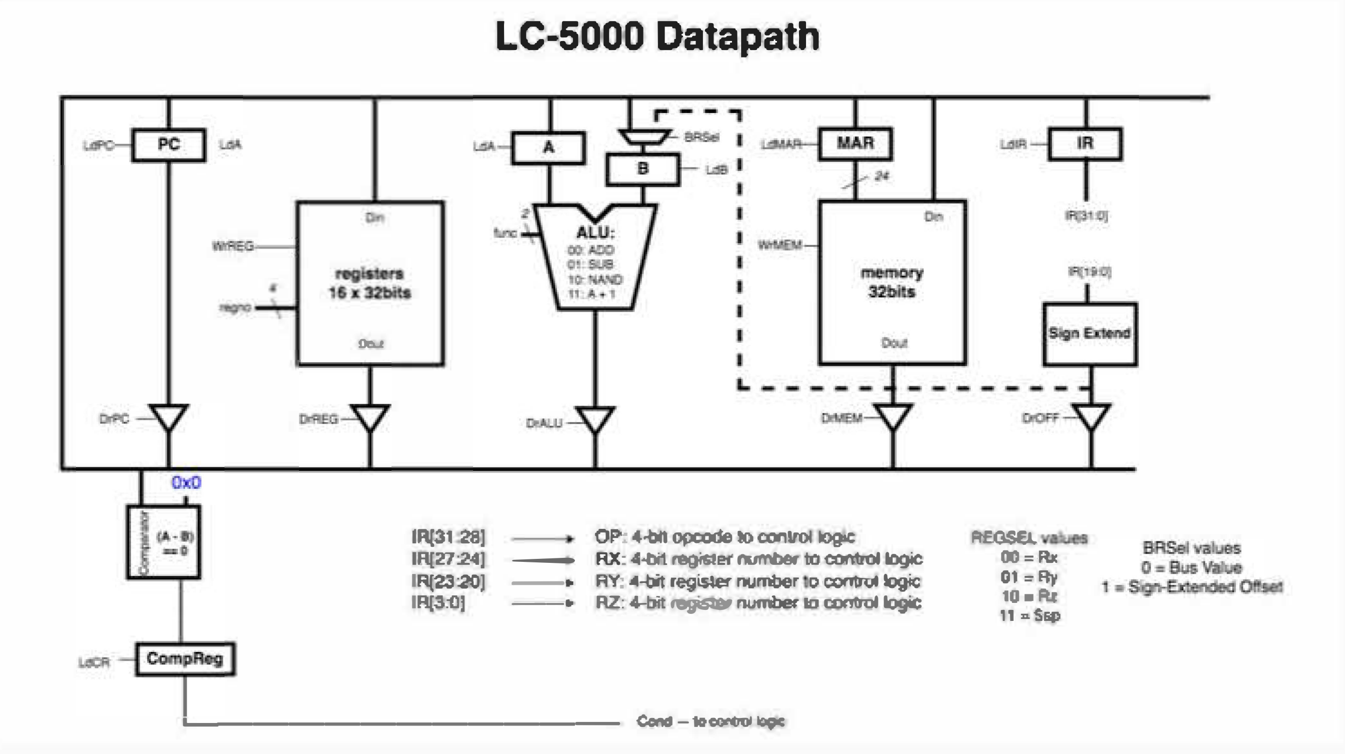
Q1 Datapath Tracing
28 Points



The above is the datapath of the LC-4000, a modified version of LC-2200. Notice the extra MUX north of the "B" register and a wire connecting the Sign Extend output to the new B mux. Also, note that the new mux uses control signals BRSel.

Q1.1 LW microcode
8 Points

Write out the microstates for an efficient `LW` instruction that makes use of the modifications on the LC-5000 datapath. This `LW` instruction accomplishes the same goals i.e., loads a 32-bit word from memory into DR using the Base Register and offset.

For each microstate, write the control signals used. Signals irrelevant to the state can be omitted and will be assumed to be zero. **You will lose points for an inefficient answer!** An example answer can be found below. *Note the use of RegSel instead of regno!*

Example: `ADD` instruction
ADD0: DrREG, LdA, RegSel=01
ADD1: DrREG, LdB, RegSel=10
ADD2: DrALU, WrReg, func=00, RegSel=00

Enter your microstates of the `LW` instruction for the LC-5000 below:

LW1: DrREG, LdA, LdB, BRSel=1, RegSel=01
LW2: DrALU, LdMAR, func=00
LW3: DrMEM, WrREG, RegSel=00

1.1 LW microcode 8 / 8 pts

✓	+ 3 pts	Asserts DrREG, LdA, LdB, RegSel=01, BrSel=1
✓	+ 2 pts	Asserts DrALU, LdMAR, func=00
✓	+ 2 pts	Asserts DrMEM, WrREG, RegSel=00
✓	+ 1 pt	Completed in three clock cycles
	+ 4 pts	Incorrect: Working LW except does not utilize BrSel
	+ 0 pts	Incorrect/blank/no answer

Q1.2 BEQ microcode
20 Points

Write out the microstates for an efficient `BEQ` instruction that makes use of the modifications on the LC-5000 datapath. For each microstate, write the control signals used. Signals irrelevant to the state can be omitted and will be assumed to be zero. **You will lose points for an inefficient answer!**

You should write out the full logic for `BEQ`; this means including the microstates for when a branch is taken. You should assume that asserting `ChkCmp` at the correct time will select the correct next state for the branch; and you should assume that the branch **is taken** in order to write out the full logic for BEQ.

Enter your microstates of the `BEQ` instruction for the LC-5000 below:

BEQ1: DrREG, LdA, RegSel=00
BEQ2: DrREG, LdB, BRSel=0, RegSel=01
BEQ3: DrALU, LdCR, func=01
BEQ4: ChkCmp
BEQ5: DrPC, LdA, LdB, BRSel=1
BEQ6: DrALU, LdPC, func=00

1.2 BEQ microcode **20** / 20 pts

✓	+ 4 pts	Asserted DrREG, LdA, RegSel = 00
✓	+ 4 pts	Asserted DrREG, LdB, RegSel = 01, BrSel = 0
✓	+ 4 pts	Asserted DrALU, LdCR, func = 01
✓	+ 0 pts	Asserted ChkCmp = 1 (Correct, but we are not requiring this triggered)
✓	+ 4 pts	Asserted DrPC, LdA, LdB, BrSel = 1
✓	+ 4 pts	Asserted DrALU, LdPC, func = 00
	+ 0 pts	Works and utilizes BrSel, but still uses more than 5 microstates (ignoring any ChkCmp only microstates)
	+ 10 pts	Works, but does not utilize BrSel at all
	− 2 pts	Second to last microstate broken up into 2 microstates and is thus inefficient
	+ 0 pts	Incorrect/Empty

Q2 Datapath Design
16 Points

In datapath design, two common approaches are to use a single-ported register file or a dual-ported register file.

Q2.1 LC 2200 Register File
4 Points

What type of register architecture does the LC-2200 have?

[✓ Single-ported]
Dual-ported

2.1 LC 2200 Register File **4** / 4 pts

✓	+ 4 pts	Correct
	+ 0 pts	Incorrect

Q2.2 Single-ported vs Dual-ported Register File
12 Points

Describe one benefit of having a single-ported register file design AND one drawback of having a single-ported register file design.

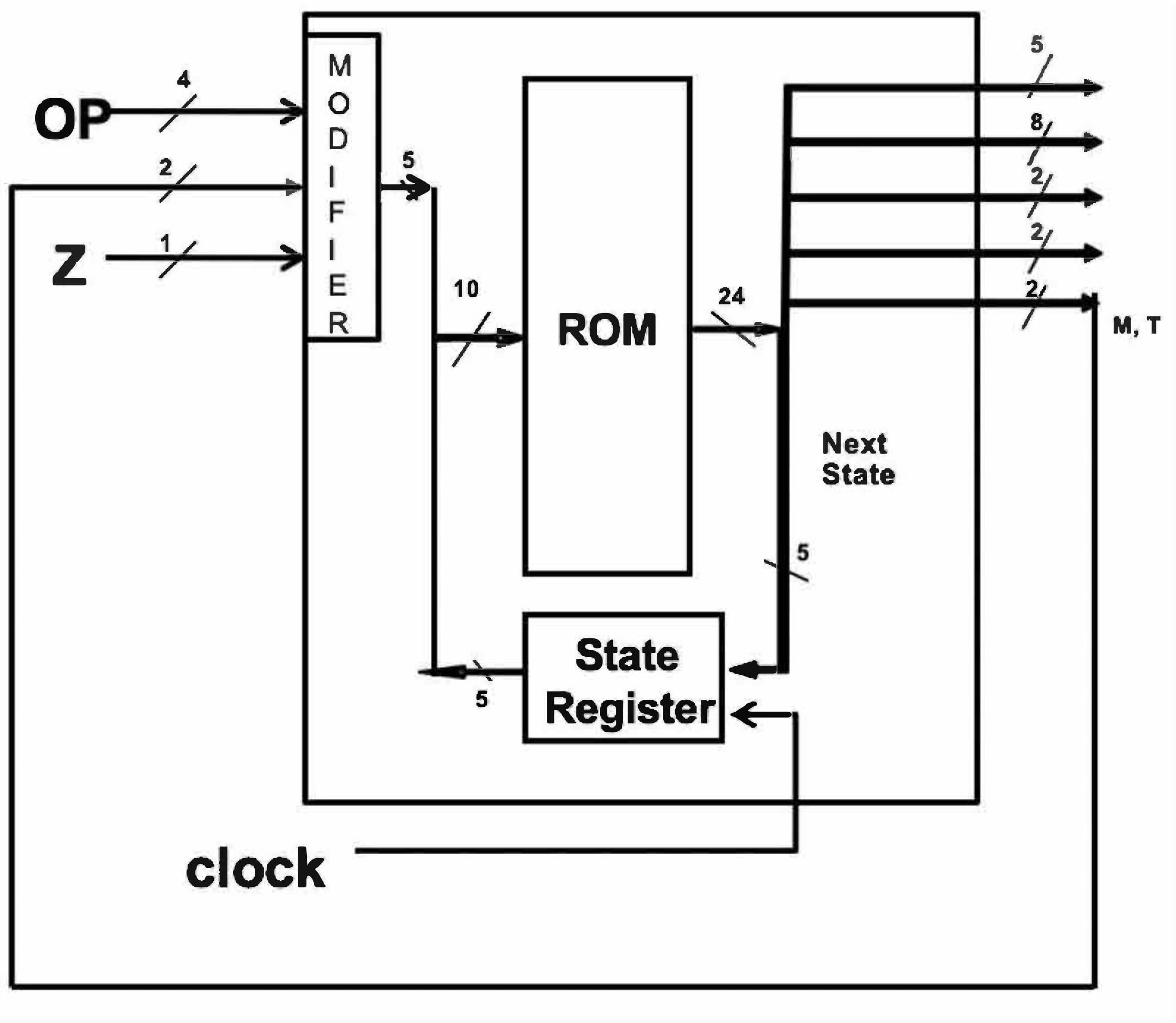
Advantage of having single-ported vs dual-ported REG FILE: Single-ported has a larger clock cycle width (dual-ported register files aren't very common in the real world because of their slow clock speed)

Disadvantage of having a single-ported vs dual-ported REG FILE: Can only read/write one register at a time whilst dual-ported can read or write two at nearly the same time.

2.2 Single-ported vs Dual-ported Register File **12** / 12 pts

✓	+ 12 pts	Correct
	+ 6 pts	Mentions a benefit of single-bus design (easier/simpler to implement, more flexibility to extend datapath because all components are connected to one another/can load multiple components at the same time, etc.)
	+ 6 pts	Mentions a drawback of single-bus design (more clock cycles because only one component can be writing at any given moment, etc.)
	+ 0 pts	Incorrect
	+ 0 pts	Incorrect/blank/no answer

Q3 Microcontroller Design
22 Points



Consider the Flat ROM from the LC-2200 microcontroller. The Z-bit is equivalent to CmpOut in the LC-902 (i.e. it tells the microcontroller whether or not to branch).

The bit layout of the input to the rom looks like this:

```
MSB | --- 4 bit OP --- | --- 1 bit Z --- | --- 5 bit State --- | LSB
```

Q3.1 Fetch microstates

4 Points

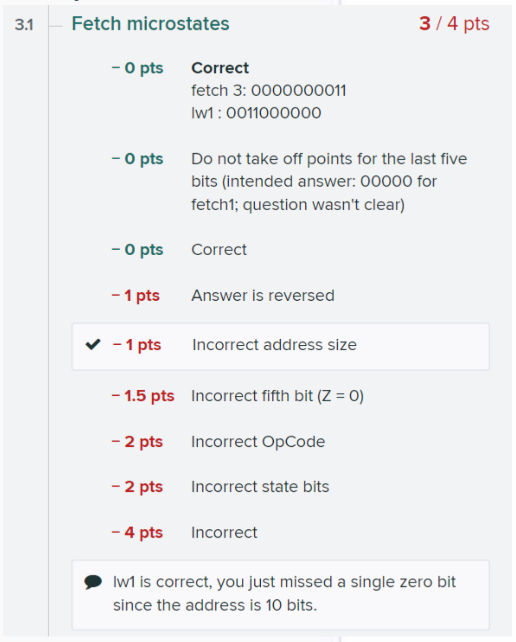
Suppose fetch contains 3 microstates (fetch1 starts at *00001*), and the base address for next state is *00000*. What is the address in ROM that stores the fetch3 and lw1 respectively?

fetch3

000000011

lw1

001100000


3.1 Fetch microstates **3** / 4 pts

	− 0 pts	**Correct** fetch 3: 0000000011 lw1 : 0011000000
	− 0 pts	Do not take off points for the last five bits (intended answer: 00000 for fetch1; question wasn't clear)
	− 0 pts	Correct
	− 1 pts	Answer is reversed
✓	− 1 pts	Incorrect address size
	− 1.5 pts	Incorrect fifth bit (Z = 0)
	− 2 pts	Incorrect OpCode
	− 2 pts	Incorrect state bits
	− 4 pts	Incorrect

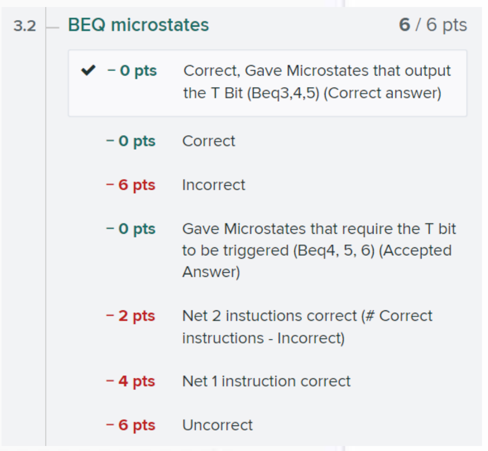
lw1 is correct, you just missed a single zero bit since the address is 10 bits.

Q3.2 BEQ microstates

6 Points

Assume BEQ has 6 microstates in total (3 microstates to compare register values and 3 microstates to jump to new address), denote them as *beq1, beq2, ..., beq6*. At which microstate(s) will we turn on the T bit?

BEQ3, BEQ4, BEQ5


3.2 BEQ microstates **6** / 6 pts

✓	− 0 pts	Correct, Gave Microstates that output the T Bit (Beq3,4,5) (Correct answer)
	− 0 pts	Correct
	− 6 pts	Incorrect
	− 0 pts	Gave Microstates that require the T bit to be triggered (Beq4, 5, 6) (Accepted Answer)
	− 2 pts	Net 2 instuctions correct (# Correct instructions - Incorrect)
	− 4 pts	Net 1 instruction correct
	− 6 pts	Uncorrect

Q3.3 M bit

8 Points

Describe the reason why we have a M bit

The M bit is for multiway branches (e.g. fetch3). It tells the ROM to get the next address using the OPCode rather than the next state encoded in the ROM.

3.3 M bit 8 / 8 pts

✓	+ 4 pts	Describes that the M bit enables multiway branching for the end of fetch
✓	+ 4 pts	Describes that the M bit allows the microcontroller to read the OpCode
	+ 2 pts	Confuses the selection functionality with that of the T bit, where we append the Z-bit instead of the opcode.
	+ 0 pts	Blank/no answer
	+ 0 pts	Incorrect

Q3.4 Minimum bitsize of Next State Register
4 Points

Assume the following:

- Fetch takes 3 microstates
- Any individual instruction, except for BEQ, takes at most 4 microstates
- The first half of branch takes 3 microstates
- The second half of branch takes 3 microstates (for a total of 6 branch microstates)

What is the smallest acceptable size of the Next State register in bits?

1

2

3

4

[✓ 5]

3.4 Minimum bitsize of Next State Register 0 / 4 pts

	+ 4 pts	Correct (Largest instruction is BEQ, 6 microstates in total. Requires 3 bits to represent all the microstates)
	+ 4 pts	Correct
✓	+ 0 pts	Incorrect

Q4
6 Points

For each of the six examples below:

- choose whether it is synchronous or asynchronous
- choose whether it is an exception, trap, or interrupt

Q4.1
1 Point

Keyboard inputs is a(n) _____ event.

Synchronous

[✓ Asynchronous]

This would be an example of a(n) _____.

Exception

Trap

[✓ Interrupt]

4.1 (no title) 1 / 1 pt

✓	+ 1 pt	Correct
	+ 0.5 pts	Only Async
	+ 0.5 pts	Only Interrupt
	+ 0 pts	Incorrect

Q4.2
1 Point

Indexing an array outside of its bounds is a(n) _____ event.

[✓ Synchronous]

Asynchronous

This would be an example of a(n) _____.

[✓ Exception]

Trap

Interrupt

4.2 (no title) 1 / 1 pt

✓	+ 1 pt	Correct
	+ 0.5 pts	Only Sync
	+ 0.5 pts	Only Exception
	+ 0 pts	Incorrect

Q4.3
1 Point

Making a System Call is a(n) ____ event.

- [✓ Synchronous]
- Asynchronous

This would be an example of a(n) _____.

- Exception
- [✓ Trap]
- Interrupt

4.3 — (no title) **1** / 1 pt

✓	**+ 1 pt**	Correct
	+ 0.5 pts	Only Sync
	+ 0.5 pts	Only Trap
	+ 0 pts	Incorrect

Q4.4
1 Point

Receiving a printer finished message is a(n) _____ event.

- [✓ Synchronous]
- Asynchronous

This would be an example of a(n) _____.

- Exception
- [✓ Trap]
- Interrupt

4.4 — (no title) **0** / 1 pt

	+ 1 pt	Correct
	+ 0 pts	Only Async
	+ 0 pts	Only Interrupt
✓	**+ 0 pts**	Incorrect

Q4.5
1 Point

Floating point underflow is a(n) ____ event.

- [✓ Synchronous]
- Asynchronous

This would be an example of a(n) _____.

- [✓ Exception]
- Trap
- Interrupt

4.5 — (no title) **1** / 1 pt

✓	**+ 1 pt**	Correct
	+ 0 pts	Only Sync
	+ 0 pts	Only Exception
	+ 0 pts	Incorrect

Q4.6
1 Point

Writing to a file is a(n) _____ event.

- [✓ Synchronous]
- Asynchronous

This would be an example of a(n) _____.

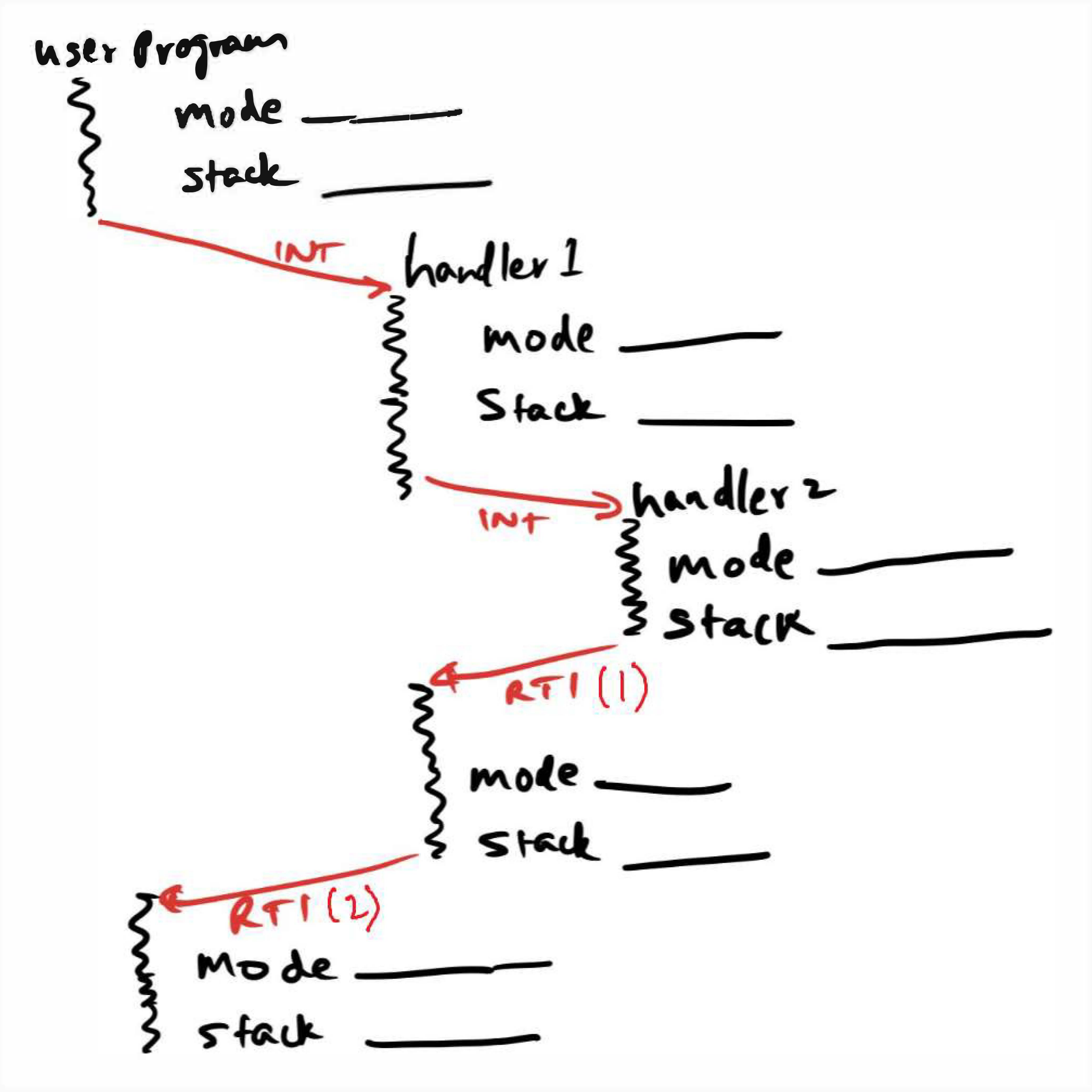
- Exception
- [✓ Trap]
- Interrupt

4.6 — (no title) **1** / 1 pt

✓	**+ 1 pt**	Correct
	+ 0.5 pts	Only Sync
	+ 0.5 pts	Only Trap
	+ 0 pts	Incorrect

Q5 System vs. User Stack

3 Points



Fill in the blanks (in questions 4.1 to 4.5) to indicate the mode (**user, kernel**) of the processor and the state (**user, system**) of the stack. The squiggly black lines indicate a program, and the red arrows indicate a change in what program is executing (with a label indicating what operation is happening).

Q5.1 User Program

1 Point

What mode is the processor in?

[✓ User]

Kernel

What state is the stack in?

[✓ User]

System

5.1 User Program **1** / 1 pt

✓	**+ 1 pt**	Correct
	+ 0 pts	Only User (Mode)
	+ 0 pts	Only User (Stack)
	+ 0 pts	Incorrect

Q5.2 Handler 1

1 Point

What mode is the processor in?

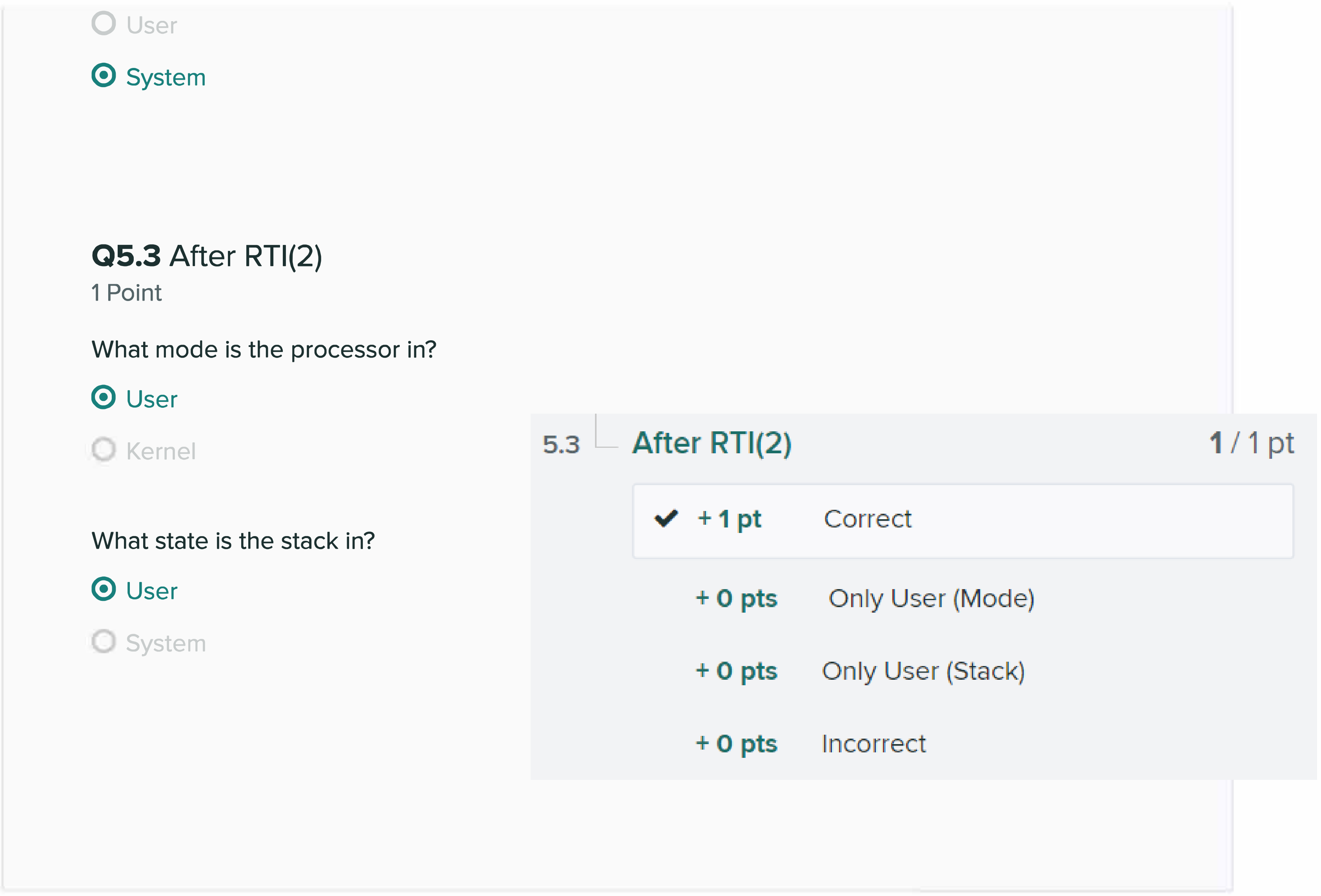
User

[✓ Kernel]

What state is the stack in?

5.2 Handler 1 **1** / 1 pt

✓	**+ 1 pt**	Correct
	+ 0 pts	Only Kernel
	+ 0 pts	Only System
✓	**+ 0 pts**	Incorrect



User [✓ System]

Q5.3 After RTI(2)

1 Point

What mode is the processor in?

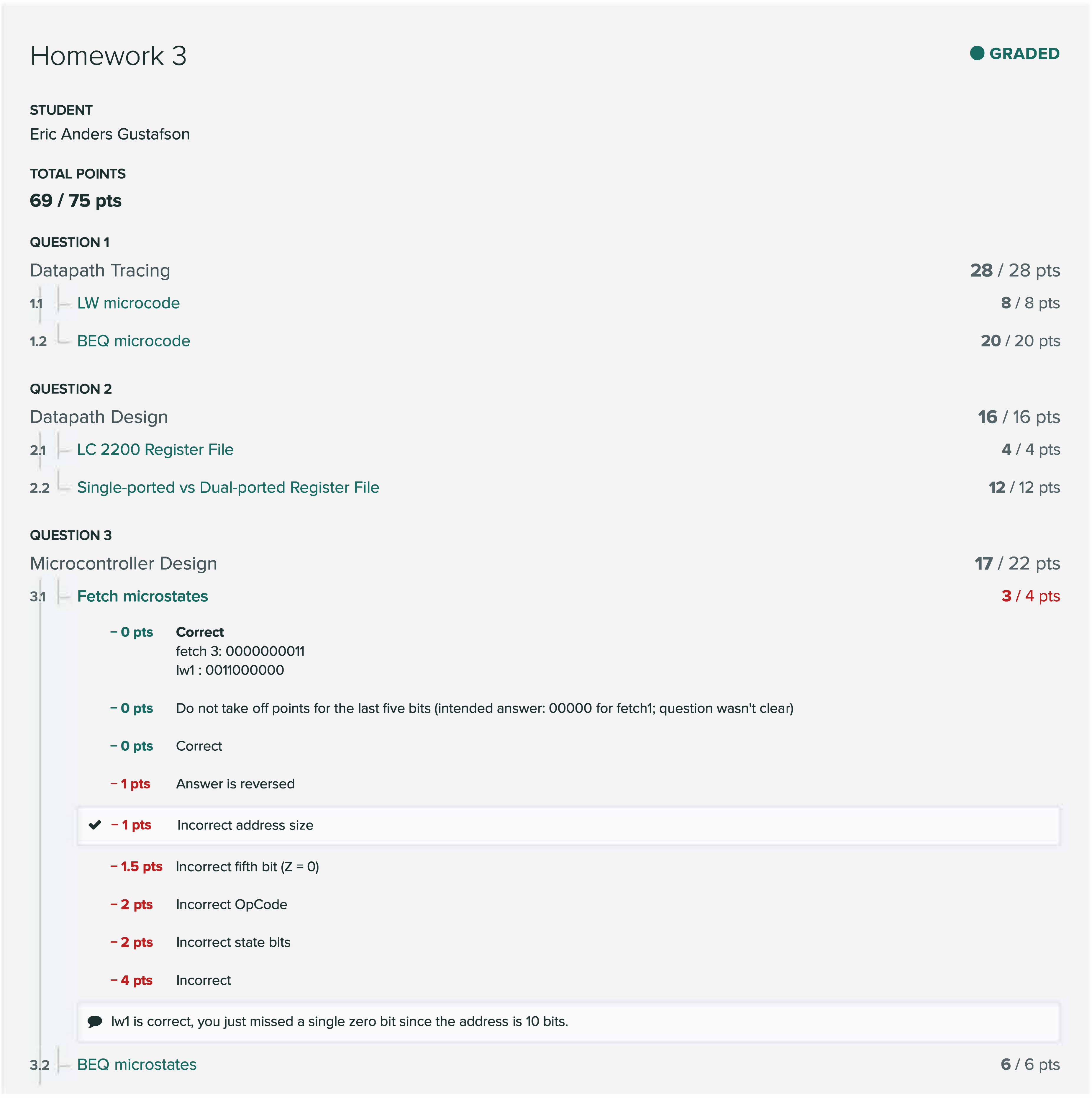
[✓ User] Kernel

What state is the stack in?

[✓ User] System

5.3 **After RTI(2)** **1** / 1 pt

✓	**+ 1 pt**	Correct
	+ 0 pts	Only User (Mode)
	+ 0 pts	Only User (Stack)
	+ 0 pts	Incorrect



Homework 3 ● **GRADED**

STUDENT

Eric Anders Gustafson

TOTAL POINTS

69 / 75 pts

QUESTION 1

	Datapath Tracing	**28** / 28 pts
1.1	LW microcode	**8** / 8 pts
1.2	BEQ microcode	**20** / 20 pts

QUESTION 2

	Datapath Design	**16** / 16 pts
2.1	LC 2200 Register File	**4** / 4 pts
2.2	Single-ported vs Dual-ported Register File	**12** / 12 pts

QUESTION 3

	Microcontroller Design	**17** / 22 pts
3.1	Fetch microstates	**3** / 4 pts

	− 0 pts	**Correct** fetch 3: 0000000011 lw1 : 0011000000
	− 0 pts	Do not take off points for the last five bits (intended answer: 00000 for fetch1; question wasn't clear)
	− 0 pts	Correct
	− 1 pts	Answer is reversed
✓	**− 1 pts**	Incorrect address size
	− 1.5 pts	Incorrect fifth bit (Z = 0)
	− 2 pts	Incorrect OpCode
	− 2 pts	Incorrect state bits
	− 4 pts	Incorrect

lw1 is correct, you just missed a single zero bit since the address is 10 bits.

3.2	BEQ microstates	**6** / 6 pts

3.3	M bit	**8** / 8 pts
3.4	Minimum bitsize of Next State Register	**0** / 4 pts

QUESTION 4

(no title) **5** / 6 pts

4.1	(no title)	**1** / 1 pt
4.2	(no title)	**1** / 1 pt
4.3	(no title)	**1** / 1 pt
4.4	(no title)	**0** / 1 pt
4.5	(no title)	**1** / 1 pt
4.6	(no title)	**1** / 1 pt

QUESTION 5

System vs. User Stack **3** / 3 pts

5.1	User Program	**1** / 1 pt
5.2	Handler 1	**1** / 1 pt
5.3	After RTI(2)	**1** / 1 pt