UNIVERSITI TEKNOLOGI MALAYSIA

SECR2033 COMPUTER ORGANIZATION AND ARCHITECTURE
SECTION-02

20202021/2

LAB 4

LECTURER'S NAME : DR AIDA ALI

VIDEO LINK:

https://youtu.be/QOpUKkEzgUU

GROUP 4.0 MEMBERS :

NAME	MATIRC NO.
AFIF HAZMIE ARSYAD BIN AGUS	A20EC0176
HONG PEI GEOK	A20EC0044
LOW JUNYI	A20EC0071
LEE JIA XIAN	A20EC0200
MUHAMMAD IMRAN HAKIMI BIN MOHD SHUKRI	A20EC0213
YONG ZHI YAN	A20EC0172

Table of Contents

Program 1

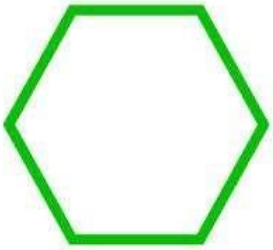


Figure 1: A hexagon

Figure 1 is illustrates a hexagon figure with 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 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 ***TotalPerimeter*** variable.

iv. Display the output as shown in Figure 2.

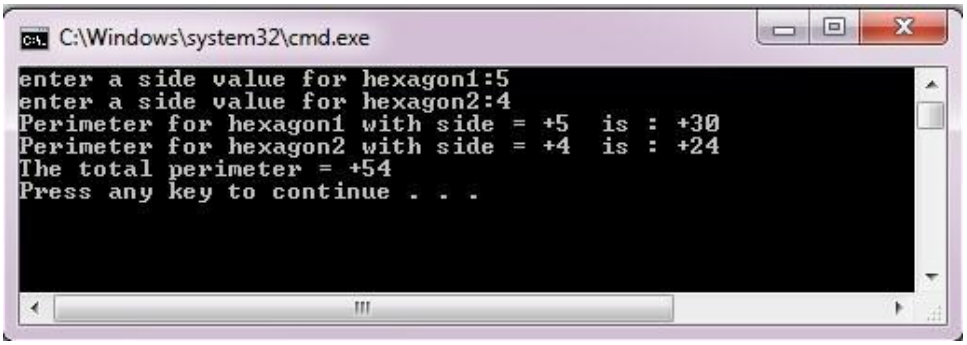


Figure 2: The Output

Extra Challenge: Rewrite your program and add 3 more library procedures based on your creativity.

```
TITLE Lab 4 Program 1          (main.asm)

INCLUDE Irvine32.inc

.data
question1 BYTE "enter a side value for hexagon1:",0
question2 BYTE "enter a side value for hexagon2:",0
answerq1 BYTE "Perimeter for hexagon1 with side = ",0
answerq2 BYTE "Perimeter for hexagon1 with side = ",0
reply BYTE " is :",0
Total BYTE "The total perimeter = ",0



sideHex1 DWORD ?
sideHex2 DWORD ?
Perimeter_hexagon1 DWORD ?
Perimeter_hexagon2 DWORD ?
TotalPerimeter DWORD ?
side DWORD 6

.code
main PROC

; call user to input side for hexagon 1
mov edx, OFFSET question1
call WriteString
call ReadDec
mov sideHex1, eax


; call user to input side for hexagon 2
mov edx, OFFSET question2
call WriteString
call ReadDec
mov sideHex2, eax

; calculate the perimeter of hexagon 1 by using loop
mov ecx, side
```

```
mov eax, 0
L1:
add eax, sideHex1
loop L1

mov Perimeter_hexagon1, eax  ;store the answer of into Perimeter_hexagon2


; calculate the perimeter of hexagon 2 by using loop
mov ecx, side
mov eax, 0
L2:
add eax, sideHex2
loop L2

mov Perimeter_hexagon2, eax ;store the answer of into Perimeter_hexagon2


; calculate the total perimeter of both hexagon
mov eax, 0
add eax, Perimeter_hexagon1
add eax, Perimeter_hexagon2
mov TotalPerimeter, eax

;display perimeter of hexagon1
mov edx, OFFSET answerq1
call WriteString
mov eax, sideHex1
call WriteInt
mov edx, OFFSET reply
call WriteString
mov eax, Perimeter_hexagon1
call WriteInt
call crlf


;display perimeter of hexagon2
mov edx, OFFSET answerq2
call WriteString
mov eax, sideHex2
call WriteInt
mov edx, OFFSET reply
call WriteString
mov eax, Perimeter_hexagon2
```

```
call WriteInt
call crlf

;display total perimeter of both hexagon
mov edx, OFFSET total
call WriteString
mov eax, TotalPerimeter
call WriteInt

        exit
main ENDP

END main
```

Code with extra challenges:

```
TITLE Lab 4 Program 1          (main.asm)

INCLUDE Irvine32.inc

.data
calculate BYTE "Calculating....",0
random BYTE "Do u want to proceed with random values? (Y/N):",0
randomhex1 BYTE "Randomizing side value of hexagon1(1-99)...",0
randomhex2 BYTE "Randomizing side value of hexagon2(1-99)...",0
rsidehex1 BYTE "The side of hexagon1 is: ",0
rsidehex2 BYTE "The side of hexagon2 is: ",0
question1 BYTE "enter a side value for hexagon1:",0
question2 BYTE "enter a side value for hexagon2:",0
answerq1 BYTE "Perimeter for hexagon1 with side = ",0
answerq2 BYTE "Perimeter for hexagon1 with side = ",0
reply BYTE " is :",0
Total BYTE "The total perimeter = ",0
continue BYTE "Do you want to continue with other values? (Y/N): ",0
bye BYTE "Thank You !!",0
process BYTE "Processing in 3 seconds...",0


sideHex1 DWORD ?
sideHex2 DWORD ?
Perimeter_hexagon1 DWORD ?
Perimeter_hexagon2 DWORD ?
TotalPerimeter DWORD ?
side DWORD 6
choice BYTE 4 dup(0)

.code
main PROC


r:
call clrscr
call Randomize

;ask the user whether want to proceed with random values
mov edx, OFFSET random
call WriteString
```

```
mov edx, OFFSET choice
mov ecx, (SIZEOF choice) - 1
call ReadString; read Y/N

mov al, [edx]
cmp al, "N"
je input ; if "N" go to input


;pick random side value for hexagon1
mov edx, OFFSET randomhex1
call WriteString
mov eax, 3000
call delay
mov eax, 100; indicate range of value from 0 to 99
call RandomRange
mov sideHex1, eax
mov edx, OFFSET rsidehex1
call WriteString
mov eax, sideHex1
call WriteInt
call crlf

;pick random side value for hexagon2
mov edx, OFFSET randomhex2
call WriteString
mov eax, 3000
call delay
mov eax, 100; indicate range of value from 0 to 99
call RandomRange
mov sideHex2, eax
mov edx, OFFSET rsidehex2
call WriteString
mov eax, sideHex2
call WriteInt
call crlf

jmp calc ; go to calc


input:
; call user to input side for hexagon 1
mov edx, OFFSET question1
call WriteString
```

```
call ReadDec
mov sideHex1, eax


; call user to input side for hexagon 2
mov edx, OFFSET question2
call WriteString
call ReadDec
mov sideHex2, eax


calc:
; calculate the perimeter of hexagon 1 by using loop
mov edx, OFFSET calculate
call WriteString
call crlf
mov eax, 3000
call delay
mov ecx, side
mov eax, 0
L1:
add eax, sideHex1
loop L1

mov Perimeter_hexagon1, eax  ;store the answer of into Perimeter_hexagon2


; calculate the perimeter of hexagon 2 by using loop
mov ecx, side
mov eax, 0
L2:
add eax, sideHex2
loop L2

mov Perimeter_hexagon2, eax ;store the answer of into Perimeter_hexagon2


; calculate the total perimeter of both hexagon
mov eax, 0
add eax, Perimeter_hexagon1
add eax, Perimeter_hexagon2
mov TotalPerimeter, eax

;display perimeter of hexagon1
```

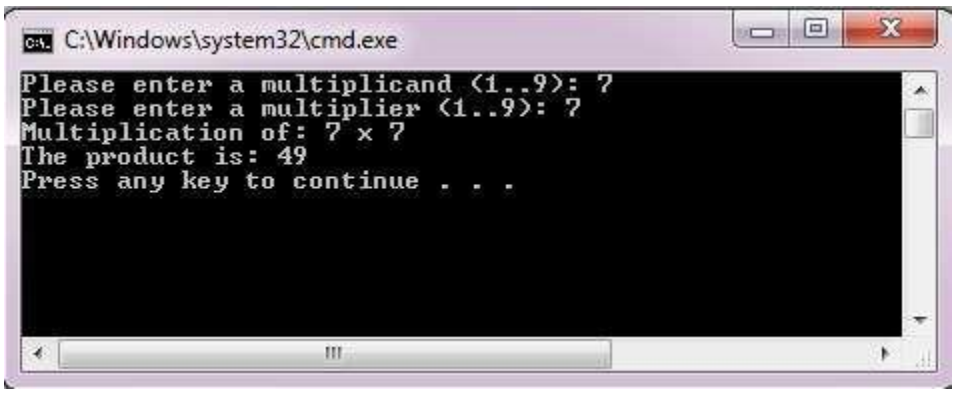
```
mov edx, OFFSET answerq1
call WriteString
mov eax, sideHex1
call WriteInt
mov edx, OFFSET reply
call WriteString
mov eax, Perimeter_hexagon1
call WriteInt
call crlf


;display perimeter of hexagon2
mov edx, OFFSET answerq2
call WriteString
mov eax, sideHex2
call WriteInt
mov edx, OFFSET reply
call WriteString
mov eax, Perimeter_hexagon2
call WriteInt
call crlf

;display total perimeter of both hexagon
mov edx, OFFSET total
call WriteString
mov eax, TotalPerimeter
call WriteInt

;ask the user whether want to repeat the program
call crlf
mov edx, OFFSET continue
call WriteString
mov edx, OFFSET choice
mov ecx, (SIZEOF choice) - 1
call ReadString

mov al, [edx]
cmp al, "Y"
je de ; if "Y" go to de
cmp al, "N"
je ex ; IF "N: go to ex
```

```
de:
;Delay to repeating program
mov edx, OFFSET process
call WriteString
mov eax, 3000
call delay
jmp r


ex:
;print "Thank you" and end the program
mov edx, OFFSET bye
call WriteString


        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



Extra Challenge: Rewrite your program and ask either user want to continue the calculation (Yes/No). If Yes, user can have a selection either perform MUL or DIV. If No, print "Thank you" and exit the program.

Code without extra challenge:

```
TITLE Lab 4 Q2          (main.asm)

INCLUDE Irvine32.inc

.data
QMC byte "Please enter a multiplicand (1...9): ",0
QMP byte "Please enter a multiplier (1...9): ",0
process byte "Multiplication of: ",0
symbol byte " x ",0
QP byte "The product is: ",0

multiplicand dword 0
multiplier dword 0
product dword 0

.code
main PROC
```

```
;ask user enter the value of multiplicand
mov edx, OFFSET QMC
call writestring
call readdec
mov multiplicand, eax

;ask user to enter the value of multiplier
mov edx, OFFSET QMP
call writestring
call readdec
mov multiplier, eax

;show the process
mov edx, OFFSET process
call writestring
mov edx, multiplicand
call writedec
mov edx, OFFSET symbol
call writestring
mov edx, multiplier
call writedec
call crlf

;show the result
mov edx, OFFSET QP
call writestring
mov eax, multiplicand
mov ebx, multiplier
mul ebx
mov product, eax
call writedec

        exit
main ENDP

END main
```

Code with extra challenge:

```
TITLE Lab 4 Q2          (main.asm)

INCLUDE Irvine32.inc

.data
QMC byte "Please enter a multiplicand (1...9): ",0
QMP byte "Please enter a multiplier (1...9): ",0
process byte "Multiplication of: ",0
symbol byte " x ",0
QP byte "The product is: ",0

QDV byte "Please enter a dividend (1...9): ",0
QDS byte "Please enter a divisor (1...9): ",0
process2 byte "Division of: ",0
symbol2 byte " / ",0
QQ byte "The quotient is: ",0

QCT byte "Do you want to continue the calculation [1] Yes  [2] No  : ", 0
QCD byte "Select the operation you want to perform   [1] Multiplication  [2] Division   : ",0
ENDING byte "Thank you",0
continue dword 0
operation dword 0

multiplicand dword 0
multiplier dword 0
product dword 0

dividend dword 0
divisor dword 0
quotient dword 0

.code
main PROC

;ask user enter the value of multiplicand
MULTI:
call clrscr
mov edx, OFFSET QMC
call writestring
call readdec
mov multiplicand, eax

;ask user to enter the value of multiplier
```

```
mov edx, OFFSET QMP
call writestring
call readdec
mov multiplier, eax

;show the multiplication  process
mov edx, OFFSET process
call writestring
mov eax, multiplicand
call writedec
mov edx, OFFSET symbol
call writestring
mov eax, multiplier
call writedec
call crlf

;show the result of product
mov edx, OFFSET QP
call writestring
mov eax, multiplicand
mov ebx, multiplier
mul ebx
mov product, eax
call writedec
call crlf
JMP CONTI

;ask user enter the value of dividend
DIVI:
call clrscr
mov edx, OFFSET QDV
call writestring
call readdec
mov dividend, eax

;ask user enter the value of divisor
mov edx, OFFSET QDS
call writestring
call readdec
mov divisor, eax

;show the division  process
mov edx, OFFSET process2
call writestring
```

```
mov eax, dividend
call writedec
mov edx, OFFSET symbol2
call writestring
mov eax, divisor
call writedec
call crlf

;show the result of quotient
mov edx, OFFSET QQ
call writestring
mov edx, 0
mov eax, dividend
mov ebx, divisor
div ebx
mov quotient, eax
call writedec
call crlf

;ask user want to continue or not
CONTI:
mov edx, OFFSET QCT
call writestring
call readint
mov continue, eax
mov eax, 1
cmp eax, continue
JE NEXT
JMP STOP

;ask user want to perform multiplication or division
NEXT:
call clrscr
mov edx, OFFSET QCD
call writestring
call crlf
call readint
mov operation, eax
mov eax, 1
cmp eax, operation
JE MULTI
JMP DIVI

;the end of the process
```

```
STOP:
call clrscr
mov edx, OFFSET ENDING
call writestring

        exit
main ENDP

END main
```

Program 3

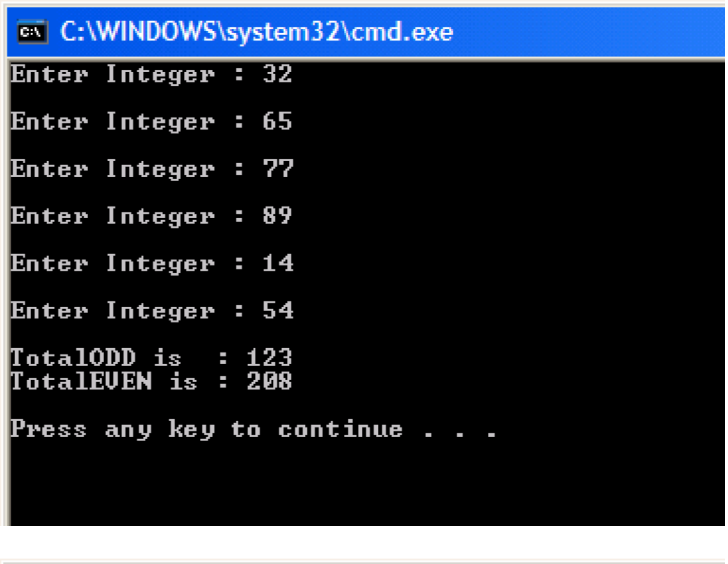
Write a program that will **interactively** ask the **user to input the values of 6 integers** in DWORD and you have to put the values into an array name HELLO.

- 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.**

Sample output



Extra Challenge: Rewrite your program and calculate the TotalALL by adding TotalODD and TotalEVEN. Finally, display the value of TotalALL at the centre of the screen.

```
TITLE Lab4Q3(main.asm)

INCLUDE Irvine32.inc

.data
line1 byte "Enter Integer : ", 0
```

```
line2 byte "TotalODD is : ", 0
line3 byte "TotalEVEN is : ", 0
HELLO dword 6 dup(0)
TotalODD dword ?
TotalEVEN dword ?

;Extra challenge to calculate TotalALL
str4 BYTE "TotalALL is        : ", 0
TotalALL DWORD ?

.code
main PROC
mov edx, offset line1
call WriteString
call ReadInt
mov HELLO + 0, eax

call crlf
mov edx, offset line1
call WriteString
call ReadInt
mov HELLO + 4, eax

call crlf
mov edx, offset line1
call WriteString
call ReadInt
mov HELLO + 8, eax

call crlf
mov edx, offset line1
call WriteString
call ReadInt
mov HELLO + 12, eax

call crlf
mov edx, offset line1
call WriteString
call ReadInt
mov HELLO + 16, eax
```

```
call crlf
mov edx, offset line1
call WriteString
call ReadInt
mov HELLO + 20, eax

call crlf
mov edi, offset HELLO
mov ecx, 3
mov eax, 0

L1 :
add eax, [edi]
add edi, 8
loop L1

mov TotalODD, eax
mov edi, offset HELLO + 4
mov ecx, 3
mov eax, 0

L2 :
add eax, [edi]
add edi, 8
loop L2

mov TotalEVEN, eax

mov edx, offset line2
call WriteString
mov eax, TotalODD
call WriteDec

call crlf
mov edx, offset line3
call WriteString
mov eax, TotalEVEN
call WriteDec
```

```
;Extra challenge to calculate TotalALL
call crlf
mov edx, OFFSET str4
call WriteString
mov eax, TotalODD
add eax, TotalEVEN
mov TotalALL, eax
call WriteDec
call crlf

call crlf
exit
main ENDP
END main
```

Conclusion

In program 1, the perimeter of two different hexagons and the total perimeter are calculated. The perimeter of the hexagon is calculated by using a loop. We had also included three library procedures in the program. Firstly, the instruction 'call Randomize' will generate random numbers in the range we set. Next, the instruction 'call clrscr' (read as call clearscreen) will clear the output screen once called. Thirdly, the instruction 'call delay' will delay the program for 3 seconds when it is called. Other than that, we also make use of the conditional jump instruction ('je') and the compare instruction (cmp) to make the conditional jumps to the process of either input the value of sides manually or randomize the values of sides. Beside that, the conditional jump is also used to determine whether to continue the program or not.

In program 2, two unsigned numbers are multiplied. The multiplicand and multiplier are entered by the user and stored in the register edx and eax. The product is then stored in the register eax and displayed to the output screen. The conditional jump instruction is also used in this program to let the user choose between the function of multiplication and division. As for division, the dividend and divisor are entered by the user and stored in the register edx and eax. The quotient is stored in the register eax and displayed as output.

In program 3, six different values are inputted by the user, the odd sequence of the values (eg.: first value, third value, and fifth value) is added, and vice versa for the even sequence. The values inputted are stored in an array named 'HELLO', which each number occupies a size of 4. The addition process for both odd numbers and even numbers are carried out by using the loop instruction and the results are stored in the variable TotalODD and TotalEVEN respectively. The total for all six values are then added together and stored in the variable TotalALL. In this program, we learn how to output a line of string by using the instruction 'call WriteString'. We also know how to read an integer input by the user by using the instruction 'call ReadInt'. Lastly, we use the instruction 'call WriteDec' to display the output integer.