1. If your data in memory is word-aligned,
   A. you are working with an integer array.
   B. you are not on a Mac.
   C. the 2 LSB's of the data values are zeros.
   **D. four divides each data's address value.**
   
   Addresses are aligned, not data at the address.

2. Your computer stores memory addresses in little-endian format.
   Which of the following could be a valid address of a MIPS instruction?
   A. 00101010011010100101010010101100
   B. 11110011000000110000110011111110
   C. 11001100010011000100110001000001
   D. 0000000010101110101110101011100

   **Mips instructions are word-aligned, and thus their address ends in 00.**
   Little-endian puts that ending byte first, so the **first** byte will end in 00.

   No students answered this question correctly, so we will discuss it on Monday.
   Send me your feedback please.

3. Why are CISC processor architectures popular?
   A. More processing can be done with fewer instructions.
   B. Because they operate on CISCO routers.
   C. Programs run faster than on a RISC processor.
   D. The architecture emphasizes software.

4. Comparing MIPS and x86 architectures reveals that...
   A. MIPS memory is byte-addressable, while x86 memory is bit-addressable.
   B. Fewer instructions result in smaller programs. **<--no, larger.**
   C. MIPS uses a stack, while x86 uses a queue.
   **D. MIPS has fewer instructions than x86.**

5. When a program is executing...
   A. each machine instruction is placed on the stack.
   B. machine instructions are copied into $pc just before execution.
   **C. machine instructions can be found in main memory.**
   D. the program counter is incremented by 4 after each instruction.

6. The instruction that follows a jump instruction in memory...
   A. cannot be another jump or branch instruction.
   B. **remains in the pipeline, and is executed.**
   C. is not executed, because the $pc was set to the jump address.
   D. is an instruction that does "no operation".
7. When a MIPS j instruction is executed, the jump address contained in the machine instruction...

   A. is copied to the $pc register. <-- not directly copied.
   B.. *is bit-shifted 2 to the left.* <-- among other things.
   C. is bit-shifted 2 to the right.
   D. is added to the 4 MSB's of the program counter.

8. This command: sll $0, $0, 0

   A. stores the local address of a long-jump address.
   B. should not be used after a jump or branch instruction.
   C. does nothing.
   **D.. ** *is safe to use anywhere in your program.*
   E is skipped by the branch controller.

Wikipedia says a nop instruction does nothing. It also says it is used for memory alignment and timing operations. That's nothing? sll is executed by the processor, takes a slot in the pipeline, and causes an increment to $pc.
Half the class chose C, so I will allow answers C and D.

9. Your program is running when it encounters a jump instruction.
Here is your encoded j instruction: 00001001101010001100101010001110
Here is the current program counter: 10011100101111000000010101100010
What address will your program jump to?

   A. 00000001101010001100101010001110
   B.. **10010110101000110010101000111000**  <--------+
   C. 00000110101000110010101000111000
   D. 00011010100011001010100011101001  |  
   |  
First 4 bits of $pc + address + 00. There it is ------+

10. What is the value in register $t2 after the following code executes.

    addiu $t0, $0, -15
    addiu $t1, $0, 5
    sltu  $t2, $t0, $t1

   A. -10
   B. 10
   C. 1
   **D..0**

11. Which of the following formulas can be used to express a computer's performance ability?

   A. time/program = cycle/time x cycles/instruction x instructions/program
   **B.. time/program = time/cycle x cycles/instruction x instructions/program**
   C. time/program = time/cycle x instructions/cycle x instructions/program
   D. time/program = time/cycle x cycles/instruction x program/instructions