Chapter 4

The Processor

[Some slides adapted from A. Sprintson, M. Irwin, D. Paterson and others]

Reading Registers “Just in Case”

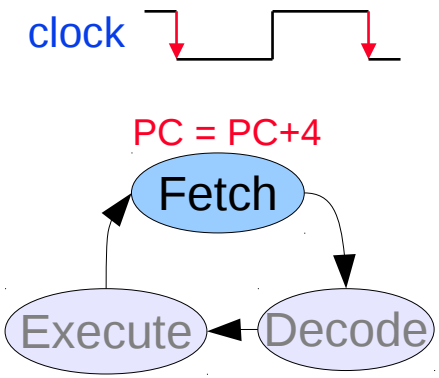
- Note that both RegFile read ports are active for all instructions during the Decode cycle using the *rn* and *rm* instruction field addresses
  - Since haven’t decoded the instruction yet, *we don’t know what the instruction is*!
  - Just in case the instruction uses values from the RegFile do “work ahead” by reading the two source operands

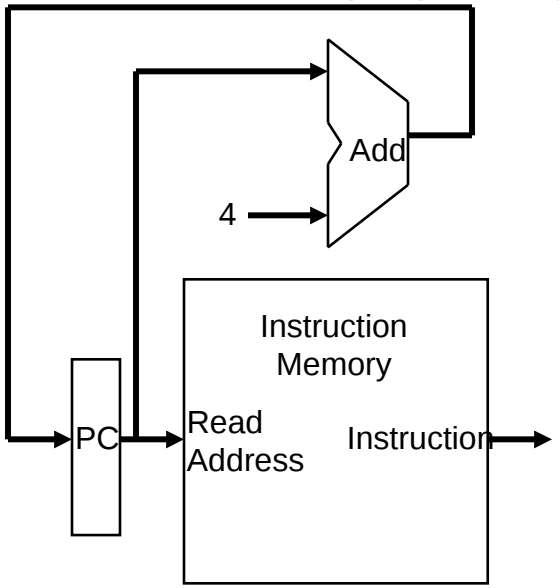
Which instructions do make use of the RegFile values?

- Also, all instructions (except **B**) use the ALU after reading the registers

Fetching Instructions

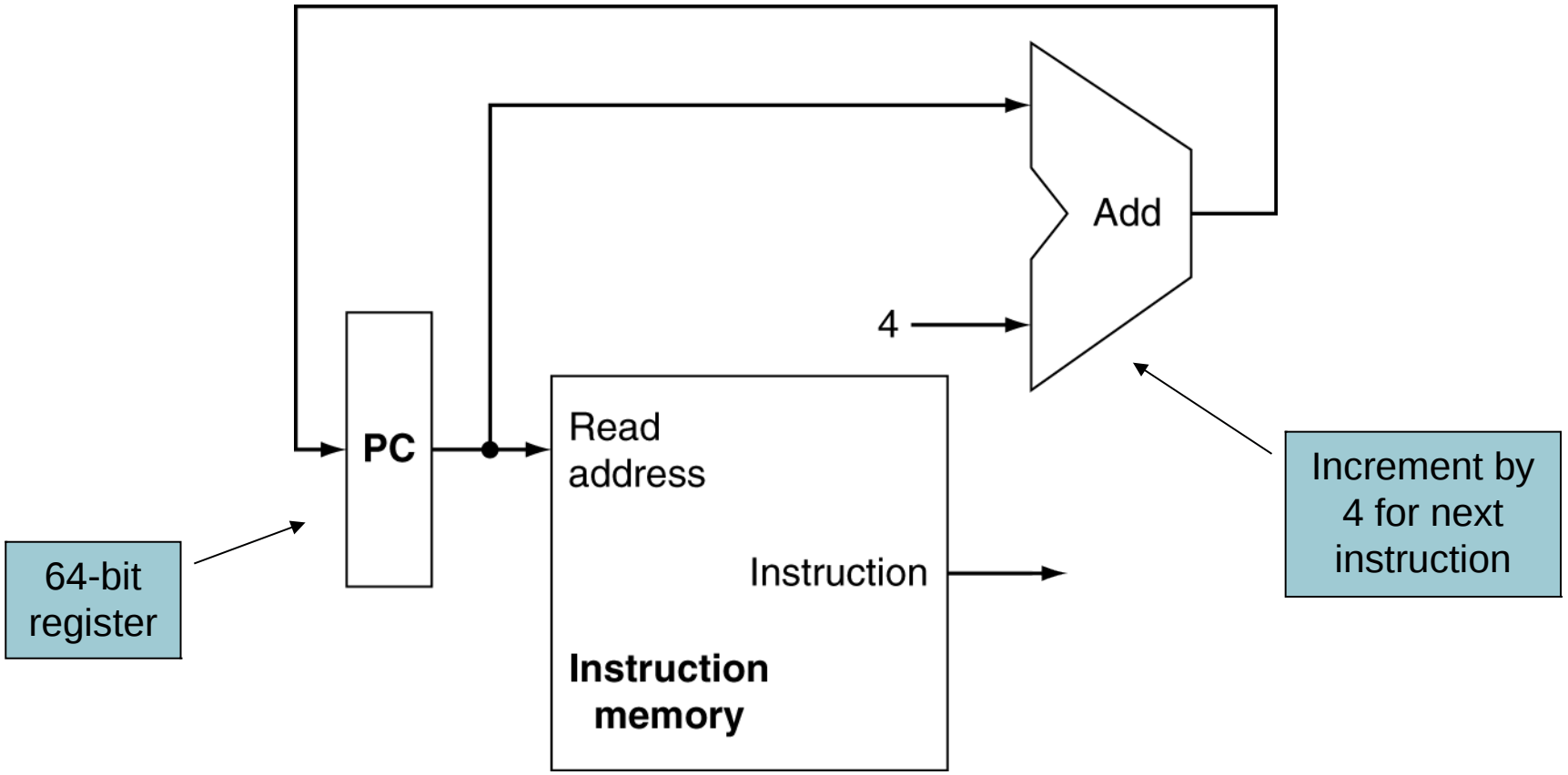
- Fetching instructions involves
  - reading the instruction from the Instruction Memory
  - updating the PC value to be the address of the next (sequential) instruction





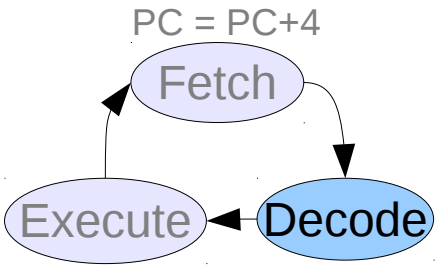
  - PC is updated every clock cycle, so it does not need an explicit write control signal
  - Instruction Memory is read every clock cycle, so it doesn't need an explicit read control signal

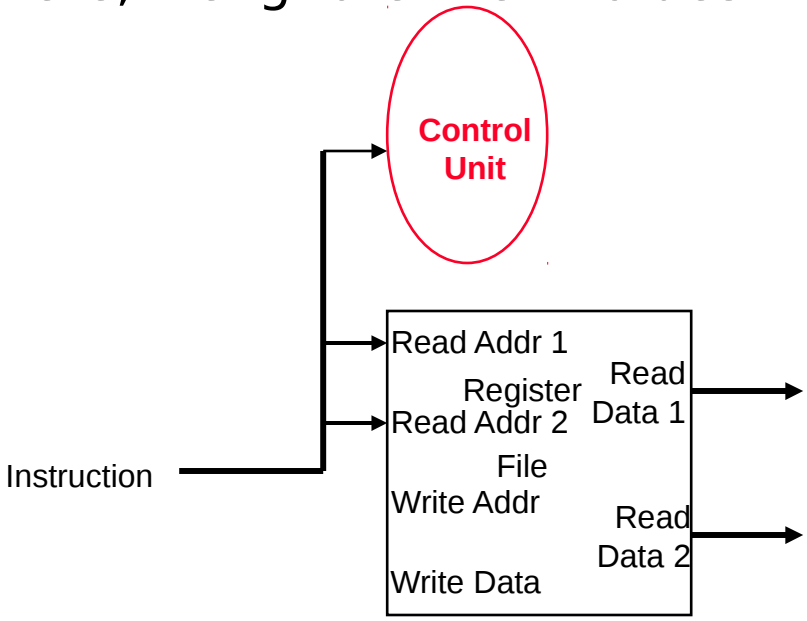
Instruction Fetch



Decoding Instructions

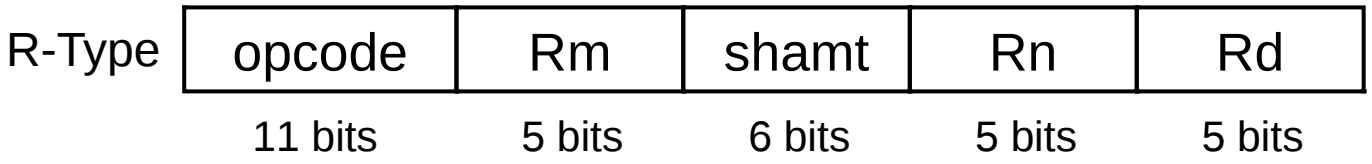
- Send the fetched instruction's opcode to the control unit
- Read two values from the Register File
  - Register File indices are contained in the instruction
  - Read data outputs are always enabled. If we don't need them elsewhere, we ignore their values.





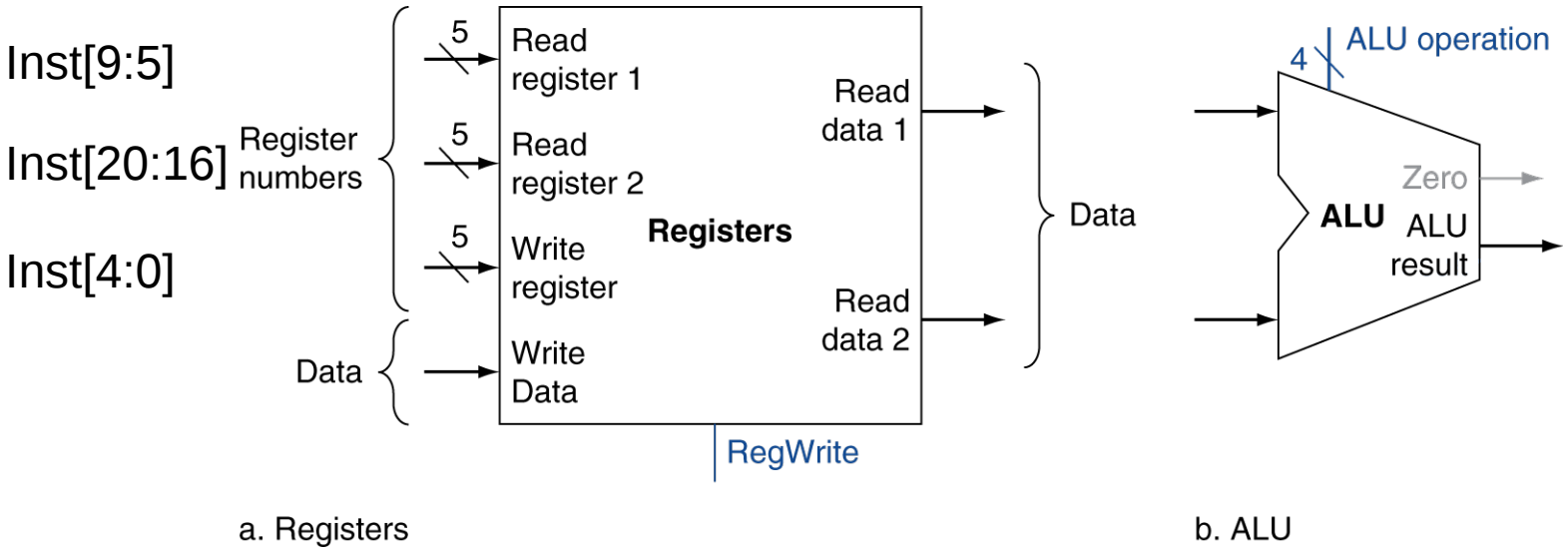
R-Format Instructions

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



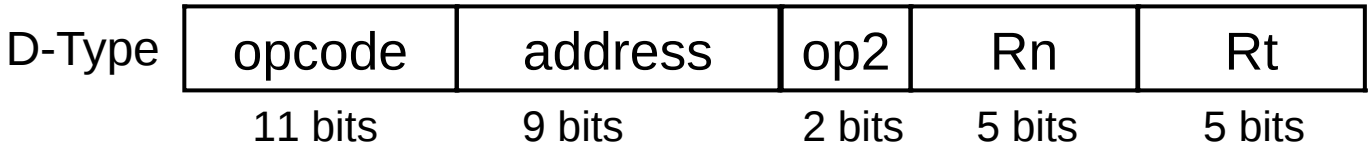
R-Format Instructions

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



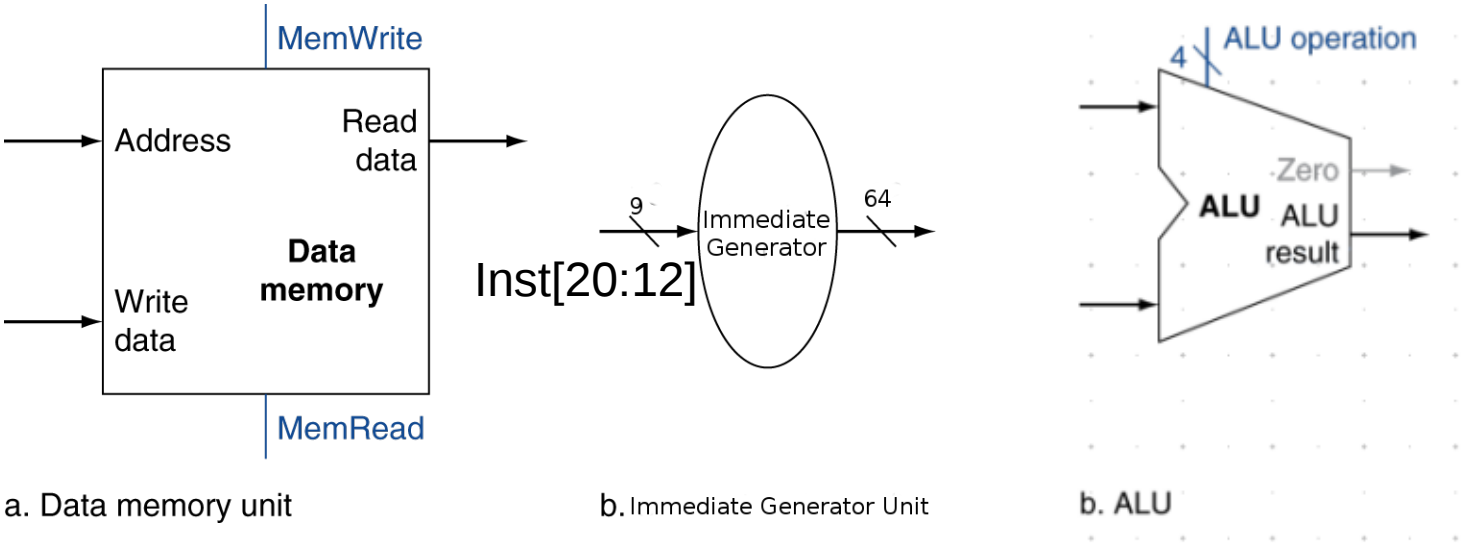
Load/Store Instructions

- Read register operands
- Calculate address using up to 9-bit offset
  - Use ALU, but sign-extend offset
- Load: Read memory and update register
- Store: Write register value to memory



Load/Store Instructions

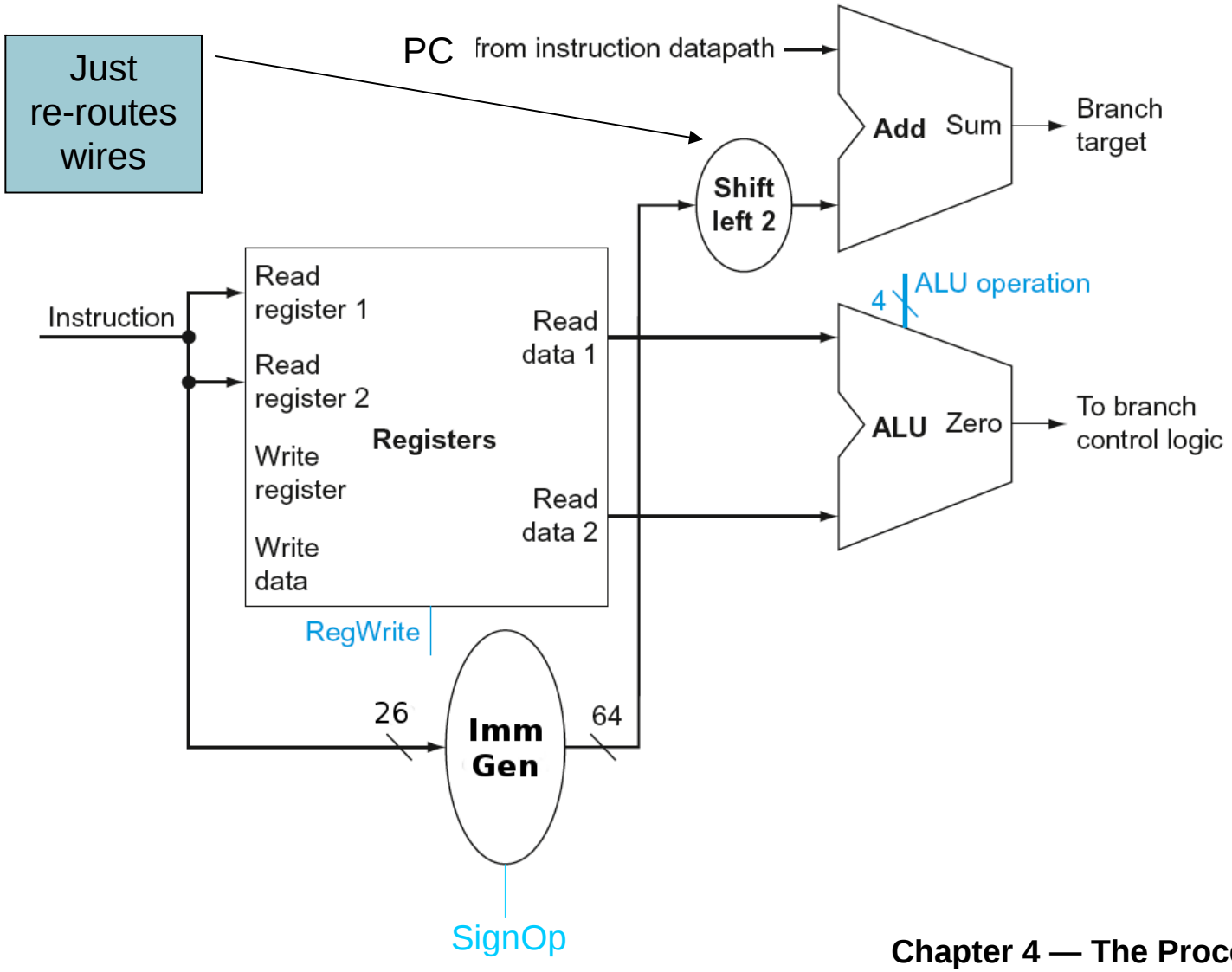
- Read register operands
- Calculate address using up to 9-bit offset
  - Use ALU, but sign-extend offset
- Load: Read memory and update register
- Store: Write register value to memory



Branch Instructions

- Read register operand
- Test for zero
  - Use ALU, check Zero output
- Calculate target address
  - Sign-extend immediate
  - Shift left 2
  - Add to PC

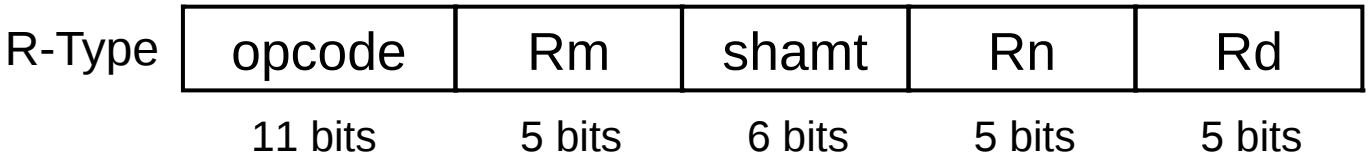
Branch Instructions

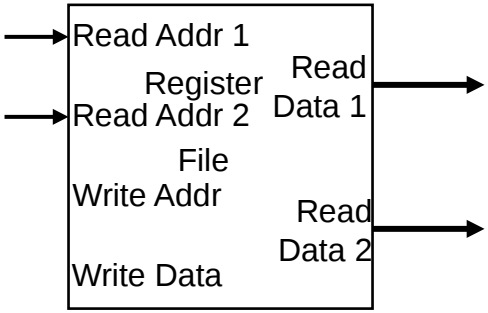


Composing the Elements

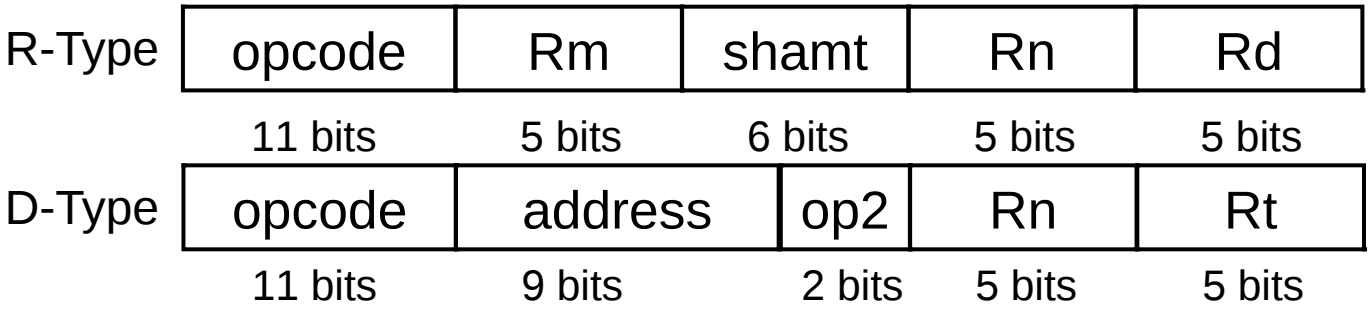
- First-cut data path does an instruction in one clock cycle
  - Each datapath element can only do one function at a time
  - Hence, we need separate instruction and data memories
- Use multiplexers where alternate data sources are used for different instructions

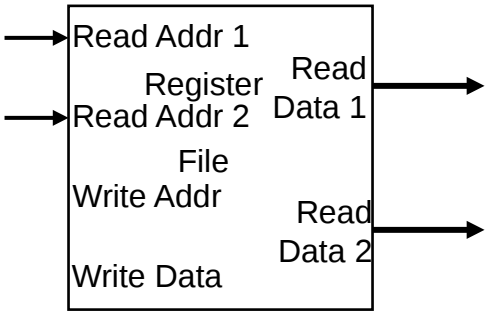
Constructing a single Datapath:R-type

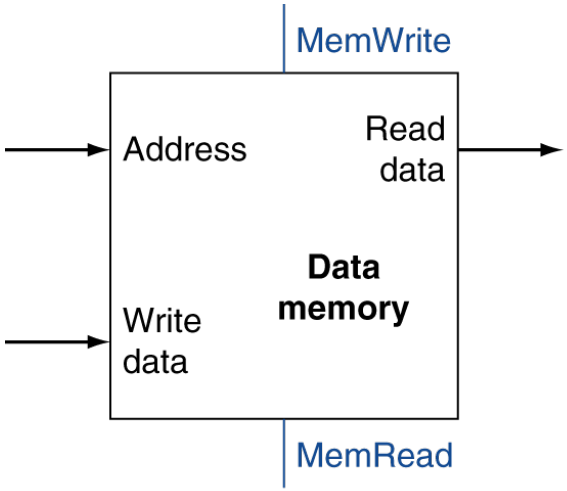


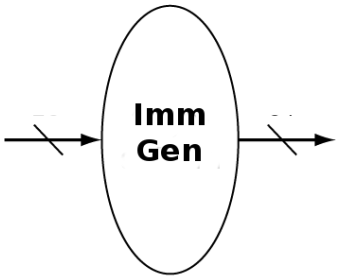


Constructing a single Datapath:R-type,D-Type

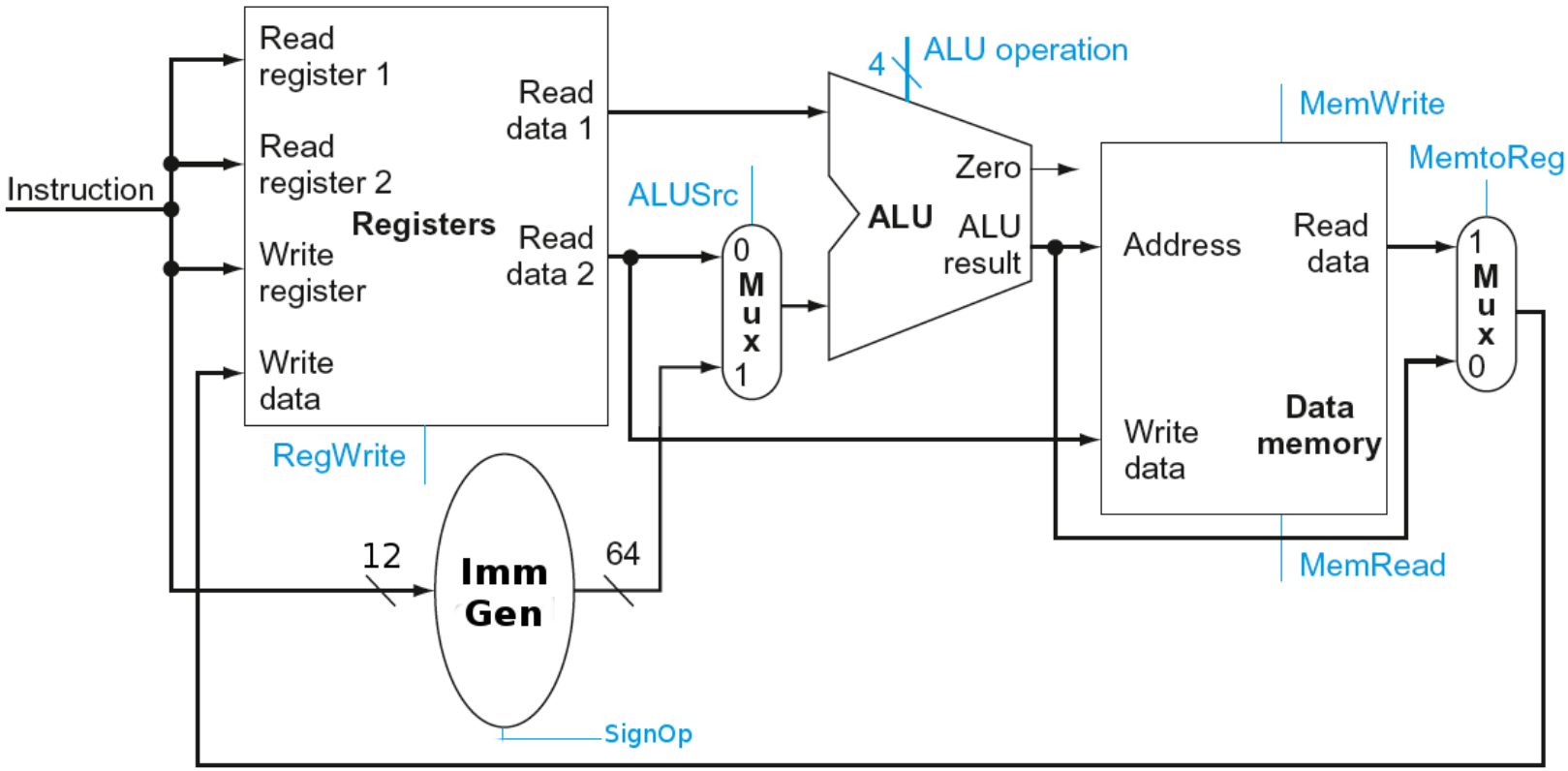








R-Type/Load/Store Datapath



Full Datapath: Adding B-Type