

LAB 06

PROCEDURES & FILE HANDLING



STUDENT NAME

ROLL NO

SEC

SIGNATURE & DATE

MARKS AWARDED: _____

NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES
(NUCES), KARACHI

Prepared by: Aamir Ali

Version: 1.0
Date: 06th Oct 2021

Lab Session 06: PROCEDURES & FILE HANDLING

Objectives:

- Built-in-Procedure
- PROC Directive
- Call & Ret Instructions
- File Handling

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.
2. **Crlf:**
Writes the end of line sequence to the console window.
3. **DumpRegs:**
Displays the EAX, EBX, ECX, EDX, ESI, EDI, ESP:EIP and EFLAG registers.
4. **DumpMem (ESI=Starting OFFSET, ECX=LengthOf, EBX=Type):**
Writes the block of memory to the console window in hexadecimal.
5. **WriteBin:**
Writes an unsigned 32-bit integer to the console window in ASCII binary format.
6. **WriteChar:**
Writes a single character to the console window.
7. **WriteDec:**
Writes an unsigned 32-bit integer to the console window in decimal format.
8. **WriteHex:**
Writes a 32-bit integer to the console window in hexadecimal format.
9. **WriteInt:**
Writes a signed 32-bit integer to the console window in decimal format.
10. **WriteString (EDX= OFFSET String):**
Write a null-terminated string to the console window.
11. **ReadChar:**
Waits for single character to be typed at the keyboard and returns that character.
12. **ReadDec:**
Reads an unsigned 32-bit integer from the keyboard.
13. **ReadHex:**
Reads a 32-bit hexadecimal integers from the keyboard, terminated by the enter key.
14. **ReadInt:**
Reads a signed 32-bit integer from the keyboard, terminated by the enter key.
15. **ReadString (EDX=OFFSET String, ECX=SIZEOF):**
Reads a string from the keyboard, terminated by the enter key.
16. **SetTextColor (Background= Upper AL, Foreground= Lower AL):**
Sets the foreground and background colors of all subsequent text output to the console.
17. **GetTextColor (Background= Upper AL, Foreground= Lower AL):**
Returns the active foreground and background text colors in the console window.
18. **MsgBox (EDX=OFFSET String, EBX= OFFSET Title):**



Displays a pop-up message box.

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

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

20. **WaitMsg:**

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

21. **Delay:**

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

22. **getDateTime:**

Gets the current date and time from system

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

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

24. **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.

25. **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	Magenta	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                ; row 5
L1: mov dl, cl
    call Gotoxy
    call WriteChar
loop L1
call CrLf
exit
main ENDP
END main

```

Here are some more:

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                ; Only call randomize once
call Random32
call WriteInt                 ; output EAX as int
call CrLf
call RandomRange
call WriteInt                 ; output EAX as int
call CrLf
; Get and display a string
mov edx, offset myprompt
call Writestring              ; Display prompt
mov ecx, 30                   ; Max length of 30
mov edx, offset myStr
call Readstring
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 of opened 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 error code.

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

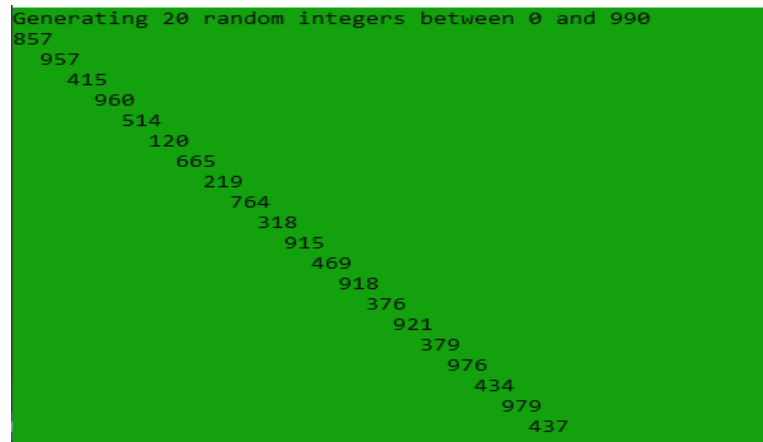
```
; Reading a File (ReadFile.asm)
; Opens, reads, and displays a text file using
; procedures from Irvine32.lib.
INCLUDE Irvine32.inc
INCLUDE macros.inc
BUFFER_SIZE = 5000
.data
buffer BYTE BUFFER_SIZE DUP(?)
filename BYTE 80 DUP(0)
fileHandle HANDLE ?
```




```
.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.

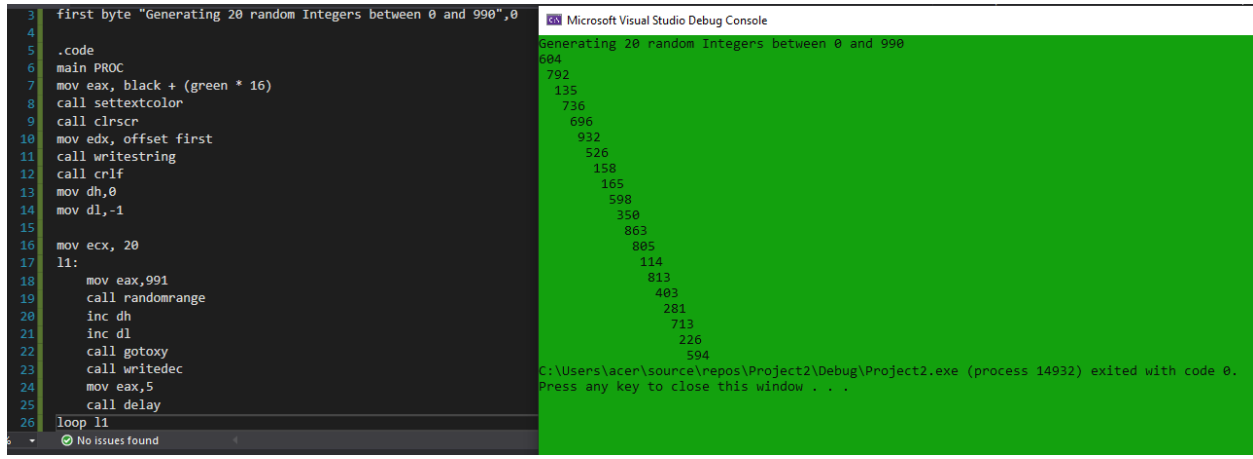


```
Generating 20 random integers between 0 and 990
857
 957
 415
 960
 514
 120
 665
 219
 764
 318
 915
 469
 918
 376
 921
 379
 976
 434
 979
 437
```

2. Write a program to display a single character at 100 random screen locations, using a timing delay of 100 millisecond. (Hint: Use GetMaxXY and movzx procedures)
3. Write a program to generate 10 unsigned integers in the range 0 to 4,294,967,294 and 10 signed integers in the range -50 to +49.
4. Make a program to create a text file name MyFile.txt and write a string in file.

LAB 6

Task 1: Code + Output



The screenshot shows the Visual Studio IDE with an assembly file open on the left and the Microsoft Visual Studio Debug Console on the right. The assembly code generates 20 random integers between 0 and 990 and displays them in a green console window.

```
3 first byte "Generating 20 random Integers between 0 and 990",0
4
5 .code
6 main PROC
7 mov eax, black + (green * 16)
8 call settextcolor
9 call clrscr
10 mov edx, offset first
11 call writestring
12 call crlf
13 mov dh,0
14 mov dl,-1
15
16 mov ecx, 20
17 l1:
18 mov eax,991
19 call randomrange
20 inc dh
21 inc dl
22 call gotoxy
23 call writedec
24 mov eax,5
25 call delay
26 loop l1
```

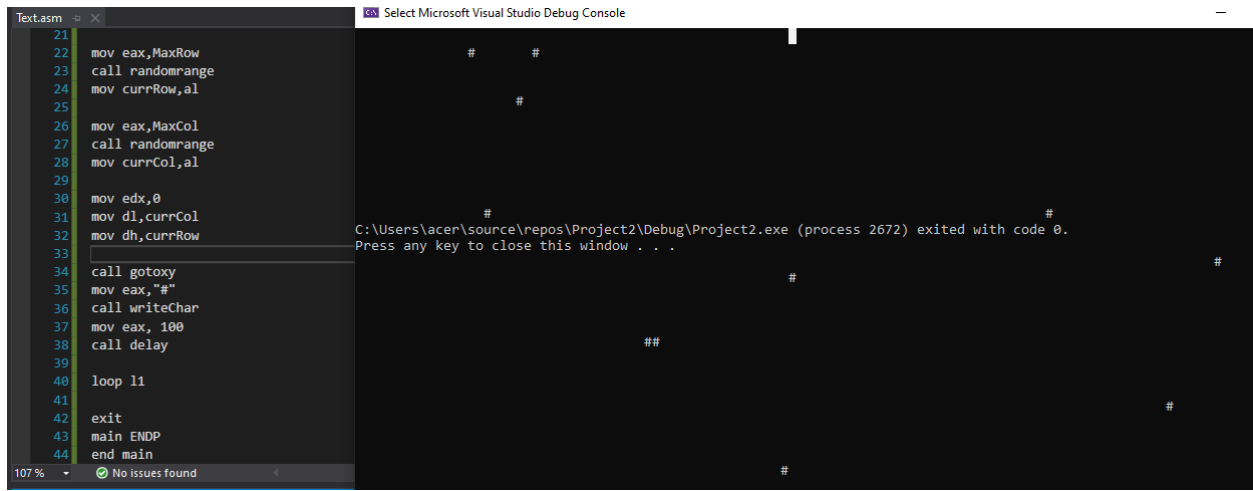
Microsoft Visual Studio Debug Console

Generating 20 random Integers between 0 and 990

604
792
135
736
696
932
526
158
165
598
350
863
805
114
813
403
281
713
226
594

C:\Users\acer\source\repos\Project2\Debug\Project2.exe (process 14932) exited with code 0.
Press any key to close this window . . .

Task 2: Code + Output



The screenshot shows the Visual Studio IDE with an assembly file open on the left and the Microsoft Visual Studio Debug Console on the right. The assembly code generates a pattern of '#' characters in a console window.

```
21
22 mov eax,MaxRow
23 call randomrange
24 mov currRow,al
25
26 mov eax,MaxCol
27 call randomrange
28 mov currCol,al
29
30 mov edx,0
31 mov dl,currCol
32 mov dh,currRow
33
34 call gotoxy
35 mov eax,"#"
36 call writeChar
37 mov eax, 100
38 call delay
39
40 loop l1
41
42 exit
43 main ENDP
44 end main
```

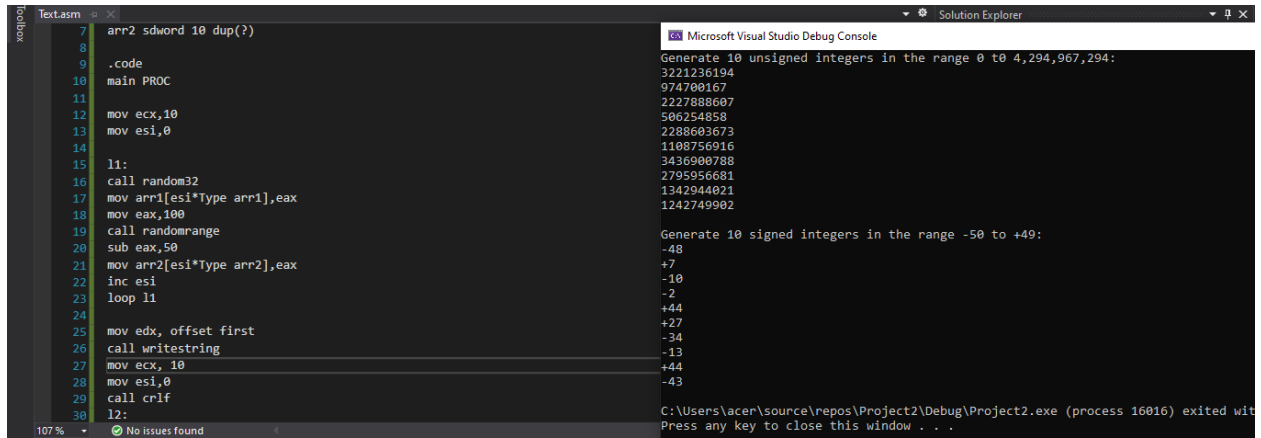
Select Microsoft Visual Studio Debug Console

#

C:\Users\acer\source\repos\Project2\Debug\Project2.exe (process 2672) exited with code 0.
Press any key to close this window . . .

LAB 6

Task 3: Code + Output



The screenshot shows the Visual Studio IDE with an assembly file named 'Text.asm' open. The code defines a 'main' procedure that generates 10 unsigned integers in the range 0 to 4,294,967,294 and 10 signed integers in the range -50 to +49. The code uses 'call random32' and 'call randomrange' to generate the numbers, stores them in arrays, and prints them using 'call writestring' and 'call crlf'. The debug console on the right shows the output of the program, displaying the generated numbers and the exit message: 'C:\Users\acer\source\repos\Project2\Debug\Project2.exe (process 16016) exited with code 0. Press any key to close this window . . .'

```
7 arr2 sdword 10 dup(?)
8
9 .code
10 main PROC
11
12 mov ecx,10
13 mov esi,0
14
15 l1:
16 call random32
17 mov arr1[esi*Type arr1],eax
18 mov eax,100
19 call randomrange
20 sub eax,50
21 mov arr2[esi*Type arr2],eax
22 inc esi
23 loop l1
24
25 mov edx, offset first
26 call writestring
27 mov ecx, 10
28 mov esi,0
29 call crlf
30 l2:
31
```

Microsoft Visual Studio Debug Console

Generate 10 unsigned integers in the range 0 to 4,294,967,294:

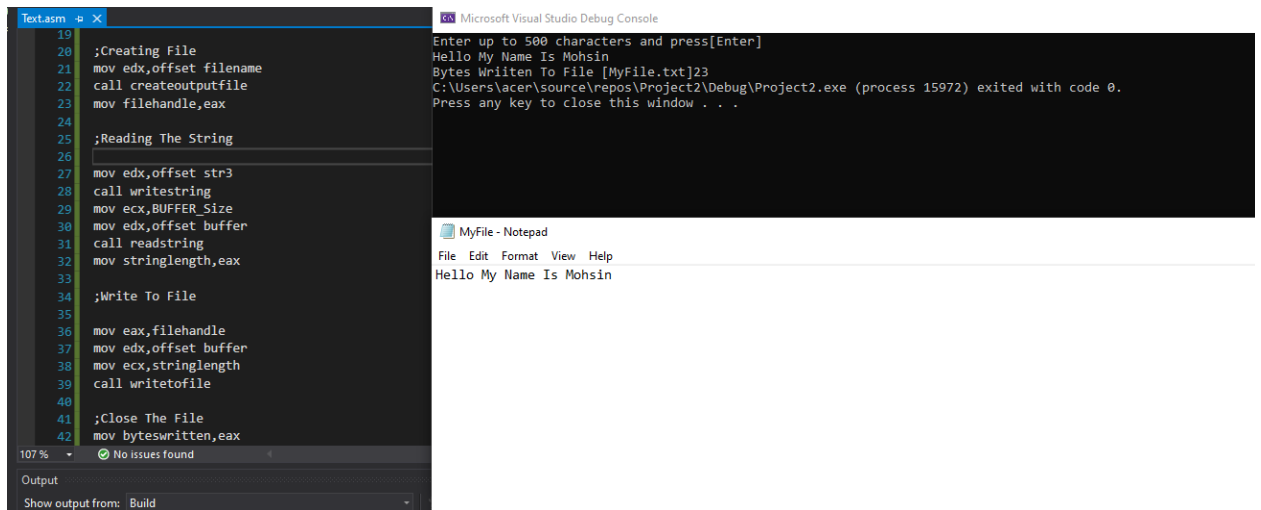
3221236194
974780167
2227888607
506254858
2288603673
1108756916
3436900788
2795956681
1342944021
1242749902

Generate 10 signed integers in the range -50 to +49:

-48
+7
-10
-2
+44
+27
-34
-13
+44
-43

C:\Users\acer\source\repos\Project2\Debug\Project2.exe (process 16016) exited with code 0.
Press any key to close this window . . .

Task 4: Code + Output



The screenshot shows the Visual Studio IDE with an assembly file named 'Text.asm' open. The code defines a 'main' procedure that creates a file named 'MyFile.txt', writes the string 'Hello My Name Is Mohsin' to it, and then reads the string back. The code uses 'call createoutputfile', 'call writestring', 'call readstring', and 'call writetofile' to perform these operations. The debug console on the right shows the output of the program, displaying the generated file name, the string written to the file, and the exit message: 'C:\Users\acer\source\repos\Project2\Debug\Project2.exe (process 15972) exited with code 0. Press any key to close this window . . .'

```
19 ;Creating File
20 mov edx,offset filename
21 call createoutputfile
22 mov filehandle,eax
23
24 ;Reading The String
25
26
27 mov edx,offset str3
28 call writestring
29 mov ecx,BUFFER_Size
30 mov edx,offset buffer
31 call readstring
32 mov stringlength,eax
33
34 ;Write To File
35
36 mov eax,filehandle
37 mov edx,offset buffer
38 mov ecx,stringlength
39 call writetofile
40
41 ;Close The File
42 mov byteswritten,eax
43
```

Microsoft Visual Studio Debug Console

Enter up to 500 characters and press[Enter]
Hello My Name Is Mohsin
Bytes Written To File [MyFile.txt]23
C:\Users\acer\source\repos\Project2\Debug\Project2.exe (process 15972) exited with code 0.
Press any key to close this window . . .

MyFile - Notepad

File Edit Format View Help
Hello My Name Is Mohsin