Outline of Lecture

- **Main Memory**
- **Building a Datapath**

Main Memory

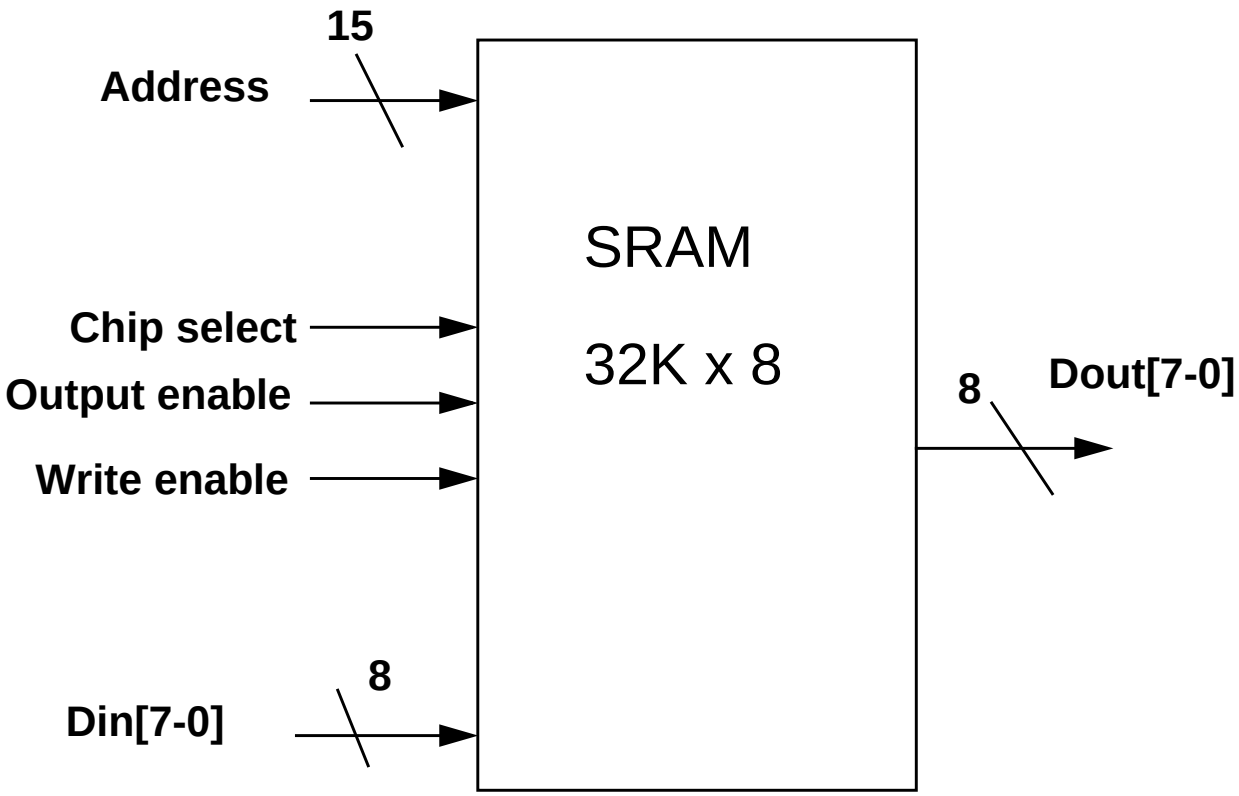
- A register file is a fast small amount of memory inside the processor.
- Larger amount of memory which resides outside the processor are built either using ***SRAMs*** (Static Random Access Memory) or ***DRAMs*** (Dynamic Random Access Memory).
- SRAMs (They are typically built using flip-flops) do not need to be ***refreshed*** periodically like DRAMs (They are typically built using capacitors) - but are more expensive and less dense.
- SRAMs are used for small memories (e.g., Cache), and DRAMs are used for large memories (e.g., main memory).

Example

a 256K x 1 SRAM contains 256K entries, and each entry is 1 bit wide.

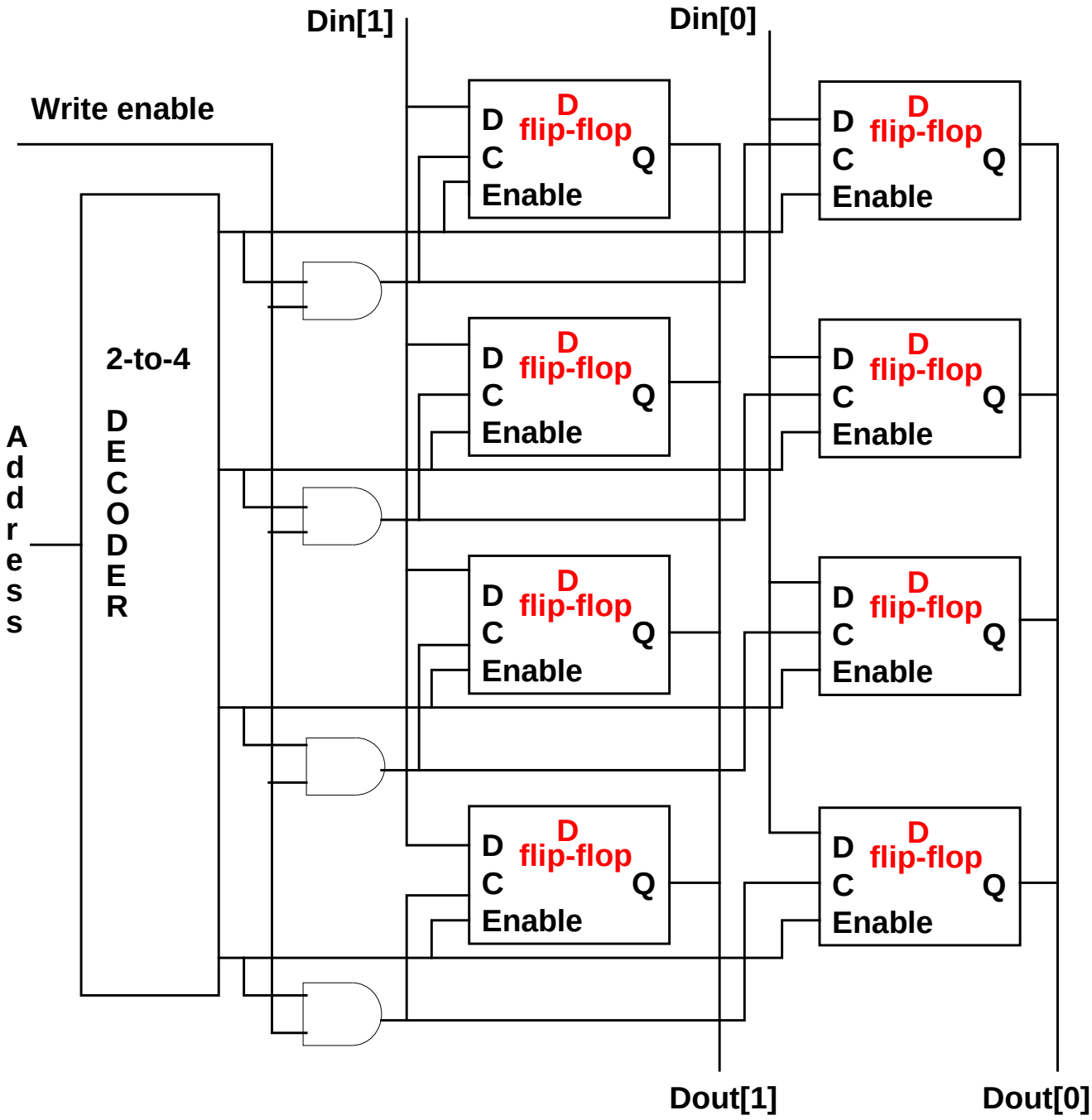
Thus we need 18 address lines (i.e., 2^{18} = 256K).

- A 32K x 8 SRAM has the same number of bits, but will have 15 address lines (to address 32K entries) and each entry holds 8 bits.



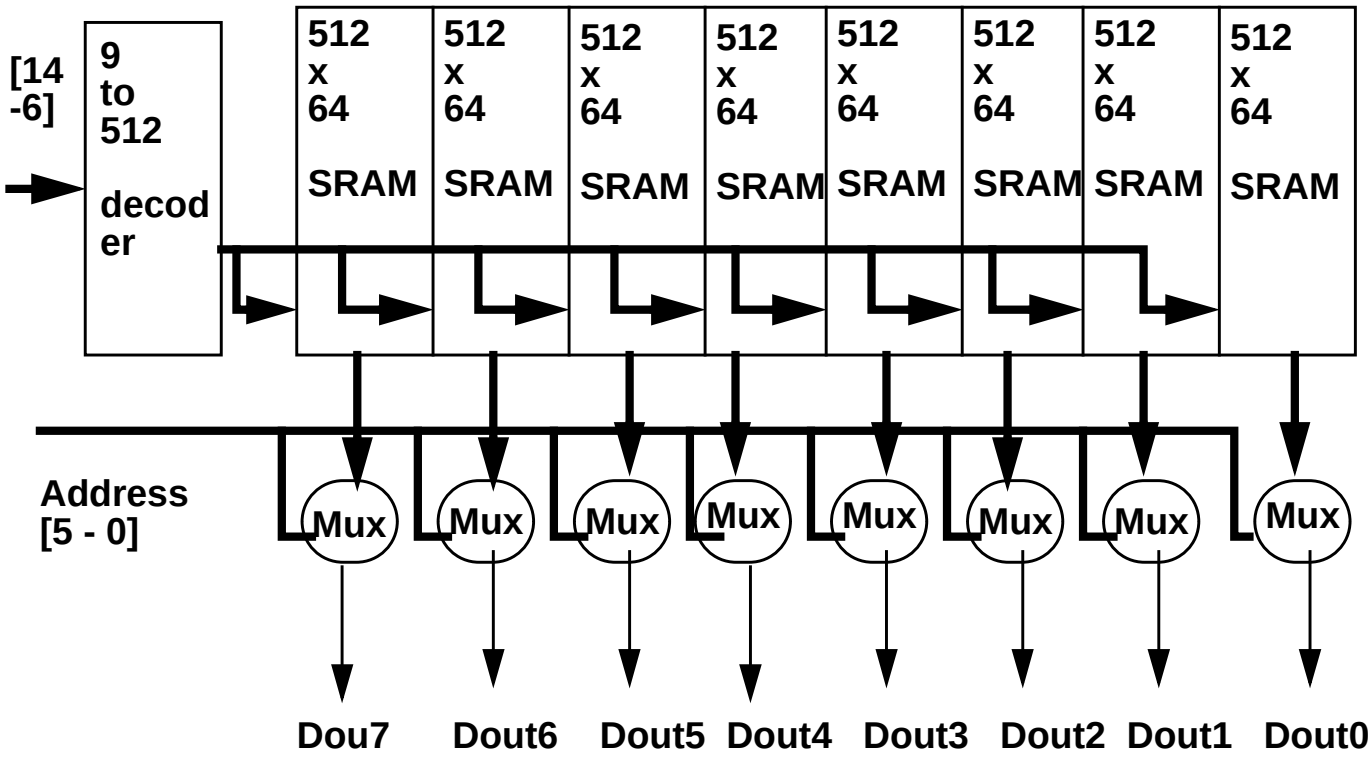
Decoder

- In order to activate individual words within a SRAM, we need a *decoder* to accomplish that.



Two-Step Decoder

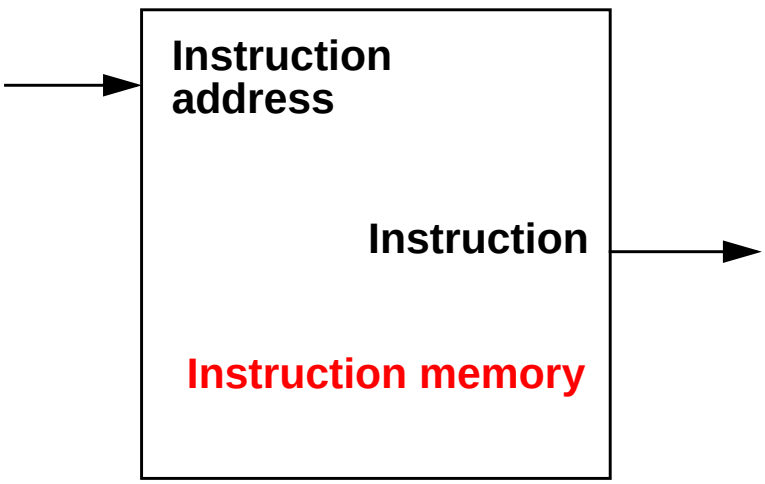
- In order to solve the problem of having large decoders for large memories, a *two-step decoding process* can be used:



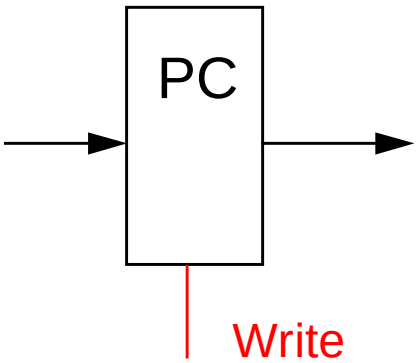
Building a Datapath

- In designing our datapath, we initially assume that each MIPS instruction takes a *single clock cycle*. Later, we will look at the more realistic case, where each instruction takes a variable number of clock cycles.

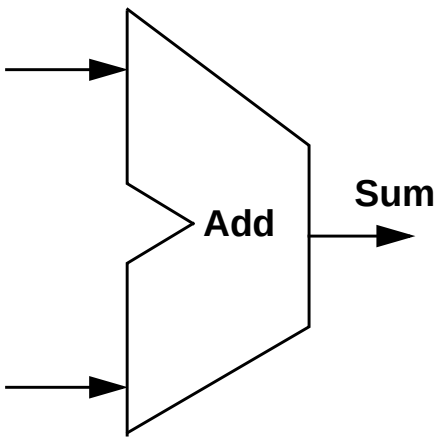
- The first element we need in a data path is a *memory unit* to store the instructions in a program and supply an instruction given its address.



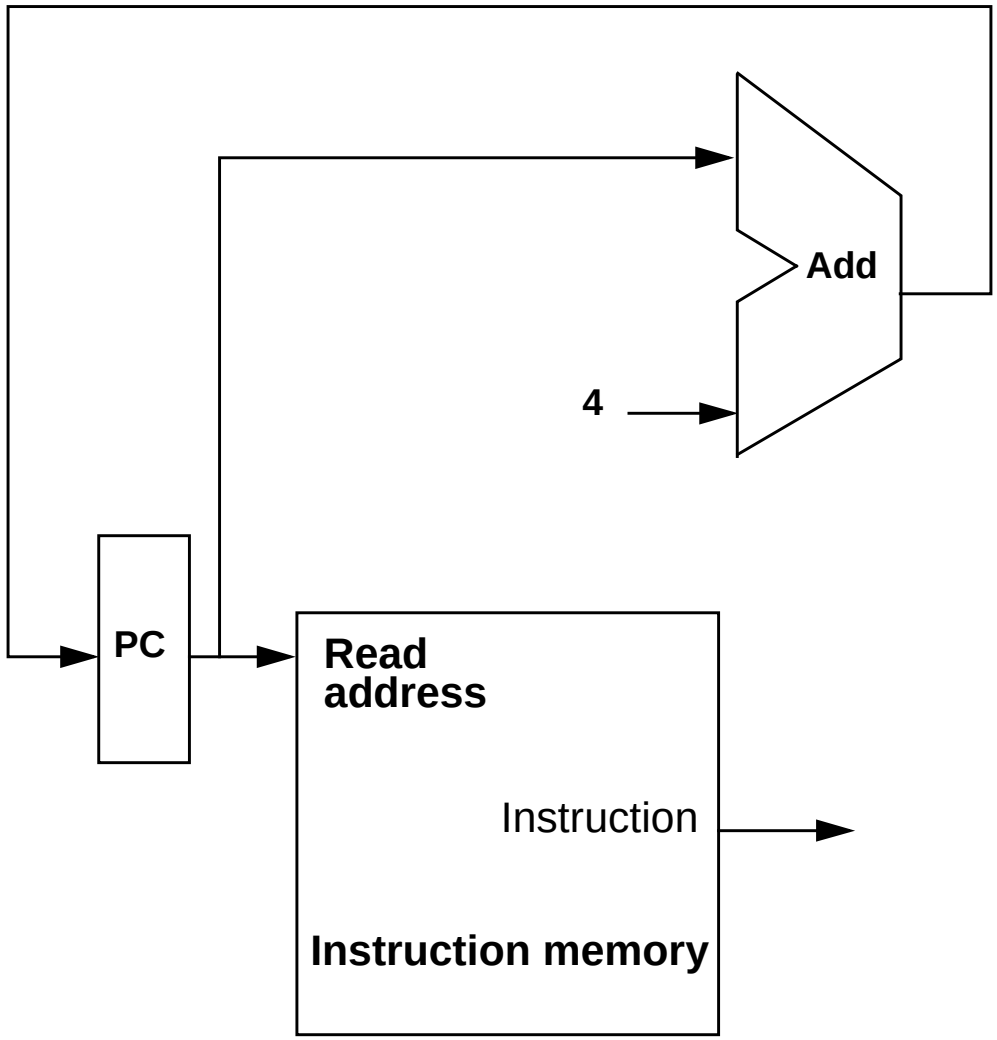
- **The address of the instruction being executed is also stored in a register called *program counter* (PC).**



- **We need to be able to increment the PC to point to the next instruction. This can be done using the ALU we designed in the previous chapter.**



- In order to execute an instruction, we need to *fetch* the instruction, and *increment* (+ 4) the program counter so that it is ready for the next instruction.

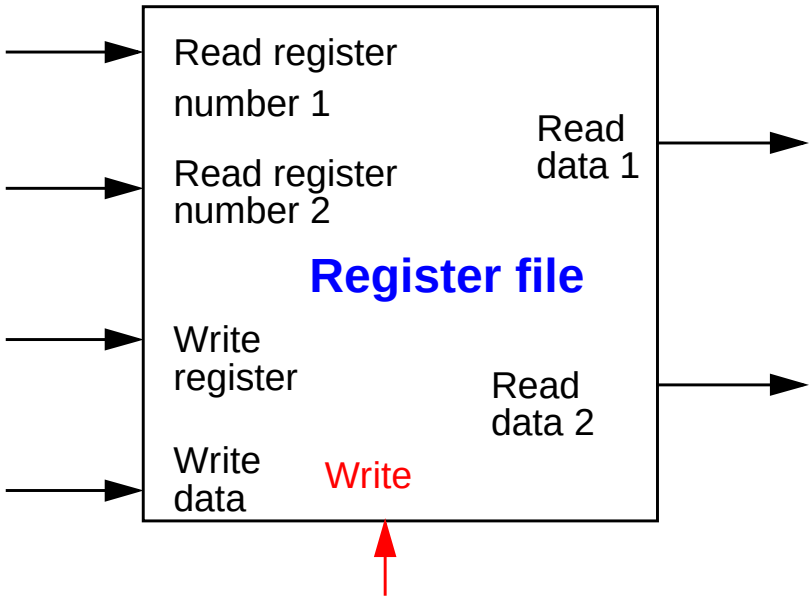


- Next we consider the execution of ***R-format instructions*** (R-type instructions). They all read two registers, perform an ALU operation (add, sub, and, `or`, and `slt`) and write the result into a register.

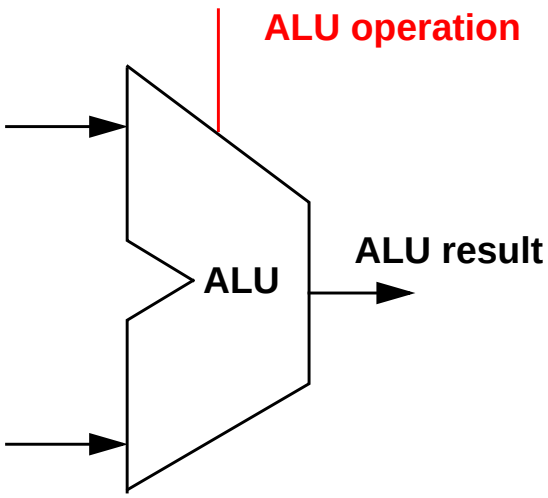
➔ **To read:** We need an input to the register file that specifies the register number (5 bits) to be read and an output (32 bits) that will carry the value read.

➔ **To write:** We need to specify the register number to be written, and we need to input the data (32 bits) that will be written into that register.

∴ We need a total of 4 inputs

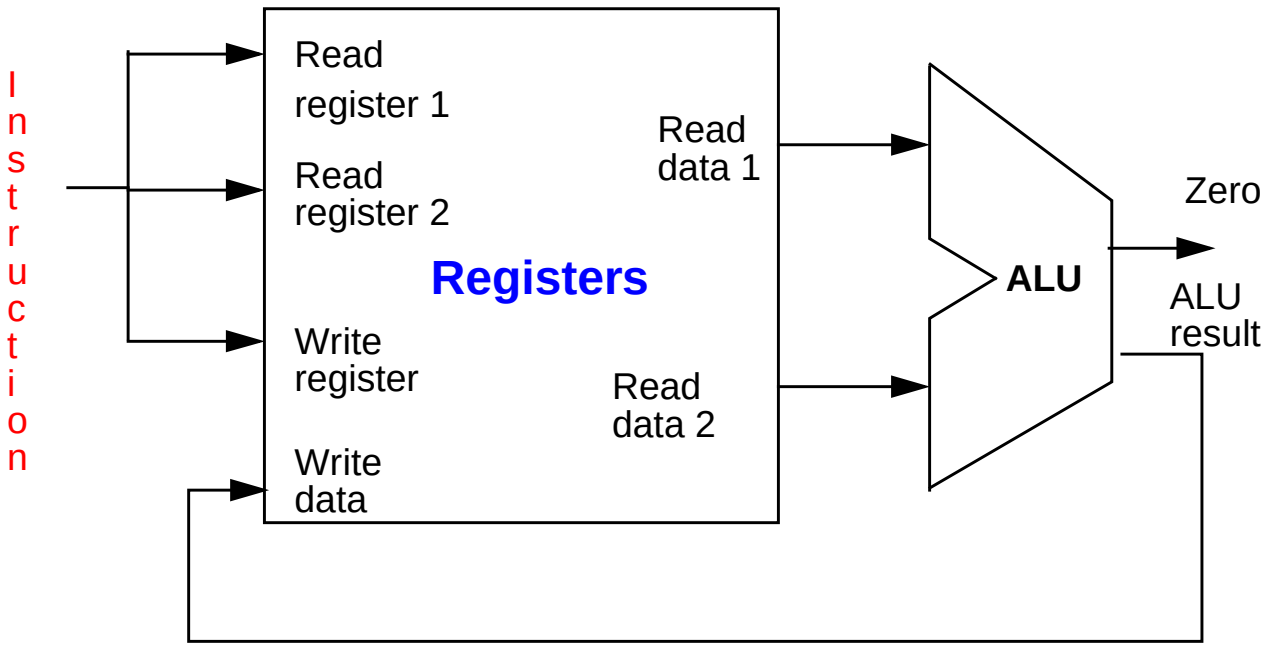


- **The ALU takes two 32-bit inputs and produces a 32-bit result - it is controlled by a 3-bit signal as described in our previous lectures.**



The Data Path for R-type Instructions

- The Data path for R-type instructions is as follows:



Further Reading

Chapter 5 & Appendix B. **David A. Patterson and John L. Hennessy. *Computer Organization & Design: The Hardware / Software Interface*. Morgan Kaufman Publishers, 1998. (343-348 and B-28 to B-33).**