Philadelphia University
Faculty of Engineering

Marking Scheme

Examination Paper

Department of CE

Module: Microprocessors (630313)

Final Exam First Semester Date: 21/01/2019

Section 1

Weighting 40% of the module total

Lecturer:	Dr. Qadri Hamarsheh
Coordinator:	Dr. Qadri Hamarsheh
Internal Examiner:	Dr. Naser Halasa

Marking Scheme
Microprocessors (630313)

The presented exam questions are organized to overcome course material, the exam contains 7 questions; *all questions* are compulsory requested to be answered. Thus, the student is permitted to answer any question out of the existing ones in this section.

Marking Assignments

The following scheme shows the marks assignments for each question. They show also the steps for which a student can get marks along the related procedure he/she achieves.

Question 1 This question is attributed with 10 marks if answered properly
The answer for this question as the following:

1) The first processor that includes **real mode** in the Intel microprocessor family was ---------------
   a) 8085 b) 8086
   c) 80286 d) 80386
2) Which of the following is an **invalid** instruction?
   a) add dx,dx b) MOV AX, CS
   c) sub bar,5 d) MOV AL, DI
3) The **directive** that can be used to declare variables to store binary-coded decimal numbers (packed BCD Integers) is ---------------
   a) SWORD b) REAL10
   c) QWORD d) TBYTE
4) The variable definition **smallArray byte 2Ch, 5 DUP ("exam")** will reserve ------- bytes of memory.
   a) 21 b) 26
   c) 6 d) None of above
5) The output of the linker (**LINK** command) is stored in a file with the extension
   a) .lis b) .obj
   c) .lnk d) .exe
6) What will be the values of the **Sign**, and **Zero** flags after the following instructions have executed?

```
mov ax,620h
sub ah,0F6h
```

   a) S=0,Z=1 b) S=0,Z=0
   c) S=1,Z=0 d) S=1,Z=1
7) The conditional branch instruction **JNS** performs the operations when if __
   a) ZF =0 b) PF=0
   c) SF=0 d) CF=0
8) The instruction **TEST** is most similar to----------
   a) OR b) AND
   c) XOR d) NOT
9) The interrupt vector for **INT 17H** is stored in memory at:
   a) 0005CH b) 00068H
   c) 000C5H d) 00017H

10) Which of the following are performed when an **interrupt** occurs:
   (I) FLAGS register is pushed to the stack
   (II) CS register is pushed to the stack
   (III) IP register is pushed to the stack

   a) (I) and (II) and (III) b) (I) and (II) only
   c) (II) and (III) only d) (I) and (III) only

Question 2 This question is attributed with 6 marks if answered properly

a) **Explain 8086 flag register?** **(3 marks)**

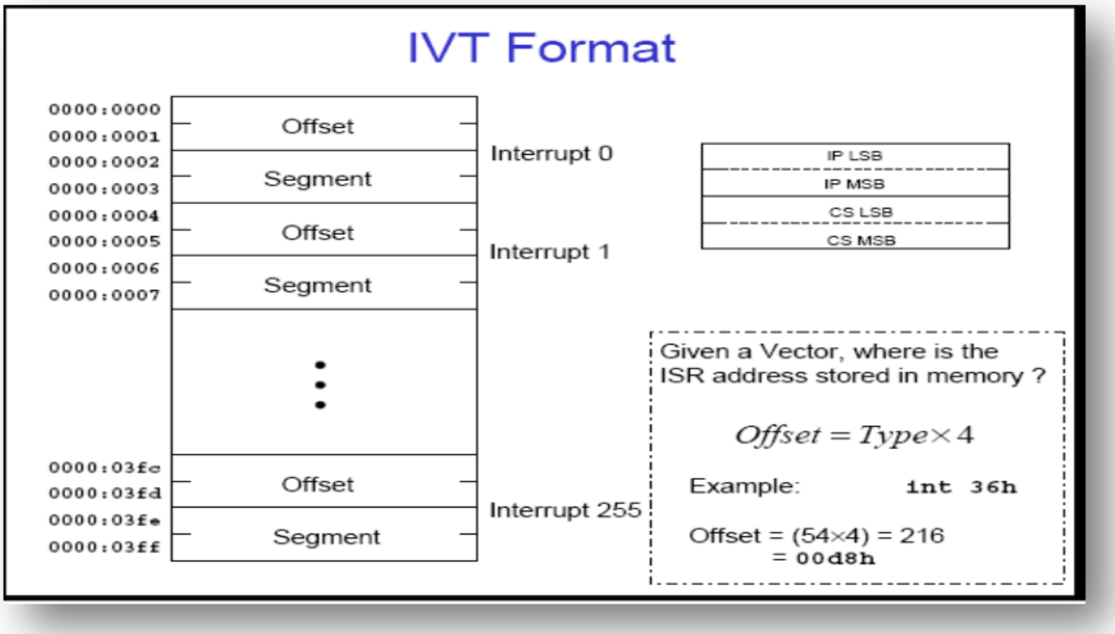
Solution

1. **Carry Flag (CF) - this flag is set to 1 when there is an unsigned overflow. For example when you add bytes 255 + 1 (result is not in range 0...255). When there is no overflow this flag is set to 0.**
2. **Parity Flag (PF) - this flag is set to 1 when there is even number of one bits in result, and to 0 when there is odd number of one bits.**
3. **Auxiliary Flag (AF) - set to 1 when there is an unsigned overflow for low nibble (4 bits).**
4. **Zero Flag (ZF) - set to 1 when result is zero. For non-zero result this flag is set to 0.**
5. **Sign Flag (SF) - set to 1 when result is negative. When result is positive it is set to 0. (This flag takes the value of the most significant bit.)**
6. **Trap Flag (TF) - Used for on-chip debugging.**
7. **Interrupt enable Flag (IF) - when this flag is set to 1 CPU reacts to interrupts from external devices.**
8. **Direction Flag (DF) - this flag is used by some instructions to process data chains, when this flag is set to 0 - the processing is done forward, when this flag is set to 1 the processing is done backward.**
9. **Overflow Flag (OF) - set to 1 when there is a signed overflow. For example, when you add bytes 100 + 50 (result is not in range -128...127).**

b) **What is the use of Interrupt vector table of 8086 microprocessor?** **(2 marks**

Solution

The interrupt vector table contains 256 four byte entries, containing the CS:IP interrupt vectors for each of the 256 possible interrupts. The table is used to locate the interrupt service routine addresses for each of those interrupts.



c) **What is an instruction queue? Explain?** **(1 mark)**

Solution

This is introduced in 8086 processor.This queue is in the BIU and is used for storing the predecoded instructions.This will overlap the fetching and execution cycle. The EU will take the instructions from the queue for decoding and execution.

Question 3 This question is attributed with 4 marks, if answered properly.
The answer for this question as the following:
Write instruction(s) to perform the following tasks:

1)	**Multiply AX by 5**	**MOV CX, 5** **MUL CX**
2)	**Three different instructions that will clear the contents of register CL**	**1) MOV CL, 0H** **2) XOR CL, CL** **3) SUB, CL, CL**
3)	**Jump to label 'HELP' if AX is negative**	**TEST AX, 8000H** **JNZ HELP**
4)	**sets (1) the right most five bits of DI without changing the remaining bits of DI.**	**OR DI,001FH**

Question 4 This question is attributed with 6 marks, if answered properly.
The complete code for this question as the following:

a) *(3 marks)*

Solution

We can count elements of the BW array as follows:

```
                        . . . BW-6    BW-4    BW-2
AW   DW    000Ah,   010Ah,   020Ah,   030Ah,   040Ah
            BW            BW+2     BW+4          BW+6
BW   DW    000Bh,         010Bh,   020Bh,        030Bh
           BW+8    BW+10    BW+12    . . .
CW   DW    000Ch,  010Ch,   020Ch,  030Ch, 040Ch,  050Ch
The following array references have the results given:
     mov   ax,    [BW + 2] ;          ax = 010Bh
     mov   ax,    [AW + 20] ;         ax = 010Ch
     mov   ax,    [BW – 4] ;          ax = 030Ah
mov ax,    1234h
     xchg  ah,    al ;                ax =3412
MOV BX,    B372h
MOVZX      EAX,  BX ;           EAX=0000B372h
MOV BX,    B372h
MOVSX       DX,   BL ;          DX=0072h
```

b) *(3 marks)*

Solution

```
        mov esi, OFFSET Arr_Bytes
        mov al, [esi] ;                    a. AL = ----FFh -------
        mov al, [esi+3] ;                       b. AL = ------3Dh -----
        mov esi, OFFSET Arr_Words + 2
        mov ax, [esi] ;                    c. AX = ------003Bh -----
        mov edi, 8
        mov edx, [Arr_DoubleWords + edi] ;           d. EDX = -----3------
        mov edx, Arr_DoubleWords[edi] ;              e. EDX = ------3------
        mov ebx, Ptr_DoubleWords
        mov eax, [ebx+4] ;                 f. EAX = ------2-----
```

Question 5 This question is attributed with 5 marks, if answered properly.
The answer for this question as the following:

Solution

```
Title string operation
.model small
.stack 100h
 .data
        String db "exercise",0
        Length db ($-String) -1
        Ans db ?                                    (1 mark)
.code
        Main proc
        MOV AX, @data
        MOV DS, AX
        MOV AL,00H
        MOV SI, offset String
        MOV CX, Length                              (1 mark)
Back: MOV BH, [SI]
        CMP BH, 'e'
        JNZ Label
        INC AL
        Label: INC SI
        LOOP Back
        MOV Ans, AL
        MOV AH, 4CH
        INT 21H
        Main endp
End Main                                            (3 marks)
```

Question 6 This question is attributed with 3 marks, if answered properly.
The answer for this question as the following:

Solution

```
mov ax, A
cmp ax, B
jne DoIF
        mov ax, X
        cmp ax, Y
jng EndOfIf
        mov ax, Z
        cmp ax, T
        jnl EndOfIf
; THEN Block:
DoIf: mov ax, D
mov C, ax
; End of IF statement
EndOfIF:
```

Question 7 This question is attributed with 6 marks, if answered properly.
The answer for t his question as the following:

Solution

```
Title ArraysOperations
.model small
.data
   InputArr    db 1,2,3,1,3,5,6,3,4,5
   OddArr  db 10 dup(?)
   EvenArr db 10 dup(?)
   OddAdd  db 0
   EvenAdd db 0
.code
  Main PROC
   mov ax,@data
   mov ds,ax
   LEA BX,InputArr
   LEA SI,OddArr
   LEA DI,EvenArr
   mov cx,10
   mov dh,02                                          (2 marks)
   L1:
      mov ah,00
      mov al,[BX]
      mov dl,al
      div dh
      cmp ah,00                                        (1 mark)
      je EVEN1
      mov [DI],dl
      add OddAdd,dl
      INC DI
      INC BX
      Loop L1
      jmp CAL                                          (1 mark)
   EVEN1:
      mov [SI],dl
      add EvenAdd,dl
      INC SI
      INC BX
      Loop L1                                          (1 mark)
   CAL:
      mov ax,0000
      mov bx,0000
      mov al,OddAdd
      mov bl,EvenAdd
      mov ax,4C00h
      int 21h
Main endp
End Main                                               (1 mark)
```