11. What ... branching instruc...

As microprocessor is a sequential machine, it executes the machine codes form the one memory location to the next. So, some mechanism is needed in order to change the sequence of a program under certain conditions. The branch instruction which are also regarded as the most powerful instructions, instruct the microprocessor to go to a different memory location and the microprocessor continues executing machine codes from that new location. These instructions add flexibility and versatility to the microprocessor,.

The various types of branching instruction in 8085 are as follows.

1) Jump Instructions

Jump instructions specify the memory location explicitly. They are 3 byte instruction; one byte for the op – code followed by a 16 bit memory address.

i) Unconditional Jump

It transfers the control to a new address specified in the instruction without checking any conditions. It helps the programmer to set up continuous loop.

The unconditional jump instruction is

JMP 16 bit address.

For example if current program sequence is at 2030 H and the program sequence is to be transferred to memory location 2080 H , then the hump instruction will be JMP 2080 H

Memory address	Hex code	Mnemonics
2030	C3	JMP 2080 H
2031	80	
2032	20	

ii) **Conditional Jumps.**

These instructions allow the microprocessor to make decisions to change or not change the sequence of program to a new address being based upon the condition of the flag (Set or reset)

Various conditional Jump Instructions along with the condition of Flags are as:

OP code	Description	Flag - status
JC	Jump on carry	CY = 1
JNC	Jump on no carry	CY = 0
JP	Jump on positive	S = 0
JM	Jump on minus	S = 1
JPE	Jump on Parity Even	P = 1
JPO	Jump on Parity Odd	P =0
JZ	Jump on Zero	Z = 1
JNZ	Jump on No Zero	z = 0

For example the current program sequence is to be transferred from 2000 H to 2050 H. Then JC 2050 H will transfer the program sequence to 2050 H if the carry flag is set i.e. CY = 1

2) **Call and Return Instructions**

CALL and RET instructions are used to implement subroutines. A subroutine is group of instructions written separately from the main program to perform a function that occurs repeatedly in the main perform.

The CALL Instructions is used in the main program to call a subroutine and the RET instructions is used at the end of the subroutine to return to the main program.

i) **Unconditional CALL and RET instructions**

The unconditional CALL an RET instructions are as:

Opcode	Operand	Description
CALL	16 bit memory address of a subroutine	it is a 3 byte instruction that transfer the program sequence to a subroutine address unconditionally.
RET		It is a 1 byte instruction that is used to return from subroutine unconditionally.

Example: The use of CALL F950 H to branch to the subroutine at F950 H and the RET instruction in the subroutine to get back to the main program is shown below.

ii) Conditional CALL and return instructions

The Conditional CALL and RET instruction are based on four at data conditions or flag : Carry, Zero, Sign and Parity flags.

In case of a conditional call instructions the program is transferred to the subroutine if the condition is met, else the main program is continued.

In case of a conditional Return instruction the sequence return to the main program if the condition is met, else the sequence in the subroutine is continued.

Conditional call:

OP – code	Description	Flag status
CC	Call on carry	CY = 1
CNC	Call with No carry	CY = 0
CP	Call on positive	S = 0
CM	Call on minus	s = 1
CPE	Call on parity Even	p = 1
CPO	Call on parity odd	p = 0
CZ	Call on Zero	z = 1
CNZ	Call on No zero	z = 0

Conditional Return:

OP – Code	Description	Flag status
RC	Return on carry	CY = 1
RNC	Return with No carry	CY = 0
RP	Return on Positive	S = 0
RM	Return on Minus	S = 1
RPE	Return on Parity Even	P = 1
RPO	Return on Parity Odd	P = 0
RZ	Return on Zero	z = 1
RNZ	Return on No zero	z = 0