Quiz 1:

1. Translate the following decimal numbers to binary (in 8 bits):
    a. 34
    b. 18
2. Translate the following decimal numbers to hexadecimal:
    a. 75
    b. 97
3. Add the following binary integers: 00101101 and 00111011
4. Translate the following signed binary numbers to decimal:
    a. 11111111
    b. 11111000
5. Translate the following signed decimal numbers to binary (in 8 bits):
    a. -7
    b. -1
6. What is the value of the boolean expression $X \vee (Y \wedge Z)$ when X=true, Y=false, and Z=false?
    a. True
    b. False
7. Is the expression $X \vee (Y \wedge Z)$ equivalent to $(X \vee Y) \wedge (Y \vee Z)$ for all inputs of X, Y, and Z?
    a. True
    b. False
8. Three hexadecimal digits can be used to represent 12 binary bits.
    a. True
    b. False
9. The most significant bit in a binary byte is numbered bit 8.
    a. True
    b. False
10. A signed integer stores the sign in the least significant bit (LSB).
    a. True
    b. False
11. If an integer’s sign bit is 1, the integer is positive.
    a. True
    b. False
12. The expression $X \wedge Y$ is true only when X and Y are both true.
    a. True
    b. False
13. The expression $X \vee Y$ is only true when X and Y are both true.
    a. True
    b. False

14. The expression ¬ (X ∧ Y) is true when X and Y are both false.
    a. True
    b. False
15. The three most basic operators in Boolean algebra are AND, OR, and NOT.
    a. True
    b. False
16. What is the largest unsigned integer that may be stored in 20 bits?
    a. $2^{15} - 1$
    b. 2^{16}
    c. $2^{20} - 1$
    d. 2^{31}
17. What is the largest signed integer that may be stored in 32 bits?
    a. $2^{32} - 1$
    b. 2^{32}
    c. $2^{31} - 1$
    d. 2^{31}
18. The two's complement of an integer is formed by doing which of the following?
    a. Reversing (inverting) the bits and adding 1
    b. Adding 1 and reversing the bits
    c. Adding 2 and reversing the bits
    d. Changing the highest bit to 1
19. Which list contains the correct hexadecimal translation (in order) of the following unsigned decimal integer? 33, 95, 257
    a. 21, 5F, 101
    b. 22, 5E, 11A
    c. 6A, 5F, 101
    d. 21, 62, 103
20. Which of the following binary values is equivalent to hexadecimal 4A2B?
    a. 0100 1010 0010 1101
    b. 0110 1010 0010 1011
    c. 0100 1010 0010 1011
    d. 0110 1011 0010 1001
21. Which of the following binary values is equivalent to hexadecimal 7CBE?
    a. 0111 1101 1011 1110
    b. 0111 1011 1011 1100
    c. 0111 1100 1100 1110
    d. 0111 1100 1011 1110
22. Which of the following is the binary translation of signed decimal -33?
    a. 11011111
    b. 10101011
    c. 11001100
    d. 11100011

Quiz 1 Answers:

1. Translate decimal to 8 bit binary:
    a. 34: 00100010
    b. 18: 00010010
2. Translate decimal to Hex:
    a. 75: 4B
    b. 97: 61
3. 01101000
4. Translate signed binary to decimal:
    a. 11111111: -1
    b. 11111000: -8
5. Translate signed decimal to signed binary:
    a. -7: 11111001
    b. -1: 11111111
6. A. True
7. A. Yes
8. A. True
9. B. False *This is the question. The most significant bit is number 7 because we start counting right to left from 0. It is the **8th bit** but it is **numbered 7**.*
10. B. False
11. B. False
12. A. True
13. B. False
14. A. True
15. A. True
16. C. $2^{20} - 1$
17. C. $2^{31} - 1$
18. A. reversing (inverting) the bits and adding 1
19. A. 21, 5F, 101
20. C. 0100 1010 0010 1011
21. D. 0111 1100 1011 1110
22. A. 11011111

Quiz 2:

1. In real-address mode, convert the following segment offset address to a linear address: 0950:0100
2. Which two 32-bit registers are known as extended index registers?
    a. SI, DI
    b. EAX, EBX
    c. ESI, EDI
    d. EBP, ESP
3. What is the name of the lowest 8 bits of the EDX register?
    a. DL
    b. DH
    c. DX
    d. None of the above
4. How much memory can be addressed in Real-address mode?
    a. 640 K
    b. 1 MB
    c. 16 MB
    d. 4 GB
5. How much memory can be addressed in Protected mode?
    a. 640 K
    b. 1 MB
    c. 16 MB
    d. 4 GB
6. Which of the following linear addresses matches the segment-offset address 08F0:0200?
    a. 09100h
    b. 09200h
    c. 0AF0h
    d. 08F2h
7. Within the CPU, all calculations and logic operations take place inside the _____.
    a. Registers
    b. ALU
    c. CU
    d. MBU
8. The four parts of the CPU are:
    a. Data bus, memory unit, control unit, arithmetic logic unit
    b. Address bus, registers, control unit, arithmetic logic unit
    c. Clock memory unit control unit, instruction fetch unit
    d. Clock, registers, control unit, arithmetic logic unit
9. Which register is known as a loop counter?
    a. EAX
    b. EBX

    c. ECX
    d. EDX
10. Which directive identifies the part of a program containing instructions?
    a. .DATA
    b. .CODE
    c. STACK
    d. .PROG
11. Which of the following are valid data definition statements that create an array of unsigned bytes containing decimal 10, 20, and 30, named **myArray**?
    a. myArray BYTE 10, 20, 30
    b. BYTE myArra 10, 20, 30
    c. BYTE myArra[3]: 10, 20, 30
    d. myArray BYTE DUP(3) 10, 20, 30
12. In the following data definition, assume that **List2** begins at offset 2000h. What is the offset of the fourth value (6)?

    List2 WORD 3 , 4 , 5 , 6 , 7

    a. 2008h
    b. 2006h
    c. 2000h
    d. 2004h
13. Which letter choice shows the memory byte order using little endian, from low to high address, of the following data definition? *BigVal DWORD 12345678h*
    a. 56h, 78h, 12h, 34h
    b. 12h, 34h, 56h, 78h
    c. 78h, 56h, 34h, 12h
    d. 34h, 12h, 78h, 56h
14. Given the following array definition, which letter choice contains a valid constant declaration named **ArrayCount** that automatically calculates the number of elements in the array?

    newArray WORD 10 , 20 , 30 , 40 , 50

    a. ArrayCount = newArray - $
    b. ArrayCount = ($ - newArray)/2
    c. ArrayCount = $ - newArray
    d. ArrayCount = (newArray - $)/4
15. Select a data definition statement that creates an array of 50 doublewords named **myList** and initializes each array element to the value 0.
    a. DWORD myList 50 DUP(0)
    b. myList 50 DWORD DUP(0)
    c. myList 50 DWORD(0)

d. myList DWORD 50 DUP(0)

16. The following is a valid data definition statement:

```
List1 BYTE 10, 20, 20
      BYTE 30, 40, 30
```

  a. True
  b. False

17. The EQU directive permits a constant to be redefined at any point in a program.
  a. True
  b. False

18. Which of the following defines a text macro named MESSAGE that contains the string data?
  “I’m good at this!”, 0
  a. MESSAGE TEXTEQU<”I’m good at this!”,0>
  b. TEXTEQU MESSAGE<”I’m good at this!”,0>
  c. <”I’m good at this!”,0> TEXTEQU MESSAGE
  d. TEXTEQU <”I’m good at this!”, 0> MESSAGE

Some of the following questions have more than one correct answer. Circle all correct answers.

19. Which utility program reads an assembly language source file and produces an object file?
  a. Compiler
  b. Linker
  c. Assembler
  d. Loader

20. Which of the following will generate assembly error?
  a. Var1 BYTE 1101b, 22, 35h
  b. Var2 BYTE “ABCDE”, 18
  c. Var3 BYTE ‘$’, ‘98778’,
  d. Var4 BYTE 256, 19, 40

21. The byte-ordering scheme used by computers to store large integers in memory with the low-order byte at the lowest address is called:
  a. Big endian
  b. Byte-major
  c. Little endian
  d. Byte-minor

22. Operands may be any of the following:
  a. Constant or constant expression
  b. Reserved word
  c. Register name
  d. Variable name (memory)

Quiz 2 answers

1. 09600
2. C. ESI, EDI
3. A. DL
4. B. 1 MB
5. D. 4 GB
6. A. 09100h
7. B. ALU
8. D. Clock, registers, control unit, arithmetic logic unit
9. C. ECX
10. B. .CODE
11. A. myArray BYTE 10, 20, 30
12. B. 2006h
13. C. 78h, 56h, 34h, 12h
14. B. ArrayCount = ($ - newArray)/2
15. D. myList DWORD 50 DUP(0)
16. A. True
17. A. False
18. A. MESSAGE TEXTEQU<"I'm good at this!", 0>
19. C. assembler
20. C. var3 BYTE '$','98778' AND D. var4 BYTE 256,19,40
21. C. little endian
22. A. Constant or constant expression, C. register name, AND D. variable name (memory)

Exam 1

1. Translate the following decimal numbers to binary (in 8 bits):
    a. 73 01001001b
    b. -78 10110010b
2. Translate the following decimal numbers to hexadecimal:
    a. 75 4Bh
    b. 97 61h
3. Add the following binary integers: 00101101 and 0011101101101000b
4. Translate the following signed decimal numbers to binary (in 8 bits):
    a. -7 11111001b
    b. -1 11111111b
5. What is the decimal representation of the unsigned binary number 11111010 250d
6. What is the decimal value of the following signed binary numbers?
    a. 10110111 -73d
    b. 01101010 106d
7. Create a truth table to show all possible input and output for the boolean function described by not(a and b) Answer at the end of the exam
8. Create a truth table to show all possible input and output for the boolean function described by (not A or not B) Answer at the end of the exam
9. In real-address mode, convert the following segment offset address to a linear address: 09F0:0100 0A000h
10. Which of the following linear addresses matches the segment-offset address 08F0:0200?
    a. 09100h
    b. 09200h
    c. 0AF0h
    d. 08F2h
11. How much memory can be addressed in Protected mode?
    a. 640 K
    b. 1 MB
    c. 16 MB
    d. 4 GB
12. Which register is known as a loop counter?
    a. EAX
    b. EBX
    c. ECX
    d. EDX
13. Which directive identifies the part of a program containing instructions?
    a. .DATA
    b. .CODE
    c. .STACK
    d. .PROG

14. Which two 32-bit registers are known as extended index registers?
    a. SI, DI
    b. EAX, EBX
    c. ESI, EDI
    d. EBP, ESP
15. Which of the following are valid data definition statements that create an array of unsigned bytes containing decimal 10, 20, and 30, named **myArray**?
    a. myArray BYTE 10, 20, 30
    b. BYTE myArray 10, 20 , 30
    c. BYTE myArray[3]: 10, 20, 30
    d. myArray BYTE DUP(3) 10,20,30
16. In the following data definition, assume that **List2** begins at offset 2000h. What is the offset of the third value(5)? **List2 WORD 3, 4, 5, 6, 7**
    a. 2008h
    b. 2002h
    c. 2000h
    d. 20004h
17. Which letter choice shows the memory byte order in little endian, from low to high address, of the following data definition? **BigVal DWORD 12345678h**
    a. 56h, 78h, 12h, 34h
    b. 12h, 34h, 56h, 78h
    c. 78h, 56h, 34h, 12h
    d. 34h, 12h, 78h, 56h
18. What is the largest signed integer that may be stored in 32 bits?
    a. 2^32 -1
    b. 2^32
    c. 2^31 -1
    d. 2^31
19. Given the following array definition, which letter choice contains a valid constant declaration named **ArrayCount** that automatically calculates the number of elements in the array? **newArray DWORD 10, 20, 30, 40, 50**
    a. ArrayCount = newArray - $
    b. ArrayCount = ($ - newArray) / 4
    c. ArrayCount DWORD $ - newArray
    d. ArrayCount = (newArray - $) / 4
20. Select a data definition statement that creates an array of 500 signed doublewords named **myList** and initializes each array element to the value -1.
    a. SDWORD myList 500 DUP (-1)
    b. myList 500 SDWORD DUP (-1)
    c. myList 500 SDWORD (-1)
    d. myList SDWORD 500 DUP (-1)
21. Select a data definition statement that creates an array of 500 signed doublewords named **myList** that contains this string data? **"I'm good at this!", 0**

a. MESSAGE TEXTEQU <"I'm good at this!",0>
b. TEXTEQU MESSAGE <"I'm good at this!",0>
c. <"I'm good at this!",0> TEXTEQU MESSAGE
d. TEXTEQU <"I'm good at this!",0> MESSAGE

22. Which directive is used when defining 64-bit IEEE long reals?
    a. REAL4
    b. REAL8
    c. REAL64
    d. REAL
23. The following are both valid data definition statements:
    i. **List1 BYTE 10, 20**
    ii. **BYTE 30, 40**
    b. True
    c. False
24. The following sequence of statements is invalid:
    i. .code
    ii. mov eax, edx
    iii. .data
    iv. myByte BYTE 10
    v. .code
    vi. mov al, myByte
    b. True
    c. False
25. The EQU directive permits a constant to be redefined at any point in a program.
    a. True
    b. False
26. The four parts of a CPU are:
    a. Data bus, memory unit, control unit, arithmetic logic unit
    b. Address bus, registers, control unit, arithmetic logic unit
    c. Clock memory unit, control unit, instruction fetch unit
    d. Clock, registers, control unit, arithmetic logic unit
27. Which of the following are true about assembly language instructions and directives?
    a. A directive is executed at runtime
    b. An instruction is executed at runtime
    c. A directive is executed at assembly time
    d. An instruction is executed at assembly time
28. Which utility program reads an assembly language source file and produces an object file?
    a. Compiler
    b. Linker
    c. Assembler
    d. Loader
29. Which of the following will generate assembly errors?

    a. Var1 BYTE 1101b, 22, 35
    b. Var2 BYTE “ABCDE”,18
    c. Var3 BYTE ‘$’, ‘98778’,
    d. Var4 BYTE 256,19,40

30. The byte-ordering scheme used by computers to store large integers in memory with the high-order byte at the lowest address is called:
    a. Big endian
    b. Byte-major
    c. Little endian
    d. Byte-minor
31. Operands may be any of the following:
    a. A constant or constant expression
    b. Reserved word
    c. Register name
    d. Variable name (memory)

Truth Tables for 7 and 8

7. Not (A and B)

A	**B**	**A AND B**	**NOT (A AND B)**
T	T	T	F
T	F	F	T
F	T	F	T
F	F	F	T

8. (NOT A or NOT B)

A	**B**	**NOT A**	**NOT B**	**NOT A OR NOT B**
T	T	F	F	F
T	F	F	T	T
F	T	T	F	T
F	F	T	T	T

Quiz 3: The bane of our existence

Using the following data definitions until notified otherwise:`

```
byte1 BYTE 0FFh,1,2
byte2 BYTE 14h
word1 WORD 0FFFFh,1,2
word2 WORD 3
word3 SWORD 7FFFh, 8000h
word4 SWORD 9000h
dword1 DWORD 10h,20h,30h,40h
dArray DWORD 10 DUP(?)
```

1. Write one or more statements that move the first element of **word1** to **word2**
   Mov esi, 0
   Mov ax, word1 [esi]
   Mov word2, ax
   (many ways to do this but basically this)

2. For each of the following instructions, indicate whether it is valid (V) or invalid (I)
```
a. mov byte2,0FFh          (V)
b. mov word1, byte1        (I)
c. mov word2, 10000h       (I)
d. mov si, word1           (V)
```

3. For each of the following instructions, indicate whether it is valid (V) or invalid (I)
```
a. movzx ax, byte1         (V)
b. movzx edx, bl           (V)
c. movzx word2, al         (I)
d. movzx dl, al            (I)
```

4. Indicate the hexadecimal value of the destination operand next to each instruction. Use the letter (I) to indicate that a particular instruction is illegal and (V) if it is valid (note: all statements run together):
```
mov dx, word3          a.dx = 7FFFh
movsx eax, byte1       b.eax = 7777 7FFFh
mov dh, al             c.dh = FFh
mov bx, dx             d.bx = FFFFh
```

5. Indicate the hexadecimal value of the destination operand next to each instruction. Use the letter (I) to indicate that a particular instruction is illegal and (V) if it is valid (note: all statements run together):

```
mov ax, [word3+2]      a. 8000h
mov eax, [dword1+4]    b. 00000020h
mov al, [byte1+1]      c. 1
mov eax, [word3+4]     d. (I)
```

6. Write an instruction that moves the 32-bit address of word1 into the ESI register (assume 32-bit Protected mode).

```
MOV esi, OFFSET word1
```

7. Write an instruction that moves the lower 16 bits of dword1 into the BX register (hint: use PTR)

```
MOV bx, WORD PTR dword1
```

8. What is the value of the expression (TYPE word1)? 2

9. What is the value of the expression (LENGTHOF word1)? 3

10. What is the value of the expression (SIZEOF word1)? 6

SHORT PROGRAMMING PROBLEMS

Use the following data definitions until notified otherwise

```
byte1 BYTE 0FFh,1,2
byte2 BYTE 14h
word1 WORD 0FFFFh,1,2
word2 WORD 3
word3 SWORD 7FFFh, 8000h
word4 SWORD 9000h
dword1 DWORD 10h,20h,30h,40h
dArray DWORD 10 DUP(?)
```

11. Implement the following expression in assembly language, using 32-bi integers (you may modify any registers you wish) : `eax = dword1 + ebx - ecx`

```
MOV eax, dword1
ADD eax, ebx
SUB eax, ecx
```

12. Implement the following expression in assembly language, using 32-bi integers (you may modify any registers you wish) : `eax = -dword1 + (edx- ecx) + 1`

```
        MOV eax, dword1
        NEG eax
        MOV ebx, edx
        SUB ebx, ecx
        ADD eax, ebx
        INC eax
```

13. Implement the following expression in assembly language, using 32-bit integers. The notion dword[1] corresponds to tan array reference in C++ or Java: `dArray[0] = dArray[1] + dArray[2]`

```
mov eax, [darray + 4]
add eax, [darray + 8]
mov [darray], eax
```

14. Use the following data declarations to write an assembly language loop that copies the string from **source** to **target**. Use indexed addressing with EDI and use the LOOP instruction.

```
        source BYTE “String to be copied”,0
        target BYTE SIZEOF source DUP(0), 0

        MOV ecx, LENGTHOF source
        MOV edi, 0
        doWhile:
                MOV al, source[edi]
                MOV target[edi], al
                INC edi
                LOOP doWhile
```

15. The MOV instruction does not permit an immediate value to be moved to a segment register.
    [✓ a. True]
    b. False

16. The MOVSX instruction sign-extends an integer into a larger operand.
    [✓ a. True]
    b. False

17. Adding 0FFh and 05h in an 8-bit register set the Overflow flag.
    a. True
    [✓ b. False]

18. Adding 5 to 0FBh in an 8-bit register sets the Zero flag.

    [✓ a. True]
    b. False

19. The following instructions will set the Carry Flag:

```
mov al, 0FEh
sub al, 2
```

    a. True
    [✓ b. False]

20. The following instructions will set the Sign Flag:

```
mov al, 0FEh
sub al, 2
```

    [✓ a. True]
    b. False

21. Select the answer choice that best implements the following expression. Do not permit **dword1,** ECX, or EDX to be modified:

```
eax = -dword1 + (edx - ecx) + 1
```

```
a. mov eax, dword1
   neg eax
   sub edx, ecx
   add eax, edx
   inc eax
```

[✓
```
b. mov eax, dword1
   neg eax
   mov ebx, edx
   sub ebx, ecx
   add eax, ebx
   inc eax
```
]

```
c. neg dword1
   mov ebx, edx
   sub ebx, ecx
   add eax, ebx
   inc eax
```

```
d. mov eax, dword1
   mov edx, ebx
```

```
sub ebx, ecx
add eax, ebx
inc eax
```

Some of the following Questions have more than one correct answer. Circle all correct answers:

Use the following data definitions until notified otherwise:

```
byte1 BYTE 0FFh, 1, 2
byte2 BYTE 14h
word1 WORD 0FFFFh, 1, 2
word2 WORD 3
word3 SWORD 7FFFh, 8000h
word4 SWORD 9000h
dword1 DWORD 10h, 20h, 30h, 40
dArray DWORD 10 DUP (?)
```

22. What is the hexadecimal value of AX when this code executes?

```
mov esi, OFFSET word1
add esi, 4
mov ax, [esi]
```

   a. 1
   [✓ b. 2]
   c. FFFFh
   d. 3

23. What is the final hexadecimal value of AX when this code executes?

```
mov ebx,OFFSET dword1
sub ebx,2
mov ax, [ebx]
```

   a. 0000h
   b. 0010h
   [✓ c. 9000h]

d. 0020h

24. What is the final hexadecimal value of AL when this code executes?

```
mov ebx,OFFSET byte1
mov al, [ebx+3]
```

a. 1
b. 2
c. [✓ 14h]
d. 3

25. What is the final hexadecimal value of EAX when this code executes?

```
mov edx, 8
mov eax, dword1[edx]
```

a. 00000010h
b. 20000000h
c. 00300000h
d. [✓ 00000030h]

26. In Protected mode, which of the following define(s) a pointer variable containing the offset of **word1**?

a. [✓ ptr1 DWORD word1]
b. word1 DWORD ptr1
c. ptr2 DWORD PTR word1
d. [✓ ptr2 DWORD OFFSET word1]

Example 2

```
1:    .data
2:    varX DWORD 9,8,7,6,5,4,3,2,1,0
3:    varY DWORD (LENGTHOF varX) DUP(0)
4:    .code
5:         mov esi,OFFSET varY + (SIZEOF varX) - 4
6:         mov edi,4
7:         mov ecx,LENGTHOF varX - 1
8:    L1:  mov eax,varX[edi]
9:         mov [esi],eax
10:        add edi,4
11:        sub esi,4
12:        loop L1
```

27. Refer to Example 2. After the loop executes, what will be the values at locations varY, varY+4, and varY+8
    a. 0,0,0
    b. 0,1,2
    c. 1,2,3
    [✓ d. 0,0,1]

Use the following data for the remaining questions in this section:

```
word1  WORD  1000h,2000h,3000h,4000h,5000h
dword1 DWORD 10000h,20000h,30000h,40000h
```

28. What is the final value of AX after this code has executed?

```
     mov esi,OFFSET word1
     mov ecx,5
     mov eax,100h

L1:  add ax,[esi]
     add ax,16
     add esi, TYPE word1
     Loop L1
```

    [✓ a. F150h]
    b. 0150h
    c. F016h
    d. 0016h

29. What is the final value of AX after this code has executed?

```
     mov edx,OFFSET word1+8
     mov ecx,2
     mov ax,0

L1:  mov ax,[edx]
     add ax,20h
     sub edx,4
     Loop L1
```

a. 8040h
b. 9040h
c. 4020h
[✓ d. 3020h]

30. Suppose we want EAX to contain the sum of the **dword1** array when following (incomplete) code finishes executing:

```
1:    mov edi,OFFSET dword1
2:    mov ecx,LENGTHOF dword1
3:    ?
4:    ?
5:    ?
6:    loop L1
```

Which of the following choices would best fill in lines 3, 4, and 5?

```
a. 3:         mov eax,[edi]
   4:    L1:  add eax,dword1
   5:         add edi,2

b. 3:         mov eax,0
   4:    L1:  add eax,[edi]
   5:         add edi, TYPE dword1

c. 3:         mov eax,0
   4:    L1:  add eax,[edi]
   5:         add edi,2

d. 3:         mov DWORD PTR [edi],0
   4:    L1:  add eax,[edi]
   5:         add edi,TYPE dword1
```

Quiz 4:

1. What will be the value of EAX when the following sequence of instructions has executed?

   ```
   push 5
   push 10
   push 20
   pop eax
   ```

   EAX = 20

2. What will be the value of EAX when the following sequence of instructions has executed?

   ```
   push 5
   push 10
   pop ebx
   pop eax
   ```

   ```
   EAX = 5
   ```

3. Code a PROC declaration for a procedure named **MySub**. Use the USES operator to preserve the EAX and EBX registers.

   ```
   MySub PROC USES eax ebx
   ```

4. Code a label named **Here** that can be the target of a jump instruction originating in a different procedure.

   ```
   Here::
   ```

5. Write a sequence of statements that read a signed integer from standard input and write the same integer to standard output. Use library procedures.

   ```
   call ReadInt
   call WriteInt
   ```

6. Given the following string definition, write a sequence of statements that write the string to standard output. Use a library procedure.

   ```
   mov EDX, OFFSET str1
   call WriteString
   ```

7. Given the following data definition, write a sequence of statements that display a dump of the data in 32-bit hexadecimal, using the DumpMem procedure from the link library.

```
array DWORD 10h, 20h, 30h, 40h
mov esi, OFFSET array
mov ecx1, LENGTHOF array
mov ebx, TYPE array
call DumpMem
```

8. Write statements that use library procedures to generate a single unsigned pseudorandom integer between 0 and 999 and write it to standard output:

```
mov eax, 1000
call RandomRange
call WriteDec
```

9. Which of the following code sequences assigns the value 10h to EBX?

```
a. mov edx,20h
   push edx
   mov ecx,10h
   push ecx
   pop ebx
   pop edx

b. mov ecx,10h
   mov edx,20h
   push ecx
   push edx
   pop ebx
   pop edx

c. push 20h
   mov ecx,10h
   push ecx
   pop eax
   pop ebx

d. push 30h
   push 10h
   push 20h
   pop edx
   pop ebx
   pop eax
```

Exam 2

```
byte1  BYTE  0FFh, 1, 2
byte2  BYTE  14h
word1  WORD  0FFFFh, 1, 2
word2  WORD  3
word3  SWORD 7FFFh, 8000h
word4  SWORD 9000h
dword1 DWORD 10h, 20h, 30h, 40h
dArray DWORD 10 DUP (?)
```

1. Where marked by a letter (a, b, c, d), indicate the hexadecimal value of the destination operand:

```
mov ax,word1
inc ax                  a. 0000h
dec ax                  b. FFFFh
mov ax,word3
neg ax                  c. 8001h
add ax,0C2A5h           d. 52A4h
```

2. For each of the following instructions, indicate whether it is valid (V) or invalid (I). (Note: each statement are independent.)

```
a. mov byte2,0FFh       (V)
b. mov word1,byte2      (I)
c. mov word2,100000h    (I)
d. mov si,word1         (V)
```

3. For each of the following instructions, indicate whether it is valid (V) or invalid (I).

```
a. movzx ax,byte1       (V)
b. movzx edx,bl         (V)
c. movzx word2,al       (I)
d. movsx dl,al          (I)
```

4. Indicate the hexadecimal value of the destination operand next to each instruction. Use the letter (I) to indicate that a particular instruction is illegal.

```
mov dx,word3            a. 7FFFh
movsx eax,byte1         b. FFFF FFFFh
mov dh,al               c. FFh
mov bx,dx               d. FFFFh
```

5. Indicate the hexadecimal value of the destination operand next to each instruction. Use the letter (I) to indicate that a particular instruction is invalid.

```
mov ax,[word+2]        a. 8000h
mov eax,[dword1+4]     b. 20h
mov al,[byte1+1]              c. 1
mov eax,[word3+4]      d. Invalid
```

6. In the following instruction sequence, show the values of the Carry, Zero, Sign, and Overflow flags where indicated.

```
                           ZF   CF   SF   OF
mov al,7Fh
add al,2              a.   0    0    1    1
sub al,5              b.   0    0    0    1
mov al,80h
add al,80h            c.   1    1    0    1
neg al                d.   1    0    0    0
```

7. What will be the value of EAX when the following sequence of instructions has executed?

```
push 5
push 15
push 10
pop ebx
pop eax
pop ecx

EAX = 15
```

8. What will be the value of ECX when the following sequence of instructions has executed?

```
push 5
push 10
pop ebx
pop eax
pop ecx

It cannot be determined.
```

9. Write an instruction that moves the lower 8 bits of word2 into the AL register.

```
mov al, PTR BYTE word2
```

10. Write an instruction that moves EBX to the location starting from word1

```
        mov DWORD PTR word1, ebx
```

11. Write one or more statements that move the first element of word1 to word2.

```
        mov ax,word1
        mov word2,ax
```

12. Implement the following expression in assembly language, using the 32-bit integers (you may modify any registers you wish):

```
        eax = -dword + (edx - ecx) + 1
```

```
        neg dword1
        sub edx,ecx
        add edx,dword
        inc edx
        mov eax,edx
```

13. Implement the following expression in assembly language, using 32-bit integers. The notation dword[1] corresponds to an array reference in C++ or Java.

```
        dArray[0] = dArray[1] + dArray[2]
```

```
        mov eax,[dArray + 4]
        add eax,[dArray + 8}
        mov dArray,eax
```

14. Use the following data declarations to write an assembly language loop that copies the string from source to target. Use indexed addressing with EDI, and use the LOOP instruction.

```
        source BYTE "String to be copied",0
        target BYTE SIZEOF source DUP(0),0
```

```
        mov esi,OFFSET source
        mov ecx,LENGTHOF source
 L1: mov target[esi],source[esi]
        mov esi,TYPE source
        LOOP L1
```

15. Code a label named Here that can be the target of a jump instruction originating in a different procedure.

```
        Here::
```

16. Given the following string definition, write a sequence of statements that write the string to standard output. Use a library procedure.

```
str1 BYTE "Display this string",0

mov edx,OFFSET str1
call WriteString
```

17. Given the following data definition, write a sequence of statements that display a dump of the data in 32-bit hexadecimal, using the DumpMem procedure from the link library.

```
array DWORD 10h,20h,30h,40h

mov esi,OFFSET array
mov ecx,LENGTHOF array
mov edx,TYPE array
call DumpMem

*Translation:
1. Load up the offset of the array into esi
2. Load the length of the array into ecx
3. Load the type of the array into edx
4. Call DumpMem
```

18. Write statements that use library procedures to generate a single unsigned pseudorandom integer between 0 and 999 and write it to standard output.

```
mov eax,1000
call RandomRange
Call WriteDec
```

19. Which of the following code sequences assigns the value 10h to EBX?

```
a. mov edx,20h
   push edx
   mov ecx,10h
   push ecx
   pop ebx
   pop edx
b. mov ecx,10h
   mov edx,20h
   push ecx
   push edx
   pop ebx
   pop edx
c. push 20h
```

```
   mov ecx,10h
   push ecx
   pop eax
   pop ebx
d. push 30h
   push 10h
   push 20h
   pop edx
   pop ebx
   pop eax
```

<u>Example 2</u>

```
.data
varX DWORD 9,8,7,6,5,4,3,2,1,0
varY DWORD (LENGTHOF varX) DUP(0)
.code
          mov esi,OFFSET varY + (SIZEOF varX) - 4
          mov edi,4
          mov ecx,LENGTHOF varX - 1
L1:       mov eax,varX[edi]
          mov [esi],eax
          add edi,4
          sub esi,4
          loop L1
```

20. Refer to example 2. After the loop executes, what will be the values at locations varY, varY+4, and varY+8?

a. 0,0,0
b. 0,1,2
c. 1,2,3
[✓ d. 0,0,1]

21. Refer to example 2. After the loop executes, what will be the values in the last three positions (array elements) of varY?

a. 0,0,0
b. 8,9,0
[✓ c. 6,7,8]
d. 7,8,9

22. Refer to example 2. If line 9 were changed to the following, what would be the final values at location varY, varY+4, varY+8?

```
          mov [esi-4],eax
```

a. 0,0,0
[✓ b. 0,1,2]
c. 1,2,3
d. 0,0,1

```
          word1  WORD  1000h,2000h,3000h,4000h,5000h
          dword1 DWORD 10000h,20000h,30000h,40000h
```

23. What is the final value of AX after this code has executed?

```
          mov esi,OFFSET word1
          mov ecx,5
          mov eax,100h

     L1:  mov ax,[edx]
          add ax,16h
          add esi,TYPE word1
          Loop L1
```

[✓ a. F150h]
b. 0150h
c. F016h
d. 0016h

24. What is the final value of AX after this code has executed?

```
          mov edx,OFFSET word1+8
          mov ecx,2
          mov ax,0

     L1:  mov ax,[edx]
          add ax,20h
          sub edx,4
          Loop L1
```

a. 8040h
b. 9040h
c. 4020h
[✓ d. 3020h]

25. Suppose we want EAX to contain the sum of the dword1 array when the following (incomplete) code finishes executing:

```
1:    mov edi,OFFSET dword1
2:    mov ecx,LENGTHOF dowrd1
3:    ?
4:    ?
5:    ?
6:    loop L1
```

Which of the following choices would best fill in lines 3,4, and 5?

```
a. 3:    mov eax,[edi]
   4:L1: add eax,dword1
   5:    add edi,2

[✓ b.] 3:    mov eax,0
   4:L1: add eax,[edi]
   5:    add edi,TYPE dword1

c. 3:    mov eax,0
   4:L1: add eax,[edi]
   5:    add edi,2

d. 3:    mov DWORD PTR [edi],0
   4:L1: add eax,[edi]
   5:    add edi,TYPE dword1
```