CSE 331L / EEE 332L: Microprocessor Interfacing & Embedded System

Section: 9, Spring 2020

Lab - 06 (String Operations)

String Instructions:

1. CLD, STD
2. MOVSB, MOVSW
3. REP, REPE, REPNE, REPZ, REPNZ (prefix)
4. SCASB. SCASW
5. CMPSB, CMPSW

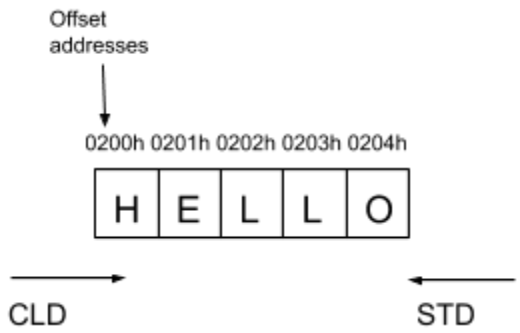
1. CLD, STD:

sets the value of the direction flag: 0 and 1 respectively. It determines the direction in which string operation will proceed.

CLD : DF=0 (SI, DI increments: byte moves from left to right)

STD : DF=1 (SI, DI decrements: byte moves from right to left)

```
ORG 100H

.MODEL SMALL
.DATA
        STR DB "HELLO$"

.CODE

MOV AX, @DATA
MOV DS, AX

CLD

LEA SI, STR
```



2. MOVSB, MOVSW:

```
.DATA
        STR1 DB "HELLO$"
        STR2 DB 5 DUP (?), '$'
.CODE
MAIN PROC
        MOV AX, @DATA
        MOV DS, AX
        LEA SI, STR1
        LEA DI, STR2
        MOV BL, [SI]
        MOV [DI], BL
PRINT:
        MOV AH, 9
        LEA DX, STR2
        INT 21H
EXIT:
        MOV AH, 4CH
        INT 21H
MAIN ENDP
END MAIN
```

```
.DATA
        STR1 DB "HELLO$"
        STR2 DB 5 DUP (?), '$'
.CODE
MAIN PROC
        MOV AX, @DATA
        MOV DS, AX
        MOV ES, AX

        CLD

        LEA SI, STR1
        LEA DI, STR2

        MOVSB
PRINT:
        MOV AH, 9
        LEA DX, STR2
        INT 21H
EXIT:
        MOV AH, 4CH
        INT 21H
MAIN ENDP
END MAIN
```

3. **REP:**
    - Checks the value of cx and repeats if not zero

```
MAIN PROC
        MOV AX, @DATA
        MOV DS, AX
        MOV ES, AX

        CLD

        LEA SI, STR1
        LEA DI, STR2
MOVE:
        CMP [SI], '$'
        JE PRINT
        MOVSB
        JMP MOVE
PRINT:
        MOV AH, 9
        LEA DX, STR2
        INT 21H
        JMP EXIT
EXIT:
        MOV AH, 4CH
        INT 21H
MAIN ENDP
```

```
MAIN PROC
        MOV AX, @DATA
        MOV DS, AX
        MOV ES, AX

        CLD

        LEA SI, STR1
        LEA DI, STR2
MOVE:
        REP MOVSB
PRINT:
        MOV AH, 9
        LEA DX, STR2
        INT 21H
        JMP EXIT
EXIT:
        MOV AH, 4CH
        INT 21H
MAIN ENDP
```

4. **SCASB, SCASW:**
    - Target byte in AL
    - Checks string pointed by ES:DI
    - Subtracts string byte from contents of AL hence, checks ZF
      ZF = 1 ;comparison result=0(equal), so ZF becomes 1
      ZF = 0 ;comparison result!=0(not equal), so ZF becomes 0

```
.DATA
        STR1 DB "HELLO CLASS!$"
        MSG1 DB "Character found$"
        MSG2 DB "Character not
found$"
.CODE
MAIN PROC
        MOV AX, @DATA
        MOV DS, AX

        MOV AL, 'C'
        LEA DI, STR1
SCAN:
        CMP [DI], '$'
        JE PRINT_N

        CMP [DI], AL
        JE PRINT

        INC DI
        JMP SCAN
PRINT:
        MOV AH, 9
        LEA DX, MSG1
        INT 21H
        JMP EXIT
PRINT_N:
        MOV AH, 9
        LEA DX, MSG2
        INT 21H
        JMP EXIT
EXIT:
        MOV AH, 4CH
        INT 21H
MAIN ENDP
```

```
.DATA
  STR1 DB "HELLO CLASS!$"
        MSG1 DB "Character found$"
        MSG2 DB "Character not found$"
.CODE

MAIN PROC
        MOV AX, @DATA
        MOV ES, AX

        CLD
        LEA DI, STR1
        MOV AL, 'C'
SCAN:
        CMP [DI], '$'
        JE PRINT_N

        SCASB
        JE PRINT

        JMP SCAN
PRINT:
        MOV AH, 9
        LEA DX, MSG1
        INT 21H
        JMP EXIT
PRINT_N:
        MOV AH, 9
        LEA DX, MSG2
        INT 21H
        JMP EXIT

EXIT:
        MOV AH, 4CH
        INT 21H
MAIN ENDP
```

REPNE/REPNZ:

```
MAIN PROC
        MOV AX, @DATA
        MOV ES, AX

        CLD
        LEA DI, STR1
        MOV AL, 'B'
        MOV CX, 12
COPY:
        REPNE SCASB
        JNE PRINT_N
PRINT:
        MOV AH, 9
        LEA DX, MSG1
        INT 21H
        JMP EXIT
PRINT_N:
        MOV AH, 9
        LEA DX, MSG2
        INT 21H
        JMP EXIT
EXIT:
        MOV AH, 4CH
        INT 21H
MAIN ENDP
```

REPNE/REPNZ:

1. Repeatedly checks if the target matches any string character
2. Update DI
3. Decrements CX until the target is found (ZF = 1) or CX = 0

5. **CMPSB, CMPSW:**
    - Subtracts string byte with address ES:DI from that with address DS:SI

```
.DATA
  STR1 DB "HELLO CLASS!$"
  STR2 DB "HELLO CLASS!$"
  MSG1 DB "STRINGS MATCH$"
  MSG2 DB "STRINGS DO NOT MATCH$"

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

        CLD
        LEA SI, STR1
        LEA DI, STR2
        MOV CX, 12
COMPARE:
        REPE CMPSB
        JNE PRINT_N
PRINT:
        MOV AH, 9
        LEA DX, MSG1
        INT 21H
        JMP EXIT
PRINT_N:
        MOV AH, 9
        LEA DX, MSG2
        INT 21H
        JMP EXIT
EXIT:
        MOV AH, 4CH
        INT 21H
MAIN ENDP
```

REPE/REPZ:

1. Repeatedly executes CMPSB/CMPSW
2. Updates SI and DI
3. Decrements CX until there is a mismatch or CX = 0

TASKS:

1. Write a program that will copy str1 to str2 in reverse order.

2. Write a program that will check if there is any vowel in a string.

3. Write a program that will check if a string is a palindrome using CMPSB/CPMSW instruction.