SEMESTER 2

SESSION 2019/2020

SECR 2033

PROJECT CASE 4

PREPARED BY :

1. NUR ALEEYA SYAKILA BINTI MUHAMAD SUBIAN (A19EC0127)
2. NUR HADIRAH MUNAWARAH BINTI ROZMIZAN (A19EC0201)

LECTURER'S NAME: DR. ZURIAHATI
SECTION : 02
SUBMITTED ON : 28/6/2020

Table of Contents

MEMBER RESPONSIBILITIES

Functional Team Leader : **NUR ALEEYA SYAKILA BINTI MUHAMAD SUBIAN**	● Generates idea and manage tasks ● Meets regularly to solve some issues ● Provides brilliant method to solve the problem ● Manage project with low risks ● Contributes to overall project especially coding ● Monitor team member to keep in tracking
Core Team Member : **NUR HADIRAH MUNAWARAH BINTI ROZMIZAN**	● Contributes to the project ● Provides requirement from team leader ● Attendant and actively participated ● Contributes to overall project especially report ● Performs the assigned tasks ● Implement the tasks given

CODING AND EXPLANATION

CODING

```
TITLE COA Project         (case4.asm)

          ; Group Members:
          ;1. NUR ALEEYA SYAKILA BINTI MUHAMAD SUBIAN (A19EC0127)
          ;2. NUR HADIRAH MUNAWARAH BINTI ROZMIZAN (A19EC0201)
          ; Section : 02

INCLUDE Irvine32.inc

.data
   num byte 10 dup(0)
   count dword 0

   LIM = 50
   inputString byte LIM+1 DUP(?)
          choice1 byte "y",0
          choice2 byte "n",0
   q1 BYTE "Please Enter 4-digit Hexadecimal integer (e.g.,A1B2): ",0
          s1 BYTE "Two's Complement of Hex ",0
          s2 BYTE " is ",0
          s3 BYTE "ERROR ",0
          q2 BYTE "Try again? (y/n) ",0
          s4 BYTE "WRONG CODE! please choose between (y/n) !!",0
          cb byte 0                                   ;Check when propagation or not

.code
main PROC

dowhile:

   mov edx, OFFSET q1 ;"Please Enter 4-digit Hexadecimal integer (e.g.,A1B2):
          call WriteString
```

```
        mov edx, OFFSET num
        mov ecx, SIZEOF num
        call ReadString
                                ;reading input from user

        mov count, eax
        mov edx, OFFSET s1      ;"Two's Complement of Hex"
        call WriteString
        mov edx, OFFSET num     ;display original input from user
        call WriteString
        mov edx, OFFSET s2      ;"is"
        call WriteString

        INVOKE Str_ucase, ADDR num
        mov esi, OFFSET num
        mov ecx, 3

FIVENBELOW:
        mov al, byte ptr [esi+ecx]
        cmp al, 53
        ja FIVENABOVE
        sub al, 48
        mov bl, 70
        sub bl, al
        jmp MSB


FIVENABOVE:
        cmp al, 57
        ja ALPHABET
        sub al, 54
        mov bl, 57
        sub bl, al
        jmp MSB
```

```
ALPHABET:
        cmp al, 70
        ja wrong
        sub al, 65
        mov bl, 53
        sub bl, al
        jmp MSB


MSB   :
        cmp ecx, 3
        jne propa

here1:

        cmp bl, 57
        jne here2
        mov bl, 65
        mov cb, 0
        jmp answer


here2:
        cmp bl, 70
        jne here3
        mov bl, 48
        mov cb, 1
        jmp answer

here3:
        add bl, 1
        mov cb, 0
        jmp answer

propa:
        cmp cb, 0
        jne here1
```

```
answer:
        mov byte ptr [esi+ecx], bl
        dec ecx
        cmp ecx, -1
        je final
        jmp FIVENBELOW



final:
        mov edx, OFFSET num
        call WriteString
        call Crlf
        jmp exitt



wrong:
        mov edx, OFFSET s3        ;"Wrong hexadecimal entered"
        call WriteString
        call Crlf
        jmp exitt

exitt:
        mov edx, OFFSET q2
        call WriteString
        mov  edx,OFFSET inputString
   mov  ecx ,LIM
   call ReadString
   INVOKE Str_compare, ADDR inputString, ADDR choice1         ;y
        jne No
        call Crlf
        jmp dowhile

No:
        INVOKE Str_compare, ADDR inputString, ADDR choice2   ;n
   jne other
        call Crlf
        jmp outt
```

```
other:
        call Crlf
        call Crlf
        mov edx, OFFSET s4
        call WriteString
        call Crlf
        call Crlf
        call WaitMsg                                            ; "Press [Enter]..."
        call Clrscr                                             ; clear screen
        jmp exitt

outt:
        call WaitMsg                                            ; "Press [Enter]..."
        call Clrscr                                             ; clear screen
        exit
main ENDP


END main
```

EXPLANATION

In .data inside the MASM coding, num is stored in an array with data type byte, count with data type dword which is double world, **LIM** that is assigned to 50 and lastly inputString with unknown value that has byte data type. In addition, there are about 8 statements, choice1, choice2, q1,s1,s2,s3, q2 and s4. The statements are stored in byte. cb is represent for the carrybit.

The main .code, inside the **dowhile**, **q1** is displayed which is *"Please Enter 4-digit Hexadecimal integer (e.g.,A1B2):"* And the size of array num which is 10 is assigned to register ecx. It then reads the user input and stores it in a data type string. After that, it is stored in register eax. And then, **s1** and **s2** are displayed in output which is *"Two's Complement of Hex "* and *"is"* respectively. Then, the first address of num is stored in register esi. Lastly, 3 is assigned to the register of ecx.

In **FIVENBELOW**, the pointer of the sum register esi and ecx is moved into al register. Then it compares with al register with 53(5 in ascii table). If it is bigger than 53, it will jump to **FIVENABOVE.** This means that if al is less or equal to 53, the number is smaller or equal to 5. Then, in it will subtract register al with 48, move 70 to register bl and sub al with bl. This clearly means that the above 3 lines are doing 1's complement. Lastly, it will jump to **MSB**.

In **FIVENABOVE**, it will compare to 57 (9 in ascii table). If al is above than 57, it means the number is bigger than 9. Then, it jump above (ja) to **ALPHABET**. If not, it will subtract with 54,move 57 into bl register and subtract al with bl register. Lastly, it will jump to **MSB**.

In **ALPHABET**, it will compare al register with 70 (F in ascii table). If al is less than or equal to 70, it means that the number is smaller to F. Proceed, we subtract between al register and 65, move 53 into bl register and subtract al register and bl register. Lastly, it will jump to **MSB**. But if bigger than F, it is invalid. It will jump to **wrong** if above than F.

In **MSB**, 3 is compared to register ecx. And it will jump if not equal (jne) to **propa**. This means that if it is not Most Significant Bit (MSB) , it will directly skip the below step to **propa**(propagation).

In **here1**, this means carry bit propagation only occurs if the radix(digit) in F. First, it will compare register bl with 57,compare if it is 9 (57th in ASCII Table), it will become A. Then it will jump if not equal (jne) to **here2** which means if it is not 9 , it will jump to **here2** to do the next task. After that, 65 is moved to register bl and 0 is moved to cb. This means that, take for example 9+1=A, there is no carry. Therefore, the carry bit is 0. Lastly, it will jump to **answer**.

In **here2** , register bl is compared to 70 . It compares with F (70 in ASCII Table) or not.It will jump to **here3** if not equal. Then, 48 is moved to register bl and also 1 is moved to cb. This means F+1=10 , there is a carry bit going on. Therefore, the carry bit is set to 1. Then it will lastly jump to **answer**.

In **propa**, 0 is compared to cb(carry bit). It will check whether there is carry bit propagation or not. If it is not equal to 0, It will go to **here1.** It will jump back to **here1** to check the next digit. If it is 0, it will proceed to **answer.**

In **answer**, we will assign a new digit(digit that has changed to 2's complement) to the char array. Register bl is move to pointer of sum of esi and ecx. Then, it will decrement the counter which is ecx to update the counter. After that, -1 will compare to the counter. This means it will jump to **final** if the counter is equal to -1. It will repeat the step by jump to **FIVENBELOW** if it is not -1.

In **final**, it will display the full 2's complement after completing and change all of the user input to 2's complement. Then it will jump to **exitt** which means jump to exit or ask the user if they want to try again or not.

In **wrong**, **s3** is displayed which is "Error" if user enters an INVALID hexadecimal number. Then it will jump to **exitt**.

In **exitt** , **q2** is displayed which is "Try again? (y/n)" and the counter is assigned to **LIM**(that has a value of 50). Here, it will compare the user entered code whether it's the same with **choice1** (y). If it is not the same, it will jne to **No.** But if it's the same, it will jump to **dowhile** and start all over again.

In **No**, it will compare the user entered code whether it's the same with choice2 (n). If it's not the same, it will jne to **other.** If it is the same, it will jump to **outt** to proceed with exiting the program.

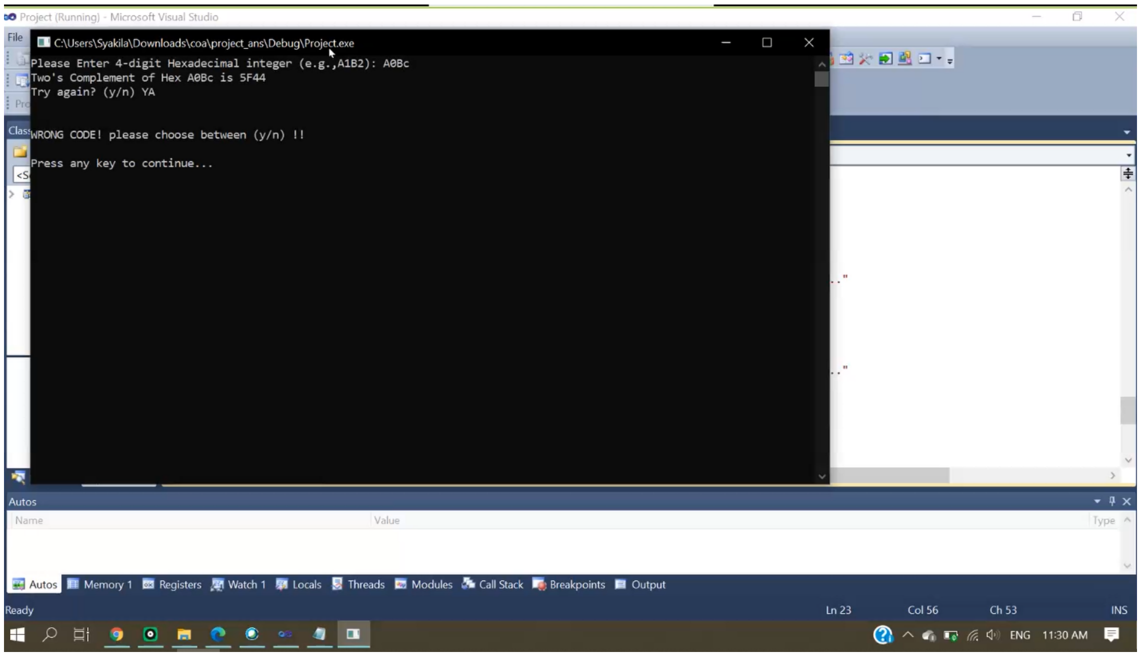
In **other**, **s4** is displayed which is *"WRONG CODE! please choose between (y/n) !!"*. Then it jumps to **exitt**.

Lastly in **outt**, if the user press enter it will freeze for a while then it clears the screen and the program will end.

EXAMPLES OF INPUTS & OUTPUTS

INPUT & OUTPUT

Example for wrong input of choice (y/n), the user input capital letter “YA” instead of y or n

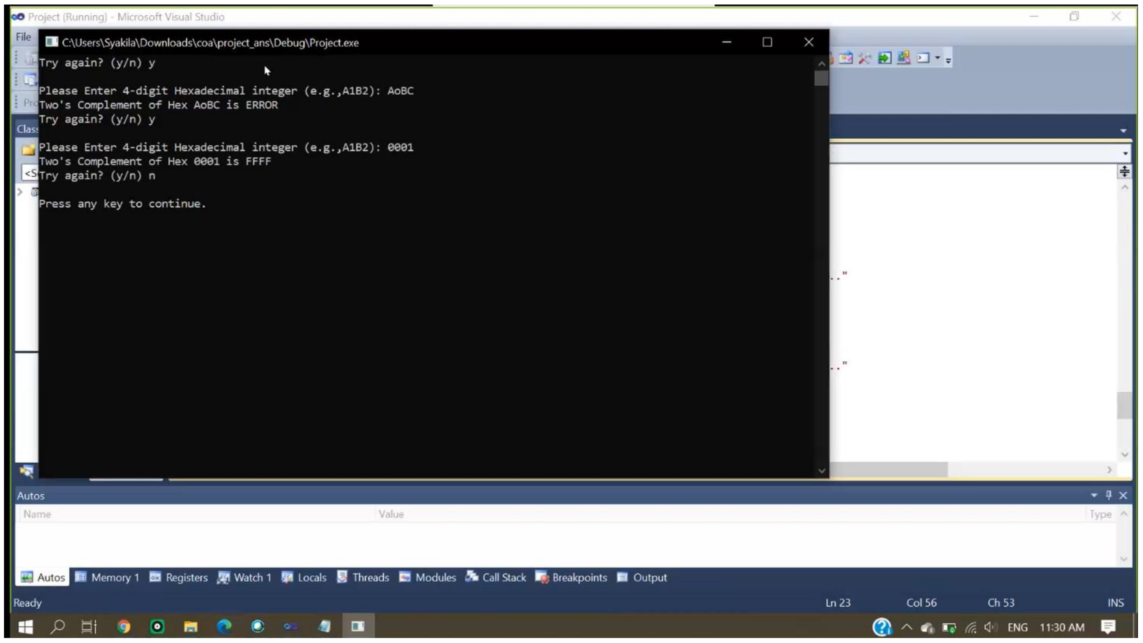


```
Please Enter 4-digit Hexadecimal integer (e.g.,A1B2): A0Bc
Two's Complement of Hex A0Bc is 5F44
Try again? (y/n) YA


WRONG CODE! please choose between (y/n) !!

Press any key to continue...
```

Example for wrong hexadecimal, correct user input of choice (y/n)



```
Try again? (y/n) y

Please Enter 4-digit Hexadecimal integer (e.g.,A1B2): AoBC
Two's Complement of Hex AoBC is ERROR
Try again? (y/n) y

Please Enter 4-digit Hexadecimal integer (e.g.,A1B2): 0001
Two's Complement of Hex 0001 is FFFF
Try again? (y/n) n

Press any key to continue.
```

DISCUSSION AND CONCLUSION

It can be discussed that the function of this coding is to read the user input which is 4 digits hexadecimal integers. If the user wrongly enters a hexadecimal integer, it will show error output. Then, the coding converts the inputs into two’s complement of HEX numbers. If the input is correct, it will display HEX numbers after applying two’s complement method. After that, the coding will ask the user if they want to repeat the question again.

However, if the user gives the input wrongly to that question, it will directly execute and give errors messages. If the input is correctly entered, it will continue to ask for entering another 4-digits hexadecimal integer.

From this, it can be seen that the user's input is important such as entering “y” or “n” choices. This is because if the user entered another than the particular character, it will straightly give wrong output. Another thing is that if the user enters the input of hexadecimal numbers wrongly, it will also show error messages because this coding will check the input one by one by comparing using ASCII value.

References

Koppella, G. (2014, April 15). *An assembly program to accept a decimal number and print it’s 2’s complement binary and hexadecimal*. From cssimplified: http://cssimplified.com/assignments/an-assembly-language-program-to-accept-a-decimal-number-and-display-its-twos-complement-representation-in-binary-and-hexadecimal-formats

Rathor, R. (2019, January 22). *program to find 1's and 2's complement of 8-bit number*. From tutorialspoint: https://www.tutorialspoint.com/8085-program-to-find-1-s-and-2-s-complement-of-8-bit-number

thehamzayy. (2015, November 18). *Twos complement of a hexadecimal to form the twos*. From coursehero: https://www.coursehero.com/file/p5i7ip0/Twos-Complement-of-a-Hexadecimal-To-form-the-twos-complement-of-a-hexadecimal/