INSTRUCTION SET OF 8085

Classification Of Instruction Set

There are 5 categories:

- (1) Data Transfer Instruction,
- (2) Arithmetic Instructions,
- (3) Logical Instructions,
- (4) Branching Instructions,
- (5) Control Instructions,

(1) Data Transfer Instructions

- **MOV Rd, Rs**
- **MOV M, Rs**
- **MOV Rd, M**
- **This instruction copies the contents of the source register into the destination register.**
- **The contents of the source register are not altered.**
- **Example: MOV B,A or MOV M,B or MOV C,M**

BEFORE EXECUTION

A	20	B

MOV B,A

AFTER EXECUTION

A	20	B	20

A		F	
B	30	C	
D		E	
H	20	L	50

MOV M,B

A		F	
B	30	C	
D		E	
H	20	L	50

30

A		F	
B		C	
D		E	
H	20	L	50

40

MOV C,M

A		F	
B		C	40
D		E	
H	20	L	50

40

(2) Data Transfer Instructions

- MVI R, Data(8-bit)
- MVI M, Data(8-bit)

- The 8-bit immediate data is stored in the destination register (R) or memory (M), R is general purpose 8 bit register such as A,B,C,D,E,H and L.

- Example: MVI B, 60H or MVI M, 40H

BEFORE EXECUTION

A	F
B	C
D	E
H	L

MVI B,60H

AFTER EXECUTION

A		F
B	60	C
D		E
H		L

BEFORE EXECUTION

MVI M,40H

AFTER EXECUTION

(3) Data Transfer Instructions

- **LDA 16-bit address**

- **The contents of a memory location, specified by a 16-bit address in the operand, are copied to the accumulator (A).**
- **The contents of the source are not altered.**

- **Example: LDA 2000H**


BEFORE EXECUTION
A
30
2000H
LDA 2000H
AFTER EXECUTION
A
30
30
2000H

(4) Data Transfer Instructions

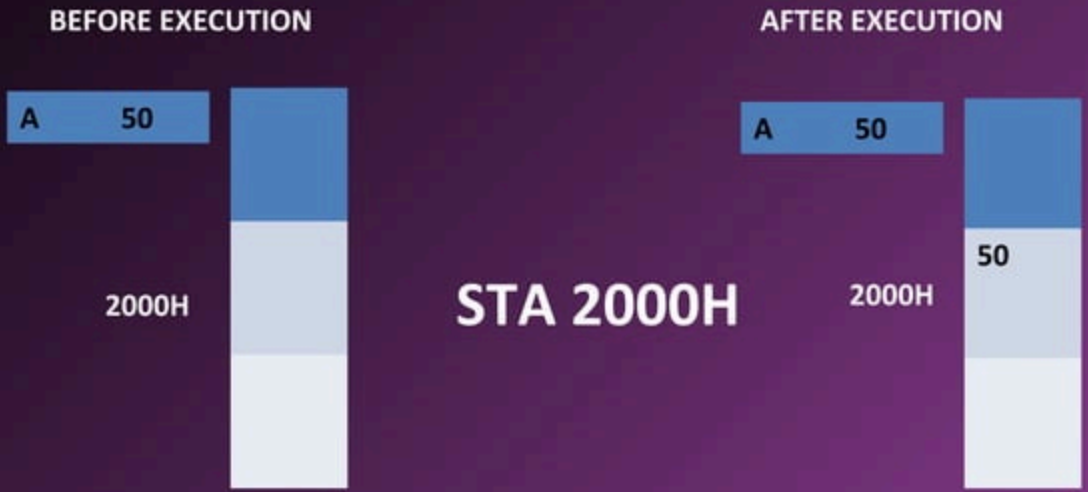
- **LDAX Register Pair**
- Load accumulator (A) with the contents of memory location whose address is specified by BC or DE or register pair.
- The contents of either the register pair or the memory location are not altered.

- **Example: LDAX D**


BEFORE EXECUTION
A F
B C
D 20 E 30
2030H
80
LDAX D
AFTER EXECUTION
A 80 F
B C
D 20 E 30
2030H
80

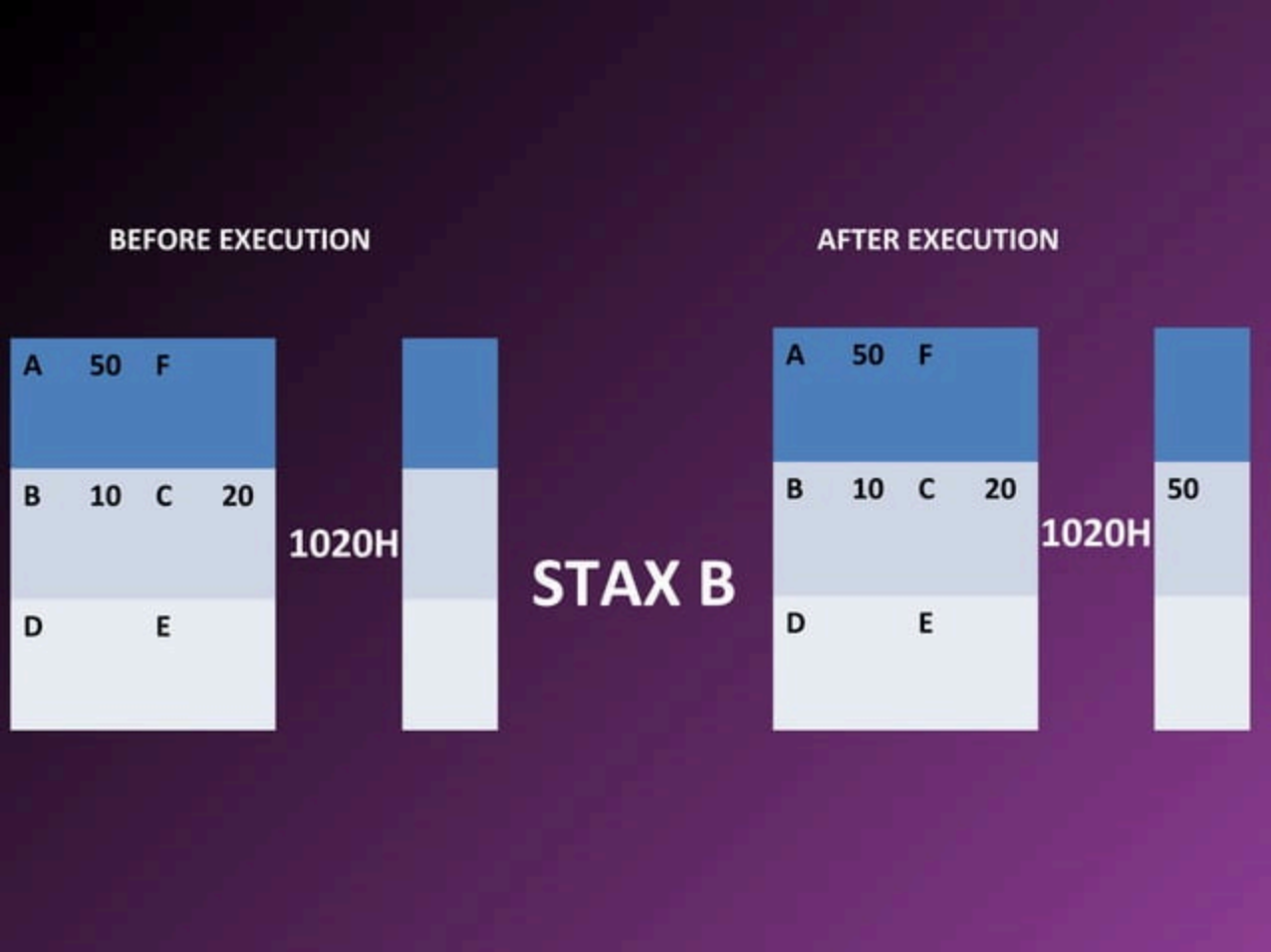
(5) Data Transfer Instructions

- **STA 16-bit address**
- **The contents of accumulator are copied into the memory location i.e. address specified by the operand in the instruction.**
- **Example: STA 2000 H**


BEFORE EXECUTION
A 50
2000H
STA 2000H
AFTER EXECUTION
A 50
50
2000H

(6) Data Transfer Instructions

- **STAX Register Pair**
- **Store the contents of accumulator (A) into the memory location whose address is specified by BC Or DE register pair.**
- **Example: STAX B**


BEFORE EXECUTION
A 50 F
B 10 C 20
D E
1020H
STAX B
AFTER EXECUTION
A 50 F
B 10 C 20
D E
1020H
50

(7) Data Transfer Instructions

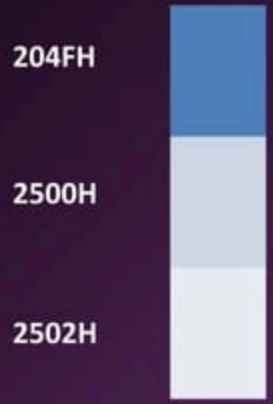
- **SHLD 16-bit address**

- Store H-L register pair in memory.
- The contents of register L are stored into memory location specified by the 16-bit address.
- The contents of register H are stored into the next memory location.

- Example: SHLD 2500 H

BEFORE EXECUTION

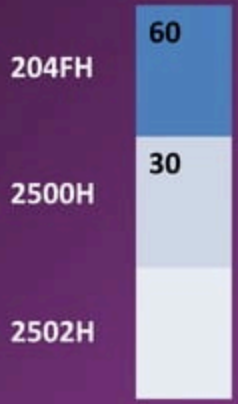
H 30 L 60



SHLD 2500H

AFTER EXECUTION

H 30 L 60



(8) Data Transfer Instructions

- **XCHG**

- **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**

BEFORE EXECUTION

D	20	E	40
H	70	L	80

XCHG

AFTER EXECUTION

D	70	E	80
H	20	L	40

(9) Data Transfer Instructions

- **SPHL**

- **Move data from H-L pair to the Stack Pointer (SP)**
- **This instruction loads the contents of H-L pair into SP.**

- **Example: SPHL**

BEFORE EXECUTION

SP			
H	25	L	00

SPHL

AFTER EXECUTION

SP		2500	
H	25	L	00

(10) Data Transfer Instructions

- **XTHL**
- **Exchange H–L with top of stack**
- **The contents of L register are exchanged with the location pointed out by the contents of the SP.**
- **The contents of H register are exchanged with the next location (SP + 1).**

- **Example: XTHL**

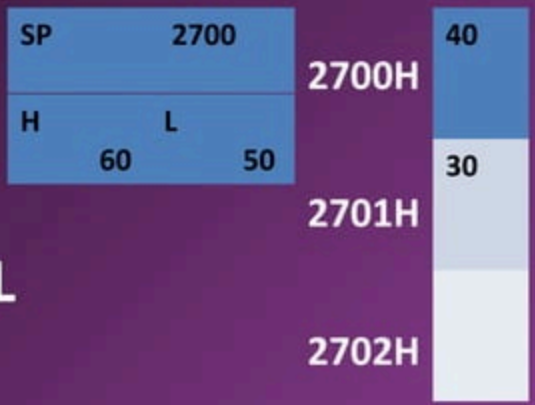
L=SP

H=(SP+1)

BEFORE EXECUTION

XTHL

AFTER EXECUTION



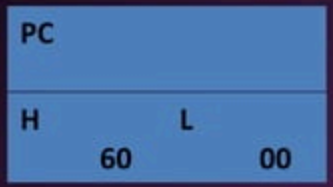
(11) Data Transfer Instructions

- **PCHL**

- **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.

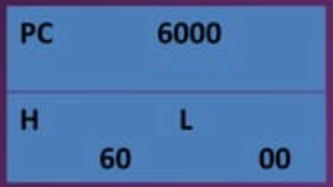
- **Example: PCHL**

BEFORE EXECUTION



PCHL

AFTER EXECUTION



(12) Data Transfer Instructions

- **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 80 H**

BEFORE EXECUTION

IN 80H

AFTER EXECUTION

(13) Data Transfer Instructions

- **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 50 H**


BEFORE EXECUTION
PORT 50H
10
A
40
OUT 50H
AFTER EXECUTION
PORT 50H
40
A
40

Arithematic Instructions

- These instructions perform the operations like:

- Addition
- Subtraction
- Increment
- Decrement

(1) Arithematic Instructions

- **ADD R**
- **ADD M**

- The contents of register 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.

- Example: ADD C or ADD M


BEFORE EXECUTION
A 20
B C 30
D E
H L
ADD C
A=A+R
AFTER EXECUTION
A 50
B C 30
D E
H L
BEFORE EXECUTION
A 20
B C
D E
H 20 L 50
10
2050
ADD M
A=A+M
AFTER EXECUTION
A 30
B C
D E
H 20 L 50
10
2050

(2) Arithematic Instructions

- **ADC R**
- **ADC M**

- The contents of register or memory and Carry Flag (CY) 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 the addition.

- Example: ADC C or ADC M

BEFORE EXECUTION

CY	1

A	50

B		C	20
D		E	
H		L	

ADC C

A=A+R+CY

AFTER EXECUTION

CY	0

A	71

B		C	20
D		E	
H		L	

BEFORE EXECUTION

CY	1

A	20

H	20	L	50

2050H

30

ADC M

A=A+M+CY

AFTER EXECUTION

CY	0

A	51

H	20	L	50

2050H

30

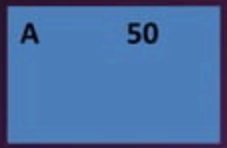
(3) Arithematic Instructions

- **ADI 8-bit data**

- The 8-bit data is added to the contents of accumulator.
- The result is stored in accumulator.

- Example: ADI 10 H

BEFORE EXECUTION



ADI 10H
A=A+DATA(8)

AFTER EXECUTION

(4) Arithematic Instructions

- **ACI 8-bit data**

- The 8-bit data and the Carry Flag (CY) are added to the contents of accumulator.
- The result is stored in accumulator.

- Example: ACI 20 H

BEFORE EXECUTION

CY	1
A	30

ACI 20H
A=A+DATA (8)+CY

AFTER EXECUTION

CY	0
A	51

(5) Arithematic Instructions

- **DAD Register 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.

- Example: DAD D

BEFORE EXECUTION

CY	0

SP			
B		C	
D	10	E	20
H	20	L	50

DAD D

HL=HL+R

AFTER EXECUTION

CY	0

SP			
B		C	
D	10	E	20
H	30	L	70

(6) Arithematic Instructions

- **SUB R**
- **SUB M**

- The contents of the register or memory location 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 H-L pair.

- Example: SUB B or SUB M

BEFORE EXECUTION

A	50	
B	30	C
D		E
H		L

SUB B
A=A-R

AFTER EXECUTION

A	20	
B	30	C
D		E
H		L

BEFORE EXECUTION

A	50	
H		L
10		20

1020H

10

SUB M
A=A-M

AFTER EXECUTION

A	40	
H		L
10		20

1020H

10

(7) Arithematic Instructions

- **SBB R**
- **SBB M**

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

- **Example: SBB C or SBB M**


BEFORE EXECUTION
CY 1
A 40
B C 20
D E
H L
SBB C
A=A-R-CY
AFTER EXECUTION
CY 0
A 19
B C 20
D E
H L
BEFORE EXECUTION
CY 1
A 50
2050H
10
H L
20 50
SBB M
A=A-M-CY
AFTER EXECUTION
CY 0
A 39
2050H
10
H L
20 50

(8) Arithematic Instructions

- SUI 8-bit data

- OPERATION: A=A-DATA(8)
- The 8-bit immediate data is subtracted from the contents of the accumulator.
- The result is stored in accumulator.

- Example: SUI 45 H

(9) Arithematic Instructions

- **SBI 8-bit data**

- The 8-bit data and the Borrow Flag (i.e. CY) is subtracted from the contents of the accumulator.
- The result is stored in accumulator.

- Example: SBI 20 H

BEFORE EXECUTION

CY	1
A	50

SBI 20H
A=A-DATA(8)-CY

AFTER EXECUTION

CY	0
A	29

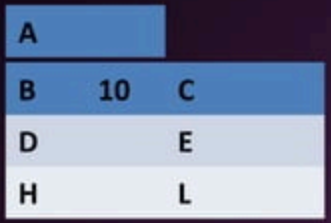
(10) Arithematic Instructions

- **INR R**
- **INR M**

- The contents of register or memory location are incremented by 1.
- The result is stored in the same place.
- If the operand is a memory location, its address is specified by the contents of H-L pair.

- Example: INR B or INR M

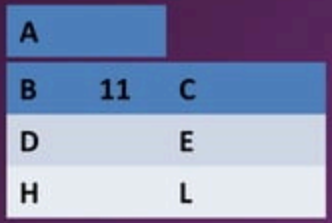
BEFORE EXECUTION



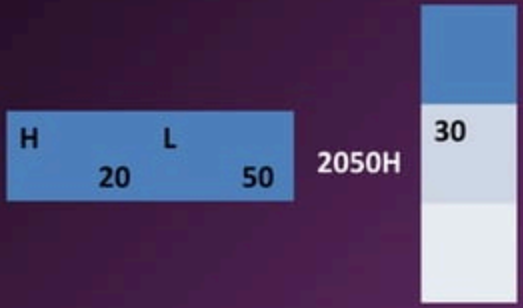
INR B

R=R+1

AFTER EXECUTION



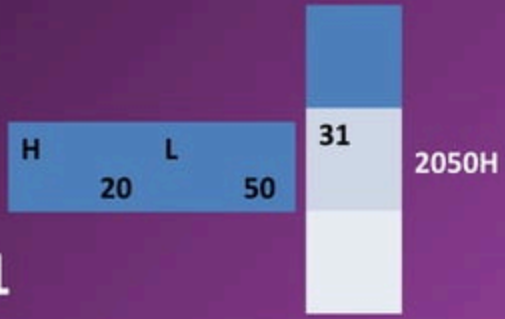
BEFORE EXECUTION



INR M

M=M+1

AFTER EXECUTION



(11) Arithematic Instructions

- **INX Rp**

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

- Example: INX H

BEFORE EXECUTION

SP			
B		C	
D		E	
H	10	L	20

INX H

RP=RP+1

AFTER EXECUTION

SP			
B		C	
D		E	
H	11	L	21

(12) Arithematic Instructions

- **DCR R**
- **DCR M**

- 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 the contents of H-L pair.
- Example: DCR E or DCR M

BEFORE EXECUTION

A		
B	C	
D	E	20
H	L	

DCR E

R=R-1

AFTER EXECUTION

A		
B	C	
D	E	19
H	L	

BEFORE EXECUTION

H	L
20	50

2050H

21

DCR M

M=M-1

AFTER EXECUTION

H	L
20	50

2050H

20

(13) Arithematic Instructions

- **DCX Rp**

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

- Example: DCX D

BEFORE EXECUTION

SP			
B		C	
D	10	E	20
H		L	

DCX D
RP=RP-1

AFTER EXECUTION

SP			
B		C	
D	10	E	19
H		L	

(1) Logical Instructions

- **ANA R**
- **ANA M**

- **AND specified data in register or memory with accumulator.**
- **Store the result in accumulator (A).**

- Example: **ANA B, ANA M**

BEFORE EXECUTION

CY		AC	
A	AA		
B	0F	C	
D		E	
H		L	

1010 1010=AAH
0000 1111=0FH

0000 1010=0AH

ANA B
A=A and R

AFTER EXECUTION

CY	0	AC	1
A	0A		
B	0F	C	
D		E	
H		L	

BEFORE EXECUTION

CY		AC	
A	55	2050H	
H	20	L	50

B3

0101 0101=55H
1011 0011=B3H

0001 0001=11H

ANA M
A=A and M

AFTER EXECUTION

CY	0	AC	1
A	11	2050H	
H	20	L	50

B3

(2) Logical Instructions

- **ANI 8-bit data**

- **AND 8-bit data with accumulator (A).**
- **Store the result in accumulator (A)**

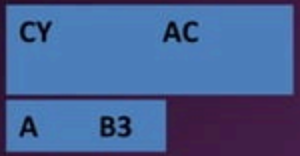
- Example: **ANI 3FH**

BEFORE EXECUTION

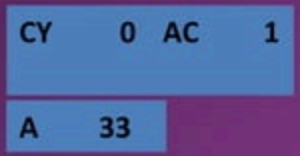
AFTER EXECUTION

1011 0011=B3H
0011 1111=3FH

0011 0011=33H



ANI 3FH
A=A and DATA(8)



(3) Logical Instructions

- **XRA Register (8-bit)**

- **XOR specified register with accumulator.**
- **Store the result in accumulator.**

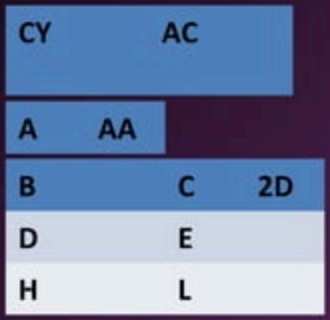
- Example: **XRA C**

BEFORE EXECUTION

1010 1010=AAH
0010 1101=2DH

1000 0111=87H

AFTER EXECUTION



XRA C

A=A xor R

(4) Logical Instructions

- **XRA M**

- XOR data in memory (memory location pointed by H-L pair) with Accumulator.
- Store the result in Accumulator.

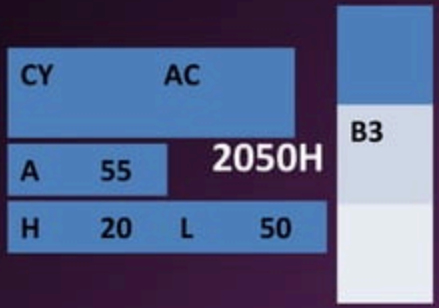
- Example: XRA M

BEFORE EXECUTION

0101 0101=55H
1011 0011=B3H

1110 0110=E6H

AFTER EXECUTION



XRA M
A=A xor M

(5) Logical Instructions

- **XRI 8-bit data**

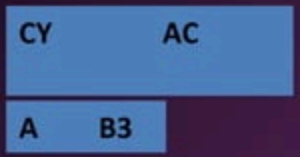
- XOR 8-bit immediate data with accumulator (A).
- Store the result in accumulator.

- Example: XRI 39H

1011 0011=B3H
0011 1001=39H

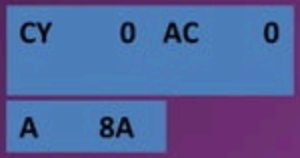
1000 1010=8AH

BEFORE EXECUTION



XRI 39H
A=A xor DATA(8)

AFTER EXECUTION



(6) Logical Instructions

- **ORA Register**

- OR specified register with accumulator (A).
- Store the result in accumulator.

- Example: ORA B

1010 1010=AAH
0001 0010=12H

1011 1010=BAH

BEFORE EXECUTION

CY		AC	

A	AA	
B	12	C
D		E
H		L

ORA B

A=A or R

AFTER EXECUTION

CY	0	AC	0

A	BA	
B	12	C
D		E
H		L

(7) Logical Instructions

- **ORA M**

- OR specified register with accumulator (A).
- Store the result in accumulator.

- Example: ORA M

0101 0101=55H
1011 0011=B3H

1111 0111=F7H

BEFORE EXECUTION

AFTER EXECUTION

ORA M

A=A or M

(8) Logical Instructions

- **ORI 8-bit data**

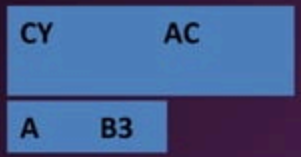
- OR 8-bit data with accumulator (A).
- Store the result in accumulator.

- Example: ORI 08H

1011 0011=B3H
0000 1000=08H

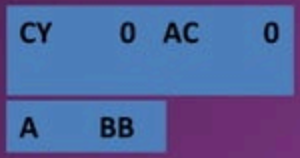
1011 1011=BBH

BEFORE EXECUTION



ORI 08H
A=A or DATA(8)

AFTER EXECUTION



(9) Logical Instructions

- **CMP Register**
- **CMP M**

- Compare specified data in register or memory with accumulator (A).
- Store the result in accumulator.

- Example: CMP D or CMP M

BEFORE EXECUTION

CY		Z	
A	B8		
B		C	
D	B9	E	
H		L	

A>R: CY=0,Z=0
A=R: CY=0,Z=1
A<R: CY=1,Z=0

CMP D
A-R

AFTER EXECUTION

CY	0	Z	0
A	B8		
B		C	
D	B9	E	
H		L	

BEFORE EXECUTION

CY		Z		
A	B8		2050H	B8
H	20	L	50	

A>M: CY=0,Z=0
A=M: CY=0,Z=1
A<M: CY=1,Z=0

CMP M
A-M

AFTER EXECUTION

CY	0	Z	1	
A	B8		2050H	B8
H	20	L	50	

(10) Logical Instructions

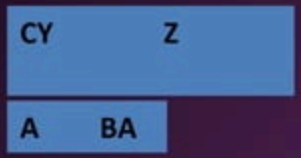
- **CPI 8-bit data**

- Compare 8-bit immediate data with accumulator (A).
- Store the result in accumulator.

- Example: CPI 30H

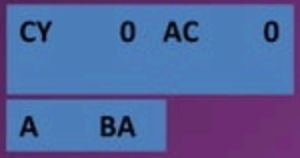
A>DATA: CY=0,Z=0
A=DATA: CY=0,Z=1
A<DATA: CY=1,Z=0

BEFORE EXECUTION



CPI 30H
A-DATA

AFTER EXECUTION



1011 1010=BAH

(11) Logical Instructions

- **STC**

- It sets the carry flag to 1.
- Example: STC

BEFORE EXECUTION

AFTER EXECUTION

CY	0

STC

CY=1

CY	1

(12) Logical Instructions

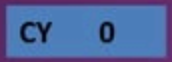
- **CMC**

- It complements the carry flag.
- Example: CMC

BEFORE EXECUTION

AFTER EXECUTION

CMC



(13) Logical Instructions

- **CMA**

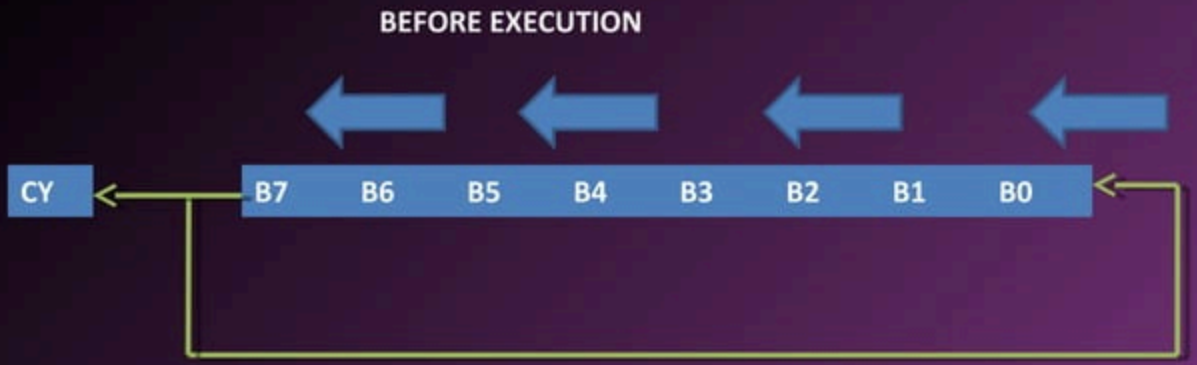
- It complements each bit of the accumulator.
- Example: CMA

(14) Logical Instructions

- **RLC**

- 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.

- Example: RLC.


BEFORE EXECUTION
CY
B7
B6
B5
B4
B3
B2
B1
B0


AFTER EXECUTION
B7
B6
B5
B4
B3
B2
B1
B0
B7

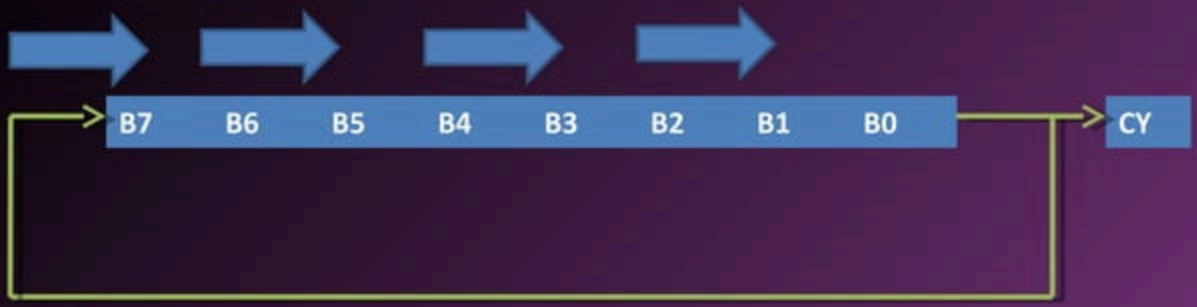
(15) Logical Instructions

- **RRC**

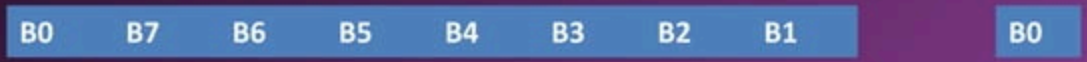
- Rotate accumulator right
- Each binary bit of the accumulator is rotated right by one
- position.
- Bit D0 is placed in the position of D7 as well as in the Carry flag.
- CY is modified according to bit D0.

- Example: RRC.

BEFORE EXECUTION



AFTER EXECUTION

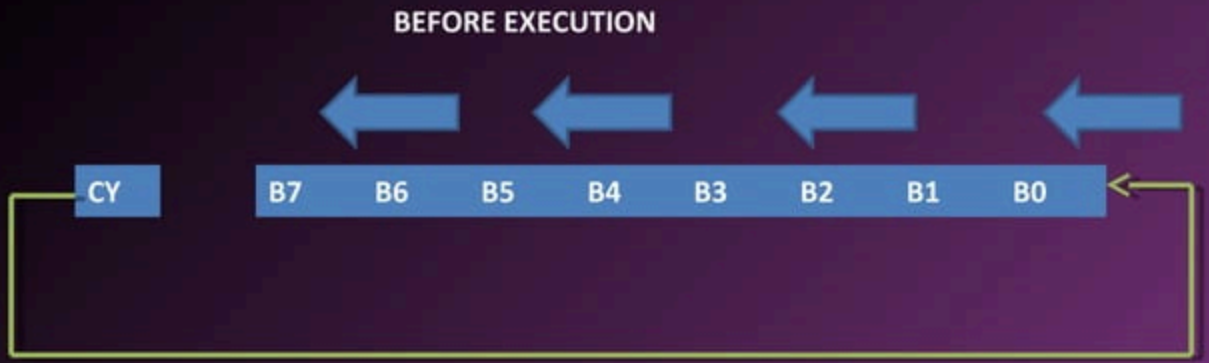


(16) Logical Instructions

- **RAL**

- 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.

- Example: RAL.


BEFORE EXECUTION
CY
B7
B6
B5
B4
B3
B2
B1
B0


AFTER EXECUTION
B7
B6
B5
B4
B3
B2
B1
B0
CY

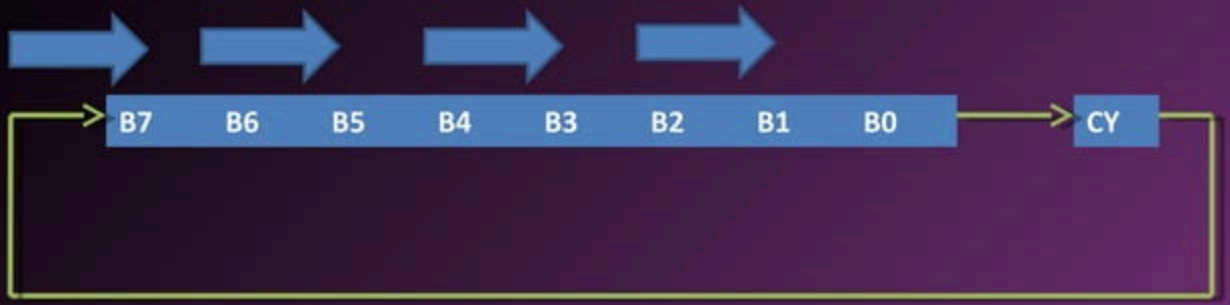
(17) Logical Instructions

- **RAR**

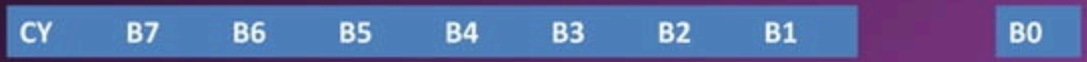
- Rotate accumulator right through carry
- Each binary bit of the accumulator is rotated left by one position through the Carry flag.
- Bit D7 is placed in the Carry flag, and the Carry flag is placed in the least significant position D0.
- CY is modified according to bit D7.

- Example: RAR

BEFORE EXECUTION



AFTER EXECUTION



Branching Instructions

- The branch group instructions allows the microprocessor to change the sequence of program either conditionally or under certain test conditions. The group includes,
- (1) Jump instructions,
- (2) Call and Return instructions,
- (3) Restart instructions,

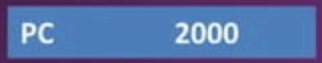
(1) Branching Instructions

- **JUMP ADDRESS**

- BEFORE EXECUTION

JMP 2000H

AFTER EXECUTION



- Jump unconditionally to the address.
- The instruction loads the PC with the address given within the instruction and resumes the program execution from specified location.
- Example: JMP 200H

Conditional Jumps

Instruction Code	Decription	Condition For Jump
JC	Jump on carry	CY=1
JNC	Jump on not carry	CY=0
JP	Jump on positive	S=0
JM	Jump on minus	S=1
JPE	Jump on parity even	P=1
JPO	Jump on parity odd	P=0
JZ	Jump on zero	Z=1
JNZ	Jump on not zero	Z=0

(2) Branching Instructions

- **CALL address**

- Call unconditionally a subroutine whose starting address given within the instruction and used to transfer program control to a subprogram or subroutine.

- Example: CALL 2000H

Conditional Calls

Instruction Code	Description	Condition for CALL
CC	Call on carry	CY=1
CNC	Call on not carry	CY=0
CP	Call on positive	S=0
CM	Call on minus	S=1
CPE	Call on parity even	P=1
CPO	Call on parity odd	P=0
CZ	Call on zero	Z=1
CNZ	Call on not zero	Z=0

(3) Branching Instructions

- **RET**

- Return from the subroutine unconditionally.
- This instruction takes return address from the stack and loads the program counter with this address.

- Example: RET

BEFORE EXECUTION

SP	27FD
PC	

27FDH	00
27FEH	62
27FFH	

RET

AFTER EXECUTION

SP	27FF
PC	6200

27FDH	00
27FEH	62
27FFH	

(4) Branching Instructions

- **RST n**

- Restart n (0 to 7)
- This instruction transfers the program control to a specific memory address. The processor multiplies the RST number by 8 to calculate the vector address.

- Example: RST 6

BEFORE EXECUTION

AFTER EXECUTION

SP-1

SP	3000
PC	2000

RST 6

SP	2999
PC	0030

ADDRESS OF THE NEXT INSTRUCTION IS 2001H

Vector Address For Return Instructions

Instruction Code	Vector Address
RST 0	0*8=0000H
RST 1	0*8=0008H
RST 2	0*8=0010H
RST 3	0*8=0018H
RST 4	0*8=0020H
RST 5	0*8=0028H
RST 6	0*8=0030H
Rst 7	0*8=0038H

(1) Control Instructions

- **NOP**

- No operation
- No operation is performed.
- The instruction is fetched and decoded but no operation is executed.

- Example: NOP

(2) Control Instructions

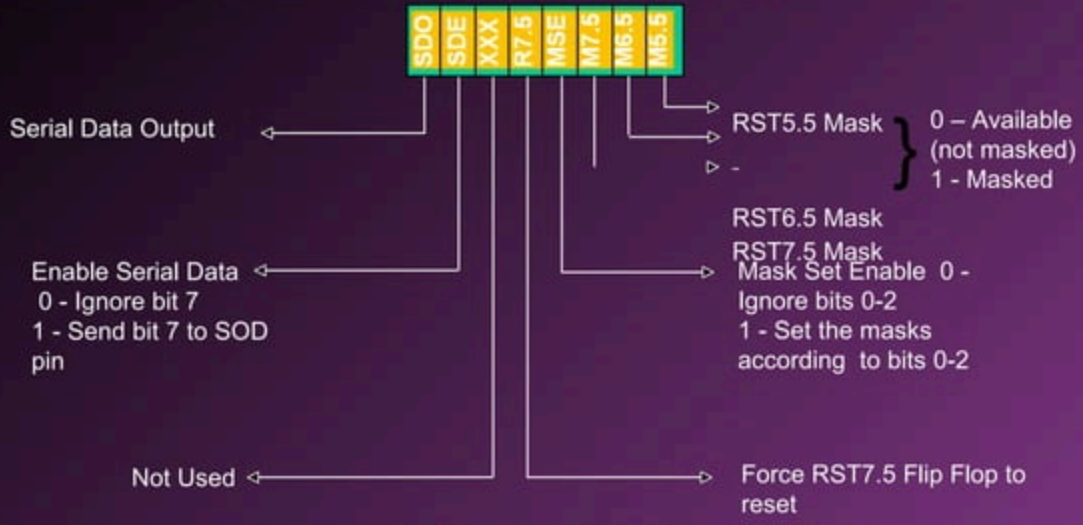
- **HLT**

- 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

(3) Control Instructions

SIM instruction can be used to perform two different tasks: 1. For masking of 3 interrupts 2. For serial data transmission (Each time a SIM instruction is executed, 7th bit of Accumulator is automatically copied to SOD pin of 8085)



While EI/DI instructions enable/disable all maskable interrupts at once, SIM instruction can be used to selectively mask (or disable) 3 out of 4 maskable interrupts which are RST7.5,RST6.5 & RST5.5. Fourth maskable interrupt INTR can only be enabled/disabled by using EI/DI instructions.

Example of how to use SIM instruction in any program

Example problem:- Set the interrupt masks so that RST5.5 is enabled, RST6.5 is masked & RST7.5 is enabled.

- We can determine the bit pattern as per format of SIM instruction given below:

- Enable 5.5	bit 0 = 0
- Disable 6.5	bit 1 = 1
- Enable 7.5	bit 2 = 0
- Allow setting the masks	bit 3 = 1
	bit 4 = 0
- Don't reset the flip flop	bit 5 = 0
- Bit 5 is not used	bit 6 = 0
- Don't use serial data	bit 7 = 0
- Serial data is ignored	

Contents of accumulator are: 0AH

- Now use following set of instructions to implement required masks using SIM

```
EI
MVI A,
0A  SIM
```

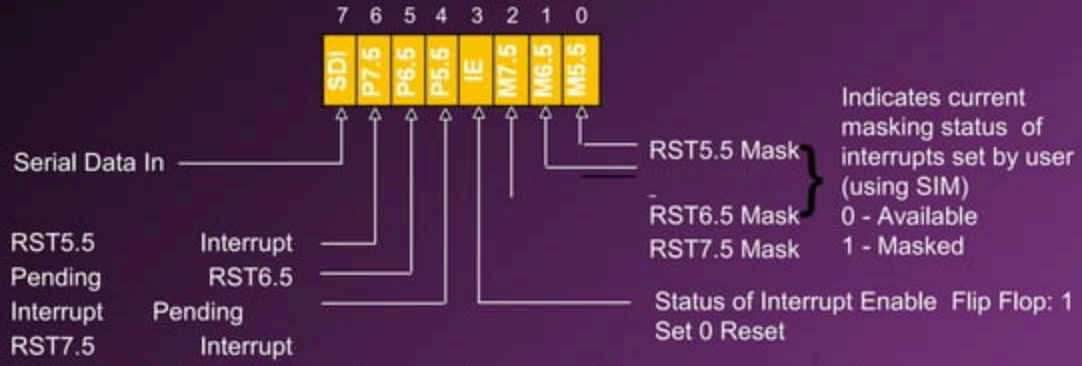
✓ First of all enable all interrupts using EI instruction without using which SIM wouldn't be effective

✓ Move the prepared bit pattern (0AH here) to Accumulator

✓ SIM instruction interprets contents of Accumulator same as per the above format & performs the desired operation of masking the respective interrupts

(4) Control Instructions

Like SIM instruction, RIM can be used to perform two different tasks: 1. To read current status of 3 maskable interrupts 2. For serial data reception (Each time a SIM instruction is executed, the bit present on SID pin of 8085 is automatically moved to 7th bit of the Accumulator)



Pending Interrupts: Since the 8085 has 5 interrupt lines, another interrupts may occur while an interrupt is being attended and thus remain pending. Such interrupts are called pending interrupts & would be attended as soon as ISR of current interrupt is executed. A programmer may know the status (current value of high/low on the respective interrupt pin) of such interrupts anytime by using RIM instruction.