UTM

UNIVERSITI TEKNOLOGI MALAYSIA

FACULTY OF ENGINEERING

SCHOOL OF COMPUTING

SEMESTER 2/20212022

SECR2033/07 - COMPUTER ORGANIZATION AND ARCHITECTURE

PROJECT

BOOLEAN CALCULATOR

LECTURERS

Dr Aida Ali

GROUP : E

VIDEO LINK :

https://drive.google.com/drive/folders/1-hOhLhj70zJ4cmSECLCjgor3Wgpo7jrt?usp=sharing

Name	Matric ID
MUHAMMAD NAJWAN HAZIM BIN KHAIRI	A21EC0087
MUHAMMAD WAFFI QAYYUM BIN DIN	A21EC0097
MUHAMMAD IRHAM HAKIM BIN ROSLAN	A21EC0081
MUHAMMAD KAMIL EIZAZ BIN OTHMAN	A21EC0084
MUHAMMAD THORIQ BIN KAHAIRI	A21EC0096

MEMBER RESPONSIBILITIES

Name	Responsibilities
NAJWAN HAZIM	1. Prepare the Visual Studio program template 2. Incharge of debugging and compiling the program 3. Responsible for the Coding and Explanation topic in the report template 4. Incharge of finalize the report template
WAFFI QAYYUM	1. Incharge of AND operation in the program 2. Re-adjust the program to produce the same output as the provided examples 3. Prepare the report template 4. Responsible for the Discussion & Conclusion topic in the report template
IRHAM HAKIM	1. Incharge of OR operation in the program 2. Responsible for the Coding and Explanation topic in the report template 3. Incharge of formatting and correcting mistakes in the report template 4. Incharge of finalize the report template
KAMIL EIZAZ	1. Incharge of NOT operation in the program 2. Responsible for the Example of Inputs & Outputs topic in the report template 3. Incharge of compiling the video presentation of each member 4. Incharge of finalize the coding
THORIQ	1. Incharge of XOR operation in the program 2. Incharge of debugging and compiling the program 3. Responsible for the Discussion & Conclusion topic in the report template 4. Incharge of finalize the coding

TABLE OF CONTENTS

CODING AND EXPLANATION

```
13      INCLUDE Irvine32.inc
14
15
16      .data
17          menuChoice DWORD ?
18          hexaNum1 DWORD ?
19          hexaNum2 DWORD ?
20          str1 BYTE "Boolean AND ", 0
21          str2  BYTE "Boolean OR ", 0
22          str3 BYTE "Boolean NOT ", 0
23          str4 BYTE "Boolean XOR ", 0
24          str5 BYTE "--- Boolean Calculator ----------", 0
25          str6 BYTE "1. x AND y", 0
26          str7 BYTE "2. x OR y", 0
27          str8 BYTE "3. x NOT y", 0
28          str9 BYTE "4. x XOR y", 0
29          str10 BYTE "5. Exit Program", 0
30          str11 BYTE "--------------------------------", 0
31          str12 BYTE "Enter your choice: ", 0
32          str13 BYTE "Bye." ,0
33          value BYTE "The 32-bit hexadecimal result is ", 0
34          errorStr BYTE "Invalid number, please enter again. ", 0
35          hexaStr1 BYTE "Input the first 32-bit hexadecimal operand: ", 0
36          hexaStr2 BYTE "Input the second 32-bit hexadecimal operand: ", 0
```

Diagram 1

The Code seen above is primarily our prompt message to the user; simply, all of the variables are strings that tell the user which option to select from a range of (1-5) and the type of choice they have made, such as "Boolean AND" and "Boolean OR." The final portion of the code asks the user to enter a number in 32-bit hexadecimal format and also displays the result string, informing the user of the outcome of the action they selected. Also, there is a variable with DWORD as the data type for restoring value from the register.

```
41      Menu:
42          call crlf                       ;newline
43          mov edx, offset str5            ;print out str5
44          call writestring
45          call Crlf
46          mov edx, offset str6            ; print out str6
47          call writestring
48          call Crlf
49          mov edx, offset str7            ; print out str7
50          call writestring
51          call Crlf
52          mov edx, offset str8            ; print out str8
53          call writestring
54          call Crlf
55          mov edx, offset str9            ; print out str9
56          call writestring
57          call Crlf
58          mov edx, offset str10           ; print out str10
59          call writestring
60          call Crlf
61          mov edx, offset str11           ; print out str11
62          call writestring
63          call Crlf
64
65          mov edx, offset str12           ; print out str12
66          call writestring
67          call ReadDec                    ; get input from user
68          mov menuChoice, eax
69          call Crlf
70          jmp L1
```

Diagram 2

In this main proc, we provide a menu for the user to choose which operator the user wanted to do. All call WriteStrings are to call the string that we have provided in the data. After the user inputs his/her choice then it will jump to L1 which is for comparing the data.

```
72      L1:
73          cmp menuChoice, 1
74          je AndInput                 ; jump if menuChoice = 1
75          cmp menuChoice, 2
76          je OrInput                  ; jump if menuChoice = 2
77          cmp menuChoice, 3
78          je NotInput                 ; jump if menuChoice = 3
79          cmp menuChoice, 4
80          je XorInput                 ; jump if menuChoice = 4
81          cmp menuChoice, 5
82          je ExitOutput               ; jump if menuChoice = 5
83          cmp menuChoice, 5
84          ja ErrorOutput              ; jump if menuChoice above than 5
85          cmp menuChoice, 1
86          jb ErrorOutput              ; jump if menuChoice below than 1
```

Diagram 3

In L2, there is a bunch of compare commands for comparing the data. All options are provided in L2 which is if the user inputs number 1-5, it will go to the option that he/she has chosen but if the user exceeds 5 or below 1, it will jump to ErrorOutput which is it will display about the error message and ask the user to re-enter the number.

```
88      AndInput:
89          call Clrscr
90          mov edx, OFFSET str1            ; print out str1
91          call WriteString
92          call crlf
93          call opAnd
94
95      OrInput:
96          call Clrscr
97          mov edx, OFFSET str2            ; print out str2
98          call WriteString
99          call crlf
100         call opOr
101
102
103     NotInput:
104         call Clrscr
105         mov edx, OFFSET str3            ; print out str3
106         call WriteString
107         call crlf
108         call opNot
109
110
111     XorInput:
112         call Clrscr
113         mov edx, OFFSET str4            ; print out str4
114         call WriteString
115         call crlf
116         call opXor
117
```

Diagram 4

```
118
119     ExitOutput:
120
121         mov edx, offset str13           ; print out str13
122         call writestring
123         call crlf
124         call crlf
125         call WaitMsg                    ; system pause
126         exit
127
128
129     ErrorOutput:
130         call Crlf
131         mov edx, OFFSET errorStr        ; print out errorStr
132         call WriteString
133         call crlf
134         mov edx, offset str12           ; print out str12
135         call writestring
136         call ReadDec                    ; get input from user
137         mov menuChoice, eax
138         call Crlf
139         jmp L1
```

Diagram 5

In Diagram 4, shows that it will print messages according to the user's input. If the user enters number 1 then it will print out "Boolean and" and the same for other operations. Then, it will call proc for the calculation part. Each operation has its own calculation part. Diagram 5 shows for exit output and the error output. If the user chooses option 5, then the user will exit the program. Then, if the user puts the wrong number, then the program will ask the user to re-enter the number again until the number option is correct. If the user enters number 5 in the option, it will go to ExitOutput to print out all about exit.

```
143     ; ----------------------------------------
144     ; AND OPERATION
145
146     opAnd PROC
147
148     mov eax, 0
149     mov edx, OFFSET hexaStr1            ; print out hexaStr1
150     call WriteString
151     call ReadHex                        ; input hexa number from user
152     mov  hexaNum1, eax
153     mov edx, OFFSET hexaStr2            ; print out hexaStr2
154     call WriteString
155     call ReadHex                        ; input hexa number from user
156     mov hexaNum2, eax
157
158     mov eax, hexaNum1                   ; calculation for AND operation
159     AND eax, hexaNum2
160
161     mov edx, offset value               ; print out the result
162     call WriteString
163     call WriteHex
164     call crlf
165     call main
166
167     ret                                 ; return from procedure
168
169     opAnd ENDP
170
```

Diagram 6

First, in this opAnd PROC, we shift 0 to the eax register. Then, using the WriteString function, we print hexaStr1 and hexaStr2. The user will then be prompted to enter two digits by the application. It will read the two numbers in hexadecimal format. The application will show the result AND operation in hexadecimal once two numbers are entered. We utilise the "ReadHex" library procedure to read numbers encoded in hexadecimal. An end-of-line sequence is written to standard output using the Crlf method.

```
171     ; ---------------------------------------
172     ; OR OPERATION
173
174     opOr PROC
175
176     mov edx, OFFSET hexaStr1            ; print out hexaStr1
177     call WriteString
178     call ReadHex                        ; input hexa number from user
179     mov hexaNum1, eax
180     mov edx, OFFSET hexaStr2            ; print out hexaStr2
181     call WriteString
182     call ReadHex                        ; input hexa number from user
183     mov hexaNum2, eax
184
185     mov eax, hexaNum1                   ; calculation for OR operation
186     OR eax, hexaNum2
187
188
189     mov edx, offset value               ; print out the result
190     call WriteString
191     call WriteHex
192     call crlf
193     call main
194
195     ret                                 ; return from procedure
196
197     opOr ENDP
198
```

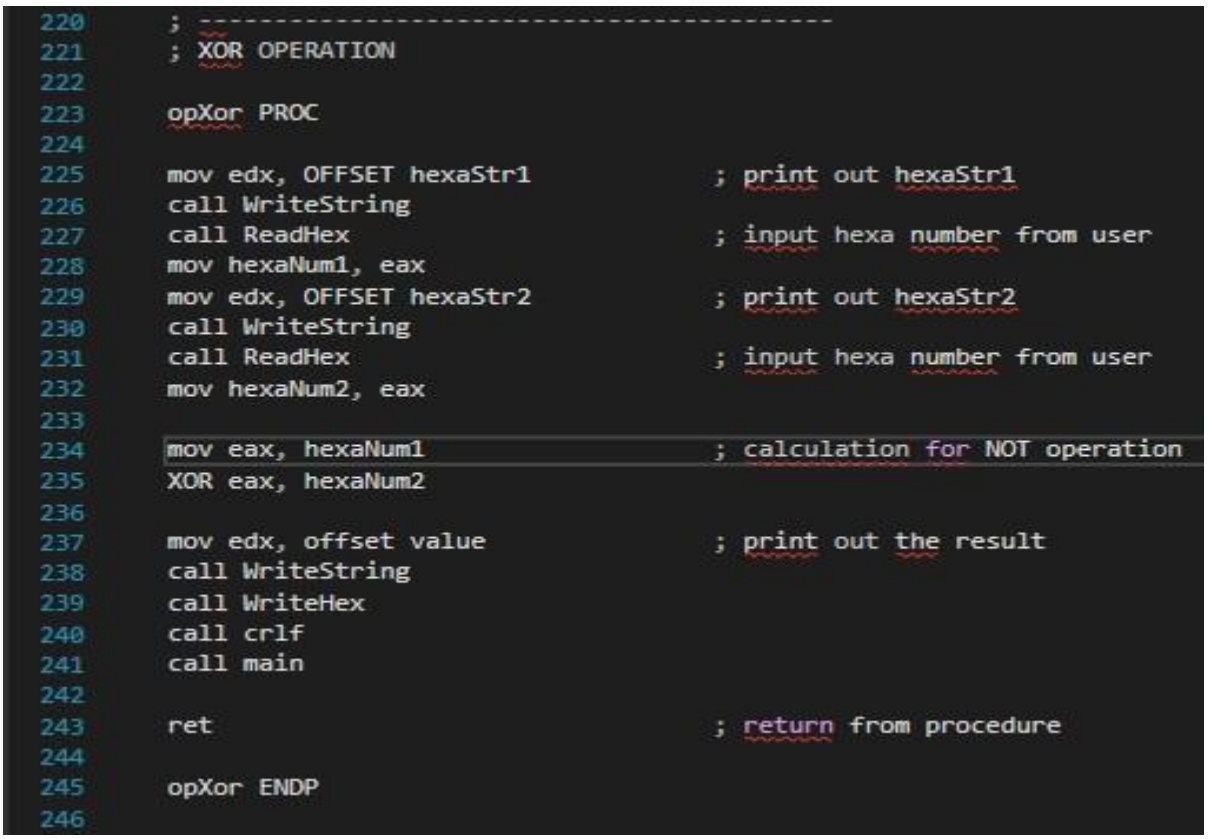
Diagram 7

Basically, all step in Diagram 6 are similar with diagram 7. The difference is we are using OR in the coding to calculate OR operation. Then, it will return to the main proc.

```
199     ; ---------------------------------------
200     ; NOT OPERATION
201
202     opNot PROC
203
204     mov edx, OFFSET hexaStr1            ; print out hexaStr1
205     call WriteString
206     call ReadHex                        ; input hexa number from user
207
208     NOT eax                             ; calculation for NOT operation
209
210     mov edx, offset value               ; print out the result
211     call WriteString
212     call WriteHex; 0000FFF2
213     call crlf
214     call main
215
216     ret                                 ; return from procedure
217
218     opNot ENDP
219
```

Diagram 8

Basically, all step in Diagram 6 are similar with diagram 8. The difference is we are using NOT in the coding to calculate NOT operation and we only need one input hexanumber from the user. Then, it will return to the main proc.



```
220     ; ---------------------------------------
221     ; XOR OPERATION
222
223     opXor PROC
224
225     mov edx, OFFSET hexaStr1            ; print out hexaStr1
226     call WriteString
227     call ReadHex                        ; input hexa number from user
228     mov hexaNum1, eax
229     mov edx, OFFSET hexaStr2            ; print out hexaStr2
230     call WriteString
231     call ReadHex                        ; input hexa number from user
232     mov hexaNum2, eax
233
234     mov eax, hexaNum1                   ; calculation for NOT operation
235     XOR eax, hexaNum2
236
237     mov edx, offset value               ; print out the result
238     call WriteString
239     call WriteHex
240     call crlf
241     call main
242
243     ret                                 ; return from procedure
244
245     opXor ENDP
246
```

Diagram 9

Same with digram 6, and 7, all step are similar with diagram 9. The only difference in opXor proc is we are using XOR in the coding to calculate XOR operation. Then, it will return to the main proc.

EXAMPLE OF INPUTS AND OUTPUTS

- Output when the user enters an incorrect/invalid number of options given

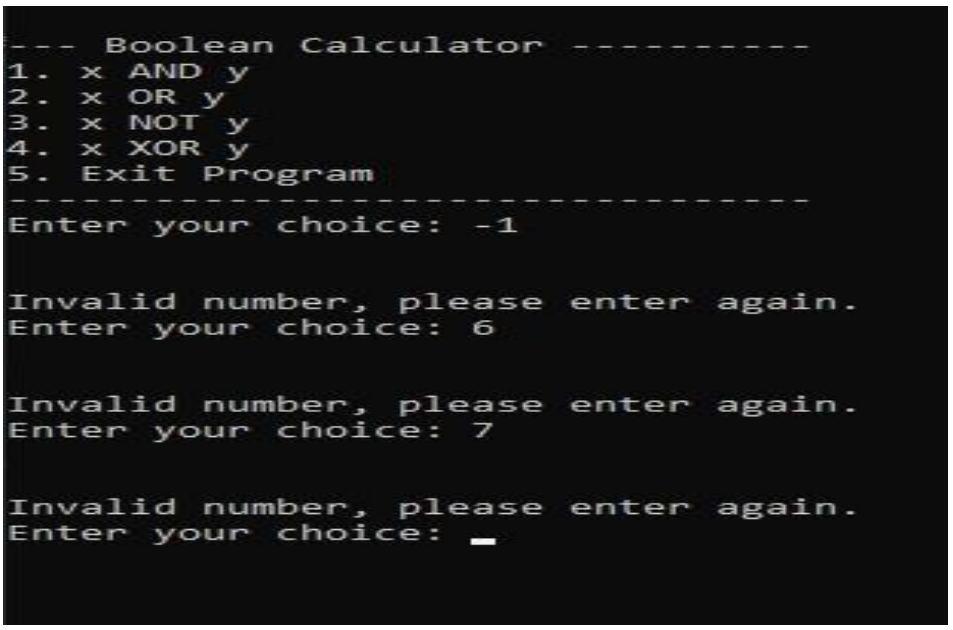


Diagram 10

The message "Invalid number, please enter again." will appear if the user inputs a number that is lower than 1 or higher than 5. Therefore, the user must provide a value between 1 and 5, or else the application will not perform the task.

- Output for AND operation when the user enters value 1

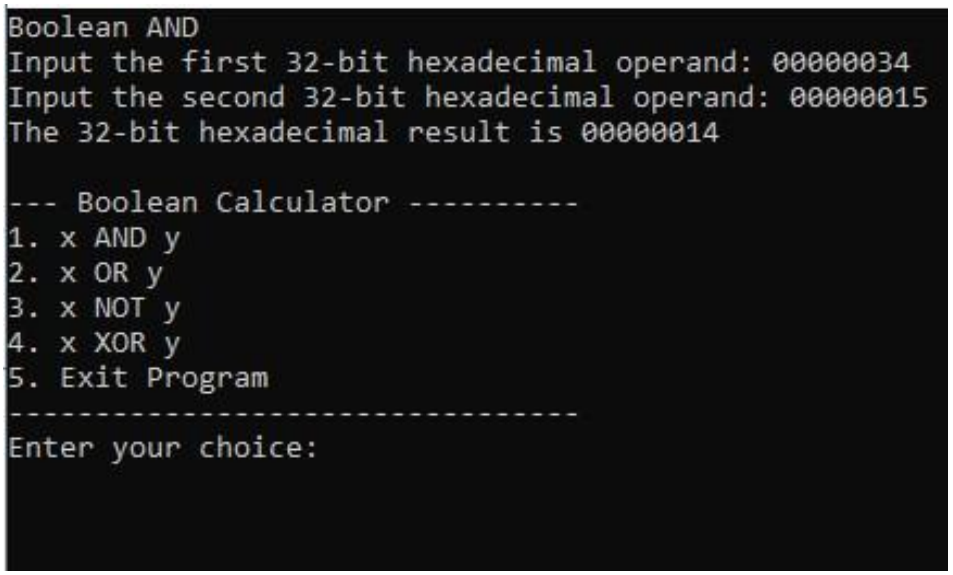


Diagram 11

X	Y	ANSWER
0	0	0
0	1	0
1	0	0
1	1	1

The program will perform an AND operation after the user selects option number 1. The user's initial input is 34h. It would be 00110100b in binary. The second input is 15h, or 00010101b in binary. These two numbers provide the result 00010100b when we use the AND operator. 00010100b would be 14h in hexadecimal conversion. The application will then return to the main menu so the user can select another choice or do the same action.

- Output for OR operation when the user enters value 2

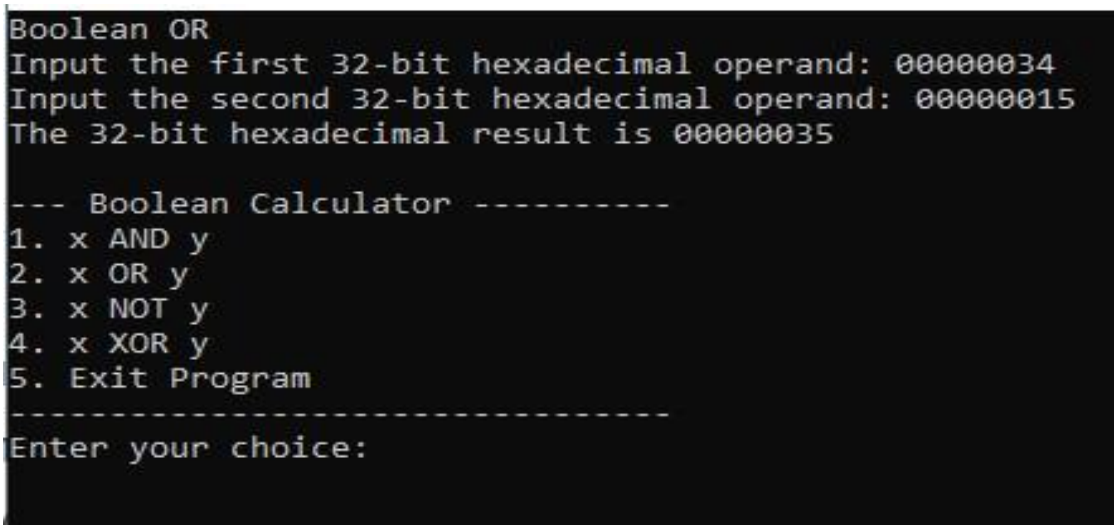


Diagram 12

X	Y	ANSWER
0	0	0
0	1	1
1	0	1
1	1	1

Following the user's selection of choice number 2, the programme will do an OR operation. 34h is the user's initial input. In binary, it would be 00110100b. The second input is 15h, or in binary, 00010101b. When we use the OR operator to these two numbers, we get the value 00110101b. If 00110101b were converted to hexadecimal, it would be 35h. When the user wants to choose another option or take the same action, the application will then return to the main menu.

- Output for NOT when the user enters value 3

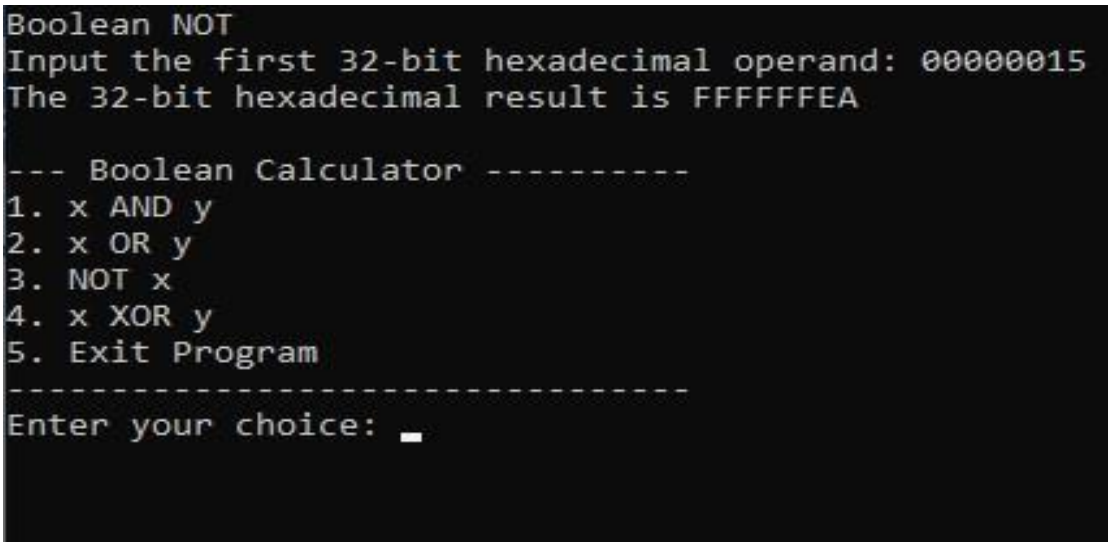


Diagram 13

X	ANSWER
0	1
1	0

When the user chooses option number 3, the application will do a NOT operator. The user's initial input is 15h. It would be 00010101b in binary. The value 11101010b is what we obtain when we apply the NOT operator. Hexadecimal representation of 11101010b is FFFFFFEAh. The application will then return to the main menu when the user wishes to select another choice or perform the same activity.

- Output for XOR when the user enters value 4

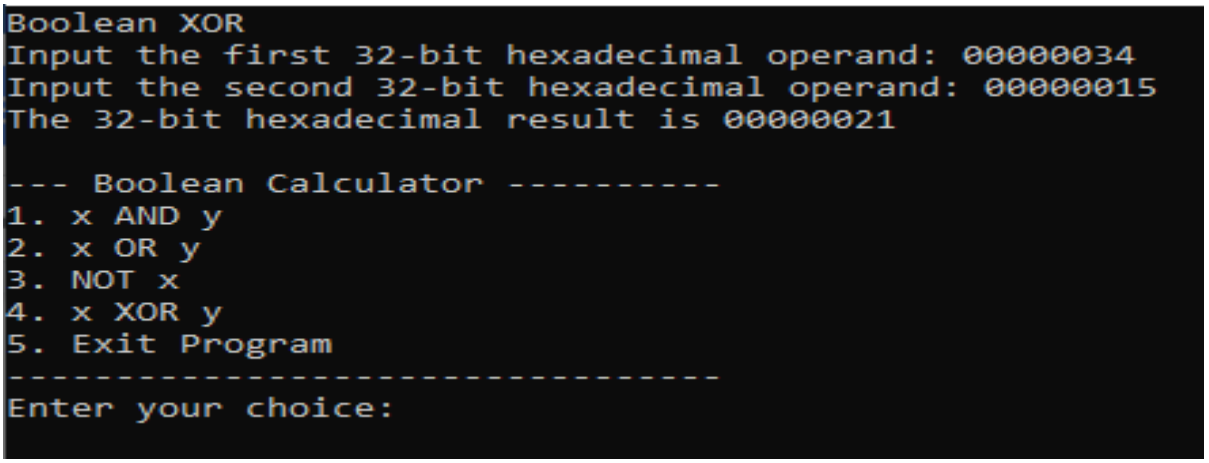


Diagram 14

X	Y	ANSWER
0	0	0
0	1	1
1	0	1
1	1	0

After the user selects option number 4, the application will do an XOR operation. The user's initial input is 34h. It would be 00110100b in binary. The second input is 15h, or 00010101b in binary. The result of applying the XOR operator to these two values is 00100001b. Hexadecimal representation of 00100001b would be 21h. The application will then return to the main menu when the user wishes to select another choice or perform the same activity.

- Output for Exit program when the user enters value 5

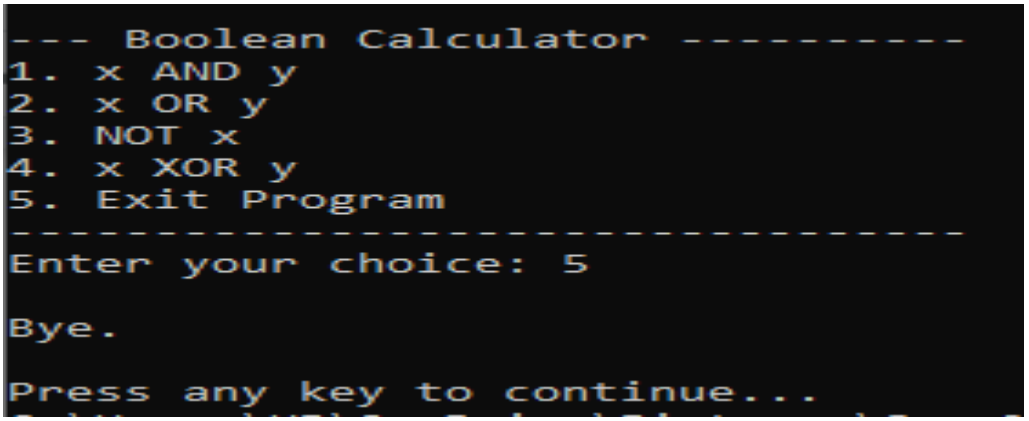


Diagram 15

DISCUSSION AND CONCLUSION

Assembly language source files must be a part of a project, which is similar to a box container, for Visual Studio to accept them. A project stores configuration data such the locations of the linker, assembler, and necessary libraries. A project contains a folder where all of its files are listed by name and location. This project makes use of the project template in the .asm format that our instructor provided at the outset of teaching us assembly language. We must take advantage of Irvine Link Library. A number of helpful routines to input data, output data, and carry out various activities that typically require numerous operating system calls are available in the Irvine link library. Here is a list of some of the Irvine32 and Irvine16 library procedures we used to create this project:

Clrscr	Clears the screen, moves the cursor to the upper-left corner.
Crlf	Writes a carriage return / linefeed to the display.
Mov	Copies a byte or word from a source operand to a destination operand.
Writestring	Write a null-terminated string. Input: EDX points to the strings offset.
ReadHex	Reads a 32-bit Hexadecimal integer from the keyboard.
WriteHex	Writes a 32-bit Hexadecimal integer to the console window in Hexadecimal format.
Jmp	(Jump Unconditionally) Jump to a code label.
JE	Jump if equal.
JA	Jump if left operand bigger than right operand.
JB	Jump if left operand smaller than right operand.
AND	Performs a Boolean AND operation between two operands
OR	Performs a Boolean OR operation between two operands
NOT	Performs a Boolean NOT operation between two operands
XOR	Performs a Boolean XOR operation between two operands

The coding we created has the flaw of being both too basic and long. We are continuously learning even though we are in the first year of our software course degree programme. Over time and with practice, it is possible to learn far more effective techniques or shortcuts for much simpler code. Nevertheless, this code still follows the project's instructions and makes use of the assembly language that we have studied in class.

In conclusion, we have applied the knowledge that we learned from this course Computer Organization and Architecture to design a simple Boolean Calculator. We have a fundamental understanding of computer programming language and the idea of assembly programming. As a programmer, it is crucial to possess a solid foundation in assembly programming. We shouldn't undervalue the significance of assembly language because computer programming languages are frequently used for creating software programmes and apps for us who will be the future Software Engineer.

REFERENCES

- Chris. (2017). *Boolean calculator - x86 assembly language*. Gist. Retrieved June 30, 2022, from https://gist.github.com/killuhwhale/94be4126b1f9d1abf3c32778f8b3cd55

- Code, H. (2022). How to convert a boolean expression into assembly code. Retrieved 30 June 2022, from https://stackoverflow.com/questions/61277688/how-to-convert-a-boolean-expression-into-assembly-code

APPENDIX

Full coding:

```
TITLE Project COA(main.asm)
; Boolean Calculator
; ---------------------------------------------------------------------------------------------------------------
; Section: 06
; Group: E
; Group member : 1) MUHAMMAD NAJWAN HAZIM BIN KHAIRI(A21EC0087)
;                2) MUHAMMAD THORIQ BIN KAHAIRI (A21EC0096)
;                3) MUHAMMAD WAFFI QAYYUM BIN DIN (A21EC0097)
;                4) MUHAMMAD IRHAM HAKIM BIN ROSLAN (A21EC0081)
;                5) MUHAMMAD KAMIL EIZAZ BIN OTHMAN (A21EC0084)
; ---------------------------------------------------------------------------------------------------------------

INCLUDE Irvine32.inc


.data
        menuChoice DWORD ?
        hexaNum1 DWORD ?
        hexaNum2 DWORD ?
        str1 BYTE "Boolean AND ", 0
        str2  BYTE "Boolean OR ", 0
        str3 BYTE "Boolean NOT ", 0
        str4 BYTE "Boolean XOR ", 0
        str5 BYTE "--- Boolean Calculator ----------", 0
        str6 BYTE "1. x AND y", 0
        str7 BYTE "2. x OR y", 0
        str8 BYTE "3. x NOT y", 0
        str9 BYTE "4. x XOR y", 0
        str10 BYTE "5. Exit Program", 0
        str11 BYTE "---------------------------------", 0
        str12 BYTE "Enter your choice: ", 0
        str13 BYTE "Bye." ,0
        value BYTE "The 32-bit hexadecimal result is ", 0
        errorStr BYTE "Invalid number, please enter again. ", 0
        hexaStr1 BYTE "Input the first 32-bit hexadecimal operand: ", 0
        hexaStr2 BYTE "Input the second 32-bit hexadecimal operand: ", 0

.code
main PROC

Menu:
        call crlf                                   ;newline
        mov edx, offset str5                        ;print out str5
        call writestring
        call Crlf
        mov edx, offset str6                        ; print out str6
        call writestring
        call Crlf
        mov edx, offset str7                        ; print out str7
```

```
        call writestring
        call Crlf
        mov edx, offset str8                    ; print out str8
        call writestring
        call Crlf
        mov edx, offset str9                    ; print out str9
        call writestring
        call Crlf
        mov edx, offset str10                   ; print out str10
        call writestring
        call Crlf
        mov edx, offset str11                   ; print out str11
        call writestring
        call Crlf

        mov edx, offset str12                   ; print out str12
        call writestring
        call ReadDec                            ; get input from user
        mov menuChoice, eax
        call Crlf
        jmp L1

L1:
        cmp menuChoice, 1
        je AndInput                             ; jump if menuChoice = 1
        cmp menuChoice, 2
        je OrInput                              ; jump if menuChoice = 2
        cmp menuChoice, 3
        je NotInput                             ; jump if menuChoice = 3
        cmp menuChoice, 4
        je XorInput                             ; jump if menuChoice = 4
        cmp menuChoice, 5
        je ExitOutput                           ; jump if menuChoice = 5
        cmp menuChoice, 5
        ja ErrorOutput                          ; jump if menuChoice above than 5
        cmp menuChoice, 1
        jb ErrorOutput                          ; jump if menuChoice below than 1

AndInput:
        call Clrscr
        mov edx, OFFSET str1                    ; print out str1
        call WriteString
        call crlf
        call opAnd

OrInput:
        call Clrscr
        mov edx, OFFSET str2                    ; print out str2
        call WriteString
        call crlf
        call opOr


NotInput:
```

```
        call Clrscr
        mov edx, OFFSET str3                    ; print out str3
        call WriteString
        call crlf
        call opNot


XorInput:
        call Clrscr
        mov edx, OFFSET str4                    ; print out str4
        call WriteString
        call crlf
        call opXor


ExitOutput:

        mov edx, offset str13                   ; print out str13
        call writestring
        call crlf
        call crlf
        call WaitMsg                            ; system pause
        exit


ErrorOutput:
        call Crlf
        mov edx, OFFSET errorStr                ; print out errorStr
        call WriteString
        call crlf
        mov edx, offset str12                   ; print out str12
        call writestring
        call ReadDec                            ; get input from user
        mov menuChoice, eax
        call Crlf
        jmp L1

main ENDP

; ---------------------------------------
; AND OPERATION

opAnd PROC

mov eax, 0
mov edx, OFFSET hexaStr1                        ; print out hexaStr1
call WriteString
call ReadHex                                    ; input hexa number from user
mov  hexaNum1, eax
mov edx, OFFSET hexaStr2                        ; print out hexaStr2
call WriteString
call ReadHex                                    ; input hexa number from user
mov hexaNum2, eax
```

```
mov eax, hexaNum1                       ; calculation for AND operation
AND eax, hexaNum2

mov edx, offset value                   ; print out the result
call WriteString
call WriteHex
call crlf
call main

ret                                     ; return from procedure

opAnd ENDP

; ----------------------------------------
; OR OPERATION

opOr PROC

mov edx, OFFSET hexaStr1                ; print out hexaStr1
call WriteString
call ReadHex                            ; input hexa number from user
mov hexaNum1, eax
mov edx, OFFSET hexaStr2                ; print out hexaStr2
call WriteString
call ReadHex                            ; input hexa number from user
mov hexaNum2, eax

mov eax, hexaNum1                       ; calculation for OR operation
OR eax, hexaNum2


mov edx, offset value                   ; print out the result
call WriteString
call WriteHex
call crlf
call main

ret                                     ; return from procedure

opOr ENDP

; ----------------------------------------
; NOT OPERATION

opNot PROC

mov edx, OFFSET hexaStr1                ; print out hexaStr1
call WriteString
call ReadHex                            ; input hexa number from user

NOT eax                                 ; calculation for NOT operation

mov edx, offset value                   ; print out the result
call WriteString
```

```
call WriteHex; 0000FFF2
call crlf
call main

ret                                     ; return from procedure

opNot ENDP

; ----------------------------------------
; XOR OPERATION

opXor PROC

mov edx, OFFSET hexaStr1                ; print out hexaStr1
call WriteString
call ReadHex                            ; input hexa number from user
mov hexaNum1, eax
mov edx, OFFSET hexaStr2                ; print out hexaStr2
call WriteString
call ReadHex                            ; input hexa number from user
mov hexaNum2, eax

mov eax, hexaNum1                       ; calculation for NOT operation
XOR eax, hexaNum2

mov edx, offset value                   ; print out the result
call WriteString
call WriteHex
call crlf
call main

ret                                     ; return from procedure

opXor ENDP

END main
```