

**LAB 14**

*for*

*Computer Organization and Assembly Language*

|  |  |
| --- | --- |
| Lab Instructor(s) | Ayesha Inam |
| Section | A |
| Semester | Spring 2020 |

Roll No 17f8202

Name usman afzal

***Problem 1:***

*PART (1)*

***Real Address Mode:***

• In real mode processor could execute 16 bit instructions using 16 bit internal registers.

• The 16 bit instruction mode of is called the real mode.

***Protected Address Mode:***

• In protected address mode uses 32 bit access to the memory and uses 32 bit registers.

• Each program that is running has its own assigned memory locations, which are protected from conflict with other programs.

*PART (2)*

A byte 10

B sbyte -19

C sbyte ?

.code

Mov al, A

Neg b

Sub al,b

Mov c,al

C= -9

*PART (3)*

Mov eax, 0001h

Jmp eax

*PART (4)*

mov AL,0EH

add AL,72H

***Problem 5:***

include Irvine32.inc

.data

arr DB 8 DUP(? )

Rollnumber DB "ROLL# 18F0391", 0dh, 0ah

DB "LAST DIGITS: 91", 0

Byte1 DB 91

msg DB "ARRAY: ", 0

.code

main PROC

mov ecx, 8

xor eax, eax

mov edx, offset Rollnumber

call WriteString

call crlf

mov al, Byte1

mov esi, 0

L1:

shl al, 1

JC Store

JNC ZeroValue

Store :

mov arr[esi], 1

JMP Next

ZeroValue :

mov arr[esi], 0

Next :

add esi, type arr

loop L1

mov edx, offset msg

call Writestring

mov ecx, 8

mov esi, offset arr

Print :

mov edx, offset br1

call WriteString

mov al, [esi]

call Writedec

mov edx, offset br2

call WriteString

add esi, type arr

loop Print

call Readchar

exit

main endp

*end main*

***Problem 2(B):***

include Irvine32.inc

.data

array DD 26 DUP(? )

str1 BYTE 26 DUP(? )

var DD ?

spacemsg DB " ", 0

msg1 DB "FREQUENCY TABLE: ", 0

msg2 DB "ENTER STRING : ", 0

str2 DB "ABCDEFGHIJKLMNOPQRSTUVWXYZ", 0

.code

str\_in PROC

mov edx, offset msg2

call WriteString

mov ecx, 26

mov edx, offset str1

call ReadString

call crlf

ret

str\_in endp

Get\_Frequencies PROC

cld

mov edi, offset array

mov ecx, 26

mov ebp, offset str2

L1 :

mov var, ecx

mov ecx, 26

mov al, [ebp]

mov edx, 0

mov ebp, offset str1

L2 :

cmp al, [ebp]

JNZ next

inc dl

next :

inc ebp

loop L2

push eax

mov eax, 0

mov al, dl

stosd

pop eax

inc ebp

mov ecx, var

loop L1

ret

Get\_Frequencies endp

main PROC

xor eax, eax

xor ebx, ebx

xor ecx, ecx

xor edx, edx

call str\_in

call Get\_Frequencies

mov esi, offset array

mov ecx, 26

mov edx, offset msg1

call WriteString

mov edx, offset spacemsg

Print :

mov eax, [esi]

call Writedec

call Writestring

loop Print

call Readchar

exit

main endp

*end main*

***Output:***

