Lab Session 08

Built-In Procedures

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. **DumpMem (ESI=Starting OFFSET, ECX=LengthOf, EBX=Type)**
Writes the block of memory to the console window in hexadecimal.
f. **getDateTime**
Gets the current date and time from system
g. **GetMaxXY (DX=col, AX=row)**
Gets the number of columns and rows in the console window buffer.
h. **GetTextColor (Background= Upper AL, Foreground= Lower AL)**
Returns the active foreground and background text colors in the console window.
i. **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.
j. **MsgBox (EDX=OFFSET String, EBX= OFFSET Title)**
Displays a pop-up message box.
k. **MsgBoxAsk (EDX=OFFSET String, EBX= OFFSET Title)**
Displays a yes/no question in a pop-up message box.
(EAX=6 YES, EAX=7 NO)
l. **ReadChar**
Waits for single character to be typed at the keyboard and returns that character.
m. **ReadDec**
Reads an unsigned 32-bit integer from the keyboard.
n. **ReadHex**
Reads a 32-bit hexadecimal integers from the keyboard, terminated by the enter key.

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

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

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

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

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

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

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

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

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

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

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

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

EXAMPLE # 01:

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 # 02:

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 # 03:

MsgBox: Offset of message to be displayed inside the pop-up is passed in EDX. Offset of caption (optional) is passed in EBX.

```
.data
        caption "Dialog Title", 0
        HelloMsg BYTE "This is a pop-up message box.", 0dh,0ah
                 BYTE "Click OK to continue...", 0
.code
        mov     ebx, 0                          ; no caption
        mov     edx, OFFSET  HelloMsg           ; contents
        call    MsgBox

        mov     ebx, OFFSET  caption            ; caption
        mov     edx, OFFSET  HelloMsg           ; contents
        call    MsgBox
```

EXAMPLE # 04:

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

a)

```
.data
var1 DWORD ?
var2 DWORD ?
arrayD word 1,2,3,4

prompt BYTE "Enter 1st a 32-bit signed integer: ", 0
prompt1 BYTE "Enter 2nd a 32-bit signed integer: ", 0

.code
main PROC
mov     edx, OFFSET prompt
call    WriteString
call readint
mov var1, eax

mov     edx, OFFSET prompt1
call    WriteString
call readint
mov var2, eax

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
call crlf

mov eax, 0

; Addition
mov eax, var1
add eax, var2
call writeint
call crlf
call writehex
```

```
call crlf
call writebin
call crlf

; Subtraction
mov eax, var1
sub eax, var2
call writeint
call crlf
call writehex
call crlf
call writebin
call crlf

; Multiplication
mov eax, var1
imul eax, var2
call writeint
call crlf
call writehex
call crlf
call writebin
call crlf
```

b)

```
.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 # 05:

MsgBoxAsk: Offset of question string is passed in EDX. Offset of caption is passed in EBX. Selected value is returned in EAX (IDYES equal to 6 or IDNO equal to 7)

```
.data
caption BYTE "Survey Completed",0
question BYTE "Thank you for completing the survey."
                BYTE 0dh, 0ah
                BYTE "Would you like to receive the results?", 0
.code
main proc

                mov     edx, OFFSET question
                call    MsgBoxAsk
                mov     edx, OFFSET caption
                call    WriteString
                ;(check return value in EAX)
```

EXAMPLE # 06:

GetMSeconds: Value is returned in EAX

```
.data
        startTime DWORD ?
.code
main proc
        call GetMseconds
        mov startTime, eax
        call writeint
        call crlf
        L1:
        ; (loop body)
        loop L1
        call GetMseconds
        sub eax, startTime
        call writeint
        call crlf
```

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

EXAMPLE:

```
.data
filehandle DWORD ?
filename BYTE "MyFile.txt", 0
buffSize = 10                              ; if we want to read just 10 bytes
buffer BYTE buffSize DUP(?)          ; buffer will contain the text read from the file

.code
main proc
        mov  edx,OFFSET filename
        call OpenInputFile
        mov  filehandle, EAX
        mov  edx, OFFSET buffer     ;buffer will contain the text read from the file
        mov  ecx, buffSize                ;specify how many bytes to read
        call ReadFromFile
        mov  edx, OFFSET buffer
        call writestring
        Call crlf
```

Writing To A File:

Takes 3 parameters
- eax == the file handle to write to
-edx == pointer to buffer that contains data to write to file
-ecx == length of data to write to file.

Call arguments:
EAX = an open file handle
EDX = offset of the buffer
ECX = maximum number of bytes to write

EXAMPLE:

```
.data
filehandle DWORD ?
filename BYTE "MyFile14.txt", 0
buffSize = 15              ; if we want to read just 10 bytes
buffer BYTE buffSize DUP(?)          ; buffer will contain the text read from the file
buffer1 BYTE "fizzaaqeel",0

.code

main proc
mov edx, offset filename
call CreateOutputFile
mov filehandle, eax

mov  eax, filehandle          ; assuming that filehandle contains handle of an open file
mov  edx, OFFSET buffer1      ;buffer from where text will be written to file
mov  ecx, buffSize            ;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
```