**Carson Blevins**

**Bellarmine University**

**Due Date: 11/6/15**

**Date Submitted: 11/19/15**

**CS 305 C Assembly Language Fall 2015**

**Due Date:** 11/9/15

**Exercise #1:**

**Source Code of Exercise #5 of Chapter 6:**

TITLE Boolean Calculator 1 (BooleanCalc1.asm)

; This functions as a simple boolean calculator for 32-bit integers.

INCLUDE Irvine32.inc

.data

CaseTable BYTE '1' ;lookup value

DWORD x\_AND\_y ;address of procedure

EntrySize = ($-CaseTable)

BYTE '2'

DWORD x\_OR\_y

BYTE '3'

DWORD NOT\_x

BYTE '4'

DWORD x\_XOR\_y

BYTE '5'

DWORD Exit\_Program

NumberOfEntries = ($-CaseTable)/EntrySize

prompt BYTE "Press 1. x AND y, 2. x OR y, 3. NOT x, 4. x XOR y, 5. Exit ",0

msg1 BYTE "x AND y ",0

msg2 BYTE "x OR y ",0

msg3 BYTE "NOT x ",0

msg4 BYTE "x XOR y ",0

msg5 BYTE "Exit Program ",0

promptx BYTE "Enter an x in Hexadecimal: ",0

prompty BYTE "Enter an y in Hexadecimal: ",0

.code

main PROC

mov edx,OFFSET prompt ;ask user for input

call WriteString

call ReadChar ;read character into AL

mov ebx,OFFSET CaseTable ;point EBX to the table

mov ecx,NumberOfEntries ;loop counter

L1:

cmp al,[ebx] ;match found?

jne L2 ;no: continue

call NEAR PTR [ebx+1] ;yes: call the procedure

;call WriteString ;display message

call Crlf

jmp L3 ;exit the search

L2:

add ebx,EntrySize ;point to the next entry

loop L1 ;repeat until ECX = 0

L3:

exit

main ENDP

x\_AND\_y PROC

call Crlf

mov edx,OFFSET msg1

call WriteString

call Crlf

mov edx,OFFSET promptx ;ask user for input

call WriteString

call ReadHex ;read decimal into EAX

mov EBX,EAX

call Crlf

mov edx,OFFSET prompty ;ask user for input

call WriteString

call ReadHex ;read decimal into EAX

AND EAX,EBX

call WriteHexB

ret

x\_AND\_y ENDP

x\_OR\_y PROC

call Crlf

mov edx,OFFSET msg2

call WriteString

call Crlf

mov edx,OFFSET promptx ;ask user for input

call WriteString

call ReadHex ;read decimal into EAX

mov EBX,EAX

call Crlf

mov edx,OFFSET prompty ;ask user for input

call WriteString

call ReadHex ;read decimal into EAX

OR EAX,EBX

call WriteHexB

ret

x\_OR\_y ENDP

NOT\_x PROC

call Crlf

mov edx,OFFSET msg3

call WriteString

call Crlf

mov edx,OFFSET promptx ;ask user for input

call WriteString

call ReadHex ;read decimal into EAX

NOT eax

call WriteHexB

ret

NOT\_x ENDP

x\_XOR\_y PROC

call Crlf

mov edx,OFFSET msg4

call WriteString

call Crlf

mov edx,OFFSET promptx ;ask user for input

call WriteString

call ReadHex ;read decimal into EAX

mov EBX,EAX

call Crlf

mov edx,OFFSET prompty ;ask user for input

call WriteString

call ReadHex ;read decimal into EAX

XOR EAX,EBX

call WriteHexB

ret

x\_XOR\_y ENDP

Exit\_Program PROC

call Crlf

mov edx,OFFSET msg5

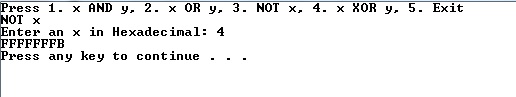
call WriteString

ret

Exit\_Program ENDP

END main

**Here’s the output:**

****

**Exercise #2:**

**Source Code of Exercise #6 of Chapter 6:**

TITLE Boolean Calculator 2 (BooleanCalc2.asm)

; This functions as a simple boolean calculator for 32-bit integers.

INCLUDE Irvine32.inc

.data

CaseTable BYTE '1' ;lookup value

DWORD x\_AND\_y ;address of procedure

EntrySize = ($-CaseTable)

BYTE '2'

DWORD x\_OR\_y

BYTE '3'

DWORD NOT\_x

BYTE '4'

DWORD x\_XOR\_y

BYTE '5'

DWORD Exit\_Program

NumberOfEntries = ($-CaseTable)/EntrySize

prompt BYTE "Press 1. x AND y, 2. x OR y, 3. NOT x, 4. x XOR y, 5. Exit ",0

msg1 BYTE "x AND y ",0

msg2 BYTE "x OR y ",0

msg3 BYTE "NOT x ",0

msg4 BYTE "x XOR y ",0

msg5 BYTE "Exit Program ",0

promptx BYTE "Enter an x in Hexadecimal: ",0

prompty BYTE "Enter an y in Hexadecimal: ",0

.code

main PROC

mov edx,OFFSET prompt ;ask user for input

call WriteString

call ReadChar ;read character into AL

mov ebx,OFFSET CaseTable ;point EBX to the table

mov ecx,NumberOfEntries ;loop counter

L1:

cmp al,[ebx] ;match found?

jne L2 ;no: continue

call NEAR PTR [ebx+1] ;yes: call the procedure

;call WriteString ;display message

call Crlf

jmp L3 ;exit the search

L2:

add ebx,EntrySize ;point to the next entry

loop L1 ;repeat until ECX = 0

L3:

exit

main ENDP

x\_AND\_y PROC

call Crlf

mov edx,OFFSET msg1

call WriteString

call Crlf

mov edx,OFFSET promptx ;ask user for input

call WriteString

call ReadHex ;read decimal into EAX

mov EBX,EAX

call Crlf

mov edx,OFFSET prompty ;ask user for input

call WriteString

call ReadHex ;read decimal into EAX

AND EAX,EBX

call WriteHexB

ret

x\_AND\_y ENDP

x\_OR\_y PROC

call Crlf

mov edx,OFFSET msg2

call WriteString

call Crlf

mov edx,OFFSET promptx ;ask user for input

call WriteString

call ReadHex ;read decimal into EAX

mov EBX,EAX

call Crlf

mov edx,OFFSET prompty ;ask user for input

call WriteString

call ReadHex ;read decimal into EAX

OR EAX,EBX

call WriteHexB

ret

x\_OR\_y ENDP

NOT\_x PROC

call Crlf

mov edx,OFFSET msg3

call WriteString

call Crlf

mov edx,OFFSET promptx ;ask user for input

call WriteString

call ReadHex ;read decimal into EAX

NOT eax

call WriteHexB

ret

NOT\_x ENDP

x\_XOR\_y PROC

call Crlf

mov edx,OFFSET msg4

call WriteString

call Crlf

mov edx,OFFSET promptx ;ask user for input

call WriteString

call ReadHex ;read decimal into EAX

mov EBX,EAX

call Crlf

mov edx,OFFSET prompty ;ask user for input

call WriteString

call ReadHex ;read decimal into EAX

XOR EAX,EBX

call WriteHexB

ret

x\_XOR\_y ENDP

Exit\_Program PROC

call Crlf

mov edx,OFFSET msg5

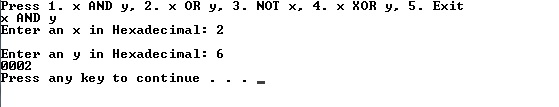
call WriteString

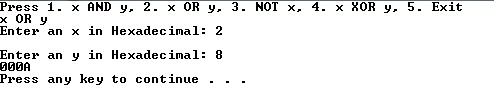
ret

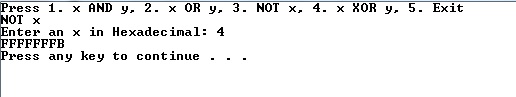
Exit\_Program ENDP

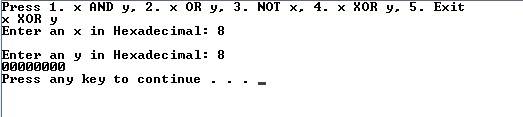
END main

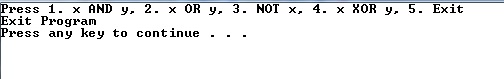
**Here’s the output:**

****

****

****

****

****

**Exercise #2:**

**Source Code of Exercise #8 of Chapter 6:**

; Encryption Program (Encrypt.asm)

INCLUDE Irvine32.inc

KEY = 236 ; any value between 1-255

BUFMAX = 128 ; maximum buffer size

.data

key1 BYTE "This\_is\_an\_Encrypted\_Message."

sPrompt BYTE "Enter the plain text:",0

sEncrypt BYTE "Cipher text: ",0

sDecrypt BYTE "Decrypted: ",0

buffer BYTE BUFMAX+1 DUP(0)

bufSize DWORD ?

.code

main PROC

call InputTheString ; input the plain text

call TranslateBuffer ; encrypt the buffer

mov edx,OFFSET sEncrypt ; display encrypted message

call DisplayMessage

call TranslateBuffer ; decrypt the buffer

mov edx,OFFSET sDecrypt ; display decrypted message

call DisplayMessage

exit

main ENDP

;-----------------------------------------------------

InputTheString PROC

;

; Prompts user for a plaintext string. Saves the string

; and its length.

; Receives: nothing

; Returns: nothing

;-----------------------------------------------------

pushad ; save 32-bit registers

mov edx,OFFSET sPrompt ; display a prompt

call WriteString

mov ecx,BUFMAX ; maximum character count

mov edx,OFFSET buffer ; point to the buffer

call ReadString ; input the string

mov bufSize,eax ; save the length

call Crlf

popad

ret

InputTheString ENDP

;-----------------------------------------------------

DisplayMessage PROC

;

; Displays the encrypted or decrypted message.

; Receives: EDX points to the message

; Returns: nothing

;-----------------------------------------------------

pushad

call WriteString

mov edx,OFFSET buffer ; display the buffer

call WriteString

call Crlf

call Crlf

popad

ret

DisplayMessage ENDP

;-----------------------------------------------------

TranslateBuffer PROC

;

; Translates the string by exclusive-ORing each

; byte with the encryption key byte.

; Receives: nothing

; Returns: nothing

;-----------------------------------------------------

pushad

mov ecx,bufSize ; loop counter

mov esi,0 ; index 0 in buffer

L1:

mov bl,key1[esi]

xor buffer[esi],bl ; translate a byte

inc esi ; point to next byte

loop L1

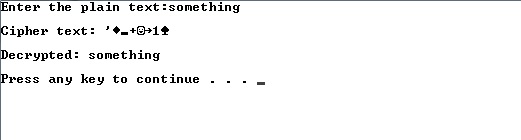
popad

ret

TranslateBuffer ENDP

END main

**Here’s the output:**

****