Subject: Microprocessor Lab

Assignment No. 02

Roll No: **21118**

Batch: E-1

Problem Statement:

Write X 86/64 ALP to accept a string and to display its length.

Hardware of PC:

- Manufacturer and model: Acer Swift-3
- Processor: Intel core i5 – 8265U @1.60 GHz
- Memory: 8GB of DDR4 RAM and 512GB of ROM
- System Type: 64-bit OS, x-64 based PC

Software Used:

- Operating system: Ubuntu 20.04 LTS on oracle virtual machine
- Text editor: Gedit (version: 3.36.2)
- Assembler: NASM (version: 2.14.02)

Theory:

*Instructions:*

1. *rol:* The left rotate instruction shifts all bits in the register or memory operand specified. The most significant bit is rotated to the carry flag, the carry flag is rotated to the least significant bit position, all other bits are shifted to the left. The result does not include the original value of the carry flag.
   Syntax: rol op, num
2. *and:* The AND instruction is used for supporting logical expressions by performing bitwise AND operation. The bitwise AND operation returns 1, if the matching bits from both the operands are 1, otherwise it returns 0.
   Syntax: and op1, op2
3. *Cmp:* The CMP instruction compares two operands. It is generally used in conditional execution. This instruction basically subtracts one operand from the other for comparing whether the operands are equal or not. It does not disturb the destination or source operands. It is used along with the conditional jump instruction for decision making.
   Syntax: CMP destination, source

4. *Jbe:* Jumps to the destination label mentioned in the instruction if the result of previous instruction (generally compare) causes either the CF or ZF to have value equal to 1, else no action is taken.
   Syntax: jbe label
5. *Jnz:* Jumps to the destination label mentioned in the instruction if the ZF is 0, else no action is taken.
   Syntax: jnz label

Algorithm:

- *Finding length of string:*
  1. Declare the byte array to store the string.
  2. Declare slen of 2 bytes to store the length of string.
  3. Read string from user.
  4. Collect the accumulator value in variable.
  5. Convert the value from hex to ascii.
  6. Display the length of string.
  7. Exit Syscall.
- *Hex to ascii conversion:*
  1. Initialize counter to 2
  2. Set rsi pointer to slen
  3. Use rol instruction to reverse the number (rol al,04h).
  4. Move al to bl.
  5. Perform and operation on bl with 0Fh
  6. Compare bl with 09h
  7. If bl is less than or equal to 09h then goto step 9
  8. Else add 07h to bl
  9. Add 30h to bl and move the content of bl in rsi.
  10. Increment rsi and decrement counter
  11. if counter is not zero then goto step 3
  12. Else display the content of slen

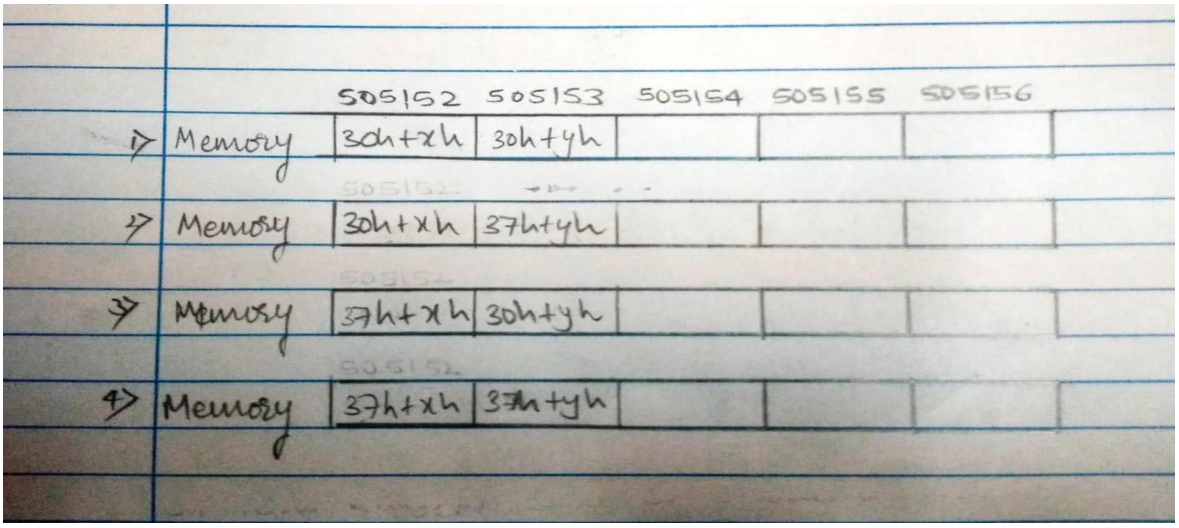
Illustration of data in memory (Diagram):

*Hex to ascii conversion*

Assume the two digits hex number as xyh

1. x <= 09h and y <= 09
2. x <= 09h and y > 09
3. x > 09h and y <= 09

4. x > 09h and y > 09




Program:

```
%macro rwm 03
        mov rax, %1
        mov rdi, 01
        mov rsi, %2
        mov rdx, %3
        syscall
%endmacro rwm

section .data

        msg1 db "Enter String: "
        l1 equ $ -msg1

        msg2 db "Length of String is: "
        l2 equ $ -msg2

        newline db 0xA

section .bss

        mystr resb 30                   ; array to store string
        slen resb 02                    ; array to store string len
        cnt resb 01

global _start
section .text
_start:
```

```
        rwm 01, msg1, l1            ; displaying msg
        rwm 00, mystr, 30           ; reading string



        mov byte[cnt], 02           ; counter

        mov rsi, slen               ; rsi point to first loc of str len array

again:                              ; loop for each digit of hex

        rol al, 04                  ; rol for rightmost digit
        mov bl, al
         and bl, 0fh                ; bl stores digit (rightmost)
        cmp bl, 09h                 ; next 5 lines => hex->ASCII for single digit

        jbe add30h
        add bl, 07h

add30h:

        add bl, 30h
        mov [rsi], bl               ; mov digit to str len array
        add rsi, 01                 ; rsi will point to next location of str
                                ; len array

        dec byte[cnt]
        jnz again


        rwm 01, msg2, l2
        rwm 01, slen, 02            ; print strlen array
         rwm 01, newline, 01

        ; exit syscall
        mov rax, 60
        mov rdi, 00
        syscall
```

Output:

```
shubham20_03@ubuntu:~/Assembly$ nasm -f elf64 str_len2.asm
shubham20_03@ubuntu:~/Assembly$ ld -o out str_len2.o
shubham20_03@ubuntu:~/Assembly$ ./out
Enter String: Hellooo!
Length of String is: 09
```

```
shubham20_03@ubuntu:~/Assembly$ nasm -f elf64 str_len2.asm
shubham20_03@ubuntu:~/Assembly$ ld -o out str_len2.o
shubham20_03@ubuntu:~/Assembly$ ./out
Enter String: Hello, Shubham here!
Length of String is: 15
shubham20_03@ubuntu:~/Assembly$
```

Conclusion:

In this assignment I learned about hex to ascii conversion in assembly programming. I also written a program to do the same.