CS 351 Exam 1, Fall 2010

Your name: ____

Rules

- You may use one handwritten 8.5 x 11" cheat sheet (front and back). This is the only resource you may consult during this exam.
- Include explanations and comments in your answers in order to maximize your partial credit. However, you will be penalized for giving extraneous incorrect information.
- You may use the backs of these pages if you need more space, but make it clear where to find your answer to each question.
- Unless otherwise specified, you do not need to work out the arithmetic on math problems. Just do enough algebra to set up an answer of the form: Answer = [arithmetic expression] [units]

Grade (instructor use only)

	Your Score	**Max Score**
Problem 1		15
Problem 2		20
Problem 3		15
Problem 4		10
Problem 5		10
Problem 6		15
Problem 7		15
Total		100

Problem 1: 15 points.

Answer the following questions. Be sure to explain your answers in order to receive partial credit.

---

a) How many bits are in a byte? How many bytes are in a megabyte? (The latter question has two different answers – what are they?)

b) Explain how function return addresses are treated differently in x86 and in MIPS.

c) Critique the following arguments. The more clear and specific your critique, the more points you will receive:

Argument 1: “My CPU has 4 cores, and yours only has 2! My computer is going to be so much faster than yours! Ha ha!”

Argument 2: “My disk’s seek time is way lower than yours! I’m going to get data retrieved from disk so much faster!”

Argument 3: “My disk uses only 2 W, and yours uses 10 W! My computer is going to use way less energy for disk-intensive workloads, sucka!”

Problem 2: 25 points.

You are considering two different processors to run various CPU-bound workloads. They both implement the same ISA.

1) A single-core processor with a CPI of 1.5 and a clock frequency of 2.5 GHz
2) A dual-core processor. Each core has a CPI of 2 and a clock frequency of 1.5 GHz

---

a) One of your workloads computes winning lottery numbers. Each time you run the program, it generates a single winning number. The winning numbers are not interdependent. Computing a single winning number cannot be parallelized.

   You need to produce a single winning number as quickly as possible. Which processor should you choose? (You should set up the arithmetic, but you don't need to fully carry it out – you should be able to see which is better without doing the full computation.)

   What is the speedup of the faster processor over the slower one for a single winning number? (Just set up the arithmetic expression – no need to do the arithmetic)

b) You are thinking of setting up a business and charging $100,000 per winning number. In order to maximize your money per unit time, which processor should you choose? (see above note about arithmetic)

c) With your proceeds from part (b), you convince the manufacturer of processor #1 to add special-purpose hardware that makes the main loop of your program run in half as much time as it used to! If this loop took 70% of your program's execution time on the old version of the processor, what is the speedup of the new version of the processor over the old version? (see above note about arithmetic)

Problem 3: 15 points.

You are considering two different quad-core processors to run various CPU-bound workloads. They both implement the same ISA.

1) A low-power processor running at 1 V and 1.8 GHz
2) A power-hungry processor running at 1.3 V and 3 GHz

Assume that they have the same static leakage power (*S*) and capacitive load (*C*).

---

a) How much power does each processor consume? (Just set up the arithmetic; no need to work it out.)

b) If the first processor runs a CPU-bound task in 60 seconds, how much energy does it consume?

c) How much energy will the second processor consume to complete the same task?

Problem 4: 10 points.

Translate the following C code to MIPS. Obey all MIPS conventions about register and stack usage. Assume that the C variable *a* is in $s0, *b* is in $s1, and *c* is in $s2.

```
while (a <= 50) {
     b += c;
     a++;
}
```

Problem 5: 10 points.

Translate the following C code to MIPS. Obey all MIPS conventions about register and stack usage.

```
int f(int x) { return x % 4; }
```

Problem 6: 15 points.

Translate the following C code to MIPS. Obey all MIPS conventions about register and stack usage. Assume that the array *a* is located at the top of the stack (so $sp points to the first element). Please put the loop counter in $s0.

```
for (int i=0; i < 50; i++) {
     a[i]++;
}
```

Problem 7: 15 points.

Translate the following C code to MIPS. Obey all MIPS conventions about register and stack usage.   Assume that the function *g* is defined and is being called correctly.

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

Extra paper