



**School of Computing**  
**Faculty of Engineering**  
**UNIVERSITI TEKNOLOGI MALAYSIA**

**SUBJECT NAME:** COMPUTER ORGANIZATION AND ARCHITECTURE

**SUBJECT CODE:** SCSR/SECR 2033

**SEMESTER:** 2019/20-2

**LAB TITLE:** Programming 4: Interactive Usage of Link Libraries

**STUDENT INFO :**

| Name          | Matric No. |
|---------------|------------|
| SEE WEN XIANG | A19EC0206  |
| LOO ZHI XUEN  | A19EC0078  |
| KONG HAO YANG | A19EC0065  |

**SECTION:** 07

**SUBMISSION DATE:** 06/06/2020 (Sat)

**COMMENTS:**

## **Part C – Let's do a little programming on your own**

**Ans:**

### **Program 1**

#### **i) Before improvement**

```
INCLUDE Irvine32.inc

.data
sideHex1 dword 0
sideHex2 dword 0
Perimeter_hexagon1 dword 0
Perimeter_hexagon2 dword 0
Totalperimeter dword 0
instruction1 byte "Enter a side value for hexagon 1:",0
instruction2 byte "Enter a side value for hexagon 2:",0
instruction3 byte "Perimeter for hexagon 1 with side = ",0
instruction4 byte "Perimeter for hexagon 2 with side = ",0
instruction5 byte "The total perimeter = ",0
instruction6 byte " is : ",0

.code
main PROC

mov edx, offset instruction1 ; Displaying instruction to ask for side value of hexagon 1 from user
call writestring
call readint ; Read input for side value of hexagon 1
mov sideHex1,eax

mov edx,offset instruction2 ; Displaying instruction to ask for side vlaue of hexagon 2 from user
call writestring
call readint ; Read input for side value of hexagon 2
mov sideHex2,eax

mov edx,offset instruction3 ;displaying perimeter hexagon 1
call writestring
mov eax,sideHex1
call writeint
mov edx,offset instruction6
call writestring

mov eax,0 ;calculate perimeter hexagon 1
mov ecx,6
L1: add eax,sideHex1
Loop L1

mov Perimeter_hexagon1,eax ;displaying perimeter hexagon 1
call writeint
call crlf

mov edx,offset instruction4 ;displaying perimeter hexagon 2
call writestring
mov eax,sideHex2
call writeint
mov edx,offset instruction6
call writestring

mov eax,0 ;display ;calculate perimeter hexagon 2
mov ecx,6
L2: add eax,sideHex2
Loop L2

mov Perimeter_hexagon2,eax ;displaying perimeter hexagon 2
call writeint
call crlf

mov eax,Perimeter_hexagon1 ;calculate and display total
add eax,Perimeter_hexagon2
mov TotalPerimeter,eax
mov edx,offset instruction5
call writestring
call writeint
call crlf

exit
main ENDP
END main
```

## ii) After improvement

INCLUDE Irvine32.inc

```
.data
sideHex1 dword 0
sideHex2 dword 0
Perimeter_hexagon1 dword 0
Perimeter_hexagon2 dword 0
Totalperimeter dword 0
instruction1 byte "enter a side value for hexagon 1:",0
instruction2 byte "enter a side value for hexagon 2:",0
instruction3 byte "Perimeter for hexagon 1 with side = ",0
instruction4 byte "Perimeter for hexagon 2 with side = ",0
instruction5 byte "The total perimeter = ",0
instruction6 byte " is :",0

.code
main PROC

mov edx,offset instruction1 ; Displaying instruction to ask for side value of hexagon 1 from user
call writestring
call readint ; Read input for side value of hexagon 1
mov sideHex1,eax

mov edx,offset instruction2 ; Displaying instruction to ask for side vlaue of hexagon 2 from user
call writestring
call readint ; Read input for side value of hexagon 2
mov sideHex2,eax

mov ecx,6
L1: ; Loop Calculating the perimeter of hexagon 1 and hexagon 2

mov eax,sideHex1
add Perimeter_hexagon1,eax
mov eax,sideHex2
add Perimeter_hexagon2,eax
Loop L1

mov eax,Perimeter_hexagon1 ; Calculating total perimeter
add Totalperimeter,eax
mov eax,Perimeter_hexagon2
add Totalperimeter,eax

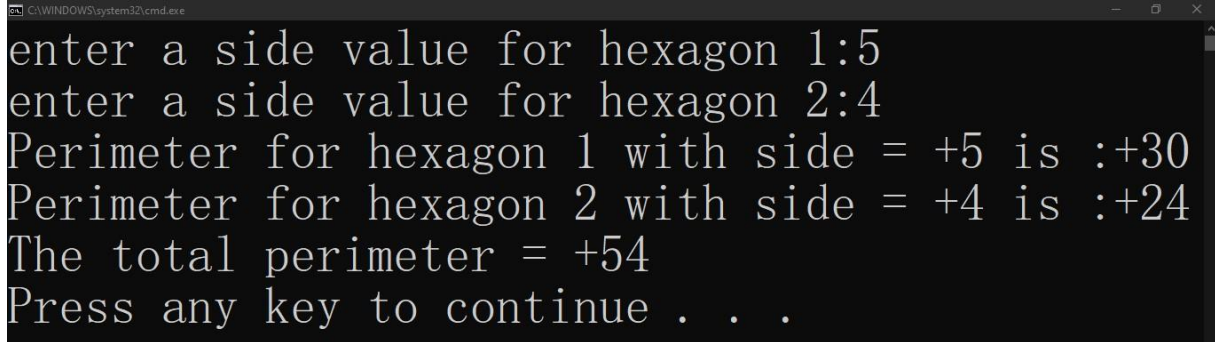
mov edx,offset instruction3 ; Displaying the perimeter of hexagon 1
call writestring
mov eax,sideHex1
call writeint
mov edx,offset instruction6
call writestring
mov eax,Perimeter_hexagon1
call writeint
call crlf

mov edx,offset instruction4 ; Displaying the perimeter of hexagon 2
call writestring
mov eax,sideHex2
call writeint
mov edx,offset instruction6
call writestring
mov eax,Perimeter_hexagon2
call writeint
call crlf

mov edx,offset instruction5 ; Displaying total perimeter for hexagon 1 and 2
call writestring
mov eax,Totalperimeter
call writeint
call crlf

        exit
main ENDP
END main
```

### iii) Output Diagram



```
C:\WINDOWS\system32\cmd.exe
enter a side value for hexagon 1:5
enter a side value for hexagon 2:4
Perimeter for hexagon 1 with side = +5 is :+30
Perimeter for hexagon 2 with side = +4 is :+24
The total perimeter = +54
Press any key to continue . . .
```

### iv) Comment

In data declaration, variables named with sideHex1 and sideHex2 are used to store the side values of hexagon 1 and 2. Furthermore, variables named with Perimeter\_hexagon1, Perimeter\_hexagon2 and Totalperimeter are used to store the perimeter of hexagon 1 and 2 and also total perimeter of both hexagons. 2 string variables are used to display instructions to ask for input from user and the rest of 4 string variables are used to show statement for the respective result obtained.

In the main program, string variables pointed to edx are called by using writestring procedures so that the output displays instructions to ask for inputs from users and readInt procedure get the user input from keyboard and then the input is saved into respective location. After that, 6 is moved into ecx so that loop instructions which are used to calculate the perimeter of hexagon 1 and 2 can repeat for 6 times. By using writeString and writeInt, statements that shows the perimeter of hexagon 1 and 2 are displayed in the output. Before the program ends, the sum of the perimeters of hexagon 1 and 2 is calculated and saved into Totalperimeter and the statement that shows the total perimeter is displayed in the output.

### v) Improvement

After our discussion, we had identified the part of the program that can be further improved which is the part of calculating the perimeter of hexagon 1 and 2. In fact, the program uses two separated loops to calculate the perimeter of hexagon 1 and 2 and this can be simplified by combining the two loops into one that can perform the same operations. This means that the process of adding sideHex1 into Perimeter\_hexagon1 and sideHex2 into Perimeter\_hexagon2 for 6 times can be done in one loop. By doing so, the length of the code and execution time of the program can be shortening which can increase the efficiency of the program.

## **Program 2**

i) Before improvement

```
INCLUDE Irvine32.inc
.data
instruction1 byte "Please enter a multiplicand (1..9): ",0
instruction2 byte "Please enter a multiplier (1..9): ",0
output1 byte "Multiplication of: ",0
multiply byte " x ",0
outputproduct byte "The product is: ",0
Multiplicand dword ?
Multiplier dword ?
Total dword ?
```

```
.code
main PROC
    mov edx, OFFSET instruction1
    call WriteString
    call ReadDec
    mov Multiplicand,eax

    mov edx, OFFSET instruction2
    call WriteString
    call ReadDec
    mov Multiplier,eax

    mov edx, OFFSET output1
    call WriteString
    mov eax,Multiplicand
    call writeDec

    mov edx, OFFSET multiply
    call WriteString
    mov eax,Multiplier
    call WriteDec
    call crlf

    mov edx, OFFSET outputproduct
    call WriteString
    mov eax,1
    mul Multiplicand
    mul Multiplier
    mov Total,eax
    call WriteDec
    call crlf

exit
main ENDP
```

ii) After improvement

```
INCLUDE Irvine32.inc
```

```
.data
```

```
instruction1 byte "Please enter a multiplicand <1..9>: ",0
```

```
instruction2 byte "Please enter a multiplier <1..9>: ",0
```

```
output1 byte "Multiplication of: ",0
```

```
multiply byte "x",0
```

```
outputproduct byte "The product is: ",0
```

```
Multiplicand dword ?
```

```
Multiplier dword ?
```

```
Result dword ?
```

```
.code
```

```
main PROC
```

```
    ;Input the Multiplicand
```

```
    mov edx, OFFSET instruction1
```

```
    call WriteString
```

```
    call ReadDec
```

```
    mov Multiplicand, eax
```

```
    ;Input the Multiplier
```

```
    mov edx, OFFSET instruction2
```

```
    call WriteString
```

```
    call ReadDec
```

```
    mov Multiplier, eax
```

```
    ;Display ( n x m )
```

```
    mov edx, OFFSET output1
```

```
    call WriteString ;print output1
```

```
    mov eax, Multiplicand
```

```
    call WriteDec
```

```
    ;Display Multiplicand
```

```
    mov edx, OFFSET multiply
```

```
    call WriteString
```

```
    ;Display multiply
```

```
    mov eax, Multiplier
```

```
    call WriteDec
```

```
    ;Display Multiplier
```

```
    call crlf
```

```
    ;math operation
```

```
    mov ebx, Multiplicand
```

```
    mul ebx
```

```
    mov Result, eax
```

```
    ;print out the result
```

```
    mov eax, Result
```

```
    mov edx, OFFSET outputproduct ;Display product
```

```
    call WriteString
```

```
    call WriteDec
```

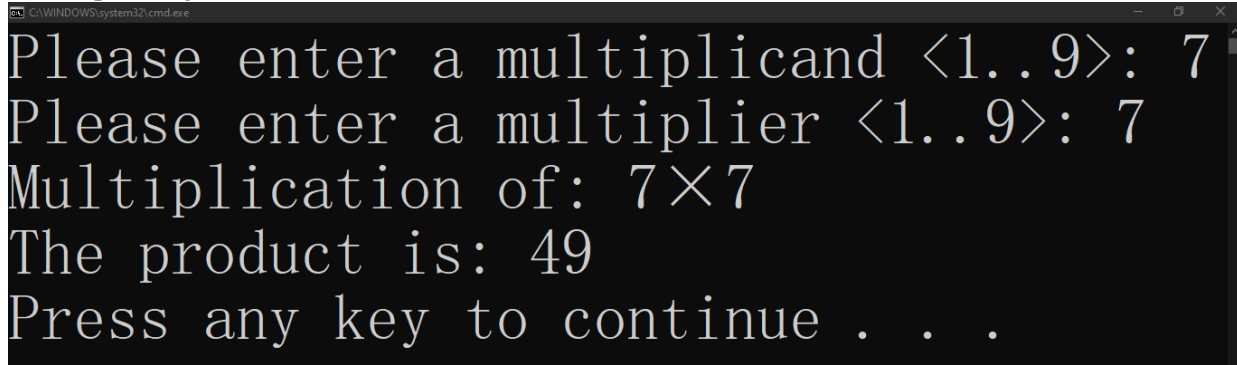
```
    call crlf
```

```
exit
```

```
main ENDP
```

```
END main
```

### iii) Output Diagram



```
C:\WINDOWS\system32\cmd.exe
Please enter a multiplicand <1..9>: 7
Please enter a multiplier <1..9>: 7
Multiplication of: 7x7
The product is: 49
Press any key to continue . . .
```

### iv) Comment

The objective of this program is to do multiplication from user input. In data declaration, variables named with Multiplicand and Multiplier are used to store the values of multiplicand and multiplier of product. Variable named with Result is used to store the values of multiplication result. Furthermore, variables named with instruction1 and instruction2 are used to display instructions to ask for input from user and the variable named with output1, multiply and outputproduct are used to show result of product.

In the main program, string variables pointed to edx are called by using WriteString procedures so that the output displays instructions to ask for inputs from users and ReadDec procedure get the user input from keyboard and then the input is saved into respective location. After that, we display the expression of calculation and do the math operation by using mul to multiply the user input. Before the program ends, the product result is calculated and saved into Result which is displayed in the output.

### v) Improvement

After the discussion with group members in Zoom Meeting, we improve the math operation coding by shorten it. Since the multiplication result will store in eax by default, so we only need to move multiplicand to ebx then write mul ebx (previously eax has stored multiplier), then will get the result of multiplication which has stored in eax. After that move eax to Result to use for display output.

### **Program 3**

#### **i) Before improvement**

```
INCLUDE Irvine32.inc
.data
HELLO DWORD ?,?,?,?,?
TotalOdd DWORD ?
TotalEven DWORD ?
str1 BYTE "Enter Integer : ",0
str2 BYTE "TotalODD is : ",0
str3 BYTE "TotalEVEN is : ",0
.code
main PROC

    mov edx,OFFSET str1      ;point to str1
    call writeString ;display string output
    call readInt             ;read input for Hello [0]
    mov HELLO[0],eax
    call Crlf                ;enter

    mov edx,OFFSET str1
    call writeString
    call readInt             ;read input for Hello [4]
    mov HELLO[4],eax
    call Crlf

    mov edx,OFFSET str1
    call writeString
    call readInt             ;read input for Hello [8]
    mov HELLO[8],eax
    call Crlf

    mov edx,OFFSET str1
    call writeString
    call readInt             ;read input for Hello [12]
    mov HELLO[12],eax
    call Crlf

    mov edx,OFFSET str1
    call writeString
    call readInt             ;read input for Hello [16]
    mov HELLO[16],eax
    call Crlf

    mov edx,OFFSET str1
    call writeString
    call readInt             ;read input for Hello [20]
    mov HELLO[20],eax
    call Crlf

    mov ecx,3                ;set number of loops
    mov eax,0
    mov ebx,0
L2:
    add eax,HELLO[ebx]        ;perform addition for odd number
    add ebx,8
    loop L2
    mov TotalOdd,eax
    mov edx,OFFSET str2
    call writeString
    call WriteDec             ;display total of odd number
    call Crlf

    mov ecx,3
    mov eax,0
    mov ebx,4
L3:
    add eax,HELLO[ebx]        ;perform addition for even number
    add ebx,8
    loop L3
    mov TotalEven,eax
    mov edx,OFFSET str3
    call writeString
    call WriteDec             ;display total of even number
    call Crlf
    call Crlf

    exit
main ENDP
END main
```



ii) After improvement

```
INCLUDE Irvine32.inc
.data
HELLO DWORD ?,?,?,?,?
TotalOdd DWORD ?
TotalEven DWORD ?
str1 BYTE "Enter Integer : ",0
str2 BYTE "TotalODD is : ",0
str3 BYTE "TotalEVEN is : ",0
.code
main PROC
mov ecx,6                ;set number of loops for input
mov ebx,0
mov edx,OFFSET str1      ;point to str1
L1: call writeString
    call readInt         ;read input
    mov HELLO[ebx],eax
    add ebx,4            ;update value of ebx for address
    call Crlf
    loop L1

mov ecx,3                ;set number of loops for addition
mov eax,0
mov ebx,0
L2:
add eax,HELLO[ebx]
add ebx,8
loop L2

mov TotalOdd,eax
mov edx,OFFSET str2      ;point to str2
call writeString
call WriteDec            ;display TotalOdd
call Crlf

mov ecx,3                ;set number of loops for addition
mov eax,0
mov ebx,4
L3:
add eax,HELLO[ebx]
add ebx,8
loop L3

mov TotalEven,eax
mov edx,OFFSET str3      ;point to str3
call WriteString
call WriteDec            ;display TotalEven
call Crlf
call Crlf

exit
main ENDP
END main
```

### iii) Output Diagram

```
C:\WINDOWS\system32\cmd.exe
Enter Integer : 32
Enter Integer : 65
Enter Integer : 77
Enter Integer : 89
Enter Integer : 14
Enter Integer : 54
TotalODD is : 123
TotalEVEN is : 208
Press any key to continue . . .
```

### iv) Comment

The objective of this program is to read the input and store it into the array. In data declaration, HELLO variable with 16 bits size is the array that can store 6 content. Next, both TotalOdd and TotalEven is to store the sum of the odd number of arrays and even number of arrays. Three string declaration are inserted for the instruction to the user to insert the data and the others would be showing the statement for the respective result obtained.

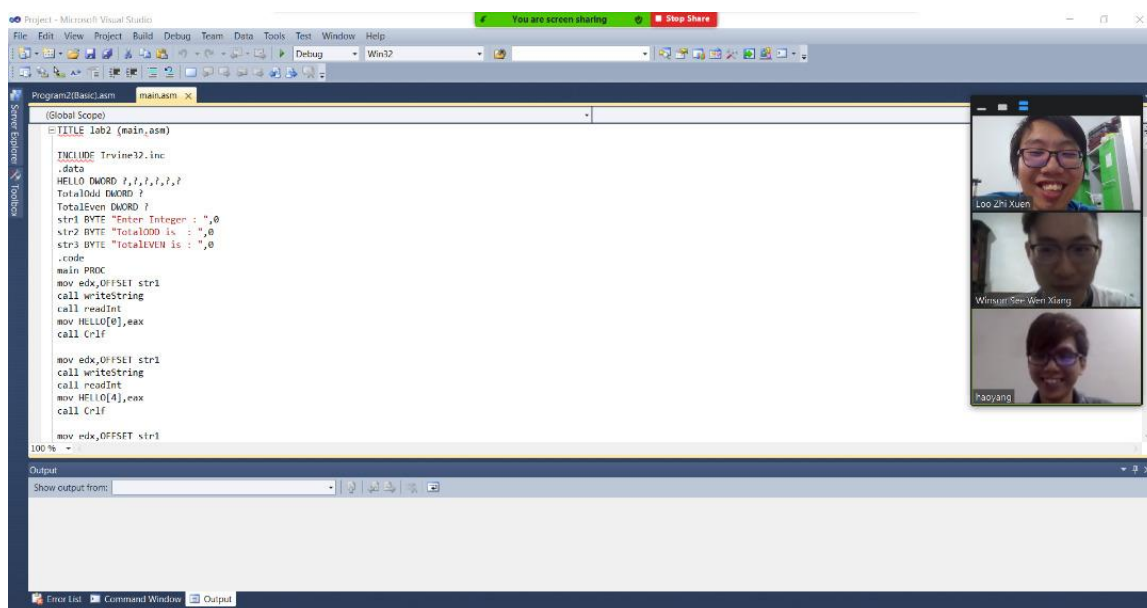
In the main function, point the edx to the string declaration and display it out to lead the user to key in the data required. Next, using readInt function to get the data from the user and then store in the respective location in array. Then, the content of array is extracted for the next step which is sum up the odd number of arrays which is the first Hello, HELLO [0], third Hello [8] and fifth Hello, HELLO [16]. The total will be stored into the variable TotalOdd. While for the total of the even number of arrays is consists of second Hello, HELLO [4], fourth Hello, HELLO [12] and sixth Hello, HELLO [20] and will be stored into variable TotalEven. Finally, with the function writeString and writeDec, the statement with the content in the variables will be displayed.

### v) Improvement

After the discussion with group member in WhatsApp, the improvement that we done in this program is at the part whereby we need to read the input. The program is initially created with the idea read the input once by once in the coding whereby we need to insert the coding for pointing the edx to str1, writeString and ReadInt until reach the iteration we required. But, after the improvement, the code above only must write once with the help of loop. This is because those code are performing the same function even though they are repeating more than once. It is better to use the loop as it provides better efficiency in term of performing the coding.

## DESCRIPTION FOR MEETING

The classification of the task for Part C is conducted by using WhatsApp. Kong Hao Yang will be responsible for Question 1, See Wen Xiang for Question 2, and Loo Zhi Xuen for Question 3. Then, we have our meeting at 4<sup>th</sup> of June at 8pm by using Zoom. In the meeting, we share our code one by one and having the discussion to improve the code during the meeting. At the same time, comments are added in the program to make the code is easier to understand. Then, we will try to run the code to make sure the output is same with the question required. After the improvements of the program, our next schedule will be discussing about the content. Finally, we decide to split the report into few parts, which the report are included with before and improved program, output of the program, description about how the program is functioning, description of the improvement that we have done and the proof that we have for the meeting.

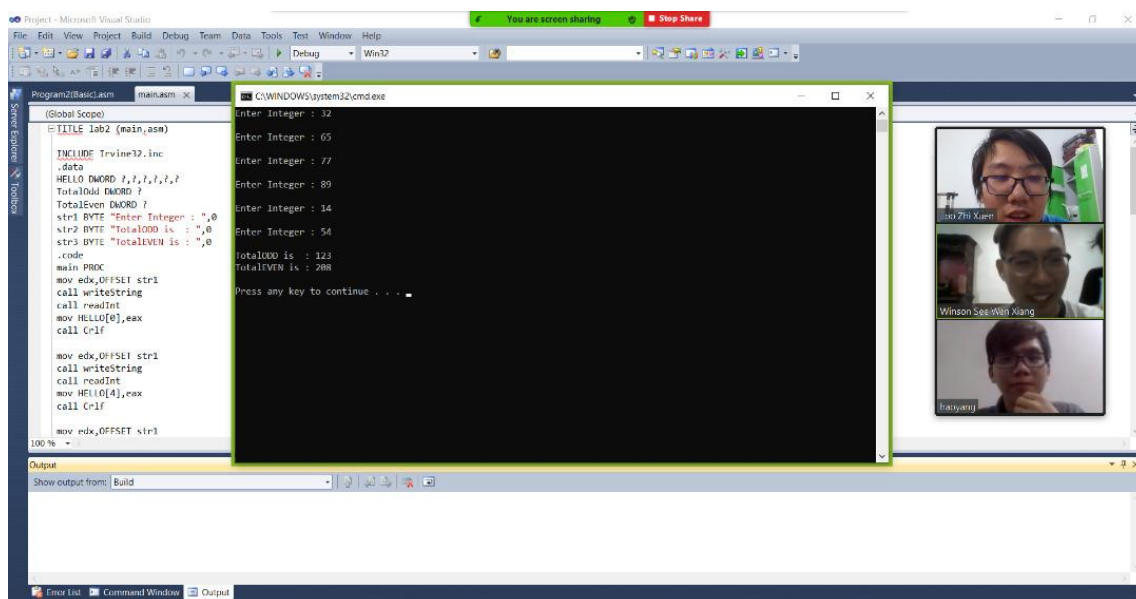


```
(Global Scope)
TITLE lab2 (main.asm)

INCLUDE Irvine32.inc
.data
HELLO DWORD ?,?,?,?,?
TotalOdd DWORD ?
TotalEven DWORD ?
str1 BYTE "Enter Integer : ",0
str2 BYTE "TotalOdd is : ",0
str3 BYTE "TotalEven is : ",0
.code
main PROC
    mov edx,OFFSET str1
    call writeString
    call readInt
    mov HELLO[0],eax
    call Crlf

    mov edx,OFFSET str1
    call writeString
    call readInt
    call readInt
    mov HELLO[4],eax
    call Crlf

    mov edx,OFFSET str1
```



```
C:\WINDOWS\system32\cmd.exe
Enter Integer : 32
Enter Integer : 65
Enter Integer : 77
Enter Integer : 89
Enter Integer : 14
Enter Integer : 54
TotalOdd is : 123
TotalEven is : 268
Press any key to continue . . .
```