MCS-284 Intra-term Exam 2

Serial #:

This exam is closed-book and mostly closed-notes. You may, however, use a single 8 1/2 by 11 sheet of paper with *hand-written* notes for reference. (Both sides of the sheet are OK.)

Please write your name only on this page. Do not turn the page until instructed, in order that everyone may have the same time. Then, be sure to look at all problems before deciding which one to do first. Some problems are easier than others, so plan your time accordingly. **You have 50 minutes to work.**

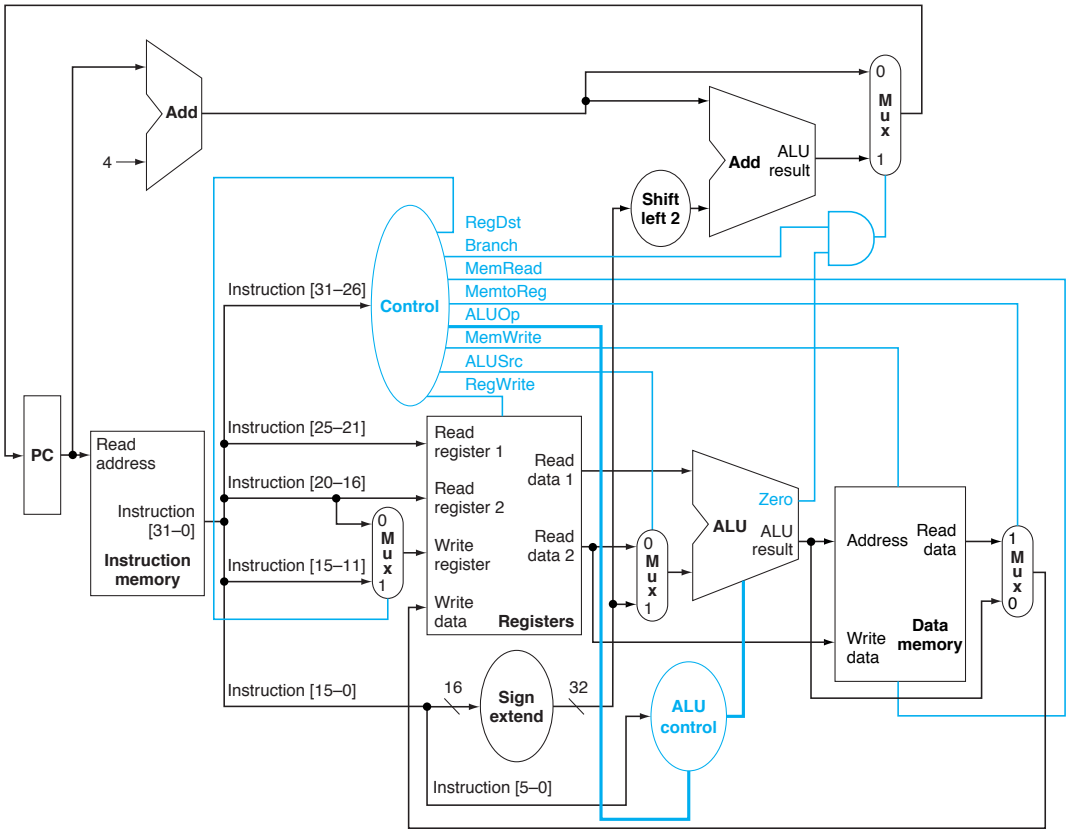
Write the answer to each problem on the page on which that problem appears. You may also request additional paper, which should be labeled with your test number and the problem number.

Printed name: _____

Problem	Page	Possible	Score
1	2	25	
2	3	24	
3	4	25	
4	6	26	
Total		100	

1. [**25 Points**] Make any necessary modifications to the datapath and control table, reproduced below, of the single-cycle processor to add a new instruction, `lwr`. The `lwr` (or Load Word Registers) instruction loads a value from memory into a register, like `lw` does. The difference is in how the address of the memory location is calculated. Recall that `lw` adds an offset specified in the instruction to the value of a base register to find the address. With `lwr`, we again add two quantities to find the address, but this time *both* of them come from registers. Thus an `lwr` instruction contains three registers: one to receive the loaded value, and two to provide the address. We will use the same general layout as for R-format instructions, with the rs and rt fields specifying the address registers, and the rd field specifying the register to receive the result. (Rs is bits 25–21, rt is bits 20–16, and rd is bits 15–11.) A different opcode is used than for normal R-format instructions, however.

   (a) Add any new features you need to the datapath below.



   (b) Fill in the extra row in the control table below for the new `lwr` instruction. If you added any new control signals in the previous part of this problem, add columns for them to this table and show the values in those new columns in all rows, old as well as new.

Instruction	Reg Dst	ALU Src	Memto Reg	Reg Write	Mem Read	Mem Write	Branch	ALU Op1	ALU Op0
R-format	1	0	0	1	0	0	0	1	0
`lw`	0	1	1	1	1	0	0	0	0
`sw`	X	1	X	0	0	1	0	0	0
`beq`	X	0	X	0	0	0	1	0	1
`lwr`									

2. [**24 Points**] For each of the following eight sequences of instructions, indicate whether chapter 6's pipelined processor would use forwarding, stalling, both, or neither. As usual, we are assuming that registers are written in the first half of the clock cycle and read during the second half; this doesn't count as forwarding.

(a)
```
add $2, $11, $3
lw $5, 16($1)
add $6, $1, $7
add $10, $8, $9
```

(b)
```
add $2, $11, $3
lw $5, 16($1)
add $6, $5, $7
add $10, $8, $9
```

(c)
```
lw $5, 16($1)
add $10, $8, $9
add $6, $5, $7
add $2, $11, $3
```

(d)
```
lw $5, 16($1)
add $10, $8, $9
add $2, $11, $3
add $6, $5, $7
```

(e)
```
add $2, $11, $3
add $5, $2, $4
lw $1, 16($7)
add $10, $8, $1
```

(f)
```
lw $5, 16($1)
add $10, $8, $9
add $6, $7, $10
add $2, $11, $3
```

(g)
```
add $2, $11, $3
sw $2, 16($1)
add $6, $1, $7
add $10, $8, $9
```

(h)
```
add $2, $11, $3
lw $5, 16($1)
add $6, $1, $7
sw $5, 16($8)
```

3. [**25 Points**] Modify the datapath and state machine of the multi-cycle processor to add a new instruction, `swai`. The `swai` (or Store Word And Increment) instruction stores a value into memory from a register, like `sw` does. However, it afterwards also increments the base address register by 4. The reason to add this instruction is that programs have many loops that store values into consecutive words of memory. Those loops often have two instructions like

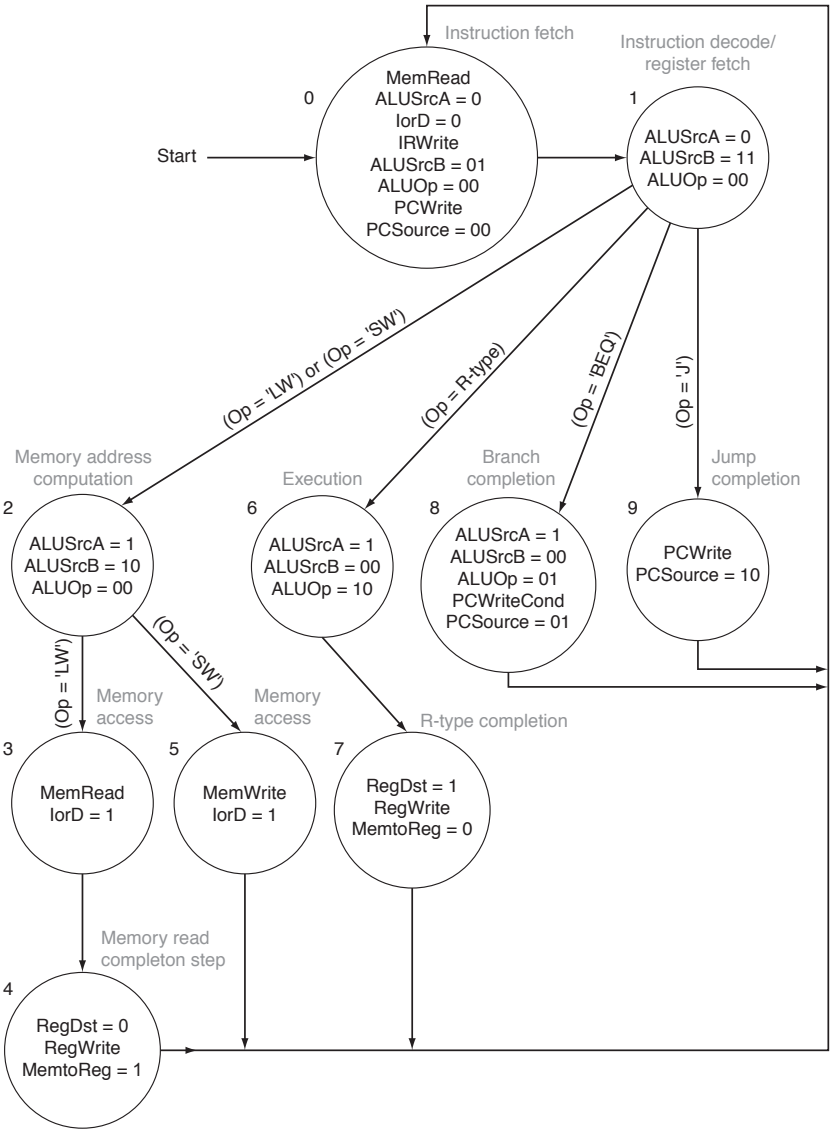
```
sw $t0, 16($t1)
addi $t1, $t1, 4
```

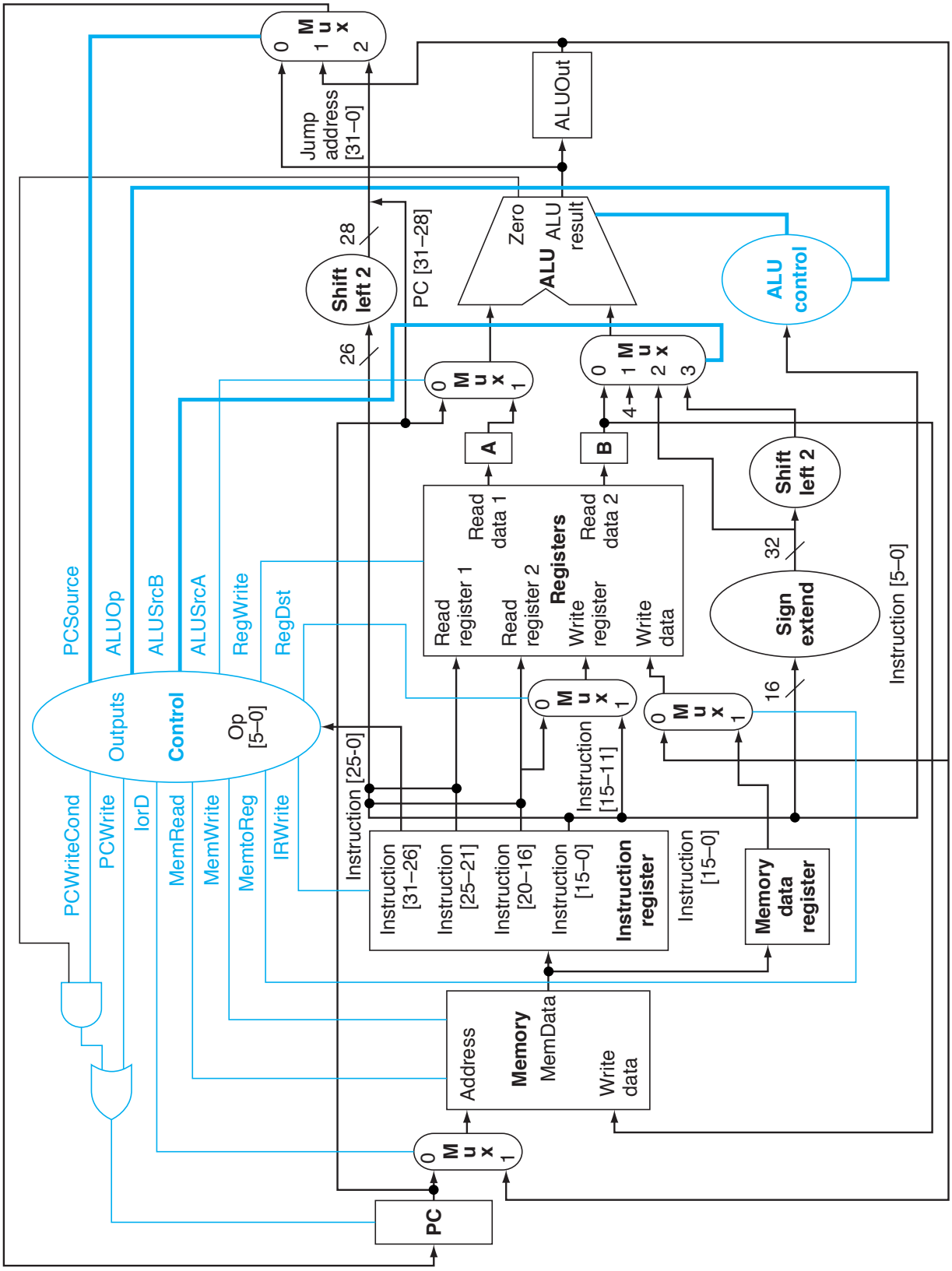
With the new `swai` instruction, the above two instructions could be replaced with one:

```
swai $t0, 16($t1)
```

The machine-language format of `swai` uses the Rs, Rt, and Imm fields in the same way as `sw`.

The datapath you are to modify is on the next page, and the state machine to modify is below.




M u x
0
1
2
Jump address [31–0]
ALUOut
Zero
ALU
ALU result
Shift left 2
28
26
PC [31–28]
ALU control
M u x
0
1
2
3
4
A
B
Shift left 2
Read data 1
Read register 1
Read register 2
Registers
Read data 2
Write register
Write data
32
Sign extend
Instruction [5–0]
16
PCSource
ALUOp
ALUSrcB
ALUSrcA
RegWrite
RegDst
Outputs
Control
Op [5–0]
PCWriteCond
PCWrite
IorD
MemRead
MemWrite
MemtoReg
IRWrite
Instruction [25-0]
Instruction [15–11]
Instruction [31–26]
Instruction [25–21]
Instruction [20–16]
Instruction [15–0]
Instruction register
Instruction [15–0]
Memory data register
Address
Memory
MemData
Write data
PC

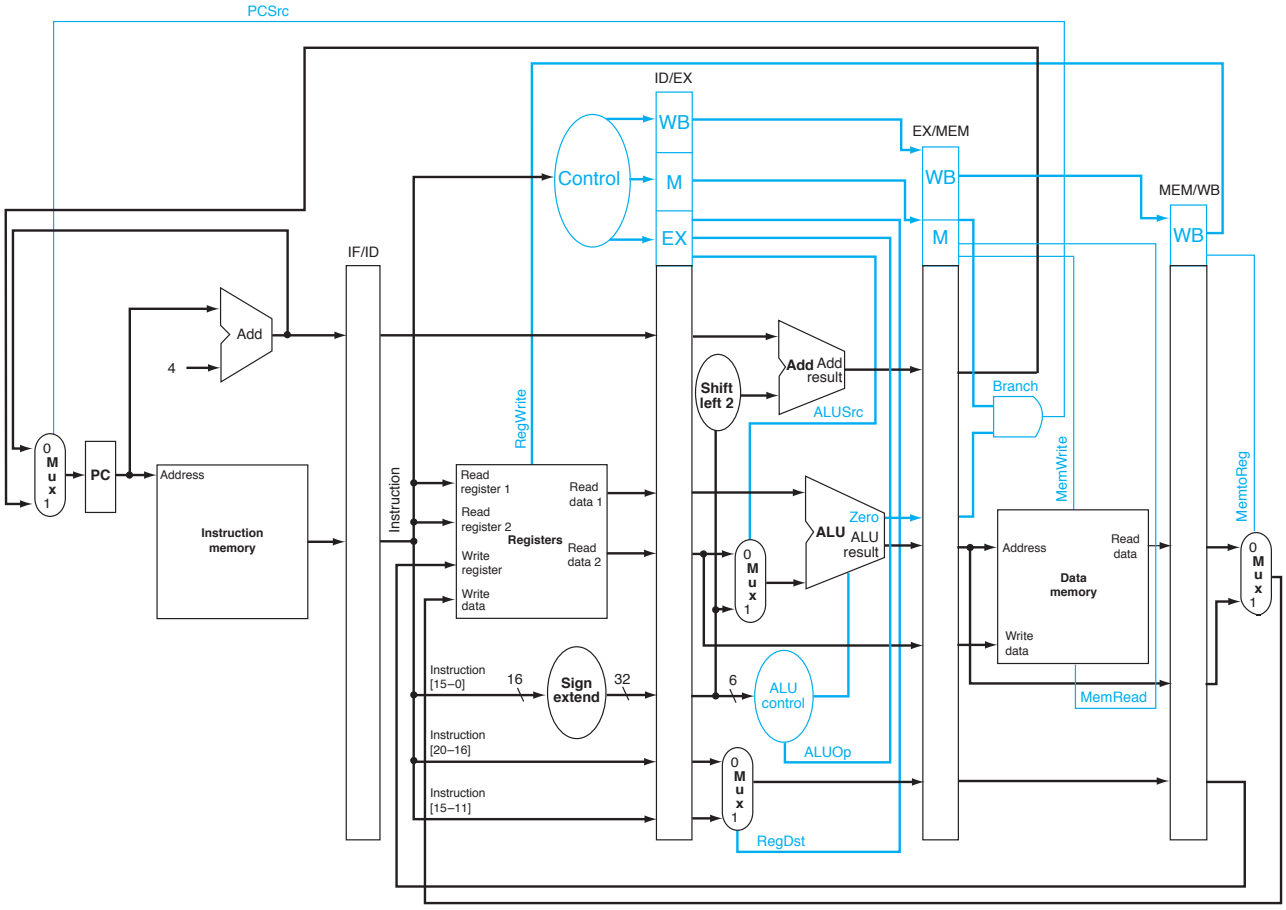
4. [**26 Points**] The diagram on the next page shows the pipelined datapath. Thirteen lines on it have been marked with circled letters, (a) through (m). Suppose we execute the following sequence of instructions. During the first clock cycle, the first instruction is fetched from address 1000 in instruction memory:

```
add $9, $5, $6
lw $4, 16($8)
sw $1, 32($2)
beq $7, $3, 100
```

Suppose further that the following registers and data memory locations contain the specified values when execution starts:

Registers		**Data Memory Locations**	
1:	100	216:	13
2:	200	232:	26
3:	300	416:	39
4:	400	432:	52
5:	500	816:	65
6:	600	832:	78
7:	700		
8:	800		
9:	900		

During the fourth clock cycle, what value is on each line? If there isn't enough information given for you to know the value, write N/A.



(a)

(b)

(c)

(d)

(e)

(f)

(g)

(h)

(i)

(j)

(k)

(l)

(m)