

National University of Computer and Emerging Sciences, Lahore Campus



Course Name: Computer Organization and Assembly Language
Program: BS(Computer Science)
Duration: 3 hours
Paper Date: 28-5-2018
Section: ALL
Exam Type: FINAL EXAM

Course Code: EE213
Semester: Spring 2018
Total Marks: 80
Weight: 45%
Page(s): 10

Student : Name: _____ Roll No. _____ Section: _____

- Instruction/Notes:**
1. Exam is Open book, Open notes.
 2. Properly comment your code.
 3. Write your answer in the space provided. You **can take extra sheets BUT they WONT BE ATTACHED WITH THE QUESTION PAPER OR MARKED.**
 4. No need to copy code from book. If you need any code/part of code, just mention the line numbers and page no.

Question1. [2x10 Marks] Tick the correct answer. No cutting/over-writing. Cutting and over-writing will give you zero marks. Any answer not clear will also give you zero marks.

- What will be the value of overflow flag after the following arithmetic statements?
mov ax, 0xD18A
sub ax, 0xFFFF
a. 0
b. 1
- While calculating the physical address by adding segment and effective addresses, a 21-bit result (or a carry-out) would mean which of the following while the logical address was only of 16-bit?
a. There is a segment wraparound but no memory wraparound
b. There is a memory wraparound but no segment wraparound
c. There is a memory and a segment wraparound
d. There is no wraparound at all
- Which of the following will result in a syntax error?
a. mov ax, [cs:si]
b. mov ax, [cs:di]
c. mov ax, [cs:sp]
d. mov ax, [cs:bx]
- Which of the following is a true statement about Far jump?
a. Far jump has a relative address of 20-bits
b. Far jump has a relative address of 16-bits
c. Far jump is an intra-segment jump
d. Far jump is an absolute jump and not position relative
- Ascii values of all the lower case letters have a difference of 0x20 with the ascii of their respective upper case letter. For example ascii of small 'a' is 0x61 and capital 'A' is 0x41. Which of the following codes is converting lower case ascii value to upper case?

- a. `mov al, 'a'`
`and al, 11011111b`
- b. `mov al, 'a'`
`or al, 11111110b`
- c. `mov al, 'a'`
`xor al, 11001111b`
- d. none of the above

vi. The programmable interrupt controller is required to:

- a. handle one interrupt request
- b. **Handle one or more interrupt requests at a time**
- c. handle one or more interrupt requests with a delay
- d. handle no interrupt request

vii. Which of the following is not a true statement about multi-tasking?

- a. The EOI signal that timer sends cannot be removed
- b. **After all 32 tasks are created task 0 will always get twice as much time as any other task**
- c. The line number parameter is necessary for my task to execute properly
- d. Timer does not know about total tasks in the list

viii. `Lds si, [bp-2]` will load the following values in registers:

- a. si will contain [bp+2] and DS will have value of [bp]
- b. si will contain [bp-2] and DS will have value of [bp-4]
- c. si will contain [bp+2] and DS will have value of [bp-2]
- d. **si will contain [bp-2] and DS will have value of [bp]**

ix. Which of the following statement has the same effect as these two valid instructions?

- `dec cx`
`jnz l3`
- a. `rep stosb`
 - b. `retf`
 - c. **`loop l3`**
 - d. `iret`

x. Which interrupt is generated as a result of divide overflow error?

- a. **Int 0**
- b. Int 1
- c. Int 2
- d. Int 3

Question2. [20 Marks] Write an assembly program to find the minimum length of a string in an array of 20 strings given in DS. All strings are null terminated. The program places the minimum length found in dx register. You have to do this question using string instructions only but you cannot use `cmps` or `cmpsb`.

For example:

`arraystr: db 'hello',0,'class room',0,'is',0,'language',0`

For this array dx will have value 2 at the end of program.

```
[org 0x0100]
```

```
mov al, 0  
push ds  
pop es
```

```
mov di, arraystr  
mov cx, 0xffff  
mov dx, 0xffff  
mov si, 21                ;size is 20 strings
```

```
l1:  
  
    dec si  
    cmp si, 0  
    je exit  
  
    mov bx, cx  
    repne scasb  
    sub bx, cx  
    dec bx
```

```
cmp bx, dx  
jae l1  
mov dx, bx  
jmp l1  
exit:
```

```
mov ax, 0x4c00  
int 21h  
arraystr: db 'hello',0,'class room',0, 'is',0,'language',0
```

Question3. [20 Marks] In this question, you will write the keyboard and timer interrupt service routines to enable a simple game. The game uses a 1D board with six slots on the screen. These slots are numbered 0, 1, 2, ...5 and occupy the six left-most cells of the top line of the screen. Therefore, slot 0 is at 0xB800:0, slot 1 is at 0xB800:2, slot 2 is at 0xB800:4, and so on. The game is played between two players, represented by symbols X and Y, who take alternate turns. Player 1 always goes first. You may assume that, initially, the board contains empty spaces.

The program at all times keeps track of a `current_slot_number` (between 0 and 5) to know which slot is currently selected.

The keyboard ISR enables the following functionality:

- If the current player (Player 1 or Player 2) presses the L key the `current_slot_number` is decremented by 1, and if he presses the R key the `current_slot_number` is incremented by 1. However, the `current_slot_number` is never allowed to go below 0 or above 5.
- If it is Player 1's turn and he presses enter, his symbol X is printed at the current slot (stored in `current_slot_number`) on the screen, and the turn switches to Player 2. If the current slot already contains a symbol it is simply overwritten.
- If it is Player 2's turn and he presses enter, his symbol Y is printed at the current slot (stored in `current_slot_number`) on the screen, and the turn switches to Player 1. If the current slot already contains a symbol it is simply overwritten.
- At the end of the ISR send EOI and IRET.

Note: the Scan Codes of L and R keys are: 0x46 and 0x33 respectively.

The Timer ISR performs the following task:

- Check if any three consecutive slots on the board contain the same symbol (i.e. either all three are X or all three are Y). If yes, print the winner and terminate the program. In this case, it should print either 'Player 1 wins' or 'Player 2 wins'.
- If no three consecutive slots on the board contain the same symbol, simply chain to the original timer ISR. Assume that the IP and CS of the original timer ISR are already stored at `oldISR` and `oldISR+2` in your program respectively.

You are only required to write the timer and keyboard ISRs, and declare the necessary variables in your program. You do not need to write the code to hook the interrupts or create TSR etc. We will not mark any redundant code.

Turn: dw 1

Position: dw 0xb800

Name: db 'player'

Win: db 'wins'

```

kbisr:  push a      push 0x60
        pop cs
        in  al, 0x60
        cmp al, 0x46
        jne right
        cmp [position], 0xb800
        je  exit
        sub [position], 2
        jmp exit

```

;(44)

```

right:  cmp  al, 0x33
        jne enter
        cmp [position], 0xb80A
        je  exit
        add [position], 2
        jmp exit

```

```

enter:  cmp  al, 0x1C
        jne exit
        mov  ah, 0x07
        cmp  [turn], 1
        jne turn2
        mov  al, 'x'
        mov  position, ax
        mov  [turn], 2
        jmp exit

```

```

turn2:  mov  al, 'y'
        mov  [position], ax
        mov  [turn], 1

```

```

exit:   mov  al, 0x20
        out  0x20, al
        pop 9

```

① check for left
dec position (2)

② check for Right
inc position (2)

③ Check enter (1)
+ exit (1)

④ put mark
(2)

10

⑤ manage
turn
(2)

Timer :

```

push a
push cs
pop ds
mov bx, 0xb80a
mov cx, 4

```

l1:

```

mov ax, [bx]
mov si, bx
add si, 2
cmp ax, [si]
jne next
mov si, [bx]
sub si, 2
cmp ax, [si]
jne next
push ax
call print
jmp exit
next:
add bx, 2
loop l1

```

next:

exit: jmp far [cs:oldisr]

print:

```

push a
push 10
push 0
push name
push 6
call printstr

push 10
push 8 [bp+4], 'X'
cmp [bp+4], 'X'
mov [bp+4], 'X'
push [bp+4]
jmp cont
call printnum
push 10
push 11

```

Cont,

① loop to find seq ⑥

② print correct player wing ②

loop ③ chaining ②

⑩

```

push wtr
push 4
call printstr
pop a
ret

```

Question4. [10+ 10 Marks] Consider elaborate multitasking example 10.2 for the following two questions. Remember these questions are ‘independent’ of each other.

Part A) The current scheduling algorithm implemented in example 10.2 executes each process for a fixed period of time, suspends it and then starts next process in the list. Your task is to change the scheduling algorithm to “priority based scheduling”. Whenever a task is created it is assigned a priority number between 1 and 4 (4 meaning highest priority). The scheduler works such that it allows every next task to run for thrice as much timer ticks as its priority. So for example, if a task has priority 4, the scheduler allows it to run for $4 \times 3 = 12$ timer ticks. When the task has run for its assigned number of timer ticks the scheduler suspends the task (just like in example 10.2) and moves on to the next task. The scheduler keeps doing this for all tasks infinitely.

For task 0, its execution time is 1 timer tick only and has no priority.

For this part you can assume that a function “generate_random_priority” is already defined and given to you which returns a priority number in ax register.

Note: Do not copy the whole multitasking code. This will earn you zero marks. Just modify the changes required in the code. Your code should be well-commented and it should explain the changes made to the code.

Tickcount: dw 0

In initpcb:

After line 118:

Call generate_random_priority

Mov cl, 3

Mul cl

Mov [pcb+bx+30], ax

In Timer:

Push ds

Push bx

Push cx

Pop ds

Mov bx, [current]

Cmp bx, 0

Je skip1

Inc word[tickcount]

Shl bx, 5

Push ax

Mov ax, [tickcount]

Cmp ax, [pcb+bx+30]

Je skip2

Mov al, 0x20

Out 0x20, al

Pop ax

Pop bx

Pop ds

Iret

Skip2:

pop ax

Skip1:

mov word [tickcount], 0

Shl bx, 5

Continue from line 140 till end

Part B) i) In example 10.2 whenever a new task is created its PCB is inserted at head of list i.e. after PCB 0. Change the insertion process in the code such that whenever initpcb wants to insert a new pcb in the list, it calls another sub-routine "insert_after". This sub-routine takes as parameter a pcb number and inserts the new pcb created after this pcb. So for example initpcb has created a pcb number 4. It calls insert_after and passes it as parameter pcb number 2. The sub-routine insert_after will insert pcb number 4 after pcb number 2 in the list. Note that parameter passed is via stack and is a valid pcb number already in the list. Make appropriate changes to the links connecting pcbs. All pcbs and tasks are created sequentially just like in book. Do not copy code from book. Write complete insert_after subroutine and mention clearly from where it will be called from initpcb.

After line 118 in initpcb

Push ax ;suppose ax has the pcb number after which to insert

Call insert_after

(omit line 119 to 122)

Insert_after:

Push bp

Mov bp, sp

Push a

Mov si, [bp+4] ;the pcb number

Shl si, 5

Mov di, [pcb+si+28] ;next of that pcb

Mov ax, [nextpcb] ;the new inserted

Mov [pcb+si+28], ax ;setting it as next

Mov si, [nextpcb]

Shl si, 5

Mov [pcb+si+28], di ;newpcb's next is now set

Pop a

Pop bp

Ret 2

Part B) ii) The tasks in example 10.2 run for an infinite number of time. Suppose that our scheduler wants to forcefully end a task by calling a sub-routine “delete_next_pcb”. This sub-routine takes as parameter a task/pcb number and removes the next of that pcb from the list. So for example, if delete_next_pcb is passed as parameter pcb 3, it removes the pcb number 2 (the next of pcb 3) from list and makes necessary adjustments to the links to update the list. You only need to write the sub-routine. The parameter is passed via stack.

Delete_next_pcb:

Push bp

Mov bp, sp

Push a

Mov si, [bp+4]

Shl si, 5

Mov di, [pcb+si+28] ;the next of pcb passed as parameter

Cmp di, 0

Je exit

Shl di, 5

Mov ax, [pcb+di+28] ;the next of pcb to be deleted

Mov [pcb+si+28], ax ;updating links

Exit:

Pop bp

Pop a

Ret 2

☺ GOOD LUCK! ☺