CS 351 Exam 1 Makeup, 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) Explain the differences in how the x86 ISA and the MIPS ISA handle memory accesses.

b) What MIPS instruction is the equivalent of the x86 CALL instruction? How is it different from the x86 CALL instruction?

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

Argument 1: Substituting a lower-power CPU will make my computer use less energy than before.

Argument 2: Substituting a CPU with a higher clock speed will make my program run faster than before.

Problem 2: 25 points.

You own a prototype of a *heterogeneous multicore* processor. It has one core that runs at 2.5 GHz and 3 cores that run at 1.5 GHz.

You can run code serially on the 2.5 GHz processor OR on up to 3 of the 1.5 GHz cores in parallel.

Answer the following questions:

---

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 core(s) should you run your workload on?

   What is the speedup of your choice over the alternative? (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. How should you utilize the cores to pump out as many numbers as possible per unit time?

c) With your proceeds from part (b), you convince the manufacturer of your processor to add special-purpose hardware to the serial core 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 core over the old version? (see above note about arithmetic) Would that change your answer to part (b)?

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 *b* is in \$s0, *c* is in \$s1, and *d* is in \$s2.

```
for (int i=0; i < N; i++) {
     b += c;
     c = d | i;
}
```

Problem 5: 10 points.

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

```
int f(int x) {
     int y = x/8;
     return y % 8;
}
```

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), and that a pointer to *b* is in $s0. Please put the loop counter in $s1.

```
for (int i=0; i < 50; i++) {
     a[i] = b[i-1];
}
```

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, int y) {
     int a[3];
     a[0] = rand() - x;
     a[1] = rand() - y;
     a[2] = rand();
     return a[0] & a[1] & a[2];
}
```

Extra paper