Philadelphia University
Faculty of Engineering

Marking Scheme

Examination Paper

Department of CE

Module: Microprocessors (630313)

Final Exam	Second Semester	Date: 01/06/2019

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 5 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) Which microprocessor accepts the program written for **8086** without any changes?

- **a)** 8085 — **b)** . 8087
- **c) 8088** — **d)** None of the above

2) One of the following is **not** a valid segment address

- **a)** 00000 — **b)** E0840
- **c)** 8CE90 — **d) 8CE91**

3) Which group of instructions do not affect the flags?

- **a)** Arithmetic operations — **b)** Branch operations
- **c)** Logic operations — **d) Data transfer operations**

4) Which of the following will generate assembly **error**?

- **a)** var1 BYTE 1101b, 22, 35
- **b) var3 BYTE '$','98778',**
- **c)** var2 BYTE "ABCDE", 18
- **d)** None of the above

5) In the following data definition, assume that **X2** begins at offset **4000h**. What is the offset of the third value (**66**)?

```
X2      DWORD      37, 49, 51, 66, 77
```

a)	4004h	b)	4008
c)	400A	d)	400C

6) ______________ can be used as **indexed** registers in **real addressing mode**.

- **a) BX, SI, DI**
- **b)** SI, DI, , DS
- **c)** AX, SI, DI
- **d)** AX, BX, CX

7) Which of the following is an **illegal** 8086 instruction?

- **a)** add ax, [si] — **b) dec [si]**
- **c)** mov ax, [si] — **d)** aDd bx, [si]

8) From hardware viewpoint, the **overflow flag (OF)** can be implemented using the following logic expression:

- **a)** OF = CF AND MSB
- **b)** OF = CF OR MSB
- **c) OF = CF XOR MSB**
- **d)** None of the above

9) Assume that the **AX** register contains the value **6521 H.** What will be the contents of **AX** after execution the instruction:

CMP AL, AH

- **a)** 65BC H — **b)** BC21 H
- **c)** 4421 H — **d) 6521 H**

10) What will be the final value of **ax**?

```
mov ax, 6
mov ecx, 4
L1:
        inc ax
loop L1
```

- **a)** 11 — **b) 10**
- **c)** 9 — **d)** None of the above

Question 2 This question is attributed with 10 marks if answered properly

a) *(1.5 marks)*

Term	Description
***Virtual memory*:**	A way of access to unlimited memory by swapping data between disk storage and RAM.
Real mode	faster operation with maximum of 1 Mbytes of memory
Protected mode	mode is slower but can use 16 Mbytes of memory

b) *(2.5 marks)*

Solution

DWORD: It defines word type variable. The defined variable may have one or more initial values in the directive statement. If there is one value, two bytes of memory space are reserved. The general format is Name of variable DW Initial value or values.

```
X DWORD 1,2,3,4
```

OFFSET: It is an operator to determine the offset (displacement) of a variable or procedure with respect to the base of the segment which contains the named variable or procedure. The operator can be used to load a register with the offset of a variable.

```
X DWORD 1,2,3,4
Mov esi, offset x
```

ENDP: (End Procedure) It informs assembler the end of a procedure. In assembly language programming, subroutines are called procedures. A procedure may be an independent program module to give certain result or the required value to the calling program. This directive is used together with PROC directive to enclose procedure. The general format of ENDP directive is:

```
ProcedureName ENDP
```

EQU:

The EQU directive associates a symbolic name with an integer expression or some arbitrary text.

There are three formats:

```
name EQU expression
Rate EQU 7
Mov Ax, Rate
```

Cannot be redefined

SIZEOF - returns number of bytes used by an array initializer

```
LENGTHOF * TYPE
data
intArray WORD 32 DUP(0)
.code
mov   eax,SIZEOF intArray ;  returns 64 = 32 * 2
```

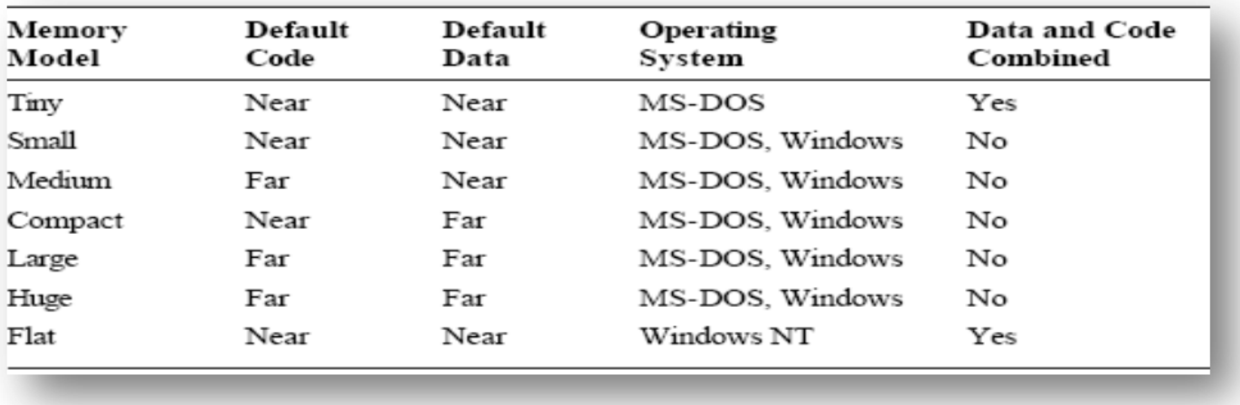
c) *(3.5 marks)*

Solution

1. **TINY MODEL (.MODEL TINY):**
   - **The model uses maximum of 64K bytes for Code and Data.**
2. **SMALL MODEL (.MODEL SMALL):**
   - **The model uses maximum of 64K bytes for Code and 64K bytes for Data (Code<=64K and Data <=64K).**
   - **This model is the most widely used memory model and is sufficient for all the programs to be used in this course.**
3. **MEDIUM MODEL, (.MODEL MEDIUM):**
   - **The model uses maximum of 64K bytes for Data and Code can exceed 64K bytes (Code>64K and Data <=64K).**

4. **COMPACT MODEL, (.MODEL COMPACT):**
   - **The model uses maximum of 64K bytes for Code and Data can exceed 64K bytes (Code<=64K and Data >64K).**
5. **LARGE MODEL, (.MODEL LARGE):**
   - **Both Code and Data can exceed 64K bytes. However no single data set (i.e. array) can exceed 64K bytes (Code>64K and Data >64K).**
6. **HUGE MODEL, (.MODEL HUGE):**
   - **Both Code and Data can exceed 64K bytes. Additionally, a single data set (i.e. array) can exceed 64K bytes (Code>64K and Data >64K).**
7. **FLAT MODEL, (.MODEL FLAT)**
   - **Window NT Application**

Attributes of Memory Models



Memory Model	Default Code	Default Data	Operating System	Data and Code Combined
Tiny	Near	Near	MS-DOS	Yes
Small	Near	Near	MS-DOS, Windows	No
Medium	Far	Near	MS-DOS, Windows	No
Compact	Near	Far	MS-DOS, Windows	No
Large	Far	Far	MS-DOS, Windows	No
Huge	Far	Far	MS-DOS, Windows	No
Flat	Near	Near	Windows NT	Yes

d) *(2.5 marks)*

Solution

Software Interrupt - Internal - from int or into
- **The INT instruction executes a software interrupt.**
- **The code that handles the interrupt is called an interrupt handler.**
- **The Interrupt Vector Table (IVT) holds a 32-bit segment-offset address for each possible interrupt handler.**
- **Interrupt Service Routine (ISR) is another name for interrupt handler.**

Hardware Interrupt - External Uses INTR and NMI
- **Generated by the Intel 8259 Programmable Interrupt Contoller (PIC)**
  - **in response to a hardware signal**

Interrupt Control Instructions
- **STI – set interrupt flag**
- **CLI – clear interrupt flag**

<u>**Question 3**</u> This question is attributed with 10 marks, if answered properly.
The answer for this question as the following:

a) *(2 marks)*

Ñ	Instruction	Answer
1)	CMP AL, GOAL	V
2)	Sub SS, MailSize	I
3)	MOV [1234H] ,AX	V
4)	xchg Goal, Name	I

b) *(2 marks)*

Solution

CF = 1, SF = 0, ZF = 1, OF = 0
CF = 0, SF = 1, ZF = 0, OF = 1
CF = 0, SF = 1, ZF = 0, OF = 0

c) *(2 marks)*

Solution
AL = 40h
AX = 200h
EAX = 20000h
edx = 50000h

d) *(2 marks)*

Solution
SUM =10

e) *(2 marks)*

Solution
0001H
0001H
0003H
0001H

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

Solution

```
TITLE Calculation of different equations Cal.asm
.MODEL          Tiny
.Data
        Y   sword           ?
        X1          sword           ?
        X2          sword           ?
        Z   word            ?
        X3 word             ?
        X4          word            ?
        C   =       200;                                            (1 mark)
.Code
        Main PROC
        mov         ax, @Data
        mov         ds, ax  ;                                           (1 mark)
        ; Initialize variables X1=FFh,  X2=-10,  X3=555,  X4 =100.
        mov         X1, 0FFh
        mov         X2, -10
        mov         X3, 555
        mov         X4, 100;                                            (1 mark)
        ; Compute Y := X1+X2-C and Z := X3+X4
        mov         ax, X1
        add         ax, X2
        sub         ax, C
        mov         Y, ax
        mov         ax, X3
        add         ax, X4
        mov         Z, ax ;                                           (1 mark)
        Main ENDP
        END Main
```

<u>Question 5</u> This question is attributed with 6 marks, if answered properly.
The answer for this question as the following:

Solution

```
Title Compare.asm
.Model flat, stdcall
.Stack 1024
strsize = 100
.Data
       str1            Byte      "enter first string ", 0
       str2            Byte      "enter second string ", 0
       instr1          Byte      strsize dup("0")
       instr2          Byte      strsize dup("0")
       msg1            Byte      "string are equal”
       msg2            Byte      "strings are not equals"
main PROC
.Code
       mov esi, offset instr1
       mov edi, offset instr2                                        (1.5 marks)
; get string
       mov edx, offset str1
       call writestring
       mov edx, offset instr1
       mov ecx, strsize
       call readstring
       mov edx, offset str2
       call writestring
       mov edx, offset instr2
       mov ecx, strsize
       call readstring                                               (1.5 marks)
; string comparision
      mov ecx, strsize
     L1: mov bl, byte ptr [esi]
       cmp byte ptr [edi], bl
       jne L2
       inc esi
       inc edi
       loop L1
       mov edx, msg1
       call writestring
       jmp L3
     L2: edx, msg2
       call writestring
      L3: call crlf
exit
main ENDP
END main                                                            (3 marks)
```