String Handling Instructions

String Handling Instructions:

String handling instructions allow the programmer to manipulate large blocks of data with relative ease.

String handling instructions use the direction flag, SI and DI registers. The Direction Flag (DF) selects auto-increment or auto-decrement operation for the DI and SI registers during string operations. Whenever a string instruction transfers a byte, the contents of SI and/or DI increment or decrement by 1. In case of a word, the contents of SI and/or DI increment or decrement by 2.

Format	Operation	Mode	Effect
CLD	Clear DF; (DF) ✍ 0	Auto Increment	SI ✍ SI + 1; DI ✍ DI +1
STD	Set DF; (DF) ✍ 1	Auto Decrement	SI ✍ SI - 1; DI ✍ DI –1

Table 16. 1: Auto- incrementing and decrementing in string instructions

The LODS Instruction:

LODS loads AL or AX with data stored at the data-segment offset address indexed by the SI register. The LODSB causes a byte to be loaded into AL, and the LODSW causes a word to be loaded into AX.

The STOS Instruction:

The STOS instruction stores AL or AX at the extra-segment memory location addressed by the DI register, (in fact ES:DI). The STOSB stores a byte in AL at the extra-segment memory indicated by DI. The STOSW stores a word in AX at the extra-segment memory indicated by DI. Program 9.1 gives an example on the use of STOS instruction to clear the video memory.

The MOVS Instruction:

The MOVS instruction transfers data from one memory location to another. This is the only memory-to-memory transfer allowed in the Intel family of Microprocessors. The MOVS instruction transfers a byte or a word from the data-segment location addressed by SI to the extra-segment location addressed by DI. The pointers then increment or decrement as indicated by the direction flag (
Table 16. **2**).

Mnemonics	Meaning	Format	Operation As per Direction Flag	Flags affected
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
MOVS	Move string	**MOVSB** **MOVSW**	((ES)0 + (DI)) ✍ ((DS)0+(SI)) (SI) ✍ (SI) ? 1 or 2 (DI) ✍ (DI) ? 1 or 2	None

Note: B stands for Byte and W for Word.

Table 16. 2: Basic String Handling Instructions

Example of a move string:

Below is an example of the MOVS instruction. The same example is repeated later but with the use of the REP prefix.

```
        MOV AX, @DATA
        MOV DS, AX
        MOV ES, AX              ; Make ES = DS
        LEA SI, BLK1            ; Source address for block1
        LEA DI, BLK2            ; Destination address for block2
        MOV CX, N               ; N = number of bytes to move
        CLD                     ; Set Auto-Increment mode
NEXT:   MOVSB                   ; Move one byte at a time
        LOOP NEXT
```

String Comparisons:

In order to allow a section of memory to be compared against a constant or another section of memory, the String Scan instruction SCAS (Table 16. 3) is used. The SCAS instruction compares the content of the AL register with a byte block of memory, or the AX register with a word block of memory. The opcode used for byte comparison is SCASB and for word comparison is SCASW (Table 16. 3).

The Compare Strings Instruction (CMPS) compares two sections of memory data as bytes (CMPSB), or words (CMPSW). The contents of the data-segment memory indicated by SI are compared with the contents of the data-segment memory indicated by DI. The CMPS instruction increment both SI and DI if the direction flag (DF) is zero, or decrements both of them if DF is set to one.

The CMPS instruction is normally used with either the REPE or REPNE prefix. Alternates to these prefixes are REPZ (*repeat while zero*) and REPNZ (*repeat while not zero*), though REPE and REPNE are more common (Table 16. 4).

Mnemonics	Meaning	Format	Operation	Flags affected
CMPS	Compare strings	**CMPSB** **CMPSW**	Set flags as per: ((ES)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

Note: B stands for Byte and W for Word.

Table 16. 3: String Compare Instructions

Repeat Prefixes:

The REP prefix:

The REP prefix is added to any data transfer or compare instruction, except the LODS instruction. The REP prefix causes the CX register to decrement by 1 each time the string instruction executes. If CX reaches 0, the instruction terminates and the program continues with the next sequential instruction. The following example illustrates the of a move string using the REP prefix:

```
        MOV AX, @DATA
```

```
MOV DS, AX
MOV ES, AX                ; Make ES = DS
CLD                       ; Set Auto-Increment mode
MOV CX, 20H
MOV SI, OFFSET DATA1
MOV DI, OFFSET DATA2
REP MOVSB
```

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

Note: B stands for Byte and W for Word.

Table 16. 4: Prefixes fo use with basic string instructions

Examples on the use of the SCAS and CMPS instructions:

The following example shows how to search a memory section of 100 bytes in length and starting at location BLOCK. The program searches if any location contains the value 45H.

```
MOV DI, OFFSET BLOCK      ;address data
CLD                       ;auto-increment
MOV CX, 100               ;load counter
MOV AL, 45H               ;AL = 45H
REPNE SCASB               ;search
```

The next example illustrates a short procedure that compares two sections of memory searching for a match. The CMPSB instruction is prefixed with a REPE. This causes the search to continue as long as an equal condition exists. When the CX register becomes 0, or an unequal condition exists, the CMPSB instruction stops execution. After the CMPSB instruction ends, the CX register is zero or the flags indicate an equal condition when the two strings match. If CX is not zero or the flags indicate a not-equal condition, the strings do not match.

```
MATCH     PROC FAR
          MOV SI, OFFSET LINE
          MOV DI, OFFSET TABLE
          CLD
          MOV CX, 10
          REPE CMPSB
          RET
MATCH     ENDP
```