CS 351 Exam 1 Wed. 2/24/2016

Name: _____

Rules and Hints

- You may use one handwritten 8.5 × 11″ cheat sheet (front and back). This is the only additional resource you may consult during this exam. *No calculators.*
- You may write your answers in the form *[mathematical expression][units]*. There is no need to actually do the arithmetic.
- You may use extra scratch paper if you need more space, but make it clear where to find your answer to each question.

Grade

	Your Score	**Max Score**
Problem 1: Short answer		25
Problem 2: Processor performance and power		25
Problem 3: MIPS arrays		20
Problem 4: MIPS functions		30
Total		**100**

Problem 1: Short answer (25 points)

Part A (2 points)

+1 extra credit point if entire class gets it right.

How many bits are in a byte?

Part B (4 points)

Imagine that the MIPS designers are considering adding a new instruction to their ISA: INC $rs, which would increment the value in register $rs. To encode this instruction, you would need only the opcode and $rs fields, for a total of 11 bits. Rounding this up to the nearest byte, we could easily encode this instruction using 16 bits instead of 32. Is there a downside to saving space this way? Explain.

Part C (10 points)

Translate these two MIPS assembly instructions to 32-bit binary:

```
L: ADDI $v0, $t2, 5
BEQ $v0, $t1, L
```

Part D (9 points)

You and a friend have the same C++ program, which is CPU-bound (meaning the processor is the sole determinant of its performance). Your friend buys a processor with a faster clock speed than yours, but the program runs slower! Why might this be?

Your frustrated friend exchanges that processor for a processor with the same clock speed as yours, and twice as many cores. But the program still runs slower for your friend! How could this happen???

Your friend gives up on beating your performance and exchanges the processor for one that uses half as much power as yours. However, your friend's processor uses more energy to run the program. What's going on?

Problem 2: Processor performance and power (35 points)

You have a workload with 100 billion dynamic instructions. 60% are integer instructions, 15% are branches, and 25% are memory instructions. You want to run it on one core of a processor with the following characteristics:

- Frequency of 3 GHz
- Average CPI of 1 for integer instructions; 2 for branches; 3 for memory
- Peak power consumption of 75 W (25 W static, 50 W dynamic)

Part A: Execution time (9 points)

What is the execution time of your workload if it runs single-threaded on a single core of this processor?

Part B: Power and energy (8 points)

You have the option to run your program in a low-power mode, with the frequency and the operating voltage both halved. How much power will your processor use in this mode?

How much energy will it use in this mode?

Part C: Optimization (8 points)

Taking another look at your program, you find a loop that you can run in parallel across 4 cores. The loop takes up 20% of your current dynamic execution time. What is the speedup of the parallel version over the original version?

Problem 3: MIPS arrays (20 points)

Translate the following C code to MIPS. Assume that *a* is a `char *` in `$s0`, *b* is a `char *` in `$s1`, *c* is an `int` in `$s2`, and *N* is an `int` in `$s3`. At the end of your MIPS code, all of these registers should have their correct values according to the C semantics. Use temporary registers for all intermediate values, including the loop counter.

```
c = 0;
for (int i = 0; i < N; i++) {
    if (a[i] < b[i]) {
       c = 1;
   }
}
```

Problem 4: MIPS arrays (30 points)

Translate the following C functions to MIPS, using only the instructions on your reference sheet. You should translate each function independently and not attempt to combine them. Obey all MIPS conventions about functions, registers, and stack usage.

```
int g(int x) {
    return x + 2;
}

int h (int x) {
    return (x % 4) * 8;
}

int f(int* x, int* y) {
   if (g(x[0]) == h(y[0])) {
      return x[1];
   }
   return y[1];
}
```