

CSE 3107: Microprocessor

Lecture # 5 Instruction Set of 8085

Sujan Sarker

Lecturer, CSE, AUST



CSE | AUST

Fall 2016

Course Website: <https://sites.google.com/site/sujanaustcse/courses-fall-2016/cse-3107/>

The 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**.
- Since the 8085 is an 8-bit device it can have up to 2^8 (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.
- 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

- There are 5 Types:
 - (1) Data Transfer Instructions
 - (2) Arithmetic Instructions
 - (3) Logical Instructions
 - (4) Branching Instructions
 - (5) Control Instructions

Instruction and Data Format

- 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**”.

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.
- While copying, the contents of source are not modified.
- No flags are affected.

Data Transfer Operations (Cont.)

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

- This instruction copies the contents of the source register into the destination register.
- The contents of the source register are not altered.
- If one of the operands is a memory location, its location is specified by the contents of the HL registers.
- Example: MOV A, D or MOV M, A

BEFORE EXECUTION

A	20	B	
---	----	---	--

MOV B,A

AFTER EXECUTION

A	20	B	20
---	----	---	----

A		F		2047	
B	30	C		2048	
D		E		2049	
H	20	L	50	2050	

MOV M,B

A		F		2047	
B	30	C		2048	
D		E		2049	
H	20	L	50	2050	30

A		F		2047	
B	30	C		2048	
D		E		2049	
H	20	L	50	2050	40

MOV C,M

A		F		2047	
B	30	C	40	2048	
D		E		2049	
H	20	L	50	2050	40

Data Transfer Operations (Cont.)

Opcode	Operand	Description
MVI	Rd, <u>Data</u> M, <u>Data</u>	Move immediate 8-bit

- The 8-bit immediate 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 A, 02H or MVI M, FFH

BEFORE EXECUTION

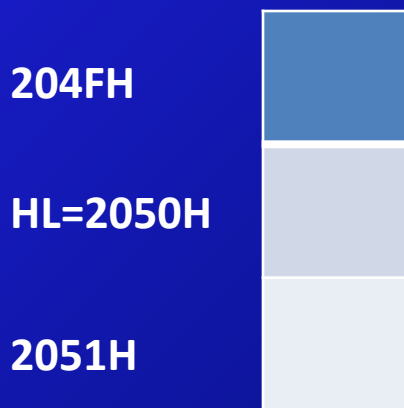
A		F	
B		C	
D		E	
H	20	L	50

MVI B,60H

AFTER EXECUTION

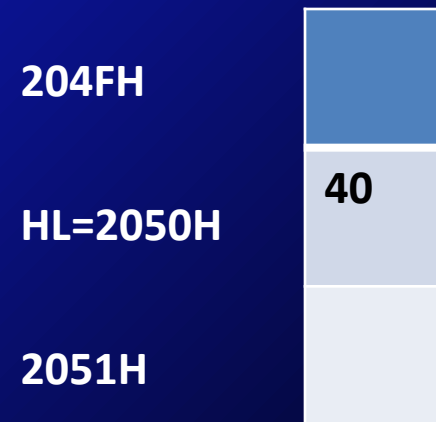
A		F	
B	60	C	
D		E	
H	20	L	50

BEFORE EXECUTION



MVI M,40H

AFTER EXECUTION



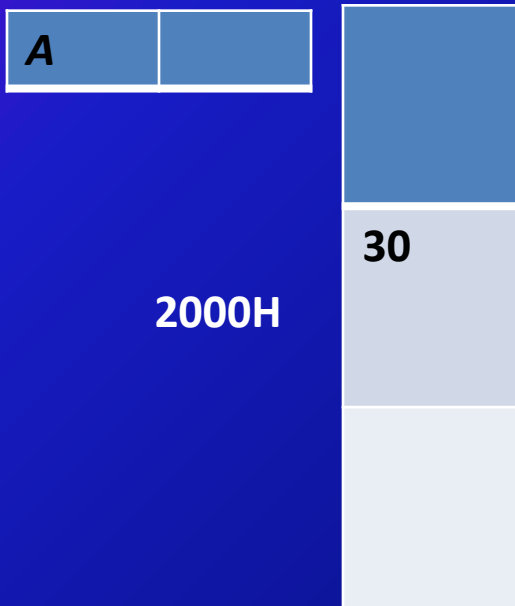
Data Transfer Operations (Cont.)

Opcode	Operand	Description
LDA	16-bit address	Load Accumulator

- The contents of a memory location, specified by a 16-bit address in the operand, are copied to the accumulator.
- The contents of the source are not altered.
- Example: LDA 0005H

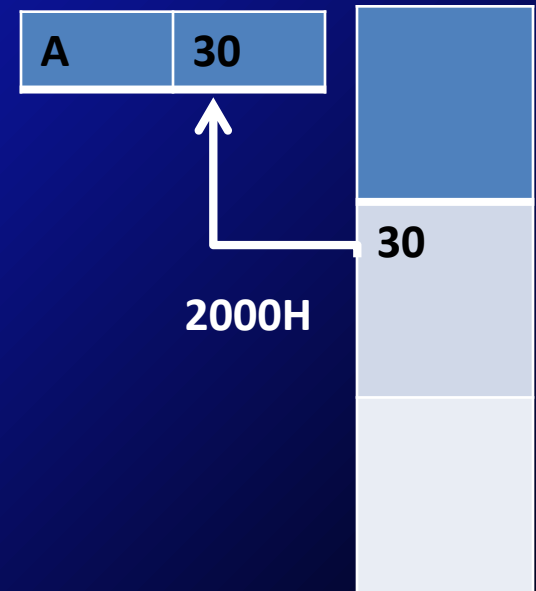
[MOV A, M]

BEFORE EXECUTION



LDA 2000H

AFTER EXECUTION



Data Transfer Operations (Cont.)

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

- 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



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

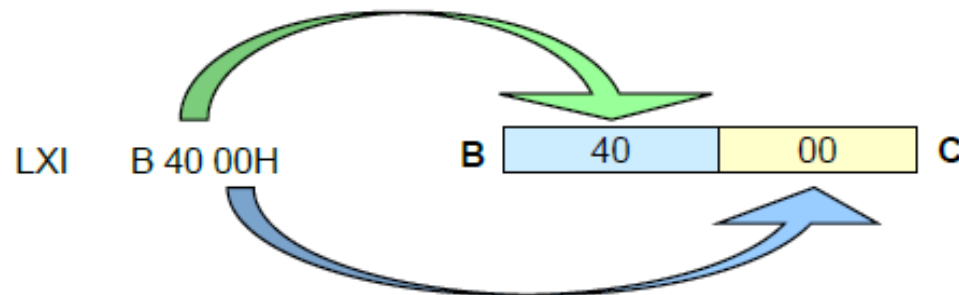
2030H

80

Data Transfer Operations (Cont.)

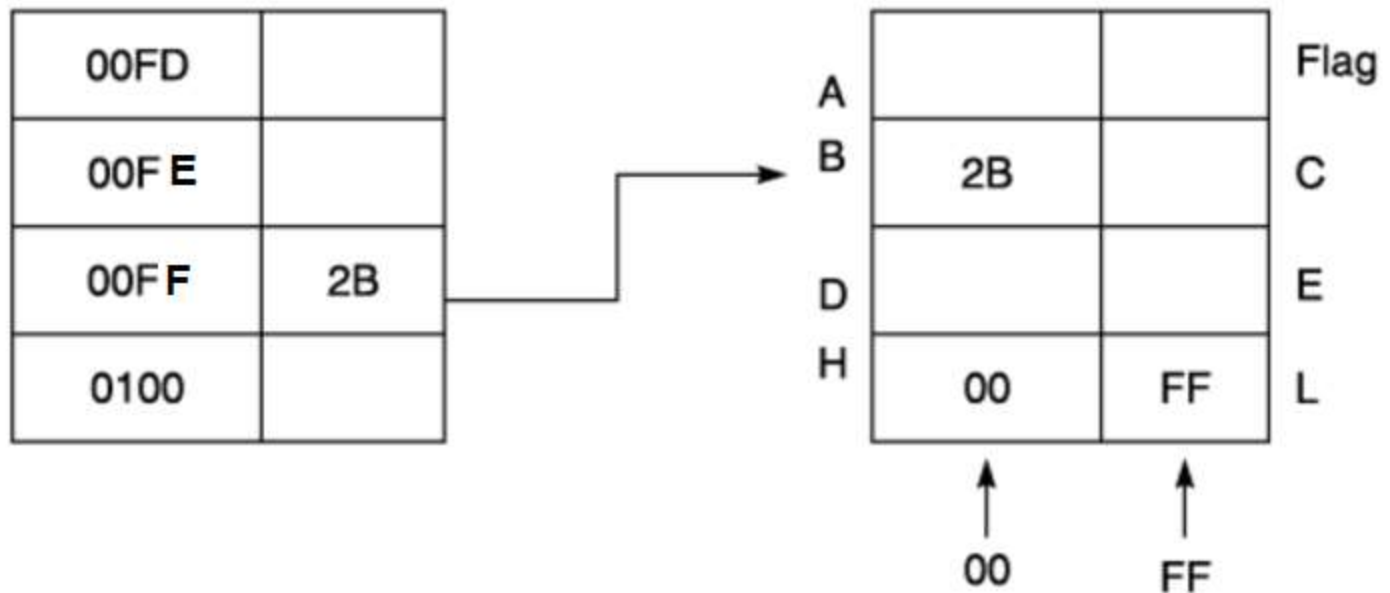
Opcode	Operand	Description
LXI(Load eXtended Immediate)	Reg. pair, 16-address	Load register pair immediate

- The instruction LXI B, 4000H will place the 16-bit number 4000 into the register pair B, C.
 - The upper two digits are placed in the 1st register of the pair and the lower two digits in the 2nd.




Data Transfer Operations (Cont.)

LXI H, 00FF
MOV B, M



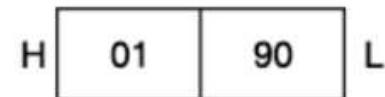
Data Transfer Instructions (Cont.)

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

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

LHLD 2050 H

2050	90
2051	01



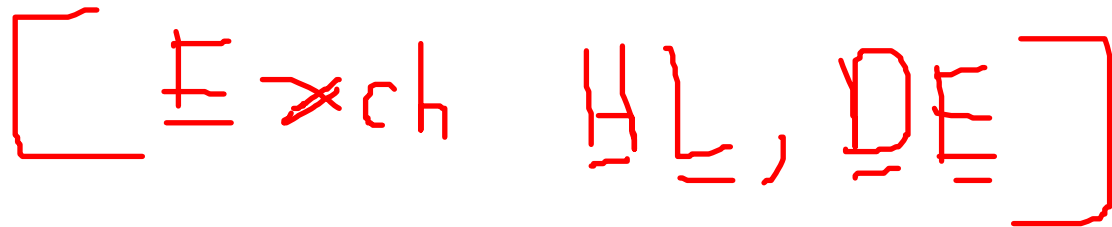
1) MOV L, M;
2) MOV H, M+1

Data Transfer Instructions (Cont.)

Opcode	Operand	Description
XCHG	None	Exchange H-L with D-E

- The contents of register H are exchanged with the contents of register D.
- The contents of register L are exchanged with the contents of register E.

• Example: XCHG



Data Transfer Instructions (Cont.)

LXI H, 0A2B

LXI D, 0A2D

MOV A, M

XCHG

ADD M

Memory

0A2A	0A
0A2B	08
0A2D	10
0A2E	11

ALU
08
+ 10
—
+ 18

D	0A	2D	E
H	0A	2B	L

XCHG

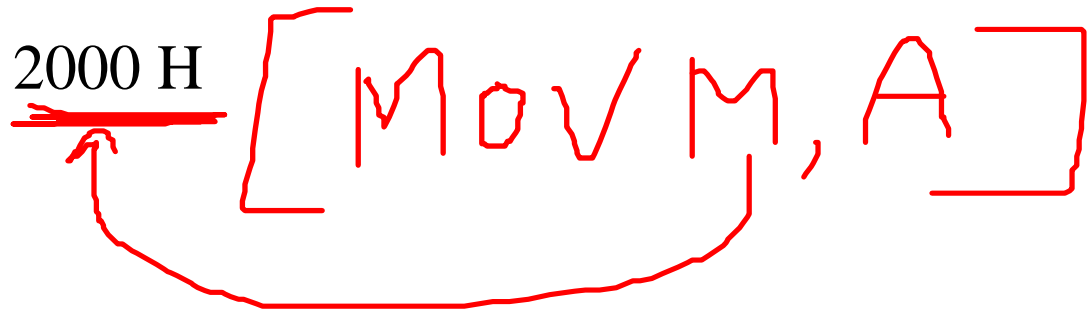
D	0A	2B	E
H	0A	2D	L

A = 08

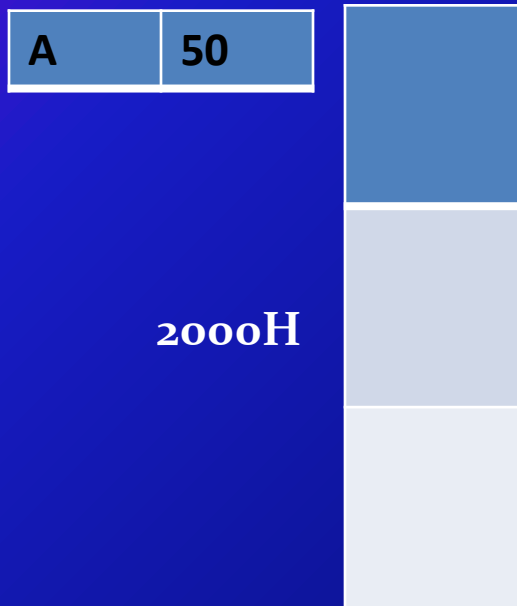
Data Transfer Instructions (Cont.)

Opcode	Operand	Description
STA	16-bit address	Store the contents of accumulator into memory location i.e. address specified by the operand in the instruction.

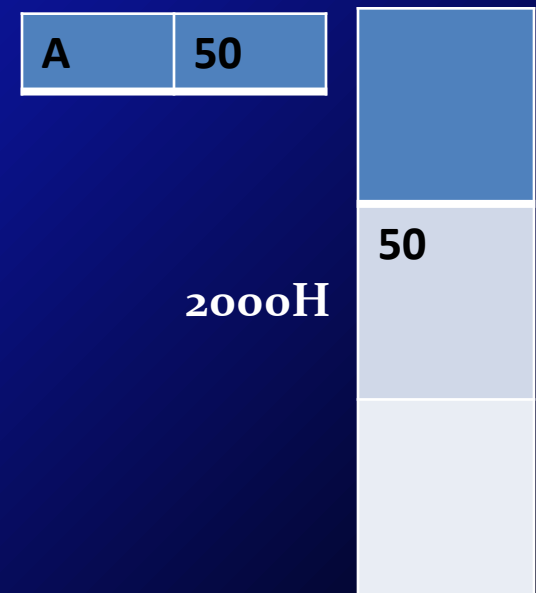
- Example: STA 2000 H



BEFORE EXECUTION



AFTER EXECUTION



STA 2000H

Data Transfer Instructions (Cont.)

Opcode	Operand	Description
STAX	Register Pair	Store the contents of accumulator into memory location whose address is specified by <u>BC</u> or <u>DE</u> register pair.

- Example: STAX B

[MOV ~~MA~~]

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

Data Transfer Instructions (Cont.)

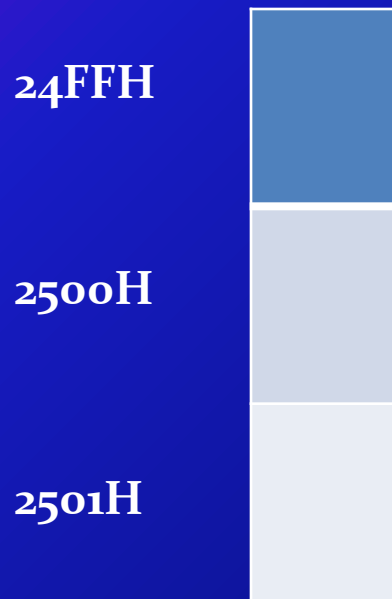
Opcode	Operand	Description
SHLD 	16-bit address	Store H-L register pair in memory.

- 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

1) MOV M, L
2) MOV M+1, H

BEFORE EXECUTION

H	30	L	60
---	----	---	----



AFTER EXECUTION

H	30	L	60
---	----	---	----

SHLD 2500H



Data Transfer Instructions (Cont.)

Opcode	Operand	Description
SPHL	None	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

Data Transfer Instructions (Cont.)

Opcode	Operand	Description
XTHL	None	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
 1) EXCH L, SP
 2) EXCH H, SP+1

$L=SP$
 $H=(SP+1)$

BEFORE EXECUTION

SP	2700		
H	30	L	40

2700H

50

2701H

60

2702H

--

AFTER EXECUTION

SP	2700		
H	60	L	50

2700H

40

2701H

30

2702H

--

XTHL

Data Transfer Instructions (Cont.)

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 

BEFORE EXECUTION

PC			
H	60	L	00

PCHL

AFTER EXECUTION

PC	6000		
H	60	L	00

Data Transfer Instructions (Cont.)

Opcode	Operand	Description
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 78 H

Data Transfer Instructions (Cont.)

Opcode	Operand	Description
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 8C H

Arithmetic Instructions

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

Addition

- Any 8-bit number, or the contents of register, or the contents of memory location can be added to the contents of accumulator.
- The result (sum) is stored in the accumulator.
- 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.
- 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.

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.
- 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.
- 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.

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.
- No need to disturb the contents of the accumulator.

Arithmetic Instructions (Cont.)

Opcode	Operand	Description
ADD	R M	<ul style="list-style-type: none"> The contents of register or memory are added to the contents of accumulator.

ADD B

A

19

B

32

$$19 + 32 = 4B$$

ADD M

A

4B

1431	2B
1432	68
1433	4E

H

14

L

32

$$4B + 68 = B3$$

Arithmetic Instructions (Cont.)

Opcode	Operand	Description
ADI	Data[8-bit]	• Add immediate data[8-bit] with accumulator.

ADI 34H

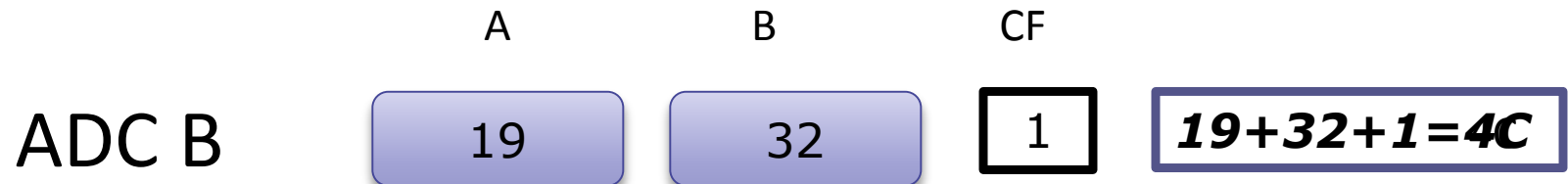
A

B3

B3 + 34H = EFH

Arithmetic Instructions (Cont.)

Opcode	Operand	Description
ADC	r M	<ul style="list-style-type: none">The contents of register or memory and Carry Flag (CY) are added to the contents of accumulator.



Arithmetic Instructions (Cont.)

Opcode	Operand	Description
ACI	Data[8-bit]	•Add with carry immediate data to accumulator

ACI 25H

A

19

CF

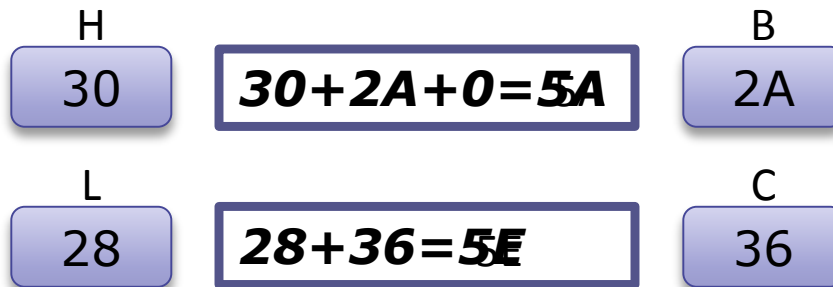
1

$19 + 25 + 1 = 3F$

Arithmetic Instructions (Cont.)

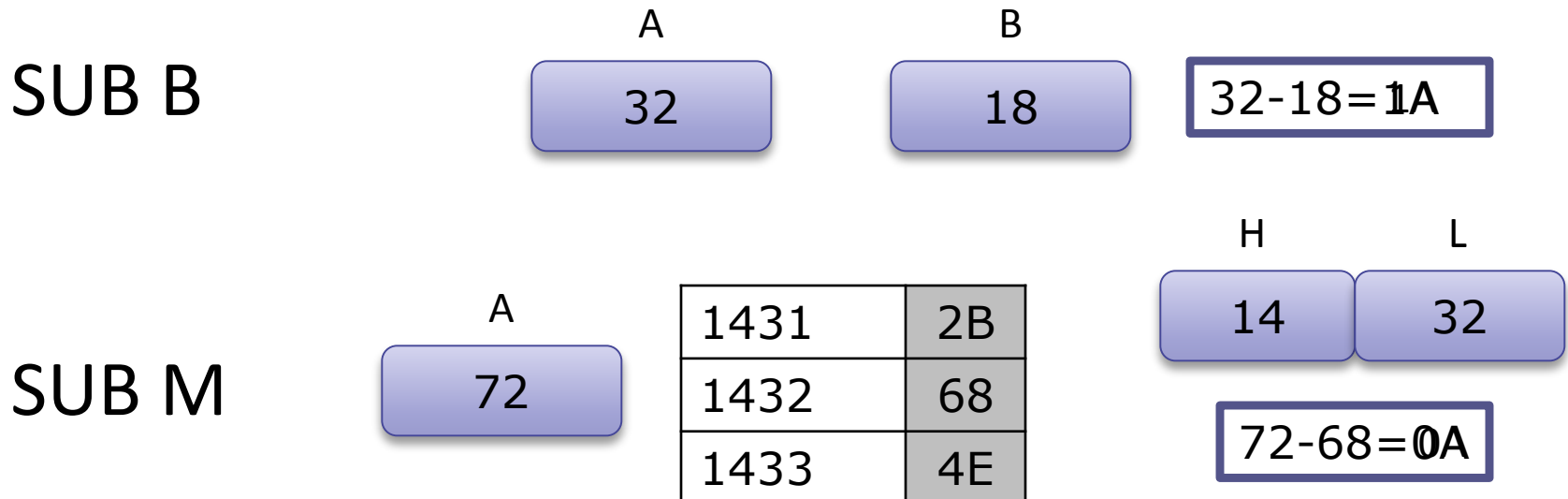
Opcode	Operand	Description
DAD	rp	<ul style="list-style-type: none">•Add register pair with [H-L] pair•If the result is larger than 16 bits, then CY is set.

DAD B



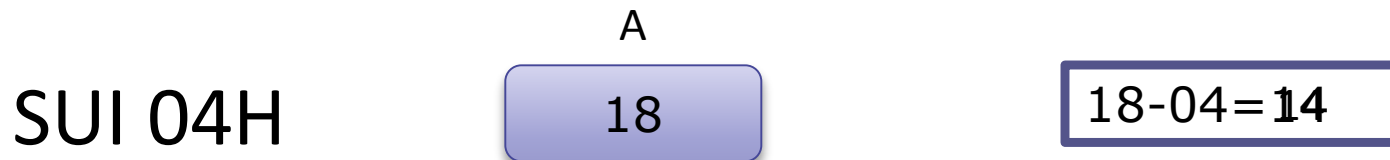
Arithmetic Instructions (Cont.)

Opcode	Operand	Description
SUB	R M	Subtract register from accumulator



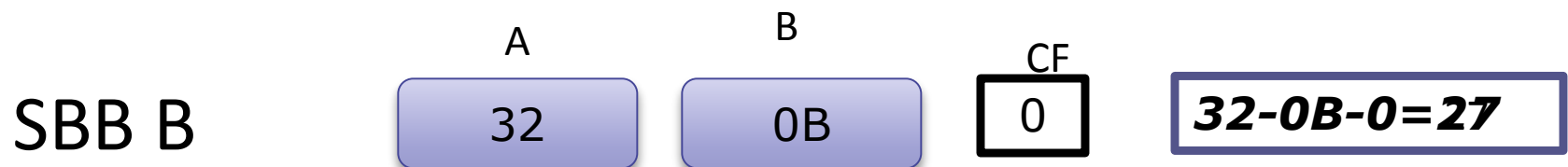
Arithmetic Instructions (Cont.)

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



Arithmetic Instructions (Cont.)

Opcode	Operand	Description
SBB	r M	Subtract register from accumulator with borrow



Arithmetic Instructions (Cont.)

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

SBI 0EH A CF **$19 - 0E - 1 = 0A$**

19 1

Arithmetic Instructions (Cont.)

- The 8085 performs the following steps internally to execute the instruction SUB (or SUI):
- Converts subtrahend (the number to be subtracted) into its 1's complement.
- Adds 1 to 1's complement to obtain 2's complement of the subtrahend.
- Add 2's complement to the minuend (the contents of the accumulator).
- Complements the Carry flag.

Step 1:

$$\begin{array}{r} 39H = 0011\ 1001 \\ \text{1's complement of } 39H = 1100\ 0110 \end{array}$$

Step 2:

$$\begin{array}{r} + \\ \text{Add } 01 = 0000\ 0001 \\ \text{2's complement of } 39H = 1100\ 0111 \end{array}$$

Step 3:

$$\begin{array}{r} + \\ \text{Add } 30H \text{ to 2's complement of } 39H = 0011\ 0000 \\ \text{CY } \boxed{0} \quad 1111\ 0111 \end{array}$$

Step 4: Complement carry

$$\boxed{1} \quad 1111\ 0111 = F7H$$

Flag Status: S = 1, Z = 0, CY = 1

Arithmetic Instructions (Cont.)

Opcode	Operand	Description
INR	r M	Increment content of register or memory

INR B

B

0B

$0B + 1 = 0C$

Arithmetic Instructions (Cont.)

Opcode	Operand	Description
DCR	r M	Decrement content of register or memory

DCR B

^B
0B

0B-1=0A^A

Arithmetic Instructions (Cont.)

Opcode	Operand	Description
INX	rp	Increment the content of register pair

INX B

B

C

12

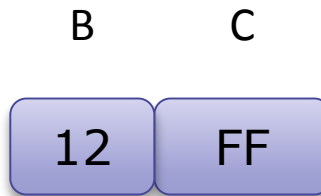
FF

$12FF + 1 = 1300$

Arithmetic Instructions (Cont.)

Opcode	Operand	Description
DCX	Rp	Decrement the content of register pair

DCX B



12FF-1=12FE

Logical Instruction

- These instructions perform logical operations on data stored in registers, memory and status flags and the result is stored on accumulator.
- The logical operations are:
 - AND
 - OR
 - XOR
 - Rotate
 - Compare
 - Complement

Logical Instruction (Cont.)

Opcode	Operand	Description
ANA	r M	AND register with accumulator

ANA L L A

 0B 1E

0B	0	0	0	0	1	0	1	1
1E	0	0	0	1	1	1	1	0
0A	0	0	0	0	1	0	1	0

Opcode	Operand	Description
ANI	Data[8-bit]	AND immediate data with accumulator

ANI 0B A

 1E

0B	0	0	0	0	1	0	1	1
1E	0	0	0	1	1	1	1	0
0A	0	0	0	0	1	0	1	0

Logical Instruction (Cont.)

Opcode	Operand	Description
ORA	r M	OR register / memory with accumulator
ORI	Data[8-bit]	OR immediate data with accumulator
XRA	r M	EXCLUSIVE-OR register / memory with accumulator
XRI	Data[8-bit]	EXCLUSIVE-OR immediate data with accumulator
CMC		Compliment the carry status
STC		Set carry status
✓ CMA		Complement the accumulator

CMA

A

2B

2B	0	0	1	0	1	0	1	1
D4	1	1	0	1	0	1	0	0

Logical Instruction (Cont.)

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

- if $(A) < (\text{reg/mem})$: carry flag is set
- if $(A) = (\text{reg/mem})$: zero flag is set
- if $(A) > (\text{reg/mem})$: carry and zero flags are reset.
- Example: CMP B or CMP M

Logical Instruction (Cont.)

Opcode	Operand	Description
CPI	8-bit data	Compare immediate with accumulator

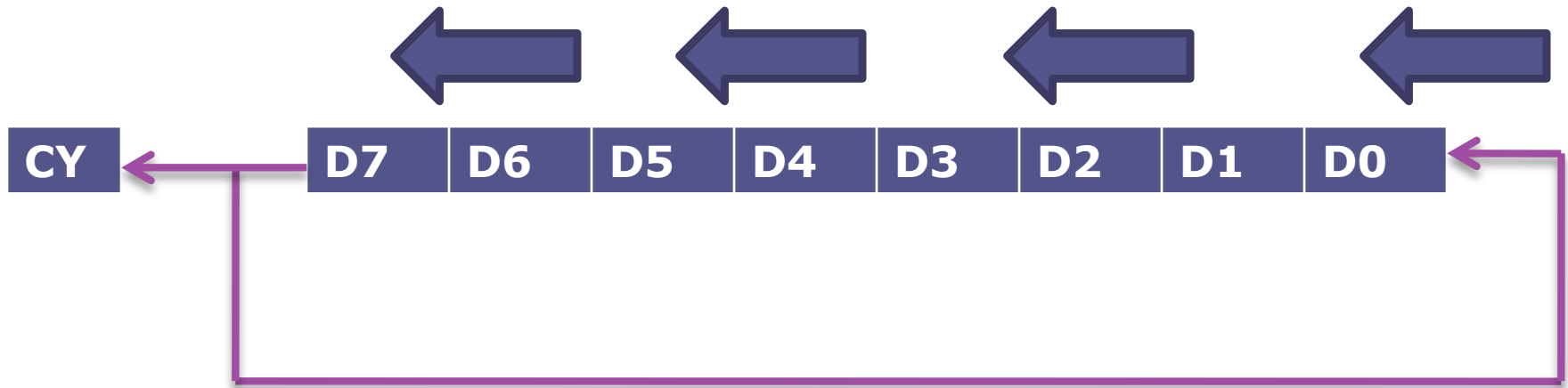
- The 8-bit data is compared with the contents of accumulator.
- The values being compared remain unchanged.
- The result of the comparison is shown by setting the flags of the PSW as follows:
 - if $(A) < \text{data}$: carry flag is set
 - if $(A) = \text{data}$: zero flag is set
 - if $(A) > \text{data}$: carry and zero flags are reset
 - Example: CPI 89H

Logical Instruction (Cont.)

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.
- Example: RLC.

BEFORE EXECUTION



AFTER EXECUTION

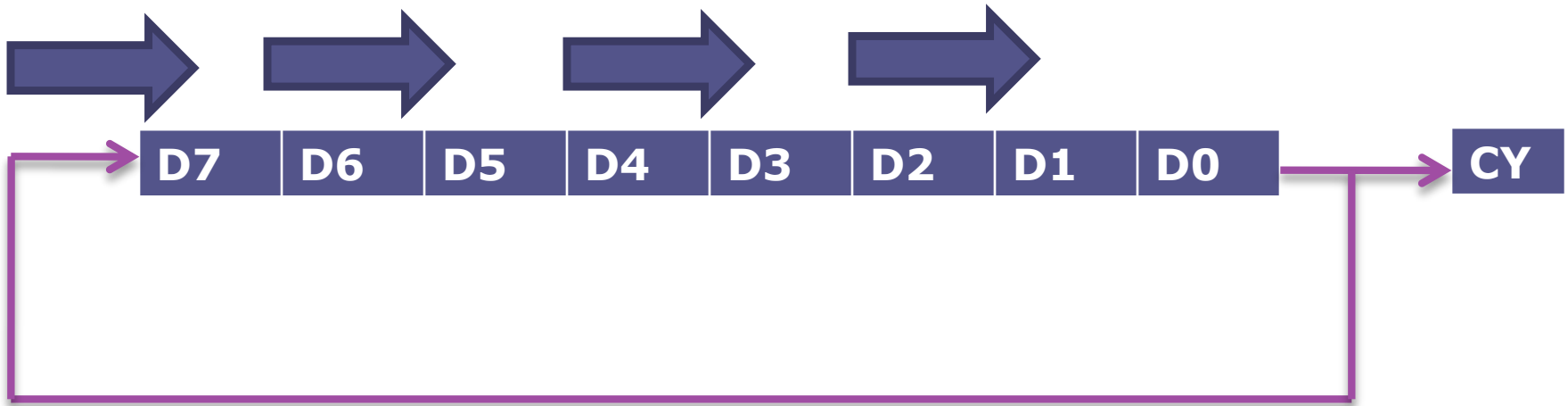


Logical Instruction (Cont.)

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.
- Example: RRC.

BEFORE EXECUTION



AFTER EXECUTION

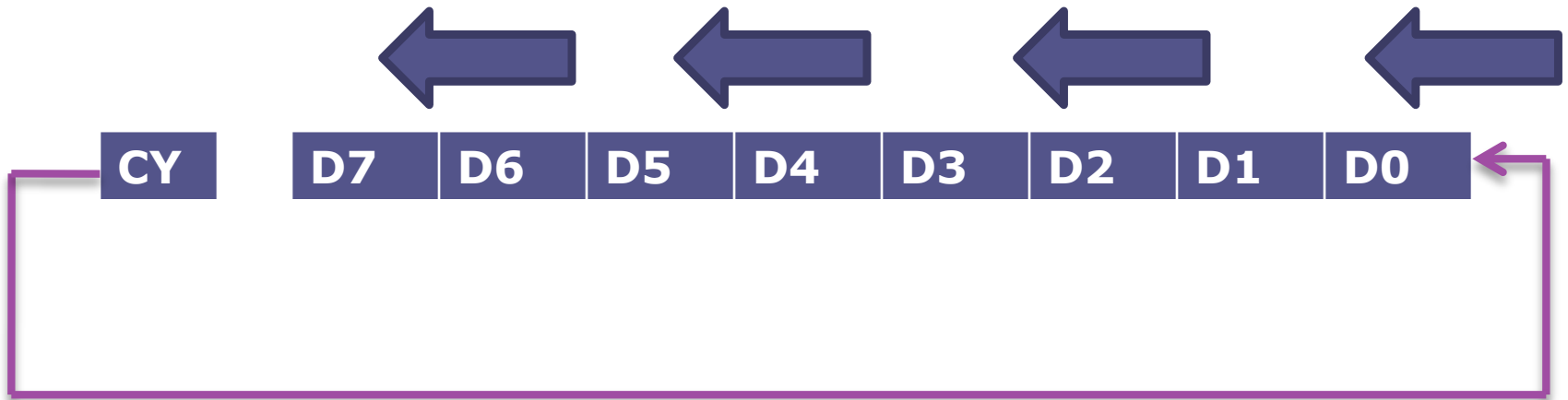


Logical Instruction (Cont.)

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.
- Example: RAL.

BEFORE EXECUTION



AFTER EXECUTION

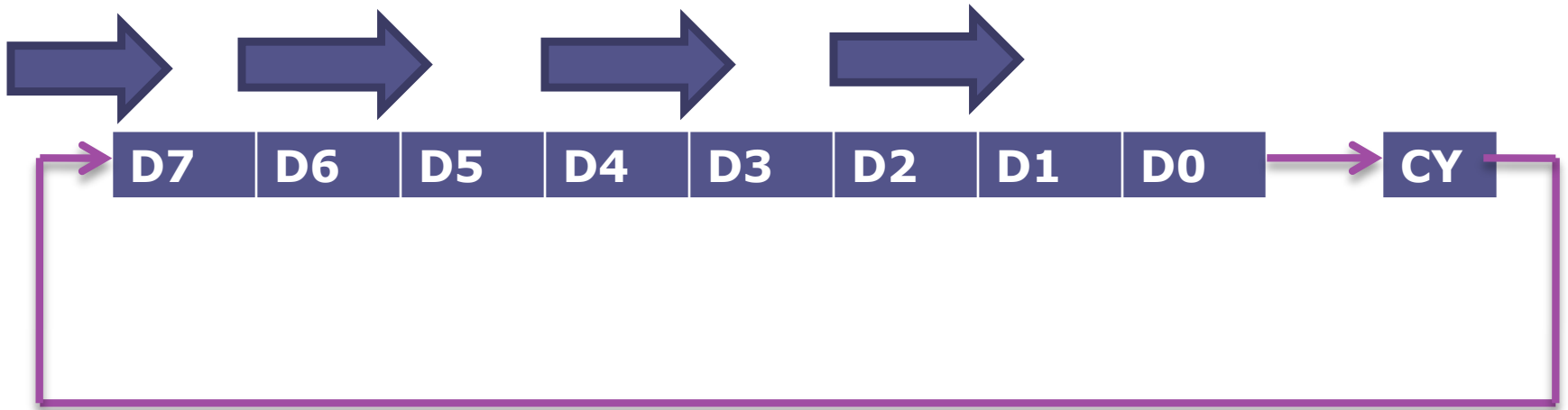


Logical Instruction (Cont.)

Opcode	Operand	Description
RAR	None	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 Instruction

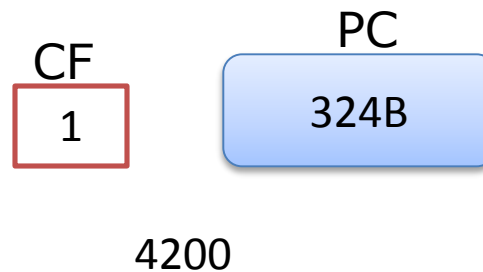
- The branch group instructions allows the microprocessor to change the sequence of program either conditionally or under certain test conditions. The group includes:
 - Jump instructions
 - Call and Return instructions
 - Restart instructions

Unconditional Branch

Opcode	Operand	Description
JMP	Address[16-bit]	Jump to the address specified (Go to).
CALL	Address[16-bit]	Jump to the address specified but treat it as a subroutine.
RET	None	Return from a subroutine.

SS

324A	xx
324B	JMP 4200
324E	xxxxxx
324F	xxxxxx

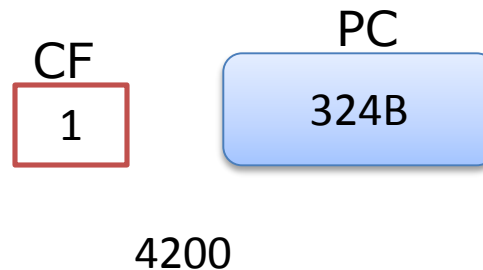


41FF	xxx
4200	xxxxxx
4201	xxxxxx
4202	xxxxxx

Conditional Branch

Opcode	Operand	Description
JC	address[16-bit]	Jump immediate address if CF=1

324A	xx
324B	JC 4200
324E	xxxxxx
324F	xxxxxx



41FF	xxx
4200	xxxxxx
4201	xxxxxx
4202	xxxxxx

Conditional Branch (Cont.)

Opcode	Operand	Description
JNC	address[16-bit]	Jump immediate address if CF=0
JZ	address[16-bit]	Jump immediate address if ZF=1
JNZ	address[16-bit]	Jump immediate address if ZF=0
JP	address[16-bit]	Jump immediate address if SF=0
JM	address[16-bit]	Jump immediate address if SF=1
JPE	address[16-bit]	Jump immediate address if PF=1
JPO	address[16-bit]	Jump immediate address if PF=0

Machine Control Instruction

Opcode	Operand	Description
HLT		Stop the execution of instruction in the microprocessor
NOP		No operation

Instruction Format

- In the 8085, “byte” and “word” are synonymous because it is an 8-bit microprocessor.
- However, instructions are commonly referred to in terms of bytes rather than words.

- 1-byte instructions

OPCODE

- 2-byte instructions

OPCODE

8-bit
data/address

- 3-byte instructions

OPCODE

Low byte
data/address

High byte
data/address

1-byte instructions

- It includes the opcode and the operand in the same byte. For example:

Opcode	Operand	Binary Code	Hex Code
MOV	C, A	0100 1111	4FH
ADD	B	1000 0000	80H
CMA		0010 1111	2FH

2-byte instructions

- The first byte specifies the operation code and the second byte specifies the operand. For example:

Opcode	Operand	Binary Code	Hex Code
MVI	A, 32H	0011 1110 0011 0010	3E [first byte] 32 [second byte]
MVI	B, F2H	0000 0110 1111 0010	06 [first byte] F2 [second byte]

3-byte instructions

- The first byte specifies the operation code and the following two bytes specify the 16-bit address.
- The second byte is the low-order address and the third byte is the high-order address.
- For example:

Opcode	Operand	Binary Code	Hex Code
LDA	2050H	0011 1010 0101 0000 0010 0000	3A [first byte] 50 [second byte] 20 [third byte]
JMP	2085H	1100 0011 1000 0101 0010 0000	C3 [first byte] 85 [second byte] 20 [third byte]

Addressing Modes

- Instructions can be categorized according to their method of addressing the hardware registers and/or memory.
 - Implied Addressing
 - Register Addressing
 - Immediate Addressing
 - Direct Addressing
 - Register Indirect Addressing

Immediate Addressing

- In immediate addressing mode, the data is specified in the instruction itself.
- The data will be a part of the program instruction.
- Example:
 - `MVI B, 3EH` - Move the data 3EH given in the instruction to B register.

Direct Addressing

- In direct addressing mode, the address of the data is specified in the instruction.
- The data will be in memory.
- Example:
 - LDA 1050H - Load the data available in memory location 1050H into accumulator

Register Addressing

- In register addressing mode, the instruction specifies the name of the register in which the data is available.
- Example:
 - MOV A, B - Move the content of B register to A register

Register Indirect Addressing

- In register indirect addressing mode, the instruction specifies the name of the register in which the address of the data is available.
- Here the data will be in memory and the address will be in the register pair.
- Example:
 - MOV A, M - The memory data addressed by H L pair is moved to A register.

Implied Addressing

- In implied addressing mode, the instruction itself specifies the data to be operated.
- Example:
 - CMA - Complement the content of accumulator
 - RAL

Example: Writing a simple program

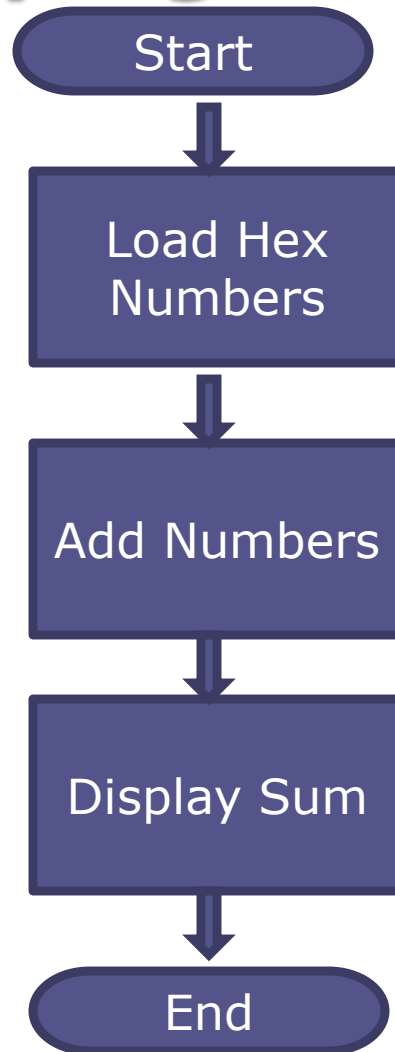
- PROBLEM STATEMENT

- Write instructions to load the two hexadecimal numbers 32H and 48H in registers A and B, respectively.
- Add the numbers, and display the sum at the LED output port .

Example: Writing a simple program (Cont.)

- PROBLEM ANALYSIS
 - Load the numbers in the registers.
 - Add the numbers.
 - Display the sum at the output port.

Example: Writing a simple program (Cont.)



Example: Writing a simple program (Cont.)

- ASSEMBLY LANGUAGE PROGRAM
 - To write an assembly language program, we need to translate the blocks shown in the flowchart into 8085 operations and then, subsequently, into mnemonics.
 - By examining the blocks, we can classify them into three types of operations: Blocks 1 and 3 are copy (data transfer) instruction.
 - Block 2 is an arithmetic operation;
 - Block 4 is a machine-control operation.
 - To translate these steps into assembly and machine languages, we should review the instruction set.

Example: Writing a simple program (Cont.)

- Let us assume that R/W memory ranges from 2000H to 20FFH, and the system has an LED output port with the address 01H.

Block 1:	MVI A,32H	Load register A with 32H
	MVI B,48H	Load register B with 48H
Block 2:	ADD B	Add two bytes and save the sum in A
Block 3:	OUT 01H	Display accumulator contents at port 01H
Block 4:	HALT	End

Example: Writing a simple program (Cont.)

Mnemonics	Hex Code	
MVI A,32H	3E	2-byte instruction
	32	
MVI B,48H	06	2-byte instruction
	48	
ADD B	80	1-byte instruction
OUT 01H	D3	2-byte instruction
	01	
HLT	76	1-byte instruction

Example: Writing a simple program (Cont.)

- In this illustrative example, the program will be stored in memory as follows:

Mnemonics	Hex Code	Memory Contents	Memory Address
MVI A,32H	3E 32	0 0 1 1 1 1 1 0 0 0 1 1 0 0 1 0	2000 2001
MVI B,48H	06 48	0 0 0 0 0 1 1 0 0 1 0 0 1 0 0 0	2002 2003
ADD B	80	1 0 0 0 0 0 0 0	2004
OUT 01H	D3 01	1 1 0 1 0 0 1 1 0 0 0 0 0 0 0 1	2005 2006
HLT	76	0 1 1 1 1 1 1 0	2007

Example: Writing a simple program (Cont.)

- EXECUTING THE PROGRAM
- To execute the program, we need to tell the microprocessor where the program begins by entering the memory address 2000H.

Example 2

- Subtract the contents of memory location 4001H from the memory location 4000H and place the result in memory location 4002H.

(4000H) = 51H

(4001H) = 19H

Result = 51H - 19H = 38H

Source program:

```
LXI H, 4000H      ; HL points 4000H
MOV A, M          ; Get first operand
INX H             ; HL points 4001H
SUB M             ; Subtract second operand
INX H             ; HL points 4002H
MOV M, A          ; Store result at 4002H.
HLT               ; Terminate program execution
```

Example 3

- Load two numbers from memory, multiply them and store the result into memory.

```
LDA 2000      ; Load multiplicand to accumulator
MOV B,A       ; Move multiplicand from A(acc) to B register
LDA 2001      ; Load multiplier to accumulator
MOV C,A       ; Move multiplier from A to C
MVI A,00      ; Load immediate value 00 to a
L: ADD B      ; Add B(multiplier) with A
DCR C         ; Decrement C, it act as a counter
JNZ L         ; Jump to L if C reaches 0
STA 2010      ; Store result in to memory
HLT          ; End
```


Thank You 😊