March 27, 2010

COMPUTER ENGINEERING DEPARTMENT

COE 205

COMPUTER ORGANIZATION & ASSEMBLY PROGRAMMING

Major Exam I

Second Semester (092)

Time: 8:00-10:00 PM

Student Name : _KEY____

Student ID.   : ____

Question	Max Points	Score
Q1	76	
Q2	12	
Q3	12	
Total	100	

Dr. Aiman El-Maleh

[76 Points]

(Q1) Fill the blank in each of the following:

(1) The advantages and disadvantages of programming in assembly language are Advantages: accessibility to system hardware and space and time efficiency.
Disadvantages: Programs are not portable and program development and maintenance is more difficult than using high level language.

(2) The advantages and disadvantages of DRAM are

Advantages: smaller in size and hence denser and cheaper.
Disadvantages: slower as it needs refreshing.

(3) The instruction set architecture of a processor is composed of the instruction set, programmer-accessible registers, and memory.

(4) In protected mode, the logical address consists of a 16-bit segment selector and a 32-bit offset.

(5) In protected mode, the linear address is computed based on adding a 32-bit offset with a 32-bit base address obtained from descriptor table indexed by segment selector.

(6) Given that variables I and J are Dword variables, their content can be swapped using the following instructions:

```
mov eax, I
xchg eax, J
mov I, eax
```

(7) Given that variable I is defined as Qword, the content of I is incremented using the following code:

```
add Dword PTR I, 1
adc Dword PTR I+4, 0
```

(8) Given a magnetic disk with the following properties: Rotation Speed = 5000 RPM (rotations per minute), Average Seek = 4 ms, Sector = 512 bytes, Track = 250 sectors. The average time to access a block of 50 consecutive sectors is
Average access time= Seek Time + Rotation Latency + Transfer Time
Rotations per second=5000/60 =83.33 RPS
Rotation time in milliseconds=1000/83.33=12 ms
Time to transfer 50 sectors=(50/250)* 12=2.4 ms
Average access time=4 + 6 + 2.4 =12.4 ms.

(9) The integer number -2000 is represented in hexadecimal using 16-bit 2's complement representation as F830.

(10) Assuming 16-bit 2's complement representation, the hexadecimal number EC10 represents the decimal number -5104.

(11) Assuming 4-bit 2's complement representation, the largest number that can be stored is +7 in decimal and 0111 in binary and the smallest number that can be stored is -8 in decimal and 1000 in binary.

(12) Given that the number 88h is represented using 8-bit 2's complement representation, the equivalent number represented using 16-bit 2's complement representation is FF88.

(13) Given that register AL=C4 stores an ASCII character, then the stored character is D and the used parity is odd. Note that 'A'=41h and 'a'=61h.

(14) The EIP register holds the address of the next instruction to be fetched from memory.

(15) Given that the instruction ADD AX, I (having the machine code 03060000) is stored at address 0000002C, then the address of the next instruction to be fetched from memory is 0000002C+4=00000030.

(16) Given a processor with an 8-stage pipeline and clock frequency of 2 GHZ, the time that will be required to execute a program of 4 billion instructions assuming that there will be no pipeline stalls is nearly $4x10^9/(2x10^9)=2$ seconds.

(17) Assume that the range of addresses from 00000 to 00A1A is used by another program. Given that a program has a code segment of 8 Kbyte and a data segment of 3 Kbyte, the code segment number allocated is 00A2 and the data segment number allocated is 02A2.

The used address space for the code segment is 00A20+1FFF=2A1F. Thus, the next available address is 2A20. Thus, the next available segment is 2A2.

(18) Assume that DS=00EF, CS=013A, ES=0112, SS=0FEC, IP=00FF, BX=309A, and SP=01FC. Based on 16-bit real-mode addressing, the linear address of the next instruction to be fetched from memory is 013A0+000FF=0149F.

(19) Assume that DS=00EF, CS=013A, ES=0112, SS=0FEC, IP=00FF, BX=309A, and SP=01FC. Based on 16-bit real addressing mode, the linear address of the source operand in the instruction MOV AX, [BX+5] is 00EF0+0309A+00005=03F8F.

(20) The addressing mode of the source operand in the instruction MOV EAX, [MSG+1] is direct addressing mode.

(21) Assume that AX=00FFh. Executing the instruction INC AL produces the result AX=<u>0000</u>.

(22) The addressing mode of the source operand in the instruction MOV SI, [EBX] is <u>register indirect addressing mode</u>.

(23) The assembler allocates <u>4*(5*(1+30))=620</u> bytes for the variable *Array* defined below:

Array Dword 5 dup(30, 30 dup(0))

(24) The content of register EAX after executing the following instructions is<u> -5+2+10=7</u>.

```
I=5
J EQU 10
MOV EAX, I-J
I=I-3
ADD EAX, I+J
```

(25) Assuming the following data segment and assuming that variable X is given the linear address 00404000h, then the linear address for variables Y and Z will be <u>00404008h</u> and <u>00404010h.</u>

```
.DATA
X         BYTE 1, 2, 3, 4, 5
ALIGN 4
Y         DWORD 4, 5
ALIGN  2
Z         WORD  7,  8, 9
```

(26) Assuming the following data segment and assuming that variable X is given the linear address 00404000h, then the content of register EAX after executing the instruction MOV EAX, Y-5 is 14000A00.

```
.DATA
X       BYTE “EXAM I”, 0
        WORD 10, 20
Y       DWORD  30, 40
```

(27) Assuming the following data segment and assuming that variable X is given the linear address 00404000h , after executing the code given below, the content of register EAX=4 and EBX=00404004h.

```
.DATA
X       WORD  10, 20, 30
Y       DWORD 30, 40, 50
.CODE
MOV EAX, TYPE Y
MOV EBX, OFFSET Y-2
```

(28) After executing the code given below, the content of registers EAX and EBX will be 00000006 and 24d=00000018h.

```
.DATA
ARRAY          DWORD          10, 20, 30,
                              40, 50, 60
.CODE
MOV EAX, LENGTHOF ARRAY
MOV EBX, SIZEOF ARRAY
```

(29) After executing the code given below, the content of register EAX will be 28001E00.

```
.DATA
ARRAY WORD  10, 20, 30, 40, 50, 60
.CODE
MOV EAX, DWORD PTR ARRAY+3
```

(30) Assuming that variable ARRAY is defined as shown below:

```
ARRAY  DWORD 1, 2, 3, 4, 5, 6
```

The content of register AX after executing the instruction MOV EAX, ARRAY+2 will be <u>0000</u>.

(31) Assume that AL=93h. Executing the instruction MOVSX EBX, AL produces the result EBX=<u>FFFFFF93</u>.

(32) Assume that AX=A100h. Executing the instruction *NEG AX* produces the following results: AX=<u>5F00</u> overflow flag=<u>0</u>, sign flag=<u>0</u>, zero flag=<u>0</u>, carry flag=<u>1</u>, auxiliary flag=<u>0</u> and parity flag=<u>1</u>.

(33) Assume that AX=ABCDh and BX=8876h. Executing the instruction *ADD AX, BX* produces the following results: AX=<u>3443,</u> overflow flag=<u>1</u>, sign flag=<u>0</u>, zero flag=<u>0</u>, carry flag=<u>1</u>, auxiliary flag=<u>1</u> and parity flag=<u>0</u>.

(34) Assume that AX= 98A0h and BX= FFDAh. Executing the instruction *SUB AX, BX* produces the following results: AX=<u>98C6,</u> overflow flag=<u>0</u>, sign flag=<u>1</u>, zero flag=<u>0</u>, carry flag=<u>1</u>, auxiliary flag=<u>1</u> and parity flag=<u>1</u>.

(35) The content of register EAX after executing the instructions below will be <u>0000000C</u>.

```
.DATA
    ARRAY          DWORD  1, 2, 3, 4
                   DWORD  5, 6, 7 , 8
                   DWORD  9, 10, 11, 12
    RS EQU   SIZEOF ARRAY
.CODE
    MOV ESI, 2*RS
    MOV EDI, 3
    MOV EAX, ARRAY[ESI+EDI*TYPE ARRAY]
```

[12 Points]

(Q2) Consider a program that has the following data segment assuming a flat memory model:

X	*EQU*	*16*
Y	*BYTE*	*17*
Z	*WORD*	*18*
W	*DWORD*	*19*

Indicate whether the following are valid **IA-32** instructions or not. **If invalid, give the reason**:

1. MOV EAX, W-1

Valid.

2. MOV Z, Word PTR Y

Invalid. Both operands are memory operands.

3. MOV DS, X

Invalid. We can’t move a constant to a segment register.

4. MOV Z, X

Valid.

5. MOV AX, OFFSET Z

Invalid. There is size mismatch because the address is 32-bit.

6. MOVSX EAX, X

Invalid. The source operand can’t be a constant in a MOVSX instruction.

7. MOV W, Dword PTR AX

Invalid. PTR operator can’t be used with a register.

8. INC [EBX]

Invalid. There is ambiguity in the size of the operand to be incremented.

[12 Points]

(Q3) Suppose that the following directives are declared in the data segment with a starting linear address of 00404000. Show the linear addresses of allocated memory and their corresponding content in hexadecimal. Note that the ASCII code for character ‘a’ is 61h and that of character ‘A’ is 41h. The ASCII code of character ‘0’ is 30h.

```
I      BYTE      10,“10”,0
       WORD      10, -10
J      DWORD     112, -112
K      EQU       100
L      BYTE      K+20
       BYTE      2, 2 dup(1,-1)
```

Variable	Linear Address (Hex.)	Content (Hex.)
I	00404000	0A
	00404001	31
	00404002	30
	00404003	00
	00404004	0A
	00404005	00
	00404006	F6
	00404007	FF
J	00404008	70
	00404009	00
	0040400A	00
	0040400B	00
	0040400C	90
	0040400D	FF
	0040400E	FF
	0040400F	FF
L	00404010	78
	00404011	02
	00404012	01
	00404013	FF
	00404014	01
	00404015	FF