Assembly Language LAB

Islamic University – Gaza
Engineering Faculty
Department of Computer Engineering
2013
ECOM 2125: Assembly Language LAB
Created by: Eng. Ahmed M. Ayash
Modified and Presented By: Eihab S. El-Radie

Lab # 10

String Primitive Instructions

Objective:

To learn more about strings in assembly language.

String Primitive Instructions:

1. Move string data (MOVSB, MOVSW, and MOVSD):

- These instructions copy (move) data from the memory location pointed to by ESI to the memory location pointed to by EDI.

MOVSB	Move (copy) bytes
MOVSW	Move (copy) words
MOVSD	Move (copy) doublewords

- ESI is automatically an offset in the segment addressed by **DS**, and EDI is automatically an offset in the segment addressed by **ES**.

- DS and ES are always set to the same value and you cannot change them.

```
mov ax,@data ; get addr of data seg
mov ds,ax ; initialize DS
mov es,ax ; initialize ES
```

Using a Repeat Prefix:

- A repeat prefix can be inserted just before MOVSB, MOVSW, or MOVSD.
- ECX controls the number of repetitions

REP	Repeat while ECX > 0
REPZ, REPE	Repeat while the Zero flag is set and ECX > 0
REPNZ, REPNE	Repeat while the Zero flag is clear and ECX > 0

- **Example: Copy a String**

  In the following example, MOVSB moves 10 bytes from **string1** to **string2**

```
cld                     ; clear direction flag DF=0
mov esi,OFFSET string1  ; ESI points to source
mov edi,OFFSET string2  ; EDI points to target
mov ecx,10              ; set counter to 10
rep movsb
```

- ESI and EDI are automatically incremented when MOVSB repeats. This behavior is controlled by the CPU's Direction flag.
- ESI and EDI are automatically incremented or decremented:

• MOVSB increments/decrements by 1

• MOVSW increments/decrements by 2

• MOVSD increments/decrements by 4

Direction Flag:

- The Direction flag controls the incrementing or decrementing of ESI and EDI.

Value of the Direction Flag	Effect on ESI and EDI	Address Sequence
Clear	Incremented	Low-high
Set	Decremented	High-low

- The Direction flag can be explicitly modified using the CLD and STD instructions:

```
CLD    ; clear Direction flag (forward direction)
STD    ; set Direction flag (reverse direction)
```

2. Compare strings (CMPSB, CMPSW, and CMPSD):

- These instructions each compare a memory operand pointed to by ESI to a memory operand pointed to by EDI.

CMPSB	Compare bytes
CMPSW	Compare words
CMPSD	Compare doublewords

- You can use a repeat prefix with CMPSB, CMPSW, and CMPSD. The Direction flag determines the incrementing or decrementing of ESI and EDI.

- **Example1:** If source > target, the code jumps to label L1; otherwise, it jumps to label L2

```
.data
source DD 1234h
target DD 5678h
.code
mov esi,OFFSET source
mov edi,OFFSET target
cmpsd              ; compare doublewords
ja L1              ; jump if source > target
jmp L2             ; jump if source <= target
```

- **Example2: Compare multiple doublewords**

```
mov esi,OFFSET source
mov edi,OFFSET target
cld                         ; direction = forward
mov ecx,LENGTHOF source     ; repetition counter
repe cmpsd                  ; repeat while equal
```

- Repeat prefix often used:
  - REPE (REPZ)
  - REPNE (REPNZ)

3. Scan string (SCASB, SCASW, and SCASD):

- These instructions compare a value in AL/AX/EAX to a byte, word, or doubleword, respectively, addressed by EDI.

- Useful types of searches:
  - Search for a specific element in a long string or array.
  - Search for the first element that does not match a given value.

- **Example: Scan for a Matching Character,** Search for the letter 'F' in a string named alpha:

```
.data
alpha BYTE "ABCDEFGH",0
.code
mov edi,OFFSET alpha    ; EDI points to the string
mov al,'F'              ; search for the letter F
mov ecx,LENGTHOF alpha  ; set the search count
cld                     ; direction = forward
repne scasb             ; repeat while not equal
jnz quit                ; quit if letter not found
dec edi                 ; found: back up EDI
```

  - JNZ was added after the loop to test for the possibility that the loop stopped because ECX = 0 and the character in AL was not found.

4. Store string data (STOSB, STOSW, and STOSD):

- These instructions store the contents of AL/AX/EAX, respectively, in memory at the offset pointed to by EDI.

- **Example: Fill an array with 0FFh**

```
.data
Count = 100
string1 BYTE Count DUP(?)
.code
mov al,0FFh                ; value to be stored
mov edi,OFFSET string1     ; ES:DI points to target
mov ecx,Count              ; character count
cld                        ; direction = forward
rep stosb                  ; fill with contents of AL
```

5. Load accumulator from string (LODSB, LODSW, and LODSD):

- LODSB, LODSW, and LODSD load a byte, word, or doubleword from memory at ESI into AL/AX/EAX, respectively.

- Meaning:

  mov al, [esi]
  inc esi

- **Example: Array Multiplication**

```
.data
array DD 1,2,3,4,5,6,7,8,9,10
multipex101lier DD 10
.code
    cld                     ; direction = up
    mov esi,OFFSET array    ; source index
    mov edi,esi             ; destination index
    mov ecx,10              ; loop counter
L1: lodsd                   ; copy [ESI] into EAX
    mul multiplier          ; multiply by a value
    stosd                   ; store EAX at [EDI]
    loop L1
```

Lab work:

Excercise1:

Write a procedure named **str_substring** that take some characters from a string and moves them to other String. There are two input parameters: a start address and end address of characters to be moved.

Administrator: cmd - edit ex1.asm

File Edit Search View Options Help
C:\Masm615\Assembly\Lab10\ex1.asm

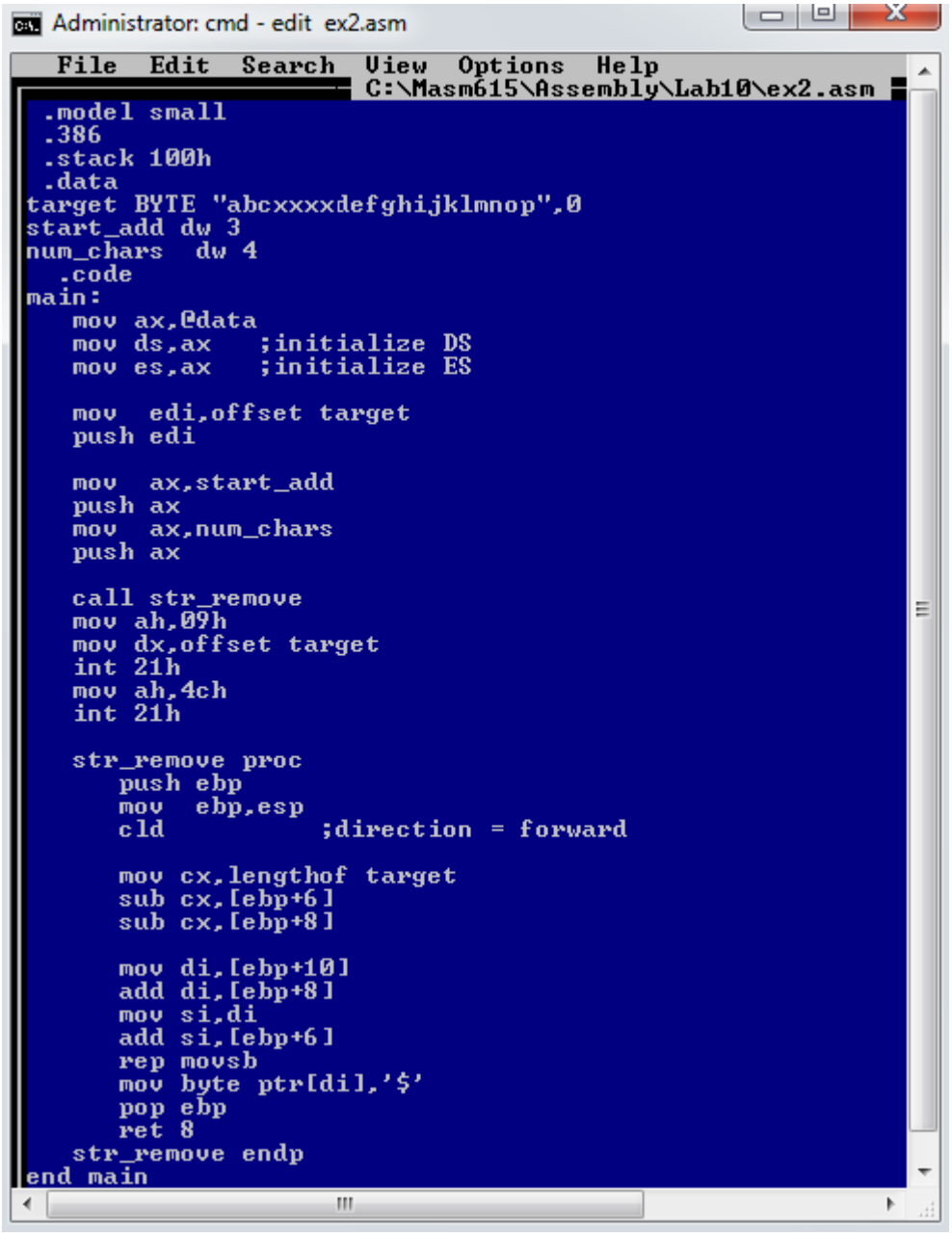
```
.model small
.386
.stack 100h
.data
source db 'Lab 10: Strings in assembly',0
target db ?
start_add dw 16
end_add dw 17
    .code
main:
    mov ax,@data
    mov ds,ax     ;initialize DS
    mov es,ax     ;initialize ES

    mov eax,offset source
    push eax
    mov eax,offset target
    push eax

    mov ax,start_add
    push ax
    mov ax,end_add
    push ax
    call str_substring
    mov ah,09h
    mov dx,offset target
    int 21h
    mov ah,4ch
    int 21h

str_substring proc
    push ebp
    mov ebp,esp
    cld               ;direction = forward
    mov cx,[ebp+6]    ;17
    sub cx,[ebp+8]    ;17-16=1
    inc cx            ;2
    mov edi,[ebp+10]
    mov esi,[ebp+14]
    add esi,[ebp+8]   ;starting
    rep movsb
    mov byte ptr[edi],'$'
    pop ebp
    ret 12
str_substring endp
end main
```

Output:

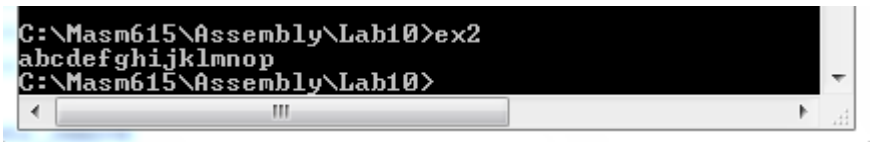
```
C:\Masm615\Assembly\Lab10>ex1
in
C:\Masm615\Assembly\Lab10>
```

Excercise2:

Write a procedure named **str_remove** that removes **n** characters from a string. Pass a pointer to the position in the string where the characters are to be removed. Pass an integer specifying the number of characters to remove.



Output:



Homework:

Write a procedure Str_length to calculate the length of a null-terminated string and returns the length in the EAX register.

Use this String:

target BYTE "abcxxxxdefghijklmnop",0