CSSE232
Computer Architecture I

Datapath

Class Status

- Reading
  - Sections 4.1-3

- Project
  - Project group milestone assigned
  - Indicate who you want to work with
  - Indicate who you don't want to work with
  - Due next Friday (before exam)

Review so far

- Performance
- Instruction Sets (ISAs)
- MIPS assembly
  - Register convention
  - Procedure calls
  - Alignment
  - Exceptions

Outline

- Today, we begin hardware
  - Datapath

- Stages of execution
- CPU overview
- Building a datapath

Introduction

- CPU performance factors
  - Instruction count
    - Determined by ISA and compiler
  - CPI and Cycle time
    - Determined by CPU hardware
- We will examine three MIPS datapaths
  - A simplified version
  - An improved split design
  - A more realistic pipelined version

Different Subsets of Instructions

- Memory reference: `lw`, `sw`
- Arithmetic/logical: `add`, `sub`, `and`, `or`, `slt`
- Control transfer: `beq`, `j`

- Instructions executed in multiple steps
  - Some steps are common to all instructions
  - Some steps shared between subsets
  - Some steps unique to single instruction

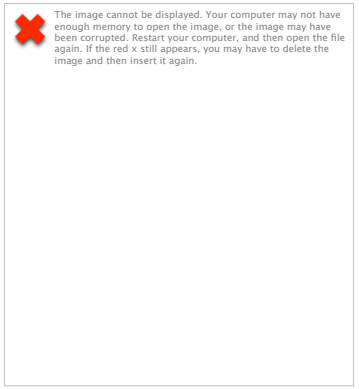
Five Stages of Instruction Execution

1. Instruction fetch
2. Instruction decode
3. Execute
4. Memory access
5. Write back

We will formalize hardware for these stages after Winter break. Until then, we'll build parts as needed.

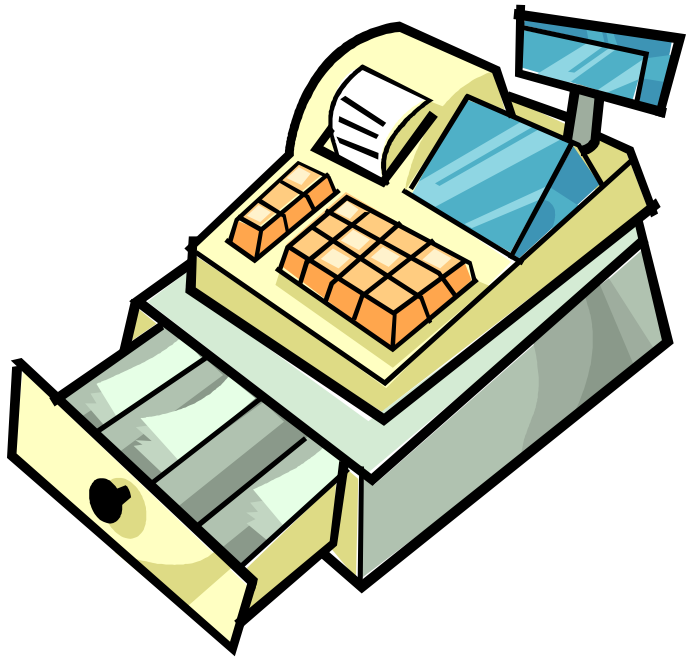
Instruction Fetch

- Fetch the instruction
  - based on value stored in PC
- Prepare for next instruction
  - Increment the PC



Instruction Decode

- Read registers from register file
- Calculate branch address



Execute

- Depending on instruction class
- Use ALU to calculate
  - Arithmetic result (R-type)
  - Memory address (lw,sw)
  - Branch compare (beq, bne)

- PC ← target address (faster branches)

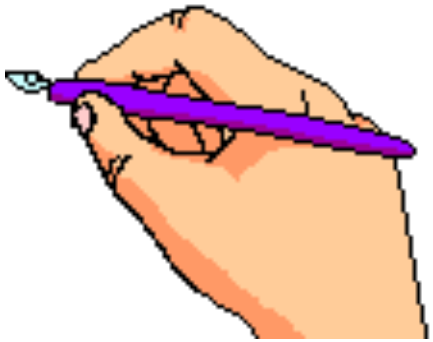
Memory

- Store data into memory (sw)
- Read data from memory (lw)

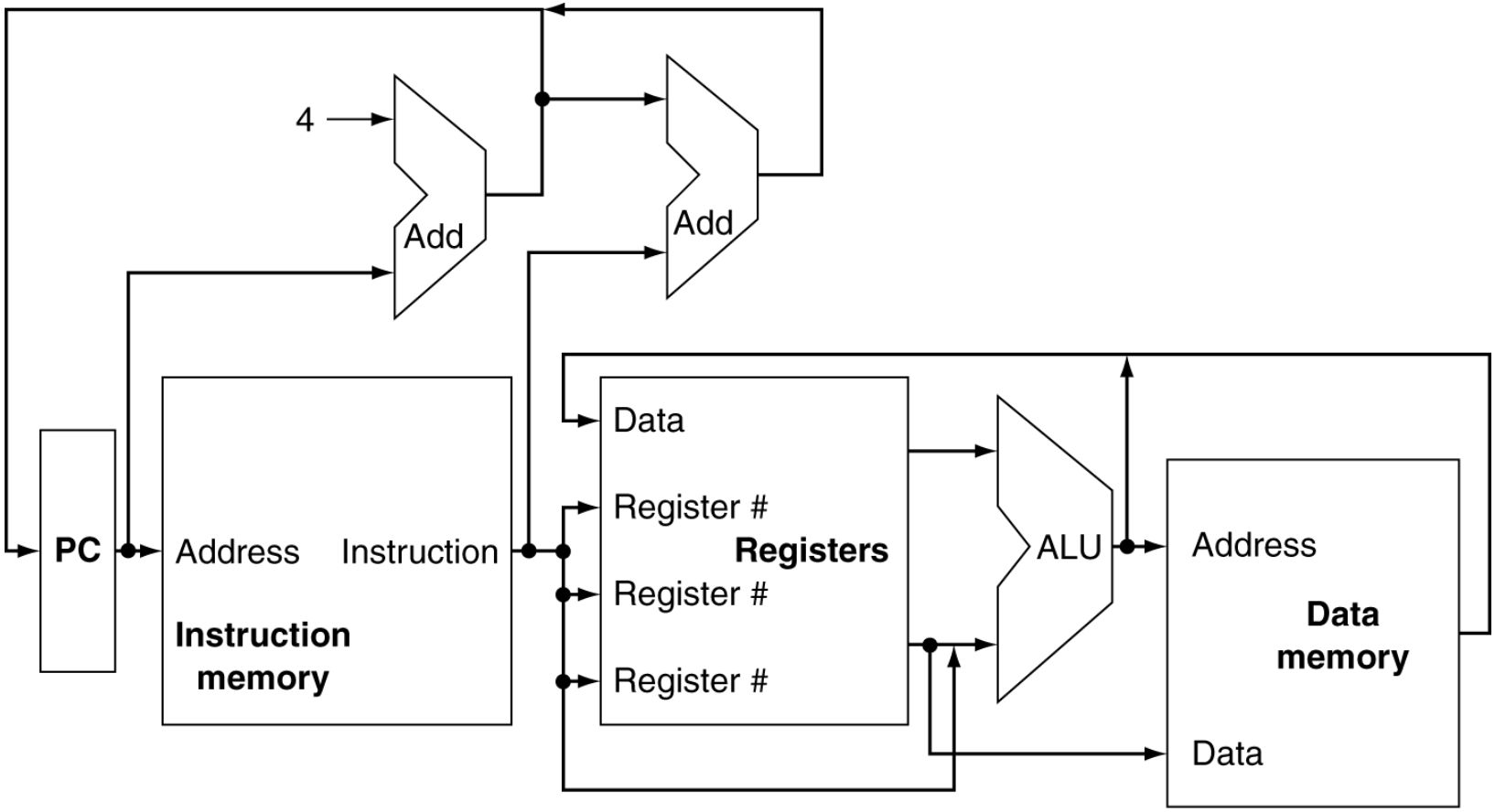
- Other things happen here too
  - Save results from R-type instructions

Write Back

- Store the data from memory into registers (lw)
  - Similar to saving R-type results

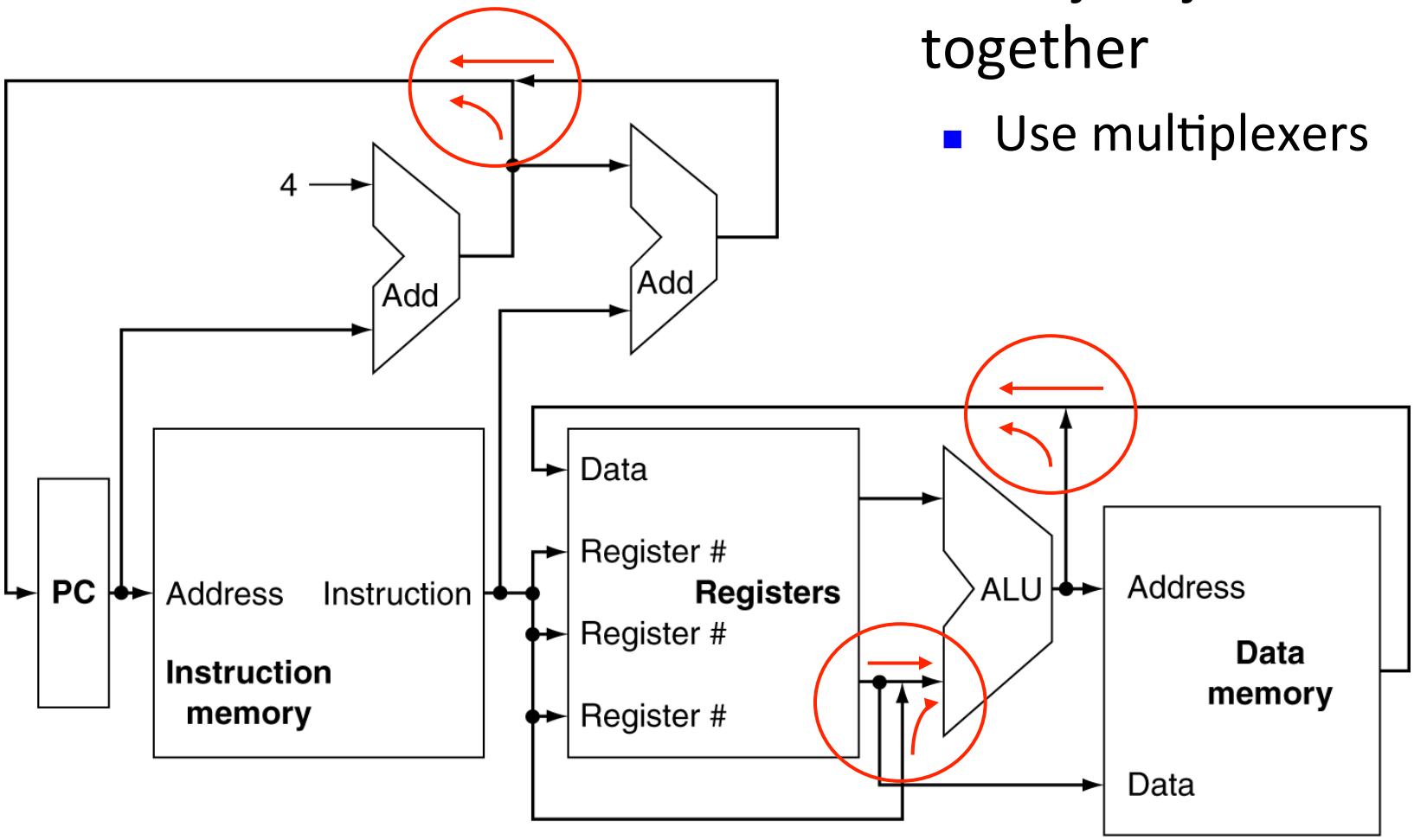


Datapath Overview

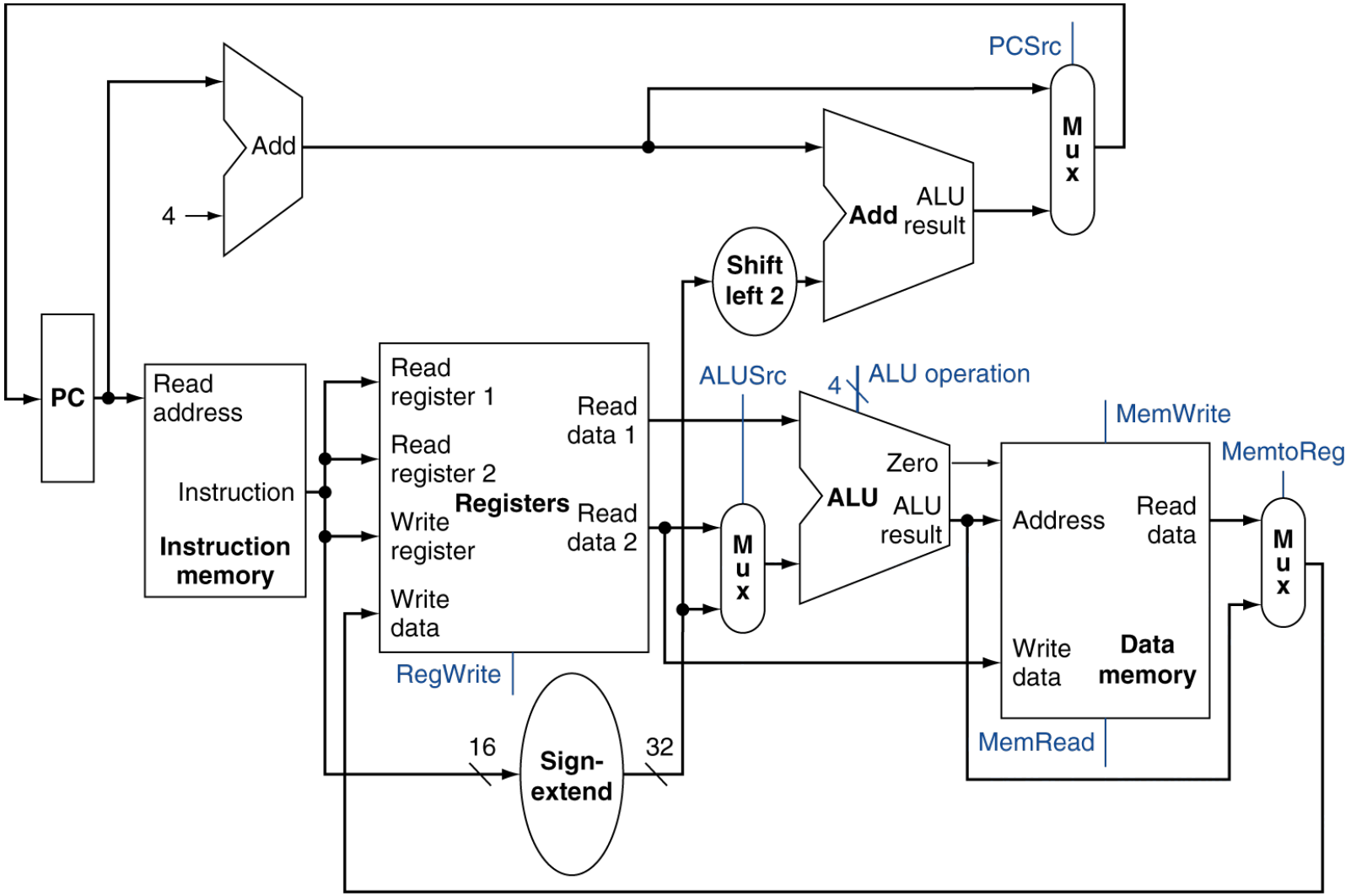


Multiplexers

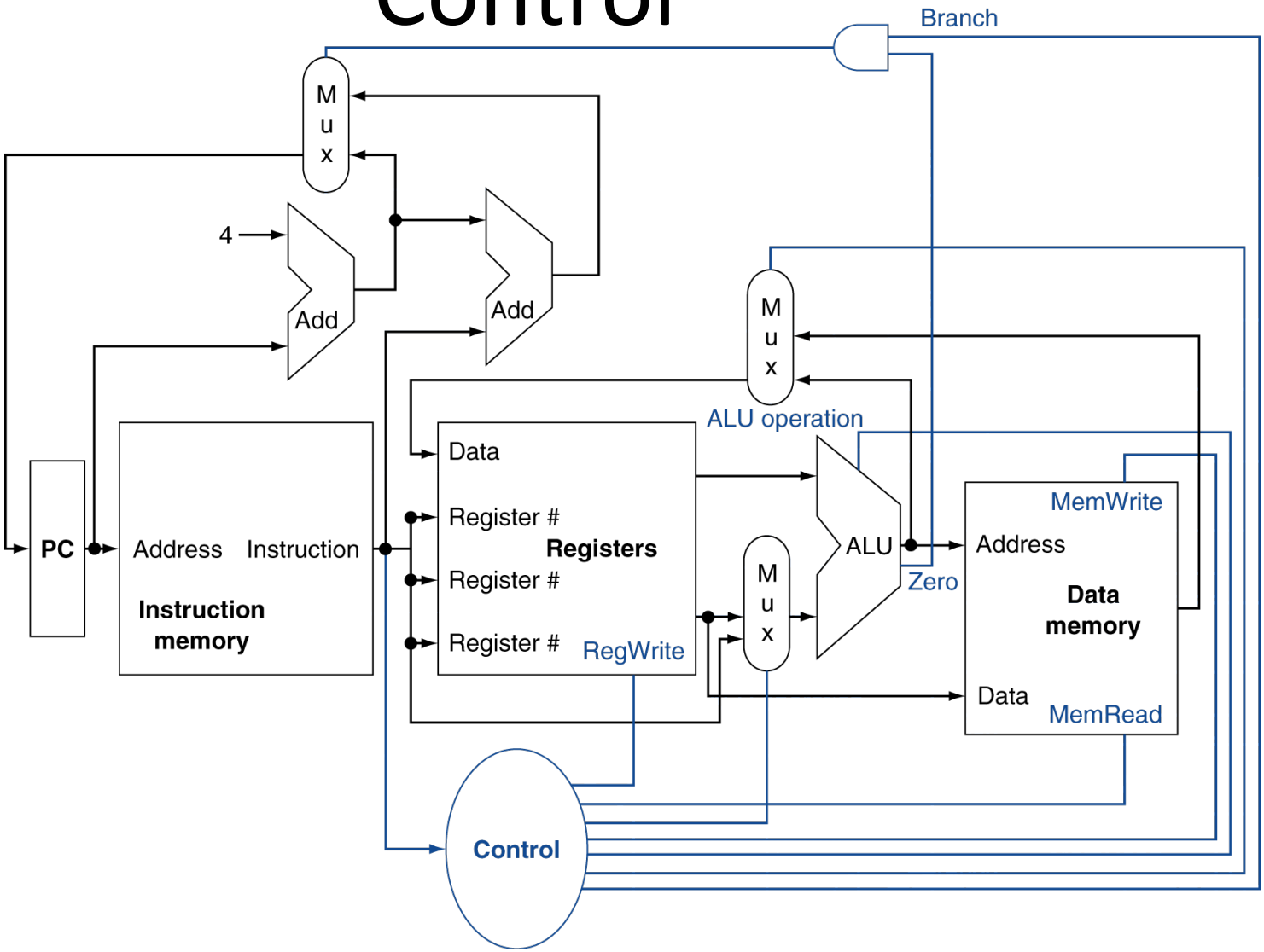
- Can't just join wires together
  - Use multiplexers



Multiplexers



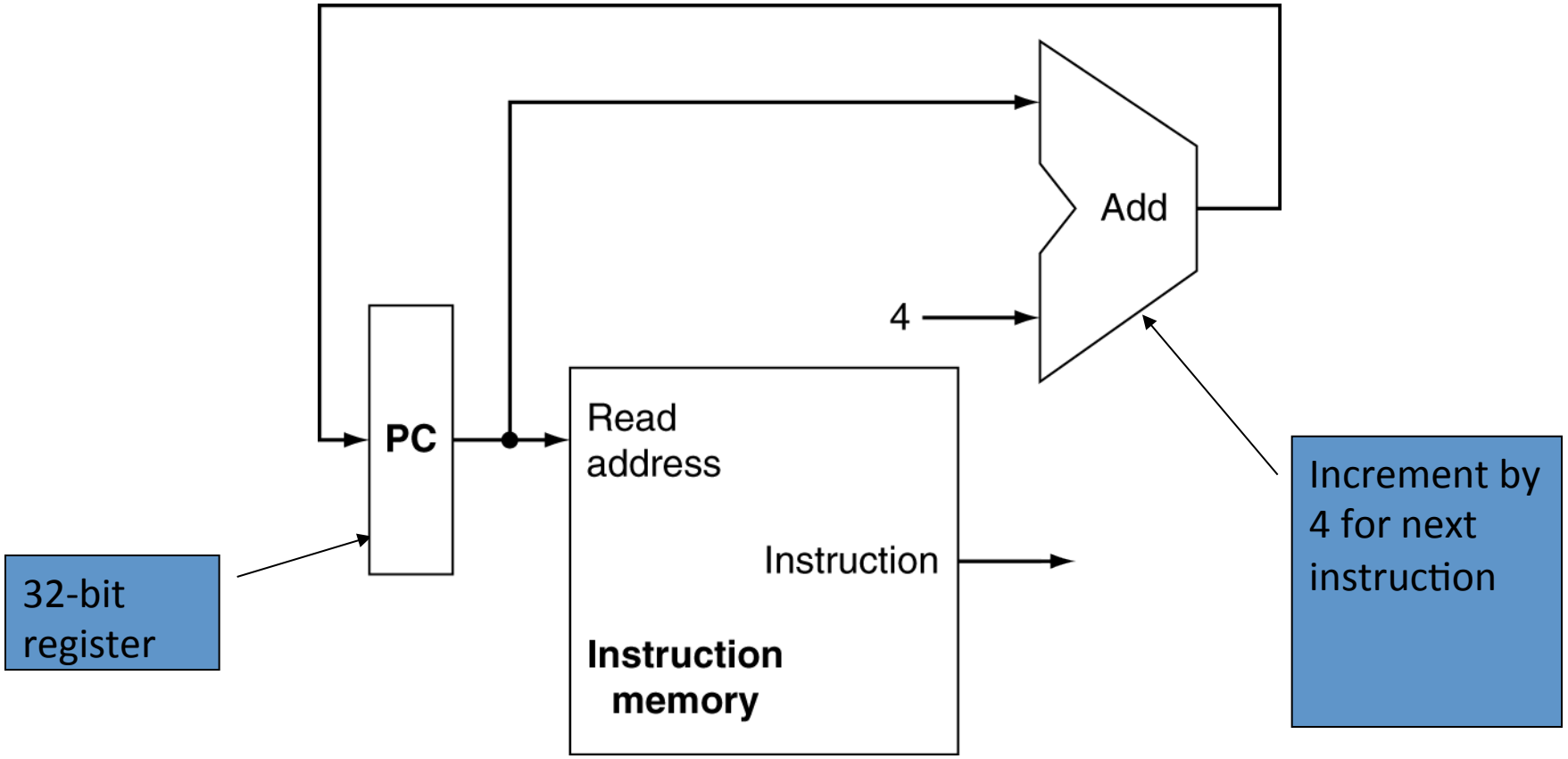
Control


Branch
M
u
x
4
Add
Add
M
u
x
ALU operation
PC
Address
Instruction
Instruction
memory
Data
Register #
Registers
Register #
Register #
RegWrite
M
u
x
ALU
Zero
MemWrite
Address
Data
memory
Data
MemRead
Control

Building a Datapath

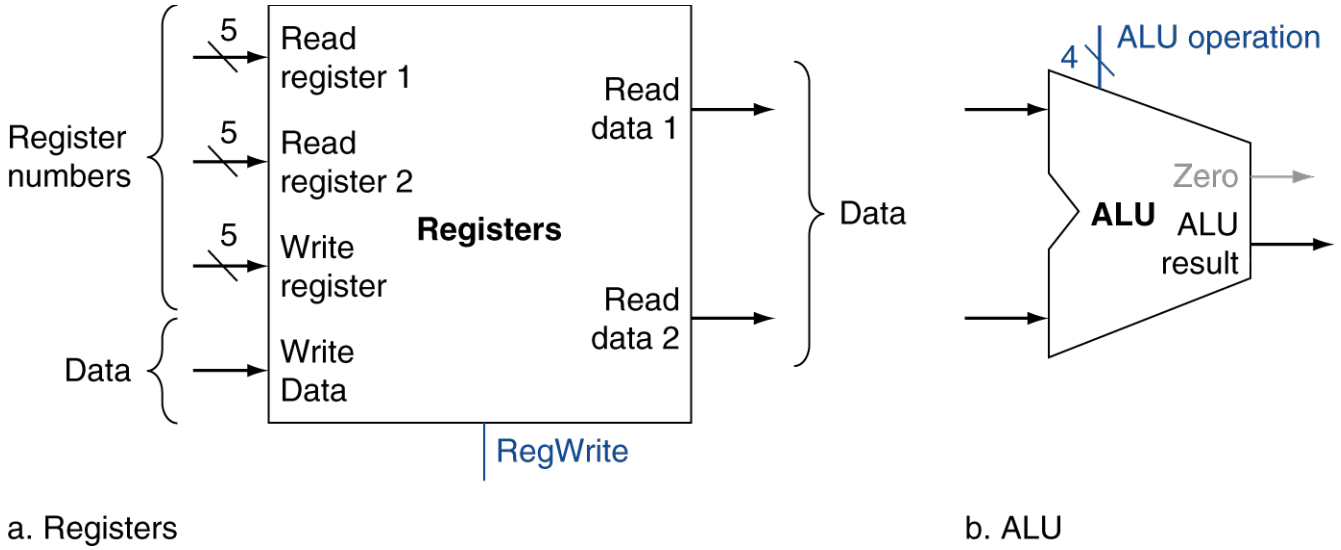
- Datapath
  - Elements that process data and addresses in the CPU
    - Registers, ALUs, mux's, memories, …
- We will build a MIPS datapath incrementally
  - Refining the overview design
  - Consider how you will build your datapath!

Instruction Fetch



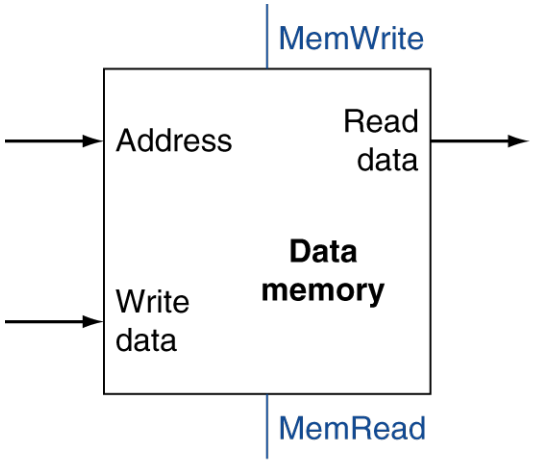
R-Format Instructions

- Read two register operands
- Perform arithmetic/logical operation
- Write register result

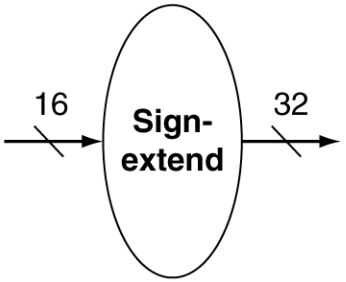


Load/Store Instructions

- Read register operands
- Calculate address using 16-bit offset
  - Use ALU, but sign-extend offset
- Load: Read memory and update register
- Store: Write register value to memory



a. Data memory unit

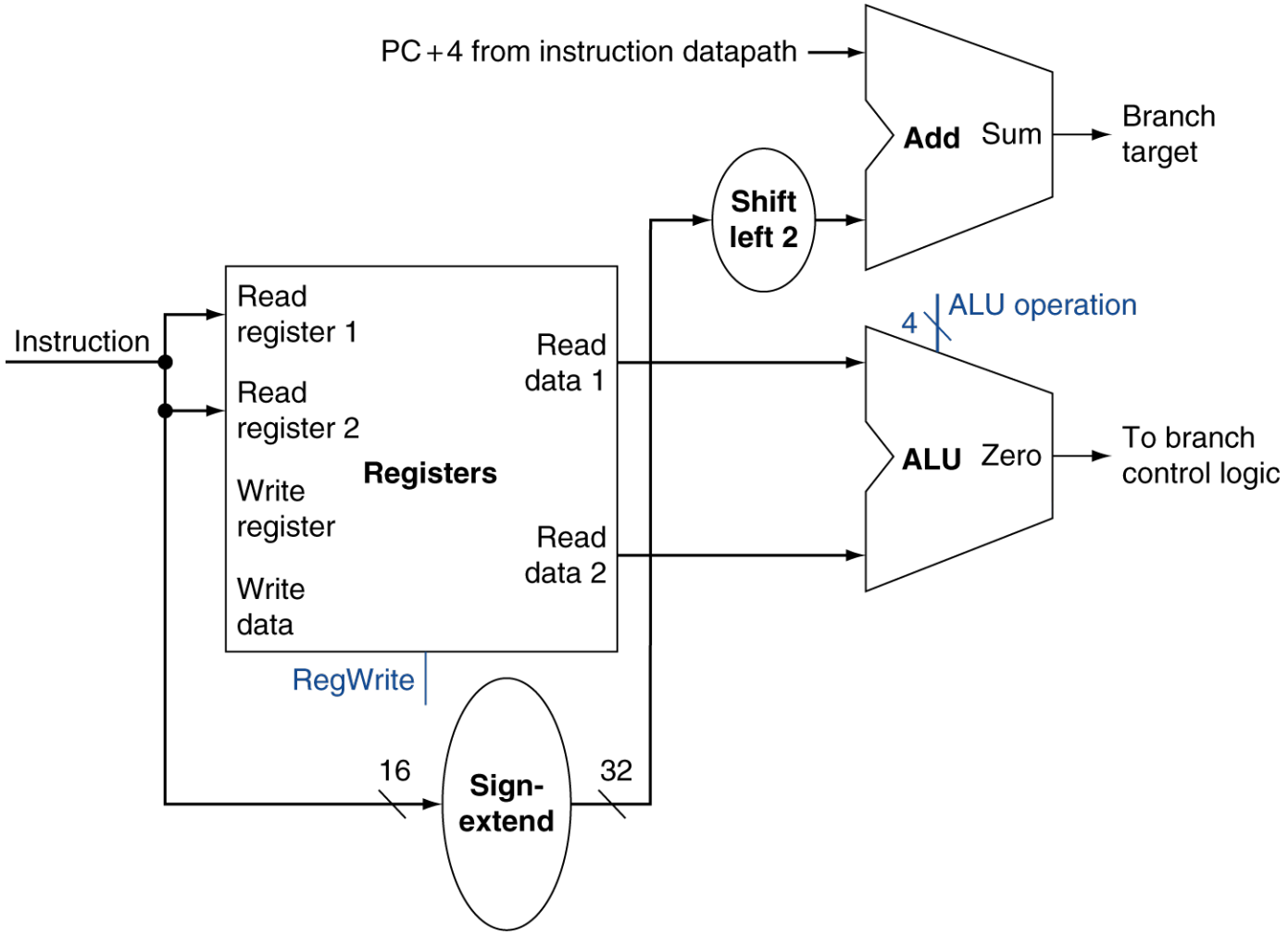


b. Sign extension unit

Branch Instructions

- Read register operands
- Compare operands
  - Use ALU, subtract and check Zero output
- Calculate target address
  - Sign-extend displacement
  - Shift left 2 places (word displacement)
  - Add to PC + 4
    - Already calculated by instruction fetch

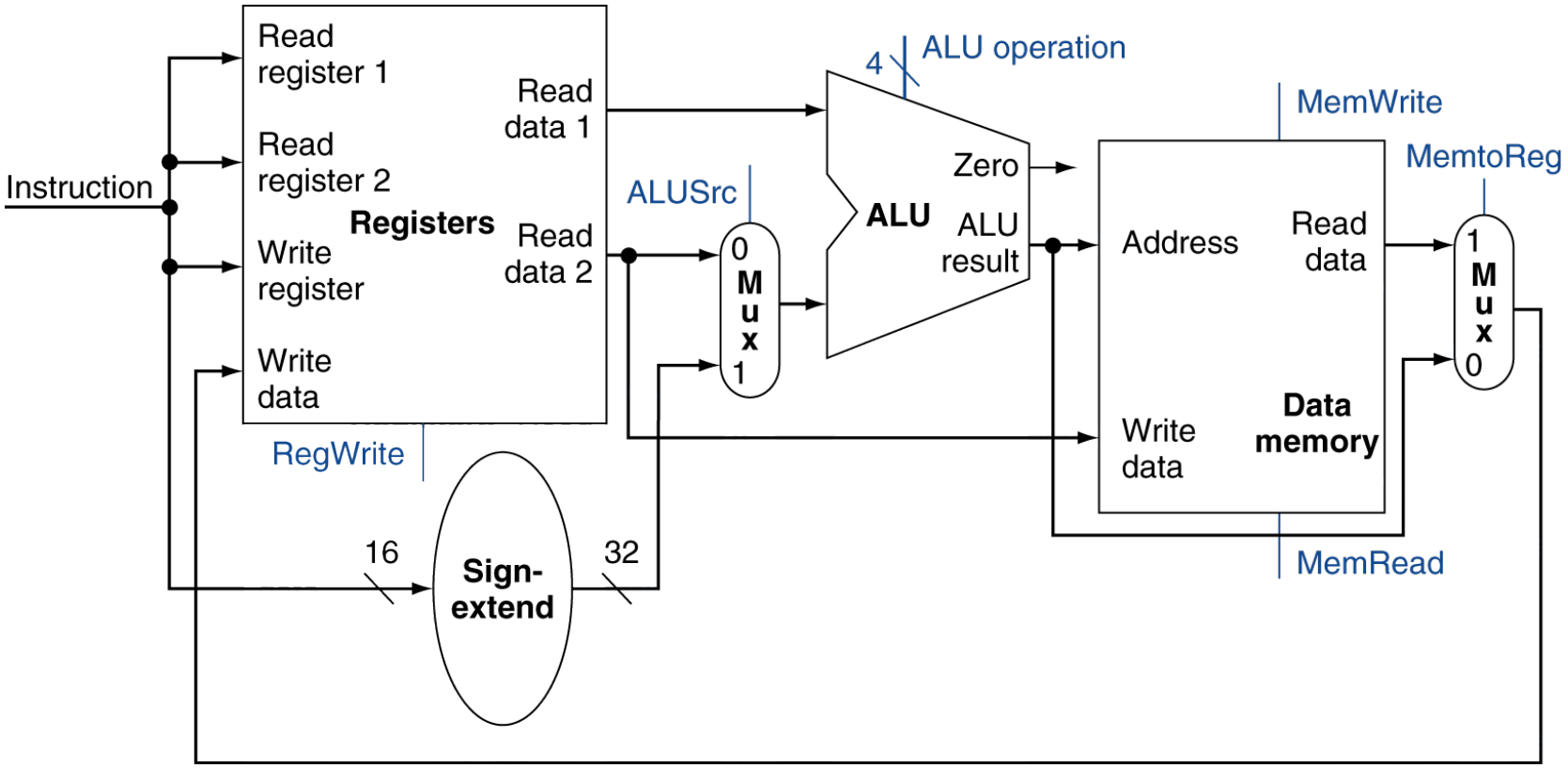
Branch Instructions



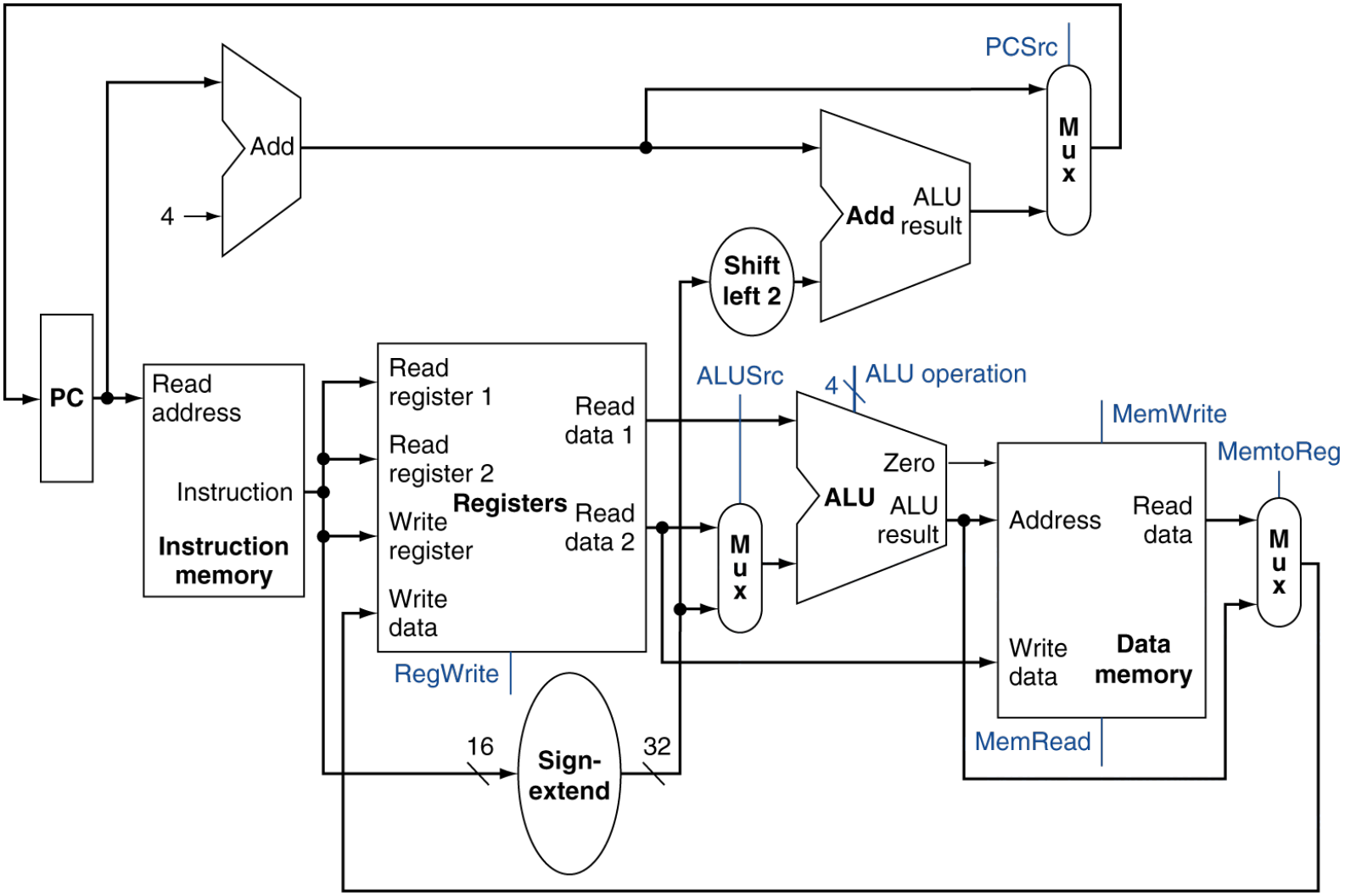
Composing the Elements

- First datapath does one instruction in one clock cycle
  - Each datapath element can only do one function at a time
  - Hence, we need separate instruction and data memories
- Use multiplexers where alternate data sources are used for different instructions

R-Type/Load/Store Datapath



Full Datapath



Performance Issues

- Longest delay determines clock period
  - Critical path: load instruction
  - InstMem→ RegFile→ ALU→ DataMem→ RegFile
- Not feasible to vary period for different instructions
- Violates design principle
  - Making the common case fast
- We will improve performance in next version

Review and Questions

- Stages of execution
- CPU overview
- Building a datapath

Different Subsets of Instructions

- Memory reference: `lw`, `sw`
- Arithmetic/logical: `add`, `sub`, `and`, `or`, `slt`
- Control transfer: `beq`

Two programs…

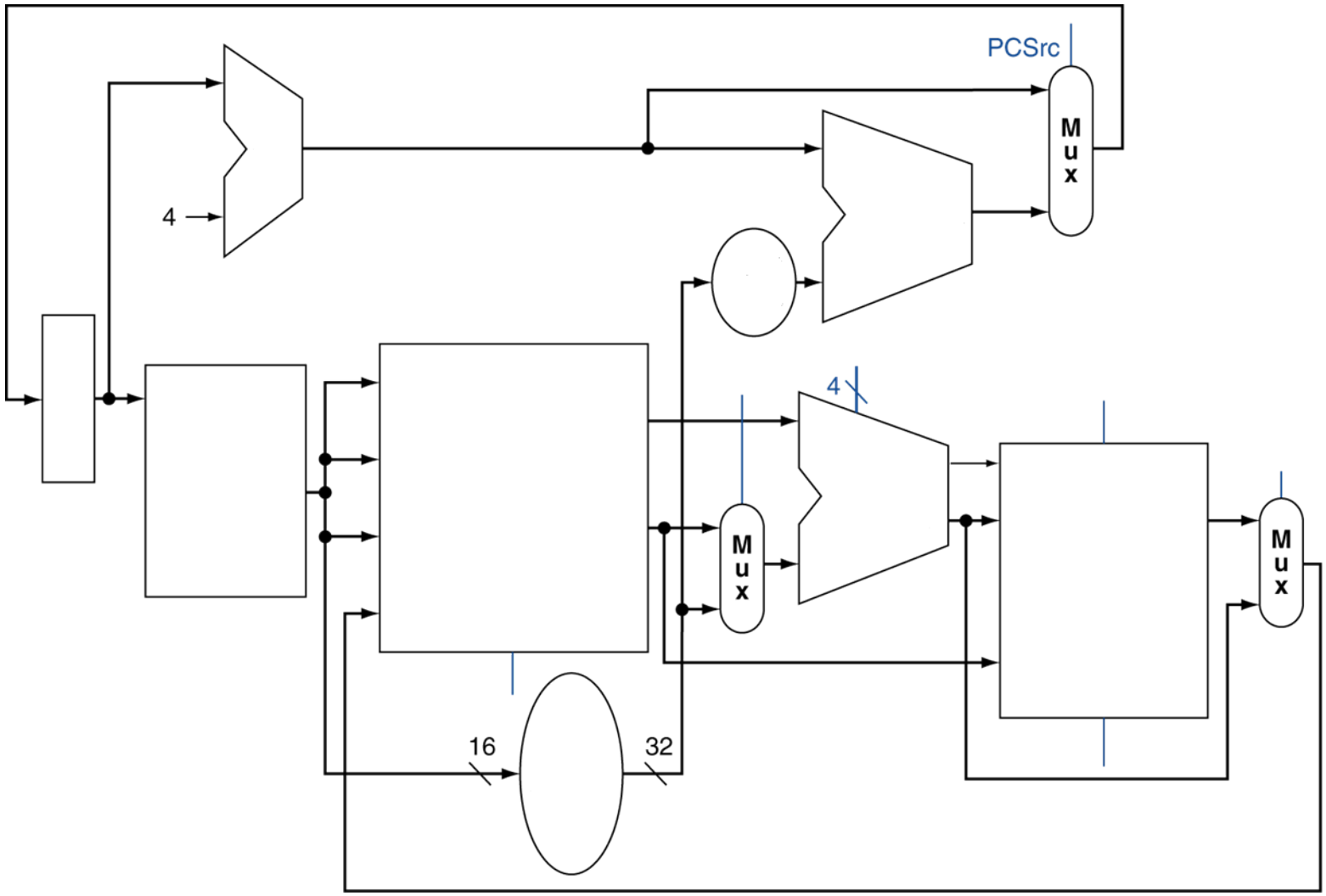
- Program 1

```
lw $t0 4($t1)
add $t0 $t0 $t2
beq $t0 $zero exit
```

- Program 2

```
sub $s1 $s1 $s2
sw $s1 0($t0)
beq $s1 $t2 loop
```

Blank Datapath



Datapath Exercise

- Trace instructions through datapath
  - Label used datapath components
  - Trace path through datapath

- Write your name at the top
- And the program you used