

Solution:

Solution:

1 Binary Warmup

(12 points)

For each of the following statements, determine whether it is either **true** or **false**. (1 point each)

1. The addition of two n -bit numbers produces at most an $n + 1$ -bit result.

Solution: True. Not sure what else to say...

2. All combinatorial circuits can be implemented purely out of AND gates (using no other gates).

Solution: False. The AND operation alone is not a complete boolean base.

3. The number 10001011 in binary is -117 in decimal (signed twos-complement interpretation).

Solution: True. What can I say?

4. One problem with ones-complement representation is that it can only represent even numbers.

Solution: False. Not sure what else to say...

5. A pseudo-instruction directs the processing of source code by the assembler.

Solution: False. A pseudo-instruction is an instruction the assembler pretends to know but the underlying hardware doesn't; the assembler translates it to actual machine instructions.

6. Using memory-mapped I/O we can keep the address bus free for RAM and ROM exclusively.

Solution: False. The idea of memory-mapped I/O is to reserve some address space for things besides RAM and ROM, namely I/O devices (or rather their controller chips).

7. An overflow condition cannot occur if we add a positive and a negative number.

Solution: True. Not sure what else to say...

8. Abstraction means ignoring certain details in order to focus more clearly on important issues.

Solution: True. Not sure what else to say...

9. The number 10001011 in binary is 139 in decimal (unsigned twos-complement interpretation).

Solution: True. What can I say?

10. The circuit for a half adder has three inputs to add two digits and a carry.

Solution: True. Not sure what else to say...

11. The `nop` or no-operation instruction is 100% useless.

Solution: False. There are a variety of places where the instruction comes in handy, for example to reserve space inside a piece of code to later modify that location (“self-modifying” code).

12. One problem with electronic voting systems is that they concentrate the elections in fewer hands.

Solution: True. It’s of course also true of mechanical voting systems to some degree. . .

Solution:

2 Tough Choices (8 points)

For each of the following questions, circle **one** answer out of the choices given. (2 points each)

1. Consider the **exclusive or** (XOR) gate. Its output A is high if and only if its two inputs X and Y are different. Which of the following is the **disjunctive normal form** (DNF) for the XOR?

- (a) $A = X \cdot Y' + X' \cdot Y$
- (b) $A = (X + Y)'$
- (c) $A = (X' + Y') \cdot (X + Y)$
- (d) $A = (X \cdot Y)'$
- (e) None of the above.

Solution: That would be (a).

2. Which of the following **most accurately** describes what an **interrupt** is?

- (a) An interrupt is a state in which the processor does not know what to do.
- (b) Interrupts were used in the 1950s to allow operators to stop overheating machines.
- (c) Interrupts allow processors to react to asynchronous events such as network traffic.
- (d) All of the above.
- (e) None of the above.

Solution: That would be (c).

3. The **range of values** representable by a **signed twos-complement 16-bit register** is which of the following?

- (a) -2^{16} to 2^{16}
- (b) -2^{15} to 2^{15}

- (c) $-2^{15} - 1$ to 2^{15}
- (d) -2^{15} to $2^{15} + 1$
- (e) None of the above.

Solution: That would be (e). Confused? Not really! The range should be -2^{15} to $2^{15} - 1$ instead.

4. Which of the following **cannot** be easily encoded for processing by a computer?
- (a) Speed of a car.
 - (b) Length of a wall.
 - (c) Image from a television camera.
 - (d) Signals from a combinatorial circuit.
 - (e) None of the above.

Solution: That would be (e).

Solution:

3 Short Answer (8 points)

For each of the following questions, answer in **one to three** sentences, the shorter the better. (2 points each)

1. You need to multiply the value in register `$s0` by 10 (that's "ten"). Give (or describe) a short code sequence that will perform this operation **without** using a `mul` or `mult` instruction.

Solution: We have $x \times 10 = x \times (8 + 2) = x \times 8 + x \times 2 = x \times 2^3 + x \times 2^1$; multiplication by powers of 2 can be done by shifting, so we get the following code sequence:

```
sll $s1, $s0, 3
sll $s2, $s0, 1
add $s0, $s1, $s2
```

We could also replace the second `sll` with an `add` instruction, but that's probably not faster (after all the `add` has to propagate a carry).

2. Show how the AND gate can be implemented in terms of **only** NOR gates (no other gates allowed).

Solution: Sorry, no time to draw circuits. :-/

3. Explain the notion of **sign extension** as well as you can. When is it relevant? How is it performed? Give an example as well.

Solution: Sign extension is necessary when we want to move (or otherwise use) a quantity from a smaller to a bigger register. For a negative quantity, all "new" bits should be one, for a positive quantity all "new" bits should be zero. For example, when adding -1 stored in 8 bits to 10 stored in 32 bits, we need to "pad" the representation of -1 with 24 one bits before performing a 32 bit addition.

4. I claimed that we can divide a signed number by a power of 2 using the `sra` (shift right arithmetic) instruction. Consider $-14/2$ for example: Shifting 10010 (which is -14) to the right one bit yields 11001 (which is -7); note that the result for $14/2$ is also 7. However, things go less smooth if we try to compute $-7/2$ this way; compare the result to $7/2$ computed the same way. Explain the problem and outline a solution.

Solution: The problem is that we “shift out” a one bit while dividing a negative number. Instead of “rounding to 0” (which is what we would want) this ends up rounding to negative infinity. We can fix things by adding one to the result.

Solution:

4 Circuit Design

(12 points)

Design a 2-to-4 decoder circuit that sets **one of four outputs** to high depending on the binary pattern supplied on the **two inputs** (the other outputs should be low); use only NOT, AND, and OR gates. You can use whatever design techniques you feel comfortable with, but you should give me more than just the finished circuit. A block diagram and some truth tables would be nice for example.

Solution: Sorry, no time to draw this. . .

Solution: