CS 351 Exam 2, Fall 2011

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		30
Problem 2		20
Problem 3		20
Problem 4		15
Problem 5		15
Total		100

Problem 1: Datapath mechanics (30 points)

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

Assembly code: SUB $t0, $t1, $zero
Instruction format: opcode / rs / rt / rd / shamt / funct
Machine code: 000000 01001 00000 01000 00000 100010

Assume that the value in $t1 is 2 before the instruction begins.

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

---

a) [2 points] What is the input to the control unit for this instruction?

b) [4 points] What values does this instruction send to the *Read Register 1* and *Read Register 2* ports?

c) [4 points] What values end up on the *Read data 1* and *Read data 2* ports?

d) [6 points] What operation does the ALU do this cycle? What are its exact operands?

e) [4 points] What gets put on the register file's *Write register* port? What data gets put on the register file's *Write data* port?

f) [4 points] What address goes to the memory's address port? What data goes to the memory's data port?

g) [2 points] How is the next PC computed?

h) [4 points] Which of your answers to parts (a)-(g) are eventually discarded? That is, which of these answers do NOT have an impact on the state of the processor?

Problem 2: Extending the datapath (20 points)

You are modifying the attached datapath to support a new instruction called INC:

INC $t0 # Adds 1 to the value in $t0

INC is an R-format instruction, encoded as follows:

- Opcode = 000000 (R-format)
- $rs = XXXXX (don't care)
- $rt = register to increment
- $rd = XXXXX
- shamt = XXXXX
- Funct = 0x3f

A. [10 points] State the value of each existing control signal for the INC instruction, and briefly explain.

   If the value of a control signal does not matter, use an X (don't care). You will lose points if you use a 1 or a 0 for a signal whose value does not matter.

MemWrite:

RegWrite:

Branch:

Jump:

RegDst:

MemtoReg:

ALUSrc:

B. [10 points] Describe *in detail* any hardware you would need to add to the existing datapath to implement this instruction.

   If you are adding any muxes, please label their inputs and outputs on the herd of muxen on the next page. You should label each input and output line so that it is 100% clear where it links up with the attached datapath diagram.

   You should label both the data and select inputs of the muxen. If the select input is a new control signal, explain how it works.

Problem 3: Pipelining and performance (20 points)

Assume a 4-stage pipeline, where the latency of each stage is as follows:

Stage 1: 400 ps
Stage 2: 500 ps
Stage 3: 300 ps
Stage 4: 350 ps

A. [5 points] What is the minimum possible cycle time for this processor? How long will a single instruction take to complete?

B. [5 points] How long does it take 10 independent instructions to complete?

C. [5 points] Consider an alternative implementation that executes each instruction in a single cycle by going through all 4 stages sequentially. Set up a mathematical expression for *N,* the minimum length of a sequence of instructions that is faster in the pipelined implementation than the single-cycle implementation. For example, N=3 if 1-2 instructions execute faster in the single-cycle implementation, but 3+ instructions execute faster in the pipelined implementation.

D. [5 points] How would your answers to A and B change if we added a pipeline stage by splitting Stage 2 into 2 separate stages: a 300 ps stage and a 200 ps stage?

Problem 4: Data hazards (15 points)

Assume the following sequence of MIPS instructions:

```
add $s0, $s0, 1
add $t1, $t1, 4
lw $t0, 4($t1)
sw $t0, 8($t1)
slti $t2, $s0, 25
```

A. [5 points] List ALL of the read-after-write data dependencies, even those that are unlikely to stall the pipeline. Specify the two instructions and the register in question.

B. [8 points] Draw a pipeline diagram for this set of instructions. Assume that all possible forwarding paths have been implemented, and indicate forwarding on your diagram.

C. [2 points] What is the CPI of this sequence of instructions? If this exact set of instructions were repeated infinitely, what would the CPI be?

Problem 5: Control hazards and branch prediction (15 points)

Consider the following branch pattern, which repeats indefinitely:

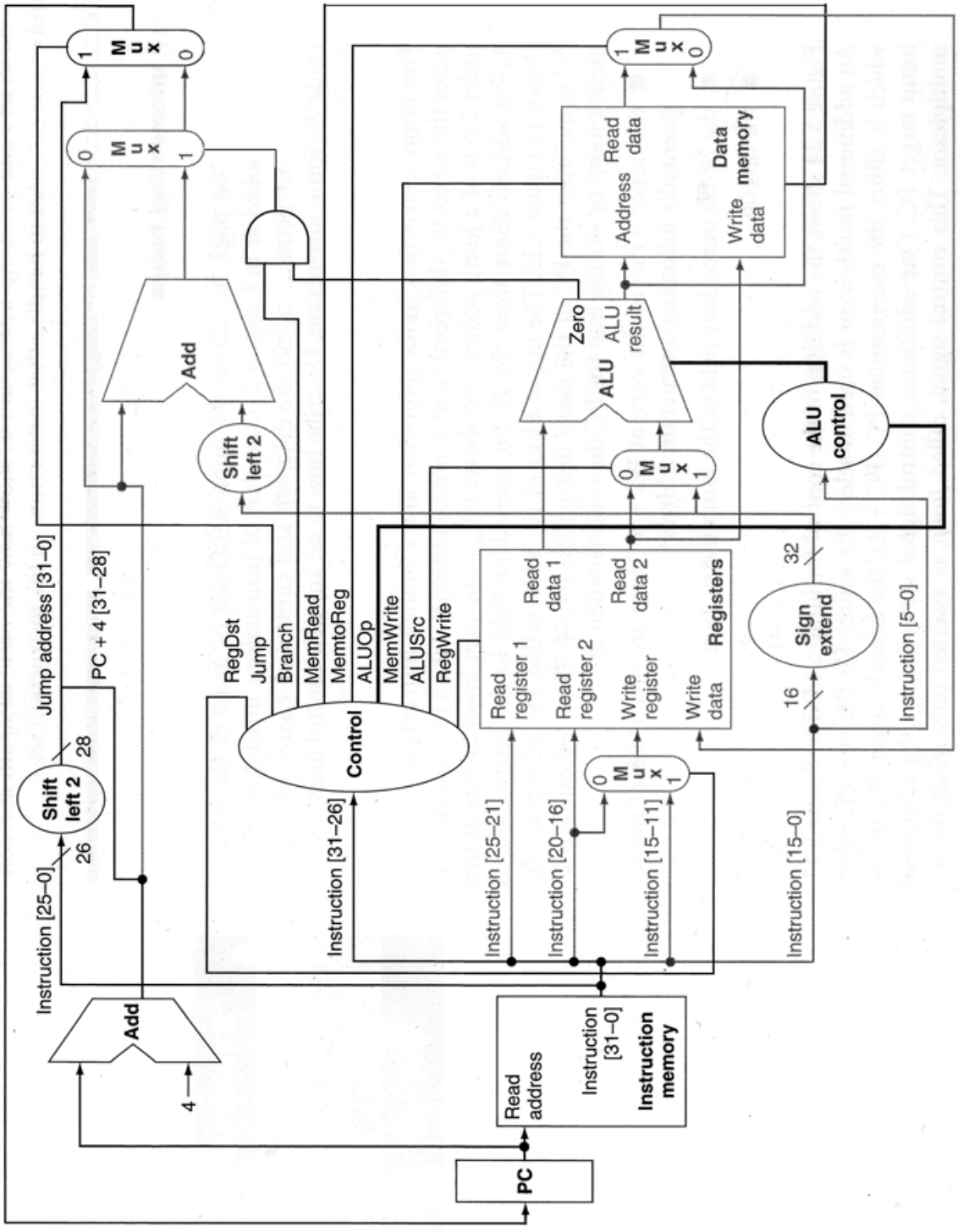
T, T, T, T, NT

A. [4 points] What is the accuracy of "predict taken" for this branch? What about "predict not taken"?

B. [3 points] What is the accuracy of a 1-bit predictor initialized to NT?

C. [4 points] What is the steady-state accuracy of a 2-bit predictior initialized to strong NT?

D. [4 points] The Intel Pentium 1 used a variant of the 2-bit predictor: it was initialized to strong NT but would transition to strong T if the first occurrence of the branch was taken. How would this design differ from the behavior in Part C?


M u x
0
1
Add
4
Instruction [25–0]
Instruction [31–0]
Instruction memory
Read address
PC
Jump address [31–0]
PC + 4 [31–28]
Shift left 2
26
28
Shift left 2
Add
RegDst
Jump
Branch
MemRead
MemtoReg
ALUOp
MemWrite
ALUSrc
RegWrite
Control
Instruction [31–26]
Instruction [25–21]
Instruction [20–16]
Instruction [15–11]
Instruction [15–0]
Read register 1
Read register 2
Write register
Write data
Read data 1
Read data 2
Registers
ALU
Zero
ALU result
Address
Read data
Write data
Data memory
ALU control
Sign extend
16
32
Instruction [5–0]