10- STRINGS AND STRING-HANDLING INSTRUCTIONS

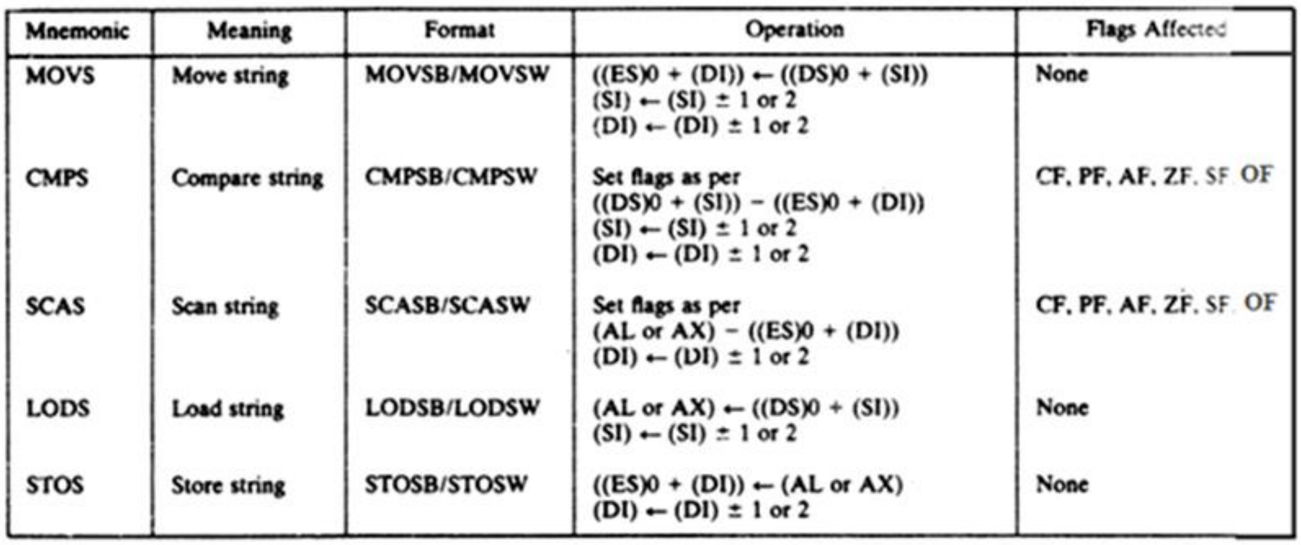
By string we mean a series of data words (or bytes) that reside in consecutive memory locations. The string instructions of the 8086 permit a programmer to implement operations such as

- move data from one block of memory to a block elsewhere in memory.
- Scanning a string of data elements stored in memory to look for a specific value.
- Comparing the elements of two strings in order to determine whether they are the same or different.

Five basic String Instructions define operations on one element of a string:

- Move byte or word string **MOVSB/MOVSW**
- Compare string **CMPSB/CMPSW**
- Scan string **SCASB/SCASW**
- Load string **LODSB/LODSW**
- Store string **STOSB/STOSW**

These instructions, listed in Fig.(a). Repetition is needed to handle more than one element of a string.



Mnemonic	Meaning	Format	Operation	Flags Affected
MOVS	Move string	MOVSB/MOVSW	((ES)0 + (DI)) ← ((DS)0 + (SI)) (SI) ← (SI) ± 1 or 2 (DI) ← (DI) ± 1 or 2	None
CMPS	Compare string	CMPSB/CMPSW	Set flags as per ((DS)0 + (SI)) − ((ES)0 + (DI)) (SI) ← (SI) ± 1 or 2 (DI) ← (DI) ± 1 or 2	CF, PF, AF, ZF, SF, OF
SCAS	Scan string	SCASB/SCASW	Set flags as per (AL or AX) − ((ES)0 + (DI)) (DI) ← (DI) ± 1 or 2	CF, PF, AF, ZF, SF, OF
LODS	Load string	LODSB/LODSW	(AL or AX) ← ((DS)0 + (SI)) (SI) ← (SI) ± 1 or 2	None
STOS	Store string	STOSB/STOSW	((ES)0 + (DI)) ← (AL or AX) (DI) ← (DI) ± 1 or 2	None

Fig.(a)

Auto-indexing of String Instructions

Execution of a string instruction causes the address indices in SI and DI to be either automatically incremented or decremented. The decision to increment or decrement is made based on the status of the direction flag (DF).

The 8086 provides two instructions, clear direction flag (CLD) and set direction flag (STD)

When CLD is executed , DF is set to 0 .this Selects the auto-increment mode, and each time a string operation is performed, SI and/or DI are incremented by 1 if byte data are processed and by 2 if word data are processed.

To select the auto decrement (DF=1) , STD instruction is executed.

Mnemonic	Meaning	Format	Operation	Flags Affected
CLD	Clear DF	CLD	(DF) ← 0	DF
STD	Set DF	STD	(DF) ← 1	DF

Move String : MOVSB, MOVSW

The instructions MOVSB and MOVSW perform the same basic operation. An element of the string specified by the source index (SI) register with respect to the current data segment (DS) register is moved to the location specified by the destination index (DI) register with respect to the current extra segment (ES) register. The move can be performed on a byte or a word of data. After the move is complete, the contents of both SI and DI are automatically incremented or decremented by 1 for a byte move and by 2 for

a word move. The address pointers in Sl and DI increment or decrement, depending on how the direction flag (DF) is set. For example, the instruction

MOVSB

can be used to move a byte.

Figure -1- (a)shows an example of a program that uses MOVSB. This program is modified version of the block-move program shown in Fig.-1-(b). Note that the two MOV instructions that perform the data transfer and two INC instructions that update the pointer have been replaced with one move-string byte instruction. We have also made DS equal to ES.

```
        MOV     AX,DATASEGADDR
        MOV     DS,AX
        MOV     ES,AX
        MOV     SI,BLK1ADDR
        MOV     DI,BLK2ADDR
        MOV     CX,N
        CLD
NXTPT:  MOVSB
        LOOP    NXTPT
        HLT
```

Fig-1-(a)

```
        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- (b)

Compare String and Scan String - CMPSB/CMPSW and SCASB/SCASW

The compare-strings instruction can be used to compare two elements in the same or different strings: it subtracts the destination operand from the source operand and adjusts the flags accordingly. The result of subtraction is not saved; therefore, the operation does not affect the operands in any way.

An example of a compare strings instruction for bytes of data is

CMPSB

Again, the address in Sl points to the source element with respect to the current value in DS, and the destination element is specified by the contents of DI relative to the contents of ES.

When executed, the operands are compared, the flags are adjusted, and both SI and DI are updated so that they point to the next elements in their respective strings.

The scan-string instruction is similar to compare strings; however, it compares the byte or word element of the destination string at the physical address derived from DI and ES to the contents of AL or AX, respectively. The flags are adjusted based on this result and DI incremented or decremented.

Ex: Explain the function of the following sequence of instructions

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

Solution:

The first 5 instructions initialize internal registers and set up a data segment the loop in the program searches the 15 memory locations starting from Memory location A001Hfor the data stored in DL (05H). As long as the value In DL is not found the zero flag is reset, otherwise it is set. The LOOPNE Decrements CX and checks for CX=0 or ZF =1. If neither of these conditions is met the loop is repeated. If either condition is satisfied the loop is complete. Therefore, the loop is repeated until either 05 is found or all locations in the address range A001H through A00F have been checked and are found not to contain 5.

Ex: Implement the previous example using SCAS instruction.

Solution:

```
        MOV AX, 0H
        MOV DS, AX
        MOV ES, AX
        MOV AL, 05
        MOV DI, A001H
        MOV CX, 0FH
        CLD
AGAIN:  SCASB
        LOOPNE AGAIN
```

Load and Store String- LODSB/LODSW and STOSB/STOSW

The last two instructions in Fig.(a), load string and store string, are specifically provided to move string elements between the accumulator and memory. LODSB loads a byte from a string in memory into AL. The address in SI is used relative to DS to determine the address of the memory location of the string element; SI is incremented by 1 after loading. Similarly, the instruction LODSW indicates that the word- string element at the physical address derived from DS and SI is to be loaded into AX. Then the index in
SI is automatically incremented by 2.

On the other hand, STOSB stores a byte from AL into a string location in memory. This time the contents of ES and DI are used to form the address of the storage location in memory.

Ex: Write a program loads the block of memory locations from 0A001H through 0A00FH with number 5H.

Solution:

```
        MOV AX, 0H
        MOV DS, AX
        MOV ES, AX
        MOV AL, 05
        MOV DI, A001H
        MOV CX, 0FH
        CLD
AGAIN:  STOSB
        LOOP AGAIN
```

Repeat String - REP

In most applications, the basic string operations must be repeated in order to process arrays of data. Inserting a repeat prefix before the instruction that is to be repeated does this, the *repeat prefixes* of the 8086 are shown in table below

For example, the first prefix, **REP**, caused the basic string operation to be repeated until the contents of register CX become equal to 0. Each time the instruction is executed, it causes CX to be tested for 0. If CX is found not to be 0, it is decremented by 1 and the basic string operation is repeated. On the other hand, if it is 0, the repeat string operation is done and the next instruction in the program is s executed, the repeat count must be loaded into CX prior to executing the repeat string instruction.

Prefix	Used with:	Meaning
REP	MOVS STOS	Repeat while not end of string CX≠ 0
REPE / REPZ	CMPS SCAS	Repeat while not end of string and strings are equal CX≠ 0 and ZF =1
REPNE / REPNZ	CMPS SCAS	Repeat while not end of string and strings are not equal CX≠ 0 and ZF =0

The prefixes REPE and REPZ stand for the same function. They are meant for use with the CMPS and SCAS instructions. With REPE/REPZ, the basic compare or scan operation repeats as long as both the count in CX is not equal to 0 and ZF is 1. The first condition, CX not equal to 0, indicates that the end of the string has not yet reached, and the second condition, ZF = 1, indicates that the elements that were compare are equal.

The last prefix, REPNE/REPNZ, works similarly as the REPE/REPZ, except that now the operation is repeated as long as CX is not equal to 0 and ZF is 0. That is, the comparison or scanning is performed as long as the string elements are unequal and the end of the string is not yet reached.

Ex: write a program to copy a block of 32 consecutive bytes from the block of memory locations starting at address 2000H in the current Data Segment(DS) to a block of locations starting at address 3000H in the current Extra Segment (ES).

Solution:

```
CLD
MOV AX, data_seg
MOV DS, AX
MOV AX, extra_seg
MOV ES, AX
MOV CX, 20H
MOV SI, 2000H
MOV DI, 3000H
REP  MOVSB
```

Ex: Write a program that scans the 70 bytes start at location D0H in the current Data Segment for the value **45H** , if this value is found replace it with the value **29H** and exit scanning.

Solution:

```
          MOV AX,EXSTRA-SEGMENT ADDRESS
          MOV ES,AX
          CLD
          MOV DI, 00D0H
          MOV CX, 0046H
          MOV AL, 45H
          REPNE SCASB
          JZ FOUND
          HLT
FOUND:   DEC DI
          MOV  [DI], 29H
          HLT
```