G. Pullaiah College of Engineering and Technology: Kurnool

Department Of Electronics and Communication Engineering

LECTURE NOTES

MICROPROCESSORS AND INTERFACING

PREPARED BY

V.SHANTHI

ASST PROFESSOR

DEPT OF ECE

GPCET

Department Of Electronics and Communication Engineering

G. Pullaiah College of Engineering and Technology: Kurnool

(Approved by AICTE, New Delhi, Recognized by UGC under 2 (f) & 12 B, Permanently Affiliated to J.N.T.U, Anantapur)

Kurnool – 518002

Andhra Pradesh

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY ANANTAPUR

B. Tech II - II sem (C.S.E)

T	Tu	C
3	1	3

(15A04407) MICROPROCESSORS & INTERFACING

Course Objective:

- *Study the instruction set of 8086 microprocessor and its architecture*
- *Learn assembly language programming using 8086 microprocessor*
- *Interfacing 8051, 8255, 8237, and 8259*

Learning Outcome:

- *Program the 8086 microprocessor*
- *Interface the 8086 microprocessor with various devices and program them*

UNIT I

Microprocessors-Evolution and Introduction: Microprocessors and Micro Controllers, Microprocessor based system, Origin of Microprocessor, Classification of Microprocessors, Types of Memory, I/O Devices, Technology Improvements Adapted to Microprocessors and Computers, Introduction to 8085 processor, Architecture of 8085, Microprocessor instructions, classification of instructions, Instruction set of 8085.

Intel 8086 Microprocessor architecture, Features, and Signals: Architecture of 8086, Accessing memory locations, PIN details of 8086.

UNIT II

Addressing Modes, Instruction Set and Programming of 8086: Addressing modes in 8086, Instruction set of 8086, 8086 Assembly Language Programming, Modular Programming.

UNIT III

8086 Interrupts: Interrupt types in 8086, Processing of Interrupts by 8086, Dedicated interrupt types in 8086, Software interrupts-types 00H-FFH, Priority among 8086 interrupts, Interrupt service routines, BIOS interrupts or functional calls, Interrupt handlers, DOS services-INT 21H, System calls-BIOS services.

Memory and I/O Interfacing: Physical memory organization in 8086, Formation of system bus, Interfacing RAM and EPROM chips using only logic gates, Interfacing RAM/ EPROM chips using decoder IC and logic gates, I/O interfacing, Interfacing 8-bit input device with 8086, Interfacing output device using 8086, Interfacing printer with 8086, Interfacing 8-bit and 16-bit I/O devices or ports with 8086, Interfacing CRT terminal with 8086.

UNIT IV

Features and Interfacing of programmable devices for 8086 systems: Intel 8255 programmable peripheral interface, Interfacing switches and LEDS, Interfacing seven segment displays, Traffic light control, Interfacing analog to digital converters, Intel Timer IC 8253, Introduction to serial communication, 8259 programmable controller, 8237 DMA controller.

UNIT V

Introduction to 8051 Micro controllers: Intel‘s MCS-51 series micro controllers, Intel 8051 architecture, Memory organization, Internal RAM structure, Power control in 8051, Stack operation.

8051 Instruction Set and Programming: Introduction, Addressing modes of 8051, Instruction set of 8051, Hardware features of 8051: Introduction, Parallel ports in 8051, External memory interfacing in 8051, Timers, Interrupts, Serial ports.

Interfacing Examples: Interfacing 8255 with 8051, Interfacing of push button switches and LEDS, Interfacing of seven segment displays.

Text Books:

- *"Microprocessor and Interfacing 8086,8051, 8096 and advanced processors", Senthil Kumar, Saravanan, Jeevanathan, Shah, 1st edition, 2nd impression, 2012, Oxford University Press.*
- *"The X86 Microprocessors", Lyla B. Das. , 2010, Pearson.*

Reference Books:

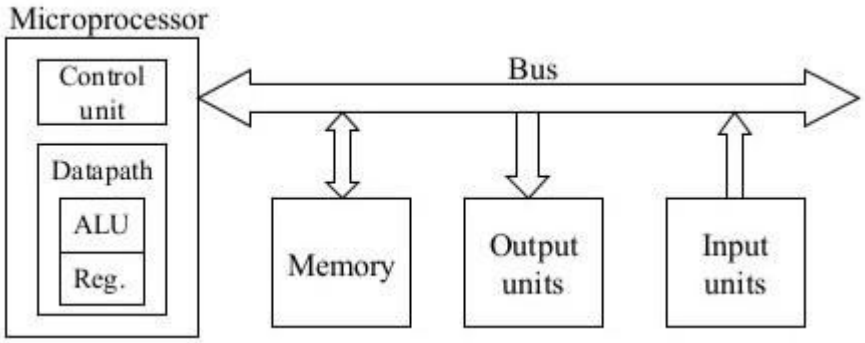
1. *"Microprocessor and Interfacing: Programming and Hardware", Douglas V.Hall, McGrawHill*
2. *"8086 microprocessor: Programming and Interfacing the PC", Kenneth Ayala, Cengage Learning*
3. *"ARM system-on-chip architecture", Steve Furber, Addison-Wesley Professional*
4. *"The Intel Microprocessors", Barry B. Brey, Prentice Hall*

UNIT- I
Microprocessors-Evolution and Introduction

INTRODUCTION:

- The microprocessor is an electronic chip that functions as the central processing unit of a computer.
- Microprocessor based systems with limited resources are called microcomputers.
- Today microprocessor found in almost all consumer electronic devices such as computer printers, washing machines, microwave ovens and in advanced applications such as satellites, radars and flights.

MICROPROCESSOR BASED SYSTEM:



A microprocessor-based system consists primarily of three components-the microprocessor unit (MPU), Memory and I/O (input / output). The MPU is the central player; it communicates with memory and I/O devices. Processes data, and controls timing of all its operations. Memory and 1I0s are integral parts of a microprocessor-based system.

The steps in performing these MPU operations can be summarized as follows (not necessarily in the order listed in every operation):

1. Identify the memory location or the peripheral with its address.
2. Provide timing or synchronization signal.
3. Transfer binary data.

Therefore, the MPU requires three sets of communication lines called buses: the first group of lines, called the address bus, to identify the memory location; the second group, called the data bus, to transfer data; and the third group, called the control lines, for timing signals.

ADDRESS BUS:

If the address size is 4 bits, the microprocessor can identify 16 (24) different memory locations. The addressing is simply a numbering scheme to identify memory registers. For example, a two-digit decimal numbering scheme can identify only 100 items, from 00 to 99. On the other hand, a four-digit numbering scheme can identify 10,000 items, from 0000 to 9999. Thus, the number of bits (address

Lines) used for addressing by the MPU clearly determines the number of memory registers it can identify. If there are n no of address line 2^n memory locations.

DATA BUS:

These lines are used to transfer data and are bidirectional-data can flow either direction. These lines are identified as D_0 to D_n where $_n$ signifies the most significant bit (MSB) of the data bus. Again, the size of the data bus determines how large a binary number can be transferred and processed at a time and thus influences the microprocessor architecture considerably

CONTROL BUS:

These are individual signal lines generated by the MPU to indicate its operations. The MPU generates a specific signal for each of its four operations-Memory Read. Memory Write, 1/0 Read, and I/O Write. These are timing signals that are used to enable, or activate, peripherals. For example, to fetch (or read) an instruction from a memory location, the MPU sends a timing pulse called Memory Read to enable the memory chip.

CLASSIFICATION OF MICROPROCESSORS:

A Microprocessor can be classified depending on many measure here we discuss classification of microprocessor based on characteristics.

The microprocessors are classified into three categories. These are as follows:

1. RISC processors
2. CISC processors
3. Special processors.

Classification of Microprocessor

RISC Processors

RISC is an acronym for reduced instruction set computer. The concept of RISC processor involves an attempt to reduce execution time by simplifying the instruction set of the computer.

The major characteristics of RISC processors are:

- Relatively few instructions
- Relatively few addressing modes
- Memory access limited to load and store instructions
- All operations done within the registers of the CPU
- All operations done within the registers of the CPU
- Fixed-length, easily decoded instruction format
- Single-cycle instruction execution
- Hardwired rather than micro programmed control.

Some architectural features of RISCs are:

- Relatively large number of registers in the processing unit
- Use of overlapped register windows to speedup procedure, call and return

- Efficient instruction pipeline
- Compiler support for efficient translation of high level language programs into machine language programs.

Some popular RISC processors are:

- Power PC: 601, 604, 615, 620
- DEC Alpha: 210642, 211066, 21068, 21164
- MIPS: TS (R10000) RISC Processor
- PA-RISC:HP 7100LC CISC Processor

CISC is an acronym for complex instruction set computer. Major characteristics of a CISC processors are:

- Large number of instruction
- Some instructions that perform specialized tasks
- Variety of addressing modes
- Variable length instruction formats
- Instructions that manipulate control
- Several cycles may be required to execute one instruction.

Examples: Intel's x86 family: Motorola's 680000, 68020, 68030, 68030, 68040 etc.

For many years, Intel is the only major supplier of x86 processor with a few authorized and controlled alternate sources. Today a large number of companies offer compatible processors with no assistance from Intel. They are:

- AMD (Advanced Micro Device) processors
- Cyrix processors
- Texas instrument (TI) processors
- NexGen processors.

All CISC microprocessors that run PC software were derived from the early Intel 8086 or x86 architecture. Today Intel controls the development of CISC microprocessor and other industries produce the Intel compatible processors.

Intel presented a new processor in 1993 called Pentium. With this processor, Intel opened a new era of technology. Day by day, new features are added to the processor and as a result new highly developed processors come out. Some of them are:

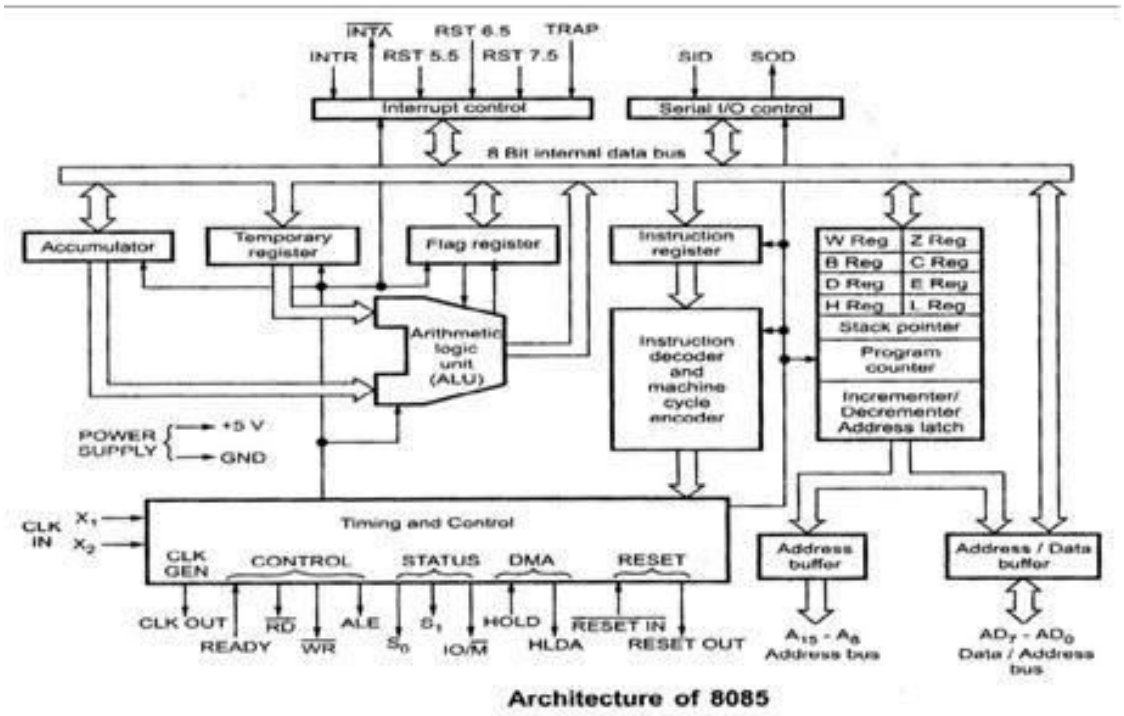
Pentium Pro: This is an optimized processor for high-performance network server implementation.
Pentium III: This is the recent Intel processor in the market, which has both the technology of Pentium Pro and also MMX (multimedia extension) technology. The new PIII system gives very good combination of price and performance and graphic applications.

8085 MICROPROCESSOR:

The Salient Features of 8085 Microprocessor:

- 8085 is an 8 bit microprocessor, manufactured with N-MOS technology.
- It has 16-bit address bus and hence can address up to 2^{16} = 65536 bytes (64KB) memory locations through A0-A15.
- The first 8 lines of address bus and 8 lines of data bus are multiplexed AD0 - AD7.
- Data bus is a group of 8 lines D0 - D7.
- It supports external interrupt request.8085 consists of 16 bit program counter (PC)
- and stack pointer (SP).
- Six 8-bit general purpose register arranged in pairs: BC, DE, HL.

Internal Architecture of 8085:

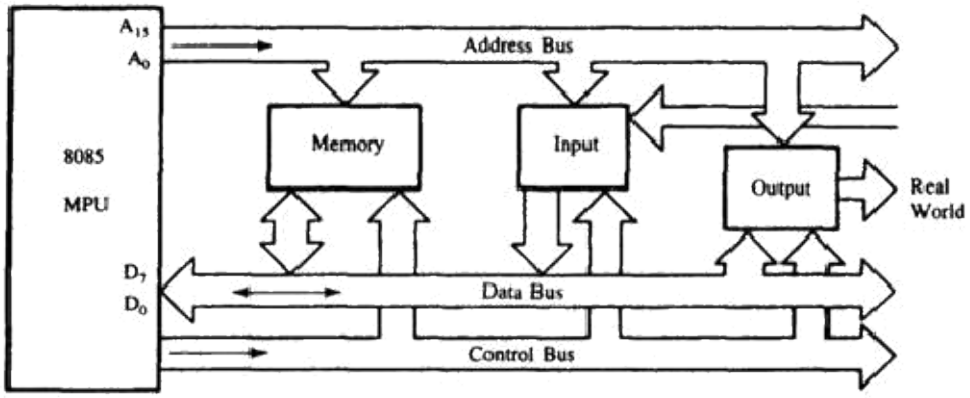


Architecture of 8085

8085 Bus Structure:

Address Bus:

- The address bus is a group of 16 lines generally identified as A0 to A15.
- The address bus is unidirectional: bits flow in one direction-from the MPU to peripheral devices.
- The MPU uses the address bus to perform the first function: identifying a peripheral or a memory location.



Data Bus:

- The data bus is a group of eight lines used for data flow.
- These lines are bi-directional - data flow in both directions between the MPU and memory and peripheral devices.
- The MPU uses the data bus to perform the second function: transferring binary information.
- The eight data lines enable the MPU to manipulate 8-bit data ranging from 00 to FF (28 = 256 numbers).
- The largest number that can appear on the data bus is 11111111.

Control Bus:

- The control bus carries synchronization signals and providing timing signals.
- The MPU generates specific control signals for every operation it performs. These signals are used to identify a device type with which the MPU wants to communicate.

Registers of 8085:

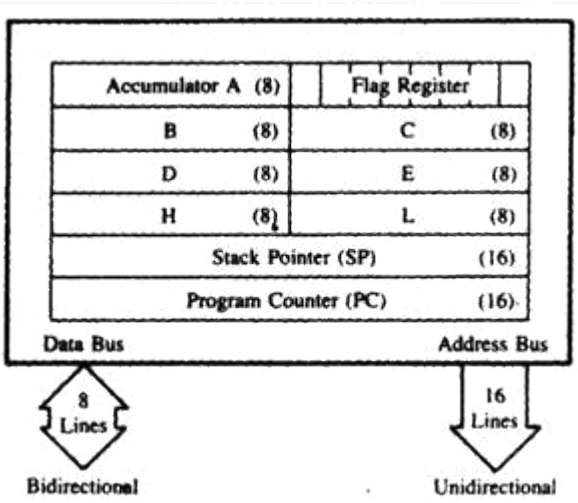
- The 8085 have six general-purpose registers to store 8-bit data during program execution.
- These registers are identified as B, C, D, E, H, and L.
- They can be combined as register pairs-BC, DE, and HL-to perform some 16-bit operations.

Accumulator (A):

- The accumulator is an 8-bit register that is part of the arithmetic/logic unit (ALU).
- This register is used to store 8-bit data and to perform arithmetic and logical operations.
- The result of an operation is stored in the accumulator.

Flags:

- The ALU includes five flip-flops that are set or reset according to the result of an operation.
- The microprocessor uses the flags for testing the data conditions.
- They are Zero (Z), Carry (CY), Sign (S), Parity (P), and Auxiliary Carry (AC) flags. The most commonly used flags are Sign, Zero, and Carry.



The bit position for the flags in flag register is,

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
S	Z		AC		P		CY

1. **Sign Flag (S):**
   - After execution of any arithmetic and logical operation, if D7 of the result is 1, the sign flag is set. Otherwise it is reset.
   - D7 is reserved for indicating the sign; the remaining is the magnitude of number.
   - If D7 is 1, the number will be viewed as negative number. If D7 is 0, the number will be viewed as positive number.
2. **Zero Flag (z):**

If the result of arithmetic and logical operation is zero, then zero flag is set otherwise it is reset.

3. Auxiliary Carry Flag (AC):

If D3 generates any carry when doing any arithmetic and logical operation, this flag is set. Otherwise it is reset.

4. Parity Flag (P):

If the result of arithmetic and logical operation contains even number of 1's then this flag will be set and if it is odd number of 1's it will be reset.

5. Carry Flag (CY):

If any arithmetic and logical operation result any carry then carry flag is set otherwise it is reset.

Arithmetic and Logic Unit (ALU):

- It is used to perform the arithmetic operations like addition, subtraction, multiplication, division, increment and decrement and logical operations like AND, OR and EX-OR.
- It receives the data from accumulator and registers.
- According to the result it set or reset the flags.

Program Counter (PC):

- This 16-bit register sequencing the execution of instructions.
- It is a memory pointer. Memory locations have 16-bit addresses, and that is why this is a 16-bit register.
- The function of the program counter is to point to the memory address of the next instruction to be executed.
- When an opcode is being fetched, the program counter is incremented by one to point to the next memory location.

Stack Pointer (SP):

- The stack pointer is also a 16-bit register used as a memory pointer.
- It points to a memory location in R/W memory, called the stack.
- The beginning of the stack is defined by loading a 16-bit address in the stack pointer (register).

Temporary Register: It is used to hold the data during the arithmetic and logical operations.
Instruction Register: When an instruction is fetched from the memory, it is loaded in the instruction register.

Instruction Decoder: It gets the instruction from the instruction register and decodes the instruction. It identifies the instruction to be performed.

Serial I/O Control: It has two control signals named SID and SOD for serial data transmission.

Timing and Control unit:

- It has three control signals ALE, RD (Active low) and WR (Active low) and three status signals IO/M(Active low), S0 and S1.
- ALE is used for provide control signal to synchronize the components of microprocessor and timing for instruction to perform the operation.
- RD (Active low) and WR (Active low) are used to indicate whether the operation is reading the data from memory or writing the data into memory respectively.
- IO/M(Active low) is used to indicate whether the operation is belongs to the memory or peripherals.

IO/M(Active Low)	S1	S2	Data Bus Status(Output)
0	0	0	Halt
0	0	1	Memory WRITE
0	1	0	Memory READ
1	0	1	IO WRITE
1	1	0	IO READ
0	1	1	Opcode fetch
1	1	1	Interrupt acknowledge

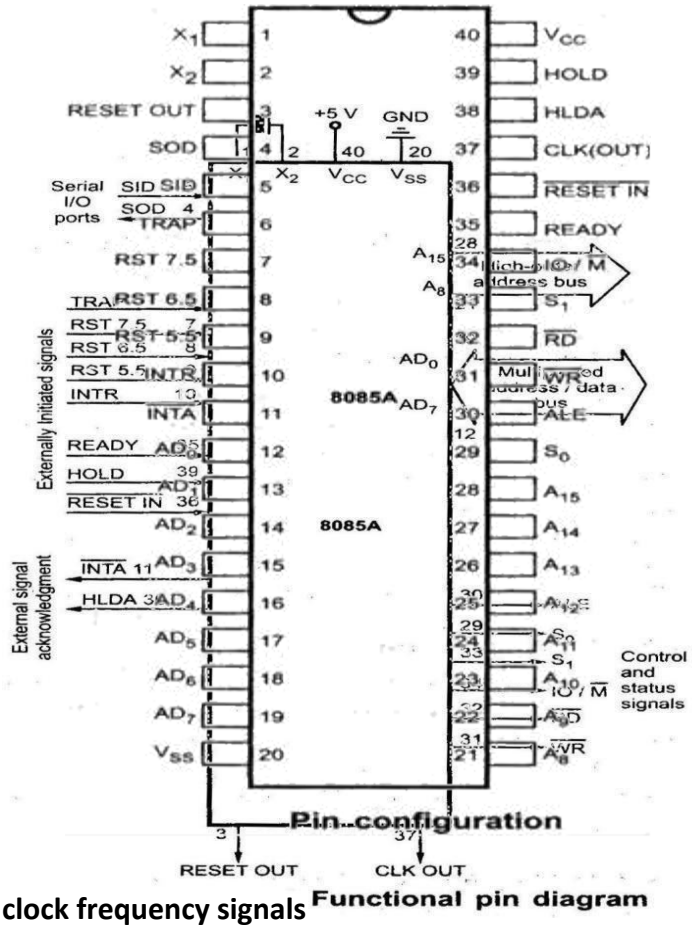
Interrupt Control Unit:

- It receives hardware interrupt signals and sends an acknowledgement for receiving the interrupt signal.

Pin Diagram and Pin Description Of 8085

8085 is a 40 pin IC, DIP package. The signals from the pins can be grouped as follows

1. Power supply and clock signals
2. Address bus
3. Data bus
4. Control and status signals
5. Interrupts and externally initiated signals
6. Serial I/O ports



1. Power supply and clock frequency signals

- Vcc + 5 volt power supply
- Vss Ground
- X1, X2: 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 externally.
- CLK (output)-Clock Output is used as the system clock for peripheral and devices interfaced with the microprocessor.

2. **Address Bus:**

- A8 - A15 (output; 3-state)
- It carries the most significant 8 bits of the memory address or the 8 bits of the I/O address;

3. **Multiplexed Address / Data Bus:**

- AD0 - AD7 (input/output; 3-state)
- These multiplexed set of lines used to carry the lower order 8 bit address as well as data bus.
- During the opcode fetch operation, in the first clock cycle, the lines deliver the lower order address A0 - A7.
- In the subsequent IO / memory, read / write clock cycle the lines are used as data bus.
- The CPU may read or write out data through these lines.

4. **Control and Status signals:**
   - ALE (output) - Address Latch Enable.
   - This signal helps to capture the lower order address presented on the multiplexed address / data bus.
   - RD (output 3-state, active low) - Read memory or IO device.
   - This indicates that the selected memory location or I/O device is to be read and that the data bus is ready for accepting data from the memory or I/O device.
   - WR (output 3-state, active low) - Write memory or IO device.
   - This indicates that the data on the data bus is to be written into the selected memory location or I/O device.
   - IO/M (output) - Select memory or an IO device.
   - This status signal indicates that the read / write operation relates to whether the memory or I/O device.
   - It goes high to indicate an I/O operation.
   - It goes low for memory operations.
5. **Status Signal**
   - It is used to know the type of current operation of the microprocessor.

IO/M (Active Low)	S1	S2	Data Bus Status (Output)
0	0	0	Halt
0	0	1	Memory WRITE
0	1	0	Memory READ
1	0	1	IO WRITE
1	1	0	IO READ
0	1	1	Opcode fetch
1	1	1	Interrupt acknowledge

6. **Interrupts and externally initiated operations:**
   - 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,
   - On receipt of an interrupt, the microprocessor acknowledges the interrupt by the active low INTA (Interrupt Acknowledge) signal.

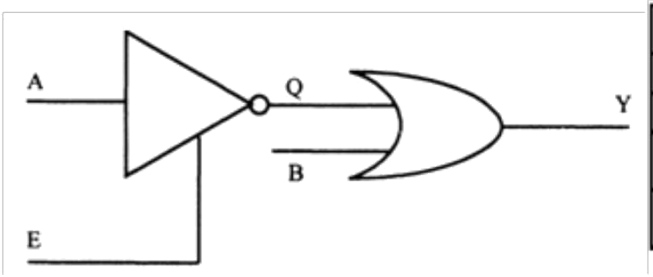
Reset In (input, active low)
   - This signal is used to reset the microprocessor.
   - The program counter inside the microprocessor is set to zero.
   - The buses are tri-stated.

Reset Out (Output)
   - It indicates CPU is being reset.
   - Used to reset all the connected devices when the microprocessor is reset

7. **Direct Memory Access (DMA):**

Tri state devices:

   - 3 output states are high & low states and additionally a high impedance state.
   - When enable E is high the gate is enabled and the output Q can be 1 or 0 (if A is 0, Q is Z1, otherwise Q is 0). However, when E is low the gate is disabled and the output Q enters into a high impedance state.



E	A	Q	State
1(high)	0	1	High
1	1	0	Low
0(low)	0	0	High impedance
0	1	0	High impedance

- For both high and low states, the output Q draws a current from the input of the OR gate.
- When E is low, Q enters a high impedance state; high impedance means it is electrically isolated from the OR gate's input, though it is physically connected. Therefore, it does not draw any current from the OR gate's input.
- When 2 or more devices are connected to a common bus, to prevent the devices from interfering with each other, the tristate gates are used to disconnect all devices except the one that is communicating at a given instant.
- The CPU controls the data transfer operation between memory and I/O device. Direct Memory Access operation is used for large volume data transfer between memory and an I/O device directly.
- The CPU is disabled by tri-stating its buses and the transfer is effected directly by external control circuits.
- HOLD signal is generated by the DMA controller circuit. On receipt of this signal, the microprocessor 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.

READY (input)

- 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.
- The microprocessor enters into WAIT state while the READY pin is disabled.

8. Single Bit Serial I/O ports:

- SID (input) - Serial input data line
- SOD (output) - Serial output data line
- These signals are used for serial communication.

CLASSIFICATION OF MICROPROCESSORS:

Microprocessor instructions can be classified into following five categories:

1. Based on functionality
2. Based on length
3. Based on addressing modes

8085 Addressing modes with example and Instruction Size:

ADDRESSING MODES:

The various ways of specifying data (or operands) for instructions are called as **addressing modes.** The 8085 addressing modes are classified into following types:

1. Immediate addressing mode
2. Direct addressing mode
3. Register addressing mode
4. Register indirect addressing mode
5. Implicit addressing mode

(i) Immediate Addressing mode

In this mode operand is a part of the instruction itself is known as Immediate Addressing mode. If the immediate data is 8-bit, the instruction will be of two bytes. If the immediate data is 16 bit, the instruction is of 3 bytes.

Ex: (1). ADI DATA ; Add immediate the data to the contents of the accumulator.
(2).LXIH 8500H : Load immediate the H-L pair with the operand 8500H
(3). MVI 08H ; Move the data 08 H immediately to the accumulator
(4). SUI 05H ; Subtract immediately the data 05H from the accumulator

(ii) Direct Addressing mode:

The mode of addressing in which the 16-bit address of the operand is directly available in the instruction itself is called Direct Addressing mode. i.e., the address of the operand is available in the instruction itself. This is a 3-byte instruction.

Ex: (1). LDA 9525H; Load the contents of memory location into Accumulator.
(2). STA 8000H; Store the contents of the Accumulator in the location 8000H
(3). IN 01H; Read the data from port whose address is 01H.

(iii). Register addressing modes:

In this mode the operands are microprocessor registers only. i.e. the operation is performed within various registers of the microprocessor.

Ex: (1). MOV A, B; Move the contents of B register to A register.
(2). SUB D; Subtract the contents of D register from Accumulator.
(3). ADD B, C; Add the contents of C register to the contents of B register.

(iv). Register indirect addressing modes:

The 16-bit address location of the operand stored in a register pair (H-L) is given in the instruction. The address of the operand is given in an indirect way with the help of a register pair. So it is called Register indirect addressing mode.

Ex: (1). LXIH 9570H : Load immediate the H-L pair with the address of the location 9570H
MOV A, M : Move the contents of the memory location pointed by the H-L pair to accumulator

(v). Implicit Addressing mode:

The mode of instruction which do not specify the operand in the instruction but it is implicated, is known as implicit addressing mode. i.e., the operand is supposed to be present generally in accumulator.

Ex: (1).CMA; complement the contents of Accumulator

(2).CMC; Complement carry

(3). RLC; Rotate Accumulator left by one bit

(4). RRC; Rotate Accumulator right by one bit

(5). STC; Set carry.

Instruction and data formats:

The format of a typical instruction is composed of two parts: an operation code or op-code and an operand. Every instruction needs an opcode to specify the operation of the instruction is and then an operand that gives the appropriate data needed for that particular operation code.

Depending upon the size of machine codes, the 8085 instructions are classified into three types.

(a) One byte (single) instructions.

(b)Two byte instructions.

(c) Three byte instructions.

One-byte instructions: A 1 byte instruction include the opcode and the operand in the 8 bits only which is one byte.

Ex: 1. MOV C, A Hex code = 4FH (one byte)

2. ADD B Hex code = 80H (one byte)
3. CMA Hex code = 2FH (one byte)

Two-byte instructions: The two byte instruction is one which contains an 8-bit op-code and 8-bit operand (Data).

Ex: 1. MVI A, 09 Hex code = 3E, 09 (two bytes)

2. ADD B, 07 Hex code = 80, 07 (two bytes)
3. SUB A, 05 Hex code = 97, 05 (two bytes)

Three-byte instructions: In a three byte instruction the first byte is opcode and second and third bytes are operands i.e. 16-bit data or 16-bit address.

1. LDA 8509Hex code = 3A, 09, 85 (Three bytes)
2. LXI 2500Hex code = 21, 00, 25 (Three bytes)
3. STA 2600 Hex code = 32, 00, 26 (Three bytes)

DATA FORMATS:

The 8085 is an 8-bit microprocessor which process only on binary numbers. Since t is very difficult to understand these numbers by a common user, So we are using different data formats to code these binary numbers. ASCII, BCD, signed integers and unsigned integers are some of the data formats. The ASCII code is a 7-bit alpha-numeric code that represents decimal numbers, English alphabets and certain special characters. The ASCII stands for “American Standard code for Information Interchange”.

The term BCD stands for binary coded decimal, used for the decimal numbers from 0-9. An 8-bit register can store two BCD numbers. A signed integer can be both either negative or positive number. In 8085 microprocessor the most significant bit is used for the sign. Here 0 denotes positive sign and 1 denotes the negative sign. An integer without a sign is represented by all the 8-bits in a microprocessor register. So, the largest number that can be processed at one time is FFH. The numbers larger than 8-bits like 16, 24, 32 bits can be processed by dividing them in groups of 8-bits.

Instruction Set 8085:

1. Control
2. Logical
3. Branching
4. Arithmetic
5. Data Transfer

Control Instructions

Opcode	Operand	Explanation of Instruction	Description
NOP	**none**	No operation	No operation is performed. The instruction is fetched and decoded. However no operation is executed. Example: NOP
HLT	**none**	Halt and enter wait state	The CPU finishes executing the current instruction and halts any further execution. An interrupt or reset is necessary to exit from the halt state. Example: HLT
DI	**none**	Disable interrupts	The interrupt enable flip-flop is reset and all the interrupts except the TRAP are disabled. No flags are affected. Example: DI
EI	**none**	Enable interrupts	The interrupt enable flip-flop is set and all interrupts are enabled. No flags are affected. After a system reset or the acknowledgement of an interrupt, the interrupt enable flipflop is reset, thus disabling the interrupts. This instruction is necessary to reenable the interrupts (except TRAP). Example: EI
RIM	**none**	Read interrupt mas	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. Example: RIM

SIM	**none**	Set interrupt mask	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 follows. Example: SIM

LOGICAL INSTRUCTIONS

Opcode	Operand	Explanation of Instruction	Description

CMP	**R**
	
	M
		Compare register or memory with accumulator	The contents of the operand (register or memory) are M compared with the contents of the accumulator. Both contents are preserved . The result of the comparison is shown by setting the flags of the PSW as follows:
			
			if (A) < (reg/mem): carry flag is set
			if (A) = (reg/mem): zero flag is set
			if (A) > (reg/mem): carry and zero flags are reset
			
			Example: CMP B or CMP M

CPI	**8-bit data**	Compare immediate with accumulator	The second byte (8-bit data) is compared with the contents of the accumulator. The values being compared remain unchanged. The result of the comparison is shown by setting the flags of the PSW as follows:
			
			if (A) < data: carry flag is set
			if (A) = data: zero flag is set
			if (A) > data: carry and zero flags are reset
			
			Example: CPI 89H

ANA	**R** **M**	Logical AND register or memory with accumulator	The contents of the accumulator are logically ANDed with M the contents of the operand (register or memory), and the result is placed in the accumulator. If the operand is a memory location, its address is specified by the contents of HL registers. S, Z, P are modified to reflect the result of the operation. CY is reset. AC is set. Example: ANA B or ANA M

ANI	**8-bit**	Logical	AND	The contents of the accumulator are logically ANDed with the

	data	immediate with accumulator	8-bit data (operand) and the result is placed in the accumulator. S, Z, P are modified to reflect the result of the operation. CY is reset. AC is set. Example: ANI 86H
XRA	**R** **M**	Exclusive OR register or memory with accumulator	The contents of the accumulator are Exclusive ORed with M the contents of the operand (register or memory), and the result is placed in the accumulator. If the operand is a memory location, its address is specified by the contents of HL registers. S, Z, P are modified to reflect the result of the operation. CY and AC are reset. Example: XRA B or XRA M
XRI	**8-bit data**	Exclusive OR immediate with accumulator	The contents of the accumulator are Exclusive ORed with the 8-bit data (operand) and the result is placed in the accumulator. S, Z, P are modified to reflect the result of the operation. CY and AC are reset. Example: XRI 86H
ORA	**R** **M**	Logical OR register or memory with accumulator	The contents of the accumulator are logically ORed with M the contents of the operand (register or memory), and the result is placed in the accumulator. If the operand is a memory location, its address is specified by the contents of HL registers. S, Z, P are modified to reflect the result of the operation. CY and AC are reset. Example: ORA B or ORA M
ORI	**8-bit data**	Logical OR immediate with accumulator	The contents of the accumulator are logically ORed with the 8-bit data (operand) and the result is placed in the accumulator. S, Z, P are modified to reflect the result of the operation. CY and AC are reset. Example: ORI 86H
RLC	**none**	Rotate accumulator left	Each binary bit of the accumulator is rotated left by one position. Bit D7 is placed in the position of D0 as well as in the Carry flag.
			CY is modified according to bit D7. S, Z, P, AC are not affected.

			Example: RLC
RRC	**none**	Rotate accumulator right	Each binary bit of the accumulator is rotated right by one position. Bit D0 is placed in the position of D7 as well as in the Carry flag. CY is modified according to bit D0. S, Z, P, AC are not affected. Example: RRC
RAL	**none**	Rotate accumulator left through carry	Each binary bit of the accumulator is rotated left by one position through the Carry flag. Bit D7 is placed in the Carry flag, and the Carry flag is placed in the least significant position D0. CY is modified according to bit D7. S, Z, P, AC are not affected. Example: RAL
RAR	**none**	Rotate accumulator right through carry	Each binary bit of the accumulator is rotated right by one position through the Carry flag. Bit D0 is placed in the Carry flag, and the Carry flag is placed in the most significant position D7. CY is modified according to bit D0. S, Z, P, AC are not affected. Example: RAR
CMA	**none**	Complement accumulator	The contents of the accumulator are complemented. No flags are affected. Example: CMA
CMC	**none**	Complement carry	The Carry flag is complemented. No other flags are affected. Example: CMC
STC	**none**	Set Carry	Set Carry Example: STC

BRANCHING INSTRUCTIONS:

<table>
<tr><th>Opcode</th><th>Operand</th><th>Explanation of Instruction</th><th>Description</th></tr>
<tr><td>JMP</td><td>16-bit address</td><td>Jump unconditionally</td><td>The program sequence is transferred to the memory location specified by the 16-bit address given in the operand.

Example: JMP 2034H or JMP XYZ</td></tr>
<tr><td><table>
<tr><th>Opcode</th><th>Description</th><th>Flag Status</th></tr>
<tr><td>JC</td><td>Jump on Carry</td><td>CY = 1</td></tr>
<tr><td>JNC</td><td>Jump on no Carry</td><td>CY = 0</td></tr>
<tr><td>JP</td><td>Jump on positive</td><td>S = 0</td></tr>
<tr><td>JM</td><td>Jump on minus</td><td>S = 1</td></tr>
<tr><td>JZ</td><td>Jump on zero</td><td>Z = 1</td></tr>
<tr><td>JNZ</td><td>Jump on no zero</td><td>Z = 0</td></tr>
<tr><td>JPE</td><td>Jump on parity even</td><td>P = 1</td></tr>
<tr><td>JPO</td><td>Jump on parity odd</td><td>P = 0</td></tr>
</table></td><td>16-bit address</td><td>Jump conditionally</td><td>The program sequence is transferred to the memory location specified by the 16-bit address given in the operand based on the specified flag of the PSW as described below.

Example: JZ 2034H or JZ XYZ</td></tr>
<tr><td><table>
<tr><th>Opcode</th><th>Description</th><th>Flag Status</th></tr>
<tr><td>CC</td><td>Call on Carry</td><td>CY = 1</td></tr>
<tr><td>CNC</td><td>Call on no Carry</td><td>CY = 0</td></tr>
<tr><td>CP</td><td>Call on positive</td><td>S = 0</td></tr>
<tr><td>CM</td><td>Call on minus</td><td>S = 1</td></tr>
<tr><td>CZ</td><td>Call on zero</td><td>Z = 1</td></tr>
<tr><td>CNZ</td><td>Call on no zero</td><td>Z = 0</td></tr>
<tr><td>CPE</td><td>Call on parity even</td><td>P = 1</td></tr>
<tr><td>CPO</td><td>Call on parity odd</td><td>P = 0</td></tr>
</table></td><td>16-bit address</td><td>Unconditional subroutine call</td><td>The program sequence is transferred to the memory location specified by the 16-bit address given in the operand. Before the transfer, the address of the next instruction after CALL (the contents of the program counter) is pushed onto the stack.

Example: CALL 2034H or CALL XYZ</td></tr>
<tr><td>RET</td><td>none</td><td>Return from subroutine unconditionally</td><td>The program sequence is transferred from the subroutine to the calling program. The two bytes from the top of the stack are</td></tr>
</table>

<table>
<tr><td></td><td></td><td></td><td>copied into the program counter,and program execution begins at the new address.

Example: RET</td></tr>
<tr><td><table>
<tr><th>Opcode</th><th>Description</th><th>Flag Status</th></tr>
<tr><td>RC</td><td>Return on Carry</td><td>CY = 1</td></tr>
<tr><td>RNC</td><td>Return on no Carry</td><td>CY = 0</td></tr>
<tr><td>RP</td><td>Return on positive</td><td>S = 0</td></tr>
<tr><td>RM</td><td>Return on minus</td><td>S = 1</td></tr>
<tr><td>RZ</td><td>Return on zero</td><td>Z = 1</td></tr>
<tr><td>RNZ</td><td>Return on no zero</td><td>Z = 0</td></tr>
<tr><td>RPE</td><td>Return on parity even</td><td>P = 1</td></tr>
<tr><td>RPO</td><td>Return on parity odd</td><td>P = 0</td></tr>
</table></td><td>none</td><td>Return from subroutine conditionally</td><td>The program sequence is transferred from the subroutine to the calling program based on the specified flag of the PSW as described below. The two bytes from the top of the stack are copied into the program counter, and program execution begins at the new address.

Example: RZ</td></tr>
<tr><td>PCHL</td><td>none</td><td>Load program counter with HL contents</td><td>The contents of registers H and L are copied into the program counter. The contents of H are placed as the high-order byte and the contents of L as the low-order byte.

Example: PCHL</td></tr>
<tr><td>RST</td><td>0-7</td><td>Restart</td><td>The RST instruction is equivalent to a 1-byte call instruction to one of eight memory locations depending upon the number. The instructions are generally used in conjunction with interrupts and inserted using external hardware. However these can be used as software instructions in a program to transfer program execution to one of the eight locations. The addresses are:</td></tr>
</table>

Instruction	Restart Address
RST 0	**0000H**
RST1	**0008H**
RST 2	**0010H**
RST 3	**0018H**
RST 4	**0020H**
RST 5	**0028H**
RST 6	**0030H**
RST 7	**0038H**

The 8085 has four additional interrupts and these interrupts generate RST instructions internally and thus do not require any external hardware. These instructions and their Restart addresses are:

Interrupt	Restart Address
TRAP	**0024H**
RST 5.5	**002CH**
RST 6.5	**0034H**
RST 7.5	**003CH**

ARITHMETIC INSTRUCTIONS:

Opcode	Operand	Explanation of Instruction	Description
ADD	**R** **M**	Add register or memory, to accumulator	The contents of the operand (register or memory) are added to the contents of the accumulator and the result is stored in the accumulator. If the operand is a memory location, its location is specified by the contents of the HL registers. All flags are modified to reflect the result of the addition. Example: ADD B or ADD M
ADC	**R** **M**	Add register to accumulator with carry	The contents of the operand (register or memory) and M the Carry flag are added to the contents of the accumulator and the result is stored in the accumulator. If the operand is a memory location, its location is specified by the contents of

			the HL registers. All flags are modified to reflect the result of the addition. Example: ADC B or ADC M
ADI	**8-bit data**	Add immediate to accumulator	The 8-bit data (operand) is added to the contents of the accumulator and the result is stored in the accumulator. All flags are modified to reflect the result of the addition. Example: ADI 45H
ACI	**8-bit data**	Add immediate to accumulator with carry	The 8-bit data (operand) and the Carry flag are added to the contents of the accumulator and the result is stored in the accumulator. All flags are modified to reflect the result of the addition. Example: ACI 45H
LXI	**Reg. pair, 16-bit data**	Load register pair immediate	The instruction loads 16-bit data in the register pair designated in the operand. Example: LXI H, 2034H or LXI H, XYZ
DAD	**Reg. pair**	Add register pair to H and L registers	The 16-bit contents of the specified register pair are added to the contents of the HL register and the sum is stored in the HL register. The contents of the source register pair are not altered. If the result is larger than 16 bits, the CY flag is set. No other flags are affected. Example: DAD H
SUB	**R** **M**	Subtract register or memory from accumulator	The contents of the operand (register or memory) are subtracted from the contents of the accumulator, and the result is stored in the accumulator. If the operand is a memory location, its location is specified by the contents of the HL registers. All flags are modified to reflect the result of the subtraction.
			Example: SUB B or SUB M
SBB	**R**	Subtract source and borrow from	The contents of the operand (register or memory) and M the Borrow flag are subtracted from the contents of the

	M	accumulator	accumulator and the result is placed in the accumulator. If the operand is a memory location, its location is specified by the contents of the HL registers. All flags are modified to reflect the result of the subtraction. Example: SBB B or SBB M
SUI	**8-bit data**	Subtract immediate from accumulator	The 8-bit data (operand) is subtracted from the contents of the accumulator and the result is stored in the accumulator. All flags are modified to reflect the result of the subtraction. Example: SUI 45H
SBI	**8-bit data**	Subtract immediate from accumulator with borrow	The contents of register H are exchanged with the contents of register D, and the contents of register L are exchanged with the contents of register E. Example: XCHG
INR	**R** **M**	Increment register or memory by 1	The contents of the designated register or memory) are incremented by 1 and the result is stored in the same place. If the operand is a memory location, its location is specified by the contents of the HL registers. Example: INR B or INR M
INX	**R**	Increment register pair by 1	The contents of the designated register pair are incremented by 1 and the result is stored in the same place. Example: INX H
DCR	**R** **M**	Decrement register or memory by 1	The contents of the designated register or memory are M decremented by 1 and the result is stored in the same place. If the operand is a memory location, its location is specified by the contents of the HL registers.
			Example: DCR B or DCR M
DCX	**R**	Decrement register pair by 1	The contents of the designated register pair are decremented by 1 and the result is stored in the same place. Example: DCX H

DAA	**none**	Decimal adjust accumulator	The contents of the accumulator are changed from a binary value to two 4-bit binary coded decimal (BCD) digits. This is the only instruction that uses the auxiliary flag to perform the binary to BCD conversion, and the conversion procedure is described below. S, Z, AC, P, CY flags are altered to reflect the results of the operation. If the value of the low-order 4-bits in the accumulator is greater than 9 or if AC flag is set, the instruction adds 6 to the low-order four bits. If the value of the high-order 4-bits in the accumulator is greater than 9 or if the Carry flag is set, the instruction adds 6 to the high-order four bits. Example: DAA

DATA TRANSFER INSTRUCTIONS

Opcode	Operand	Explanation of Instruction	Description
MOV	**Rd, Rs** **M, Rs** **Rd, M**	Copy from source(Rs) to destination(Rd)	This instruction copies the contents of the source register into the destination register; the contents of the source register are not altered. If one of the operands is a memory location, its location is specified by the contents of the HL registers. Example: MOV B, C or MOV B, M
MVI	**Rd, data** **M, data**	Move immediate 8-bit	The 8-bit data is stored in the destination register or memory. If the operand is a memory location, its location is specified by the contents of the HL registers.
			Example: MVI B, 57H or MVI M, 57H
LDA	**16-bit address**	Load accumulator	The contents of a memory location, specified by a 16-bit address in the operand, are copied to the accumulator. The contents of the source are not altered. Example: LDA 2034H

LDAX	**B/D Reg. pair**	Load accumulator indirect	The contents of the designated register pair point to a memory location. This instruction copies the contents of that memory location into the accumulator. The contents of either the register pair or the memory location are not altered. Example: LDAX B
LXI	**Reg. pair, 16-bit data**	Load register pair immediate	The instruction loads 16-bit data in the register pair designated in the operand. Example: LXI H, 2034H or LXI H, XYZ
LHLD	**16-bit address**	Load H and L registers direct	The instruction copies the contents of the memory location pointed out by the 16-bit address into register L and copies the contents of the next memory location into register H. The contents of source memory locations are not altered. Example: LHLD 2040H
STA	**16-bit address**	16-bit address	The contents of the accumulator are copied into the memory location specified by the operand. This is a 3-byte instruction, the second byte specifies the low-order address and the third byte specifies the high-order address. Example: STA 4350H
STAX	**Reg. pair**	Store accumulator indirect	The contents of the accumulator are copied into the memory location specified by the contents of the operand (register pair). The contents of the accumulator are not altered.
			Example: STAX B
SHLD	**16-bit address**	Store H and L registers direct	The contents of register L are stored into the memory location specified by the 16-bit address in the operand and the contents of H register are stored into the next memory location by incrementing the operand. The contents of registers HL are not altered. This is a 3-byte instruction, the second byte specifies the low-order address and the third byte specifies the high-order address.

			Example: SHLD 2470H
XCHG	**none**	Exchange H and L with D and E	The contents of register H are exchanged with the contents of register D, and the contents of register L are exchanged with the contents of register E. Example: XCHG
SPHL	**none**	Copy H and L registers to the stack pointer	The instruction loads the contents of the H and L registers into the stack pointer register, the contents of the H register provide the high-order address and the contents of the L register provide the low-order address. The contents of the H and L registers are not altered. Example: SPHL
XTHL	**none**	Exchange H and L with top of stack	The contents of the L register are exchanged with the stack location pointed out by the contents of the stack pointer register. The contents of the H register are exchanged with the next stack location (SP+1); however, the contents of the stack pointer register are not altered. Example: XTHL
PUSH	**Reg. pair**	Push register pair onto stack	The contents of the register pair designated in the operand are copied onto the stack in the following sequence. The stack pointer register is decremented and the contents of the highorder register (B, D, H, A) are copied into that location. The stack pointer register is decremented again and the contents of the low-order register (C, E, L, flags) are
			copied to that location. Example: PUSH B or PUSH A
POP	**Reg. pair**	Pop off stack to register pair	The contents of the memory location pointed out by the stack pointer register are copied to the low-order register (C, E, L, status flags) of the operand. The stack pointer is incremented by 1 and the contents of that memory location are copied to the high-order register (B, D, H, A) of the operand. The stack pointer register is again incremented by

			1. Example: POP H or POP A
OUT	**8-bit port address**	Output data from accumulator to a port with 8-bit address	The contents of the accumulator are copied into the I/O port specified by the operand. Example: OUT F8H
IN	**8-bit port address**	Input data to accumulator from a port with 8-bit address	The contents of the input port designated in the operand are read and loaded into the accumulator. Example: IN 8CH

Intel 8086 Microprocessor architecture, Features, and Signals:

Overview or Features of 8086

- It is a 16-bit Microprocessor (µp).It's ALU, internal registers works with 16bit binary word.
- 8086 has a 20 bit address bus can access up to 2^{20} = 1 MB memory locations.
- 8086 has a 16bit data bus. It can read or write data to a memory/port either 16bits or 8 bit at a time.
- It can support up to 64K I/O ports.
- It provides 14, 16 -bit registers.
- Frequency range of 8086 is 6-10 MHz
- It has multiplexed address and data bus AD0- AD15 and A16 – A19.
- It requires single phase clock with 33% duty cycle to provide internal timing.
- It can prefetch up to 6 instruction bytes from memory and queues them in order to speed up instruction execution.
- It requires +5V power supply.
- A 40 pin dual in line package.
- 8086 is designed to operate in two modes, Minimum mode and Maximum mode.

Register Organization of 8086:

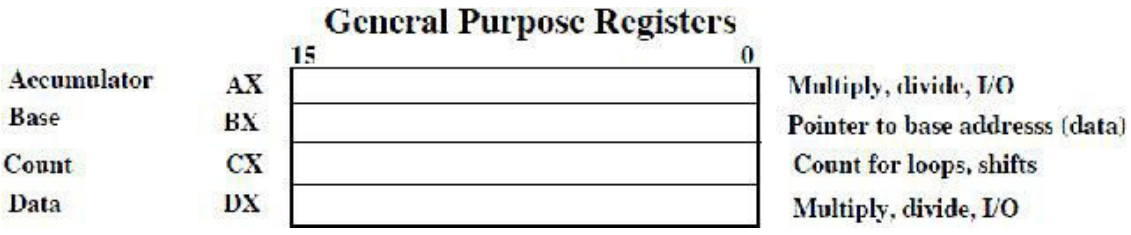
The 8086 microprocessor has a total of fourteen registers that are accessible to the programmer. It is divided into four groups. They are:

- Four General purpose registers
- Four Index/Pointer registers
- Four Segment registers
- Two Other registers

General purpose registers:

Accumulator register consists of two 8-bit registers AL and AH, which can be combined together and used as a 16-bit register AX. AL in this case contains the low order byte of the word, and AH contains the high-order byte. Accumulator can be used for I/O operations and string manipulation.

Base register consists of two 8-bit registers BL and BH, which can be combined together and used as a 16-bit register BX. BL in this case contains the low-order byte of the word, and BH contains the high-order byte. BX register usually contains a data pointer used for based, based indexed or register indirect addressing.



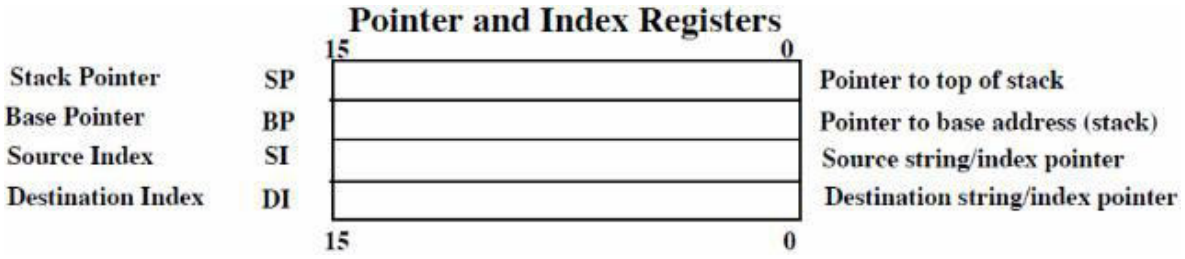
Count register consists of two 8-bit registers CL and CH, which can be combined together and used as a 16-bit register CX. When combined, CL register contains the low order byte of the word, and CH contains the high-order byte. Count register can be used in Loop, shift/rotate instructions and as a counter in string manipulation

Data register consists of two 8-bit registers DL and DH, which can be combined together and used as a 16-bit register DX. When combined, DL register contains the low order byte of the word, and DH contains the high-order byte. Data register can be used as a port number in I/O operations. In integer 32-bit multiply and divide instruction the DX register contains high-order word of the initial or resulting number.

Index or Pointer Registers

These registers can also be called as Special Purpose registers.

Stack Pointer (SP) is a 16-bit register pointing to program stack, i.e. it is used to hold the address of the top of stack. The stack is maintained as a LIFO with its bottom at the start of the stack segment (specified by the SS segment register).Unlike the SP register, the BP can be used to specify the offset of other program segments.



Base Pointer (BP) is a 16-bit register pointing to data in stack segment. It is usually used by subroutines to locate variables that were passed on the stack by a calling program. BP register is usually used for based, based indexed or register indirect addressing.

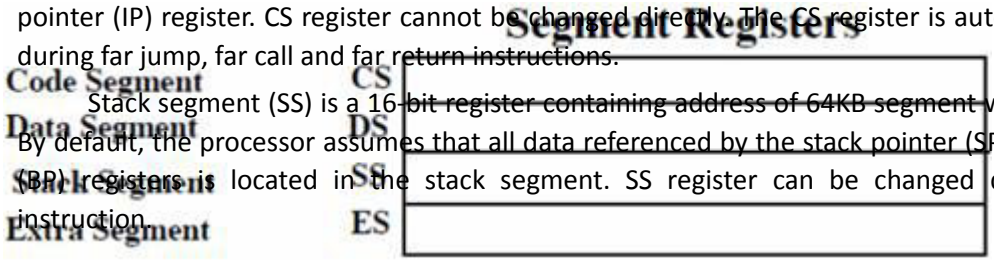
Source Index (SI) is a 16-bit register. SI is used for indexed, based indexed and register indirect addressing, as well as a source data address in string manipulation instructions. Used in conjunction with the DS register to point to data locations in the data segment.

Destination Index (DI) is a 16-bit register. Used in conjunction with the ES register in string operations. DI is used for indexed, based indexed and register indirect addressing, as well as a destination data address in string manipulation instructions. In short, Destination Index and SI Source Index registers are used to hold address.

Segment Registers

Most of the registers contain data/instruction offsets within 64 KB memory segment. There are four different 64 KB segments for instructions, stack, data and extra data. To specify where in 1 MB of processor memory these 4 segments are located the processor uses four segment registers.

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

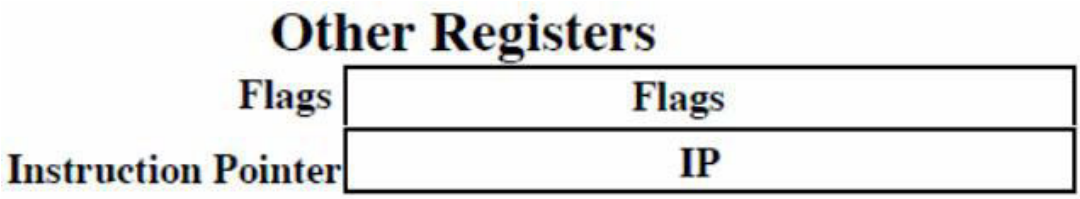


Stack segment (SS) is a 16-bit register containing address of 64KB segment with program stack. By default, the processor assumes that all data referenced by the stack pointer (SP) and base pointer (BP) registers is located in the stack segment. SS register can be changed directly using POP instruction.

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

Extra segment (ES) used to hold the starting address of Extra segment. Extra segment is provided for programs that need to access a second data segment. Segment registers cannot be used in arithmetic operations.

Other registers of 8086:



Instruction Pointer (IP) is a 16-bit register. This is a crucially important register which is used to control which instruction the CPU executes. The IP, or program counter, is used to store the memory location of the next instruction to be executed. The CPU checks the program counter to ascertain which instruction to carry out next. It then updates the program counter to point to the next instruction. Thus the program counter will always point to the next instruction to be executed.

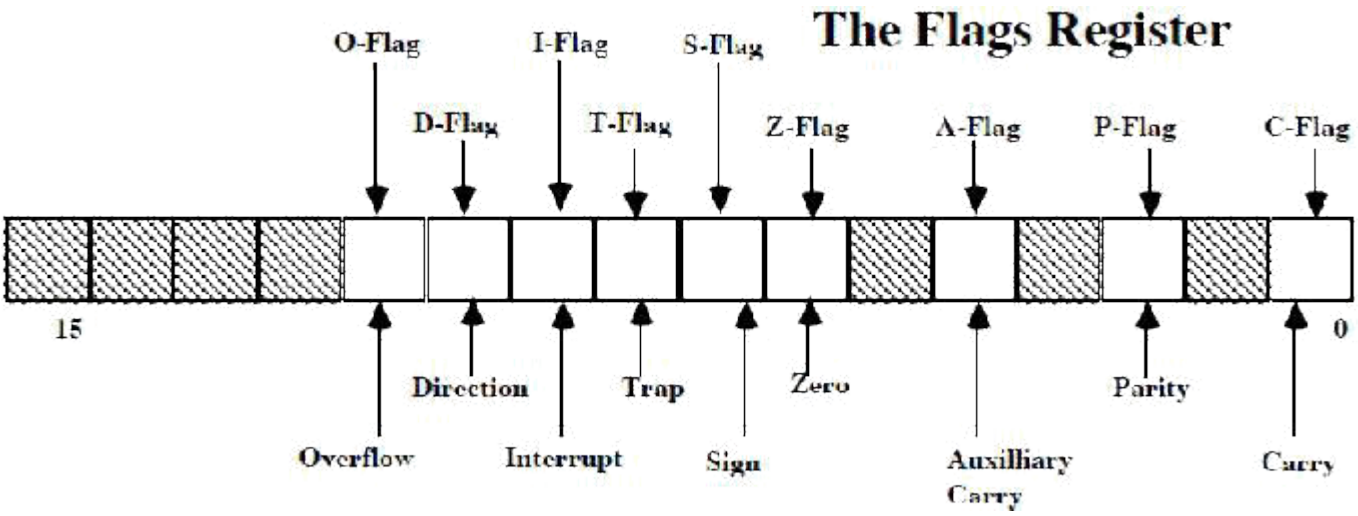
Flag Register contains a group of status bits called flags that indicate the status of the CPU or the result of arithmetic operations. There are two types of flags:

1. The **status flags** which reflect the result of executing an instruction. The programmer cannot set/reset these flags directly.

2. The **control flags** enable or disable certain CPU operations. The programmer can set/reset these bits to control the CPU's operation.

Nine individual bits of the status register are used as control flags (3 of them) and status flags (6 of them).The remaining 7 are not used.

A flag can only take on the values 0 and 1. We say a flag is set if it has the value 1.The status flags are used to record specific characteristics of arithmetic and of logical instructions.



Control Flags: There are three control flags

1. **The Direction Flag (D):** Affects the direction of moving data blocks by such instructions as MOVS, CMPS and SCAS. The flag values are 0 = up and 1 = down and can be set/reset by the STD (set D) and CLD (clear D) instructions.

2. **The Interrupt Flag (I):** Dictates whether or not system interrupts can occur. Interrupts are actions initiated by hardware block such as input devices that will interrupt the normal execution of programs. The flag values are 0 = disable interrupts or 1 = enable interrupts and can be manipulated by the CLI (clear I) and STI (set I) instructions.

3. **The Trap Flag (T):** Determines whether or not the CPU is halted after the execution of each instruction. When this flag is set (i.e. = 1), the programmer can single step through his program to debug any errors. When this flag = 0 this feature is off. This flag can be set by the INT 3 instruction.

Status Flags: There are six status flags

1. **The Carry Flag (C):** This flag is set when the result of an unsigned arithmetic operation is too large to fit in the destination register. This happens when there is an end carry in an addition operation or there an end borrows in a subtraction operation. A value of 1 = carry and 0 = no carry.

2. **The Overflow Flag (O):** This flag is set when the result of a signed arithmetic operation is too large to fit in the destination register (i.e. when an overflow occurs). Overflow can occur when adding two numbers with the same sign (i.e. both positive or both negative). A value of 1 = overflow and 0 = no overflow.

3. **The Sign Flag (S):** This flag is set when the result of an arithmetic or logic operation is negative. This flag is a copy of the MSB of the result (i.e. the sign bit). A value of 1 means negative and 0 = positive.

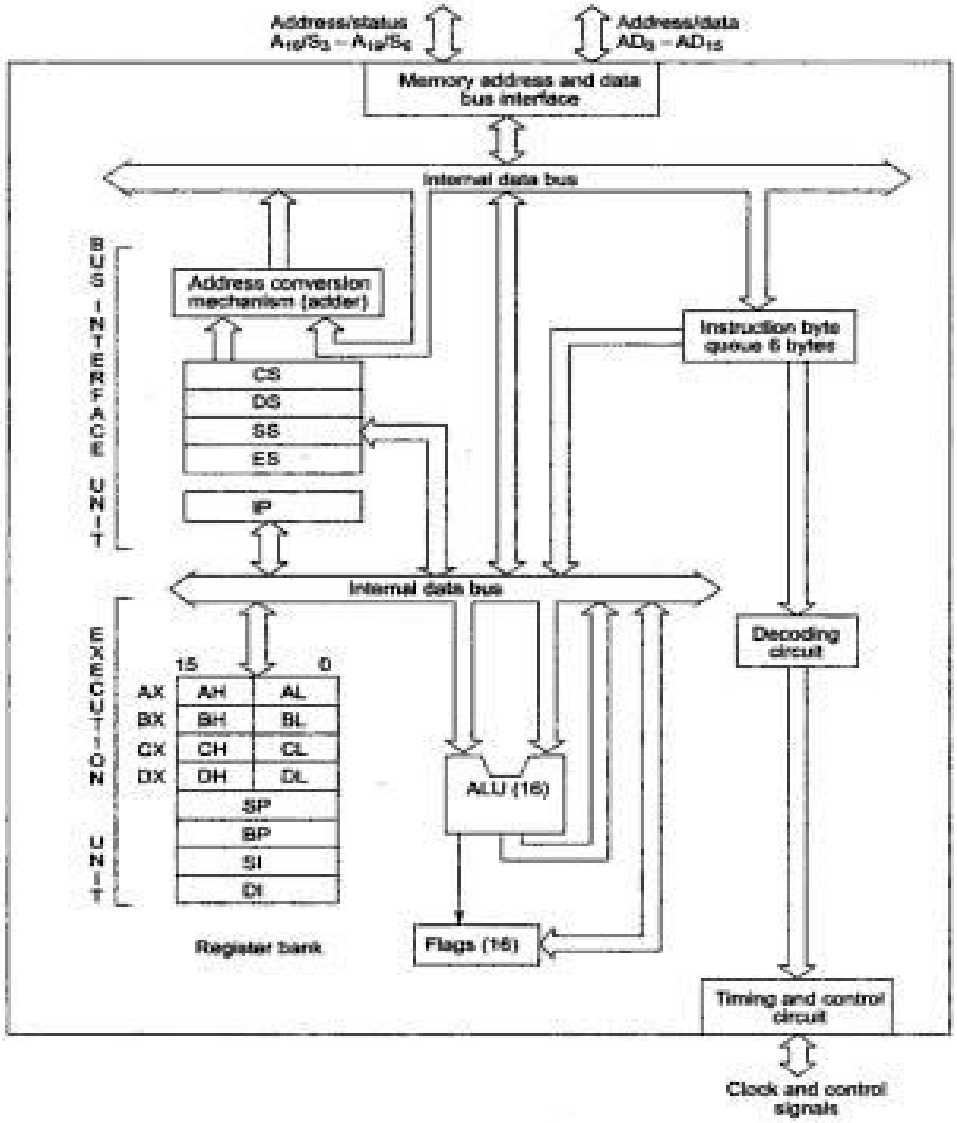
4. **The Zero Flag (Z):** This flag is set when the result of an arithmetic or logic operation is equal to zero. A value of 1 means the result is zero and a value of 0 means the result is not zero.

5. **The Auxiliary Carry Flag (A):** This flag is set when an operation causes a carry from bit 3 to bit 4 (or a borrow from bit 4 to bit 3) of an operand. A value of 1 = carry and 0 = no carry.

6. **The Parity Flag (P):** This flags reflects the number of 1s in the result of an operation. If the number of 1s is even its value = 1 and if the number of 1s is odd then its value = 0.

Architecture of 8086 or Functional Block diagram of 8086

- 8086 has two blocks Bus Interface Unit (BIU) and Execution Unit (EU).
- 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 executes instructions from the instruction system byte queue.
- Both units operate asynchronously to give the 8086 an overlapping instruction fetch and execution mechanism which is called as Pipelining. This results in efficient use of the system bus and system performance.
- BIU contains Instruction queue, Segment registers, Instruction pointer, Address adder.
- EU contains Control circuitry, Instruction decoder, ALU, Pointer and Index register, Flag register.



Explanation of Architecture of 8086

Bus Interface Unit:

- It provides a full 16 bit bidirectional data bus and 20 bit address bus.
- The bus interface unit is responsible for performing all external bus operations.

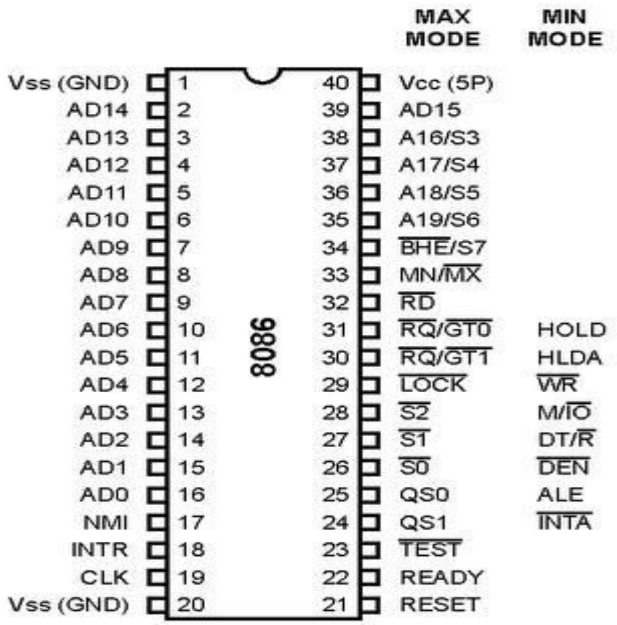
Specifically it has the following functions:

- Instruction fetch Instruction queuing, Operand fetch and storage, Address relocation and Bus control.
- The BIU uses a mechanism known as an instruction stream queue to implement pipeline architecture.
- This queue permits prefetch of up to six bytes of instruction code. When ever the queue of the BIU is not full, it has room for at least two more bytes and at the same time the EU is not requesting it to read or write operands from memory, the BIU is free to look ahead in the program by prefetching the next sequential instruction.
- These prefetching instructions are held in its FIFO queue. With its 16 bit data bus, the BIU fetches two instruction bytes in a single memory cycle.
- After a byte is loaded at the input end of the queue, it automatically shifts up through the FIFO to the empty location nearest the output.
- The EU accesses the queue from the output end. It reads one instruction byte after the other from the output of the queue. If the queue is full and the EU is not requesting access to operand in memory.
- These intervals of no bus activity, which may occur between bus cycles are known as Idle state.
- If the BIU is already in the process of fetching an instruction when the EU request it to read or write operands from memory or I/O, the BIU first completes the instruction fetch bus cycle before initiating the operand read / write cycle.
- The BIU also contains a dedicated adder which is used to generate the 20bit physical address that is output on the address bus. This address is formed by adding an appended
- 16 bit segment address and a 16 bit offset address.
- For example: The physical address of the next instruction to be fetched is formed by combining the current contents of the code segment CS register and the current contents of the instruction pointer IP register.
- The BIU is also responsible for generating bus control signals such as those for memory read or write and I/O read or write.

Execution Unit

- The Execution unit is responsible for decoding and executing all instructions.
- The EU extracts instructions from the top of the queue in the BIU, decodes them, generates operands if necessary, passes them to the BIU and requests it to perform the read or write bus cycles to memory or I/O and perform the operation specified by the instruction on the operands.
- During the execution of the instruction, the EU tests the status and control flags and updates them based on the results of executing the instruction.
- If the queue is empty, the EU waits for the next instruction byte to be fetched and shifted to top of the queue.
- When the EU executes a branch or jump instruction, it transfers control to a location corresponding to another set of sequential instructions.
- Whenever this happens, the BIU automatically resets the queue and then begins to fetch instructions from this new location to refill the queue.

Pin Diagram and Pin description of 8086:



The Microprocessor 8086 is a 16-bit CPU available in different clock rates and packaged in a 40 pin CERDIP or plastic package.

- The 8086 operates in single processor or multiprocessor configuration to achieve high performance. The pins serve a particular function in minimum mode (single processor mode) and other function in maximum mode configuration (multiprocessor mode).

· The 8086 signals can be categorized in three groups.

- The first are the signal having common functions in minimum as well as maximum mode.
- The second are the signals which have special functions for minimum mode
- The third are the signals having special functions for maximum mode.

· The following signal descriptions are common for both modes.

· **AD15-AD0:** These are the time multiplexed memory I/O address and data lines.

- Address remains on the lines during T1 state, while the data is available on the data bus during T2, T3, Tw and T4. These lines are active high and float to a tristate during interrupt acknowledge and local bus hold acknowledge cycles.

A19/S6, A18/S5, A17/S4, and A16/S3: These are the time multiplexed address and status lines.

- During T1 these are the most significant address lines for memory operations.

- During I/O operations, these lines are low.
- During memory or I/O operations, status information is available on those lines for T2, T3, Tw and T4.
- The status of the interrupt enable flag bit is updated at the beginning of each clock cycle.
- The S4 and S3 combine indicate which segment registers is presently being used for memory accesses as in below fig.

These lines float to tri-state off during the local bus hold acknowledge. The status line S6 is always low.

o The address bit is separated from the status bit using latches controlled by the ALE signal.

S_4	S_3	Indication
0	0	Alternate Data
0	1	Stack
1	0	Code or None
1	1	Data
0	0	Whole word
0	1	Upper byte from or to even address
1	0	Lower byte from or to even address

· **BHE/S7:** The bus high enable is used to indicate the transfer of data over the higher order (D15-D8) data bus as shown in table. It goes low for the data transfer over D15-D8 and is used to derive chip selects of odd address memory bank or peripherals. BHE is low during T1 for read, write and interrupt acknowledge cycles, whenever a byte is to be transferred on higher byte of data bus. The status information is available during T2, T3 and T4. The signal is active low and tristated during hold. It is low during T1 for the first pulse of the interrupt acknowledge cycle.

· **RD – Read:** This signal on low indicates the peripheral that the processor is performing memory or I/O read operation. RD is active low and shows the state for T2, T3, Tw of any read cycle. The signal remains tristated during the hold acknowledge.

READY: This is the acknowledgement from the slow device or memory that they have completed the data transfer. The signal made available by the devices is synchronized by the 8284A clock

generator to provide ready input to the 8086. the signal is active high.

- **INTR-Interrupt Request:** This is a triggered input. This is sampled during the last clock cycles of each instruction to determine the availability of the request. If any interrupt request is pending, the processor enters the interrupt acknowledge cycle. This can be internally masked by resulting the interrupt enable flag. This signal is active high and internally synchronized.

- **TEST:** This input is examined by a 'WAIT' instruction. If the TEST pin goes low, execution will continue, else the processor remains in an idle state. The input is synchronized internally during each clock cycle on leading edge of clock.

- **CLK- Clock Input:** The clock input provides the basic timing for processor operation and bus control activity. It's an asymmetric square wave with 33% duty cycle.

The following pin functions are for the minimum mode operation of 8086.

- **M/IO – Memory/IO:** This is a status line logically equivalent to S2 in maximum mode.

  When it is low, it indicates the CPU is having an I/O operation, and when it is high, it indicates that the CPU is having a memory operation. This line becomes active high in the previous T4 and remains active till final T4 of the current cycle. It is tristated during local bus "hold acknowledge ".

- **INTA – Interrupt Acknowledge:** This signal is used as a read strobe for interrupt acknowledge cycles. i.e. when it goes low, the processor has accepted the interrupt.

- **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 and is never tristated.

- **DT/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.

- **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. It is active from the middle of T2 until the middle of T4. This is tristated during 'hold acknowledge' cycle.

- **HOLD, HLDA- Acknowledge:** When the HOLD line goes high; it indicates to the processor that another master is requesting the bus access. The processor, after receiving the HOLD

request, issues the hold acknowledge signal on HLDA pin, in the middle of the next clock cycle after completing the current bus cycle.

- At the same time, the processor floats the local bus and control lines. When the processor detects the HOLD line low, it lowers the HLDA signal. HOLD is an asynchronous input, and is should be externally synchronized. If the DMA request is made while the CPU is performing a memory or I/O cycle, it will release the local bus during T4 provided :

1. The request occurs on or before T2 state of the current cycle.
2. The current cycle is not operating over the lower byte of a word.
3. The current cycle is not the first acknowledge of an interrupt acknowledge sequence.
4. A Lock instruction is not being executed.

The following pin functions are applicable for maximum mode operation of 8086.

- **S2, S1, and S0 – Status Lines:** These are the status lines which reflect the type of operation, being carried out by the processor. These become activity during T4 of the previous cycle and active during T1 and T2 of the current bus cycles.

- **LOCK:** This output pin indicates that other system bus master will be prevented from gaining the system bus, while the LOCK signal is low. The LOCK signal is activated by the 'LOCK' prefix instruction and remains active until the completion of the next instruction. When the CPU is executing a critical instruction which requires the system bus, the LOCK prefix instruction ensures that other processors connected in the system will not gain the control of the bus.

The 8086, while executing the prefixed instruction, asserts the bus lock signal output, which may be connected to an external bus controller. By prefetching the instruction, there is a considerable speeding up in instruction execution in 8086. This is known as **instruction pipelining**.

At the starting the CS: IP is loaded with the required address from which the execution is to be started. Initially, the queue will be empty and the microprocessor starts a fetch operation to bring one byte (the first byte) of instruction code, if the CS: IP address is odd or two bytes at a time, if the CS: IP address is even.

S_2	S_1	S_0	Indication
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

- The first byte is a complete opcode in case of some instruction (one byte opcode instruction) and is a part of opcode, in case of some instructions (two byte opcode instructions), the remaining part of code lie in second byte.
- The second byte is then decoded in continuation with the first byte to decide the instruction length and the number of subsequent bytes to be treated as instruction data. The queue is updated after every byte is read from the queue but the fetch cycle is initiated by BIU only if at least two bytes of the queue are empty and the EU may be concurrently executing the fetched instructions.
- The next byte after the instruction is completed is again the first opcode byte of the next instruction. A similar procedure is repeated till the complete execution of the program. The fetch operation of the next instruction is overlapped with the execution of the current instruction. As in the architecture, there are two separate units, namely Execution unit and Bus interface unit.
- While the execution unit is busy in executing an instruction, after it is completely decoded, the bus interface unit may be fetching the bytes of the next instruction from memory, depending upon the queue status.

QS_1	QS_0	Indication
0	0	No Operation
0	1	First Byte of the opcode from the queue
1	0	Empty Queue
1	1	Subsequent Byte from the Queue

RQ/GT0, RQ/GT1 – Request/Grant:

These pins are used by the other local bus master in maximum mode, to force the processor to release the local bus at the end of the processor current bus cycle.

Each of the pin is bidirectional with RQ/GT0 having higher priority than RQ/GT1.

- RQ/GT pins have internal pull-up resistors and may be left unconnected. Request/Grant

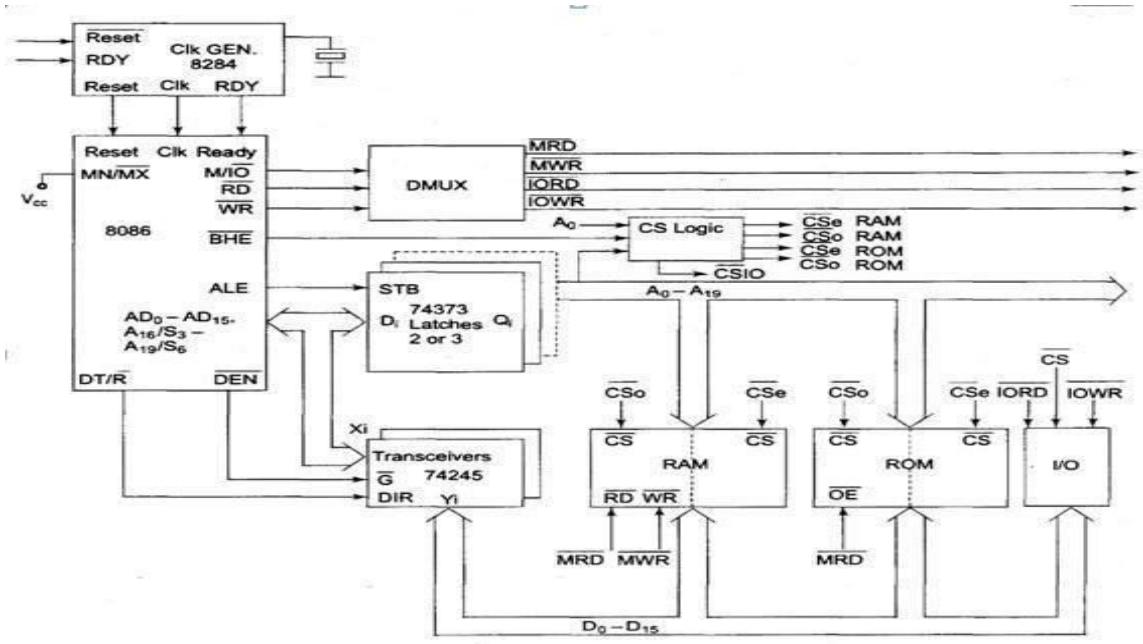
sequence is as follows:

1. A pulse of one clock wide from another bus master requests the bus access to 8086.
2. During T4(current) or T1(next) clock cycle, a pulse one clock wide from 8086 to the requesting

master, indicates that the 8086 has allowed the local bus to float and that it will enter the 'hold acknowledge' state at next cycle. The CPU bus interface unit is likely to be disconnected from the local bus of the system.

3. A one clock wide pulse from another master indicates to the 8086 that the hold request is about to end and the 8086 may regain control of the local bus at the next clock cycle. Thus each master to master exchange of the local bus is a sequence of 3 pulses. There must be at least one dead clock cycle after each bus exchange. The request and grant pulses are active low. For the bus request those are received while 8086 is performing memory or I/O cycle, the granting of the bus is governed by the rules as in case of HOLD and HLDA in minimum mode.

Minimum Mode 8086 System



Minimum mode 8086 system

- In a minimum mode 8086 system, the microprocessor 8086 is operated in minimum mode by strapping its MN/MX pin to logic 1.

- In this mode, all the control signals are given out by the microprocessor chip itself. There is a single microprocessor in the minimum mode system.

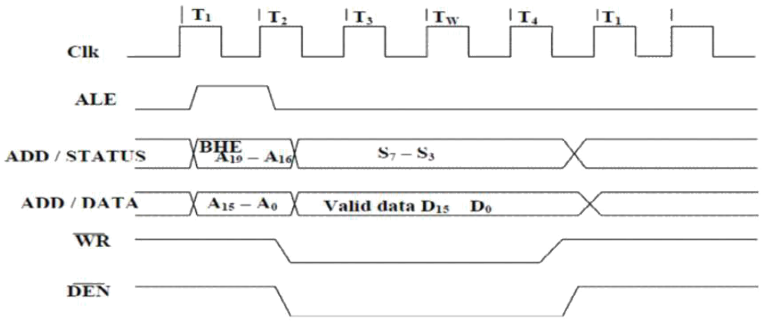
- The remaining components in the system are latches, transceivers, clock generator,

memory and I/O devices. Some type of chip selection logic may be required for selecting memory or I/O devices, depending upon the address map of the system.

- Latches are generally buffered output D-type flip-flops like 74LS373 or 8282. They are used for separating the valid address from the multiplexed address/data signals and are controlled by the ALE signal generated by 8086.
- Transceivers are the bidirectional buffers and some times they are called as data amplifiers. They are required to separate the valid data from the time multiplexed address/data signals.
- They are controlled by two signals namely, DEN and DT/R.
- The DEN signal indicates the direction of data, i.e. from or to the processor. The system contains memory for the monitor and users program storage.
- Usually, EPROM is used for monitor storage, while RAM for users program storage. A system may contain I/O devices.

Write Cycle Timing Diagram for Minimum Mode

- The working of the minimum mode configuration system can be better described in terms of the timing diagrams rather than qualitatively describing the operations.
- The opcode fetch and read cycles are similar. Hence the timing diagram can be categorized in two parts, the first is the timing diagram for read cycle and the second is the timing diagram for write cycle.
- The read cycle begins in T1 with the assertion of address latch enable (ALE) signal and also M / IO signal. During the negative going edge of this signal, the valid address is latched on the local bus.
- The BHE and A0 signals address low, high or both bytes. From T1 to T4 , the M/IO signal indicates a memory or I/O operation.
- At T2, the address is removed from the local bus and is sent to the output. The bus is then tristated. The read (RD) control signal is also activated in T2.
- The read (RD) signal causes the address device to enable its data bus drivers. After RD goes low, the valid data is available on the data bus.



- The addressed device will drive the READY line high. When the processor returns the read signal to high level, the addressed device will again tristate its bus drivers.

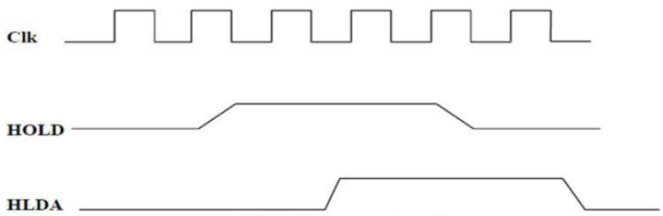
- A write cycle also begins with the assertion of ALE and the emission of the address. The M/IO signal is again asserted to indicate a memory or I/O operation. In T2, after sending the address in T1, the processor sends the data to be written to the addressed location.

- The data remains on the bus until middle of T4 state. The WR becomes active at the beginning of T2 (unlike RD is somewhat delayed in T2 to provide time for floating).

- The BHE and A0 signals are used to select the proper byte or bytes of memory or I/O word to be read or write.

- The M/IO, RD and WR signals indicate the type of data transfer as specified in table below.

Bus Request and Bus Grant Timings in Minimum Mode System of 8086



Bus Request and
Bus Grant Timings in Minimum Mode System

- Hold Response sequence: The HOLD pin is checked at leading edge of each clock pulse.

  If it is received active by the processor before T4 of the previous cycle or during T1 state of the current cycle, the CPU activates HLDA in the next clock cycle and for succeeding bus cycles, the bus will be given to another requesting master.

- The control of the bus is not regained by the processor until the requesting master does not drop the HOLD pin low. When the request is dropped by the requesting master, the HLDA is dropped by the processor at the trailing edge of the next clock.

Maximum Mode 8086 System

- In the maximum mode, the 8086 is operated by strapping the MN/MX pin to ground.

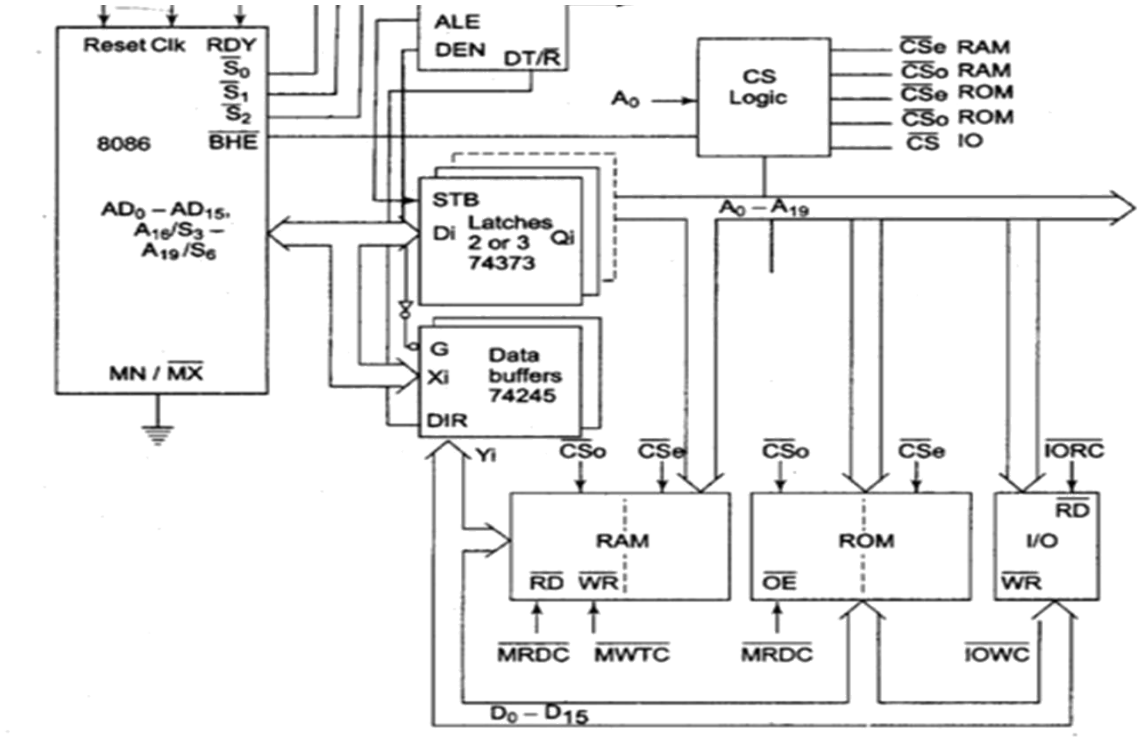
- In this mode, the processor derives the status signal S2, S1, S0. Another chip called bus controller derives the control signal using this status information.

- In the maximum mode, there may be more than one microprocessor in the system configuration.

- The components in the system are same as in the minimum mode system.

- The basic function of the bus controller chip IC8288 is to derive control signals like RD and WR (for memory and I/O devices), DEN, DT/R, ALE etc. using the information by the processor on the status lines.

- The bus controller chip has input lines S2, S1, S0 and CLK. These inputs to 8288 are driven by CPU.



- It derives the outputs ALE, DEN, DT/R, MRDC, MWTC, AMWC, IORC, IOWC and AIOWC. The AEN, IOB and CEN pins are especially useful for multiprocessor systems.
- AEN and IOB are generally grounded. CEN pin is usually tied to +5V. The significance of the MCE/PDEN output depends upon the status of the IOB pin.
- If IOB is grounded, it acts as master cascade enable to control cascade 8259A, else it acts as peripheral data enable used in the multiple bus configurations.
- INTA pin used to issue two interrupt acknowledge pulses to the interrupt controller or to an interrupting device.
- IORC, IOWC are I/O read command and I/O write command signals respectively.
  - These signals enable an IO interface to read or write the data from or to the address port.
  - The MRDC, MWTC are memory read command and memory write command signals respectively and may be used as memory read or write signals.
  - All these command signals instructs the memory to accept or send data from or to the bus.
  - For both of these write command signals, the advanced signals namely AIOWC and AMWTC are available.

- Here the only difference between in timing diagram between minimum mode and maximum mode is the status signals used and the available control and advanced command signals.

- R0, S1, S2 are set at the beginning of bus cycle.8288 bus controller will output a pulse as on the ALE and apply a required signal to its DT / R pin during T1.

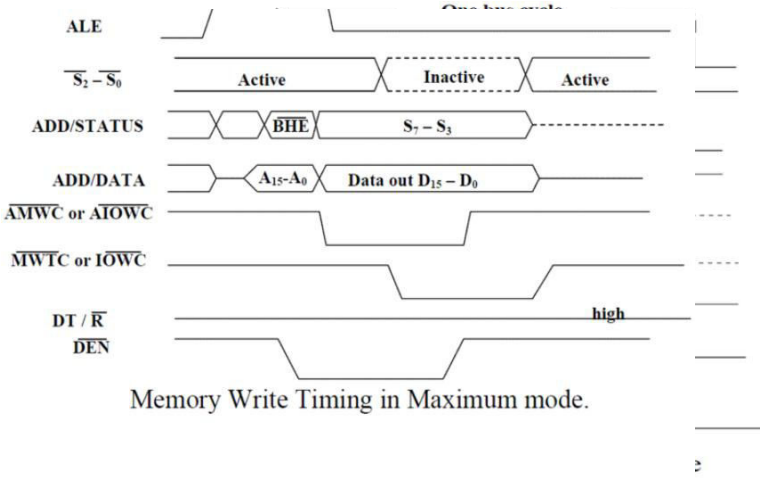
- In T2, 8288 will set DEN=1 thus enabling transceivers, and for an input it will activate MRDC or IORC. These signals are activated until T4. For an output, the AMWC or AIOWC is activated from T2 to T4 and MWTC or IOWC is activated from T3 to T4.

- The status bit S0 to S2 remains active until T3 and become passive during T3 and T4.

- If reader input is not activated before T3, wait state will be inserted between T3 and T4.

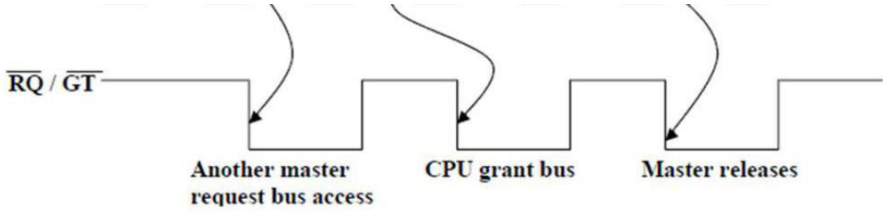
Memory Read Timing Diagram in Maximum Mode of 8086

Memory Write Timing in Maximum mode of 8086



Memory Write Timing in Maximum mode.

RQ/GT Timings in Maximum Mode



$\overline{RQ}/\overline{GT}$ Timings in Maximum Mode

- The request/grant response sequence contains a series of three pulses. The request/grant pins are checked at each rising pulse of clock input.
- When a request is detected and if the condition for HOLD request is satisfied, the processor issues a grant pulse over the RQ/GT pin immediately during T4 (current) or T1 (next) state.
- When the requesting master receives this pulse, it accepts the control of the bus, it sends a release pulse to the processor using RQ/GT pin.

UNIT II
Addressing Modes, Instruction Set and Programming of 8086

Addressing Modes of 8086:

Addressing mode indicates a way of locating data or operands. Depending up on the data type used in the instruction and the memory addressing modes, any instruction may belong to one or more addressing modes or same instruction may not belong to any of the addressing modes.

The addressing mode describes the types of operands and the way they are accessed for executing an instruction. According to the flow of instruction execution, the instructions may be categorized as

1. Sequential control flow instructions and
2. Control transfer instructions.

Sequential control flow instructions are the instructions which after execution, transfer control to the next instruction appearing immediately after it (in the sequence) in the program. For example the arithmetic, logic, data transfer and processor control instructions are Sequential control flow instructions.

The control transfer instructions on the other hand transfer control to some predefined address or the address somehow specified in the instruction, after their execution. For example INT, CALL, RET & JUMP instructions fall under this category.

The addressing modes for Sequential and control flow instructions are explained as follows.

1. Immediate addressing mode:

In this type of addressing, immediate data is a part of instruction, and appears in the form of successive byte or bytes.

Example: MOV AX, 0005H.

In the above example, 0005H is the immediate data. The immediate data may be 8- bit or 16-bit in size.

2. Direct addressing mode:

In the direct addressing mode, a 16-bit memory address (offset) directly specified in the instruction as a part of it.

Example: MOV AX, [5000H].

3. Register addressing mode:

In the register addressing mode, the data is stored in a register and it is referred using the particular register. All the registers, except IP, may be used in this mode.

Example: MOV BX, AX

4. Register indirect addressing mode:

Sometimes, the address of the memory l o c a t i o n which contains data or operands is determined in an indirect way, using the offset registers. The mode of addressing is known as register indirect mode.

In this addressing mode, the offset address of data is in either BX or SI or DI Register. The default segment is either DS or ES.

Example: MOV AX, [BX].