UTM

UNIVERSITI TEKNOLOGI MALAYSIA

SECR2033 COMPUTER ORGANIZATION AND ARCHITECTURE

SECTION-02

20202021/2

PROJECT

LECTURER'S NAME : DR AIDA ALI

VIDEO LINK:

GROUP 4.0 MEMBERS :

NAME	MATIRC NO.
AFIF HAZMIE ARSYAD BIN AGUS	A20EC0176
HONG PEI GEOK	A20EC0044
LOW JUNYI	A20EC0071
LEE JIA XIAN	A20EC0200
MUHAMMAD IMRAN HAKIMI BIN MOHD SHUKRI	A20EC0213
YONG ZHI YAN	A20EC0172

1.0 Member Responsibilities

NAME	TASKS
AFIF HAZMIE ARSYAD BIN AGUS	
HONG PEI GEOK	Coding: ● Convert char digit to a number value Report: ● Coding and Explanation Video
LOW JUNYI	Coding: ● Receive a 4 char Hex digits ● Get a Hex char digit Report: ● Example inputs and outputs Video
LEE JIA XIAN	Coding: ● Convert number value to char digit Report: ● Coding and Explanations Video
MUHAMMAD IMRAN HAKIMI BIN MOHD SHUKRI	

YONG ZHI YAN	Coding: ● Loop for repeating the process Report: ● Coding and Explanation ● Conclusion Video

2.0 Table of Contents

3.0 Coding and Explanation

```
TITLE PROJECT   (main.asm)

INCLUDE Irvine32.inc
INCLUDE Macros.inc

.data
value           BYTE 5 dup(0)
checkvalue      BYTE 5 dup(0)
choice          BYTE 4 dup(0)


.code
main PROC
START :

;low (reveice input)
mwrite      "Please Enter 4-digit Hexadecimal integer (e.g., A1BE): "
mov         edx, offset value
mov         ecx, (SIZEOF value)
call        ReadString

mov     esi, offset value + 3
mov     edi, offset checkvalue + 3
mov     ecx, 3
```

In the declaration of data, the variable value is used to receive and store the four chaar digit input by the user. The variable checkvalue is used to store the final answer of the value (2's complement). The variable choice is used to receive and store the decision made by the user (y/n),

In the code part, it starts from the START. A message is popped out to prompt the user to enter a 4-digit hexadecimal and it is stored in the variable value. The offset of value and checkvalue is added by 3 to locate the last digit (LSB) input by the user. While the 3 that moved into the ecx register acts as a counter for a loop that run 4 times.

```
;afif (make upper case)
CHECK :
mov al, [esi]
cmp al, 70      ;70 in decimal = F in char
JBE DECIMAL     ;check if char is upper case jump to DECIMAL
Sub al, 32      ; change to upper case

;py (convert char digit to a number)
;convert A-F to decimal (10-15)
DECIMAL:
cmp al,  65;    ; 65 in decimal = A in char
JB REVERSE
sub al, 7   ; we assume 58 in decimal is 10 as 57 = 9 in  char


REVERSE:
mov bl, 63  ; we used (48-63) to act as (0-15) in this operation
sub bl, al  ;user 63(15) to minus the single bit
mov al, bl  ;mov the answer into al back
;check input of user is whether valid or not
cmp al, 0 ;
JB ERRORIN
cmp al, 15
JA ERRORIN
```

In “CHECK”, we move the value of address held by register esi to register al and compare it with 70 in decimal. If the value of register al is smaller or equal to 70, the code will jump to “DECIMAL”. This is because 70 in decimal represents char F in Ascii table and the part “DECIMAL” will convert the char digit to a number which is equal to its second complement value. Otherwise, the code will continue to subtract register al with 32 before going to the part “DECIMAL” and the purpose is to make the value of al become upper case. In the Ascii table, char a to f is equal to decimal 97-102 and the difference between A to F and a to f in Ascii table is 32. Thus, the subtraction with 32 is used to convert the content of register al to upper case.

After going through “CHECK”, the “DECIMAL” part is responsible to determine the content of al by comparing it with decimal 65. If the value of al is below 65, the code will jump to “REVERSE”. Else, it will continue subtracting al with 7. This is because 65 is equal to A in char, if below 65 it means the value of al is either decimal or error and it will jump to “REVERSE” in order to get the second complement or catch an error. Next, if al is larger than

65, it will proceed to subtraction with 7 and continue with “REVERSE” and the value of al is in between A-F.

In “REVERSE”, ……

```
SET:
mov checkvalue[ecx], al
dec esi
dec ecx
cmp ecx, 0
JGE CHECK


;imran (ADD 1)
mov ecx, 4
L1:
mov al, checkvalue[ecx-1]
add al, 1
cmp al, 16 ; if value = 16 (G)
JNE NOCARRY
mov checkvalue[ecx-1], 0
loop L1

NOCARRY:
mov checkvalue[ecx-1], al
```

The aim of “SET” is used to set the value of checkvalue with the content of al. In this block, it will also decrease the value of the esi and ecx register before going to check the next digit store in value. This part of the code will decide whether we should jump to check the next digit or proceed to the next stage of finding the second complement by comparing the ecx register with 0. If the ecx register is greater or zero this means the system has not checked all the four digits and it will jump to “CHECK” otherwise it will continue with the next stage of checking.

```
;ljx  (change back to alpha)
mov ecx, 4
L2:
mov al, checkvalue[ecx-1]
cmp al, 10
JAE ITOA
add al, 48
ALPHA:
mov checkvalue[ecx-1], al
loop L2
jmp SHOW

ITOA:
add al, 55
jmp ALPHA


SHOW:
mwrite  "Two's Complement of Hex "
mov edx, OFFSET value
call writestring
mwrite  " is "
mov edx, OFFSET checkvalue
call writestring
call crlf
```

The purpose of L2 is to change all decimal values back to characters. The ecx register is declared as 4 as a counter for L2. In the L2 loop, the LSB number is first compared with the decimal number 10. If it is greater than or equal to 10, the code will jump to ITOA, otherwise it will add decimal 48. According to the ASCII table, we noticed that the decimal value (48-57) is equal to (0-9) in char, and the difference between them is 48. Therefore, if the value of al is a decimal value from 0 to 9, add 48 to al to convert it back to (0-9) in char. And if the value of al is decimal 10 to 15, add 55 to it to convert it back to (A-F) in characters. This is because (65-70) in decimal is equal to (A-F) in char and the difference between (10-15) and (65-70) is 55. After that, the code continues to ALPHA to store the converted value back to the variable checkvalue. This process is repeated 4 times for the 4 hexadecimal integers. After that, the SHOW block will display the value in the variable checkvalue, which is the 2's complement of the hexadecimal integer entered by the user.

```
;zhiyan (Repeat process)
TRYAGAIN:
mwrite "Try again? (y/n) "
mov edx, OFFSET choice
mov ecx, (SIZEOF choice)-1
call readString
call crlf
mov al, [edx]
cmp al, 'y'
je START
jne ERRORC


;IF user input invalid hexadecimal
ERRORIN:
mwrite  "Two's Complement of Hex "
mov edx, OFFSET value
call writestring
mwrite  " is Error"
call crlf
JMP TRYAGAIN
```

```
;if user input invalid choice (y/n)
ERRORC:
cmp al, 'n'
je BYE
mwrite "Invalid input!"
call crlf
JMP TRYAGAIN

;End Program
BYE:
mwrite "Thank You!"

    exit
main ENDP

END main
```

4.0 Example of inputs & outputs

```
Please Enter 4-digit Hexadecimal integer (e.g., A1B2): 6666
Two's Complement of Hex 6666 is 999A
Try Again ? (y/n) y

Please Enter 4-digit Hexadecimal integer (e.g., A1B2): 9999
Two's Complement of Hex 9999 is 6667
Try Again ? (y/n) y

Please Enter 4-digit Hexadecimal integer (e.g., A1B2): AAAA
Two's Complement of Hex AAAA is 5556
Try Again ? (y/n) y

Please Enter 4-digit Hexadecimal integer (e.g., A1B2): F1F5
Two's Complement of Hex F1F5 is 0E0B
Try Again ? (y/n) y

Please Enter 4-digit Hexadecimal integer (e.g., A1B2): FFF9
Two's Complement of Hex FFF9 is 0007
Try Again ? (y/n) y

Please Enter 4-digit Hexadecimal integer (e.g., A1B2): y999
Two's Complement of Hex y999 is ERROR
Try Again ? (y/n) y

Please Enter 4-digit Hexadecimal integer (e.g., A1B2): 83B3
Two's Complement of Hex 83B3 is 7C4D
Try Again ? (y/n) y

Please Enter 4-digit Hexadecimal integer (e.g., A1B2): 7CD4
Two's Complement of Hex 7CD4 is 832C
Try Again ? (y/n) y

Please Enter 4-digit Hexadecimal integer (e.g., A1B2): 9F9F
Two's Complement of Hex 9F9F is 6061
Try Again ? (y/n) y

Please Enter 4-digit Hexadecimal integer (e.g., A1B2): BBBB
Two's Complement of Hex BBBB is 4445
Try Again ? (y/n) y

Please Enter 4-digit Hexadecimal integer (e.g., A1B2): ABCD
Two's Complement of Hex ABCD is 5433
Try Again ? (y/n) y

Please Enter 4-digit Hexadecimal integer (e.g., A1B2): abcd
Two's Complement of Hex abcd is 5433
Try Again ? (y/n) n
```

5.0 Discussion

6.0 Conclusion

7.0 References

8.0 Appendix

```
TITLE PROJECT       (main.asm)

INCLUDE Irvine32.inc
INCLUDE Macros.inc

.data
value                   BYTE 5 dup(0)
checkvalue              BYTE 5 dup(0)
choice                  BYTE 4 dup(0)


.code
main PROC
START :

;low (reveice input)
mwrite          "Please Enter 4-digit Hexadecimal integer (e.g., A1BE): "
mov             edx, offset value
mov             ecx, (SIZEOF value)
call            ReadString

mov             esi, offset value + 3
mov             edi, offset checkvalue + 3
mov             ecx, 3

;afif (make upper case)
CHECK :
mov al, [esi]
cmp al, 70              ;70 in decimal = F in char
JBE DECIMAL             ;check if char is upper case jump to DECIMAL
Sub al, 32              ; change to upper case

;py (convert char digit to a number)
;convert A-F to decimal (10-15)
DECIMAL:
cmp al,  65;    ; 65 in decimal = A in char
JB REVERSE
sub al, 7       ; we assume 58 in decimal is 10 as 57 = 9 in  char


REVERSE:
mov bl, 63      ; we used (48-63) to act as (0-15) in this operation
```

```
sub bl, al        ;user 63(15) to minus the single bit
mov al, bl        ;mov the answer into al back
;check input of user is whether valid or not
cmp al, 0 ;
JB ERRORIN
cmp al, 15
JA ERRORIN


SET:
mov checkvalue[ecx], al
dec esi
dec ecx
cmp ecx, 0
JGE CHECK


;imran (ADD 1)
mov ecx, 4
L1:
mov al, checkvalue[ecx-1]
add al, 1
cmp al, 16 ; if value = 16 (G)
JNE NOCARRY
mov checkvalue[ecx-1], 0
loop L1

NOCARRY:
mov checkvalue[ecx-1], al

;ljx（change back to alpha）
mov ecx, 4
L2:
mov al, checkvalue[ecx-1]
cmp al, 10
JAE ITOA
add al, 48
ALPHA:
mov checkvalue[ecx-1], al
loop L2
jmp SHOW

ITOA:
add al, 55
```

```
jmp ALPHA


SHOW:
mwrite "Two's Complement of Hex "
mov edx, OFFSET value
call writestring
mwrite " is "
mov edx, OFFSET checkvalue
call writestring
call crlf

;zhiyan (Repeat process)
TRYAGAIN:
mwrite "Try again? (y/n) "
mov edx, OFFSET choice
mov ecx, (SIZEOF choice)-1
call readString
call crlf
mov al, [edx]
cmp al, 'y'
je START
jne ERRORC


;IF user input invalid hexadecimal
ERRORIN:
mwrite "Two's Complement of Hex "
mov edx, OFFSET value
call writestring
mwrite " is Error"
call crlf
JMP TRYAGAIN

;if user input invalid choice (y/n)
ERRORC:
cmp al, 'n'
je BYE
mwrite "Invalid input!"
call crlf
JMP TRYAGAIN

;End Program
BYE:
```

```
mwrite "Thank You!"

        exit
main ENDP

END main
```