Assembly Program Practice Programming

Dept. of CSIE,
Fu Jen Catholic University,
Hsin Chuang, 24205,
Taipei Taiwan.

Str_concatenate program

Program No.1

Write a procedure named **str_concat** that concatenates a source string to the end of a target string. Sufficient space must exist in the target string to accommodate the new characters. Pass pointers to the source and target strings. Here is a sample call:

```
.data
targetStr BYTE "LAST Week of class and preparation ",0
sourceStr BYTE "Before final exam.",0

.code
INVOKE Str_concat, ADDR targetStr, ADDR sourceStr
```

First, one must declare the **PROTO** in order to work with the **INVOKE** function.

```
INCLUDE Irvine32.inc

Str_concat PROTO,
 source:PTR BYTE, ; source string
 target:PTR BYTE; target string

.data
targetStr BYTE "LAST Week of class and so preparation ",0
sourceStr BYTE "Before final exam.",0
```

```
.code
main PROC
call Clrscr

; Sample procedure call
INVOKE Str_concat, ADDR targetStr, ADDR sourceStr

; Display the target string
mov  edx,OFFSET targetStr
call WriteString
call Crlf

exit
main ENDP
```

```
;-----------------------------------------------------
Str_concat PROC USES eax ecx esi edi,
 target:PTR BYTE, ; source string
 source:PTR BYTE; target string
; Copy a string from source to target.
; Requires: the target string must contain enough
;          space to hold a copy of the source string.
;-----------------------------------------------------
INVOKE Str_length,target            ; get length of target string
addtarget,eax                       ; move to end of target string
INVOKE Str_length,source            ; get length of source string
mov ecx, eax                        ; insert length in REP count
inc ecx                             ; add 1 for null byte
mov esi,    source                  ; move the source pointer
mov edi,    target                  ; move the target pointer
cld                                 ; direction = up
repmovsb                            ; copy the string
ret
Str_concat ENDP

END main
```

Let us take another program and try to write this program called **str_nextWord** Procedure

Str_nextWord Procedure

Write a procedure called str_nextWord that scans a string for the first occurrence of a certain delimiter character and replaces the delimiter character with a null byte. After the call, if the delimiter was found, the Zero flag is set and EAX contains the offset of the next character beyond the delimiter. Otherwise, the Zero flag is clear and EAX is undefined. The following example code passes the address of Target and a comma as the delimiter:

```
.data
target BYTE “Johnson,Calvin,goes,to sleep,by,tonight”,0
.code
INVOKE Str_nextword, ADDR targe, “,”
JNZ notfound
```

Run of the program is as follows:

C:\Windows\system32\cmd.exe

```
Johnson,Calvin,goes,to sleep,by,tonight
Calvin,goes,to sleep,by,tonight
goes,to sleep,by,tonight
to sleep,by,tonight
by,tonight
tonight
請按任意鍵繼續 . . .
```

微軟新注音 半 :

```
INCLUDE Irvine32.inc

Str_nextWord PROTO,
pString:PTR BYTE,              ; pointer to string
delimiter:BYTE                 ; delimiter to find

.data
testStr BYTE  "Johnson,Calvin,goes,to sleep,by,tonight",0
.code
main PROC
call Clrscr
mov  edx,OFFSET testStr        ; display starting string
call WriteString
call Crlf

; Loop through the string, replace each delimiter, and
; display the remaining string.
```

```
mov esi,  OFFSET testStr

L1:  INVOKE Str_nextword, esi, ","   ; look for delimiter
jnz  Exit_prog                       ; quit if not found
mov esi, eax                         ; point to next substring
mov  edx,  eax
call WriteString        ; display remainder of string
call Crlf
jmp L1

Exit_prog:
exit
main ENDP
```

```
;-----------------------------------------------------------
Str_nextWord PROC,
pString:PTR BYTE,; pointer to string
delimiter:BYTE; delimiter to find
;
; Scans a string for the first occurrence of a certain delimiter
; character and replaces the delimiter with a null byte.
; Returns: If ZF = 1, the delimiter was found and EAX contains
;   the offset of the next character beyond the delimiter.
;   Otherwise, ZF = 0 and EAX is undefined.
;-----------------------------------------------------------
push  esi
mov   al, delimiter
mov   esi, pString
Cld                     ; clear Direction flag (forward)
L1:
  lodsb                 ; AL = [esi], inc(esi)
cmp al, 0               ; end of string?
je    L3                ; yes: exit with ZF = 0
cmp al,delimiter        ; delimiter found?
jne L1                  ; no: repeat loop
```

```
L2:
mov BYTE PTR [esi-1], 0      ; yes: insert null byte
mov eax, esi                 ; point EAX to next character
Jmp Exit_proc                ; exit with ZF = 1

L3:
   or    al,1                ; clear Zero flag

Exit_proc:
pop esi
ret
Str_nextWord  ENDP

END main
```

Let us write a third program that is the solution to Quiz 2

小考2 ODD Program

Given the following main procedure, write the needed two procedures to complete the program to run and show the output to the Instructor or the Assistants. Get a signature. Write the code needed for the two procedures before you submit your exam paper.

; 2018 ODD program for Quiz No.2. (ODDmain.asm)

COMMENT !

This program requires two procedures. 1). The first procedure uses INVOKE method **Get_frequencies** of the letters appearing in a string. **Ex:** "Not Now" N: 2 o:2 t:1 w:1 (Shows letter N appears 2 times and o appears 2 times and t appears 1 time and w appears 1 time.) It only calculates the frequencies and stores on a table with the letters appearing.

2). The second procedure uses "call **DisplayTable** procedure" to display the frequencies.

The sample output will be shown on the slides. you can download them.

!

```
include Irvine32.inc

Get_frequencies PROTO,
pString:PTR BYTE,              ; points to string
pTable:PTR DWORD    ; points to frequencey table

.data
freqTable DWORD 256 DUP(0)
; Optional string that can be used when testing the program:
;aString BYTE "THE QUICK BROWN FOX JUMPED OVER THE LAZY DOGS BACK",0
aString BYTE 80 DUP(0),0
str1 BYTE   "*** Constructing a Frequency Table *** (DEMO)", 0dh,0ah,0dh,oah
        "Enter between 1 and 80 characters: ",0
```

Use the main code given and the complete the procedure to run the program.

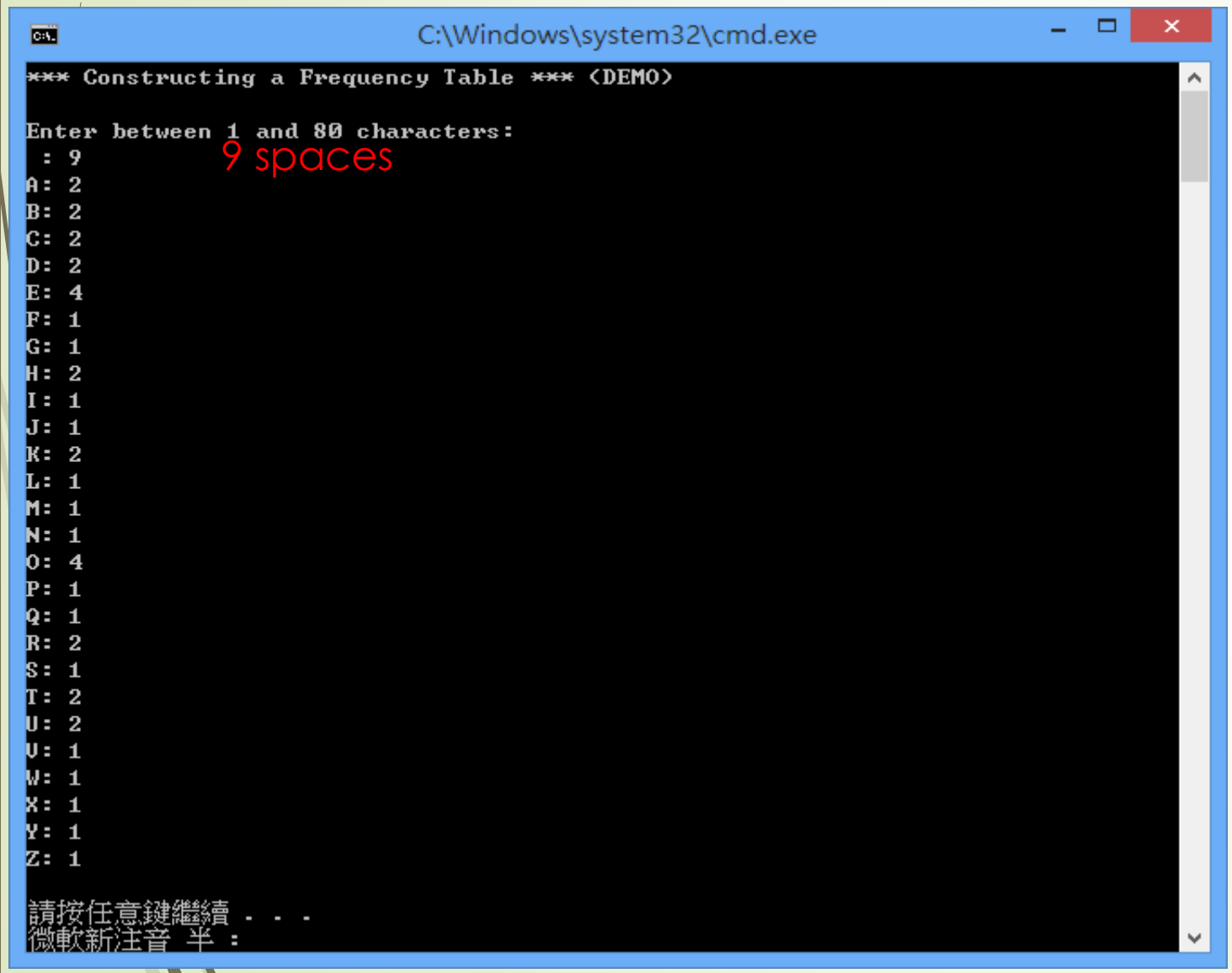
```
.code
main proc
call    Clrscr
mov  edx,OFFSET str1
call WriteString
mov  ecx,SIZEOF aString - 1
mov  edx,OFFSET aString
call ReadString
INVOKE Get_frequencies, ADDR aString, ADDR freqTable
call DisplayTable
exit
main endp
Get_frequencies PROC,    ; Write this procedure and complete the code here.

Get_frequencies ENDP
DisplayTable PROC  ; Write the display Table procedure to display the table.
DisplayTable ENDP
end main
```

One Sample output of the given string in the program is as follows:

THE QUICK BROWN FOX JUMPED OVER THE LAZY DOGS BACK

小考 **ODD Program**



Frequency of letters As follows:

A:2
B:2
C:2
D:2
.
.
.
and so on.

Another example: **"User entered"** string and the output of the string is as follows

This is my Final Exam Program Assignment.

小考 **ODD Program**

C:\Windows\system32\cmd.exe

```
*** Constructing a Frequency Table *** (DEMO)

Enter between 1 and 80 characters: This is my Final exam Program Assignment.

 : 6
.: 1
A: 1
F: 1
P: 1
T: 1
a: 3
e: 2
g: 2
h: 1
i: 4
l: 1
m: 4
n: 3
o: 1
r: 2
s: 4
t: 1
x: 1
y: 1

請按任意鍵繼續 . . .

微軟新注音 半 :
```

6 spaces

```
.code
main proc
call    Clrscr

    mov  edx,OFFSET str1
    call WriteString

    mov  ecx,SIZEOF aString - 1
    mov  edx,OFFSET aString
    call ReadString

    INVOKE Get_frequencies, ADDR aString, ADDR freqTable
    call DisplayTable

exit
main endp
```

```
;-----------------------------------------------------------
Get_frequencies PROC,
        pString:PTR BYTE,               ; points to string
        pTable:PTR DWORD                ; points to frequencey table
; Constructs a character frequency table. Each array position
; is indexed by its corresponding ASCII code.
; Returns: Each entry in the table contains a count of how
; many times that character occurred in the string.
;-----------------------------------------------------------
        mov     esi,  pString
        mov     edi,  pTable
        cld                             ; clear Direction flag (forward)

L1:     mov     eax,0                   ; clear upper bits of EAX
        lodsb                           ; AL = [ESI], inc ESI
        cmp     al,0                    ; end of string?
        je      Exit_proc               ; yes: exit
        shl     eax,2                   ; multiply by 4
        inc     DWORD PTR[edi+eax]      ; add to table entry
        jmp L1                          ; repeat loop
Exit_proc:
        ret
Get_frequencies END
```

```
;-----------------------------------------------------
DisplayTable PROC
;
; Display the non-empty entries of the frequency table.
; This procedure was not required, but it makes it easier
; to demonstrate that Get_frequencies works.
;-----------------------------------------------------
.data
colonStr BYTE ": ",0
.code
    call Crlf
    mov   ecx,LENGTHOF freqTable ; entries to show
    mov   esi,OFFSET freqTable
    mov   ebx,0                  ; index counter
```

```
L1:  mov     eax,[esi]                    ; get frequency count
     cmp     eax,0                    ; count = 0?
     jna  L2                      ; if so, skip to next entry

     mov     eax,ebx                      ; display the index
     call WriteChar
     mov     edx,OFFSET colonStr          ; display ": "
     call WriteString
     mov     eax,[esi]                    ; show frequency count
     call WriteDec
     call Crlf

L2:  addesi,TYPE freqTable            ; point to next table entry
     inc  ebx                     ; increment index
     loop L1

     call Crlf
     ret
DisplayTable ENDP
 end main
```

Let us write the EVEN Program of the Quiz2 program.

小考2 EVEN Program

Given the following main procedure, write the needed two procedures to complete the program to run and show the output to the Instructor or the Assistants. Get a signature. Write the code needed for the two procedures before you submit your exam paper.

; 2017 Replace the delimiter character with the space Even.asm

Comment !

Description: Write a procedure called **Str_replace** that scans a string for all the occurrence of a certain delimiter character and replaces the delimiter with the space. There are three input parameters: a pointer to the string, parameter: a pointer to the clean string and the delimiter character. After the call, if the delimiter was found, it replaces the delimiter with the space. Writes the clean string in another string.

Displayresult procedure, displays the clean string and the number of delimiter characters found.

!

Results of running the string "ABC\DE\FGHIJK\LM"

小考 EVEN Program

C:\Windows\system32\cmd.exe

```
The Clean string after removing the delimiter is:

ABC DE FGHIJK LM
Number of Delimiter(s) found in the string are: 3
請按任意鍵繼續 . . .
```

微軟注音 半 :

User Entered string is as follows:

I\will\do\well\in\my\final\exam.\Because\I\have\studied\well.

小考 **EVEN Program**

C:\Windows\system32\cmd.exe

```
I\will\do\well\in\my\final\exam.\Because\I\have\studied\well.

The Clean string after removing the delimiter is:

I will do well in my final exam. Because I have studied well.
Number of Delimiter(s) found in the string are: 12
請按任意鍵繼續 . . .
```

微軟注音 半 :

```
INCLUDE Irvine32.inc

Str_replace PROTO,
    pString:PTR BYTE,        ; pointer to string
    cString:PTR BYTE,    ; clean string
    delimiter:BYTE       ; delimiter to find

.data
msg0 BYTE "The Clean string after removing the delimiter is: ",0dh,0a
msg1 BYTE "Number of Delimiter(s) found in the string are: ",0
testStr BYTE "ABC\DE\FGHIJK\LM",0
aStr    BYTE 80 DUP(0),0
cleanstr BYTE 80 DUP(0),0
```

```
.code
main PROC
    call Clrscr
    mov  ecx,SIZEOF aStr - 1
    mov  edx,OFFSET aStr
    ;call ReadString

   INVOKE Str_replace, ADDR testStr, ADDR cleanstr, "\"        ; look for delimiter
   call Displayresult
Exit_prog:
    exit
main ENDP
```

```
;This is the diplayresult procedure to display the number of delimiter used.
Displayresult PROC uses eax
    call crlf
    mov edx, OFFSET msg0
     call Writestring
     call crlf
     mov  edx,OFFSET cleanstr          ; display starting string
     call WriteString
     call Crlf
     mov edx, OFFSET msg1
     call Writestring
     call writedec
     call Crlf
     ret
Displayresult ENDP
```

```
;-----------------------------------------------------------
Str_replace PROC,
    pString:PTR BYTE,        ; pointer to string
    cString: PTR BYTE,
    delimiter:BYTE           ; delimiter to find
    LOCAL first:DWORD
; Scans a string for the first occurrence of a certain delimiter
; character and replaces the delimiter with a null byte.
;-----------------------------------------------------------

    push  esi
    push  edi
    mov first, 0
    mov   al,delimiter
    mov  edi, cString
    mov   esi,pString
    cld                      ; clear Direction flag (forward)
```

```
L1:
    mov al, [esi]
      inc esi
      cmp al, 0
      je L4
      cmp al, delimiter
      je L3
L2: mov [edi],al
   inc edi
   jmp L1

L3:
    mov al, ' '
    mov BYTE PTR [esi-1],al
      mov [edi],al
      inc edi
      inc first
      jmp L1
L4:
  or    al,1                    ; clear Zero flag
```

```
Exit_proc:
    pop edi
     pop    esi
     mov eax, first
     ret
Str_replace ENDP

END main
```

Prepare for the Final exam.