April 15, 2009

COMPUTER ENGINEERING DEPARTMENT

COE 205

COMPUTER ORGANIZATION & ASSEMBLY PROGRAMMING

Major Exam I

Second Semester (082)

Time: 7:00-9:00 PM

Student Name : ____

Student ID. : ____

Question	Max Points	Score
Q1	60	
Q2	10	
Q3	10	
Q4	20	
Total	100	

Dr. Aiman El-Maleh

[60 Points]

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

(1) There is one-to-one correspondence between machine language and ________________________.

(2) _____________ translate high-level programs to machine code.

(3) One of the main advantages of programming in high level languages is that programs are _______________.

(4) Programs written in assembly language have the advantage of being both ___________ and ___________ efficient.

(5) ____ of a processor provides a hardware/software interface.

(6) The size of the address bus of the 8086 processor is ______ bits while it is ______ bits for the Pentium IV processor.

(7) With the Pentium processor having a 32 bit address bus and a 64 bit data bus, the physical address space is _________________ bytes and the maximum number of bytes that can be transferred in a single read/write cycle is _______ bytes.

(8) _____________ RAM is slower than ___________ RAM but is denser and cheaper.

(9) ______________________ can help bridge the widening speed gap between CPU and main memory.

(10) Given a magnetic disk with the following properties: Rotation speed = 5400 RPM (rotations per minute), Average seek = 6 ms, Sector = 512 bytes, Track = 256 sectors. The average time to access a block of 64 consecutive sectors is ____.

(11) The integer number -4992 is represented in hexadecimal using 16-bit 2's complement representation as _______________.

(12) Assume that AX=8411h and BX=F857. Executing the instruction *ADD AX, BX* produces the following results: AX=______, overflow flag=___, sign flag=___, zero flag=___, carry flag=___, auxiliary flag=___ and parity flag=___.

(13) Assume that AX=8411h and BX=F857. Executing the instruction *SUB AX, BX* produces the following results: AX=______, overflow flag=___, sign flag=___, zero flag=___, carry flag=___, auxiliary flag=___ and parity flag=___.

(14) As part of the instruction set architecture of the Pentium-IV processor, it has ___________________ general-purpose registers, _____________ segment registers in addition to _______________________ and ______________________ registers.

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

(16) Given that the instruction MOV ECX, 1000 (having the machine code B9 000003E8) is stored at address 00000005, then the address of the next instruction to be fetched from memory is __________________.

(17) In real addressing mode, assume that the code segment occupies the address range from 00000 to 010F5. The next available free segment is _______________.

(18) Assume that DS=10AF, CS=4FE5, ES=F030, SS=E123, IP=0059, BX=1055, and SI=577F. Based on 16-bit real-mode addressing, the linear address of the next instruction to be fetched from memory is ____.

(19) Assume that DS=10AF, CS=4FE5, ES=F030, SS=E123, IP=0059, BX=1055, and SI=577F. Based on 16-bit real addressing mode, the linear address of the source operand in the instruction MOV AX, [SI] is ____.

(20) In protected mode, ___________________ translates logical address to linear address while ___________________ translates linear address to physical address.

(21) ___________________ provide information to the assembler while translating a program and are non-executable.

(22) The assembler allocates ________ bytes for the variable *Array* defined below:

Array DWORD 5, 5 dup(5, 5 dup(0))

(23) 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 ____________ and ___________.

```
.DATA
X       BYTE 10, 11, 12, 13, 14
Y       WORD 15
ALIGN   4
Z       DWORD  16
```

(24) 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, OFFSET Y-2 is ________________.

```
.DATA
X       BYTE “COE205”, 10, 13
        WORD 1, 2, 3, 4
Y       DWORD  16
```

(25) After executing the code given below, the content of registers EAX and EBX will be _______________ and ______________.

```
.DATA
ARRAY DWORD     -1, 50,
                0FEh, -200,
                1010b, 0ABCDh
.CODE
MOV EAX, LENGTHOF ARRAY
MOV EBX, SIZEOF ARRAY
```

(26) After executing the code given below, the content of register EAX will be _______________.

```
.DATA
ARRAY BYTE   1, 2, 3, 4, 5, 6, 7, 8
.CODE
MOV EAX, DWORD PTR ARRAY
```

(27) Assuming variable ARRAY is defined as shown below:

```
ARRAY  WORD 1, 2, 3, 4, 5, 6, 7, 8
```

The content of register AX after executing the instruction MOV AX, ARRAY+3 will be _____________.

(28) The addressing mode of the source operand in the instruction MOV EAX, offset ARRAY+4 is __________________________.

(29) The addressing mode of the source operand in the instruction MOV EAX, ARRAY+20[EBX*2-4] is __________________________.

(30) Assume that AX=50A3h. Executing the instruction MOVSX EBX, AL produces the result EBX=________________.

(31) After executing the code shown below, the content of register EAX will be __________ and the content of register ECX will be _________.

```
        MOV ECX, 10
        MOV EAX, 0
NEXT:
        ADD EAX, ECX
        LOOP NEXT
```

(32) Considering the code below, the value stored in the address field for NEXT in the LOOP instruction is __________________.

```
Offset      Machine Code                 Source Code

00000000  B9 00000004                    MOV ECX, 4
00000005  B8 00000002                    MOV EAX, 2
0000000A                     NEXT:
0000000A  03 C0                          ADD EAX, EAX
0000000C  40                             INC EAX
0000000D  E2 ??                          LOOP NEXT
```

(33) Considering the code below, the content of register AX after executing the code will be __________________.

```
.DATA
    ARRAY           WORD  1, 2, 3, 4, 5
                    WORD  6, 7,8 , 9, 10
                    WORD 11, 12, 13, 14, 15
                    WORD 16, 17, 18, 19, 20
    RSIZE EQU   SIZEOF ARRAY
.CODE
    MOV ESI, 3*RSIZE
    MOV EDI, 4
    MOV AX, ARRAY[ESI+EDI*TYPE ARRAY]
```

[10 Points]

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

I	*EQU*	*1*
J	*EQU*	*AX*
K	*BYTE*	*10*
L	*WORD*	*I+10*

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

1. MOV AX, L-1

2. MOV AX, offset K+1

3. MOV DS, J

4. MOV ES, K

5. MOV [2*EAX+EAX], 20

6. SUB [ESI*4], AX

7. MOV EAX, OFFSET L[EBX]

8. MOV EAX, DWORD PTR BX

9. MOVSX EAX, AL

10. MOV [EAX+ESI], L

[10 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     -20, ‘20’
      WORD     20
J     DWORD    0FEh
K     EQU      67H
L     BYTE     K-5
      BYTE     2, 2 dup(2,’C’)
```

Variable	Linear Address (Hex.)	Content (Hex.)
I	00404000	

[20 Points]

(Q4) Assume that you have a two-dimensional array of integers, declared as Array, with each integer defined as a DWORD. Write an assembly program to swap the content of any two columns assuming that the two column numbers to be swapped are stored in registers AL and AH. Assume that the number of rows and number of columns in the array are defined in the constants NRow and NCol, respectively. **<u>Your program should work for any array size</u>**.

For example, assume the following array definition:

```
NRow EQU 4
NCol EQU 5
Array       DWORD 1, 2, 3, 4, 5
            DWORD 6, 7, 8, 9, 10
            DWORD 11, 12, 13, 14, 15
            DWORD 16, 17, 18, 19, 20
```

After executing the program assuming AH=1 and AL=2, the content of Array will be:

```
Array       DWORD 1, 3, 2, 4, 5
            DWORD 6, 8, 7, 9, 10
            DWORD 11, 13, 12, 14, 15
            DWORD 16, 18, 17, 19, 20
```