SOME SAMPLE MIDTERM2 AND FINAL EXAM EXERCISES

3) [30 marks total]

Let us suppose that the Processor's register contents are the following (displayed in Hex)

```
AX = FFFF                    BX = 001C
CX = 0000                    DX = 0000
SP = 00FE                    BP = 0000
SI = 0000                    DI = 0000
DS = 2789                    ES = 277D
SS = 2792                    CS = 278D
IP = 0021
```

The following lines display a memory dump for a given area of our memory, like the ones given by our debugger:

```
2790:0000  01 83 F9 00 75 F4 CC 5B 5F C3 33 00 01
2790:000D  00 02 00 03 00 04 00 05 00 00 10 00 00
2790:001A  00 00 64 20 00 00 08 0E 00 00 20 39 00
2790:0027  10 00 00 00 10 00 00 00 00 64 20 00 00
2790:0034  69 17 00 00 E0 4B 00 02 00 00 00 20 E0
2790:0041  4B 00 20 E0 4B 00 20 E0 4B 00 20 08 0E
2790:004E  00 00 00 00 00 00 08 0E 00 16 00 43 56
2790:005B  20 2F 35 30 20 43 48 30 35 5C 53 55 42
2790:0068  53 32 2E 65 78 65 20 0D 8B 3E 72 0E 8B
2790:0075  5E 0A 89 1D 33 C0 89 46 FE 89 46 FC 8B
```

We want to execute the following instructions:

```
     mov   di, Var           ; (1) ____________________________________
     mov   dx, ax            ; creates copy of AX
CS1: mov   ax, [BX]          ; (2) ____________________________________
     sub   di, 1             ; ____________________________________
     add   bx, 2             ; ____________________________________
     add   cx, ax            ; add the value of ax into cx
     cmp   di, 0             ; ____________________________________
     jne   CS1               ; (3) ____________________________________
```

a) [5 marks]
Let us suppose that the Var value is 3 (Var is a 16-bit variable that has been defined earlier in the program). After executing the code fragment, which are the values of the following registers?

```
CX = ______(HEX)          DI = ______(HEX)                DX = ______(HEX)
```

b) [3 marks]
What are we doing with this code fragment?

c) [3 marks]
Fill-in the missing comments in the program, explaining what is each of the instructions is doing.

d) [3 marks]
Identify all the addressing modes used in the lines commented (1), (2) and (3) in the code fragment

Question 2 [15 marks] **Program the following exercises in Assembly**

a) [10 marks] Consider the problem of calculating the value of an exponential number 2^{power}. (Example : $2^3=8$). The pseudo-code for the recursive solution for this problem (where a subroutine calls itself) is given below. Provide the complete implementation of the following subroutine in Assembly code. Use the course policies for programming subroutines.

```
int exponential ( unsigned int power)
   if (power > 0)
   {
      twoToN = exponential (power-1)
      result = twoToN + twoToN // Yes, it's '+', not '*'
      if (overflow) {result = -1 }
      return result
   }
   else
      return -2
```

b) [5 marks] Write a main program that invokes this subroutine and displays the results. The main program must also recognize errors returned by the subroutine and print the corresponding error messages.

a) [5 marks] Explain briefly the main differences between Hardware and Software interrupts.

b) [7 marks] Explain, in detail, and step by step, what happens if we receive a Non Maskable Interrupt from an external device (for instance, a switch indicating an alarm in a power plant).

c) Recall the Keyboard ISR we discussed in class. The ISR stores the read characters in the global variable `KEYBOARD_CHARACTER` as follows:

…

```
; Convert the make code to ASCII
     LEA  BX , SCAN_TABLE
     XLAT
     CMP  AL , 0              ; some keys ignored !
     JZ   SEND_EOI

; Put ASCII encoded value in shared variable
     MOV  KEYBOARD_CHARACTER , AL
SEND_EOI:
```

...

The procedure `GET_CHAR` reads that character, as follows:

```
GET_CHAR PROC NEAR
          ; poll until char received from ISR
          ; check for "no data" value
          ; (1)
          CMP KEYBOARD_CHARACTER, 0FFH
          JZ  GET_CHAR
          ; (2)

          ; get ASCII character
          MOV  AL , KEYBOARD_CHARACTER
          MOV  KEYBOARD_CHARACTER , 0FFH
          RET
GET_CHAR  ENDP
```

i) [5 marks] Are the instructions in the bold part of a Critical Section? Justify your answer with detail.

ii) [3 marks] What happens in this case if we disable interrupts at (1) and enable them at (2) shown in the code on the previous page? Justify your answer.

The PC I/O Map

Parallel Output port (LEDs)	378h	Parallel Input port (Switches)	379h
PIC Command Register	20h	PIC Mask Register	21h
PIT Timer 0 Data Register	40h	PIT Control Register	43h
Keyboard Data Port	60h	Keyboard Control Port	61h

An instruction summary

Transfer instructions
MOV dest, src PUSH reg/mem POP reg/mem
IN ALorAX, imm8 IN ALorAX, DX
OUT DX, ALorAX OUT imm8, ALorAX

Arithmetic Instructions:
ADD dest, src SUB dest, src CMP dest, src
MUL reg DIV reg
INC operand DEC operand

Logical instructions:
AND dest, src TEST dest,src
OR dest, src XOR dest,src NOT operand

Shift Instructions:
- Arithmetic :
  - SAR operand, 1 SAR operand, CL
  - SAL operand, 1 SAL operand, CL
- Logical:
  - SHR operand, 1 SHR operand, CL
  - SHL operand, 1 SHL operand, CL
  - ROR/ROL/RCR/RCL (rotate operator)

Control Flow
- Unconditional: JMP target
- Simple Branches: JC JS JO JZ LOOP target
- **Unsigned Branches: JA JB JAE JBE**
- Signed Branches: JG JGE JL JLE
- Subroutines: CALL target RET
- ISRs: INT type IRET INTO