CS 351 Exam 2 Mon. 4/6/2015

Name: ____

Rules and Hints

- **The MIPS cheat sheet and datapath diagram are attached at the end of this exam for your reference.**
- 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.
- Include step-by-step explanations and comments in your answers, and show as much of your work as possible, in order to maximize your partial credit.
- 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: Tracing the MIPS datapath		30
Problem 2: Pipeline performance and implementation		25
Problem 3: Average memory access time		20
Problem 4: Cache tracing		25
Total		**100**

Problem 1: Tracing the MIPS datapath (30 points)

Answer the following questions about the single-cycle datapath provided at the end of this exam. This is the datapath that executes a single instruction, from start to finish, on every clock cycle. Be sure to explain your answers if you want to receive partial credit.

Consider the execution of this instruction: `BEQ $t0, $zero, $L`

Assume the following:

- The bits of this instruction are stored in addresses 1000–1003 in memory.
- The initial value in \$t0 is 0.
- The immediate operand (L) is 0xfffe.

Provide the exact numeric answer (in decimal, hex, or binary) to each question if possible. If not, describe the value without stating an exact number.

Part A: Instruction memory (4 points)

What value goes into the instruction memory's *Read Address* port? How many bits is that value?

What is the value of the instruction memory's *Instruction[31:0]* output?

Part B: Register file (7 points)

What values go into the register file's *Read Register 1* and *Read Register 2* ports? How many bits are those values?

What values go into the register file's *Write Register* and *Write Data* ports? How many bits are those values?

What is the value of the *RegWrite* control signal?

Part C: ALU (7 points)

What are the values of the ALU's two data inputs? How many bits are those values?

What two values go into the ALU control unit? How many bits are those values?

What are the values of the ALU's *ALU result* and *Zero* outputs? How many bits are those values?

Part D: Branch target adder (7 points)

What two values are input to the top right adder? How many bits are those values?

What is the output of the top right adder?

What is the output of the top right mux? Conceptually, what is this mux choosing between?

Part E: Modifying the datapath (5 points)

How would you change this datapath to support BNE instead of BEQ?

Problem 2: Pipelining (25 points)

You have a 5-stage MIPS pipeline in which each stage takes the following number of cycles:

- IF: 200 ps
- ID: 300 ps
- EX: 200 ps
- MEM: 250 ps
- WB: 150 ps

Part A: Cycle time (8 points)

What is the minimum cycle time of a **single-cycle** version of this processor?

What is the minimum cycle time of a **pipelined** version of this processor?

Part B: Performance (8 points)

How many **cycles** will each version of this processor take to execute a sequence of 1,000,000 instructions with no stalls due to data or control hazards?

Which version will be faster for this sequence of instructions? Explain.

Part C: Branch prediction (9 points)

Consider a branch that alternates forever between T and NT. What is the *long-term accuracy* of each of the following predictors for this branch?

- Always predict not taken
- A 1-bit predictor
- A 2-bit predictor

Explain your answers.

Problem 3: Average memory access time (20 points)

You have a system with the following memory hierarchy:

- Separate L1 instruction and data caches, which each have access times of 1 cycle and hit rates of 95%
- A unified L2 cache with an access time of 5 cycles and a hit rate of 80%
- A unified L3 cache with an access time of 10 cycles and a hit rate of 60%
- Main memory with an access time of 100 cycles

Part A: AMAT (10 points)

What is the average memory access time, in cycles, on this system?

Part B: CPI (10 points)

If 30% of instructions are loads or stores, how many memory accesses does the average instruction make?

If the base CPI of this system is 1 cycle, and accesses to L1 are included in this base CPI, what is the actual CPI including memory stalls?

Problem 4: Cache tracing (25 points)

For this problem, consider a cache that

- Has 16 blocks (entries)
- Is direct-mapped
- Has 1 byte per block
- Is write-back

Memory addresses on this system are 10 bits.

Part A: Tracing (15 points)

Give a sequence of at least 6 memory references that will have a hit rate of exactly 50%, assuming the cache starts out empty. Then, draw the final cache state (data and metadata).

Part B: Other mapping schemes (10 points)

Give a sequence of at least 6 memory references that will have a hit rate of 0% for the cache from Part A, but at least 50% for a fully associative cache that has 16 blocks and 1 byte per block. Explain.

Give a sequence of at least 6 memory references that will have a hit rate of 0% for the cache from Part A, but at least 50% for a direct-mapped cache that has 4 blocks and 4 bytes per block. Explain.

MIPS Arithmetic Instructions

Instruction	Operation	Fmt	Opcode	Funct
ADD $rd, $rs, $rt	reg[rd] = reg[rs] + reg[rt]	R	0	0x20
ADDI $rt, $rs, imm	reg[rt] = reg[rs] + SignExt(imm)	I	0x08	
ADDU $rd, $rs, $rt	reg[rd] = reg[rs] + reg[rt]	R	0	0x21
ADDIU $rt, $rs, imm	reg[rt] = reg[rs] + SignExt(imm)	I	0x09	
SUB $rd, $rs, $rt	reg[rd] = reg[rs] - reg[rt]	R	0	0x22
SUBU $rd, $rs, $rt	reg[rd] = reg[rs] - reg[rt]	R	0	0x23
LUI $rt, imm	reg[rt] = (imm << 16) | 0	I	0x0f	

The signed instructions (ADD, ADDI, SUB) can cause overflow exceptions.

MIPS Logical Instructions

Instruction	Operation	Fmt	Opcode	Funct
AND $rd, $rs, $rt	reg[rd] = reg[rs] & reg[rt]	R	0	0x24
ANDI $rt, $rs, imm	reg[rt] = reg[rs] & ZeroExt(imm)	I	0x0c	
NOR $rd, $rs, $rt	reg[rd] = ~(reg[rs] | reg[rt])	R	0	0x27
OR $rd, $rs, $rt	reg[rd] = reg[rs] | reg[rt]	R	0	0x25
ORI $rt, $rs, imm	reg[rt] = reg[rs] | ZeroExt(imm)	I	0x0d	
SLL $rd, $rt, shamt	reg[rd] = reg[rt] << shamt	R	0	0x00
SRL $rd, $rt, shamt	reg[rd] = reg[rt] >> shamt	R	0	0x02

The SLL and SRL instructions fill in the "shifted-out" bits with 0.

MIPS Branch and Jump Instructions

Instruction	Operation	Fmt	Opcode	Funct
BEQ $rs, $rt, label	if (reg[rs] == reg[rt]) PC=BrAddr	I	0x04	
BNE $rs, $rt, label	if (reg[rs] != reg[rt]) PC=BrAddr	I	0x05	
J label	PC = JumpAddr	J	0x02	
JAL label	$ra = PC+4; PC = JumpAddr	J	0x03	
JR $rs	PC = reg[$rs]	R	0	0x08

BrAddr = PC + 4 + SignExt (imm << 2)
JumpAddr = (PC+4)[31:28] | (26-bit imm << 2)

MIPS Memory Access Instructions

Instruction	Operation	Fmt	Opcode	Funct
LW $rt, imm($rs)	reg[rt] = Mem[rs + SignExt(imm)]	I	0x23	
SW $rt, imm($rs)	Mem[rs + SignExt(imm)] = reg[rt]	I	0x2b	
LH $rt, imm($rs)	Loads 16b from memory	I	0x21	
LHU $rt, imm($rs)	Loads 16b from memory	I	0x25	
SH $rt, imm($rs)	Stores 16b to memory	I	0x29	
LB $rt, imm($rs)	Loads 8b from memory	I	0x20	
LBU $rt, imm($rs)	Loads 8b from memory	I	0x24	
SB $rt, imm($rs)	Stores 8b to memory	I	0x28	

LH and LB sign-extend the values read from memory in order to fill the leftmost bits of the 32-bit register. LHU/LBU zero-extend.

MIPS Comparison Instructions

Instruction	Operation	Fmt	Opcode	Funct
SLT $rd, $rs, $rt	reg[rd] = ($rs < $rt)	R	0	0x2a
SLTI $rt, $rs, imm	reg[rt] = ($rs < SignExt(imm))	I	0x0a	
SLTU $rd, $rs, $rt	reg[rd] = ($rs < $rt)	R	0	0x2b
SLTIU $rt, $rs, imm	reg[rt] = ($rs < SignExt(imm))	I	0x0b	

Instruction Formats

R-format

6b: opcode	5b: rs	5b: rt	5b: rd	5b: shamt	6b: funct

I-format

6b: opcode	5b: rs	5b: rt	16b: immediate operand

J-format

6b: opcode	26b: JumpAddr[28:2]

Register List

Name	Number	Use	Preserved across function call?
$zero	0	Constant value 0	Y
$at	1	Assembler temporary	N
$v0-$v1	2-3	Function return values	N
$a0-$a3	4-7	Function arguments	N
$t0-$t7	8-15	Temporary values	N
$s0-$s7	16-23	Saved values	Y
$t8-$t9	24-25	More temporary values	N
$k0-$k1	26-27	Reserved for OS kernel	N
$gp	28	Global (heap) pointer	Y
$sp	29	Stack pointer	Y
$fp	30	Frame pointer	Y
$ra	31	Return address	Y

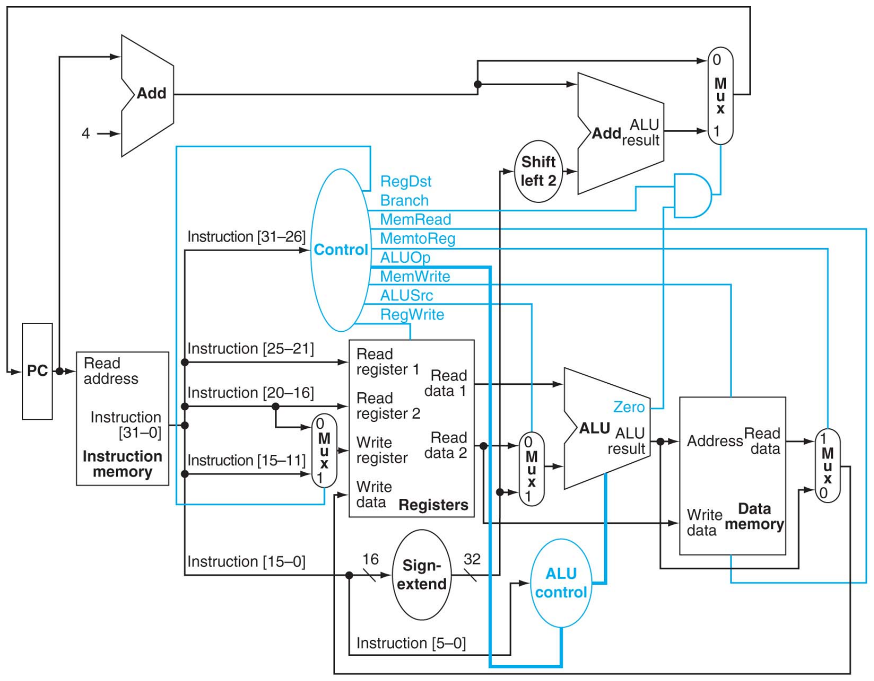

FIGURE 4.17 The simple datapath with the control unit. The input to the control unit is the 6-bit opcode field from the instruction. The outputs of the control unit consist of three 1-bit signals that are used to control multiplexors (RegDst, ALUSrc, and MemtoReg), three signals for con trolling reads and writes in the register file and data memory (RegWrite, MemRead, and MemWrite), a 1-bit signal used in determining whether to possibly branch (Branch), and a 2-bit control signal for the ALU (ALUOp). An AND gate is used to combine the branch control signal and the Zero output from the ALU; the AND gate output controls the selection of the next PC. Notice that PCSrc is now a derived signal, rather than one coming directly from the control unit. Thus, we drop the signal name in subsequent figures. Copyright © 2009 Elsevier, Inc. All rights reserved.