All Labs CS401 P- Copy - for better preparation

Introduction to assembly language (Virtual University of Pakistan)

Lab 1

Assembly language programming is taught by practicing code over **"NASM"** assembler and debugger.

So, it is required for every student to learn that how to install assembly language assembler and debugger.

32bit and 64bit Operating system does not support "nasm "assembler directly, so DOSBOX is introduced that is an emulator of the DOS operating system that will provide you compatibility of NASM on all windows platforms.

This LAB will give student a complete guideline that how to install nasm and DOSBOX

Mechanism to Conduct Lab:

Students and teacher communicate through google meet. Students will write code using Notepad or Programmer's Notepad and will share code and screen output.

Nasm: https://vulms.vu.edu.pk/Courses/CS401/Downloads/AssmSoft.zip

DosBOX: http://sourceforge.net/projects/dosbox/files/dosbox/0.74-2/DOSBox0.74-2-win32-installer.exe/download

Programmers Notepad: https://github.com/simonsteele/pn/releases/download/v2.4.2/portable-pn2421440.zip

Lab2

You are required to write assembly language code in which you will implement loop:

- As there are no loops available in assembly language so will write it manually.
- Declare one variable as ‘total’.
- Initialize bx = 5, ax = 0, cx = 10
- You are required to use jump and labels to implement loop.
- Loop will run 10 times and subtract 1 from cx at every iteration.
- At each iteration of loop the value of bx will be added into ax.
- At the end of program, the value of ax will be transferred to ‘total’.

Run the program in AFD and verify that your program is running properly.

Solution

```
[org 0x0100]
mov bx,5
mov cx,10
mov ax,0
l1:
add ax, bx
sub cx, 1
jnz l1
mov [total], ax
mov ax,0x4c00
int 0x21
total: dw 0
```

Lab3 Problem Statement

Write an Assembly language program that finds the biggest digit of your VUID

Instructions

- Store your VUID (only numbers) in the array of digits in the memory
- Implement labels and the jumps
- Declare a variable in memory named as ‘max’ and at the end of the program, store the biggest digit of your VUID in that variable.
- Use your own VUID
- Example: BC191234567 is your VUID then, biggest digit is 9.

Solution

```
[org 0x0100]
array_nums: db 1,2,3,4,5,6,7,8,9
max: db 0
jmp start
findBiggest:
mov ax, [bx] ; first number to ax
mov cx, 0
maxvalue: cmp ax, [bx] ; find the maximun number
jge maxloop ; if greater or equal number
mov ax, [bx] ; maximum number was larger
maxloop:
add bx, 2 ; advance bx to next index
loop maxvalue
```

```
ret
start: mov bx, array_nums
mov ax,0
mov cx, 15
call findBiggest
mov [max], ax
mov ax, 0x4c00 ; terminate program
int 0x21
```

Lab4 Problem Statement

Instructions

- Store your VUID (only numbers) in the array of digits in the memory
- Declare a variable named as ‘sum’ and initialize it with 0
- Initialize all the registers
- Set the counter to numbers of digits of your VUID
- Implement the summation logic to sum all the numbers (use labels and jumps to implement the loop)

Note

- Use your own VUID
- Example : BC191234567 is your VUID then, then sum of the VUID is 38

Debug the program in AFD to confirm that program is running properly.

```
[org 0x100]
        mov al,0
        mov di,0
        mov cx,9
        mov bl,0
summation:      add al,[vuid+di]
        inc di
        dec cx
        jnz summation
        mov [sum],al
        mov ax, 0x4c00
        int 0x21
vuid: db 1,2,3,4,5,6,7,8,9
sum: db 0
```

LAB 5

Make Binary of you Roll Number first and then use this binary converted Roll number in program by storing this binary in AX register.

Write a program in Assembly Language to swap every pair of bits in the AX register

Solution

```
[org 0x0100]
mov ax,1001100110011001b
mov bx,0
mov dx, 0 ; initialization
mov bx, ax ; get a copy of ax
and bx, 0011001100110011b ; bx now has 01, 45, 89, 1213 bits
rol bx, 2
mov dx, ax ; get a copy of ax
and dx, 1100110011001100b ; dx now has 23, 67, 1011, 1415 bits
ror dx,2
mov cx, 0
mov cx, bx
or cx, dx ; ORing of bx and dxwill produce desired result
mov ax, cx ; ax now has the result
end: mov ax, 4c00h
int 21h
```

Lab 6

You are required to write an assembly language program which will sort array using bubble sort, the sorting must be signed.

- Following numbers must be sorted:
  -15, -30, -12, -50, 10, 20, 55, 40, 7, 0
- After sorting the numbers, the numbers must be in the following order:
  -50, -30, -15, -12, 0, 7, 10, -20, 40,55
- You will store the sorted numbers in the same variable(array) declared earlier for unsorted numbers.
- Hint: You can use different type of jumps to sort the array.
- JE, JNE JZ JNZ

Solution:

```
[org 0x0100]
jmp start
data: dw -15, -30, -12, -50, 10, 20, 55, 40, 7, 0
swap: db 0
start: mov bx, 0 ; initialize array index to zero
mov byte [swap], 0 ; rest swap flag to no swaps
loop1: mov ax, [data+bx] ; load number in ax
cmp ax, [data+bx+2] ; compare with next number
jle noswap ; no swap if already in order
mov dx, [data+bx+2] ; load second element in dx
mov [data+bx+2], ax ; store first number in second
mov [data+bx], dx ; store second number in first

mov byte [swap], 1 ; flag that a swap has been done
noswap:
add bx, 2 ; advance bx to next index
cmp bx, 18 ; are we at last index
jne loop1 ; if not compare next two
cmp byte [swap], 1 ; check if a swap has been done
je start ; if yes make another pass
mov ax, 0x4c00 ; terminate program
int 0x21
```

Lab 7

You are required to write an assembly language program which will print the hexadecimal representation of the numbers included in your Roll number on screen.

- Write one subroutine to clear the screen.
- Write another subroutine to print the number.
- Store numbers from your VUID to any register.
- Then push the VUID to stack.

Solution:

```
[0x0100]
jmp start
; subroutine to clear the screen
clrscr: push es
push ax
push di
mov ax, 0xb800
mov es, ax ; point es to video base
mov di, 0 ; point di to top left column
nextloc: mov word [es:di], 0x0720 ; clear next char on screen
add di, 2 ; move to next screen location
cmp di, 4000 ; has the whole screen cleared
jne nextloc ; if no clear next position
pop di
pop ax
                                pop es

ret
; subroutine to print a number at top left of screen
; takes the number to be printed as its parameter
printnum: push bp
mov bp, sp
push es
push ax
push bx
push cx
push dx
push di
mov ax, 0xb800
mov es, ax ; point es to video base
mov ax, [bp+4] ; load number in ax
mov bx, 16 ; use base 10 for division
mov cx, 0 ; initialize count of digits
nextdigit:
mov dx, 0 ; zero upper half of dividend
div bx ; divide by 10
add dl, 0x30 ; convert digit into ascii value
```

```
push dx ; save ascii value on stack
inc cx ; increment count of values
cmp ax, 0 ; is the quotient zero
jnz nextdigit ; if no divide it again
mov di, 0 ; point di to top left column
nextpos: pop dx ; remove a digit from the stack
mov dh, 0x07 ; use normal attribute
                mov [es:di], dx ; print char on screen

add di, 2 ; move to next screen location
loop nextpos ; repeat for all digits on stack
pop di
pop dx
pop cx
pop bx
pop ax
pop es
pop bp
ret 2
start:
 call clrscr ; call the clrscr subroutine
mov ax, 123456789
push ax ; place number on stack
call printnum ; call the printnum subroutine
mov ax, 0x4c00 ; terminate program
                                int 0x21
```

LAB 8

Write a that's program takes two strings from the user and concatenates the two strings into one and displays it.

For concatenation the second string is copied at the end of the first string.

The first string array should have enough memory to accommodate the length of the second string.

LAB 9

You are required to print your VU ID on the console screen using BIOS Video Services.

```
; print string using bios service
[org 0x0100]
jmp start
message: db 'Student Name'
start: mov ah, 0x13 ; service 13 - print string
mov al, 1 ; subservice 01 – update cursor
mov bh, 0 ; output on page 0
mov bl, 7 ; normal attrib
mov dx, 0x0A03 ; row 10 column 3
mov cx, 11 ; length of string
push cs
pop es ; segment of string
mov bp, message ; offset of string
int 0x10 ; call BIOS video service
mov ax, 0x4c00 ; terminate program
int 0x21
```

LAB 10

Write program that get terminated when you will press T from your keyboard.

Solution

```
attempt to terminate program with T that hooks ke
yboard interrupt
[org 0x0100]
jmp start
; keyboard interrupt service routine
kbisr: push ax
push es
mov ax, 0xb800
mov es, ax ; point es to video memory
in al, 0x60 ; read a char from keyboard port
cmp al, 0x2a ; is the key left shift
jne nextcmp ; no, try next comparison
mov byte [es:0], 'L' ; yes, print L at top left
jmp nomatch ; leave interrupt routine
nextcmp: cmp al, 0x36 ; is the key right shift
jne nomatch ; no, leave interrupt routine
mov byte [es:0], 'R' ; yes, print R at top left
nomatch: mov al, 0x20 out 0x20, al ; send EOI to PIC
start: xor ax, ax
```

```
mov es, ax ; point es to IVT base
cli ; disable interrupts
mov word [es:9*4], kbisr ; store offset at n*4
mov [es:9*4+2], cs ; store segment at n*4+2
sti ; enable interrupts
l1: mov ah, 0 ; service 0 – get keystroke
int 0x16 ; call BIOS keyboard service
cmp al, 27 ; is the Esc key pressed
jne l1 ; if no, check for next key
mov ax, 0x4c00 ; terminate program
int 0x21
```

LAB 11

Write a program to make an asterisks travel the border of the screen, from upper left to upper right to lower right to lower left and back to upper left indefinitely, making each movement after one second.

Solution:

```
; program to move astericks
[org 0x0100]
jmp start
seconds: dw 0
timerflag: dw 0
oldkb: dd 0
; keyboard interrupt service routine
kbisr: push ax
in al, 0x60 ; read char from keyboard port
cmp al, 0x2a ; has the left shift pressed
jne nextcmp ; no, try next comparison
cmp word [cs:timerflag], 1; is the flag already set
je exit ; yes, leave the ISR
mov word [cs:timerflag], 1; set flag to start printing
jmp exit ; leave the ISR
nextcmp: cmp al, 0xaa ; has the left shift released
jne nomatch ; no, chain to old ISR
mov word [cs:timerflag], 0; reset flag to stop printing
jmp exit ; leave the interrupt routine
```

```
nomatch: pop ax

jmp far [cs:oldkb] ; call original ISR

exit: mov al, 0x20

out 0x20, al ; send EOI to PIC

pop ax

iret ; return from interrupt

; timer interrupt service routine

timer: push ax

cmp word [cs:timerflag], 1 ; is the printing flag setjne skipall ; no, leave
the ISR

inc word [cs:seconds] ; increment tick count

push word [cs:seconds]

call printnum ; print tick count

skipall: mov al, 0x20

out 0x20, al ; send EOI to PIC

pop ax

iret ; return from interrupt

start: xor ax, ax

mov es, ax ; point es to IVT base

mov ax, [es:9*4]

mov [oldkb], ax ; save offset of old routine

mov ax, [es:9*4+2]

mov [oldkb+2], ax ; save segment of old routine

cli ; disable interrupts

mov word [es:9*4], kbisr ; store offset at n*4

mov [es:9*4+2], cs ; store segment at n*4+2

mov word [es:8*4], timer ; store offset at n*4
```

```
mov [es:8*4+2], cs ; store segment at n*4+

sti ; enable interrupts

mov dx, start ; end of resident portion

add dx, 15 ; round up to next para

mov cl, 4

shr dx, cl ; number of paras

mov ax, 0x3100 ; terminate and stay resident

int 0x21
```

LAB 12

Write an Assembly program to underline each character of the screen font using BIOS video services.

Solution:

```
; put underlines on screen font
[org 0x0100]
jmp start
font: times 256*16 db 0 ; space for font
start: mov ax, 0x1130 ; service 11/30 – get font info
mov bx, 0x0600 ; ROM 8x16 font
int 0x10 ; bios video services
mov si, bp ; point si to rom font data
mov di, font ; point di to space for font
mov cx, 256*16 ; font size
push ds
push es
pop ds ; ds:si to rom font data
pop es ; es:di to space for font
cld ; auto increment mode
rep movsb ; copy font
push cs
pop ds ; restore ds to data segment
mov si, font-1 ; point si before first char
mov cx, 0x100 ; total 256 characters
```

```
change: add si, 16 ; one character has 16 bytes
mov byte [si], 0xFF ; change last line to all ones
loop change ; repeat for each character
mov bp, font ; es:bp points to new font
mov bx, 0x1000 ; bytes per char & block number
mov cx, 0x100 ; number of characters to change
xor dx, dx ; first character to change
mov ax, 0x1110 ; service 11/10 – load user font
int 0x10 ; bios video services
mov ax, 0x4c00 ; terminate program
int 0x21
```

LAB 13

You are required to write a TSR program that will print total number of characters in your name (including spaces within complete name) on top left corner of screen once you will press first letter of your name from keyboard.

Solution:

```
; Example student name : Muhammad Rizan Mir
; Name length : 18
[org 0x0100]
jmp start
oldisr: dd 0 ; space for saving old isr
; keyboard interrupt service routine
kbisr: push ax
push es
mov ax, 0xb800
mov es, ax ; point es to video memory
in al, 0x60 ; read a char from keyboard port
cmp al, 0x32 ; check if m has been pressed
jne nomatch
mov byte [es:0], '1' ; yes, print 1 at first column
mov byte [es:2], '8' ; yes, print 8 at second column
;jmp exit ; leave interrupt routine
nomatch: pop es
pop ax
jmp far [cs:oldisr] ; call the original ISR
exit: mov al, 0x20
out 0x20, al ; send EOI to PIC
```

```
pop es
pop ax
iret ; return from interrupt
start: xor ax, ax
mov es, ax ; point es to IVT base
mov ax, [es:9*4]
mov [oldisr], ax ; save offset of old routine
mov ax, [es:9*4+2]
mov [oldisr+2], ax ; save segment of old routine
cli ; disable interrupts
mov word [es:9*4], kbisr ; store offset at n*4
mov [es:9*4+2], cs ; store segment at n*4+2sti ; enable interrupts
mov dx, start ; end of resident portion
add dx, 15 ; round up to next para
mov cl, 4
shr dx, cl ; number of paras
mov ax, 0x3100 ; terminate and stay resident
int 0x21
```

LAB 14

You are required to write a TSR program that will print characters of your name on new line once you press ‘p’ from keyboard.

LAB 15,16

Write a program that will transfer Student name to another Computer using serial port. You will type on your name on command prompt of one computer and it will be displayed on 2nd computer's command prompt.