# Microprocessor
# Chapter-2
# 8085 Architecture and Instruction Set

# Introduction

8085 is an 8 bit microprocessor designed by Intel in 1977 using NMOS technology.
The acronym of NMOS is a negative channel metal oxide semiconductor; it is pronounced as en-moss. It is a kind of semiconductor that charges negatively.
It has 16 bit address bus which can address upto 64 KB. So, it has 16 bit program counter.
It has 16 bit stack pointer.
It has 6 general purpose 8 bit registers. They are: B, C, D, E, H, L. These six registers can be arranged in pairs: BC, DE, HL to store 16 bit of data.

# 8085 Microprocessor Architecture & Functional Units

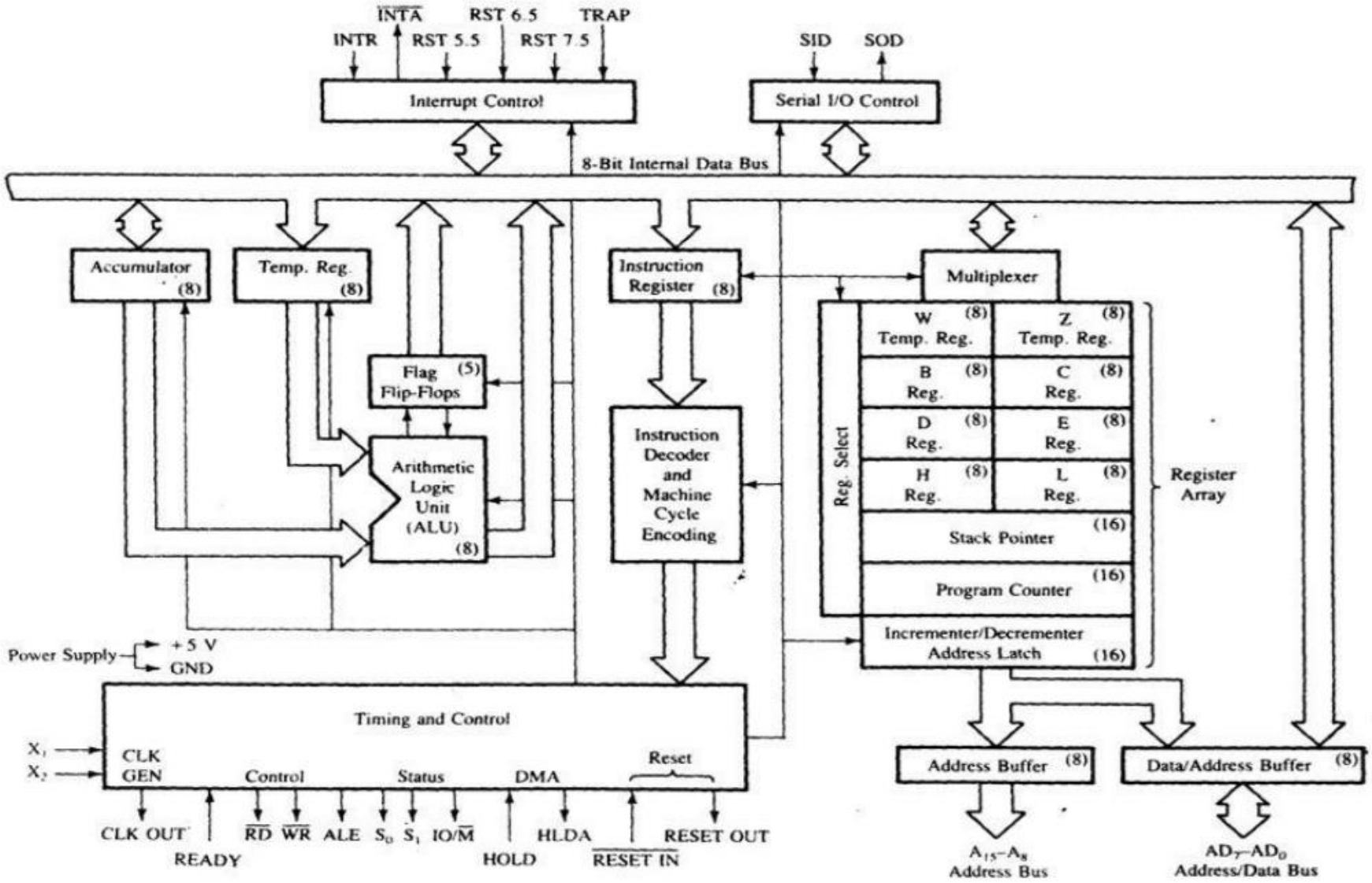



**Fig: Block Diagram of 8085**

# Introduction

8085 consists of the following functional units −

1) **Accumulator** It is an 8-bit register used to perform arithmetic, logical, I/O & LOAD/STORE operations. It is connected to internal data bus & ALU.
2) **Arithmetic and logic unit** As the name suggests, it performs arithmetic and logical operations like Addition, Subtraction, AND, OR, etc. on 8-bit data.
3) **General purpose register** There are 6 general purpose registers in 8085 processors, i.e. B, C, D, E, H & L. Each register can hold 8-bit data. These registers can work in pair to hold 16-bit data and their pairing combination is like B-C, D-E & H-L.
4) **Program counter** It is a 16-bit register used to store the memory address location of the next instruction to be executed. Microprocessor increments the program whenever an instruction is being executed, so that the program counter points to the memory address of the next instruction that is going to be executed.

# Introduction

8085 consists of the following functional units −

1) **Stack pointer** It is also a 16-bit register works like stack, which is always incremented/decremented push & pop operations.
2) **Temporary register** It is an 8-bit register, which holds the temporary data of arithmetic and logical operations.
3) **Flag register** It is an 8-bit register having five 1-bit flip-flops, which holds either 0 or 1 depending upon the result stored in the accumulator. It gives the status of the current result.

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

These are the set of 5 flip-flops –

- Sign (S)
- Zero (Z)
- Auxiliary Carry (AC)
- Parity (P)
- Carry (C)

# Introduction

**Sign flag(S):** After any operation if result is negative sign flag becomes set i.e. if result is positive sign flag becomes reset i.e. 0.
**Zero flag (Z):** After any arithmetic or logical operation if result is 0 the zero flag becomes set i.e. 1 otherwise it becomes reset i.e. 0.
**Auxiliar carry flag (AC):** if intermediate carry is generated this flag is set to 1 otherwise it is reset to 0.
**Parity flag (P):** If after any arithmetic or logical operation the result has be even parity, an even number of 1 bits, the parity register becomes set i.e. 1, otherwise it becomes reset.
**Carry Flat (CY):** Carry is generated when performing n bit operation and the result is more than n bits, then this flag becomes set i.e. 1 otherwise it becomes reset i.e. 0. During subtraction (A-B), if A>B it becomes reset and if A<B it becomes set. So carry flag is also called borrow flag.

# Introduction

**Sign flag(S):** After any operation if result is negative sign flag becomes set i.e. if result is positive sign flag becomes reset i.e. 0.
**Zero flag (Z):** After any arithmetic or logical operation if result is 0 the zero flag becomes set i.e. 1 otherwise it becomes reset i.e. 0.
**Auxiliar carry flag (AC):** if intermediate carry is generated this flag is set to 1 otherwise it is reset to 0.
**Parity flag (P):** If after any arithmetic or logical operation the result has be even parity, an even number of 1 bits, the parity register becomes set i.e. 1, otherwise it becomes reset.
**Carry Flat (CY):** Carry is generated when performing n bit operation and the result is more than n bits, then this flag becomes set i.e. 1 otherwise it becomes reset i.e. 0. During subtraction (A-B), if A>B it becomes reset and if A<B it becomes set. So carry flag is also called borrow flag.

# Introduction

**Instruction register and decoder:** It is an 8 bit register. When an instruction is fetched from memory then it is stored in the instruction register. Instruction decoder decodes the information present in the instruction register. Here the instruction is decoded and decoded information is given to the timing and control circuit where the instruction is executed.

**Stack pointer (SP):** It is a 16 bit special purpose register. It holds address of the top of the stack. Stack is a set of memory locations operating in LIFO method. SP is decremented on every **PUSH** operation and incremented on every **POP** operation.

**Serial I/O control**: It controls the serial data communication using the instruction SID (Serial Input data) and SOD (Serial output data).

**Address buffer and Address data buffer:** This content stored in stack pointer and program counter is loaded into the address buffer and address-data buffer to communicate with the CPU. The memory and I/O chips are connected to these buses. The memory and I/O chips are connected to these buses. CPU can exchange the desired data with memory and I/O chips.

# Introduction

**Temporary Registers (WZ):** This is a 16 bit register pair. It is used by microprocessor to hold temporary values in some instruction like CALL/JMP/LDA/XCHG, etc. The programmer has no access to this register.

**INR/DCR register:** This is 16 bit shift register. It is used to increment PC after every insruction byte is fetched and increment or decrement SP after a POP or PUSH operation respectively. It is not available for programmer.

# Introduction

**Interrupt Control:** This block is responsible for controlling hardware interrupts of 8085. 8085 supports the following interrupts:

a. **Trap:** This is an edge as well as level triggered vectored interrupt. It cannot be masked by SIM instruction and can neither be displayed by DI instruction . Value for TRAP is 4.5. To calculate memory location value: $4.5X8=(0036)_{10}$ i.e. 0024H. IT has highest priority Its vector address is 0024h i.e. control is transferred to 0024h memory location.

b. **RST 7.5**: RST means restart. This is an edge triggered, vectored interrupt. It can be masked by SIM instruction and can also be disabled by DI instruction. It has second highest priority. Its vector address is 003ch. Here maskable interrupt means it can be disabled or ignored by instruction of CPU.

c. **RST 6.5:** This is a level triggered vectored interrupt. It can be masked by SIM instruction and can also be displayed by DI instruction. It has third highest priority. Its vector address is 0034h

d. **RST 5.5:** This is a level triggered vectored interrupt. It can be masked by SIM instruction can also be displayed by DI instruction. It has fourth highest priority. Its vector address is 002CH.

# Introduction

**Interrupt Control:** This block is responsible for controlling hardware interrupts of 8085. 8085 supports the following interrupts:

e. **INTR:** This is level triggered, non-vectored interrupt. It cannot be masked by SIM instruction but can be disabled by DI instruction. It has lowest priority. It has acknowledgement signal INTA'. The address for ISR is fetched from external hardware. INTA' is an acknowledgement signal for INTR.

**Here vectored interrupts are those interrupts which have fixed vector address (starting address of sub-routine) and after executing these program control is transferred to that address.**

**Maskable interrupts are those which can be disabled or ignored by microprocessor.**

# Introduction

**Software Interrupt :** Software interrupts vector addresses are given by:

| Interrupt | Vector Address |
|---|---|
| RST 0 | 00h |
| RST 1 | 08h |
| RST 2 | 10h |
| RST3 | 18h |
| RST4 | 20h |
| RST5 | 28h |
| RST6 | 30h |
| RST7 | 38h |

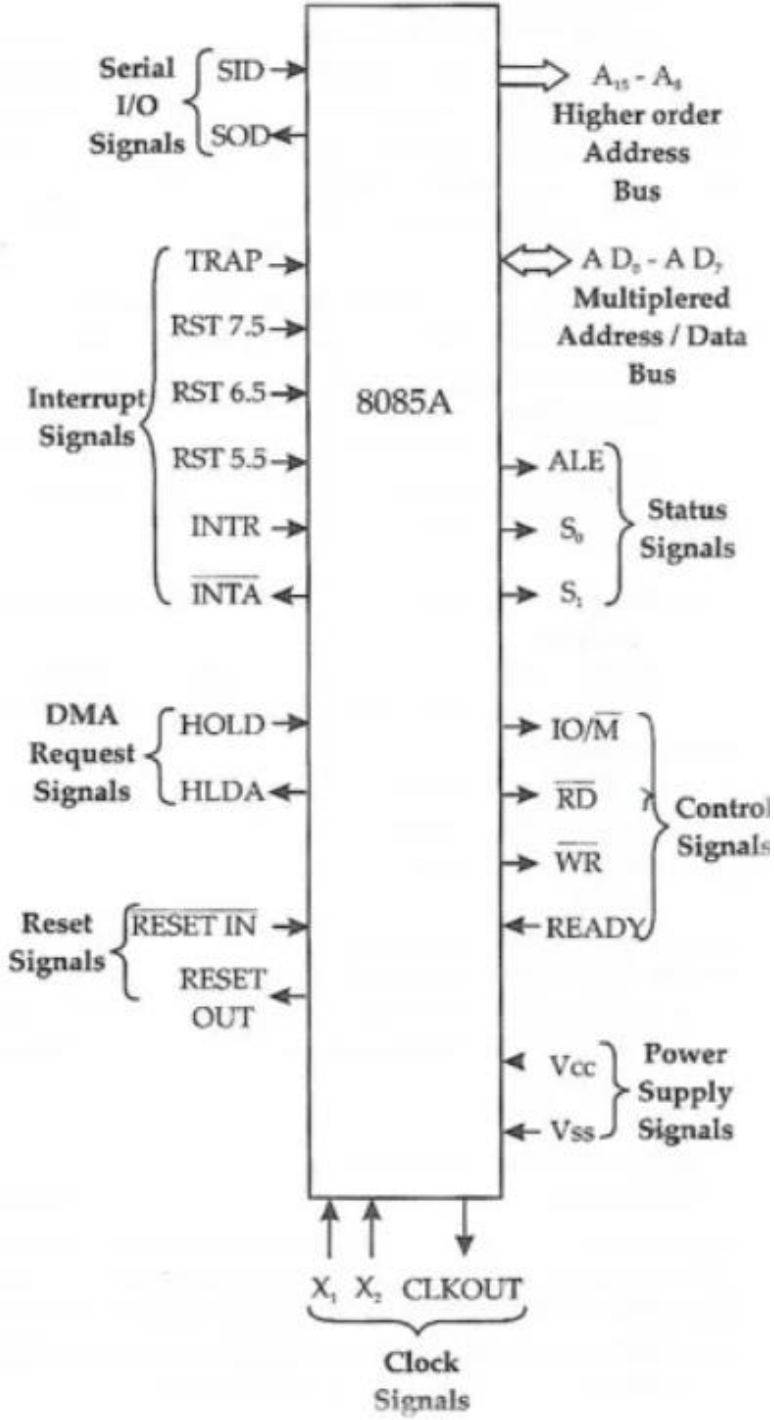
# Introduction

**Timing and control unit:** We use timing and control unit in 8085 for generation of timing signals and signals to control. All operations and functions both interior and exterior of a microprocessor are controlled by this unit.

- **Control signals:** Ready, RD', WR', ALE
- **Status Signals:** S0, S1, IO/M'
- **DMA signals:** HOLD, HLDA
- **Reset signals:** $\overline{\text{RESET IN}}$, RESET OUT

# Pin diagram 8085

8085 microprocessor is of 40 pins and we can divide into nine groups: address and data bus, control signals, status signals, interrupt signals, clock signals, reset signals, DMA request signals, serial I/O signals, power supply signals.

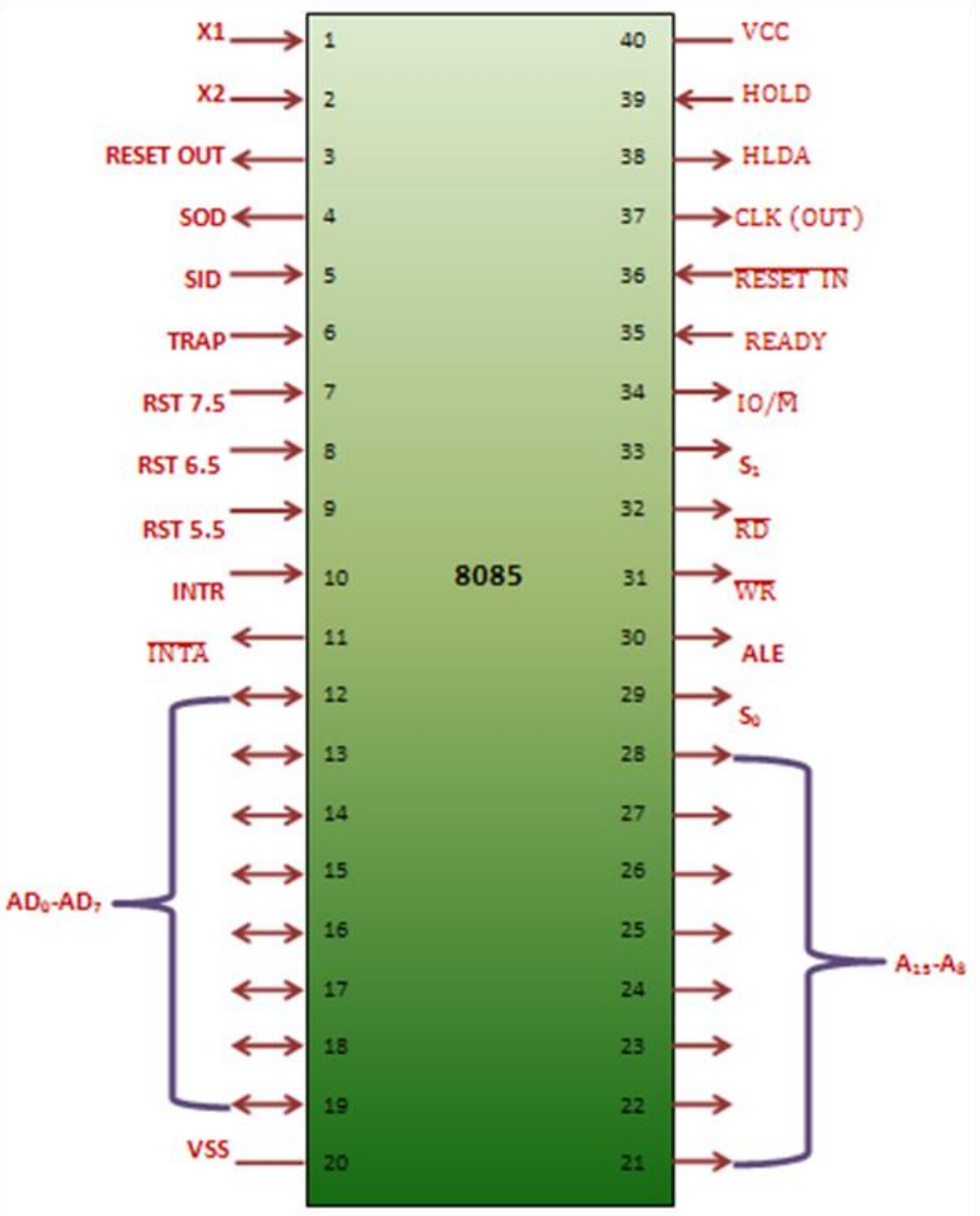


# Pin diagram 8085

**Address and Data Bus**

**Address bus (A8-A15):** High order address bus. The address bus pins are ranges from A8-A15.

Address and Data bus (Pins AD0-AD7): Lower order address/data bus. These pins are multiplexed i.e. it performs two tasks. The address bus is used to connect with IO devices or memory. The 8085 has another pin, which helps in demultiplexing these 8 pins i.e. ALE (Address Latch Enable). For first machine cycle, you can access address lines. These are then latched to an external latch. At second cycle 8 address bits are latched and we source the data lines from AD0-AD7.



# Pin diagram 8085

**Status Signals (IO/$\overline{M}$)**

The status signal IO/$\overline{M}$ (also can be termed as control signals) resolves whether the address is intended for memory or input/output. When the IO/M is high then the address of the address bus is used for the devices of input/output devices. When the IO/M is low then the address of the address bus is used for the memory.

**Status signals (S0-S1)**

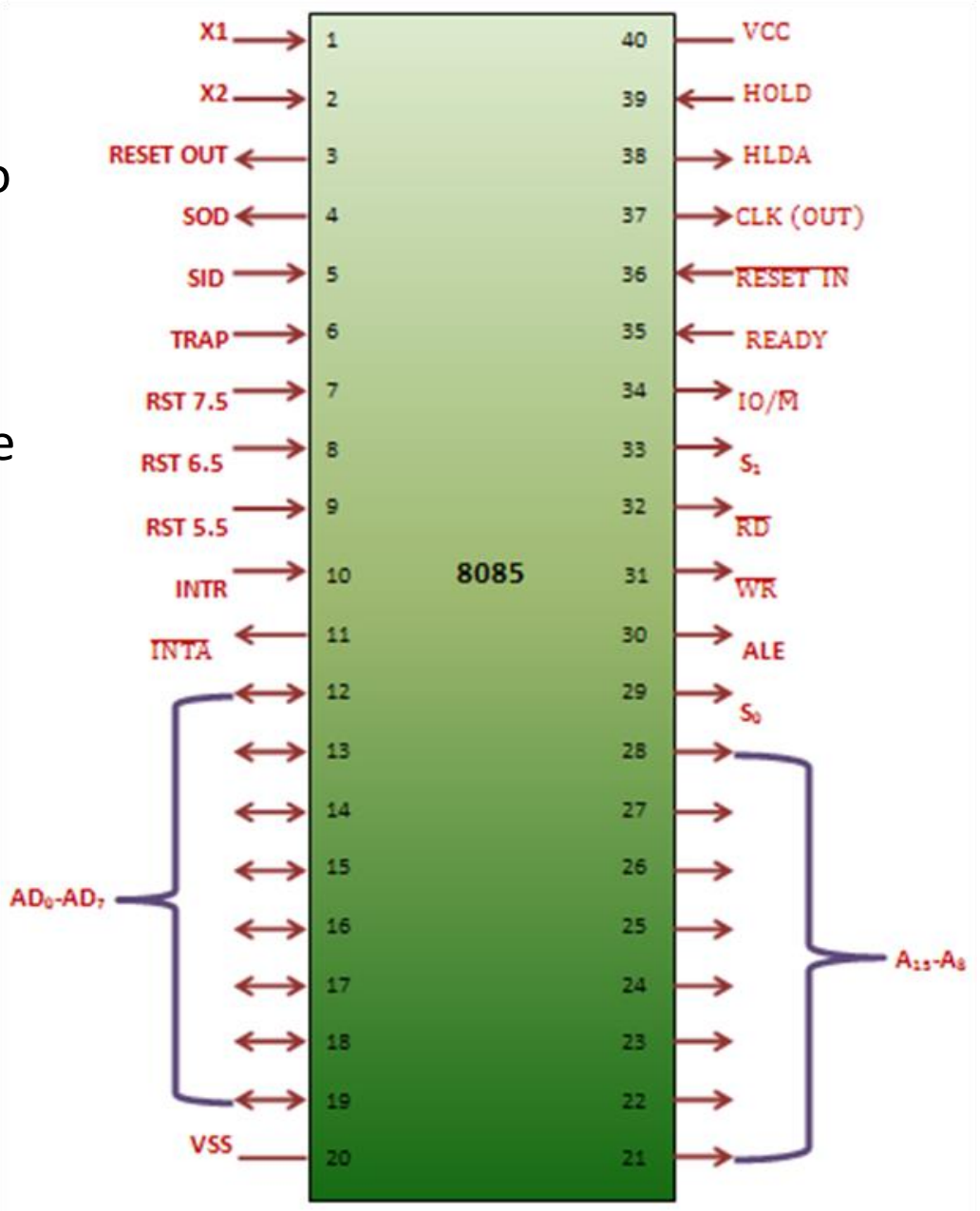
The status signals S0, S1 gives different functions as well as status based on their status.

| IO/M' | S1 | S0 | Data Bus Status |
|---|---|---|---|
| 0 | 1 | 1 | Opcode fetch |
| 0 | 1 | 0 | Memory read |
| 0 | 0 | 1 | Memory write |
| 1 | 1 | 0 | I/O read |
| 1 | 0 | 1 | I/O write |
| 1 | 1 | 1 | Interrupt acknowledge |
| 0 | 0 | 0 | Halt |

# Pin diagram 8085

## Status Signals

**ALE (Address Latch Enable):** ALE is used to separate the AD0-AD7 bus to A0-A7 bus and D0-D7 bus. It goes high during first T state of a machine and enables a lower 8 bits of the address if its value is 1 and otherwise data bus is activated.



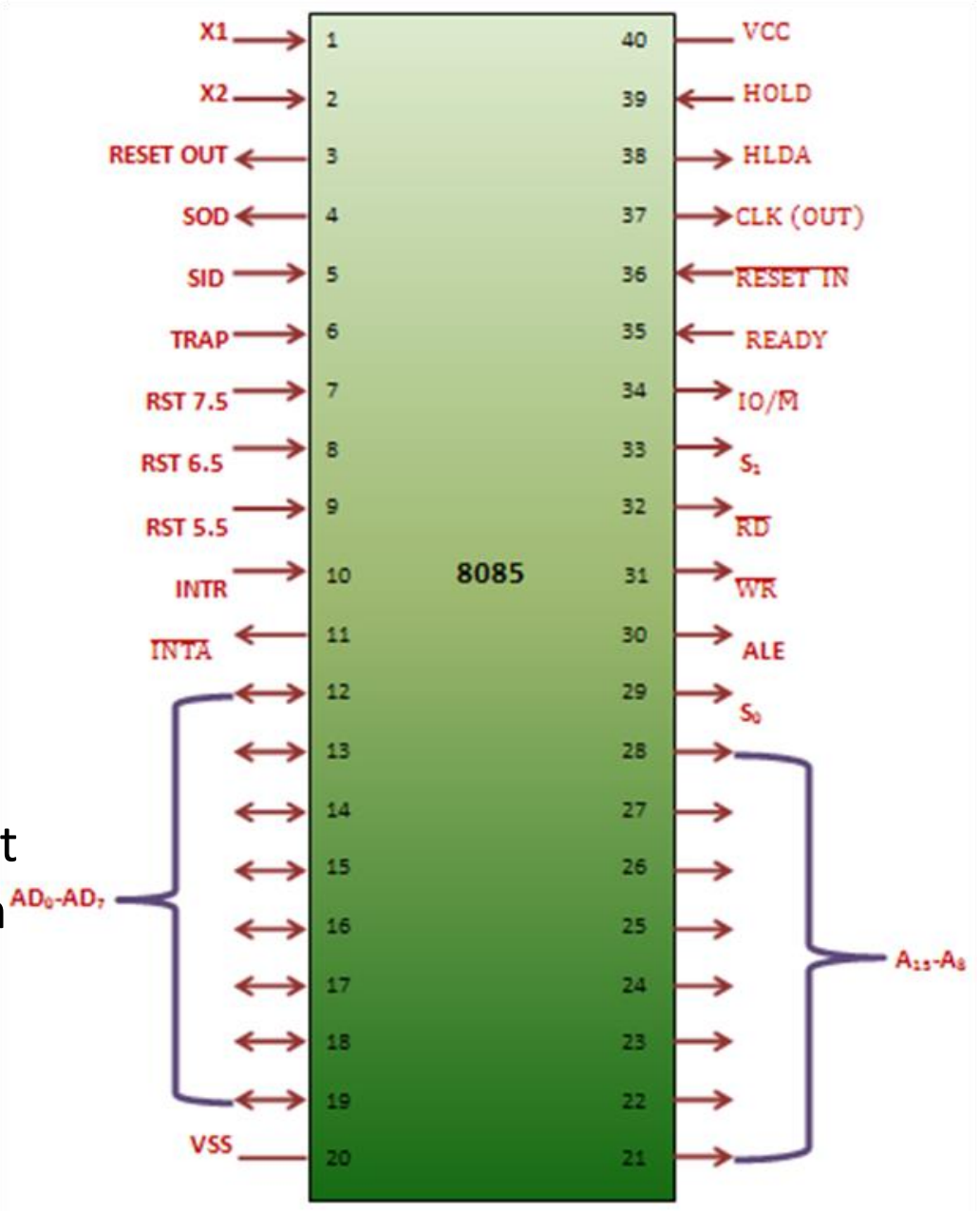
# Pin diagram 8085

**Control Signals (RD', WR', IO/M')**

**RD' (Read)**: This is an active low input. When goes low, the microprocessor reads data from either I/O device or from memory.

**WR' (Write)**: Another active low pin. When it goes low, the data on the data bus is written to a memory location or an I/O device.

**Ready:** This pin used for synchronize slower peripheral devices with fast microprocessor. A low value causes the microprocessor to enter into wait state. The microprocessor remains in wait state until the input at this pin goes high.



# Pin diagram 8085

## DMA signals (Pins HOLD and HLDA)

-**HOLD**: The hold pin can be used to communicate to the microproessor control mechanism that an external device is requesting the use of address bus and data bus just like DMA controller.

-**HLDA**: Hold acknowledge. It indicates that HOLD request has been received. After the removal of HOLD request the HLDA goes low.

## Serial I/O signals (Pins SID and SOD):

SID (Serial Input data) pins accepts serial input data. The data transfers from the pin to the $7^{th}$ bit of the accumulator when a RIM(Read interrupt mask) instruction executes.

SOD(Serial Output data) pin outputs serial data onto the $7^{th}$ bit of the accumulator when a SIM (set interrupt mask) instruction is executed.

# Pin diagram 8085

**Power supply signals (Pins Vcc and Vss)**
**Vcc –** +5v power supply
**Vss –** Ground Reference

**X1, X2**: These X1 and X2 pins are also called crystal input pins. 8085 microprocessor can generate clock signals internally. To generate internal clock signals, 8085 microprocessor requires external inputs from X1 and X2 pins.
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 microprocessor.

Note: Because the operating frequency of 8085 Microprocessor is half (1/2) of Crystal frequency ( frequency offered by Crystal oscillator).

# Pin diagram 8085

## Reset signals

**RESETIN:**

-It is active low signal.

-If this pin in 0, then microprocessor will get reset. When microprocessor is reset, then all registers including PC program counter will get cleared to zero. i.e. PC=0000H

-So microprocessor will fetch it's next instruction from 0000h location of address.

-During reset operation all the buses are in high impedence state or tristated, so power consumption is also very less.

-At first it clears PC and IR. Then at second, disable all interrupts (except trap). At third disable SOD pin, fourthly all buses are tristated and finally gives high output to RESET OUT pin.

**RESETOUT**

-It is active high signal

-When this pin is active high, it will reset interfaced device with it. All the peripherals are connected to Reset out along with clock out.

# Addressing Mode of 8085 microprocessor

-The different ways in which microprocessor access the data is referred to as addressing mode.

-In Assembly language statements, the addressing mode is indicated in the instruction itself.

-There are five types of addressing mode in 8085. They are:

- -Register Addressing mode
- -Immediate addressing mode
- -Direct addressing mode
- -Register indirection addressing mode
- -Implied Addressing mode

# Addressing Mode of 8085 microprocessor

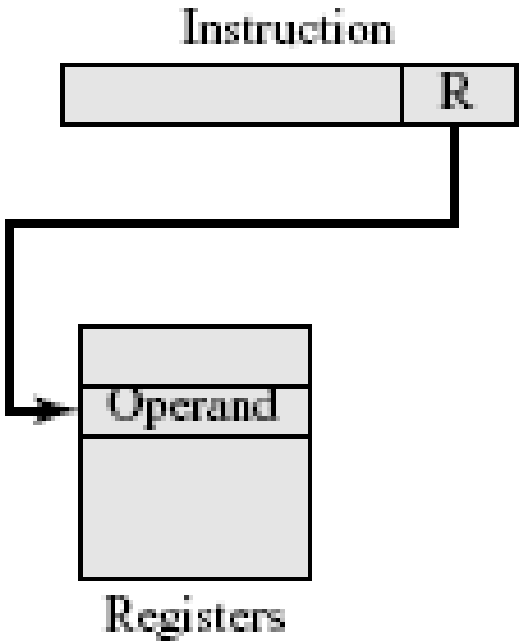
**Register Addressing mode:**

- Operands are in registers that reside within the CPU

-The particular register is selected from a register field in the instruction.

-It is carried out with 8 bit registers A, B, C, D, E, H, L.

-Mov a, b



# Addressing Mode of 8085 microprocessor

**Immediate Addressing mode:**

- Operand is specified in the instruction itself.

-In other words, an immediate-mode instruction has an operand field rather than an address field.

-Immediate mode of instructions is useful for initializing register to constant value.

-Example: mvi a,05h

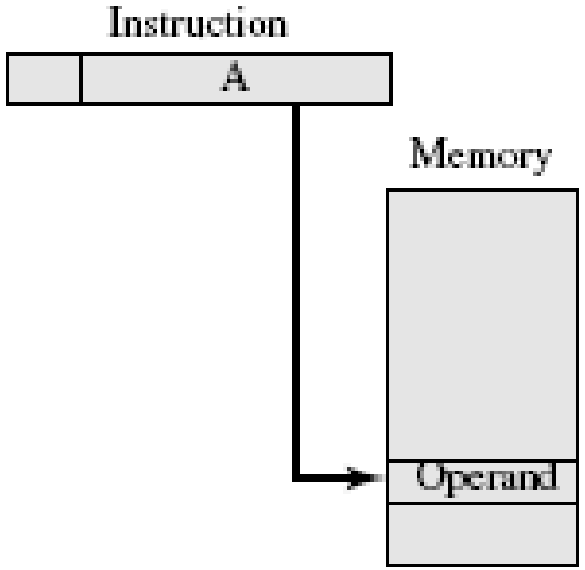
# Addressing Mode of 8085 microprocessor

**Direct Addressing mode:**

- In this mode the effective address is equal to the address part of the instruction.

-the operand resides in memory and its address is given directly by the address field of the instruction.

-Example: LDA 4000h

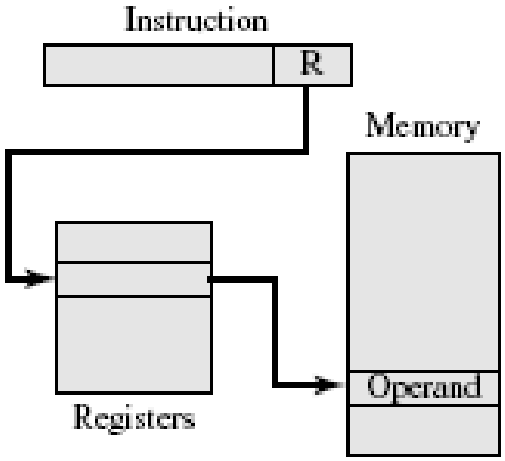


# Addressing Mode of 8085 microprocessor

**Register Indirect Addressing mode:**

- In this mode the instruction specifies a register in the CPU whose contents give the address of the operand in memory.

-Before using a register indirect mode instruction, the programmer must ensure that the memory address of the operand is placed in the processor register with a previous instruction.



# Addressing Mode of 8085 microprocessor

**Implied Addressing mode:**

-Operands are specified implicitly in the definition of the instruction.

-For example, the instruction CMC is an implied mode instruction as the operand in carry flag is implied in the definition of the instruction.

-Example: STC, CMC, DAA

# Instruction set of 8085 microprocessor

-Instruction is a binary pattern designed inside a microprocessor to perform a specific function.
-The entire group of instructions that a microprocessor supports is called instruction set.
-8085 has 246 instructions.
-Each instruction is represented by an 8 bit binary value.
-These 8 bits of binary value is called op-code or instruction byte.
-8085 instruction set can be classified into following categories:

- -Data transfer instruction
- -Arithmetic instruction
- -Logical instruction
- -Branching instruction
- -Control Instruction

# Instruction set of 8085 microprocessor

## Data transfer instructions

-These instructions move data between registers or between memory and registers.

-These instruction copy data from source to destination.

-While copying, the contents of source are not modified.

| Opcode | Operand | Description |
|---|---|---|
| MOV | Rd, Rs<br>M, Rs<br>Rd, M | Copy from source to destination. |

-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 HL registers.

-Example: MOV B, C

MOV B,M

# Instruction set of 8085 microprocessor

## Data transfer instructions

| Opcode | Operand | Description |
|---|---|---|
| MVI | Rd, Data<br>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 content of H-L registers.
-Example: MVI B, 57H,
MVI M, 57H

| Opcode | Operand | Description |
|---|---|---|
| LDA | 16-bit address | Load Accumulator |

-The content of a memory location, specified by a 16 bit address in the operand are copied to the accumulator.
-The content of source are not altered.
-Example: LDA 2034H

# Instruction set of 8085 microprocessor

## Data transfer instructions

| Opcode | Operand | Description |
|---|---|---|
| LDAX | B/D Register Pair | Load accumulator indirect |

-The contents of the designated register pair point to a memory location.

-This instruction copies the content of that memory location into the accumulator.

-The content of either the register pair or the memory location are not altered.

-Example: LDAX B

| Opcode | Operand | Description |
|---|---|---|
| LXI | Reg. pair, 16-bit data | Load register pair immediate |

-This instruction loads 16 bit data in the register pair.

-Example: LXI H, 2034h

# Instruction set of 8085 microprocessor

## Data transfer instructions

| Opcode | Operand | Description |
|---|---|---|
| LHLD | 16-bit address | Load H-L registers direct |

-This instruction copies the content of memory location pointed out by 16 bit address into register L.
-It copies the content of next memory location into register H.
-Example: LHLD 2040H

| Opcode | Operand | Description |
|---|---|---|
| STA | 16-bit address | Store accumulator direct |

-The contents of accumulator are copied into the memory location specified by the operand.
-Example: STA 2500H

# Instruction set of 8085 microprocessor

## Data transfer instructions

| Opcode | Operand | Description |
|---|---|---|
| STAX | Reg. pair | Store accumulator indirect |

-The contents of accumulator are copied into the memory location specified by the contents of the register pair.
-Example: STAX B

| Opcode | Operand | Description |
|---|---|---|
| SHLD | 16-bit address | Store H-L registers direct |

-The content of register L are stored into memory location specified by the 16 bit address.
-The content of register H are stored into the next memory location.
-Example: SHLD 2550H

# Instruction set of 8085 microprocessor

## Data transfer instructions

| Opcode | Operand | Description |
|---|---|---|
| XCHG | None | Exchange H-L with D-E |

-The contents of register H are exchanged with the contents of register D. The content of register L are exchanged with the content of register E.

-During this, WZ register is used as a temporary register to exchange the value between HL and DE register.

-Example: XCHG

| Opcode | Operand | Description |
|---|---|---|
| SPHL | None | Copy H-L pair to the Stack Pointer (SP) |

-This instruction loads the contents of H-L pair into SP.

# Instruction set of 8085 microprocessor

## Data transfer instructions

| Opcode | Operand | Description |
|---|---|---|
| XTHL | None | Exchange H–L with top of stack |

-The contents of L register are exchanged with the location pointed out by the contents of SP.
-The content of H register are exchanged with the next location (SP+1)
-Example: XTHL

| Opcode | Operand | Description |
|---|---|---|
| PCHL | None | Load program counter with H-L contents |

-The contents of registers H and L are copied into the program counter (PC).
-The contents of H are placed as the high order byte and the contents of L as the low order byte.
-Example: PCHL

# Instruction set of 8085 microprocessor

## Data transfer instructions

| Opcode | Operand | Description |
|---|---|---|
| PUSH | Reg. pair | Push register pair onto stack |

-The content of register pair are copied into stack.
-SP is decremented and the contents of high order registers (B, D, H, A are copied into stack
-SP is again decremented and contents of low order registers (C, E, L, flags are copied into stack.
-Example: PUSH B

# Instruction set of 8085 microprocessor

## Data transfer instructions

| Opcode | Operand | Description |
|---|---|---|
| POP | Reg. pair | Pop stack to register pair |

-The content of top of stack are copied into register pair.
-The contents of location pointed out by SP are copied to low order register (C, E, L, Flags).
-SP is incremented and the contents of location are copied to high order register (B, D, H, A).
-Example: POP H

# Instruction set of 8085 microprocessor

## Data transfer instructions

| Opcode | Operand | Description |
|---|---|---|
| OUT | 8-bit port address | Copy data from accumulator to a port with 8-bit address |

The contents of accumulator are copied into the I/O port.

Example: OUT 60H

| Opcode | Operand | Description |
|---|---|---|
| IN | 8-bit port address | Copy data to accumulator from a port with 8-bit address |

The contents of I/O port are copied into accumulator.

Example: IN 8C H

# Instruction set of 8085 microprocessor

## Arithmetic instructions

-These instructions perform the operationslike:

- -Addition
- -Subtraction
- -Increment
- -Decrement

# Instruction set of 8085 microprocessor

## Arithmetic instructions: Addition

-Any 8-bit number or the contents of register or the content of memory location can be added to the contents of accumulator.

-The result is stored in accumulator

-No two other 8 bit registers can be added directly. i.e. contents of B register cannot be added directly to the content of register C.

| Opcode | Operand | Description |
|---|---|---|
| ADD | R<br>M | Add register or memory to accumulator |

-The contents of registers or memory are added to the contents of accumulator.

-The result is stored in accumulator.

-If the operand is memory location, its address is specified by H-L pair.

-All flags are modified to reflect the result of addition.

-Example: ADD B

ADD M

# Instruction set of 8085 microprocessor

## Arithmetic instructions: Addition

| Opcode | Operand | Description |
|---|---|---|
| ADC | R<br>M | Add register or memory to accumulator with carry |

-The content of register or memory and Carry flag are added to the content of accumulator.
-The result is stored in accumulator.
-If the operand is memory location, its address is specified by H-L pair.
-All flags are modified to reflect teh result of addition.
-Example: ADC B or ADC M

| Opcode | Operand | Description |
|---|---|---|
| ADI | 8-bit data | Add immediate to accumulator |

-The 8 bit data is added to the contents of accumulator.
-The result is stored in accumulator.
-All flags are modified to reflect the result of the addition.
-Example: ADI 45

# Instruction set of 8085 microprocessor

## Arithmetic instructions: Addition

| Opcode | Operand | Description |
|---|---|---|
| ACI | 8-bit data | Add immediate to accumulator with carry |

-The 8 bit data and the carry flag are added to the contents of accumulator.
-The result is stored in accumulator.
-All flags are modified to reflect the result of addtion.
-Example: ACI 45H

| Opcode | Operand | Description |
|---|---|---|
| DAD | Reg. pair | Add register pair to H-L pair |

-The 16 bit contents of register pair are added to the contents of H-L pair.
-The result is stored in H-L pa ir. If the result is larger than 16 bits, then CY is set. No other flags are changed.
-Example: DAD B

# Instruction set of 8085 microprocessor

## Arithmetic instructions: subtraction

| Opcode | Operand | Description |
|---|---|---|
| SUB | R<br>M | Subtract register or memory from accumulator |

-The content of register or memory location are subtracted from the content of the accumulator.

-The result is stored in accumulator

-If the operand is memory location, its address is specified by HL pair.

-All flags are modified to reflect the result of subtraction.

-Example: SUB B

SUB M

# Instruction set of 8085 microprocessor

## Arithmetic instructions: subtraction

| Opcode | Operand | Description |
|---|---|---|
| SBB | R<br>M | Subtract register or memory from accumulator with borrow |

-The content of register or memory location and borrow flag i.e.CF are subtracted from the contents of the accumulator.
-The result is stored in accumulator
-If the operand is memory location, its address is specified by HL pair.
-All flags are modified to reflect the result of subtraction.
-Example: SBB B or SBB M

| Opcode | Operand | Description |
|---|---|---|
| SUI | 8-bit data | Subtract immediate from accumulator |

-The 8 bit data is subtracted from the content of the accumulator.
-The result is stored in accumulator
-All flags are modified to reflect the result of subtraction.
-Example: SUI 45h

# Instruction set of 8085 microprocessor

## Arithmetic instructions: subtraction

| Opcode | Operand | Description |
|---|---|---|
| SBI | 8-bit data | Subtract immediate from accumulator with borrow |

-The 8 bit data and borrow flag (CF) is subtracted from the contents of accumulator.
-The result is stored in accumulator.
-All flags are modified to reflect the result of subtraction.
-Example: SBI 45H

| Opcode | Operand | Description |
|---|---|---|
| INR | R<br>M | Increment register or memory by 1 |

-The contents of register or memory location are incremented by 1f.
-The result is stored in same place.
-If operand is a memory location, its address is specified by the contents of HL pair.
-Example: INR B or INR M

# Instruction set of 8085 microprocessor

## Arithmetic instructions:

| Opcode | Operand | Description |
|---|---|---|
| INX | R | Increment register pair by 1 |

-The contents of register are incremented by 1.
-The result is stored in same place.
-INX H

| Opcode | Operand | Description |
|---|---|---|
| DCR | R<br>M | Decrement register or memory by 1 |

-The contents of register or memory location are decremented by 1.
-The result is stored in the same place.
-If the operand is a memory location, its address is specified by contents of HL pair.
-Example: DCR B or DCR M.

# Instruction set of 8085 microprocessor

## Arithmetic instructions:

| Opcode | Operand | Description |
|---|---|---|
| DCX | R | Decrement register pair by 1 |

-The contents of register pair are decremented by 1.
-The result is stored in the same place.
-Example: DCX H

# Instruction set of 8085 microprocessor

## Logical instruction

-Used to perform logical operation on data stored in registers, memory and status flag.

-Logical operations are:

- -AND (ANA, ANI)
- -OR(ORA, ORI)
- -XOR (XRA, XRI
- -Rotate
- -Compare
- -Complement

# Instruction set of 8085 microprocessor

## Logical instruction : AND, OR, XOR

-Perform logical operations on any 8 bit data, or contents of register or memory location with the contents of accumulator.

-The result is stored in accumulator.

| Opcode | Operand | Description |
|---|---|---|
| ANA | R<br>M | Logical AND register or memory with accumulator |

-The contents of accumulator are logically ANDed with the contents of register or memory.

-The result is placed in the accumulator.

-If the operand is a memory location, its address is specified by teh contents of HL pair.

-S, Z, P are modified to reflect the result of operation.

-CY is reset and AC is set.

-Example: ANA B or ANA M

# Instruction set of 8085 microprocessor

## Logical instruction : AND, OR, XOR

| Opcode | Operand | Description |
|---|---|---|
| ANI | 8-bit data | Logical AND immediate with accumulator |

-The contents of accumulator are logically ANDed with 8 bit data.
-The result is placed in the accumulator.
-S, Z, P are modified to reflect the result.
-CY is reset, AC is set.
-Example: ANI 86H.

# Instruction set of 8085 microprocessor

## Logical instruction : AND, OR, XOR

| Opcode | Operand | Description |
|---|---|---|
| ORA | R<br>M | Logical OR register or memory with accumulator |

-The contents of the accumulator are logically ORed with the contents of register or memory.
-The result is placed in the accumulator.
-If the operand is a memory location, its address is specified by the contents of HL par.
-S,Z, P are modified to reflect the result.
-CY and AC are reset.
-Example: ORA B or ORA M

# Instruction set of 8085 microprocessor

## Logical instruction : AND, OR, XOR

| Opcode | Operand | Description |
|---|---|---|
| ORI | 8-bit data | Logical OR immediate with accumulator |

-The contents of accumulator are logically ORed with 8 bit data.
-The result is placed in the accumulator.
-S, Z, P are modified to reflect the result.
-CY and AC are reset.
-Example: ORI 86H

# Instruction set of 8085 microprocessor

## Logical instruction : AND, OR, XOR

| Opcode | Operand | Description |
|---|---|---|
| XRA | R<br>M | Logical XOR register or memory with accumulator |

-The contents of accumulator are XORed with the contents of register or memory.
-The result is placed in the accumulator.
-If the operand is a memory location, its address is specified by the contents of HL pair
-S, Z, P are modified to reflect the result of the operation.
-CY and AC are reset.
-Example: XRA B or XRA M

# Instruction set of 8085 microprocessor

## Logical instruction : AND, OR, XOR

| Opcode | Operand | Description |
|---|---|---|
| XRI | 8-bit data | XOR immediate with accumulator |

-The contents of the accumulator are XORed with the 8 bit data.
-The reuslt is placed in the accumulator.
-S, Z, P are modified to reflect the result.
-CY and AC are reset
-Example: XRI 86h

# Instruction set of 8085 microprocessor

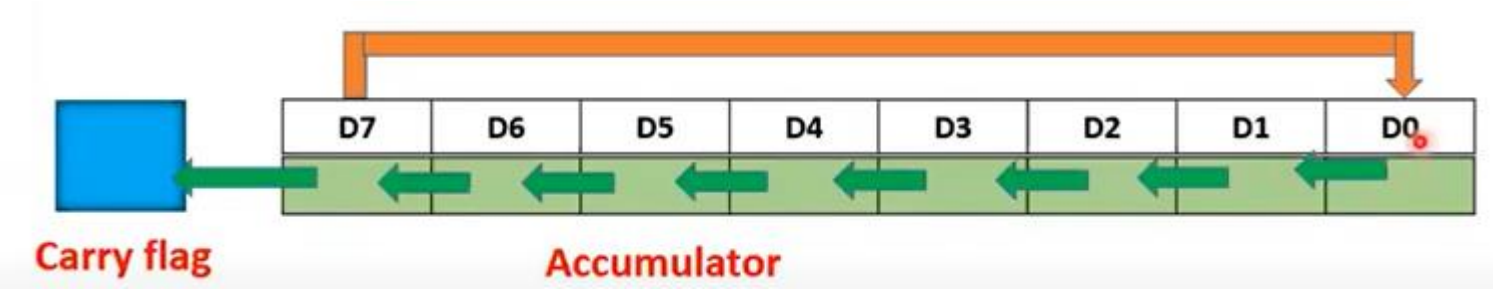
**Rotate**

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

-There are 4 categories of Rotate instruction. They are:

**RLC (Rotate Accumulator left)**

-Each bit is shifted to the adjacent left position. Bit D7 becomes D0. Carry flag is modified according to bit D7.



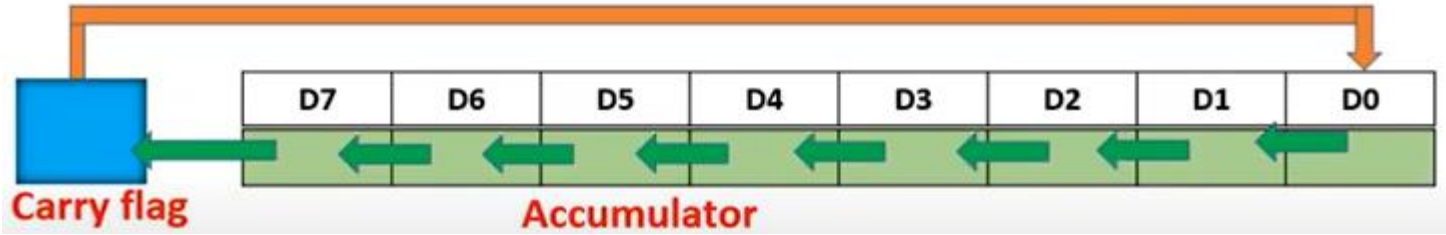
# Instruction set of 8085 microprocessor

**Logical instruction :**

**Rotate**

**RAL (Rotate Accumulator left through carry)**

-Each bit is shifted to the adjacent left position. Bit D7 becomes carry bit and carry flag bit is shifted to DO . Carry flag is modified according to bit D7.



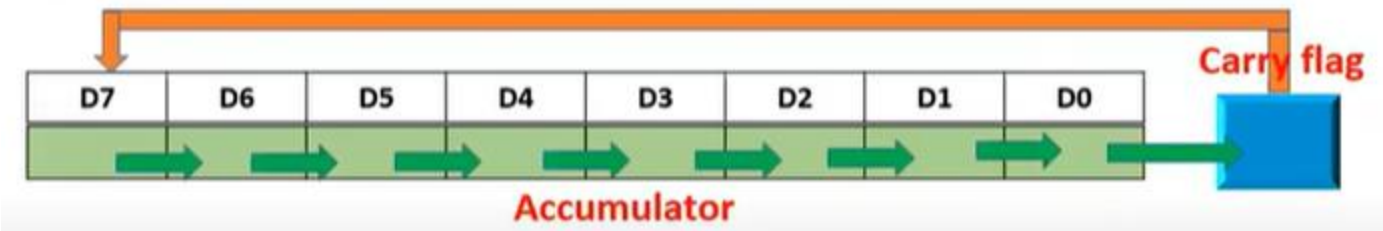
# Instruction set of 8085 microprocessor

**Logical instruction :**

**Rotate**

**RAR (Rotate Accumulator Right through carry)**

-In this instruction, each bit is shifted to the adjacent right position. Bit D0 becomes the carry bit and carry bit shifted to D7, Carry flag is modified according to bit D0.



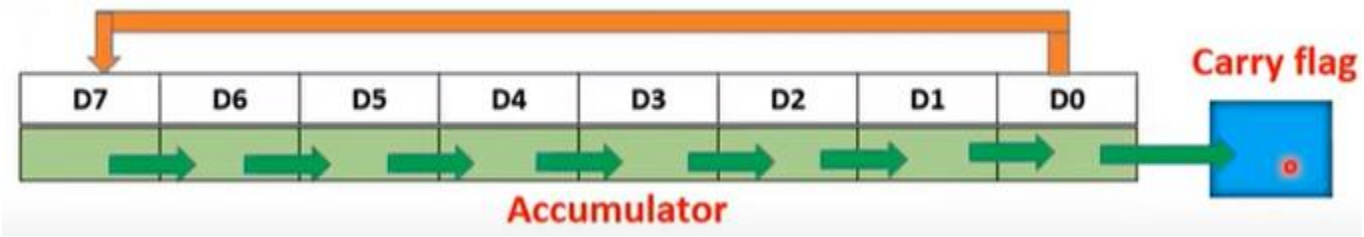
# Instruction set of 8085 microprocessor

**Logical instruction :**

**Rotate**

**RRC (Rotate Accumulator left through carry)**

-Each bit is shifted to the adjacent right position. Bit D0 becomes D7. Carry flag CY is modified according to the bit D0




**What is the application of Rotate instructions?**

-The rotate instructions are primarily used in arithmetic multiply and divide operations and for serial data transfer.

# Instruction set of 8085 microprocessor

## Logical instruction

| Opcode | Operand | Description |
|---|---|---|
| INX | R | Increment register pair by 1 |

-The contents of register are incremented by 1.
-The result is stored in same place.
-INX H

| Opcode | Operand | Description |
|---|---|---|
| DCR | R<br>M | Decrement register or memory by 1 |

-The contents of register or memory location are decremented by 1.
-The result is stored in the same place.
-If the operand is a memory location, its address is specified by contents of HL pair.
-Example: DCR B or DCR M.

# Instruction set of 8085 microprocessor

## Complement:

| Opcode | Operand | Description |
|---|---|---|
| CMA | None | Complement accumulator |

-The content of accumulator can be complemented. In this operation, each 0 is replaced by 1 and 1 by 0.
-No flags are affected.
-Instruction used for this is **CMA**.

| Opcode | Operand | Description |
|---|---|---|
| CMC | None | Complement carry |

-The carry flag is complemented.
-No other flags are affected.
-Example: CMC

# Instruction set of 8085 microprocessor

## STC

| Opcode | Operand | Description |
|---|---|---|
| STC | None | Set carry |

-The carry flag is set to 1.

-No other flags are affected.

-Example: STC

# Instruction set of 8085 microprocessor

## Compare Instruction

-Any 8 bit data or then contents of register or memory location can be compares for Equality, Greater than, Less than with the contents of accumulator.

-The result is reflected in status flag.

# Instruction set of 8085 microprocessor

## Logical Instruction

| Opcode | Operand | Description |
|---|---|---|
| CMP | R<br>M | Compare register or memory with accumulator |

-The content of operand (register or memory) are compared with contents of accumulator.

-However both contents are preserved.

-The result of comparison is shown by setting the flag of PSW as:

-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.

# Instruction set of 8085 microprocessor

## Logical Instruction

| Opcode | Operand | Description |
|---|---|---|
| CMP | R<br>M | Compare register or memory with accumulator |

-The content of operand (register or memory) are compared with contents of accumulator.

-However both contents are preserved.

-The result of comparison is shown by setting the flag of PSW as:

-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.

# Instruction set of 8085 microprocessor

## Logical Instruction

| Opcode | Operand | Description |
|---|---|---|
| CPI | 8-bit data | Compare immediate with accumulator |

-The 8 bit data is compared with the contents of accumulator.

-The values being compared remain unchanged.

-The result of comparison is shown by setting the flags of PSW as:

-If (A)< data ; carry flag is set

-If (A)=data ; zero flag is set

-If (A)>data ; carry and zero flag are reset.

# Instruction set of 8085 microprocessor

## Branching Instruction

-The branching instruction alter the normal sequential flow.

-These instructions alter either unconditionally or conditionally.

| Opcode | Operand | Description |
|---|---|---|
| JMP | 16-bit address | Jump unconditionally |

-The program sequence is transferred to the memory location specified by 16 bit address given in the operand.

-JMP 2034H

# Instruction set of 8085 microprocessor

## Branching Instruction

| Opcode | Operand | Description |
|---|---|---|
| Jx | 16-bit address | Jump conditionally |

-The program sequence is transferred to the memory location specified by 16 bit address given in the operand based on the specified flag of PSW.Example: JZ 2034H

| Opcode | Description | Status Flags |
|---|---|---|
| JC | Jump if Carry | CY = 1 |
| JNC | Jump if No Carry | CY = 0 |
| JP | Jump if Positive | S = 0 |
| JM | Jump if Minus | S = 1 |
| JZ | Jump if Zero | Z = 1 |
| JNZ | Jump if No Zero | Z = 0 |
| JPE | Jump if Parity Even | P = 1 |
| JPO | Jump if Parity Odd | P = 0 |

# Instruction set of 8085 microprocessor

## Branching Instruction

| Opcode | Operand | Description |
|---|---|---|
| CALL | 16-bit address | Call unconditionally |

-The program sequence is transferred to the memory location specified by 16 bit address given in the operand.
-Before the transfer, the address of the next instruction after CALL (the contents of program counter) is pushed into the stack.
-Example: CALL 2034H

# Instruction set of 8085 microprocessor

## Branching Instruction

| Opcode | Operand | Description |
|---|---|---|
| Cx | 16-bit address | Call conditionally |

-The program sequence is transferred to the memory location specified by the 16 bit address given in the operand based on specified flag of PSW.

-Before the transfer, the address of next instruction after the call (the contents of program counter) is pushed onto the stack.

-Example: CZ 2034H

# Instruction set of 8085 microprocessor

## Branching Instruction

| Opcode | Operand | Description |
|---|---|---|
| Cx | 16-bit address | Call conditionally |

| Opcode | Description | Status Flags |
|---|---|---|
| CC | Call if Carry | CY = 1 |
| CNC | Call if No Carry | CY = 0 |
| CP | Call if Positive | S = 0 |
| CM | Call if Minus | S = 1 |
| CZ | Call if Zero | Z = 1 |
| CNZ | Call if No Zero | Z = 0 |
| CPE | Call if Parity Even | P = 1 |
| CPO | Call if Parity Odd | P = 0 |

# Instruction set of 8085 microprocessor

## Branching Instruction

| Opcode | Operand | Description |
|---|---|---|
| RET | None | Return unconditionally |

-The program sequence is transferred from the subroutine to the calling program.
-The two bytes from the top of the stack are copied into the program counter and program execution begins at the new address.
-Example: RET

# Instruction set of 8085 microprocessor

## Branching Instruction

| Opcode | Operand | Description |
|---|---|---|
| Rx | None | Call conditionally |

-The program sequence is transferred from the subroutine to the calling program based on the specified flag of PSW.
-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

# Instruction set of 8085 microprocessor

## Branching Instruction

| Opcode | Operand | Description |
|---|---|---|
| Rx | None | Call conditionally |

| Opcode | Description | Status Flags |
|---|---|---|
| RC | Return if Carry | CY = 1 |
| RNC | Return if No Carry | CY = 0 |
| RP | Return if Positive | S = 0 |
| RM | Return if Minus | S = 1 |
| RZ | Return if Zero | Z = 1 |
| RNZ | Return if No Zero | Z = 0 |
| RPE | Return if Parity Even | P = 1 |
| RPO | Return if Parity Odd | P = 0 |

# Instruction set of 8085 microprocessor

## Branching Instruction

| Opcode | Operand | Description |
|---|---|---|
| RST | 0 – 7 | Restart (Software Interrupts) |

-The RST instruction jumps the control to one of eight memory locations depending upon the number.
-These are used as software instructions in a program to transfer program execution to one of the eight locations.
-Example: RST 3

# Instruction set of 8085 microprocessor

## Branching Instruction

| Opcode | Operand | Description |
|---|---|---|
| RST | 0 – 7 | Restart (Software Interrupts) |

| Instructions | Restart Address |
|---|---|
| RST 0 | 0000 H |
| RST 1 | 0008 H |
| RST 2 | 0010 H |
| RST 3 | 0018 H |
| RST 4 | 0020 H |
| RST 5 | 0028 H |
| RST 6 | 0030 H |
| RST 7 | 0038 H |

# Instruction set of 8085 microprocessor

## Control Instruction

-The control instructions control the operation of microprocessor.

| Opcode | Operand | Description |
|---|---|---|
| NOP | None | No operation |

-NOP stands for No Operation. When this instruction encounter, no operation is performed.

-The instruction is fetched and decoded but not operation is executed.

-Example: NOP

# Instruction set of 8085 microprocessor

## Control Instruction

| Opcode | Operand | Description |
|---|---|---|
| HLT | None | Halt |

-The CPU finishes executing current instruction and halts any further execution.
-An interrupt or reset is necessary to exit from the halt state.
-Example: HLT

| Opcode | Operand | Description |
|---|---|---|
| DI | None | Disable interrupt |

-The interrupt enable flip-flop is reset and interrupts except TRAP are disabled.
-No flags are affected.
-Example: DI

# Instruction set of 8085 microprocessor

## Control Instruction

| Opcode | Operand | Description |
|---|---|---|
| EI | None | Enable interrupt |

-The interrupt enable flip-flop is set and all interrupts are enabled.
-No flags are affected.
-This instruction is necessary to reenable the interrupts (except TRAP).
-Example: EI

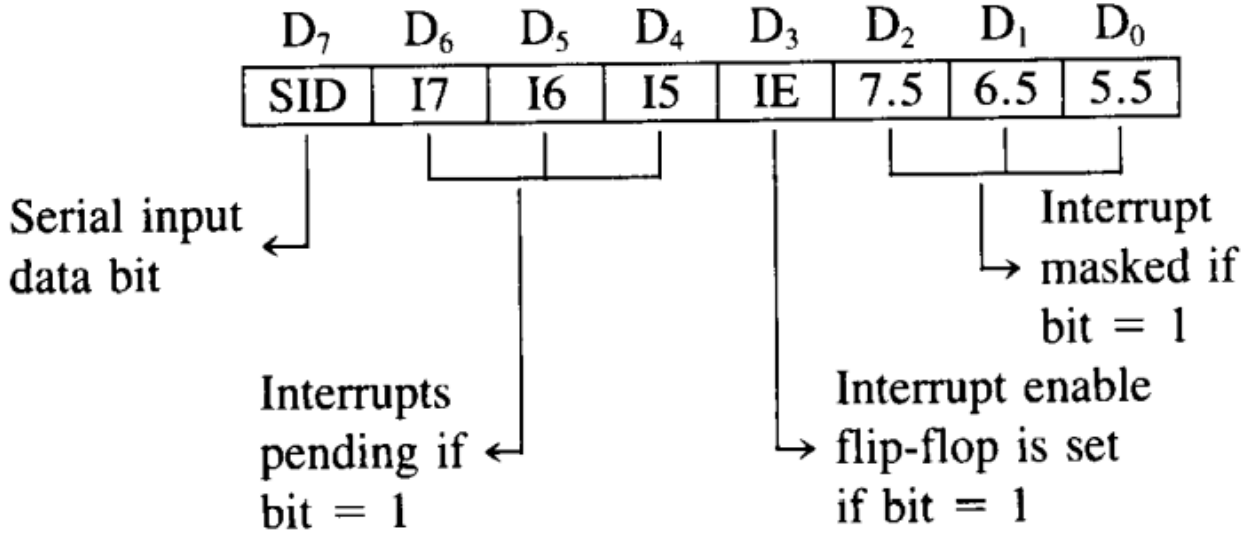
# Instruction set of 8085 microprocessor

## Control Instruction

| Opcode | Operand | Description |
|---|---|---|
| RIM | None | Read Interrupt Mask |

-This is a multipurpose instruction used to read the status of interrupt 7.5, 6.5, 5.5 and read serial data input bit.

-The instruction loads eight bits in accumulator with following interpretations:




-Exampe: RIM

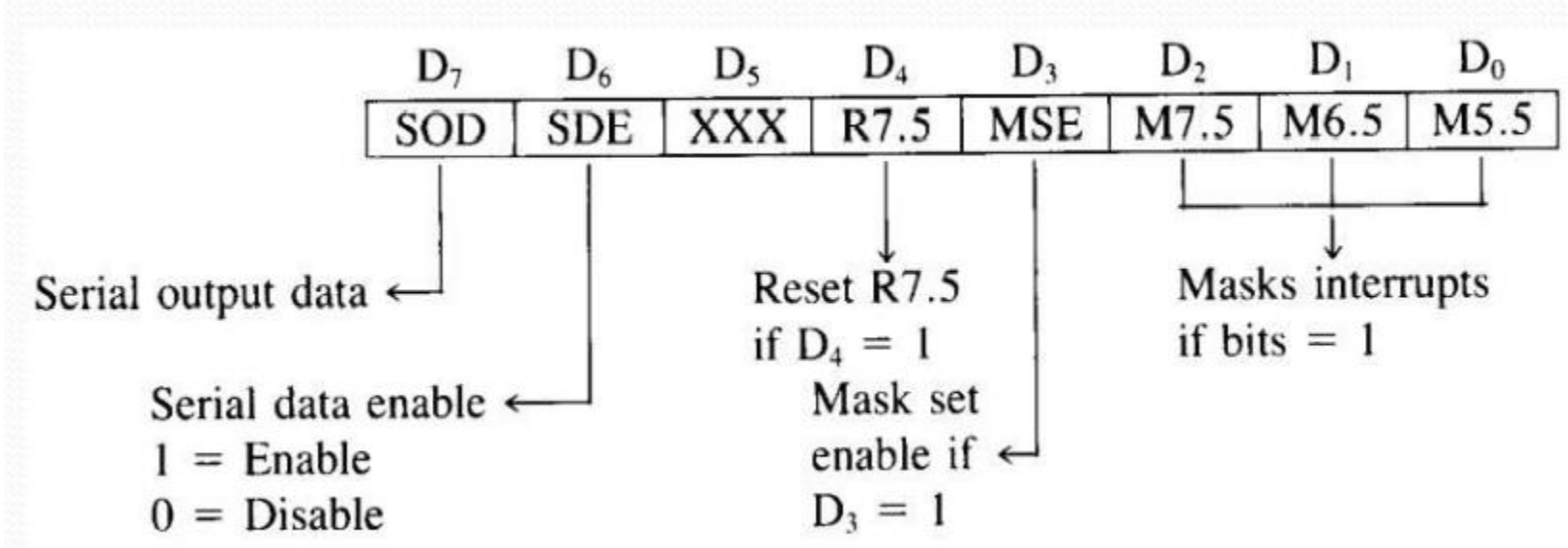
# Instruction set of 8085 microprocessor

## Control Instruction

| Opcode | Operand | Description |
|---|---|---|
| SIM | None | Set Interrupt Mask |

-This is multipurpose instruction used to implement 8085 interrupts 7.5, 6.5, 5.5 and serial data output.

-The instruction interprets the accumulator content as follows;




-Exampe: SIM

# Program List: Assignment

1. Write ALP to add two 8 bit number from memory location 2050H and 3050H and display the result in output port 03H.
2. Write ALP to add two number from the location 2050H and 2051H and store the result in memory location 2055H.
3. Write ALP to add two 8 bit number input from input port 01H and 02H and display the result in output port 05H.
4. Write ALP to subtract two 8 bit number from the location 20H and 25H and display the result in output port 02H
5. Write ALP to add two number from the location 2050H and 2051H and store the result in memory location 2055H.
6. Write ALP to find 1's complement of a number stored in memory location 2020H and store the result in memory location 3030H.
7. Write ALP to find 2's complement of number stored in memory location 40H and store the result in memory location 5000H
8. Write program to perform the right shift on 8 bit number for three times and store the result in memory location 2050H.
9. Write ALP to add two 16 bit numbers and store the result in the memory location 2055h, 2056h.

# Program List: Assignment

9. Write ALP to subtract two 16 bit numbers where first 16 bit number is in the location 2050H and second number from the memory location 2052H and store the result in memory location 2055H.
10. Write ALP to multiply two 8 bit numbers. Store the result in memory lcoation 2050H
11. Write ALP to divide 8 bit numbers and store the quotient in memory location 2055H and remainder in memory location 2056H.
12. Write ALP to find whether a number is odd or even.
13. Write ALP to count number of 1's in given 8 bit number.
14. Write ALP to display number from 1 to 10.
15. Write ALP to find the sum of numbers from 1 to 10.
16. Write ALP to display all odd numbers from 1 to 10.
17. Write ALP to display all even numbers from 1 to 20.
18. Write ALP to display all even number from 20 to 50.
19. Write ALP to find sum of 10 numbers in array.
20. Write ALP to find the greatest number among 10 different numbers.

# Program List: Assignment

21. Write ALP to print fibonacci series upto $8^{th}$ term.
22. Write ALP to sort 10 different numbers in ascending order in array.
23. Write ALP to multiply tow 8 bit numbers 43H and 07H and stored the result in memory address 3050.