**LAB 06**

**PROCEDURES & FILE HADILING**



Mohammad Yehya Hayati K213309 G

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_ \_\_\_

STUDENT NAME ROLL NO SEC

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

SIGNATURE & DATE



EL

2003

COMP ORG & ASSEMBLY

LANGUAGE

LAB

LABORATORY MANUAL



Fall 2021



Instructor:

Aamir Ali

, Aashir Mehboob, Amin Sadiq, Qurat

-

ul

-

Ain, M Kariz, Rabia

**MARKS AWARDED: \_\_\_\_\_\_\_\_\_\_\_\_\_\_**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES**

**(NUCES), KARACHI**

Version: 1.0

Prepared by: Aamir Ali th

Date: 06 Oct 2021

Lab Session 06:PROCEDURES & FILE HANDILING

# Objectives:

* Built-in-Procedure

* PROC Directive
* Call & Ret Instructions
* File Handiling

# Procedure in Irvine32 Library:

Some of the procedures available in Irvine32 library are:

1. **Clrscr:**

Clears the console window and locates the cursor at the above left corner.

1. **Crlf:**

Writes the end of line sequence to the console window.

1. **DumpRegs:**

Displays the EAX, EBX, ECX, EDX, ESI, EDI, ESP:EIP and EFLAG registers.

1. **DumpMem(ESI=Starting OFFSET, ECX=LengthOf, EBX=Type):**

Writes the block of memory to the console window in hexadecimal.

1. **WriteBin:**

Writes an unsigned 32-bit integer to the console window in ASCII binary format.

1. **WriteChar:**

Writes a single character to the console window.

1. **WriteDec:**

Writes an unsigned 32-bit integer to the console window in decimal format.

1. **WriteHex:**

Writes a 32-bit integer to the console window in hexadecimal format.

1. **WriteInt:**

Writes a signed 32-bit integer to the console window in decimal format.

1. **WriteString(EDX= OFFSET String):**

Write a null-terminated string to the console window.

1. **ReadChar:**

Waits for single character to be typed at the keyboard and returns that character.

1. **ReadDec:**

Reads an unsigned 32-bit integer from the keyboard.

1. **ReadHex:**

Reads a 32-bit hexadecimal integers from the keyboard, terminated by the enter key.

1. **ReadInt:**

Reads a signed 32-bit integer from the keyboard, terminated by the enter key.

1. **ReadString(EDX=OFFSET String, ECX=SIZEOF):**

Reads a string from the keyboard, terminated by the enter key.

1. **SetTextColor(Background= Upper AL, Foreground= Lower AL):**

Sets the foreground and background colors of all subsequent text output to the console.

1. **GetTextColor(Background= Upper AL, Foreground= Lower AL):**

Returns the active foreground and background text colors in the console window.

1. **MsgBox(EDX=OFFSET String, EBX= OFFSET Title):**

Displays a pop-up message box.

1. **MsgBoxAsk (EDX=OFFSET String, EBX= OFFSET Title):**

Displays a yes/no question in a pop-up message box.

1. **WaitMsg:**

Display a message and wait for the Enter key to be pressed.

1. **Delay:**

Pauses the program execution for a specified interval (in milliseconds).

1. **getDateTime:**

Gets the current date and time from system

1. **GetMaxXY(DX=col, AX=row):**

Gets the number of columns and rows in the console window buffer.

1. **Gotoxy (DH=row , DL=col):**

Locates the cursor at a specific row and column in the console window. By default X coordinate range is 0-79 and Y coordinate range is 0-24.

1. **Randomize:**

Seeds the random number generator with a unique value.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | Color and Its Value | |  |  |  |
| Color | Value | Color | Value | Color | Value | Color | Value |
| Black | 0 | Red | 4 | Gray | 8 | Light Red | C |
| Blue | 1 | Magneta | 5 | Light Blue | 9 | Light Magenta | D |
| Green | 2 | Brown | 6 | Light Green | A | Yellow | E |
| Cyan | 3 | Light Gray | 7 | Light Cyan | B | White | h |

**Example 01:**

**Gotoxy (DH=row , DL=col):**

Locates the cursor at a specific row and column in the console window. By default X coordinate range is 0-79 and Y coordinate range is 0-24.

Include Irvine32.inc

.code main proc call Clrscr mov dh, 24

mov dl, 79 ; bottom-right corner

call Gotoxy ; Move cursor there mov al, '\*'

call WriteChar ; Write '\*' in bottom right

call ReadChar ; Character entered by user is in AL mov dh, 10 mov dl, 10 call Gotoxy

call WriteChar ; Output the character entered at 10,10 call CrLf ; Carriage return to line 11

call DumpRegs ; Output registers

; output a row of '&'s to the screen, minus first column

mov al, '&' mov cx, 79

|  |  |  |  |
| --- | --- | --- | --- |
| mov dh, 5 L1: mov dl, cl  call Gotoxy    call WriteChar loop L1 call CrLf exit  main ENDP END main    **Here are some more:** |  |  | ; row 5 |
| **Randomize** | Initialize random number seed | | |
| **Random32** | Generate a 32 bit random integer and return it in eax | | |
| **RandomRange** | Generate random integer from 0 to eax-1 | | |

**Example 02:**

Include Irvine32.inc

.data

myInt DWORD ? myChar BYTE ?

myStr BYTE 30 dup(0) myPrompt BYTE "Enter a string:",0

myPrompt2 BYTE "Enter a number:",0

.code main proc

; Output 2 random numbers

|  |  |
| --- | --- |
| call Randomize  call Random32 | ; Only call randomize once |
| call WriteInt call Crlf  call RandomRange | ; output EAX as int |
| call WriteInt  call Crlf  ; Get and display a string mov edx, offset myprompt | ; output EAX as int |
| call Writestring | ; Display prompt |
| mov ecx, 30 mov edx, offset myStr call Readstring | ; Max length of 30 |
| call Writestring | ; Output what was typed |

Call Crlf

; Get a number and display it mov edx, offset myprompt2 call Writestring ; Display prompt

call ReadInt ; Int stored in EAX call Crlf call WriteInt

call Crlf

exit

main endp end main

**Example 03:**

Include Irvine32.inc

.data

msg byte "Genrating 50 number",0

.code

main PROC mov edx,offset msg call WriteString

call crlf

mov ecx,50 L1:

mov eax,+33 call RandomRange call writeDec

call Crlf Loop L1

exit

main endp

end main

# Writing Procedures

You have already been defining your own procedures – the main procedure works just like any other procedure.

The format to define a procedure is:

<Procedure-Name> proc

…

… ; code for procedure

…

ret ; Return from the procedure

<Procedure-Name> endp

The keyword proc indicates the beginning of a procedure, and the keyword endp signals the end of the procedure. Your procedure must use the RET instruction when the procedure is finished. This causes the procedure to return by popping the instruction pointer off the stack.

**To invoke a procedure, use call:**

**call procedure-name**

**Example** **04: (Addition of Two Numbers)**

INCLUDE Irvine32.inc

.data

var1 DWORD 5 var2 DWORD 6

.code

main PROC call AddTwo

call writeint call crlf

exit

main ENDP AddTwo PROC

mov eax,var1 mov ebx,var2

add eax,var2

ret

AddTwo ENDP

END main

**Example 05: (Addition of Elements within an Array)**

INCLUDE Irvine32.inc

.data

myarray DWORD 1,2,3,4,5,6

.code

main PROC call ArraySum call writeint call crlf

exit

main ENDP ArraySum PROC mov esi,0 mov eax,0

mov ecx, LENGTHOF myarray

L1:

add eax,myarray[esi] add esi,4

Loop L1

ret

ArraySum ENDP

END main

# FILING HANDLING

**Creating a New File**

EAX contains the newly created file’s handle or INVALID\_HANDLE\_VALUE if creation is unsuccessful.

**Opening an Existing File**

Offset of file name is passed to EDX. Handle ofopened file is returned in EAX

**Reading From a File**

**Call arguments:**

EAX = an open file handle

EDX = offset of the input buffer

ECX = maximum number of bytes to read

**Return arguments:**

If CF = 0, EAX contains the number of bytes read. If CF = 1, EAX contains a system error code.

**Writing To a File:**

**Call arguments:**

EAX = an open file handle

EDX = offset of the buffer

ECX = maximum number of bytes to write

**Return arguments:**

If CF = 0, EAX contains the number of bytes written.

If CF = 1, EAX contains a system errorcode.

**Example 06**

; Creating a File (CreateFile.asm)

INCLUDE Irvine32.inc

BUFFER\_SIZE = 501

.data

buffer BYTE BUFFER\_SIZE DUP(?) filename BYTE "output.txt",0 fileHandle HANDLE ? stringLength DWORD ? bytesWritten DWORD ?

str2 BYTE "Bytes written to file [output.txt]:",0

str3 BYTE "Enter up to 500 characters and press"

BYTE "[Enter]: ",0dh,0ah,0

.code

main PROC

; Create a new text file. mov edx,OFFSET filename call CreateOutputFile mov fileHandle,eax

; Ask the user to input a string.

mov edx,OFFSET str3 ; "Enter upto ...." call WriteString

mov ecx,BUFFER\_SIZE ; Input a string mov edx,OFFSET buffer call ReadString mov stringLength,eax ; counts chars entered ; Write the buffer to the output file.

mov eax,fileHandle mov edx,OFFSET buffer mov ecx,stringLength call WriteToFile

mov bytesWritten,eax ; save return value

call CloseFile ; Display the return value.

mov edx,OFFSET str2 ; "Bytes written"

call WriteString mov eax,bytesWritten call WriteDec call Crlf

exit

main ENDP

END main

**Example 07**

; Reading a File (ReadFile.asm)

; Opens, reads, and displays a text file using ; procedures from Irvine32.lib. INCLUDE Irvine32.inc INCLUDE macros.inc

BUFFER\_SIZE = 5000

.data

buffer BYTE BUFFER\_SIZE DUP(?) filename BYTE 80 DUP(0) fileHandle HANDLE ?

.code

main PROC

; Let user input a filename.

mWrite "Enter an input filename: "

mov edx,OFFSET filename

mov ecx,SIZEOF filename call ReadString ; Open the file for input. mov edx,OFFSET filename call OpenInputFile mov fileHandle,eax

; Read the file into a buffer. mov edx,OFFSET buffer mov ecx,BUFFER\_SIZE call ReadFromFile mov buffer[eax],0 ; insert null terminator mWrite "File size: " call WriteDec ; display file size

call Crlf ; Display the buffer.

mWrite <"Buffer:",0dh,0ah,0dh,0ah> mov edx,OFFSET buffer ; display the buffer

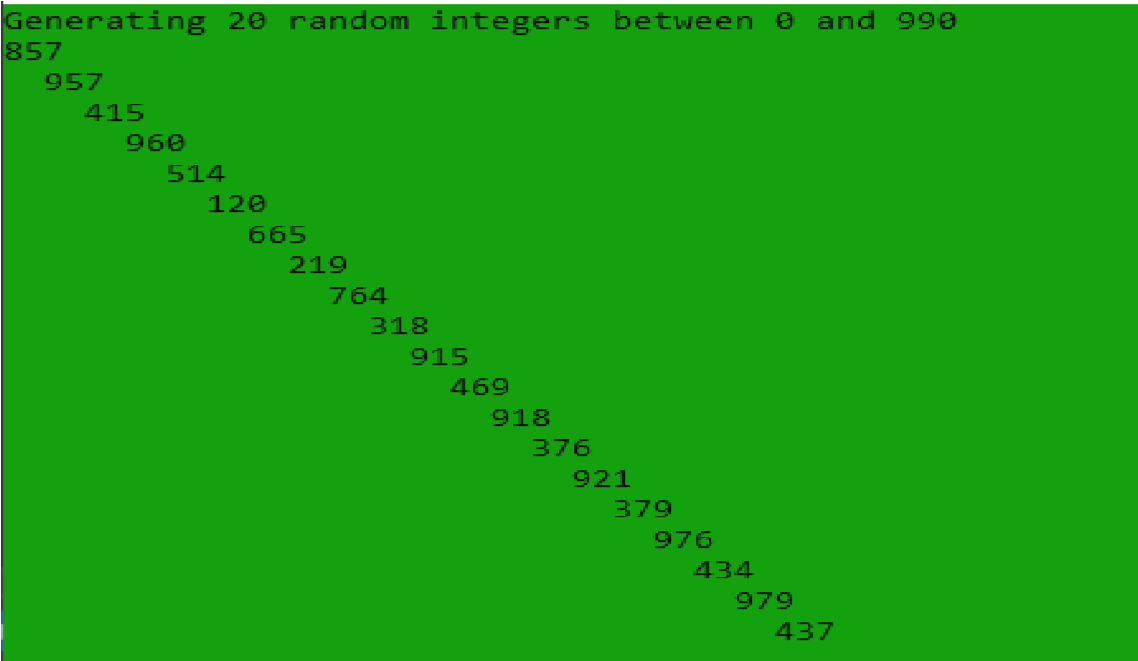
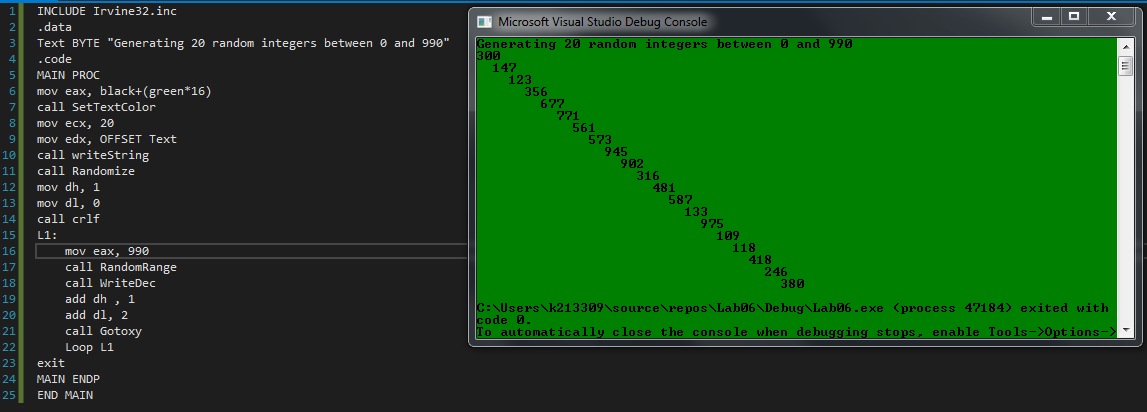
call WriteString call Crlf mov eax,fileHandle call CloseFile exit

main ENDP

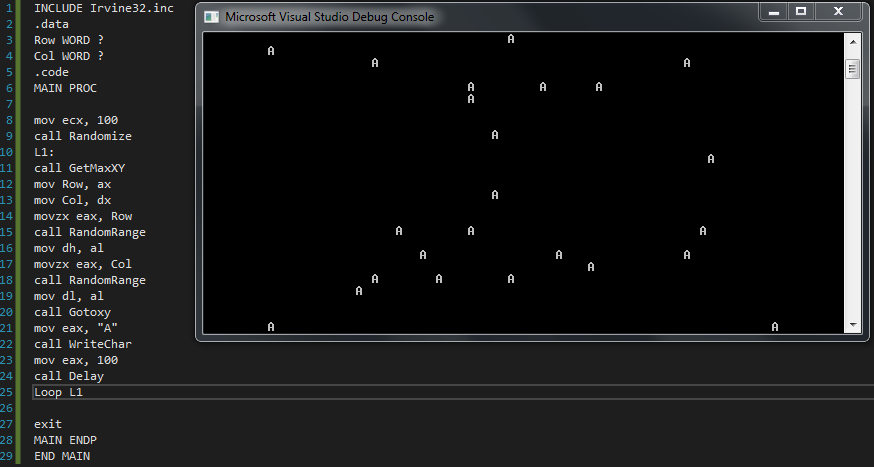
END main

# Lab Task(s):

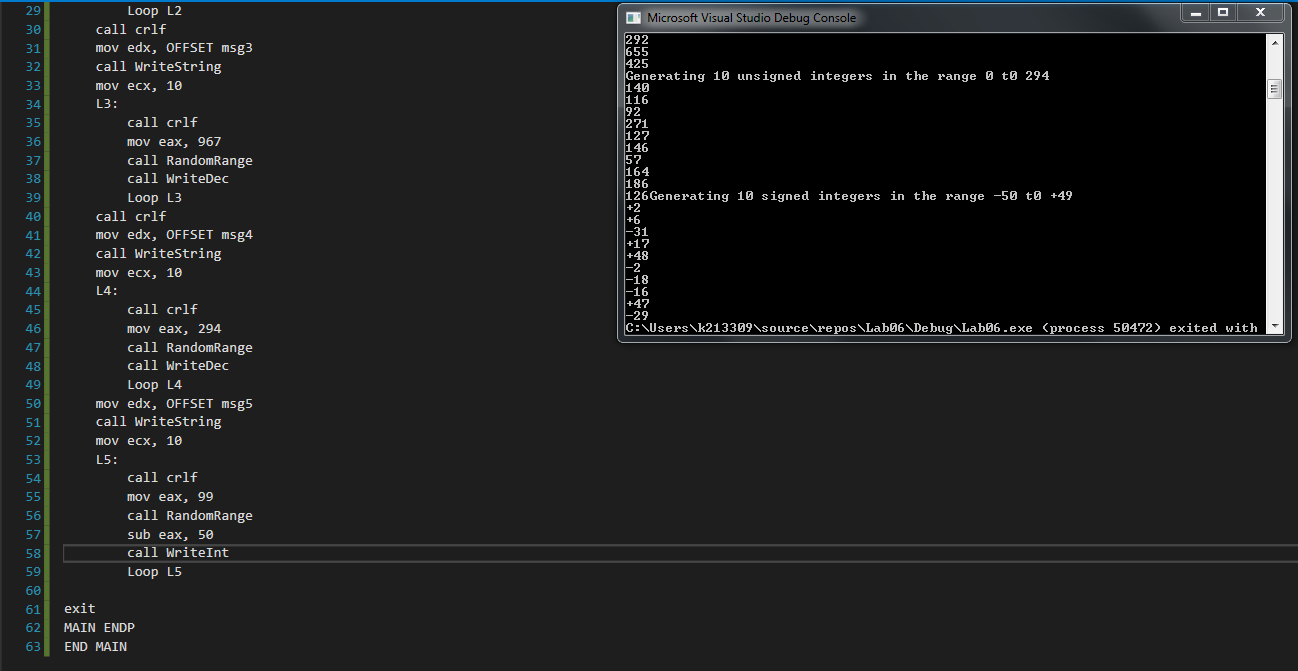
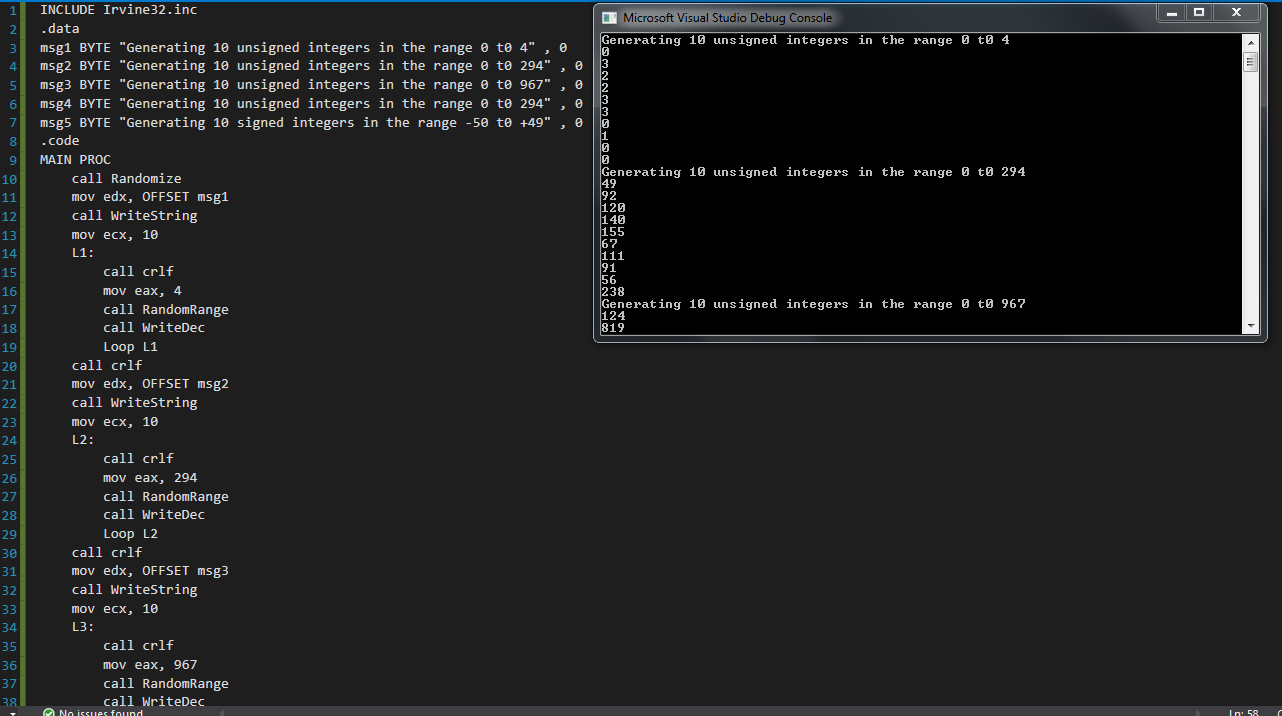
1. Write a program to display a list of 20 random numbers in diagonal pattern. Add a 5 millisecond delay before displaying each number.

1. Write a progam to display a single character at 100 random screen locations, using a timing delay of 100 millisecond. (**Hint: Use GetMaxXY and movzx procedures)**



1. Write a program to generate 10 unsigned integers in the range 0 t0 4,294,967,294 and 10 signed integers in the range -50 t0+49.



1. Make a program to create a text file name MyFile.txt and write a stringin file.