9- LOOPS AND LOOP-HANDLING INSTRUCIONS

The 8086 microprocessor has three instructions specifically designed for implementing loop operations. These instructions can be used in place of certain conditional jump instructions and give the programmer a simpler way of writing loop sequences. The loop instructions are listed in Fig.(a).

Mnemonic	Meaning	Format	Operation
LOOP	Loop	LOOP Short-label	(CX) ← (CX) − 1 Jump is initiated to location defined by short-label if (CX) ≠ 0; otherwise, execute next sequential instruction
LOOPE/LOOPZ	Loop while equal/ loop while zero	LOOPE/LOOPZ Short-label	(CX) ← (CX) − 1 Jump to location defined by short-label if (CX) ≠ 0 and (ZF) = 1; otherwise, execute next sequential instruction
LOOPNE/ LOOPNZ	Loop while not equal/ loop while not zero	LOOPNE/LOOPNZ Short-label	(CX) ← (CX) − 1 Jump to location defined by short-label if (CX) ≠ 0 and (ZF) = 0; otherwise, execute next sequential instruction

Fig(a)

The first instruction, *loop* (**LOOP**), works with respect to the contents of the CX register. CX must be preloaded with a count that represents the number of times the loop is to repeat. Whenever LOOP is executed, the contents of CX are first decremented by one and then checked to determine if they are equal to zero. If equal to zero, the loop is complete and the instruction following LOOP is executed; otherwise, control is returned to the instruction at the label specified in the loop instruction. In this way, we see that LOOP is a single instruction that functions the same as a decrement CX instruction followed by a JNZ instruction.

For example, the LOOP instruction sequence shown in Fig.-1-(a) causes the part of the program from label NEXT through the instruction LOOP to repeat a number of times equal to the value stored in CX.

```
        MOV CX,COUNT      Load count for the number of repeats
NEXT:       .
            .
            .
            .             Body of routine that is repeated
            .
            .
            .
LOOP NEXT                 Loop back to label NEXT if count not zero
```

Fig-1- (a)

For example, if CX contains 000AH, the sequence of instructions included in the loop executes 10 times.

Figure-1-(b) shows a practical implementation of a loop. Here we find the block move program in fig-1-(c) is rewritten using the LOOP instruction.

We see that the instruction LOOP NXTPT has replaced both the DEC and JNZ instructions.

```
        MOV     AX, DATASEGADDR
        MOV     DS, AX
        MOV     SI, BLK1ADDR
        MOV     DI, BLK2ADDR
        MOV     CX, N
NXTPT:  MOV     AH, [SI]
        MOV     [DI], AH
        INC     SI
        INC     DI
        DEC     CX
        JNZ     NXTPT
        HLT
```

Fig-1- (b)

```
        MOV     AX,DATASEGADDR
        MOV     DS,AX
        MOV     SI,BLK1ADDR
        MOV     DI,BLK2ADDR
        MOV     CX,N
NXTPT:  MOV     AH,[SI]
        MOV     [DI],AH
        INC     SI
        INC     DI
        LOOP    NXTPT
        HLT
```

Fig-1- (c)

The other two loop instructions in Fig.-a- operate in a similar way except that check for two conditions.

The instruction **loop while equal** (LOOPE) / **loop while zero** (LOOPZ) checks the contents of both CX and the zero flag (ZF). Each time the

loop instruction is executed, CX decrements by 1 without affecting the flags, its contents are checked for 0, and the state of ZF that results from execution of the previous instruction is tested for 1. If CX is not equal to 0 and ZF equals 1, a jump is initiated to the location specified with the Short-label operand and the loop continues. If either CX or ZF is 0, the loop is complete, and the instruction following the loop instruction is executed.

Instruction **loop while not equal** (LOOPNE) / **loop while not zero** (LOOPNZ) works in a similar way to the LOOPE/LOOPZ instruction. The difference is that it checks ZF and CX looking for ZF equal to 0 together with CX not equal to 0. If these conditions are met, the jump back to the location specified with the Short-label operand is performed and the loop continues.

Ex: Explain what happens as the following sequence of instructions is executed.

```
        MOV DL , 05H
        MOV AX, 0A00H
        MOV DS, AX
        MOV SI, 0H
        MOV CX, 0FH
AGAIN:  INC SI
        CMP [SI] , DL
        LOOPNE AGAIN
```

Solution:

The first five instructions are for initializing internal registers. Data register DL is loaded with 05H; data segment register DS is loaded via AX with the value 0A00 H; source index register SI is loaded with 0000 H; and count register CX is loaded with 0FH (15 Dec.).after initialization, a data segment is set up at physical address 0A000H, and SI points to the memory location at offset 0000H in this data segment. DL contains the data 5H and the CX register contains the loop count 15 Dec.

The part of the program that starts at the label AGAIN and ends with the LOOPNE instruction is a software loop. The first instruction in the loop increments the value in SI by 1. Therefore, the first time through the loop SI points to the memory address 0A001H. The next instruction compares the contents of this memory location with the content of DL, 5Dec.. If the data held at 0A001H are 05H, the zero flag is set; otherwise, it is reset . the LOOPNE instruction then decrements CX (making it 0EH) and then checks for CX = 0 or ZF = 1. If neither of these two conditions is satisfied, program control is returned to the instruction with the label AGAIN. This causes the comparison to be repeated for the examination of the contents of the next byte in memory. On the other hand, if either condition is satisfied, the loop is complete. In this way, we see that the loop is repeated until either a number 5Dec. is found or all locations in the address range 0A001H through 0A00FH are tested and found not to contain 5Dec..