

Name: _____

Problem 1: (25 points) Arithmetic

The original reason for Booth's algorithm was to reduce the number of operations by avoiding operations when there were string of 0s and 1s. Revise the algorithm presented in class to look at 4 bits at a time and compute the product three bits at a time.

Recall the table for 2 bits (A is the multiplicand, B is the multiplier):

A_i	A_{i-1}	Operation
0	0	Do nothing
0	1	Add B
1	0	Subtract B
1	1	Do nothing

Fill in the following table to determine the 3-bit Booth encoding. Explain the reason for the operation on each line. Two of the cases are given for you.

Current Bits			Previous Bit	Operation	Reason
A_{i+2}	A_{i+1}	A_i	A_{i-1}		
0	0	0	0	None	Middle of a string of 0s
0	0	0	1		
0	0	1	1		
0	1	0	1		
1	0	1	0		
1	1	0	0		
1	1	0	1		
1	1	1	0		
1	1	1	1		

Name: _____

Problem 2: (35 points) Single Cycle Processor

The following single cycle processor seems to be executing random instructions. You must investigate and find out why. You suspect that the control is the cause of the problem.

On the next page you will find a picture of the datapath, and the corresponding control signals are below.

	PCSrc	RegDst	RegWr	ExtOp	ALUSrc	ALUCtr	MemWr	MemToReg
addu	0	0	1	1	x	0	0	0
subu	0	1	1	x	0	0	0	0
ori	0	1	1	0	x	2	0	0
lw	0	1	1	1	1	0	1	1
sw	0	x	0	0	1	0	1	x
beq	Equal	x	0	x	0	3	0	x
jr	2	x	0	x	x	x	0	x

The ALUCtr encoding is as follows:

Control Bits	Operation
0	add
1	sub
2	or
3	xor

Assume the following:

- “Equal” means that PCSrc takes on the value of the equal signal coming out of the “= 0?” module. This will either be 0 or 1.
- The “= 0?” module outputs 1 if the input to the module is 0, otherwise it outputs 0.
- The modules within the datapath (i.e. extender, alu) all work correctly.
- The register file (RegWr) and data memory (MemWr) both write when their respective write signals are 1
- The extender will zero extend if ExtOp is 0, and will sign extend when ExtOp is 1

Part a) (25 points) What does the broken processor actually do for the following sequence of instructions? The first instruction has been done for you as an example. If there is more than one possibility, please list all of them (note that this may be a different instruction, correct, behavior, or an undefined instruction). If the incorrect result does not match a valid MIPS instruction, give a sequence of instructions that correspond to the behavior. Also give a very brief explanation of your possibilities. For simplicity, we have used the actual register number rather than the names.

Name: _____

Intended Instruction	Actual Instruction(s)
addu \$1, \$2, \$0	addu \$1, \$2, \$0 (if ALUSrc = 0) - correct addiu \$1, \$2, 33 (if ALUSrc = 1) - incorrect
subu \$4, \$5, \$6	
ori \$7, \$8, 0x0025	
beq \$11, \$12, 24	
sw \$10, -12(\$31)	
lw \$9, -16(\$29)	

Name: _____

Part b) (10 points) Describe the changes to the control signals that must be done to make the processor function correctly. The control signals from the previous page have been reprinted for your convenience.

	PCSrc	RegDst	RegWr	ExtOp	ALUSrc	ALUCtr	MemWr	MemToReg
addu	0	0	1	1	x	0	0	0
subu	0	1	1	x	0	0	0	0
ori	0	1	1	0	x	2	0	0
lw	0	1	1	1	1	0	1	1
sw	0	x	0	0	1	0	1	x
beq	Equal	x	0	x	0	3	0	x
jr	2	x	0	x	x	x	0	x

Name: _____

Problem 3: (25 points) Multiple Cycle Processor

MIPS is a register-register architecture, where the arithmetic source and destinations must be registers. But let's say we wanted to add a register-memory instruction:

```
addm rd, rs, rt    # rd = rs + Mem[rt]
```

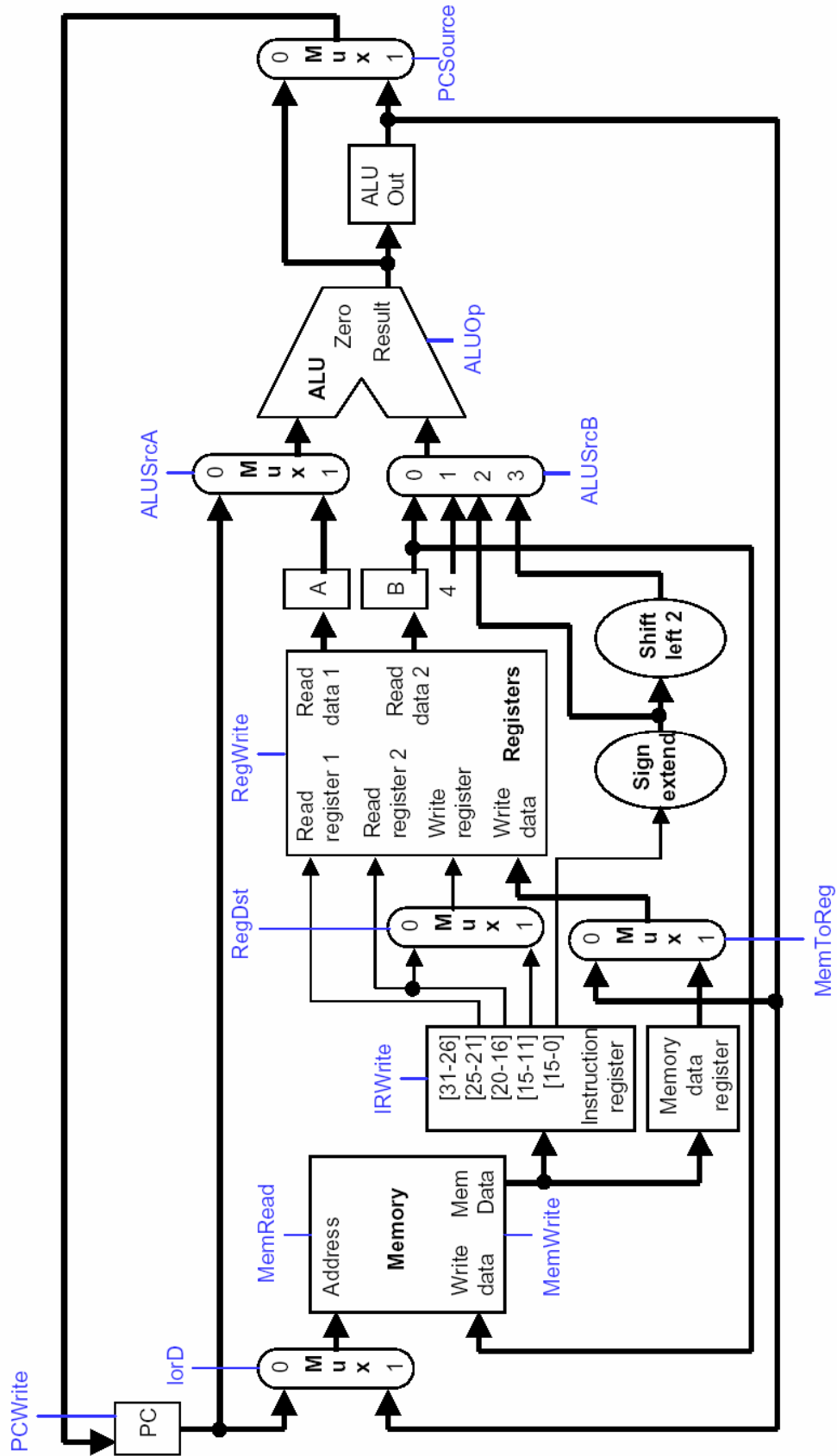
It is an R-type instruction. Here is the R-type format, for your reference (shamt and func are not used):

Field	op	rs	rt	rd	shamt	func
Bits	[31:26]	[25:21]	[20:16]	[15:11]	[10:6]	[5:0]

The multiple cycle datapath is shown on the next page. You may assume that $ALUOp = 010$ performs an addition.

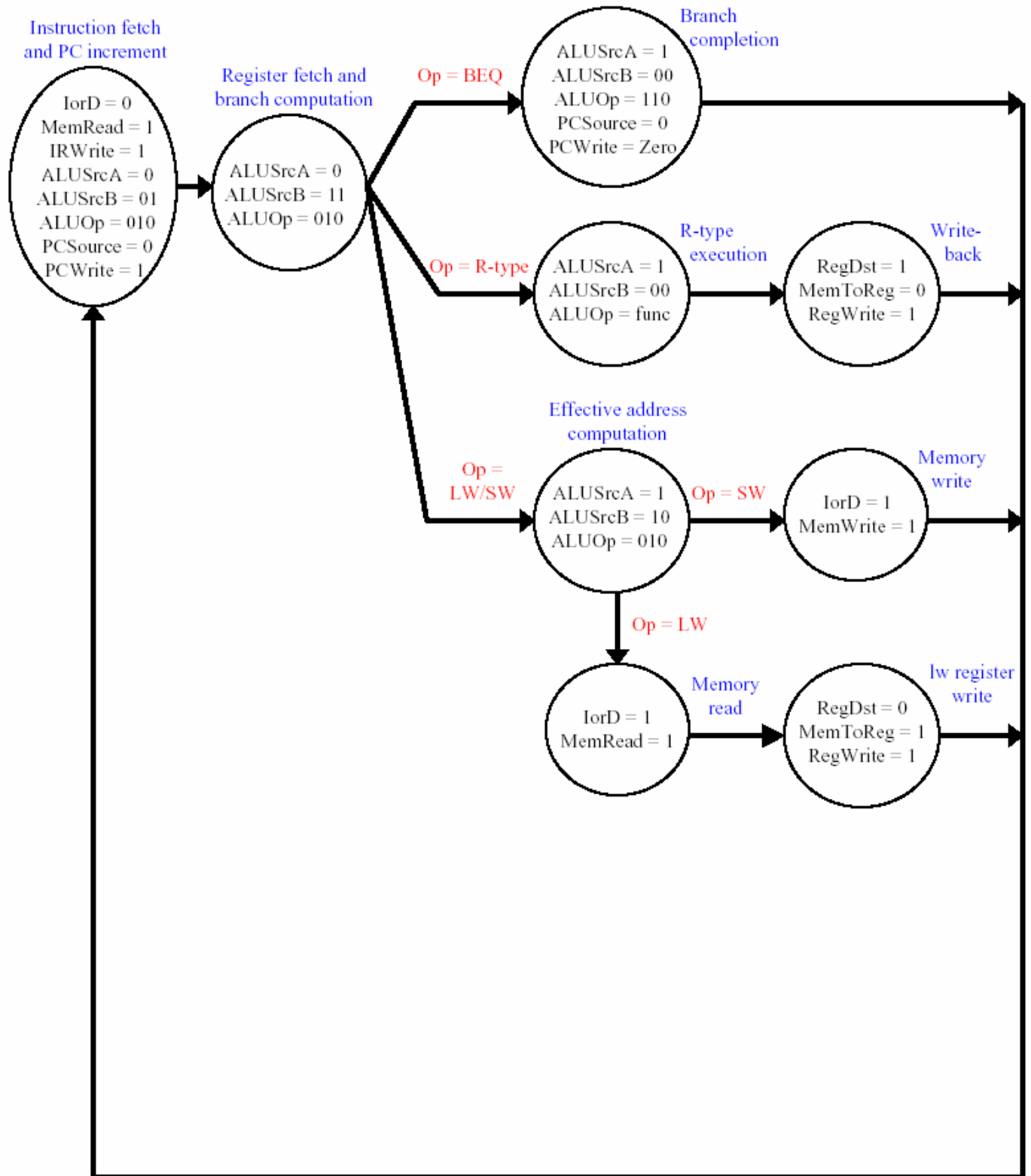
Part a) (10 points) On the next page, show what changes are needed to support *addm* in the multicycle datapath.

Name: _____



Part b) (15 points)

Complete this finite state machine diagram for the *addm* instruction. Be sure to include any new control signals you may have added.



Name: _____

Problem 4: (25 points) Pipelined Processor

The next page shows a diagram of the pipelined datapath with forwarding, but no hazard detection unit.

Part a) (5 points)

Both of the code fragments below have dependencies, but only one of them will execute correctly with the forwarding datapath. Tell us which one, and why. If you like, you can draw a pipeline diagram to aid in your explanation.

```
add $t0, $a0, $a1
add $v0, $t0, $t0
```

```
lw $t0, 0($a0)
add $v0, $t0, $t0
```

Part b) (5 points)

Here is one more code fragment. How is this one different from the previous ones?

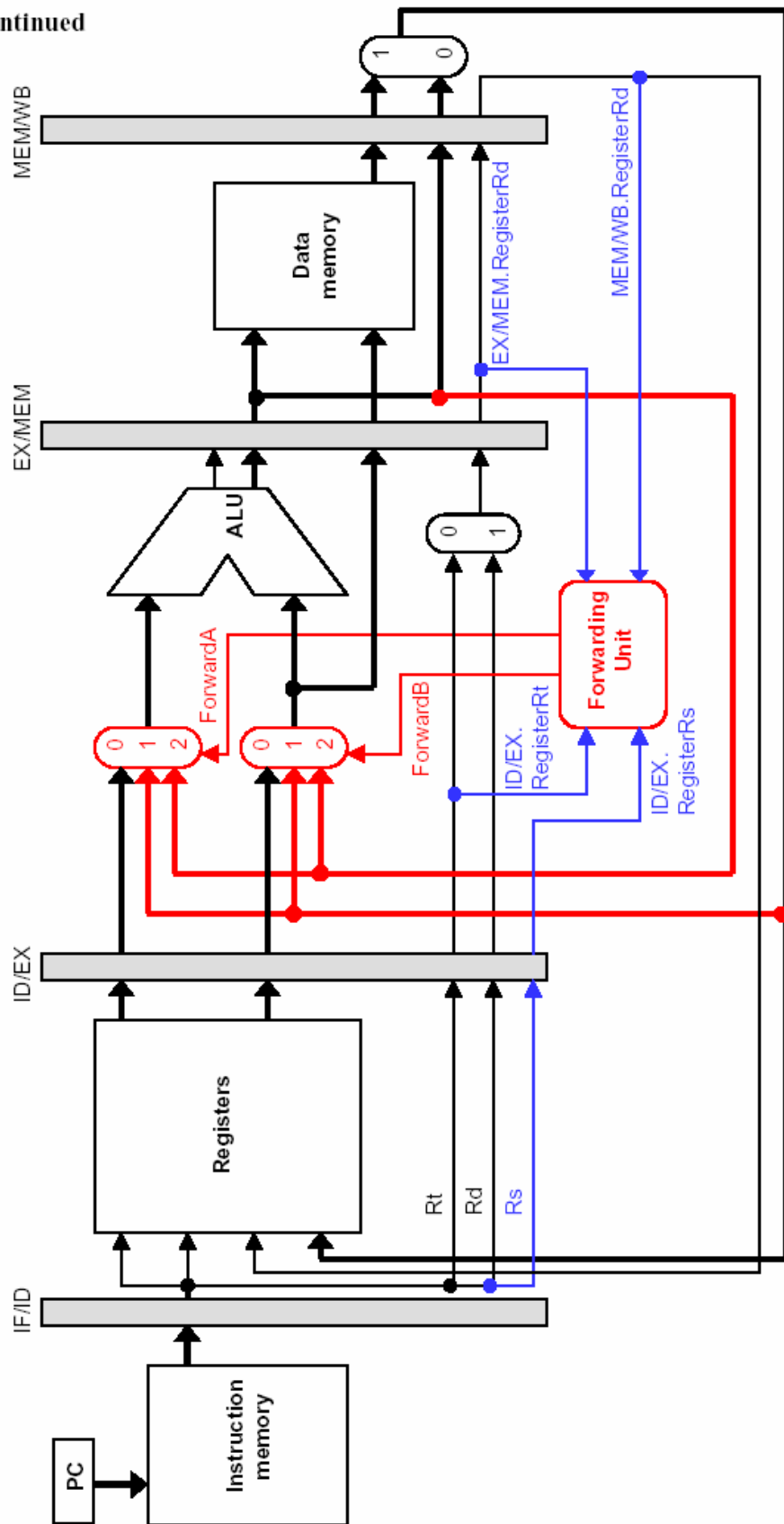
```
lw $t0, 0($a0)
sw $t0, 0($a1)
```

Part c) (15 points)

It is possible to modify the datapath so the code in Part b) executes without *any* stalls. Explain how this could be done, and show your changes on the next page.

Name: _____

continued



Name: _____

Problem 5: (40 points) Caches

$$AMAT = \text{Hit time} + (\text{Miss rate} \times \text{Miss penalty})$$

(4 points) A processor has a direct-mapped cache capable of holding B bytes of data in total, and each block of the cache contains b bytes of data (both B and b are powers of two). How many bits does it take to implement the cache?

Now you must figure out what the values of B and b are for the processor. Consider the following code:

```
// We will choose values for LENGTH and STEP
char a[LENGTH];
int i, j, temp;
for(i = 0; i < 10000; i++)
    for(j = 0; j < LENGTH; j = j + STEP)
        temp = temp + a[j];
```

How does this code help us?

Part a) Suppose STEP is initially 1.

(4 points) If $LENGTH \leq B$, how many cache misses would we expect each time the outer for-loop is executed?

(4 points) How would things change if we increased LENGTH until it became larger than $2B$?

Name: _____

(4 points) What would happen if we kept increasing LENGTH?

Part b) Now suppose we fix LENGTH to some value larger than $2B$, and increase STEP:

(4 points) If STEP is less than b , how many cache misses would we expect each time the outer for-loop is executed?

(4 points) After STEP equals b , what would happen if we kept increasing STEP?

Now that we know what to expect, let's run some experiments. We will vary the values of LENGTH and STEP, and compute the average memory-access time (AMAT) in the statement

```
temp = temp + a[j];
```

The table below lists the AMAT (in nanoseconds) for different values of LENGTH and STEP. Our time measurements are not perfectly accurate, so there will be some fluctuations in the data:

Name: _____

		STEP							
LENGTH		1	2	4	8	16	32	64	128
8		50	53	50	50	50	50	50	50
16		50	47	50	51	50	50	50	50
32		54	49	50	50	50	50	50	50
64		106	168	275	500	500	502	52	50
128		108	162	275	506	500	500	497	50
256		101	162	272	500	507	498	500	500

Part c) Drawing inferences from the data:

(4 points) What is the value of B? Explain your answer.

(4 points) What is the value of b? Explain your answer.

(4 points) What is the approximate time for a cache hit?

(4 points) What is the approximate time for a cache miss?