Lab 5: *Procedures & Filing*	**Session:** Fall 2019
Instructor(s): Sumaiyah Zahid & Fahim Ahmed	

CALL Instruction

The call instruction is used to call a procedure.

Procedures in Irvine32 Library

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

b. **Crlf**
Writes the end of line sequence to the console window.

c. **Delay (EAX)**
Pauses the program execution for a specified interval (in milliseconds).

d. **DumpRegs**
Displays the EAX, EBX, ECX, EDX, ESI, EDI, ESP, EIP and EFLAG registers.

e. **GetMaxXY (DX=col, AX=row)**
Gets the number of columns and rows in the console window buffer.

f. **GetTextColor (Background= Upper AL, Foreground= Lower AL)**
Returns the active foreground and background text colors in the console window.

g. **SetTextColor (EAX= Foreground + (Background*16))**
Sets the foreground and background colors of all subsequent text output to the console.

black = 0	red = 4	gray = 8	lightRed = 12
blue = 1	magenta = 5	lightBlue = 9	lightMagenta = 13
green = 2	brown = 6	lightGreen = 10	yellow = 14
cyan = 3	lightGray = 7	lightCyan = 11	white = 15

h. **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.

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

j. **ReadDec**
Reads an unsigned 32-bit integer from the keyboard.

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

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

m. **ReadString (EDX=OFFSET, ECX=SIZEOF)**
Reads a string from the keyboard, terminated by the enter key.

n. **WriteBin**

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

o. **WriteChar**
Writes a single character to the console window.

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

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

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

s. **WriteString (EDX= OFFSET String)**
Write a null-terminated string to the console window.

t. **Randomize**
Seeds the random number generator with a unique value.

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

v. **DumpMem (ESI=Starting OFFSET, ECX=LengthOf, EBX=Type)**
Writes the block of memory to the console window in hexadecimal.

Example 1:

WriteDec: The integer to be displayed is passed in EAX
WriteString: The offset of string to be written is passed in EDX
WriteChar: The character to be displayed is passed in AL

```
.data
        divider    BYTE " - ", 0
        codepage   DWORD 1252
.code
        mov  ecx, 255
        mov  eax,1
        mov  edx, OFFSET divider
        L1:
                call    WriteDec            ; EAX is a counter
                call    WriteString         ; EDX points to string
                call    WriteChar           ; AL is the character
                call    Crlf
                inc     al                  ; next character
        Loop L1
```

Example 2:

SetTextColor: Background & foreground colors are passed to EAX

```
.data
        str1 BYTE "Sample string in color", 0dh, 0ah, 0
.code
        mov     eax, yellow + (blue * 16)
        call    SetTextColor

        mov     edx, OFFSET str1
        call    WriteString

        call    DumpRegs
        exit
```

Example 3:

DumpMem: Pass offset of array in ESI, length of array in ECX & type in EBX
ReadInt: Reads the signed integer into EAX
WriteInt: Signed integer to be written is passed in EAX
WriteHex: Hex value to be written is passed in EAX
WriteBin: Binary value to be written is passed in EAX

```
.data
        COUNT = 4
        BlueTextOnGray = blue + (lightGray * 16)
        DefaultColor = lightGray + (black * 16)
        arrayD SDWORD 12345678h, 1A4B2000h, 3434h, 7AB9h
        prompt BYTE "Enter a 32-bit signed integer: ", 0
.code
; Set text color to blue text on a light gray background
        mov eax, BlueTextOnGray
        call  SetTextColor
        call  Clrscr                            ; clear the screen

; Display an array using DumpMem.

        mov     esi, OFFSET arrayD              ; starting OFFSET
        mov     ebx, TYPE arrayD                ; doubleword = 4 bytes
        mov     ecx, LENGTHOF arrayD            ; number of units in arrayD
        call    DumpMem                         ; display memory

    ; Ask the user to input a sequence of signed integers
        call    Crlf                            ; new line
        mov     ecx, COUNT

L1:
        mov     edx, OFFSET prompt
        call    WriteString
        call    ReadInt                 ; input integer into EAX
        call    Crlf                            ; new line

; Display the integer in decimal, hexadecimal, and binary
        call    WriteInt                        ; display in signed decimal
        call    Crlf
        call    WriteHex                        ; display in hexadecimal
        call    Crlf
        call    WriteBin                        ; display in binary
        call    Crlf
        call    Crlf
Loop  L1                                ; repeat the loop

; Return console window to default colors.
        call    WaitMsg                         ; "Press any key..."
        mov     eax, DefaultColor
        call    SetTextColor
        call    Clrscr
```

Example 4:

GetMSeconds: Value is returned in EAX

```
.data
        startTime DWORD ?
.code
```

```
call GetMseconds
mov startTime, eax
L1:
; (loop body)
loop L1
call GetMseconds
sub eax, startTime
```

Creating A New File

EAX contains the newly created file's handle or INVALID_HANDLE_VALUE if creation is unsuccessful

Example:

```
.data
        filehandle DWORD ?
        filename BYTE "MyFile.txt", 0
.code
        mov edx, offset filename
        call CreateOutputFile
        mov filehandle, eax
```

Opening An Existing File

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

Example:

```
.data
        filehandle DWORD ?
        filename BYTE "MyExistingFile.txt", 0
.code
        mov  edx,OFFSET filename
        call OpenInputFile
        mov  filehandle, 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

Example:

```
.data
        buffSize = 10                   ; if we want to read just 10 bytes
        buffer BYTE buffSize DUP(?)     ; buffer will contain the text read from the file

.code
        mov  eax, filehandle            ;assuming filehandle contains handle of an open file
        mov  edx, OFFSET buffer         ;buffer will contain the text read from the file
        mov  ecx, BUFSIZE               ;specify how many bytes to read
        call ReadFromFile
```

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:

```
.data
        bufferSize = 10                 ;if we want to write just 10 bytes
        buffer BYTE bufferSize DUP(?);uninitialized in this example but buffer will contain the text to be
written to file

.code
        mov  eax, filehandle            ; assuming that filehandle contains handle of an open file
        mov  edx, OFFSET buffer         ;buffer from where text will be written to file
        mov  ecx, bufferSize            ;number of bytes to be written to file from the buffer
        call WriteToFile
```

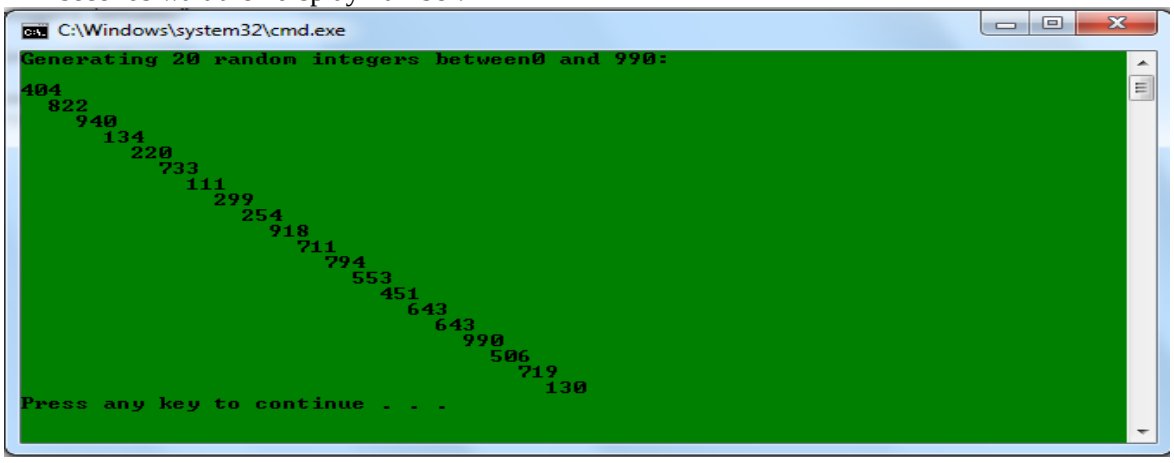
Closing A File

Example:

```
mov  eax, filehandle            ;assuming filehandle contains handle of an open file
call CloseFile
```

Activity:

Write a program to display random number list in diagonal pattern before each number display 5 milliseconds wait then display number.



Write a program to take input data for an employee and store it in appropriate variables. The program should ask for Employee ID, Name, Year of Birth & Annual Salary from the user. The program should then calculate the annual tax on that employee's annual salary if it exceeds Rs. 50,000 and display the tax message in a message box. The tax is calculated according to formula:
Tax = Monthly Salary / 2

Make a program to create a text file name Fibo.txt and write the first 8 fibonnaci numbers to that file.

Print the following pattern (using GotoXY and any other library procedure) without using the “Space" character.

```
    *
   **
  ***
 ****
*****
```