SUNBEAM

Institute of Information Technology

PreCAT

MicroProcessor

CONTENTS

INTRODUCTION

Microprocessor is an electronic chip that functions as the central processing unit (CPU) of a computer. In other words, we can call microprocessor as the heart of any computer system. Some call the microprocessors as the brain of the computers. The microprocessor based systems with limited resources are called as microcomputers.

Programs are written using mnemonics called the assembly level language and then they are converted into binary machine level language.

Basic Concept

Chip

A chip or an integrated circuit is a small, thin piece of silicon with the required circuit and transistors etched on it to perform a particular function.

Bit

A bit means a single binary digit. Also, the bit is the fundamental storage unit of computer memory. In binary, bit can have only two values, 0 or 1.

Word

A number of bits grouped together for processing is called as word. 16-bit binary number is called a word in a 16-bit processor

Byte ->An 8-bit word is referred to as a byte.

Nibble ->A 4-bit word is referred to as a nibble.

Kilobyte ->A collection of 1024 bytes is called a kilobyte. Megabyte ->A collection of 1024 Kbytes is called a megabyte.

Bus

The bus in a microprocessor system refers to a group of wires or signals having a common functionality. There are three types of buses

- Data Bus
- Address Bus
- Control Bus

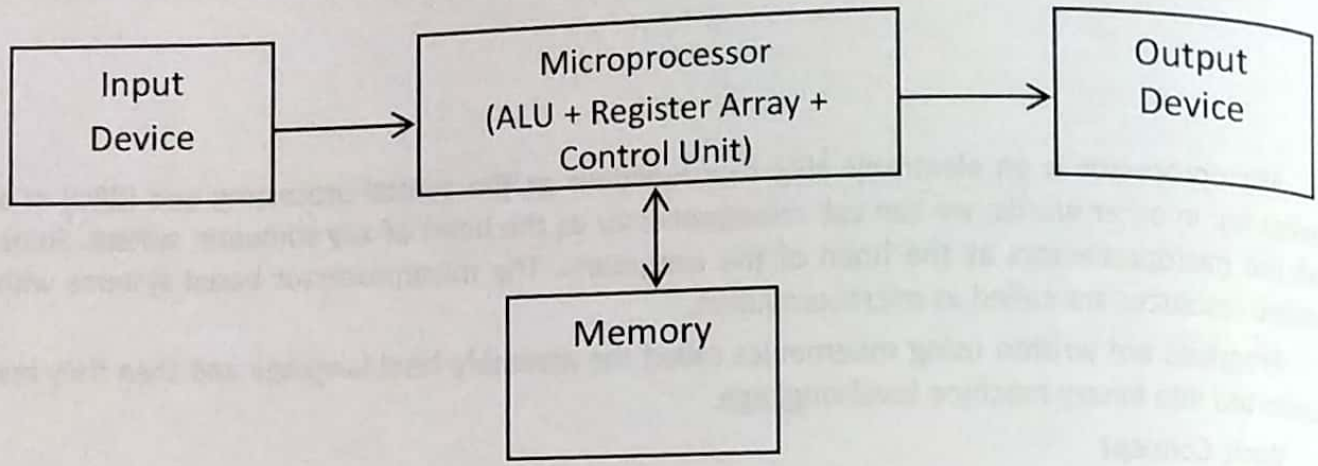
Microprocessor

Microprocessor is a controlling unit of a micro-computer, fabricated on a small chip capable of performing Arithmetic Logical Unit (ALU) operations and communicating with the other devices connected to it.

Microcontroller

A Microcontroller is a programmable digital processor with necessary peripherals. Microcontroller may be called computer on chip since it has basic features of microprocessor with internal ROM, RAM, Parallel and serial ports, timer, interrupt schedule circuitry in single chip. Microcontrollers are also called embedded controllers.

Block Diagram of a Basic Microcomputer



Microprocessor

Microprocessor means a small (micro) IC that can process data i.e. perform arithmetic and logical operations. Microprocessor reads the instructions from memory and executes it line by line.

Microprocessor consists of an ALU, register array, and a control unit. ALU performs arithmetical and logical operations on the data received from the memory or an input device. Register array consists of registers identified by letters like B, C, D, E, H, L and accumulator. The control unit controls the flow of data and instructions within the computer

Input device

The devices that are used for giving data to the microcomputer system are called as input devices. Usually keyboard and mouse are the input devices through which data and instructions are given to computer.

Output device

The devices that are used for getting data out from the microcomputer system are called as output devices. A display screen, printer and displays are the common output devices.

Memory

It stores both the instructions to be executed and the data involved. It consists of RAMs (random-access memories),PROM,EPROM,static RAM, dynamic RAM and ROMS (read-only memories)etc.

The microprocessor follows a sequence: Fetch, Decode, and then Execute.

Initially, the instructions are stored in the memory in a sequential order. The microprocessor fetches those instructions from the memory, then decodes it and executes those instructions till stop instruction is reached. Later, it sends the result in binary to the output port. Between these processes, the register stores the temporarily data and ALU performs the computing functions.

Classification of Microprocessor

Based on size of data bus

- 4 - bit microprocessor
- 8 - bit microprocessor
- 16 - bit microprocessor
- 32 - bit microprocessor

Based on architecture

- Reduced Instruction Set Computer / Computing (RISC) processors
- Complex Instruction Set Computer / Computing (CISC) processors
- Special Processor

RISC Processor

RISC stands for Reduced Instruction Set Computer. It is designed to reduce the execution time by simplifying the instruction set of the computer. Using RISC processors, each instruction requires only one clock cycle to execute results in uniform execution time. This reduces the efficiency as there are more lines of code, hence more RAM is needed to store the instructions. The compiler also has to work more to convert high-level language instructions into machine code.

RISC processors are Power PC, ARM (Advance RISC machine), AVR (Advance Virtual RISC machine), SUN SPARC, 8051, MIPS.

Architecture of RISC

RISC microprocessor architecture uses highly-optimized set of instructions. It is used in portable devices like Apple iPod due to its power efficiency.

Characteristics of RISC

The major characteristics of a RISC processor are as follows –

1. It consists of simple instructions.
2. It supports various data-type formats.
3. It utilizes simple addressing modes and fixed length instructions for pipelining.
4. It supports register to use in any context.
5. One cycle execution time.
6. "LOAD" and "STORE" instructions are used to access the memory location.
7. It consists of larger number of registers.
8. It consists of less number of transistors.

CISC Processor

CISC stands for Complex Instruction Set Computer / Computing. It is designed to minimize the number of instructions per program, ignoring the number of cycles per instruction. The emphasis is on building complex instructions directly into the hardware.

The compiler has to do very little work to translate a high-level language into assembly level language/machine code because the length of the code is relatively short, so very little RAM is required to store the instructions.

Some of the CISC Processors are

Intel 8080,8085, 8086, 80286, 80386, Pentium etc.

Architecture of CISC

Its architecture is designed to decrease the memory cost because more storage is needed in larger programs resulting in higher memory cost. To resolve this, the number of instructions per program can be reduced by embedding the number of operations in a single instruction.

Characteristics of CISC

1. Variety of addressing modes.
2. Larger number of instructions.

3. Variable length of instruction formats.
4. Several cycles may be required to execute one instruction.
5. Instruction-decoding logic is complex.
6. One instruction is required to support multiple addressing modes.

Special Processors

These are the processors which are designed for some special purposes. Few of the special processors are briefly discussed –

Coprocessor

A coprocessor is a specially designed microprocessor, which can handle its particular function many times faster than the ordinary microprocessor.

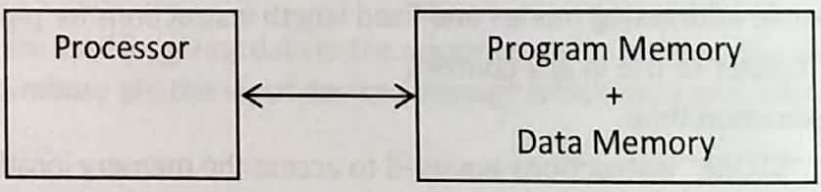
For example – Math Coprocessor.

Some Intel math-coprocessors are –

- 8087-used with 8086
- 80287-used with 80286
- 80387-used with 80386

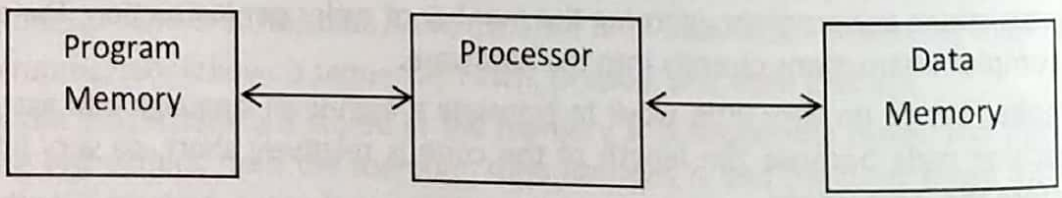
Von Neumann Architecture:

The computer has single storage memory for storing data as well as program to be executed. It works on CICS architecture.



Harvard Architecture:

The computer has two separate memories for storing data and program. Processor can complete an instruction in one cycle. It works on RISC architecture.



Objective Questions:

Q.1. Which are the processors based on RISC?

a) SPARC
b) 80386
c) MC68030
d) MC68020

Q.2. Which of the following processors execute its instruction in a single cycle?

a) 8086
b) 8088
c) 8087
d) 8051

Q.3. How is memory accessed in RISC architecture?

a) load and store instruction
b) opcode instruction
c) memory instruction
d) bus instruction

Q.4. Which of the following processors uses Harvard architecture?

a) AVR
b) 80386
c) 80286
d) 8086

Q.5. Which of the following statements are true for von Neumann architecture?

a) shared bus between the program memory and data memory
b) separate bus between the program memory and data memory
c) external bus for program memory and data memory
d) external bus for data memory only

Q.6. Which of the following processors has CISC architecture?

a) AVR
b) ARM
c) Power PC
d) Zilog Z80

Q.7. Which one of the following offers CPUs as integrated memory or peripheral interfaces?

a) Microcontroller
b) Microprocessor
c) Embedded system
d) Memory system

Q.8. Which of the following offers external chips for memory and peripheral interface circuits?

a) Microcontroller
b) Microprocessor
c) Peripheral system
d) Embedded system

Q.9. What is CISC?

a) Computing instruction set complex
b) Complex instruction set computing
c) Complementary instruction set computing
d) Complex instruction set complementary

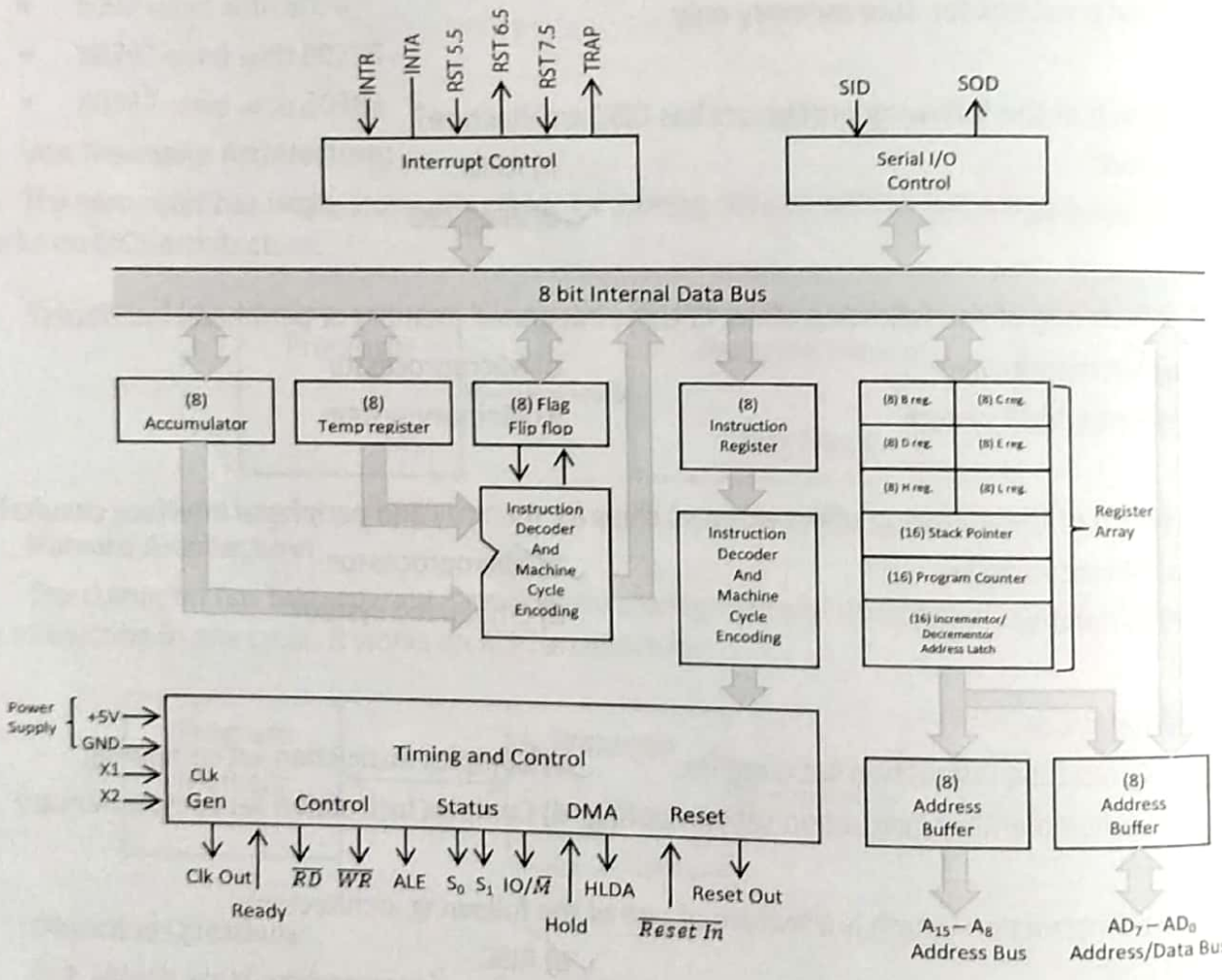
10) Fixed instruction length is a feature of one of the following architecture.

a) CISC
b) RISC
c) X86
d) None of the above

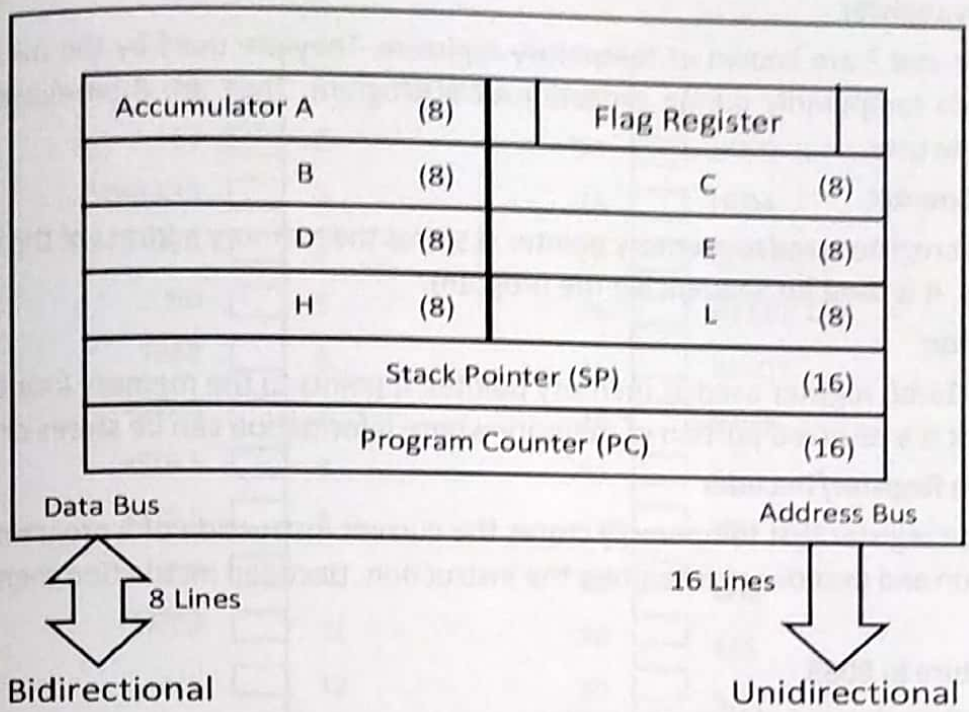
MICROPROCESSOR 8085

8085 microprocessors are an 8-bit microprocessor designed by Intel in 1977 using NMOS technology. It is a 40 pin IC. It works on +5V power supply. It can run at a maximum frequency of 3MHz. Its data bus width is 8-bit and address bus width is 16-bit, thus it can address 2^{16} = 64 KB of memory.

8085 Architecture



Registers: -These are general purposes registers. Microprocessor consists of 6 general purpose registers each of 8-bit named as B, C, D, E, H and L. It carries the 8-bits data. These are used only during the execution of the instructions. These registers can also be used to carry the 16 bits data by making the registers pair. The valid register pairs available are BC, DE and HL. These registers are programmed by user.



Accumulator

It is an 8-bit register which is used to perform arithmetical and logical operation such as Addition, Subtraction, AND, OR etc. The results of the arithmetic and logical operations are stored in the accumulator.

Flag Register

It consists of five flip flop which changes its status according to the result stored in an accumulator. It is also known as status registers

There are five flip-flops in the flag register are as follows:

1. Sign(S)
2. Zero(Z)
3. Auxiliary carry(AC)
4. Parity(P)
5. Carry(C)

The bit position of the flip flop in flag register is:

D7	D6	D5	D4	D3	D2	D1	D0
S	Z		AC		P		CY

For example, after an addition of two numbers, if the result in the accumulator is larger than 8 - bit, CY flag is set to 1. When an arithmetic operation results in zero, Z flag is set to 1. The S flag is just a copy of the bit D7 of the accumulator. A negative number has a 1 in bit D7 and a positive number has a 0 in bit D7. The AC flag is set to 1, when a carry result from bit D3 and passes to bit D4. The P flag is set to 1, when the result in accumulator contains even number of 1s.

Temporary register

Register W and Z are known as temporary registers. They are used by the microprocessor for storing the data temporarily during execution of a program. They are 8-bitregisters and are not accessible to the user.

Program Counter

It is a 16-bit register used as memory pointer. It stores the memory address of the next instruction to be executed. It is used for sequencing the program.

Stack Pointer

It is also a 16-bit register used as memory pointer. It points to the memory location called stack. Generally, stack is a reserved portion of memory where information can be stores or taken back.

Instruction Register/Decoder

It is an 8-bit register that temporarily stores the current instruction of a program. Decoder then takes instruction and decodes or interprets the instruction. Decoded instruction then passed to next stage.

Bus Structure in 8085

There are three buses in Microprocessor:

1. Address Bus 2. Data Bus 3. Control Bus

Data Bus

The data bus width is 8-bit i.e. 2^8 = 255 combination of binary digits, identified as D0 – D7. Data bus is bidirectional since it carries data in binary form between microprocessor and other external units such as memory. It is used to transmit data. As it is 8-bit wide then largest number is 11111111.

Address Bus

8085 microprocessor contains 16-bit address bus i.e. 2^{16}= 65536= 64KB memory location it can access, identified as A0 - A15. Address bus is unidirectional. The address bus carries addresses from microprocessor to the memory or other devices. The higher order address lines are A8 – A15 and the lower order lines (A0 – A7) are multiplexed with the eight bits data lines (D0 – D7).

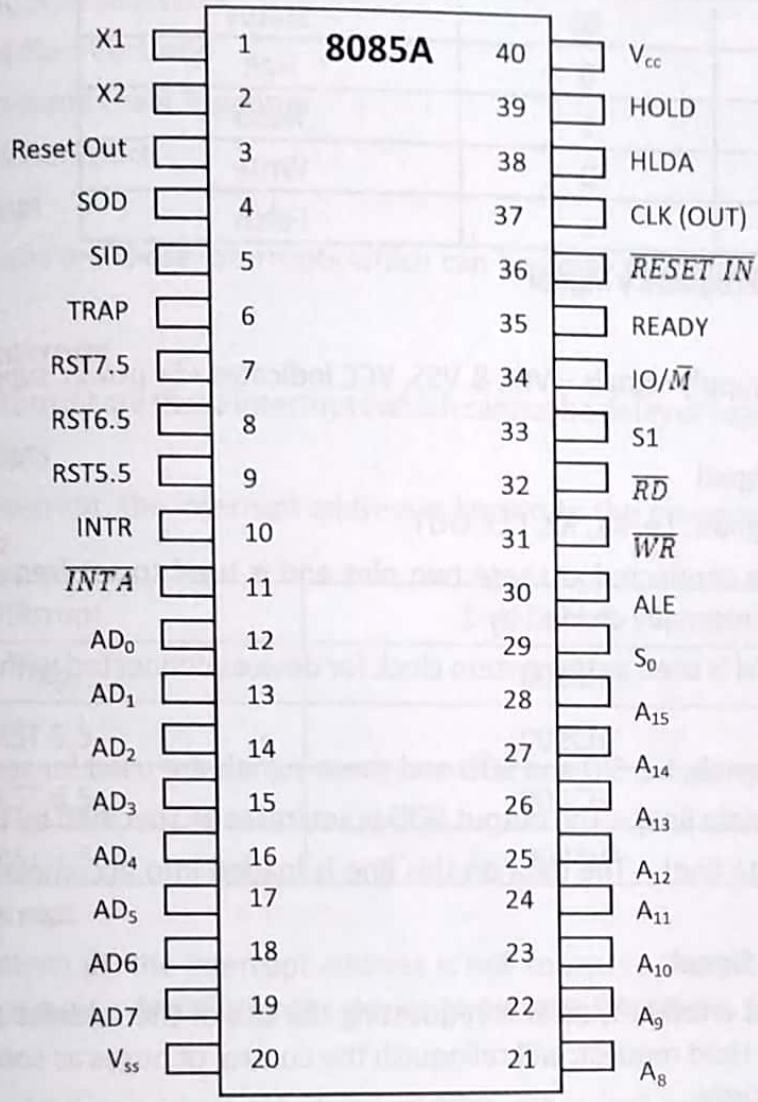
Control Bus

Control bus are various lines which have specific functions for coordinating and controlling microprocessor operations. The control bus carries control signals partly unidirectional and partly bidirectional.

8085 PIN DESCRIPTION

8085 is a 40pin IC. Works on +5v power supply with 3MHz frequency. 8085 signals are classified into six groups:

- Address bus
- Data bus
- Control & status signals
- Power supply and frequency signals
- Externally initiated signals
- Serial I/O signals



PIN DIAGRAM of 8085

Address bus

A_{15}-A_8, it carries the most significant 8-bits of memory or IO address.

Data bus

AD_7-AD_0, it carries the least significant 8-bit address and data bus.

Control and status signals

There are 3 control signals as RD, WR & ALE and 3 status signals as IO/M, S0 & S1.

- **$\overline{RD}$** –The Read signal indicates that data are being read from the selected I/O or memory device that are available on the data bus.
- **$\overline{WR}$** – The Write signal indicates that data on the data bus are to be written into a selected memory or I/O location.
- **ALE** – Address Latch Enable is a signal that is provided. when the pulse goes high an address appears on the AD0 – AD7 lines, after which it becomes 0indicates data.
- **IO/$\overline{M}$** - It is a signal that distinguished between a memory operation and an I/O operation. When IO/= 0 it is a memory operation and IO/= 1 it is an I/O operation.

- **S1 & S0** - These are status signals used to specify the type of operation being performed

S1	S0	Status
0	0	Halt
0	1	Read
1	0	Write
1	1	Fetch

1. **Power Supply and Frequency Signal**
2. **Power Supply**

There are 2 power supply signals – VCC & VSS. VCC indicates +5v power supply and VSS indicates ground signal.

Clock(frequency) Signal

There are 3 clock signals, i.e. X1, X2, CLK OUT.

X1, X2 – A crystal is connected at these two pins and is used to set frequency of the internal clock. This frequency is internally divided by 2.

CLK OUT – This signal is used as the system clock for devices connected with the microprocessor.

Serial I/O signals

There are 2 serial signals, i.e. SID and SOD and these signals are used for serial communication.

SOD (Serial output data line) – The output SOD is set/reset as specified by the SIM instruction.

SID (Serial input data line) – The data on this line is loaded into accumulator whenever a RIM instruction is executed.

Externally Initiated Signal

HOLD: indicates that another Master is requesting the use of the Address and Data Buses. The CPU, upon receiving the Hold request. will relinquish the control of buses as soon as the completion of the current machine cycle.

$\overline{\textbf{HLDA}}$: Hold Acknowledge. This signal indicates that the CPU has received the Hold request and that it will relinquish the buses in the next clock cycle. HLDA goes low after the Hold request is removed.

Ready - The Ready signal is used to synchronize slower peripherals with the microprocessor. If the signal at READY pin is low the microprocessor enters into a wait state.

$\overline{\textbf{RESET}}$ - This is an active low signal. When it is activated the microprocessor suspends all the internal operations and the program counter is cleared.

RESET OUT - This signal indicates that the microprocessor is being reset. It can be used to reset other devices also.

$\overline{\textbf{INTA}}$ - Interrupt Acknowledge. This signal is used to acknowledge an interrupt.

Interrupt

Interrupt means interrupting the normal execution of the microprocessor. When microprocessor receives interrupt signal, it discontinues whatever it was executing. It starts executing new program indicated by the interrupt signal. Interrupt signals are generated by external peripheral devices to request the microprocessor to perform a task. There are 5 interrupt signals, i.e. TRAP, RST 7.5, RST 6.5, RST 5.5 and INTR.

Classification of Interrupts

- Maskable and Non-Maskable
- Vectored and Non-Vectored
- Edge Triggered and Level Triggered
- Software and Hardware

Maskable Interrupt

Maskable interrupts are those interrupts which can be delay or rejected. For example: RST7.5, RST6.5, RST5.5, INTR.

Non Maskable Interrupt

Non Maskable interrupt are those interrupts which cannot be delay or rejected. For example: Trap.

Vectored Interrupts

In this type of interrupt, the interrupt address is known to the processor. For example: RST7.5, RST6.5, RST5.5, TRAP.

Interrupt	Address
Trap	0024H
RST 5.5	002CH
RST 6.5	003CH
RST 7.5	0034H

Non-Vector interrupt

In this type of interrupt, the interrupt address is not known to the processor so, the interrupt address needs to be supply externally by the device to perform interrupts. For example: INTR.

Edge Triggered

The interrupts which are triggered at rising edge are called edge triggered interrupts. For example: RST 7.5 is an edge triggered interrupt.

Level Triggered

The interrupts which are triggered at falling edge are called level triggered interrupts. RST 5.5, RST 6.5, INTR.

Priority Based Interrupts

Whenever there exists a simultaneous request at two or more pins, then the pin with higher priority is selected by the microprocessor.

Priority of interrupts:

Interrupt	Priorities
Trap	1
RST 7.5	2
RST 6.5	3
RST 5.5	4
INTR	5

TRAP

It is a non-Maskable interrupt. It has the highest priority. It cannot be disabled. It is both edge as well as level triggered.

TRAP is usually used for power failure and emergency shutoff.

RST 7.5

It is a Maskable interrupt. It has the second highest priority. It is positive edge triggered only. The internal flip-flop is triggered by the rising edge. The flip-flop remains high until it is cleared by RESET IN.

RST 6.5

It is a Maskable interrupt. It has the third highest priority. It is level triggered only. The pin has to be held high for a specific period of time. RST 6.5 can be enabled by EI instruction. It can be disabled by DI instruction.

RST 5.5

It is a Maskable interrupt. It has the fourth highest priority. It is also level triggered. The pin has to be held high for a specific period of time.

INTR

It is a Maskable interrupt. It has the lowest priority. It is also level triggered.

Software Interrupts

8085 instructions set includes eight software interrupt instructions called Restart (RST) instructions. These are one byte instructions that make the processor execute a subroutine at predefined locations. Instructions and their vector addresses are given below.

RST 0 -- 0000H

RST 1 -- 0008H

RST 2 -- 0010H

RST 3 -- 0018H

RST 4 -- 0020H

RST 5 -- 0028H

RST 6 -- 0030H

RST 7 -- 0038H

Hardware Interrupts

8085 have five hardware interrupts – INTR, RST 5.5, RST 6.5, RST 7.5 and TRAP

Interrupt Handling Procedure:

The following sequence of operations takes place when an interrupt signal is recognized:

1. Save the PC content and information about current state (flags, registers etc.) in the stack.
2. Jump to the IVT(Interrupt Vector Table) table which has ISR(Interrupt Service Routine) address. It will jump to the ISR address and load PC with the beginning address of an ISR and start to execute it.
3. Finish ISR when the return instruction (IRET – Interrupt return) is executed.
4. Return to the point in the interrupted program where execution was interrupted.

Instruction of 8085

The 8085 microprocessor has 246 bit patterns amounting to 74 different instructions. These 74 different instructions are therefore called its instruction set.

Instruction Set Classification

An instruction is a binary pattern designed inside a microprocessor to perform a specific function. The entire group of instructions, called the instruction set, determines what functions the microprocessor can perform.

These instructions can be classified into the following five functional categories:

1. Data transfer (copy) operations
2. Arithmetic operations
3. Logical operations
4. Branching operations
5. Machine-control operations.

1. Data Transfer (Copy) Operations

This group of instructions copy data from a location called a source to another location called a destination, without modifying the contents of the source. The term data transfer is used for copying function. However, the term transfer is misleading, it creates the impression that the contents of the source are destroyed, in fact, the contents are retained without any modification.

The various types of data transfer (copy) are listed below with examples of each type:

Type	Example
1. Between Registers.	1. Copy the contents of the register B into register D.
2. Specific data byte to a register or a memory location.	2. Load register B with the data byte 32H.
3. Between a memory location and a register.	3. From a memory location 2000H to register B.
4. Between an I/O device and the accumulator.	4.From an input keyboard to the accumulator.

2. Arithmetic Operations

These instructions perform arithmetic operations such as addition, subtraction, increment, and decrement.

Addition - Any 8-bit number, or the contents of a register or the contents of a memory location can be added to the contents of the accumulator and the sum is stored in the accumulator. No two others 8-bit registers can be added directly (e.g., the contents of register B cannot be added directly to the contents of the register C). The instruction DAD is an exception, it adds 16-bit data directly in register pairs.

Subtraction - Any 8-bit number, or the contents of a register, or the contents of a memory location can be subtracted from the contents of the accumulator and the results stored in the accumulator. The subtraction is performed in 2's complement, and the results if negative, are expressed in 2's complement. No two other registers can be subtracted directly.

Increment/Decrement - The 8-bit contents of a register or a memory location can be incremented

or decrement by 1. Similarly, the 16-bit contents of a register pair (such as BC) can be incremented or decrement by 1.

3. Logical Operations

These instructions perform various logical operations with the contents of the accumulator.

AND, OR Exclusive-OR - Any 8-bit number, or the contents of a register, or of a memory location can be logically ANDed, Ored, or Exclusive-ORed with the contents of the accumulator. The results are stored in the accumulator.

Rotate- Each bit in the accumulator can be shifted either left or right to the next position.

Compare- Any 8-bit number, or the contents of a register, or a memory location can be compared for equality, greater than, or less than, with the contents of the accumulator.

Complement - The contents of the accumulator can be complemented. All 0s are replaced by 1s and all 1s are replaced by 0s.

4. Branching Operations

This group of instructions alters the sequence of program execution either conditionally or unconditionally.

Jump - Conditional jumps are an important aspect of the decision-making process in the programming. These instructions test for a certain condition (e.g., Zero or Carry flag) and alter the program sequence when the condition is met. In addition, the instruction set includes an instruction called unconditional jump.

Call, Return, and Restart - These instructions change the sequence of a program either by calling a subroutine or returning from a subroutine. The conditional Call and Return instructions also can test condition flags.

5. Machine Control Operations

These instructions control machine functions such as Halt, Interrupt, or do nothing.

Instruction Format

An instruction is a command to the microprocessor to perform a given task on a specified data. Each instruction has two parts: one is task to be performed, called the operation code (opcode), and the second is the data to be operated on, called the operand. The operand (or data) can be specified in various ways. It may include 8-bit (or 16-bit) data, an internal register, a memory location, or 8-bit (or 16-bit) address. In some instructions, the operand is implicit (which doesn't have operand).

Instruction word size

The 8085-instruction set is classified into the following three groups according to word size:

- 1-byte instructions
- 2-byte instructions
- 3-byte instructions

One-byte instructions: Instruction having one byte in machine code. Examples shown in table.

Opcode	Operand	Machine code/Hex code
MOV	A, B	78
ADD	M	86

Two-byte instructions: Instruction having two bytes in machine code. Examples shown in table.

Opcode	Operand	Machine code /Hex Code	Byte description
MVI	A,7FH	3E	First byte
		7F	Second Byte
ADI	0FH	C6	First byte
		0F	Second Byte

Three-byte instructions: Instruction having three bytes in machine code. Examples shown in table.

Opcode	Operand	Machine code /Hex Code	Byte description
JMP	4050H	C3	First byte
		50	Second Byte
		40	Third Byte
LDA	6050H	3A	First byte
		50	Second Byte
		60	Third Byte

Addressing Modes in Instructions

The process of specifying the data to be operated on by the instruction is called addressing. The various formats for specifying operands are called addressing modes.

The 8085 has the following five types of addressing:

I. Immediate addressing
II. Direct addressing
III. Register direct addressing
IV. Register Indirect addressing
V. Implicit / Implied addressing

Immediate Addressing: In this mode, the operand given in the instruction - a byte or word – transfers to the destination register or memory location.

Example: MVI A, 30H

- 30H is copied into the register A

MVI B, 40H

- 40H is copied into the register B

Direct Addressing: Data is directly copied from the given address to the register. The memory location address is given in the instruction.

Example: LDA 850FH

- This instruction is used to load the content of memory address 850FH in the accumulator.

Register Direct Addressing: Register direct addressing transfer a copy of a byte or word from source register to destination register. Data is copied from one register to another register.

Example: MOV B, C

- It copies the content of register C to register B

MOV A, C

- the content of C is copied into the register A

Register Indirect Addressing: Indirect addressing transfers a byte or word between a register and a memory location. The data is transferred from the address pointed by the data in a register to other register.

Example: LXI H,1020H

MOV A,M ; indirect addressing mode

- Here M points to(contains) address 1020H, at his address operand is stored. By executing this instruction, the content of 1020H is loaded into accumulator A.

Implicit / Implied Addressing: This mode doesn't require any operand. The data is specified by opcode itself.

Ex: CMA

- The instruction complements the content of the accumulator. No specific data or operand is mentioned in the instruction.

Data Transfer instruction

1. **MVI** - Move immediate

   MVI Rd, data -Transfers 'data' value into register Rd.

   Ex: MVI A,50H

2. **MOV** – Move from one register to another register

   MOV Rd,Rs -Here Rd is destination register, Rs is source register. Transfers data in Rs into Rd. Rs content will not be changed.

   Ex: MOV B,A

3. **IN**

   IN port_address -This instruction reads data from IO device connected at specified port address and loads accumulator with that data.

   Ex: IN 10H

4. **OUT**

   OUT port_address -This instruction sends content of accumulator to IO device connected at specified port address.

   Ex: OUT 05H

5. **LXI**- Load Immediate data into register pair

   LXI Rp,data -This instruction loads 16-bit immediate data into specified register pair(Rp)

   LXI B, 16-bit data; loads BC with 16-bit data

   LXI D, 16-bit data; loads DE with 16-bit data

   LXI H,16-bit data; loads HL with 16-bit data

   LXI SP,16-bit data; loads stack pointer with immediate data

   Ex: LXI H,2050H

   LXI SP,1090H

6. **LDAX**-Load accumulator indirect

   LDAX B/D - This instruction loads Accumulator with memory location pointed by content of B or D

   LDAX B - loads accumulator with content of memory location pointed by BC pair

   LDAX D - loads accumulator with content of memory location pointed by DE pair

7. **LDA**-Load accumulator direct

LDA 16-bit address - This instruction loads accumulator with the content of memory location specified in the instruction.

Ex: LDA 2005H -This instruction loads accumulator with content of [2005H]

8. **STAX**-Store accumulator indirect

STAX B/D -This instruction stores the content of accumulator into memory location pointed by B or D register pair.

Example: STAX B

stores accumulator content in memory location pointed by BC pair

STAX D

stores accumulator content in memory location pointed by DE pair

9. **STA** -Store accumulator direct

STA 16-bit address -This instruction stores accumulator content in 16-bit address location specified in the instruction.

Ex: STA 1020H

stores accumulator content in 1020H address

Arithmetic Instructions

1. **ADD**-Add Register to Accumulator

ADD R; A=A+R

ADD M; A=A+[HL]

This instruction performs addition of content of specified register or memory location pointed by register pair HL and stores the result in accumulator.

Example: ADD B; where A=A+B

2. **ADI**- Add immediate data

ADI data - This instruction adds 8-bit "data" to accumulator and places the result in accumulator.

Example: ADI 50H

Add the data byte 50H to contents of accumulator.

3. **SUB**- subtract register from accumulator

SUB R; A=A-R

SUB M; A=A-[HL]

This instruction subtracts content of register R or content of memory location pointed by HL register pair from accumulator and stores the result in accumulator.

Ex: SUB B

4. **SUI**- subtract immediate data

SUI data - This instruction subtracts 8-bit "data" from accumulator and stores the result in accumulator.

Ex: SUI 34H

5. **INR**- increments register by 1

This instruction operates on 8-bit data.

INR R; increments register R content by 1.

INR M; increments content of memory location pointed by HL register pair by 1.

Example: INR B

6. **DCR** – decrements register by 1

This instruction operates on 8-bit data.

DCR R; decrements register R content by 1

DCR M; decrements content of memory location pointed by HL register pair by 1

Example: DCR B

7. **INX** - increments 16-bit data

This instruction increments 16-bit data by 1

INX Rp; increments register pair Rp

INX B; increments register pair BC

INX D; increments register pair DE

INX H; increments register pair HL

INX SP; increments stack pointer SP by 1

8. **DCX**- decrements register pair by 1

This instruction decrements 16-bit data by 1

DCX Rp; decrements register pair Rp

DCX B; decrements register pair BC

DCX D; decrements register pair DE

DCX H; decrements register pair HL

DCX SP; decrements stack pointer SP by 1

Logical instructions

1. **ANA** - logical AND with accumulator

ANA R -This instruction performs bit wise AND operation between content of specified register and the accumulator and stores the result in accumulator.

Example : ANA B

2. **ANI** -logical AND immediate data with accumulator

ANI 8-bit data -This instruction performs bit wise AND operation between 8-bit immediate data & accumulator content and stores the result in accumulator

Example: ANI 30H

3. **ORA** -logical OR with accumulator

ORA R -This instruction performs bit wise OR operation between content of accumulator and specified register and stores result in accumulator.

Example: ORA C

4. **ORI** -logical OR with immediate data

ORI 8-bit data -This instruction performs bit wise OR operation between content of accumulator and 8-bit immediate data specified in the instruction and stores theresult in accumulator.

Example: ORI 39H

5. **XRA**-logical exclusive OR operation with accumulator

XRA R -This instruction performs bit wise exclusive OR operation between content of accumulator and specified register in the instruction and stores the result in accumulator.

Example: XRA A

6. **XRI**-Logical exclusive OR with 8-bit data

XRI 8-bit data -This instruction performs bit wise logical exclusive OR between immediate data and accumulator and stores the result in accumulator.

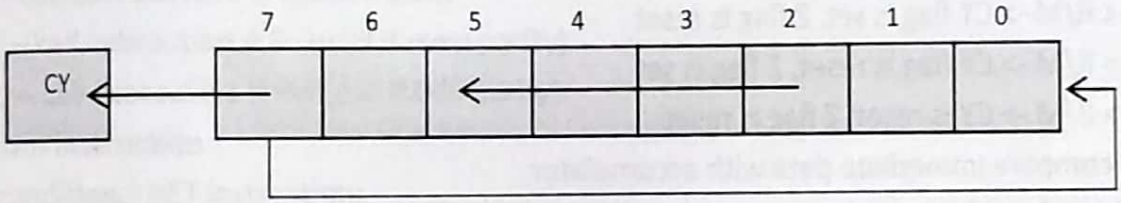
Example: XRI 52H

7. **CMA**-complement accumulator

This instruction performs bit wise inversion of content of accumulator and result will be stored in accumulator.

8. **RLC** -rotate accumulator left

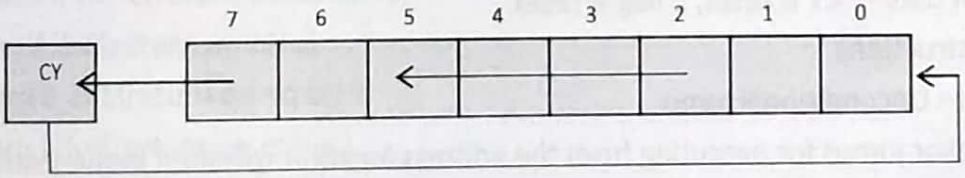
This instruction rotates left the content of accumulator by one bit. The MSB of accumulator is shifted out and copied into LSB. Copy of MSB is kept in carry flag.



9. **RAL**-rotate accumulator left through carry

This instruction rotates left the content of accumulator along with carry by one bit.

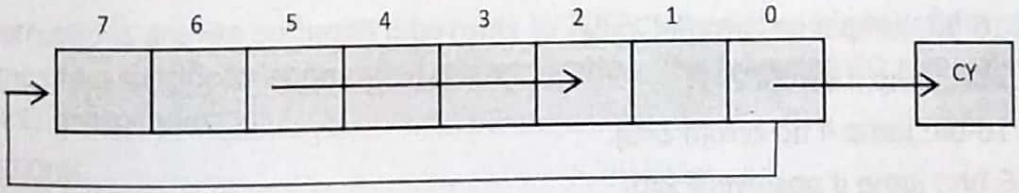
This instruction shifts the MSB of accumulator out and copies it into carry flag and content of carry flag is stored in LSB.



10. **RRC**- rotate accumulator right

This instruction rotates the content of accumulator right by one bit.

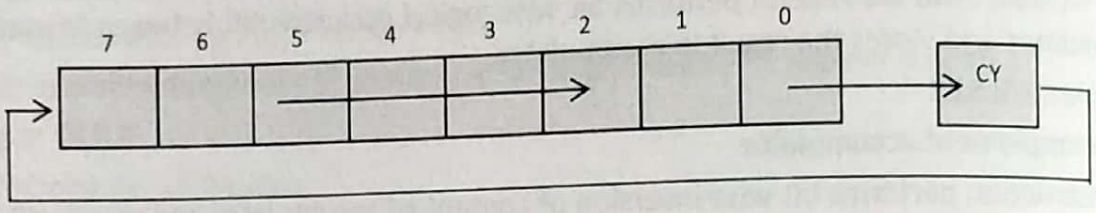
This instruction shifts the LSB of accumulator out and copies that into MSB. Copy of LSB is also stored in the carry flag



11. **RAR** - rotate accumulator right through carry

This instruction rotates the content of accumulator through carry right by one bit.

This instruction shifts the LSB of accumulator out and copies it into carry and content of carry is stored in the MSB.



12. **CMP**- compare with accumulator

CMP R; compare the content of specified register R with accumulator

CMP M; compare content of memory location pointed by HL register pair with accumulator.

Depending on the content of R or M carry flag and zero flag are affected. This in turn subtracts content of R/M from accumulator but result is not stored anywhere, only flags are affected

ACC < R/M -> CY flag is set, Z flag is reset

ACC = R/M -> CY flag is reset, Z flag is set

ACC > R/M -> CY is reset, Z flag is reset

13. **CPI** - compare immediate data with accumulator

CPI 8-bit data This instruction compares 8-bit data with the content of accumulator(this in turn subtracts 8-bit data from accumulator but result is not stored anywhere, only flags are affected). Depending on the content Carry flag and Zero flag are affected.

ACC < 8-bit data -> CY flag is set, Z flag is reset

ACC = 8-bit data -> CY flag is reset, Z flag is set

ACC > 8-bit data -> CY is reset, Z flag is reset

Branch instructions

1. **JMP 16-bit**– Unconditional jump

The Processor jumps for executing from the address location specified in the instruction.

Example: JMP 2050H

Conditional jump instructions:

This instruction jumps to the specified location in the instruction depending on the certain conditions indicated by flags.

1. JC 16-bit;jump if carry(if CY=1)
2. JNC 16-bit; jump if no carry(if CY=0)
3. JZ 16-bit;jump if zero(if Z=1)
4. JNZ 16-bit; jump if no zero(if Z=0)
5. JP 16-bit ; jump if positive(if S=0)
6. JM 16-bit ; jump if minus(if S=1)
7. JPE 16-bit ; jump on even parity(if P=1)
8. JPO 16-bit ; jump on odd parity(if P=0)

2. **CALL16-bit** - Unconditional CALL

This instruction causes processor to start executing from the specified location in the instruction.

Subroutines can be implemented using CALL instruction. Subroutines are a way for code reuse. When it is required to perform certain operations again and again than those operations can be written separately as a subroutine. Whenever those operations are required simply call the subroutine.

Example: CALL 6050H

Conditional CALL instruction:

This instruction calls the subroutine depending on the certain conditions indicated by flags.

CC – call subroutine if CY=1(Call if carry)

CNC – call subroutine if CY=0(call if no carry)

CZ – call subroutine if Z=1(call if zero)

CNZ – call subroutine if Z=0(call if no zero)

CM – call subroutine if S=1(call if minus)

CP – call subroutine if S=0(call if plus)

CPE – call subroutine if P=1(call if even parity)

CPO – call subroutine if P=0(call if odd parity)

3. **Return Instruction**

Unconditional RET instruction:

This instruction is used to return from the subroutine.

Conditional RET instruction:

This instruction returns from the subroutine depending certain conditions indicated by flags.

RC – return if CY=1(Return on carry)

RNC – return if CY=0(return on no carry)

RZ – return if Z=0(return on zero)

RNZ - return if Z=1(return on no zero)

RM – return if S=1(return on minus)

RP – return if S=0(return on plus)

RPE – return if P=1(return on even parity)

RPO – return if P=0(return on odd parity)

4. **Restart Instructions**

RST n -> 'n' value -- Vector location

These instructions are like software interrupts to 8085. When these instructions are executed processor jumps to a specific location called restart location. The following list gives restart location for different RST instructions.

RST 0 -- 0000H

RST 1 -- 0008H

RST 2 -- 0010H

RST 3 -- 0018H

RST 4 -- 0020H

RST 5 -- 0028H

RST 6 -- 0030H

RST 7 -- 0038H

To get the vector location 'n' value is multiplied by 8 and the result is converted to hexadecimal notation. For Example: RST 3 instruction, multiply 3*8 = 24.24 in hexadecimal notation is 18H. So, vector address is 0018H.

Stack related instructions

PUSH

PUSH Rp - This instruction stores the content of register pair into stack. Stack is Last in first out data structure.

Example:

PUSH B; push BC pair on to stack

PUSH D; push DE pair on to stack

PUSH H; push HL pair on to stack

PUSH PSW; push PSW (Accumulator + Flag register) on to stack

When this instruction is executed by the processor first it decrements SP by 1 and stores higher byte of the specified register pair. Then, it again decrements SP by 1 and stores lower byte of the specified register pair. SP always points to top of stack.

8085 Maintains a stack which grows downwards. That is Higher address to lower address

POP

POP Rp - This instruction is used to retrieve data from stack. i.e. 16-bit data from top of stack is stored in the specified register pair.

Example:

POP B; store data from top of stack in BC pair

POP D; store data from top of stack in DE pair

POP H; store data from top of stack in HL pair

POP PSW; store data from top of stack in Accumulator and Flag register

When this instruction is executed by the processor, it copies a byte from top of stack into lower byte of register pair and increments SP by 1, Then it again copies a

byte from top of stack into higher byte of the register pair specified in the instruction and increments SP by 1.

Machine-Control Operations.

HLT- Halt

When this instruction is executed by the processor it stops executing and enters into wait state. Address and data bus of the processor are kept in high impedance state.

NOP - No operation

This instruction performs nothing except wasting processor time. This instruction is used for writing delays.

EI- Enable interrupts

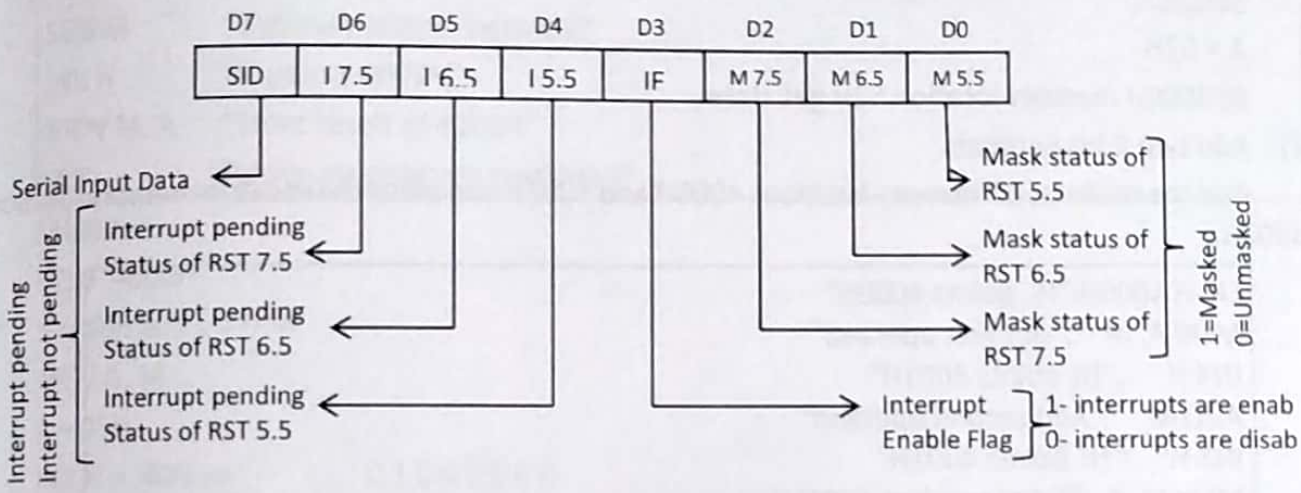
This instruction is used to enable the interrupts

DI - disable interrupts

This instruction is used to disable the interrupts

Read interrupt mask (RIM)

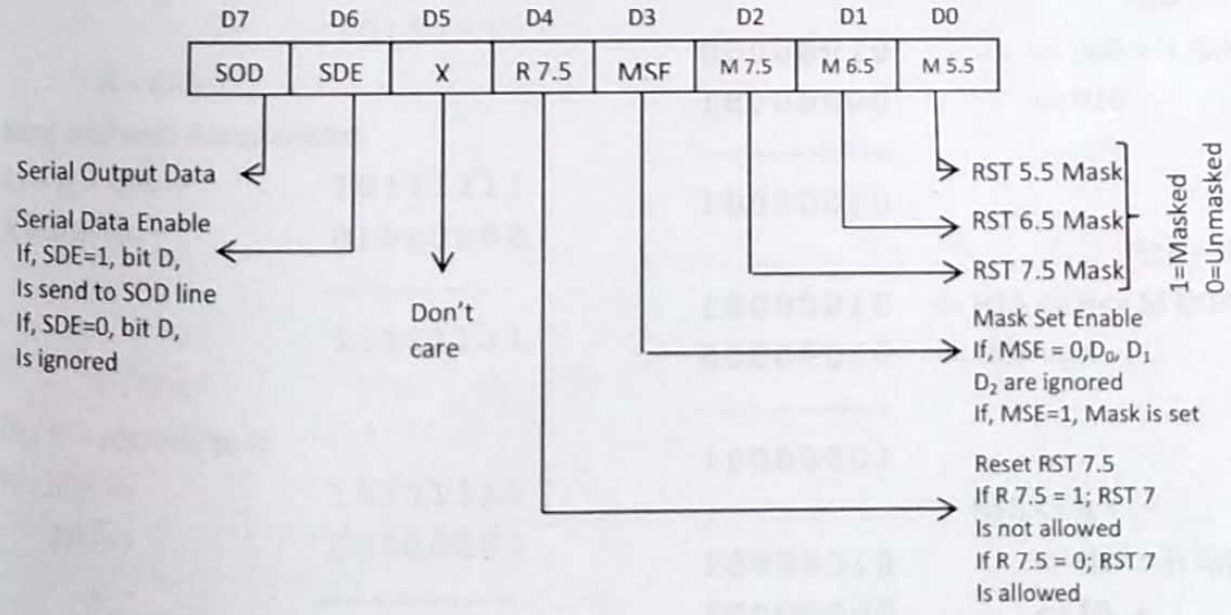
RIM -This is a multipurpose instruction used to read the status of interrupts 7.5, 6.5, 5.5 and read serial data input bit. The instruction loads eight bits in the accumulator with the following interpretations.



Set interrupt mask (SIM)

SIM none This is a multipurpose instruction and used to implement the 8085 interrupts 7.5, 6.5, 5.5, and serial data output. The instruction interprets the accumulator contents as

Example: SIM



Simple Programs

1) Store 8-bit data in memory

```
MVI A, 52H  :"Store 32H in the accumulator"
STA 4000H  :"Copy accumulator contents at address  4000H"
HLT        :"Terminate program execution"
```

Solution:

A = 52H

At 4000H memory location 52H get store.

2) Add two 8-bit numbers

Add the contents of memory locations 4000H and 4001H and place the result in memory location 4002H.

```
LXI H,4000H:"HL points 4000H"
MOV A, M   :"Get first operand"
INX H      :"HL points 4001H"
ADD M      :"Add second operand"
INX H      :"HL points 4002H"
MOV M, A   :"Store result at 4002H"
HLT        :"Terminate program execution"
```

Solution:

LXI H, 4000H

H = 40H and L = 00H

MOV A, M

A = 40H

```
INX H = 40H =>          01000000
        01H =>          00000001
                        --------
                        01000001
H = 41H
ADD M =>H => 41H =>     01000001
          A => 40H =>   01000000
                        --------
                        10000001
          A => 81H
INX H = 41H =>          01000001
      +  01 =>          00000001
                        --------
                        01000010
          H = 42H
```

3) Subtract two 8-bit numbers

Subtract the contents of memory location 4001H from the memory location 2000H and place the result in memory location 4002H.

```
LXI H,4000H:"HL points 4000H"
MOV A, M    :"Get first operand"
INX H       :"HL points 4001H"
SUB M       :"Subtract second operand"
INX H       :"HL points 4002H"
MOV M, A    :"Store result at 4002H"
HLT         :"Terminate program execution"
```

Solution:

LXI H, 4000H

H = 40H and L = 00H

MOV A, M

A = 40H

INX H = 40H => 0 1 0 0 0 0 0 0

01H => 0 0 0 0 0 0 0 1

---

0 1 0 0 0 0 0 1

H = 41H

SUB M = 41H => 0 1 0 0 0 0 0 1

Take 2's complement of 41H

1 0 1 1 1 1 1 1

H = BFH

Now add with Accumulator

M = H = BFH=> 1 0 1 1 1 1 1 1

A = 40H => 0 1 0 0 0 0 0 0

---

1 1 1 1 1 1 1 1

A = FFH

INX H – add one to H

H = BFH => 1 0 1 1 1 1 1 1

01H => 0 0 0 0 0 0 0 1

---

1 1 0 0 0 0 0 0

H = C0H

Answer A= FFH.

4) Finding one's complement of a number

Find the 1's complement of the number stored at memory location 4400H and store the complemented number at memory location 4300H.

```
LDA 4400B:"Get the number"
CMA       :"Complement accumulator value"
STA 4300H:"Store the result"
HLT       :"Terminate program execution"
```

Solution:

A = 44H

CMA means take 1's complement

A= 44H => 0 1 0 0 0 1 0 0

1's complement of 44H replace 0 with 1 and 1 with 0

1 0 1 1 1 0 1 1

A = BBH

5) Finding Two's complement of a number

Find the 2's complement of the number stored at memory location 4200H and store the complemented number at memory location 4300H

```
LDA 4200H:"Get the number"
CMA       :"Complement the number"
ADI 01H   :"Add one in the number"
STA 4300H :"Store the result"
HLT       :"Terminate program execution"
```

Solution:

A = 42H

CMA means take 1's complement

A= 42H => 0 1 0 0 0 0 1 0

1's complement of 42H replace 0 with 1 and 1 with 0

1 0 1 1 1 1 0 1

A = BDH

ADI 01H

A = BDH => 1 0 1 1 1 1 0 1

01H => 0 0 0 0 0 0 0 1

---

1 0 1 1 1 1 1 0

So, the result is A = BEH

INSTRUCTION EXECUTION AND TIMING DIAGRAM

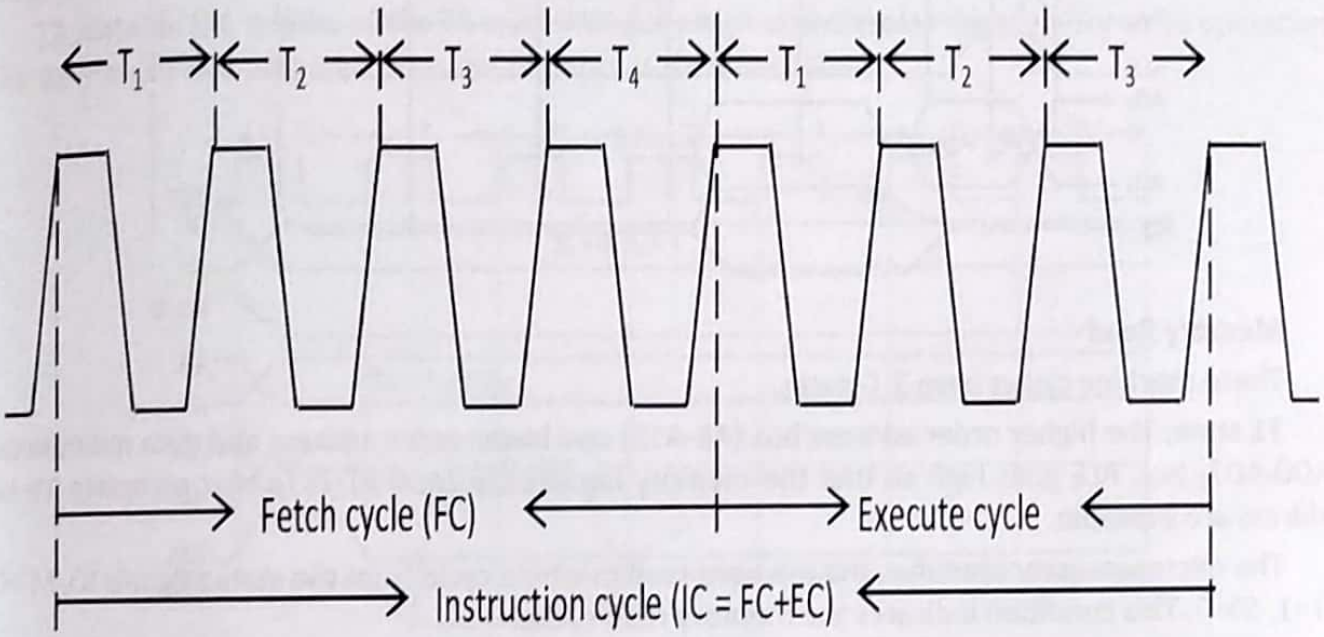
Each instruction in 8085 microprocessor consists of two parts- operation code (opcode) and operand. The opcode is a command such as ADD and the operand is an object to be operated on, such as a byte or the content of a register.

Timing Diagram is a graphical representation. It represents the execution time taken by each instruction in a graphical format. The execution time is represented in T-states.

Instruction Cycle: The time taken by the processor to complete the execution of an instruction.

Machine Cycle: The time required to complete one operation, accessing either the memory or I/O device. A machine cycle consists of three to six T-states.

T-State: Time corresponding to one clock period. It is the basic unit to calculate execution of instructions or programs in a processor.

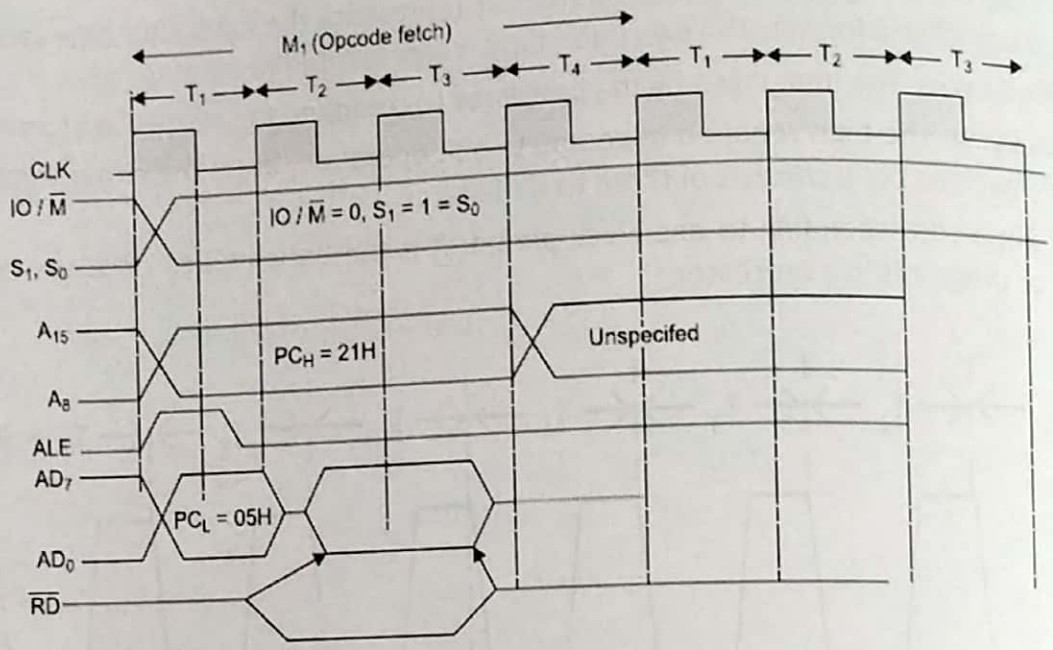


To execute a program, 8085 performs various operations as:

- Opcode fetch
- Operand fetch
- Memory read/write
- I/O read/write

Opcode fetch

- Each instruction of the processor has one byte opcode.The opcodes are stored in memory. So, the processor executes the opcode fetch machine cycle to fetch the opcode from memory.
- Hence, every instruction starts with opcode fetch machine cycle.
- The time taken by the processor to execute the opcode fetch cycle is 4T.In this time, the first, 3 T-states are used for fetching the opcode from memory and the remaining T-states are used for internal operations by the processor.



Memory Read

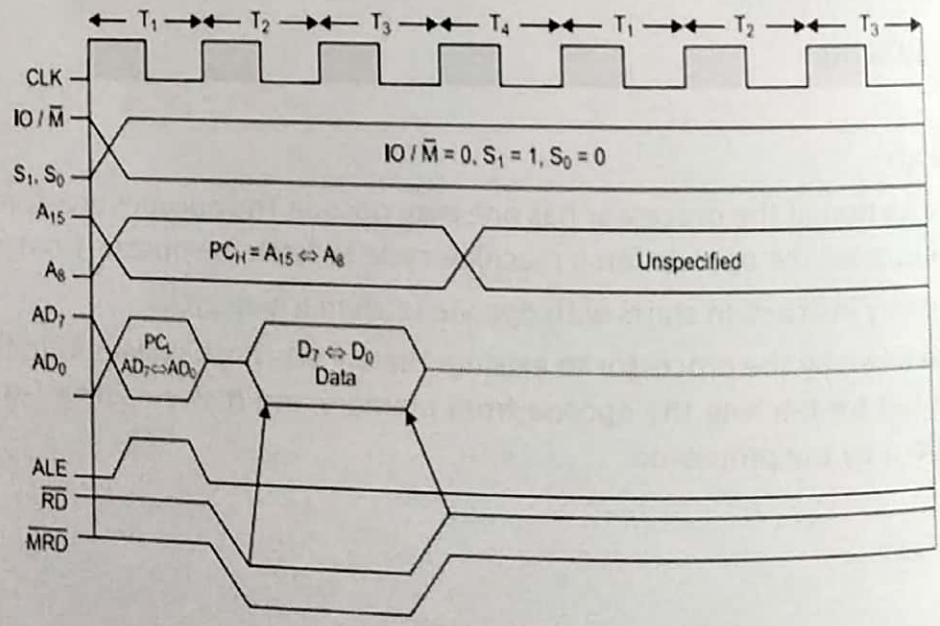
These machine cycles have 3 T-states.

T1 state: The higher order address bus (A8-A15) and lower order address and data multiplexed (AD0-AD7) bus. ALE goes high so that the memory latches the (AD0-AD7) so that complete 16-bit address are available.

The microprocessor identifies the memory read machine cycle from the status signals IO/M'=0, S1=1, S0=0. This condition indicates the memory read cycle.

T2 state: Selected memory location is placed on the (D0-D7) of the A/D multiplexed bus. RD' goes LOW

T3 State: The data which was loaded on the previous state is transferred to the microprocessor. In the middle of the T3 state RD' goes high and disables the memory read operation. The data which was obtained from the memory is then decoded.



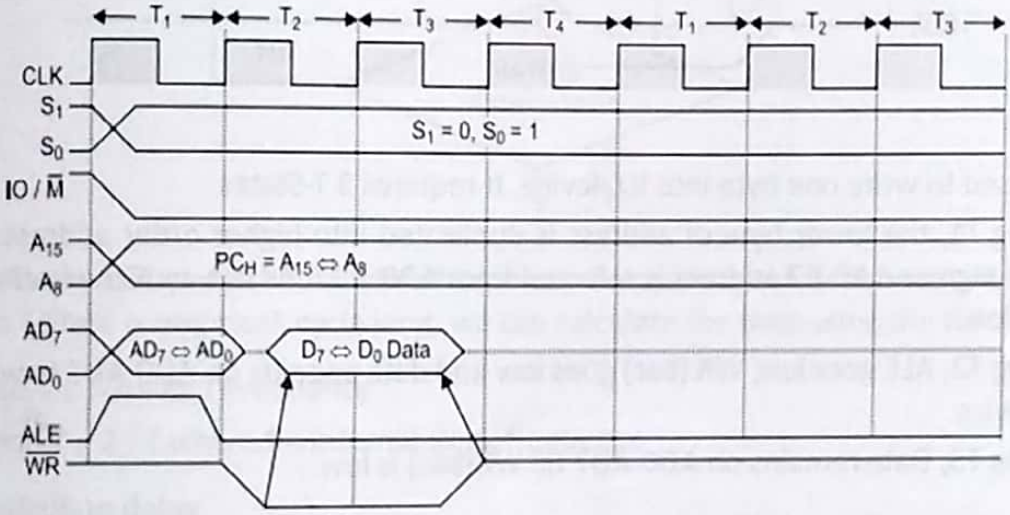
Memory Write

These machine cycles require3T-states.

T1 state: The higher order address bus (A8-A15) and lower order address and data multiplexed (AD0-AD7) bus. ALE goes high so that the memory latches the (AD0-AD7) so that complete 16-bit address are available. The microprocessor identifies the memory read machine cycle from the status signals IO/M'=0, S1=0, S0=1. This condition indicates the memory read cycle.

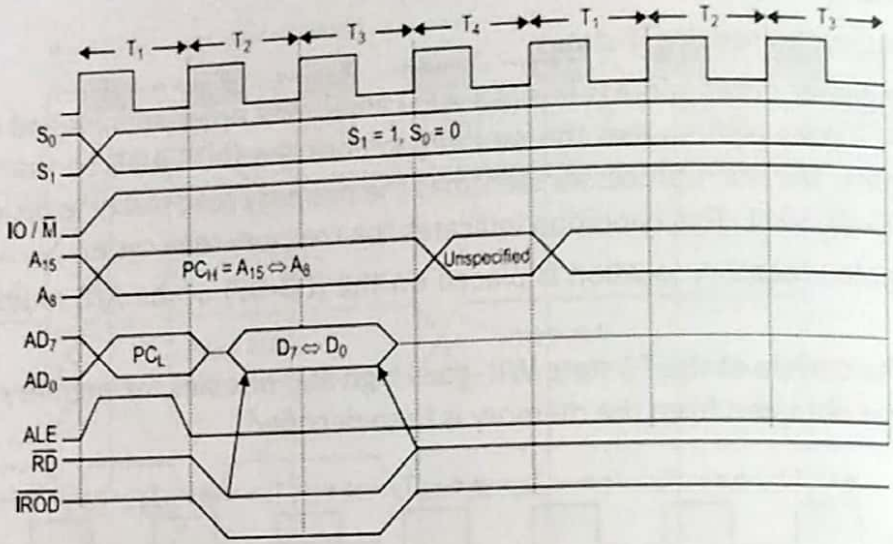
T2 state: Selected memory location is placed on the (D0-D7) of the A/D multiplexed bus. WR' goes LOW

T3 State: In the middle of the T3 state WR' goes high and disables the memory write operation. The data which was obtained from the memory is then decoded.



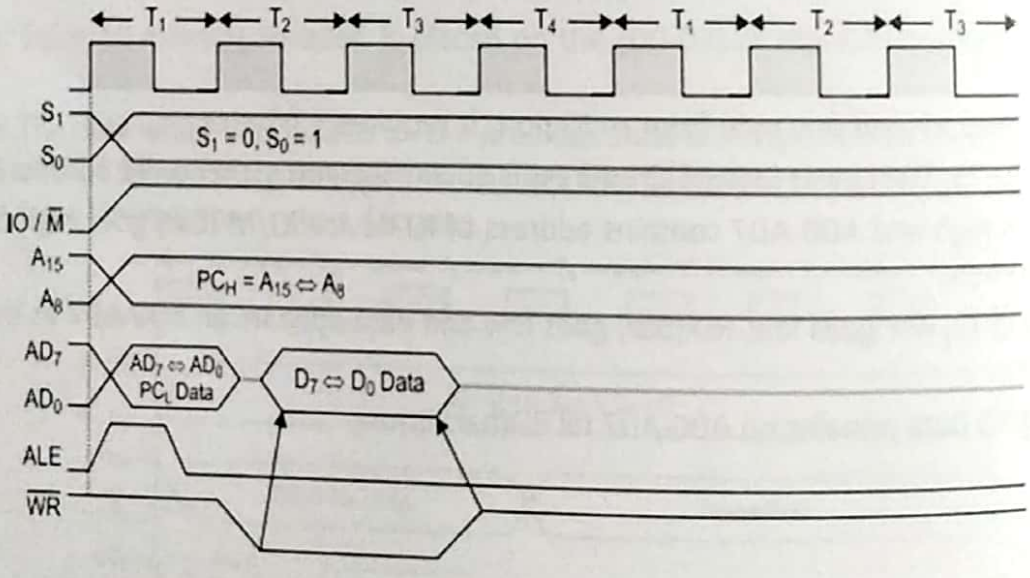
I/O read

- It is used to read one byte from an IO port. It requires 3 T-States.
- During T1, The Lower Byte of IO address is duplicated into higher order address bus A8-A15.
- ALE is high and AD0-AD7 contains address of IO device.IO/M (bar) goes high as it is an IO operation.
- During T2, ALE goes low, RD (bar) goes low and data appears on AD0-AD7 as input from IO device.
- During T3 Data remains on AD0-AD7 till RD(bar) is low.



I/O Write

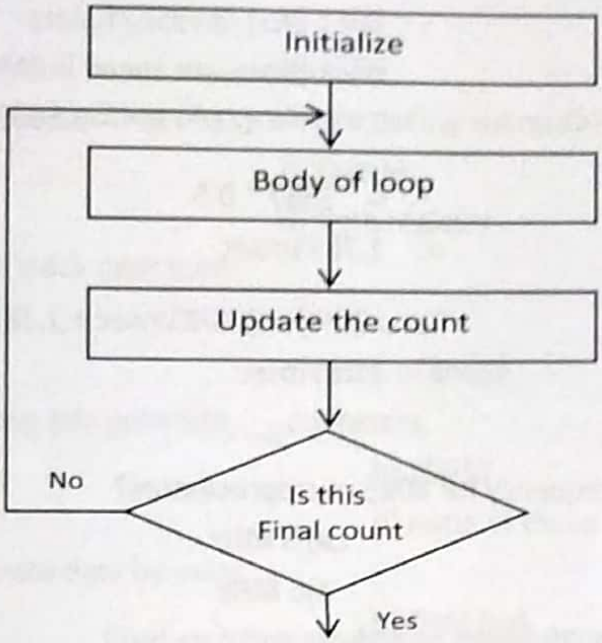
- It is used to write one byte into IO device. It requires 3 T-States.
- During T1, the lower byte of address is duplicated into higher order address bus A8-A15. ALE is high and A0-A7 address is selected from AD0-AD7. As it is an IO operation IO/M (bar) goes low.
- During T2, ALE goes low, WR (bar) goes low and data appears on AD0-AD7 to write data into IO device.
- During T3, Data remains on AD0-AD7 till WR(bar) is low.



Counter and Delay

A loop counter is set up by loading a register with a certain value. Then using the DCR (to decrement) and INR (to increment) the contents of the register are updated. A loop is set up with a conditional jump instruction that loops back or not depending on whether the count has reached the termination count.

The operation of a loop counter can be described using the following flowchart.



To calculate the delay, we should know how many T-States an instruction requires, and keeping in mind that a T-State is one clock cycle long, we can calculate the time using the following formula:

Delay = No. of T-States / Frequency

Time period T = 1 / f where f = Internal clock frequency

How to calculate delay

Examples of delay loop with up frequency = 2MHz

MVI C, FFH	7T-state
LOOP: DCR C	4T-state
JNZ LOOP	10T-state

1) To find the no of T-state required for each instruction

   Outside the loop requires 7T-state

   Inside the loop requires (4 + 10)= 14T-state

2) Next convert the count from Hex to Decimal

   Count =FFH convert into decimal => 255

As the last rotation of the loop the JNZ will not JMP to the address so it will require 7T-stateinstead of 10T-state, hence we have to deduct 3T-state from inner loop.

Delay = No of T-state/frequency

To calculate the delay formula :->

T_{delay} (total delay) = T_0 (outside the loop)+ T_L (delay of inner loop)

Outer loop delay calculation

$$T_0 = 7 * 0.5 * 10^{-6}$$

$$= 3.5\text{usec}$$

$$= 0.0035\text{msec}$$

Inner loop delay calculation

= (14 * 255) -3 =3567T-state

microprocessor speed is 2MHz

T = 1/F = 1/2 * 10^{-6} = 0.5usec

T_L = 3567 * 0.5

= 1.7835msec

Total Delay = 0.0035msec + 1.7835msec

= 1.787msec

Objective Questions:

Q.1. What is the clock frequency for 8085 microprocessors?

a) 2 MHz
b) 3 MHz
c) 4 MHz
d)6 MHz

Q.2. Signal used for demultiplexing of address and data bus?

a) HOLD
b) ALE
c) READY
d) INTR

Q.3. Which of the following is 16-bitregister?

a) PC
b) SP
c) None of these
d) Both (a) and (b)

Q.4. How many flags are available in 8085?

a) 4
b) 6
c) 5
d) 7

Q.5. Stack pointer register is used for accessing

a) Stack
b) Memory
c) Strings
d) None of above

Q.6. Which of the following interrupt has the highest priority?

a) RST 7.5
b) RST 6.5
c) RST 5.5
d) TRAP

Q.7. Which register is not available for user in microprocessor?

a) A
b)B
c) D
d) W

Q.8. Which signal is used for synchronizing the speed of 8085 with slower peripheral devices?

a) RESET
b)READY
c) HOLD
d) INTR

Q.9. The register used for sequencing the execution of instructions is

a) Stack Pointer
b) Accumulator
c) Program counter
d) W and Z

Q.10. Maximum memory which 8085 microprocessors can address

a) 64KB
b) 32KB
c) 4KB
d) 16KB

Q.11. Which is used to store critical pieces of data during subroutines and interrupts:

a) Stack
b) Queue
c) Accumulator
d) Data register

Q.12. Which is the basic stack operation:

a) PUSH
b) POP
c) BOTH A and B
d) None of these

Q.13. A 16-bit address bus can generate___ addresses:

a) 32767
b) 25652
c) 65536
d) none of these

Q.14. CPU can read & write data by using

a) Control bus
b) Data bus
c) Address bus
d) None of these

Q.15. What is SIM?

a) Select Interrupt Mask
b) Sorting Interrupt Mask
c) Set Interrupt Mask.
d) none of these

Q.16. Address line for TRAP is?

a) 0023H
b) 0024H
c) 0033H
d) 0099H

Q.17. Which of the following is not an arithmetic instruction?

a) INC (increment)
b) CMP (compare)
c) DEC (decrement)
d) ROL (rotate left)

Q.18. Which group of instructions do not affect the flags?

a) Arithmetic operations
b) Logic operations
c) Data transfer operations
d)None of the above

Q.19. Which is not the control bus signal:

a) READ
b)WRITE
c)RESET
d)None of these

MICROPROCESSOR 8086

Introduction of 8086

The Microprocessor 8086 was a 16-bit processor developed by Intel in 1978 on HMOS Technology but now it works on HCMOS technology. It is a 16-bit Microprocessor. It gave rise to the x86 architecture. Itshaving16-bit data bus, a 20-bit address bus that provides up to 1MB storage, with +5v power supply and works on 5MHz frequency. It consists of powerful instruction set, which provides operations like multiplication and division easily. It consists of 29,000 transistors.

It supports two modes of operation, i.e. Maximum mode and Minimum mode. Maximum mode is suitable for system having multiple processors and Minimum mode is suitable for system having a single processor.

Architecture 8086

The Architecture of 8086 supports a 16-bit ALU, a set of 16-bit register and provides segmented memory addressing capability, a rich instruction set, powerful interrupt structure, fetched instruction queue for overlapped fetching and execution etc.

The complete architecture of 8086 can be divided into two parts

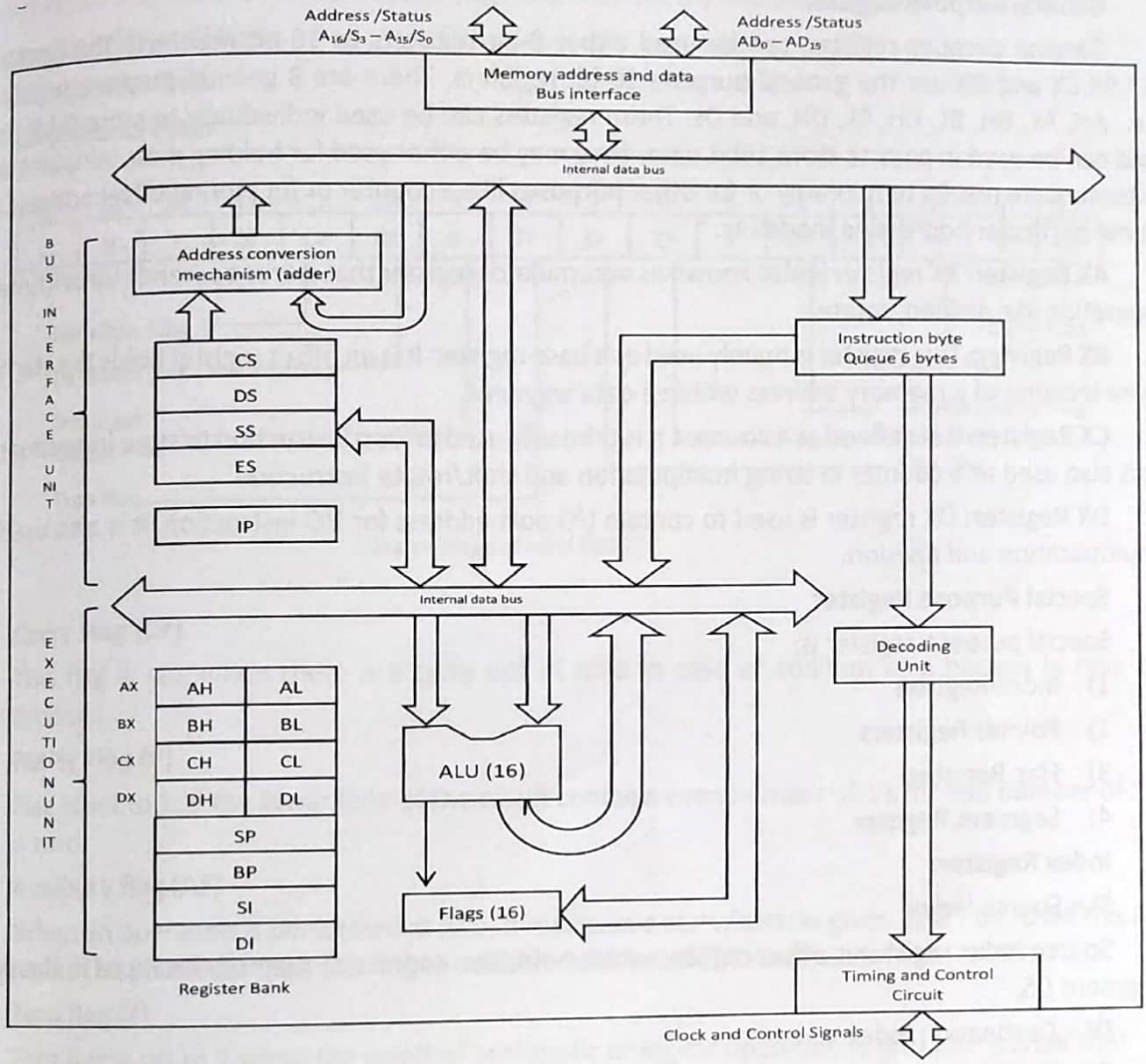
a) **Bus Interface Unit (BIU)**

BIU contains Instruction queue, Segment registers, Instruction pointer, Address adder.

b) **Execution Unit (EU)**

EU contains Control circuitry, Instruction decoder, ALU, Registers, Flag register

The **BIU** performs all bus operations such as instruction fetching, reading and writing operands for memory and calculating the addresses of the memory operands. The instruction bytes are transferred to the instruction queue.



EU 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.

Both units operate simultaneously but works asynchronously to give the 8086 an overlapping instruction fetch and execution mechanism which is called as Pipelining.

Let discuss the functional parts of 8086 microprocessors.

ALU (Arithmetic & Logic Unit):

This unit can perform various arithmetic and logical operation, based on the instruction to be executed. It can perform arithmetical operations, such as add, subtract, increment, decrement, multiply, divide and compare etc. And logical operations, such as AND, OR, exclusive OR, shift/rotate etc.

Register Organisation

8086 has a powerful set of registers known as general purpose register and special purpose register. All of them are 16-bit register.

General Purpose Register

General purpose register, can be used either 8-bit registers or 16-bit registers. The registers AX,BX,CX and DX are the general purpose 16-bit registers. There are 8 general purpose registers, i.e., AH, AL, BH, BL, CH, CL, DH, and DL. These registers can be used individually to store 8-bit data and can be used in pairs to store 16bit data. They may be either used for holding data, variables and intermediate results temporarily or for other purposes like a counter or for storing offset address for some particular addressing mode etc.

AX Register: AX register is also known as accumulator register that stores operands for arithmetic operation like divided, rotate.

BX Register: This register is mainly used as a base register. It is an offset register holds the starting base location of a memory address within a data segment.

CX Register: It is defined as a counter. It is primarily used in loop instruction to store loop counter. It is also used as a counter in string manipulation and shift/rotate instruction

DX Register: DX register is used to contain I/O port address for I/O instruction. It is also used in multiplication and division.

Special Purpose Register

Special purpose register is:

1) Index Register
2) Pointer Registers
3) Flag Register
4) Segment Register

Index Register:

SI = Source Index

Source index register is offset register which holds the address of memory locations in the data segment DS.

DI = Destination Index

Destination index register is offset register which holds the address of memory locations in the Extra segment ES.

Pointer Registers

SP (Stack Pointer):

stack pointer is an offset register which holds the address of memory location in the stack. It is used in conjunction with SS for accessing the stack segment.

BP (Base Pointer):

Base pointer is an offset register which holds the address of memory location in the stack.

IP(Instruction Pointer):

It Always points to the next instruction to be executed within the currently executing code segment. This register contains the 16-bit offset address pointing to the next instruction code within the 64Kb of the code segment area.

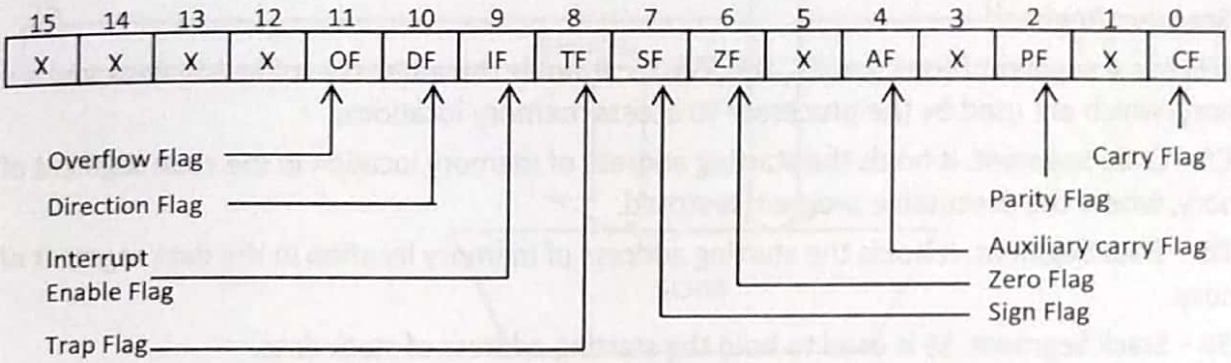
Flag Register

It is a 16-bit register that behaves like a flip-flop, i.e. it changes its status according to the result

stored in the accumulator. It has 9 active flags and they are divided into 2 groups

- Conditional Flags
- Control Flags

Conditional Flags



Status Flags of Intel 8086

Carry Flag (CY)

This flag is set, when there is a carry out of MSB in case of addition or a borrow in case of subtraction.

Parity Flag (P)

Flag is set to 1, if the lower byte of the result contains even number of 1's for odd number of 1's set to zero.

Auxiliary flag (AF)

When an operation is performed at ALU, it results in a carry/barrow given by D3 bit to D4 this AF flag is set. The processor uses this flag to perform binary to BCD conversion.

Zero flag (Z)

Zero flag is set to 1 when the result of arithmetic or logical operation is zero else it is set to 0.

Sign flag(S)

Sign flag holds the sign of the result, i.e. when the result of the operation is negative, then the sign flag is set to 1 else set to 0.

Overflow flag (OF)

This flag is set, if an overflow occurs, i.e., if the result of a signed operation is large enough to accommodate in a destination register. The result is more than 15-bits in size, then the overflow will be set.

Control Flags

Trap Flag (TP):

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.

Interrupt Flag (IF):

It is an interrupt enable/disable flag. If it is set, the Maskable interrupt of 8086 is enabled and if it is reset, the interrupt is disabled.

Direction Flag (DF):

It is used in string operation. If it is set, string bytes are accessed from higher memory address to lower memory address (auto-decrement).

When it is reset, the string bytes are accessed from lower memory address to higher memory address (auto-increment).

Segment Register

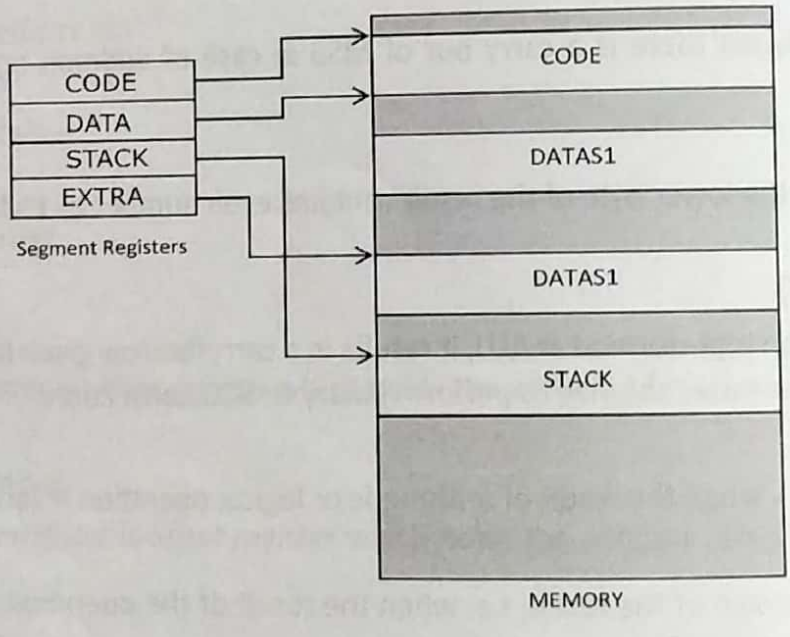
BIU has 4 segment buses, i.e. CS, DS, SS& ES. It holds the addresses of instructions and data in memory, which are used by the processor to access memory locations.

CS – Code Segment. It holds the starting address of memory location in the code segment of the memory, where the executable program is stored.

DS – Data Segment. It holds the starting address of memory location in the data segment of the memory.

SS – Stack Segment. SS is used to hold the starting address of stack data.

ES – Extra Segment. ES is additional data segment, which is used by the string to hold the starting address of extra destination data.



Each segment register is 16-bit. Address bus in 8086 is 20-bit address lines so, it can access maximum of 1MB (1048576byte) of memory. Complete 1MB of memory is logically divided into 16 logical segments. Each segment thus contains 64kbyte of memory. While addressing any location in the memory bank, the physical address is calculated from two parts, the first is **segment address** and the second is **offset.**

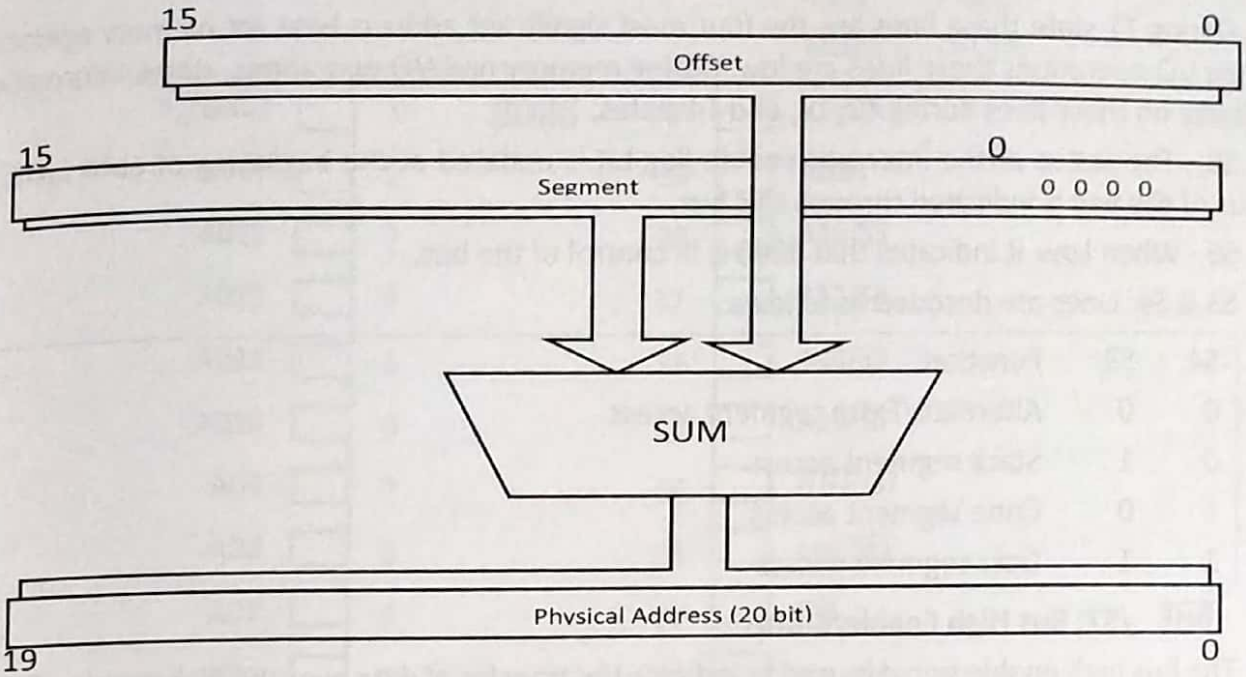
For generating a physical address, content of segment registers also called as segment address is shifted left bit-wise four times and to this result, content of offset register also called as offset address is added, to produce a 20-bit physical address.

Example: Assume that CS register is 3000H & IP is 2000H. To fetch an instruction from memory find the memory address from which the next instruction will be fetch

Solution=> physical address = CS x 10 + IP

= 3000H x 10 + 2000H

= 32000H



Accessing Memory location

Segment	Offset Register	Function
CS	IP	Address of next instruction
DS	BX, DI, SI ,8-bit or 16-bit	Address of data
SS	SP & BP	Address of stack
ES	DI for string operation	Address of destination data

Instruction Queue

A group of First-In-First Out (FIFO) in which up to 6 bytes of instruction code are pre-fetched from the memory ahead of time. This is done in order to speed up the execution by overlapping instruction fetch with execution. This mechanism is known as pipelining

Fetching the next instruction while the current instruction executes is called **pipelining**.

8086 Pin Description

Power supply and frequency signals

It uses 5V DC supply at VCC, and uses ground at VSS for its operation.

Clock signal

Clock signal provides timing to the processor for operations. Its frequency is different for different versions, i.e. 5MHz, 8MHz and 10MHz.

Address/data bus

AD0-AD15. These are 16 address/data bus. AD0-AD7 carries low order byte data and AD8-AD15 carries higher order byte data. During the first clock cycle, it carries 16-bit address and after that it carries 16-bit data.

Address/status bus

A19/S6, A18/S5, A17/S4, A16/S3: 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.

S3 & S4: Lines are decoded as follows:

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

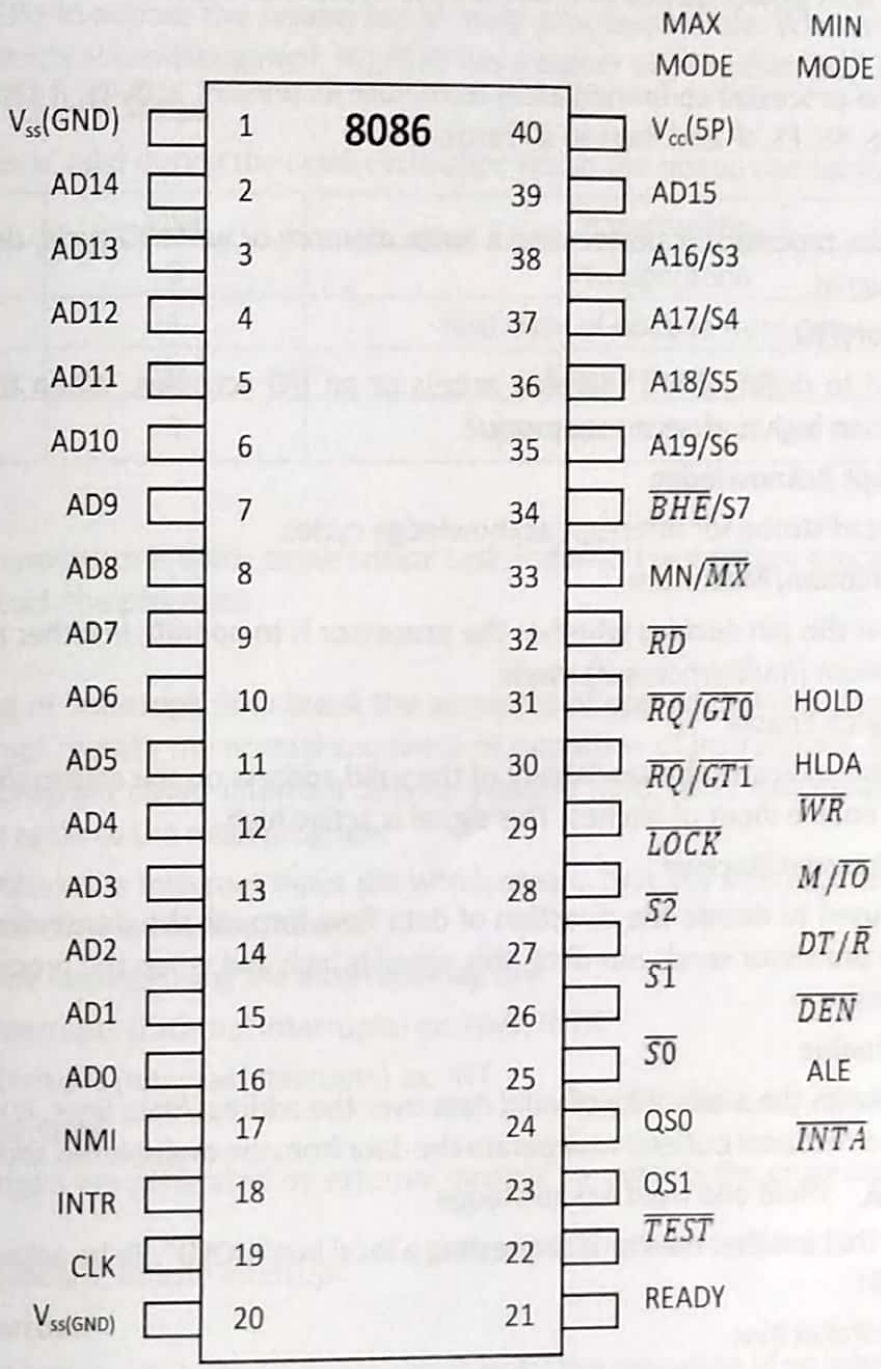
$\overline{BHE}$ /S7: Bus High Enable/Status

The bus high enable signal is used to indicate the transfer of data over the higher order (D15-D8) data bus. If BHE is low then D15-D8 used to transfer data.

A0	BHE	Characteristics
0	0	Whole word
0	1	Upper byte from/to odd address
1	0	Lower byte from/to even address
1	1	None

$\overline{RD}$: READ

The Read strobe indicates that the processor is performing a memory or I/O read cycle.



$\overline{\text{TEST}}$: TEST pin is examined by the "WAIT" instruction. If the TEST pin is Low, execution continues. Otherwise the processor waits in an "idle" state.

INTR: 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.

NMI: Non-Maskable 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.

READY

It synchronizes with slower device that its ready to accept the data.

Reset

Reset causes the processor to immediately terminate its present activity. It causes the 8086 to initialize registers DS, SS, ES, IP and flags to all zeros.

$\overline{WR}$: Write

Indicates that the processor is performing a write memory or write IO cycle, depending on the state of the M /IO signal.

M/ $\overline{IO}$: Memory/IO

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 accesses memory.

$\overline{INTA}$: Interrupt Acknowledge

It is used as a read strobe for interrupt acknowledge cycles.

MN/ $\overline{MX}$: Minimum/Maximum

The logic level at this pin decides whether the processor is to operate in either minimum (single processor) or maximum (multiprocessor) mode.

ALE: Address Latch Enable

This output signal indicates the availability of the valid address on the address/data lines, and is connected to latch enable input of latches. This signal is active high

DT/ $\overline{R}$: Data Transmit/Receive

This output is used to decide the direction of data flow through the transceivers (bidirectional buffers). When the processor sends out data, this signal is high and when the processor is receiving data, this signal is low.

$\overline{DEN}$: Data Enable

This signal indicates the availability of valid data over the address/data lines. It is used to enable the transceivers (bidirectional buffers) to separate the data from the multiplexed address/data signal.

HOLD & $\overline{HLDA}$: Hold and Hold Acknowledge

Hold indicates that another master is requesting a local bus "HOLD". To be acknowledged, HOLD must be active HIGH

$\overline{S}_0$, $\overline{S}_1$, $\overline{S}_2$:Status Pins

These status lines are encoded as shown

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 Access
1	0	1	Read Memory
1	1	0	Write Memory
1	1	1	Passive

$\overline{RQ}$ / $\overline{GT0}$, $\overline{RQ}$ / $\overline{GTI}$: Request/Grant

These signal is same as HOLD/HLDA signal. Request/Grant signals used by the other processors requesting the CPU to release the system bus in multi-processor mode. When the signal is received by CPU, then it sends acknowledgment. RQ/GT0 has a higher priority than RQ/GT1.

$\overline{SQ0}$, $\overline{SQ1}$: Queue Status

Queue Status is valid during the clock cycle after which the queue operation is performed.

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

$\overline{LOCK}$:

When microprocessor is doing some critical task, it doesn't want other processor to take control of buses. It will Lock the processor.

Interrupt

The meaning of 'Interrupt' is to break the sequence of operation. While the CPU is executing a program, 'interrupt' breaks the normal sequence of execution of instructions, diverts its execution to some other program called Interrupt Service Routine (ISR). After executing ISR, the control is transferred back again to the main program.

NMI: Non- Maskable Interrupt input pin which means that any interrupt request at NMI input cannot masked or disabled by any means.

INTR: It can be masked using the Interrupt Flag (IF).

Hardware Interrupts (External Interrupts) ex: NMI, INTR

Software Interrupts (Internal Interrupts) ex: INT

1. **External interrupts**

These interrupts are generated by external devices i.e outside the processor (using NMI, INTR pins).

Example: Keyboard, Mouse interrupt.

2. **Internal interrupts**

It is generated internally by the process circuit or by the execution of an interrupt instruction.

Example: INT instruction, overflow interrupt, divide by zero

Addressing modes

Immediate addressing Mode - the data is provided in the instruction directly.

MOV CX,4374H

Direct Addressing Modes – the instruction operand specifies the memory address where data is located.

MOV BL, [4374H]

Register Addressing Mode – copy the data from one to another register.

MOV CX, AX

Register Indirect Addressing Modes - Instruction specifies a register containing an address, where data is located. This addressing mode work with SI, DI, BX and BP register.

MOV AL, [BX] or MOV AL, CS: [BX]

Register Relative Addressing Modes- Effective address is formed by adding an 8-bit or 16-bit displacement with contents of any one of register BX, BP, SI and DI

MOV AX,50H[BX][SI]

Indexed Addressing Modes - Offset is stored in one of the index register DS is default for register SI & DI

MOV AX, [SI]

Based Indexed Addressing Modes - Effective address of data is formed by adding contents of base register (BX or BP)

MOV AX, [BX][SI]

Relative Based Indexed Addressing Modes - Effective address is formed by adding an 8-bit or 16-bit displacement

MOV AX,50H[BX][SI]

Intersegment jump =>is a jump to any memory location within the entire memory system

Intersegment jump=>is a jump anywhere within the current code segment.

Intersegment Addressing Modes - Address to which the control is to be transferred lies in the same segment in which control transferred lies in the same segment in which control transfer instruction lies and appear directly in instruction.

(8 -bit) short jump - 128 to +127

(16 -bit) long jump - 32768 to +32767

Intersegment Indirect Addressing Modes –

JMP [BX]

Intersegment Addressing Mode-Transfer to different segment CS & IP Destination are specified directly

JMP 5000H:2000H

Intersegment Indirect Addressing Mode - Jump to an address in other segment specified at effective address 2000H in DS/CS (far jmp)

JMP [2000H]

Instruction Set of 8086

An instruction is a binary pattern designed inside a microprocessor to perform a specific function. The entire group of instructions that a microprocessor supports is called Instruction Set.8086 has more than 20,000 instructions.

Classification of Instruction

1) Data transfer instruction

It is used to transfer the data from the source operand to the destination operand. All the store, move, load, exchange, input and output instruction belong to data transfer instruction.

MOV – Used to copy the byte or word from the provided source to the provided destination.

PUSH – Used to put a word at the top of the stack.

POP – Used to get a word from the top of the stack to the provided location.

PUSHA – Used to put all the registers into the stack.

POPA – Used to get words from the stack to all registers.

XCHG – Used to exchange the data from two locations.

XLAT – Used to translate a byte in AL using a table in the memory.

Instructions for input and output port transfer

IN – Used to read a byte or word from the provided port to the accumulator.

OUT – Used to send out a byte or word from the accumulator to the provided port.

Instructions to transfer the address

LEA – Used to load the address of operand into the provided register.

LDS – Used to load DS register and other provided register from the memory

LES – Used to load ES register and other provided register from the memory.

Instructions to transfer flag registers

LAHF – Used to load AH with the low byte of the flag register.

SAHF – Used to store AH register to low byte of the flag register.

PUSHF – Used to copy the flag register at the top of the stack.

POPF – Used to copy a word at the top of the stack to the flag register.

2) Arithmetic and Logical instruction

Instructions performing arithmetic, logical, increment, decrement, compare and scan belong to arithmetic instruction.

Instructions to perform addition

ADD – Used to add the provided byte to byte/word to word.

ADC – Used to add with carry.

INC – Used to increment the provided byte/word by 1.

AAA – Used to adjust ASCII after addition.

DAA – Used to adjust the decimal after the addition/subtraction operation.

Instructions to perform subtraction

SUB – Used to subtract the byte from byte/word from word.

SBB – Used to perform subtraction with borrow.

DEC – Used to decrement the provided byte/word by 1.

NPG – Used to negate each bit of the provided byte/word and add 1/2's complement.

CMP – Used to compare 2 provided byte/word.

AAS – Used to adjust ASCII codes after subtraction.

DAS – Used to adjust decimal after subtraction.

Instruction to perform multiplication

MUL – Used to multiply unsigned byte by byte/word by word.

IMUL – Used to multiply signed byte by byte/word by word.

AAM – Used to adjust ASCII codes after multiplication.

Instructions to perform division

DIV – Used to divide the unsigned word by byte or unsigned double word by word.

IDIV – Used to divide the signed word by byte or signed double word by word.

AAD – Used to adjust ASCII codes after division.

CBW – Used to fill the upper byte of the word with the copies of sign bit of the lower byte.

CWD – Used to fill the upper word of the double word with the sign bit of the lower word.

Instructions to perform logical operation

NOT – Used to invert each bit of a byte or word.

AND – Used for adding each bit in a byte/word with the corresponding bit in another byte/word.

OR – Used to multiply each bit in a byte/word with the corresponding bit in another byte/word.

XOR – Used to perform Exclusive-OR operation over each bit in a byte/word with the corresponding bit in another byte/word.

TEST – Used to add operands to update flags, without affecting operands.

3) Branching Instruction

In this instruction transfer control of execution to the specified address. Instruction such as call, jump, interrupt and return.

CALL – Used to call a procedure and save their return address to the stack.

RET – Used to return from the procedure to the main program.

JMP – Used to jump to the provided address to proceed to the next instruction.

JA/JNBE – Used to jump if above/not below/equal instruction satisfies.

JAE/JNB – Used to jump if above/not below instruction satisfies.

JBE/JNA – Used to jump if below/equal/ not above instruction satisfies.

JC – Used to jump if carry flag CF = 1

JE/JZ – Used to jump if equal/zero flag ZF = 1

JG/JNLE – Used to jump if greater/not less than/equal instruction satisfies.

JGE/JNL – Used to jump if greater than/equal/not less than instruction satisfies.

JL/JNGE – Used to jump if less than/not greater than/equal instruction satisfies.

JLE/JNG – Used to jump if less than/equal/if not greater than instruction satisfies.

JNC – Used to jump if no carry flag (CF = 0)

JNE/JNZ – Used to jump if not equal/zero flag ZF = 0

JNO – Used to jump if no overflow flag OF = 0

JNP/JPO – Used to jump if not parity/parity odd PF = 0

JNS – Used to jump if not sign SF = 0

JO – Used to jump if overflow flag OF = 1

JP/JPE – Used to jump if parity/parity even PF = 1

JS – Used to jump if sign flag SF = 1

4) Loop Instruction

If these instructions have REP prefix with CX used as count register, they can be used to implement unconditional and conditional loop. Instruction such as LOOP, LOOPNZ and LOOPZ.

LOOP – Used to loop a group of instructions until the condition satisfies, i.e., CX = 0

LOOPE/LOOPZ – Used to loop a group of instructions till it satisfies ZF = 1 & CX = 0

LOOPNE/LOOPNZ – Used to loop a group of instructions till it satisfies ZF = 0 & CX = 0

JCXZ – Used to jump to the provided address if CX = 0

5) **Machine control Instruction**

These instructions control the machine status. NOP, HLT, WAIT and LOCK are control Instruction.

6) **Flag manipulation Instruction**

All the instructions which directly affects the flag register, come under flag instruction. Instruction like CLD, STD, CLI, STI, CLC, CMC, STC.

STC – Used to set carry flag CF to 1

CLC – Used to clear/reset carry flag CF to 0

CMC – Used to complement at the state of carry flag CF.

STD – Used to set the direction flag DF to 1

CLD – Used to clear/reset the direction flag DF to 0

STI – Used to set the interrupt enable flag to 1, i.e., enable INTR input.

CLI – Used to clear the interrupt enable flag to 0, i.e., disable INTR input.

7) **Shift & rotate Instruction**

Instructions such as bitwise shifting or rotation in either direction with or without a count in CX.

Instructions to perform shift operations

SHL/SAL – Used to shift bits of a byte/word towards left and put zero(S) in LSBs.

SHR – Used to shift bits of a byte/word towards the right and put zero(S) in MSBs.

SAR – Used to shift bits of a byte/word towards the right and copy the old MSB into the new MSB.

Instructions to perform rotate operations

ROL – Used to rotate bits of byte/word towards the left, i.e. MSB to LSB and to Carry Flag [CF].

ROR – Used to rotate bits of byte/word towards the right, i.e. LSB to MSB and to Carry Flag [CF].

RCR – Used to rotate bits of byte/word towards the right, i.e. LSB to CF and CF to MSB.

RCL – Used to rotate bits of byte/word towards the left, i.e. MSB to CF and CF to LSB.

8) **String Instruction**

Instruction involve various string manipulation operations like load, move, scan, compare, store etc.

REP – Used to repeat the given instruction till CX ≠ 0.

REPE/REPZ – Used to repeat the given instruction until CX = 0 or zero flag ZF = 1.

REPNE/REPNZ – Used to repeat the given instruction until CX = 0 or zero flag ZF = 1.

MOVS/MOVSB/MOVSW – Used to move the byte/word from one string to another.

COMS/COMPSB/COMPSW – Used to compare two string bytes/words.

INS/INSB/INSW – Used as an input string/byte/word from the I/O port to the provided memory location.

OUTS/OUTSB/OUTSW – Used as an output string/byte/word from the provided memory location to the I/O port.

SCAS/SCASB/SCASW – Used to scan a string and compare its byte with a byte in AL or string

word with a word in AX.

LODS/LODSB/LODSW – Used to store the string byte into AL or string word into AX.

80286

- 16-bit up
- 24-bit address bus to address 16Mbyte of physical memory
- 1st processor to incorporate integrated memory management unit
- works in 2 operating modes real address mode and protected virtual address mode.
- 1GB of virtual memory
- swapping concept introduce

80386

- 32-bit up
- 32-bit address bus can address up to 4GByte of physical memory
- support virtual memory, paging
- page of size 4Kbytes each under segmented memory
- Memory management unit consists of segmentation unit and paging unit
- Allows only 256 pages
- Advantage of paging scheme is that complete segment of a task need not be in physical memory at any time only few pages of segments which are required currently for the execution need to be available in physical memory.

Objective Questions:

Q.1. The size of each segment in 8086 is

a) 64 kb
b) 24 kb
c) 50 kb
d) 16kb

Q.2. The _____ address of a memory is a 20-bit address for the 8086 microprocessors

a) Physical
b) Logical
c) Both
d) None of these

Q.3. The pin configuration of 8086 is available in the_____

a) 40 pin
b) 50 pin
c) 30 pin
d) 20 pin

Q.4. BP stand for

a) Bit pointer
b) Base pointer
c)Bus pointer
d) Byte pointer

Q.5. The processor uses the stack to keep track of where the items are stored on it this by using the

a) Stack pointer register
b) Queue pointer register
c) Both a & b
d) None of these

Q.6. How many type of addressing in memory

a) Logical address
b) Physical address
c) Both A and B
d) None of these

Q.7. The physical address of memory is

a) 20-bit
b) 16-bit
c) 32-bit
d) 64-bit

Q.8. AH stand for

a) Accumulator high
b) Address high
c) Appropriate high
d) Application high

Q.9. The Intel 8086 microprocessor is a _____ processor

a) 8-bit
b) 16-bit
c) 32-bit
d) 4-bit

Q.10. The 16-bit flag of 8086 microprocessor is responsible to indicate _____

a) the condition of result of ALU operation
b) the condition of memory
c) the result of addition
d) the result of subtraction

Q.11. The BIU prefetches the instruction from memory and store them in _____

a) queue
b) register
c) memory
d) stack

Q.12. The BIU contains FIFO register of size _____ bytes

a) 8
b) 6
c) 4
d) 12

Q.13. A _____ Instruction at the end of interrupt service program takes the execution back to the interrupted program

a) forward
b) return
c) data
d) line

Q.14. In 8086 the overflow flag is set when _____.

a) the sum is more than 16 bits.
b) signed numbers go out of their range after an arithmetic operation.
c) carry and sign flags are set.
d) subtraction

Q.15. A 20-bit address bus can locate _____.

a) 1,048,576 locations
b) 2,097,152 locations
c) 4,194,304 locations
d) 65536 locations

3modes of data communication

1) Simplex – data transmitted in one direction

2) Half Duplex – data transmitted in both the direction one at a time

3) Duplex or Full Duplex - data is transmitted in both the direction simultaneously

Rate of transmission is measured in bits/second or baud

Baud = bits/second

While transmitting data it should follow the bit pattern correctly

1) Synchronous transmission

While sending the data first 2 sync pulses are transmitted and then 8-bit data.

2) Asynchronous transmission

While sending the data first start bit, then parity bit, then 8-bit data and then stop bit.

Start bit is kept low and we can send one stop bit or two stop bit depending on the data transmission.

Objective Questions

Q.1. What is the address range in 80286?

a) 1 Mbytes
b) 2 Mbytes
c) 16 Mbytes
d) 32 Mbytes

Q.2. Which are the two modes of 80286?

a) Real mode and protected mode
b) Mode1 and mode2
c) Alternate and main
d) Mode A and mode B

Q.3. What does USART stand for?

a) universal synchronous asynchronous receiver transmitter
b) unique synchronous asynchronous receiver transmitter
c) universal sync address receiver transmitter
d) unique sync address receiver transmitter

Q.4. What rate can define the timing in the USART?

a) bit rate
b) baud rate
c) speed rate
d) voltage rate

Q.5. Which is the most commonly used USART?

a) 8253
b) 8254
c) 8259
d) 8251

Q.6. Which determines the mode 1 in the Intel 8253?

a) interrupt on terminal count
b) programmable one-shot
c) rate generator
d) square wave rate generator

ANSWER KEY FOR OBJECTIVE QUESTIONS				
Chapter	1.	2.	3.	4.
Q.1	A	B	D	C
Q.2	D	B	A	A
Q.3	A	D	A	A
Q.4	A	C	B	B
Q.5	A	A	A	D
Q.6	D	D	C	B
Q.7	A	D	A	
Q.8	B	B	A	
Q.9	B	C	B	
Q.10	B	A	A	
Q.11		A	A	
Q.12		B	B	
Q.13		C	B	
Q.14		B	A	
Q.15		C	A	
Q.16		B		
Q.17		D		
Q.18		C		
Q.19		C		
Q.20				