INSTRUCTION SET OF 8085

Instruction Set of 8085

- An instruction is a Command/ 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 word is represented by an 8-bit binary value.
- These 8-bits of binary value is called ***Op-Code***or ***Instruction Byte***.

Classification of Instruction Set

- Data Transfer Instruction
- Arithmetic Instructions
- Logical Instructions
- Branching Instructions
- I/O and Machine Control Instructions

Data Transfer Instruction

Data Transfer Instructions

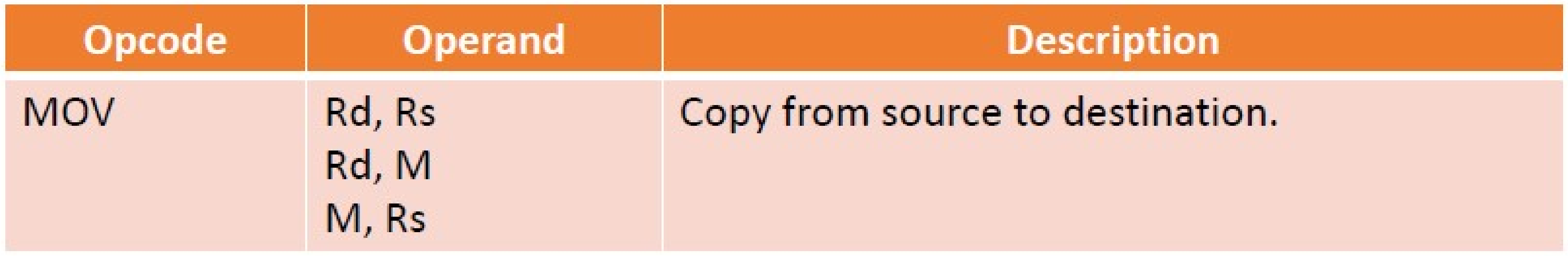
- These instructions move/ copy data between registers (R or Rp), or between memory (M) and registers.
- These instructions copy data from source to destination.
- While copying, the contents of destination is modified by content of source.
- But, content of source are not modified.

Data Transfer Instructions

i. MOV Rd, Rs - Move Data; Move content of the source register to destination register.

ii. MOV Rd, M -Move content of memory register to destination register.
iii. MOV M, Rs. -Move the content of register to memory.
iv. MVI R, data. -Move immediate data to register.
v. MVI M, data- Move immediate data to memory.
vi. LXI Rp, data 16- Load register pair immediate.
vii. LDA addr- Load Accumulator direct.
viii. STA addr- Store accumulator direct.
ix. LHLD addr- Load H-L pair direct
x. SHLD addr- Store H-L pair direct
xi. LDAX Rp. -LOAD accumulator indirect
xii. STAX Rp- Store accumulator indirect
xiii. XCHG- Exchange the contents of H-L with D-E pair [H-L] <--> [D-E].

Data Transfer Instructions-1



Opcode	Operand	Description
MOV	Rd, Rs Rd, M M, Rs	Copy from source to destination.

- This instruction copies 8 bit content of Rs into Rd.
- It can be of 3 types
  - Register to Register; eg. MOV A, C
  - Register to Memory; eg.MOV M, C;
  - Memory to Register; eg.MOV B, M
- If one of the operands is a memory location, its location is specified by the contents of the HL registers.(eg. using LXI)
- **Practice:** copy data 32H stored at 3020H memory to Reg. C.

Data Transfer Instructions-2

Opcode	Operand	Description
MVI	Rd, Data M, Data	Move immediate 8-bit

- Copy 8-bit data directly into the
  - Destination register; MVI A, 57H
  - or
  - Memory; MVI M, 57H

- If the operand is a memory location, its location is specified by the contents of the H-L registers.

- Practice: copy data 30H to memory location 3025H.

Data Transfer Instructions-3, 4

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

- This instruction loads 16-bit data in the register pair (BC-B, DE-D, HL-H)
- HL pair is used as memory Pointer.
- Eg. LXI H, 2034 H

Opcode	Operand	Description
LDA	16-bit address	Load Accumulator

- The contents of a memory location, specified by a 16-bit address in the operand, are copied to the accumulator.
- **Example:** LDA 2034H
- **Practice**: copy content of memory address 2060H to A without using any other reg.

Data Transfer Instructions-5, 6

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

- This instruction copies the contents of that memory location pointed by register pair (BC or DE) into the accumulator.
- **Example:** LDAX B
- Note: HL pair is not used in this instruction.
- **Practice:** Copy 10H stored at memory location 3020H to accumulator using LDAX instruction.

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

- This instruction copies the contents of memory location pointed out by 16-bit address into register L. (([16 bit address] → Reg. L)

- It copies contents of next memory location into reg. H. eg.[16 bit address +1] → Reg. H)
- **Example:** LHLD 2040 H

Data Transfer Instructions-7, 8

Opcode	Operand	Description
STA	16-bit address	Store accumulator direct

- The contents of accumulator are stored into the memory location specified by the operand.
- **Example:** STA 2500 H
- **Practice:** copy content of A to memory address 2500H without using any other register.

Opcode	Operand	Description
STAX	Reg. pair	Store accumulator indirect

- The contents of accumulator are stored/copied into the memory location specified by the contents of the register pair.
- **Example:** STAX B
- **Practice:** copy 15H in A and store content of A to memory address 3100H using STAX.

Data Transfer Instructions-9, 10

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

- The contents of register L are stored into memory location specified by the 16-bit address. ([16 bit address] $\leftarrow$ Reg. L)

- The contents of register H are stored into the next memory location. [16 bit address +1] $\leftarrow$ Reg. H)

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 contents of register L are exchanged with the contents of register E.

- **Example :** XCHG i.e. H $\leftrightarrow$ D; L $\leftrightarrow$ E

Arithmetic Instructions

2. Arithmetic Instructions

These instructions perform the operations like:

- Addition
- Subtraction
- Increment
- Decrement

Addition:

- Any 8-bit number, or the contents of register, or the contents of memory location can be added to the contents of accumulator.
- The result (sum) is stored in the accumulator.
- No two other 8-bit registers can be added directly.

Subtraction:

- Any 8-bit number, or the contents of register, or the contents of memory location can be subtracted from the contents of accumulator.
- The result is stored in the accumulator.
- Subtraction is performed in 2’s complement form.
- If the result is negative, it is stored in 2’s complement form.
- No two other 8-bit registers can be subtracted directly.

Arithmetic Instructions

Increment / Decrement

- The 8-bit contents of a register or a memory location can be incremented or decremented by 1.

- The 16-bit contents of a register pair can be incremented or decremented by 1.

- Increment or decrement can be performed on any register or a memory location.

Arithmetic Instructions

i. ADD r. (Add register to accumulator) [A] ← [A] + [r].
ii. ADD M. (Add memory to accumulator) [A] ← [A] + [[H-L]].
iii. ADC r. (Add register with carry to accumulator). [A] ← [A] + [r] + [CS].
iv. ADC M. (Add memory with carry to accumulator) [A] ← [A] + [[H-L]] [CS].
v. ADI data (Add immediate data to accumulator) [A] ← [A] + data.
vi. ACI data (Add with carry immediate data to accumulator). [A] ← [A] + data + [CS].
vii. DAD rp. (Add register paid to H-L pair). [H-L] ← [H-L] + [rp].
viii. SUB r. (Subtract register from accumulator). [A] ← [A] – [r].
ix. SUB M. (Subtract memory from accumulator). [A] ← [A] – [[H-L]].
x. SBB r. (Subtract register from accumulator with borrow). [A] ← [A] – [r] – [CS].
xi. SBB M. (Subtract memory from accumulator with borrow). [A] ← [A] – [[H- L]] – [CS].
xii. SUI data. (Subtract immediate data from accumulator) [A] ← [A] – data.
xiii. SBI data. (Subtract immediate data from accumulator with borrow). [A] ← [A] – data – [CS].
xiv. INR r (Increment register content) [r] ← [r] +1.
xv. INR M. (Increment memory content) [[H-L]] ← [[H-L]] + 1.
xvi. DCR r. (Decrement register content). [r] ← [r] – 1.
xvii. DCR M. (Decrement memory content) [[H-L]] ← [[H-L]] – 1.
xviii. INX rp. (Increment register pair) [rp] ← [rp] + 1.
xix. DCX rp (Decrement register pair) [rp] ← [rp] -1.
xx. DAA (Decimal adjust accumulator) .

Arithmetic Instructions- Addition

Opcode	Operand	Description
ADD	R M	Add register or memory to accumulator

- Contents of register or memory are added to the contents of accumulator.
- The result is stored in accumulator. [A] ← [A] + [r]

- If the operand is memory location, its address is specified by H-L pair.
- [A] ← [A] + [M] i.e. [A] ← [A] + [[H-L]].

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

- **Example: ADD B or ADD M**
- **Practice:** A= FFH and B= 82H

Arithmetic Instructions- Addition

Opcode	Operand	Description
ADC	R M	Add register or memory to accumulator with carry

- Contents of register or memory and Carry Flag (CY) are added to the contents of accumulator. The result is stored in accumulator.
  [A] ← [A] + [r] + [CY].

- If the operand is memory location, its address is specified by H-L pair.
- [A] ← [A] + [[H-L]] [CY].
- All flags are modified to reflect the result of the addition.
- **Example : ADC B or ADC M**

- **Note:** This instruction is used only when some carry is generated from previous addition.eg. It can be used to add two 16 bit data.
- **Practice:** Add 3465H and 2AB1H using ADC instruction. (HL ← DE+BC)

Arithmetic Instructions- Addition

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

- 8-bit data is added to the contents of accumulator.
- The result is stored in accumulator. [A] ← [A] + data.
- All flags are modified to reflect the result of the addition.
- **Example : ADI 45 H**

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

- 8-bit data and the Carry Flag (CY) are added to contents of accumulator.
- The result is stored in accumulator. [A] ← [A] + data + [CY].
- All flags are modified to reflect the result of the addition.
- **Example : ACI 45 H**

Arithmetic Instructions- Addition

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

- The 16-bit contents of the register pair are added to the contents of H-L pair.
- The result is stored in H-L pair.
- If the result is larger than 16 bits, then CY is set.

- [H-L] ← [H-L] + [rp].
- No other flags are changed.

- **Example : DAD B**

Arithmetic Instructions- Subtraction

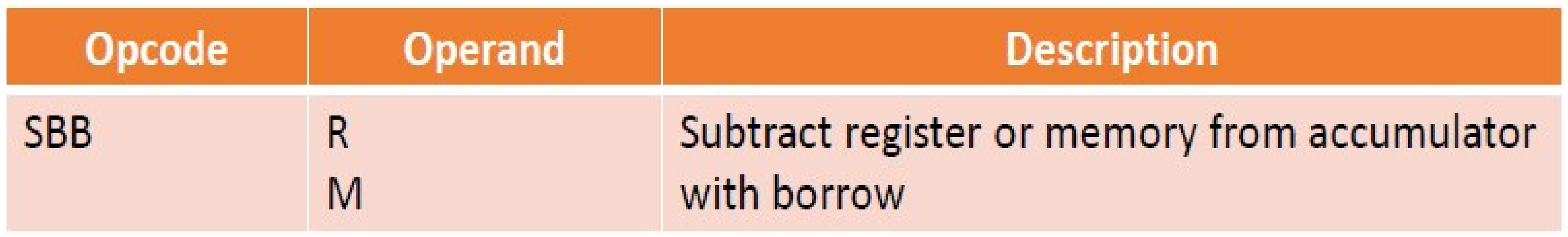
Opcode	Operand	Description
SUB	R M	Subtract register or memory from accumulator

- The contents of the register or memory location are subtracted from the contents of the accumulator.
- The result is stored in accumulator. [A] ← [A] – [r].

- If the operand is memory location, its address is specified by H-L pair. [A] ← [A] – [[H-L]]

- All flags are modified to reflect the result of subtraction.
- **Example : SUB B or SUB M**

Arithmetic Instructions- Subtraction



Opcode	Operand	Description
SBB	R M	Subtract register or memory from accumulator with borrow

- The contents of the register or memory location and Borrow Flag (i.e. CY)are subtracted from the contents of the accumulator.
- The result is stored in accumulator. [A] ← [A] – [r] – [CY]

- If the operand is memory location, its address is specified by H-L pair. [A] ← [A] – [[H- L]] – [CY]
- All flags are modified to reflect the result of subtraction.
- **Example : SBB B or SBB M**
- **Note:** CY flag has dual function, represents carry in addition and borrow in subtraction.

Arithmetic Instructions- Subtraction

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

- The 8-bit data is subtracted from the contents of the accumulator.
- The result is stored in accumulator. [A] ← [A] – data.
- All flags are modified to reflect the result of subtraction.
- **Example : SUI 45 H**

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

- 8-bit data and the Borrow Flag (i.e. CY) is subtracted from contents of accumulator.
- The result is stored in accumulator. . [A] ← [A] – data – [CY]
- All flags are modified to reflect the result of subtraction.
- **Example : SBI 45 H**

Arithmetic Instructions- Increment

Opcode	Operand	Description
INR	R M	Increment register or memory by 1

- The contents of register or memory location are incremented by 1.
- The result is stored in the same place. [r] ← [r] +1
- If the operand is a memory location, its address is specified by the contents of H-L pair. [[H-L]] ← [[H-L]] + 1
- **Example : INR B or INR M**

Opcode	Operand	Description
INX	R	Increment register pair by 1

- The contents of register pair are incremented by 1.
- The result is stored in the same place. [rp] ← [rp] + 1.
- **Example : INX H**

Arithmetic Instructions- Decrement

Opcode	Operand	Description
DCR	R 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. [r] ← [r] – 1
- If the operand is a memory location, its address is specified by the contents of H-L pair. [[H-L]] ← [[H-L]] – 1
- **Example : DCR B or DCR M**

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. [rp] ← [rp] -1
- **Example : DCX H**

Arithmetic Instructions

- DAA- Decimal Adjust Accumulator
- Operand is Implicit.
- Adjust the contents of accumulator A into BCD for after BCD addition.
- Used after ADD or ACI instructions.
- Format- DAA
- In BCD addition:
- If result in A (D3- D0)> 9, add 6 to LSB four bits (i.e. to D3-D0)
- If result in A (D7- D4)> 9, add 6 to MSB four bits (i.e. to D7-D4)

Logical Instructions

Logical 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

Logical Instructions

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.

Logical Instructions

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 Instructions

i. ANA r. (AND register with accumulator)

ii. ANA M. (AND memory with accumulator).

iii. ANI data. (AND immediate data with accumulator)

iv. ORA r. (OR register with accumulator)

v. ORA M. (OR memory with accumulator)

vi. ORI data. (OR immediate data with accumulator)

vii. XRA r. (EXCLUSIVE – OR register with accumulator)

viii. XRA M. (EXCLUSIVE-OR memory with accumulator)

ix. XRI data. (EXCLUSIVE-OR immediate data with accumulator).

x. CMA. (Complement the accumulator)

x. CMC. (Complement the carry status)

xi. STC. (Set carry status)

xii. CMP r. (Compare register with accumulator)

xiii. CMP M. (Compare memory with accumulator

xiv. CPI data. (Compare immediate data with accumulator)

xv. RLC (Rotate accumulator left)

xvi. RRC. (Rotate accumulator right)

xvii. RAL. (Rotate accumulator left through carry)

xviii. RAR. (Rotate accumulator right through carry)

Logical Instructions : AND

Opcode	Operand	Description
ANA	R M	Logical AND register or memory with accumulator

- The contents of the 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 the contents of H-L pair.
- S, Z, P are modified to reflect the result of the operation.
- CY is reset and AC is set.
- **Example: ANA B or ANA M.**

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

- The contents of the accumulator are logically ANDed with 8-bit data.
- **Example: ANI 86H.**

Logical Instructions - OR

Opcode	Operand	Description
ORA	R M	Logical OR register or memory with accumulator

- The contents of the accumulator are logically Ored with the contents of the 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 H-L pair.
- **Example : ORA B or ORA M.**

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

- The contents of the accumulator are logically ORedwith the 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**

Logical Instructions : OR

Opcode	Operand	Description
XRA	R M	Exclusive OR register or memory with accumulator

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

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

- The contents of the accumulator are XORed with the 8-bit data.
- The result is placed in the accumulator.
- **Example : XRI 86H.**

Logical Instructions : Compare

Opcode	Operand	Description
CMP	R M	Compare register or memory with accumulator

- The contents of the operand (register or memory) are compared with the contents of the accumulator.
- Both contents are preserved.
- The result of the comparison is shown by setting the flags of the PSW as follows:

- if (A) < (reg/mem): carry flag is set
- if (A) = (reg/mem): zero flag is set
- if (A) > (reg/mem): carry and zero flags are reset.

- **Example : CMP B or CMP M**

Logical Instructions : Compare

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 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**

Logical Instructions- Rotate

Opcode	Operand	Description
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.**

Opcode	Operand	Description
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.**

Logical Instructions - Rotate

Opcode	Operand	Description
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.

Opcode	Operand	Description
RAR	None	Rotate accumulator right through carry

- Each binary bit of A 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 MSB position D7.
- CY is modified according to bit D0.
- S, Z, P, AC are not affected.
- **Example : RAR.**

Logical Instructions - Complement

Opcode	Operand	Description
CMA	None	Complement accumulator

- The contents of the accumulator are complemented.
- No flags are affected.
- Example : CMA.

Opcode	Operand	Description
CMC	None	Complement carry

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

Opcode	Operand	Description
STC	None	Set carry

- The Carry flag is set to 1.
- No other flags are affected.
- **Example : STC.**

Branching Instructions

Branching Instructions

- The branching instruction alter the normal sequential flow and allow MP to change the sequence of the program (either unconditionally or udder certain test condition).
- MP executes machine codes in sequential manner (one memory location to next)
- Branch instructions instruct MP to go to different memory location and MP continues executing codes from that new location.
- Branch instructions are of three types
- Jump
- Call and Return
- Restart

Branching Instructions : JUMP Unconditionally

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 2034 H.** (C3, 34H, 20H)

- Can be specified using a ***Label*** (or a name).
  - While writing a program we may not know the exact memory location to which program sequence should be directed.
  - Use either label or 16 bit address
  - Same label can not be used for different locations.

Branching Instructions : Jump Conditionally

Opcode	Operand	Description
Jx	16-bit address	Jump conditionally

- The program sequence is transferred to the memory location specified by the 16-bit address given in the operand.
-
- MP make decisions based on certain conditions indicated by the flags. Then, MP decides to change or not to change the sequence of a program accordingly.
- Four flags used by jump instructions

    1. Carry Flag
    2. Zero Flag
    3. Sign Flag
    4. Parity Flag

- **Example : JZ 2034 H; JNZ down**

Branching Instructions : Jump Conditionally

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

Branching Instructions : CALL

Opcode	Operand	Description
CALL	16-bit address	Call unconditionally

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

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

- **Example : CALL 2034 H.**

Branching Instructions: Call Conditionally

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 the specified flag of the PSW.
- 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 2034 H.**

Branching Instructions: 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

Branching Instructions : Return

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.**

Branching Instructions : Return conditionally

Opcode	Operand	Description
Rx	None	~~Call~~ conditionally Return

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

Branching Instructions: Return 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

Branching Instructions: Restart

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.**

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

I/O, Control and Stack Instructions

I/O Instructions

Opcode	operand	Description
IN	8 Bit Port address	Copies data from I/O Port to accumulator

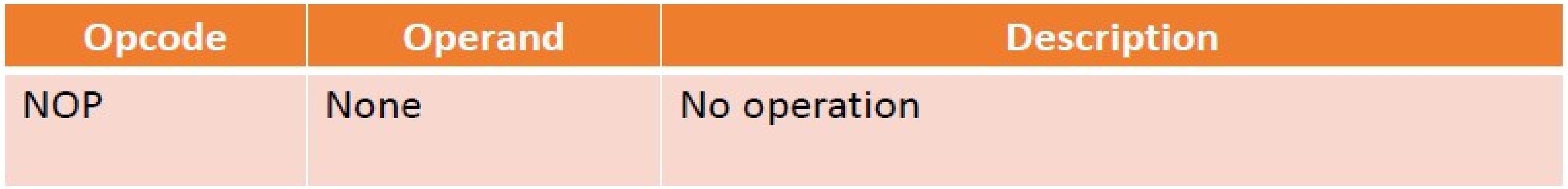
- This instruction takes (copies/ reads) data from an input device and places the data byte in the accumulator.
- [IN PORT] → A
- Example: IN 00H

Opcode	operand	Description
OUT	8 Bit Port address	Transfer data from accumulator to I/O Port

- This instruction transfers (copies/ stores) the content of accumulator to the output device.
- A → [OUT PORT]
- Example: OUT 01H
- Address may range from 00H to FF H (Total 256 different output devices can be used)

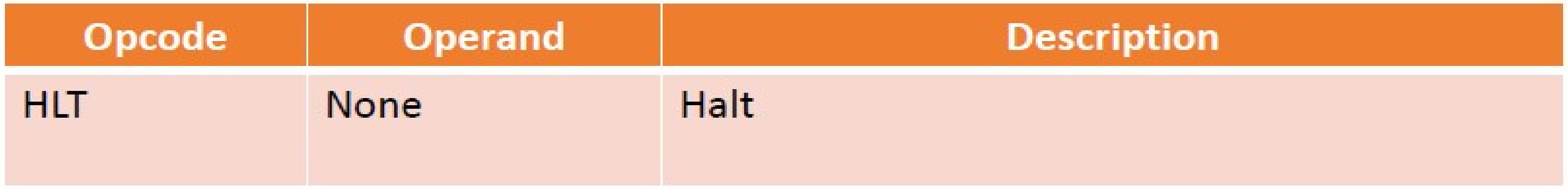
Machine Control Instructions

- The machine control instructions control the operation of microprocessor.



Opcode	Operand	Description
NOP	None	No operation

- No operation is performed.
- The instruction is fetched and decoded but no operation is executed.
- **Example: NOP**



Opcode	Operand	Description
HLT	None	Halt

- 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**

Machine Control Instructions

Opcode	Operand	Description
DI	None	Disable interrupt

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

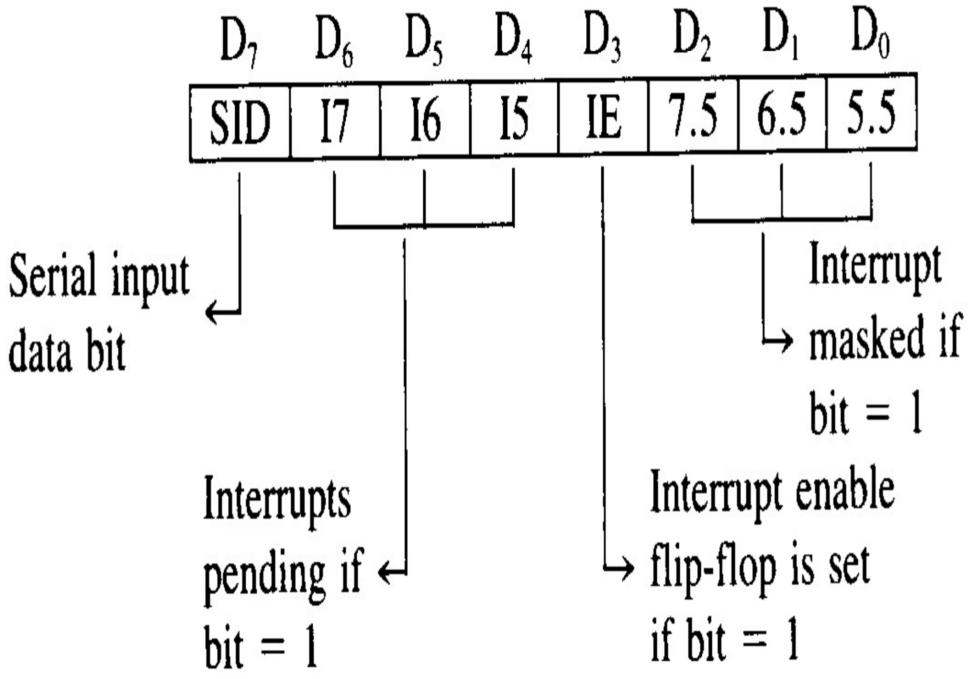
Opcode	Operand	Description
EI	None	Enable interrupt

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

Control Instructions

Opcode	Operand	Description
RIM	None	Read Interrupt Mask

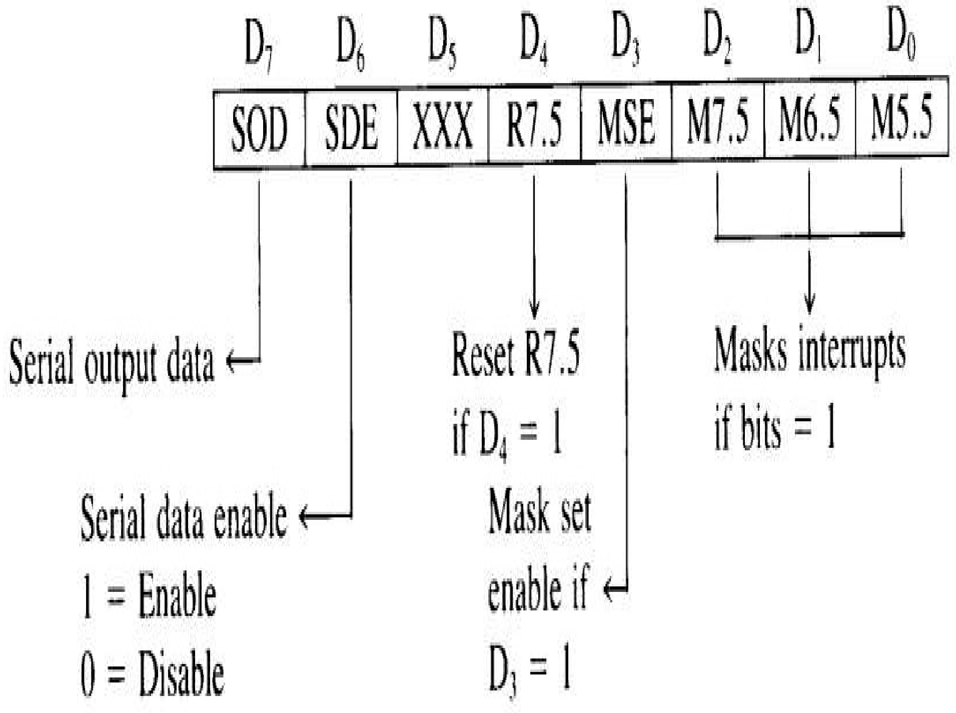
- This is a multipurpose instruction used to read the status of interrupts 7.5, 6.5 5.5.
-
- Read serial data input bit.
- The instruction loads eight bits in the accumulator with the following interpretations
- **Example: RIM**


D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0
SID
I7
I6
I5
IE
7.5
6.5
5.5
Serial input data bit
Interrupts pending if bit = 1
Interrupt enable flip-flop is set if bit = 1
Interrupt masked if bit = 1

Control Instructions

Opcode	Operand	Description
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**



D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0
SOD SDE XXX R7.5 MSE M7.5 M6.5 M5.5
Serial output data
Serial data enable
1 = Enable
0 = Disable
Reset R7.5 if D_4 = 1
Mask set enable if D_3 = 1
Masks interrupts if bits = 1

Stack Instruction

- Stack is a reserved area of the memory where we can store temporary information.
- Programmers use the stack to store data and the MP use the stack to execute subroutines.

- **Stack Pointer** holds the starting address of the stack. This address can be decided by the programmer.

- The stack operates on the **Last In, First Out** (LIFO) principle. The location of the most recent data on the stack is known as the **TOP** of the stack. The stack pointer always points to the top of the stack.

Stack Instruction

- LXI SP, 16-bit - Load the stack pointer register with a 16-bit address
- XTHL - Exchange stack-top with H-L
- SPHL - Move the contents of H-L pair to stack pointer
- PUSH rp - Push the content of register pair to stack
- PUSH PSW - PUSH Processor Status Word
- POP rp - Pop the content of register pair, which was saved, from the stack
- POP PSW - Pop Processor Status Word

Stack Instruction

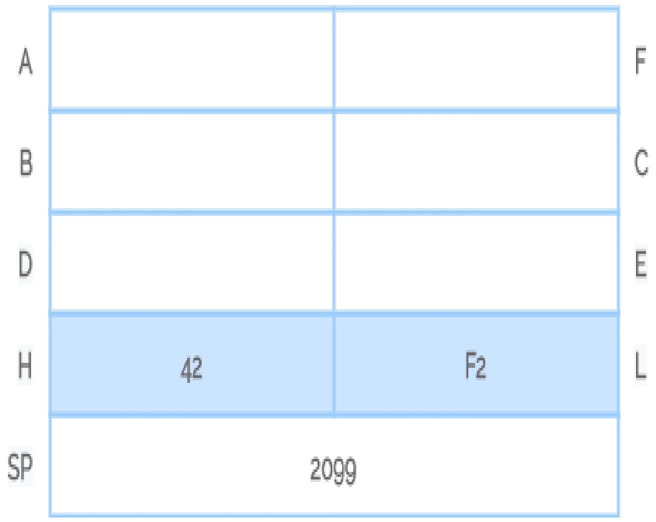
PUSH R_p

- This instruction copies the contents of the specified register pair on the stack as described below:
- The stack pointer is decremented and the contents of the higher-order register are copied to the location shown by the stack pointer register.
- The stack pointer is again decremented and the contents of the low-order register are copied to that location.

Example:

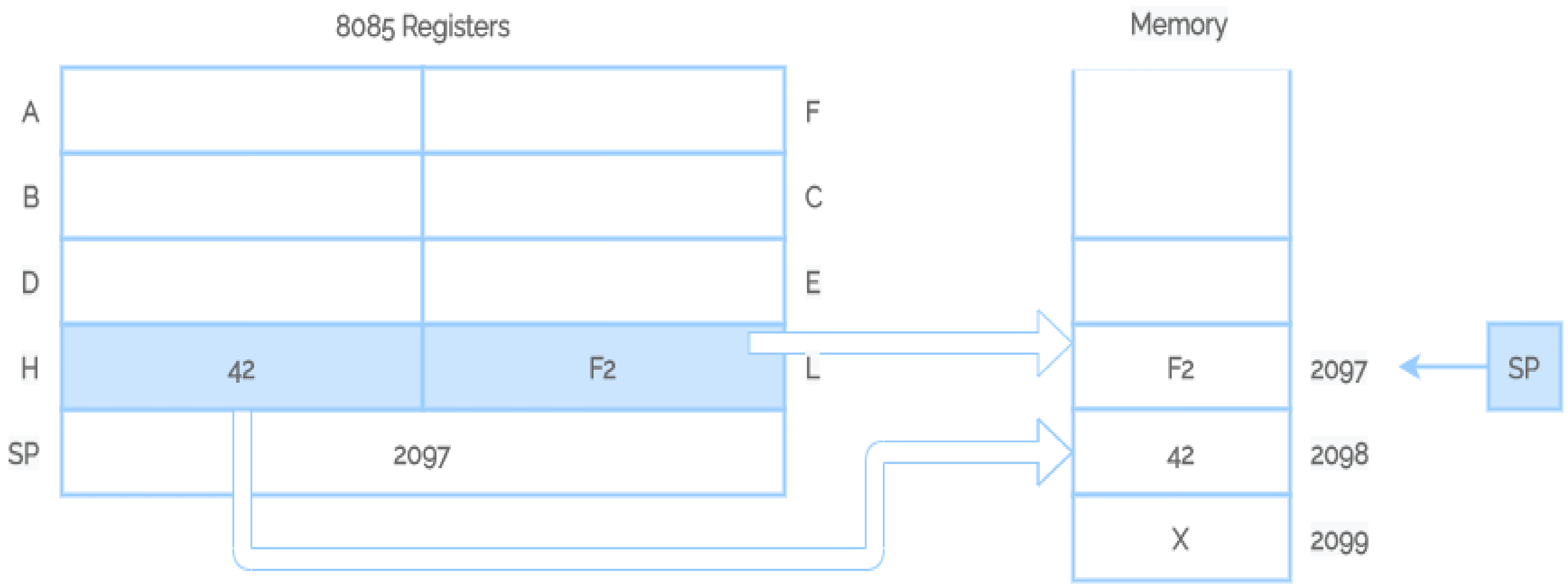
LXI SP,*2099H*
LXI H, *42F2H*
PUSH H

- **LXI** SP, *2099H* - initialize SP with the address of 2099H.
- **LXI** H, *42F2H* - initialize or load HL register pair with *42F2H* data so **H** = 42 and **L** = F2



- After the execution of **PUSH** H instruction the stack pointer is decreased by one to *2098H* and the contents of the H register are copied to memory location *2098H*

- The stack pointer is again decreased by one to *2097H* and the contents of the L register are copied to memory location *2097H*



Contents of Stack and Registers After PUSH Operation

Stack Instruction

POP R_p

- This instruction copies the contents of the top two memory locations of the stack into the specified register pair.
- First, the contents of the memory location indicated by the stack pointer register are copied into the low-order register and then the stack pointer register is incremented by 1.
- The contents of the next memory location are copied into the high-order register and the stack pointer register is again incremented by 1.

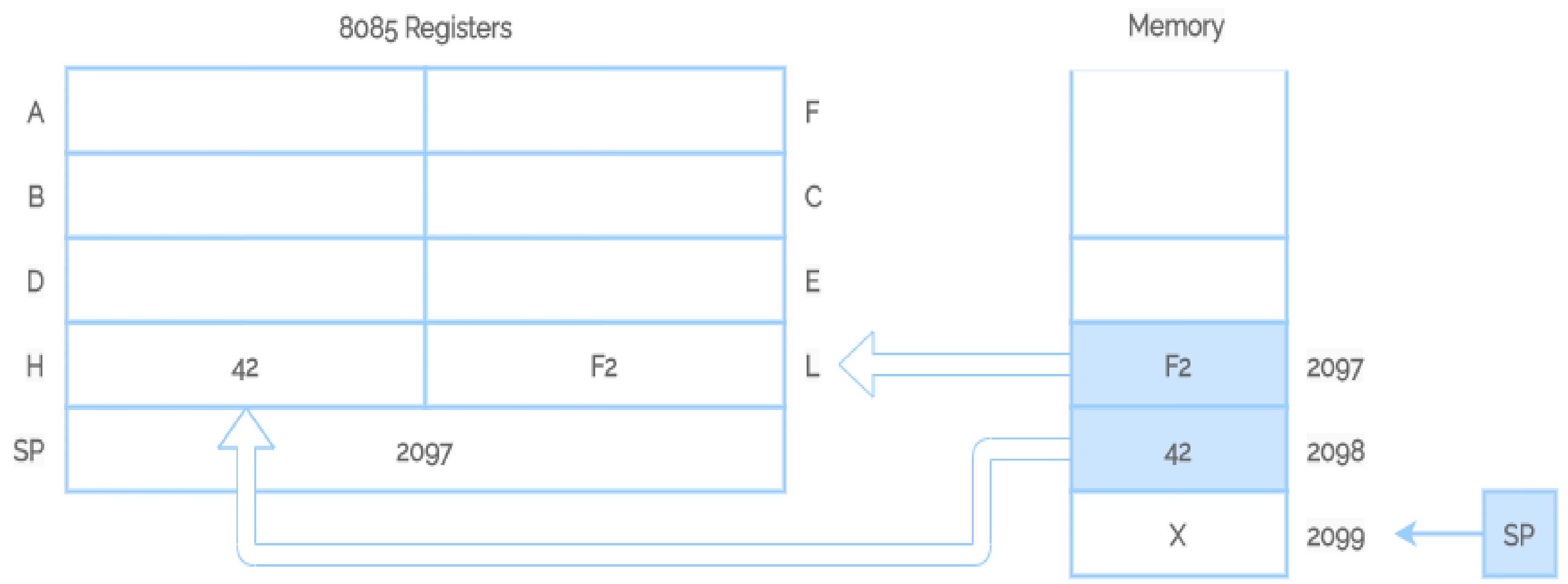
Example:

LXI SP,*2099H*
LXI H, *42F2H*
PUSH H
Delay Counter
POP H

- After the execution of **POP** H instruction, the contents of the top of the stack location shown by the stack pointer are copied in the L register and the stack pointer is increased by one to *2098H*

- The contents of the top of the stack are copied in the H register and the stack pointer is increased by one.

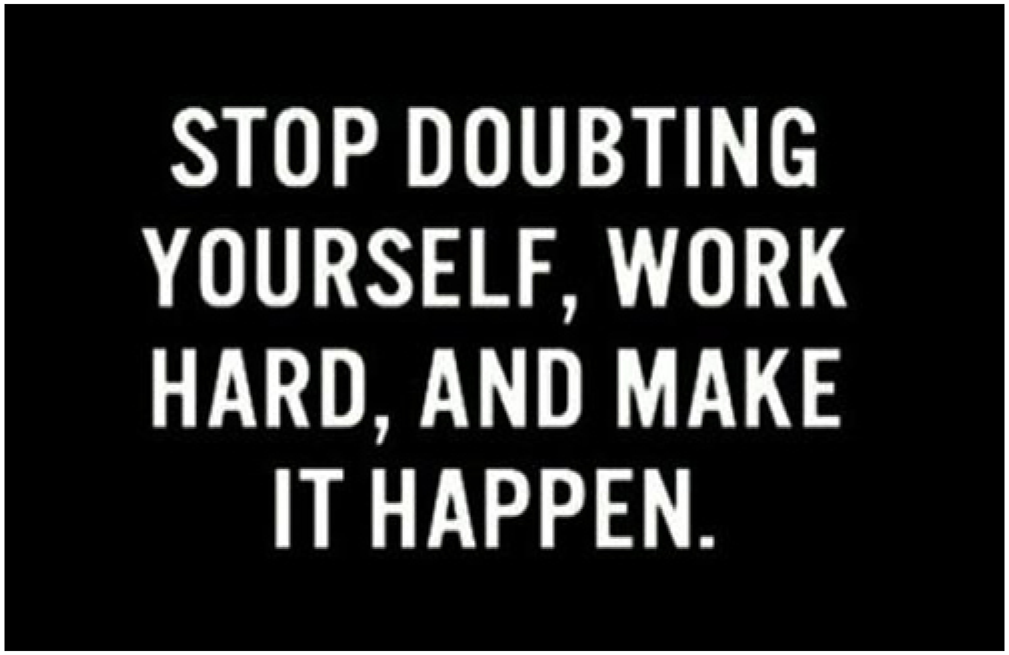
- The contents of the memory locations *2097H* and *2098H* are not destroyed until some other data bytes are stored in these locations



Contents of Stack and Registers After POP Operation

Dream big, stay positive, work hard, and enjoy the journey.

Urijah Faber



STOP DOUBTING YOURSELF, WORK HARD, AND MAKE IT HAPPEN.

Have a Great Day…
GOOD LUCK !!!!!!