Microprocessor and its Applications

1. R.S. Gaonkar: “Microprocessor architecture, Programming and Applications with 8085/8080”, Penram Publication.
2. B.Ram : “ Fundamental of Microprocessor and Microcomputer”, Dhanpat Rai Publication, 4th edition.
3. R. Singh and B.P. Singh: “ Microprocessor Interfacing and its application”, New Age International Publishers, 2nd Edition.
4. D.V. Hall: “ Microprocessor Interfacing”, TMH (Revised 2nd Edition).
5. R. Singh and B.P. Singh: “Advanced Microprocessor and Micro-controllers”, New Age International Publishers, 2nd Edition

UNIT 1

Category of Memory, Microprocessor, Microcontroller, Buses, machine Language, Assembly Language, High Level Language, Assembly Language Program Development Tool.

8085 Microprocessor:

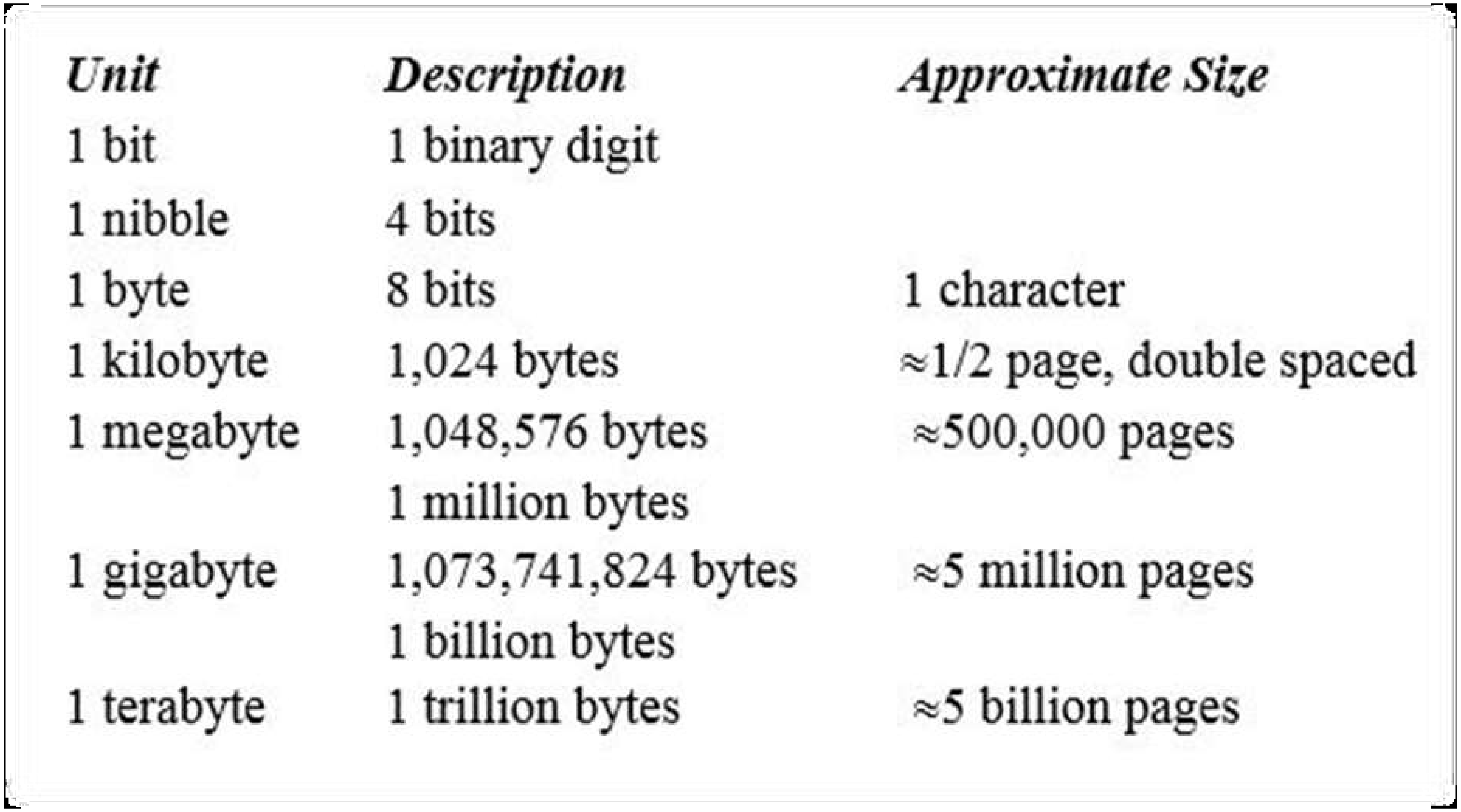
Architecture, Pin diagram, Instruction Type, Instruction Cycle, Timing Diagram, Addressing Modes, Instruction Set. Assembly Programming based on 8085, Interrupt and Interrupt Service Routine.

What is Memory?

Computer memory is any physical device capable of storing information temporarily or permanently.

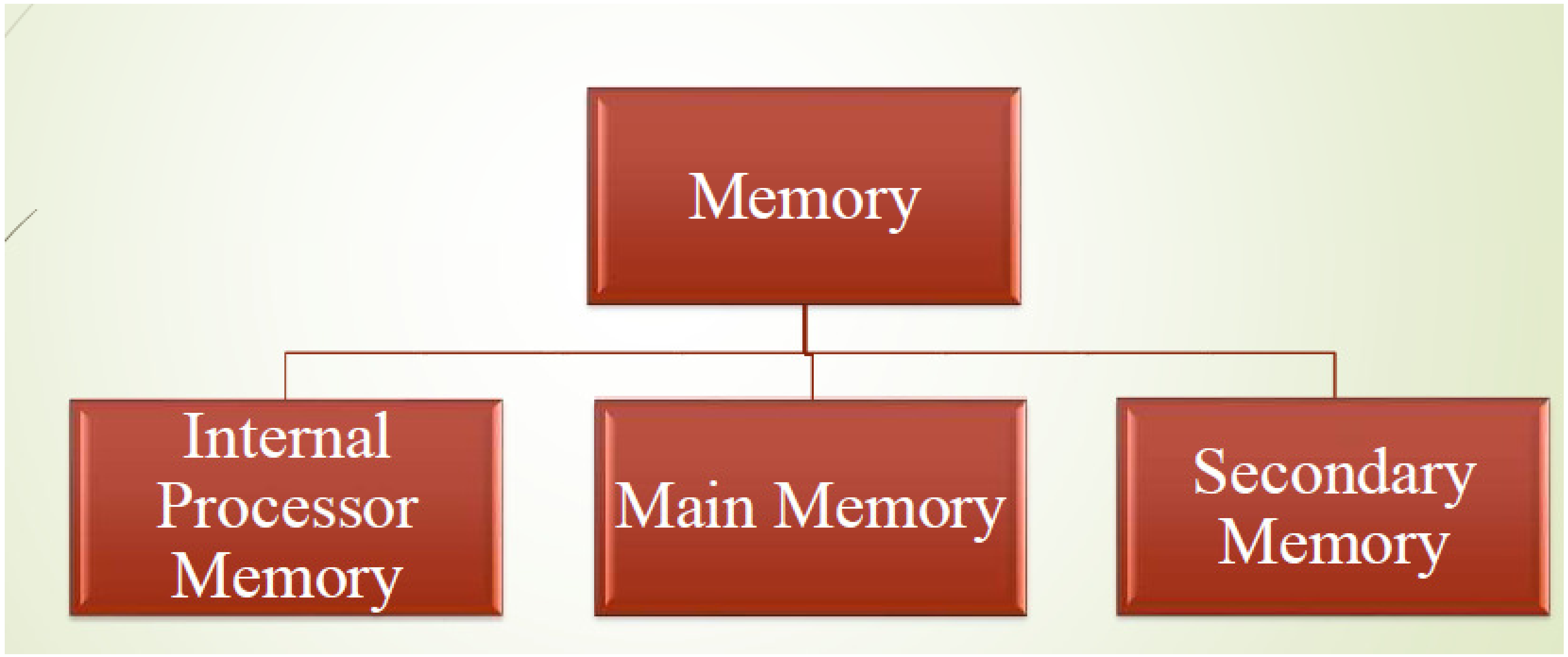
- Electronic components that store instructions, data, and results.
- Each byte stored in unique location called an address, similar to seats in a concert hall

MEMORY CAPACITY



Unit	*Description*	*Approximate Size*
1 bit	1 binary digit	
1 nibble	4 bits	
1 byte	8 bits	1 character
1 kilobyte	1,024 bytes	≈1/2 page, double spaced
1 megabyte	1,048,576 bytes	≈500,000 pages
	1 million bytes	
1 gigabyte	1,073,741,824 bytes	≈5 million pages
	1 billion bytes	
1 terabyte	1 trillion bytes	≈5 billion pages

CLASSIFICATION OF MEMORY



INTERNAL PROCESSOR MEMORY

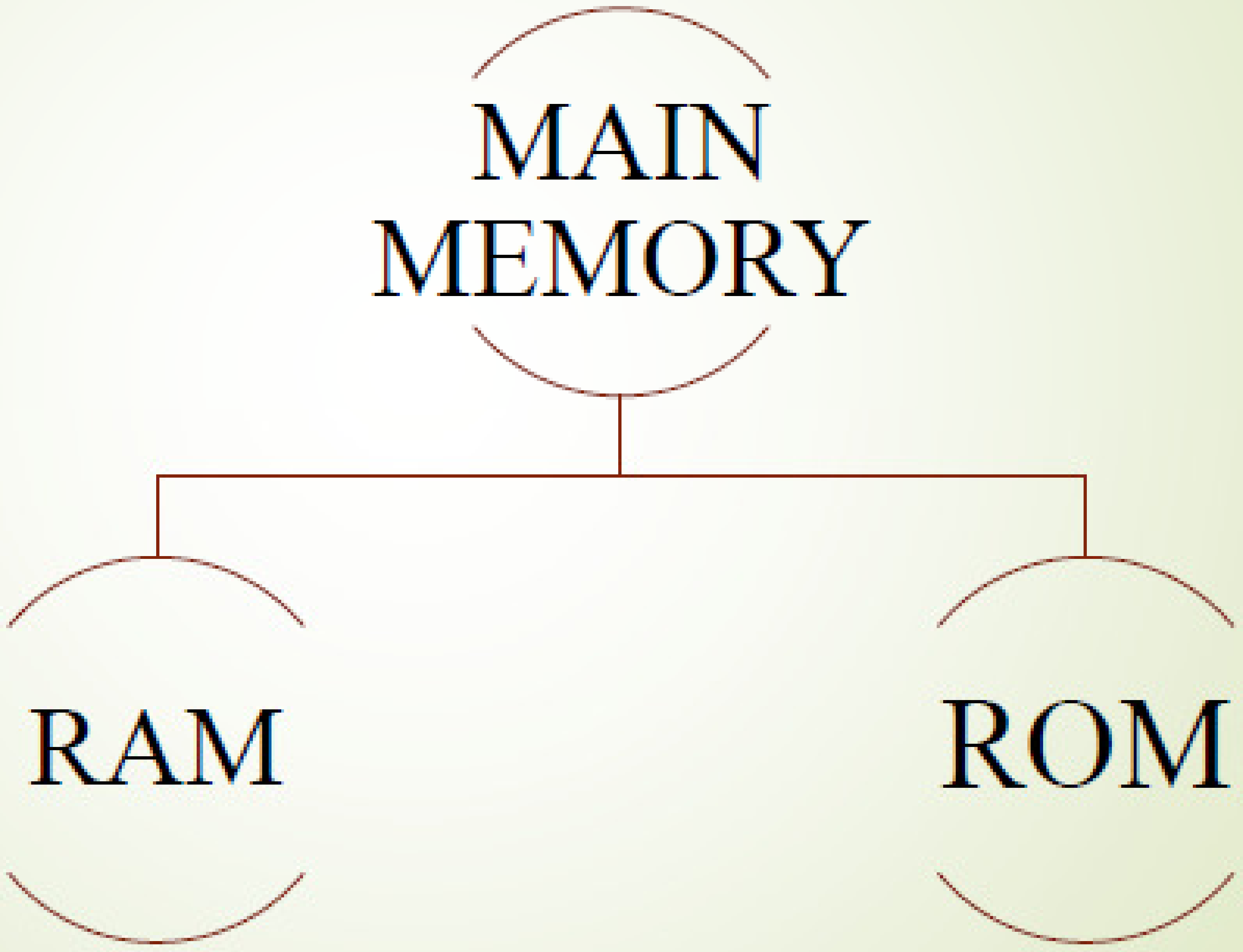
- A small high speed memory inside the processor.
- Temporary storage of instruction and data.
- Example: Registers, built-in cache.
- Like in Intel 8085 Microprocessor there are 6, 8-bit Registers (B,C,D,E,H,L)

MAIN MEMORY

- It is relatively large memory placed outside the processor.
- Data and instruction storage for the operation of the processor.
- Can be accessed directly and rapidly by the CPU.
- Example: RAM, ROM

IMAGES OF MAIN MEMORY

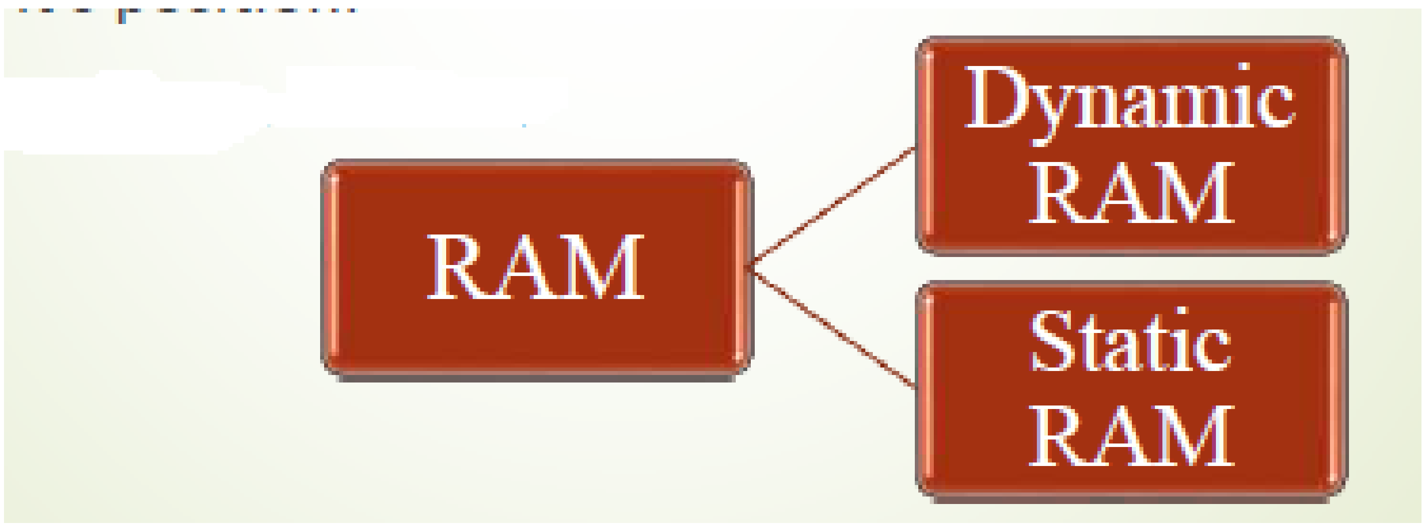
TYPES



RANDOM ACCESS MEMORY

- A temporary storage that can be read from or written into by the user.
- Volatile memory.
- Every location can be accessed independently.
- Access time for every location is constant and independent of it’s position.

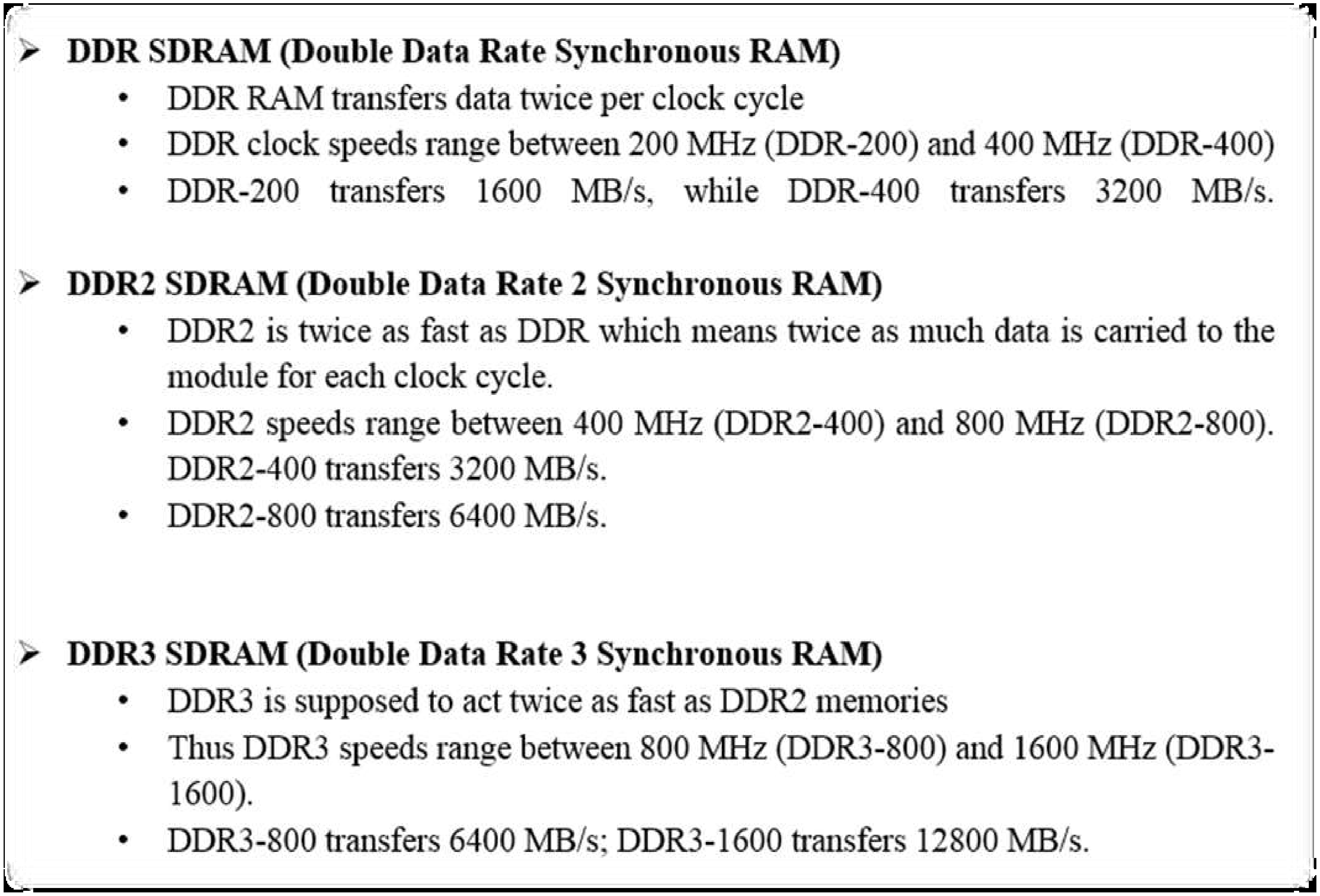
Two types of RAM



DYNAMIC RAM

- DRAM stands for Dynamic RAM.
- Relatively slower and low cost memory.
- Used for main memory.
- Contents are constantly refreshed 1000 times per second
- Access time 60 – 70 nanoseconds

TYPES OF DYNAMIC RAM



- **DDR SDRAM (Double Data Rate Synchronous RAM)**
  - DDR RAM transfers data twice per clock cycle
  - DDR clock speeds range between 200 MHz (DDR-200) and 400 MHz (DDR-400)
  - DDR-200 transfers 1600 MB/s, while DDR-400 transfers 3200 MB/s.
- **DDR2 SDRAM (Double Data Rate 2 Synchronous RAM)**
  - DDR2 is twice as fast as DDR which means twice as much data is carried to the module for each clock cycle.
  - DDR2 speeds range between 400 MHz (DDR2-400) and 800 MHz (DDR2-800). DDR2-400 transfers 3200 MB/s.
  - DDR2-800 transfers 6400 MB/s.
- **DDR3 SDRAM (Double Data Rate 3 Synchronous RAM)**
  - DDR3 is supposed to act twice as fast as DDR2 memories
  - Thus DDR3 speeds range between 800 MHz (DDR3-800) and 1600 MHz (DDR3-1600).
  - DDR3-800 transfers 6400 MB/s; DDR3-1600 transfers 12800 MB/s.

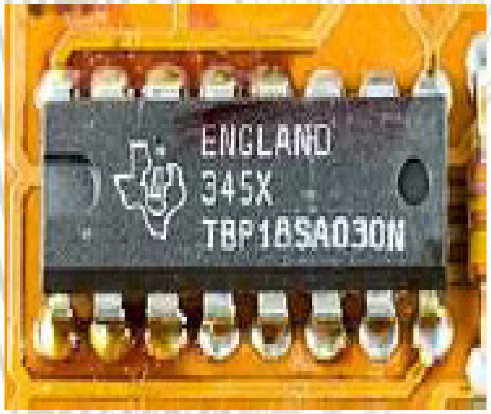
STATIC RAM

- SRAM stands for static RAM.
- Characterized by high speed and high cost.
- Use six transistors to store data.
- Access time 60 – 70 nanoseconds
- Can accept one command and transfer one word of
- data per clock cycle.

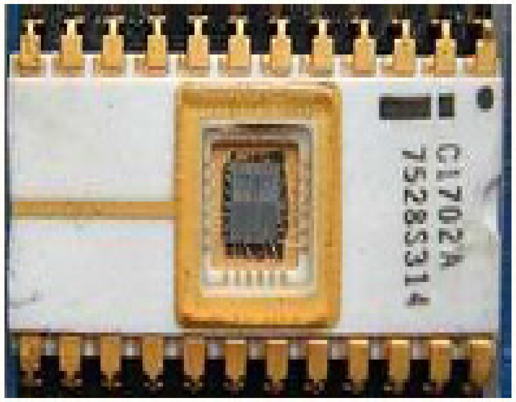
READ ONLY MEMORY (ROM)

- A class of storage used in computer and other electronic
- devices.
- Data stored in it cannot be modified
- Non-volatile memory.
- In modern PCs, ROM is used to store the basic bootstrapping
- firmware for the main processor, as well as the various firmware
- needed to internally control self-contained devices such as graphics card, hard disks, DVD drives, etc.

TYPES OF ROM



Programmable read-only memory (PROM)



Easable programmable read-only memory(EPROM)

Electrically Erasable Programmable Read-Only Memory(EEPROM)

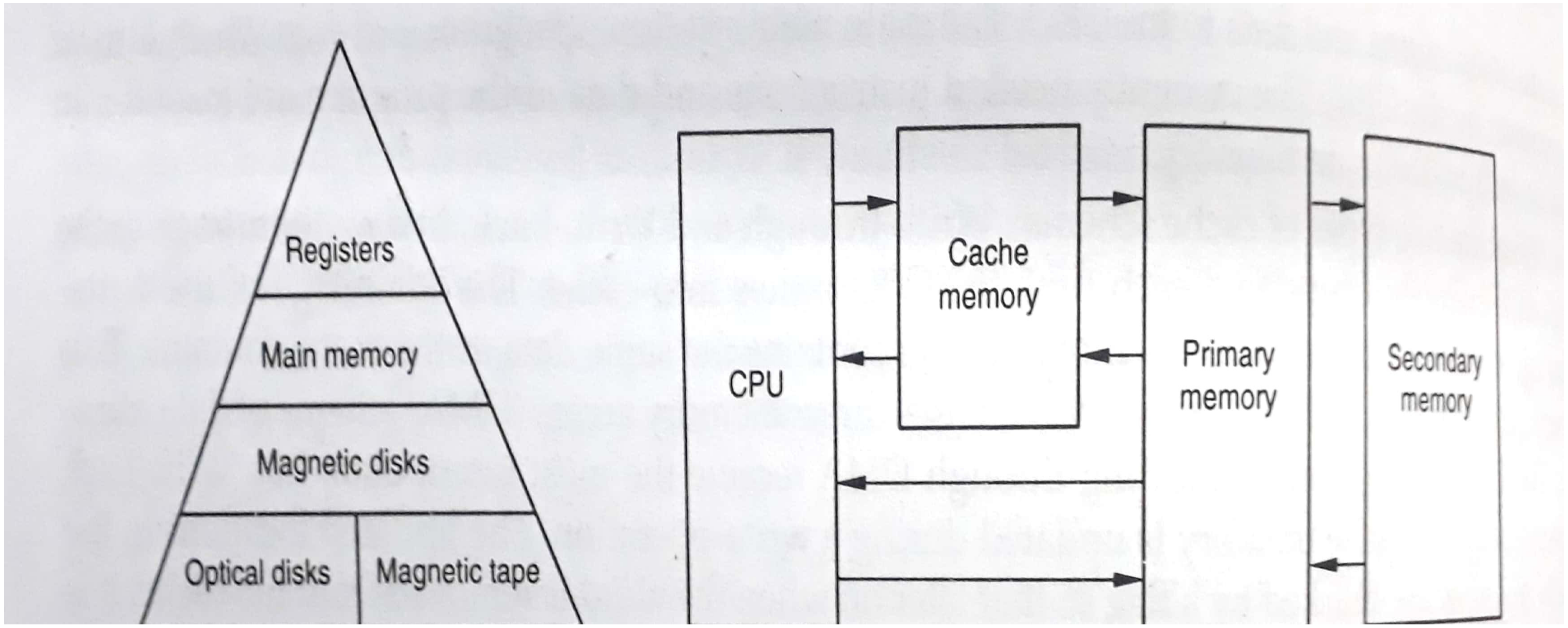
SECONDARY MEMORY

- Much larger in capacity but slower than main memory.
- Permanent storage of data and instruction.
- Example: Hard disk, CD, Floppy etc.

MEMORY HIERARCHY

- In computer architecture the **memory hierarchy is a concept used** to discuss performance issues in computer architectural design, algorithm predictions, and lower level programming constructs.
- Involving locality of reference. The **memory hierarchy in computer** storage separates each of its levels based on response time

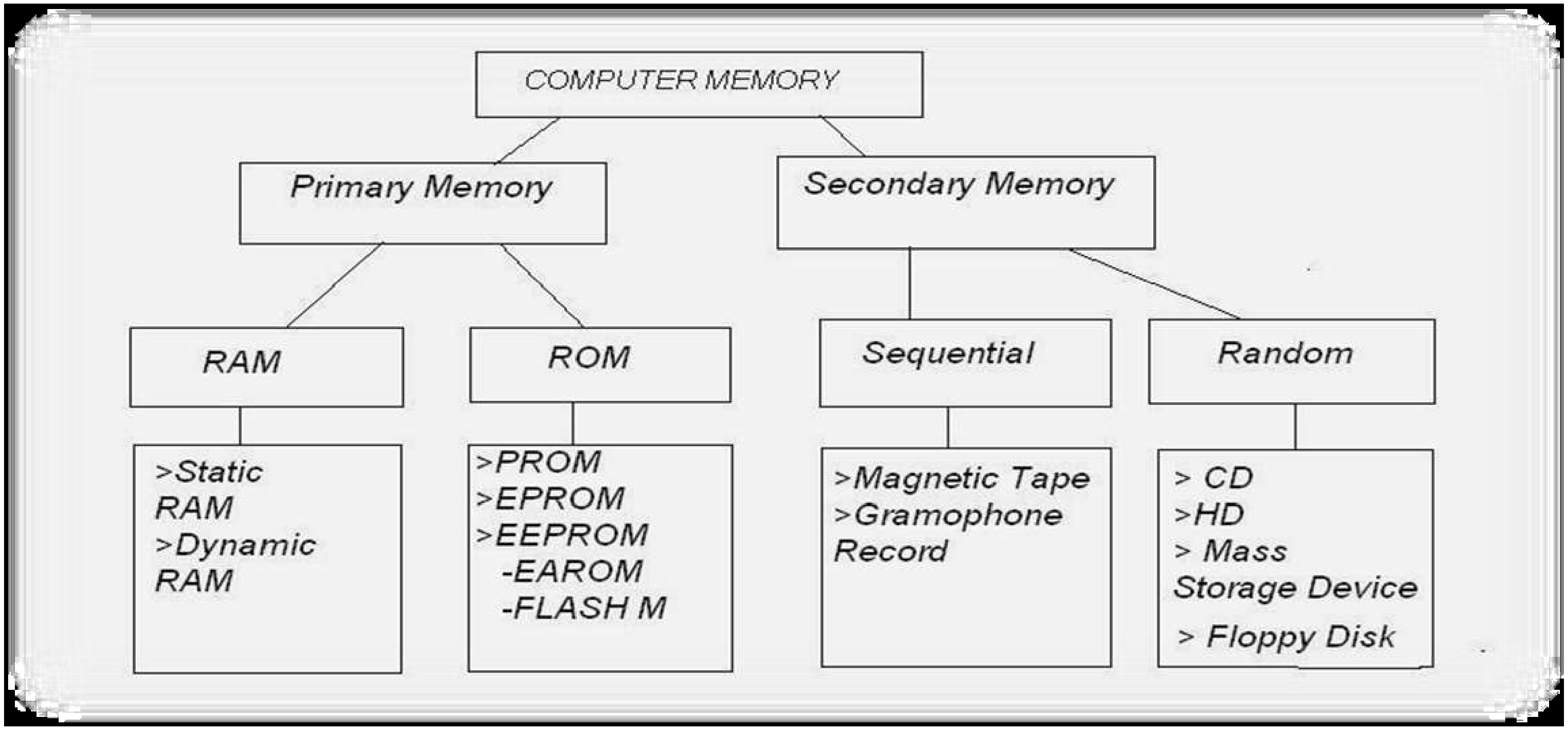
Memory Hierarchy



Cache Memory

- System performance suffers when processor waits for data from slow memory device.
- Cache memory is introduced between the CPU and the main memory.
- Cache is a high speed memory for holding recently accessed data in main memory.
- The amount of cache memory has a tremendous impact on the computer's speed.

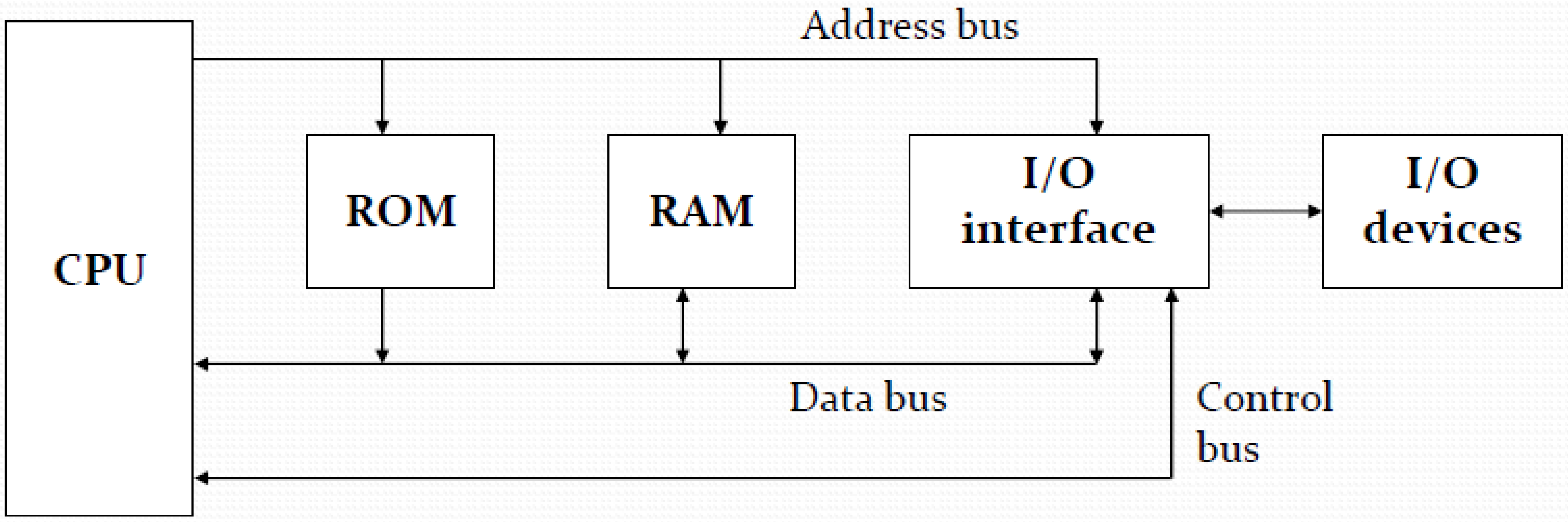
AT A GALNCE


COMPUTER MEMORY
Primary Memory
Secondary Memory
RAM
ROM
Sequential
Random
>Static RAM
>Dynamic RAM
>PROM
>EPROM
>EEPROM
-EAROM
-FLASH M
>Magnetic Tape
>Gramophone Record
> CD
>HD
> Mass Storage Device
> Floppy Disk

Introduction

Computer: A computer is a programmable machine that receives input, stores and manipulates data/information, and provides output in a useful format.

Basic computer system consist of a CPU, memory and I/O unit.



Block diagram of a basic computer system

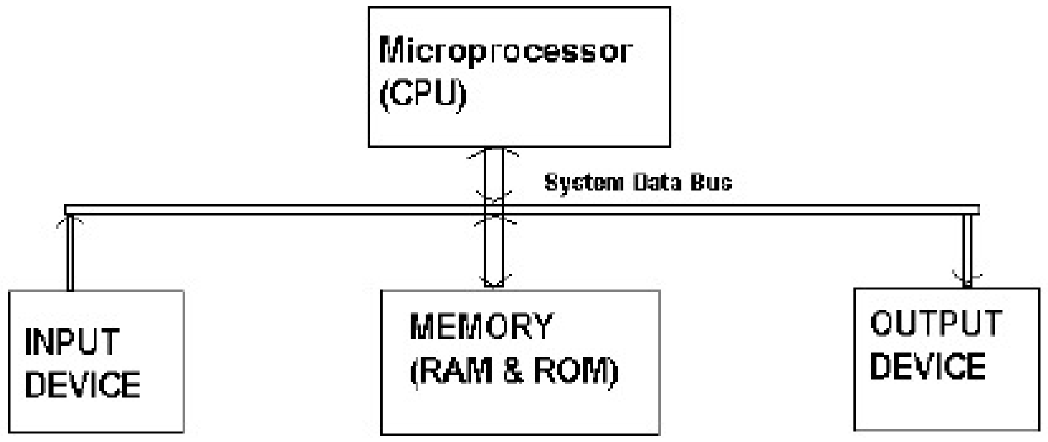
Basic Concepts of Microprocessors

- **Microcomputer:-** It is a programmable machine. The two principal characteristics of a computer are:
  - Responds to a specific set of instructions in a well-defined manner.
  - It can execute a prerecorded list of instructions (a program)
  - Its main components are
    - CPU
    - Input & Output devices
    - Memory

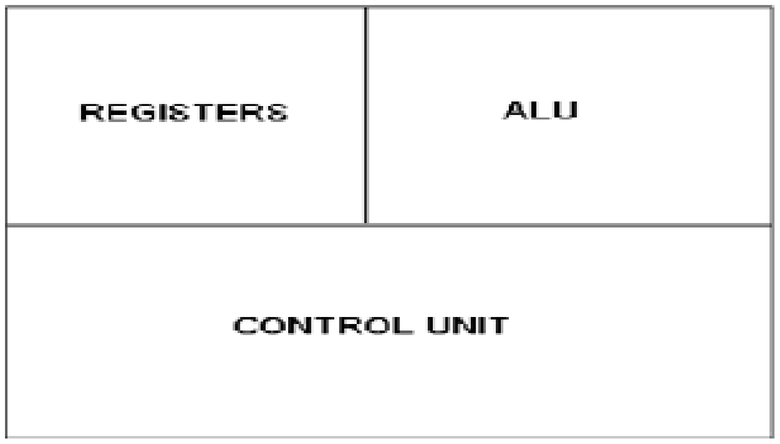
- **Microprocessor:-** It is a programmable VLSI chip which includes ALU, register circuits & control circuits. Its main units are-
  - ALU
  - Registers
  - Control Unit

- **Microcontroller:-** Silicon chip which includes microprocessor, memory & I/O in a single package

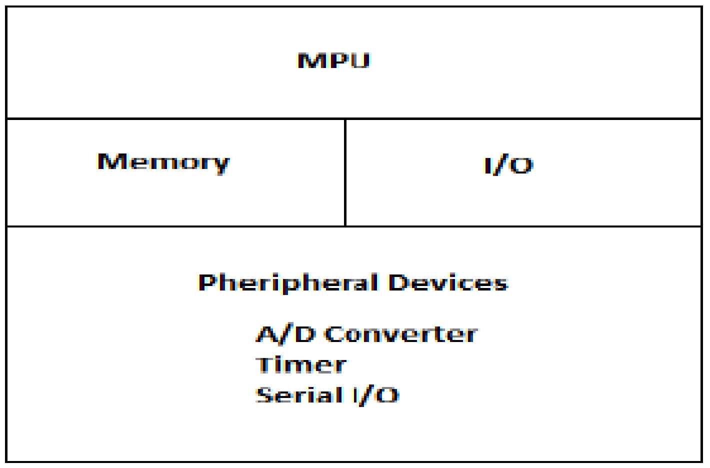
Block Diagram



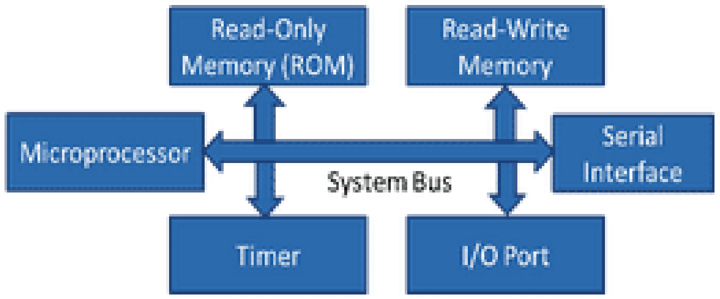
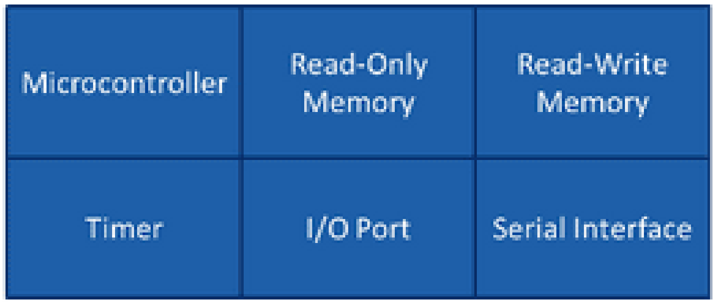
Microcomputer



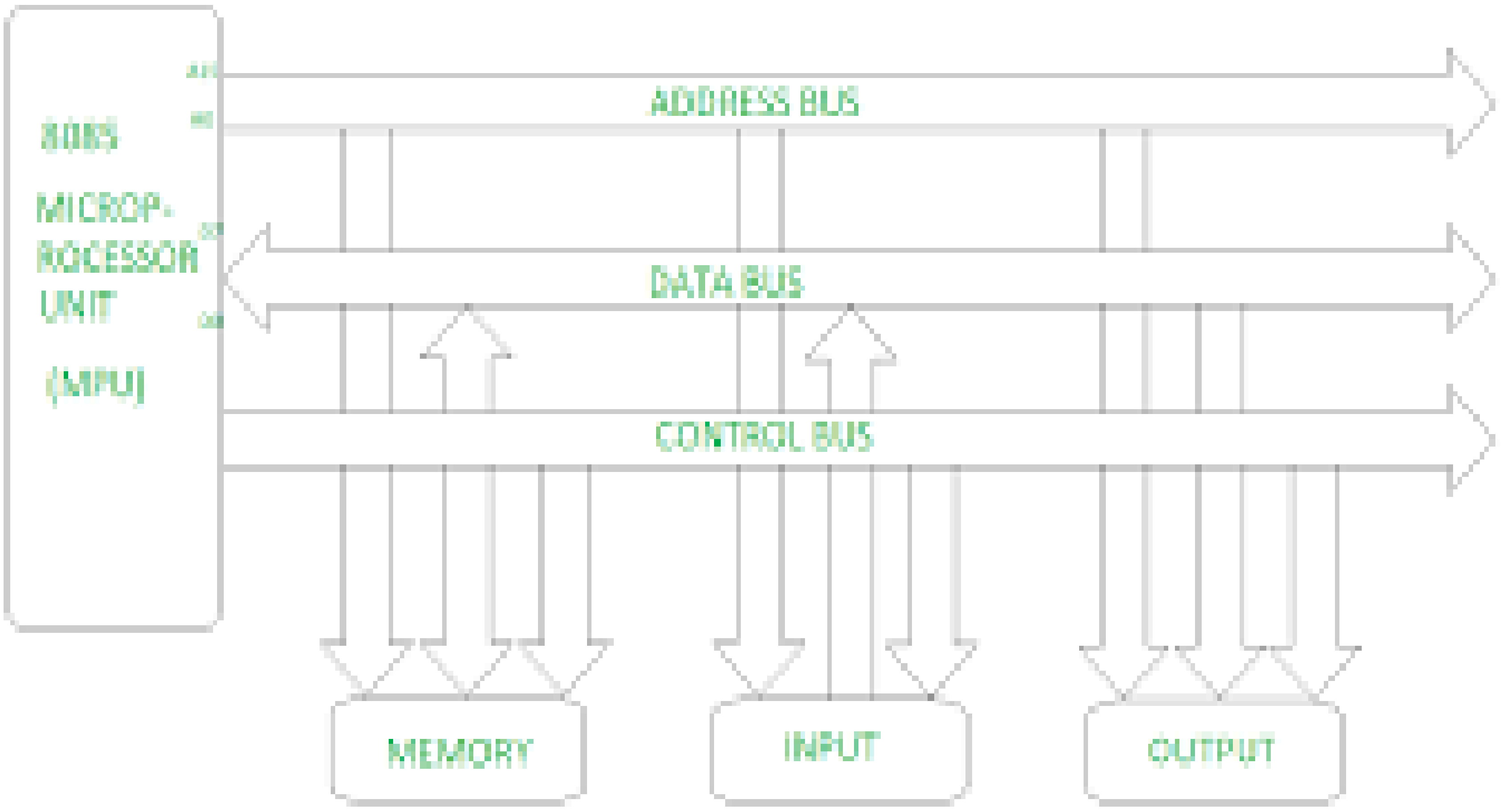
Microprocessor



Microcontroller

Microprocessor	Micro Controller
 Read-Only Memory (ROM) Read-Write Memory Microprocessor Serial Interface System Bus Timer I/O Port	 Microcontroller Read-Only Memory Read-Write Memory Timer I/O Port Serial Interface
Microprocessor is heart of Computer system.	Micro Controller is a heart of embedded system.
It is just a processor. Memory and I/O components have to be connected externally	Micro controller has external processor along with internal memory and i/O components
Since memory and I/O has to be connected externally, the circuit becomes large.	Since memory and I/O are present internally, the circuit is small.
Cannot be used in compact systems and hence inefficient	Can be used in compact systems and hence it is an efficient technique
Cost of the entire system increases	Cost of the entire system is low
Due to external components, the entire power consumption is high. Hence it is not suitable to used with devices running on stored power like batteries.	Since external components are low, total power consumption is less and can be used with devices running on stored power like batteries.
Most of the microprocessors do not have power saving features.	Most of the micro controllers have power saving modes like idle mode and power saving mode. This helps to reduce power consumption even further.

Buses

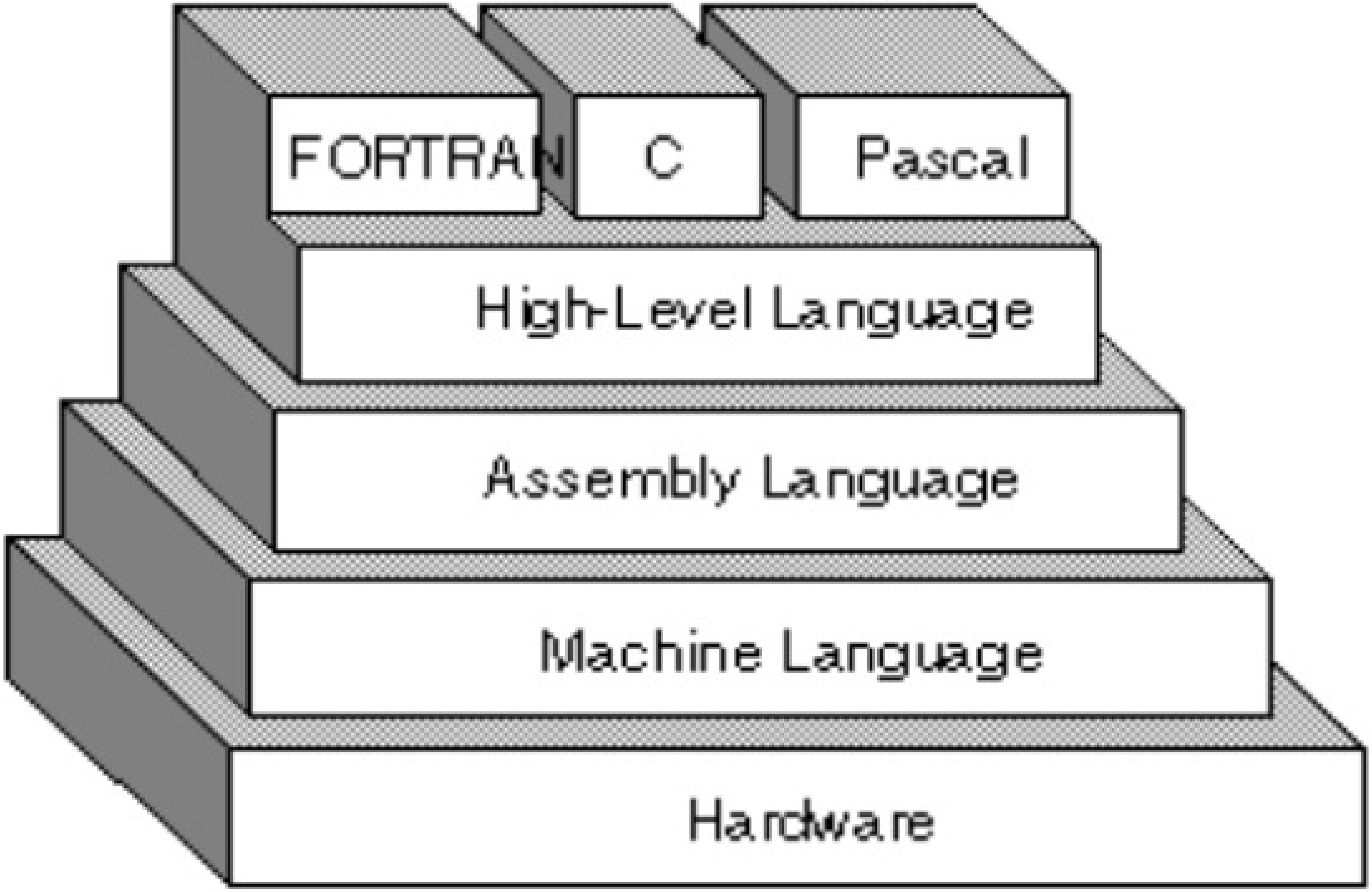

8085
MICROPROCESSOR UNIT
(MPU)
ADDRESS BUS
DATA BUS
CONTROL BUS
MEMORY
INPUT
OUTPUT

Computer Languages

Human languages are known as natural languages. Unfortunately, computers can not understand natural languages, as a result we must communicate with computers using computer languages. These languages are:

- High Level Languages
- Low Level Languages
    - Assembly Language
    - Machine Language

Architecture



* High level languages

It is a set of words and symbol which a programmer uses to write a program

High-level languages are much closer to human language.

A programming language such as C, FORTRAN or Pascal that enables to write programs which is understandable to programmer (Human) and can perform any sort of task, such languages are considered high-level because they are closer to human languages.

High level language must use interpreter, compiler or translator to convert human understandable program to computer readable code (machine code).

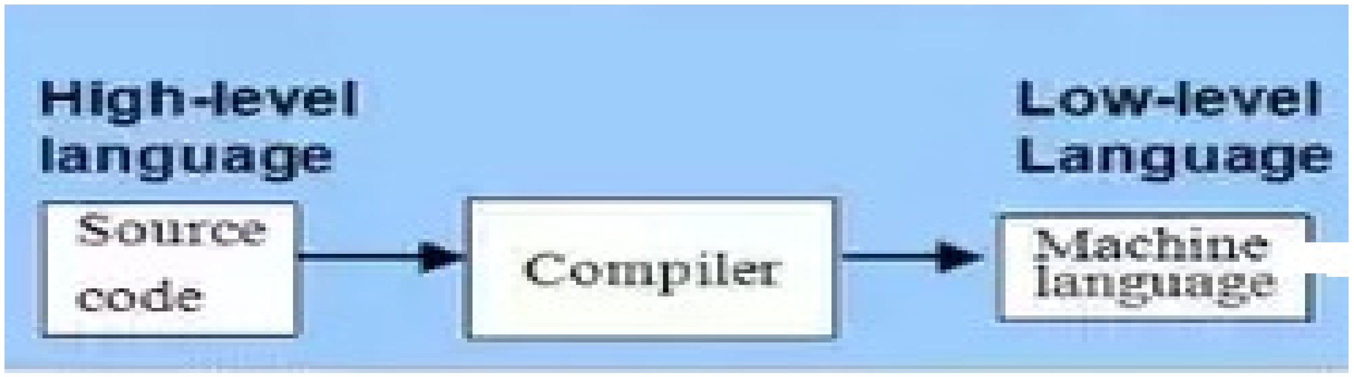
- There are many high level languages
  - Some Examples:
  - COBOL Business applications
  - FORTRAN Engineering & Scientific Applications
  - PASCAL General use and as a teaching tool
  - C & C++ General Purpose - currently most popular.
  - PROLOG Artificial Intelligence
  - JAVA General all purpose programming
  - .NET General or web applications

Advantages of High level language over low level language

- They are near to English language, that is they are easier to read, write and maintain.
- High-level languages make complex programming simpler.
- High level languages is portable, i.e., they can work on different operating system.
- Length of the program is also small compared with low level.
- Many real time problems can be easily solved with high level language.

Disadvantages of High level language over low level language

- They need to be translated for the computer to understand, hence work slower than machine code.



Low Level language

- A computer low level language that deals with hardware registers by name is known as assembly language.
- Assembly language is the best example of low level language, it is in between machine language and high-level language.
- A low-level language does not need a compiler or interpreter to run the program, the processor run low-level code directly.

Assembly languages

- Assembly languages have the same structure and set of commands as machine languages, but they enable a programmer to use names instead of numbers.
- In the early days of programming all programs were written in assembly language but now almost programs are written in a highlevel language.
- Each assembly language is specific to a particular computer architecture, in contrast to most high level programming languages, which are generally portable across multiple systems.
- *Programmers still use assembly language when speed is essential or when they need to perform an operation that isn't possible in a high-level language.

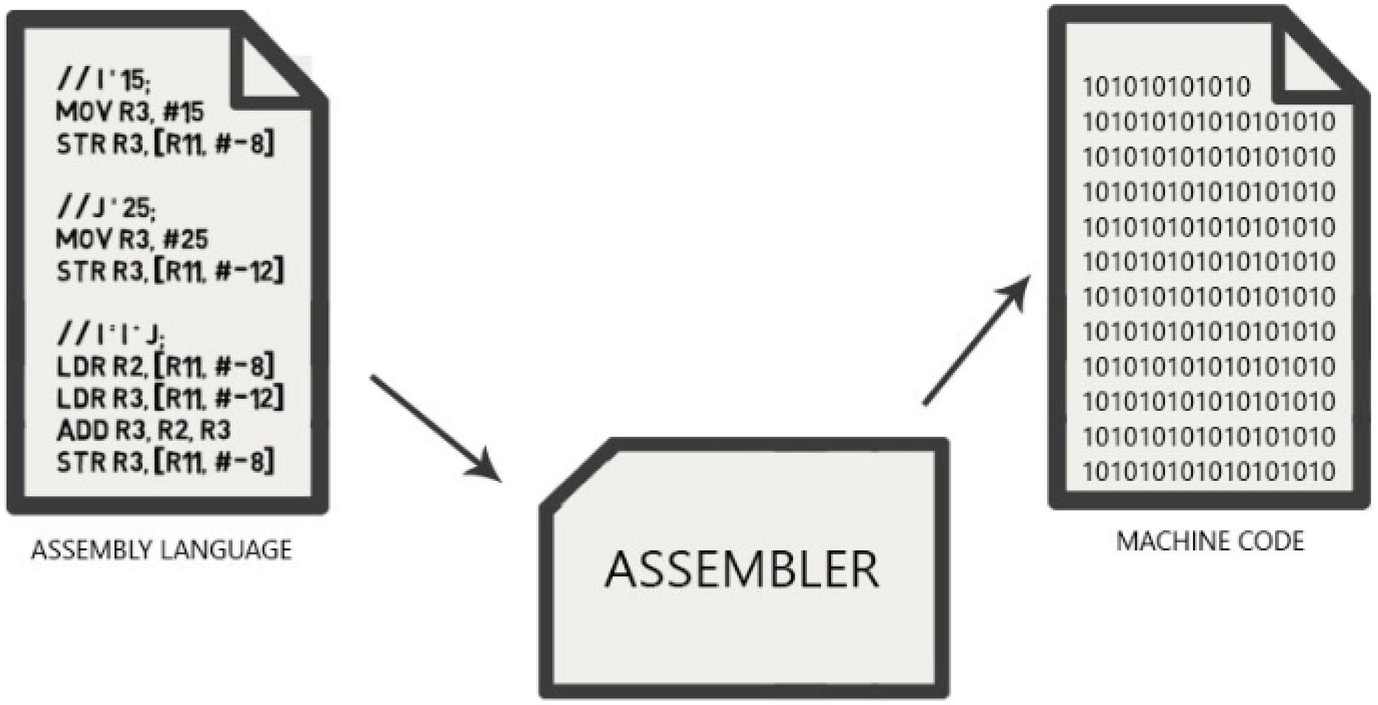
Applications of Assembly Language

- Writing games or simulation program
- Writing operating systems
- Programming robots
- Computers viruses, certain device drivers or other items very close to the hardware or low-level operating system.

Assembly languages Instructions

- It uses mnemonic codes (short forms) for instructions and allows the programmer to introduce names for blocks of memory that hold data.
- Assembly language is designed to be easily translated into machine language.
- Examples:
  - MOV AL, 1h ;
  - MVI A, 01
  - SUB AX, BX
  - MUL AL,02

Conversion of Assembly language to Machine Language



```
//I'15;
MOV R3, #15
STR R3, [R11, #-8]

//J'25;
MOV R3, #25
STR R3, [R11, #-12]

//I'I'J;
LDR R2, [R11, #-8]
LDR R3, [R11, #-12]
ADD R3, R2, R3
STR R3, [R11, #-8]
```

```
101010101010
101010101010101010
101010101010101010
101010101010101010
101010101010101010
101010101010101010
101010101010101010
101010101010101010
101010101010101010
101010101010101010
101010101010101010
101010101010101010
```

Machine language

Machine code or machine language is a system of instructions and data executed directly by a computer's CPU. The lowest-level programming language that only be understood by computers.

Computer language that is directly executable by a computer without the need for translation by a compiler or an assembler.

Machine language

- The native language of the computer, The set of symbolic instructions in binary that is used to represent operations and data in a machine called machine code
- Machine Language: “0110101100101000”
- machine language is a collection of binary digits or bits that the computer reads and interprets. Machine language is the only language a computer understands. It is almost impossible for humans to use because they consist entirely of numbers.

Assembly Language Program Development Tools

1. **Editor**

   - An editor is a program which allows you to create a file containing the assembly language statements for your program. Example: PC-Write, Wordstar.

   - When you have typed in all your program, you then save the file on the hard disk. This file is called *source file* and the extension is *.asm.*

2. **Assembler**

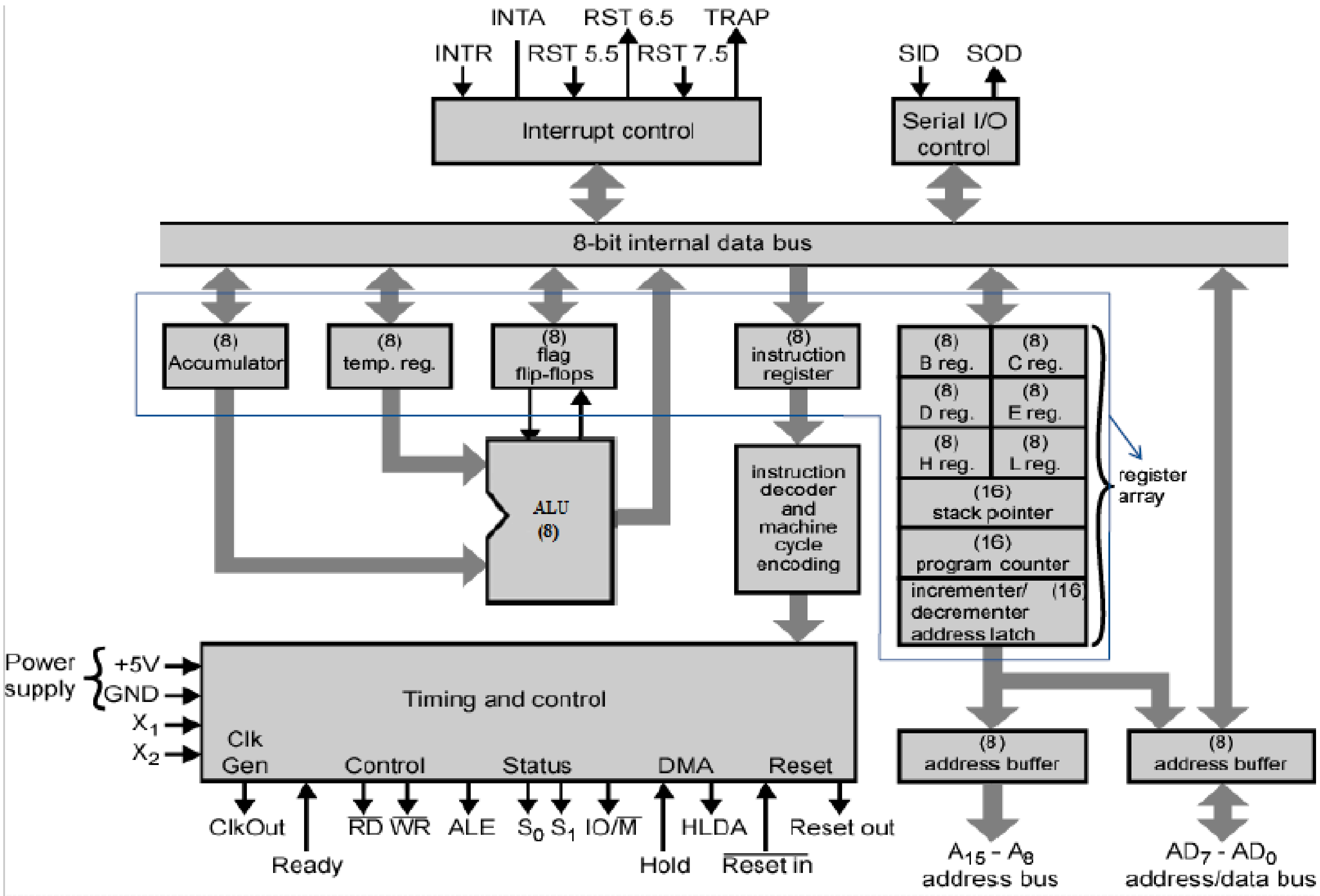
   - An assembler program is used to translate the assembly language mnemonics for instructions to corresponding binary codes. When you run the assembler, it reads the source file of your program from the disk where you have saved it after editing.

Assembly Language Program Development Tools

3. **Linker**- A linker is a program used to join several object files into one large object file.
4. **Locator** - A locator is a program used to assign the specific address of where the segments of object code are to be loaded into memory.
5. **Debugger**-A debugger is a program which allows you to load your object code program into system memory, execute the program and troubleshoot or debug it.

Features of Microprocessor-8085

8085 is developed by INTEL
8 bit microprocessor: can accept 8 bit data simultaneously
Operates on single +5V D.C. supply.
Designed using NMOS technology
6200 transistor on single chip
Operates on 3MHz clock frequency.
8-bit multiplexed address/data bus, which reduce the number of pins.
16address lines, hence it can address 2^16 = 64 K bytes of memory
It generates 8 bit I/O addresses, hence it can access 2^8 = 256 I/O ports.
5 hardware interrupts i.e. TRAP, RST6.5, RST5.5, RST4.5, and INTR



Internal Architecture (functional block diagram)of 8085

8085 Architecture…….cont…

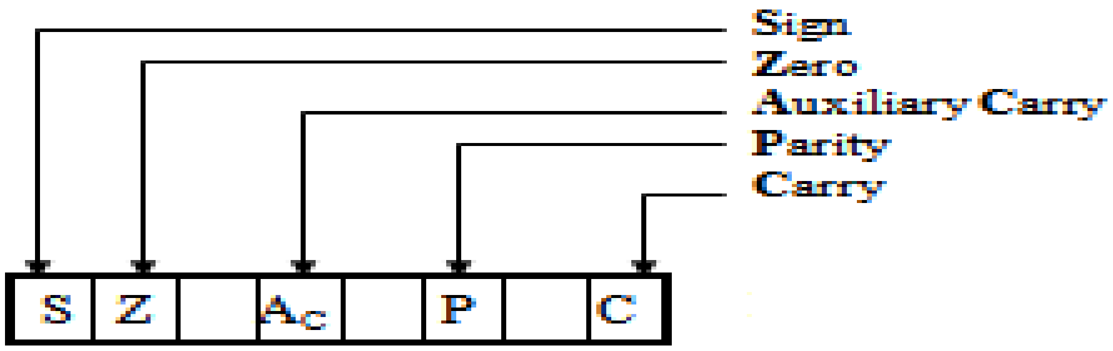
- 8085 architecture consists of following blocks:
  - Registers
  - ALU & Logical Group
  - Instruction decoder and machine cycle encoder
  - Timing and control circuitry
  - Interrupt control Group
  - Serial I/O control Group

8085 Architecture……cont…

- **Registers Array : 14 register out of which 12 are 8 bit capacity and 2 of 16 bit. Classify into 4 types**
  - **(a) General purpose register: (user accessible)**
    - B,C,D,E,H,L are 8 bit register.(can be used singly)
    - Can also be used for 16-bit register pairs- BC, DE & HL.
    - Used to store the intermediate data and result
    - **H & L can be used as a data pointer(holds memory address)**
- **(b) Special Purpose Register[A, Instruction Register and Flag]**
  - **(b.1) Accumulator (A): (user accessible)**
    - 8 bit register
    - All the ALU operations are performed with reference to the contents of Accumulator.
    - Result of an operation is stored in A.
    - Store 8 bit data during I/O transfer
  - **(b.2) Instruction Register: (user not accessible)**
    - When an instruction is fetched from memory, it is loaded in IR. Then transferred to the decoder for decoding.
    - It is not programmable and can not be accessed through any instruction.

8085 Architecture…….cont…

- **(b.3) Flag Register(F): (user accessible)**
  - ⍰ 8 bit Register
  - ⍰ Indicates the status of the ALU operation.
  - ⍰ ALU includes 5 flip flop, which are set or reset after an operation
  - according to data conditions of the result in the accumulator.



(Flag Register)

Flag Register…… cont….

Flag	Significance
C or CY (Carry)	CY is set when an arithmetic operation generates a carry out, otherwise it is 0 (reset)
P (Parity)	P= 1; if the result of an ALU operation has an even number of 1's in A; P= 0; if number of 1 is odd.
AC (Auxiliary carry)	Similar to CY, AC= 1 if there is a carry from D3 to D4 Bit AC= 0 if there is a no carry from D3 to D4 Bit (not available for user)
Z(zero)	Z = 1; if result in A is 00H 0 otherwise
S(Sign)	S=1 if D7 bit of the A is 1(indicate the result is -ive) S= 0 if D7 bit of the A is 0(indicate the result is +ive)

Flag Register…… cont….

Example

```
  10011001
+ 10101110
  01000111
```

Carry-1 (set to 1)

AC-1 (set to 1)

- **© Temporary Register[Temporary data register]**

  Internally used by the MP(user not accessible)

  - 8 bit capacity
  - Used to hold temporary data during ALU operations.

8085 Architecture …… cont….

- **(d) Pointer Register or special purpose [SP, PC]**
  - **(d.1) Stack Pointer(SP)**
    - 16 bit address which holds the address of the data present at the top of the stack memory
    - It is a reserved area of the memory in the RAM to store and retrieve the temporary information.
    - Also hold the content of PC when subroutines are used.
    - When there is a subroutine call or on an interrupt. ie. pushing the return address on a jump, and retrieving it after the operation is complete to come back to its original location.
  - **(d.3) Program Counter(PC)**
    - 6 bit address used for the execution of program Contain the address of the next instruction to be executed after fetching the instruction
    - it is automatically incremented by 1.
    - Not much use in programming, but as an indicator to user only.

8085 Architecture …… cont….

- **In addition to register MP contains some latches and buffer**
  - **Increment and decrement address latch**
    - 16 bit register
    - Used to increment or decrement the content of PC and SP
  - **Address buffer**
    - 8 bit unidirectional buffer
    - Used to drive high order address bus(A8 to A15)
    - When it is not used under such as reset, hold and halt etc this buffer is used tristate high order address bus.
  - **Data/Address buffer**
    - 8 bit bi-Directional buffer
    - Used to drive the low order address (A0 to A7) and data (D0 to D7) bus.
    - Under certain conditions such as reset, hold and halt etc this buffer is used tristate low order address bus.

8085 Architecture …… cont….

- **ALU & Logical Group: it consists ALU, Accumulator, Temporary register and Flag Register.**
  - **(a) ALU**
    - Performs arithmetic and logical operations
    - Stores result of arithmetic and logical operations in accumulator
- **(b) Accumulator**
    - General purpose register
    - Stores one of the operand before any arithmetic and logical operations and result of operation is again stored back in Accumulator
    - Store 8 bit data during I/O transfer

8085 Architecture …… cont….

(2) ALU & Logical Group…………………..cont…………………

(c) Temporary Register

- 8 bit register
- During the arithmetic and logical operations one operand is available in A and other operand is always transferred to temporary register
- For Eg.: ADD B – content of B is transferred into temporary register before actual addition

(d) Flag Register

- Five flag is connected to ALU
- After the ALU operation is performed the status of result will be stored in five flags.

8085 Architecture ...... cont...

(3) Instruction decoder and machine cycle encoder, Timing and control circuitry

(a) Instruction decoder and machine cycle encoder :

- Decodes the op-code stored in the Instruction Register (IR) and establishes the sequence of events to follow.
- Encodes it and transfer to the timing & control unit to perform the execution of the instruction.

(b) Timing and control circuitry

- works as the brain of the CPU
- For proper sequence and synchronization of all the operations of MP, this unit generates all the timing and control signals necessary for communication between microprocessor and peripherals.

8085 Architecture …… cont….

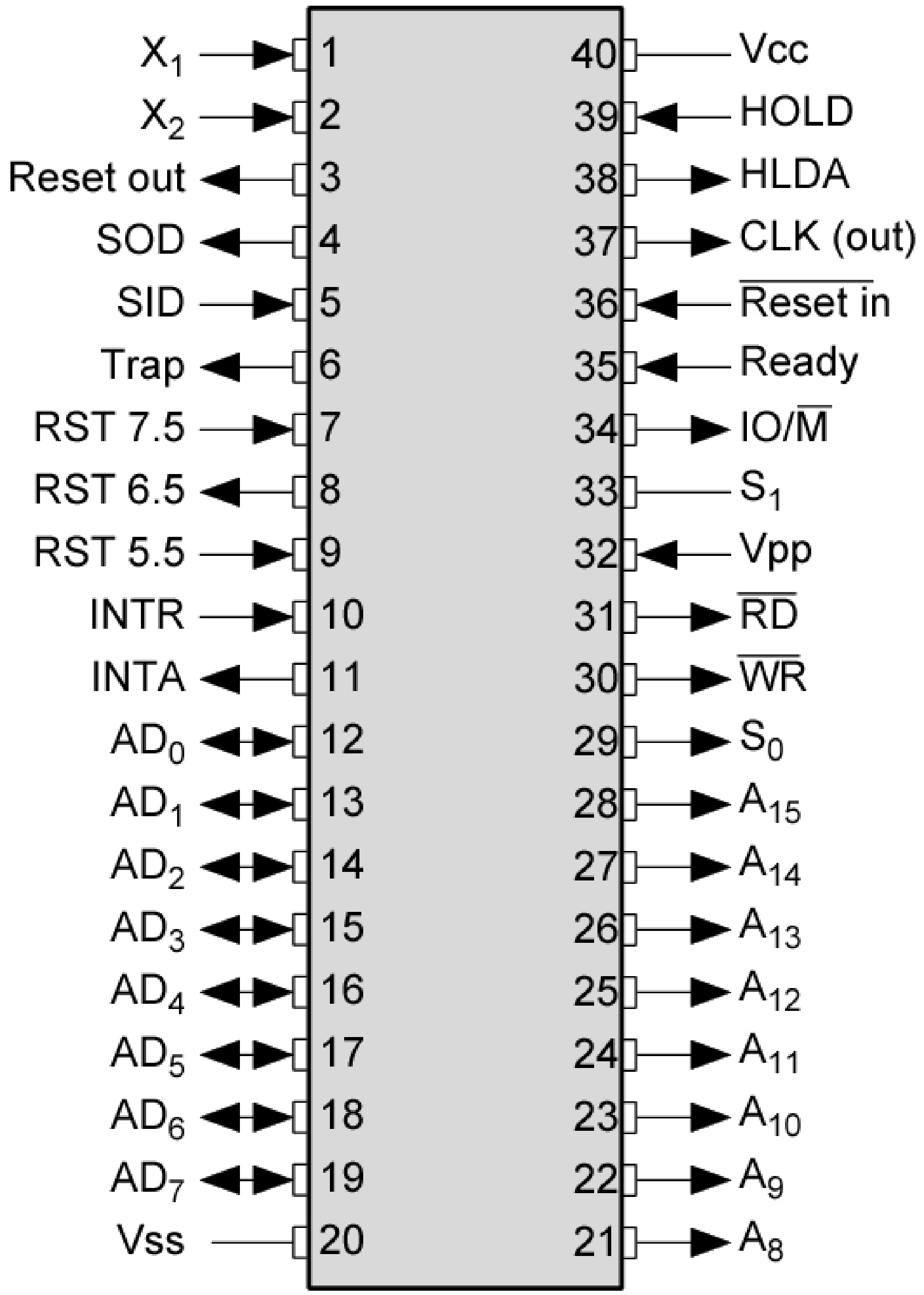
(4) Interrupt Control group

- **Interrupt:- Occurrence of an external disturbance**
- After servicing the interrupt, 8085 resumes its normal working sequence
- Transfer the control to special routines
- Five interrupts: - TRAP, RST7.5, RST6.5, RST5.5, INTR
- In response to INTR, it generates INTA signal

(5) Serial I/O control Group

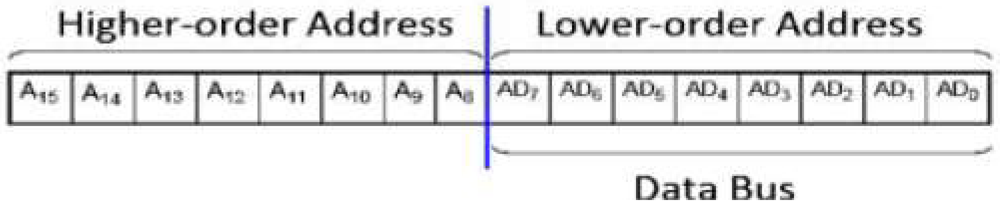
- Data transfer red on D0- D7 lines is parallel data But under some condition it is used serial data transfer
- Serial data is entered through SID(serial input data) input (received)
- Serial data is outputted on SOD(serial output data) input (send)

8085 Pin Diagram



8085 Pin Description

- The 8085 is an 8-bit general purpose microprocessor that can address 64K Byte of memory.
- It has 40 pins and uses +5V for power. It can run at a maximum frequency of 3 MHz.
- **Address Bus (Pin 21-28)**
  - 16 bit address lines A0 to A15
  - The address bus has 8 signal lines A8 – A15 which are unidirectional.
  - The other 8 address lines A0 to A7 are multiplexed (time shared) with the 8 data bits.


Higher-order Address
Lower-order Address
A_{15} A_{14} A_{13} A_{12} A_{11} A_{10} A_9 A_8
AD_7 AD_6 AD_5 AD_4 AD_3 AD_2 AD_1 AD_0
Data Bus

8085 Pin Description..Cont..

Data Bus (Pin 19-12)

- To save the number of pins lower order address pin are multiplexed with 8 bit data bus (bidirectional)
- So, the bits AD0 – AD7 are bi-directional and serve as A0 – A7 and D0 – D7 at the same time.
- During the execution of the instruction, these lines carry the address bits during the early part (T1 state), then during the late parts(T2 state) of the execution, they carry the 8 data bits.

8085 Pin Description..Cont..

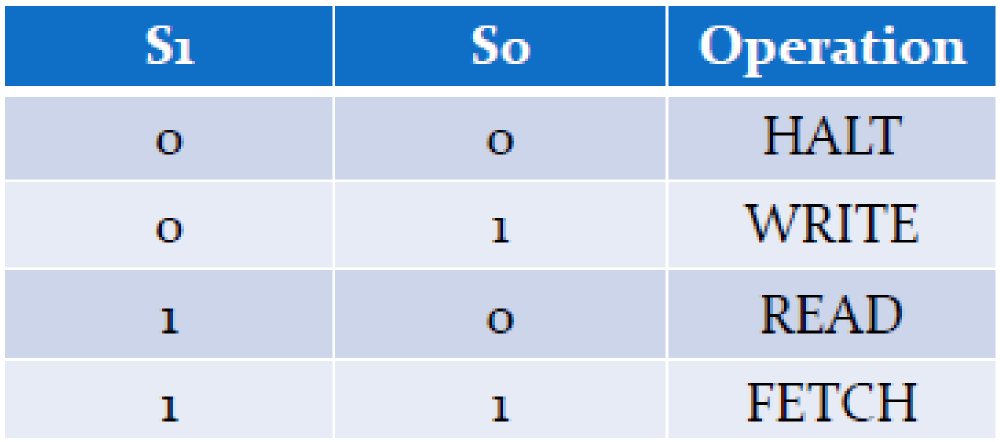
Status Pins – ALE, S1, S0

- **1. ALE(Address Latch Enable): (Pin 30)**
  - Used to demultiplexed the address and data bus
  - +ive going pulse generated when a new operation is started by uP.
  - ALE = 1 when the AD0 – AD7 lines have an address
  - ALE = 0 When it is low it indicates that the contents are data.
  - This signal can be used to enable a latch to save the address bits from the AD lines.

8085 Pin Description..Cont..

2. S1 and S0 (Status Signal): (Pin 33 and 29)

– Status signals to specify the kind of operation being performed.



S1	S0	Operation
0	0	HALT
0	1	WRITE
1	0	READ
1	1	FETCH

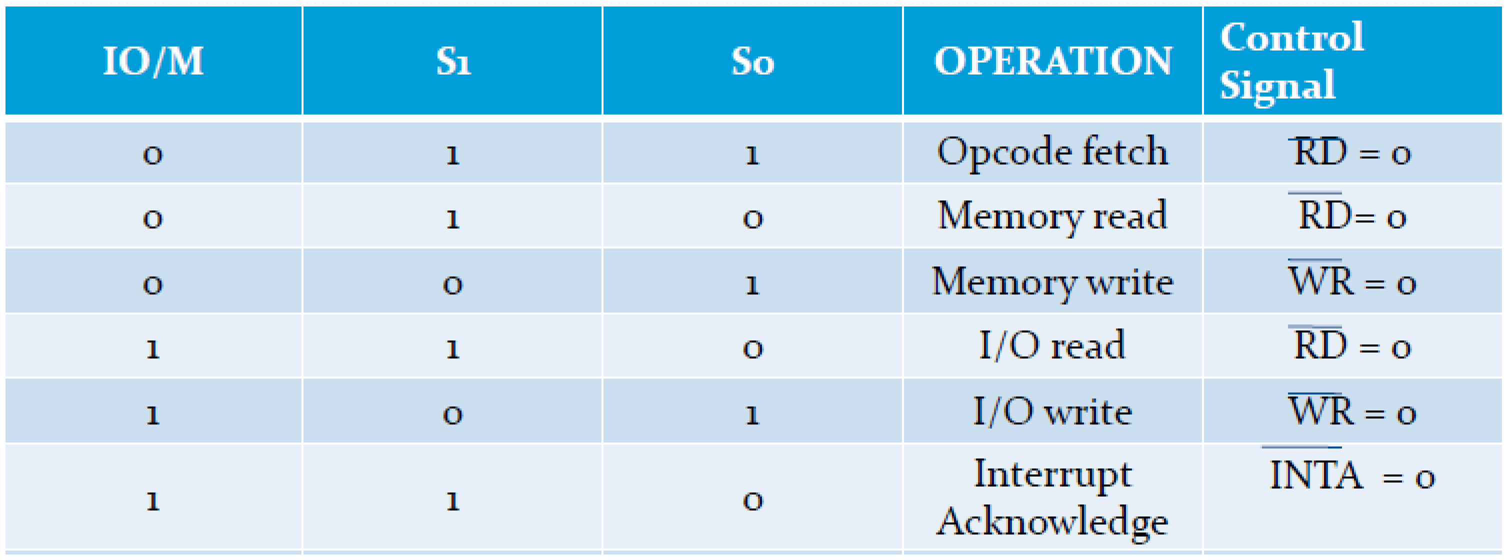
8085 Pin Description..Cont..

Control Pins – RD, WR, IO/M(active low)

- **1. RD: Read(Active low) (Pin 32)**
  - Read Memory or I/O device
  - Indicated that data is to be read either from memory or I/P device and data bus is ready for accepting data from the memory or I/O device.
- **2. WR:Write(Active low) (Pin 31)**
  - Write Memory or I/O device
  - Indicated that data on the data bus are to be written into selected memory or I/P device.
- **3. IO/M: (Input Output/Memory-Active low) (Pin 34)**
  - Signal specifies that the read/write operation relates to whether memory or I/O device.
  - When (IO/M=1) the address on the address bus is for I/O device
  - When (IO/M=0) the address on the address bus is for memory

8085 Pin Description..Cont..

- **Control and status Signals**
  - When S0, S1 is combined with IO/M(active low), we get status of machine cycle

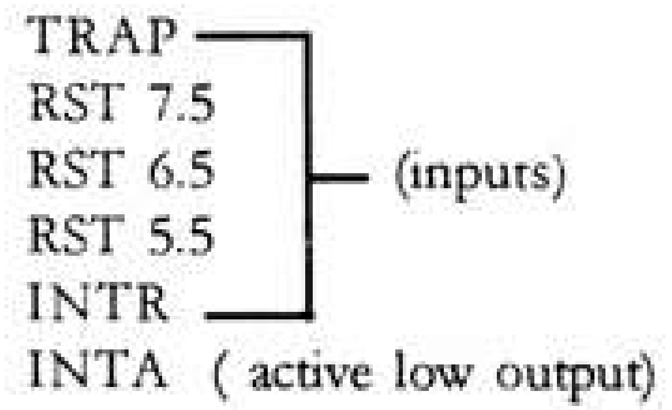


IO/M	S1	S0	OPERATION	Control Signal
0	1	1	Opcode fetch	$\overline{RD} = 0$
0	1	0	Memory read	$\overline{RD} = 0$
0	0	1	Memory write	$\overline{WR} = 0$
1	1	0	I/O read	$\overline{RD} = 0$
1	0	1	I/O write	$\overline{WR} = 0$
1	1	0	Interrupt Acknowledge	$\overline{INTA} = 0$

8085 Pin Description..Cont..

Interrupts

- They are the signals initiated by an external device to request the microprocessor to do a particular task or work.
- There are five hardware interrupts called, **(Pin 6-11)**
- On receipt of an interrupt, the microprocessor acknowledges the interrupt by the active low INTA (Interrupt Acknowledge) signal.



8085 Pin Description..Cont..

- **Power supply and Clock Signal**
  - Vcc (Pin 40) : single +5 volt power supply
  - Vss (Pin 20) : Ground
  - There are 3 important pins in this group.
- **X0 and X1 :((Pin 1-2)**
  - Crystal or R/C network or LC network connections to set the frequency of internal clock generator.
  - The frequency is internally divided by two.
  - Since the basic operating timing frequency is 3 MHz, a 6 MHz crystal is connected to the X0 and X1 pins.
- **CLK (output): (Pin 37)**
- Clock Output is used as the system clock for peripheral and devices interfaced with the microprocessor.

8085 Pin Description..Cont..

- **Reset Signals**
  - **Reset In (input, active low) (Pin 36)**
    - This signal is used to reset the microprocessor.
    - The program counter inside the microprocessor is set to zero(0000H)
- **Reset Out (Output, Active High) (Pin 3)**
    - It indicates MP is being reset.
    - Used to reset all the connected devices when the microprocessor is reset.

8085 Pin Description..Cont..

DMA Request Signals

- **HOLD (Pin 38)**
  - This signal indicates that another device is requesting the use of address and data bus.
  - So it relinquish the use of buses as soon as the current machine cycle is completed.
  - MP regains the bus after the removal of a HOLD signal
- **HLDA (Pin 39)**
  - On receipt of HOLD signal, the MP acknowledges the request by sending out HLDA signal and leaves out the control of the buses.
  - After the HLDA signal the DMA controller starts the direct transfer of data.
  - After the removal of HOLD request HLDA goes low.

8085 Pin Description..Cont..

- **Serial I/O Signals**
  - These pins are used for serial data communication
- **SID (input) Serial input data (Pin 4)**
  - It is a data line for serial input
  - Used to accept serial data bit by bit from external device
  - The data on this line is loaded into accumulator bit 7 whenever a RIM instruction is executed.
- **SOD (output) Serial output data (Pin 5)**
  - It is a data line for serial output
  - Used to transmit serial data bit by bit to the external device
  - The 7th bit of the accumulator is outputted on SOD line when SIM instruction is executed.

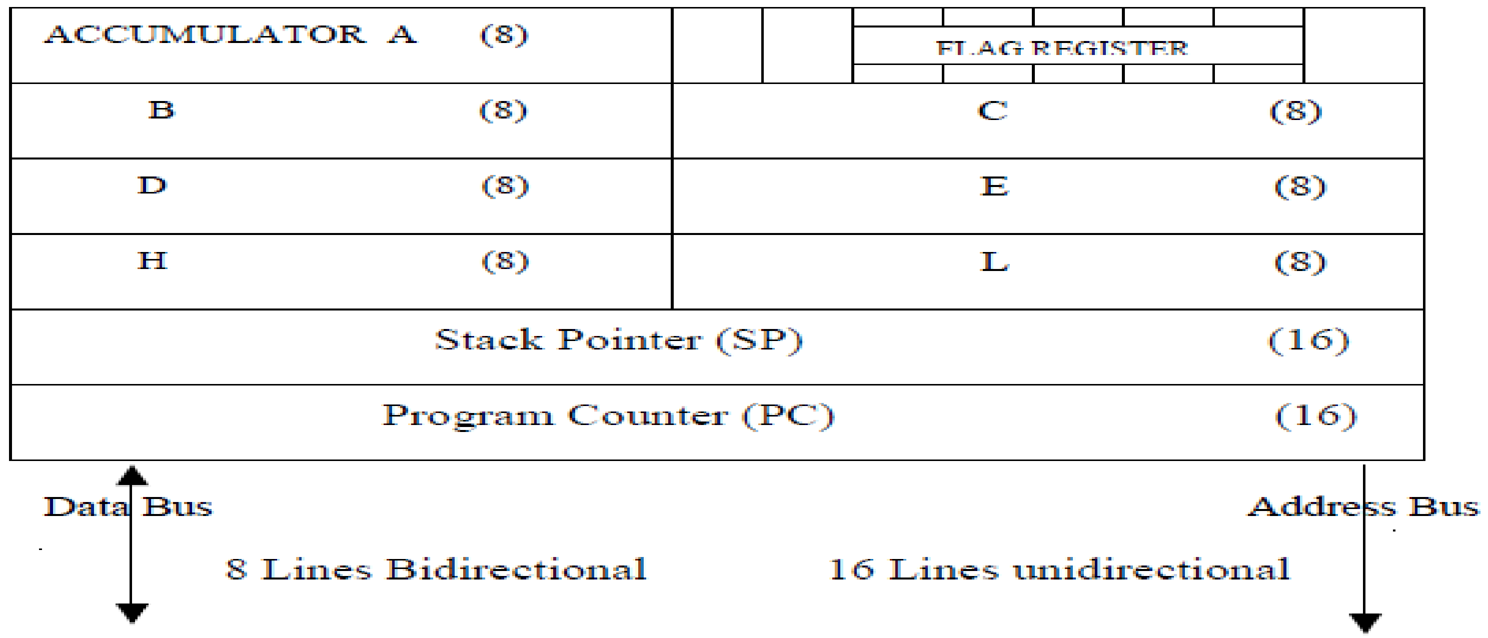
8085 Pin Description..Cont..

Ready (input) (Pin 35)

- Memory and I/O devices will have slower response compared to microprocessors.
- Before completing the present job such a slow peripheral may not be able to handle further data or control signal from CPU.
- The processor sets the READY signal after completing the present job to access the data.
- It synchronize slower peripheral to the processor.
- The microprocessor enters into WAIT state while the READY pin is disabled.

8085 Programming register

- The register which are programmable and available for the use are six general, purpose register, A, F, PC, SP.



Fetching & Execution Cycles

- **Fetching Cycles**
  The fetch cycle takes the instruction required from memory, stores it in the instruction register, and increment the program counter by one so that it points to the next instruction.
  Execute cycle
  The actual actions which occur during the execute cycle of an instruction. Depend on both the instruction itself and the addressing mode specified to be used to access the data that may be required.

Instruction Cycle

The necessary steps that a CPU carries out to fetch an instruction and necessary data from the memory and execute it, constitute an instruction cycle.

$$IC = FC + EC$$

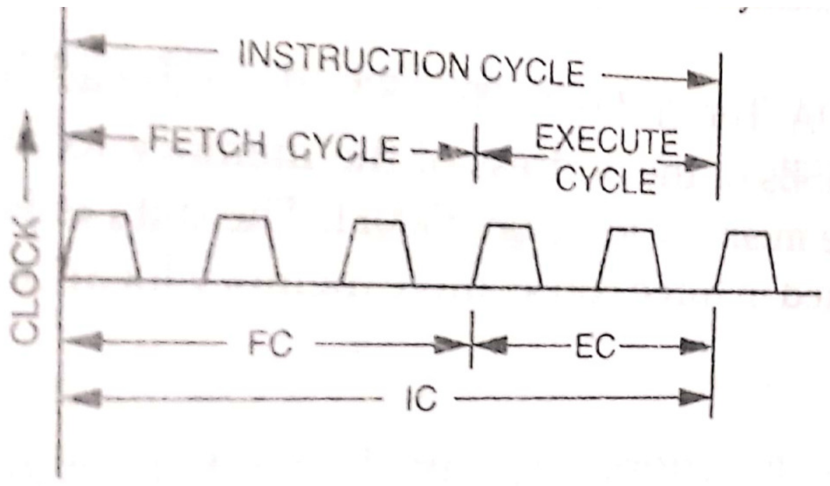


Fig. 3.4 Instruction cycle showing FC, EC, IC

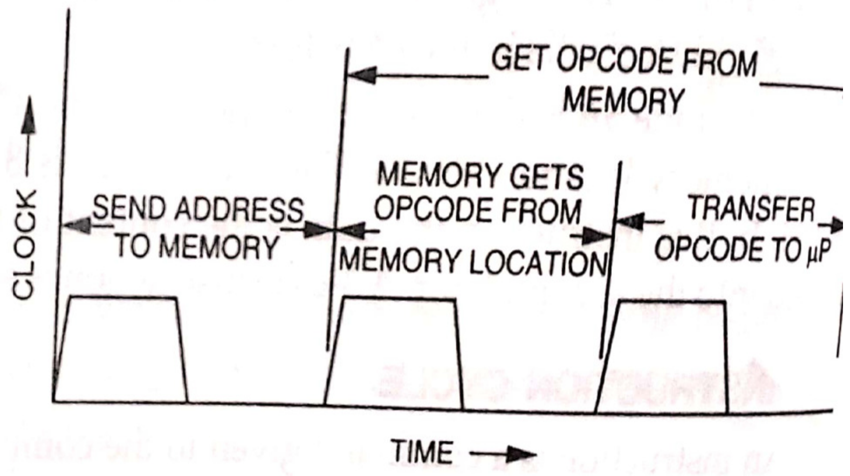


Fig. 3.5. Fetch Cycle

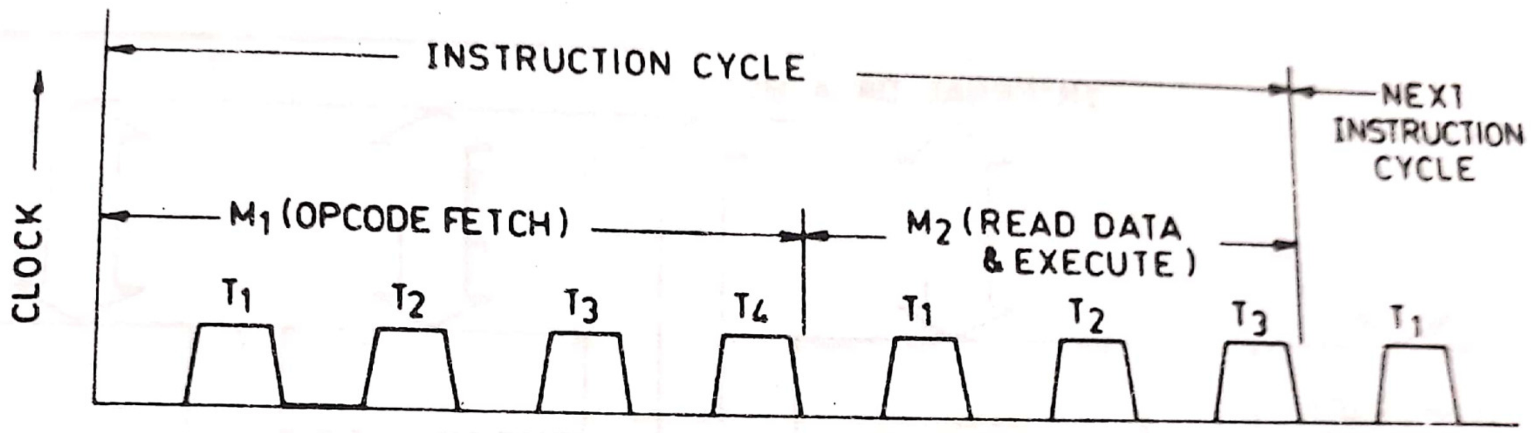
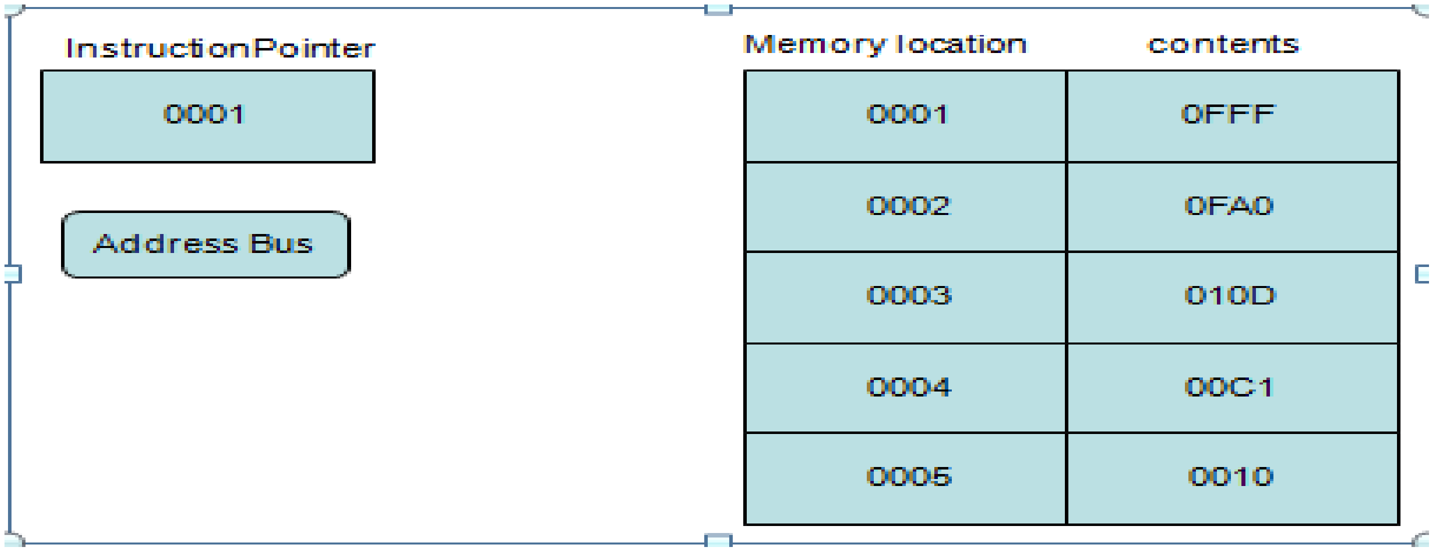


Fig. 3.6. Typical Instruction Cycle

Fetching an instruction

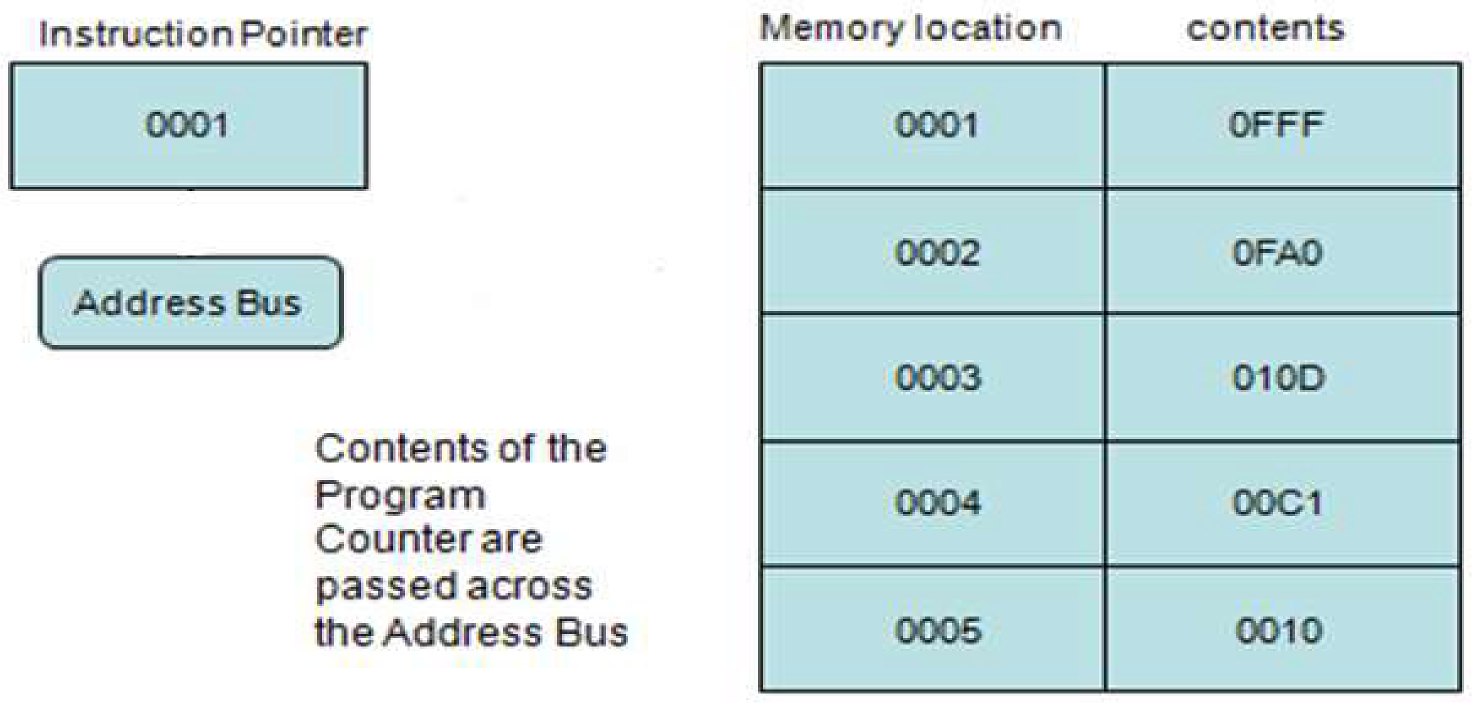
Step 1: Instruction pointer (program counter) hold the address of the next instruction to be fetch.



Memory location	contents
0001	0FFF
0002	0FA0
0003	010D
0004	00C1
0005	0010

Fetching an instruction....Cont....

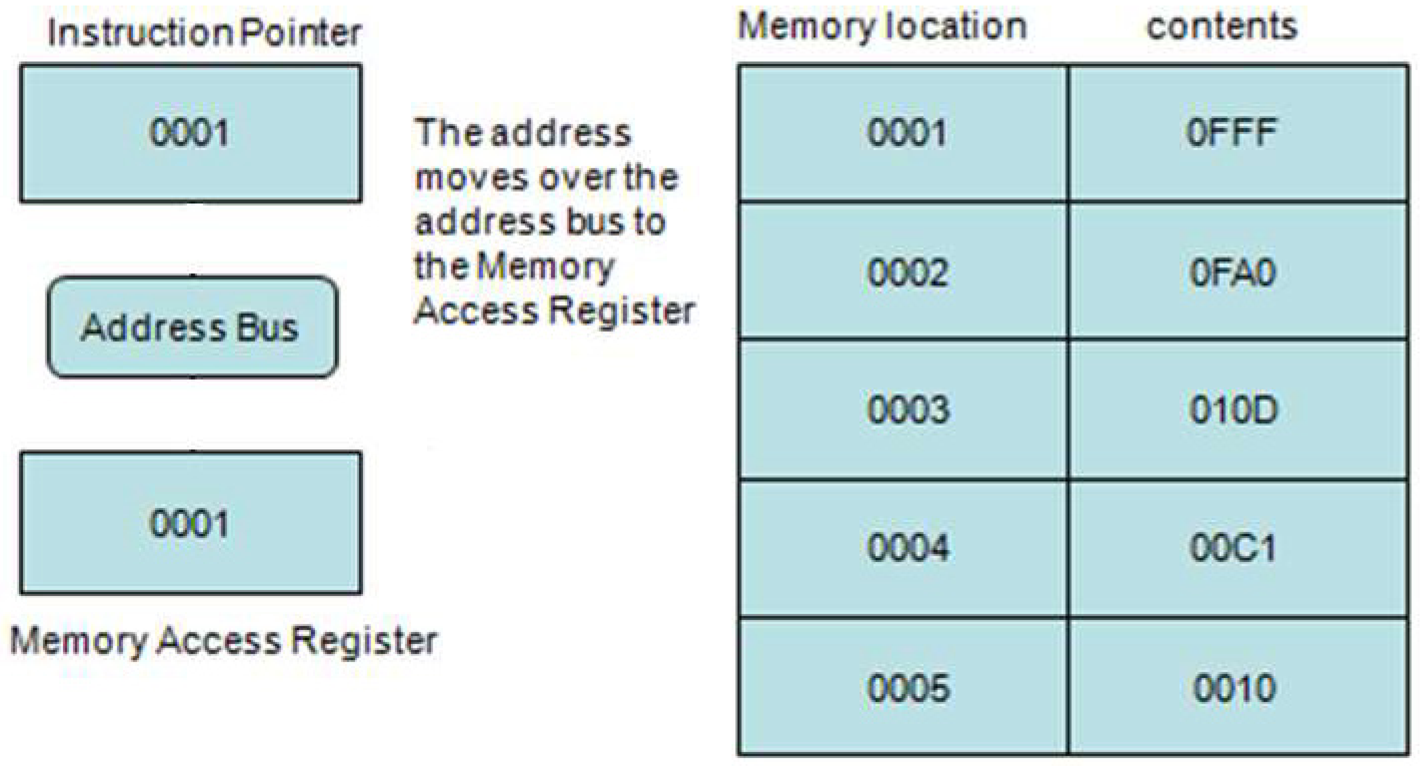
- **Step 2**


Instruction Pointer
0001
Address Bus
Contents of the Program Counter are passed across the Address Bus

Memory location	contents
0001	0FFF
0002	0FA0
0003	010D
0004	00C1
0005	0010

Fetching an instruction....Cont....

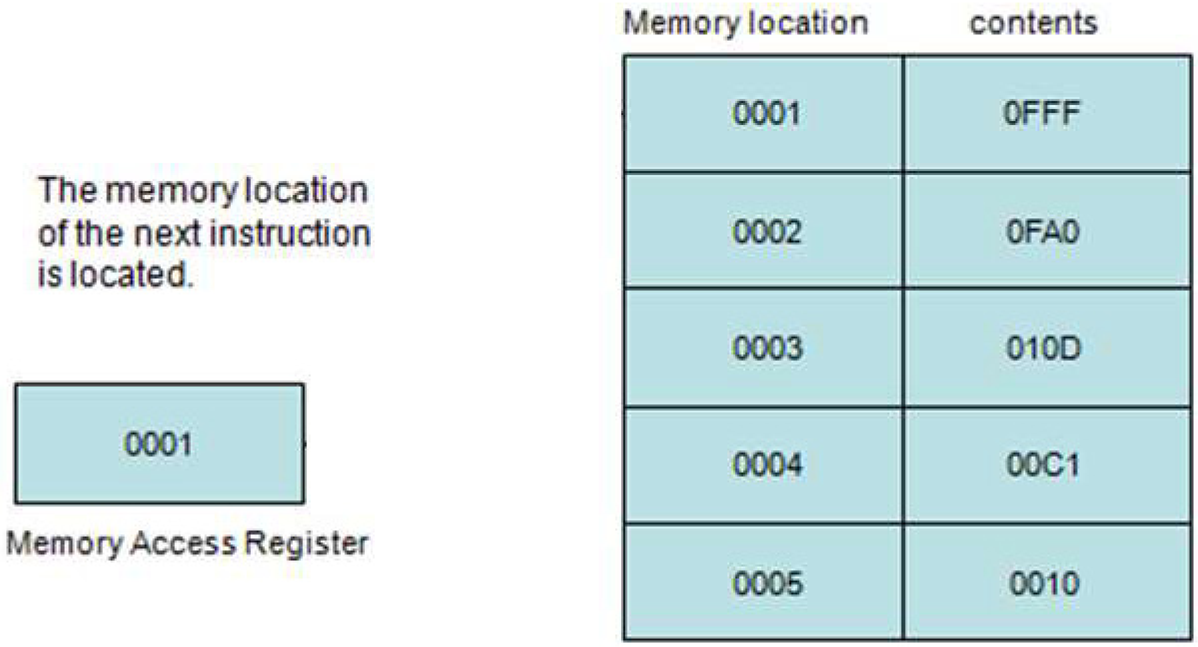
- **Step 3**



Memory location	contents
0001	0FFF
0002	0FA0
0003	010D
0004	00C1
0005	0010

Fetching an instruction….Cont….

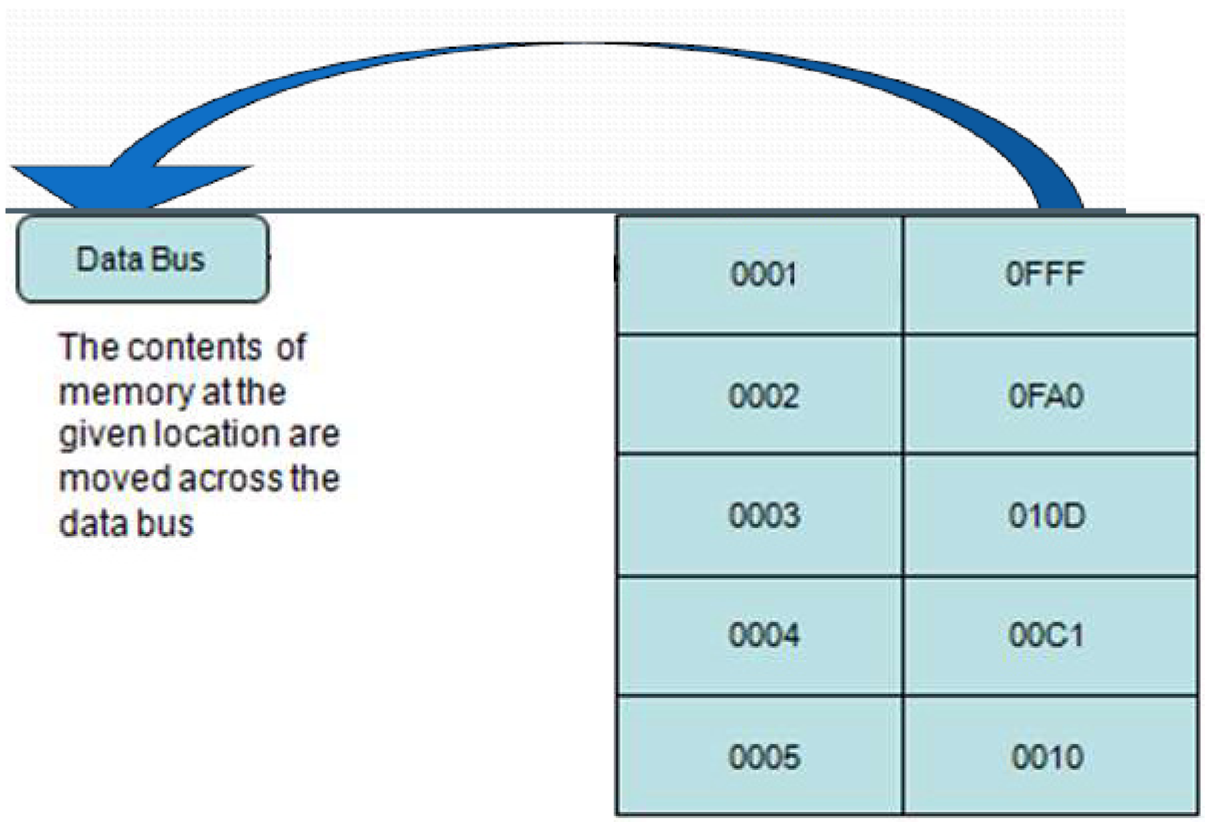
- **Step 4**



Memory location	contents
0001	0FFF
0002	0FA0
0003	010D
0004	00C1
0005	0010

Fetching an instruction….Cont….

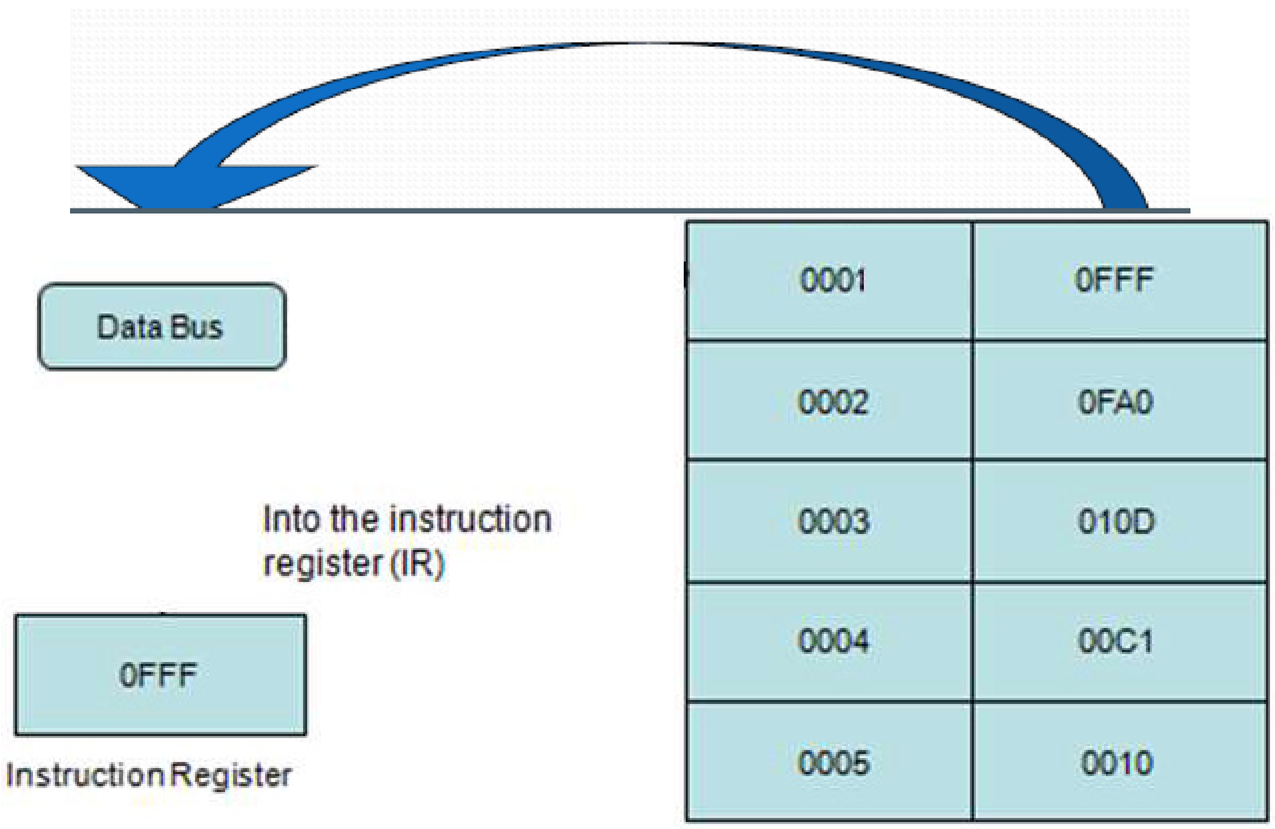
- **Step 5**


Data Bus
The contents of memory at the given location are moved across the data bus

0001	0FFF
0002	0FA0
0003	010D
0004	00C1
0005	0010

Fetching an instruction....Cont....

- **Step 6**



Data Bus
Into the instruction register (IR)
0FFF
Instruction Register
0001
0FFF
0002
0FA0
0003
010D
0004
00C1
0005
0010