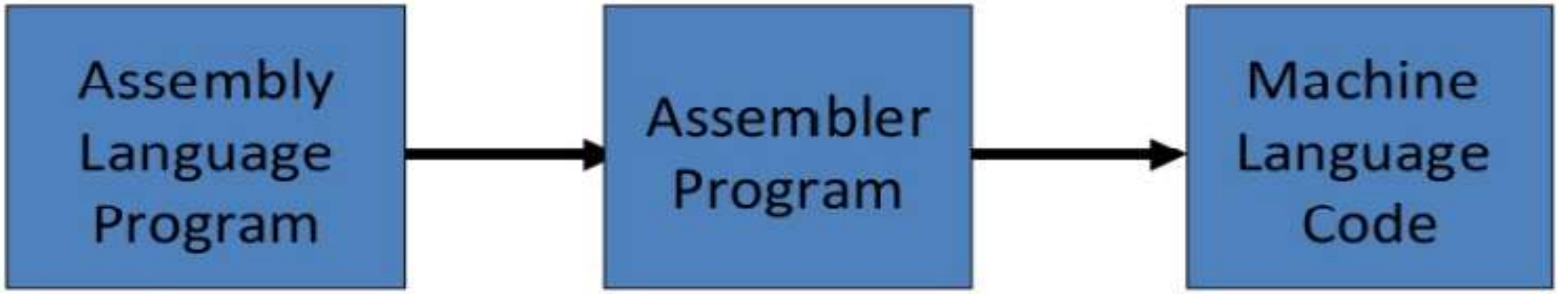
UNIT 2 – 8085 Instruction set and assembly language programming

- 8085 assembly language programming
- Addressing modes
- 8085 instruction set
- Instruction formats
- Instruction Classification:
  - Data transfer
  - Arithmetic operations
  - Logical operations
  - Branching operations
  - Machine control —Stack and subroutines
- Example Programs

Microprocessor understands Machine Language only!

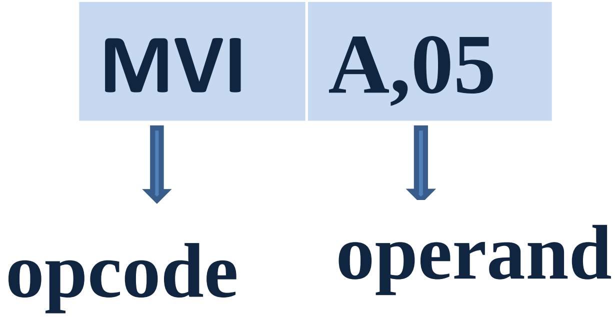
- Microprocessor cannot understand a program written in Assembly language
- A program known as **Assembler** is used to convert a Assembly language program to machine language



Instruction Format

- Program is defind as sequence of instructions
- Instructions are also called as mnemonics
- Each instruction has two parts.
  - The first part is the task or operation to be performed.
    - This part is called the “opcode” (operation code).
  - The second part is the data to be operated on
    - Called the “operand”.

Example :



MVI A,05

opcode operand

Addressing Modes in Instructions

- The process of specifying the data to be operated on by the instruction is called addressing. The various formats for specifying operands are called addressing modes. The 8085 has the following five types of addressing:

  - ❑ **Immediate addressing**
  - ❑ **Memory direct addressing**
  - ❑ **Register direct addressing**
  - ❑ **Indirect addressing**
  - ❑ **Implicit addressing**

Addressing Modes

- The microprocessor has different ways of specifying the data for the instruction. These are called “addressing modes”.

The 8085 has five addressing modes:

Sl.No.	Addressing Mode	Example
1	Immediate	MVI B, 45
2	Register	MOV A,B
3	Direct	LDA 4000
4	Indirect	LDAX B
5	Implied	CMA

Addressing Modes Contd.

Immediate Addressing:

In this mode, the operand given in the instruction - a byte or word – transfers to the destination register or memory location.

Ex: MVI A, 9A H

- The immediate data is part of the instruction.
- The operand is stored in the register mentioned in the instruction.

Direct Addressing:

Memory direct addressing moves a byte or word between a memory location and register. The memory address is given in the instruction.

Ex: LDA 850FH

This instruction is used to load the content of memory address 850FH in the accumulator.

Addressing Modes Contd.

Register Addressing:

Register direct addressing transfer a copy of a byte or word from source register to destination register.

Ex: MOV B, C

It copies the content of register C to register B.

Indirect Addressing:

Indirect addressing transfers a byte or word between a register and a memory location.

Ex: MOV A, M

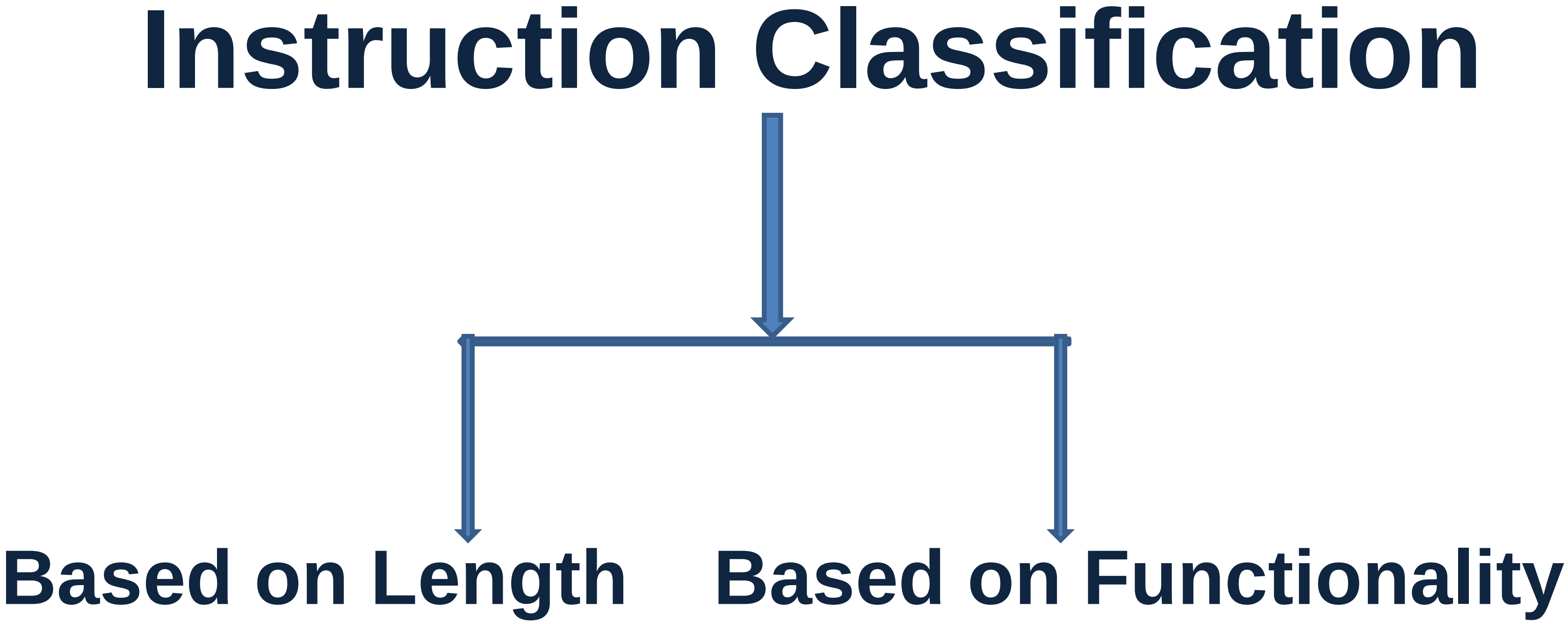
Here the data is in the memory location pointed to by the contents of HL pair. The data is moved to the accumulator.

Implicit Addressing

In this addressing mode the data itself specifies the data to be operated upon.

Ex: CMA

The instruction complements the content of the accumulator. No specific data or operand is mentioned in the instruction.



Instruction Classification

Based on Length

Based on Functionality

Classification based on length

❑ One-byte instructions: Instruction having one byte in machine code

Example: MOV A,B , ADD M

Opcode	Operand	Machine code/Hex code
MOV	A, B	78
ADD	M	86

❑ Two-byte instructions: Instruction having two byte in machine code

Example: MVI A,7F

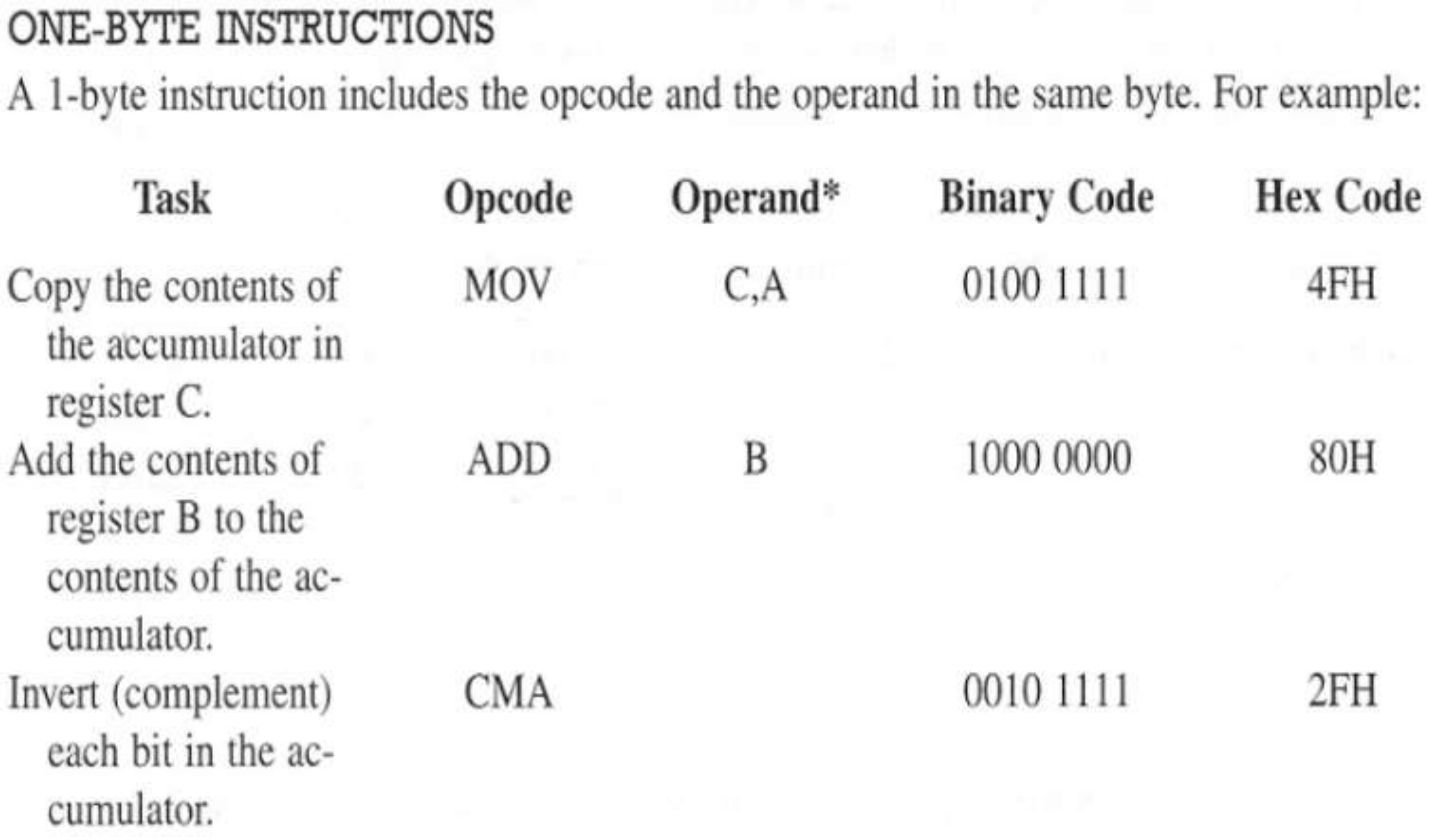
Opcode	Operand	Machine code/Hex code	Byte description
MVI	A, 7FH	3E	First byte
		7F	Second byte
ADI	0FH	C6	First byte
		0F	Second byte

❑ Three-byte instructions: instruction having three byte in machine code.

Example.: LDA 8850

Opcode	Operand	Machine code/Hex code	Byte description
JMP	9050H	C3	First byte
		50	Second byte
		90	Third byte
LDA	8850H	3A	First byte
		50	Second byte
		88	Third byte

8085 Instruction Types

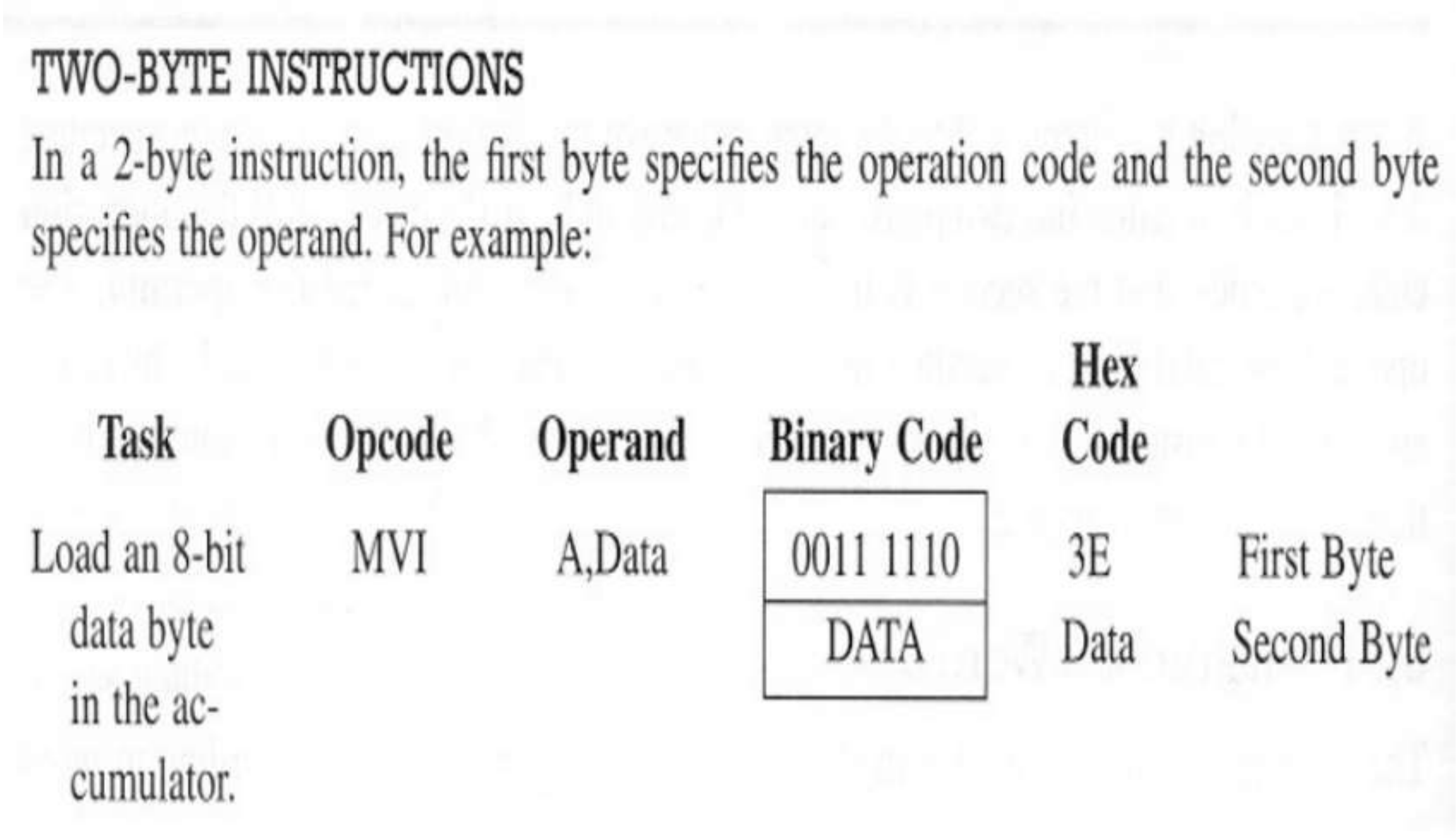


ONE-BYTE INSTRUCTIONS

A 1-byte instruction includes the opcode and the operand in the same byte. For example:

Task	Opcode	Operand*	Binary Code	Hex Code
Copy the contents of the accumulator in register C.	MOV	C,A	0100 1111	4FH
Add the contents of register B to the contents of the accumulator.	ADD	B	1000 0000	80H
Invert (complement) each bit in the accumulator.	CMA		0010 1111	2FH

8085 Instruction Types

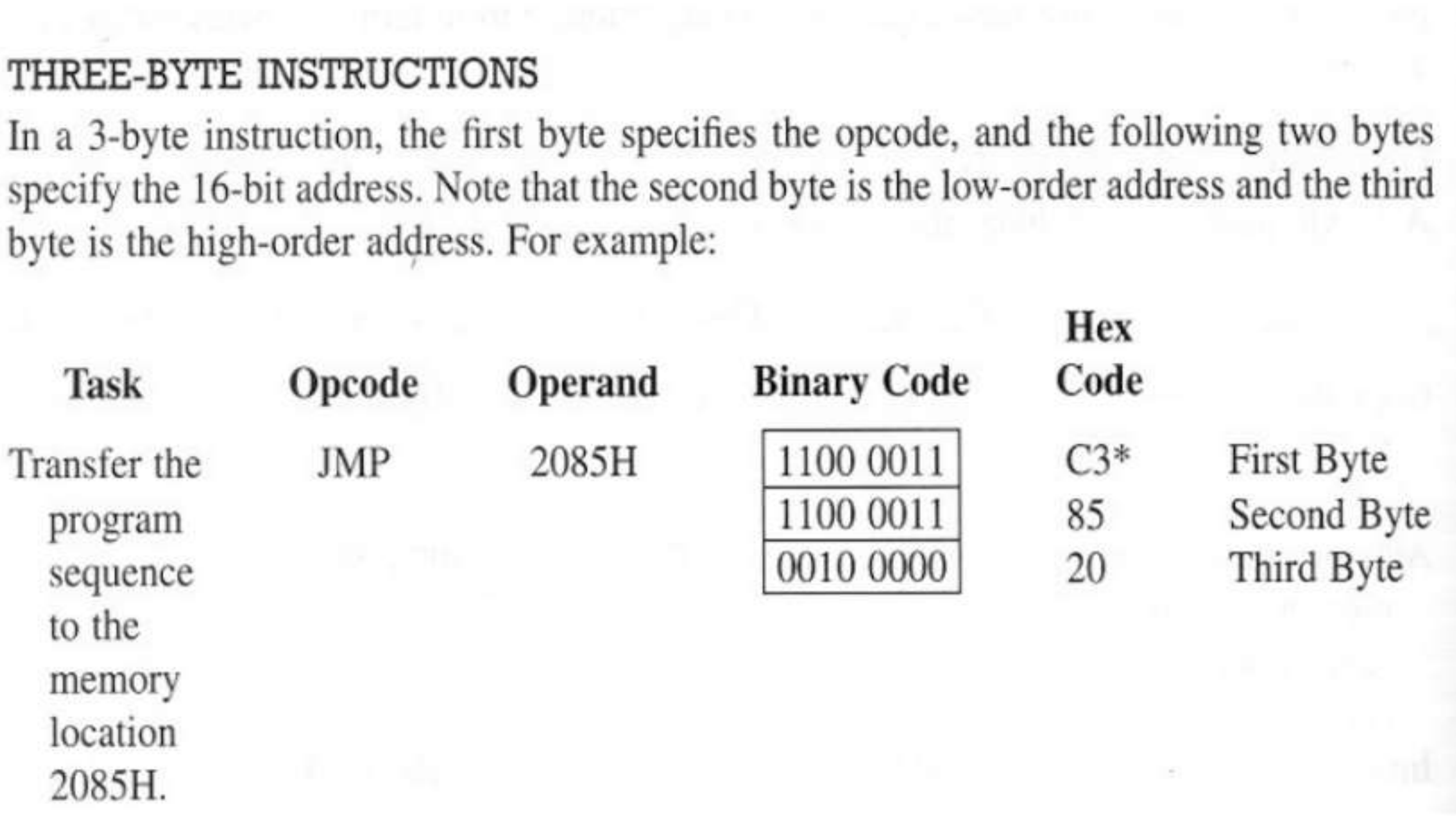


TWO-BYTE INSTRUCTIONS

In a 2-byte instruction, the first byte specifies the operation code and the second byte specifies the operand. For example:

Task	Opcode	Operand	Binary Code	Hex Code	
Load an 8-bit data byte in the accumulator.	MVI	A,Data	0011 1110	3E	First Byte
			DATA	Data	Second Byte

8085 Instruction Types



THREE-BYTE INSTRUCTIONS

In a 3-byte instruction, the first byte specifies the opcode, and the following two bytes specify the 16-bit address. Note that the second byte is the low-order address and the third byte is the high-order address. For example:

Task	Opcode	Operand	Binary Code	Hex Code	
Transfer the program sequence to the memory location 2085H.	JMP	2085H	1100 0011	C3*	First Byte
			1100 0011	85	Second Byte
			0010 0000	20	Third Byte

Classification based on functionality

- Data Transfer
- Arithmetic
- Logical
- Branching
- Machine control operations

The 8085 Instruction Set

The 8085 Instructions

Since the 8085 is an 8-bit device it can have up to 28 (256) instructions. However, the 8085 only uses 246 combinations that represent a total of 74 instructions. Most of the instructions have more than one format.

These instructions can be grouped into five different groups:

- Data Transfer Operations
- Arithmetic Operations
- Logic Operations
- Branch Operations
- Machine Control Operations

Data Transfer Instructions

Data transfer instructions move the data from source to destination. The content of source which is moved remains unchanged.

Data Transfer Operations

- ❑ These operations simply COPY the data from the source to the destination.
- ❑ MOV, MVI, LDA, and STA
- ❑ They transfer:Data between registers.
- ❑ Data Byte to a register or memory location.
- ❑ Data between a memory location and a register.
- ❑ Data between an I\O Device and the accumulator.

❖ The data in the source is not changed.

Data Transfer Instructions

- ❑ MOV
- ❑ MVI
- ❑ LXI
- ❑ LDA
- ❑ LDAX
- ❑ LHLD
- ❑ STA
- ❑ STAX
- ❑ SHLD
- ❑ XCHG

MOV

MOV	R1, R2	[R1] <- [R2]
	R, M	Copy from the location pointed by the HL pair (M) to register [R] <- [[HL]]
	M, R	Copy from register to the location pointed by the HL pair (M) [[HL]] <- [R]

- Example: MOV B,A or MOV M,B or MOV C,M

Before Execution

A	20	B	

A		F	
B	30	C	
D		E	
H	20	L	50

Memory

	2050

A		F	
B		C	
D		E	
H	20	L	50

Memory

40	2050

After Execution

A	20	B	20

A		F	
B	30	C	
D		E	
H	20	L	50

Memory

30	2050

A		F	
B		C	40
D		E	
H	20	L	50

Memory

40	2050

MVI

MVI	R, Data	Move immediate 8-bit into a register [R] <- Data
	M, Data	Move immediate 8-bit data into the memory location pointed by the HL pair (M) [[HL]] <- Data

Here, an 8-bit immediate data is stored in the destination register or a memory location, which is pointed by the HL register pair.

Size of instruction

2 bytes

Addressing mode

Immediate

Flags affected

None

Example

MVI A, 57H

MVI M, 57H

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

204FH	
HL=2050H	
2051H	

MVI M,40H

AFTER EXECUTION

204FH	
HL=2050H	40
2051H	

LXI

LXI	Rp, Data	Load the register pair with an immediate value [rp] <- 16-bit data 8 MSBs in the upper register 8 LSBs in the lower register

This instruction loads a 16-bit immediate data in the memory location pointed by the HL pair (M).

Size of instruction

3 bytes

Addressing mode

Immediate

-

LXI means..Load Register Pair with Immediate data.. 16bit data

LXI B ,2050H
Loads BC pair with value 2050H
B-> 20H
C-> 50H

Its similar to 2 MVI instruction
ie
MVI B,20H
MVI C,50H

LDA

LDA	**16-bit address**	**Load Accumulator with contents stored at an address** **[A] <- [[address]]**

The contents from the address (addresses in 8085 are 16-bit) are copied to the accumulator register. The contents of the source location remain unaltered.

Size of instruction

3 bytes

Addressing mode

Direct

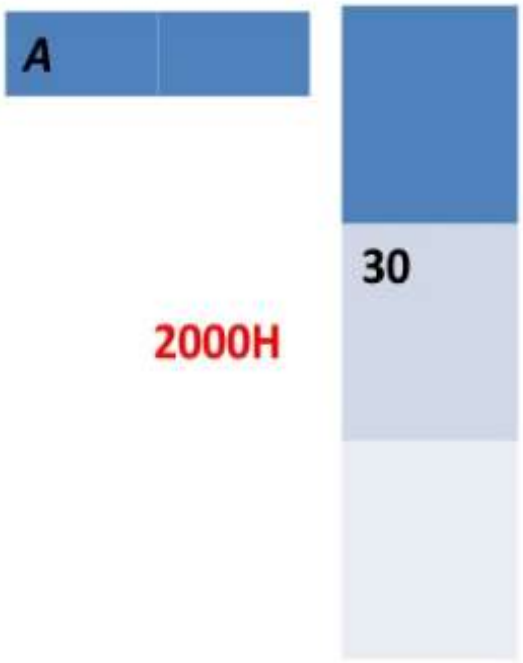
Flags affected

None

Example

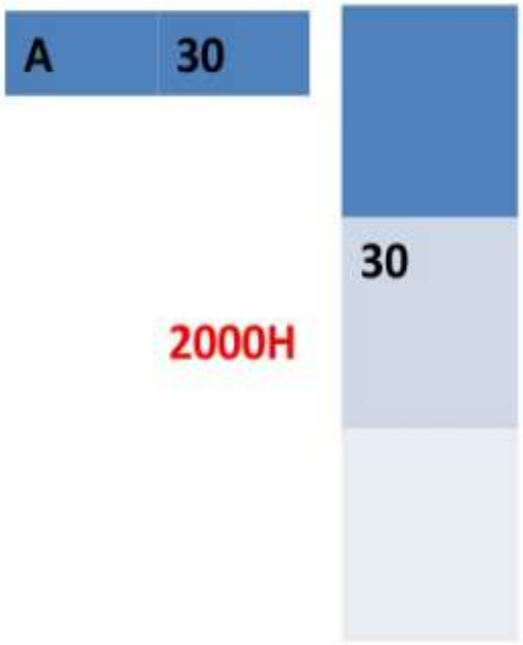
LDA 2034H //Contents stored at memory location 2034H are copied into the accumulator.

BEFORE EXECUTION



LDA 2000H

AFTER EXECUTION



STA

STA	16-bit address value	Store from the accumulator into a memory location [address] <- [A]

The contents of the accumulator register are copied into a memory location, which is specified by the operand.

Size of instruction

3 bytes

Addressing mode

Direct

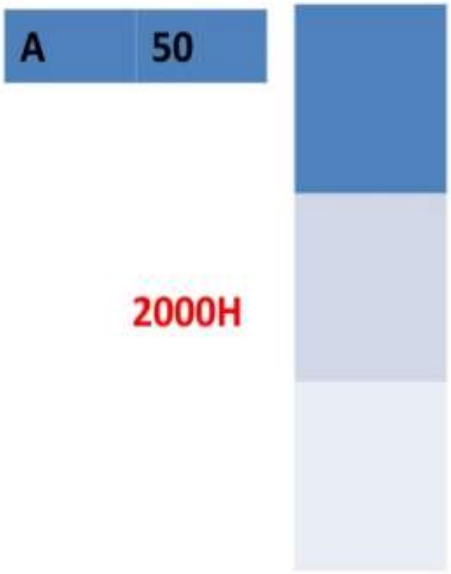
Flags affected

None

Example

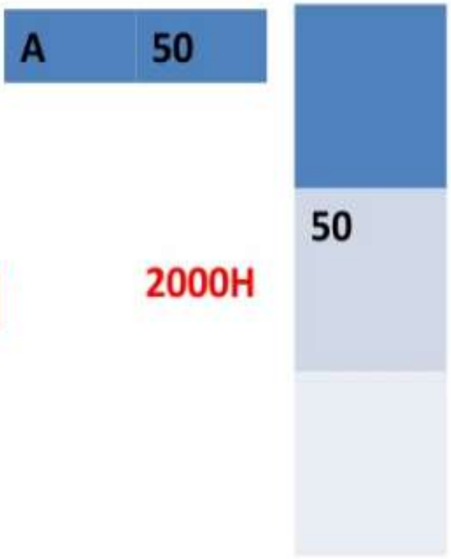
STA 1000H

BEFORE EXECUTION



STA 2000H

AFTER EXECUTION



LDAX

LDAX	Either B/D Register Pair	Load accumulator with data at the memory location pointed by the contents of the register pair. [A] <- [[rp]]

The contents of the mentioned register pair point to a memory location. LDAX instruction copies the contents of that particular memory location into the accumulator register. Neither the contents of the register pair nor that of the memory location is altered.

Size of instruction

3 bytes

Addressing mode

Register Indirect

Flags affected

None

Example

LDAX B

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

STAX

STAX	**Register pair**	**Store contents of the accumulator into other register pair** **[[RP]] <- [A]**

The contents of the accumulator register are copied into the memory specified by the register pair in the operand.

Size of instruction

1 byte

Addressing mode

Indirect

Flags affected

None

Example

STAX B

STAX D

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

LHLD

LHLD	**16-bit address**	**Load the H-L registers with contents location at memory locations given by the address value** **[L] <- [[address]]** **[H] <- [[address+1]]**

The LHLD instruction copies the contents of a specified memory location pointed out by a 16-bit address into register L. The contents of the next memory location are copied into register H.

Size of instruction

3 bytes

Addressing mode

LHLD Memory

Flags affected

None

Example

LHLD 2100H

- **LHLD 2050** Means..copy content of 2050 and 2051 to HL pair

  if
  2050 -> 90H
  2051->5AH

  LHLD 2050 implies..
  L -> 90H
  H -> 5AH

SHLD

SHLD	**16-bit address**	**Stores from the H-L registers into the specified location** **[address] <- [L]** **[address+1] <- [H]**

The first 8-bit contents of the register L is stored into the memory location specified by the 16-bit address. The next 8-bit contents of the register H are stored into the subsequent memory location.

Size of instruction

3 bytes

Addressing mode

Direct

Flags affected

None

Example

SHLD 1200H

BEFORE EXECUTION

H	30	L	60

204FH	
2500H	
2502H	

SHLD 2500H

AFTER EXECUTION

H	30	L	60

204FH	60
2500H	30
2502H	

- LXI B,2105
- LHLD 9206
- MOV B,C
- MOV A,M
- STA 9206
- STAX D
- XCHG

XCHG

XCHG	**None**	**Exchange H-L with D-E** **[HL] <-> [DE]**

The contents of register pair H-L are exchanged with the contents of the register-pair D-E. The information stored in register H is exchanged with that of D; similarly, that in register L is exchanged with the contents of the register E.

Size of instruction

1 byte

Addressing mode

Implicit

Flags affected

None

Example

XCHG

.

BEFORE EXECUTION

D	20	E	40
H	70	L	80

XCHG

AFTER EXECUTION

D	70	E	80
H	20	L	40

ARITHMATIC INSTRUCTIONS

- These instructions perform the operations like:
  - Addition
  - Subtract
  - Increment
  - Decrement

- These instructions perform arithmetic operations such as addition, subtraction, increment, and decrement.
- **Addition** - Any 8-bit number, or the contents of a register or the contents of a memory location can be added to the contents of the accumulator and the sum is stored in the accumulator. No two other 8-bit registers can be added directly (e.g., the contents of register B cannot be added directly to the contents of the register C). The instruction DAD is an exception; it adds 16-bit data directly in register pairs.
- **Subtraction -** Any 8-bit number, or the contents of a register, or the contents of a memory location can be subtracted from the contents of the accumulator and the results stored in the accumulator. The subtraction is performed in 2's compliment, and the results if negative, are expressed in 2's complement. No two other registers can be subtracted directly.
- **Increment/Decrement -** The 8-bit contents of a register or a memory location can be incremented or decrement by 1. Similarly, the 16-bit contents of a register pair (such as BC) can be incremented or decrement by 1.

Arithmetic Instructions

- ADD
  - ADD r or M
  - ADC r or M
  - ADI 8 bit data
  - ACI 8 bit data
  - DAD rp
- SUB
  - SUB r or M
  - SBB r or M
  - SUI 8 bit data
  - SBI 8 bit data
- INR r or M
- DCR r or M
- INX rp
- DCX rp
- DAA

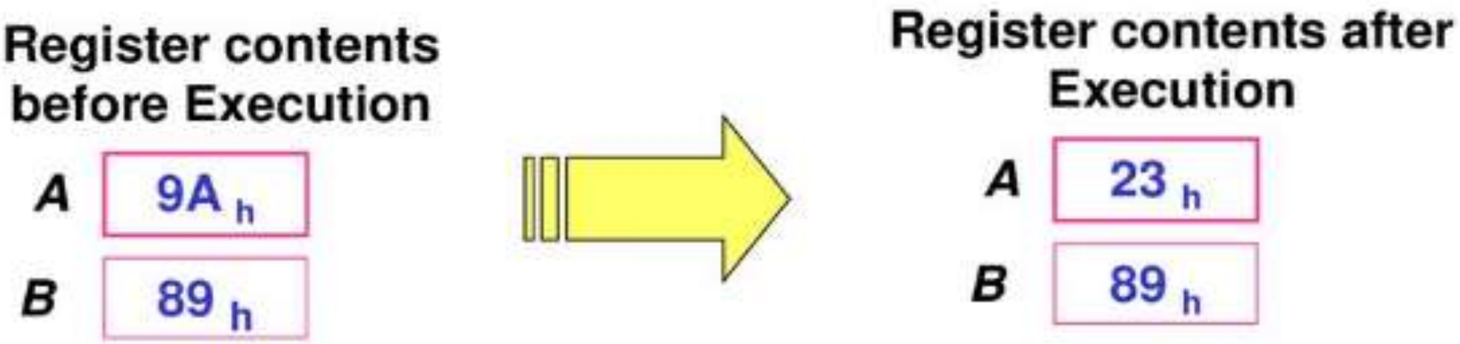
ADD

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

Example

Instruction: **ADD B**



Register contents before Execution

A 9A h

B 89 h

Register contents after Execution

A 23 h

B 89 h

```
1001 1010
1000 1001
0010 0011
```

Flag: S=0, Z=0, AC=1 , P=0 and CY=1

Note: All flags are affected during the execution of arithmetic instruction.

Example : ADD M

ADD M

A=A+M

Before Execution

A	04

B		C	
D		E	
H	20	L	50

10

2050

After Execution

A	14

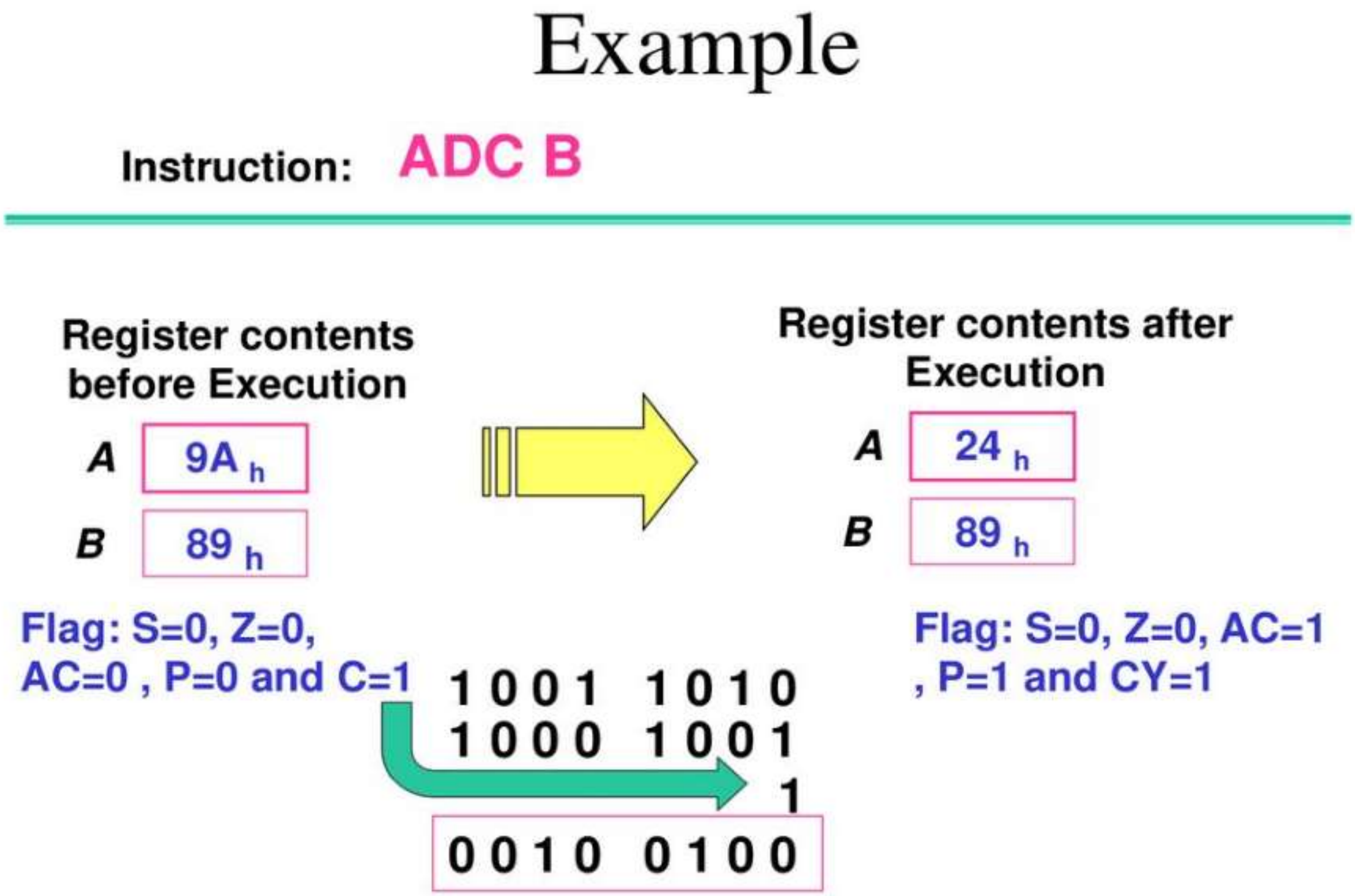
B		C	
D		E	
H	20	L	50

10

2050

04+10=14

ADC



Example

Instruction: **ADC B**

Register contents before Execution

A	9A h
B	89 h

Flag: S=0, Z=0, AC=0 , P=0 and C=1

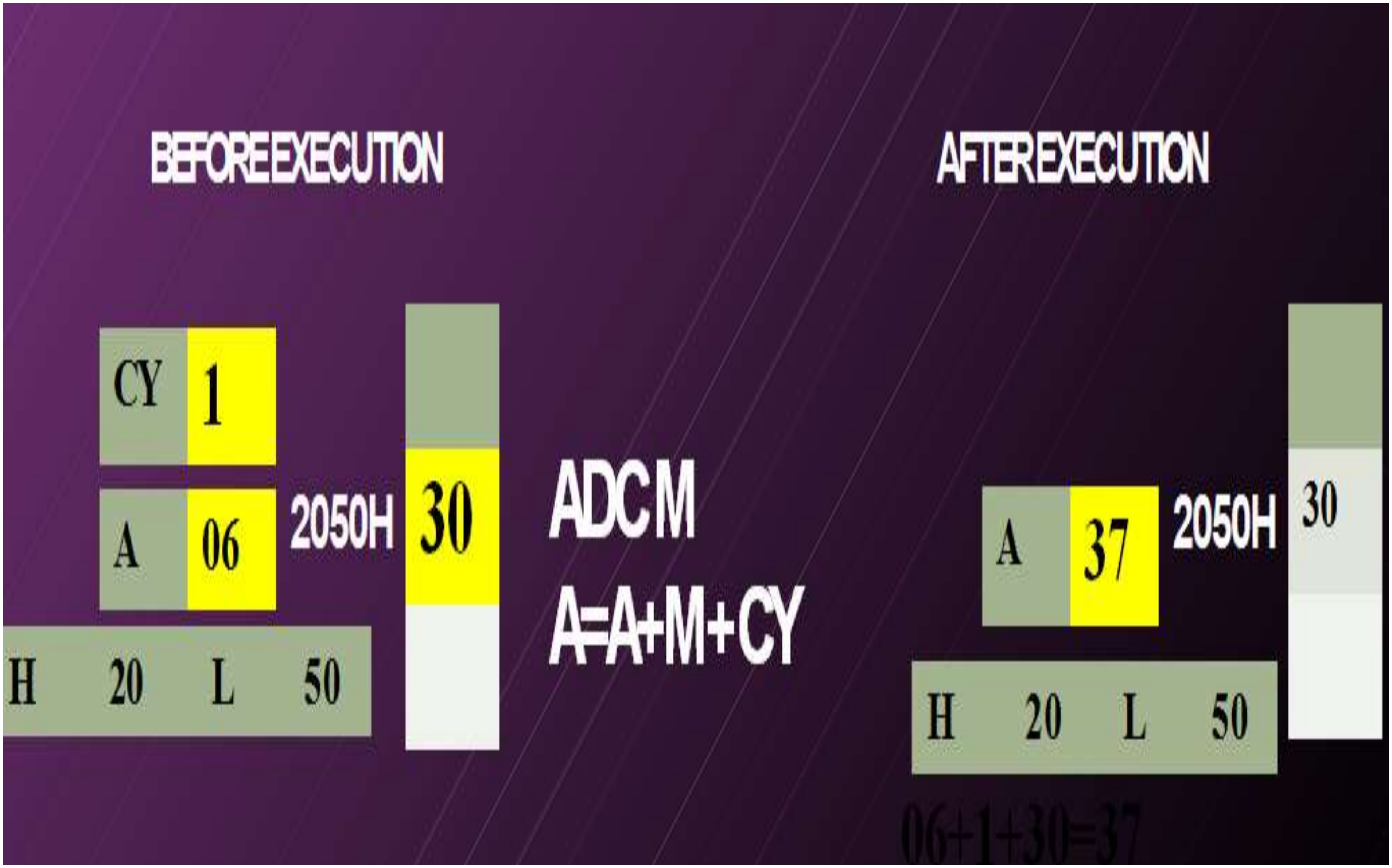
Register contents after Execution

A	24 h
B	89 h

Flag: S=0, Z=0, AC=1 , P=1 and CY=1

```
1001 1010
1000 1001
        1
0010 0100
```

ADC M



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

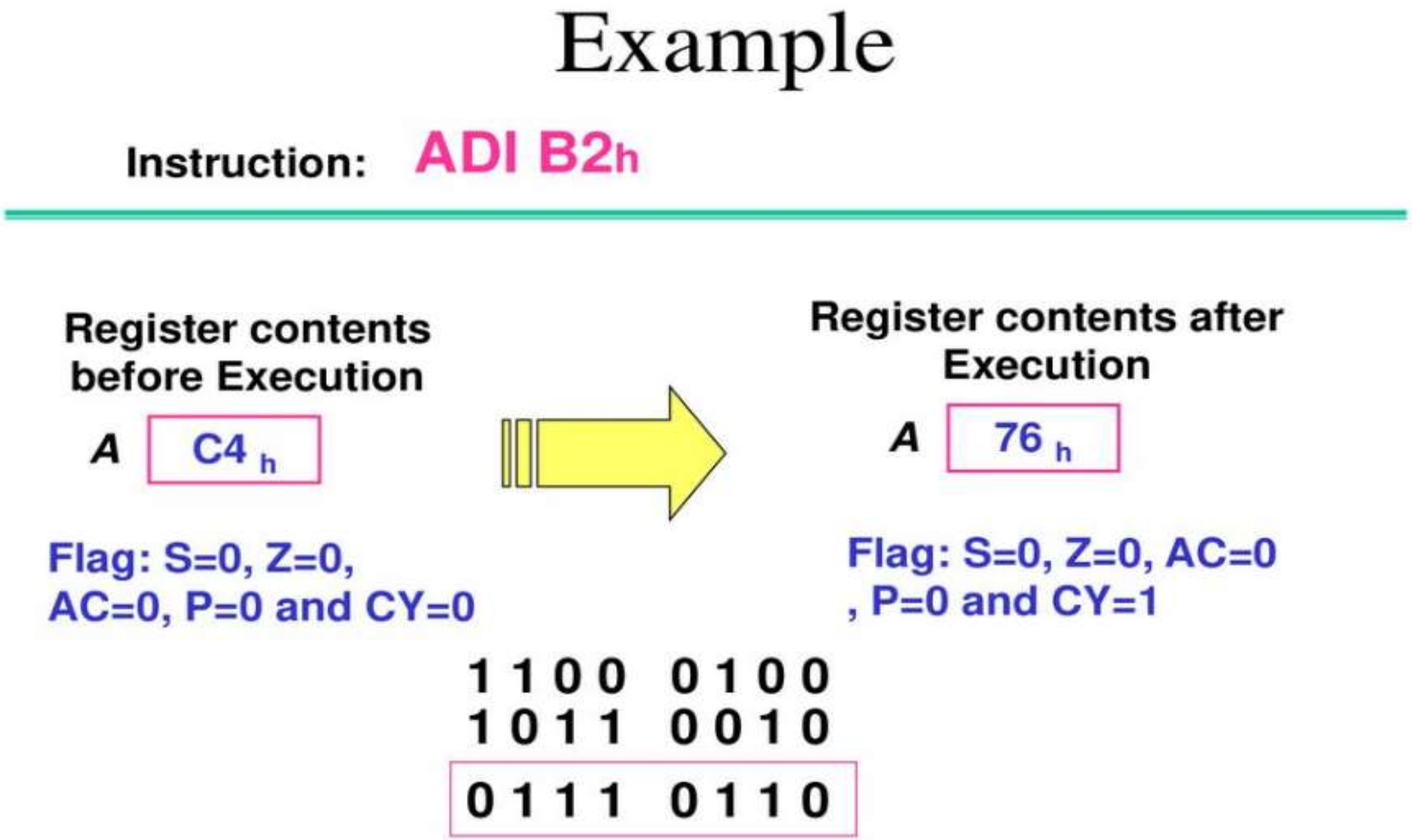
06+1+30=37

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**

ADI DATA



Example

Instruction: **ADI B2h**

Register contents before Execution

A C4 h

Flag: S=0, Z=0, AC=0, P=0 and CY=0

Register contents after Execution

A 76 h

Flag: S=0, Z=0, AC=0 , P=0 and CY=1

```
1 1 0 0   0 1 0 0
1 0 1 1   0 0 1 0
0 1 1 1   0 1 1 0
```

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

Instruction: ACI 15_h

Register contents before Execution

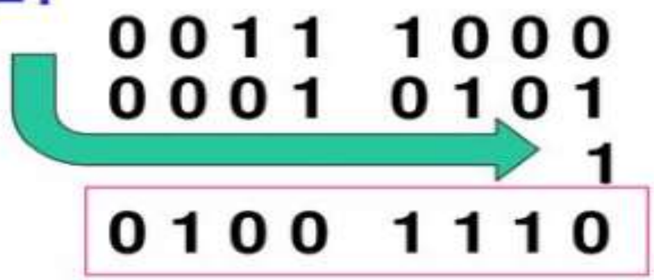
A 38_h

Flag: S=0, Z=0, AC=0 , P=0 and C=1

Register contents after Execution

A $4E_h$

Flag: S=0, Z=0, AC=0 , P=1 and CY=0



```
0 0 1 1   1 0 0 0
0 0 0 1   0 1 0 1
                1
0 1 0 0   1 1 1 0
```

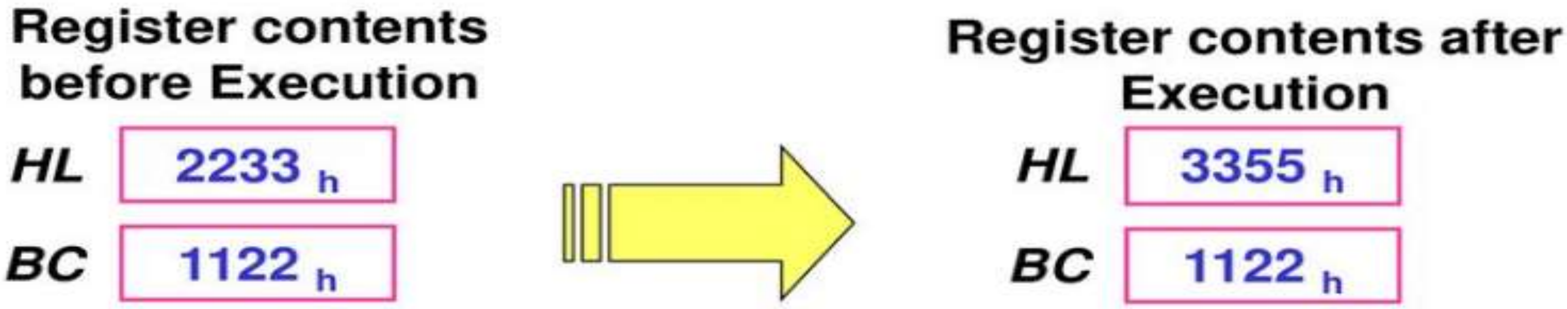
DAD

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

Example

Instruction: **DAD B**



Register contents before Execution

HL 2233_h

BC 1122_h

Register contents after Execution

HL 3355_h

BC 1122_h

Note: No flags are affected except Carry Flag.

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	

SUBC

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

10

2050

SUBM

A=A-M

AFTER EXECUTION

A	04		
B		C	
D		E	
H	20	L	50

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.

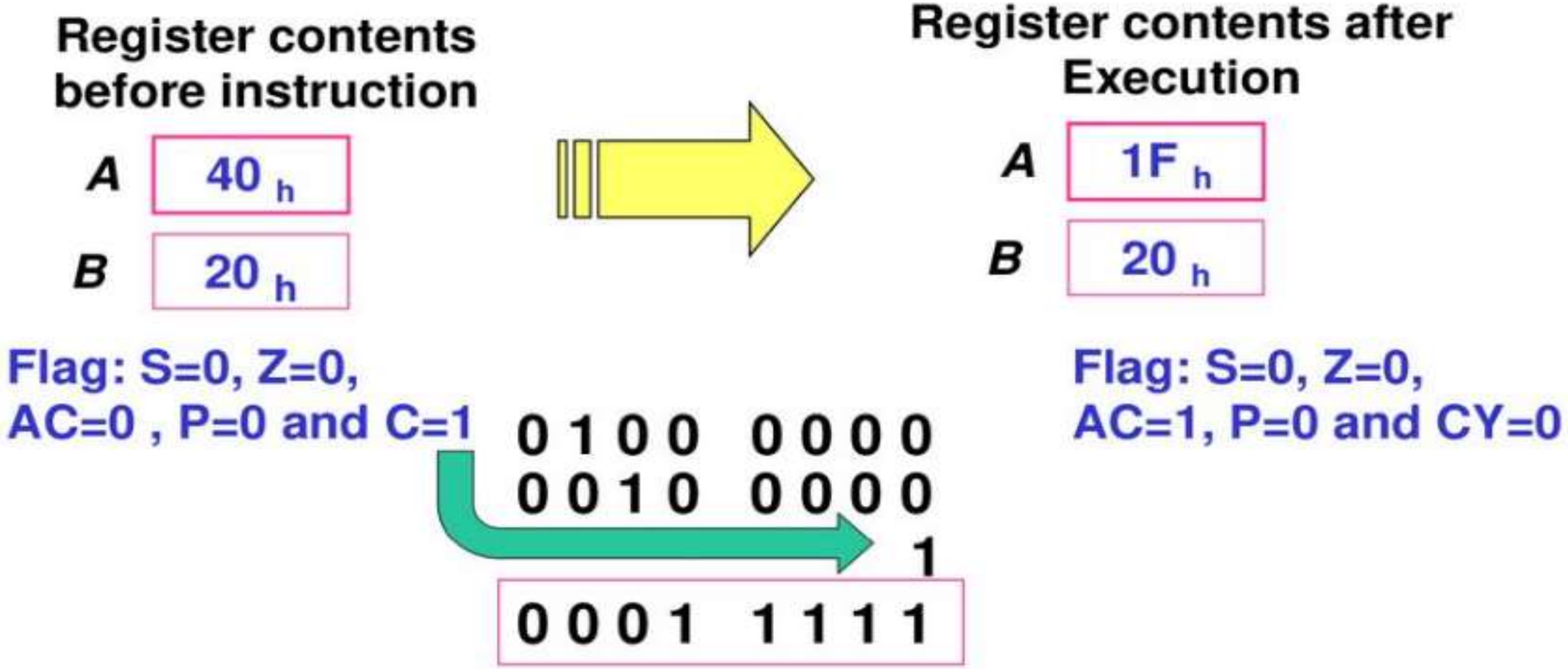
- If the operand is memory location, its address is specified by H-L pair.

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

- **Example:** SBB B or SBB M

Example

Instruction: **SBB B**



Register contents before instruction

A	40_h
B	20_h

Flag: S=0, Z=0, AC=0 , P=0 and C=1

Register contents after Execution

A	$1F_h$
B	20_h

Flag: S=0, Z=0, AC=1, P=0 and CY=0

```
 0 1 0 0   0 0 0 0
 0 0 1 0   0 0 0 0
                 1
 0 0 0 1   1 1 1 1
```

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 05H

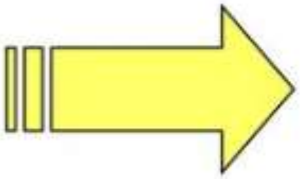
Example

Instruction: **SUI 13h**

Register contents before Execution

A 05 h

Flag: S=0, Z=0, AC=0 , P=0 and CY=0



Register contents after Execution

A F2 h

Flag: S=1, Z=0, AC=0, P=0 and CY=1

```
0 0 0 0   0 1 0 1
0 0 0 1   0 0 1 1
1 1 1 1   0 0 1 0
```

SBI

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

Example

Instruction: **SBI 13h**

Register contents before Execution

A 18 h

Flag: S=0, Z=0,
AC=0 , P=0 and C=1

Register contents after Execution

A 04 h

Flag: S=0, Z=0,
AC=0, P=0 and CY=0

```
0 0 0 1   1 0 0 0
0 0 0 1   0 0 1 1
                1
0 0 0 0   0 1 0 0
```

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.

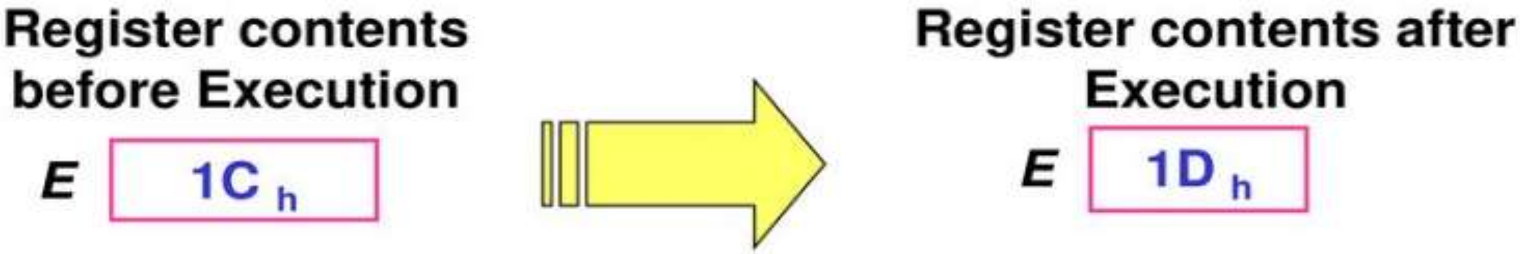
□ The result is stored in the same place.

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

□ **Example:** INR B or INR M

Example

Instruction: **INR E**



Register contents before Execution

E $1C_h$

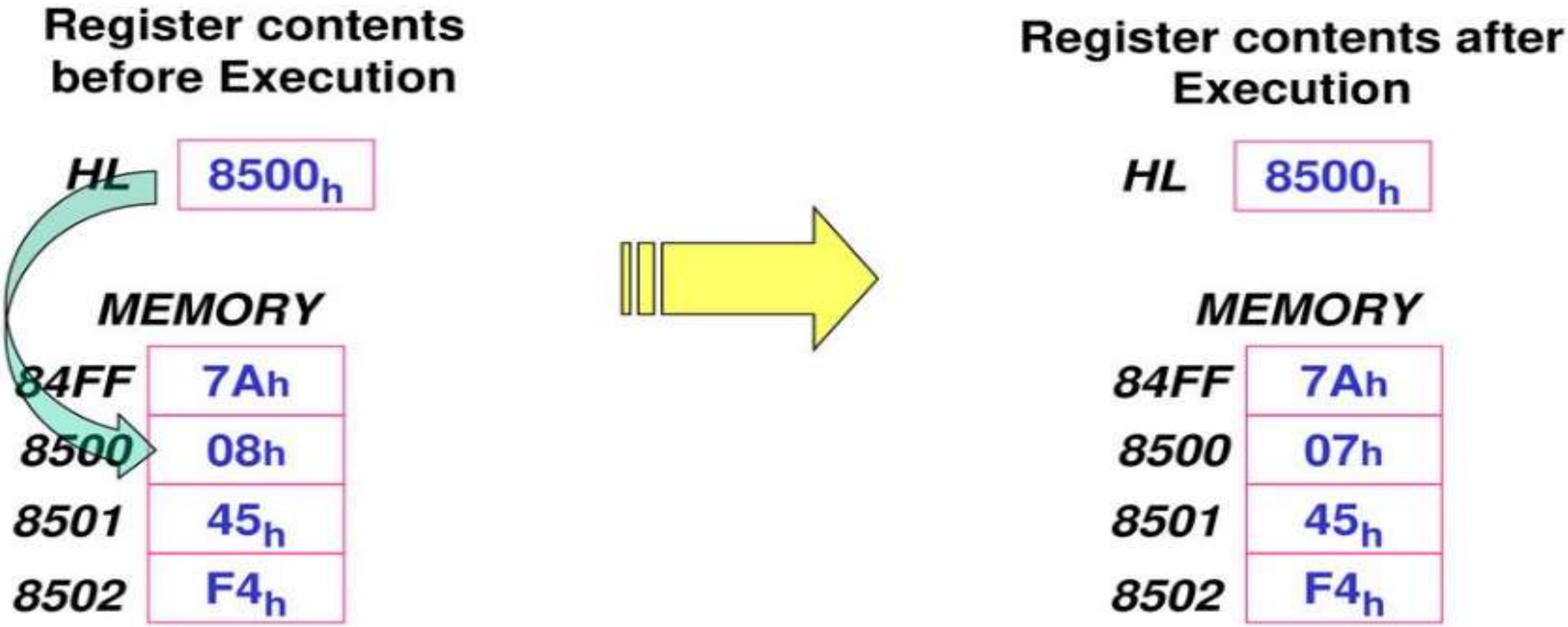
Register contents after Execution

E $1D_h$

Note: Except Carry Flag, all flags are affected depend upon the result.

Example

Instruction: **DCR M**



Register contents before Execution

HL 8500_h

MEMORY

84FF	7Ah
8500	08h
8501	45_h
8502	$F4_h$

Register contents after Execution

HL 8500_h

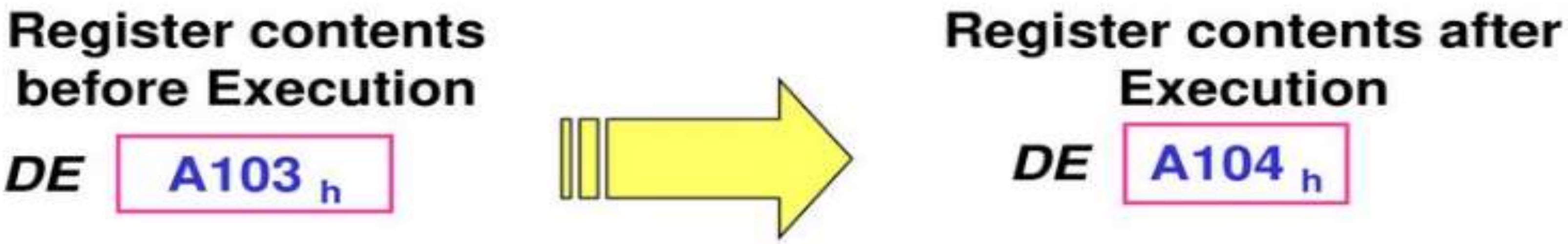
MEMORY

84FF	7Ah
8500	07h
8501	45_h
8502	$F4_h$

Note: Except Carry Flag, all flags are affected depend upon the result.

Example

Instruction: **INX D**



Register contents before Execution

DE A103$_h$

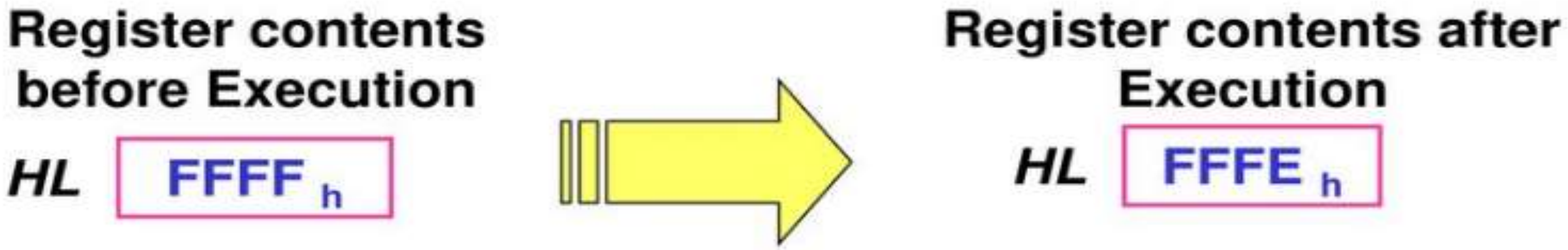
Register contents after Execution

DE A104$_h$

Note: No Flags are affected.

Example

Instruction: **DCX H**



Register contents before Execution

HL FFFF $_h$

Register contents after Execution

HL FFFE $_h$

Note: No Flags are affected.

Example

Instruction: **DAA**

Register contents before Execution

A 6C $_h$

Flag: S=0, Z=0, AC=0 , P=0 and CY=0

```
0 1 1 0   1 1 0 0
0 0 0 0   0 1 1 0
0 1 1 1   0 0 1 0
```

Register contents after Execution

A 72 $_h$

Flag: S=0, Z=0, AC=1 , P=1 and CY=0

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

AND, OR,XOR

- Any 8-bit data, or the contents of register, or memory location can logically have
  - AND operation
  - OR operation
  - XOR operation

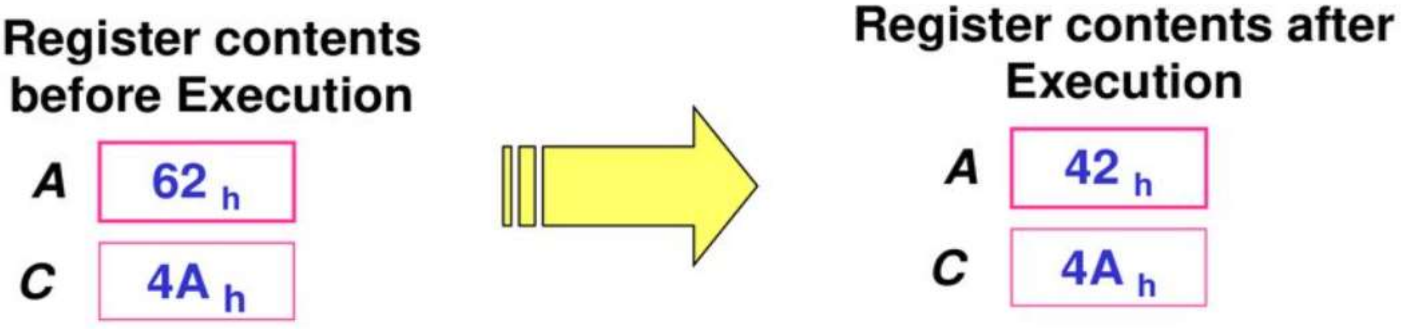
  with the contents of accumulator.
- The result is stored in accumulator.

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.

Example

Instruction: **ANA C**



Register contents before Execution

A 62 h

C 4A h

Register contents after Execution

A 42 h

C 4A h

Flag: S=0, Z=0, AC=0 , P=0 and CY=0

```
0110 0010
0100 1010
0100 0010
```

Flag: S=0, Z=0, AC=1 , P=1 and CY=0

Note: S, Z, P flags are affected during the execution of AND instruction. CY=0 AC = 1

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.

Example

Instruction: **ANI 15h**

Register contents before Execution

A 62_h

Flag: S=0, Z=0, AC=0 , P=0 and CY=0

Register contents after Execution

A 00_h

Flag: S=0, Z=1, AC=1 , P=1 and CY=0

0 1 1 0 0 0 1 0

0 0 0 1 0 1 0 1

$\boxed{0\,0\,0\,0\quad 0\,0\,0\,0}$

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

Example

Instruction: ORA C

Register contents before Execution

A 62_h

C $4A_h$

Flag: S=0, Z=0, AC=0 , P=0 and CY=0

Register contents after Execution

A $6A_h$

C $4A_h$

Flag: S=0, Z=0, AC=0 , P=1 and CY=0

```
0 1 1 0   0 0 1 0
0 1 0 0   1 0 1 0
0 1 1 0   1 0 1 0
```

Note: S, Z, P flags are affected during the execution of OR instruction. CY=0 AC = 0

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.

Example

Instruction: **ORI 15h**

Register contents before Execution

A 62 h

Register contents after Execution

A 77 h

Flag: S=0, Z=0, AC=0 , P=0 and CY=0

```
0 1 1 0   0 0 1 0
0 0 0 1   0 1 0 1
0 1 1 1   0 1 1 1
```

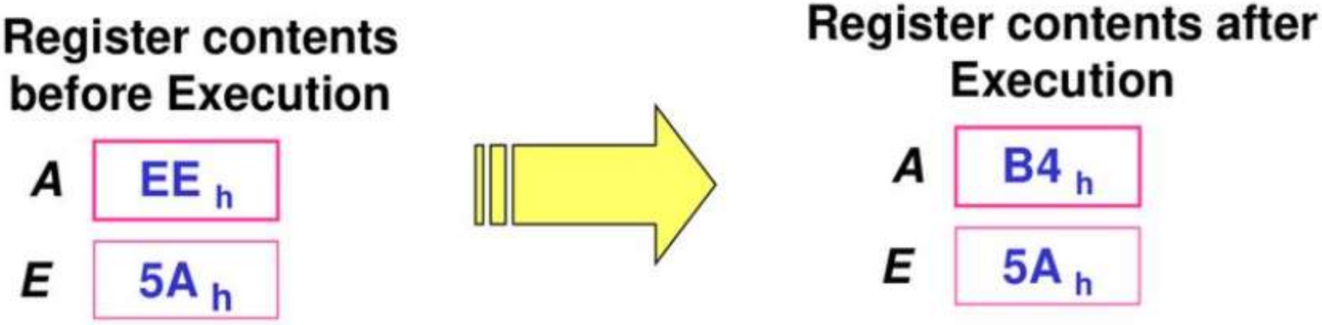
Flag: S=0, Z=0, AC=0 , P=1 and CY=0

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.

Example

Instruction: **XRA E**



Register contents before Execution

A EE h

E 5A h

Register contents after Execution

A B4 h

E 5A h

Flag: S=0, Z=0, AC=0 , P=0 and CY=0

```
1 1 1 0  1 1 1 0
0 1 0 1  1 0 1 0
1 0 1 1  0 1 0 0
```

Flag: S=0, Z=0, AC=0 , P=1 and CY=0

Note: S, Z, P flags are affected during the execution of XOR instruction. CY=0 AC = 0

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 86H.

Example

Instruction: **XRI 18h**

Register contents before Execution

A | C3 h

Register contents after Execution

A | DB h

Flag: S=0, Z=0, AC=0 , P=0 and CY=0

1 1 0 0 0 0 1 1
0 0 0 1 1 0 0 0
$\boxed{1\,1\,0\,1\quad 1\,0\,1\,1}$

Flag: S=0, Z=0, AC=0 , P=1 and CY=0

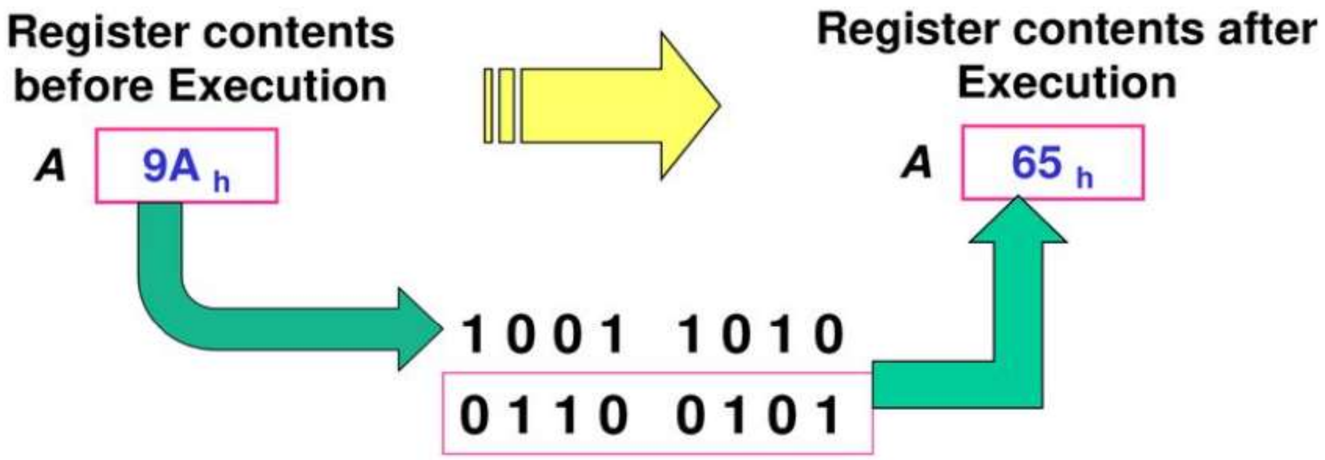
Complement

Opcode	Operand	Description
CMA	None	Complement accumulator

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

Example

Instruction: **CMA**



Register contents before Execution

A 9A h

Register contents after Execution

A 65 h

1 0 0 1 1 0 1 0

0 1 1 0 0 1 0 1

Note: No Flags are affected.

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.

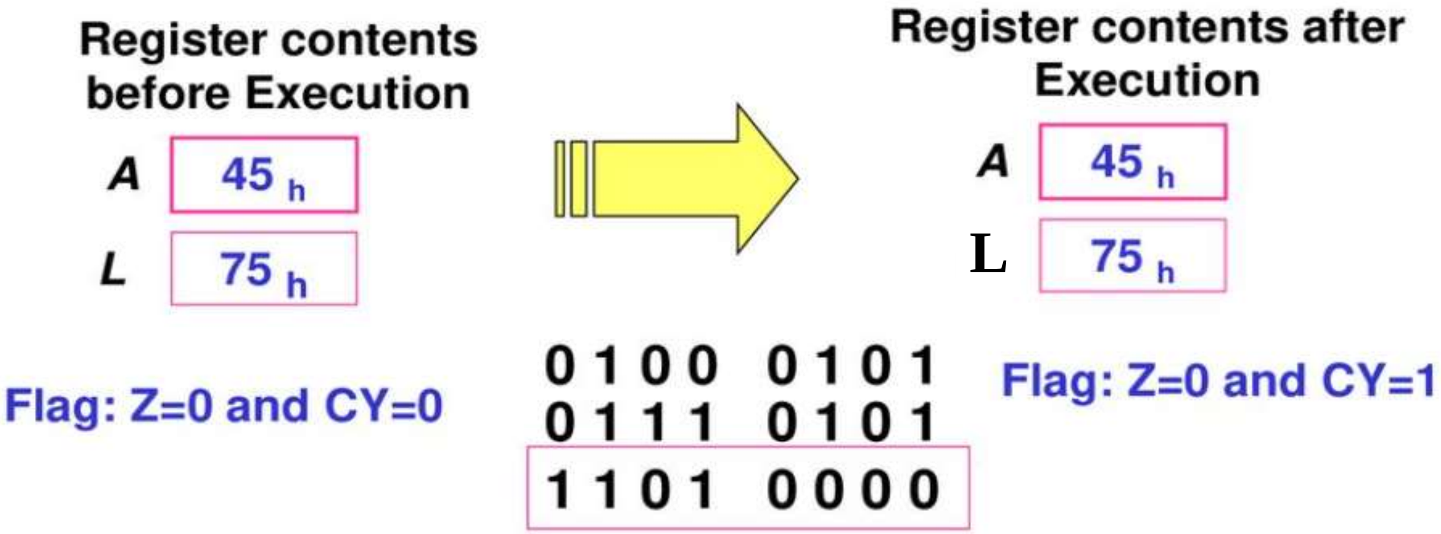
Opcode	Operand	Description
CMP	R M	Compare register or memory with accumulator

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

- Both contents are preserved .

Example

Instruction: **CMP L**



Register contents before Execution

A 45 h

L 75 h

Flag: Z=0 and CY=0

Register contents after Execution

A 45 h

L 75 h

Flag: Z=0 and CY=1

0 1 0 0 0 1 0 1
0 1 1 1 0 1 0 1
1 1 0 1 0 0 0 0

Note: Besides Z and CY, All other Flags are also affected.

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

Example

Instruction: **CPI 45h**

Register contents before Execution

A 62 h

Flag: Z=0 and CY=0

Register contents after Execution

A 62 h

Flag: Z=0 and CY=0

$$\begin{array}{l} 0110\ \ 0010 \\ 0100\ \ 0101 \\ \boxed{0001\ \ 1101} \end{array}$$

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

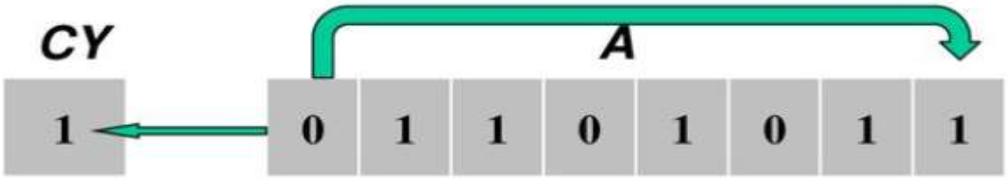
Opcode	Operand	Description
RLC	None	Rotate accumulator left

- ❑ 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.

Example

Instruction: **RLC**

Register contents before Execution



CY	A							
1	0	1	1	0	1	0	1	1

Register contents after Execution

CY	A							
0	1	1	0	1	0	1	1	0

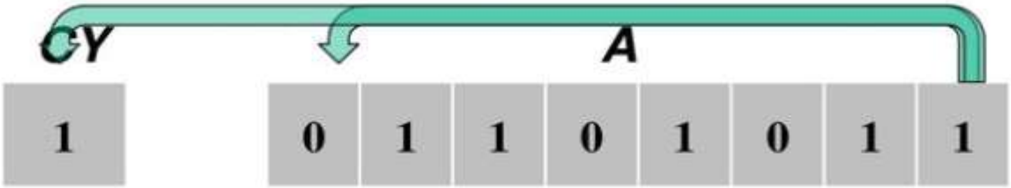
Opcode	Operand	Description
RRC	None	Rotate accumulator right

- 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.

Example

Instruction: **RRC**

Register contents before Execution



Register contents after Execution

CY		A							
1		1	0	1	1	0	1	0	1

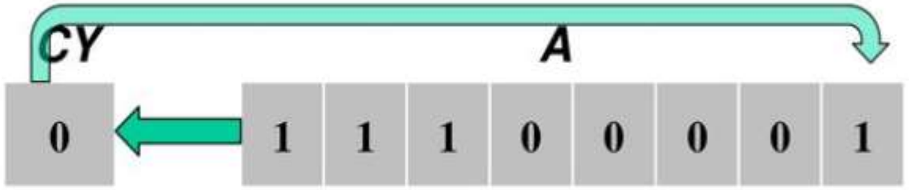
Opcode	Operand	Description
RAL	None	Rotate accumulator left through carry

- 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.

Example

Instruction: **RAL**

Register contents before Execution



CY		A							
0		1	1	1	0	0	0	0	1

Register contents after Execution

CY		A							
1		1	1	0	0	0	0	1	0

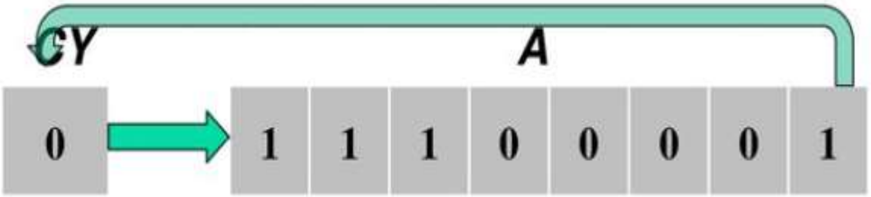
Opcode	Operand	Description
RAR	None	Rotate accumulator right through carry

- 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.

Example

Instruction: **RAR**

Register contents before Execution



CY	A							
0	1	1	1	0	0	0	0	1

Register contents after Execution

CY	A							
1	0	1	1	1	0	0	0	0

Opcode	Operand	Description
CMC	None	Complement carry

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

Opcode	Operand	Description
STC	None	Set carry

- The Carry flag is set to 1.

- No other flags are affected.

- **Example:** STC CF=1

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,

Branching Instructions

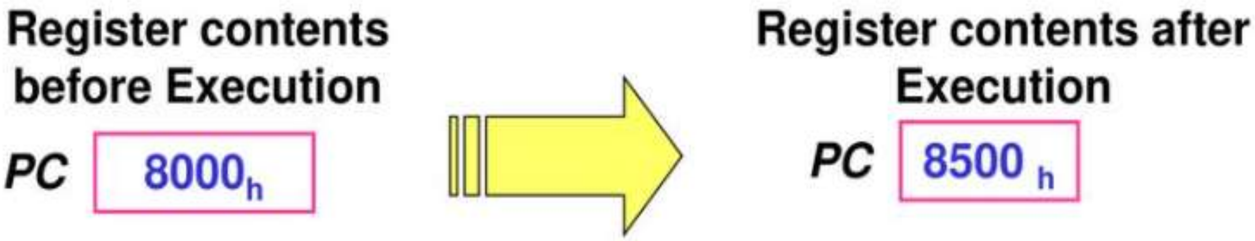
- ❖Jump Unconditionally
- ❖Jump Conditionally
  - •JC, JNC
  - •JP, JM
  - •JPE, JPO
  - •JZ, JNZ
- ❖Call Unconditionally
- ❖Call Conditionally
  - •CC, CNC
  - •CP, CM
  - •CPE, CPO
  - •CZ, CNZ
- ❖Return Unconditionally
- ❖Return Conditionally
- ❖RST
- ❖PCHL

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.

Example

Instruction: **JMP 8500h**



Register contents before Execution

PC 8000_h

Register contents after Execution

PC 8500_h

Note: No Flags are affected.

Opcode	Operand	Description
J x	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

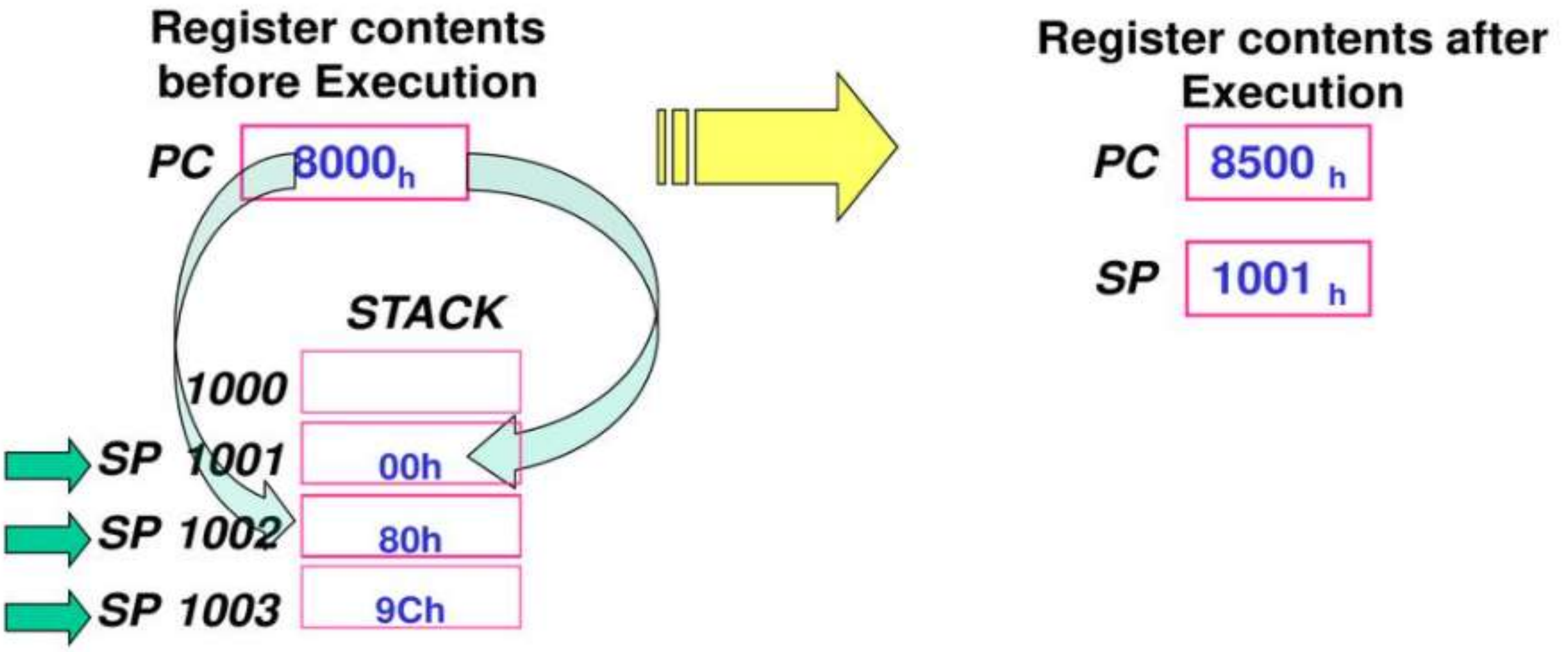
Opcode	Description	Status Flags
JC	Jump if Carry	CY = 1
JNC	Jump if No Carry	CY = 0
JZ	Jump if Zero	Z = 1
JNZ	Jump if No Zero	Z = 0
JPE	Jump if Parity Even	P = 1
JPO	Jump if Parity Odd	P = 0
JP	Jump if positive	S=0
JM	Jump if negative	S=1

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.

Example

Instruction: **CALL 8500h**



Register contents before Execution

PC 8000_h

STACK

1000

SP 1001 00h

SP 1002 80h

SP 1003 9Ch

Register contents after Execution

PC 8500_h

SP 1001_h

Note: No Flags are affected.

Call Conditionally

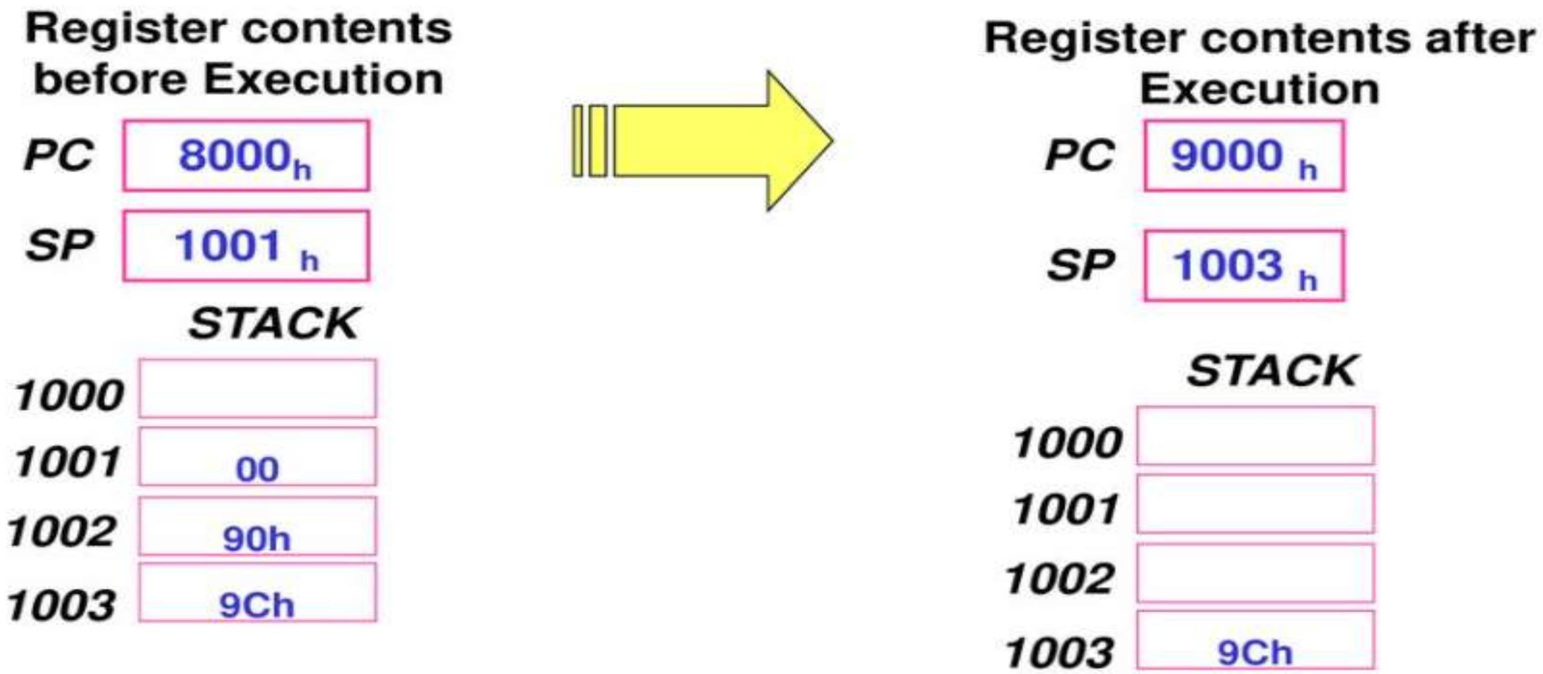
Opcode	Description	Status Flags
CC	Call if Carry	CY = 1
CNC	Call if No Carry	CY = 0
CP	Call if Positive	S = 0
CM	Call if Minus	S = 1
CZ	Call if Zero	Z = 1
CNZ	Call if No Zero	Z = 0
CPE	Call if Parity Even	P = 1
CPO	Call if Parity Odd	P = 0

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.

Example

Instruction: **RET**



Register contents before Execution

PC	8000h
SP	1001 h

STACK

1000	
1001	00
1002	90h
1003	9Ch

Register contents after Execution

PC	9000 h
SP	1003 h

STACK

1000	
1001	
1002	
1003	9Ch

Note: No Flags are affected.

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

STACK,I/O & MACHINE INSTRUCTIONS

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		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

SP		2700	
H	30	L	40

2700H	50
2701H	60
2702H	

XTHL

AFTER EXECUTION

SP		2700	
H	60	L	50

2700H	40
2701H	30
2702H	

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

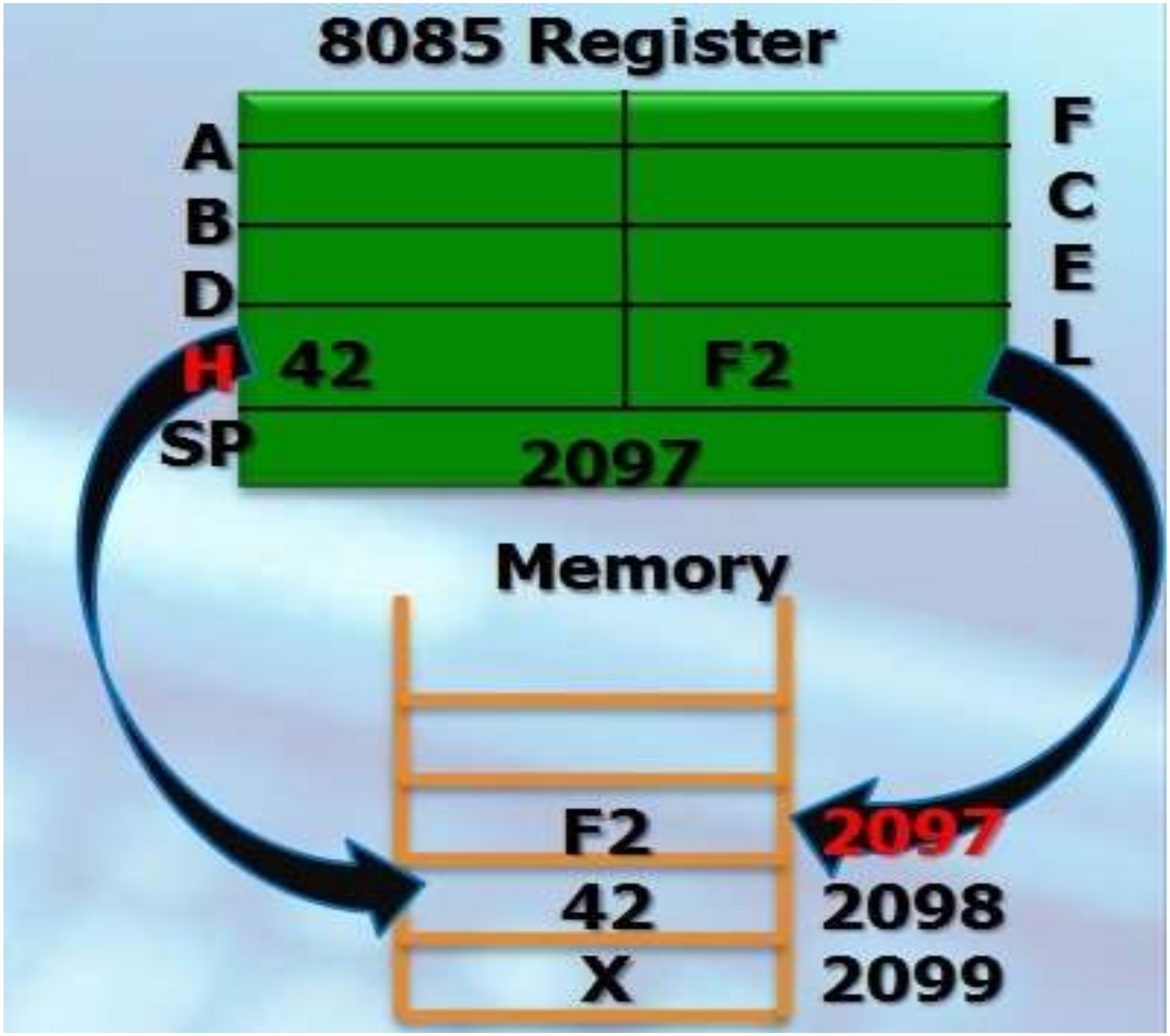
PUSH-Push register pair onto stack

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
F
C
E
L
42
F2
2097
Memory
F2
42
X
2097
2098
2099

POP- Pop stack to register pair

Opcode	Operand
POP	Reg. pair

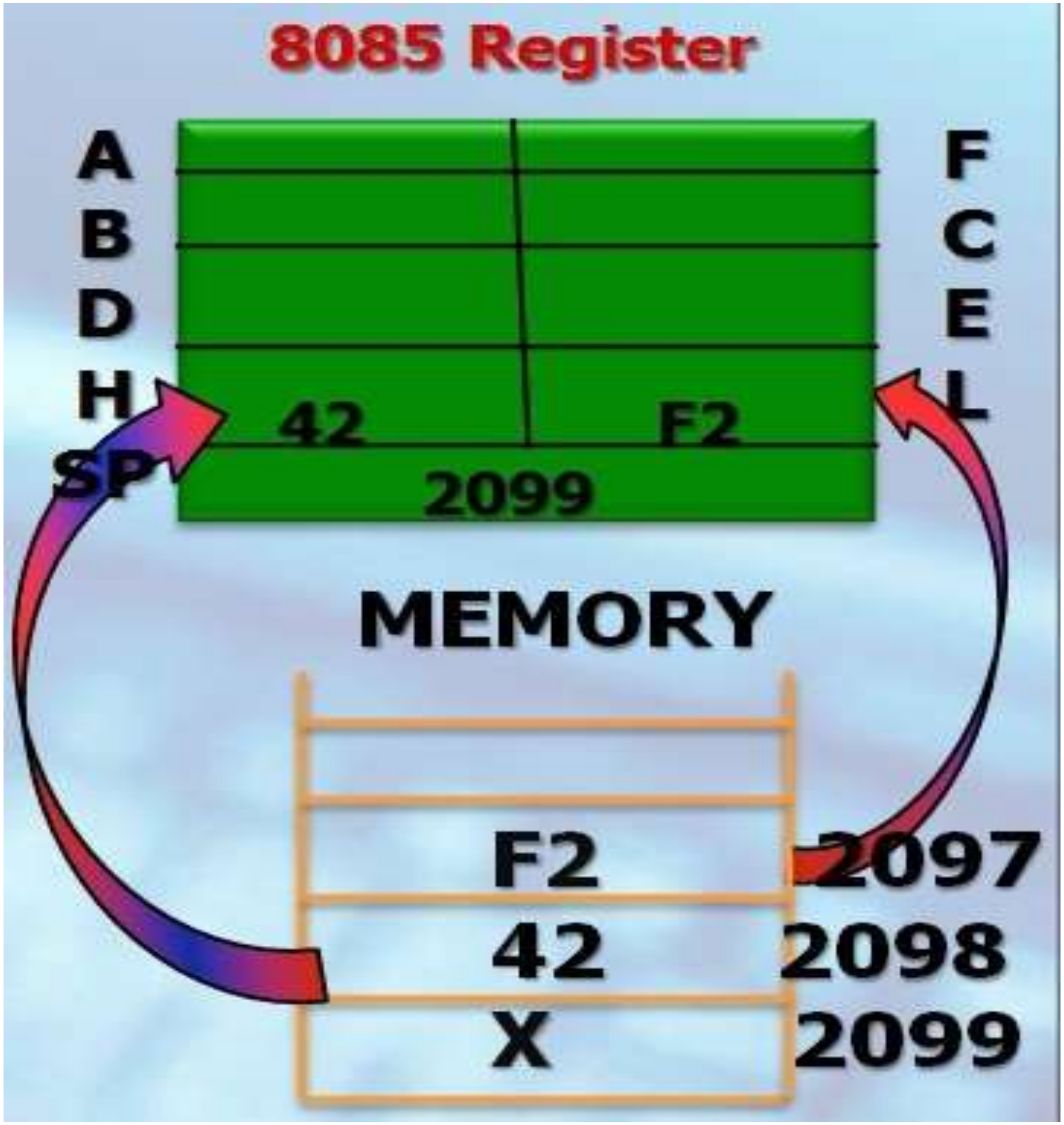
❑The contents of **top of stack** are **copied into register pair**.

❑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
F
B
C
D
E
H
L
42
F2
SP
2099
MEMORY
F2
2097
42
2098
X
2099

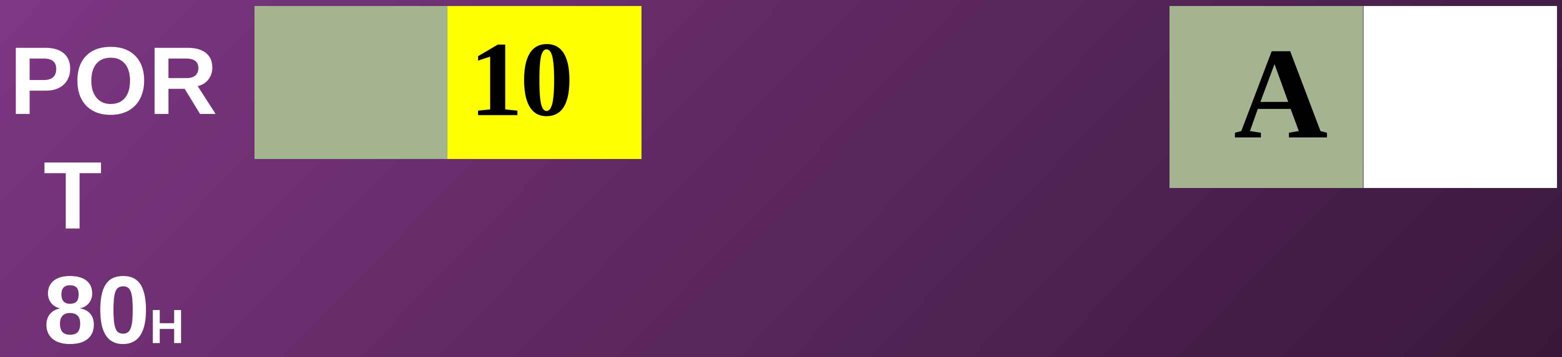
IN- Copy data to accumulator from a port with 8-bit address

Opcode	Operand
IN	8-bit port address

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

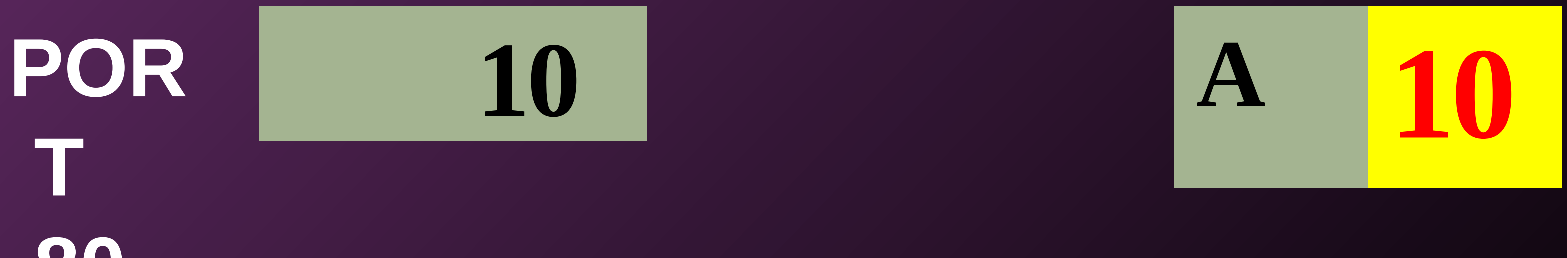
Example: IN 8CH

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 78H

BEFORE EXECUTION

PORT 50H

10

A 40

OUT 50H

AFTER EXECUTION

PORT

40

A 40

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

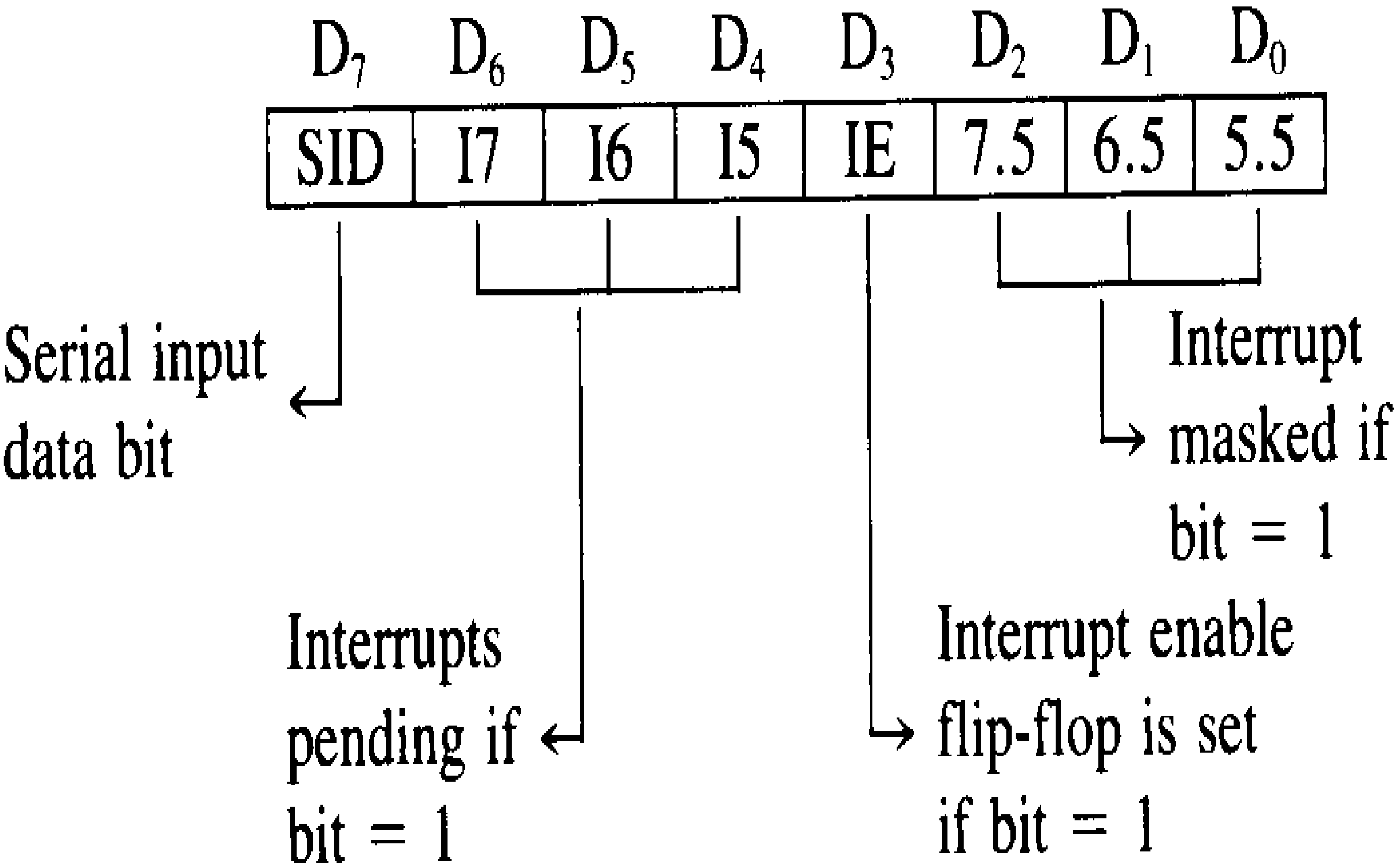
Opcode	Operand	Description
RIM	None	Read Interrupt Mask

- 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



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

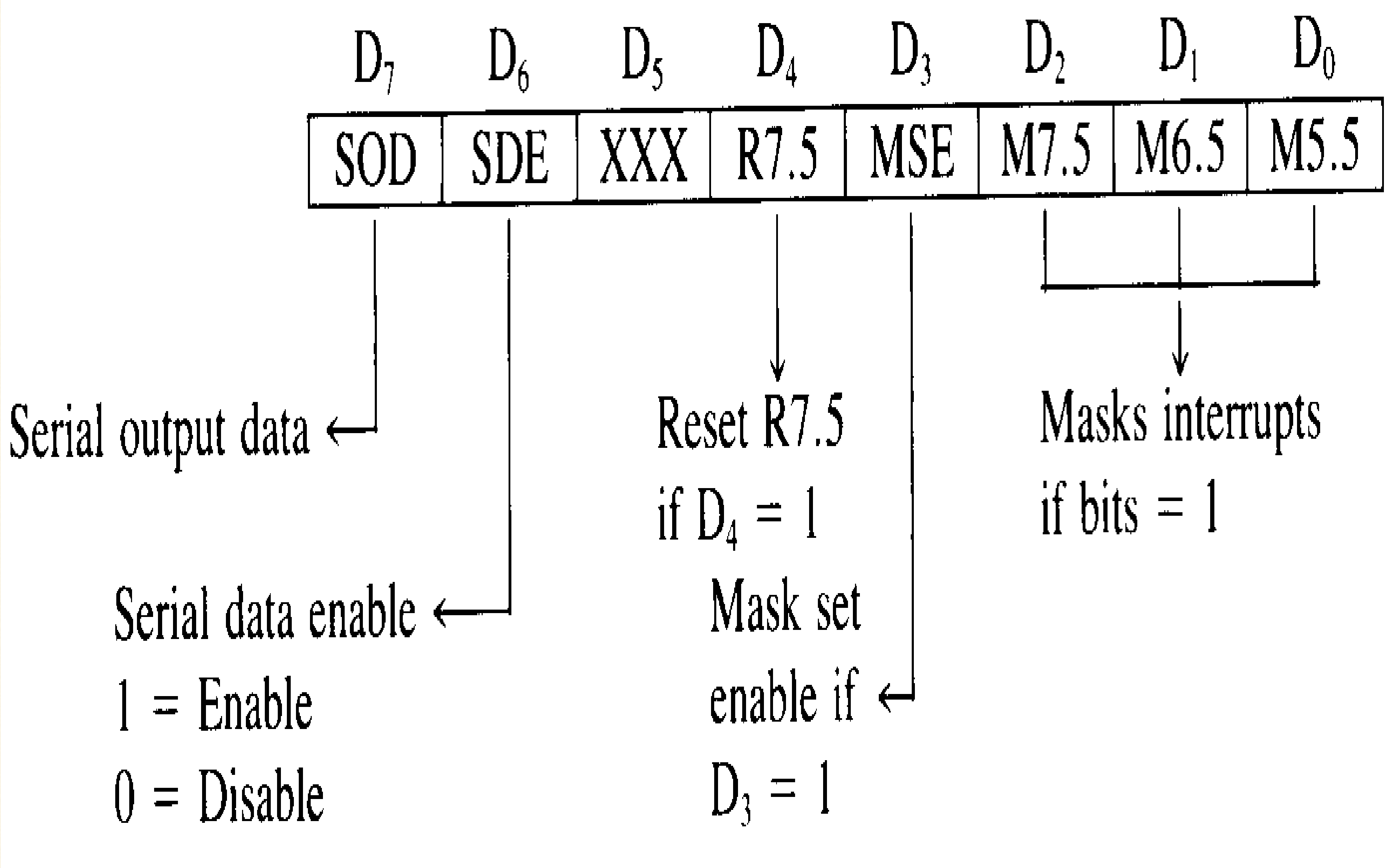
Opcode	Operand	Description
SIM	None	Set Interrupt Mask

- 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



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

Multi Byte Addition Algorithm

1. Clear the carry flag
2. Initialize one register (Reg C) with the length of the data
3. Initialize one register pair (HL) with the starting address of the first data
4. Initialize one register pair (DE) with the starting address of the second data
5. Move the content of DE to accumulator
6. Add the contents of DE and memory with carry
7. Move the accumulator content to memory
8. Increment HL and DE pairs
9. Decrement the counter register (Reg C)
10. Check for zero, if not zero, go to step 5
11. Stop

Multi byte Addition

```
        STC   cy=1
        CMC  cy=0
        LDA      9100       9100= length of the data  =4
        MOV    C, A   c=4
        LXI       H, 9200    HL=9200
        LXI       D, 9300    DE=9300
L1:     LDAX   D     A=(9301)
        ADC      M   A= A+(M)+CY
        MOV    M, A
        INX      D  DE=9304
        INX      H  HL= 9204
        DCR      C   C=0
        JNZ      L1
        HLT
```

02030805
0104121A

Largest value in an array

1. Initialize one register pair (HL) with the starting address of the array
2. Initialize one register (Reg B) with one count less than the length of the array
3. Move the memory content to accumulator
4. Increment HL pair
5. Compare the contents of accumulator and memory
6. Check for carry, if no carry, go to step 8
7. Move the memory content to accumulator
8. Decrement Reg B
9. Check for zero, if not zero, go to step 4
10. Store the accumulator content in a memory location
11. Stop

Largest value in an array

```
LXIH, 9100
    MVIB, 07
    MOV A, M
L2 :INX H
    CMP M
    JNC L1
    MOV A, M
L1 :DCR B
    JNZ  L2
    STA  9200
     RST 1
```