String Representation

Muhammad Afzaal
m.afzaal@nu.edu.pk

Book Chapter

- “Assembly Language for x86 Processors”
- Author “Kip R. Irvine”
- 6th Edition
- Chapter 9
  - Section 9.2

String Instructions (1/2)

- Five groups of instructions for processing array of **B**ytes, **W**ords and **D**ouble-words
- Called String Primitives but not limited to character arrays only
- These instructions use ESI/SI and EDI/DI registers to address memory
- Array indexes are repeated and incremented automatically

String Instructions (2/2)

Instruction	Description
MOVSB, MOVSW, MOVSD	Copy data from memory addressed by SI to memory addressed by DI
CMPSB, CMPSW, CMPSD	Compare the contents of memory addressed by SI to memory addressed by DI
SCASB, SCASW, SCASD	Compare the accumulator register (AL, AX or EAX) to the contents of memory addressed by DI
STOSB, STOSW, STOSD	Store the contents of accumulator register into memory location addressed by DI
LODSB, LODSW, LODSD	Load the contents of memory addressed by SI into the accumulator register

Direction Flag

- String instructions increment/decrement SI and DI based on the state of Direction Flag
- DF can be explicitly modified using instructions
  - `CLD` ;clear Direction Flag (forward direction)
  - `STD` ;set Direction Flag (reverse direction)

```
if(DF=0) then
    SI=SI+1
    DI=DI+1
else
    SI=SI-1
    DI=DI-1
end if
```

Value of DF	Effect on SI and DI	Address Sequence
Clear	Increment	Low-High
Set	Decrement	High-Low

Repeat Prefix

- REP (a repeat prefix) can be used just before MOVSB, MOVSW, MOVSD
- By default CX controls the number of repetitions

Prefix	Description
REP	Repeat while CX>0
REPZ, REPE	Repeat while ZF=1 and CX>0
REPNZ, REPNE	Repeat while ZF=0 and CX>0

REP Prefix

- Repeat while CX>0
- Value of CX is checked before execution of instruction
- If CX is zero, string instruction is not executed

```
while(CX≠0)
    execute the string instruction
    CX = CX – 1
end while
```

REPE, REPZ Prefixes

```
while(CX≠0)
    execute the string instruction
    CX = CX – 1
    if (ZF=0) then
        exit loop
    end if
end while
```

REPNE, REPNZ Prefixes

```
while(CX≠0)
    execute the string instruction
    CX = CX – 1
    if (ZF=1) then
        exit loop
    end if
end while
```

MOVSB, MOVSW, MOVSD (1/2)

- Copy data from memory location pointed to by SI to memory location pointed to by DI
- SI and DI are either incremented or decremented based on the value of DF
- SI/DI incremented/decremented by
  - 1 when used with MOVSB
  - 2 when used with MOVSW
  - 3 when used with MOVSD

MOVSB, MOVSW, MOVSD (2/2)

```
.data
    src DB “Hello World!”
    src_len DB $-src
    dst DB src_len DUP(?)
.code
    MOV CX, src_len
    MOV SI, OFFSET src
    MOV DI, OFFSET dst
    REP MOVSB
```

CMPSB, CMPSW, CMPSD (1/2)

- Compare memory operand pointed to by SI to memory operand pointed to by DI
- SI and DI are either incremented or decremented based on the value of DF
- SI/DI incremented/decremented by
  - 1 when used with CMPSB
  - 2 when used with CMPSW
  - 3 when used with CMPSD

CMPSB, CMPSW, CMPSD (2/2)

```
.data
    src DB “Hello World!”
    src_len DB $-src
    dst DB “Hello! World”
.code
    MOV CX, src_len
    MOV SI, OFFSET src
    MOV DI, OFFSET dst
    REPE CMPSB
```

SCASB, SCASW, SCASD (1/2)

- Compare the value in accumulator to the memory value pointed to by DI
- Useful when looking for a single value in string or array
- When combined with REPNE prefix, SCAS***X*** scans until either accumulator is matched a value in memory or CX=0
- When combined with REPE prefix, string is scanned while CX>0 and value in accumulator matches each subsequent value in string

SCASB, SCASW, SCASD (2/2)

```
.data
    dst DB “Hello! World”
.code
    MOV CX, src_len
    MOV DI, OFFSET dst
    MOV AL, ‘H’
    REPE SCASB
```

STOSB, STOSW, STOSD (1/2)

- Store the contents of accumulator in the memory addressed by DI
- When used with REP prefix, these instructions can be used to fill all elements of string with the same value

STOSB, STOSW, STOSD (2/2)

```
.data
    dst DB 5 DUP(?)
    dst_len DW $-dst
.code
    MOV CX, dst_len
    MOV DI, OFFSET dst
    MOV AL, 'H'
    REP STOSB
```

LODSB, LODSW, LODSD

- Load a byte/word from memory at SI into accumulator register
- Used to load a single value because older value is always overwritten in accumulator