ICS 312 String Instructions

Textbook Reading (Jones): Chapter 17

String - a byte or word array.

Operations that can be performed with string instructions:

- copy a string into another string
- search a string for a particular byte or word
- store characters in a string
- compare strings of characters alphanumerically

Direction Flag

- one of 8086 processor control flags.
- controls the direction of string operations:
- DF = 0 => forward (left to right) processing
- DF = 1 => backward (right to left) processing

CLD - clears the DF; sets DF = 0
STD - sets DF = 1

Moving (Copying) Strings

Instructions:

MOVSB - copies contents of BYTE given by **DS:SI** into **ES:DI**
MOVSW - copies contents of WORD given by **DS:SI** into **ES:DI**
MOVSD - copies contenst of DOUBLE WORD given by **DS:SI** into **ES:DI**

Notes:

- the string instructions use **DS:SI** as the Source string and **ES:DI** as the destination string. (For the string instructions, DI references an offset in the ES by default).
- MOVSD requires the target processor be at least .386 (32-bit processor)

Example:

```
.DATA
STRING1 DB 'HELLO'
STRING2 DB 5 DUP (0)

.CODE
        MOV AX, @DATA
        MOV DS, AX            ; init DS
        MOV ES, AX            ; init ES

        LEA SI, STRING1       ; source
        LEA DI, STRING2       ; destination
        CLD                   ; DF = 0

        MOVSB                 ; mov 1st byte
        MOVSB                 ; mov 2nd byte
```

Note: MOVSB moves only 1 byte at a time. Set CX = count and use the REP prefix to move a specified number of bytes.

Example:

```
        CLD                     ; forward direction
        PUSH DS                 ; set ES = DS
        POP ES
        LEA SI, STRING1         ; set SI to source
        LEA DI, STRING2         ; set DI to destination
        MOV CX, 5

        REP MOVSB               ; copies 5 chars
```

Note: both SI and DI are incremented for each byte that is copied from SI to DI (in the forward direction)

Reverse:

```
        STD                          ; reverse direction
        PUSH DS
        POP ES                       ; ES = DS
        LEA SI, STRING1+4            ; end of string
        LEA DI, STRING2+4            ; end of string
        MOV CX, 5

        REP MOVSB                    ; copy 5 chars
```

Note: both SI and DI are decremented by one for each byte that is copied from SI to DI (in the reverse direction).

- MOVSW works the same way as MOVSB and moves one word (2 bytes) at a time. Consequently, SI/DI will be incremented or decremented by 2 bytes for each word copied.
- MOVSD moves one double word (4 bytes) at a time. Consequently, SI and DI will be incremented or decremented by 4 for each word copied.
- Note: take reverse byte ordering into account when moving WORD or DOUBLE WORD strings.

Storing Strings

Instructions:

STOSB - copies contents of AL to BYTE address given by ES:DI. DI is incremented/decremented by 1.
STOSW - copies the contents of AX to the WORD address given by ES:DI. DI is incremented/decremented by 2.
STOSD - copies contents of EAX to the DOUBLE WORD address given by ES:DI. DI is incremented/decremented by 4.

Example:

```
    MOV AX, @DATA
    MOV ES, AX ; initialize ES

    LEA DI, STRING1         ; assume BYTE string
    CLD
    MOV AL, 'A'

    STOSB                   ; store 1st byte of A
    STOSB                   ; store 2nd byte of A
```

Load String

Instructions:

LODSB - moves the BYTE at address DS:SI into AL. SI is incremented/decremented by 1.
LODSW - moves the WORD at address DS: SI into AX. SI is incremented/decremented by 2.
LODSD - moves the DOUBLE WORD at address DS:SI into EAX. SI is incremented/decremented by 4.

Example:

```
        MOV AX, @DATA
        MOV DS, AX
        LEA SI, STRING1
        CLD

        LODSB
        LODSB
```

Loads two bytes from STRING1 into AL (second byte overwrites the first).

Scan String

Instructions:

SCASB - compares **BYTE** at **ES:DI** with **AL** and sets flags according to result.
SCASW - compares **WORD** at **ES:DI** with **AX** and sets flags.
SCASD - compares **DOUBLE WORD** at **ES:DI** with **EAX** and sets flags.

Example:

```
.DATA
STRING1 DB 'ABC'

.CODE
        MOV AX, @DATA
        MOV AX, ES ; initialize ES

        CLD ; left to right
        LEA DI, STRING1
        MOV AL, 'B' ; target character

        SCASB ; scan first byte
        SCASB ; scan 2nd byte
```

Note: when the target ("B") is found, ZF = 1 and DI points to the byte following the target since DI is automatically incremented by SCASB.

Also, set CX = count and use:

```
        REPNE SCASB
        REPNZ SCASB
```

to repeat the scan until the target byte is found, or until the entire string has been searched (i.e., CX = 0).

Compare String

Instructions:

CMPSB - compares BYTE at ES:DI with BYTE at DS:SI and sets flags.
CMPSW - compares WORD at ES:DI with WORD at DS:SI and sets flags.
CMPSD - compares DOUBLE WORD at ES:DI with WORD at DS:SI and sets flags.

Example:

```
.DATA
STRING1 DB 'ACD'
STRING2 DB 'ABC'

.CODE
        MOV AX, @DATA
        MOV DS, AX
        MOV ES, AX

        CLD
        LEA SI, STRING1
        LEA DI, STRING2
        MOV CX, 3            ; string length

        REPE CMPSB           ; repeat while strings match
```

Increments (or decrements) each string pointer and successively compares bytes until there is a mismatch between the bytes being compared, or until CX = 0.

CMPSB can be used to determine whether two strings match, or whether one string is a substring of another string.

BYTE, WORD, and DOUBLE WORD form of string instructions:

INSTRUCTION	**DEST**	**SOURCE**	**BYTE**	**WORD**	**DOUBLE WORD**
Move String	ES:DI	DS:SI	MOVSB	MOVSW	MOVSW
Compare String	ES:DI	DS:SI	CMPSB	CMPSW	CMPSW
Store string	ES:DI	AL/AX/EAX	STOSB	STOSW	STOSW
Load string	AL/AX/EAX	DS:SI	LODSB	LODSW	LODSW
Scan string	ES:DI	AL/AX/EAX	SCASB	SCASW	SCASW

The operands are **IMPLICIT**.

GENERAL FORM of STRING instructions (optional):

MOVS *destination, source*
CMPS *destination, source*
STOS *destination, source*
LODS *destination, source*
SCAS *destination, source*

To use the general form of the instructions, SI must be in the DS, DI must be in ES, and the strings must both be of the same type.

Example:

```
.DATA
STRING1   DB   'ABCDE'
```

```
STRING2   DB   'EFGH'
STRING3   DB   'IJKL'
STRING4   DB   'MNOP'
STRING5   DW   1,2,3,4,5
STRING6   DW   7,8,9
```

Then, the following pairs of instructions are equivalent:

General form of instruction:	**Specific:**
MOVS STRING2, STRING1	MOVSB
MOVS STRING6, STRING5	MOVSW
LODS STRING4	LODSB
LODS STRING5	LODSW
SCAS STRING1	SCASB
SCAS STRING6	STOSW

Note: When the general forms of the instructions are used as above, you must still pre-load the addresses into SI/DI because the general form of the instruction is actually converted into the BYTE or WORD form of the instruction by the assembler translator. (I.e., the general form of the instruction requires just as much preparation as the byte, word, or double word forms of the instructions.)