition, subtraction, negation, multiplication and comparing two values.

ADD :

The add instruction adds the contents of the source operand to the destination operand.
Eg.

```
ADD AX, 0100H
ADD AX, BX
ADD AX, [SI]
ADD AX, [5000H]
ADD [5000H], 0100H
ADD 0100H
```

ADC : Add with Carry

This instruction performs the same operation as ADD instruction, but adds the carry flag to the result.

```
Eg.  ADC  0100H AX,
     ADC  BX AX,
     ADC  [SI] AX,
     ADC  [5000]
          [5000], 0100H
     ADC
     ADC
```

SUB : Subtract

The subtract instruction subtracts the source operand from the destination operand and the result is left in the destination operand.

Eg. SUB AX, 0100H

SUB AX, BX
SUB AX,
[5000H]
SUB [5000H], 0100H

SBB : Subtract with Borrow

The subtract with borrow instruction subtracts the source operand and the borrow flag (CF) which may reflect the result of the previous calculations, from the destination

SBB AX, 0100H
SBB AX, BX SBB
AX, [5000H]
SBB [5000H], 0100H .

INC : Increment

This instruction increases the contents of the specified Register or memory location by 1. Immediate data cannot be operand of this instruction.

Eg. INC AX INC
[BX] INC
[5000H]

DEC : Decrement

The decrement instruction subtracts 1 from the contents of the specified register or memory location.

Eg. DEC AX DEC
[5000H]

NEG : Negate

The negate instruction forms 2's complement of the specified destination in the instruction. The destination can be a register or a memory location. This instruction can be implemented by inverting each bit and adding 1 to it.

Eg. NEG AL

AL = 0011 0101 35H Replace number in AL with its 2's complement
AL = 1100 1011 = CBH

CMP : Compare

This instruction compares the source operand, which may be a register or an immediate data or a memory location, with a destination operand that may be a

register or a memory location

Eg. CMP BX, 0100H CMP
AX, 0100H

CMP [5000H], 0100H
CMP BX, [SI]
CMP BX, CX

MUL :Unsigned Multiplication Byte or Word

This instruction multiplies an unsigned byte or word by the contents of AL.

```
Eg.  MUL BH                  ;  (AX)        (AL) x (BH)
     MUL CX                  ;  (DX)(AX)    (AX) x (CX)
     MUL WORD PTR [SI]       ;  (DX)(AX)    (AX) x ([SI])
```

IMUL :Signed Multiplication

This instruction multiplies a signed byte in source operand by a signed byte in AL or a signed word in source operand by a signed word in AX.

```
Eg. IMUL BH
    IMUL CX
    IMUL [SI]
```

CBW : Convert Signed Byte to Word

This instruction copies the sign of a byte in AL to all the bits in AH. AH is then said to be sign extension of AL.

Eg. CBW

AX= 0000 0000 1001 1000 Convert signed byte in AL signed word in AX.
Result in AX = 1111 1111 1001 1000

CWD : Convert Signed Word to Double Word

This instruction copies the sign of a byte in AL to all the bits in AH. AH is then said to be sign extension of AL.

Eg. CWD

Convert signed word in AX to signed double word in DX : AX
DX= 1111 1111 1111 1111
Result in AX = 1111 0000 1100 0001

DIV : Unsigned division

This instruction is used to divide an unsigned word by a byte or to divide an unsigned double word by a word.

```
Eg.  DIV CL    ;  Word in AX / byte in CL
                  ;  Quotient in AL, remainder in AH
     DIV CX    ;   Double word in DX and AX / word
                  ;  in CX, and Quotient in AX,
                  ;   remainder in DX
```

AAA : ASCII Adjust After Addition

The AAA instruction is executed aftr an ADD instruction that adds two ASCII coded operand to give a byte of result in AL. The AAA instruction converts the resulting contents of Al to a unpacked decimal digits.

```
Eg.  ADD CL, DL    ;  [CL] = 32H = ASCII for 2
                   ;  [DL] = 35H = ASCII for 5
                   ;  Result [CL] = 67H
     MOV AL, CL    ;  Move ASCII result into AL since
                   ;   AAA adjust only [AL]
     AAA           ;   [AL]=07, unpacked BCD for 7
```

AAS : ASCII Adjust AL after Subtraction

This instruction corrects the result in AL register after subtracting two unpacked ASCII operands. The result is in unpacked decimal format. The procedure is similar to AAA instruction except for the subtraction of 06 from AL.

AAM : ASCII Adjust after Multiplication

This instruction, after execution, converts the product available In AL into unpacked BCD format.

```
Eg.  MOV AL, 04     ;  AL = 04
     MOV BL ,09     ;  BL = 09
     MUL BL         ;  AX = AL*BL  ; AX=24H
     AAM            ;  AH = 03,  AL=06
```

AAD : ASCII Adjust before Division

This instruction converts two unpacked BCD digits in AH and AL to the equivalent binary number in AL. This adjustment must be made before dividing the two unpacked BCD digits in AX by an unpacked BCD byte. In the instruction sequence, this instruction appears Before DIV instruction.

Eg. AX 05 08

AAD result in AL 00 3A 58D = 3A H in AL

The result of AAD execution will give the hexadecimal number 3A in AL and 00 in AH. Where 3A is the hexadecimal Equivalent of 58 (decimal).

DAA : Decimal Adjust Accumulator

This instruction is used to convert the result of the addition of two packed BCD numbers to a valid BCD number. The result has to be only in AL.

```
Eg.  AL = 53CL = 29
     ADD AL, CL     ; AL   (AL) + (CL)
                    ; AL   53 + 29
                    ; AL   7C
     DAA            ; AL   7C + 06 (as C>9)
                    ; AL   82
```

DAS : Decimal Adjust after Subtraction

This instruction converts the result of the subtraction of two packed BCD numbers to a valid BCD number. The subtraction has to be in AL only.

Eg. AL = 75, BH = 46

```
SUB AL, BH      ; AL    2 F =  (AL) - (BH)
                ; AF = 1
DAS             ; AL    2 9 (as F>9, F - 6 = 9)
```

Logical Instructions

AND : Logical AND

This instruction bit by bit ANDs the source operand that may be an immediate register or a memory location to the destination operand that may a register or a memory location. The result is stored in the destination operand.

```
Eg.   AND AX, 0008H
      AND AX, BX
```

OR : Logical OR

This instruction bit by bit ORs the source operand that may be an immediate , register or a memory location to the destination operand that may a register or a memory location. The result is stored in the destination operand.

```
Eg.   OR AX, 0008H
      OR AX, BX
```

NOT : Logical Invert

This instruction complements the contents of an operand register or a memory location, bit by bit.

```
Eg.   NOT AX
      NOT [5000H]
```

XOR : Logical Exclusive OR

This instruction bit by bit XORs the source operand that may be an immediate , register or a memory location to the destination operand that may a register or a memory location. The result is stored in the destination operand.

```
Eg.   XOR AX, 0098H
      XOR AX, BX
```

TEST : Logical Compare Instruction

The TEST instruction performs a bit by bit logical AND operation on the two operands. The result of this ANDing operation is not available for further use, but flags are affected.

```
Eg.TEST     AX, BX
   TEST     [0500], 06H
```

SAL/SHL : SAL / SHL destination, count.

SAL and SHL are two mnemonics for the same instruction. This instruction shifts each bit in the specified destination to the left and 0 is stored at LSB position. The MSB is shifted into the carry flag. The destination can be a byte or a word.
It can be in a register or in a memory location. The number of shifts is indicated

by count.

```
SAL CX, 1
SAL AX, CL
```

SHR : SHR destination, count

This instruction shifts each bit in the specified destination to the right and 0 is stored at MSB position. The LSB is shifted into the carry flag. The destination can be a byte or a word.

It can be a register or in a memory location. The number of shifts is indicated by count.

```
Eg.  SHR CX, 1
     MOV CL,
     05H SHR AX,
     CL
```

SAR : SAR destination, count

This instruction shifts each bit in the specified destination some number of bit positions to the right. As a bit is shifted out of the MSB position, a copy of the old MSB is put in the MSB position. The LSB will be shifted into CF.

```
Eg.  SAR BL, 1
     MOV CL,
     04H SAR DX,
     CL
```

ROL Instruction : ROL destination, count

This instruction rotates all bits in a specified byte or word to the *left* some number of bit positions. MSB is placed as a new LSB and a new CF.

```
Eg.  ROL CX, 1
     MOV CL, 03H
     ROL BL, CL
```

ROR Instruction : ROR destination, count

This instruction rotates all bits in a specified byte or word to the *right* some number of bit positions. LSB is placed as a new MSB and a new CF.

```
Eg.  ROR CX, 1
     MOV CL, 03H
     ROR BL, CL
```

RCL Instruction : RCL destination, count

This instruction rotates all bits in a specified byte or word some number of bit positions to the *left along with the carry flag*. MSB is placed as a new carry and previous carry is place as new LSB.

```
Eg.   RCL CX, 1
      MOV CL, 04H
      RCL AL, CL
```

RCR Instruction : RCR destination, count

This instruction rotates all bits in a specified byte or word some number of bit positions to the right *along with the carry flag*. LSB is placed as a new carry and previous carry is place as new MSB.

```
Eg.   RCR CX, 1
      MOV CL, 04H
      RCR AL, CL
```

ROR Instruction : ROR destination, count

This instruction rotates all bits in a specified byte or word to the *right* some number of bit positions. LSB is placed as a new MSB and a new CF.

```
Eg.   ROR CX, 1
      MOV CL, 03H
      ROR BL, CL
```

RCL Instruction : RCL destination, count

This instruction rotates all bits in a specified byte or word some number of bit positions to the *left along with the carry flag*. MSB is placed as a new carry and previous carry is place as new LSB.

```
Eg.   RCL  CX, 1
      MOV  CL, 04H
      RCL  AL, CL
```

RCR Instruction : RCR destination, count

This instruction rotates all bits in a specified byte or word some number of bit positions to the right *along with the carry flag*. LSB is placed as a new carry and previous carry is place as new MSB.

```
Eg.   RCR CX, 1
      MOV CL, 04H
      RCR AL, CL
```

Branch Instructions :

Branch Instructions transfers the flow of execution of the program to a new address specified in the instruction directly or indirectly. When this type of instruction is executed, the CS and IP registers get loaded with new values of CS and IP corresponding to the location to be transferred.

The Branch Instructions are classified into two types

i. Unconditional Branch Instructions.

ii. Conditional Branch Instructions.

Unconditional Branch Instructions :

In Unconditional control transfer instructions, the execution control is transferred to the specified location independent of any status or condition. The CS and IP are unconditionally modified to the new CS and IP.

CALL : Unconditional Call

This instruction is used to call a Subroutine (Procedure) from a main program. Address of procedure may be specified directly or indirectly.

There are two types of procedure depending upon whether it is available in the same segment or in another segment.

i. Near CALL i.e., ±32K displacement.

ii. For CALL i.e., anywhere outside the segment.

On execution this instruction stores the incremented IP & CS onto the stack and loads the CS & IP registers with segment and offset addresses of the procedure to be called.

RET: Return from the Procedure.

At the end of the procedure, the RET instruction must be executed. When it is executed, the previously stored content of IP and CS along with Flags are retrieved into the CS, IP and Flag
registers from the stack and execution of the main program continues further.

INT N: Interrupt Type N.

In the interrupt structure of 8086, 256 interrupts are defined corresponding to the types from 00H to FFH. When INT N instruction is executed, the type byte N is multiplied by 4 and the contents of IP and CS of the interrupt service routine will be taken from memory block in 0000 segment.

INTO: Interrupt on Overflow

This instruction is executed, when the overflow flag OF is set. This is equivalent to a Type 4 Interrupt instruction.

JMP: Unconditional Jump

This instruction unconditionally transfers the control of execution to the specified address using an 8-bit or 16-bit displacement. No Flags are affected by this instruction.

IRET: Return from ISR

When it is executed, the values of IP, CS and Flags are retrieved from the stack to continue the execution of the main program.

LOOP : LOOP Unconditionally

This instruction executes the part of the program from the Label or address specified in the instruction upto the LOOP instruction CX number of times. At each iteration, CX is decremented automatically and JUMP IF NOT ZERO structure.

```
Example:    MOV CX, 0004H
            MOV BX, 7526H
            Label 1 MOV AX, CODE
            OR  BX, AX
            LOOP Label 1
```

Conditional Branch Instructions

When this instruction is executed, execution control is transferred to the address specified relatively in the instruction, provided the condition implicit in the Opcode is satisfied. Otherwise execution continues sequentially.

JZ/JE Label

Transfer execution control to address 'Label', if ZF=1.

JNZ/JNE Label

Transfer execution control to address 'Label', if ZF=0

JS Label

Transfer execution control to address 'Label', if SF=1.

JNS Label

Transfer execution control to address 'Label', if SF=0.

JO Label

Transfer execution control to address 'Label', if OF=1.

JNO Label

Transfer execution control to address 'Label', if OF=0.

JNP Label

Transfer execution control to address 'Label', if PF=0.

JP Label

Transfer execution control to address 'Label', if PF=1.

JB Label

Transfer execution control to address 'Label', if CF=1.

JNB Label

Transfer execution control to address 'Label', if CF=0.

JCXZ Label

Transfer execution control to address 'Label', if CX=0

Conditional LOOP Instructions.

LOOPZ / LOOPE Label

Loop through a sequence of instructions from label while ZF=1 and CX=0.

LOOPNZ / LOOPENE Label

Loop through a sequence of instructions from label while ZF=1 and CX=0.

String Manipulation Instructions

A series of data byte or word available in memory at consecutive locations, to be referred as Byte String or Word String. A String of characters may be located in consecutive memory locations, where each character may be represented by its ASCII equivalent.

The 8086 supports a set of more powerful instructions for string manipulations for referring to a string, two parameters are required.

I. Starting and End Address of the String.
II. Length of the String.

The length of the string is usually stored as count in the CX register.The incrementing or decrementing of the pointer, in string instructions, depends upon the Direction Flag (DF) Status. If it is a Byte string operation, the index registers are updated

by one. On the other hand, if it is a word string operation, the index registers are updated by two.

REP : Repeat Instruction Prefix

This instruction is used as a prefix to other instructions, the instruction to which the REP prefix is provided, is executed repeatedly until the CX register becomes zero (at each iteration CX is automatically decremented by one).

i. REPE / REPZ- repeat operation while equal / zero.
ii. REPNE / REPNZ - repeat operation while not equal / not zero.

These are used for CMPS, SCAS instructions only, as instruction prefixes.

MOVSB / MOVSW :Move String Byte or String Word

Suppose a string of bytes stored in a set of consecutive memory locations is to be moved to another set of destination locations.The starting byte of source string is located in the memory location whose address may be computed using SI (Source Index) and DS (Data Segment) contents.

The starting address of the destination locations where this string has to be relocated is given by DI (Destination Index) and ES (Extra Segment) contents.

CMPS : Compare String Byte or String Word

The CMPS instruction can be used to compare two strings of byte or words. The length of the string must be stored in the register CX. If both the byte or word strings are equal, zero Flag is set.

The REP instruction Prefix is used to repeat the operation till CX (counter) becomes zero or the condition specified by the REP Prefix is False.

SCAN : Scan String Byte or String Word

This instruction scans a string of bytes or words for an operand byte or word specified in the register AL or AX. The String is pointed to by ES:DI register pair. The length of the string s stored in CX. The DF controls the mode for scanning of the string. Whenever a match to the specified operand, is found in the string, execution stops and the zero Flag is set. If no match is found, the zero flag is reset.

LODS : Load String Byte or String Word

The LODS instruction loads the AL / AX register by the content of a string pointed to by DS : SI register pair. The SI is modified automatically depending upon DF, If it is a byte transfer (LODSB), the SI is modified by one and if it is a word transfer (LODSW), the SI is modified by two. No other Flags are affected by this instruction.

STOS : Store String Byte or String Word

The STOS instruction Stores the AL / AX register contents to a location in the string pointer by ES : DI register pair. The DI is modified accordingly, No Flags are affected by this instruction.

The direction Flag controls the String instruction execution, The source index SI and Destination Index DI are modified after each iteration automatically. If DF=1, then the execution follows autodecrement mode, SI and DI are decremented automatically after each iteration. If DF=0, then the execution follows autoincrement mode. In this mode, SI and DI are incremented automatically after each iteration.

Flag Manipulation and a Processor Control Instructions

These instructions control the functioning of the available hardware inside the processor chip. These instructions are categorized into two types:

1. Flag Manipulation instructions.
2. Machine Control instructions.

Flag Manipulation instructions

The Flag manipulation instructions directly modify some of the Flags of 8086.

i. CLC – Clear Carry Flag.
ii. CMC – Complement Carry Flag.
iii. STC – Set Carry Flag.
iv. CLD – Clear Direction Flag.
v. STD – Set Direction Flag.
vi. CLI – Clear Interrupt Flag.
vii. STI – Set Interrupt Flag.

Machine Control instructions

The Machine control instructions control the bus usage and execution

i. WAIT – Wait for Test input pin to go low.
ii. HLT – Halt the process.
iii. NOP – No operation.
iv. ESC – Escape to external device like NDP
v. LOCK – Bus lock instruction prefix.

8086 Assembly Language Program 16 Bit Addition

```
DATA SEGMENT
   NUM DW 1234H, 0F234H
   SUM DW 2 DUP(0)
DATA ENDS
CODE SEGMENT
   ASSUME CS: CODE, DS:DATA
   START: MOV AX,DATA
   MOV DS,AX
   MOV AX,NUM ; First number loaded into AX
   MOV BX,0H ; For carry BX register is cleared
   ADD AX,NUM+2 ; Second number added with AX
   JNC DOWN ; Check for carry
   INC BX ; If carry generated increment the BX
   DOWN: MOV SUM,AX ; Storing the sum value
   MOV SUM+2,BX ; Storing the carry value
   MOV AH,4CH
   INT 21H
CODE ENDS
END START


INPUT : 1234H, F234H
OUTPUT : 10468H
```

8086 Assembly Language Program 16 Bit Subtraction

```
DATA SEGMENT
NUM DW 4567H,2345H
DIF DW 1 DUP(0)
DATA ENDS
CODE SEGMENT
ASSUME
CS:CODE,DS:DATA
START: MOV AX,DATA
MOV DS,AX
CLC
; Clearing Carry
LEA SI,NUM ; SI pointed to the NUM
MOV AX,[SI] ; Move NUM1 to AX
SBB AX,[SI+2] ; Move the SI to Num2 and subtract with AX(Takes
;care for both smaller as well as larger
;Number subtraction)
```

```
MOV DIF,AX ;Store the result
MOV AH,4CH
INT 21H
CODE ENDS
END START
```

8086 Assembly Language Program-16 Bit multiplication for unsigned numbers

```
DATA SEGMENT
  NUM DW 1234H,1234H
  PROD DW 2 DUP(0)
DATA ENDS
CODE SEGMENT
  ASSUME CS:CODE,DS:DATA
  START: MOV AX,DATA
  MOV DS,AX
  LEA SI,NUM ; SI pointed to the Multiplicand
  MOV AX,[SI] ; Multiplicand has to be in AX register
  MOV BX,[SI+2] ; SI+2 pointed to the Multiplier and move it to BX
  MUL BX ;Perform the multiplication
  MOV PROD,AX ;32 bit product stored in DX-AX registers
  MOV PROD+2,DX
  MOV AH,4CH
  INT 21H
CODE ENDS
END START
```

8086 Assembly Language Program 16 Bit Multiplication for signed numbers

```
DATA SEGMENT
NUM DW -2,1
PROD DW 2 DUP(0)
DATA ENDS
CODE SEGMENT
ASSUME CS:CODE,DS:DATA
START: MOV AX,DATA
MOV DS,AX
LEA SI,NUM ; SI pointed to the Multiplicand
MOV AX,[SI] ; Multiplicand has to be in AX register
MOV BX,[SI+2] ; SI+2 pointed to the Multiplier and move it to BX
IMUL BX ; Perform the sign multiplication using sign
;Multiplication operator (IMUL)
MOV PROD,AX ; 32 bit product stored in DX-AX registers
MOV PROD+2,DX
```

```
MOV AH,4CH
INT 21H
CODE ENDS
END START
```

8086 Assembly Language Program 16 Bit Division for Unsigned Numbers

```
DATA SEGMENT
NUM1 DW 4567H,2345H
NUM2 DW 4111H
QUO DW 2 DUP(0)
REM DW 1 DUP(0)
DATA ENDS
CODE SEGMENT
ASSUME CS:CODE,DS:DATA
START: MOV AX,DATA
MOV DS,AX
MOV AX,NUM1 ;Move the lower bit of Dividend to AX
MOV DX,NUM1+2
DIV NUM2
MOV QUO,AX
MOV REM,DX
MOV AH,4CH
INT 21H
CODE ENDS
END START
```

8086 Assembly Language Program for given data is positive or negative

```
DATA SEGMENT
NUM DB 12H
MES1 DB 10,13,'DATA IS POSITIVE $'
MES2 DB 10,13,'DATA IS NEGATIVE $'
DATA ENDS
CODE SEGMENT
ASSUME CS:CODE,DS:DATA
START: MOV AX,DATA
MOV DS,AX
MOV AL,NUM
ROL AL,1
JC NEGA
```

```
;Move the Number to AL.
;Perform the rotate left side for 1 bit position.
;Check for the negative number.
MOV DX,OFFSET MES1 ;Declare it positive.
JMP EXIT ;Exit program.
NEGA: MOV DX,OFFSET MES2;Declare it negative.
EXIT: MOV AH,09H
INT 21H
MOV AH,4CH
INT 21H
CODE ENDS
END START
```

8086 Assembly Language Program for given data is odd or even

```
DATA SEGMENT
X DW 27H
MSG1 DB 19,13,'NUMBER IS EVEN$'
MSG2 DB 10,13,'NUMBER IS ODD$'
DATA ENDS
CODE SEGMENT
ASSUME CS:CODE,DS:DATA
START: MOV AX,DATA
MOV DS,AX
MOV AX,X
TEST AX,01H
JNZ EXIT
LEA DX,MSG1
MOV AH,09H
INT 21H
JMP LAST
;Test for Even/Odd number.
;If it is Even go to Exit label.
;(alternate logic)
;MOV BL,2
;DIV BL
;CMP AH,0H
;JNZ EXIT
;Declare it is Even number.
EXIT: LEA DX,MSG2 ;Declare it is Odd number.
MOV AH,09H
INT 21H
LAST: MOV AH,4CH
INT 21H
CODE ENDS
END START
```

8086 Assembly Language Program to find Bit wise palindrome

```
DATA SEGMENT
  X DW 0FFFFH
  MSG1 DB 10,13,'NUMBER IS PALINDROME$'
  MSG2 DB 10,13,'NUMBER IS NOT PALINDROME$'
DATA ENDS
CODE SEGMENT
 ASSUME CS:CODE,DS:DATA
 START: MOV AX,DATA     ;Load the Data to AX.
       MOV DS,AX       ;Move the Data AX to DS.
       MOV AX,X        ;Move DW to AX.
       MOV CL,10H      ;Initialize the counter 10.
   UP: ROR AX,1        ;Rotate right one time.
      RCL DX,1        ;Rotate left with carry one time.
      LOOP UP         ;Loop the process.
      CMP AX,DX       ;Compare AX and DX.
      JNZ DOWN        ;If no zero go to DOWN label.
      LEA DX,MSG1     ;Declare as a PALINDROME.
      MOV AH,09H
      INT 21H
      JMP EXIT        ;Jump to EXIT label.
  DOWN: LEA DX,MSG2     ; Declare as not a PALINDROME
       MOV AH,09H
       INT 21H
  EXIT:MOV AH,4CH
      INT 21H
CODE ENDS
END START
```

8086 Assembly Language Program to Find Largest Number Among the Series

```
DATA SEGMENT
X DW 0010H,52H,30H,40H,50H
LAR DW ?
DATA ENDS
CODE SEGMENT
ASSUME CS:CODE,DS:DATA
START: MOV AX,DATA
MOV DS,AX
MOV CX,05H
LEA SI,X
```

```
MOV AX,[SI]
DEC CX
UP: CMP AX,[SI+2]
JA CONTINUE
MOV AX,[SI+2]
CONTINUE:ADD SI,2
DEC CX
JNZ UP
MOV LAR,AX
MOV AH,4CH
INT 21H
CODE ENDS
END START
```

8086 Assembly Language Program to find smallest number among the series

```
DATA SEGMENT
  X DW 0060H,0020H,0030H,0040H,0050H
  MES DB 10,13,'SMALLEST NUMBER AMONG THE SERIES IS $'
DATA ENDS
CODE SEGMENT
  ASSUME CS:CODE,DS:DATA
  START: MOV AX,DATA
        MOV DS,AX
        MOV CX,05H
        LEA SI,X
        MOV AX,[SI]
        DEC CX
    UP: CMP AX,[SI+2]
        JB CONTINUE
        MOV AX,[SI+2]
 CONTINUE:ADD SI,2
         DEC CX
         JNZ UP
         AAM
         ADD AX,3030H
         MOV BX,AX
         MOV AH,09H
         LEA DX,MES
         INT 21H
         MOV DL,BH
         MOV AH,02H
         INT 21H
         MOV DL,BL
         INT 21H
```

```
        MOV AH,4CH
        INT 21H
CODE ENDS
END START
```

8086 Assembly Language Program to compute the factorial of a positive integer 'n' using recursive procedure.

```
DATA SEGMENT
     N DB 06H
      FACT DW ?
DATA ENDS
CODE SEGMENT
ASSUME CS:CODE, DS:DATA
START:
       MOV AX, DATA
       MOV DS, AX
       MOV AX, 1
       MOV BL, N
       MOV BH, 0
       CALL FACTORIAL
       MOV FACT, AX
       MOV AH, 4CH
       INT 21H

       FACTORIAL PROC
       CMP BX, 1
       JE L1
       PUSH BX
       DEC BX
       CALL FACTORIAL
       POP BX
       MUL BX
L1:   RET
       FACTORIAL ENDP
CODE ENDS
END START
```

8086 Assembly Language Program to find square and cube of a number

```
DATA SEGMENT
  X DW 04H
  SQUARE DW ?
  CUBE DW ?
```

```
DATA ENDS
CODE SEGMENT
  ASSUME CS:CODE,DS:DATA
  START: MOV AX,DATA
  MOV DS,AX
  MOV AX,X
  MOV BX,X
  MUL BX
  MOV SQUARE,AX
  MUL BX
  MOV CUBE,AX
  MOV AH,4CH
  INT 21H
CODE ENDS
END START
```

15. Additional Topics

1. Computer Types
2. Bus Structure
3. Different I/O Devices
4. Arithmetic operations
5. EMU8086 simulator

16. UNIVERSITYPREVIOUS QUESTION PAPERS:

(2013)

1. Perform arithmetic on the following numbers using their binary equivalents
   i) 3+4 ii) -3+4 iii) 3-4 iv) -3-4 [16]

2.a) Explain the instruction cycle with a neat flow chart.
b) Explain RISC. [16]

3.a) Explain booths multiplication algorithm with 2's compliment data.
b) Explain about the floating point arithmetic operations. [16]

4.a) Explain about microprogrammed control.
b) Explain about sequencer. [16]

5. Define virtual memory. Explain the process of converting virtual address to physical address with a neat diagram. [16]

6.a) Explain about asynchronous data transfer.
b) Explain about modes of transfer. [16]

7.a) Explain the execution of four segment CPU pipeline with a neat diagram.
b) List out the limitations of instruction pipeline. [16]

8.a) What is cache coherence problem? Explain the conditions for incoherence.
b) Discuss the mechanisms and possible solutions to overcome cache coherence problem. [16]

II B.Tech II Semester Examinations, May/June 2012
COMPUTER ORGANIZATION

1. Design a 4 bit binary adder - subtract or circuit which has the following values for input mode M and data inputs A and B in each case determine the values of the outputs: s3 , s2 , s − 1,s0 and s4. [15]
M A B
(a) 0 0111 0110
(b) 0 1000 1001
(c) 1 1100 1000
(d) 1 0101 1010
2. (a) What is meant by cycle stealing? Explain?
(b) List and explain data transfer signals on the PCI bus/ . [7+8]
3. Write in detail about loosely coupled micro processor and tightly coupled micro processors. [15]
4. (a) Give the register configuration for floating point arithmetic operations.
(b) Multiply +15 with -13 using Booth algorithm. [7+8]
5 (a) How to deal with overflow in integer arithmetic? Explain with appropriate examples.
(b) What are the influencing factors of performance improvement in a computer? [7+8]
6. (a) What is the general relationship among access time, memory cost and capacity?
(b) How does SDRAM differ from ordinary DRAM? [7+8]
7. (a) Consider a computer with a four floating point pipeline processors. Supports that each processor uses a cycle time of 20ns. How along will it take to perform 200 floating point operations? Is there any difference if the same 200 operations are carried out using a single pipeline processor with a cycle type of 10ns?
(b) Discuss the features of super computer? [10+5]
8. (a) Support or Oppose the statement “Instruction set architecture has impact on the processors micro architecture”.
(b) What is a pipeline register? What is the use of it? Explain? [7+8] ? ? ? ? ?

1. (a) Design a functional unit using 4*1 MUX to output AND, OR, XOR or NAND between two bits.
(b) Discuss Organization of stack and arithmetic expression evaluation using stack. [7+8]
2. Explain various methods that enable a device to communicate with the processor. Use necessary diagrams to support your explanation. [15]
3. Describe in detail the concept of address sequencing in control memory? [15]
4. (a) What is a pipeline? Illustrate the behavior of a pipeline with a space time diagram?
(b) Discuss about multiple module memory organization? [7+8]
5. (a) Discuss the usage of TSL instruction?
(b) What is a critical section? Explain?
(c) Give a note on mutual exclusion mechanism? [5+5+5]
6. A tape drive has the following parameters:
Bit density 2000 bits/c Tape speed 800 cm/s Time to reverse direction of motion 225 ms

Minimum time spent at an inter record gap 3 ms Average record length 4000 characters Estimate the percentage gain in time resulting from the ability to read records in both the forward and backward directions. Assume that records are accessed at random and that on average, the distance between two records accessed in sequence is 4 records. [15]
7. Derive an algorithm in flowchart from for the non restoring method of fixed point binary division. [15]
8. (a) Give a detail note on SPEC rating?
(b) Show the bit configuration of a 24 bit register when its content represents the decimal equivalent of 295
i. in binary
ii. in BCD
iii. in ASIII using 8 bit with EVEN parity. [7+8]

II B.Tech II Semester Examinations,May/June 2012
COMPUTER ORGANIZATION

1. What is parallel processing? Explain Flynn's classification of computer. Explain memory interleaving? [15]
2. Explain briefly various inter connection structures for multiprocessor system. [15]
3. Explain micro instruction format for the control memory and various micro instructions [15]
4. (a) List and explain the SCSI bus signals.
(b) Draw a flowchart to describe the CPU-I/O channel communication in the IBM 370? [7+8]
5. (a) Discuss about logical and shift micro operations?
(b) What is a stack? Explain push and pop instructions using stack with examples? [7+8]
6. (a) Prove that the multiplication of two n- digit numbers in base B gives a product of no more than 2n digits.
(b) Discuss the hardware implementation of Booth algorithms. [7+8]
7. (a) What is an Optimizing Compiler? What is its role in performance improval?
(b) Define Elapsed time?
(c) Give a detail note on alphanumeric representation? [7+3+5]
8. (a) A digital computer has a memory unit of 64k 16 and a cache memory of 1k words. The cache uses direct mapping with a block size of 4 words.
i. How many bits are there in the tag, index, block and word fields of the address format?
ii. How many bits are there in each word of cache and how are they divided into function? Include a valid bit.
(b) Discuss about Ram bus memory? [10+5]
? ? ? ? ?

II B.Tech II Semester Examinations,May/June 2012
COMPUTER ORGANIZATION

1. (a) Discuss the main virtue of the single bus structure? 6m
(b) Convert the following numbers with the indicated bases to Decimal.
i. (12121)3
ii. (4310)5
iii. (50)7
iv. (198)12
v. (F 3A7C2)16 . [7+8]
2. (a) Discuss the four types of commands for an i/o interface?
(b) Explain briefly the concept of Asynchronous data transfer? [7+8]
3. (a) Explain functioning of a control unit. Discuss the terms control word,control memory, control register and control buffer register.
(b) Distinguish between micro programmed control and hardwired control. [10+5]
4. (a) Discuss the delayed branch in RISC?
(b) Explain an auxiliary array processor of a array computer? [7+8]
5. Derive a Combinational circuit that selects and generates any of the 16 logic micro operations? [15]
6. (a) Write a detailed note on shared memory multi processors.
(b) Draw a diagram showing the structure of 4D hypercube network. List all the paths available from node 7 to node 9. That use a minimum number of intermediate nodes. [5+10]
7. (a) Explain about the normalized and biased exponent.
(b) Perform (-35)-(+40), use serial bits and signed 2's complement. [7+8]
8. (a) Discuss about ram bus memory?
(b) Write a detail note on magnetic disk principles? [5+10]

1. (a) Diferentiate between microprogramming and nanoprogramming.
(b) Hardwired control unit is faster than microprogammed control unit. Justify this statement.
2. What are the different types of Mapping Techniques used in the usage of Cache Memory? Explain.
3. (a) Draw a flowchart to explain how addition and subtraction of two fixed point numbers can be done. Also, draw a circuit using full adders for the same.
(b) Explain Booth's alogorithm with its theoretical basis.
4. (a) Draw the block diagram of a computer system and describe each of its parts along with their functions. Also designate the information flow between the parts with arrows.
(b) Explain the term `memory bus bottleneck'.
(c) Distinguish between multiprocessor and a multicomputer.
5. Explain the following:
(a) Asynchronous Serial Transfer
(b) Asynchronous Communication Interface.
6. (a) What is meant by arithmetic pipeline? Explain.
(b) Explain pipeline for floating point addition and subtraction.
7. (a) Explain commonly employed bit shift operators such as shift left, right, circular

shift left/right and arithmetic shift left/right. Give 8-bit examples.
(b) Design a circuit for combined shift left/right operations. Assume register length is 4 bits and employ RS flip-flops.
8. (a) Explain multiport memory organization with a neat sketch.
(b) Explain system bus structure for multiprocessors with a neat sketch.
9. (a) How many bits are needed to store the result addition, subtraction, multiplication and division of two n-bit unsigned numbers. Prove.
(b) What is overflow and underflow? What is the reason? If the computer is considered as infinite system do we still have these problems.
10. (a) Design a circuit transferring data from a 4bit register which uses D ip-ops to another register which employs RS flip-flops.
(b) What are register transfer logic languages? Explain few RTL statement for branching with their actual functioning.
11. Draw the general block diagram of a microsequencer. Explain clearly the inputs and outputs of the same along with their functioning.
12. (a) Explain bit oriented and character oriented protocols in serial communication.
(b) What are the different issues behind serial communication? Explain.
13. (a) Find out what are the actual values of the following IEEE 754 single precision number.
i. Sign=0, exponent=all 1s and fraction _field is not 0.
ii. Sign=0, exponent=all 0s and fraction _field is 0.
iii. Sign=1, exponent=all 1s and fraction _field is 0.
iv. Sign=0, exponent=1000 0001, fraction=1100 0000 0000 0000 0000 000
(b) Why is bus arbitration required. Discuss about any two bus arbitration methods.
14. (a) Explain how Flynn classified the processors into different streams by giving an example for each stream.
(b) Explain tightly coupled and loosely coupled systems with suitable examples.

15. Explain the following in related with Vector Processing
(a) Super Computers
(b) Vector operations
(c) Matrix multiplication
(d) Memory interleaving
16. (a) A two-way set associative cache memory uses blocks of 4 words. The cache can accommodate a total of 2048 words from main memory. The main memory size is 124K _ 32
i. Formulate the information required to construct cache Memory.
ii. What is the size of cache Memory?
(b) The access time of cache memory is 100ns and that of main memory is 1000ns. It is estimated that 80% of memory requests are for read and the remaining 20% for write. The hit ratio for read access only is 0.9. A write through procedure is used.
i. What is the average access time of the system considering only read cycles?
ii. What is the average access time of the system considering both read and write cycles?
17. (a) Explain RISC pipeline in detail.

(b) Explain vector processing.
18. Draw circuit for BCD addition and subtraction. Explain its functionality with mathematical background.
19. (a) Explain the working of 8 _ 8 Omega Switching network.
(b) Explain the functioning of Binary Tree network with 2 _ 2 Switches. Show a neat sketch.
20. (a) What are the different types of I/O communication techniques? Give brief notes.
(b) In the above techniques, which is the most efficient? Justify your answer.
21. (a) Explain the variety of techniques available for sequencing of microinstructions based on the format of the address information in the microinstruction.
(b) Hardwired control unit is faster than microprogammed control unit. Justify this statement.
22 (a) If cache access time is 70 ns, memory access time is 700ns.
i. Compute the formula for average access time.
ii. Compute hit and miss ratios if the average access time is 140ns.
(b) What is RAID? What are the advantages of using this technique.
(c) Show the memory hierarchy and give the brief explanation.
23. What do you mean by instruction set architecture (ISA). What is the completeness and orthogonality with respect to ISA. What are the design issues related to ISA?
24. (a) Find the actual number from its IEEE 754 representation.
Sign = 0
Exponent = 1000 0000
Mantissa = 1100 0000 0000 0000 0000 000
(b) What is meant by normalization in oating point representation? Why do we need it? What is bias? What normalization is used in IEEE 754 standard?
25. (a) Differentiate I/O Bus and memory Bus.
(b) What are major functional differences between memory mapped I/O and Isolated I/O.
26. What is paging? Explain how the technique of paging can be implemented.
27. (a) Explain nanoinstructions and nanometry. Why do we need them?
(b) Describe advantages and disadvantages of horizontal and vertical microcoded systems.
28. (a) Distinguish between high level and low level languages?. What are the requirements for a good programming language?
(b) Explain with an example IEEE 754 representation for single precision and double precision floating point numbers.
29. Write short notes on the following:
(a) RISC pipeline
(b) Vector processing
(c) Array processors.
30. (a) Represent two n-bit unsigned numbers multiplications with a series of n/2-bit multiplications.
(b) Explain single precision and double precision calculations. In general how many bytes are used for both and what is the precision we get. Give some examples where double precision calculations are needed.

31. (a) What are the different physical forms available to establish an inter-connection network? Give the summary of those.
(b) Explain time-shared common bus Organization.
(c) Explain system bus structure for multiprocessors.
32. There exists three 4-bit registers (A,B, and C) and 4-bit data lines (say D). Design a circuit to transfer data from any register to any other register including data lines and vice versa.
33.a) Give a good account of Fixed Point and Floating point representation of data in a computer with examples.
b) Explain pipelining and super scalar operation.
34. Explain the organization of 4-bit binary adder and 4 bit binary incrementor. Illustrate the functioning of these with examples.
35. With reference to microprogram control explain Address sequencing in detail with examples.
36. With the aid of a flow chart explain a Hardware algorithm for 'Divide' operation.
37. List and explain various issues related to performance consideration of memory.
38. Give a detailed account of priority interupt.
39. Write notes on
a) Vector Processing
b) Array Processors.
40. Explain Interprocessor communication and synchrounization in Multiprocessors
41.a) Give a good Account of performance measurement of a Digital Computer
b) Explain the role of system software in the context of a Digital Computer.
42. Explain the organization of 4 bit arithmetic circuit with a schematic. Illustrate its functioning with an example.
43. Explain the organization and functioning of Micro program sequencer.
44. With the aid of flow chart explain Hardware implementation of Decimal Division.
45. Explain in detail secondary storage Medial concepts.
46. Explain in detail any two modes of Data Transfer.
47. Give a good account of Instruction Hazards in pipelining.
48. Explain Inter process Arbitration in Multiprocessors.
49.a) Give a detailed account of how a digital computer is organized and explain its functions clearly.
b) Give a detailed explanation of Data Representation in a digital computer with suitable examples.
50. Explain the organization of 4 bit combinational circuit shifter. Illustrate its functioning with an example.
51. Explain the following with examples:
a) Micro Operations
b) Micro Instructions
c) Micro Program
d) Micro Code.
52. With the aid of a flow chart explain either addition or subtraction operation of floating point numbers at hard ware level.
53. Explain in detail the organization of a functioning of secondary storage media.
54. Write notes on the following communication protocols:
a) RS 232

b) USB
c) IEEE 1394.
55. Give a good account of Data Hazards in pipelining.
56. Explain cache coherence in multiprocessors.

Set-4

57.a) Explain the functioning of a Digital Computer.
b) Give a note on error detection codes.
58.a) Explain Register Transfer Process and control function with suitable examples.
b) Explain the concept of Three state Bus Buffers.
59. Develop and explain a block schematic for decoding of micro operation fields.
60. With the aid of a flow chart explain Booth's algorithm for multiplication of signed-2's complement numbers.
61. Give a good account of Asynchronous and Synchronous DRAMs.
62. Give a detailed Account of Direct Memory Access (DMA).
63. Write notes on:
a) Concepts of Pipelining.
b) Vector Processing.
64. Explain Inter Connection Structures in Multiprocessors.

65. (a) Differentiate I/O Bus and memory Bus.
(b) What are major functional differences between memory mapped I/O and Isolated I/O.
2. (a) Explain RISC pipeline in detail.
(b) Explain vector processing.
3. Design a circuit which can be used to transfer data from any register to any other register out of four 4-bit registers A,B,C,D which uses RS ip-ops.
4. (a) Explain the functioning of a control unit explaining the terms control word, control memory, control address register and control buffer register.
(b) Support the statement. Instruction Set Architecture has impact on the processors microarchitecture.
5. Explain the following:
(a) Magnetic Tape Systems
(b) Optical Disc
(c) DVD Technology.
6. (a) Explain the terms NaN, overflow and underflow in the IEEE 754 representation of loating point numbers.
(b) Explain 2's complement method of representing numbers. When can you say that an overflow has occurred when adding or subtracting two _fixed point numbers.
(c) What do you mean by a parity bit? Explain with an example how even and odd parity bits are generated. Is it possible to correct errors using parity bits.

7. (a) Draw a circuit for 10's complement of a given numbers.
(b) Explain the relative advantages and disadvantages of 9's and 10's complement methods.
8. (a) What are the total number of switches in a 8_8 omega network? Show a neat

sketch.
(b) How many nodes will be there in a 4 Dimensional hyper cube? Show a neat sketch.

set-2

1. (a) What are the relative advantages of ones complement and two's complement methods.
(b) Explain subtraction procedure using complement representation.
2. (a) What is Flynn's classification? Categorize the different streams in it.
(b) Explain SIMD array processor organization.
3. (a) What are the di_erent interconnection structures used in multiprocessors? Explain about multistage crossbar switch.
(b) Support or oppose the statement? Every efficient serial program is efficient parallel program?
4. (a) What are the different types of I/O communication techniques? Give brief notes.
(b) In the above techniques, which is the most efficient? Justify your answer.
5. (a) Support or oppose the statement. If we want to add a new machine language instruction to a processors instruction set, simply write a C program and compiler and store the resultant code in control memory.
(b) Why do we need subroutine register in a control unit? Explain.
6. (a) Explain the functioning of omega switching network with a neat sketch.
(b) In 8 _ 8 omega switching network how many stages are there and in each stage how many Switches are there.
(c) How many stages and how many Switches in each stage are needed in a n _ n omega switching network.
7. (a) Differentiate between paging and segmentation.
(b) What are the relative advantages and disadvantages of using the technique of paged segmentation.
(c) In the Paged Segmentation technique, if the logical address format is as shown in the diagram below

Segment No. Page No. O_set
2 bits 4 bits 10 bits

i. What is the maximum size of virtual memory required?
ii. What is the maximum size of one segment?
iii. What is the size of each page?

8. Design a circuit to increment, decrement, complement and clear a 4 bit register using RS ip-ops. Explain the control logic.

Set-3

1. (a) Draw a owchart to explain how addition and subtraction of two _xed point numbers can be done. Also, draw a circuit using full adders for the same.
(b) Explain Booth's logorithm with its theoretical basis.
2. Describe commonly employed bit shift operators such as shift left, right and arith-metic shift left/right. Design a circuit for register length of 4 bits using D Flip-Flops.
3. (a) What is a virtual memory technique? Explain di_erent virtual memory tech-niques.

(b) Explain how the technique of paging can be implemented.
4. (a) What is meant by arithmetic pipeline? Explain.
(b) Explain pipeline for oating point addition and subtraction.
5. (a) Explain how Flynn classified the processors into different streams by giving an example for each stream.
(b) Explain tightly coupled and loosely coupled systems with suitable examples. [7+8]
6. (a) Give the typical horizontal and vertical microinstruction formats.
(b) Describe how microinstructions are arranged in control memory and how they are interpreted.
7. What are relative advantages and disadvantages of I/O communication techniques? Explain.
8. Distinguish between error detection and correction codes. What do you understand by odd parity and even parity? What is odd function and even function? To calculate odd and even parity values which functions can be used? Calculate Odd and even parity values for all hexadecimal digits 0-9 and A-F.

set-4

1. (a) What is Flynn's classiffication? Categorize.
(b) Explain each stream of the Flynn's classiffication with an example.
2. What are various components involved in serial communication? Explain.
3. Perform the following arithmetic operations using 2's complement method and 1's complement method.
(a) 1101□011
(b) 1111□1001
(c) 1111□1111
(d) 0111□1101. [4+4+3+4]
4. (a) Explain how the Bit Cells are organized in a Memory Chip.
(b) Explain the organization of a 1K _ 1 Memory with a neat sketch.
5. (a) Draw the block diagram of a computer system and describe each of its parts along with their functions. Also designate the information ow between the parts with arrows.
(b) Explain the term `memory bus bottleneck'.
(c) Distinguish between multiprocessor and a multicomputer.
6. (a) Explain the variety of techniques available for sequencing of microinstructions based on the format of the address information in the microinstruction.
(b) Hardwired control unit is faster than microprogammed control unit. Justify this statement.
7. (a) What are the different physical forms available to establish an inter-connection network? Give the summary of those.
(b) Explain time-shared common bus Organization.
(c) Explain system bus structure for multiprocessors.
8. Explain about instruction, fetch, and decode cycles for a memory reference instruction. Draw a ow chart also to explain the same. Indicate clearly where and which processor registers comes into picture. Now let us assume while a instruction is in the middle of its decode cycle a interrupt is arrived. What is going to happen? Is the instruction is completed or not. If we want to stop there itself and handle the

interrupt what are the difficulties?

(2007)

1. (a) What are the different performance measures used to represent a computer systems performance?
(b) Explain about Amdhal's Law.
2. Design a circuit which can be used to transfer data from any register to any other register out of four 4-bit registers A,B,C,D which uses RS flip-flops.
3. (a) How do we reduce number of microinstructions. What are micro-subroutines?

(b) Explain nanoinstructions and nanometry. Why do we need them?
4. (a) Multiply 10111 with 10011 using, Booths algorithm.
(b) Explain booths algorithm with its theoretical basis.
5. What are the different types of Mapping Techniques used in the usage of Cache Memory? Explain.
6. Write short notes on the following:
(a) RS 232
(b) USB
(c) IEEE 1394.
7. Explain the following with related to the Instruction Pipeline
(a) Pipeline conflicts
(b) Data dependency
(c) Hardware interlocks
(d) Operand forwarding
(e) Delayed load
(f) Pre-fetch target instruction
(g) Branch target buffer
(h) Delayed branch.
8. (a) What is the need of interprocessor synchronization? Explain.
(b) Explain hardwarelock mechanism.

Set-2

1. (a) Explain the terms compiler, linker, assembler, loader and describe how a C program or any other high level language program is executed in a system. Indicate entire process with a figure.
(b) Distinguish between high level and low level languages?. What are the requirements for a good programming language?
2. Design a circuit for parallel load operation into one of the four 4-bit registers from a bus. Mention clearly control/selection bits and selection logic. Assume JK flip-flops.
3. (a) Support or oppose the statement. If we want to add a new machine language instruction to a processors instruction set, simply write a C program and compile and store the resultant code in control memory.
(b) Why do we need subroutine register in a control unit? Explain.
4. (a) Multiply 10111 with 10011 using, Booths algorithm.
(b) Explain booths algorithm with its theoretical basis.

5. (a) ”In paged segmentation, the reference time increases and fragmentation decreases”, Justify your answer.
(b) A Virtual Memory System has an address space of 8K words and a Memory space of 4K words and page and block sizes of 1K words. Determine the number of page faults for the following page replacement algorithms: 1) FIFO 2) LRU if the reference string is as follows: 4,2,0,1,2,6,1,4,0,1,0,2,3,5,7.
6. What are relative advantages and disadvantages of I/O communication techniques? Explain.
7. Explain array processors. Explain SIMD array processor organization in detail.

8. (a) Explain the functioning of omega switching network with a neat sketch.
(b) In 8 x 8 omega switching network how many stages are there and in each stage how many Switches are there.
(c) How many stages and how many Switches in each stage are needed in a n x n omega sitching network.

set-3

1. (a) What are the different interconnection structures used in multiprocessors? Explain about multistage crossbar switch.
(b) Support or oppose the statement ?Every efficient serial program is efficient parallel program?
2. Design a circuit for parallel load operation into one of the four 4-bit registers from a bus. Mention clearly control/selection bits and selection logic. Assume RS flip-flops.
3. (a) What are the design goals for a designer while deciding a hardwired or micro-programmed CU for a CPU?
(b) Explain nanoinstructions and nanometry. Why do we need them.
4. (a) Explain Booth’s algorithm with its theoretical basis.
(b) Represent two n-bit unsigned numbers multiplications with a series of n/2-bit multiplications.
5. Explain two-way set associative mapping and four-way set associative mapping techniques with an example for each.
6. Explain the following with respect to serial communication:
(a) Data Communication processor
(b) Modem
(c) Block Transfer
(d) CRC
(e) Full Duplex
(f) Bit oriented protocol.
7. Write short notes on the following:
(a) RISC pipeline
(b) Vector processing
(c) Array processors.
8. (a) Explain multiport memory organization with a neat sketch.
(b) Explain system bus structure for multiprocessors with a neat sketch.

Set-4

1. (a) What are the different interconnection structures used in multiprocessors?

Explain about multistage crossbar switch.
(b) Support or oppose the statement ?Every efficient serial program is efficient parallel program?
2. There exists three 4-bit registers (A,B, and C) and 4-bit data lines (say D). Design a circuit to transfer data from any register to any other register including data lines and vice versa.
3. (a) Differentiate between microprogramming and nanoprogramming.
(b) Hardwired control unit is faster than microprogammed control unit. Justify this statement.
4. Explain the computational errors. Why do they occur?. Give some problems where these errors are catastrophic. Also, give some practical examples (algorithms) where error gets
(a) accumulated and
(b) multiplies.
5. (a) Compare and contrast FIFO and LRU Cache Replacement Algorithms.
(b) Compare and contrast direct and associative mapping Techniques.
6. (a) What is Direct Memory Access? Explain the working of DMA.
(b) What are the different kinds of DMA transfers? Explain.
(c) What are the advantages of using DMA transfers?
7. (a) What is meant by instruction pipeline? Explain four segment Instruction Pipeline.
(b) Give the timing diagram of instruction pipeline.
8. (a) Explain serial abitration (Daisy Chain).
(b) Explain prallel abitration.

17. Question Bank

Unit-I

PART-A

1. Give the name of the Von Neumann Computer.
2. Define Time sharing.
3. Define parallel processing.
4. Define Pipeline processing.
5. What is an operating system?
6. Define system throughput.
7. Mention some applications of parallel processing.
8. List the steps involved in the instruction execution.
9. Define Microcomputer.
10. Classify Parallel Computers.

PART-B

1. Explain the various Instruction types?
2. Write in detail about various addressing modes.
3. Explain the architecture of a basic Computer.
4. Explain Program Control.

5.Explain different types of Interrupts.

Unit-II

PART-A

1. Define intra segment and inter segment communication.
2. Mention the group of lines in the system bus.
3. What is bus master and slave master?
4. Differentiate synchronous and asynchronous bus.
5. What is strobe signal?
6. What is bus arbitration?
7. Mention types of bus arbitration.
8. What is IO control method?
9. What is DMA?
10. Why does the DMA priority over CPU when both request memory transfer?
11. List out the types of interrupts.
12. What is dumb terminal?
13. What is the need for DMA transfer?
14. List down the functions performed by an Input/Output.

PART-B (16 MARKS)

1. Explain with the block diagram the DMA transfer in a computer system.
2. Describe in detail about IOP organization.
3. Describe the data transfer method using DMA.
4. Write short notes on the following

(a) Magnetic disk drive (8)

(b) Optical drives (8)

5. Discuss the design of a typical input or output interface.
6. What are interrupts? How are they handled?

Unit –III

PART-A

1. What is Memory system?
2. Give classification of memory.
3. Define cache.
4. What is Read Access Time?
5. Define Random Access Memory.
6. What are PROMS?
7. Define Memory refreshing.
8. What is SRAM and DRAM?
9. What is Volatile memory?
10. Define data transfer or bandwidth.
11. What is Flash memory?
12. What are multilevel memories?
13. Give the basic structure of cache and what is its use?

14. What is associate memory?
15. Define Seek time and latency time.
16. What is DVD?
17. Define Magneto Optical Disk.
18. Define Virtual Memory.
19. Distinguish between Static and Dynamic.
20. Define the term LRU and LFU.

PART-B (16 MARKS)
1. Illustrate the characteristics of some common memory technologies.
2. Describe in detail about associative memory.
3. What is Memory Interleaving? Explain the addressing of multiple module memory system.
4. Discuss the different mapping techniques used in cache memories and their relative merits and demerits.
5. Comparing paging and segmentation mechanisms for implementing the virtual memory.
6. What do you mean by virtual memory? Discuss how paging helps in implementing virtual memory.
7. Discuss any six ways of improving the cache performance.

UNIT IV

PART -A

1. What is addressing?
2. What are modes in which 8086 can operate?
3. List the segment registers of 8086?
4. What is NMI?
5. What are the Flags of 8086?

PART- B

1. Draw the architecture block diagram of 8086.
2. Explain with examples addressing modes of 8086 processor.
3. Draw & explain the modes of operation of 8086.
4. Explain the instruction set of 8086 with examples.
   Explain in detail about the functions of various data transfer and arithmetic operations of 8086

Unit -V

Which of the following are valid and invalid assembler language instructions for 8086? Explain each instruction. State the error for each invalid instruction. Assume that all identifiers are variables and are associated with words. [8+8]
MOV BP, AL MOVI IX, 10 MOV CS, AX XLAT 3.

18. Assignment Questions

Unit - 1

1. Explain various instruction formats for the expression X= (a+b)*(c+d)?
2. Briefly explain memory instructions?
3. Numerically explain about different Addressing modes?
4. Explain about Program interrupts in detail.

Unit - 2

1. Explain about I/O interface with example?
2. Discuss about I/O Vs Memory Bus, Isolated Vs Memory –mapped?
3. Explain about Asynchronous Data Transfer in detail.
4. Explain about data transfer between I/O devices And CPU through Programmed I/O mode of Transfer.
5. Explain DMA transfer with neat diagram?.
6. Explain the communication between CPU-IOP through Flow chart.

Unit-3

1. Explain Memory Hierarchy in detail.
2. How Address in Mapped in memory?
3. Explain memory connections to CPU through Circuit diagram.
4. Give Hardware Configuration of Associative Memory?
5. Explain the mapping techniques of cache memory?
6. In detail explain about virtual memory?

Unit-4

1. Explain about Pins configuration of signal description of 8086.
1. . Explain in detail about addressing modes of 8086.
2. Explain about 8086 flag register.
3. Explain about register organization of 8086.
4. Write about concept of pipelining.

UNIT-5

1. Write an Assembly language program for adding and subtracting two hexadecimal numbers.
   . Write an Assembly language program for ascending order of given series
2. 58H, 24H, 19H, 02H.
3. Write assembly language program to count number of even and odd numbers from the given series. 1234H,0299H,3682H,0087H,1235H,0289H.
4. . Write assembly language program for arithmetic expression
   X=(A+B) * (C+D)
5. Write an assembly language program for multiplication of two hexadecimal numbers.

6. Write assembly language program to count number of positive and negative numbers from the given series. 1234H,0299H,3682H,0087H,1235H,0289H.
7. Write an assembly language program for finding square and cube of the number

7. Unit wise Quiz questions

Unit-I

Q.1 In Reverse Polish notation, expression A*B+C*D is written as
(A) AB*CD*+ (B) A*BCD*+ (C) AB*CD+* (D) A*B*CD+
Ans: A

Q.2Assembly language
(A) uses alphabetic codes in place of binary numbers used in machine language (B) is the easiest language to write programs (C) need not be translated into machine language
Ans :a

Q.3 Computers use addressing mode techniques for
(A) giving programming versatility to the user by providing facilities as pointers to memory counters for loop control (B) to reduce no. of bits in the field of instruction (C) specifying rules for modifying or interpreting address field of the instruction (D) All the above
Ans: D

Q.4 _____ register keeps track of the instructions stored in program stored in memory.
(A) AR (Address Register) (B) XR (Index Register) (C) PC (Program Counter) (D) AC (Accumulator)
Ans: C

Q.5 The addressing mode used in an instruction of the form ADD X Y, is
(A) Absolute (B) indirect (C) index (D) none of these
Ans: C

Q.6 A Stack-organised Computer uses instruction of
(A) Indirect addressing (B) Two-addressing (C) Zero addressing (D) Index addressing

Ans: C

Q.7 PSW is saved in stack when there is a
(A)interrupt recognised (B)execution of RST instruction (C)Execution of CALL instruction (D)All of these
Ans: A

Q.8_____ register keeps tracks of the instructions
stored in program stored in memory.
(A) AR (Address Register)
(B) XR (Index Register) (C) PC (Program Counter) (D) AC (Accumulator)
Ans: C

Q.9The instruction ‘ORG O’ is a
(A) Machine Instruction. (B) Pseudo instruction. (C) High level instruction. (D) Memory instruction.
Ans: B

Q.10The BSA instruction is

(A)Branch and store accumulator (B)Branch and save return address (C)Branch and shift address (D)Branch and show accumulator
Ans: B
Q.11The load instruction is mostly used to designate a
transfer from memory to a
processor register known as
(A) Accumulator (B) Instruction Register (C) Program counter (D) Memory address Register
Ans: A
Q12.Data input command is just the opposite of a
(A) Test command (B) Control command (C) Data output (D) Data channel
Ans: C
Q.13MRI indicates
(A)Memory Reference Information.
(B)Memory Reference Instruction.
(C)Memory Registers Instruction.
(D)Memory Register information
Ans: B
Q.14 If the value V(x) of the target operand is contained in the address field itself, the addressing mode is
(A)immediate.
(B)direct.
(C)indirect.
(D) implied.
Ans:B
Q15.The instructions which copy information from one location to another either in the processor's internal register set or in the external main memory are called
(A)Data transfer instructions.
(B)Program control instructions.
(C)Input-output instructions.
(D)Logical instructions.
Ans:A

UNIT-II

Q.1A group of bits that tell the computer to perform aspecific operation is known as
(A) Instruction code (B) Micro-operation (C) Accumulator (D) Register
Ans: A
Q.2he communication between the components in a microcomputer takes place via the address and
(A) I/O bus (B) Data bus (C) Address bus (D) Control lines
Ans: B
Q.3 A microprogram sequencer
(A)generates the address of next micro instruction to be executed.
(B)generates the control signals to execute a microinstruction.
(C)sequentially averages all microinstructions in the control memory.
(D)enables the efficient handling of a micro program subroutine.

Ans: A
Q4. Microinstructions are stored in control memory groups, with each group specifying a
(A) Routine
(B) Subroutine
(C) Vector
(D) Address
Ans: A
Q.5Status bit is also called
(A) Binary bit (B) Flag bit
(C) Signed bit (D) Unsigned bit
Ans: B
Q6.The maximum addressing capacity of a micro processor which uses 16 bit database & 32 bit address base is
(A)64 K.
(B)4 GB.
(C)both (A)& (B)
.(D)None of these.
Ans:B

7. A microprogram is sequencer perform the operation...
A) read
B) write
C) read and write
D) read and execute
Ans :D

Q.1 Suppose that a bus has 16 data lines and requires 4 cycles of 250 nsecs each to transfer data. The bandwidth of this bus would be 2 Megabytes/sec. If the cycle time of the bus was reduced to 125 nsecs and the number of cycles required for transfer stayed the same what would the bandwidth of the bus?
(A) 1 Megabyte/sec (B) 4 Megabytes/sec (C) 8 Megabytes/sec (D) 2 Megabytes/sec
Ans: D
Q.2 In a memory-mapped I/O system, which of the following will not be there?
(A) LDA (B) IN (C) ADD (D) OUT
Q3.An interface that provides a method for transferring binary information between internal storage and external devices is called
(A) I/O interface
(B) Input interface
(C) Output interface
(D) I/O bus
Ans: A
Q4.An interface that provides I/O transfer of data directly to and form the memory unit and peripheral is termed as
(A)DDA.
(B)Serial interface.
(C)BR.

(D)DMA.
Ans:D
Q5. State whether the following statement is True or False for PCI bus.
i) The PCI bus tuns at 33 MHZ and can transfer 32-bits of data(four bytes) every clock tick.
ii) The PCI interface chip may support the video adapter, the EIDE disk controller chip and may be two external adapter cards.
iii) PCI bus deliver the different throughout only on a 32-bit interface that other parts of the machine deliver through a 64-bit path.
A) i- True, ii- False, iii-TrueB) i- False, ii- True, iii-True
C) i-True, ii-True, iii-False
D) i- False, ii- False, iii-True
Ans:c
6 The I/O processor has a direct access to ..... and contains a number of independent data channels.
A) main memory
B) secondary memory
C) cache
D) flash memory
Ans:A

Q.1 SIMD represents an organization that _____.
(A) refers to a computer system capable of processing several programs at the same time. (B) represents organization of single computer containing a control unit, processor unit and a memory unit. (C) includes many processing units under the supervision of a common control unit (D) none of the above.
Ans: C
Q.2 In a vectored interrupt.
(A) the branch address is assigned to a fixed location in memory. (B) the interrupting source supplies the branch information to the processor through an interrupt vector. (C) the branch address is obtained from a register in the processor (D) none of the above
Ans: B
3.MIMD stands for
(A)Multiple instruction multiple data (B)Multiple instruction memory data (C)Memory instruction multiple data (D)Multiple information memory data
Ans: A
4.An instruction pipeline can be implemented by meansof
(A) LIFO buffer (B) FIFO buffer (C) Stack (D) None of the above
Ans: B
Q.1 An n-bit microprocessor has
(A)n-bit program counter (B)n-bit address register
(C)n-bit ALU (D)n-bit instruction register
Ans: D

Unit-III:

Q.1The amount of time required to read a block of data from a disk into memory is composed of seek time, rotational latency, and transfer time. Rotational latency refers to

(A) the time its takes for the platter to make a full rotation (B) the time it takes for the read-write head to move into position over the appropriate track (C) the time it takes for the platter to rotate the correct sector under the head (D) none of the above

Ans: A

Q.2 What characteristic of RAM memory makes it not suitable for permanent storage?

(A) too slow (B) unreliable (C) it is volatile (D) too bulky

Ans: C

Q.3 The circuit used to store one bit of data is known as

(A) Register (B) Encoder (C) Decoder (D) Flip Flop

Ans :d

Q.4The average time required to reach a storage location in memory and obtain its contents is called the

(A) seek time (B) turnaround time (C) access time (D) transfer time

Ans: C

Q.5 The idea of cache memory is based

(A) on the property of locality of reference (B) on the heuristic 90-10 rule (C) on the fact that references generally tend to cluster (D) all of the above

Ans: A

Q.6If memory access takes 20 ns with cache and 110 ns with out it, then the ratio (cache uses a 10 ns memory) is

(A) 93% (B) 90% (C) 88% (D) 87%

Ans: B

Q.7 Cache memory acts between

(A) CPU and RAM (B) RAM and ROM (C) CPU and Hard Disk (D) None of these

Ans: A

Q.8 Write Through technique is used in which memory for updating the data

(A) Virtual memory (B) Main memory

(C) Auxiliary memory (D) Cache memory

Ans: D

Q.9Generally Dynamic RAM is used as main memory in a computer system as it

(A) Consumes less power (B) has higher speed (C) has lower cell density (D) needs refreshing circuitary

Ans: B

Q.10Virtual memory consists of

(A) Static RAM (B) Dynamic RAM (C) Magnetic memory (D) None of these

Ans: A

Q.11Virtual memory consists of

(A) Static RAM (B) Dynamic RAM (C) Magnetic memory (D) None of these

Ans: A

Q.12 If the main memory is of 8K bytes and the cache memory is of 2K words. It uses associative mapping. Then each word of cache memory shall be

(A) 11 bits (B) 21 bits

(C) 16 bits (D) 20 bits
Ans: C
Q.13Cache memory works on the principle of
(A)Locality of data (B)Locality of memory (C)Locality of reference D)Locality of reference & memory
Ans: C
Q.14The main memory in a Personal Computer (PC) is madeof
(A)cache memory. (B)static RAM (C)Dynamic Ram (D)both (A)and (B)
. Ans: D
Q.15Memory unit accessed by content is called
(A) Read only memory (B) Programmable Memory (C) Virtual Memory D) Associative Memory
Ans: D

UNIT 4

1.Which of the following register is not present in EU(Execution Unit) of the 8086 microprocessor? [d]
a. SP b.BP c.SI d.IP

2. Which signal is used to demultiplex the Adress bus of 8086 [d]
a. BHE b.LOCK c.READY d.ALE

3. Which of the following signal is used to control the direction of dataflow of bus transcievers [a]

a. $\overline{DT}$ / R b. M / $\overline{IO}$ c. $\overline{DEN}$ d.ALE

4. Which of the following signal belongs to maximum mode only [c]

a. $\overline{RD}$ b. $\overline{TEST}$ c. LOCK d. $\overline{DEN}$

5. Which of the following pair of registers is invalid [a]
a. CS:SP b. SS: SP c. DS:BX d. ES:DI

6. Which of the following flags is used for single stepping [c]
a. Direction Flag b. Interrupt Flag c. Trap Flag d. Overflow Flag

7. Which of the following are the three basic sections of a microprocessor unit? [b]
(A) operand, register, and arithmetic/logic unit (ALU)
(B) control and timing, register, and arithmetic/logic unit (ALU)
(C) control and timing, register, and memory
(D) arithmetic/logic unit (ALU), memory, and input/output

8. Which bus is a bidirectional bus? [b]

(A) address bus (B) data bus (C) address bus and data bus (D) none of the above

9. Direction flag is used with [a]
(A) String instructions (B) Stack instructions
(C) Arithmetic instructions (D) Branch instructions

10. What does microprocessor speed depends on? [c]
(A) Clock (B) Data bus width (C) Address bus width (D)size of register

11. A register capable of shifting its binary information either to the right or the left is called a[c] (A)parallel register (B) serial register (C) shift register (D) storage register

12. What is meant by Mask able interrupts? [b]
(A) An interrupt which can never be turned off
(B) An interrupt that can be turned off by the programmer
(C) An interrupt which can be turned off automatically
(D) none

13. In a DMA write operation the data is transferred [a]
(A) from I/O to memory (B) from memory to I/O
(C) from memory to memory (D) from I/O to I/O

Fill in the blanks

1. Flags are divided into **control flags** & **conditional flags**

2. Data bus is used for **sending and receiving data between CPU and other devices**

3. The function of instruction CLC is to **reset the carry flag to zero**

3. **ALE(address latch enable)** signal is provided by 8086 to demultiplex the AD_0-AD_{15} into A_0-A_{15} & D_0-D_{15} using external latches.

4. The part of operating system which is loaded in RAM during power up from harddisk or floppy disk is known as **DOS(disk operating system)**
5. The cycle required to read from memory is called **Machine Cycle.**
6. The address available at 8086 pins is called **Physical Addres**
7. The Condition which makes CPU execute next instruction after WAIT instruction is **TEST Becomes Low**

UNIT-5

1. Which of the following register holds the address of I/O port of I/O instructions of 8086 [b]

a. BX b.DX c.SI d.DI

2. What do the symbols [ ] indicate? [c]

(A) Direct addressing (B) Register Addressing

(C) Indirect addressing (D) None of the above

Fill in the blanks

3. The instruction MOV AX,BX gives that **the contents of register BX are copied to AX and stored in it**

4. The instruction MOV 1234[BX],3456H is **Six (6)** bit (opcode) Instruction.

1. The advantage of memory mapped I/O over I/O mapped I/O is, [d]

(A) Faster (B) Many instructions supporting memory mapped I/O

(C) Require a bigger address decoder (D) All the above

2. Which pins are general purpose I/O pins during mode-2 operation of the 8255? [a]

(A) PA0 – PA7 (B) PB0-PB7 (C) PC3-PC7 (D) PC0-PC2

3. An example for a D/A converter is **IC 1408,DAC0830/DAC0831/DAC0832**

4. **Port A** port of 8255 can be programmed in three modes: mode 0, mode 1, mode 2.

5.The modes used in display mode are **left entry mode (type wroter mode)** & **right entry mode (calculator type mode)**

8. Tutorial Problems

Unit-1

1. Instruction types
2. Addressing modes
3. Types of interrupts
4. Program Control.

Unit-2

1. DMA Transfer & Control
2. I/O Interface
3. Asynchronous Data Transfer
4. Asynchronous Communication Interface
5. IOP-CPU Communication
6. INTEL 8089

Unit-3

1. Memory Hierarchy
2. Associative Memory
3. Cache Memory Mapping Techniques
4. Virtual Memory
5. Page Replacement Techniques

Unit 4

1. Signal Description of 8086
2. Register Organization of 8086
3. Concept of Pipelining
4. Adressing Modes of 8086

Unit -5

1. Assembly Directives
2. Assembly Programs for 8- bit & 16-bit Addition , Subtraction , division & Multiplication
3. Assembly Programs to find even & odd number from list

4. Assembly Programs to find positive & negative number from list
5. Assembly Programs to find BCD Addition , Subtraction , division & Multiplication
6. Assembly Programs to find parity bits of numbers
7. Assembly language program for arithmetic expressions.

21. Known Gaps

In this year the course was advanced with the concepts of micro processor. Any language needs practice to learn it. 8086 assembly language also need more programming examples and exercises. This is evident from the course end survey analysis of previous academic year i.e. 2014-15. So, EMU8086 simulator is provided to the students to check and understand the executing of the programming exercises.

22.Discussion topics

1. Functions of CPU
2. I/O Bus and interface
3. Main Memory, RAM,ROM
4. Arithmetic operations
5. 8086 microprocessor addressing modes

23. References, Journals, websites and E-links

Websites

1. https://www.cs.dal.ca/~mheywood/CSCI3121.
2. *uic.edu.hk/~hpguo/teaching/spring2011/co/index.html*
3. *http://emu8086.com*

REFERENCES

Text Books

- Computer System Architecture, M. Morris Mano (UNIT-1,2,3).
- Advanced Micro Processor and Peripherals, Hall/A K Ray (UNIT-4,5).

References

- Computer Organization and Architecture, William Stallings, Sixth Edition, Pearson/PHI.
- Structured Computer Organization-Andrew S Tanenbaum, 4th Edition, PHI/Pearson.
- Fundamentals of Computer Organization and Design, Sivaraama Dandamudi, Springer Int. Edition.

- Computer Architecture a Quantitative Approach, John L. Hennessy and David A. Patterson, 4th Edition, Elsevier.
- Computer Architecture: Fundamentals and Principles of Computer Design, SJoseph D. Dumas II, BS Publication.

24. Quality Measurement sheets

Course Assessment

Class: II CSE-A A.Y: 2015-2016
Subject: Computer Organization Sem: II
Faculty:

Assessment	Criteria Used	Attainment Level			Remarks
Direct(d)	**Theory:**				
	External Marks				
	Internal Marks(Theory)				
	Assignments				
	Tutorials				
	Lab:				
	Internal Marks				
	External Marks				
Indirect(id)	Course End Survey				
Course Assessment(06*d+0.4*id)					

Faculty Program Coordinator HOD

Course End Survey Analysis

A.Y :2015-2016	CLASS : 2 CSE-A	SEM : II	COURSE: Computer Organization
FACULTY:	NO OF STUDENTS:	DATE:	ASSESSMENT:

Summary:
Strengths:
Weakness:

Suggestions/Improvements required:

(Signature of Faculty) **(HOD)**

Course Assessment

Class: II CSE-A
Subject: Computer Organization
Faculty: N.Radhika Amareshwari

A.Y: 2014-2015
Sem: II

Assessment	Criteria Used	Attainment Level			Remarks
Direct(d)	**Theory:**				
	External Marks				
	Internal Marks(Theory)	78.95			
	Assignments	100			
	Tutorials	76.87			
	Lab:				
	Internal Marks				
	External Marks				
Indirect(id)	Course End Survey			79.87	
Course Assessment(06*d+0.4*id)					

Faculty Program Coordinator HOD

Course End Survey Analysis

A.Y :2014-2015	CLASS : 2 CSE-A	SEM : II	COURSE: Computer Organization
FACULTY: N.Radhika Amareshwari	NO OF STUDENTS: 57	DATE:27/04/2015	ASSESSMENT:

Summary:
Computer architecture is a specification detailing how a set of software and hardware technology standards interact to form a computer system or platform. Computer architecture refers to how a computer system is designed and what technologies it is compatible with.
Whether the course delivery process was able to meet the suggestions/expectations given in the previous course end survey assessment: YES

Strengths:
Syllabus was less in computer organization for R-13 Regulation; Student developed the ability and confidence to use the fundamentals of computer organization as a tool in digital systems and leant about the fundamentals in Assembly language Programming by using 8086 CPU Architecture.

Weakness:
Practical's for ALP was required to make concepts in Assembly Programming Clear.

Suggestions/Improvements required:
Lab is needed for 8086 ALP Programming

(Signature of Faculty) **(HOD)**

Course Assessment

Class: II CSE-B A.Y: 2014-2015
Subject: Computer Organization Sem: II
Faculty: N.Radhika Amareshwari

Assessment	Criteria Used	Attainment Level			Remarks
Direct(d)	**Theory:**				
	External Marks				
	Internal Marks(Theory)	85.96			
	Assignments	100			
	Tutorials	84.01			
	Lab:				
	Internal Marks				
	External Marks				
Indirect(id)	Course End Survey			89.84	
Course Assessment(06*d+0.4*id)					

Faculty Program Coordinator HOD

Course End Survey Analysis

A.Y : 2014-15	CLASS : 2 CSE-B	SEM : I	COURSE: Computer Organization
FACULTY: N.Radhika Amareshwari	NO OF STUDENTS: 57	DATE: 27-04-2105	ASSESSMENT:1

Summary:

Computer architecture is a specification detailing how a set of software and hardware technology standards interact to form a computer system or platform. Computer architecture refers to how a computer system is designed and what technologies it is compatible with.

Whether the course delivery process was able to meet the suggestions/expectations given in the previous course end survey assessment: Yes

Strengths:

Syllabus was less in computer organization for R-13 Regulation; Student developed the ability and confidence to use the fundamentals of computer organization as a tool in digital systems and leant about the fundamentals in Assembly language Programming by using 8086 CPU Architecture.

Weakness:

Practical's for ALP was required to make concepts in Assembly Programming Clear.

Suggestions/Improvements required:

Lab is needed for 8086 ALP Programming

(Signature of Faculty) **(HOD)**

Course Assessment

Class: II B.TECH CSE-C A.Y: 2014-2015
Subject: Computer Organization Sem: II
Faculty: P. Gowtamee Radha

Assessment	Criteria Used	Attainment Level			Remarks
Direct(d)	**Theory:**				
	External Marks				
	Internal Marks(Theory)	57.89%			
	Assignments	89.47%			
	Tutorials	85.38%			
	Lab:				
	Internal Marks				
	External Marks				
Indirect(id)	Course End Survey	91.2%			
Course Assessment(0.6*d+0.4*id)					

Faculty Program Coordinator HOD

Course End Survey Analysis

A.Y : 2014-15	CLASS : II CSE-C	SEM : II	COURSE:CO
FACULTY: P. GOWTAMEE RADHA	NO OF STUDENTS:50	DATE:	ASSESSMENT:91.2%

Summary:
Computer architecture is a specification detailing how a set of software and hardware technology standards interact to form a computer system or platform. Computer architecture refers to how a computer system is designed and what technologies it is compatible with.
Whether the course delivery process was able to meet the suggestions/expectations given in the previous course end survey assessment: Yes

Strengths:

Syllabus was less in computer organization for R-13 Regulation; Student developed the ability and confidence to use the fundamentals of computer organization as a tool in digital systems and

leant about the fundamentals in Assembly language Programming by using 8086 CPU Architecture.

Weakness:

Practical's for ALP was required to make concepts in Assembly Programming Clear.

Suggestions/Improvements required:

Lab is needed for 8086 ALP Programming

(Signature of Faculty) (HOD)

Course Assessment

Class: II B.TECH CSE-D A.Y: 2014-2015
Subject: COMPUTER ORGANIZATION Sem: II
Faculty: M. VAMSI KRISHNA

Assessment	Criteria Used	Attainment Level			Remarks
Direct(d)	**Theory:**				
	External Marks				
	Internal Marks(Theory)	42.59%			
	Assignments	96.30%			
	Tutorials	86.03			
	Lab:				
	Internal Marks				
	External Marks				
Indirect(id)	Course End Survey	85.5%			
Course Assessment(0.6*d+0.4*id)					

Faculty Program Coordinator HOD

Course End Survey Analysis

A.Y : 2014-15	CLASS : II CSE-D	SEM : II	COURSE:CO
FACULTY: M.VAMSI KRISHNA	NO OF STUDENTS:48	DATE:	ASSESSMENT:85.5%

Summary:
Computer architecture is a specification detailing how a set of software and hardware technology standards interact to form a computer system or platform. Computer architecture refers to how a computer system is designed and what technologies it is compatible with.
Whether the course delivery process was able to meet the suggestions/expectations given in the previous course end survey assessment: Yes

Strengths:

Syllabus was less in computer organization for R-13 Regulation; Student developed the ability and confidence to use the fundamentals of computer organization as a tool in digital systems and leant about the fundamentals in Assembly language Programming by using 8086 CPU Architecture.

Weakness:

Practical's for ALP was required to make concepts in Assembly Programming Clear.

Suggestions/Improvements required:

Lab is needed for 8086 ALP Programming

(Signature of Faculty) (HOD)

25.Students list

Class / Section: CSE 22A		
SlNo	**AdmnNo**	**StudentName**
1	13R11A05G3	SRIKANTH AKKANAPALLY
2	14R11A0501	AKHILESH G
3	14R11A0502	AKULA KRISHNA
4	14R11A0503	ALLAM USHA SREE
5	14R11A0504	ANANT SRIVATS
6	14R11A0505	AYYAGARI V S V VARDHAMAN
7	14R11A0506	B MOHANA SREYA
8	14R11A0507	B PRASHANTH
9	14R11A0508	BANDA GAYATHRI VEENA
10	14R11A0509	BATTAR SRIKANTH RAGHAVA
11	14R11A0510	BHONSLE SAI KRUTHI
12	14R11A0511	BONAGIRI TEJASHWINI
13	14R11A0512	BOTLA MOUNIKA
14	14R11A0513	D VENKATESH
15	14R11A0514	DASARI PRANAY KUMAR
16	14R11A0515	DEEPIKA TUDIMILA
17	14R11A0516	DOKKA DIVYA
18	14R11A0517	E MADHU BHARGAVA
19	14R11A0518	K JWALA
20	14R11A0519	K SAMHITHA REDDY
21	14R11A0520	K THAPASVI REDDY
22	14R11A0521	KALAKONDA VAISHNAVI
23	14R11A0522	KAMPALLY YAMINI
24	14R11A0523	KONTHAM RAVITEJA REDDY
25	14R11A0524	KOSARAJU LEELA KRISHNA
26	14R11A0525	KRISHNA SWETHA R
27	14R11A0526	KUNDUR ROHAN REDDY
28	14R11A0527	KUNNUMAL MAVILA AMRITHA
29	14R11A0528	KURMA BHARGAV
30	14R11A0529	KUTHURU PRIYANKA
31	14R11A0530	KYASNOORI SHIVANI
32	14R11A0531	M K SREE HARSHA
33	14R11A0532	M POOJA
34	14R11A0533	M SAI KRISHNA
35	14R11A0534	M SHANMUGHA PRIYA

36	14R11A0535	MALLADI RAMYA
37	14R11A0536	MEESALA SHAILAJA
38	14R11A0537	N SRI MANASVI
39	14R11A0538	NADENDLA VENKATA VINEET MURARI
40	14R11A0539	NAKKA AKHIL
41	14R11A0540	NALLANAGULA BHANU PRAKASH
42	14R11A0541	P SOWMYA
43	14R11A0542	PARVATHALA CHETANA
44	14R11A0543	PENDYALA HEMANTH SAI
45	14R11A0544	R CHANDRA MOHAN
46	14R11A0545	REKULA CHANDRAHASA
47	14R11A0546	RUDRARAJU PAVANI
48	14R11A0547	S SWATHI
49	14R11A0548	SAI SINDHU BEERAM
50	14R11A0549	SOPPADANDI AISHWARYA RAJ
51	14R11A0550	SYED VAQUAS ASHRAF
52	14R11A0551	TALLURI MURALI KRISHNA
53	14R11A0552	TAMMEWAR SNEHIT
54	14R11A0553	THIRIVEEDI DHEERAJ KUMAR
55	14R11A0554	THUMMA VINEETHA REDDY
56	14R11A0555	VALLURU VENUMADHURI
57	14R11A0556	VAMANA GUNTLA MANOJ
58	14R11A0557	VENKATA SAITEJA Y
59	14R11A0558	VUSAKA VIKAS REDDY
60	14R11A0559	Y JYOTSNA

Class / Section: CSE 22B		
SlNo	**AdmnNo**	**StudentName**
1	13R11A0561	BATTU SHIRISH KUMAR REDDY
2	14R11A0560	AKELLA SUBRAMANYA SOUJANYA
3	14R11A0561	ALEENA CHACKO K
4	14R11A0562	BARLA PRAVALIKA
5	14R11A0563	BATHINI DIVYA
6	14R11A0564	BOLLA NAVEEN REDDY
7	14R11A0565	CHAVALI SWATHI
8	14R11A0566	CHEPYALA AKHIL
9	14R11A0567	CHITTARVU LAKSHMI ISHWARYA
10	14R11A0568	DAMAGALLA KARTIK
11	14R11A0569	DHATRI NAIDU BHOGADI
12	14R11A0571	DODDAPANENI SANOJ
13	14R11A0572	DUGGISHETTY SAI PRASANNA

14	14R11A0573	E PAVAN CHANDRA
15	14R11A0574	G BHAVANA
16	14R11A0575	G KISHAN
17	14R11A0576	G LASYA PRIYA
18	14R11A0577	G SIRIVENNELA
19	14R11A0578	GANAPATHI RAJU MADHURI
20	14R11A0579	GANGAVARAPU MANASA
21	14R11A0580	GATTOJI HARITHA
22	14R11A0581	GAYATHRI MANDAL
23	14R11A0582	GORTI SANTOSH KUMAR
24	14R11A0583	HRITIK S
25	14R11A0584	KALYANAM KRUTHIKA REDDY
26	14R11A0585	KANUMURI SRI PRATHYUSHA
27	14R11A0586	KOLLU VINESH BABU
28	14R11A0587	KONDISETTI NIRANJANA KUMARI
29	14R11A0588	KONIKA VENKATA PADMA PRIYA HASINI
30	14R11A0589	KURUP MEGHANA
31	14R11A0590	MADARAJU RAMYA SAI
32	14R11A0591	MEDIPALLY SRIYA
33	14R11A0592	MUDDU DEEPTHI
34	14R11A0593	N LAHARI
35	14R11A0594	N VIKHYAT
36	14R11A0595	NAKEERTHA SANDHYA RANI
37	14R11A0596	NALLAMILLI JYOTHI
38	14R11A0597	NALLAPANENI ADITYA SAI
39	14R11A0598	NAMBURI MANISHA
40	14R11A0599	NIKHIL SINGH P
41	14R11A05A0	ORUGANTI KALPANA
42	14R11A05A1	PATLORI SAI KUMAR
43	14R11A05A2	PEESAPATI VENKATA SAI VAMSI
44	14R11A05A3	PINDIPOLU SRAVYA
45	14R11A05A4	PRANATHI KRISHNA SANGANI
46	14R11A05A5	PUDHOTA AVINASH
47	14R11A05A6	S ASHA LAKSHMI
48	14R11A05A7	S CHANDANA REDDY
49	14R11A05A8	SHAIK HAFEEZ HUSSAIN
50	14R11A05A9	SHEELAM SUSHMA
51	14R11A05B0	SRAVIKA SANKU
52	14R11A05B1	THUMMALA RISHIKA REDDY
53	14R11A05B2	VARSHA CHAHAL
54	14R11A05B3	VEERAVALLI SHILPA

55	14R11A05B4	VUPPUTTURI SUSHANTH KUMAR
56	14R11A05B5	Y ADITHYA
57	14R11A05B6	YEDDU SHASHI KUMAR
58	14R11A05B7	KATAMONI HARSHITHA
59	15R15A0501	B VINESH
60	15R15A0502	B SHRAVAN KUMAR

Class / Section: CSE 22C		
SlNo	**AdmnNo**	**StudentName**
1	14R11A05C0	A S SRUJAN SAI
2	14R11A05C1	AITHA RANJEETH KUMAR
3	14R11A05C2	ALIGETI MAHITHA
4	14R11A05C3	BAGGU VISHNU SAI PRASAD
5	14R11A05C4	BARGELA PRANAY RAJ
6	14R11A05C5	BIROJU SAI SUGANDH CHARY
7	14R11A05C6	G AKHILA
8	14R11A05C7	GANGJI MANISHA
9	14R11A05C8	GARIKAPATI NAMRATHA CHOWDARY
10	14R11A05C9	GARLAPATI RENUKA
11	14R11A05D0	GUNDARAPU SOUJANYA REDDY
12	14R11A05D1	GUNGI SAI KUMAR
13	14R11A05D2	GUNNALA RAMYA SREE
14	14R11A05D3	ILA BHAVANA
15	14R11A05D4	J HEMANTH SAI
16	14R11A05D5	JELLA MOUNICA
17	14R11A05D6	K TEJA ABHINAV
18	14R11A05D7	KANAKADANDI NAGA VENKATA APARNA
19	14R11A05D8	KASI SRAVYA
20	14R11A05D9	KAVYA S
21	14R11A05E0	KUKKALA DEEPIKA
22	14R11A05E1	LAHIRI KOTAMRAJU
23	14R11A05E2	MARAM RAJEEV KUMAR
24	14R11A05E3	MARGONWAR GAYATRI
25	14R11A05E4	MEKALA SWATHI
26	14R11A05E5	MOTHABOINA KAMESHWAR RAO ROHAN MUDH
27	14R11A05E6	NALABOTHULA HARIKA
28	14R11A05E7	NALADALA SAI CHARITHA
29	14R11A05E8	NEMURI SAI SAMPATH GOUD
30	14R11A05E9	NIKHIL THAPA
31	14R11A05F0	P ANUDEEP
32	14R11A05F1	P SAI PRASAD

33	14R11A05F2	PARIMI SAIESH
34	14R11A05F3	PINNINTI SAIKIRAN REDDY
35	14R11A05F4	POLAGOUNI VAISHNAVI GOUD
36	14R11A05F5	POTHURI VENKATA SAI PRIYANKA
37	14R11A05F6	RAHUL N D
38	14R11A05F7	RAI MEGHANA
39	14R11A05F8	RAYALA PRIYANKA
40	14R11A05F9	RAYAPROLU ASWINI
41	14R11A05G0	REEBA FATIMA
42	14R11A05G1	ROSHAN ROY
43	14R11A05G2	S HARI PRASAD
44	14R11A05G3	SAI PRIYA MALLIKA K
45	14R11A05G4	SANKARAMANCHI MOUNIKA
46	14R11A05G5	SEEMALA MAHENDER SUNNY SAMUEL
47	14R11A05G6	SUJAN MOHAN PILLI
48	14R11A05G7	T C KAVERI
49	14R11A05G8	T SWATHI
50	14R11A05G9	TERETIPALLY KARUNA SRI
51	14R11A05H0	V ABHISHEK
52	14R11A05H1	V VENKATA SAI YASASWI
53	14R11A05H2	VALLAP NITHIN REDDY
54	14R11A05H3	VASI SAI DINESH
55	14R11A05H4	VATTI BHARGAVI
56	14R11A05H5	VELPULA SANDHYA RANI
57	14R11A05H6	VIDYA SRIJA VOLETI
58	14R11A05H7	Y SREEJA
59	15R15A0503	MAALOTHU GANESH
60	15R18A0501	SANDEEP G BURUD

Class / Section: CSE 22D		
SlNo	**AdmnNo**	**StudentName**
1	14R11A05H9	A SAHITH REDDY
2	14R11A05J0	ADITYA V S P
3	14R11A05J1	ANEM PUNEET SURYA
4	14R11A05J2	ANNAVARAPU NAGA SAI SRILEKHA
5	14R11A05J4	B VINAY
6	14R11A05J5	BALABADRA SNEHA
7	14R11A05J6	BAYYAPU KEERTHANA
8	14R11A05J7	BHASWANTH K
9	14R11A05J8	BIJUMALLA SETHU MADHAV
10	14R11A05J9	BILLA HRIDAY VIKAS

11	14R11A05K0	BOLLEPALLY DIVYA
12	14R11A05K1	BUGATHA BALA GAYATRI
13	14R11A05K2	BYRU AKHILA
14	14R11A05K3	DHONEPUDI SHASHIKANTH
15	14R11A05K4	G DEEPIKA
16	14R11A05K5	GADDAM SUMIKENDAR REDDY
17	14R11A05K6	GARIKIPATI VINEETHA
18	14R11A05K7	GUNISETTY AKHIL
19	14R11A05K8	J ALIVELUMANGAVATHI
20	14R11A05K9	JESSICA SHINY THOMAS
21	14R11A05L0	KANDI SAI JAGADISH
22	14R11A05L1	KARTHIK KALVAGADDA
23	14R11A05L2	KATARAY SAMYUKTHA DEVI
24	14R11A05L3	KATHI GANESH GOUD
25	14R11A05L4	KATKAM SAI SRI
26	14R11A05L5	KATKURI NIHARIKA
27	14R11A05L6	KODANDAM SAI PRATHYUSH REDDY
28	14R11A05L7	KONDOJU ABHISHEK KUMAR
29	14R11A05L8	KOONAPAREDDY ETHESH KUMAR
30	14R11A05L9	MADDURU VENKATA SAI SHRUTHI
31	14R11A05M0	MANDAVILLI LALITH KUMAR
32	14R11A05M1	MARGUM ALEKHYA
33	14R11A05M2	MD AKBAR
34	14R11A05M3	MEKA NAVEENA
35	14R11A05M4	MOHAMMED FURQUAN AHMED
36	14R11A05M5	MOUNIKA V
37	14R11A05M6	MUTHYAM LIKITHA SREE
38	14R11A05M7	NARU SIVA SAI KARTHIK
39	14R11A05M8	NEMANI HEMANTH KUMAR
40	14R11A05M9	NUTHALAKANTI PRANAV KUMAR NANU
41	14R11A05N0	P DIVYA TEJA
42	14R11A05N1	PANANGIPALLI NAGA SAI MOUNIKA
43	14R11A05N2	PATURU NAGA SRI NIKHILENDRA
44	14R11A05N3	PRIYANKA PANDA
45	14R11A05N4	PS MANIVENKAT RATNAM MUTYALA
46	14R11A05N5	R.HARI SHANKER REDDY
47	14R11A05N6	RAMYA DUBAGUNTA
48	14R11A05N7	SATLA MOUNIKA
49	14R11A05N8	SHAIK JAVEERIA
50	14R11A05N9	SINGIREDDY AKHILA
51	14R11A05P0	SINGIREDDY MANEESHA

52	14R11A05P1	SOMIDI LEKHANA
53	14R11A05P2	T HARSHINI
54	14R11A05P3	TALATH FATHIMA
55	14R11A05P4	THOUDOJU RAMAKRISHNA
56	14R11A05P5	TIRUMALARAJU KARTHIK RAMA KRISHNA V
57	14R11A05P6	VALABOJU VAMSHI KRISHNA
58	14R11A05P7	VARUN R SHAH
59	15R15A0504	ACHHI PHANINDRA
60	15R15A0505	P NANDA SAI

26. Group wise Student list for Discussion

Class / Section: CSE 22A			
SlNo	**AdmnNo**	**StudentName**	**Group**
1	13R11A05G3	SRIKANTH AKKANAPALLY	Group-1
2	14R11A0501	AKHILESH G	
3	14R11A0502	AKULA KRISHNA	
4	14R11A0503	ALLAM USHA SREE	
5	14R11A0504	ANANT SRIVATS	Group-2
6	14R11A0505	AYYAGARI V S V VARDHAMAN	
7	14R11A0506	B MOHANA SREYA	
8	14R11A0507	B PRASHANTH	
9	14R11A0508	BANDA GAYATHRI VEENA	Group-3
10	14R11A0509	BATTAR SRIKANTH RAGHAVA	
11	14R11A0510	BHONSLE SAI KRUTHI	
12	14R11A0511	BONAGIRI TEJASHWINI	
13	14R11A0512	BOTLA MOUNIKA	Group-4
14	14R11A0513	D VENKATESH	
15	14R11A0514	DASARI PRANAY KUMAR	
16	14R11A0515	DEEPIKA TUDIMILA	
17	14R11A0516	DOKKA DIVYA	Group-5
18	14R11A0517	E MADHU BHARGAVA	
19	14R11A0518	K JWALA	
20	14R11A0519	K SAMHITHA REDDY	
21	14R11A0520	K THAPASVI REDDY	Group-6
22	14R11A0521	KALAKONDA VAISHNAVI	
23	14R11A0522	KAMPALLY YAMINI	
24	14R11A0523	KONTHAM RAVITEJA REDDY	
25	14R11A0524	KOSARAJU LEELA KRISHNA	Group-7
26	14R11A0525	KRISHNA SWETHA R	
27	14R11A0526	KUNDUR ROHAN REDDY	
28	14R11A0527	KUNNUMAL MAVILA AMRITHA	
29	14R11A0528	KURMA BHARGAV	Group-8

30	14R11A0529	KUTHURU PRIYANKA	
31	14R11A0530	KYASNOORI SHIVANI	
32	14R11A0531	M K SREE HARSHA	
33	14R11A0532	M POOJA	Group-9
34	14R11A0533	M SAI KRISHNA	
35	14R11A0534	M SHANMUGHA PRIYA	
36	14R11A0535	MALLADI RAMYA	
37	14R11A0536	MEESALA SHAILAJA	Group-10
38	14R11A0537	N SRI MANASVI	
39	14R11A0538	NADENDLA VENKATA VINEET MURARI	
40	14R11A0539	NAKKA AKHIL	
41	14R11A0540	NALLANAGULA BHANU PRAKASH	Group-11
42	14R11A0541	P SOWMYA	
43	14R11A0542	PARVATHALA CHETANA	
44	14R11A0543	PENDYALA HEMANTH SAI	
45	14R11A0544	R CHANDRA MOHAN	Group-12
46	14R11A0545	REKULA CHANDRAHASA	
47	14R11A0546	RUDRARAJU PAVANI	
48	14R11A0547	S SWATHI	
49	14R11A0548	SAI SINDHU BEERAM	Group-13
50	14R11A0549	SOPPADANDI AISHWARYA RAJ	
51	14R11A0550	SYED VAQUAS ASHRAF	
52	14R11A0551	TALLURI MURALI KRISHNA	
53	14R11A0552	TAMMEWAR SNEHIT	Group-14
54	14R11A0553	THIRIVEEDI DHEERAJ KUMAR	
55	14R11A0554	THUMMA VINEETHA REDDY	
56	14R11A0555	VALLURU VENUMADHURI	
57	14R11A0556	VAMANA GUNTLA MANOJ	Group-15
58	14R11A0557	VENKATA SAITEJA Y	
59	14R11A0558	VUSAKA VIKAS REDDY	
60	14R11A0559	Y JYOTSNA	

Class / Section: CSE 22B			
SlNo	**AdmnNo**	**StudentName**	**Group**
1	13R11A0561	BATTU SHIRISH KUMAR REDDY	Group-1
2	14R11A0560	AKELLA SUBRAMANYA SOUJANYA	
3	14R11A0561	ALEENA CHACKO K	
4	14R11A0562	BARLA PRAVALIKA	
5	14R11A0563	BATHINI DIVYA	Group-2
6	14R11A0564	BOLLA NAVEEN REDDY	
7	14R11A0565	CHAVALI SWATHI	

8	14R11A0566	CHEPYALA AKHIL	
9	14R11A0567	CHITTARVU LAKSHMI ISHWARYA	Group-3
10	14R11A0568	DAMAGALLA KARTIK	
11	14R11A0569	DHATRI NAIDU BHOGADI	
12	14R11A0571	DODDAPANENI SANOJ	
13	14R11A0572	DUGGISHETTY SAI PRASANNA	Group-4
14	14R11A0573	E PAVAN CHANDRA	
15	14R11A0574	G BHAVANA	
16	14R11A0575	G KISHAN	
17	14R11A0576	G LASYA PRIYA	Group-5
18	14R11A0577	G SIRIVENNELA	
19	14R11A0578	GANAPATHI RAJU MADHURI	
20	14R11A0579	GANGAVARAPU MANASA	
21	14R11A0580	GATTOJI HARITHA	Group-6
22	14R11A0581	GAYATHRI MANDAL	
23	14R11A0582	GORTI SANTOSH KUMAR	
24	14R11A0583	HRITIK S	
25	14R11A0584	KALYANAM KRUTHIKA REDDY	Group-7
26	14R11A0585	KANUMURI SRI PRATHYUSHA	
27	14R11A0586	KOLLU VINESH BABU	
28	14R11A0587	KONDISETTI NIRANJANA KUMARI	
29	14R11A0588	KONIKA VENKATA PADMA PRIYA HASINI	Group-8
30	14R11A0589	KURUP MEGHANA	
31	14R11A0590	MADARAJU RAMYA SAI	
32	14R11A0591	MEDIPALLY SRIYA	
33	14R11A0592	MUDDU DEEPTHI	Group-9
34	14R11A0593	N LAHARI	
35	14R11A0594	N VIKHYAT	
36	14R11A0595	NAKEERTHA SANDHYA RANI	
37	14R11A0596	NALLAMILLI JYOTHI	Group-10
38	14R11A0597	NALLAPANENI ADITYA SAI	
39	14R11A0598	NAMBURI MANISHA	
40	14R11A0599	NIKHIL SINGH P	
41	14R11A05A0	ORUGANTI KALPANA	Group-11
42	14R11A05A1	PATLORI SAI KUMAR	
43	14R11A05A2	PEESAPATI VENKATA SAI VAMSI	
44	14R11A05A3	PINDIPOLU SRAVYA	
45	14R11A05A4	PRANATHI KRISHNA SANGANI	Group-12
46	14R11A05A5	PUDHOTA AVINASH	
47	14R11A05A6	S ASHA LAKSHMI	
48	14R11A05A7	S CHANDANA REDDY	

49	14R11A05A8	SHAIK HAFEEZ HUSSAIN	Group-13
50	14R11A05A9	SHEELAM SUSHMA	
51	14R11A05B0	SRAVIKA SANKU	
52	14R11A05B1	THUMMALA RISHIKA REDDY	
53	14R11A05B2	VARSHA CHAHAL	Group-14
54	14R11A05B3	VEERAVALLI SHILPA	
55	14R11A05B4	VUPPUTTURI SUSHANTH KUMAR	
56	14R11A05B5	Y ADITHYA	
57	14R11A05B6	YEDDU SHASHI KUMAR	Group-15
58	14R11A05B7	KATAMONI HARSHITHA	
59	15R15A0501	B VINESH	
60	15R15A0502	B SHRAVAN KUMAR	

Class / Section: CSE 22C			
SlNo	**AdmnNo**	**StudentName**	**Group**
1	14R11A05C0	A S SRUJAN SAI	Group-1
2	14R11A05C1	AITHA RANJEETH KUMAR	
3	14R11A05C2	ALIGETI MAHITHA	
4	14R11A05C3	BAGGU VISHNU SAI PRASAD	
5	14R11A05C4	BARGELA PRANAY RAJ	Group-2
6	14R11A05C5	BIROJU SAI SUGANDH CHARY	
7	14R11A05C6	G AKHILA	
8	14R11A05C7	GANGJI MANISHA	
9	14R11A05C8	GARIKAPATI NAMRATHA CHOWDARY	Group-3
10	14R11A05C9	GARLAPATI RENUKA	
11	14R11A05D0	GUNDARAPU SOUJANYA REDDY	
12	14R11A05D1	GUNGI SAI KUMAR	
13	14R11A05D2	GUNNALA RAMYA SREE	Group-4
14	14R11A05D3	ILA BHAVANA	
15	14R11A05D4	J HEMANTH SAI	
16	14R11A05D5	JELLA MOUNICA	
17	14R11A05D6	K TEJA ABHINAV	Group-5
18	14R11A05D7	KANAKADANDI NAGA VENKATA APARNA	
19	14R11A05D8	KASI SRAVYA	
20	14R11A05D9	KAVYA S	
21	14R11A05E0	KUKKALA DEEPIKA	Group-6
22	14R11A05E1	LAHIRI KOTAMRAJU	
23	14R11A05E2	MARAM RAJEEV KUMAR	
24	14R11A05E3	MARGONWAR GAYATRI	
25	14R11A05E4	MEKALA SWATHI	Group-7
26	14R11A05E5	MOTHABOINA KAMESHWAR RAO ROHAN MUDH	

27	14R11A05E6	NALABOTHULA HARIKA	
28	14R11A05E7	NALADALA SAI CHARITHA	
29	14R11A05E8	NEMURI SAI SAMPATH GOUD	Group-8
30	14R11A05E9	NIKHIL THAPA	
31	14R11A05F0	P ANUDEEP	
32	14R11A05F1	P SAI PRASAD	
33	14R11A05F2	PARIMI SAIESH	Group-9
34	14R11A05F3	PINNINTI SAIKIRAN REDDY	
35	14R11A05F4	POLAGOUNI VAISHNAVI GOUD	
36	14R11A05F5	POTHURI VENKATA SAI PRIYANKA	
37	14R11A05F6	RAHUL N D	Group-10
38	14R11A05F7	RAI MEGHANA	
39	14R11A05F8	RAYALA PRIYANKA	
40	14R11A05F9	RAYAPROLU ASWINI	
41	14R11A05G0	REEBA FATIMA	Group-11
42	14R11A05G1	ROSHAN ROY	
43	14R11A05G2	S HARI PRASAD	
44	14R11A05G3	SAI PRIYA MALLIKA K	
45	14R11A05G4	SANKARAMANCHI MOUNIKA	Group-12
46	14R11A05G5	SEEMALA MAHENDER SUNNY SAMUEL	
47	14R11A05G6	SUJAN MOHAN PILLI	
48	14R11A05G7	T C KAVERI	
49	14R11A05G8	T SWATHI	Group-13
50	14R11A05G9	TERETIPALLY KARUNA SRI	
51	14R11A05H0	V ABHISHEK	
52	14R11A05H1	V VENKATA SAI YASASWI	
53	14R11A05H2	VALLAP NITHIN REDDY	Group-14
54	14R11A05H3	VASI SAI DINESH	
55	14R11A05H4	VATTI BHARGAVI	
56	14R11A05H5	VELPULA SANDHYA RANI	
57	14R11A05H6	VIDYA SRIJA VOLETI	Group-15
58	14R11A05H7	Y SREEJA	
59	15R15A0503	MAALOTHU GANESH	
60	15R18A0501	SANDEEP G BURUD	

Class / Section: CSE 22D			
SlNo	**AdmnNo**	**StudentName**	**Group**
1	14R11A05H9	A SAHITH REDDY	Group-1
2	14R11A05J0	ADITYA V S P	
3	14R11A05J1	ANEM PUNEET SURYA	
4	14R11A05J2	ANNAVARAPU NAGA SAI SRILEKHA	

5	14R11A05J4	B VINAY	Group-2
6	14R11A05J5	BALABADRA SNEHA	
7	14R11A05J6	BAYYAPU KEERTHANA	
8	14R11A05J7	BHASWANTH K	
9	14R11A05J8	BIJUMALLA SETHU MADHAV	Group-3
10	14R11A05J9	BILLA HRIDAY VIKAS	
11	14R11A05K0	BOLLEPALLY DIVYA	
12	14R11A05K1	BUGATHA BALA GAYATRI	
13	14R11A05K2	BYRU AKHILA	Group-4
14	14R11A05K3	DHONEPUDI SHASHIKANTH	
15	14R11A05K4	G DEEPIKA	
16	14R11A05K5	GADDAM SUMIKENDAR REDDY	
17	14R11A05K6	GARIKIPATI VINEETHA	Group-5
18	14R11A05K7	GUNISETTY AKHIL	
19	14R11A05K8	J ALIVELUMANGAVATHI	
20	14R11A05K9	JESSICA SHINY THOMAS	
21	14R11A05L0	KANDI SAI JAGADISH	Group-6
22	14R11A05L1	KARTHIK KALVAGADDA	
23	14R11A05L2	KATARAY SAMYUKTHA DEVI	
24	14R11A05L3	KATHI GANESH GOUD	
25	14R11A05L4	KATKAM SAI SRI	Group-7
26	14R11A05L5	KATKURI NIHARIKA	
27	14R11A05L6	KODANDAM SAI PRATHYUSH REDDY	
28	14R11A05L7	KONDOJU ABHISHEK KUMAR	
29	14R11A05L8	KOONAPAREDDY ETHESH KUMAR	Group-8
30	14R11A05L9	MADDURU VENKATA SAI SHRUTHI	
31	14R11A05M0	MANDAVILLI LALITH KUMAR	
32	14R11A05M1	MARGUM ALEKHYA	
33	14R11A05M2	MD AKBAR	Group-9
34	14R11A05M3	MEKA NAVEENA	
35	14R11A05M4	MOHAMMED FURQUAN AHMED	
36	14R11A05M5	MOUNIKA V	
37	14R11A05M6	MUTHYAM LIKITHA SREE	Group-10
38	14R11A05M7	NARU SIVA SAI KARTHIK	
39	14R11A05M8	NEMANI HEMANTH KUMAR	
40	14R11A05M9	NUTHALAKANTI PRANAV KUMAR NANU	
41	14R11A05N0	P DIVYA TEJA	Group-11
42	14R11A05N1	PANANGIPALLI NAGA SAI MOUNIKA	
43	14R11A05N2	PATURU NAGA SRI NIKHILENDRA	
44	14R11A05N3	PRIYANKA PANDA	
45	14R11A05N4	PS MANIVENKAT RATNAM MUTYALA	Group-12

46	14R11A05N5	R.HARI SHANKER REDDY	
47	14R11A05N6	RAMYA DUBAGUNTA	
48	14R11A05N7	SATLA MOUNIKA	
49	14R11A05N8	SHAIK JAVEERIA	Group-13
50	14R11A05N9	SINGIREDDY AKHILA	
51	14R11A05P0	SINGIREDDY MANEESHA	
52	14R11A05P1	SOMIDI LEKHANA	
53	14R11A05P2	T HARSHINI	Group-14
54	14R11A05P3	TALATH FATHIMA	
55	14R11A05P4	THOUDOJU RAMAKRISHNA	
56	14R11A05P5	TIRUMALARAJU KARTHIK RAMA KRISHNA V	
57	14R11A05P6	VALABOJU VAMSHI KRISHNA	Group-15
58	14R11A05P7	VARUN R SHAH	
59	15R15A0504	ACHHI PHANINDRA	
60	15R15A0505	P NANDA SAI	