Oct. 28, 2010

COMPUTER ENGINEERING DEPARTMENT

COE 205

COMPUTER ORGANIZATION & ASSEMBLY PROGRAMMING

Major Exam I

First Semester (101)

Time: 1:00-3:00 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) Two advantages of programming in assembly language include accessibility to system hardware and space and time efficiency while two disadvantages include not portable and program development and maintenance is more difficult than high-level languages.

(2) There is one-to-one correspondence between assembly instructions and machine instructions.

(3) The linker combines program's object file with other object files and link libraries, and produces a single executable program.

(4) The Instruction Set Architecture (ISA) provides a hardware/software interface.

(5) The 8086 processor is a 16 bit machine with an address bus of 20 bits and a data bus with 16 bits.

(6) With a 36-bit address bus, the physical address space is 2^{36}=64 GByte.

(7) Cache memory is used to bridge the CPU-memory performance gap.

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

(9) A segment descriptor table stores for each segment 32-bit base address, 20-bit segment size and access rights.

(10) Given that variables I and J are **Dword** variables, the equation I = I + 2*J can be implemented using the following minimal instructions:

MOV EAX, J
ADD EAX, EAX
ADD I, EAX

(11) Given that variable I is defined as **Qword**, the equation I = I + 5 can be implemented using the following minimal instructions:

ADD DWORD PTR I, 5
ADC DWORD PTR I+4, 0

(12) Given a magnetic disk with the following properties: Rotation Speed = 4000 RPM (rotations per minute), Average Seek = 4 ms, Sector = 512 bytes, Track = 500 sectors. The average time to access a block of 100 consecutive sectors is
Average access time= Seek Time + Rotation Latency + Transfer Time
Rotation time in milliseconds=1000*60/4000=15 ms
Time to transfer 100 sectors=(100/500)* 15=3 ms
Average access time=4 + 15/2 + 3 =14.5 ms.

(13) The integer number -1000 is represented in hexadecimal using 16-bit 2's complement representation as FC18.

(14) Assuming 16-bit 2's complement representation, the hexadecimal number FE00 represents the decimal number -512.

(15) Assuming 8-bit 2's complement representation, the largest number that can be stored is +127 in decimal and 01111111 in binary and the smallest number that can be stored is -128 in decimal and 10000000 in binary.

(16) Given that the number A80h is represented using 16-bit 2's complement representation, the equivalent number represented using 32-bit 2's complement representation is FFFFFA80.

(17) Given that register AL=E6 stores an ASCII character, then the stored character is f and the used parity is odd. Note that 'A'=41h and 'a'=61h.

(18) Given that the instruction MOV EAX, 10 (having the machine code B8 0000000A) is stored at address 00000008, then the address of the next instruction to be fetched from memory is 00000008+5=0000000D.

(19) Given a processor with a 5-stage pipeline and clock frequency of 4 GHZ, the time that will be required to execute a program of 8 billion instructions assuming that there will be no pipeline stalls is nearly $8x10^9/(4x10^9)=2$ seconds.

(20) Assume that in real mode the range of addresses from 00000 to 00109 is used by another program. Given that a program has a code segment of 4 Kbyte and a data segment of 2 Kbyte, the code segment number allocated is 0011 and the data segment number allocated is 0111.

(21) Assume that DS=0010, CS=0030, ES=0060, SS=0080, IP=0035, BX=00AF, and SP=020C. Based on 16-bit real-mode addressing, the linear address of the next instruction to be fetched from memory is 00300+0035=00335.

(22) Assume that DS=0010, CS=0030, ES=0060, SS=0080, IP=0035, BX=00AF, and SP=020C. Based on 16-bit real addressing mode, the linear address of the source operand in the instruction MOV AX, [BX] is 00100+00AF=001AF.

(23) The addressing mode of the source operand in the instruction MOV EAX, Array[10] is direct.

(24) Assume that AX=0000h. Executing the instruction DEC AL produces the result AX=00FF.

(25) The addressing mode of the source operand in the instruction MOV EAX, [ESI] is register indirect.

(26) The assembler allocates 4*10*10=400 bytes for the variable *Array* defined below:

Array Dword 10 dup(10 dup(0))

(27) 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 00404004h and 00404008h.

```
.DATA
X       BYTE 1, 2, 3
ALIGN 2
Y       WORD 4, 5
ALIGN  4
Z       DWORD  6,  7
```

(28) 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, Dword PTR X+2 is 00030002.

```
.DATA
X       WORD  1, 2, 3, 4
```

(29) 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=1 and EBX=00404003h.

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

(30) After executing the code given below, the content of registers EAX and EBX will be 2 and 4.

```
.DATA
ARRAY       WORD  10, 20
            WORD  30, 40
.CODE
MOV EAX, LENGTHOF ARRAY
MOV EBX, SIZEOF ARRAY
```

(31) Assume that AL=A0h. Executing the instruction MOVSX EAX, AL produces the result EAX=FFFFFFA0.

(32) Assume that AX=0F90h. Executing the instruction *NEG AX* produces the following results: AX=F070, overflow flag=0, sign flag=1, zero flag=0, carry flag=1, auxiliary flag=0 and parity flag=0.

(33) Assume that AX=800Fh and BX=FFF1h. Executing the instruction *ADD AX, BX* produces the following results: AX=8000, overflow flag=0, sign flag=1, zero flag=0, carry flag=1, auxiliary flag=1 and parity flag=1.

(34) Assume that AX=800Fh and BX=00F1h. Executing the instruction *SUB AX, BX* produces the following results: AX=7F1E, overflow flag=1, sign flag=0, zero flag=0, carry flag=0, auxiliary flag=0 and parity flag=1.

[12 Points]

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

```
A       EQU     10
B       BYTE    11, 12
C       WORD    13
D       DWORD   14
```

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

1. MOV AX, B

Invalid. Size mismatch as AX is a word while B is a byte.

2. MOV B, A

Valid.

3. MOV EAX, C+1

Invalid. Size mismatch as EAX is a dword while C+1 is a word.

4. MOV AL, [EBX]

Valid.

5. MOV AX, OFFSET C

Invalid. Size mismatch as AX is a word while offset C is a dword.

6. MOVSX EAX, AX

Valid.

7. ADD [EAX], A+1

Valid.

8. MOV DS, ES

Invalid. We cannot move a segment register into a segment register directly.

[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        -1, 100, “AB”
        WORD        -16
J       DWORD       -30
K       EQU         10
L       BYTE        K-15
        BYTE        5, 3 dup(1,2)
```

Variable	Linear Address (Hex.)	Content (Hex.)
I	00404000	FF
	00404001	64
	00404002	41
	00404003	42
	00404004	F0
	00404005	FF
J	00404006	E2
	00404007	FF
	00404008	FF
	00404009	FF
L	0040400A	FB
	0040400B	05
	0040400C	01
	0040400D	02
	0040400E	01
	0040400F	02
	00404010	01
	00404011	02