SEMESTER II 2020/2021

LAB 4

SUBJECT:

SECR2033-02 ORGANISASI & SENIBINA KOMPUTER (COMPUTER ORGANIZATION AND ARCHITECTURE)

LECTURER:

DR. AIDA BINTI ALI

GROUP NAME:

DREAMERS

NAME	MATRIC NUM
AHMAD AIMAN HAFIZI BIN MUHAMMAD	A20EC0177
MADIHAH BINTI CHE ZABRI	A20EC0074
MADINA SURAYA BINTI ZHARIN	A20EC0203
MAIZATUL AFRINA SAFIAH BINTI SAIFUL AZWAN	A20EC0204
NAYLI NABIHAH BINTI JASNI	A20EC0105
ONG HAN WAH	A20EC0129

<u>Program 1</u>

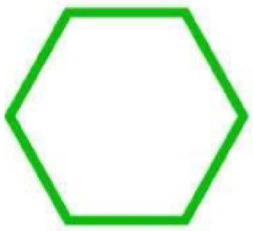


Figure 1: A hexagon

Figure 1 illustrates a hexagon figure with the same length of side. To calculate the perimeter of the hexagon, the following formula is given.

Perimeter_hexagon1 = side1 + side2 + side3 + side4 + side5 + side6
Perimeter_hexagon2 = side1 + side2 + side3 + side4 + side5 + side6
TotalPerimeter = Perimeter_hexagon1 + Perimeter_hexagon2

Write a complete program using assembly language to calculate the perimeter of TWO different hexagons with different sizes.

In the program, you should do these steps:

I. Get two values from the keyboard (32-bit unsigned integer) and save into the variable name **sideHex1** for the first hexagon and **sideHex2** for the second hexagon.
II. Calculate both of the perimeters (Example: Perimeter_hexagon1=18 3+3+3+3+3+3) by using LOOP instruction. Save the first result in **Perimeter_hexagon1** and the second result in **Perimeter_hexagon2** (as 32-bit unsigned integer).
III. Then, add the two perimeters and save in the **TotalPerimeter** variable.
IV. Display the output as shown in Figure 2.

```
TITLE Lab 4 Program 1        (main.asm)
; This program calculate perimeter of 2 hexagon

INCLUDE Irvine32.inc

.data
sideHex1 DWORD ?
sideHex2 DWORD ?
Perimeter_hexagon1          DWORD ?
Perimeter_hexagon2 DWORD ?
TotalPerimeter DWORD ?

str1 BYTE "enter a side value hexagon1:", 0
```

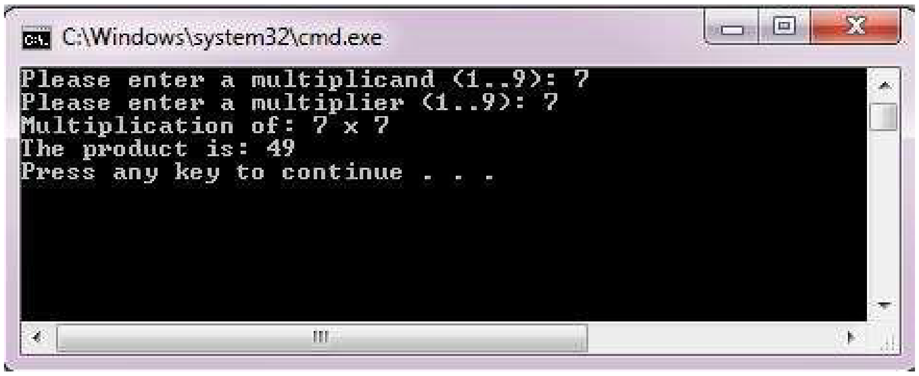
```
str2 BYTE "enter a side value for hexagon2:", 0
str3 BYTE "Perimeter for hexagon1 with side = +", 0
str4 BYTE "Perimeter for hexagon2 with side = +", 0
str5 BYTE " is : +", 0
str6 BYTE "The total perimeter = +", 0

.code
main PROC

call Clrscr

mov edx, OFFSET str1
call WriteString
call ReadDec
mov sideHex1, eax

mov edx, OFFSET str2
call WriteString
call ReadDec
mov sideHex2, eax

mov eax, Perimeter_hexagon1
mov ecx, 6

L1:
add eax, sideHex1
loop L1

mov Perimeter_hexagon1, eax

mov edx, OFFSET str3
call WriteString
mov eax, sideHex1
call WriteDec

mov edx, OFFSET str5
call WriteString
mov eax, Perimeter_hexagon1
call WriteDec

call Crlf

mov eax, Perimeter_hexagon2
mov ecx, 6
```

```
L2:
add eax, sideHex2
loop L2

mov Perimeter_hexagon2, eax

mov edx, OFFSET str4
call WriteString
mov eax, sideHex2
call WriteDec

mov edx, OFFSET str5
call WriteString
mov eax, Perimeter_hexagon2
call WriteDec

call Crlf

mov eax, Perimeter_hexagon1
add eax, Perimeter_hexagon2
mov TotalPerimeter, eax

mov edx, OFFSET str6
call WriteString
mov eax, TotalPerimeter
call WriteDec

call Crlf

exit
main ENDP

END main
```

Program 2

• Write a program in assembly language to multiple two unsigned numbers.
• Your program should ask the user to input the multiplicand (n) and the multiplier (m).
• The program will do multiplication of (n x m) using MUL.
• Your program should store the multiplicand, multiplier and the result in these variables multiplicand, multiplier and product respectively

Sample Output



```
Please enter a multiplicand (1..9): 7
Please enter a multiplier (1..9): 7
Multiplication of: 7 x 7
The product is: 49
Press any key to continue . . .
```

```
TITLE Lab 4 Progarm 2        (main.asm)
;Multiplication program

INCLUDE Irvine32.inc

.data
str1 BYTE "Please enter a multiplicand <1...9>: ", 0
str2 BYTE "Please enter a multiplier <1...9>: ", 0
str3 BYTE "Multiplication of: ", 0
str4 BYTE "The product is: ", 0
str5 BYTE " x ", 0
multiplicand dword ?
multiplier dword ?
product dword 0

.code
main PROC
call clrscr
mov edx, offset str1; move the address of str1 to edx
call WriteString; display the string "str1"
call ReadDec; read a 32bit unsigned number input from user
mov multiplicand, eax; move the input from user into "multiplicand"

mov edx, offset str2; move the address of str2 to edx
```

```
call WriteString; display the string "str1"
call ReadDec; read a 32bit unsigned number input from user
mov multiplier, eax; move the input from user into "multiplier"

mov edx, offset str3; move the address of str3 to edx
call WriteString; display the string "str3"
mov eax, multiplicand
call WriteDec; display the value of eax from multiplicand
mov edx, offset str5; move the address of str5 to edx
call WriteString; display the string "str5"
mov eax, multiplier
call WriteDec

mov ecx, multiplier
mov eax, product
L1:
   add eax, multiplicand; add the value of multiplicand to eax
   loop L1; Looping until the value of ecx deducted to "0"

mov product, eax
call crlf
mov edx, offset str4; move the address of str4 to edx
call WriteString; display the string "str4"
call WriteDec; display the value of eax, the product of addition from looping
call crlf; add newline

        exit
main ENDP

END main
```

<u>Program 2 (Extra Challenge)</u>

Rewrite your program and ask either user to continue the calculation (Yes/No). If Yes, users can have a selection to either perform MUL or DIV. If No, print “Thank you” and exit the program.

```
TITLE Lab 4 Program 2 (EXTRA)    (main.asm)
;This program adds and subtracts 32 bit integers

INCLUDE Irvine32.inc

.data
str1 BYTE "Please enter a multiplicand <1...9>: ", 0
str2 BYTE "Please enter a multiplier <1...9>: ", 0
str3 BYTE "Multiplication of: ", 0
str4 BYTE "The product is: ", 0
str5 BYTE " x ", 0
str6 BYTE "Do you want to continue ? [1 => YES   2 => NO] : ", 0
str7 BYTE " Choose Program --- 1 => Multiplication      2 => Division : ", 0
str8 BYTE "Thank You", 0
str9 BYTE "Please enter a dividend <1...9>: ",0
str10 BYTE "Please enter a divisor <1...9>: ", 0
str11 BYTE "Division of: ",0
str12 BYTE "The quotient is: ", 0
str13 BYTE " / ", 0
multiplicand dword ?
multiplier dword ?
product dword 0
dividend dword ?
divisor dword ?
quotient dword ?

.code
main PROC
call clrscr

multiplication:
mov edx, offset str1; move the address of str1 to edx
call WriteString; display the string "str1"
call ReadDec; read a 32bit unsigned number input from user
mov multiplicand, eax; move the input from user into "multiplicand"

mov edx, offset str2; move the address of str2 to edx
call WriteString; display the string "str1"
call ReadDec; read a 32bit unsigned number input from user
```

```
mov multiplier, eax; move the input from user into "multiplier"

mov edx, offset str3; move the address of str3 to edx
call WriteString; display the string "str3"
mov eax, multiplicand
call WriteDec; display the value of eax from multiplicand
mov edx, offset str5; move the address of str5 to edx
call WriteString; display the string "str5"
mov eax, multiplier
call WriteDec

mov ecx, multiplier
mov eax, product
L1 :
add eax, multiplicand; add the value of multiplicand to eax
loop L1; Looping until the value of ecx deducted to "0"

mov product, eax
call crlf
mov edx, offset str4; move the address of str4 to edx
call WriteString; display the string "str4"
call WriteDec; display the value of eax, the product of addition from looping
call crlf; add newline

question:
mov edx, offset str6
call crlf
call WriteString
call ReadDec
call clrscr

cmp eax, 1
je  choose
mov edx, offset str8
call WriteString
call crlf
jmp none
choose:
  mov edx, offset str7
  call WriteString
  call ReadDec
  call crlf
```

```
   cmp eax, 1
   je mult
   jmp division

mult:
   mov product,0
   jmp multiplication

division:
   mov edx, offset str9
   call WriteString
   call ReadDec
   mov dividend, eax

   mov edx, offset str10
   call WriteString
   call ReadDec
   mov divisor, eax

   mov edx, offset str11
   call WriteString
   mov eax, dividend
   call WriteDec
   mov edx, offset str13
   call WriteString
   mov eax, divisor
   call WriteDec

   mov edx, 0
   mov eax, dividend
   mov ebx, divisor
   div ebx
   call crlf
   mov edx, offset str12
   call WriteString
   call WriteDec
   mov quotient, eax
   call crlf; add newline
   jmp question

 none:
            exit
main ENDP
END main
```

<u>Program 3</u>

Write a program that will interactively ask the user to input the values of 6 integers in DWORD and you have to put the values into an array name HELLO.

- Example of HELLO array after the user input the values:

1st Value	2nd Value	3rd Value	4th Value	5th Value	6th Value
HELLO[0]	HELLO[4]	HELLO[8]	HELLO[12]	HELLO[16]	HELLO[20]
32	*65*	*77*	*89*	*14*	*54*

- Your CountEVEN will count the value of HELLO[0], HELLO[8] and HELLO[16] and store it in variable name TotalEVEN

- Your CountODD will count the value of HELLO[4], HELLO[12] and HELLO[20] store it in variable name TotalODD

- Lastly, display the value of TotalEVEN and TotalODD

- You must use LOOP instruction to do the addition process.

<u>Code</u>

```
TITLE Lab 4 - Program 3(main.asm)

INCLUDE Irvine32.inc

.data
HELLO         DWORD 6 DUP(0)
TotalEVEN     DWORD 0
TotalODD      DWORD 0
TotalALL      DWORD 0
enterINT      BYTE "Enter Integer : ", 0
resultODD     BYTE "TotalODD is  : ", 0
resultEVEN    BYTE "TotalEVEN is : ", 0
resultALL     BYTE "TotalALL is : ", 0

.code
main PROC
```

```
mov ecx, 6
mov ebx, OFFSET HELLO
mov edx, OFFSET enterINT

;---------------------------------(Input Integers)----------------------------
L1 :
call WriteString                        ; Start with message enterINT
call ReadInt                            ; Read any integer input
mov [ebx], eax                          ; Input all integers of eax into array ebx
add ebx, 4                              ; Initially ebx is HELLO[0], then plus 4
call crlf                               ; Next line
loop L1

;---------------------------------(Adding Odd Integer)----------------------
mov ecx, 3                              ; Loop 3 times
mov ebx, OFFSET HELLO                   ; To add HELLO[0], HELLO[8], HELLO[16]
CountODD :
mov eax, [ebx]                          ; Move the current array value of ebx into eax
add TotalODD, eax                       ; Add previous integer with new integer
add ebx, 8
loop CountODD

;---------------------------------(Adding Even Integer)---------------------
mov ecx, 3                              ; Loop 3 times
mov ebx, OFFSET HELLO + 4               ; To add HELLO[4], HELLO[12], HELLO[20]
CountEVEN:
mov eax, [ebx]                          ; Move the current array value of ebx into eax
add TotalEVEN, eax                      ; Add previous integer with new integer
add ebx, 8
loop CountEVEN

;---------------------------------(Result for Odd)----------------------------
mov edx, OFFSET resultODD               ; Start with message resultODD
call WriteString
mov eax, TotalODD                       ; Display value of TotalODD
add TotalALL, eax                       ; Add value TotalODD to TotalALL (challenge)
call WriteDec
call crlf

;---------------------------------(Result for Even)-------------------------
mov edx, OFFSET resultEVEN              ; Start with message resultEVEN
call WriteString
mov eax, TotalEVEN                      ; Display value of TotalEVEN
```

```
add TotalALL, eax                        ; Add value TotalEVEN to TotalALL (challenge)
call WriteDec
call crlf

;-----------------------------------(Result for All)-------------------------------
mov edx, OFFSET resultALL                ; Start with message resultALL
call WriteString
mov eax, TotalALL                        ; Display value of resultALL
call WriteDec

exit
main ENDP

END main
```

- **Link for video demonstration for Lab 4**

  https://youtu.be/mS4qUxGnRxI