Introduction to microprocessor and microcomputer

Reference Books:

1. Ramesh S. Gaonkar, "Microprocessor Architecture, Programming and Application with the 8085".
2. Anokh Singh , A.K. Chhabra ,Fundamentals Of Microprocessors And Its Applications.

INTRODUCTION TO MICROPROCESSOR:

Microprocessor is an electronic circuit that functions as the central processing unit (CPU) of a computer, providing computational control. Microprocessors are also used in other advanced electronic systems, such as computer printers, automobiles.

Typical microprocessors combine arithmetic and logic functional units, instruction processing circuitry, and a portion of the memory hierarchy and the input/output (I/O) and memory subsystems. While many microprocessors and single-chip designs, some high performance designs depend on a few chips to provide multiple functional units and relatively large caches. When combined with other integrated circuits that provide storage for data and programs, often on a single semiconductor base to form a chip, the microprocessor becomes the heart of a small computer, or microcomputer. Microprocessors may be classified by the semiconductor technology (TTL, CMOS, … etc.), or by the width of the data format (4-bit, 8-bit, 16-bit, 32-bit, or 64-bit) they process; and by their instruction set (CISC, complex-instruction-set computer, or RISC, reduced-instruction-set computer; see RISC processor).

Introduction to microprocessor and microcomputer

A microprocessor can do any information-processing that can be expressed, precisely, as a plan. It is a truly general-purpose information processing device.

Basic Term :

Bit: Single binary digit, 0 or 1

Byte: 8 bits. (Smallest unit of memory used in modern computers)

Nibble: 4 bits (2 nibbles = 1 byte), Used to be useful, isn't anymore

Word: 8-64 bits (1 to 8 bytes) Depends on machine!

Microcomputer Architecture : Computer system consist primary of :-

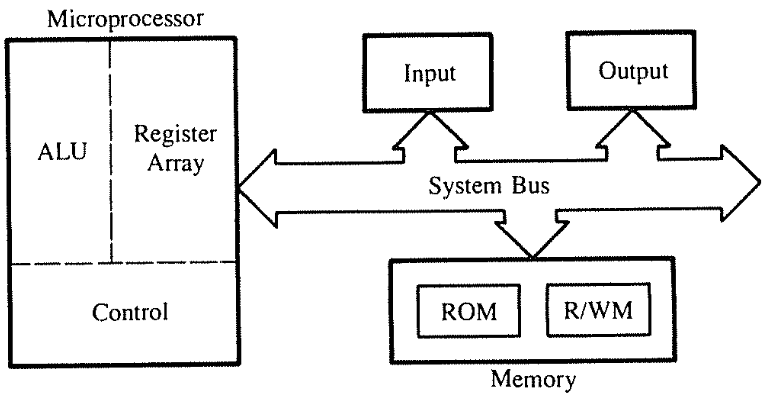
1- Microprocessor.

2-Memory.

3-Input.

4-Output.

5- Buses [Data bus, Address bus, Control bus].



Microcomputer with Bus Architecture

Introduction to microprocessor and microcomputer

Microprocessor Architecture : The internal logic design of the microprocessor called its (architecture), determine how and what various operations are performed by (Microprocessor).

Arithmetic Logic Unit (ALU): The ALU performs the actual numerical and logical operations such as Addition (ADD), Subtraction (SUB), AND, OR etc. It uses data from memory and from Accumulator to perform operations. The results of the arithmetic and logical operations are stored in the accumulator.

Timing and control unit: It generates timing an control signals which are necessary for the execution of the instructions.it controls the data flow between CPU and peripherals.

Registers:

Registers: It is a collection of flip flops use to store a binary word. They are used by the microprocessor for the temporary storage and manipulation of data and instructions.

8085 has the following registers:

1- 8 bit accumulator i.e. register A
2- six 8 bits general purpose registers i.e. B,C,D,E,H,L
3- one 16 bit register i.e. stack pointer.
4- 16 bit Program counter, Status register, Temporary register, Instruction Register.

Introduction to microprocessor and microcomputer

Memory: Memory is an essential component of a microprocessor system; it stores binary information. The memory is made up of semiconductor material used to store the programs and data. All systems contain two main types of memory read-only memory (**ROM**) and random access memory (**RAM**) or read/write memory.

Read-Only Memory (ROM) : It works as a storage medium, used in computers and other electronic devices. Data stored in this memory cannot be modified, It is not a volatile memory . ROM usually stores the startup instructions for the computer, and it also helps to load the operating system.

Random Access Memory (RAM): Is a type of memory that computers used to store data and software's to which it needs to access quickly. It is a volatile memory, that is, the information stored inside vanishes when the computer is turned off .

System Bus : Microprocessor performed these functions using sets of buses [Data bus, Address bus, Control bus].

Data Bus: is a group of 8 lines used for data flow, these lines are bidirectional of 2^{8} =256 numbers. The largest number = 1111 1111 = FF , thus 8085 Microprocessor is called 8bit Microprocessor.

Address Bus: Is a group of 16 lines, identified as A0 – A15. This bus is unidirectional (bit flow in one direction) from Microprocessor to peripheral. Each memory location or peripheral identified with binary number called address. (2^{16} =65536=64K).

Control bus: The control is comprised of various single lines that carry synchronization signals.

<u>Introduction to microprocessor and microcomputer</u>

Clock signal : The time base for synchronization of the internal and external operations of the microprocessor in a microcomputer system is provided by the clock (CLK) input signal. The 8085 generates the clock signal internally by dividing the external supplied clock signal by two. These microprocessor require an external crystal or an RC network to be connected to appropriate pins for setting the operating frequency .

Introduction to 8085 Microprocessor Architecture and Pin Diagram

Internal Architecture of 8085 : the major component of 8085 microprocessor architecture is ALU, Registers, Memory, Buses and control unit (as mentioned in Lect. 1). in Lect. 2 dealing with these components in more details.

Registers unit : The 8085 programming model includes six registers, one accumulator, and one flag register, as shown in Figure (2.1). In addition, it has two 16-bit registers (the stack pointer and the program counter). They can be combined as register pairs - BC, DE, and HL - to perform some 16-bit operations. The programmer can use these registers to store or copy data into the registers by using data copy instructions.

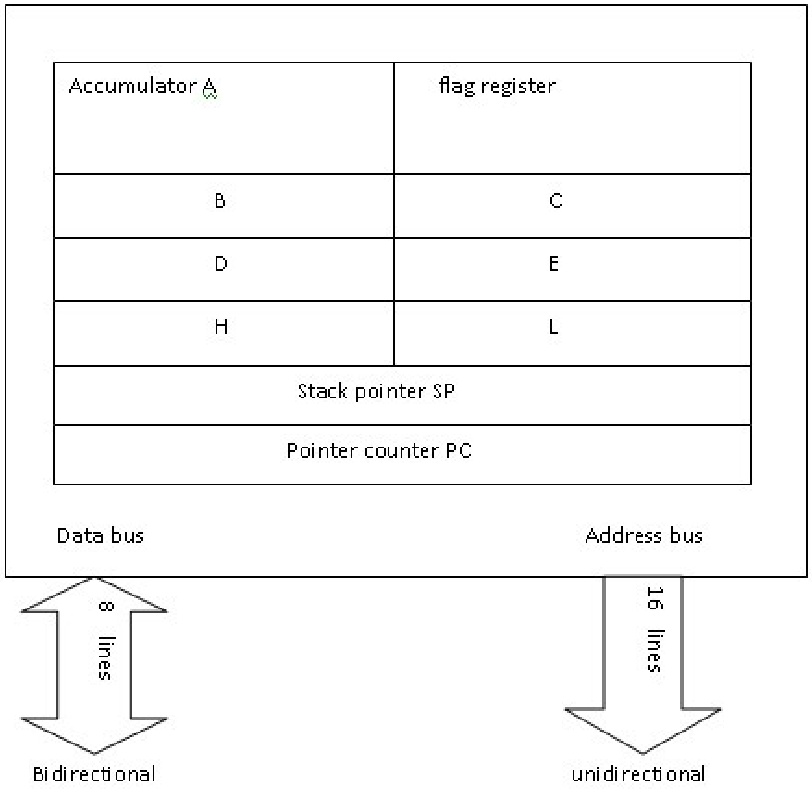


Figure 2.1: 8085 Registers.

Accumulator (A) : The accumulator is an 8-bit register that is a part of (ALU). This register is used to store 8-bit data and to perform arithmetic and logical operations. The result of an operation is stored in the accumulator. The accumulator is also identified as register A.

Flags

The ALU includes five flip-flops, which are set or reset after an operation according to data conditions of the result in the accumulator and other registers. They are called **Zero(Z), Carry (CY), Sign (S), Parity (P), and Auxiliary Carry (AC) flags**; their bit positions in the flag register are shown in the Figure (2.2).

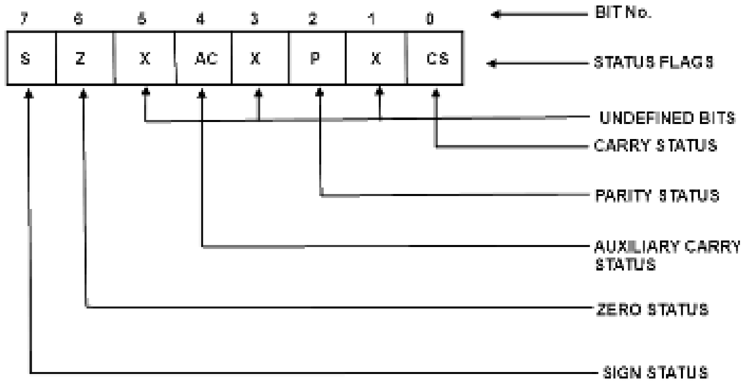


Figure 2.2: Flags Register.

Example : after an addition of two numbers, if the sum in the accumulator is larger than eight bits, the flip-flop uses to indicate a carry (called the Carry flag (CY)) is set to one.

When an arithmetic operation results in zero, the flip-flop called the Zero(Z) flag is set to one. Figure (2.1) shows an 8-bit register, called the flag register, adjacent to the accumulator. However, it is not used as a register; five bit positions out of eight are used to store the outputs of the five flip-flops. The flags are stored in the 8-bit register so that the programmer can examine these flags (data conditions) by accessing the register through an instruction.

Program Counter (PC)

This 16-bit register deals with sequencing the execution of instructions. This register is a memory pointer. The function of the program counter is to point to the memory address from which the next byte is to be fetched. When a byte (machine code) is being fetched, the program counter is incremented by one to point to the next memory location .

Stack Pointer (SP)

The stack pointer is also a 16-bit register used as a memory pointer. It points to a memory location in memory, called the **stack**. The beginning of the stack is defined by loading 16-bit address in the stack pointer. The stack concept is explained later .

8085 System Bus

Typical system uses a number of busses, collection of wires, which transmit binary numbers, one bit per wire. A typical microprocessor communicates with memory and other devices (input and output) using three busses:

- Address Bus
- Data Bus
- Control Bus.

Address Bus : Address Bus consists of 16 wires, A 16 bit binary number allows 2^{16} different numbers, i.e. 0000000000000000 up to 1111111111111111. The size of the address bus determines the size of memory, which can be used. Address bus is unidirectional, i.e. numbers only sent from microprocessor to memory, no other way.

Data Bus

Data Bus: carries ‘data’, in binary form, between µP and other external units, such as Memory Units or I/O Units. Therefor Data Bus is bi-directional. Typical size is 8 or 16 bits. The 8085 Data Bus typically consists of 8 wires. Therefore.

Control Bus

Control Bus are various lines which have specific functions for coordinating and controlling uP operations, some of control line are:

- $\overline{RD}$
- $\overline{WR}$
- IO\$\overline{M}$ (Input Output or Memory)

The Control Bus carries control signals partly unidirectional, partly bi-directional.

8085 Pin description.

- ***Properties***

Single + 5V Supply.

4 Vectored Interrupts (One is Non Maskable).

Serial In/Serial Out Port.

Decimal, Binary, and Double Precision (computer number format) Arithmetic.

Direct Addressing Capability to 64K (2^{16} $=2^{6} * 2^{10}$ =16K) bytes of memory.

The Intel 8085A is a new generation, complete 8 bit parallel central processing unit (CPU).

The 8085A uses a multiplexed data bus. The address is split between the 8bit address bus and the 8bit data bus.

Pin Description

The following describes the function of each pin:

AD0 – AD7 (Input/output)

Multiplexed Address/Data Bus; which are bidirectional, Lower 8 bits of the memory address (or I/0 address) appear on the bus during the first clock cycle of a machine state. It then becomes the data bus during the second and third clock cycles.

A_8 –A_{15} (output)

The most significant 8 bit of the address bus, which are unidirectional.

ALE (Output)

Address Latch Enable: It is a control signal occurs during the first clock cycle of a machine state and enables the address to be latched . It used to separate the address from the data, ***ALE=1 (A0 – A7) and ALE=0 (D0 – D7)***

SO, S1 (Output)

Data Bus Status. Encoded status of the bus cycle:

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

RD (Output 3state)

READ; indicates the selected memory or 1/0 device is to be read and that the Data Bus is available for the data transfer.

WR (Output 3state)

WRITE; indicates the data on the Data Bus is to be written into the selected memory or 1/0 location.

READY (Input)

If Ready is high during a read or write cycle, it indicates that the memory or peripheral is ready to send or receive data. If Ready is low, the CPU will wait for Ready to go high before completing the read or write cycle.

HOLD (Input)

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

HLDA (Output)

HOLD ACKNOWLEDGE; indicates that the CPU has received the Hold request and that it will relinquish the buses in the next clock cycle. HLDA goes low after the Hold request is removed. The CPU takes the buses one half clock cycle after HLDA goes low.

INTR (Input)

It is a general purpose interrupt request signal, It is an active high signal .

$\overline{INTA}$ (Output)

It is used to acknowledge a interrupt. It is an active low signal.

RST5.5, RST6.5, RST 7.5(input)

These are vectored interrupts that transfer the program control to specific memory locations. They have higher priority than INTR interrupts.

TRAP

TRAP has the highest priority. It is used in emergency situation. it is an non-mask able interrupt.

Order of priority

The priority of these interrupts is ordered as shown below

TRAP → RST 7.5 → RST 6.5 → RST 5.5 →INTR

$\overline{\text{RESET IN}}$ (Input)

When the signal on this pin goes low the program counter set to zero and the processor is reset. It is active low signal.

RESETOUT (output)

This signal can be used to reset other device. It is an active high signal.

X1, X2 (input)

these are terminals to be connected to an external crystal oscillator which drives an internal circuitry of the microprocessor to produce a suitable clock for the operation of microprocessor.

CLK (output)

It is a clock output for user, which can be used for other digital integrated circuits.

SID (input)

It is data line for serial input. The data on this line is loaded into the 7th bit of the accumulator when rim (read interrupt mask) instruction is executed.

SOD (output)

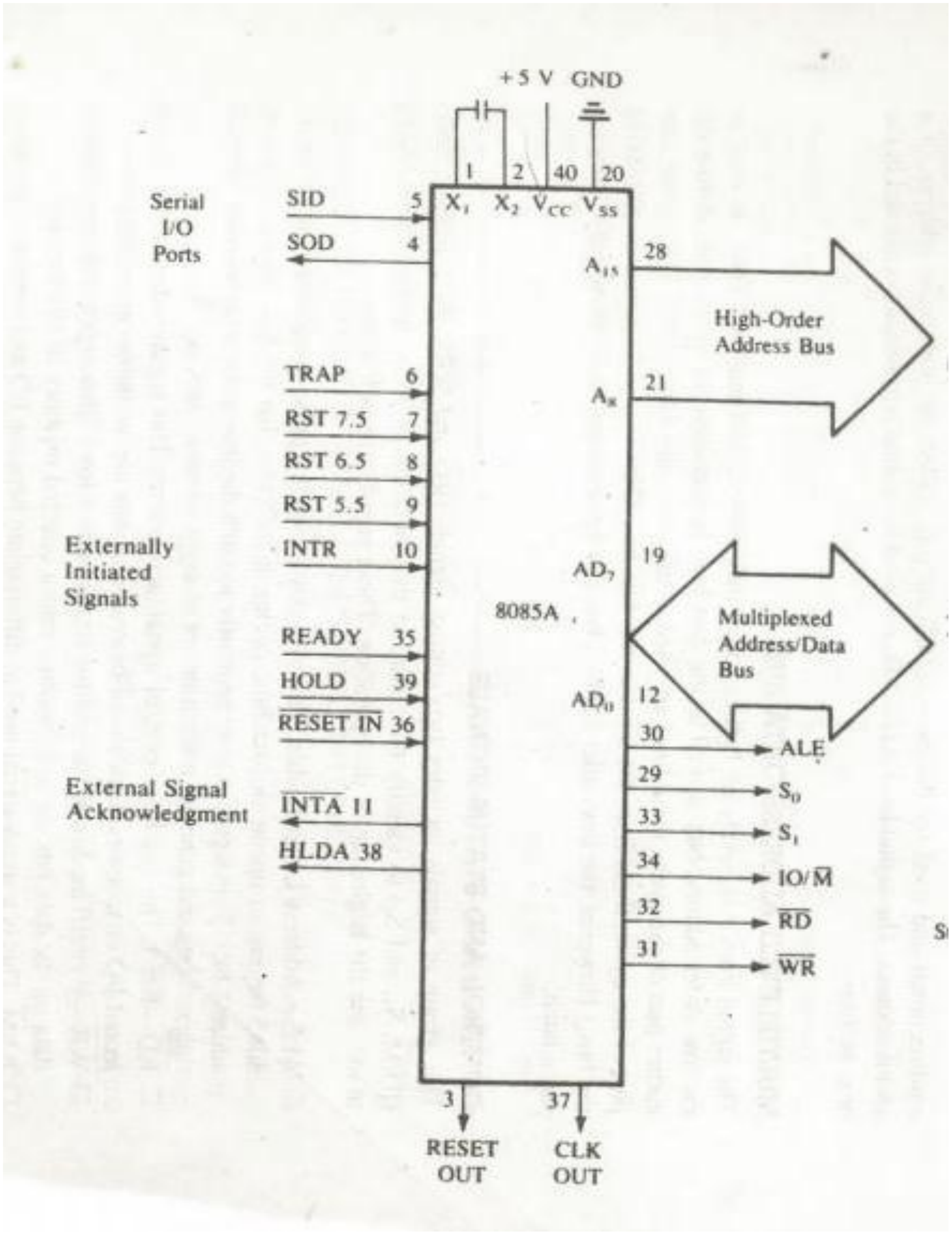
It is data line for serial output. The 7th bit of the accumulator is output on sod line when sim instruction is executed.

Vcc

It is +5 volt dc supply.

Vss

It is the ground reference.



`8085 Microprocessor Pin Diagram

<u>8085 microprocessor instruction set</u>

Instruction Set : An instruction is a command given to the computer to perform a specified operation on a given data . A memory location for Intel 8085 microprocessor is designed to accumulate 8-bit data. If 16- bit data are to be stored, they are stored in consecutive memory locations. The various techniques to specify data for instructions are:

(1) 8-bit or 16-bit data may be directly given in the instruction itself.

(2) The address of the memory location, I/O port or I/O device, where data resides, may be given in the instruction itself.

(3)In some instructions only one register is specified. The content of the specified register is one of the operand and other operand is the accumulator.

(4) Some instructions specify two registers. The contents of the registers are the required data.

Due to different ways of specifying data for instruction are not of same length.

So there are three types of instructions of Intel 8085:

(1)Single byte instruction

The content information including operands in the opcode itself .These are of one byte.

Ex-MOV A,B ; Move the content of register B to A

(2)two - byte instruction

In case of two byte instruction the 1st byte of the instruction is opcode and 2nd byte is either data or address. Both bytes are stored in two consecutive memory locations.

Ex-MVI B,05; Move 05 to register B

8085 microprocessor instruction set

(3)three - byte instruction

In case of three bytes instruction the 1st byte of instruction is opcode and 2nd and 3rd byte of instruction are either 16-bit data or 16-bit address. They are stored in three consecutive memory locations.

Ex-LXI H, 2400H ; load H-L pair with 2400H.

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

1- data transfer operations
2- arithmetic operations
3- logical operations
4- branching operations
5- machine-control operations

- **Data Transfer Operations Group**

Opcode	**Operand**	**Description**
MOV	Rd, Rs	This instruction copies the contents of the source register into the destination register; the contents of the source register are not altered. If one of the operands is a memory location, its location is specified by the contents of the HL registers.
Copy from source	M, Rs	
to destination	Rd, M	
		Example: MOV B, C or MOV B, M

8085 microprocessor instruction set

MVI ***Move immediate 8-bit***	Rd, data M, data	The 8-bit data is stored in the destination register or memory. If the operand is a memory location, its location is specified by the contents of the HL registers. ***Example:*** MVI B, 57 or MVI M, 57
LDA ***Load accumulator***	16-bit address	The contents of a memory location, specified by a16-bit address in the operand, are copied to the accumulator. The contents of the source are not altered. ***Example:*** LDA 2034
LDAX ***Load accumulator indirect***	B/D Reg. pair	The contents of the designated register pair point to a memory location. This instruction copies the contents of that memory location into the accumulator. The contents of either the register pair or the memory location are not altered. ***Example:*** LDAX B
STA ***Store accumulator***	16-bit address	The contents of the accumulator are copied into the memory location specified by the operand. This is a 3-byte instruction, the

direct		second byte specifies the low-order address and the third byte specifies the high-order address. ***Example:*** STA 4350
STAX ***Store accumulator indirect***	Reg. pair	The contents of the accumulator are copied into the memory location specified by the contents of the operand (register pair). The contents of the accumulator are not altered. ***Example:*** STAX B
LXI ***Load register pair immediate***	Reg. pair, 16-bit data	The instruction loads 16-bit data in the register pair designated in the operand. ***Example:*** LXI H, 2034
LHLD ***Load H and L registers direct***	16-bit address	The instruction copies the contents of the memory location pointed out by the 16-bit address into register L and copies the contents of the next memory location into register H. The contents of source memory locations are not altered. ***Example:*** LHLD 2040

8085 microprocessor instruction set

SHLD ***Store H and L registers direct***	16-bit address	The contents of register L are stored into the memory location specified by the 16-bit address in the operand and the contents of H register are stored into the next memory location by incrementing the operand. The contents of registers HL are not altered. This is a 3-byte instruction, the second byte specifies the low-order address and the third byte specifies the high-order address. ***Example:*** SHLD 2470
XCHG ***Exchange H and L with D and E***	none	The contents of register H are exchanged with the contents of register D, and the contents of register L are exchanged with the contents of register E. ***Example:*** XCHG
PCHL ***Load program counter with HL contents***	none	The contents of registers H and L are copied into the program counter. The contents of H are placed as the high-order byte and the contents of L as the low-order byte. ***Example:*** PCHL

SPHL ***Copy H and L registers to the stack pointer***	none	The instruction loads the contents of the H and L registers into the stack pointer register, the contents of the H register provide the high-order address and the contents of the L register provide the low-order address. The contents of the H and L registers are not altered. ***Example:*** SPHL
XTHL ***Exchange H and L with top of stack***	none	The contents of the L register are exchanged with the stack location pointed out by the contents of the stack pointer register. The contents of the H register are exchanged with the next stack location (SP+1); however, the contents of the stack pointer register are not altered. ***Example:*** XTHL

8085 microprocessor instruction set

Example : move the contents of memory location 2850 to accumulator.

```
1-     LDA 2850

       HLT

2-     LXI    H,2850
       MOV  A,M
       HLT


3-     LXI     B, 2850
       LDAX  B
       HLT : Terminate program execution
```

Example : move the contents of memory location 2850 to register C .

```
1-     LXI       H,2850
       MOV   C,M
       HLT
2-     LDA     2850
       MOV   C,A
       HLT
```

Example : Store the data byte 32H into memory location 4000H.

```
     MVI A, 32H          : Store 32H in the accumulator
     STA 4000H           : Copy accumulator contents at address 4000H
      HLT
```

8085 microprocessor instruction set

Example : Exchange the contents of memory locations 2000H and 4000H

1- LDA 2000H : Get the contents of memory location 2000H into accumulator
MOV B, A : Save the contents into B register
LDA 4000H : Get the contents of memory location 4000H into accumulator
STA 2000H : Store the contents of accumulator at address 2000H
MOV A, B : Get the saved contents back into A register
STA 4000H : Store the contents of accumulator at address 4000H
HLT

2- LXI H 2000H : Initialize HL register pair as a pointer to memory location 2000H.
LXI D 4000H : Initialize DE register pair as a pointer to memory location 4000H.
MOV B, M : Get the contents of memory location 2000H into B register.
LDAX D : Get the contents of memory location 4000H into A register.
MOV M, A : Store the contents of A register into memory location 2000H.
MOV A, B : Copy the contents of B register into accumulator.
STAX D : Store the contents of A register into memory location 4000H.
HLT

Logical and Branching Operations

Logical Operation Instructions : These instructions perform logical operations on data stored in registers, memory and status flags.

The logical operations are:

- AND
- OR
- XOR
- Rotate
- Compare
- Complement

AND, OR, XOR : Any 8-bit data, or the contents of register, or memory location can logically have AND operation, OR operation, XOR operation with the contents of accumulator. The result is stored in accumulator.

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

Compare : Any 8-bit data, or the 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 flags.

Complement : The contents of accumulator can be complemented. Each 0 is replaced by 1 and each 1 is replaced by 0.

Logical and Branching Operations

Opcode	**Operand**	**Description**
CMP	R	*Compare register or memory with accumulator*
	M	The contents of the operand (register or memory) are compared with the contents of the accumulator. The result of the comparison is shown by setting the flags of the PSW as follows:
		if (A) < (reg/mem): carry flag is set
		if (A) = (reg/mem): zero flag is set
		if (A) > (reg/mem): carry and zero flags are reset
		Example: CMP B or CMP M
CPI	8-bit data	*Compare immediate with accumulator*
		The second byte (8-bit data) is compared with the contents of the accumulator. The result of the comparison is shown by setting the flags of the PSW as follows:
		if (A) < data: carry flag is set
		if (A) = data: zero flag is set
		if (A) > data: carry and zero flags are reset
		Example: CPI 89H
ANA		*Logical AND register or memory with accumulator*
		The contents of the accumulator are logically ANDed with the contents of the operand (register or memory), and the result is placed in the accumulator. If the operand is a memory location, its address is specified by the contents of HL registers. S, Z, P are modified to reflect the result of

		the operation. CY is reset. AC is set. *Example:* ANA B or ANA M
ANI	8-bit data	*Logical AND immediate with accumulator* The contents of the accumulator are logically ANDed with the 8-bit data (operand) and the result is placed in the accumulator. S, Z, P are modified to reflect the result of the operation. CY is reset. AC is set. *Example:* ANI 86H
XRA	R M	*Exclusive OR register or memory with accumulator* The contents of the accumulator are Exclusive ORed with the contents of the operand (register or memory), and the result is placed in the accumulator. If the operand is a memory location, its address is specified by the contents of HL registers. S, Z, P are modified to reflect the result of the operation. CY and AC are reset. *Example:* XRA B or XRA M
XRI	8-bit data	*Exclusive OR immediate with accumulator* The contents of the accumulator are Exclusive ORed with the 8-bit data (operand) and the result is placed in the accumulator. S, Z, P are modified to reflect the result of the operation. CY and AC are reset. *Example:* XRI 86H

Logical and Branching Operations

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

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

Logical and Branching Operations

STC	none	*Set Carry* The Carry flag is set to 1. No other flags are affected. *Example:* STC

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

sample problem:

(4400H) = 55H   Result = (4300B) = AAB

program:

LDA 4400 : Get the number

CMA : Complement number

STA 4300H : Store the result

HLT : Terminate program execution

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

Sample problem:

(4200H) = 55H   Result = (4300H) = AAH + 1 = ABH

program:

LDA 4200H : Get the number

Logical and Branching Operations

CMA : Complement the number

ADI, 01 H : Add one in the number

STA 4300H : Store the result

HLT : Terminate program execution

Example : Pack the two unpacked BCD numbers stored in memory locations 4200H and 4201H and store result in memory location 4300H. Assume the least significant digit is stored at 4200H.

Sample problem:

(4200H) = 04 (4201H) = 09 Result = (4300H) = 94

Program :

LDA 4201H : Get the Most significant BCD digit

RLC

RLC

RLC

RLC : Adjust the position of the second digit (09 is changed to 90)

ANI FOH : Make least significant BCD digit zero

MOV C, A : store the partial result

LDA 4200H : Get the lower BCD digit

ADD C : Add lower BCD digit

Logical and Branching Operations

STA 4300H : Store the result

HLT : Terminate program execution

Branching Operation Instructions : These instructions allows the microprocessor to change the sequence of program either conditionally or under certain test conditions. The branch instructions includes :

(1) Jump instructions.

(2) Call and Return instructions.

(3) Restart instructions.

Opcode	**Operand**	**Description**	
JMP	16-bit address	*Jump unconditionally* The program sequence is transferred to the memory location specified by the 16-bit address given in the operand. *Example*: JMP 2034H or JMP XYZ	
JX Where X is the condition	16-bit address	*Jump conditionally* The program sequence is transferred to the memory location specified by the 16-bit address given in the operand based on the specified flag of the PSW as described below. *Example*: JZ 2034H or JZ XYZ	
JC	16-bit address	Jump on Carry	CY=1
JNC	16-bit address	Jump on no Carry	CY=0

Logical and Branching Operations

JP	16-bit address	Jump on positive	S=0
JM	16-bit address	Jump on minus	S=1
JZ	16-bit address	Jump on zero	Z=1
JNZ	16-bit address	Jump on no zero	Z=0
JPE	16-bit address	Jump on parity even	P=1
JPO	16-bit address	Jump on parity odd	P=0
CALL	16-bit address	*Unconditional subroutine call* The program sequence is transferred to the memory location specified by the 16-bit address given in the operand. Before the transfer, the address of the next instruction after CALL (the contents of the program counter) is pushed onto the stack. *Example*: CALL 2034H or CALL XYZ	
CX Where x is the condition	Operand: 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 the specified flag of the PSW as described below. Before the transfer, the address of the next instruction after the call (the contents of the program counter) is pushed onto the stack. *Example:* CZ 2034H or CZ XYZ	
CC	16-bit address	Call on Carry CNC	CY = 1
CNC	16-bit address	Call on no Carry	CY = 0
CP	16-bit address	CP Call on positive	S = 0

<u>Logical and Branching Operations</u>

CM	16-bit address	Call on minus	S = 1
CZ	16-bit address	Call on zero	Z = 1
CNZ	16-bit address	Call on no zero	Z = 0
CPE	16-bit address	Call on parity even	P = 1
CPO	16-bit address	Call on parity odd	P = 0
RET	none	*Return from subroutine 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	
RX Where X is the condition	none	*Return from subroutine conditionally* The program sequence is transferred from the subroutine to the calling program based on the specified flag of the PSW as described below. The two bytes from the top of the stack are copied into the program counter, and program execution begins at the new address. *Example*: RZ	
RC	none	Return on Carry	CY = 1
RNC	none	Return on no Carry	CY = 0
RP	none	Return on positive	S = 0
RM	none	Return on minus	S = 1
RZ	none	Return on zero	Z = 1
RNZ	none	Return on no zero	Z = 0

Logical and Branching Operations

RPE	none	Return on parity even	P = 1
RPO	none	Return on parity odd	P = 0

RST 0-7 *Restart*

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

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

Logical and Branching Operations

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

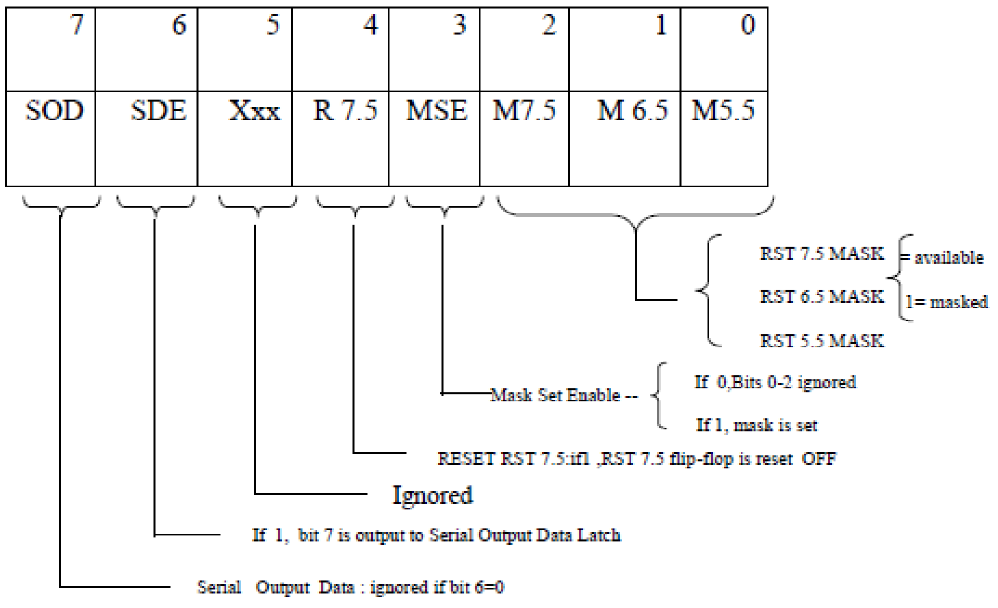
Interrupt	Restart Address
TRAP	0024H
RST 5.5	002CH
RST 6.5	0034H
RST 7.5	003CH

Control Operation Instructions : The control instructions control the operation of microprocessor.

Opcode	**Operand**	**Description**
NOP	none	*No operation* No operation is performed. The instruction is fetched and decoded. However no operation is executed. *Example*: NOP
HLT	none	*Halt and enter wait state* The CPU finishes executing the current instruction and halts any further execution. An interrupt or reset is necessary to exit from the halt state. *Example*: HLT
DI	none	*Disable interrupts* The interrupt enable flip-flop is reset and all the interrupts except the TRAP are disabled. No flags are affected.

Logical and Branching Operations

		Example: DI
EI	none	*Enable interrupts* The interrupt enable flip-flop is set and all interrupts are enabled. No flags are affected. After a system reset or the acknowledgement of an interrupt, the interrupt enable flip-flop is reset, thus disabling the interrupts. This instruction is necessary to reenable the interrupts (except TRAP). *Example*: EI
SIM	none	*Set interrupt mask* This is a multipurpose instruction and used to implement the 8085 interrupts 7.5, 6.5, 5.5, and serial data output. The instruction interprets the accumulator contents as follows. *Example*: SIM



Logical and Branching Operations

Example -1: Enable all the interrupts in an 8085 system ?

Instructions

```
EI                 ; Enable interrupts
MVI A ,08H         ; Load bit Pattern to enable RST 7.5 ,6.5 and 5.5
SIM                ;Enable RST 7.5,6.5,and 5.5
```

Bit D3= 1 in the accumulator makes the instruction SIM functional , and bits D2,

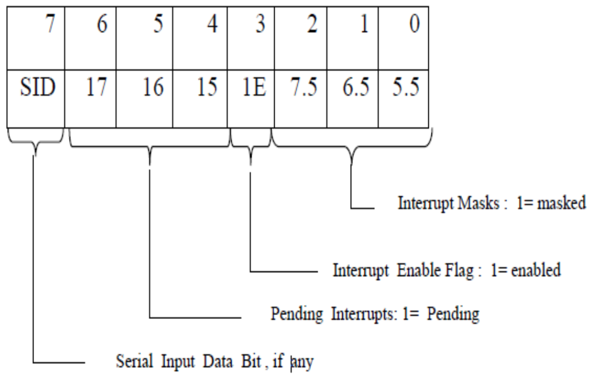
D1, and D0 =0 enable the interrupts 7.5 ,6.5 and 5.5

Example: Reset the 7.5 interrupt from Example - 1

Instructions

```
MVI A, 18H         ; Set D4 = 1
SIM                ; Reset 7.5 interrupt flip-flop
```

RIM	none	*Read interrupt mask* This is a multipurpose instruction used to read the status of interrupts 7.5, 6.5, 5.5 and read serial data input bit. The instruction loads eight bits in the accumulator with the following interpretations. *Example*: RIM


7 6 5 4 3 2 1 0
SID 17 16 15 1E 7.5 6.5 5.5
Interrupt Masks : 1= masked
Interrupt Enable Flag : 1= enabled
Pending Interrupts: 1= Pending
Serial Input Data Bit , if any

Logical and Branching Operations