COURSE: UCS1502 - MICROPROCESSORS AND INTERFACING

Instruction set of 8086 – Part 2 (String manipulation instructions)

Dr. K. R. Sarath Chandran
Assistant Professor, Dept. of CSE

This presentation covers

- Instruction set of 8086 (String manipulation instructions)

Learning Outcome of this module

- To understand string manipulation instructions of 8086

Contents

- Different types of string related instructions of 8086
- Explanation of all string manipulation instructions of 8086

Types of instructions in 8086

1. Data transfer instructions
2. Arithmetic and logical instructions
3. Branch instructions
4. Loop instructions
5. Machine control instructions
6. Flag manipulation instructions
7. Shift and rotate instructions
8. String manipulation instructions

String manipulation instructions

REP

Used to repeat the given instruction till CX = 0.

REP <instruction>

```
Step 1:   Check CX
          If CX ≠ 0 then
                    Do the following instruction
                    CX = CX – 1
                    Goto Step1
          Else
                    Exit
```

REPE / REPZ

Used to repeat the next instruction until CX=0 or ZF = 0.

REPE <instruction>

```
Step 1:   Check CX
          If CX ≠ 0 then
                    Do the following instruction
                    CX = CX – 1
                    If ZF=1 then goto Step1
                              else exit
          Else
                    Exit
```

REPNE / REPNZ

Used to repeat the next instruction until CX = 0 or ZF =1.

REPNE <instruction>

```
Step 1:   Check CX
          If CX ≠ 0 then
                    Do the following instruction
                    CX = CX – 1
                    If ZF=0 then goto Step1
                              else exit
          Else
                    Exit
```

String manipulation instructions

MOVSB / MOVSW - Used to copy the byte/word from one string to another. Count of the string should be in CX.

```
MOV CX,0006H
MOV SI,2000H
MOV DI,2500H
CLD
REP MOVSB
HLT
```

- Copy byte from DS:SI to ES:DI
- Increment or Decrement offset registers based on DF

repeat the given instruction till CX = 0.

```
assume cs:code,ds:data,es:extra
data segment
        count dw 0002h
        str1 db 11h,12h,13h,14h
data ends
extra segment
        str2 db 00h,00h,00h,00h
extra ends

code segment
        org 0100h
start:  mov ax,data
        mov ds,ax
        mov ax,extra
        mov es,ax
        mov cx,count
        mov si, offset str1
        mov di, offset str2
        cld
        rep movsb
        mov ah,4ch
        int 21h
code ends
end start
```

String manipulation instructions

CMPSB / CMPSW - Used to compare two string bytes/words.

```
MOV CX,[2400]
MOV SI,2000
MOV DI,2500
CLD
REPE CMPSB
MOV [1500],CX
HLT
```

- Compare 2 bytes from DS:SI and ES:DI
- Increment or decrement offset registers based on DF

Used to repeat the next instruction until CX=0 or ZF = 0.

```
-d 076a:0000
076A:0000  06 00 AA BB CC DD EE
076A:0010  00 00 00 00 00 00 00
```

```
-d 076c:0000
076C:0000  AA DB CC DD EE
```

After execution

```
-d 076a:0000
076A:0000  06 00 AA BB CC DD EE
076A:0010  04 00 00 00 00 00 00
```

```
assume cs:code,ds:data,es:extra
data segment
        count dw 0006h
        str1 db 0aah,0bbh,0cch,0ddh,0eeh
        org 0010h
        status dw 0000h
data ends
extra segment
        str2 db 0aah,0dbh,0cch,0ddh,0eeh
extra ends
code segment
        org 0100h
start:  mov ax,data
        mov ds,ax
        mov ax,extra
        mov es,ax
        mov cx,count
        mov si, offset str1
        mov di, offset str2
        cld
        repe cmpsb
        mov status, cx
        mov ah,4ch
    int 21h
        code ends
end start
```

String manipulation instructions

SCASB / SCASW - Used to scan a string and compare its bytes with a byte in AL or string word with a word in AX.

```
MOV CX,0006 ;  count of bytes
MOV DI,2500; string starting address should be in ES:DI
MOV AL,22H
CLD
REPNE SCASB
MOV [1500],CX
HLT
```

repeat the next instruction until CX = 0 or ZF =1.

- Compare byte in AL with a byte pointed by ES:DI.
- Increment or decrement offset register based on DF

String manipulation instructions

LODSB/LODSW

AL/AX ⇦ [DS:SI]; then SI is incremented (if DF=0) or decremented(if DF=1)

STOSB/STOSW

AL/AX ⇨ [ES:DI]; then DI is incremented (if DF=0) or decremented(if DF=1)

References

- Doughlas V. Hall, “Microprocessors and Interfacing, Programming and Hardware”, Second Edition, TMH.

Thank you