**Assignment No-03**

|  |
| --- |
| **SUBJECT: MICROPROCESSOR LAB (MPL)**  **NAME: PRIYANKA SALUNKE** |
| **CLASS: SE COMP A ROLL NO.: F19111151** |
| **SEMESTER: SEM-IV YEAR: 2020-21** |
| **DATE OF PERFORMANCE: DATE OF SUBMISSION:** |
| **EXAMINED:** |

**Title:-** Find the largest of given numbers.

**Assignment Name: -** an X86/64 ALP to find the largest of given Byte/Word/Dword/64-bit numbers.

**Objective-**

* To understand the assembly language program
* To understand 64 bit interrupt.

**Outcome-**

* Students will be able to write code for how to find the largest of given
* Students will be able to understand different assembly language instruction.

### [Prerequisite](http://dictionary.reference.com/browse/prerequisite) -

System call of Unix for Assembly language Program.

**Hardware Requirement-**

Desktop PC

**Software Requirement-**

Ubuntu 14.04,

Assembler: NASM version 2.10.07

Linker: ld

**Introduction:-**

**Theory:**

**Algorithm:**

1. Start

2. Initialise section .data

3. Define variable for array,large

4. Using cmp instruction find larger number from array.

5. Display largest number

6. Terminate program using system call

6. Stop

**Conclusion: -** Hence we implemented an ALP find the largest of given array.

**Questions:-**

Q.1.Explain macro used with Example?

Q.2 Explain CMP instruction?

Q.3 Draw and explain TSS segment of 80386?

**Programs:-**

section .data

array db 10,2,13,24h,5

msg1 db 10,13,"Largest no in an array is:"

len1 equ $-msg1

section .bss

cnt resb 1

result resb 16

large resb 1

section .text

global \_start

\_start:

;display

mov Rax,1

mov Rdi,1

mov Rsi,msg1

mov Rdx,len1

syscall

mov byte[cnt],5

mov rsi,array

mov al,0

LP: cmp al,[rsi]

jg skip

xchg al ,[rsi]

skip: inc rsi

dec byte[cnt]

jnz LP

mov [large],al

call display

;exit system call

mov Rax ,60

mov Rdi,0

syscall

%macro dispmsg 2

mov Rax,1

mov Rdi,1

mov rsi,%1

mov rdx,%2

syscall

%endmacro

display:

mov bl,[large] ; store no in rbx

mov rdi,result ;point rdi to result variable

mov cx,02 ;load count of rotation in cl

up1:

rol bl,04 ;rotate no of left by four bits

mov al,bl ; move lower byte in dl

and al,0fh ;get only LSB

cmp al,09h ;compare with 39h

jg add\_37 ;if greater than 39h skip add 37

add al,30h

jmp skip1 ;else add 30

add\_37:

add al,37h

skip1:

mov [rdi],al ;store ascii code in result variable

inc rdi ; point to next byte

dec cx ; decrement counter

jnz up1 ; if not zero jump to repeat

dispmsg result,16 ;call to macro

ret

**Output:-**

