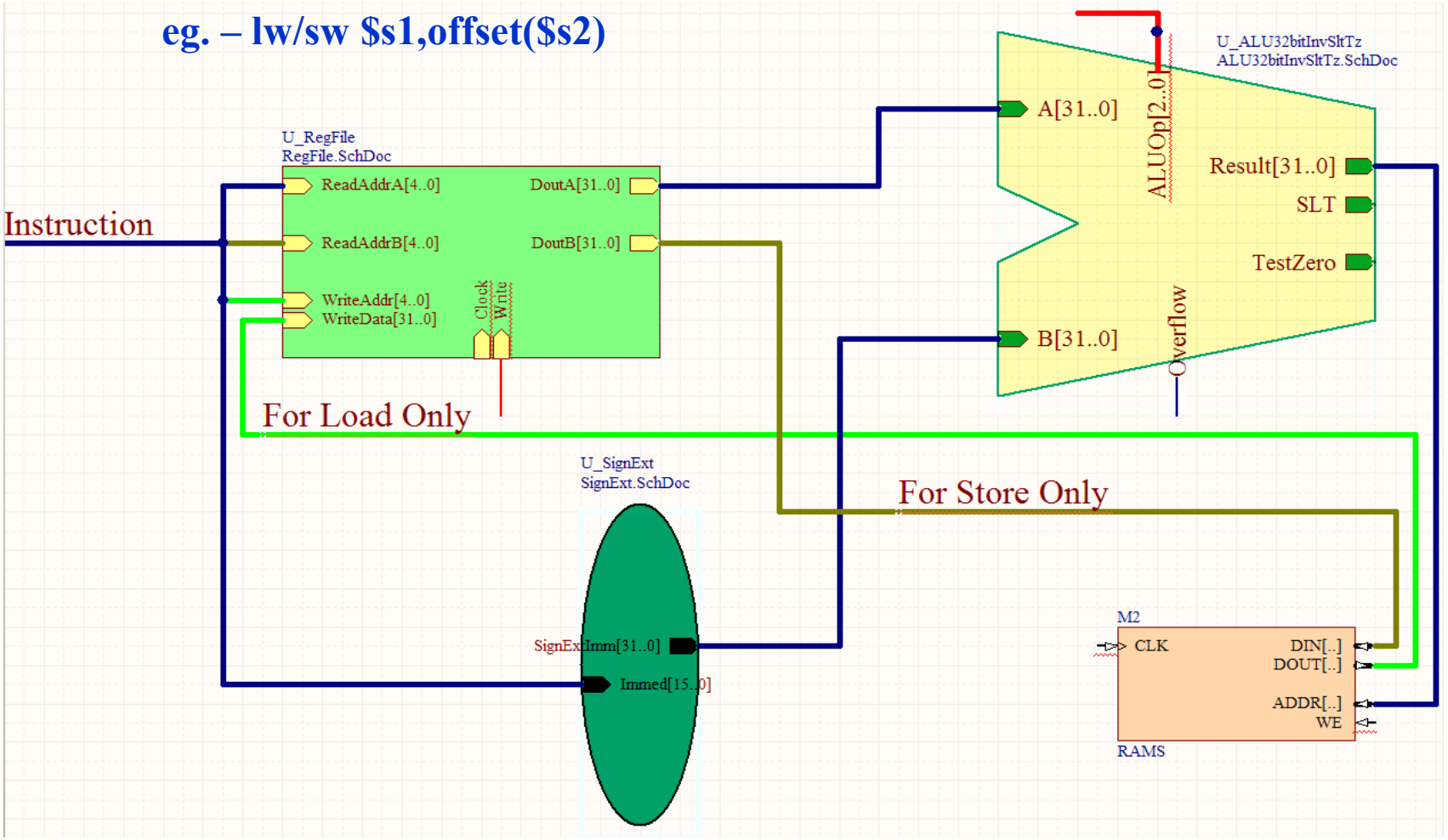
CSE 2021
COMPUTER ORGANIZATION

HUGH CHESSER
CSE B 1012U

WP-M

Datapath: Data transfer Instruction (3)

Load and store instruction combined



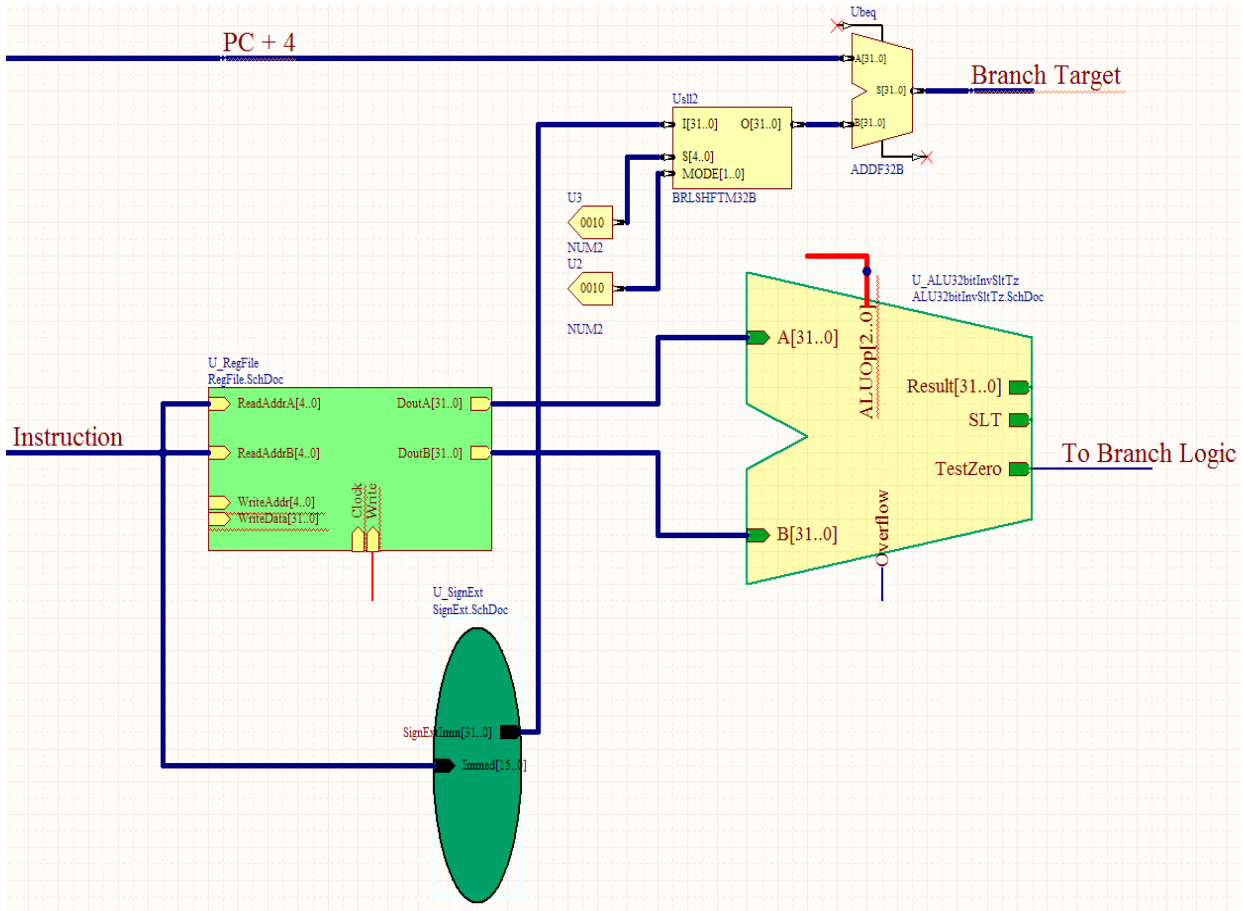
Datapath: Branch Instructions

Example: `beq $s1,$s2,Loop`

Compiler translation:

```
beq $s1,$s2,w_offset
#if $s1==$s2, goto (PC+4+4*w_offset)
```

1. Read two registers ($s2,$s3) specified in the instruction
2. ALU compares content of specified registers ($s1,$s2)
3. Adder computes the branch address
4. If equal (zero = 1), branch address is copied to PC



Agenda

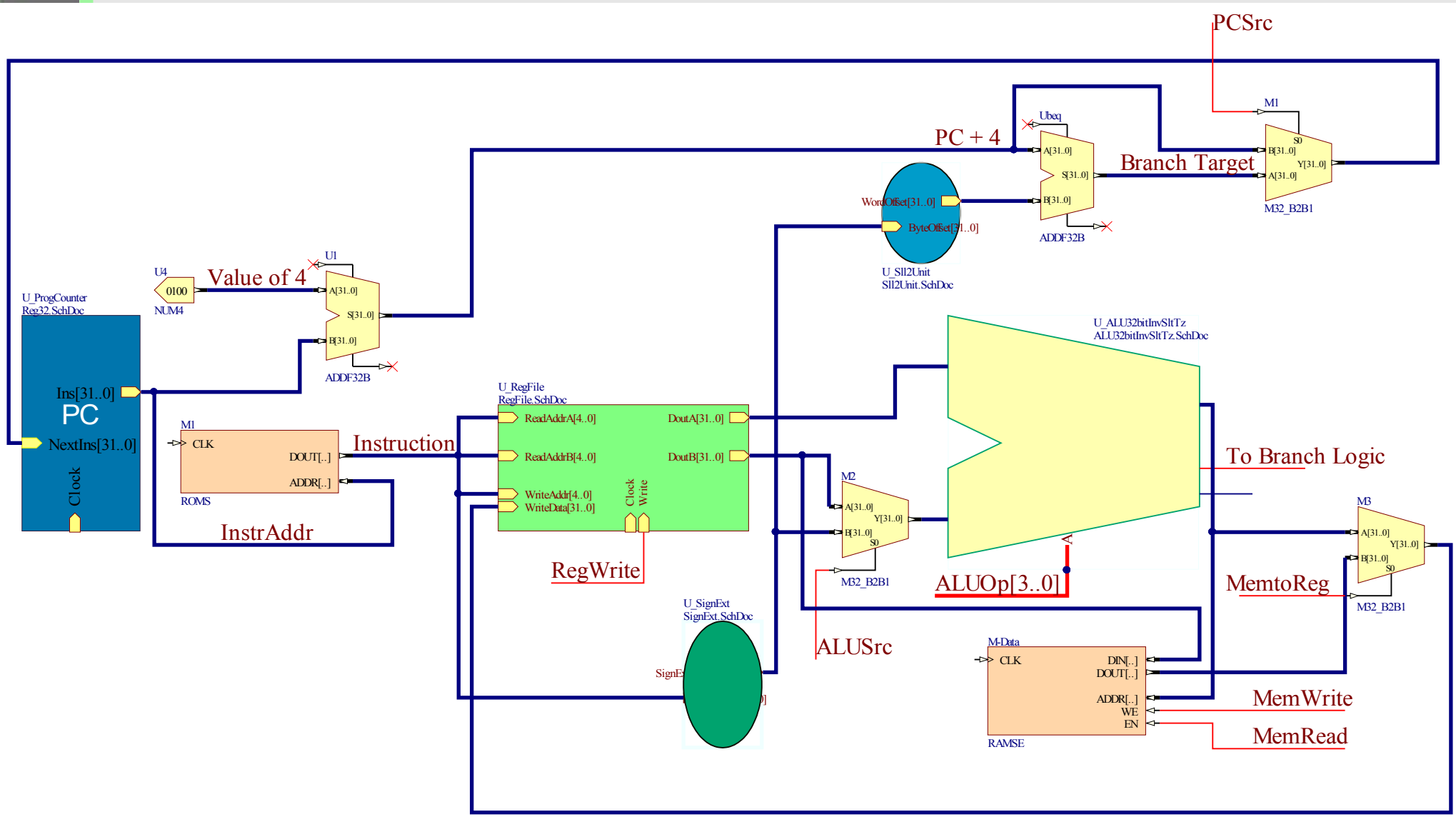
Topics:

1. A single cycle implementation
2. State Diagrams
3. A multiple cycle implementation

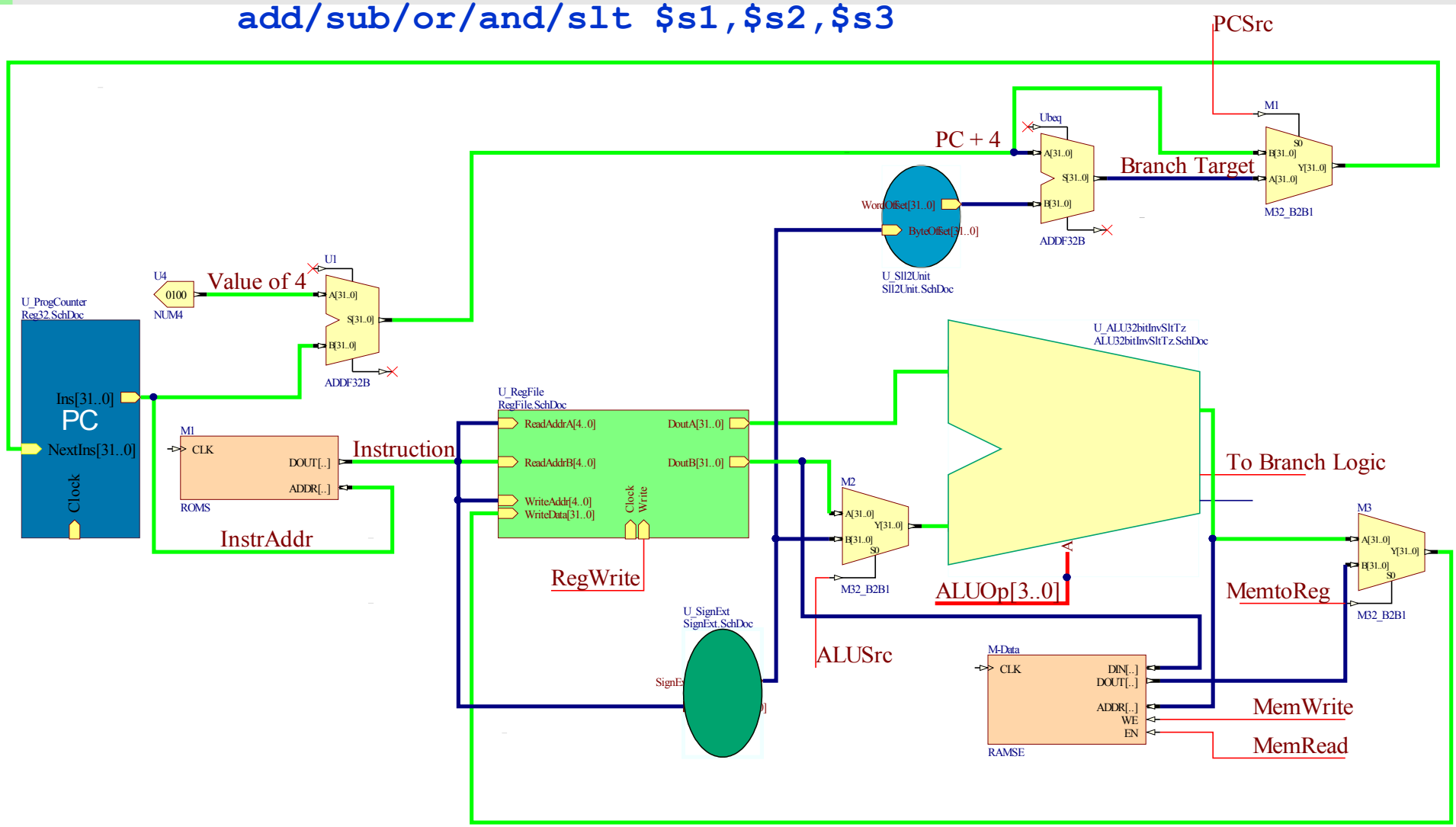
Today and Wednesday - Patterson: Section 4.3, 4.4

Reminder: Make-up Labs tonight and tomorrow

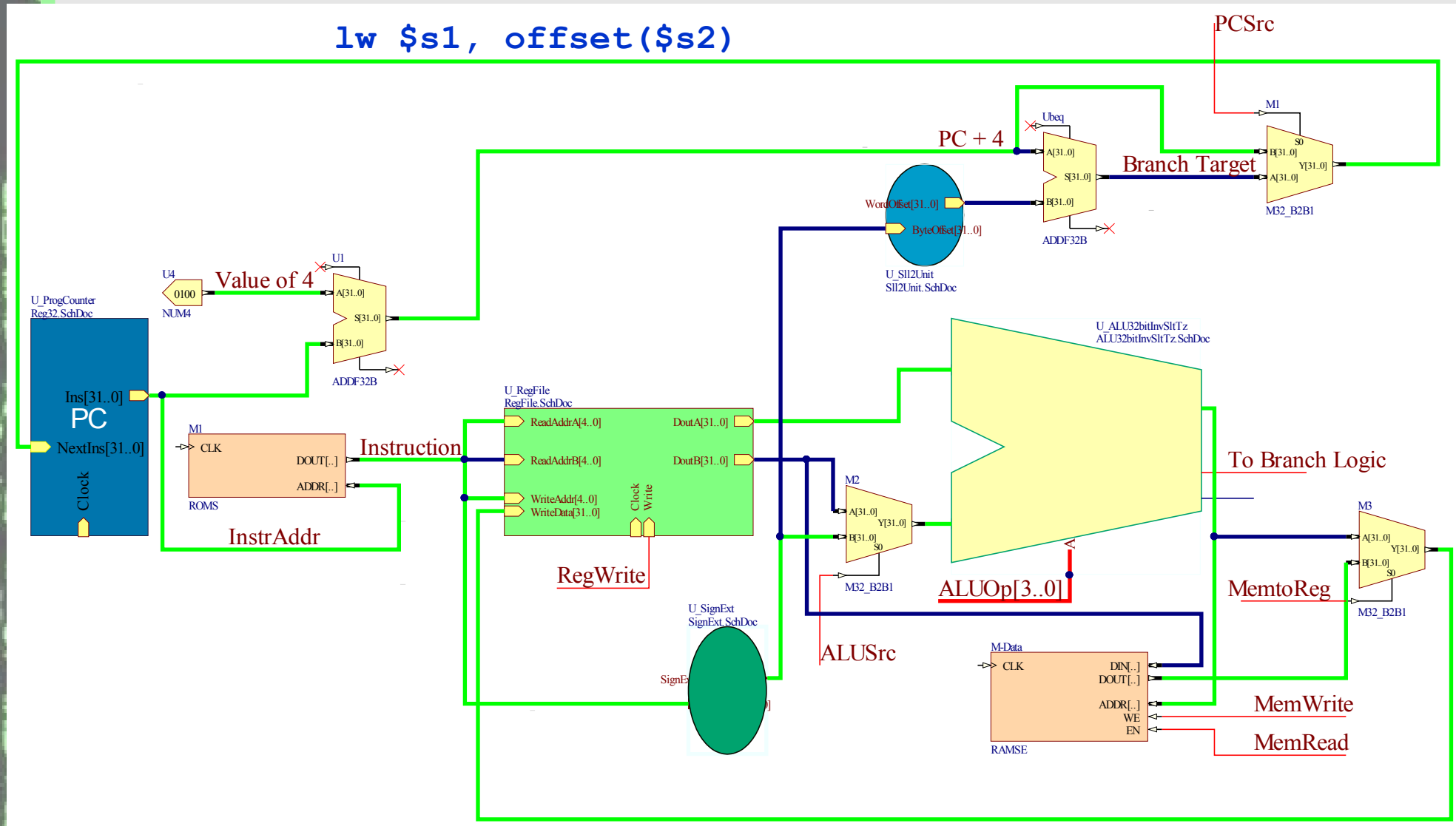
Combined Datapath



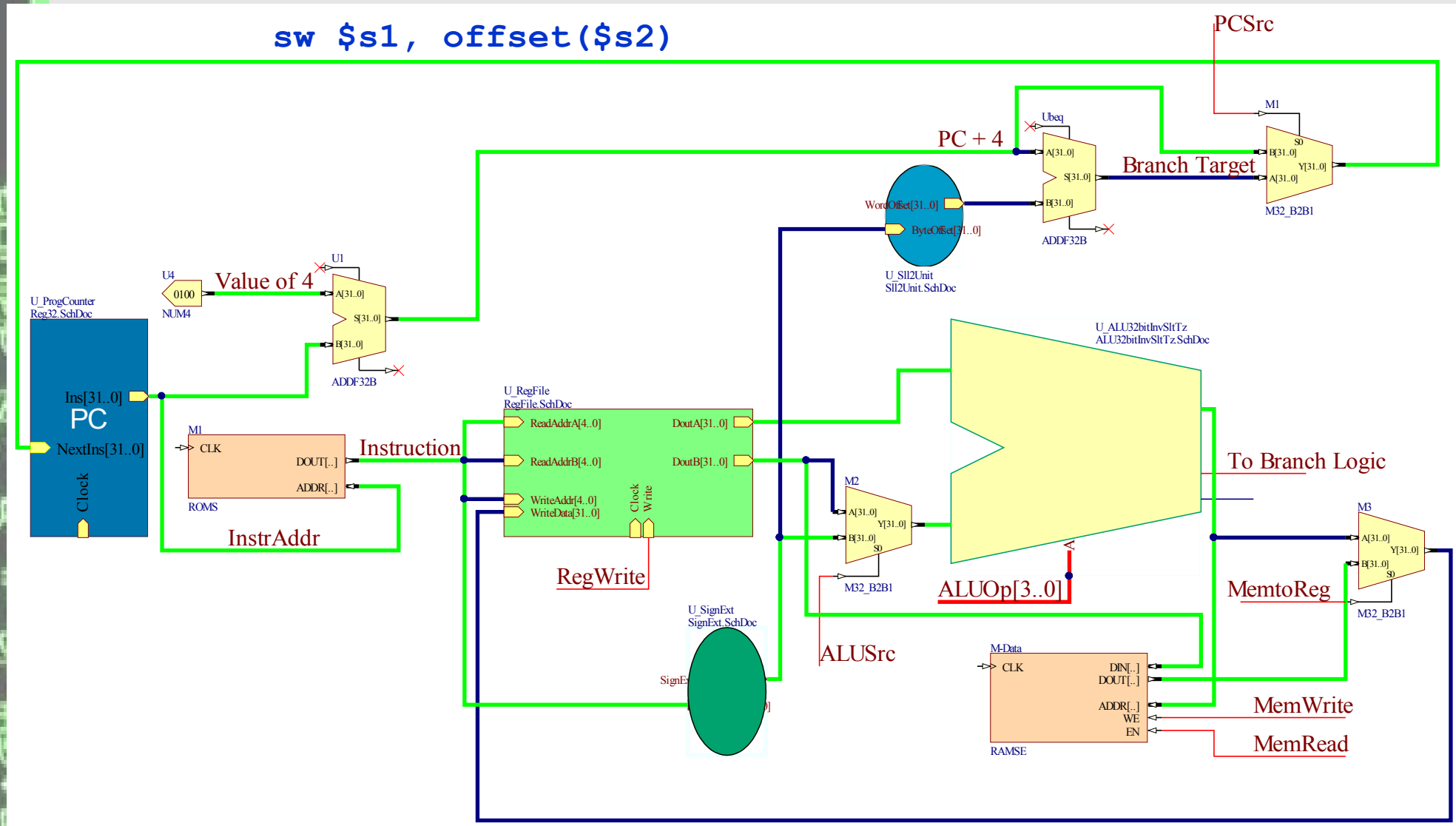
Combined Datapath: Arithmetic Operations (add,sub,or,and,slt)



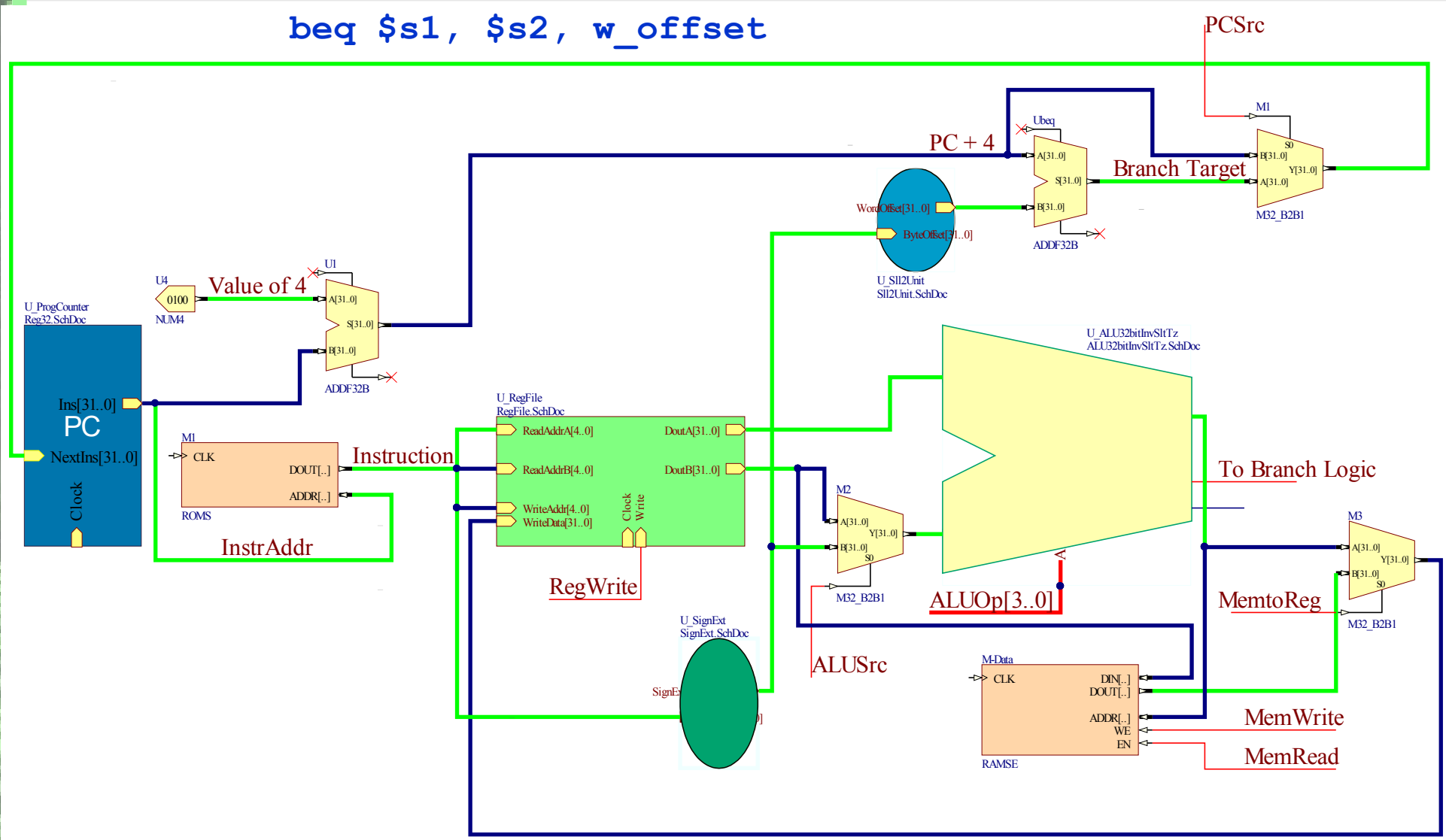
Combined Datapath: Data Transfer (load)



Review: Data Transfer (store)



Review: Data Transfer (branch)



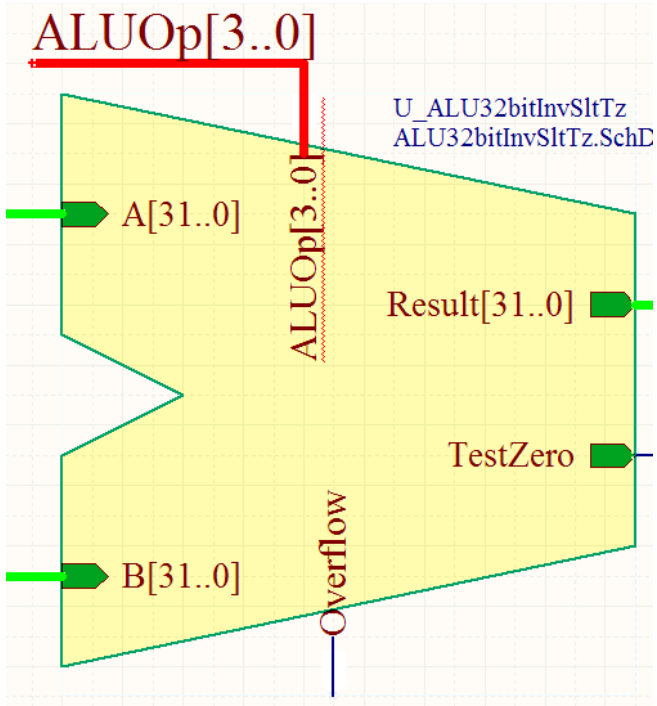
Types of Datapath

- Datapaths can be divided into two categories
  1. Single cycle datapath: An instruction (e.g. `lw`) is performed in 1 clock cycle
  2. Multiple cycle datapath: An instruction is performed in multiple clock cycles
- Advantage of single datapath: Keeps the design simpler
- Disadvantage of single datapath:
  1. No datapath resource can be used more than once in a clock cycle
  2. Elements being accessed more than once in an instruction must be duplicated or have multiple inputs and outputs.
  3. An instruction memory separate from data memory is required
- In section 4.3, a single cycle datapath is described. We expand the discussion to multiple cycle datapaths in section 4.4.

Control

- What needs to be added to complete the design of the single cycle datapath?
- Control Unit activates appropriate controls at
  1. ALU
  2. MUX's (3)
  3. Read/write signals for register file
  4. Read/write signals for data memory
- Consider the design of the ALU Control Unit

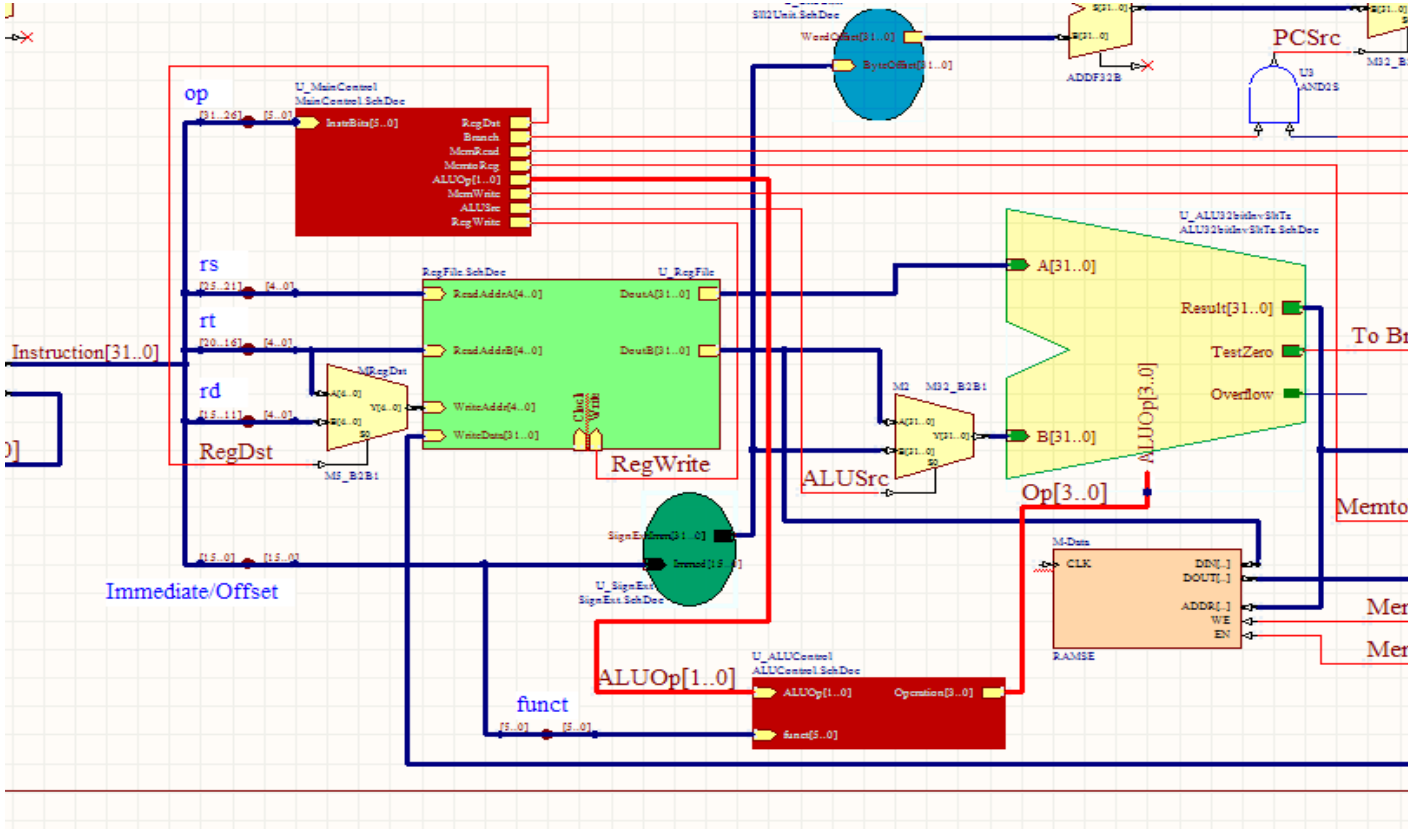
ALU Control Lines	Result
0000	AND
0001	OR
0010	Add
0110 (1010)	Subtract
0111 (1011)	SLT
1100 (0100)	NOR



Control: ALU Control Unit (1)

— Inputs to ALU Control:

1. Function fields of instructions
2. 2-bit control field (ALUOp)
3. Output of ALU Control: 4-bit signal that controls the ALU



Control: ALU Control Unit (2)

I-format:	op (31..26)	rs (25..21)	rt (20..16)	Immediate / Offset (15..0)

1. I-format instructions (**`lw/sw/beq`**) does not have a function field
2. Use ALUOp to differentiate the operation from the rest
   e.g., ALUOp = 10 for lw/sw; ALUOp = 00 for beq

R-format:	op (31..26)	rs (25..21)	rt (20..16)	rd (15..11)	shamt (10..6)	funct (5..0)

1. R-format instructions (**`add/sub/and/or`**) does have a function field
2. ALUOp = 10 for all instructions
3. Function field is 32 for add, 34 for sub, 36 for AND, 37 for OR, and 42 for SLT

Control: ALU Control Unit (3)

Instruction (funct)	Inputs: ALUOp (ALUOp1 – ALUOp0)	Inputs: Function Field (F5 – F0)	Desired ALU action	Outputs: Operation (Op3 – Op0)
lw (I)	0 0	X X X X X X	**add**	**0 0 1 0**
sw (I)	0 0	X X X X X X	**add**	**0 0 1 0**
beq (I)	0 1	X X X X X X	**sub**	**0 1 1 0**
add (32)	1 0	1 0 0 0 0 0	**add**	**0 0 1 0**
sub (34)	1 0	1 0 0 0 1 0	**sub**	**0 1 1 0**
and (36)	1 0	1 0 0 1 0 0	**and**	**0 0 0 0**
or (37)	1 0	1 0 0 1 0 1	**or**	**0 0 0 1**
slt (42)	1 0	1 0 1 0 1 0	**slt**	**0 1 1 1**

1. The first two rows of the truth table are the same.
2. Number of inputs in the truth table are 8 while number of rows in the table are 8.
3. Only 8 of the $2^8 = 128$ possible combinations of input are being used.
4. Some "DON'T CARE" entries can be added to simplify the above truth table.

Control: ALU Control Unit (4)

Instruction (funct)	Inputs: ALUOp (ALUOp1 – ALUOp0)	Inputs: Function Field (F5 – F0)	Desired ALU action	Outputs Operation (Op3 – Op0)
lw (I)	0 X (0 0)	X X X X X X	**add**	**0 0 1 0**
sw (I)	0 X (0 0)	X X X X X X	**add**	**0 0 1 0**
beq (I)	X 1 (0 1)	X X X X X X	**sub**	**0 1 1 0**
add (32)	1 X (1 0)	X X 0 0 0 0	**add**	**0 0 1 0**
sub (34)	1 X (1 0)	X X 0 0 1 0	**sub**	**0 1 1 0**
and (36)	1 X (1 0)	X X 0 1 0 0	**and**	**0 0 0 0**
or (37)	1 X (1 0)	X X 0 1 0 1	**or**	**0 0 0 1**
slt (42)	1 X (1 0)	X X 1 0 1 0	**slt**	**0 1 1 1**

ALUOp does not use encoding 11, so 10 is replaced with 1X and 01 is replaced by X1. The first two fields of the function fields are always 10, so, they are replaced by XX.

Control: ALU Control Unit (4)

Instruction (funct)	Inputs: ALUOp (ALUOp1 – ALUOp0)	Inputs: Function Field (F5 – F0)	Desired ALU action	Outputs Operation (Op3 – Op0)
lw (I)	0 X (0 0)	X X X X X X	**add**	**0 0 1 0**
sw (I)	0 X (0 0)	X X X X X X	**add**	**0 0 1 0**
beq (I)	X 1 (0 1)	X X X X X X	**sub**	**0 1 1 0**
add (32)	1 X (1 0)	X X 0 0 0 0	**add**	**0 0 1 0**
sub (34)	1 X (1 0)	X X 0 0 1 0	**sub**	**0 1 1 0**
and (36)	1 X (1 0)	X X 0 1 0 0	**and**	**0 0 0 0**
or (37)	1 X (1 0)	X X 0 1 0 1	**or**	**0 0 0 1**
slt (42)	1 X (1 0)	X X 1 0 1 0	**slt**	**0 1 1 1**

Simplified Expressions:

$$Op0 = ALUOp1 \cdot (F0 + F3)$$
$$Op1 = ??$$
$$Op2 = ??$$