COAL ASSIGNMENT 3

23k-0575 **BCS-3D**

Questions:
Question 1.

Write a recursive procedure in x86 assembly language that divides a number by another number and stops when dividend is less than or equal to 5h. Consider dividend = D4A4h and divisor = Ah. The Intel IA 32version of this program is required.

Questions 2

Write a recursive procedure to find a value in a large integer array. Ask the user to enter an integer value in the main program. You should pass user supplied value as parameter to the recursive function using the INVOKE directive. Also, draw labeled diagrams to show stack values at each iteration of this recursive

function.

Question 3

Write an assembly language program to copy the characters of a string to a target string. The characters are stored in such a way that only a single instance of any character in the string is stored. Initialize a source string to: "This is the source string".

Question 4

Write an assembly language program to read a string of characters from the user and prints/store the vowel count. For each vowel, the count includes both uppercase and lowercase letters.

Question 5

Write a procedure named DifferentInputs that returns EAX = 1 if the values of its three input parameters are all different; otherwise, return with EAX = 0. Use the PROC directive with a parameter list when declaring the procedure. Create a PROTO declaration for your procedure, and call it five times from a test program that passes different inputs.

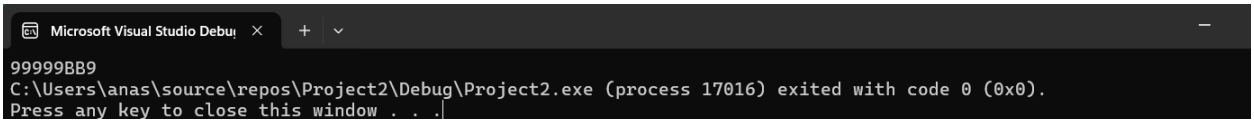
Question 6

Create a variant of the Str_trim procedure that lets the caller remove all instances of a leading character from a string. For example, if you were to call it with a pointer to the string “###ABC” and pass it the # character, the resulting string would be “ABC”.

Q1

Code:

```
INCLUDE Irvine32.inc

.data

.code
main PROC

mov eax , 0D4A4h
mov edx , 0
call Divide

call Writehex

exit

Divide PROC
cmp eax , 5h
jle done
mov ecx , 0Ah
div ecx
call Divide

done:
ret
Divide ENDP

main ENDP
END main
```



Q2

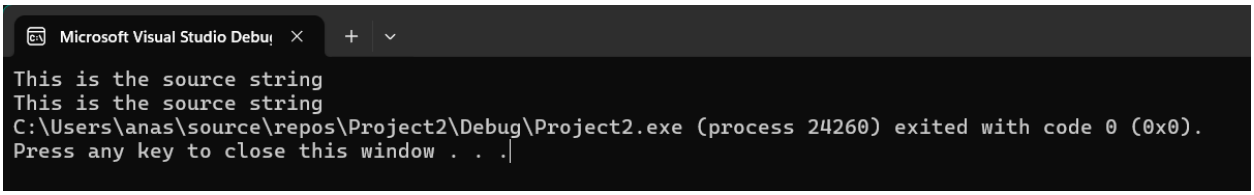
```
INCLUDE Irvine32.inc

.data
str1 BYTE "Enter the integer value: " , 0
arr DWORD 12 , 13 , 14 , 20 , 21 , 22, 25 , 26 ; arr to be searched for
value DWORD ?
size1 DWORD ?
str2 BYTE "Value found" , 0
str3 BYTE "Value not found" , 0

.code

findvalue PROTO Searchvalue:DWORD , pArray: PTR DWORD , size1:DWORD

main PROC
mov edx , offset str1
call Writestring
call Readint
```

```
mov value , eax

mov size1 , LENGTHOF arr

INVOKE findvalue , value , ADDR arr , size1

exit
main ENDP


findvalue PROC,
Searchvalue:DWORD , pArray:PTR DWORD , size2:DWORD ; func recieves the value and
the address of the array

push ebp
mov ebp, esp

cmp size2 , 0
je not_found


mov eax , [pArray]
cmp Searchvalue  , eax
je found

add pArray , 4
sub size2 , 1

INVOKE findvalue , Searchvalue , pArray , size2
jmp end_proc

not_found:
mov edx , offset str3
call Writestring
;ret
jmp end_proc

found:
mov edx , offset str2
call Writestring
;ret

end_proc:
pop ebp
ret
findvalue ENDP

END main
```

Q3

```
INCLUDE Irvine32.inc

.data
str1 BYTE "This is the source string" , 0
target BYTE 40 DUP(?)

.code

Str length PROTO ,
pString:PTR BYTE

Str_copy PROTO ,
source : PTR BYTE ,
target : PTR BYTE

main PROC

INVOKE Str_copy , ADDR str1 , ADDR target

;print the original string
mov edx , offset str1
call Writestring
call crlf

;print the copied string
mov edx , offset target
call Writestring
call crlf

exit
main ENDP

Str_length PROC USES edi,
pString:PTR BYTE ; pointer to string

mov edi,pString
mov eax,0 ; character count

L1:

cmp byte ptr [edi],0 ; end of string?
je L2 ; yes: quit
inc edi ; no: point to next
inc eax ; add 1 to count
jmp L1

L2:
ret
Str_length ENDP

Str_copy PROC  ,
source:PTR BYTE,
target1:PTR BYTE

INVOKE Str_length , source
mov ecx , eax ; move the length into ecx
inc ecx  ; add 1 for the null byte
mov esi , source
mov edi , target1
```

```
cld
rep movsb ; copy the string
ret

Str_copy ENDP

END main
```



Q4

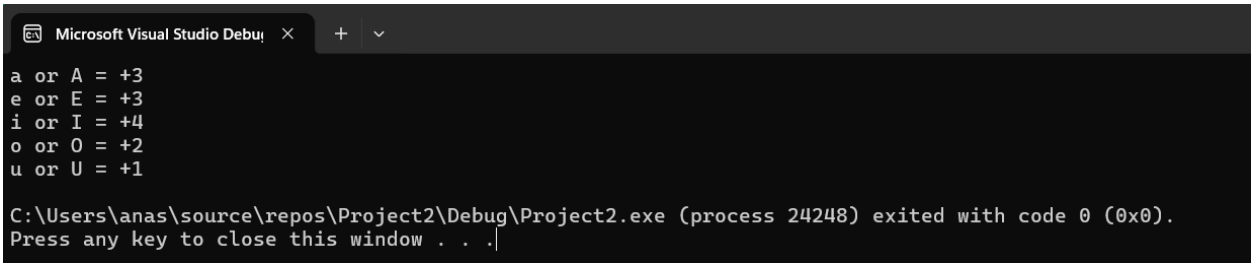
```
INCLUDE Irvine32.inc

.data
Sentence BYTE "Advanced Programming in UNIX Environment" , 0
countA DWORD ?
countE DWORD ?
countI DWORD ?
countO DWORD ?
countU DWORD ?

str1 BYTE "a or A = " , 0
str2 BYTE "e or E = " , 0
str3 BYTE "i or I = " , 0
str4 BYTE "o or O = " , 0
str5 BYTE "u or U = " , 0

.code
main PROC

mov eax , 0
mov countA , 0
mov countE , 0
mov countI , 0
mov countO , 0
mov countU , 0
mov edi , offset Sentence
mov al , 'A'
mov ecx , LENGTHOF Sentence
cld ; direction = forward

L1:
SCASB
jz foundA
loop L1
jmp for_a

foundA:
add countA , 1 ;
loop L1

;for 'a'
```

```
for_a:
mov edi , offset Sentence
mov al , 'a'
mov ecx , LENGTHOF Sentence
cld ; direction = forward

L2:
SCASB
jz found_a
loop L2
jmp forE

found_a:
add countA , 1 ;
loop L2


;Now for E or e
forE:
mov edi , offset Sentence
mov al , 'E'
mov ecx , LENGTHOF Sentence
cld ; direction = forward

L3:
SCASB
jz found_E
loop L3
jmp for_e

found_E:
add countE , 1 ;
loop L3

for_e:
mov edi , offset Sentence
mov al , 'e'
mov ecx , LENGTHOF Sentence
cld ; direction = forward

L4:
SCASB
jz founde
loop L4
jmp forI ;

founde:
add countE , 1 ;
loop L4

;NOW FOR I or i
forI:
mov edi , offset Sentence
mov al , 'I'
mov ecx , LENGTHOF Sentence
cld ; direction = forward

L5:
SCASB
jz foundI
```

```
loop L5
jmp for_i ;

foundI:
add countI , 1 ;
loop L5

for_i:
mov edi , offset Sentence
mov al , 'i'
mov ecx , LENGTHOF Sentence
cld ; direction = forward

L6:
SCASB
jz found_i
loop L6
jmp forO
found_i:
add countI , 1 ;
loop L6


;NOW FOR O or o
forO:
mov edi , offset Sentence
mov al , 'O'
mov ecx , LENGTHOF Sentence
cld ; direction = forward

L7:
SCASB
jz foundO
loop L7
jmp for_o ;

foundO:
add countO , 1 ;
loop L7

for_o:
mov edi , offset Sentence
mov al , 'o'
mov ecx , LENGTHOF Sentence
cld ; direction = forward

L8:
SCASB
jz found_o
loop L8
jmp forU ;

found o:
add countO , 1 ;
loop L8


;NOW FOR U or u
forU:
mov edi , offset Sentence
```

```
mov al , 'U'
mov ecx , LENGTHOF Sentence
cld ; direction = forward

L9:
SCASB
jz foundU
loop L9
jmp for_u ;

foundU:
add countU , 1 ;
loop L9

for u:
mov edi , offset Sentence
mov al , 'u'
mov ecx , LENGTHOF Sentence
cld ; direction = forward

L10:
SCASB
jz found_u
loop L10
jmp done ; as the sentence ends , jump to done to print results

found_u:
add countU , 1 ;
loop L10

;printing all counts
done:
mov edx , offset str1
call Writestring
mov eax , countA
call Writeint
call crlf

mov edx , offset str2
call Writestring
mov eax , countE
call Writeint
call crlf

mov edx , offset str3
call Writestring
mov eax , countI
call Writeint
call crlf

mov edx , offset str4
call Writestring
mov eax , countO
call Writeint
call crlf

mov edx , offset str5
call Writestring
mov eax , countU
call Writeint
```

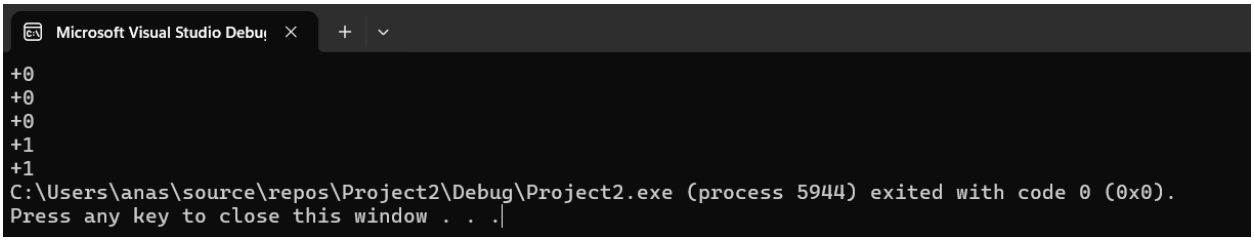
```
call crlf

exit
main ENDP
END main
```



Q5

```
INCLUDE Irvine32.inc

.data


.code

DifferentInputs PROTO value1:DWORD , value2:DWORD , value3:DWORD

main PROC
mov eax , 0
mov ebx , 0
mov ecx , 0

INVOKE DifferentInputs , 3 , 3 , 3
call Writeint
call crlf

INVOKE DifferentInputs , 3 , 2 , 3
call Writeint
call crlf

INVOKE DifferentInputs , 1 , 1 , 1
call Writeint
call crlf

INVOKE DifferentInputs , 12 , 0 , 2
call Writeint
call crlf

INVOKE DifferentInputs , 4 , 3 , 5
call Writeint


exit
main ENDP

DifferentInputs PROC ,
value1:DWORD , value2:DWORD , value3:DWORD

mov eax , value1
mov ebx , value2
```

```
mov ecx , value3

cmp eax , ebx
jne check2
mov eax , 0
jmp quit

check2:
cmp eax , ecx
jne notequal
mov eax , 0
jmp quit

notequal:
mov eax , 1

quit:
ret
DifferentInputs ENDP

END main
```



Q6

```
INCLUDE Irvine32.inc

.data
myString BYTE "###ABC",0

.code
main PROC

;original string
mov edx , offset myString
call Writestring

INVOKE Str_trim, ADDR myString, '#'
;after trimming
Mov edx , offset myString
Call Writestring

exit
main ENDP

Str_length PROC USES edi,
pString:PTR BYTE ; pointer to string
```

```
mov edi,pString
mov eax,0 ; character count

L1:

cmp byte ptr [edi],0 ; end of string?
je L2 ; yes: quit
inc edi ; no: point to next
inc eax ; add 1 to count
jmp L1

L2:
ret
Str_length ENDP

Str_trim PROC ,
pString:PTR BYTE, char1:BYTE

mov edi,pString

INVOKE Str_length,edi

;mov eax , value1 ; returns length in EAX

cmp eax,0 ; zero-length string?
je L2 ; yes: exit

mov ecx,eax ; no: counter = string length
dec eax

add edi,eax ; EDI points to last char
mov al,char1 ; char to trim

std ; direction = reverse

repe scasb ; skip past trim character

jne L1 ; removed first character?
dec edi ; adjust EDI: ZF=1 && ECX=0

L1:
mov BYTE PTR [edi+2],0 ; insert null byte

L2:
ret
Str_trim ENDP

END main
```