INSTRUCTION SET OF 8085

Instruction Set of 8085

- An instruction is a binary pattern designed inside a microprocessor to perform a **specific function.**
- The entire group of instructions that a microprocessor supports is called ***Instruction Set***.
- 8085 has **246** instructions.
- Each instruction is represented by an 8-bit binary value.
- These 8-bits of binary value is called ***Op-Code*** or ***Instruction Byte***.

Classification of Instruction Set

- Data Transfer Instruction
- Arithmetic Instructions
- Logical Instructions
- Branching Instructions
- Control Instructions

1.Data Transfer Instructions

- These instructions move data between registers, or between memory and registers.
- These instructions copy data from source to destination(without changing the original data).

MOV-Copy from source to destination

Opcode	Operand
MOV	**Rd, Rs** **M, Rs** **Rd, M**

- This instruction copies the contents of the source register into the destination register. (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.
- **Example:** MOV B, C or MOV B, 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

MVI-Move immediate 8-bit

Opcode	Operand
MVI	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 H-L registers.
- **Example:** MVI B, 60_H or MVI M, 40_H

BEFORE EXECUTION

A		F
B		C
D		E
H		L

MVI B, 60_H

AFTER EXECUTION

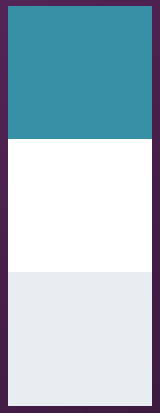
A		F
B	60	C
D		E
H		L

BEFORE EXECUTION

204FH

HL=2050

2051H



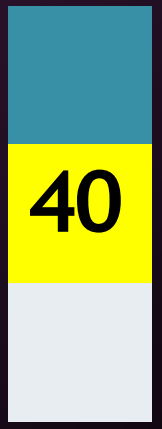
MVI M, 40_H

AFTER EXECUTION

204F

HL=2050

2051H



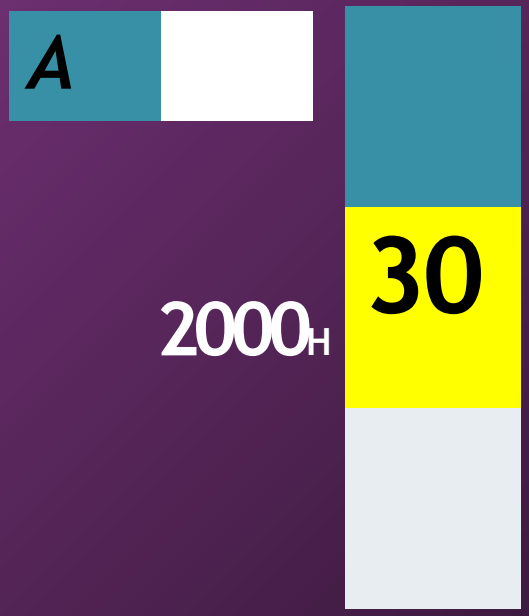
LDA-Load accumulator

Opcode	Operand
LDA	16-bit address

- The contents of a memory location, specified by a 16-bit address in the operand, are copied to the accumulator.
- The contents of the source are not altered.
- **Example:** LDA 2000H

LDA 2000_H

BEFORE EXECUTION



AFTER EXECUTION

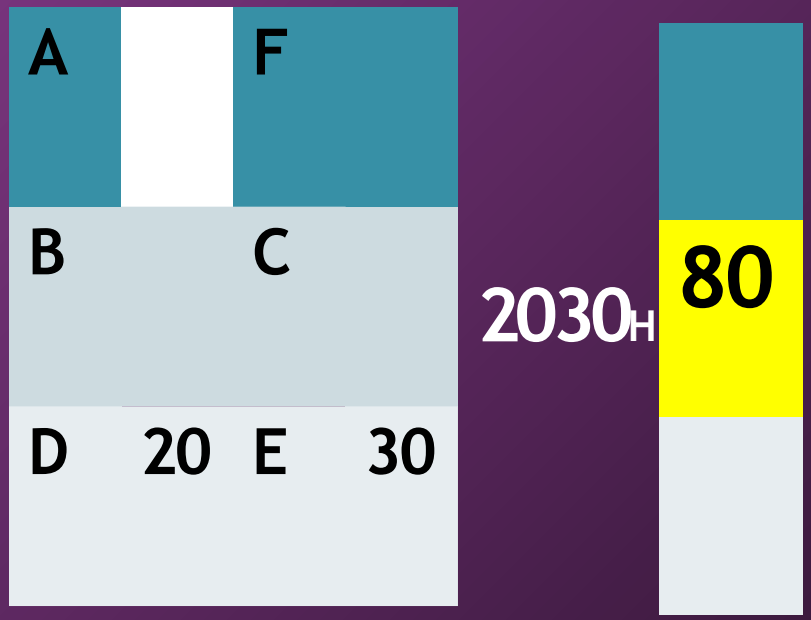
LDAX-Load accumulator indirect

Opcode	Operand
LDAX	B/D Register Pair

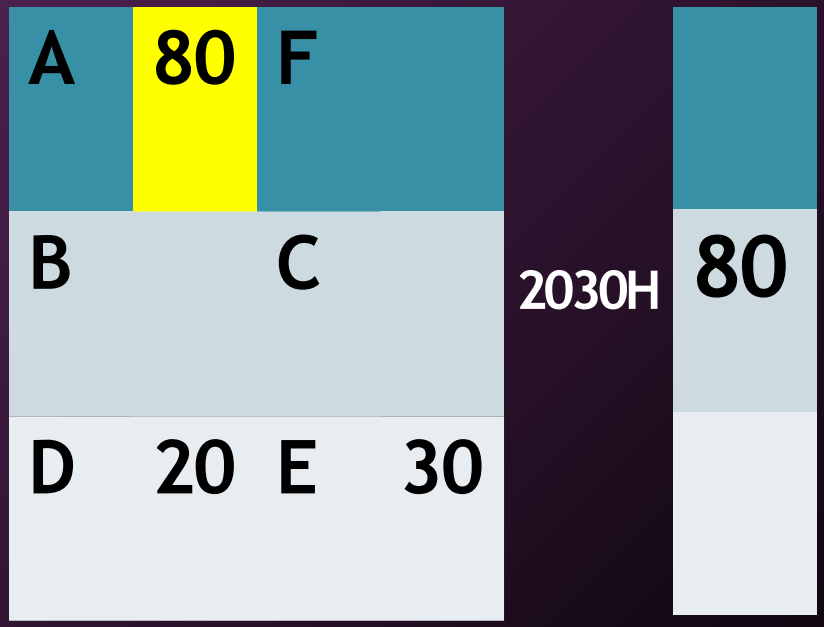
- The contents of the designated register pair point to a memory location.
- This instruction copies the contents of that memory location into the accumulator.
- The contents of either the register pair or the memory location are not altered.
- **Example:** LDAX D

LDAX D

BEFORE EXECUTION



AFTER EXECUTION



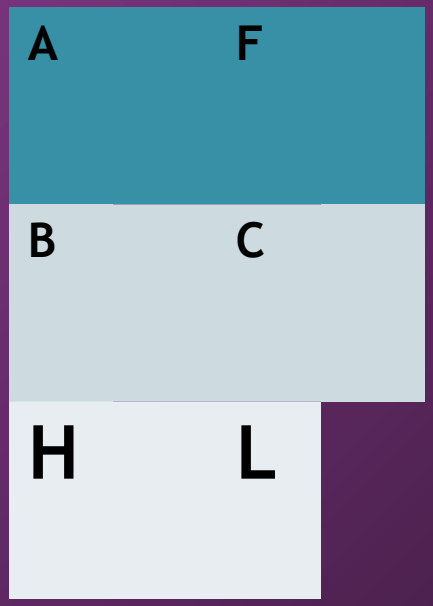
LXI-Load register pair immediate

Opcode	Operand
LXI	Reg. pair, 16-bit data

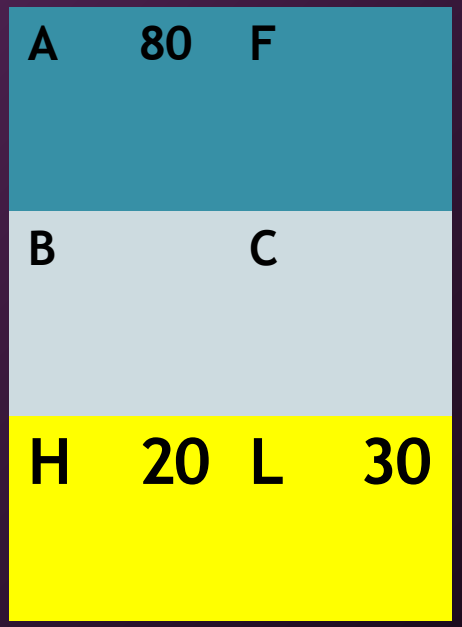
- This instruction loads 16-bit data in the register pair.
- **Example:** LXI H, 2030 H

LXI H, 2030

BEFORE EXECUTION



AFTER EXECUTION



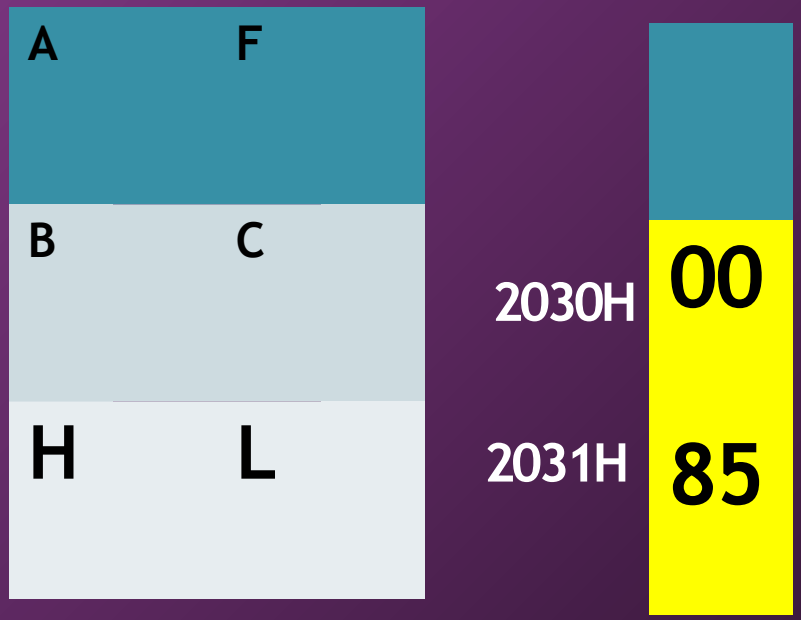
LHLD-Load H and L registers direct

Opcode	Operand
LHLD	16-bit address

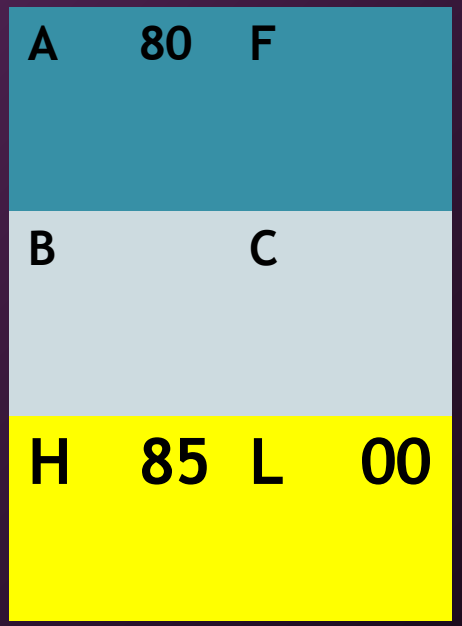
- This instruction copies the contents of memory location pointed out by 16-bit address into register L.
- It copies the contents of next memory location into register H.
- **Example:** LHLD 2030 H

LHLD 2030

BEFORE EXECUTION



AFTER EXECUTION

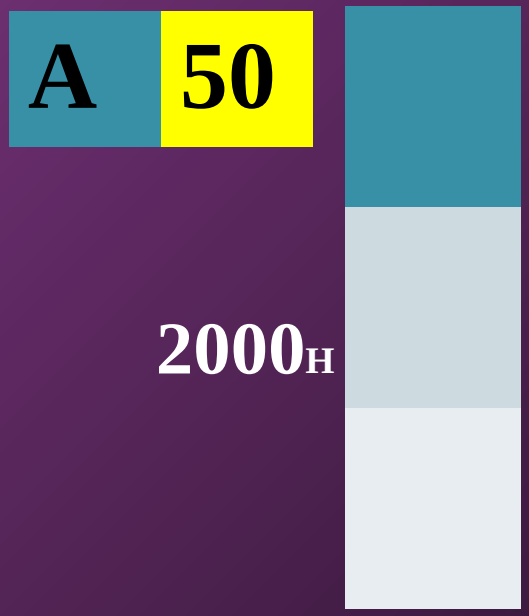


STA-Store accumulator direct

Opcode	Operand
STA	16-bit address

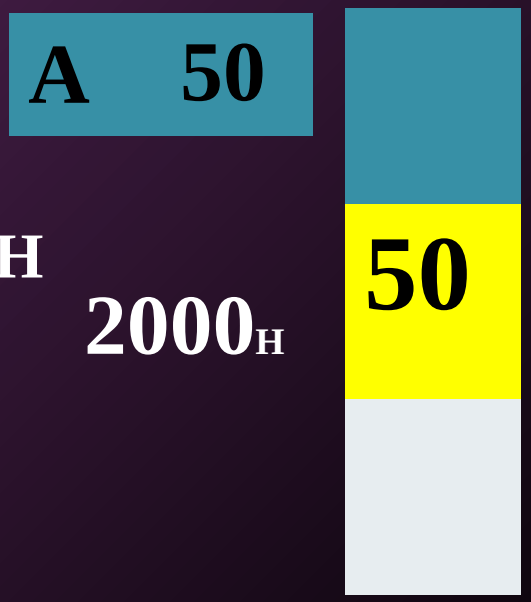
- The contents of accumulator are copied into the memory location specified by the operand.
- **Example:** STA 2000_H

BEFORE EXECUTION



STA 2000H

AFTER EXECUTION



STAX-Store accumulator indirect

Opcode	Operand
STAX	Reg. pair

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

BEFORE EXECUTION

B	85 C	00
A=1A$_H$		

STAX B

8500$_H$

AFTER EXECUTION

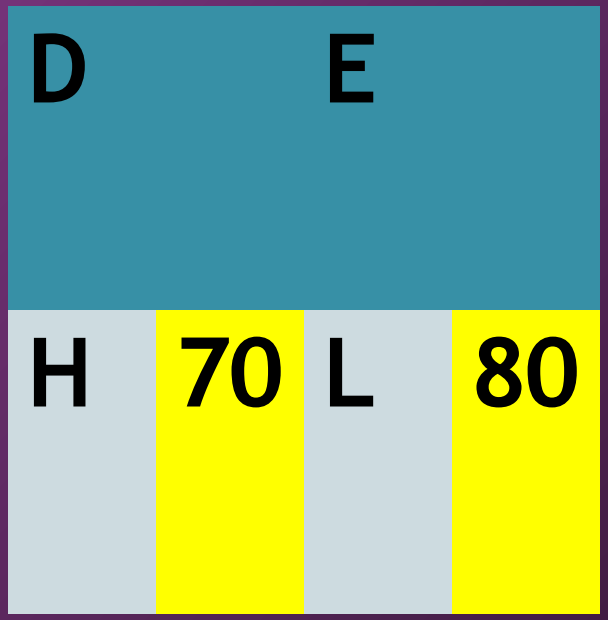
1A

SHLD-Store H and L registers direct

Opcode	Operand
SHLD	**16-bit address**

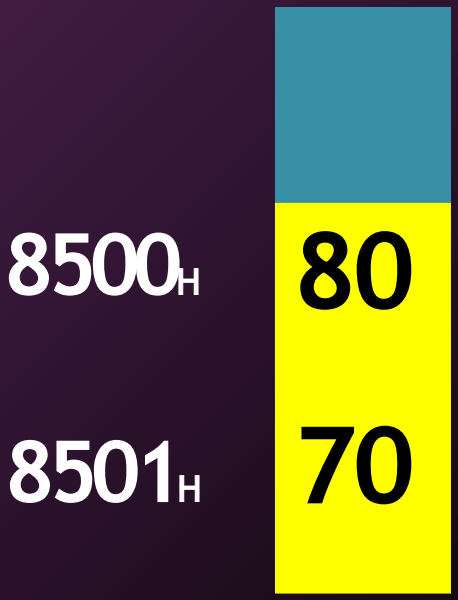
- The contents of register L are stored into memory location specified by the 16-bit address.
- The contents of register H are stored into the next memory location.
- **Example:** SHLD 2550_H

BEFORE EXECUTION



SHLD 8500

AFTER EXECUTION



XCHG-Exchange H and L with D and E

Opcode	Operand
XCHG	None

- 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

SPHL-Copy H and L registers to the stack pointer

Opcode	Operand
SPHL	None

- 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

XTHL-Exchange H and L with top of stack

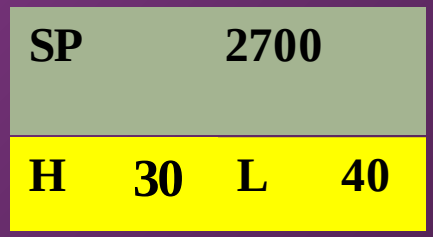
Opcode	Operand
XTHL	None

- 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

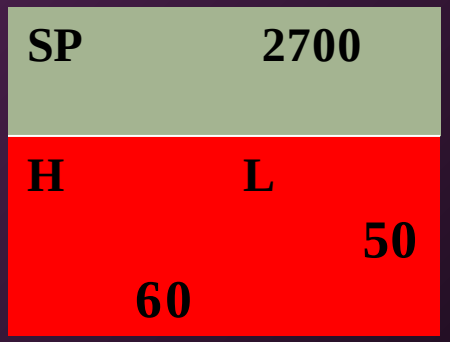
H=(SP+1)

BEFORE EXECUTION



XTHL

AFTER EXECUTION



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

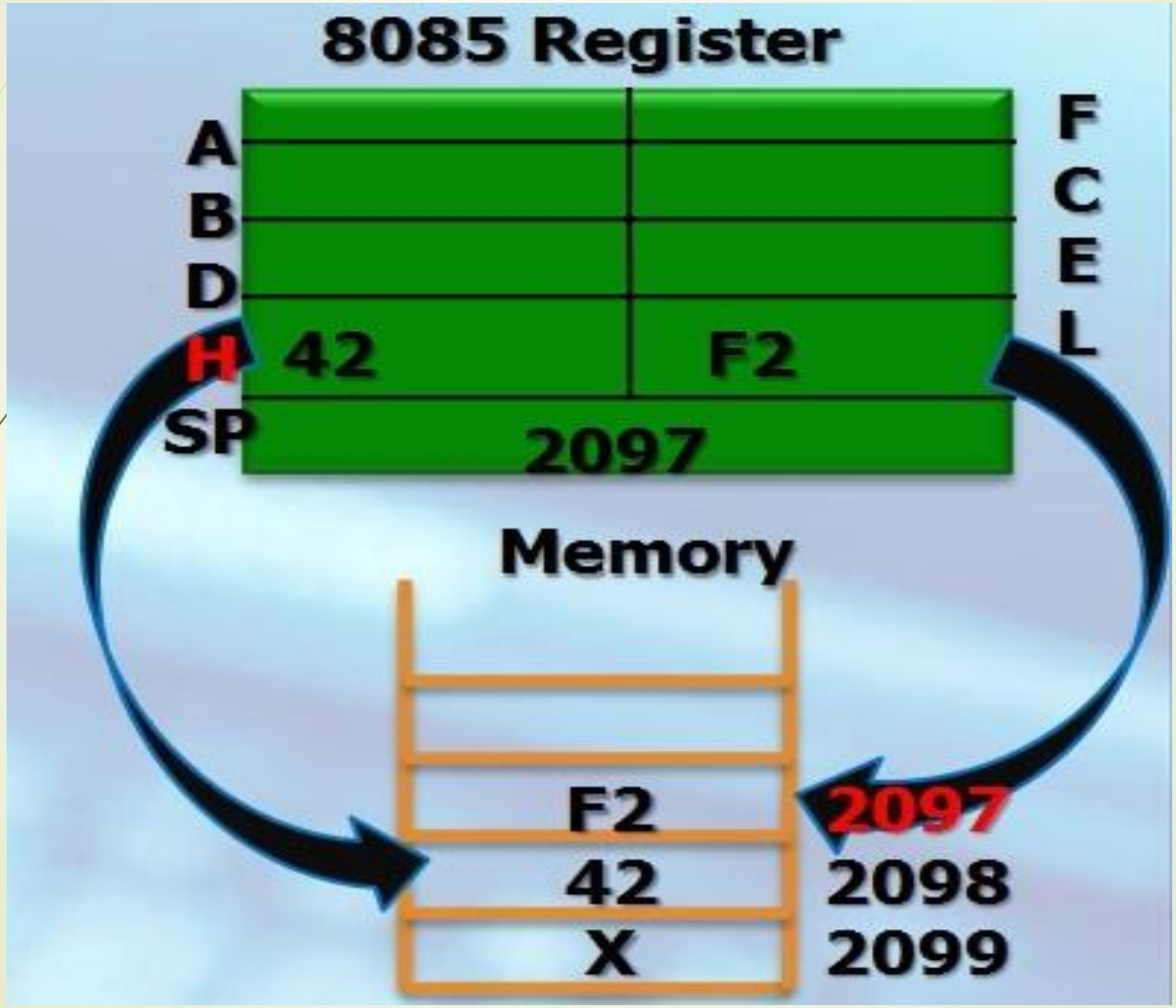
- The contents of registers H and L are copied into the program counter (PC).
- The contents of H are placed as the high-order byte and the contents of L as the low-order byte.
- **Example:** PCHL

PUSH-Push register pair onto stack

Opcode	Operand
PUSH	Reg. pair

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

PUSH H

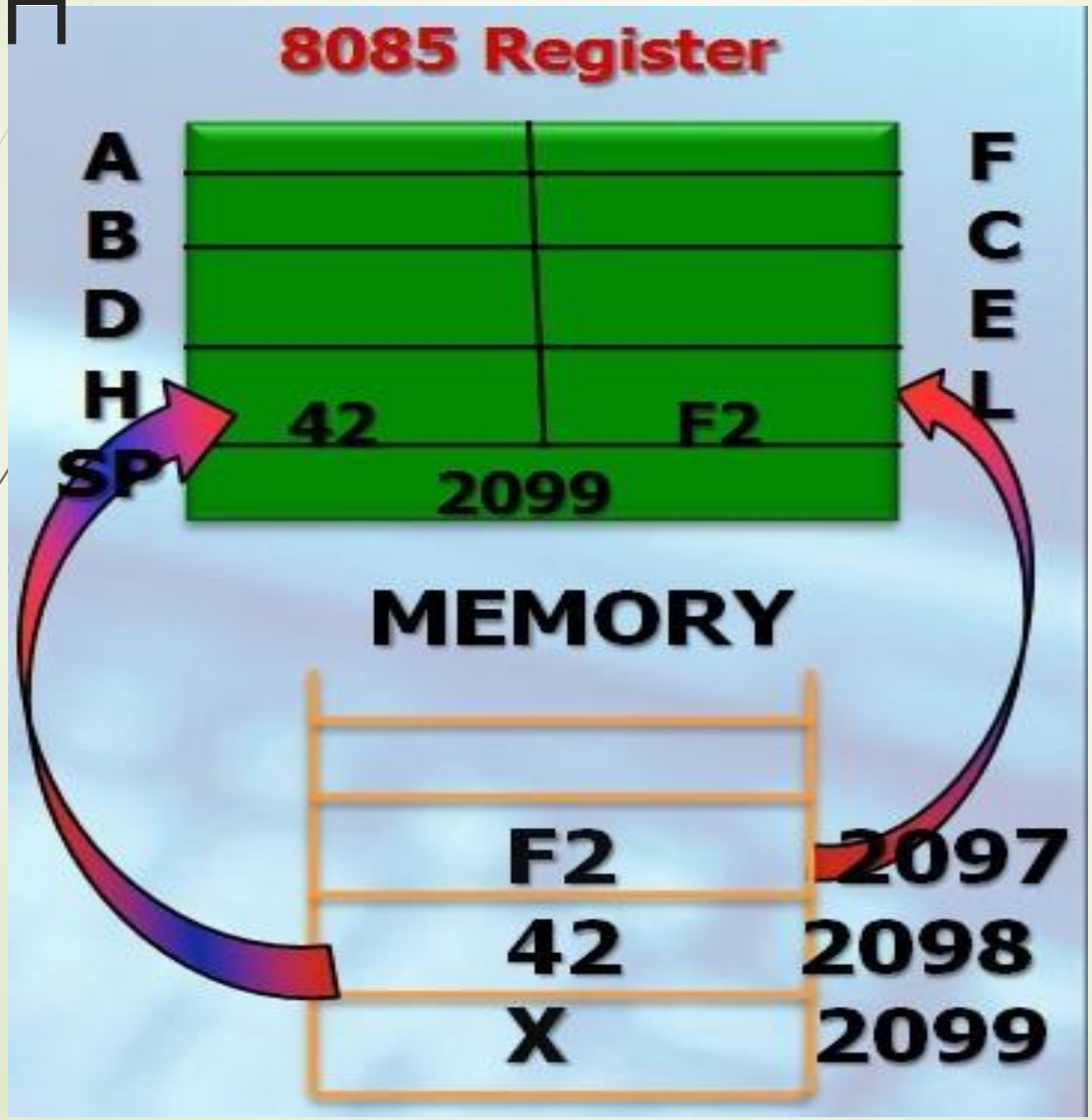

8085 Register
A
B
D
H
SP
42
F2
2097
F
C
E
L
Memory
F2
42
X
2097
2098
2099

POP- Pop stack to register pair

Opcode	Operand
POP	Reg. pair

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

POP H


8085 Register
A
B
D
H
SP
F
C
E
L
42
F2
2099
MEMORY
F2
42
X
2097
2098
2099

IN- Copy data to accumulator from a port with 8-bit address

Opcode	Operand
IN	8-bit port address

- The contents of I/O port are copied into accumulator.
- **Example:** IN $8C_H$

BEFORE EXECUTION

IN 80H

AFTER EXECUTION

OUT- Copy data from accumulator to a port with 8-bit address

Opcode	Operand
OUT	8-bit port address

- The contents of accumulator are copied into the I/O port.
- **Example:** OUT 78_H

BEFORE EXECUTION

OUT 50H

AFTER EXECUTION

2. Arithmetic Instructions

- These instructions perform the operations like:
  - Addition
  - Subtract
  - 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.
- **Example:** The contents of register B cannot be added directly to the contents of register C.

ADD

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

- 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 B or ADD M

BEFORE EXECUTION

A	04

B	C	05
D	E	
H	L	

ADD C
A=A+C

AFTER EXECUTION

A.	09

B.	C	05
D	E	
H	L	

04+05=09

BEFORE EXECUTION

A	04

B		C	
D		E	
H	20	L	50

ADD M
A=A+M

10

2050

AFTER EXECUTION

A	14

B		C	
D		E	
H	20	L	50

04+10=14

1

0

2050

ADC

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

- 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 B or ADC M

BEFORE EXECUTION

CY	01
A	50

B		C	05
D		E	
H		L	

ADC C

A=A+C+CY

AFTER EXECUTION

A	56

B		C	20
D		E	
H		L	

50+05+01=56

BEFORE EXECUTION

CY	1
A	06

2050H

30

H	20	L	50

ADC M

A=A+M+CY

AFTER EXECUTION

A	37

2050H

30

H	20	L	50

06+1+30=37

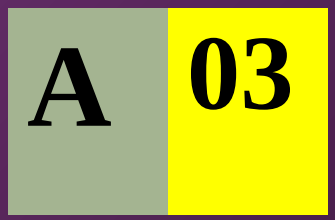
ADI

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

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

BEFORE EXECUTION

AFTER EXECUTION



ADI 05_H

A=A+DATA(8)

03+05=08

ACI

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

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

BEFORE EXECUTION

AFTER EXECUTION

ACI 20_H

A=A+DATA (8)+CY

CY	1
A	05

A	26

05+20+1=26

DAD

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.
- **No other flags are changed.**
- **Example:** DAD B or DAD D

BEFORE EXECUTION

D	12	E	34
H	23	L	45

DAD D

AFTER EXECUTION

D	12	E	34
H	35	L	79

```
1234
2345  +
-------
3579
```

DAD D HL=HL+DE
DAD B HL=HL+BC

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.

SUB

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.
- If the operand is memory location, its address is specified by H-L pair.
- All flags are modified to reflect the result of subtraction.
- **Example:** SUB B or SUB M

BEFORE EXECUTION

A	09		
B		C	04
D		E	
H		L	

SUB C
A=A-C

AFTER EXECUTION

A.	05		
B.		C	04
D		E	
H		L	

09-04=05

BEFORE EXECUTION

A	14		
B		C	
D		E	
H	20	L	50

SUB M
A=A-M

10

2050

AFTER EXECUTION

A	04		
B		C	
D		E	
H	20		50

L

10

2050

14-10=04

SBB

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.
- If the operand is memory location, its address is specified by H-L pair.
- All flags are modified to reflect the result of subtraction.
- **Example:** SBB B or SBB M

BEFORE EXECUTION

CY	01
A	08

B		C	05
D		E	
H		L	

SBB C

A=A-C-CY

AFTER EXECUTION

A	02

B		C	05
D		E	
H		L	

08-05-01=02

BEFORE EXECUTION

CY	1
A	06

2050H

02

H	20	L	50

SBB M

A=A-M-CY

AFTER EXECUTION

A	03

2050 H

02

H	20	L	50

06-02-1=03

SUI

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.
- All flags are modified to reflect the result of subtraction.
- **Example:** SUI 05_H

BEFORE EXECUTION

SUI 05_H

A=A-DATA(8)

AFTER EXECUTION

08-05=03

SBI

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

- 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.
- All flags are modified to reflect the result of subtraction.
- **Example:** SBI 45 H

BEFORE EXECUTION

AFTER EXECUTION

CY	1
A	25

SBI 20_H

A=A-DATA(8)-CY

A	04

25-20-01=04

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.

INR

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.

S,Z,P,AC flags are modified, but CY is not modified

- 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

A		
B	10	C
D		E
H		L

INR B

R=R+1

AFTER EXECUTION

A		
B	11	C
D		E
H		L

10+1=11

BEFORE EXECUTION

H	L
20	50

2050 H

10

INR M

M=M+1

AFTER EXECUTION

H	L
20	50

11

2050 H

10+1=11

INX

Opcode	Operand	Description
INX	Rp	Increment register pair by 1

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

No Flags are affected.

- **Example:** INX H or INX B or INX D

BEFORE EXECUTION

SP			
B		C	
D		E	
H	10	L	20

INX H
RP=RP+1

AFTER EXECUTION

SP			
B		C	
D		E	
H	10	L	21

1020+1=1021

DCR

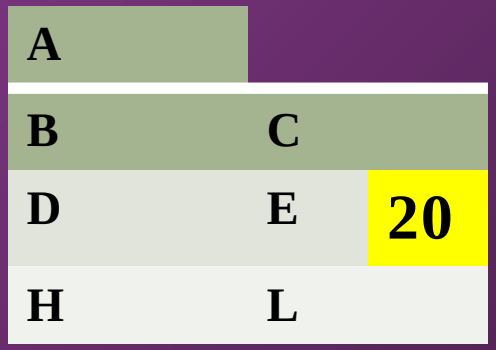
Opcode	Operand	Description
DCR	R M	Decrement register or memory by 1

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

S,Z,P,AC flags are modified, but CY is not modified

- If the operand is a memory location, its address is specified by the contents of H-L pair.
- **Example:** DCR B or DCR M

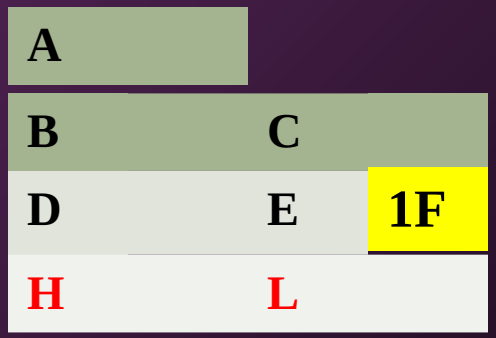
BEFORE EXECUTION



DCR E

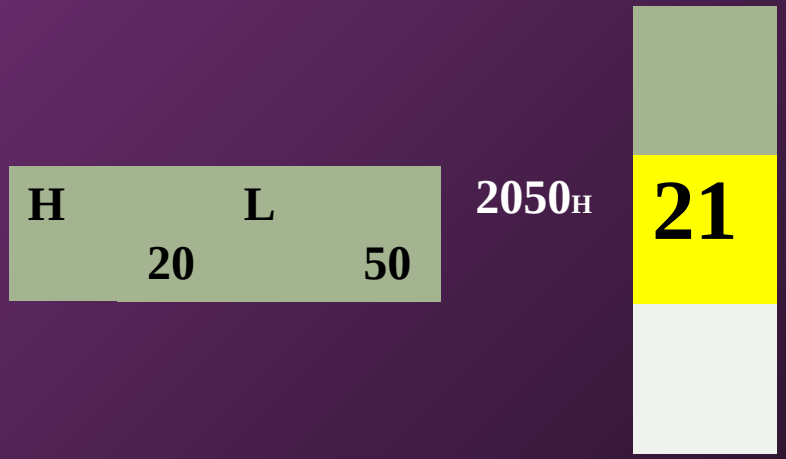
R=R-1

AFTER EXECUTION



20-1=19

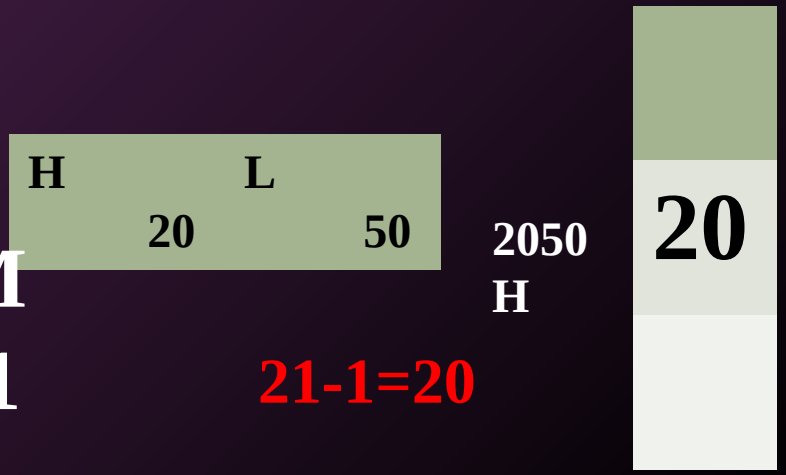
BEFORE EXECUTION



DCR M

M=M-1

AFTER EXECUTION



21-1=20

DCX

Opcode	Operand	Description
DCX	Rp	Decrement register pair by 1

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

No flag bit modifies

- **Example:** DCX H or DCX B or DCX D

BEFORE EXECUTION

SP			
B		C	
D		E	
H	10	L	21

DCX H
RP=RP-1

AFTER EXECUTION

SP			
B		C	
D		E	
H	10	L	20

3.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(NOT)

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.

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.

BEFORE EXECUTION

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

1010 1010=AA$_H$
0000 1111=0F$_H$

0000 1010=0A$_H$

ANA B
A=A and R

AFTER EXECUTION

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

BEFORE EXECUTION

C Y		A C	
A	55	2050 H	
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

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

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

BEFORE EXECUTION

AFTER EXECUTION

$1011\ 0011 = B3_H$

$0011\ 1111 = 3F_H$

$0011\ 0011 = 33_H$

CY	AC
A	B3

ANI 3FH

A=A and DATA(8)

CY	0	AC	1
A	33		

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.
- S, Z, P are modified to reflect the result.
- CY and AC are reset.
- **Example:** ORA B or ORA M.

1010 1010=AAH
0001 0010=12H

1011 1010=BAH

BEFORE EXECUTION

CY		AC	

AFTER EXECUTION

CY	0	AC	0

ORA B
A=A or R

A	AA		
B	12	C	
D		E	
H		L	

A	BA		
B	12		
D		E	
H		L	

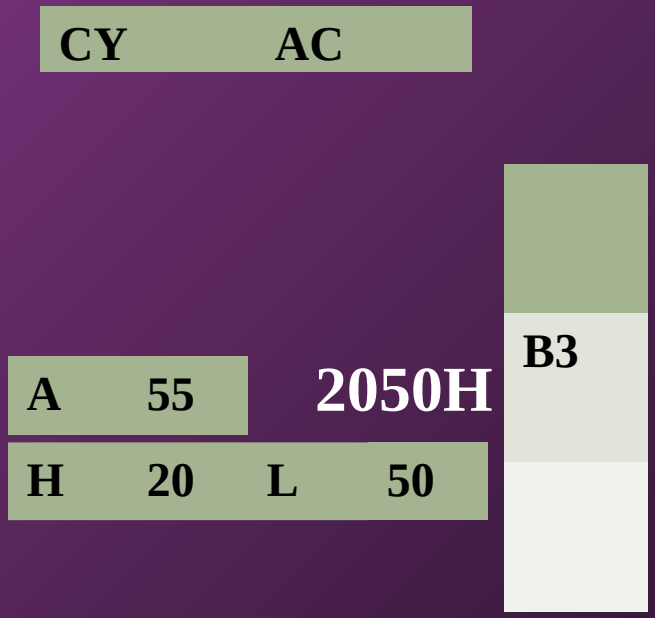
0101 0101=55H
1011 0011=B3H

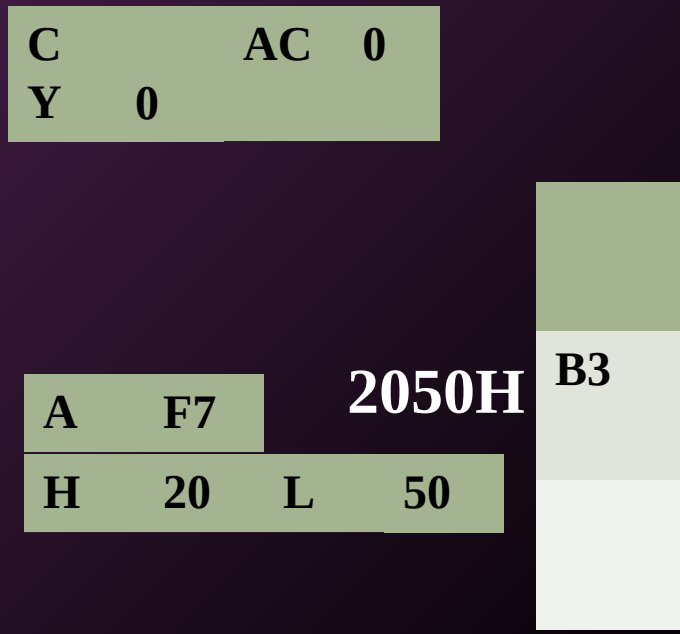
1111 0111=F7H

BEFORE EXECUTION

AFTER EXECUTION

ORA M
A=A or M





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

- The contents of the accumulator are logically ORed with 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.

1011 0011=B3H
0000 1000=08H

1011 1011=BBH

BEFORE EXECUTION

CY	AC

A	B3

AFTER EXECUTION

CY	AC
0	0

A	BB

ORI 08H
A=A or DATA(8)

Opcode	Operand	Description
XRA	R M	Logical XOR 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.
- 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 and AC are reset.
- **Example:** XRA B or XRA M.

BEFORE EXECUTION

C Y		A C	
A	AA		
B		C	2D
D		E	
H		L	

1010 1010=AA_H
0010 1101=$2D_H$

1000 0111=87_H

XRA C
A=A xor R

AFTER EXECUTION

CY	0	AC	0
A	87		
B		C	2D
D		E	
H		L	

BEFORE EXECUTION

0101 0101=55H
1011 0011=B3H

1110 0110=E6H

AFTER EXECUTION

C		A	
Y		C	
A	55		
H	20	L	50

2050H

B3

XRA M
A=A xor M

CY	0	AC	0
A	E6		
H	20	L	50

2050H

B3

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.
- S, Z, P are modified to reflect the result.
- CY and AC are reset.
- **Example:** XRI 86_H.

1011 0011=B3H
0011 1001=39H

1000 1010=8AH

BEFORE EXECUTION

C Y	A C
A	B3

XRI 39H
A=A xor DATA(8)

AFTER EXECUTION

CY	AC
0	0
A	8A

Opcode	Operand	Description
CMA	None	Compliment Accumulator

- The contents of Accumulator are modified
- The result is placed in the accumulator.
- No flags are modified
- **Example:** CMA

	Before execution			**After execution**	
A	1 0 0 0 1 0 0 1	89H	A	0 1 1 1 0 1 1 0	76H

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.

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** .

BEFORE EXECUTION

CY		Z	

A	10		
B		C	
D	20	E	
H		L	

A>R: CY=0
A=R: ZF=1
A<R: CY=1

CMP D
A-R

AFTER EXECUTION

CY	01	Z	0

A	10		
B		C	
D	20	E	
H		L	

10<20:CY=01

BEFORE EXECUTION

C Y		Z	

A	B8	2050H	

H	20	L	50

B8

A>M: CY=0
A=M: ZF=1
A<M: CY=1

CMP M
A-M

AFTER EXECUTION

CY	0	ZF	1

A	B8	2050H	

H	20	L	50

B8

B8=B8 :ZF=01

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.

BEFORE EXECUTION

C Y	Z
A	BA

A>DATA: CY=0
A=DATA: ZF=1
A<DATA: CY=1

CPI 30H
A-DATA

AFTER EXECUTION

CY	AC
0	0
A	BA

BA>30 : CY=00

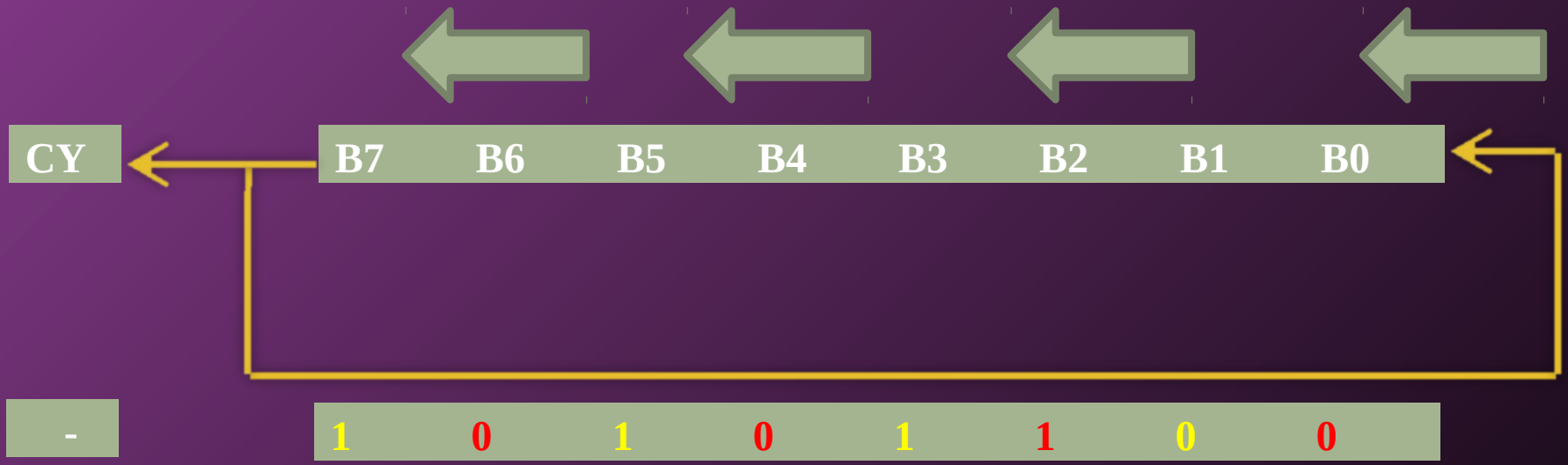
Rotate

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

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.

BEFORE EXECUTION



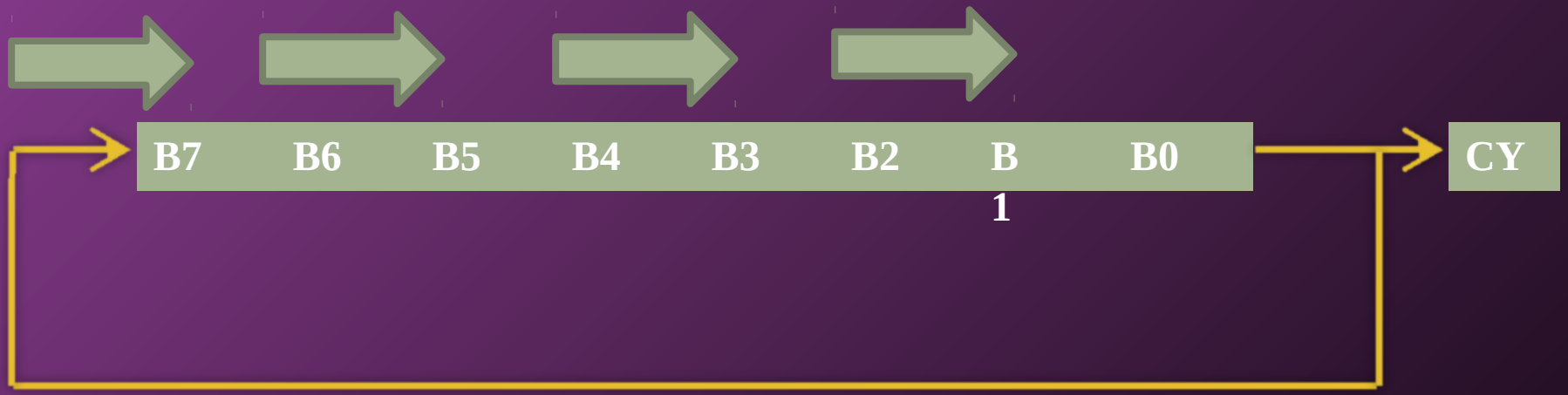
AFTER EXECUTION

B7	B6	B5	B4	B3	B2	B1	B0	B7
1	0	1	0	1	1	0	0	1

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.

BEFORE EXECUTION

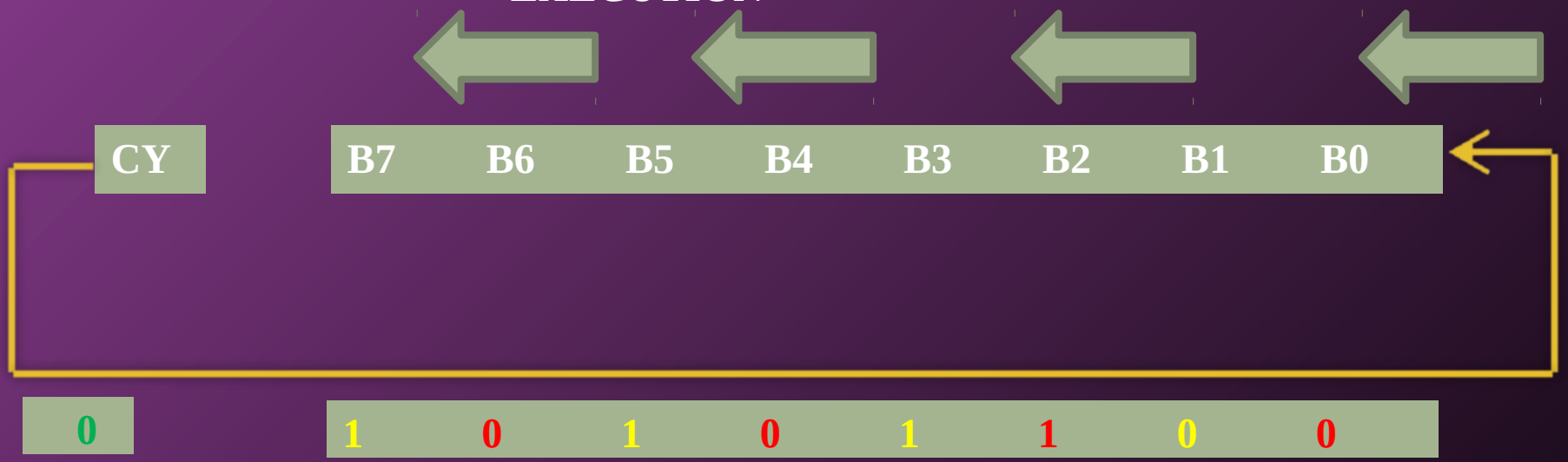


AFTER EXECUTION

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.

BEFORE EXECUTION



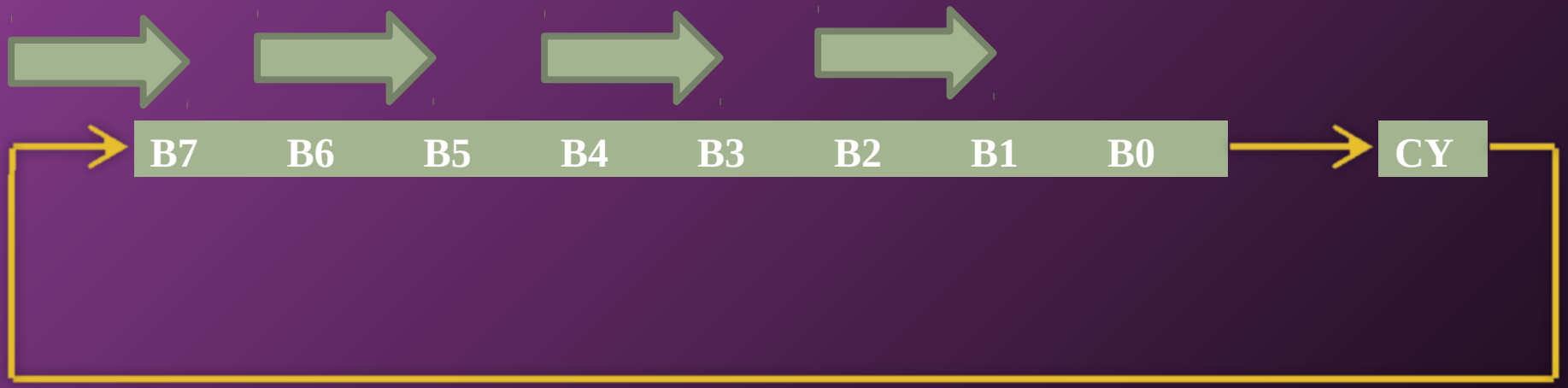
AFTER EXECUTION

B7	B6	B5	B4	B3	B2	B1	B0	CY
1	0	1	0	1	1	0	0	0

Opcode	Operand	Description
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.

BEFORE EXECUTION



AFTER EXECUTION

Complement

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

Opcode	Operand	Description
CMA	None	Complement accumulator

- The contents of the accumulator are complemented.
- No flags are affected.
- **Example:** CMA. A=A’

BEFORE EXECUTION

A	00

AFTER EXECUTION

A	FF

Opcode	Operand	Description
CMC	None	Complement carry

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

BEFORE EXECUTION

CY	0

AFTER EXECUTION

CY	1

Opcode	Operand	Description
STC	None	Set carry

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

S-set (1) **C-clear (0)**

4.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,

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.

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 based on the specified flag of the PSW.
- **Example:** JZ 2034 H.

Jump Conditionally

Opcode	Description	Status Flags
JC	Jump if Carry	CY = 1
JNC	Jump if No Carry	CY = 0
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
JM	Jump on Minus	S=1
JP	Jump on Positive	S=0

S- Sign, C-Carry , Z-Zero , P-Parity

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.
- 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.

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

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.

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

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 1 or RST 2 ….

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

- The control instructions control the operation of microprocessor.

5. Control Instructions

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

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

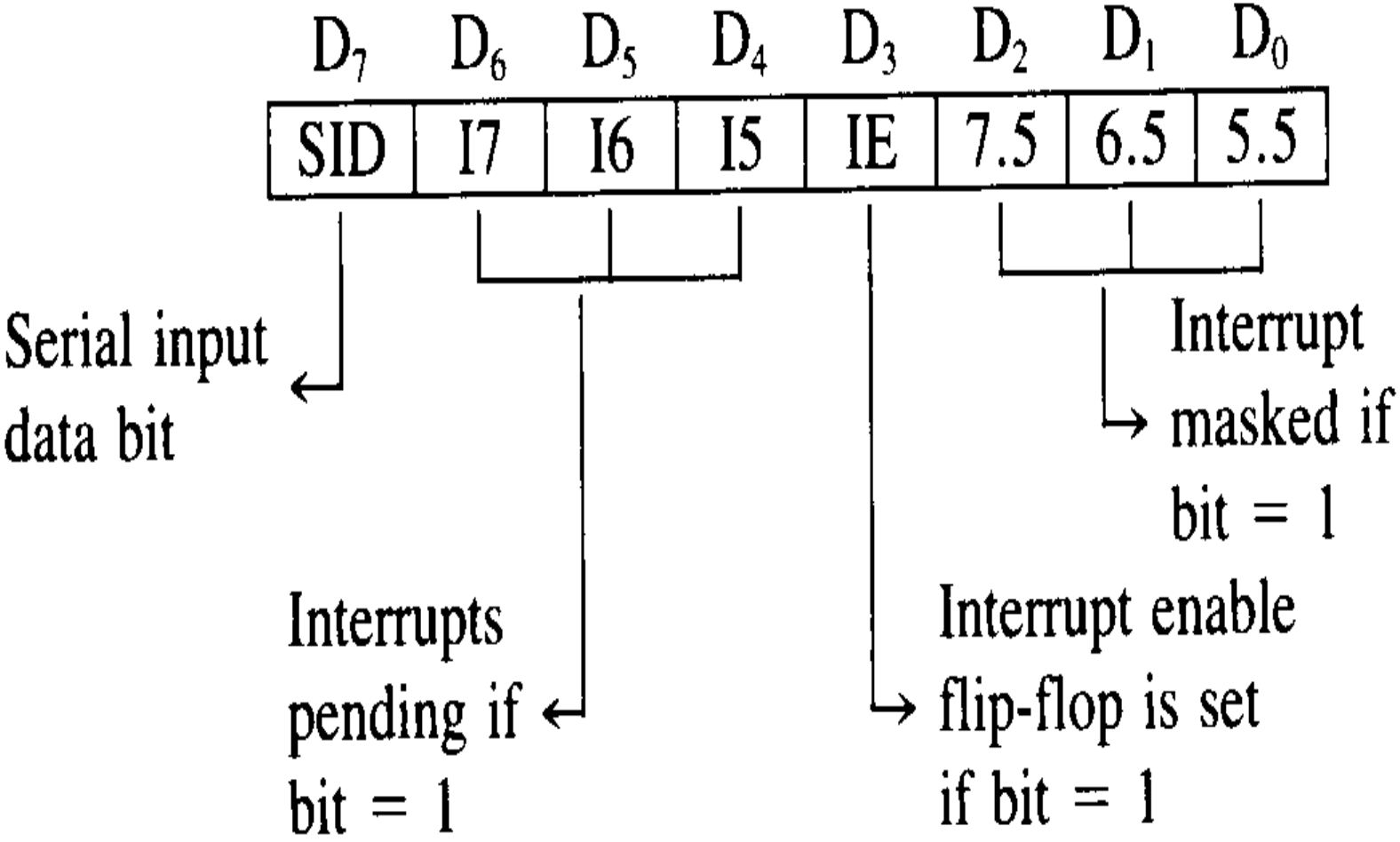
Opcode	Operand	Description
EI	None	Enable interrupt

- 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

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 and read serial data input bit.
- The instruction loads eight bits in the accumulator with the following interpretations.
- **Example:** RIM

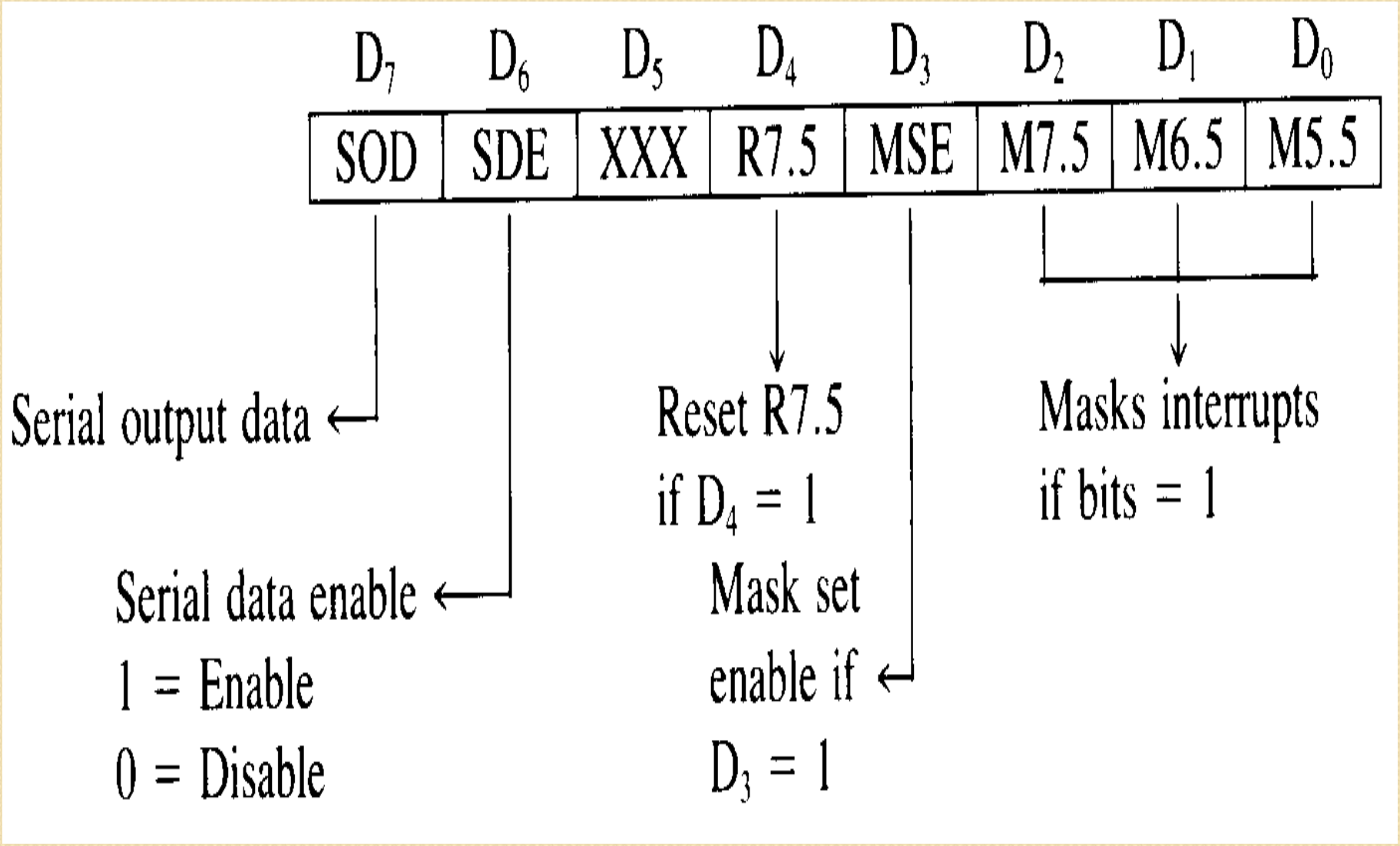
RIM Instruction



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

SIM Instruction



References

- www.slideshare.net
- www.docstoc.com
- www.slideworld.com
- www.nptel.ac.in
- www.scribd.com
- http://opencourses.emu.edu.tr/
- http://engineeringppt.blogspot.in/
- http://www.pptsearchengine.net/
- www.4shared.com
- www.eazynotes.com
- http://8085projects.info/

Books:

Microprocessors and microcontrollers by krishnakanth

Microprocessors and microcontrollers by Nagoor Kani

Staff references

- **8085 microprocessor** by Sajid Akram, researcher/lecturer at c.abdul hakeem college of engineering and technology
- **Timingdiagram** by puja00 (slideshare.net)
- **Microprocessor 8086** by Gopikrishna Madanan, Assistant Professor of Physics at Collegiate Education, Kerala, India