Quiz 1:

* The 2's complement of  1111 0000 is done by simply reversing the bits to get 0000 1111
  + False
  + To take the 2's complement, invert each bit and then add 1 to the inverted result.  
    Therefore the answer is 0000 1111 + 1 = 0001 0000
* The byte with a value of 1001 1100 can be a negative number or a positive number
  + True
  + The binary value can be a positive or negative value, depending on how it is used in the program.  
    If it is a signed integer, it is a negative value because the MSB (sign bit) is 1.  
    If it is an unsigned integer, it is large a positive value.
* The number -123,456 can be stored in one WORD.
  + False
  + The range for a signed WORD is -32,768 to +32,767
* Converting data from binary to decimal format is the same for signed vs. unsigned data
  + False
  + The conversion steps are not the same for signed vs. unsigned data. For signed data, if the value is negative, its absolute value must be calculated first.
* Each assembly language instruction corresponds to one machine language instruction that the CPU runs
  + True
  + There is a one-to-one correspondence between assembly language instructions and machine code instructions. Assembly language is simply an easier-to-read format of machine code.
* Every binary number is a sum of the powers of 2, such as 1, 2, 4, 8, 16...
  + True
  + Each bit in a binary number is a power of 2, therefore the binary number is the sum of the powers of 2
* The binary representation of decimal 33 is 0010 0001
  + True
  + 33 / 2 = 16 R 1, 16 / 2 = 8 R 0, 8 / 2 = 4 R 0, 4 / 2 = 2 R 0, 2 / 2 = 1 R 0, 1 / 2 = 0 R 1  
    Reading the remainder value from LSB to MSB: 10 0001
  + Zero filled to make up 8 bits: 0010 0001
* The CPU only works with binary data, whether the data is a text string or a numeric value
  + True
  + Data is always binary on a computer system. Both numeric data and text are stored as a sequence of 1's and 0's.
* The byte 1001 0001 is always a negative value.
  + False
  + It is not possible to determine whether a binary data value is positive or negative, unless we also know whether it's signed or unsigned data.  
    If 1001 0001 is unsigned data, then it's a positive value.  
    If 1001 0001 is signed data, then it's a negative value.
* Each high level language statement is translated to one machine code instruction
  + False
  + A high level language statement typically is translated into multiple machine code instructions

Quiz 2

* The step to fetch an operand is always necessary in the instruction cycle.
  + False
  + There is no need for an operand fetch if the operand is already in a register or is an immediate value. Operand fetch is only for data that are in memory.
* The EIP register is updated when an instruction is fetched.
  + True
  + The EIP points to the next instruction that needs to be executed. During the instruction fetch, the CU fetches the instruction that has the address in EIP. After the instruction is fetched, the EIP is updated to point to the next instruction.
* The value stored in the flag register is for one specific status flag.
  + True
  + Each bit in the flag register is for one status flag, so the flag register (EFLAGS) stores the values of many flags.
* In an instruction cycle, the operands are fetched before the instruction is fetched.
  + False
  + The instruction is fetched first and then decoded, and then the operand is fetched if needed. Some operands do not need to be fetched.
* The total memory used by all running programs can never be larger than the computer's physical memory.
  + False
  + Virtual memory means the memory used can actually be larger than the physical memory available on the system. Paging swaps in and out the data on the hard disk into RAM so that the program 'thinks' that there is a large size memory.
* The control unit (CU) coordinates the sequencing of execution steps in an instruction cycle.
  + True
  + The CU coordinates the steps of the instruction cycle
* The following data locations are in order of fastest access time to slowest access time: cache, registers, main memory
  + False
  + Data access is fastest at the registers, then in cache, and then in main memory
* The segment registers are considered general purpose registers.
  + False
  + The segment registers hold the address of the code segment, data segment, or stack segment.  They are maintained by the OS, and they are not general purpose registers that a programmer can use.
* All CPU and bus operations are synchronized to the clock cycle.
  + True
  + CPU and bus operations occur during a rising edge or falling edge of the clock signal so that they can be synchronized to work in the correct order
* It takes multiple clock cycles to access data from memory.
  + True
  + Data access from memory takes multiple cycles because the address and the control signals have to be sent out and decoded, and then data have to be loaded on to the bus and read in

Quiz 3:

* It is possible for a hexadecimal value to be an immediate value in a program
  + True
  + An immediate value is a literal constant, and constants can be in base 2 (binary), base 10 (decimal) or base 16 (hexadecimal) in a program
* A label represents an address of a memory location, and the memory location can be in the code segment or the data segment
  + True
  + A data label (variable name) is the address where the data is stored in memory in the data segment. Likewise, a code label is an address of a particular line of code in memory in the code segment.
* A directive is an instruction that the CPU executes.
  + False
  + A directive is an instruction that the CPU executes.
* String constants can be in single quotes
  + True
  + String constants can be in single quotes or double quotes
* WORD, DWORD, SBYTE are assembly data types that are used by all assemblers
  + False
  + WORD, DWORD, SBYTE are MASM specific data types. They get translated into the corresponding 16-bit, 32-bit, 8-bit data types that the CPU uses
* count = 7 \* 12 is a valid use of the = directive
  + True
  + This is a valid use of the = directive. The assembler calculates the product and uses it as the value for count in the translated code
* A character constant must be enclosed in single quotes only
  + False
  + Character constants can be in single or double quotes
* greeting byte “Hello”,0dh, 0ah, 0 is a valid data definition
  + true
  + Characters are stored as bytes, so the string greeting is made up of the characters: h, e, l, l, o, carriage return, line feed, null termination
* A code label must end with a colon, which is the symbol :
  + True
  + A code label must end with a colon, a data label does not end with a colon
* data WORD 12345h, 67890h is a valid data definition statement
  + false
  + 12345h and 67890h is larger size than a WORD

Quiz 4:

* The MUL instruction sets the Overflow flag if the upper half of the product is equal to zero.
  + False
  + CF is set when there the upper half of the product is non-zero
* If BL = 10h, then after the instruction: MOVSX AX, BL  
  AX = FF10h
  + False
  + If BL = 10h, then the MSB of BL is 0, which means AX will be 0010h
* The instructions:  
  **mov bx, -4  
  movsx eax, bx**  
    
  will produce FFFF FFFCh in EAX
  + True
  + movsx will extend the sign of BX into the upper word of EAX, and since BX is negative, the sign bit is 1. This means the upper half of EAX will be all 1's, or all F in hex
* Given the definition: arr DWORD 10,20,30,40,50,60  
  *arr + 4* will access the value 5
  + False
  + A DWORD is 4 bytes, so arr + 4 is the value 20
* The following is the correct way to implement the division:  -15 / 3  
  **mov dx, 0  
  mov ax,-15  
  mov bx, 3  
  idiv bx**
  + False
  + Since this is a signed division, use cwd to sign extend from AX to DX. Don't zero fill DX because it turns the dividend in DX:AX into a positive number
* The following instructions will set the Carry flag:  
  **mov al,125  
  sub al, 4**
  + False
  + CF is not set because 125 + (-4) = 0111 1101 + 1111 1100 = 0111 1001 with a carry of 1. But SUB means that the CF value is the reverse of the carry, so CF = 0.  
    Also, to explain it in another way: 125 - 4 = 121, which is within range of a byte and the result is valid if the values are unsigned. Since the result of an unsigned operation is valid, CF must be 0.
* After the following instructions have executed:  
  **mov dx,0  
  mov ax,123h  
  mov bx,10h  
  div bx**  
    
  Then DX=0003h and AX=0012h.
  + True
  + 123h/10h = 12h with remainder 3h. The remainder 3 is in DX, the quotient 12 is in AX
* When the MUL CL instruction executes, the 16-bit product ends up in the AX register
  + True
  + The operand CL causes an 8-bit multiplication to take place (AL \* CL), resulting in a 16 bit product in AX
* The instruction IDIV CX means that the quotient will be a 16 bit value
  + True
  + Since CX is 16 bits, the dividend will be a 32-bit value in DX:AX, which means the quotient will be 16 bits and in AX, and the remainder is also 16 bits and in DX
* The following instructions will set the Overflow flag:  
  mov al,125  
  sub al, 4
  + False
  + OF is not set because 125 + (-4) = 0111 1101 + 1111 1100 = 0111 1001 with a carry out of MSB and a carry into MSB. This means 1 XOR 1 = 0 = OF. Another way of looking at the subtraction: 125 - 4 = 121, which is within the range of a signed byte. This means the result is valid and therefore OF has to be 0
* 