Lecture Three

Computer organization & Architecture

Yarmouk University College

Dept. of Cybersecurity Engineering

2^{nd} Stage

By Asst. Lect. Anas Yassin Abbas

Instructions of ADD

Instruction	Full meaning	Operation
ACI data	Add with Carry Immediate	A ← A + data + CY
ADC r	Add with Carry	A ← A + r + CY
ADC M	Add memory with Carry	A ← A + [HL] + CY
ADD r	Add Register	A ← A + r
ADD M	Add Memory	A ← A + [HL]
ADI data	Add Immediate	A ← A + data
DAD rp	Double Add	HL ← HL + register pair

Example 1: ADD

```
MVI A, 05H
MVI B, 03H
ADD B
HLT
```

Result: A = 08H (5 + 3).

Example 2: ADI

```
MVI A, 05H
ADI 04H
HLT
```

Result: A = 09H (5 + 4).

Example 3: ADC

Suppose A = 05H, B = 03H, and CY = 1.

```
MVI A, 05H
STC
MVI B, 03H
ADC B
HLT
```

Result: A = 09H (5 + 3 + 1). STC sets the Carry Flag to 1.

Example 4: ACI

```
MVI A, 05H
STC
ACI 02H
HLT
```

Result: A = 08H (5 + 2 + 1).

Example 5: DAD

```
LXI H, 1000H
LXI B, 2000H
DAD B
HLT
```

Result: HL = 3000H (1000H + 2000H).

Instructions of SUB

Instruction	Full meaning	Operation
SUB r	Subtract Register	A ← A − r
SUB M	Subtract Memory	A ← A − [HL]
SUI data	Subtract Immediate	A ← A − data
SBB r	Subtract with Borrow	A ← A − r − CY
SBB M	Subtract Memory with Borrow	A ← A − [HL] − CY
SBI data	Subtract Immediate with Borrow	A ← A − data − CY

Example 6: SUB

```
MVI A, 09H
MVI B, 04H
SUB B
HLT
```

Result: A = 05H (9 − 4).

Example 7: SUI

```
MVI A, 0AH
SUI 03H
HLT
```

Result: A = 07H (10 − 3).

Example 8: SBB

Suppose A = 09H, B = 03H, and CY = 1.

```
MVI A, 09H
STC
MVI B, 03H
SBB B
HLT
```

Result: A = 05H (9 − 3 − 1).

Example 9: SBI

```
MVI A, 0AH
STC
SBI 02H
HLT
```

Result: A = 07H (10 − 2 − 1).

Example 10: SBB M

Suppose memory location 2050H contains 03H, A = 09H, and CY = 1.

```
LXI H, 2050H
MVI A, 09H
STC
SBB M
HLT
```

Result: A = 05H (9 − 3 − 1).

Instruction	Full meaning	Operation
INR r	Increment Register	r ← r + 1
INR M	Increment Memory	[HL] ← [HL] + 1
DCR r	Decrement Register	r ← r − 1
DCR M	Decrement Memory	[HL] ← [HL] − 1
INX rp	Increment Register Pair	rp ← rp + 1
DCX rp	Decrement Register Pair	rp ← rp − 1

Example 11: INR

```
MVI A, 05H
INR A
HLT
```

Result: A = 06H.

Example 12: DCR

```
MVI B, 05H
DCR B
HLT
```

Result: B = 04H.

Example 13: INX

```
LXI H, 2000H
INX H
HLT
```

Result: HL = 2001H.

Example 14: DCX

```
LXI B, 2000H
DCX B
HLT
```

Result: BC = 1FFFH.

- *INR and DCR affect the Sign, Zero, Auxiliary Carry, and Parity flags, but do not change CY.*
- *INX and DCX do not affect the condition flags.*
- *ADD, ADC, ADI, ACI, SUB, SBB, SUI, and SBI update the arithmetic condition flags.*
- *DAD updates only CY.*

Accumulator (A)

1. ANA: This instruction mean take **AND** for Accumulator **A**

Example: if (A=48) and (L=53) Find ANA?

A = 48 = 0100 1000

L = 52 = 0101 0010

[48]	0100 1000	
[52]	0101 0010	AND
[40]	0100 0000	

2. **ANI**: **AND** for accumulator **A** with Immediate data **I** without dealing with specific register.

Example: if (A=84) and (I=90) Find ANI?

A = 84 = 1000 0100

I = 90 = 1001 0000

[84]	1000 0100	
[90]	1001 0000	AND
[80]	1000 0000	

- **Every AND operation: carry flag=0 & Ac flag=1**

Complement Accumulator (CMA)

An instruction its purpose to find the **First Complement** for the **register A.**

1^{st} Complement such as NOT gate (0 ⟶ 1 & 1 ⟶ 0)

Example: if (A=47) Find **CMA**?

A = 47 = 0100 0111

[47]	0100 0111	CMA
[B8]	1011 1000	

Complement with Carry (CMC)

This instruction finds the complement for the carry and dose not effect on another flag.

Compare (CMP)

Comparing **Memory (M)** or **Register (R)** contains with **Accumulator (A)**.

- **Effect on flags (carry and zero flag)**

Compare	Carry	Zero
A > B	0	0
A = B	0	1
A < B	1	0

Example: if (A=84) and (B=90) Find CMP?

A = 84 = 1000 0100

B = 90 = 1001 0000

[84]	1000 0100	
[90]	1001 0000	Sub. (-)
[04]	0000 0100	

CY= 1

Z= 0

Compare (CPI)

Comparing register **A** with Immediate data **I**

Example: if (**A**=52) Find **CPI 72**?

A = 53

data = 72

A – data =

53 – 72 = -19

CY=1

Z=0

ORA

This instruction mean take **OR** for Accumulator **A**

Example: if (A=48) and (L=53) Find ANA?

A = 48 = 0100 1000

L = 52 = 0101 0010

[48]	0100 1000	
[52]	0101 0010	OR
[40]	0101 1010	

- **Every AND operation: carry flag=0 & Ac flag=0**

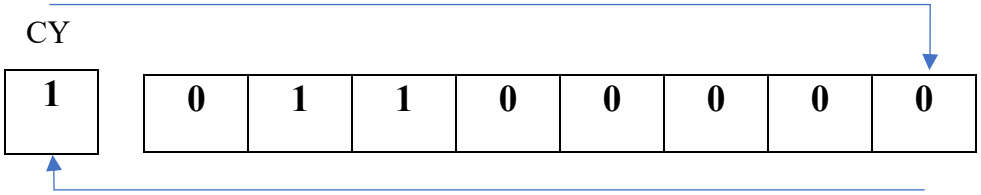
ORI **Home work**

RAL

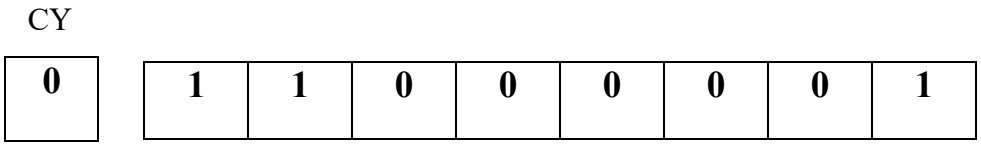
Rotate the **Accumulator** left through **carry**

Example: if (A=60) and (CY=1) Find RAL?

[60] = 0110 0000



After rotate A register =

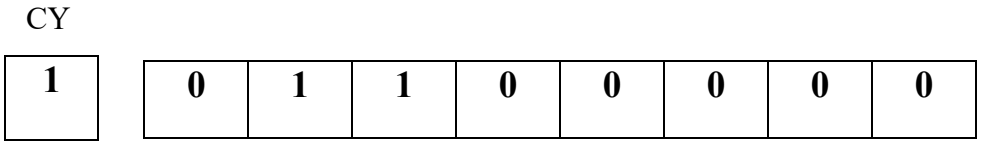


RAR

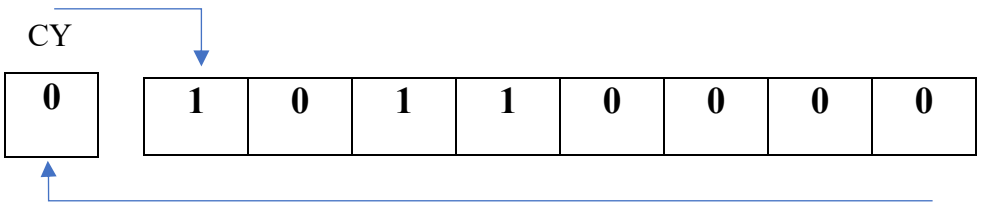
Rotate the **Accumulator** right through **carry**

Example: if (A=60) and (CY=1) Find RAR?

[60] = 0110 0000



After rotate A register =

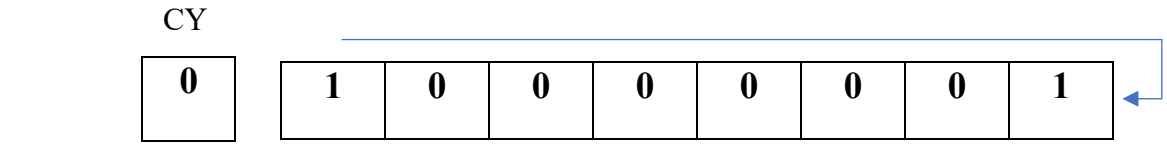


RLC

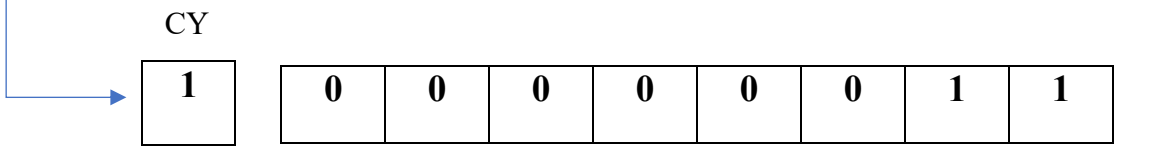
Rotate the **Accumulator** left without rotating **carry**

Example: if (A=81) and (CY=0) Find RLC?

[81] = 1000 0001



After rotate A register =

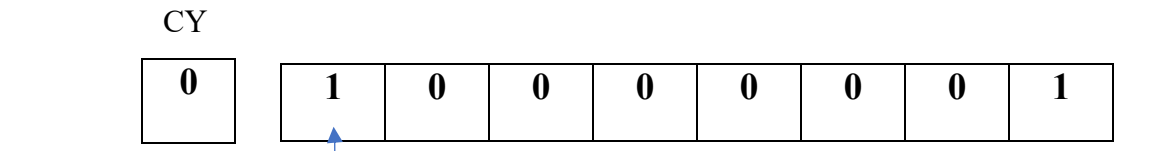


RRC

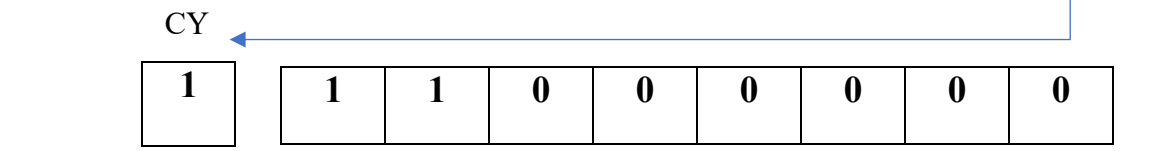
Rotate the **Accumulator** right without rotating **carry**

Example: if (A=81) and (CY=0) Find RRC?

[81] = 1000 0001



After rotate A register =



STC

This instruction set **carry** to **1**

If carry flag = 0 ⟹ 1

XRA

Exclusive OR for **register(R)** or **memory(M)** with **Accumulator(A)** contains

Example: if (A=A5) and (B=B7) Find XRA B?

A = A5 = 1010 0101

L = B7 = 1011 0111

[A5]	1010 0101	
[B7]	1011 0111	XOR
[12]	0001 0010	

- Carry = 0 always

XRI

Example: if (A=A5) and data = 07 Find XRI?

XRI 07

Homework

Branch Group

JMP

It is used to change the normal sequence of program execution by transferring control to another location in the program.

JMP label

JMP: the jump instruction.

label: the destination where program execution continues.

Example:

```
MOV AX, 5
JMP NEXT
MOV AX, 10     ; This instruction is skipped
NEXT:
MOV BX, 20     ; Execution continues here
```

Example:

```
org 100h
MOV AX, 3
JMP SKIP
MOV AX, 8
SKIP:
ADD AX, 2
RET
```

AX =???

JC: Jump if carry

It is a conditional jump instruction that checks the Carry Flag (CF) in the flag register.

If CF = 1, the processor jumps to the specified label.

If CF = 0, the processor continues to the next instruction.

Example:

```
org 100h
MOV AL, 10
ADD AL, 5
JC CARRY_FOUND
MOV BL, 0
JMP FINISH
CARRY_FOUND:
MOV BL, 1
FINISH:
RET
```

JNC: Jump if no carry

It is a conditional jump instruction that checks the Carry Flag (CF).

If CF = 0, the processor jumps to the specified label.

If CF = 1, the processor continues to the next instruction.

Example: Addition without carry

```
ORG 100H
MOV AL, 10
ADD AL, 5
JNC NO_CARRY
MOV BL, 1      ; Carry occurred
JMP FINISH
NO_CARRY:
MOV BL, 0      ; No carry occurred
FINISH:
MOV AH, 4CH
INT 21H
```

The result is:

AL = 0FH (15 decimal)

CF = 0

BL = 00H (no carry)

Example: Addition with carry

```
ORG 100H
MOV AL, 250
ADD AL, 10
JNC NO_CARRY
MOV BL, 1      ; Carry occurred
JMP FINISH
NO_CARRY:
MOV BL, 0      ; No carry
FINISH:
MOV AH, 4CH
INT 21H
```

The result is:

AL = 04H (the low 8 bits of the result)

CF = 1

BL = 01H (carry occurred)

JZ: Jump if zero

It is a conditional jump that checks the Zero Flag (ZF) in the flag register.

If ZF = 1, the processor jumps to the specified label.

If ZF = 0, the processor continues to the next instruction.

Example: Check whether two numbers are equal

```
ORG 100H
MOV AL, 5
MOV BL, 5
CMP AL, BL
JZ EQUAL
MOV CL, 0      ; Numbers are not equal
JMP FINISH
EQUAL:
MOV CL, 1      ; Numbers are equal
FINISH:
MOV AH, 4CH
INT 21H
```

Final result: CL = 01H, because both numbers are equal.

JNZ: Jump if Not Zero

The JNZ instruction means Jump if Not Zero. It jumps to a specified label if the Zero Flag (ZF) = 0.

ZF = 0: Jump is performed.

ZF = 1: Jump is not performed.

Example: When the numbers are different

Change the first number to 5 and the second to 3:

```
ORG 100H
MOV AL, 5
MOV BL, 3
CMP AL, BL
JZ EQUAL
MOV CL, 0
JMP FINISH
EQUAL:
MOV CL, 1
FINISH:
MOV AH, 4CH
INT 21H
```

Now ZF = 0, so JZ EQUAL does not jump. The final value is CL = 00H.

JPE: Jump if Parity even

It is a conditional jump instruction that checks the Parity Flag (PF).

If PF = 1, the processor jumps to the specified label.

If PF = 0, the processor continues to the next instruction.

Example:

```
ORG 100H
MOV AL, 00000011B
ADD AL, 00000001B
JPE EVEN_PARITY
MOV BL, 0
JMP FINISH
EVEN_PARITY:
MOV BL, 1
FINISH:
MOV AH, 4CH
INT 21H
```

PF=1

JPO: Jump if Parity Odd

jumps to a specified label when the Parity Flag (PF) = 0, meaning the lowest 8 bits of the previous result contain an odd number of 1 bit.

PF = 0: Jump is performed.

PF = 1: Jump is not performed.

Example:

```
MVI A, 01H
ADI 01H
JPO ODD_PARITY
MVI B, 00H
JMP FINISH
ODD_PARITY:
MVI B, 01H
FINISH:
HLT
```

PF=0

JM: Jump if Minus

JM means Jump if Minus. It transfers program execution to a specified address if the Sign Flag (S) = 1.

S = 1: Jump is performed (result is negative).

S = 0: Jump is not performed (result is non-negative).

Example:

```
MVI A, 05H
SUI 08H
JM NEGATIVE
MVI B, 00H
JMP FINISH
NEGATIVE:
MVI B, 01H
FINISH: HLT
```

JP: Jump if Plus (8085 Assembly)

JP means Jump if positive. It jumps to a specified address when the Sign Flag (S) = 0.

S = 0: Jump is performed (result is non-negative).

S = 1: Jump is not performed (result is negative).

Example:

```
MVI A, 08H
SUI 03H
JP POSITIVE
MVI B, 00H
JMP FINISH
POSITIVE:
MVI B, 01H
FINISH:
HLT
```