Branch Instructions Group

Lecture objectives: at the end of this lecture the student will able to:

1- **Explain the working of jumping instructions.**

2- **Explain the working of call and return instructions.**

3- **Write programs including jump or calling instructions.**

4.1 Branch Group Instructions:

The branch instructions allow the microprocessor to change the sequence of program execution. A 8085 microprocessor has two types of jumping are unconditional and conditional.

The branch group instructions is classified into three categories:

1- Jump instructions.

2- Call instructions.

3- Return instructions.

The branching instruction alter the normal sequential flow. These instructions alter either unconditionally or conditionally.

4.1.1 Jumping instructions

A- **JMP operand [three bytes instruction]**

This instruction is loading the program counter with operand where the fetch and execute cycle of program starting from memory location that addressed by operand, where, this lead to change the sequence of instructions execution. Operand is 16-bit number.

JMP 2300 changing the execution of program starting from address (**2300**).

B- **JC operand [three bytes instruction]**

This instruction is loading the program counter with operand where the fetch and execute cycle of program starting from memory location that addressed by operand if the (**CY=1**), where, this lead to change the sequence of instructions execution. Operand is 16-bit number.

JC 2300 changing the execution of program starting from address (**2300**) if (**CY=1**).

C- **JNC operand [three bytes instruction]**

This instruction is loading the program counter with operand where the fetch and execute cycle of program starting from memory location that addressed by operand if the (**CY=0**), where, this lead to change the sequence of instructions execution. Operand is 16-bit number.

JNC 2300 changing the execution of program starting from address (**2300**) if (**CY=0**).

D- **JPO operand** **[three bytes instruction]**

This instruction is loading the program counter with operand where the fetch and execute cycle of program starting from memory location that addressed by operand if the (**P=0, odd parity**), where, this lead to change the sequence of instructions execution. Operand is 16-bit number.

JPO 2300 changing the execution of program starting from address (**2300**) if (**P=0**).

E- **JPE operand** **[three bytes instruction]**

This instruction is loading the program counter with operand where the fetch and execute cycle of program starting from memory location that addressed by operand if the (**P=1, even parity**), where, this lead to change the sequence of instructions execution. Operand is 16-bit number.

JPE 2300 changing the execution of program starting from address (**2300**) if (**P=1**).

F- **JZ operand** **[three bytes instruction]**

This instruction is loading the program counter with operand where the fetch and execute cycle of program starting from memory location that addressed by operand if the (**Z=1**), where, this lead to change the sequence of instructions execution. Operand is 16-bit number.

JZ 2300 changing the execution of program starting from address (**2300**) if (Z=**1**).

G- **JNZ operand** **[three bytes instruction]**

This instruction is loading the program counter with operand where the fetch and execute cycle of program starting from memory location that addressed by operand if the (**Z=0**), where, this lead to change the sequence of instructions execution. Operand is 16-bit number.

JNZ 2300 changing the execution of program starting from address (**2300**) if (Z=**0**).

H- **JP operand** **[three bytes instruction]**

This instruction is loading the program counter with operand where the fetch and execute cycle of program starting from memory location that addressed by operand if the (S=**0**), where, this lead to change the sequence of instructions execution. Operand is 16-bit number.

JP 2300 changing the execution of program starting from address (**2300**) if (S=**0**).

I- **JM operand** **[three bytes instruction]**

This instruction is loading the program counter with operand where the fetch and execute cycle of program starting from memory location that addressed by operand if the (S=**1**), where, this lead to change the sequence of instructions execution. Operand is 16-bit number.

JM 2300 changing the execution of program starting from address (**2300**) if (S=**1**).

4.1.2 Call instructions

A- Call operand [three bytes instruction]

This instruction is loading the program counter with operand where the fetch and execute cycle of program starting from memory location that addressed by operand unconditionally, where, this lead to execute the subroutine that stored in memory starting from operand. Operand is 16-bit number.

Call 2300 calling the subroutine which stored in memory starting from (**2300**).

B- CC operand [three bytes instruction]

This instruction is loading the program counter with operand where the fetch and execute cycle of program starting from memory location that addressed by operand if (**CY=1**), where, this lead to execute the subroutine that stored in memory starting from operand. Operand is 16-bit number.

CC 2300 calling the subroutine which stored in memory starting from (**2300**) if **CY**=1

C- CNC operand [three bytes instruction]

This instruction is loading the program counter with operand where the fetch and execute cycle of program starting from memory location that addressed by operand if (**CY=0**), where, this lead to execute the subroutine that stored in memory starting from operand. Operand is 16-bit number.

CNC 2300 calling the subroutine which stored in memory starting from (**2300**) if **CY=0**

D- CPO operand [three bytes instruction]

This instruction is loading the program counter with operand where the fetch and execute cycle of program starting from memory location that addressed by operand if (**P=0**), where, this lead to execute the subroutine that stored in memory starting from operand. Operand is 16-bit number.

CPO 2300 calling the subroutine which stored in memory starting from (**2300**) if **P=0**.

E- CPE operand [three bytes instruction]

This instruction is loading the program counter with operand where the fetch and execute cycle of program starting from memory location that addressed by operand if (**P=1**), where, this lead to execute the subroutine that stored in memory starting from operand. Operand is 16-bit number.

CPE 2300 calling the subroutine which stored in memory starting from (**2300**) if **P=1**.

F- CZ operand [three bytes instruction]

This instruction is loading the program counter with operand where the fetch and execute cycle of program starting from memory location that addressed by operand if (**Z=1**), where, this lead to execute the subroutine that stored in memory starting from operand. Operand is 16-bit number.

CZ 2300 calling the subroutine which stored in memory starting from (**2300**) if **Z=1**.

G- CNZ operand [three bytes instruction]

This instruction is loading the program counter with operand where the fetch and execute cycle of program starting from memory location that addressed by operand if (**Z=0**), where, this lead to execute the subroutine that stored in memory starting from operand. Operand is 16-bit number.

CNZ 2300 calling the subroutine which stored in memory starting from (**2300**) if **Z=0**.

H- CP operand [three bytes instruction]

This instruction is loading the program counter with operand where the fetch and execute cycle of program starting from memory location that addressed by operand if (S=**0**), where, this lead to execute the subroutine that stored in memory starting from operand. Operand is 16-bit number.

CP 2300 calling the subroutine which stored in memory starting from (**2300**) if S=**0**.

I- CM operand

This instruction is loading the program counter with operand where the fetch and execute cycle of program starting from memory location that addressed by operand if (S=**1**), where, this lead to execute the subroutine that stored in memory starting from operand. Operand is 16-bit number..

CM 2300 calling the subroutine which stored in memory starting from (**2300**) if **S=1**.

4.1.3 Return instructions:

Return to the main program from subroutine.

A- **RET** changing the execution of program from subroutine to the main program unconditionally

B- RC changing the execution of program from subroutine to the main program if **CY=1**

C- RNC changing the execution of program from subroutine to the main program if **CY=0**

D- RZ changing the execution of program from subroutine to the main program if **Z=1**

E- RNZ changing the execution of program from subroutine to the main program if **Z=0**

F- RPE changing the execution of program from subroutine to the main program if **P=1**

G- RPO changing the execution of program from subroutine to the main program if **P=1**

H- RP changing the execution of program from subroutine to the main program if **S=0**

I- RM changing the execution of program from subroutine to the main program if **S=1**

Home work:

Write the content of accumulator after execute the following program:

```
LXI H,2000
MVI C,67
MOV M,C
MVI A,FF
CMP M
JNC A1
ANA C
A1: HLT
```