National University of Computer and Emerging Sciences, Lahore Campus

Course:	Computer Organization and Assembly Language	Course Code:	EE213
Program:	BS(Computer Science)	Semester:	Fall 2016
Duration:	180 Minutes	Total Marks:	60
Paper Date:	29-12-2016	Weight	45%
Section:	ALL	Page(s):	3
Exam:	Final	Roll No:	15L-4122

Instruction/Notes: Exam is Open book, Open notes.

Question 1: Short Questions. [5+5 Marks] (No partial marking)

i. The following piece of code starts executing at L1. How many times 'hello world' will be displayed on the display memory? Give reason for your answer.

```
; ISR_INT1 is hooked with Interrupt 1
ISR_INT1:
; It prints 'hello world' on display memory and updates print location
IRET
MyISR_80:
L1:pushf
pop ax
or ax , 0x0100
push ax
popf
mov ax, 0xEE43
mov dx , 0x5543
add ax, dx
IRET
```

ii. Every time we perform a shift of multiple bits, we lose a significant bit. For example doing SHR AX, 2 with AX = 0x0005; we will get a 0 bit in carry and loose the last bit. You have to write a program that right shifts the bits by amount stored in CL register; but instead of losing the least significant bit, save it in carry flag. For example, a right shift of 4 on AX with AX = 0x000A will save a 0 in carry flag and AX becomes 0x0000.
<u>You are not allowed to use any extra registers!</u>

More examples could be:

Before	Shift Amount	After
AX = 0x000E	3	AX = 0x0001, C = 0
AX = 0x0009	4	AX = 0x0000, C = 1

Question 2: [10 Marks]

We can represent 3 octal digits in a byte by applying 2–3–3 system. For instance, 11010110_2 is $\boxed{11}\ \boxed{010}\ \boxed{110}_2$ or 326_8 . Bit 6 and 7 determine the left octal digit (which can never have a value greater than 3), bit 3, 4, and 5 the

middle digit, and bit 0, 1, and 2 the right digit. The 16-bit binary value is represented in split octal by applying the 2–3–3 system to the high-order and low-order bytes separately.
Write a procedure splitOctal which converts a 16 bit binary value, passed to it as parameter, into its equivalent splitOctal value as a string of exactly six characters (where each character is one splitOctal digit). The procedure will have two parameters, passed through stack:
a. The word value
b. the address of the destination string
Ascii of 1 is 49 (decimal)/0x31 (hexa)

Question 3: [20 Marks]

The motherboard contains circuitry capable of turning a small speaker ON and OFF. By turning the speakeron and off at different rates, we can generate audio tones of various frequencies. For example by turning the speaker ON/OFF at the rate of 2*119 times per second we can generate a beep of frequency 10,000 Hz. DOS itself uses this technique to generate a short beep of a fixed frequency when the keyboard buffer is full, as a warning to stop typing. We can switch the speaker ON/OFF by writing a byte to parallel port 61h. The speaker in turned ON if the two least significant bits of this byte are set and OFF if both these bits are clear. Following is a snippet of code that switches the speaker ON,

```
inal,0x61 ;read from port 61h
or al,0x03 ;set the two LSBs
out 0x61,al ;switch the speaker on
```

Similarly you can write code to switch the speaker off. But the actual task is to toggle the speaker between ON/OFF at a particular frequency to generate the tone of that frequency. You might be thinking about using the timer interrupt (INT 8) for this purpose but as we saw normally the timer interrupt comes only at 18.2 times per second on average. But we can speed up the timer by changing the value of the timer divisor, this is called reprogramming the channel 0 (the channel for the timer that we have been using). To change the divisor at channel 0* we have to write to port 40h, here is a little subroutine that I wrote to do that, you can use this routine as such,

```
repch0:
push bp
movbp,sp
push ax
mov ax,[bp+4] ;the parameter contains the new divisor
out 0x40,al ;load lower word of the divisor
moval,ah
out 0x40,al ;load higher word of the divisor
;channel 0 reprogrammed
pop ax
pop bp
ret 2
```

For example, since the original timer frequency is 1.19 MHz, to get 10,000 timer ticks per second we change the divisor to 119. Now your job is to make a rudimentary piano with your keyboard (what a disgrace to the real piano!!!). The number keys 1-9 should produce beeps of frequency produced by 1000 timer ticks per second to 9000 timer ticks per second respectively. The working should be such that as long as one keeps a key pressed the tone of corresponding frequency should be heard and as one releases the key the tone should stop. Make your program a TSR.

Press Code of 1 - 0x02
Press Code of 2 - 0x03
Press Code of 3 - 0x04
Press Code of 4 - 0x05
Press Code of 5 - 0x06
Press Code of 6 - 0x07
Press Code of 7 - 0x08
Press Code of 8 - 0x09
Press Code of 9 - 0x0A

Question 4: [20 Marks]

Consider the scheduler code (timer interrupt) in Example 10.2. The current scheduling algorithm implemented is called Round Robin. It executes each process for a fixed period of time, suspends it and then starts another process next in the PCB. Your task is to change the scheduling algorithm to Shortest Job First (SJF). In SJF, the process whose execution time is the shortest executes first. Start off the kernel from the process with shortest execution time. During execution, the remaining execution time should constantly decrement. When the process is terminated (i.e. its execution time becomes 0), then the process with least execution time among the other processes initialized will execute next. If two processes have same execution time, then the process which comes first should run first. **Each process is given an execution time in seconds which is stored in PCB at the time of initialization. Assume that the execution time of Task 0 is infinite as in example 10.2.**

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.

☺💣 GOOD LUCK 💣☺