**Department of Electrical Engineering**

|  |  |
| --- | --- |
| **Faculty Member: Ma’am Qurat-ul-ain** | **Dated: November 29, 2020** |
|  |  |
| **Course/Section: BSCS-9B** | **Semester: 3rd** |
|  |  |

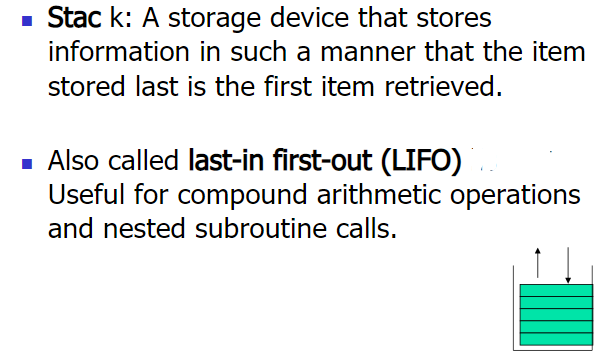
**Computer Organization and**

**Assembly Language (CS235)**

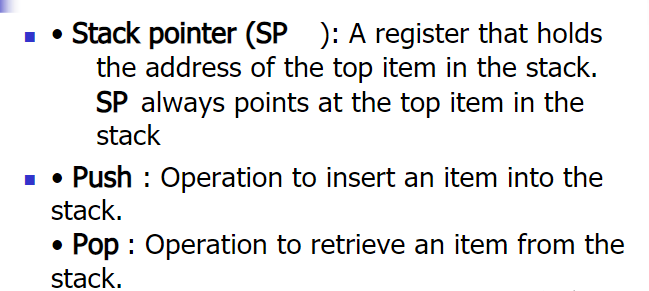
**Lab #7 Stacks and Procedures in Assembly language**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | | **PLO4 / CLO4** | | **PLO5/ CLO5** | **PLO8/ CLO 6** | **PLO9/ CLO 7** |  |
| **Name** | **Roll number** | **Viva /Quiz/ Lab performance**  **5 marks** | **Analysis of data in lab report**  **5 marks** | **Modern tool Usage**  **5 marks** | **Ethics and Safety**  **5 marks** | **Individual and team-work**  **5 marks** | **Total**  **25 marks** |
| **Fatima Seemab** | **291310** |  |  |  |  |  |  |
| **Mahum Samar** | **290647** |  |  |  |  |  |  |
| **Maryam Fatima** | **290479** |  |  |  |  |  |  |

**Objective:** The aim of this lab is to write procedures that are called from the main procedure to use stack related instructions.









Syntax  
push <reg32>  
push <mem>  
push <con32>



Examples  
push eax — push eax on the stack  
push [var] — push the 4 bytes at address var onto the stack



Syntax  
pop <reg32>  
pop <mem>



Examples  
pop edi — pop the top element of the stack into EDI.  
pop [ebx] — pop the top element of the stack into memory at the four bytes starting at location EBX.

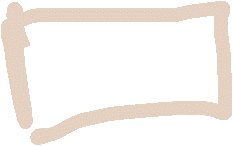


**Writing Procedure (Function) in Assembly**

Procedure is a part of code that can be called from your program in order to make some specific task. Procedures make program more structural and easier to understand. Generally, procedure returns to the same point from where it was called. The syntax for procedure declaration:



*name PROC  
      ; here goes the code  
      ; of the procedure ...  
RET  
name ENDP*



name - is the procedure name, the same name should be in the top and the bottom, this is used to check correct closing of procedures.   
  
Probably, you already know that **RET** instruction is used to return to operating system. The same instruction is used to return from procedure (actually operating system sees your program as a special procedure).   
  
PROC and ENDP are compiler directives, so they are not assembled into any real machine code. Compiler just remembers the address of procedure.   
  
CALL instruction is used to call a procedure.   
  
Here is an example: 



|  |
| --- |
| *ORG 100h*  *CALL m1*  *MOV AX, 2*  *RET ; return to operating system.*  *m1 PROC*  *MOV BX, 5*  *RET ; return to caller.*  *m1 ENDP*  *END* |

The above example calls procedure m1, does MOV BX, 5, and returns to the next instruction after CALL: MOV AX, 2.



**Exercise 1:**

Write and execute the assembly following code

TITLE defining variables

; just using variables

Include irvine32.inc

.stack 100h



.data



mbyte BYTE 05,65,81

Sum DWORD ?



.code

Main Proc

movzx eax,mByte

movzx ebx, mByte+1



movzx ecx, mByte+2

call mySum ; cal procedure

mov edx,sum

call dumpregs



Exit

Main endp



; mySum procedure

mySum Proc

PUSHAD ; Push all data registers on stack



add eax,ebx

add eax,ecx



mov sum,eax

call dumpregs

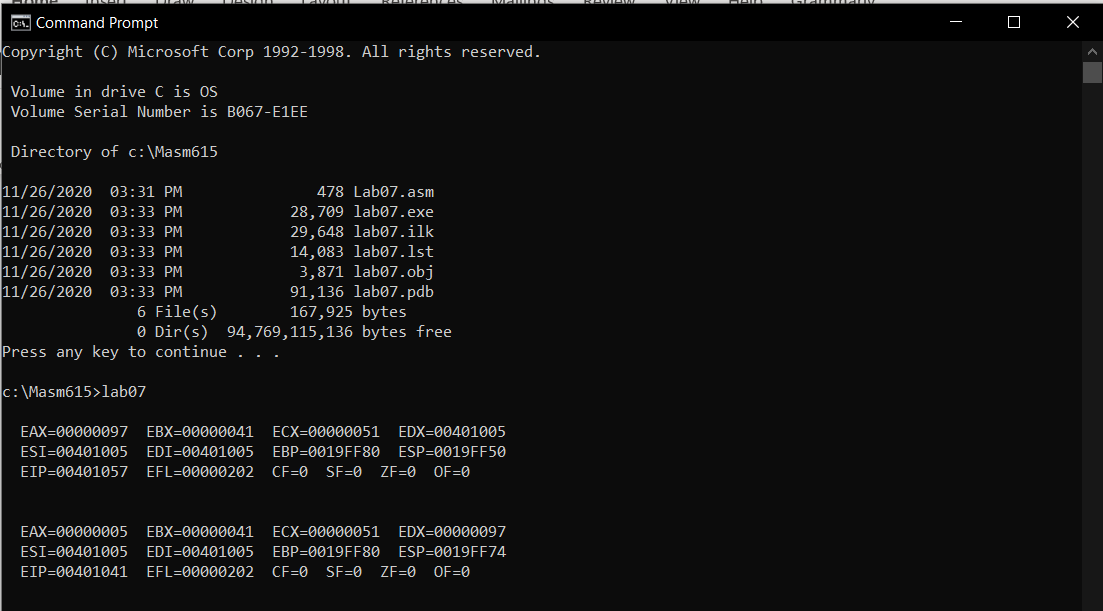
POPAD ; pop all data registers

RET

mySum endp

End main

**Output:**

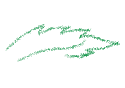


**Exercise 2:**

Write an assembly language code in that main procedure performs the following:

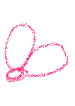
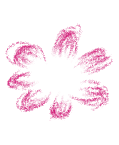
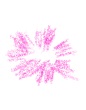
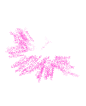
1. Calls a procedure to ask the user to input their first and last names, stores the names, and displays the number of characters in the names;



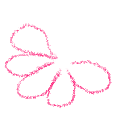
Hint (use readstring and writeint to display number of characters’)





stringname **BYTE** 50 DUP**(?) ; To be enter by the user**





**Mov edx,**offset stringname

**Mov ecx,**50 //maximum number of name characters

**call** readstring

Hint : when we use call readstring, total number of character enter by us, is display by eax automatically.



Hint: Before using push and pop your name, shift the offset of string name to ESI or EDI or EBP



1. Calls a procedure to reverse the order of names string using stack

Hint (use push pop and loop);

1. Calls a procedure to display the names in reverse order.

Program output:

Please type in your first and last names:

Muhammad Jawed

Number of characters in your name is=+14

Your name in reversed order is:

dewaJ dammahuM

**Code:**

TITLE reverse name

Include irvine32.inc

.stack 100h

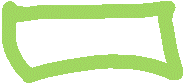
.data

lengthOfName BYTE 0

stringName BYTE 50 dup (0) ; store input name



reverseName BYTE 50 dup (0) ; store reverse name



st1 BYTE "Please type in your first and last names:",0

st2 BYTE "Number of characters in your name is= ",0



st3 BYTE "Your name in reversed order is:",0



.code

Main Proc

call inputName ; cal procedures

call reverseNameMethod

call PrintReverseName

Exit

Main endp

;--------------------------------------------------------------------------------

inputName Proc

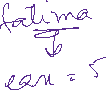
MOV edx,OFFSET st1 ; prompt user to input data



call writestring



call crlf



MOV edx,OFFSET stringName ; move offset of string to edx



MOV ecx,50 ; move size of string to ecx



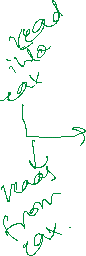
call readstring ; reads the input name



MOV edx,OFFSET st2 ; move offset of string to edx



call writestring



call crlf

call writedec ; displays the number of characters

MOV lengthOfName,al



call crlf



RET

inputName endp

;---------------------------------------------------------------------------------

reverseNameMethod PROC

movzx ecx,lengthOfName ; use length of name as counter

mov esi,0



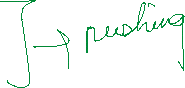
L1:



movzx eax,stringName[esi] ; get char from name string



push eax ; push char to stack



inc esi



LOOP L1



movzx ecx,lengthOfName ; use length of name as counter

mov esi, 0

L2:

pop eax ; pop char from stack

mov reverseName[esi], al ; store char in reverse name string



inc esi

LOOP L2



RET

reverseNameMethod ENDP

;---------------------------------------------------------------------------------

PrintReverseName PROC

mov edx,OFFSET st3 ; move offset of string to edx



call writestring



mov edx,OFFSET reverseName ; move offset of string to edx

call writestring



RET

PrintReverseName ENDP

;---------------------------------------------------------------------------------

End main

**Output:**

