CS 4, Winter 2007: Example Final Questions

This is not a practice final, but rather just a collection of assorted questions, illustrative of the types of questions that will be asked. Some of these are relatively short answers, and some require a bit more, so their point values would vary. Answering these questions is voluntary: do so only if you feel it is helping you with the final. We will not grade this.

1. What is "Debugging"? How does it fit in the process of writing software? What are the other steps?
2. When writing large programs, what is modularity? Why do we employ it?
3. We have used many libraries in JavaScript (the DOM being the main example) and treated them as "black boxes". Why do we need to introduce these abstractions? Are there other examples you can think of in Computer Science? And in other sciences or math?
4. Pick two positive numbers smaller then 100. Write them in binary, and perform binary addition on them, showing carry bits. To verify your answer, compute the decimal value of the sum. If you want exercise more, just change the numbers and repeat.
5. What is "Two's complement"? Is this the only way to represent negative integer values in binary? If there is more than one way to do so, how does the computer know which one to pick?
6. How is text represented in binary?
7. Write the truth table and draw a circuit to do the following. There are three inputs, A, B, and C, and one output, D. D is true if and only if A is false and C and B are not equal. If you want exercise more, just change the truth table and repeat.
8. Why did we introduced gates (AND, NOT, OR)? Are switches and wires not enough to build every computer?
9. Why do we introduce the von Neumann machine model? How is memory described in it? What is the function of the control unit?
10. What is the "address space"? Why is it important to know how large an address space your machine can index?
11. What is the largest integer you can represent with 4, 8, and 16 bits?
12. Translate the following algorithm, to find the smallest of a set of numbers, to the assembly language covered in class and in the textbook. The function `read` inputs a value, `print` outputs it. Your code should use labels and .DATA pseudo-ops to define the locations of the variables and jump targets. If you want exercise more, just change the code and repeat.

```
n = read(); // assume it's at least 1
min = read();
while(n > 1) {
  y = read();
  if(y < min) {
    min = y;
  }
  n = n - 1;
}
print(min)
```

For your reference, here are the instructions in this assembly language.

Instruction	Function
LOAD x	R = M[x]
STORE x	M[x] = R
CLEAR x	M[x] = 0
ADD x	R = R + M[x]
INCREMENT x	M[x] = M[x] + 1
SUBTRACT x	R = R - M[x]
DECREMENT x	M[x] = M[x] - 1
COMPARE x	if M[x] > R, then GT = 1
	if M[x] = R, then EQ = 1
	if M[x] < R, then LT = 1

Instruction	Function
JUMP x	PC = x
JUMPGT x	if GT = 1, then PC = x
JUMPEQ x	if EQ = 1, then PC = x
JUMPLT x	if LT = 1, then PC = x
JUMPNEQ x	if EQ = 0, then PC = x
IN x	read input device
	and store result in M[x]
OUT x	write M[x] to output device
HALT x	processor stops

13. What is an “assembler”? Why do we need it?

14. How can a language like JavaScript be executed? Remember that computers only see bits.

15. What software let’s your computer access files? And manage memory?

16. Study the fundamental concepts of each application we covered in class. In Computer Graphics, why is it important to simulate the physical behavior of light? In Artificial Intelligence, how do we search for the next move in game. In Robotics, what is a “degree of freedom”. In Web Search, what are the three main components we need for a search engine like Google.

17. What do we mean when we say that something is decideable? Give examples? What is the Halting problem? Prove the Halting problem using the diagram example used in class or Javascript. What is the Church-Turing thesis and why it is important?

18. What is a Turing machine? Why is it useful?