CS 351 Exam 2, Fall 2010

Your name: _____

Rules

- You may use one handwritten 8.5 x 11" cheat sheet (front and back). This sheet and the attached diagram are the only resources 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		20
Problem 2		15
Problem 3		20
Problem 4		10
Problem 5		20
Problem 6		15
Total		100

Problem 1: 20 points.

You are tracing the following instruction on the *single-cycle* datapath (attached):

Assembly code: lw $t0, 4($t1)
Instruction format: opcode / rs / rt / imm
Machine code: 100011 01001 01000 0000 0000 0000 0100

Provide the exact numerical answer to each question if possible. If not, describe the value without stating an exact number.

---

a) What is the input to the control unit for this instruction?

b) What values does this instruction send to the *Read Register 1* and *Read Register 2* ports?

c) Describe what values the register file puts on the *Read data 1* and *Read data 2* ports.

d) What operation does the ALU do this cycle? What are its operands?

e) What gets put on the register file's *Write register* port? What data gets put on the register file's *Write data* port?

f) What value is stored in the PC at the beginning of the next cycle?

Problem 2: 15 points.

You are modifying the datapath to support the SB instruction, which stores a byte into memory instead of a 32-bit word like SW. Answer the following questions:

---

a) What should the values of the following control signals be?

RegWrite:

MemRead:

MemWrite:

ALUSrc:

RegDst:

Branch:

MemtoReg:

b) What hardware, including control signals, would you have to add or modify to implement this instruction? Explain.

Problem 3: 20 points.

Refer to the *pipelined* datapath (attached) for this question. Assume the following latencies for the different functional units:

Instruction or data memory: 500 ps
Register file: 250 ps
ALU: 300 ps
Simple adder: 150 ps
Mux: 50 ps
ALU control: 100 ps
Everything else: zero

---

a) What is the latency of the IF stage? The IF stage starts when we read the PC and finishes when the correct values are in the IF/ID register.

b) What is the latency of the EX stage? The EX stage starts when we read the ID/EX register and finishes when we read the EX/MEM register.

c) Based on your answers to (a) – (b), what can you conclude about this processor's clock cycle time?

d) How would your answer to (c) change if simple adders were 50 ps faster?

Problem 4: 10 points.

Answer the following questions about the single-cycle and pipelined MIPS implementations.

Assume that your program consists of 1000 MIPS instructions and that there are no data or control dependences between consecutive instructions.

---

a) How many cycles will your program take to run on the single-cycle implementation?

b) How many cycles will your program take to run on the pipelined implementation?

c) Refute the following obnoxious argument. Assume that *A* refers to your numerical answer from part (a), and *B* refers to your numerical answer from part (b).

"So if *A* is less than *B*, that means the single-cycle implementation is faster, right? But if *B* is less than *A,* that means the pipelined implementation is faster, right? Hey, look at me, I'm a computer architect!"

Problem 5: 20 points.

For each of the following sequences of instructions, state whether a data dependence exists between the two instructions.

If there is a data dependence, state

- which pipeline stage of the first instruction produces the needed data
- which pipeline stage of the second instruction consumes that data
- whether it is necessary to stall the pipeline because of this hazard.

Assume that the pipeline fully supports forwarding.

---

a.

```
lw $t0, 4($t1)
sw $t0, 8($t1)
```

b.

```
add $t0, $t1, $t2
add $t3, $t0, $t4
```

c.

```
lw $t0, 0($t1)
add $t3, $t0, $t4
```

d.

```
sw $t0, 0($t1)
add $t3, $t0, $t4
```

Problem 6: 15 points.

Assume a branch with the following repeated pattern:

T, NT, T, NT, T, NT, T, NT, …

---

a) Write code (MIPS or C/C++) that exhibits this behavior. Explain. (You will almost certainly need nested loops.)

b) What is the accuracy of "predict taken" for this branch? What about "predict not taken"? Is there any reason to prefer one policy to the other?

c) What is the accuracy of the 2-bit saturating predictor for this branch?

Initial state = 00:

Initial state = 01: