DEPARTMENT OF COMPUTER & SOFTWARE ENGINEERING

COLLEGE OF E&ME, NUST, RAWALPINDI

Microprocessor and Microcontroller Based Design

Lab 05

SUBMITTED TO:
Dr Taimoor Zahid

SUBMITTED BY:
AMINA QADEER
Reg # 00000359607
DE-42 (C&SE)-A

Submission Date: 31/10/2022

Objectives:

2-3 lines

Related Topic/Chapter in theory class:

None

Hardware/Software required:

Hardware: PC
Software Tool: emu8086 v2.57

Task 1:

What do POPF and PUSHF instructions do? Do they affect flags in any way?

Solution:

PUSHF simply pushes the current contents of the Flags register onto the top of the stack. *The top of the stack* is defined as the word at SS:SP, and there is no way to override that with prefixes.

SP is decremented *before* the word goes onto the stack. Remember that SP always points to either an empty stack or else real data. There is a separate pair of instructions, PUSH and POP, for pushing and popping other register data and memory data.

The Flags register is not affected when you *push* the flags, but only when you pop them back with POPF.

PUSHF and POPF are most used in writing interrupt service routines, where you must be able to save and restore the environment, that is, all machine registers, to avoid disrupting machine operations while servicing the interrupt.

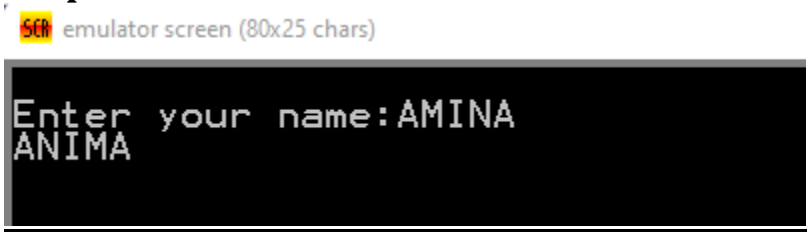
2. Prompt the user for entering a word terminated by enter. Display the characters of the word in reverse order using stack.

Solution:

```
; You may customize this and other start-up templates.
; The location of this template is c:\emu8086\inc\0_com_template.txt

org 100h
.data

  str db 13,10, 'Enter your name:$'
  buffer DB 13,10, 08, 09 DUP(?) 13,0ah
  space DB 20h
.code
main proc
  mov AX, @data
```

```
    mov ds,ax


    mov dx,offset str
    mov ah,9
    int 21h

    MOV AH, 0AH
    MOV DX, OFFSET buffer
    INT 21H

    mov  SI,OFFSET buffer+2
    mov cx,8


    L1:

    mov bx,[SI]
    push bx
    inc SI
    Loop L1
    mov cx,8


    L2:

    pop dx


    mov ah,2
    int 21h

    loop L2
    mov ah,4ch
    int 21h


  main endp

ret
```

Output:



Task 3:

Write an assembly code for calculating the factorial of a number input by user. Attach code and screenshot of console.

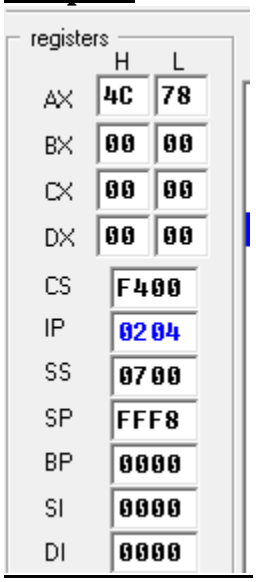
Solution:

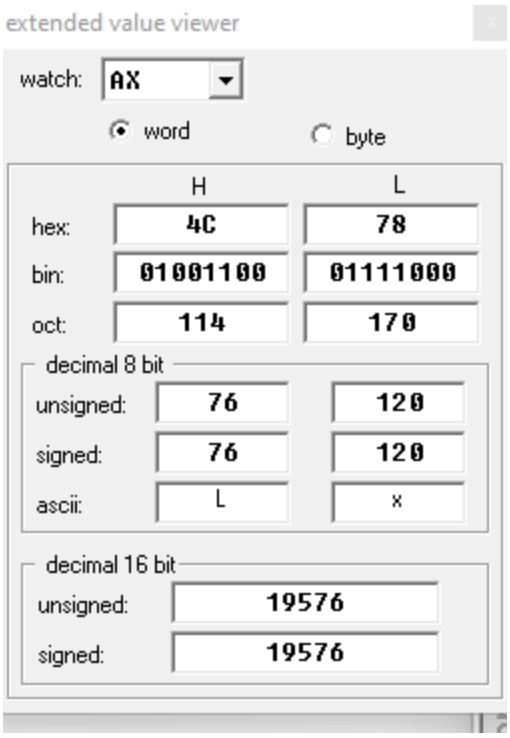
```
; You may customize this and other start-up templates;
; The location of this template is c:\emu8086\inc\0_com_template.txt

org 100h

org 100h
.data

  str db 13,10, 'Enter your name:$'
  buffer DB 13,10, 08, 09 DUP(?) 13,0ah
  space DB 20h
.code
main proc
  mov AX, @data
  mov ds,ax


  mov ax,5
  mov cx,ax
  dec cx

  label:
  mul cx
  loop label


  mov ah,4ch
  int 21h
```

main endp

ret

Output:





Conclusion:

The FAR CALL can call a procedure anywhere in the system memory.

The near call pushes the 16 bit offset of the next instruction following the call onto the stack.It copies the 16 bit effective address into the IP register. The CALL instruction transfers the flow of the program to the procedure. The CALL instruction differs from the jump instruction in the sense that a CALL saves a return address on the stack. The RET instruction return control to the instruction that immediately follows the CALL. There exist two types of calls: FAR and NEAR. .