Name: ----------------------------------------------------------------

Registration No: --------------------------

Section: --------------

* It is an open book and open notes exam.
* Write neat and well-commented programs.
* You are not allowed to use any assembly instruction that is not taught in the class.

**Q1.** What is the value of AX register after the execution of the following pieces of code, where each part is independent of each other? **[5\*2 Marks]**

|  |  |
| --- | --- |
| 1. and ax, 0x5500   mov bx, ax  inc bx  xor ax,bx  AX=0001 |  |
| 1. nums: dw 0x0000, 0x1111, 0x2222, 0x3333, 0x4444, 0x5555, 0x6666, 0x7777, 0x8888, 0x9999   ; at this moment DS=1234 and ES=1235  mov ax,[es:nums]  Ax will have any random value. Assuming that org 100h is written at the start, then ax=0x8888. | |
| 1. mov ax,1   test ax,3  AX=0001 | 1. num: dw 0x1122, 0x3344   mov ah,0  mov al,[num+2]  AX=0044 |
| 1. mov al,0x33   mov [start+2],al  start: and ax,0x5500  Ax=0000 |  |

**Q2.** One way to divide two numbers is to subtract divisor from dividend unless the dividend is less than divisor. Counting the number of subtractions will give the quotient and the dividend left will give the remainder e.g. in order to divide 24 by 10, subtract 10 from 24 unless the answer is less than 10. This will give quotient equals to 2 as subtracting 10 once from 24 gives answer 14, which is greater than 10 and again subtracting 10 gives answer 4, which is less than 10. The remainder is 4 because the number left is 4.

Write an assembly code to divide two numbers in memory using the above algorithm and store the quotient and remainder back in the memory. **[10 Marks]**

**Solution**

|  |
| --- |
| [org 100h]  mov ax, [dividend]  xor cx, cx  check\_and\_subtract:     cmp ax, [divisor]     jl exit     sub ax, [divisor]     inc cx     jmp check\_and\_subtract  exit:     mov [rem], ax     mov [quotient], cx  mov ax, 4ch  int 21h  dividend: dw 5  divisor: dw 2  quotient: dw 0  rem: dw 0 |

**Q3.** A computer maintains a log of the I/O devices attached to it. After every two hours the log is updated, and it is verified, which devices attached to processor are still in use and which devices are not being used. Devices which are not used consecutively for 4 hours, (i.e. two logs) are shut down to save power.

Following is a sample data of 16 devices, where 1 means the device was in use and 0 means device was not in use.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **D1** | **D2** | **D3** | **D4** | **D5** | **D6** | **D7** | **D8** | **D9** | **D10** | **D11** | **D12** | **D13** | **D14** | **D15** | **D16** |
| First log | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 |
| Second Log | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |

Write an assembly program, that determines which of these devices should be shutdown to save power supply. **[10 Marks]**

For the above data, define labels in your code as follows:

firstLog: db 0x99, 0xDA

secondLog: db 0x52, 0xA4

deviceSD: db

(Put the output device numbers in the label: deviceSD, where each device number should be represented in a byte.)

**Solution**

|  |
| --- |
| [org 100h]  xor cx, cx  xor si, si  mov ax, [firstLog]  mov bx, [secondLog]  or ax, bx  check\_device:     inc cx     shl ax, 1     jc check\_end     mov [deviceSD + si], cx     inc si  check\_end:     cmp cx, 16     jb check\_device  exit:  mov ax, 4ch  int 21h  firstLog: db 0x99, 0xDA  secondLog: db 0x52, 0xA4  deviceSD: times db 16 0 |

    •☠ GOOD LUCK ☠•