CS305
Computer Architecture

Single Cycle Implementation of MIPS ISA (Subset)

Bhaskaran Raman
Room 406, KR Building
Department of CSE, IIT Bombay

http://www.cse.iitb.ac.in/~br

MIPS ISA Subset

- Reg-reg: **`add, sub, and, or, slt`**
- Memory: **`lw, sw`**
- Branch: **`beq`** (and **`j`** later)
- Subset ➔ keep it simple, understand techniques

Before Proceeding, Test Your Understanding

- What kind of arguments does **`add`** take ?
- How many registers need to be specified in **`lw`** ?
- How many registers need to be specified in **`sw`** ?
- What is the least integer value of offset in **`lw`** ?
- What is the largest integer value of offset in **`sw`** ?
- What instruction format is used by **`beq`** ?
- What is the role of the immediate operand in **`beq`** ?

Recall: MIPS Instruction Format

opcode (6)	rs (5)	rt (5)	rd (5)	shamt (5)	funct (6)

R-type instruction: register-register operations

opcode (6)	rs (5)	rt (5)	immediate/constant or offset (16)

I-type instruction: loads, stores, all immediates, *conditional branch, jump register, jump and link register*

opcode (6)	offset relative to PC (26)

J-type instruction: *jump, jump and link, trap and return*

Steps in Program Execution

✓ (1) Fetch

Execution

add, sub, and, or, slt

lw, sw

beq

✓ (2a) read register file

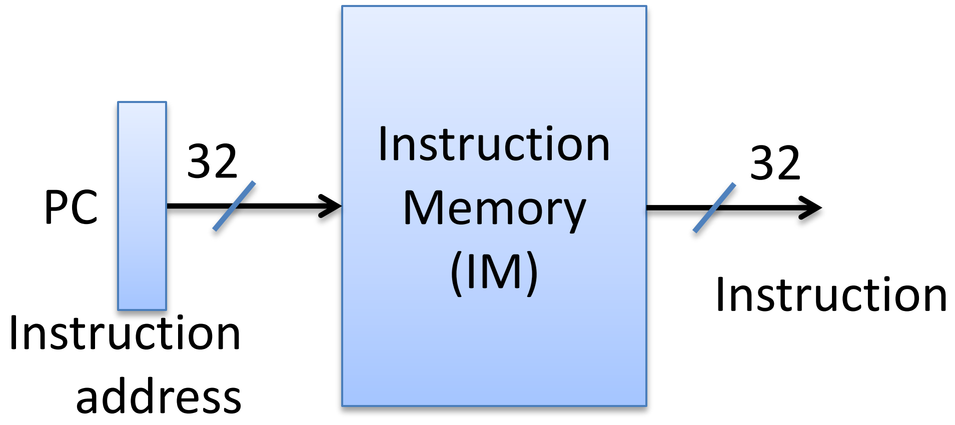
(2b) Specific to instruction

✓ (2c) $PC = \underbrace{PC+4}$

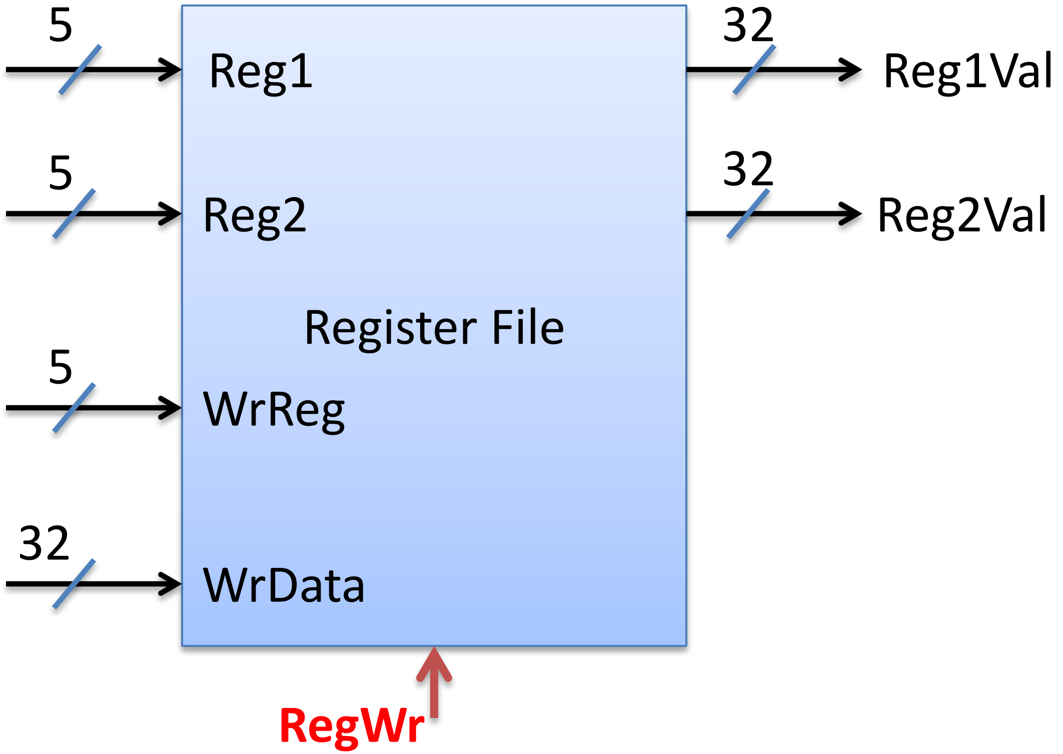
Hardware components

- put them together

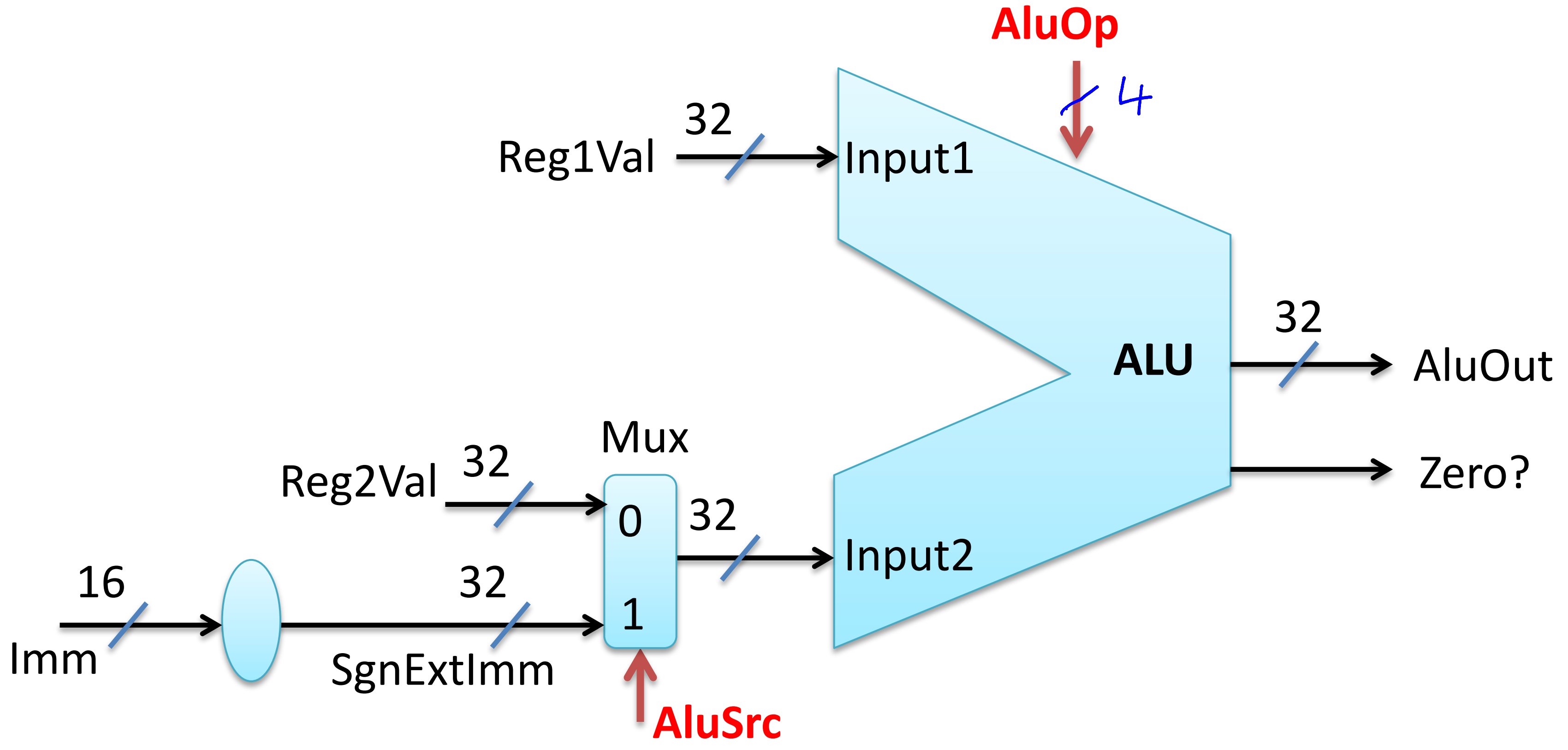
Elements for Instruction Fetch



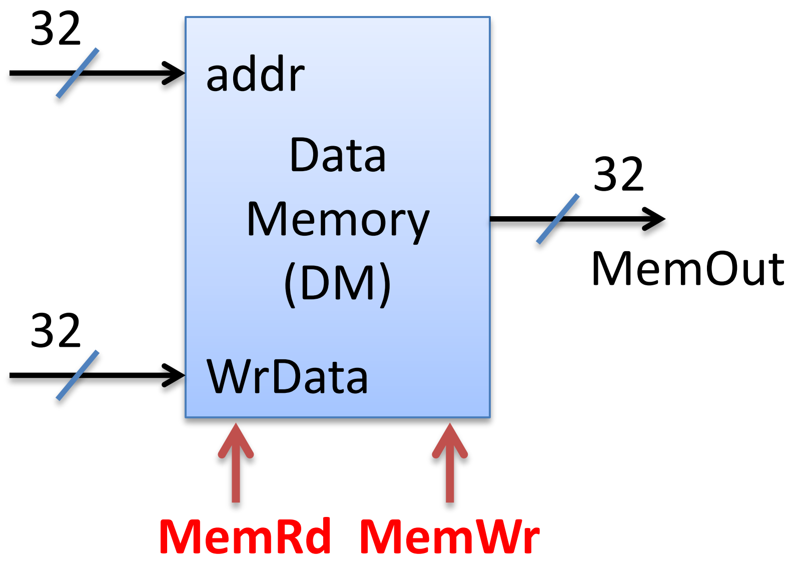
Element for Register Read/Write: The Register File



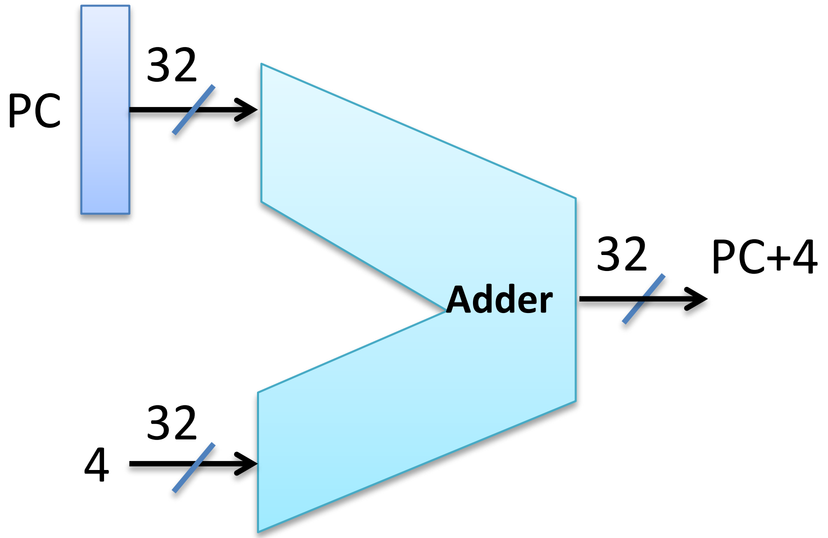
ALU: Arithmetic Logic Unit



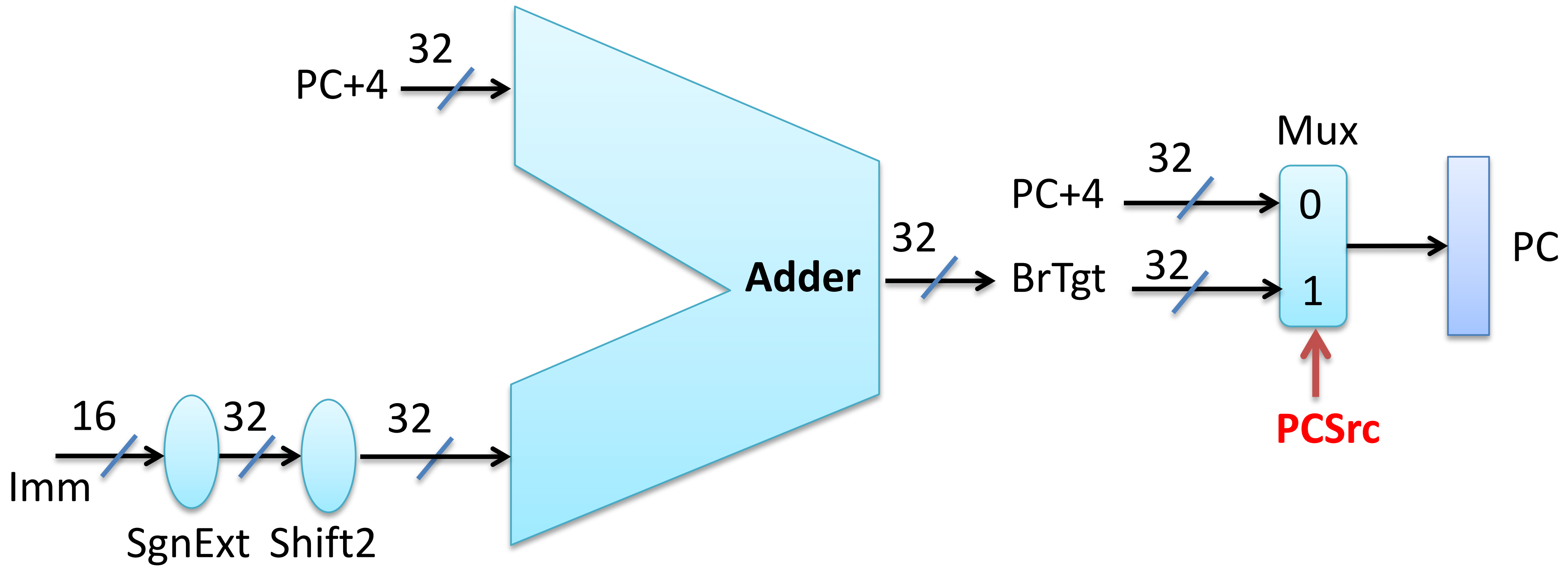
Data Memory



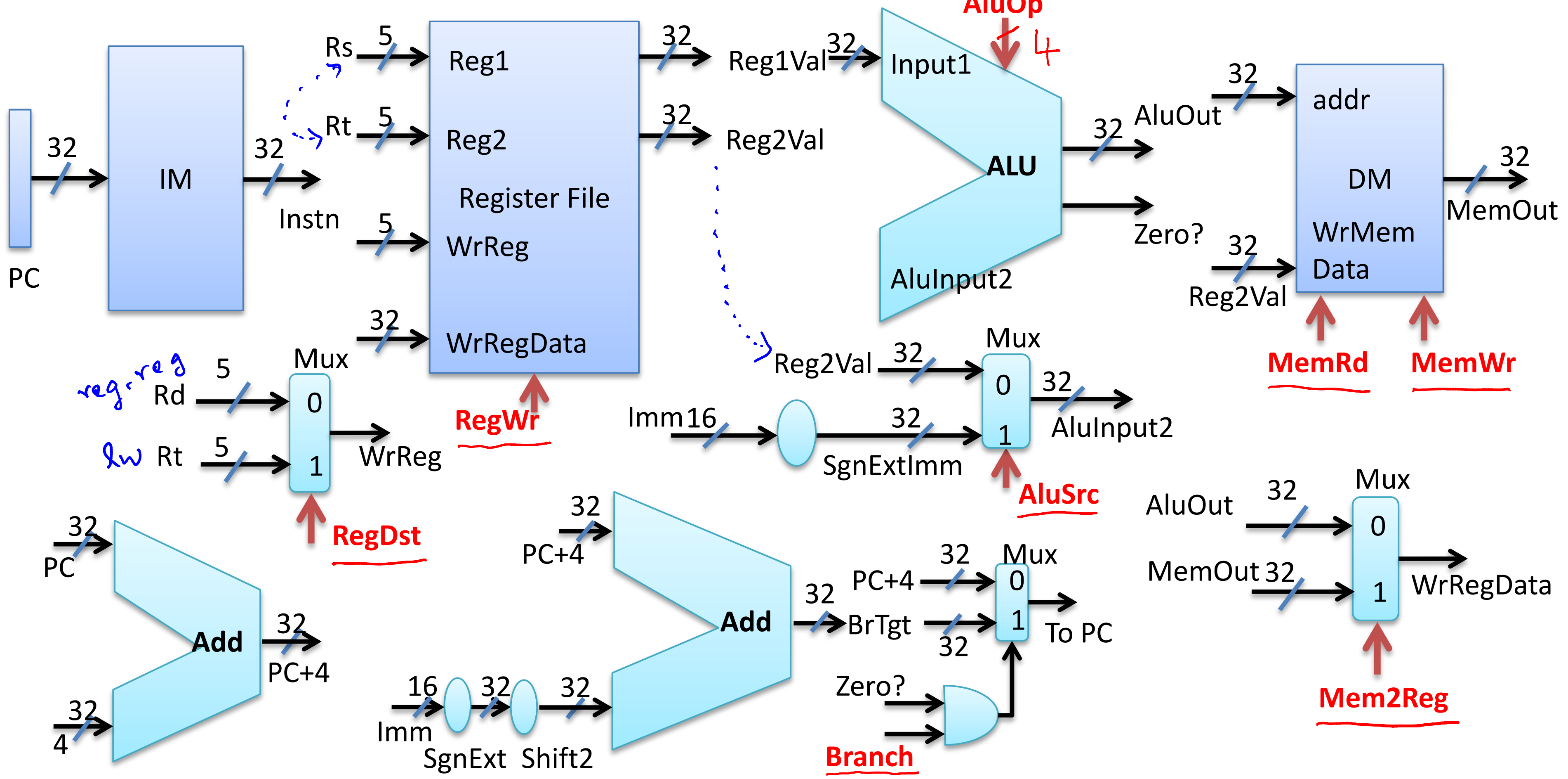
Element for PC+4 Computation



Additional Elements to Implement beq



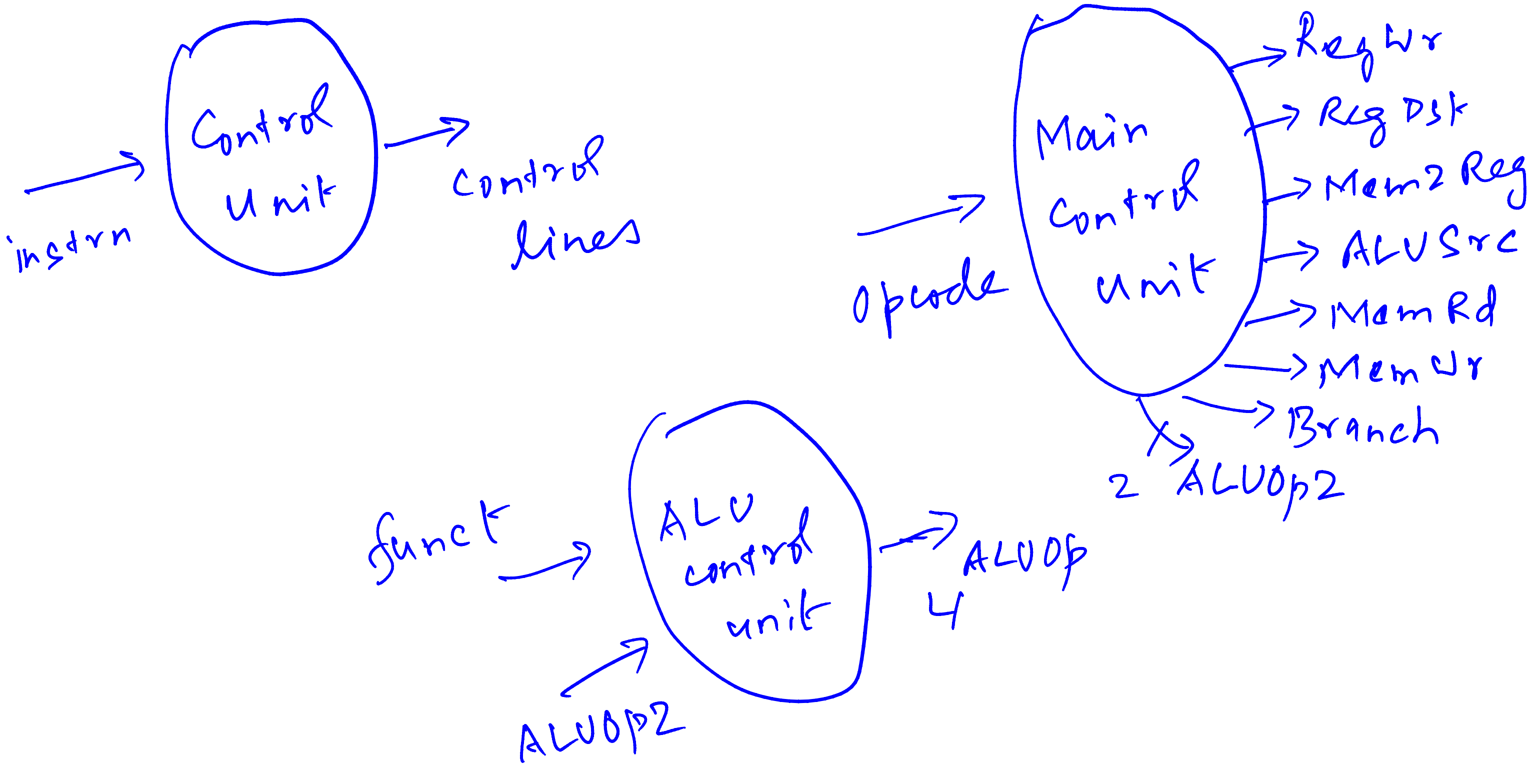
Putting it All Together



Summary of Control Lines

- RegDst (1): to decide Rd vs Rt
- RegWr (1): should register file be written?
- ALUSrc (1): to decide Rt vs SignExtImm
- MemRead (1): should data memory be read?
- MemWrite (1): should data memory be written?
- Mem2Reg (1): to decide ALUOut vs MemOut
- Branch (1): is this a **`beq`** instruction?
- AluOp (4): which ALU operation to perform

Main Control Unit, ALU Control Unit



Truth Table for Main Control Unit

	RegDst	RegWr	ALUSrc	MemRd	MemWr	Mem2Reg	Branch	ALUOp2
Reg-Reg	Rd (0)	1	Reg2Val (0)	0	0	ALUOut (0)	0	10
lw	Rt (1)	1	SgnExt-Imm (1)	1	0	MemOut (1)	0	00
sw	x	0	SgnExt-Imm (1)	0	1	x	0	00
beq	x	0	Reg2Val (0)	0	0	x	1	01

- Can produce optimized combinational circuit to implement truth table
- Similar truth table for ALU control unit as well
- Q: why is MemRd always explicitly enabled or disabled?

Summary

- Single cycle implementation of MIPS ISA subset
  - Sequential, combinational components for different instructions
  - Put together in a datapath
  - Control lines define the control path
  - Control lines generated from opcode + funccode
- Next: extending the implementation to support other instructions