CS202 Final
Due: 2013-12-13, 05:00p

This exam is open book, open notes, open course materials (anything posted on the homepage or on Piazza). It can also be considered to be open professor, in that you can ask me anything you want (you aren't guaranteed an answer, however...).

You are welcome to hand write or type your solutions. Please include the honor code on your submission (I will accept "digital signatures" -- just type your name and student ID number).

1. (5 points) We would like to implement procedures on the ∑niac, and as we saw, the stack is an important piece of this puzzle. To implement a stack, we need push and pop functions. Describe why this is impossible with the current architecture of the ∑niac (and it isn't because we don't have a stack pointer register - we could always dedicate addresses in memory to specific functions).

2. (5 points) On the first day of your new job at Acme Programming, you are handed some old code that is running slow. After doing some profiling, you identify a function that is consuming 30% of the overall runtime of the application. You bring all of the optimizing tools that you have picked up in this class to bear and rewrite the function so that is three times faster.
   a. How much faster is the program now?

   b. You start looking for other ways to improve the application and realize that the function isn't actually needed at all. After factoring it out, how fast is the application now?

3. (5 points) It takes .4μs to access main memory. You have cache memory that only requires .02μs available. How much faster can you access memory if you can guarantee a hit rate of 95% on the cache?

4. (5 points) You have been tinkering with a new machine. By adding a cache, you have managed to make loads 5 times faster, but stores are now 30% slower. If 20% of a program is load operations and 10% are stores, how much faster (or slower) is the new machine?

5. (8 points) Suppose we have a system with the following properties:
   - the memory is byte addressable
   - addresses are 13 bits wide
   - the cache is four-way set associative, with 4-byte blocks and eight sets
   a. What is the size of the cache in bytes?

   b. How would the bits of the address (A_{12} - A_0) be distributed between the **cache tag**, **cache set index**, and **block offset**?

6. (10 points) In class, I showed you that every Boolean function can be completely described by a truth table. I also demonstrated that given a truth table, I can write a sum of products expression for the function. Thus, any Boolean function can be implemented with only AND, OR and NOT gates. In truth, it is possible to implement all Boolean functions using only NOR gates. Prove it.

7. (12 points) Consider the following C function:

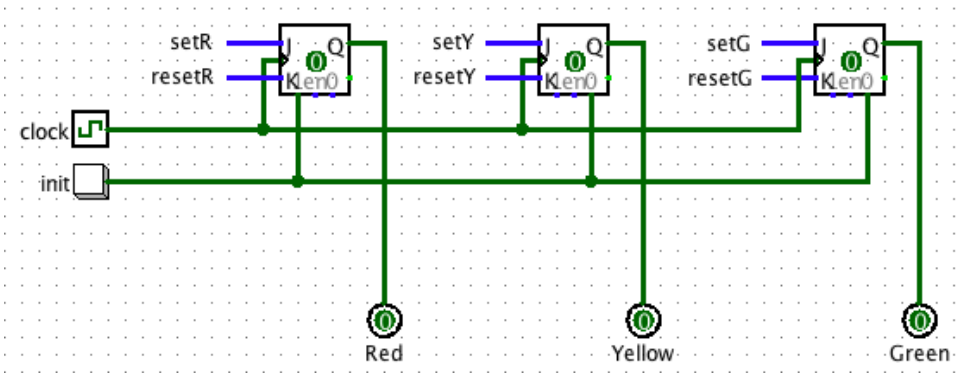
```
void test(){
  float f = -1.0;
  int a[1];
  a[0] = -1;
  fprintf(stdout,"a[0] = %x\n", a[0]);
  fprintf(stdout,"a[1] = %x\n", a[1]);
  fprintf(stdout,"a[2] = %x\n", a[2]);
  fprintf(stdout,"a[3] = %x\n", a[3]);
}
```

When called from main() under Linux (compiled with -m32), it prints:

```
a[0] = ffffffff
a[1] = bf800000
a[2] = 4fa173c4
a[3] = 2f
```

Give a detailed explanation for each of these four numbers. Be as specific as possible (Is it a value? If so, of which variable? What number does it represent? Or is it an address? If so, of what? What is its function? In what part of memory?) Hint: A sketch of the stack frame for test() might help.

8. (12 points) In Russia, Germany, and a collection of other European countries, traffics look like the ones we have here in the US (Red, Yellow, and Green lights), but the sequence is a little different. They go Red, Red +Yellow, Green, Yellow, Red,... (In theory, this allows you to stop the engine when the light turns red, and gives you enough time to restart before it is green again. My brief trip to Russia, however, it seemed that it was used more like a drag race light tree, with drivers revving up to be first off the line when the light goes green...). Here is the start of a synchronous circuit using JK flipflops to implement this pattern:



On each iteration of the clock, the circuit should advance to the next state [100, 110, 001, 010, 100…]. Derive equations for the 6 control signals setR, resetR, setY, resetY, setG, and resetG, using the outputs R,Y, and G. For full credit, your equations must be as simple as possible. (hint, remember that J-K flipflops can toggle).

9. (30 points) An eager CS 202 student has added a direct mapped cache to the ∑niac circuit (which, as you know, has 5-bit addresses). The cache holds 8 bytes and is organized in 2 lines of 4 byes each. (We are going to ignore for the moment the question about the utility of a cache for the ∑niac...)
   a. (3 points) Given an address of the form $A_4A_3A_2A_1A_0$, how are the bits allocated between the **tag**, the **set index**, and the **block offset**?

   b. (2 points) How much memory is required to implement the entire cache?

   For the remaining questions, assume the following program is stored in memory:

```
00: 08 29 4a ff 00 00 00 00
08: 0f 10 02 03 04 05 06 07
```

   c. (4 points) Disassemble the first four instructions (000-load, 001-sub, 010-add, 011-store, 100-b, 101-bneg, 111-halt).

Opcode	Binary opcode	Meaning
08		
29		
4a		
ff		

   d. (4 points) Assume the program is run. What computations are performed and what value will be contained in the accumulator once the program has terminated?

   e. (8 points) Assume the program has been run with an initially cold cache. Create four tables like the one below, and fill them in to show a snapshot of the cache ***after* each of the first four memory accesses** (remember that many instructions involve **two** memory accesses!). Use **binary** numbers for the addresses and tag fields, "Y"/"N" for the valid field, and hexadecimal numbers for the byte fields. If a line is invalid, you can leave the tag and byte fields blank.

Memory access 1: address_____ cache hit_____

	valid	tag	byte0	byte1	byte2	byte3
line 0						
line 1						

   f. (4 points) Extrapolating your results, what is the total number of memory accesses the entire program makes? How many of them are cache hits and how many of them are misses?

   g. (5 points) This illustrates a particular problem with direct mapped caches. What is it called? How could the program be changed to avoid it? How could the cache organization be changed?

10.(20 points) The following is a function that returns either 1 or 0 based on the provided input.

```
_test:
          pushl  %ebp
          movl   %esp, %ebp
          subl   $24, %esp
          movl   8(%ebp), %eax
          movl   %eax, -4(%ebp)
          movl   -4(%ebp), %eax
          movl   %eax, (%esp)
          call   _strlen
          movl   %eax, -16(%ebp)
          movl   -16(%ebp), %eax
          cmpl   $1, %eax
          jg     LBB1_2
          movl   $1, -12(%ebp)
          jmp    LBB1_5
LBB1_2:
          movl   -4(%ebp), %eax
          movb   (%eax), %al
          movl   -16(%ebp), %ecx
          subl   $1, %ecx
          movl   -4(%ebp), %edx
          movb   (%edx,%ecx), %cl
          cmpb   %cl, %al
          je     LBB1_4
          movl   $0, -12(%ebp)
          jmp    LBB1_5
LBB1_4:
          movl   -16(%ebp), %eax
          subl   $1, %eax
          movl   -4(%ebp), %ecx
          movb   $0, (%ecx,%eax)
          movl   -4(%ebp), %eax
          addl   $1, %eax
          movl   %eax, (%esp)
          call   _test
          movl   %eax, -12(%ebp)
LBB1_5:
          movl   -12(%ebp), %eax
          movl   %eax, -8(%ebp)
          movl   -8(%ebp), %eax
          addl   $24, %esp
          popl   %ebp
          ret
```

a. Provide an input for which this function will return 1. What does this function do?

b. Write the equivalent C function.

c. The performance of this function will be abysmal. There are two major issues with the code. Identify them and rewrite the function to perform better.