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 : ___KEY_____

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 assembly language.

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

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

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

(5) The instruction set architecture of a processor provides a hardware/software interface.

(6) The size of the address bus of the 8086 processor is 20 bits while it is 36 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 2^{32}=4 Gbytes and the maximum number of bytes that can be transferred in a single read/write cycle is 8 bytes.

(8) Dynamic RAM is slower than static RAM but is denser and cheaper.

(9) Cache memory 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 14.34 ms.

Average access time= Seek Time + Rotation Latency + Transfer Time
Rotations per second=5400/60 =90 RPS
Rotation time in milliseconds=1000/90=11.11 ms
Time to transfer 64 sectors=(64/256)* 11.11=2.78 ms
Average access time=6 + 5.56 + 2.78 =14.34 ms.

(11) The integer number -4992 is represented in hexadecimal using 16-bit 2's complement representation as EC80.

(12) Assume that AX=8411h and BX=F857. Executing the instruction *ADD AX, BX* produces the following results: AX=7C68, overflow flag=1, sign flag=0, zero flag=0, carry flag=1, auxiliary flag=0 and parity flag=0.

(13) Assume that AX=8411h and BX=F857. Executing the instruction *SUB AX, BX* produces the following results: AX=8BBA, overflow flag=0, sign flag=1, zero flag=0, carry flag=1, auxiliary flag=1 and parity flag=0.

(14) As part of the instruction set architecture of the Pentium-IV processor, it has 8 general-purpose registers, 6 segment registers in addition to EIP and EFLAGS registers.

(15) The EIP 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 00000005+5=0000000A.

(17) In real addressing mode, assume that the code segment occupies the address range from 00000 to 010F5. The next available free segment is 0110.

(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 4FE50+0059=4FEA9.

(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 10AF0+577F =1626F.

(20) In protected mode, segment unit translates logical address to linear address while paging unit translates linear address to physical address.

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

(22) The assembler allocates 31*4=124 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 00404005h and 00404008h.

```
.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 0040400Eh.

```
.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 00000006 and 00000018.

```
.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 04030201.

```
.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 0300.

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

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

(30) Assume that AX=50A3h. Executing the instruction MOVSX EBX, AL produces the result EBX=FFFFFFA3.

(31) After executing the code shown below, the content of register EAX will be 55d=00000037 and the content of register ECX will be 0.

```
        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 Next-PC=0000000A-0000000F=FB.

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 20d=0014.

```
.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

Valid.

2. MOV AX, offset K+1

Invalid. Size mismatch as AX is a word while offset K+1 is a 32-bit address.

3. MOV DS, J

Valid.

4. MOV ES, K

Invalid. Size mismatch as ES is a word while K is a Byte.

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

It is supposed to be **Invalid** as there is ambiguity since constants do not have size. However, with MASM615, for the MOV instruction if size is not specified it will assume it a double word!!

6. SUB [ESI*4], AX

Valid.

7. MOV EAX, OFFSET L[EBX]

Invalid. Offset operator can only be used with direct addressing mode.

8. MOV EAX, DWORD PTR BX

Invalid. PTR operator can only be used with memory operands.

9. MOVSX EAX, AL

Valid

10. MOV [EAX+ESI], L

Invalid. Both source and destination are memory operands.

[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	EC
	00404001	32
	00404002	30
	00404003	14
	00404004	00
J	00404005	FE
	00404006	00
	00404007	00
	00404008	00
L	00404009	62
	0040400A	02
	0040400B	02
	0040400C	43
	0040400D	02
	0040400E	43

[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. **Your program should work for any array size**.

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
```

```
        MOV AH, 1
        MOV AL, 2
        MOV ECX, NRow
        MOVZX ESI, AL       ; index of first column
        MOVZX EDI, AH       ; index of second column
        XOR EDX, EDX
Next:
        MOV EBX, Array[EDX+ESI*TYPE ARRAY]
        XCHG EBX, Array[EDX+EDI*TYPE ARRAY]
        MOV Array[EDX+ESI*TYPE ARRAY], EBX
        ADD EDX, Sizeof Array
        Loop   Next
```