The actual processing of information in the computer is **NOT** done in the control unit

**Chapter 1**

* 1. A higher-level programming language can instruct a computer to compute more than a lower-level programming language
     + **FALSE**

* 1. Every computer can do the same thing as every other computer. A smaller or slower computer will just take longer.
     + **TRUE**

* 1. Which  characteristic of natural languages that prevents them from being used as programming languages?
     + **AMBIGUITY**

* 1. Which are the characteristics of algorithms? Answer can be more than one.
     + **Definiteness**
     + **Effective Computability**
     + **Finiteness**

* 1. Two computers, A and B, are identical except for the fact that A has a subtract instruction and B does not. Both have add instructions. Both have instructions that can take a value and produce the negative of that value. Which computer is able to solve more problems ?
  + **both**

* + The advantage of programming in a higher-level language compared to a lower-level language is
  + Easier to read
  + Fewer instructions are required to do the same amount of work
  + **all of the above**

* + Select those things speciﬁed by an ISA.  Answer can be more than one.
    - **Operations**
    - **Data types**
    - **Addressing Modes**

* 1. How many ISAs are normally implemented by a single microarchitecture?
     + **1**

* 1. How many levels of transformation?
     + **7**

* 1. Say you go to the store and buy some word processing software. What form is the software actually in?
     + **ISA of the computer on which you'll run it**

**Chapter 2**

* 1. If numbers are to be represented with 10 bits, we can represent more unique numbers if we use an unsigned integer representation than if we use signed integer representations
     + **False**

* 1. The 2's complement representation for -2 is 111111111110, if we have 12 bits available to represent signed integers.
     + **True**

* 1. If we add the 2's complement representation of a number to itself, and if the sign bit does not change, the result will simply be the same as if we had shifted the original number one bit to the left, and inserted a 0 at the right end.
     + **True**

* 1. A and B are integers expressed in 2's complement.  If both are positive, and their sum produces a result that is the representation of a negative number, we know we MUST have caused an overflow.
     + **True**

* 1. A AND B is true if at least one of the two variables A, B is true.
     + **False**

* 1. A XOR B is true if one of A,B is true and one of A,B is false.
     + **True**

* 1. The logical NOT function is a unary function, because it has only one argument.
     + **True**

* 1. Using DeMorgan's Theorem we can convert any AND-OR structure into
     + **NAND-NAND**

* 1. The decimal digit 5 can be represented as a bit string:
  + 000000000101
  + 000101
  + 00000101
    - **ALL OF THE ABOVE**

* + If we add two negative integers, represented in 2's complement notation, and the result is positive, we know:
    - **that we must have caused an overflow condition**

* 1. Two values A = 11111111111111111111111010101 and B = 1111010101 are representations of 2's complement integers.
     + **A and B are EQUAL**

* 1. If A is true, and B is false, the expression NOT(A) OR B is
     + **FALSE**

* 1. In hex notation, each digit can have a value in the range:
     + **0 to 15**

* 1. For the value of A OR B to be true,
     + **At least one of the two values A, B is true**

* 1. In 2's complement arithmetic, an overflow occurs if:
     + *two positive numbers produce a negative result*
     + *two negative numbers produce a positive result*
     + **ALL OF THE ABOVE**

* 1. With 12 bits, we can represent uniquely:
     + **exactly 4096 distinct items**

* 1. With k bits, we can uniquely represent ? distinct element
     + **2k (2^k***) (address space)*

* 1. Add the following bit patterns. Leave your results in binary form.

0101 + 0110

* + **01011**

* + The hexadecimal representation of the binary string 0000 1111 0010 1010
    - **0F2A**

**Chapter 3**

* 1. It is possible to have a 5 input AND gate
  2. **TRUE**

* 1. An 8-bit register contains the value 5. The value 1 is written into it. The original value 5 can still be recovered
  + **FALSE**

* + It is possible to have circuits that contains both P-type and N-type transistors.
  + **TRUE**

* + It is possible for a 3:8 decoder to have 3 out of 8 inputs asserted
  + **FALSE**

* + If a memory has addressability of 4 bits, then we need 2 bits to speccify the address space
  + **FALSE**

* + Addressability of a machine can never be greater than its address space
  + **FALSE**

* + For a logic circuit to work as a storage element it is necessary that the output be fed back to the input
  + **TRUE**

* + It is not possible to have a 1 bit register
  + **FALSE**

* + The two outputs of a latch are always the complement of each other
  + **TRUE**

* + For a memory with a 16-bit address space, the addressability is:
  + **CANNOT BE DETERMINED**

* + Because we wish to allow each ASCII code to occupy one location in memory, most memories are \_\_\_\_ addressable
  + **BYTE**

* + When the write enable input is not asserted the gated D-latch \_\_\_\_\_\_ its output
  + **CANNOT CHANGE**

* + A structure that stores a number of bits taken "together as a unit" is a
  + **REGISTER**

* + We say that a set of gates is logically complete if we can build any circuit without using any other kind of gates. Which of the following sets are logically complete?

* + **set of {AND, OR, NOT}**

* + Of the following circuits, the one which involves storage is:
  + **RS LATCH**

* + Which of the following pair of gates can form a latch?
  + **A PAIR OF CROSS COUPLED NAND**

* + Which of the following conditions is not allowed in an RS Latch?
  + **R IS NEGATED, S IS NEGATED**

* + A ???? is used to select one of many inputs
  + **MUX (Multiplexer)**

* + If A[15:0] = 1000110001110001, A[12:9] =
  + **0110**

* + One can write into a gated D latch only while the \_\_\_ signal is asserted
  + **WE (Write Enable)**

**Chapter 4**

* + The Function of the Instruction Register (IR) is to point to the next instruction to be processed
  + **FALSE**

* + The actual processing of information in the computer is done in the control unit
  + **FALSE**

* + Most ISAs provide some small temporary storage very close to the ALU to allow the results to be temporarily stored if they will be needed to produce results in the near future
  + **TRUE**

* + The time taken to access registers is less than the time taken to access memory
  + **TRUE**

* + Some instructions to not require the FETCH phase
  + **FALSE**

* + The decode phase is required of all instructions
  + **TRUE**

* + The evaluate address phase can happen before the instruction decode phase
  + **FALSE**

* + For the LC-3 ISA, the ADD instruction does not require an EVALUATE ADDRESS phase.
  + **TRUE**

* + A computer memory contains:
  + **BOTH INSTRUCTIONS AND DATA**

* + The Von Neumann model consists of \_\_\_\_ parts
  + **5**

* + Typically the size of each register is \_\_\_\_\_\_ the size of values proessed by the ALU
  + **IDENTICAL TO**

* + The Decode phase of the Instruction Cycle examines the \_\_\_\_\_ part of the instruction:
  + **OPCODE**

* + The sequence of an Instruction Cycle is:
  + **FETCH -> DECODE -> EVALUATE ADDRESS -> FETCH OPERANDS -> EXECUTE -> STORE RESULT**

* + The \_\_\_\_\_\_ Instructions are used to change the sequence of instruction execution
  + **CONTROL**

* + ADD is an example of a/an
  + **OPERATE INSTRUCTION**

* + Control Instructions change the value of the PC in the
  + **EXECUTE phase**

* + Which of the following instructions needs an EVALUATE ADDRESS phase:
  + **LD**

* + Of the following devices, \_\_\_\_\_ is an example of an *output* device
  + **PRINTER**

* + in the FETCH phase, an instruction is transferred from \_\_\_\_\_ to the Instruction Register
  + **MEMORY**

* + If memory accesses take 100 times more than register accesses, then a LD instruction will take \_\_\_\_\_ machine cycles than a JMP R3 instruction
  + **MORE**

**Chapter 5**

* + The data type supported by the LC-3 ISA is 1's complement integers
  + **FALSE**

* + For an LC-3 instruction that uses the PC-Relative addressing mode, bits [15:9] of the address of the operand are the same as bits [15:9] of the address of the instruction.
  + **FALSE**

* + The TRAP instruction changes the PC to a memory address, which is part of the operating system.
  + **TRUE**

* + Every LC-3 operate and data-movement instruction either reads from or writes into the register file in the LC-3 datapath.
  + **TRUE**

* + Which of the following instructions can reference a memory location that is #1000 locations from the instruction?
  + **STR**

* + The *addressability* of LC-3 memory is:
  + **16 bits**

* + For which of the following LC-3 instructions is no memory access performed during the instruction cycle?
  + LDR
  + STI
  + LEA
  + **FOR ALL OF THE ABOVE instructions memory is accessed during the instruction cycle**

* + Which of the following LC-3 instructions does not access memory to fetch its operand?
  + **LEA**

* + The LC-3 ADD instruction:
  + **can add a 16-bit value to a sign-extended 5-bit value**

* + If the value stored in R0 is 1 at the end of the execution of the following instructions, what can be inferred about R5?

Address                  Instruction

-------                     ------------------------

0x3000                  0101000000100000

0x3001                  0101100101100001

0x3002                  0000010000000001

0x3003                  0001000000100001

* + **-R5 IS ODD**

* + \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ change the sequence of instructions that are executed
  + **CONTROL INSTRUCTIONS**

* + The number of system calls that could be uniquely identified using a trapvector of 12 bits would be
  + **4096**

* + The LC-3 operate instructions that perform binary operations are:
  + **ADD, AND**

* + If an ISA has 256 opcodes, the number of bits in the instruction encoding used to specify the opcode should be at least
  + **8 BITS**

* + The LEA instruction is used to write a \_\_\_\_\_\_ into a register
  + **ADDRESS**