National University of Computer and Emerging Sciences

Lahore Campus

COAL (EE2003)

Computer organization and assembly language

Date: December 27th 2024

Course Instructor(s)

AA,AA,SF,SI,SM

Final Exam

Total Time (Hrs):	**3**
Total Marks:	**90**
Total Questions:	**4**

Solution _____ _____ _____

Roll No Section Student Signature

Instructions: Attempt all questions. It is an open book exam. Only textbook *"Assembly Language Programming Lecture Notes by Bilal Hashmi"* is allowed. ANY NOTES OR ATTACHMENTS ARE NOT ALLOWED. Attach question paper with answer book.

ATTEMPT QUESTION 1, 2 and 3 ON THIS QUESTION PAPER, SOLUTION ON ANSWER SHEET WILL NOT BE MARKED. Attempt question 4 on answer sheet.

CLO # 2: Describe the working of important x86 assembly primitives, including arithmetic, branching, bit manipulation, addressing modes and interrupt handling.

Q1: [15 marks] Short Questions.

I.	[3 Marks] Replace the following code with just one assembly line code. L1 is a label placed in the code.	dec cx Jnz l1	loop L1
II.	[3 Marks] Write the final value of AX after execution of following statement?	mov ax, 0x8000 sar ax, 1	AX = 0xC000

III.	[3 Marks] What will be the final value of CX after executing this piece of code? Assume Str1 and str2 are two strings in code.
	CX = 0
		mov si, str1
		mov di, str2
		mov cx, 5
		rep movsb	

IV. [6 Marks]

Initial value of SP= 0xFFFE. Value of CX, DX and SP after execution of following code will be:
CX = 0x0005
DX = 0x000A
SP = 0xFFFC
	[org 0x0100]
	jmp start
	label1:
	mov cx,5
	mov dx, 10
	jmp end123
	function1:
	push label1
	ret
		start:
		call function1
		mov cx, 15
		mov dx, 20
		
		end123:
		mov ax, 0x4c00
		int 0x21

CLO # 1: Demonstrate the basic concepts of computer organization including CPU, memories, and input/output and explain their purposes and interactions.

Q2: [30 marks] Short Questions.

National University of Computer and Emerging Sciences
Lahore Campus

I. **[5 Marks]** Given, BX=00FFh, CS = 1001h, DS = 3333h, SS = 2526h, IP = 1332h, SP = 1100h, and DI = 0020h. What are the effective and physical addresses generated by the following memory access?
Memory Access: [cs: bx + di]

Effective Address: 0x011F **Physical Address:** 1001 * 10 + 011F = 0x1012F

II. **[5 Marks]** Following code is trying to make whole display screen blink. Add the missing instruction such that it successfully accomplishes this task using MASK Operation. Assume that there is no blinking character in the screen originally.

```
mov ax, 0xb800          cld
mov es, ax              loop1:                    or
mov ds, ax              lodsw
xor di, di              xor al, 0x80 ; Add missing instruction here
xor si, si              stosw
mov cx, 2000            loop loop1
```

III. **[5 Marks]** What will be the final value of memory location result after the following code executes?
result = 0x0020

```
[org 0x0100]
mov ax, 0x2000
mov ds, ax
mov bx, [num1]
add bx, 0x0020
mov [result], bx
mov ax, 0x4c00
int 0x21
num1: dw 0x0010
result: dw 0x0000
```

IV. **[5 Marks]** What will be the value of BX after the following code executes?
BX = 15

```
[org 0x0100]
mov cx, 5
mov si, array
xor ax, ax
xor bx, bx
loop_start:
lodsw
add bx, ax
loop loop_start
mov ax, bx
mov ax, 0x4c00
int 0x21
array: dw 1, 2, 3, 4, 5
```

V. **[10 Marks]** The code given below is subtracting two 64 bit numbers and storing the result in the result label. You have to complete the code. Num1 = 0x880060ff4000ffff, Num2= 0x850f0000400018000

```
[org 0x0100]
jmp start
num1: dd 0x880060ff,0x4000fff
num2: dd 0x850f0000,0x40018000
result: dq 0x0

start:
mov ax,[num1+....4....]
mov bx,[num2+....4....]
sub ax,bx
mov [result+....0....],ax

mov ax,[num1+....6....]
mov bx,[num2+....6....]
sbb ax,bx
mov [result+....2....],ax
mov ax,[num1+0]
mov bx,[num2+0]
sbb ax,bx
mov [result+....4....],ax

mov ax,[num1+2]
mov bx,[num2+2]
sbb ax,bx
mov [result+....6....],ax
```

CLO # 3: Apply the knowledge of Intel x86 architecture to develop moderately complex and well-modularized assembly programs.

Q3 [5x3 = 15 Marks]: An Elaborate Multitasking example 10.2 is provided in the book. Make the following changes to the example. *In your solution only following is required:*

Lines of code which need to be removed
Lines of code which need to be modified along with modification
Lines of code which need to be added.

No credit will be given for anything else.

I. **Each task should run for 1 second before switching to the new task.**

Add code at start of timer ISR.

```
pushi ax
inc word [cs: tickcout]
cmp word [cs:- bickcount],18
je next
mov al, 0x20
out 0x20, al
```

```
tickcount: dw 0
```

```
pop ax
iret
next: pop ax
      mov word [cs: bickcount],0
```

II. **In example 10.2, if 32 tasks are initialized, the tasks are running in the following order 0,31,30,29,------3,2,1, 0, and so on. You have to change the sequence of tasks such that the tasks should be run in the following fashion: 0,1,2,3,4,5,6, -------,31, and back to the 0th task.**

Remove Lines 119 to 123 in initpcb
Replace with the following

```
mov word [pcb+bx+28], 0
mov si, [nextpcb]
dec si
shl si, 5
mov ax, [nextpcb]
```

```
mov [pcb+si+28], ax
```

III. **The stack size for each task is increased from 256 words to 512 words.**

Change Line 10 to following

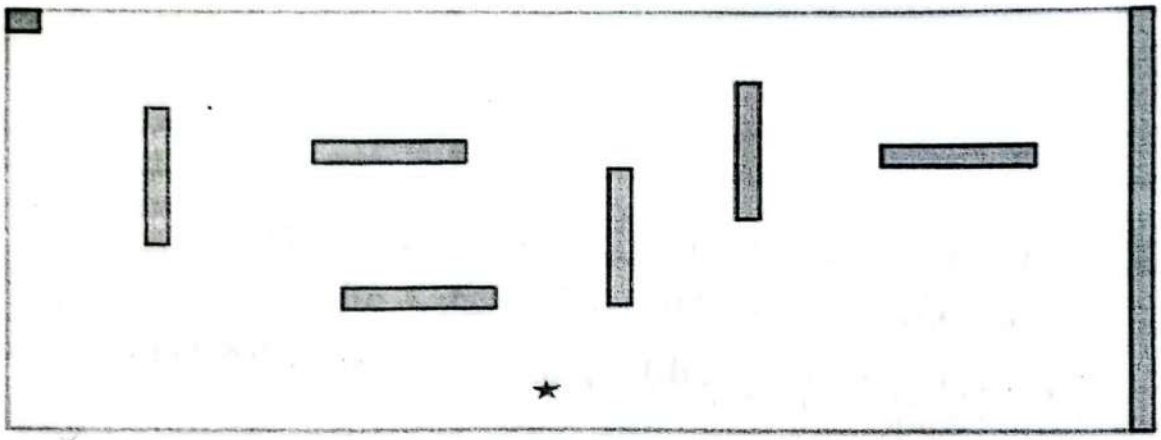
```
stackR: dw times 32*512 dw 0
```

Change Initpcb line 109-111

```
mov cl, 10
shl si, cl
add si, 512*2+stackR
```

CLO # 3: Apply the knowledge of Intel x86 architecture to develop moderately complex and well-modularized assembly programs.

Q4 [30 Marks]: You are required to implement a game "MovingStar" in 8088 Assembly Language with the following requirements:



Screen 25*80

1. **[Initial Screen – 2+2+2+2 Marks]** You have to do the following to initialize the screen.
   a. Clear the screen.
   b. Place the obstacles using *"place_obstacles"* procedure. *place_obstacles* subroutine is a built-in thing you don't need to write it, just call the function where required. It will initialize all the Green (0x2220) obstacles on the screen as shown in the picture. The full right boundary is also a green obstacle.
   c. Place the goal at the top left corner of the screen. The goal is a space with a Red background (0x4420).
   d. Place the player, a blue color asterisk character (* = '0x2A') at the center of the last row.
2. **[Game Mechanics– 7+7+4 Marks]**
   a. After every 2 timer interrupts, the player moves one step in the direction of the last pressed arrow key. Initial Direction will be Rightward.
   b. Use the keyboard arrow keys to move the player in a specific direction.
      Up: Arrow Key ↑ (Scan Code 4Ah)
      Right: Arrow Key → (Scan Code 4Dh)
      Due to shortage of time we are not handling other directions.
   c. If the player collides with an obstacle or the right-most column, the game ends.
3. **[Safe Termination – 4 Marks]:** After the end of the game, other programs should run smoothly on DOS Box.

Important Instructions:

- If you want to use any function from textbook examples, simply call it PROPERLY. YOU DO NOT NEED TO RE-WRITE IT.
- If you want to use the code to hook or unhook Keyboard or Timer. Simply write:
  "---Code to hook Timer comes here---"
  "---Code to unhook Timer comes here---"
  You do not need to re-write the code but do clearly mention this statement.
- Properly comment your code and write functions where needed. Freely use global variables, if required.