Datapath Components

CS240 Laboratory 9

Description of Mini-MIPS Architecture

16 bit data bus

8 bit address bus

Starting address of every program = 0 (PC initialized to 0 by a reset to begin execution)

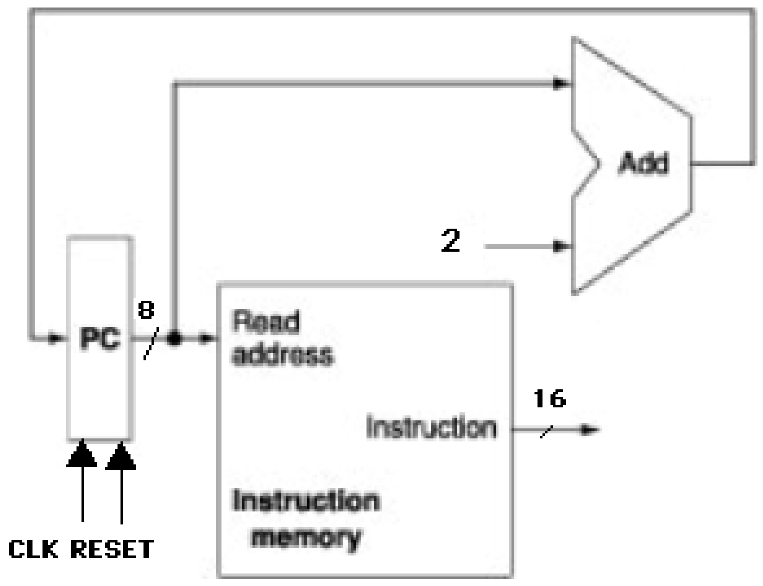
PC incremented by 2 to move to the next instruction.

16 registers
R0 = 0 (constant)
R1 =1 (constant)
R2-R15 general purpose

Instruction Set

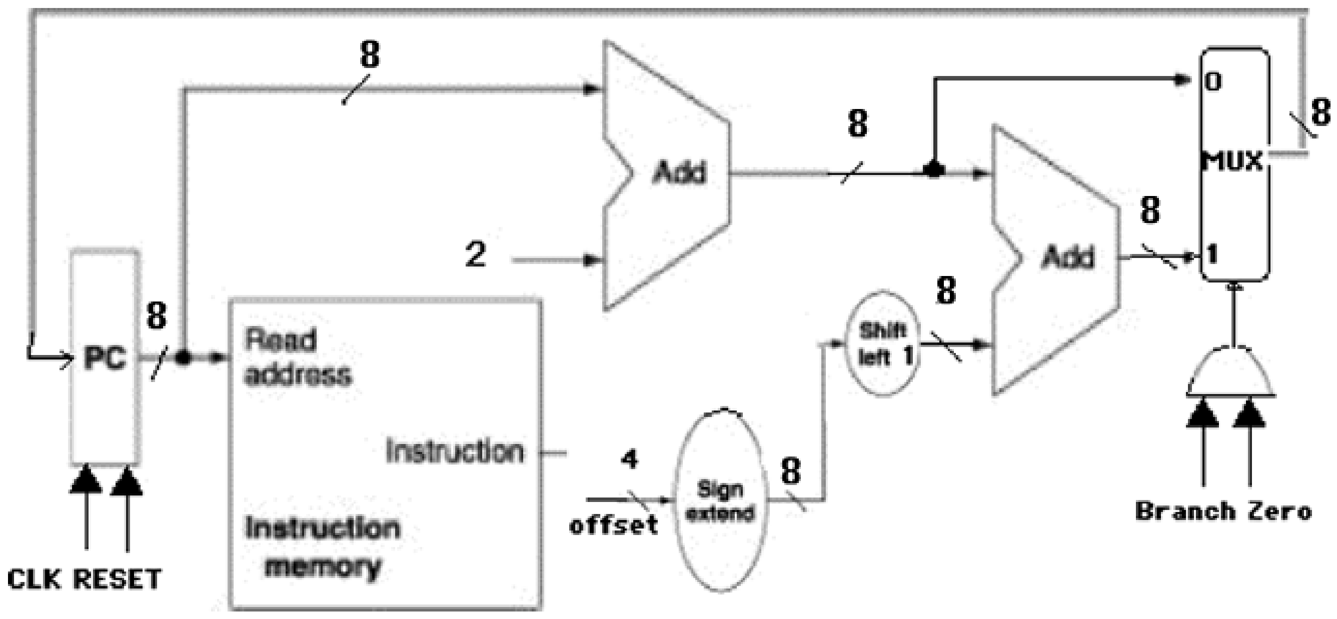
Instruction	**Meaning**	**Op** 4-bit	**Rs** 4-bit	**Rt** 4-bit	**Rd** 4-bit
LW Rt,offset(Rs)	Rt loaded with word from Data Memory at address(Rs + offset)	0000	0-15	0-15	offset
SW Rt,offset(Rs)	Data Memory address(Rs + offset) stored with word from Rt	0001	0-15	0-15	offset
ADD Rs,Rt,Rd	Rd := Rs + Rt	0010	0-15	0-15	0-15
SUB Rs,Rt,Rd	Rd := Rs - Rt	0011	0-15	0-15	0-15
AND Rs,Rt,Rd	Rd := Rs AND Rt	0100	0-15	0-15	0-15
OR Rs,Rt,Rd	Rd := Rs OR Rt	0101	0-15	0-15	0-15
SLT Rs,Rt,Rd	If Rs<Rt then Rd:=1 else Rd := 0	0110	0-15	0-15	0-15
BEQ Rs,Rt,offset	If Rs=Rt then pc:=pc+2+(offset*2) else pc:=pc+2	0111	0-15	0-15	offset
JMP offset	Jump to abs. addr = offset*2	1000	---12 bit offset-----		

Instruction Fetch



Branch Address

Either **PC + 2** or **PC + 2 + (2*offset)** is the next value of the PC.



On a BEQ instruction, **BEQ Rs,Rt,offset**

- The **offset** = number of instructions away from the next value of the PC to branch to, so must be multiplied by 2. This is done with a **shift-left** circuit.

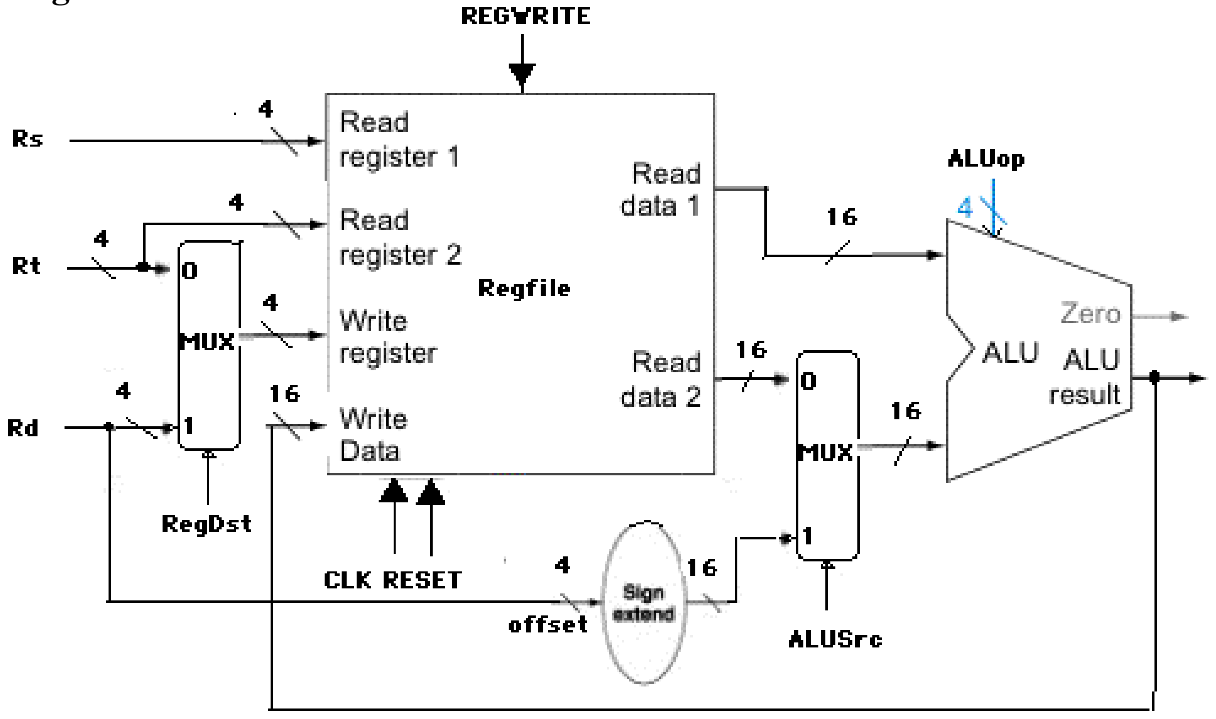
- Since **offset** is 4 bits, it must be sign-extended to 8 bits to be added to the PC. This is done with a **sign-extend** circuit.

A **2x8 multiplexer** circuit to selects the next value of the PC. The value of the **Branch** and **Zero** bits are used to determine which is used:

- The **Branch** control line = 1 if a BEQ instruction is being executed.

- The **Zero** bit from the ALU is used to check whether Rs = Rt: it is 1 if Rs – Rt = 0 (meaning they're equal). If **Branch** = 1 and **Zero** = 1, then the next value of the PC will be the branch address ; otherwise, it will simply be PC + 2:

Register File and ALU



R-type instructions ADD,SUB,AND,OR,SLT (opcode Rs Rt Rd)

- read Rs and Rt from register file
- perform an ALU operation on the contents of the registers
- write the result to register Rd in register file

Memory Access instructions LW,SW (opcode Rs Rt offset)

- memory address = Rs + sign-extended 4-bit offset
- if **SW**, the value to be stored to memory is from Rt.
- if **LW**, Rt is loaded with the value read from memory

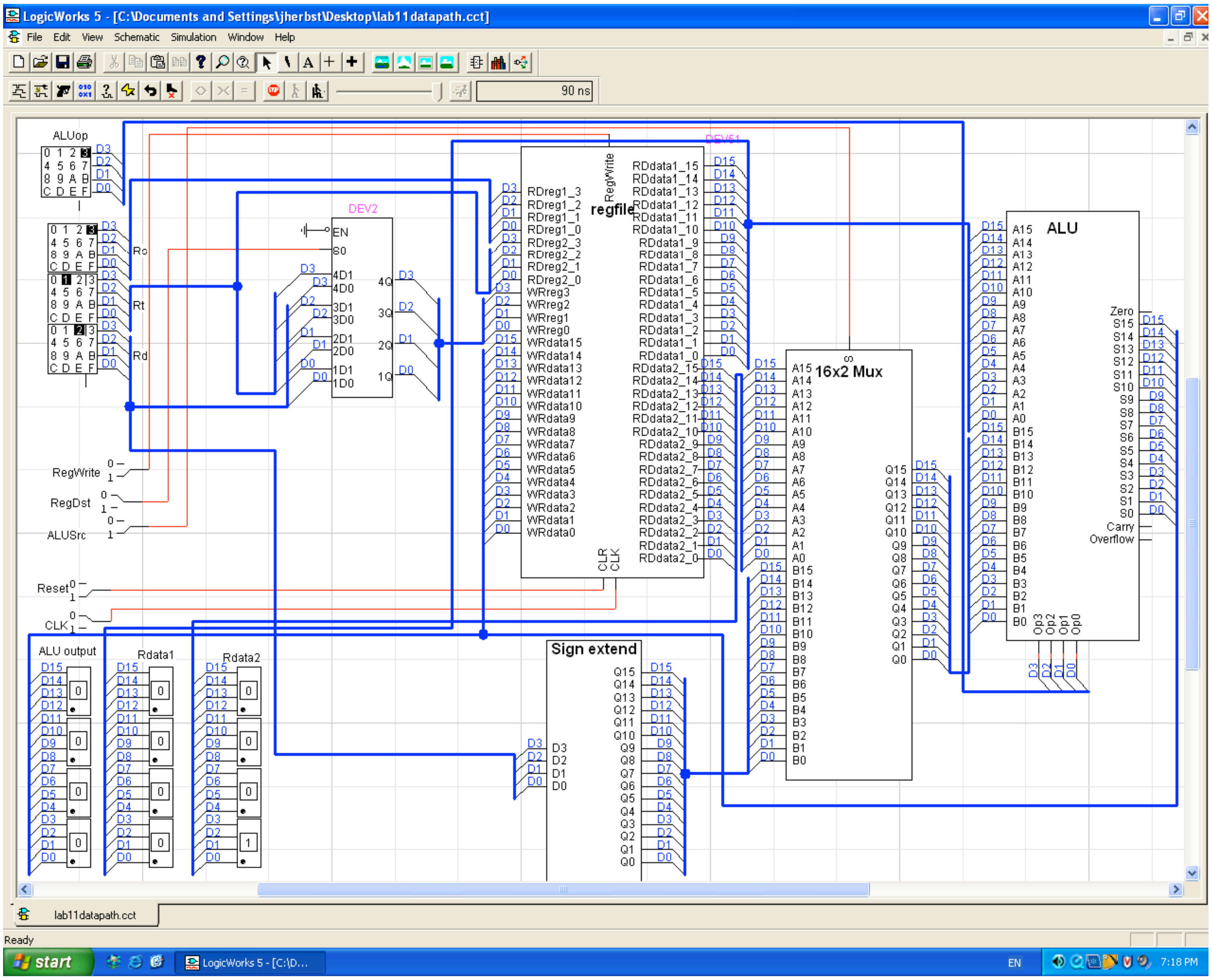
Register written to (**Write Register**) is Rd **or** Rt if a **LW** instruction (chosen by a 2x4 MUX which is controlled by **RegDst**)

ALU calculates Rs + Rt, **or** Rs + sign-extended offset.

- Input A of the ALU is always Rs

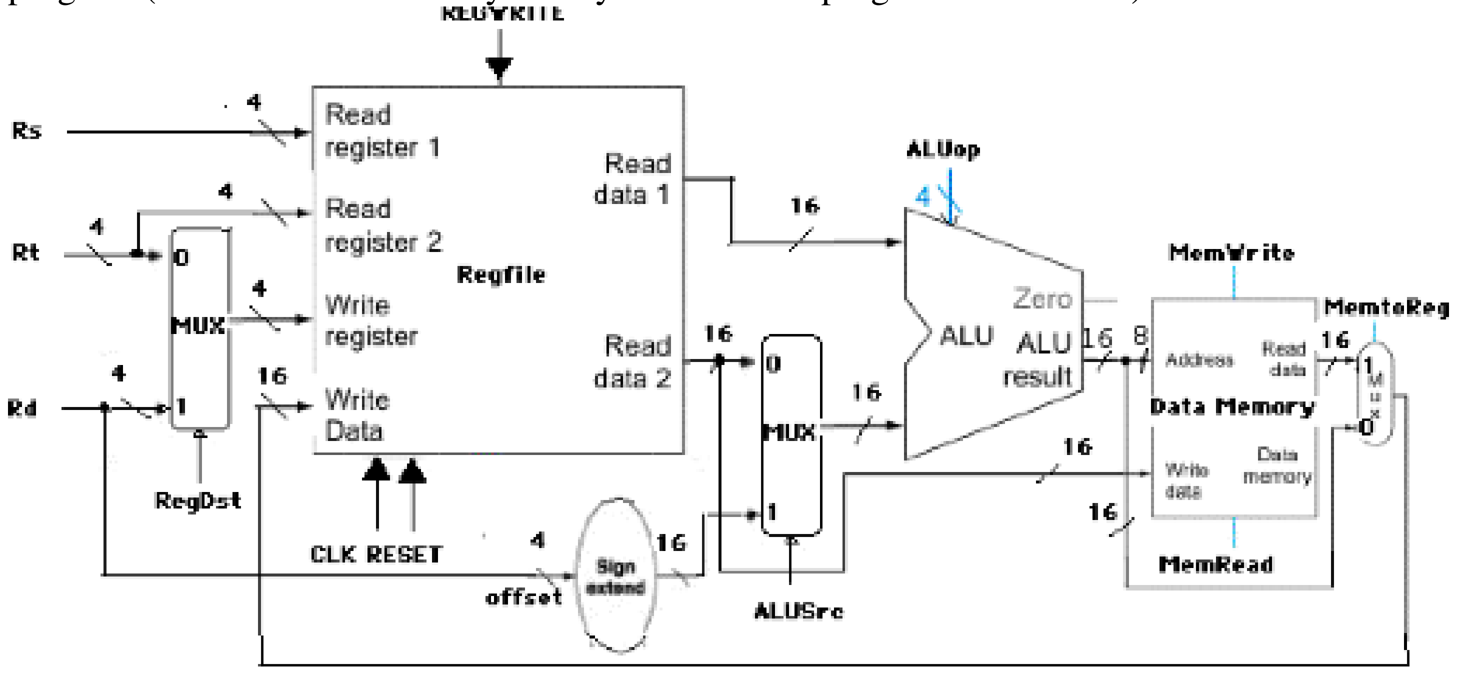
- Input B of the ALU is Rt **or** the offset (chosen by a 2x16 multiplexer, which is controlled by **ALUSrc**):

LogicWorks version of the circuit:



Data Memory

We need an additional memory for values loaded or stored (**LW** or **SW**) during execution of the program (the instruction memory is only used to store program instructions).



RegDst (chooses whether Rd or Rt goes to the **Write data** input on the Regfile)
If 0, destination is Rd. If 1, destination is Rt.

RegWr (control line to RegFile)
If 1, writes the value on the **Write data** input to the register specified by **Write register**

ALUSrc (chooses the source of the second ALU operand)
If 0, the operand is the second register file output.
If 1, the operand is the sign-extended, lowest 4 bits of the instruction.

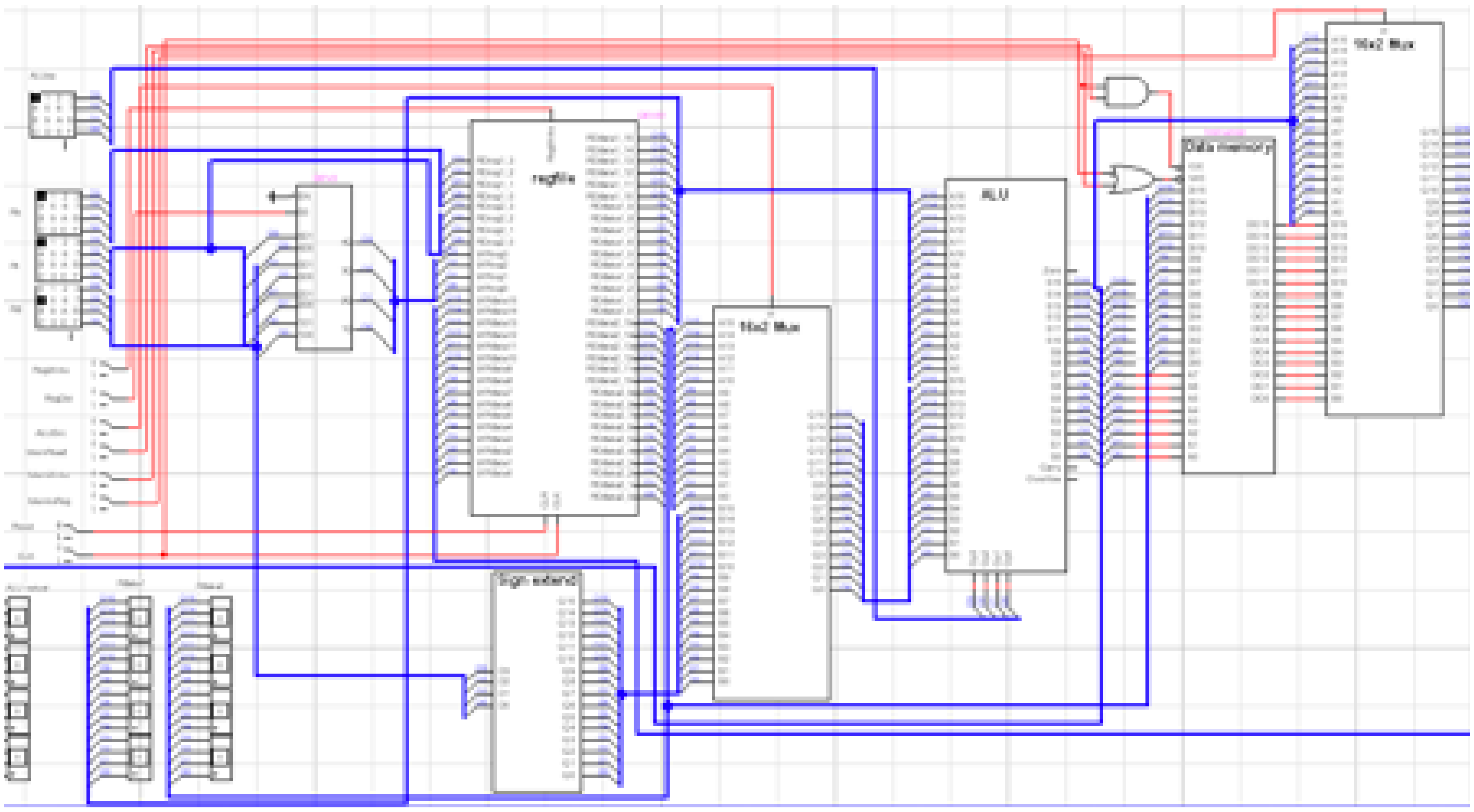
MemRd (control signal to data memory)
If 0, value stored at address in data memory is read from **Read data**.

MemWr (control signal to data memory)
If 0, data memory address written with value from **Rt** on the **Write data** input.

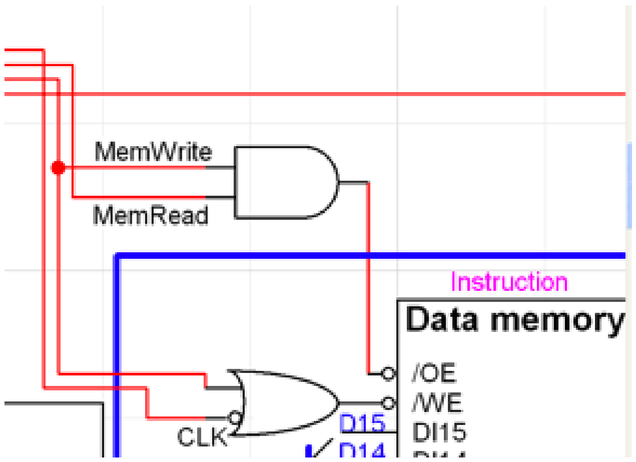
MemtoReg (chooses the value to be written back to the Regfile)
If 0, the value comes from the ALU (**R-type instruction**)
If 1, the value comes from data memory (**LW**)

ALUop(**4 bits**)	ALU function
0	a AND b
1	a OR b
2	a + b (add)
6	a - b
7	set on less than

LogicWorks version of the full DataPath



Magnified view of the logic for connecting the enable lines on the Data Memory:



Examples of Tests

In the lab, you should analyze your results with the following type of explanation:

LUop	Rs	Rt	Rd	RegWrite	RegDst	ALUSrc	MemRead	MemWrite	MemtoReg	ALU result
2	1	1	5	1	0	0	1	1	0	**2**

ALUop = 2, add

RegDst = 0, use Rt as second operand, so R1 + R1 = 2, ALU result = 2

MemRead = 1, MemWrite (not reading or writing to memory, so it is an ADD instruction)

MemToReg = 0, send ALU result back to RegFile

RegWrite = 1, so

R5 gets 2

LUop	Rs	Rt	Rd	RegWrite	RegDst	ALUSrc	MemRead	MemWrite	MemtoReg	ALU result
2	0	5	0	0	1	1	1	0	0	**0**

Assume this is a SW instruction. What tells you this?

MemWrite = 0 means data memory gets written (which is a SW)

ALUop = 2, add operation, used to calculate the address to write to

ALUSrc = 1, take offset from Rd and add to base address in Rs to get target address = 0, applied to address of data memory

RegWrite = 0, which means a register does not get a new value (it goes to data memory, instead)

Rt (R5) is connected to the data inputs on the data memory, so when the MemWrite occurs,

address 0 gets contents of R5 = 2

LUop	Rs	Rt	Rd	RegWrite	RegDst	ALUSrc	MemRead	MemWrite	MemtoReg	ALU result
2	0	3	0	1	1	1	0	1	1	**0**

This time, assume a LW instruction. What tells you this?

MemRead= 0 means data memory is read (which is a LW)

Explain the operation and result: what value is loaded from what address into what register?

ALUop = 2, add operation, used to calculate the address to read from

ALUSrc = 1, take offset from Rd and add to base address in RS to get target address = 0

RegDst = 1, register written to will be Rt, or R3

RegWrite = 1, R3 will get contents of address 0 , which is 2

R3 gets 2