④ Branch Control Group :-

- The instruction of this group change the normal sequence of the program.
- There are two types of branch instruction

(i) Unconditional Branch Instruction :-

It transfer the program to the specified label or address unconditionally i.e without satisfying any condition.

(ii) Conditional Branch Instruction :-

It transfer the program to the specified label or address when certain condition is specified or satisfied. The condition depends upon the status of Zero, Parity, Sign, Carry Flags.

1. JMP addr (label)

→ Unconditional Jump

→ Jump to the address of the label unconditionally. The address of the label is the address of memory location for next instruction to be executed. The instruction load the address of label in PC. (Immediate)

eg JMP 7000

Before Execution	After Execution
PC = 5000 H	PC = 7000 H

This instruction will load the PC with 7000 H and the processor fetch or execute the instruction from this address.

Conditional Jump Instructions :-

If the condition is true or satisfied the only jump is made at specified address.

If the condition is false or not satisfied then the next instruction in the sequence is executed.

2) JZ add (label) : Jump if the result is zero i.e Z=1.

3) JNZ add (label) : Jump if the result is not zero ie Z=0.

4) JC adde (label) : Jump if there is carry i.e CY =1

5) JNC adde (label) : Jump if there is no carry i.e CY=0

6) JP adde (label) : Jump if the result is plus i.e S=0

7) JM adde (label) : Jump if the result is Minus ie S=1

8) JPE adde (label) : Jump if even parity i.e P=1

9) JPO adde (label) : Jump if odd parity i.e P=0

If condition is true PC ← adde (label)

else PC ← PC + 3

(execute next instruction in sequence)

(Immediate)

eg JZ 7000H

Before Execution	After Execution
[PC] = 5000H	[PC] = 7000 H
[Z] = 1	[Z] = 1

eg JZ 7000 H

[PC] = 5000H	[PC] = 50003 H
Z = 0	Z = 0

10) Call addr (label)

→ Unconditional Call

→ It is used to call subroutine.

→ Before calling or jumping to the subroutine, the address of next instruction of the main program is saved in the stack.

→ The content of the stack pointer is decremented by two.

→ Then the program jumps to subroutine starting at address specified by label.

→ Immediate / Indirect

eg CALL 5000

Before E.

[SP] = 7000

[PC] = 6000

After E.

[6FFF] = 60

[6FFE] = 00

[SP] = 6FFE

[PC] = 5000

When the instruction is executed, the program counter contents 6000 will be stored on the stack and microprocessor will load the PC with 5000H and starts executing instruction from 5000 onwards.

Conditional CALL Instruction :-

If condition is true then program calls the specified subroutine.

If condition is false, then the next instruction in the sequence is executed.

Before call, the address of next instruction of the main program is saved in the stack. The stack pointer is decremented by 2

11) CZ addr (label): Call a subroutine if the result is zero, Z=1.

12) CNZ addr (label): Call a subroutine if the result is not zero, Z=0.

13) CC addr (label): Call a subroutine if there is carry i.e CY=1.

14) CNC addr (label): Call a subroutine if there is no carry i.e CY=0.

15) CP addr (label): Call a subroutine if the result is plus ie S=0.

16) CM addr (label): Call a subroutine if the result is negative, S=1.

17) CPE addr (label): Call a subroutine if even parity i-e P=1.

18) CPO addr (label): Call a subroutine if odd parity i-e P=0.

If Condition is true

$$[[SP]-1] \leftarrow PCH$$

$$[[SP]-2] \leftarrow PCL$$

$$[SP] \leftarrow [SP] - 2$$

$$[PC] \leftarrow \text{addr (label)}$$

else

$$PC \leftarrow PC + 3$$

i-e execute the next instruction in the sequence.

Immediate / Indirect.

Example:

1) CZ 7000

Before E.	After E.
[SP] = 5000	[4FFF] = 60
[PC] = 6000	[4FFE] = 00
[Z] = 1	[SP] = 4FFE
	[PC] = 7000
	[Z] = 1

The program calls a subroutine from 7000 because condition is true i.e Z = 1.

2) CZ 7000

Before E.	After E.
[SP] = 5000	[SP] = 5000
[PC] = 6000	PC = 6003
[Z] = 0	[Z] = 0

The program does not call a subroutine because condition is false i.e Z = 0.

19) RET

→ Unconditional Return

→ Return from subroutine.

→ RET instruction is used at the end of subroutine.

→ Before the execution of a subroutine the address of the next instruction of the main program is saved in the stack.

→ The execution of RET instruction brings back the saved address from the stack to the program counter.

→ The content of stack pointer is incremented by 2. Then the program jumps to the next instruction after call in the main program.

→ Indirect.

eg
=)

RET

Before E.	After E.
[SP] = 7000 H	[SP] = 7002 H
[7000] = 00 H	[7000] = 00H
[7001] = 50 H	[7001] = 50H
[PC] = 2000 H	[PC] = 5000 H

So the next instruction will be executed from 5000 instead of 2000.

Conditional RET Instruction :-

If condition is true, then program returns from the subroutine.

If condition is false, then the next instruction in the sequence is executed.

20) RZ : Return from subroutine, if the result is zero i.e Z = 1.

21) RNZ : Return from subroutine, if the result is not zero, i.e Z = 0.

22) RC : Return from subroutine, if there is carry, i.e CY = 1

23) RNC : Return from subroutine, if there is no carry i.e CY = 0.

24) RPE: Return from subroutine, if even parity i.e P = 1.

25) RPO: Return from subroutine, if odd parity i.e P = 0.

If Condition is true

$$[PCL] \leftarrow [[SP]]$$

$$[PCH] \leftarrow [[SP]] + 1$$

$$[SP] \leftarrow [SP] + 2$$

else

$$PC \leftarrow PC + 1$$

i.e execute the next instruction in the sequence.

eg RZ

Before E.

[SP] = 7000H [Z] = 1

[7000] = 00H

[7001] = 50H

[PC] = 2000H

After E.

[SP] = 7002H [Z] = 1

[7000] = 00H

[7001] = 50H

[PC] = 5000

The program will return from subroutine to main program and start the execution of instruction from 5000 because condition is true i.e Z=1.

24) RZ

Before E.	After E.
[SP] = 7000 H	[SP] = 7000 H
[7000] = 00 H	[7000] = 00 H
[7001] = 50 H	[7001] = 50 H
[PC] = 2000 H	[PC] = 2001 H (PC+1)
[Z] = 0	[Z] = 0

The program does not return from subroutine because condition is false i.e Z=0.

26) RP : Return from subroutine, if result is plus ; S=0.

27) RM : Return from subroutine, if result is negative ; S=1

28) PCHL

→ The content of HL are transfered to program counter. The content of register H are moved to high - order 8- bits of PC. The content of register L are transfer to low - order 8- bits of register PC.

→ Register

eg PCHL

Before Execution	After Execution
[HL] = 7000 H	[HL] = 7000 H
[PC] = 5000 H	[PC] = 7000 H

The program start the execution of instruction