Le-Ment College of Advanced Studies-Pattambi

University of Calicut

B.Sc Computer Science (CUCBCSS) - (2019 admission onwards)

Fourth semester

COMMON COURSE

A14 - **Microprocessors-Architecture and Programming**

Pradeep C P (M.Sc,,M.Phil)

Assistant professor (Electronics)

Le-Ment College Of Advanced Studies.

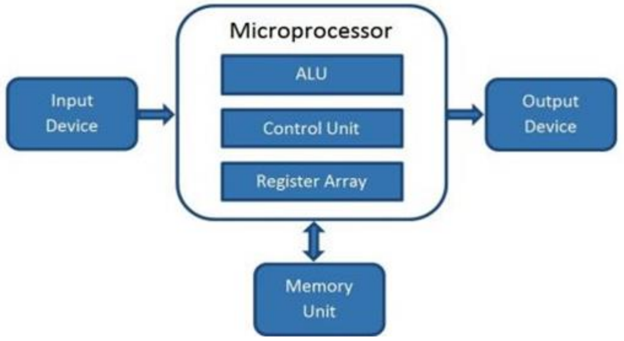
(Affiliated to Calicut University)

Pattambi

Palakkad

UNIT-1: General Architecture of computer:

Block Diagram of a Microcomputer:



A microcomputer is a complete computer on a small scale, designed for use by one person at a time. An antiquated term, a microcomputer is now primarily called a personal computer (PC), or a device based on a single-chip microprocessor. Common microcomputers include laptops and desktops. Beyond standard PCs, microcomputers also include some calculators, mobile phones, notebooks, workstations and embedded systems.

Smaller than a mainframe or minicomputer, a microcomputer uses a single integrated semiconductor chip for its central processing unit (CPU). They also contain memory in the form of read-only memory (ROM) and random access memory (RAM), input/output (I/O) ports, and a bus or system of interconnecting wires, all housed in a single unit usually referred to as a motherboard.

Common I/O devices include keyboards, monitors, printers and external storage.

The term microcomputer dates back to the 1970s. The advent of the Intel 4004 microprocessor in 1971, and later the Intel 8008 and Intel 8080 microprocessor in 1972 and 1974 respectively, paved the path to the creation of the microcomputer.

By the 1980s, microcomputers were being used for more than games and computer-based recreation, finding widespread use in personal computing, workstations and academia. By the 1990s, microcomputers were being produced as pocket-sized personal digital assistants (PDAs), and later came in the form of cell phones and portable music players.

Microcomputer applications

Personal microcomputers are often used for education and entertainment. Beyond laptops and desktops, microcomputers can include video game consoles, computerized electronics and smartphones.

In the workplace, microcomputers have been used for applications including data and word processing, electronic spread sheets, professional presentation and graphics programs, communications and database management systems. They have been used in business for tasks such as bookkeeping, inventory and communication; in medical settings to record and recall patient data, manage healthcare plans, complete schedule and for data processing; in financial institutions to record transactions, track billing, prepare financial statements and payrolls, and auditing; and in military applications for training devices, among other uses.

Introduction to microprocessor.

Microprocessor is a type of miniature digital electronic device that contains the arithmetic, logic, and control circuitry necessary to perform the functions of a digital computer's central processing unit. In effect, this kind of integrated circuit can interpret and execute program instructions as well as handle arithmetic and logical operations.

How does a Microprocessor Work?

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.

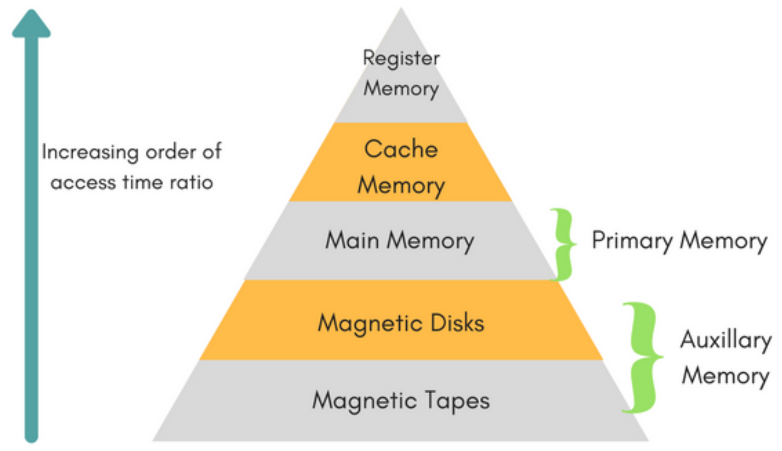
The microprocessor also permitted the development of so-called intelligent terminals, such as automatic teller machines and point-of-sale terminals employed in retail stores. The microprocessor also provides automatic control of industrial robots, surveying instruments, and various kinds of hospital equipment. It has brought about the computerization of a wide array of consumer products, including programmable microwave ovens, television sets, and electronic games. In addition, some automobiles feature microprocessor-controlled ignition and fuel systems designed to improve performance and fuel economy.

Memory organization

A memory unit is the collection of storage units or devices together. The memory unit stores the binary information in the form of bits. Generally, memory/storage is classified into 2 categories:

- **Volatile Memory**: This loses its data, when power is switched off.
- **Non-Volatile Memory**: This is a permanent storage and does not lose any data when power is switched off.

Memory Hierarchy



A memory unit is an integral part of any microcomputer, and its primary purpose is to hold instructions and data. The major design goal of a memory unit is to allow it to operate at a speed close to that of a microprocessor. A microcomputer memory system can be divided into three groups:

- Microprocessor memory (registers and external addressable memory)
- Primary or main memory (RAM,ROM)
- Secondary memory (Magnetic Discs, HDD, magnetic tapes)

Microprocessor memory comprises to a set of microprocessor registers. These registers are used to hold temporary results when a computation is in progress.

Introduction to Intel 8085 microprocessor.

Intel 8085 is a programmable, multipurpose, clock -driven, register-based electronic device that reads binary instructions from a storage device called memory, accepts binary data as input and processes data according to those instructions and provides results as output.

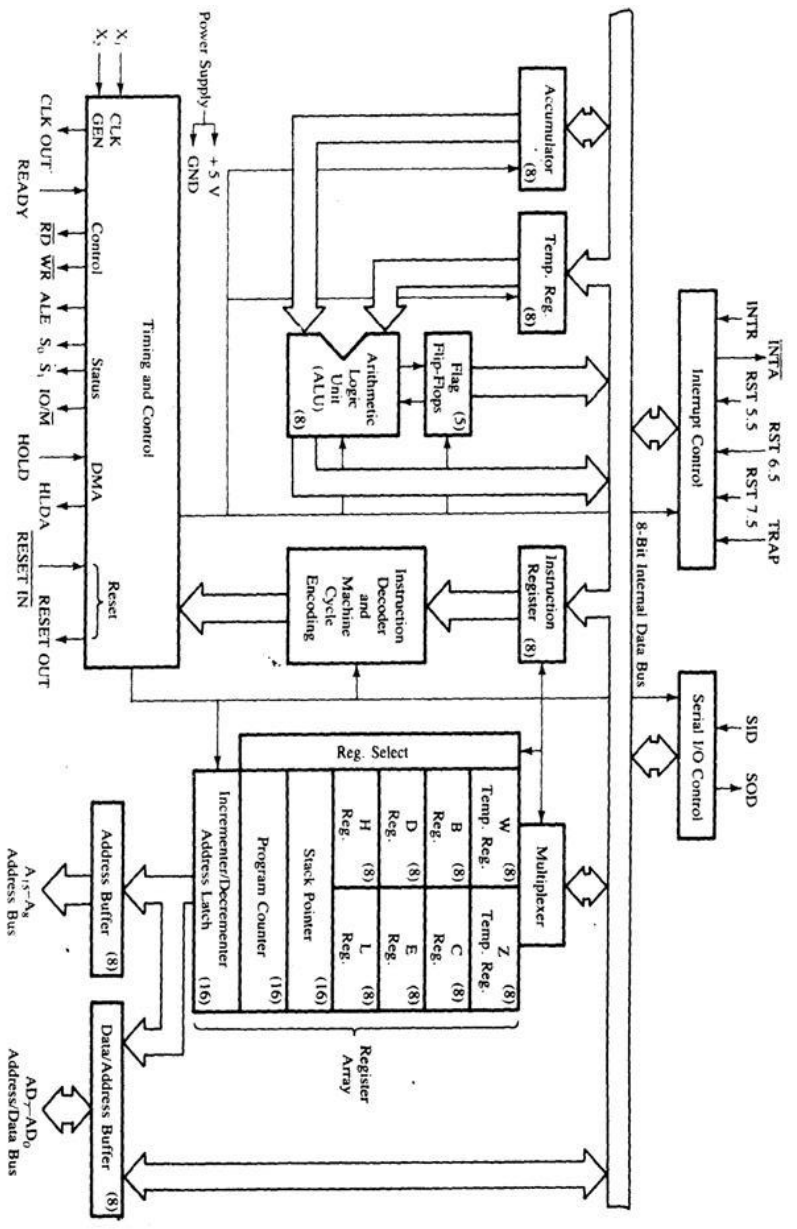
A microprocessor consists of an ALU, control unit and register array. Where **ALU** performs arithmetic and logical operations on the data received from an input device or memory. Control unit controls the instructions and flow of data within the computer. And, **register array** consists of registers identified by letters like B, C, D, E, H, L, and accumulator.

It has following configuration:

- It is a 40 pin I.C. package fabricated on a single LSI chip.
- The Intel 8085 uses a single +5Vd.c. supply for its operation.
- Intel 8085s clock speed is about 3 MHz; the clock cycle is of 320ns.
- 8 bit data bus.
- Address bus is of 16-bit, which can address up to 64KB
- 16-bit stack pointer
- 16 bit PC (Program Counter)
- Six 8-bit registers are arranged in pairs :BC, DE, HL

Intel 8085 is used in mobile phones, microwave ovens, washing machines etc.

Architecture of 8085 microprocessor



ALU

The **Arithmetic and Logic Unit**, ALU performs the arithmetic and logical operations:

- Addition
- Subtraction
- Logical AND
- Logical OR
- Logical EXCLUSIVE OR
- Complement (Logical NOT)
- Increment (add 1)
- Decrement (subtract 1)
- Left shift, Rotate left, Rotate right
- Clear, etc.

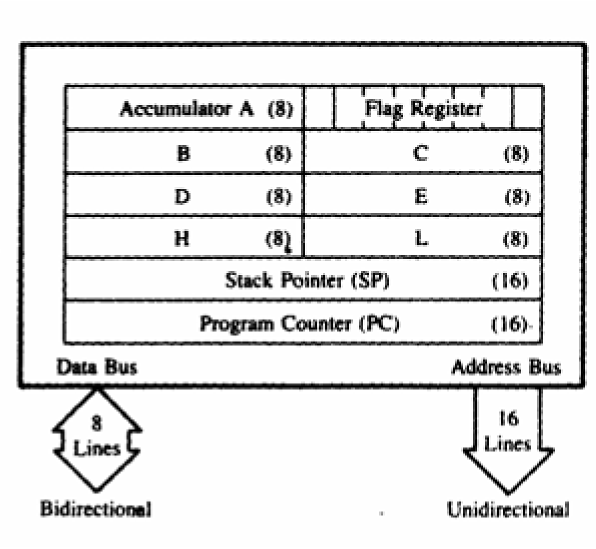
Timing and Control Unit

The timing and control unit is the section of the CPU.

- It is used to generate timing and control signals which are necessary for the execution of instructions.
- It is used to control data flow between CPU and peripherals (including memory).
- It is used to provide status, control and timing signals which are required for the operation of memory and I/O devices.
- It is used to control the entire operations of the microprocessor and peripherals connected to it.

Thus we can see that the control unit of the CPU acts as the brain of the computer system.

Register organization



Registers are used for temporary storage and manipulation of data and instructions by the microprocessor. Data remain in the registers till they are sent to the I/O devices or memory. Intel 8085 microprocessor has the following registers:

- One 8-bit accumulator (ACC) i.e. register A
- Six general purpose registers of 8-bit, these are B,C, D, E, H and L
- One 16-bit stack pointer, SP
- One 16-bit Program Counter, PC
- Instruction register
- Temporary register

In addition to the above mentioned registers the 8085 microprocessor contains a set of five flip-flops which serve as flags (or status flags).

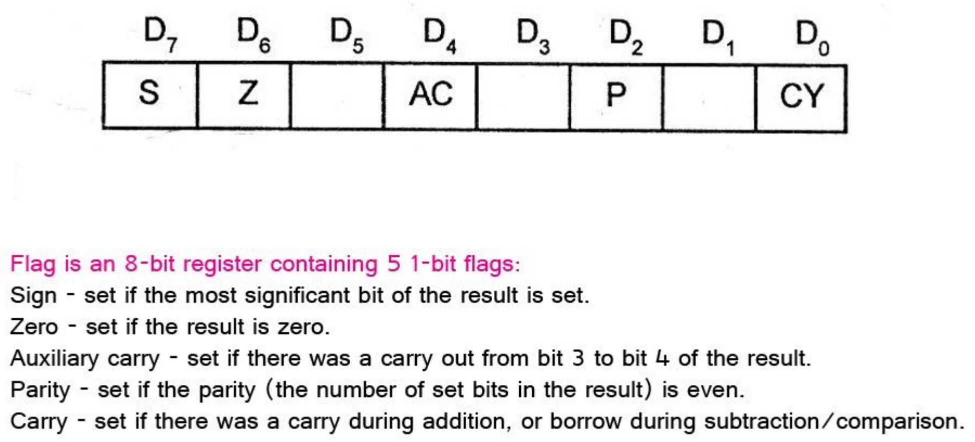
A flag is a flip-flop which indicates some conditions which arises after the execution of an arithmetic or logical instruction.

- **Accumulator (ACC):** The accumulator is an 8-bit register associated with the ALU. The register 'A' is an accumulator in the 8085. It is used to hold one of the operands of an arithmetic and logical operation. The final result of an arithmetic or logical operation is also placed in the accumulator.
- **General-Purpose Registers:** The 8085 microprocessor contains six 8-bit general purpose registers. They are: B, D, C, E, H and L register. To hold data of 16-bit a combination of two 8-bit registers can be employed. The combination of two 8-bit registers is called **register pair**. The valid register pairs in the 8085 are: D-E, B-C and H-L. The H-L pair is used to act as a memory pointer.
- **Program Counter (PC):** It is a 16-bit special purpose register. It is used to hold the address of memory of the next instruction to be executed. It keeps the track of the instruction in a program while they are being executed. The microprocessor increments the content of the next program counter during the execution of an instruction so that at the end of the execution of an instruction it points to the next instructions address in the program.
- **Stack Pointer (SP):** It is a 16-bit special function register used as memory pointer. A stack is nothing but a portion of RAM. In the stack, the contents of only those registers are saved, which are needed in the later part of the program. The stack pointer (SP) controls the addressing of the stack. The Stack Pointer contains the address of the top element of data stored in the stack.
- **Instruction Register:** The instruction register holds the opcode (operation code or instruction code) of the instruction which is being decoded and executed.

- **Temporary Register:** It is an 8-bit register associated with the ALU. It holds data during an arithmetic/logical operation. It is used by the microprocessor. It is not accessible to programmer.
- **Flags:** The Intel 8085 microprocessor contains five flip-flops to serve as a status flags. The flip-flops are reset or set according to the conditions which arise during an arithmetic or logical operation.

Flag register in 8085 microprocessor

The Flag register is a Special Purpose Register. Depending upon the value of result after any arithmetic and logical operation the flag bits become set (1) or reset (0).



Flag is an 8-bit register containing 5 1-bit flags:
Sign - set if the most significant bit of the result is set.
Zero - set if the result is zero.
Auxiliary carry - set if there was a carry out from bit 3 to bit 4 of the result.
Parity - set if the parity (the number of set bits in the result) is even.
Carry - set if there was a carry during addition, or borrow during subtraction/comparison.

The five status flags of Intel 8085 are:

1. **Sign Flag (S) –** After any operation if the MSB (B(7)) of the result is 1, it indicates the number is negative and the sign flag becomes set, i.e. 1. If the MSB is 0, it indicates the number is positive and the sign flag becomes reset i.e. 0.

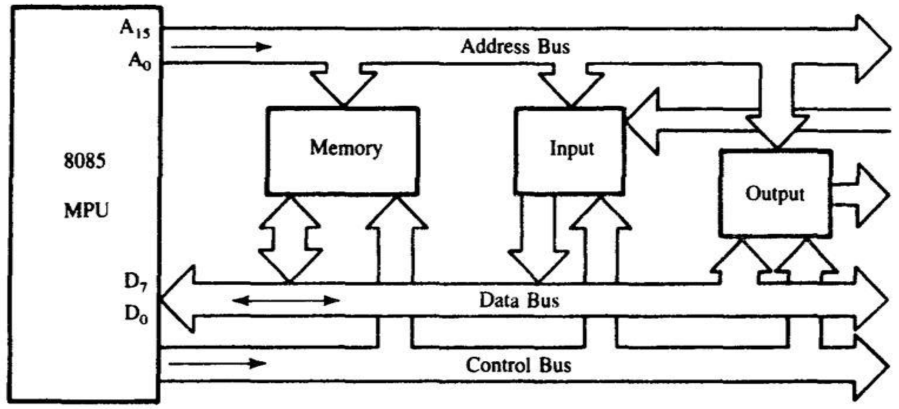
2. **Zero Flag (Z) –** After any arithmetical or logical operation if the result is 0 (00)H, the zero flag becomes set i.e. 1, otherwise it becomes reset i.e. 0.

3. **Auxiliary Carry Flag (AC) –** This flag is used in BCD number system(0-9). If after any arithmetic or logical operation D(3) generates any carry and passes on to B(4) this flag becomes set i.e. 1, otherwise it becomes reset i.e. 0.

4. **Parity Flag (P) –** If after any arithmetic or logical operation the result has even parity, an even number of 1 bits, the parity register becomes set i.e. 1, otherwise it becomes reset i.e. 0.

5. **Carry Flag (CY) –** Carry is generated when performing n bit operations and the result is more than n bits, then this flag becomes set i.e. 1, otherwise it becomes reset i.e. 0.

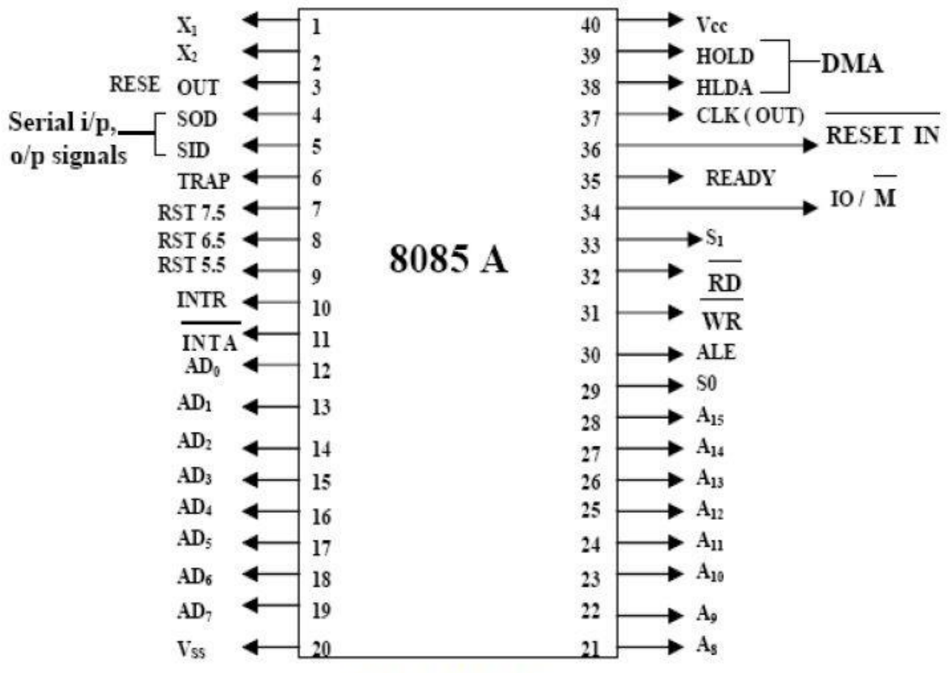
Bus organization of 8085



- **Data bus** is bidirectional because data flow in both directions, from microprocessor to memory or Input/output devices and from memory or Input/output devices to microprocessor..
- **Address bus** is unidirectional because data flow in one direction, from microprocessor to memory or from microprocessor to Input/output devices .The 8 most significant bits of the address are transmitted by the address bus, A-bus (pins A8 to A15). The 8 least significant bits of the address are transmitted by data/address bus, AD-bus (pins AD0 to AD7).

- **Control bus**: which is used to generate timing and control signals to control all the associated peripherals, microprocessor uses control bus to process data that is what to do with selected memory location. Some control signals are:
  - Memory read
  - Memory write
  - I/O read
  - I/O Write
  - Op-code fetch

8085-Pin Configuration



8085 Pin Diagram

Address Bus and Data Bus

A8 to A15 (Output): These are **address bus** and are used for the most significant bits of the memory address or 8-bits of I/O address.

AD0 to AD7 (Input/output): These are time multiplexed **address/data bus** i.e. they serve dual purpose. They are used for the least significant 8 bits of the memory address or I/O address during the first cycle. Again they are used for data during 2nd and 3rd clock cycles.

Control and Status Signals

ALE (Output): ALE stands for **Address Latch Enable** signal. ALE goes high during first clock cycle of a machine cycle and enables the lower 8-bits of the address to be latched either into the memory or external latch.

IO/M (Output): It is a **status signal** which distinguishes whether the address is for memory or I/O device.

S0, S1 (Output): These are **status signals** sent by the microprocessors to distinguish the various types of operation given in table below:

RD (Output): RD is a **signal to control READ operation**. When it goes low, the selected I/O device or memory is read.

WR (Output): WR is a **signal to control WRITE operation**. When it goes low, the data bus' data is written into the selected memory or I/O location.

READY (Input): It is used by the microprocessor to sense whether a peripheral is ready to transfer a data or not. If READY is high, the peripheral is ready. If it is low the microprocessor waits till it goes high.

Interrupts and Externally Initiated Signals

HOLD (INPUT): HOLD indicates that another device is requesting for the use of the address and data bus.

<u>HLDA (OUTPUT):</u> HLDA is a signal for **HOLD acknowledgement** which indicates that the HOLD request has been received. After the removal of this request the HLDA goes low.

<u>INTR (Input):</u> INTR is an **Interrupt Request Signal**. Among interrupts it has the lowest priority. The INTR is enabled or disabled by software.

<u>INTA (Output):</u> INTA is an **interrupt acknowledgement** sent by the microprocessor after INTR is received.

RST 5.5, 6.5, 7.5 and TRAP (Inputs): These **all are interrupts**. When any interrupt is recognized the next instruction is executed from a fixed location in the memory as given below:

RST 7.5, RST 6.5 and RST 5.5 are the restart interrupts which cause an internal restart to be automatically inserted.

<u>The TRAP (non maskable interrupt)</u> has the highest priority among interrupts.

The order of priority of interrupts is as follows:

- TRAP (Highest priority)
- RST 7.5
- RST 6.5
- RST 5.5
- INTR (Lowest priority).

<u>Reset Signals</u>

<u>RESET IN (Input):</u> It resets the program counter (PC) to 0. It also resets interrupt enable and HLDA flip-flops. The CPU is held in reset condition till RESET is not applied.

<u>RESET OUT (Output):</u> RESET OUT indicates that the CPU is being reset.

<u>Clock Signals</u>

<u>X1, X2 (Input):</u> X1 and X2 are terminals to be connected to an external crystal oscillator which drives an internal circuitry of the microprocessor. It is used to produce a suitable clock for the operation of microprocessor.

<u>CLK (Output):</u> CLK is a **clock output** for user, which can be used for other digital ICs. Its frequency is same at which processor operates.

<u>Serial I/O Signals</u>

<u>SID (Input):</u> SID is data line for **serial input**. The data on this line is loaded into the seventh bit of the accumulator when RIM instruction is executed.

<u>SOD (Output):</u> SOD is a data line for **serial output**. The seventh bit of the accumulator is output on SOD line when SIM instruction is executed.

UNIT 2

8085 Instructions

An **instruction** of computer is a command given to the computer to perform a specified operation on given data. Some instructions of Intel 8085 microprocessor are: MOV, MVI, LDA, STA, ADD, SUB, RAL, INR, MVI, etc.

Op-code and Operands

Each instruction contains two parts: Op-code (Operation code) and Operand.

The 1st part of an instruction which specifies the task to be performed by the computer is called Op-code.

The 2nd part of the instruction is the data to be operated on, and it is called Operand. The Operand (or data) given in the instruction may be in various forms such as 8-bit or 16-bit data, 8-bit or 16-bit address, internal registers or a register or memory location.

Instruction Word Size

A digital computer understands instruction written in binary codes (machine codes). The binary codes of all instructions are not of the same length.

According to the word size, the Intel 8085 instructions are classified into the following three types:

1. One byte instruction

2. Two byte instruction

3. Three byte instruction

1. One-byte instruction: Examples of one byte instructions are:

- **MOV A, B -** Move the content of the register B to register A.
- **ADD B** Add the content of register B to the content of the accumulator.

All the above two examples are only one byte long. All one-byte instructions contain information regarding operands in the op-code itself.

2. Two-byte instruction: In a two byte instruction the first byte of the instruction is its op-code and the second byte is either data or address.

Example:

- **MVI B, 05;** *the data 05H moved to register B.*

  06, 05.

The first byte 06 is the op-code for MVI B and second byte 05 is the data which is to be moved to register B.

3. Three-byte instruction: The first byte of the instruction is its op-code and the second and third bytes are either 16-bit data or 16-bit address.

Example:

- **LXI H, 2400H**; *Load H-L Pair with 2400H*

  21, 00, 24

The first byte 21 is the op-code for the instruction LXI H. The second 00 is 8 LSBs of the data (2400H), which is loaded into register L. The third byte 24 is 8 MSBs of the data (2400H), which is loaded into register H.

Instructions sets in 8085

Instructions sets in 8085 microprocessor can be classified into five groups. Those are

▪ Data transfer (copy) group
▪ Arithmetic group
▪ Logical group
▪ Branch group
▪ Machine control group.

Data transfer (copy) group:

As name suggests instructions in this group are used to copy data form one place to another. Data transfer can be possible different way. The different types of data transfer operations possible are cited below:

▪ Between two registers.

▪ Between a register and a memory location.

▪ A data byte can be transferred between a register and a memory location.

▪ Between an I/O device and the accumulator.

▪ Between a register pair and the stack.

It is very important to clear your concept that the term 'data transfer' is a misnomer—actually data is not transferred, but copied from source to destination.

Examples: MOV A,B

MVI A, 05

LXI H,C050

Arithmetic group:

As name suggest that instructions in this group are used for arithmetic operation. The arithmetic operations possible are addition, subtraction, increment and decrement.

Example: ADD B; SUB C; INR C; DCR C.

Logical group:

Instructions in this group is used to do the logical operation. The logical operations include AND, OR, EXOR, compare, complement.

Example: ANI 05; CMP B.

Shift and Rotate Instructions

Shift instructions: Shift instructions can perform two basic types of shift operations; the logical shift and the arithmetic shift. Also, each of these operations can be performed to the right or to the left.

Example: SHL, SHR, SAL, and, SAR instructions:

Branching/Jump instruction group:

In 8085 microprocessor branch operations are Jump, Call, Return and Restart instructions.

Example: JMP C050; CALL D050; RET.

The Jump instruction, which can transfer program control to a certain memory location.

These jump instructions can be divided into two categories

- Unconditional jump instructions (Ex:- JMP)
- Conditional jump instructions (Ex :- JNZ, JZ)

Machine control group:

This instructions are used to control the operation of 8085 microprocessor. The machine control operations are Halt, Interrupt and NOP (no operation).

Example: HLT; NOP.

Instruction Cycle

The time required to fetch an instruction and necessary data from memory and to execute it, is called an **instruction cycle**. Or the total time required to execute an instruction is given by:

IC = FC + EC

Where,

IC = Instruction Cycle

FC = Fetch Cycle

EC = Execute Cycle

Addressing modes in 8085 microprocessor

The way of specifying data to be operated by an instruction is called addressing mode.

Types of addressing modes – In 8085 microprocessor there are 5 types of addressing modes:

1. **Immediate Addressing Mode –** In immediate addressing mode the source operand is always data. If the data is 8-bit, then the instruction will be of 2 bytes, if the data is of 16-bit then the instruction will be of 3 bytes.

   Examples: MVI B 45 (move the data 45H immediately to register B) LXI H 3050 (load the H-L pair with the operand 3050H immediately) JMP address (jump to the operand address immediately)
2. **Register Addressing Mode –** In register addressing mode, the data to be operated is available inside the register(s) and register(s) is(are) operands. Therefore the operation is performed within various registers of the microprocessor.

   Examples: MOV A, B (move the contents of register B to register A)
   ADD B (add contents of registers A and B and store the result in register A)
   INR A (increment the contents of register A by one)
3. **Direct Addressing Mode** – In direct addressing mode, the data to be operated is available inside a memory location and that memory location is directly specified as an operand. The operand is directly available in the instruction itself.

   Examples: LDA 2050 (load the contents of memory location into accumulator A) LHLD address (load contents of 16-bit memory location into H-L register pair) IN 35 (read the data from port whose address is 01)
4. **Register Indirect Addressing Mode –** IN register indirect addressing mode, the data to be operated is available inside a memory location and that memory location is indirectly specified by a register pair.

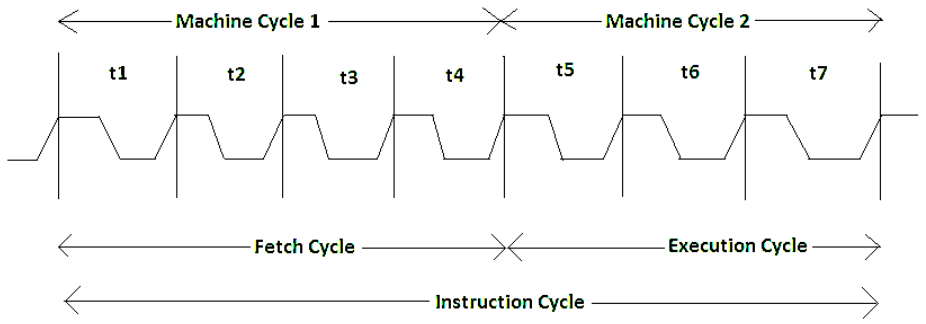
   Examples: MOV A, M (move the contents of the memory location pointed by the H-L pair to the accumulator) LDAX B (move contains of B-C register to the accumulator) LXIH 9570 (load immediate the H-L pair with the address of the location 9570)
5. **Implied/Implicit Addressing Mode –** In implied/implicit addressing mode the operand is hidden and the data to be operated is available in the instruction itself.

   Examples: CMA (finds and stores the 1's complement of the contains of accumulator A in A) RRC (rotate accumulator A right by one bit) RLC (rotate accumulator A left by one bit)

Instruction cycle in 8085 microprocessor

Time required to execute and fetch an entire instruction is called *instruction cycle*. It consists:

- **Fetch cycle –** The next instruction is fetched by the address stored in program counter (PC) and then stored in the instruction register.
- **Decode instruction –** Decoder interprets the encoded instruction from instruction register.
- **Reading effective address –** The address given in instruction is read from main memory and required data is fetched. The effective address depends on direct addressing mode or indirect addressing mode.
- **Execution cycle –** consists memory read (MR), memory write (MW), input output read (IOR) and input output write (IOW)



Instruction cycle in 8085 microprocessor

The time required by the microprocessor to complete an operation of accessing memory or input/output devices is called ***machine cycle.***

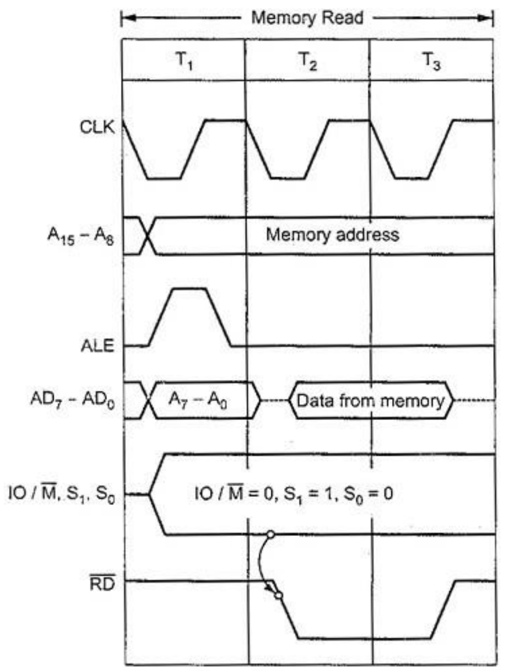
One time period of frequency of microprocessor is called ***t-state.***

A t-state is measured from the falling edge of one clock pulse to the falling edge of the next clock pulse.

Fetch cycle takes four t-states and execution cycle takes three t-states.

Memory Read (MR) machine cycle in 8085 Microprocessor.

Contents from a memory location are read during the memory read machine cycle (MRMC). This cycle is also known as the operand fetch machine cycle. But there are cases when MRMC is not used for operand fetch but for reading data at given memory location. This machine cycle spans over three T states. Each of these T states is explained here along with a timing diagram. The first three T states are almost the same as the first three T states of Opcode Fetch Machine Cycle.



1st T state

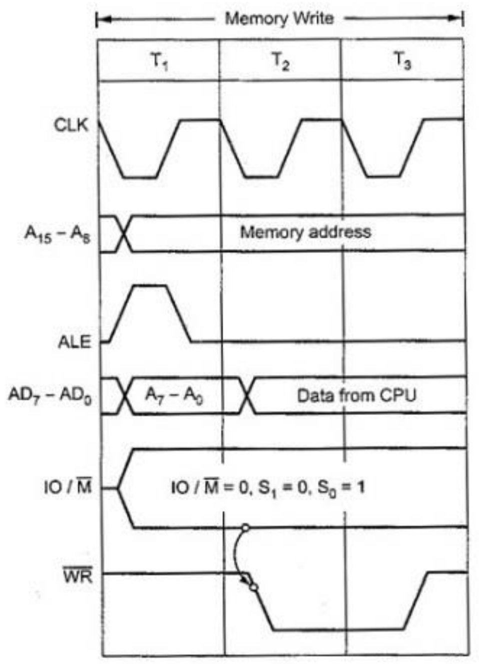
- Higher address bits loaded into **A8-A15.**
- Lower address bits loaded into **AD0-AD7**.
- ALE signal goes high in the beginning to indicate that **AD0-AD7** contains lower address bits.
- **IO/M** goes low since it is a memory operation.
- S1 and S0 become 1 and 0 respectively, indicating Memory Read Machine Cycle.
- **ALE** goes low by the end of the first T state. Lower address bits are expected to be latched by this time.

2nd and 3rd T states

- **RD** goes low, indicating the initiation of the read operation.
- Data is read from the memory location and is loaded into the data bus **AD0-AD7**. The data is loaded into the data bus at the beginning of the 2nd T state and exists until the end of the third T state.
- By the end of the third T state, **RD** goes high, indicating the end of the read operation.
- PC is incremented by 1 (only in cases described in the note in the red box above).

Memory write machine cycle

Contents are written to a memory location/stack during a memory write machine cycle (MWMC). This machine cycle spans over three T states. Each of these T states are explained here along with the timing diagram. PC is not incremented in this machine cycle. This is very similar to MRMC, except a few differences.



1st T state

- Higher address bits loaded into A8-A15.
- Lower address bits loaded into AD0-AD7.
- ALE signal goes high in the beginning to indicate that AD0-AD7 contains lower address bits.
- IO/M goes low since it is a memory operation.
- S1 and S0 become 0 and 1 respectively, indicating MWMC.
- ALE goes low by the end of the first T state. Lower address bits are expected to be latched by this time.

2nd and 3rd T states

- WR goes low, indicating the initiation of the write operation.
- Data to be written is loaded on the data bus at the beginning of the second T state and exists until the end of the third T state when the data is transferred from the data bus to the memory location.
- By the end of the third T state, WR goes high, indicating the end of the write operation. Thus, MWMC comes to an end.

The microprocessor cannot do anything by itself therefore; It needs to be linked with memory, extra peripherals, or IO devices. This linking is called Interfacing.

The interfacing of the I/O devices in 8085 can be done in two ways:

1. **Memory-Mapped I/O Interfacing :**

In this kind of interfacing, we assign a memory address that can be used in the same manner as we use a normal memory location.

2. **I/O Mapped I/O Interfacing :**

A kind of interfacing in which we assign an 8-bit address value to the input/output devices which can be accessed using IN and OUT instruction is called I/O Mapped I/O Interfacing.

<u>Difference between Memory-Mapped I/O Interfacing and I/O Mapped I/O Interfacing :</u>

Features	Memory Mapped IO	IO Mapped IO
Addressing	IO devices are accessed like any other memory location.	They cannot be accessed like any other memory location.
Address Size	They are assigned with 16-bit address values.	They are assigned with 8-bit address values.
Instructions Used	The instruction used are LDA and STA, etc.	The instruction used is IN and OUT.
Cycles	Cycles involved during operation are Memory Read, Memory Write.	Cycles involved during operation are IO read and IO writes in the case of IO Mapped IO.
Registers Communicating	Any register can communicate with the IO device in case of Memory Mapped IO.	Only Accumulator can communicate with IO devices in case of IO Mapped IO.
Space Involved	2^{16} IO ports are possible to be used for interfacing in case of Memory Mapped IO.	Only 256 I/O ports are available for interfacing in case of IO Mapped IO.
IO/M` signal	During writing or read cycles (IO/M` = 0) in case of Memory Mapped IO.	During writing or read cycles (IO/M` = 1) in case of IO Mapped IO.
Control Signal	No separate control signal required since we have unified memory space in the case of Memory Mapped IO.	Special control signals are used in the case of IO Mapped IO.
Arithmetic and Logical operations	Arithmetic and logical operations are performed directly on the data in the case of Memory Mapped IO.	Arithmetic and logical operations cannot be performed directly on the data in the case of IO Mapped IO.

UNIT 3

Looping, counting and indexing of 8085.

LOOPING:

- The programming technique used to instruct the microprocessor to repeat tasks is called looping.
- This task is accomplished by using jump instructions.

CLASSIFICATION OF LOOPS:

1. Continuous loop
2. Unconditional loop

CONTINUOUS LOOP:

- Repeats a task continuously.
- A continuous loop is set up by using the unconditional jump instruction
- A program with a continuous loop does not stop repeating the tasks until the system is reset.

CONDITIONAL LOOP:

- A conditional loop is set up by conditional jump instructions.
- These instructions check flags (Z, CY, P, S) and repeat the tasks if the conditions are satisfied.
- These loops include counting and indexing.

COUNTER

- A counter is a typical application of the conditional loop.
- A microprocessor needs a counter, flag to accomplish the looping task.
- Counter is set up by loading an appropriate count in a register.
- Counting is performed by either increment or decrement the counter.
- Loop is set up by a conditional jump instruction.
- End of counting is indicated by a flag.

INDEXING:

- Pointing of referencing objects with sequential numbers.
- Data bytes are stored in memory locations and those data bytes are referred to by their memory locations.

Delay generation in 8085.

The delay will be used in different places to simulate clocks, or counters or some other area.

When the delay subroutine is executed, the microprocessor does not execute other tasks. For the delay we are using the instruction execution times. executing some instructions in a loop, the delay is generated. There are some methods of generating delays. These methods are as follows.

- Using NOP instructions
- Using 8-bit register as counter
- Using 16-bit register pair as counter.

Using NOP instructions:

One of the main usages of NOP instruction is in delay generation. The NOP instruction is taking four clock pulses to be fetching, decoding and executing. If the 8085 MPU is working on 6MHz clock frequency, then the internal clock frequency is 3MHz. So from that we can easily determine that each clock period is 1/3 of a microsecond. So the NOP will be executed in 1/3 * 4 = 1.333µs. If we use the entire memory with NOP instruction, then 64K NOP instructions will be executed. Then the overall delay will be 2^{16} * 1.333µs = 87359.488µs, though the time is not so large and the program size is also large. So this type of NOP instruction can be used to generate a short time delay for few milliseconds.

Using 8-bit register as counter:

Counter is another approach to generate a time delay. In this case the program size is smaller. So in this approach we can generate more time delay in less space. The following program will demonstrate the time delay using 8-bit counter.

```
        MVI  B, FFH
LOOP:   DCR  B
        JNZ  LOOP
        RET
```

Here the first instruction will be executed once, it will take 7 T-states. DCR C instruction takes 4 T-states. This will be executed 255 (FF) times. The JNZ instruction takes 10 T-states when it jumps (It jumps 254 times), otherwise it will take 7 T-States. And the RET instruction takes 10 T-States.

7 + ((4*255) + (10*254)) + 7 + 10 = 3584. So the time delay will be 3584 * 1/3μs = 1194.66μs. So when we need some small delay, then we can use this technique with some other values in the place of FF.

Using 16-bit register-pair as counter:

Instead of using 8-bit counter, we can do that kind of task using 16-bit register pair. Using this method more time delay can be generated. This method can be used to get more than 0.5 seconds delay. Let us see and example.

```
      LXI  B,FFFFH
LOOP: DCX  B
      MOV  A,B
      ORA  C
      JNZ  LOOP
      RET
```

In the above table we have placed the T-States. From that table, if we calculate the time delay, it will be like this:

10 + (6 + 4 + 4 + 10) * 65535H – 3 + 10 = 17 + 24 * 65535H = 1572857. So the time delay will be 1572857 * 1/3μs = 0.52428s. Here we are getting nearly 0.5s delay.

In different program, we need 1s delay. For that case, this program can be executed twice. We can call the Delay subroutine twice or use another outer loop for two-time execution.

Stack in 8085

The stack is a reserved area of the memory in RAM where we can store temporary information. Interestingly, the stack is a shared resource as it can be shared by the microprocessor and the programmer. The programmer can use the stack to store data. And the microprocessor uses the stack to execute subroutines. The 8085 has a 16-bit register known as the ‘Stack Pointer’.

This register’s function is to hold the memory address of the stack. This control is given to the programmer. The programmer can decide the starting address of the stack by loading the address into the stack pointer register at the beginning of a program uses the instruction LXI SP.

The stack works on the principle of First In Last Out. The memory location of the most recent data entry on the stack is known as the Stack Top.

We use two main instructions to control the movement of data into a stack and from a stack. These two instructions are PUSH and POP.

PUSH – This is the instruction we use to write information on the stack. POP – This is the instruction we use to read information from the stack.

```
LXI SP, 8000H
LXI H, 1234H
PUSH H
POP D
HLT
```

Explanation of the code

LXI SP, 8000H – The address of the stack pointer is set to 8000H by loading the number into the stack pointer register.

LXI H, 1234H – Next, we add a number to the HL pair. The most significant two bits will enter the H register. The least significant two bits will enter the L register.

PUSH H – The PUSH command will push the contents of the H register first to the stack. Then the contents of the L register will be sent to the stack. So the new stack top will hold 34H.

POP D – The POP command will remove the contents of the stack and store them to the DE register pair. The top of the stack clears first and enters the E register. The new top of the stack is 12H now. This one clears last and enters the D register. The contents of the DE register pair is now 1234H.

HLT – HLT indicates that the program execution needs to stop.

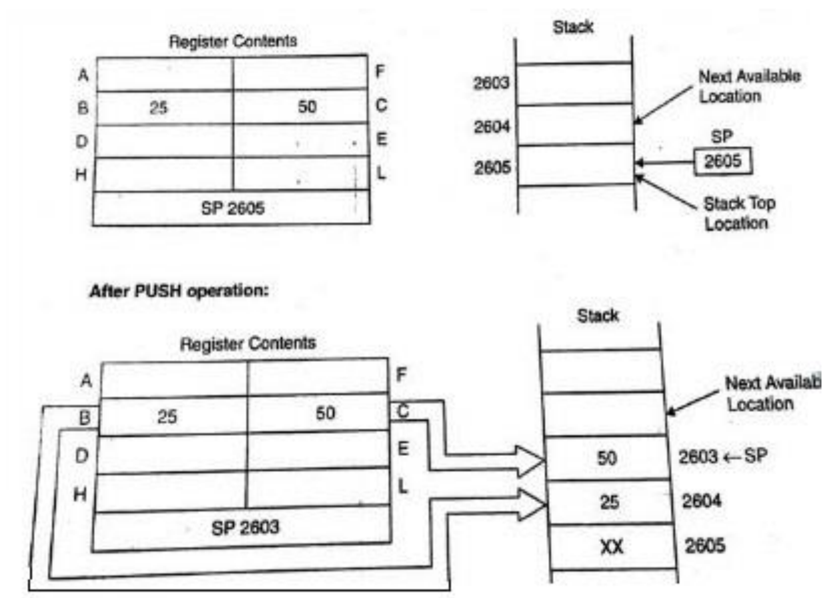
On a stack, we can perform two operations. PUSH and POP. In case of PUSH operation, the SP register gets decreased by 2 and new data item used to insert on to the top of the stack. On the other hand, in case of POP operation, the data item will have to be deleted from the top of the stack and the SP register will get increased by the value of 2.

PUSH and POP operation in 8085.

- **PUSH Rp**
- PUSH instruction can be explained with an example

```
LXI SP,2605H
LXI B, 2550H
PUSH B
Delay Counter
POP B
```

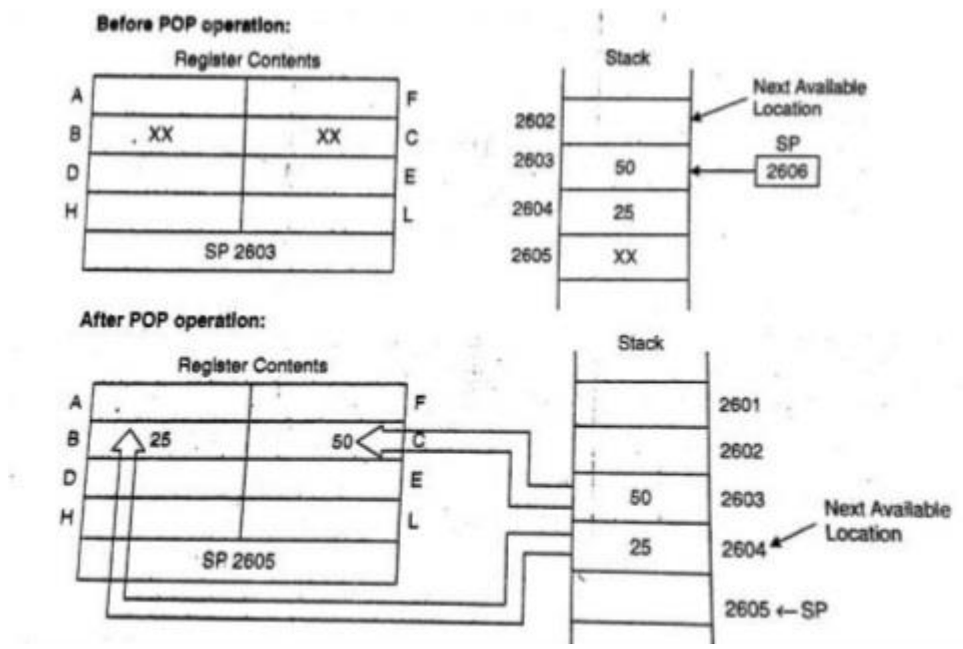
- Step 1: LXI SP, 2605 will initialize SP register 2605
- Step 2: LXI B, 2550H will initialize or load BC register pair with 2550H data so B = 25 and C = 50.
- Step 3: The execution of PUSH b instruction will be The stack pointer is decreased by one to 2604H and the contents of the B register are copied to memory location 2604H.
- The stack pointer is again decreased by one to 2603H and the contents of the C register are copied to memory location 2603H.
- The contents of the register pair BC are not destroyed. However , BC is made available for the delay counter.
- The execution of PUSH operation is shown in Figure.



- **POP Rp**
- After the delay count, the instruction POP B restores the original contents of the register pair BC.
- The execution of POP B instruction will be
- The contents of the top of the stack location shown by the stack pointer are copied in the C register and the stack pointer is increased by one to 2604H

  [C] [SP], [SP] [SP] + 1
- The contents of the top of the stack are copied in the B register and the stack pointer is increased by one

  [B] [SP], [SP] [SP] + 1
- The contents of the memory locations 2603H and2604H are not destroyed until some other data bytes are stored in these locations.
- The execution of POP operation is shown in Figure.



<u>Subroutines</u>

- A subroutine is a group of instructions that will be used repeatedly in different locations of the program.
- Rather than repeat the same instructions several times, they can be grouped into a subroutine that is called from the different locations.
- In Assembly language, a subroutine can exist anywhere in the code.
- However, it is customary to place subroutines separately from the main program.

<u>The 8085 has two instructions for dealing with subroutines.</u>

- The **CALL** instruction is used to redirect program execution to the subroutine.
- The **RET** instruction is used to **return** the execution to the calling routine.

CALL instruction:

- ***CALL 4000H (3 byte instruction)*** – When CALL instruction is fetched, the microprocessor knows that the next two Memory locations contains 16bit subroutine address in the memory.

RET instruction:

- ***RET (1 byte instruction)*** – Retrieve the return address from the top of the stack – Load the program counter with the return address.

Interrupts in 8085.

Interrupts are the signals generated by the external 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.

Interrupt is a mechanism by which an I/O or an instruction can suspend the normal execution of processor and get itself serviced. Generally, a particular task is assigned to that interrupt signal. In the microprocessor based system the interrupts are used for data transfer between the peripheral devices and the microprocessor.

Interrupt Service Routine (ISR)

A small program or a routine that when executed services the corresponding interrupting source is called as an ISR.

Maskable /Non-Maskable Interrupt

An interrupt that can be disabled by writing some instruction is known as Maskable Interrupt otherwise it is called Non-Maskable Interrupt.

There are 6 pins available in 8085 for interrupt:

1. TRAP
2. RST 7.5
3. RST6.5
4. RST5.5
5. INTR
6. INTA

When there is an interrupt requests to the Microprocessor then after accepting the interrupts Microprocessor send the INTA (active low) signal to the peripheral. The vectored address of particular interrupt is stored in program counter. The processor executes an interrupt service routine (ISR) addressed in program counter.

There are two types of interrupts used in 8085 Microprocessor:

- Hardware Interrupts
- Software Interrupts

Software Interrupts

A software interrupts is a particular instructions that can be inserted into the desired location in the program. There are eight Software interrupts in 8085 Microprocessor. From RST0 to RST7.

Software Interrupt is invoked by the use of INT instruction. This event immediately stops execution of the program and passes execution over to the INT handler. The INT handler is usually a part of the operating system and determines the action to be taken. It occurs when an application program terminates or requests certain services from the operating system.

1. RST0
2. RST1
3. RST2
4. RST3
5. RST4
6. RST5
7. RST6
8. RST7

They allow the microprocessor to transfer program control from the main program to the subroutine program. After completing the subroutine program, the program control returns back to the main program.

Hardware Interrupt.

Hardware Interrupt is caused by some hardware device such as request to start an I/O, a hardware failure or something similar. Hardware interrupts were introduced as a way to avoid wasting the processor's valuable time in polling loops, waiting for external events.

For example, when an I/O operation is completed such as reading some data into the computer from a tape drive.

There are 6 interrupt pins in the microprocessor used as Hardware Interrupts given below:

1. TRAP
2. RST7.5
3. RST6.5
4. RST5.5
5. INTR

TRAP

It is non maskable edge and level triggered interrupt. TRAP has the highest priority and vectors interrupt. Edge and level triggered means that the TRAP must go high and remain high until it is acknowledged. In case of sudden power failure, it executes a ISR and send the data from main memory to backup memory.

INTR

It is level triggered and maskable interrupt. The following sequence of events occurs when INTR signal goes high:

1. The 8085 checks the status of INTR signal during execution of each instruction.
2. If INTR signal is high, then 8085 complete its current instruction and sends active low interrupt acknowledge signal, if the interrupt is enabled.
3. On receiving the instruction, the 8085 save the address of next instruction on stack and execute received instruction.

INTA is not an interrupt. INTA is used by the microprocessor for sending the acknowledgement.

TRAP has highest priority and RST 7.5 has second highest priority and so on.

Interfacing:

8255A - Programmable Peripheral Interface:

The 8255A is a general purpose programmable I/O device designed to transfer the data from I/O to interrupt I/O under certain conditions as required. It can be used with almost any microprocessor.

It consists of three 8-bit bidirectional I/O ports (24I/O lines) which can be configured as per the requirement.

Ports of 8255A

8255A has three ports, i.e., PORT A, PORT B, and PORT C.

- **Port A** contains one 8-bit output latch/buffer and one 8-bit input buffer.
- **Port B** is similar to PORT A.
- **Port C** can be split into two parts, i.e. PORT C lower (PC0-PC3) and PORT C upper (PC7-PC4) by the control word.

These three ports are further divided into two groups, i.e. Group A includes PORT A and upper PORT C. Group B includes PORT B and lower PORT C. These two groups can be programmed in three different modes, i.e. the first mode is named as mode 0, the second mode is named as Mode 1 and the third mode is named as Mode 2.

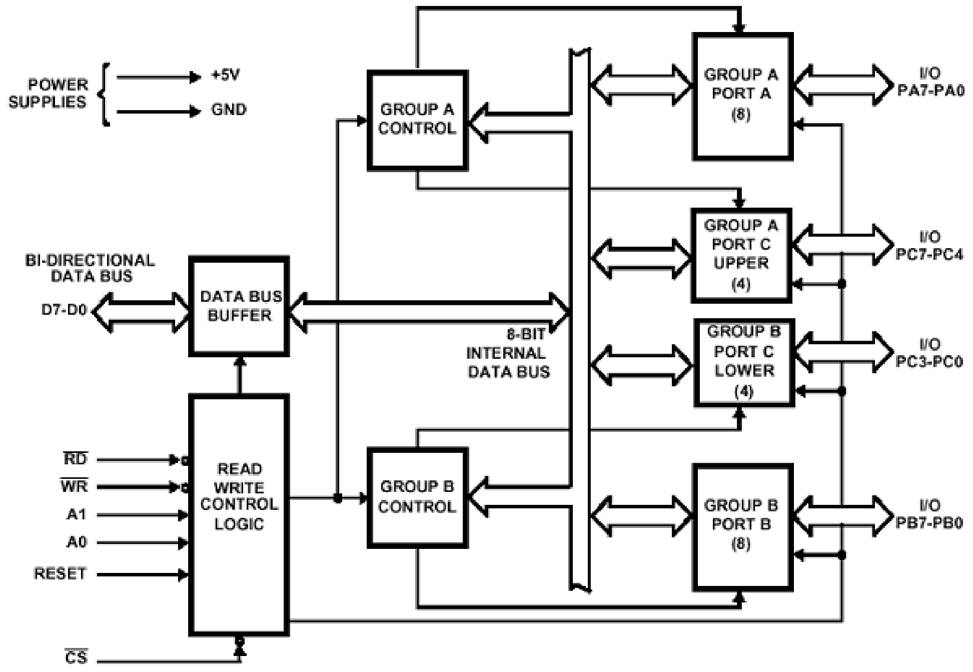
Operating Modes

8255A has three different operating modes –

- **Mode 0** – In this mode, Port A and B is used as two 8-bit ports and Port C as two 4-bit ports. Each port can be programmed in either input mode or output mode where outputs are latched and inputs are not latched. Ports do not have interrupt capability.
- **Mode 1** – In this mode, Port A and B is used as 8-bit I/O ports. They can be configured as either input or output ports. Each port uses three lines from port C as handshake signals. Inputs and outputs are latched.
- **Mode 2** – In this mode, Port A can be configured as the bidirectional port and Port B either in Mode 0 or Mode 1. Port A uses five signals from Port C as handshake signals for data transfer. The remaining three signals from Port C can be used either as simple I/O or as handshake for port B.

8255 Architecture

The following figure shows the architecture of 8255A.



Features of 8255A

The prominent features of 8255A are as follows −

- It consists of 3 8-bit IO ports i.e. PA, PB, and PC.
- Address/data bus must be externally demux'd.
- It is TTL compatible.
- It has improved DC driving capability.

8255 pin configuration:

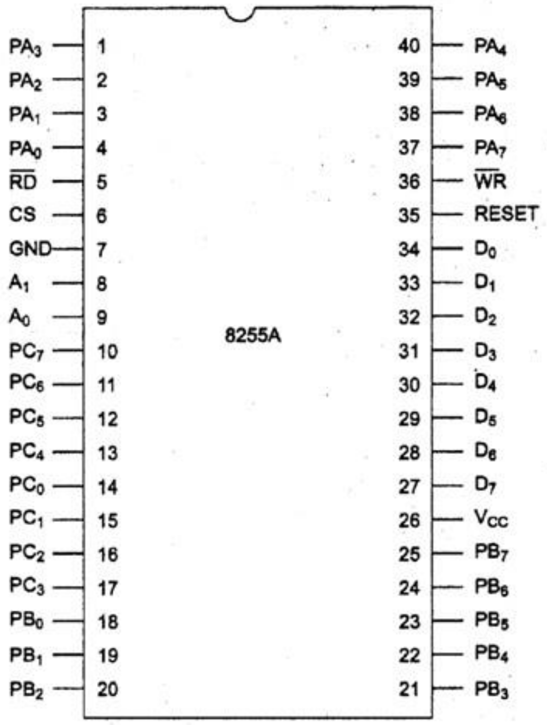
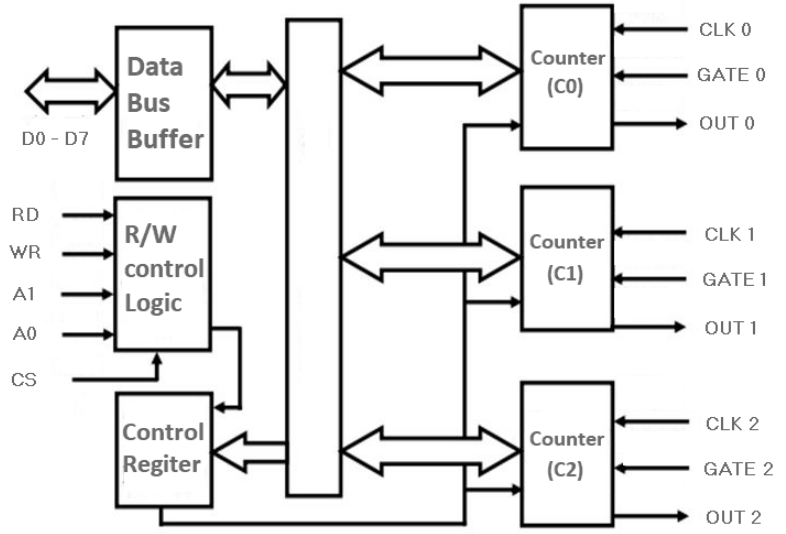


Fig. 5.17(b) *8255A Pin Configuration*

8254 programmable interval timer:

8254 is a device designed to solve the timing control problems in a microprocessor. It has 3 independent counters, each capable of handling clock inputs up to 10 MHz and size of each counter is 16 bit. It operates in +5V regulated power supply and has 24 pin signals. All modes are software programmable.



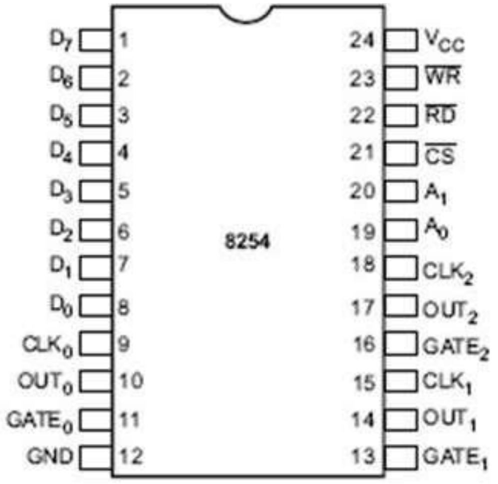
It has 3 counters each with two inputs (Clock and Gate) and one output. Gate is used to enable or disable counting.

When any value of count is loaded and value of gate is set (1), after every step value of count is decremented by 1 until it becomes zero.

Depending upon the value of CS, A1 and A0 we can determine addresses of selected counter.

CS	A1	A0	SELECETION
0	0	0	C0
0	0	1	C1
0	1	0	C2
0	1	1	Control Register

8254 pin configuration:



Applications

1. To generate accurate time delay
2. As an event counter
3. Square wave generator
4. Rate generator
5. Digital one shot

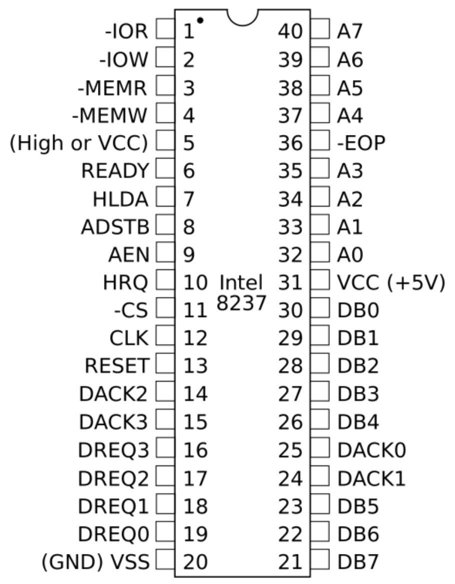
Introduction of 8237

- Direct Memory Access (DMA) is a method of allowing data to be moved from one location to another in a computer without intervention from the central processor (CPU).
- It is also a fast way of transferring data within (and sometimes between) computer.
- The DMA I/O technique provides direct access to the memory while the microprocessor is temporarily disabled.
- The DMA controller temporarily borrows the address bus, data bus and control bus from the microprocessor and transfers the data directly from the external devices to a series of memory locations (and vice versa).

Basic DMA Operation:

- Two control signals are used to request and acknowledge a direct memory access (DMA) transfer in the microprocessor-based system.
    1. The HOLD signal as an input(to the processor) is used to request a DMA action.
    2. The HLDA signal as an output that acknowledges the DMA action.
- When the processor recognizes the hold, it stops its execution and enters hold cycles.
- HOLD input has higher priority than INTR or NMI.
- The only microprocessor pin that has a higher priority than a HOLD is the RESET pin.
- HLDA becomes active to indicate that the processor has placed its buses at high-impedance state.

The 8237 DMA Controller



- The 8237 supplies memory & I/O with control signals and memory address information during the DMA transfer.
- It is actually a special-purpose microprocessor whose job is high-speed data transfer between memory and I/O
- 8237 is not a discrete component in modern microprocessor-based systems.
- It appears within many system controller chip sets
- 8237 is a four-channel device compatible with 8086/8088, adequate for small systems.
- Expandable to any number of DMA channel inputs
- 8237 is capable of DMA transfers at rates up to 1.6MB per second.
- Each channel is capable of addressing a full 64K-byte section of memory.
-

UNIT – IV

Introduction to 8086/8088 microprocessors:

8086 Microprocessor is an enhanced version of 8085Microprocessor that was designed by Intel in 1976. It is a 16-bit Microprocessor having 20 address lines and16 data lines that provides up to 1MB storage. It consists of powerful instruction set, which provides operations like multiplication and division easily.

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.

Features of 8086

The most prominent features of a 8086 microprocessor are as follows −

- It has an instruction queue, which is capable of storing six instruction bytes from the memory resulting in faster processing.
- It was the first 16-bit processor having 16-bit ALU, 16-bit registers, internal data bus, and 16-bit external data bus resulting in faster processing.
- It is available in 3 versions based on the frequency of operation −
  - 8086 → 5MHz
  - 8086-2 → 8MHz
  - 8086 (c)-1 → 10 MHz

- It uses two stages of pipelining, i.e. Fetch Stage and Execute Stage, which improves performance.
- Fetch stage can prefetch up to 6 bytes of instructions and stores them in the queue.
- Execute stage executes these instructions.
- It has 256 vectored interrupts.
- It consists of 29,000 transistors.

Comparison between 8085 & 8086 Microprocessor

- **Size** – 8085 is 8-bit microprocessor, whereas 8086 is 16-bit microprocessor.
- **Address Bus** – 8085 has 16-bit address bus while 8086 has 20-bit address bus.
- **Memory** – 8085 can access up to 64Kb, whereas 8086 can access up to 1 Mb of memory.
- **Instruction** – 8085 doesn't have an instruction queue, whereas 8086 has an instruction queue.
- **Pipelining** – 8085 doesn't support a pipelined architecture while 8086 supports a pipelined architecture.
- **I/O** – 8085 can address 2^8 = 256 I/O's, whereas 8086 can access 2^{16} = 65,536 I/O's.
- **Cost** – The cost of 8085 is low whereas that of 8086 is high.

8086 internal architecture.

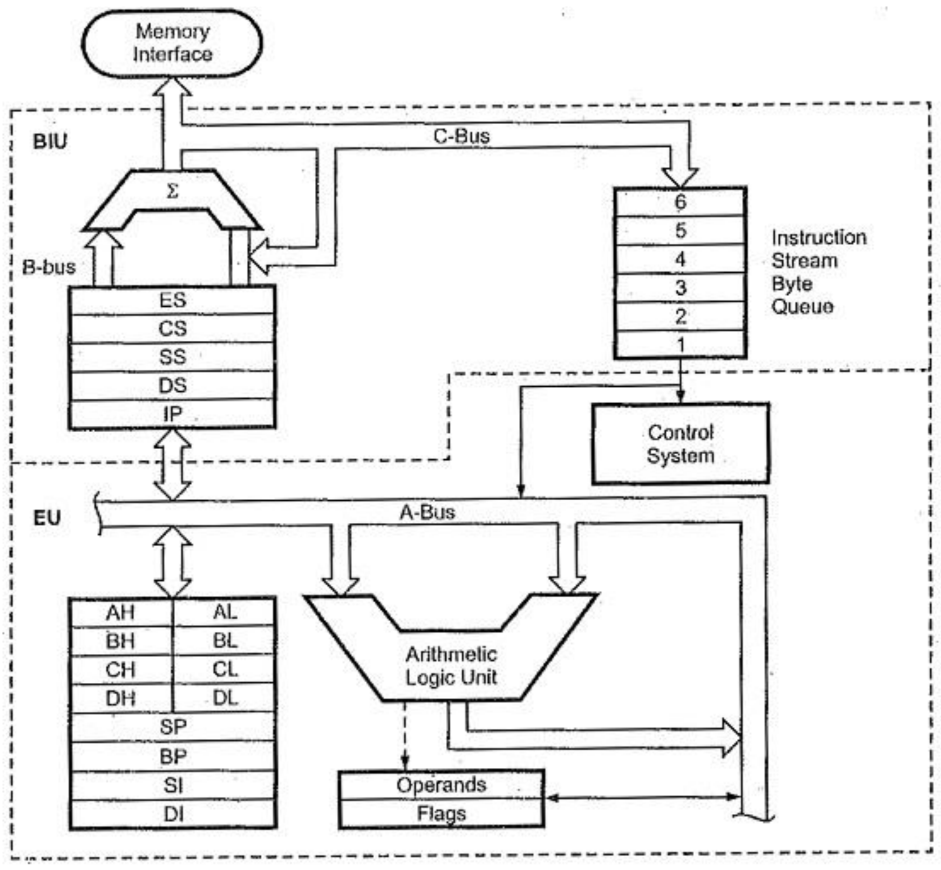


Fig. 6.2 8086 Internal block diagram

8086 Microprocessor is divided into two functional units, i.e., EU (Execution Unit) and BIU (Bus Interface Unit).

EU (Execution Unit)

Execution unit gives instructions to BIU stating from where to fetch the data and then decode and execute those instructions. Its function is to control operations on data using the instruction decoder & ALU. EU has no direct connection with system buses as shown in the above figure, it performs operations over data through BIU.

ALU

It handles all arithmetic and logical operations, like +, −, ×, /, OR, AND, NOT operations.

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 flags and they are divided into 2 groups − Conditional Flags and Control Flags.

Conditional Flags

It represents the result of the last arithmetic or logical instruction executed. Following is the list of conditional flags −

- **Carry flag** − This flag indicates an overflow condition for arithmetic operations.
- **Auxiliary flag** − When an operation is performed at ALU, it results in a carry/barrow from lower nibble (i.e. D0 – D3) to upper nibble (i.e. D4 – D7), then this flag is set, i.e. carry given by D3 bit to D4 is AF flag. The processor uses this flag to perform binary to BCD conversion.
- **Parity flag** − This flag is used to indicate the parity of the result, i.e. when the lower order 8-bits of the result contains even number of 1's, then the Parity Flag is set. For odd number of 1's, the Parity Flag is reset.
- **Zero flag** − This flag is set to 1 when the result of arithmetic or logical operation is zero else it is set to 0.
- **Sign flag** − This 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** − This flag represents the result when the system capacity is exceeded.

Control Flags

Control flags controls the operations of the execution unit. Following is the list of control flags −

- **Trap flag** − It is used for single step control and allows the user to execute one instruction at a time for debugging. If it is set, then the program can be run in a single step mode.
- **Interrupt flag** − It is an interrupt enable/disable flag, i.e. used to allow/prohibit the interruption of a program. It is set to 1 for interrupt enabled condition and set to 0 for interrupt disabled condition.
- **Direction flag** − It is used in string operation. As the name suggests when it is set then string bytes are accessed from the higher memory address to the lower memory address and vice-a-versa.

General purpose register

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. The valid register pairs are AH and AL, BH and BL, CH and CL, and DH and DL. It is referred to the AX, BX, CX, and DX respectively.

- **AX register** − It is also known as accumulator register. It is used to store operands for arithmetic operations.

- **BX register** – It is used as a base register. It is used to store the starting base address of the memory area within the data segment.
- **CX register** – It is referred to as counter. It is used in loop instruction to store the loop counter.
- **DX register** – This register is used to hold I/O port address for I/O instruction.

Stack pointer register

It is a 16-bit register, which holds the address from the start of the segment to the memory location, where a word was most recently stored on the stack.

BIU (Bus Interface Unit)

BIU takes care of all data and addresses transfers on the buses for the EU like sending addresses, fetching instructions from the memory, reading data from the ports and the memory as well as writing data to the ports and the memory. EU has no direction connection with System Buses so this is possible with the BIU. EU and BIU are connected with the Internal Bus.

It has the following functional parts –

- **Instruction queue** – BIU contains the instruction queue. BIU gets upto 6 bytes of next instructions and stores them in the instruction queue. When EU executes instructions and is ready for its next instruction, then it simply reads the instruction from this instruction queue resulting in increased execution speed.
- Fetching the next instruction while the current instruction executes is called **pipelining**.
- **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. It also contains 1 pointer register IP, which holds the address of the next instruction to executed by the EU.

  ✓ **CS** – It stands for Code Segment. It is used for addressing a memory location in the code segment of the memory, where the executable program is stored.

  ✓ **DS** – It stands for Data Segment. It consists of data used by the program andis accessed in the data segment by an offset address or the content of other register that holds the offset address.

  ✓ **SS** – It stands for Stack Segment. It handles memory to store data and addresses during execution.

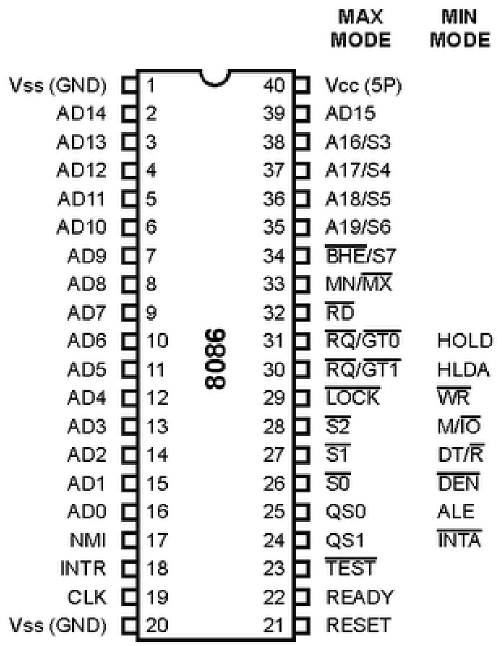
  ✓ **ES** – It stands for Extra Segment. ES is additional data segment, which is used by the string to hold the extra destination data.

- **Instruction pointer** – It is a 16-bit register used to hold the address of the next instruction to be executed.

The 8086 microprocessor supports 8 types of instructions:

- Data Transfer Instructions
- Arithmetic Instructions
- Bit Manipulation Instructions
- String Instructions
- Program Execution Transfer Instructions (Branch & Loop instructions)
- Processor Control Instructions
- Iteration Control Instructions
- Interrupt Instructions

Pin diagram of 8086:



Addressing modes of 8086:

The different ways in which a source operand is denoted in an instruction is known as addressing modes. There are 8 different addressing modes in 8086 programming.

Immediate addressing mode

The addressing mode in which the data operand is a part of the instruction itself is known as immediate addressing mode.

```
Examples :- MOV CX, 4929 H, ADD AX, 2387 H,  MOV AL,
FFH
```

Register addressing mode

It means that the register is the source of an operand for an instruction.

```
Example :- MOV CX, AX   ; copies the contents of the
16-bit AX register into ; the 16-bit CX register),
```

Direct addressing mode

The addressing mode in which the effective address of the memory location is written directly in the instruction.

```
Example :- MOV AX, [1592H], MOV AL, [0300H]
```

Register indirect addressing mode

This addressing mode allows data to be addressed at any memory location through an offset address held in any of the following registers: BP, BX, DI & SI.

```
Example

MOV AX, [BX]  ; Suppose the register BX contains
4895H, then the contents

                  ; 4895H are moved to AX
```

Based addressing mode

In this addressing mode, the offset address of the operand is given by the sum of contents of the BX/BP registers and 8-bit/16-bit displacement.

```
Example :- MOV DX, [BX+04], ADD CL, [BX+08]
```

Indexed addressing mode

In this addressing mode, the operands offset address is found by adding the contents of SI or DI register and 8-bit/16-bit displacements.

```
Example :- MOV BX, [SI+16], ADD AL, [DI+16]
```

Based-index addressing mode

In this addressing mode, the offset address of the operand is computed by summing the base register to the contents of an Index register.

`Example :- ADD CX, [AX+SI], MOV AX, [AX+DI]`

Based indexed with displacement mode

In this addressing mode, the operands offset is computed by adding the base register contents. An Index registers contents and 8 or 16-bit displacement.

`Example :- MOV AX, [BX+DI+08], ADD CX, [BX+SI+16]`

Memory Segmentation in 8086 Microprocessor

Segmentation is the process in which the main memory of the computer is divided into different segments and each segment has its own base address. It is basically used to enhance the speed of execution of the computer system, so that processor is able to fetch and execute the data from the memory easily and fast.

Need for Segmentation

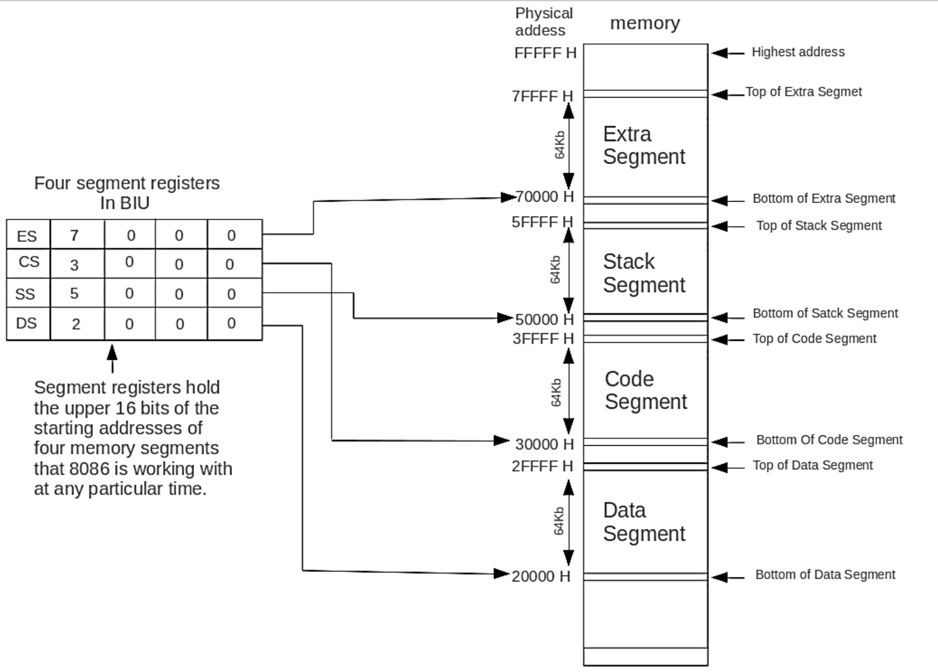
The Bus Interface Unit (BIU) contains four 16 bit special purpose registers (mentioned below) called as Segment Registers.

- **Code segment register (CS):** is used for addressing memory location in the code segment of the memory, where the executable program is stored.
- **Data segment register (DS):** points to the data segment of the memory where the data is stored.
- **Extra Segment Register (ES):** also refers to a segment in the memory which is another data segment in the memory.
- **Stack Segment Register (SS):** is used for addressing stack segment of the memory. The stack segment is that segment of memory which is used to store stack data.

The number of address lines in 8086 is 20, 8086 BIU will send 20bit address, so as to access one of the 1MB memory locations. The four segment registers actually contain the upper 16 bits of the starting addresses of the four memory segments of 64 KB each with which the 8086 is working at that instant of time. A segment is a logical unit of memory that may be up to 64 kilobytes long. Each segment is made up of contiguous memory locations. It is independent, separately addressable unit. Starting address will always be changing. It will not be fixed.

Note that the 8086 does not work the whole 1MB memory at any given time. However it works only with four 64KB segments within the whole 1MB memory.

Below is the one way of positioning four 64 kilobyte segments within the 1M byte memory space of an 8086.



Types Of Segmentation

1. **Overlapping Segment** – A segment starts at a particular address and its maximum size can go up to 64kilobytes. But if another segment starts along this 64kilobytes location of the first segment, then the two are said to be *Overlapping Segment*.

2. **Non-Overlapped Segment –** A segment starts at a particular address and its maximum size can go up to 64kilobytes. But if another segment starts before this 64kilobytes location of the first segment, then the two segments are said to be *Non-Overlapped Segment*.

Advantages of the Segmentation:

The main advantages of segmentation are as follows:

- It provides a powerful memory management mechanism.
- Data related or stack related operations can be performed in different segments.
- Code related operation can be done in separate code segments.
- It allows to processes to easily share data.
- It allows to extend the address ability of the processor, i.e. segmentation allows the use of 16 bit registers to give an addressing capability of 1 Megabytes. Without segmentation, it would require 20 bit registers.
- It is possible to enhance the memory size of code data or stack segments beyond 64 KB by allotting more than one segment for each area.

Memory address calculation in 8086:

8086 has a concept of Memory Segmentation. It is a method where the whole memory is segmented (divided) into smaller parts called segments. These segments are

- Code Segment (CS)
- Stack Segment (SS)
- Data Segment (DS)
- Extra Segment (ES)

Each Segment has a corresponding 16-bit Segment Register which holds the Base Address (starting Address) of the Segment.

At any given time, 8086 can address **16-bit x 64KB = 256 KB** of memory chunk out of 1MB.

8086 has 20bit address line. So the maximum value of address that can be addressed by 8086 is 2^{20} = 1MB.

*So 8086 can address the locations ranging between **00000 H to FFFFF H**. This 1MB memory is divided into 16 logical segments, each with a memory of 64KB.*

To locate any address in the memory bank, it needs the Physical address of that memory location. It cannot get the 20-bit Physical address using the 8086 Address Line or 16-bit Segment Registers alone.

In order to access memory location, you cannot pass 20-bit address directly to the processor. You need to tell the 16-bit address with respect to the segment. This 16-bit address with respect to the part (segment of 64KB) of the memory bank is called the offset.

So, **Physical Address = Base Address + Offset.**

Suppose the Data Segment holds the Base Address as 1000H and the data you need is present in the 0020H memory location (Offset) of the Data Segment. The calculation of the actual address is done as follows.

1. Left shift the 16-bit address present in the segment register by 4-bits

   0001 0000 0000 0000 (0000)

2. Add the 16-bit offset address to this shifted base address

```
  0001 0000 0000 0000 (0000)
+ 0000 0000 0010 0000
 ----------------------------------------
  0001 0000 0000 0010 0000
```

So the actual address turns out to be 10020h.

At any point of time we can change the base address of the segment registers and use the memory locations in those segments using the offset.