Computer Organization and Architecture
SECR 2033
Semester 2, 2019/2020

Programming 4: Interactive Usage of Link Libraries

Part A – Programming review

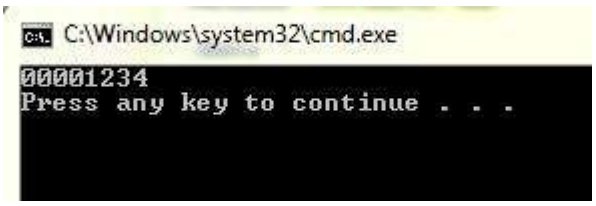
Link Library Overview

- A file containing procedures that have been compiled into machine code
- These procedures are ready to be used (via the CALL instruction) within your program. They have their own unique name.
- Most, if not all, of these procedures run on pre-selected registers which you MUST use.

Calling a Library Procedure

- Call a library procedure using the CALL instruction. Some procedures require input arguments.
- The INCLUDE directive copies in the procedure prototypes (declarations).
- The following example displays "1234" on the console (as shown in output):

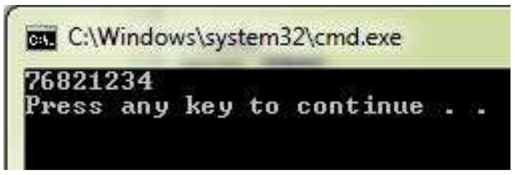
```
INCLUDE Irvine32.inc
.code
    mov eax,1234h        ; input argument      call
WriteHex ; show hex number
    call Crlf            ; end of line
```



```
C:\Windows\system32\cmd.exe
00001234
Press any key to continue . . .
```

- In the example above, register EAX must be used for it to work correctly. Try the command below and see what happens.

```
INCLUDE Irvine32.inc
.code
    mov ax,1234h         ; input argument
    call WriteHex        ; show hex number
    call Crlf            ; end of line
```



```
C:\Windows\system32\cmd.exe
76821234
Press any key to continue . .
```

Library Procedures – Overview

Here are some of the procedures available to you. You can find more from the Internet and reference books (Kip Irvine, Assembly Language programming books are a good place to start).

- Clrscr - Clears the console and locates the cursor at the upper left corner.
- Crlf - Writes an end of line sequence to standard output.
- Delay - Pauses the program execution for a specified *n* millisecond interval.
- DumpMem - Writes a block of memory to standard output in hexadecimal.
- DumpRegs - Displays the EAX, EBX, ECX, EDX, ESI, EDI, EBP, ESP, EFLAGS, and EIP registers in hexadecimal. Also displays the Carry, Sign, Zero, and Overflow flags.
- GetCommandtail - Copies the program’s command-line arguments (called the *command tail*) into an array of bytes.
- GetMseconds - Returns the number of milliseconds that have elapsed since midnight.
- Gotoxy - Locates cursor at row and column on the console.
- Random32 - Generates a 32-bit pseudorandom integer in the range 0 to FFFFFFFFh.
- Randomize - Seeds the random number generator.
- RandomRange - Generates a pseudorandom integer within a specified range.
- ReadChar - Reads a single character from standard input.
- ReadHex - Reads a 32-bit hexadecimal integer from standard input, terminated by the Enter key.
- ReadInt - Reads a 32-bit signed decimal integer from standard input, terminated by the Enter key.
- ReadString - Reads a string from standard input, terminated by the Enter key.
- SetTextColor - Sets the foreground and background colors of all subsequent text output to the console.
- WaitMsg - Displays message, waits for Enter key to be pressed.
- WriteBin - Writes an unsigned 32-bit integer to standard output in ASCII binary format.
- WriteChar - Writes a single character to standard output.
- WriteDec - Writes an unsigned 32-bit integer to standard output in decimal format.
- WriteHex - Writes an unsigned 32-bit integer to standard output in hexadecimal format.
- WriteInt - Writes a signed 32-bit integer to standard output in decimal format.
- WriteString - Writes a null-terminated string to standard output.

Computer Organization and Architecture
SECR 2033
Semester 2, 2019/2020

<u>Part B – Let's do a little programming by example</u>
You are given a few examples here. Try them out.

Example 1
Clear the screen, delay the program for **500 milliseconds**, and dump the registers and flags.

```
.code
 call Clrscr  mov  eax,500  call
Delay
          call DumpRegs
```

Example 2
Display a null-terminated **string** and move the cursor to the beginning of the next screen line. Attach the output screen capture for this example.

```
.data
str1 BYTE "Assembly language is easy!",0
.code
 mov  edx,OFFSET str1  call WriteString
          call Crlf
```

Example 3
Display an unsigned integer in binary, decimal, and hexadecimal, each on a separate line. Attach the output screen capture for this example.

```
.data
IntVal = 35
.code
 mov  eax,IntVal  call WriteBin ; display binary
          call Crlf
          call WriteDec ; display decimal
          call Crlf
          call WriteHex ; display hexadecimal
          call Crlf
```

Example 4
Input a string from the user (*ReadString*). EDX points to the string. Attach the output screen capture for this example. *(**Tips: It is always a good practice to have a string to ask for input)*

```
.data
str2 BYTE "Give me your name:  ",0
buffer2 BYTE 21 DUP(0) ; input buffer
```

```
.code    mov
edx,OFFSET
buffer2 ; point
to the buffer
mov
ecx,SIZEOF
buffer2 ; specify
max characters
    call ReadString ; input the string

    mov edx, OFFSET buffer2 ; point to the buffer
    call WriteString
    call crlf
```

Example 5

Input a decimal number from the user (*ReadDec*).. The procedure reads a 32bit unsigned decimal integer from the keyboard and returns the value in EAX. **Output a number** to screen (*WriteDec*). The procedure writes a 32-bit unsigned integer to the console window in decimal format with no leading zeros. Pass the integer in EAX. Attach the output screen capture for this example. *(**Tips: It is always a good practice to have a string to ask for input)*

```
.data
str1 BYTE "Enter a decimal:  ",0 val1 dword ?

.code
    mov edx, offset str1
    call writestring

    call ReadDec              mov
val1,eax

    mov eax,val1
    call WriteDec
```

Example 6

Generate and display ten pseudorandom signed integers in the range 0 – 99. Pass each integer to WriteInt in EAX and display it on a separate line. Attach the output screen capture for this example.

```
.code
    mov ecx,10     ; loop counter

L1:  mov    eax,100  ;  ceiling  value    call  RandomRange  ;
generate random int  call WriteInt  ; display signed int
    call Crlf        ; goto next display line
    loop L1          ; repeat loop
```

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

Program 1

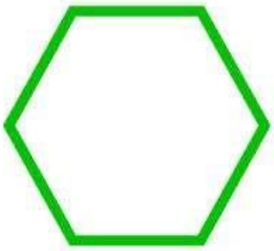


Figure 1: A hexagon

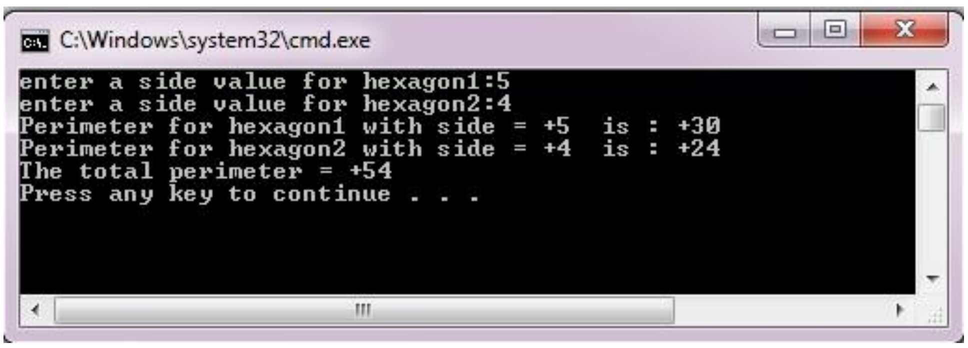
Figure 1 is illustrates a hexagon figure with same length of side. To calculate the perimeter of the hexagon, the following formula is given.

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.



```
enter a side value for hexagon1:5
enter a side value for hexagon2:4
Perimeter for hexagon1 with side = +5  is : +30
Perimeter for hexagon2 with side = +4  is : +24
The total perimeter = +54
Press any key to continue . . .
```

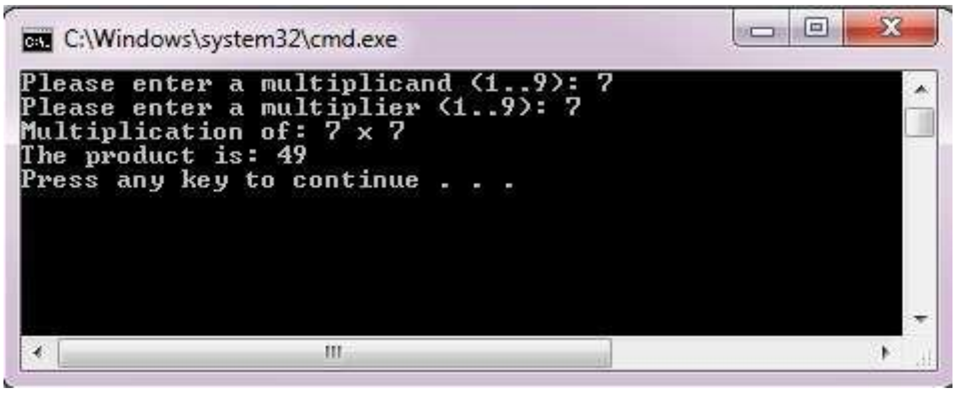
Figure 2: The Output

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

Program 2

- Write a program in assembly language to multiply 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 . . .
```

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.

Program 3

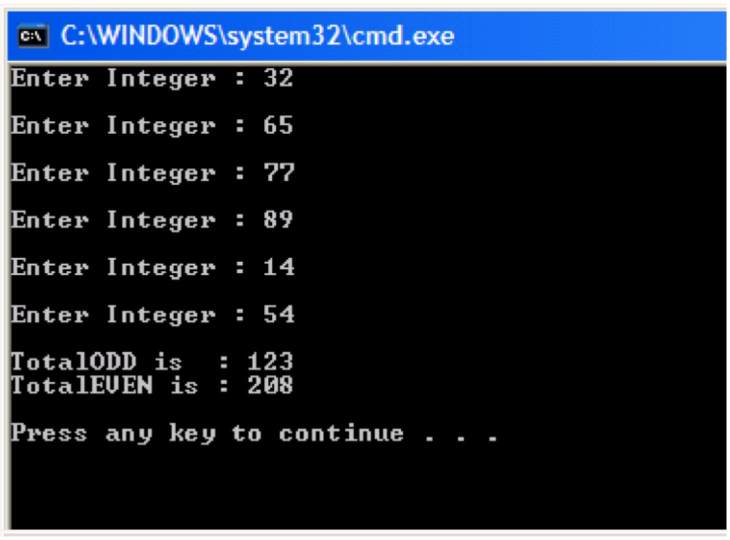
Write a program that will **interactively** ask the **user to input the values of 6 integers** in DWORD and you have to put the values into an array name HELLO.

- 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



```
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 . . .
```

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.