Levels in Processor Design

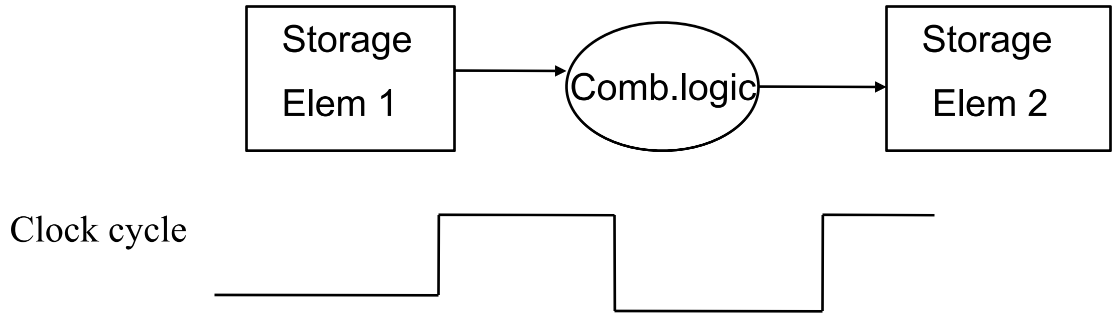
- Circuit design
  - Keywords: transistors, wires etc.Results in gates, flip-flops etc.
- Logical design
  - Putting gates (AND, NAND, …) and flip-flops together to build basic blocks such as registers, ALU's etc (cf. CSE 370)
- **Register transfer**
  - Describes execution of instructions by showing data flow between the basic blocks
- **Processor description** (the ISA)
- System description
  - Includes memory hierarchy, I/O, multiprocessing etc

Register transfer level

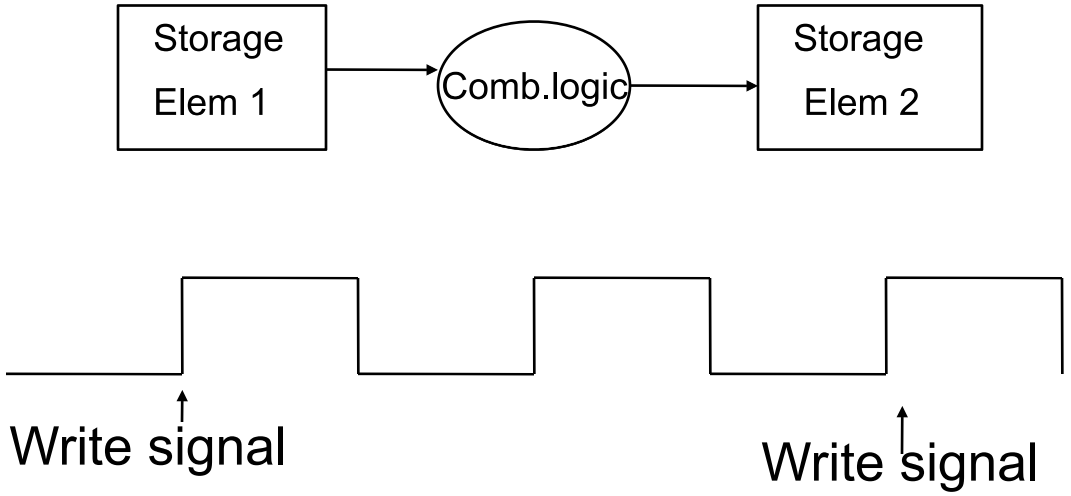
- Two types of components (cf. CSE 370)
  - *Combinational* : the output is a function of the input (e.g., adder)
  - *Sequential*: state is remembered (e.g., register)

Synchronous design

- Use of a periodic clock
  - *edge-triggered* clocking determines when signals can be read and when the output of circuits is stable
  - Values in storage elements can be updated only at clock edges
  - Clock tells when events can occur, e.g., when signals sent by control unit are obeyed in the ALU



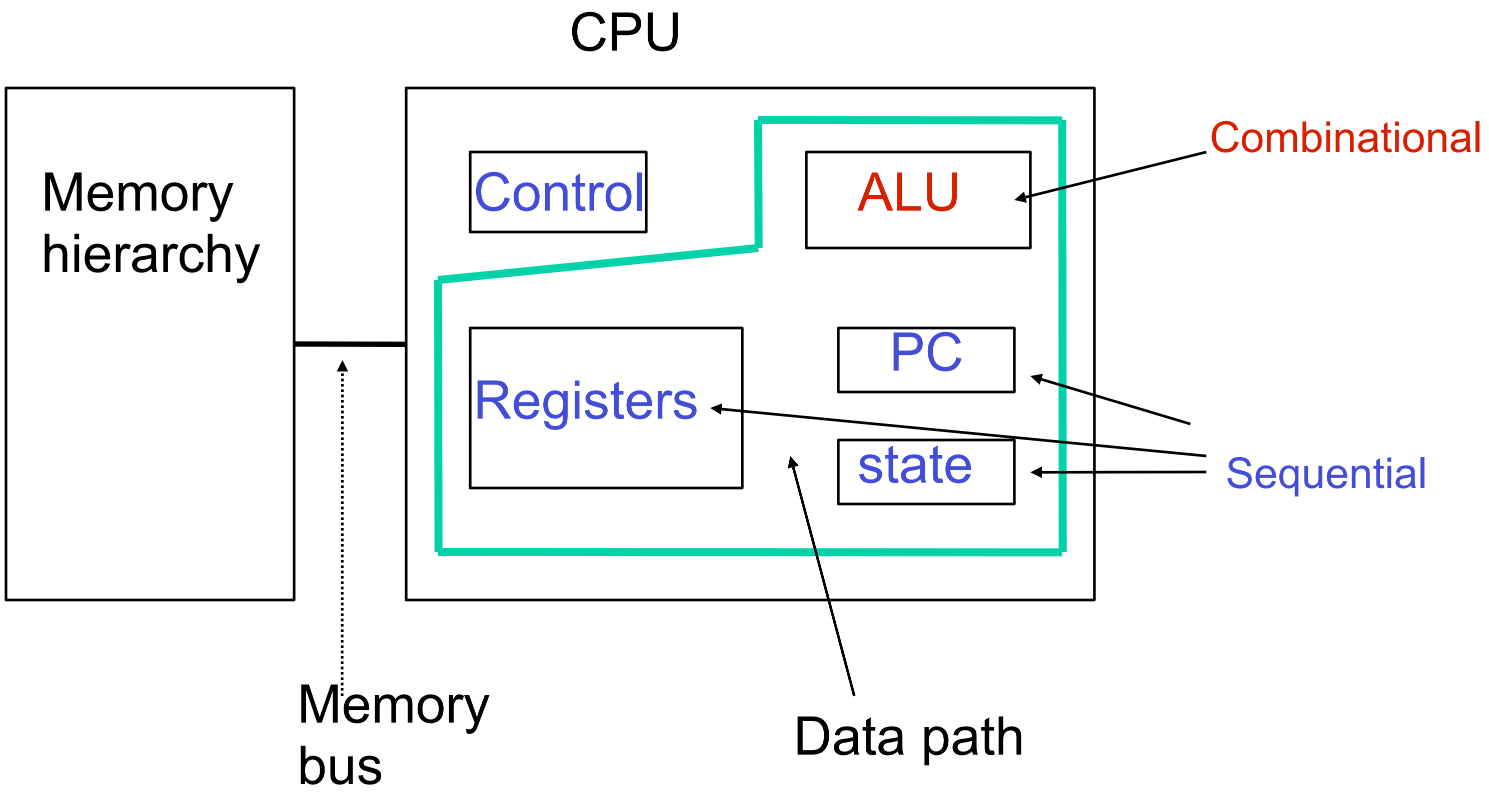
Note: the same storage element can be read/ written in the same cycle



Logic may need several cycles to propagate values

True in designs today with very high clock frequency

Processor design: data path and control unit

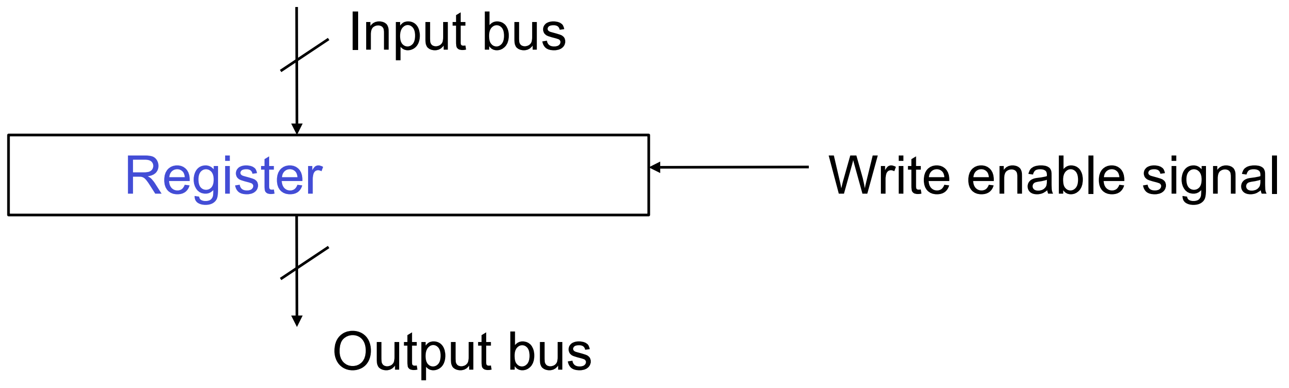


Processor design

- Data path
  - How does data flow between various basic blocks
  - What operations can be performed when data flows
  - What can be done in one clock cycle
- Control unit
  - Sends signals to data path elements
  - Tells what data to move, where to move it, what operations are to be performed
- Memory hierarchy
  - Holds program and data

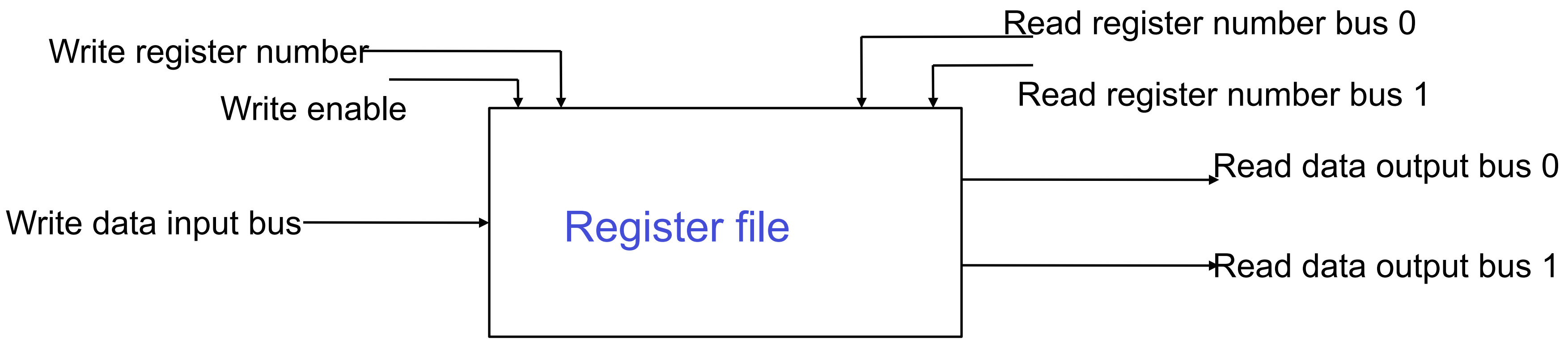
Data path basic building blocks. Storage elements

- Basic building block (at the RT level) is a register
- In our mini-MIPS implementation registers will be 32-bits
- A register can be read or written



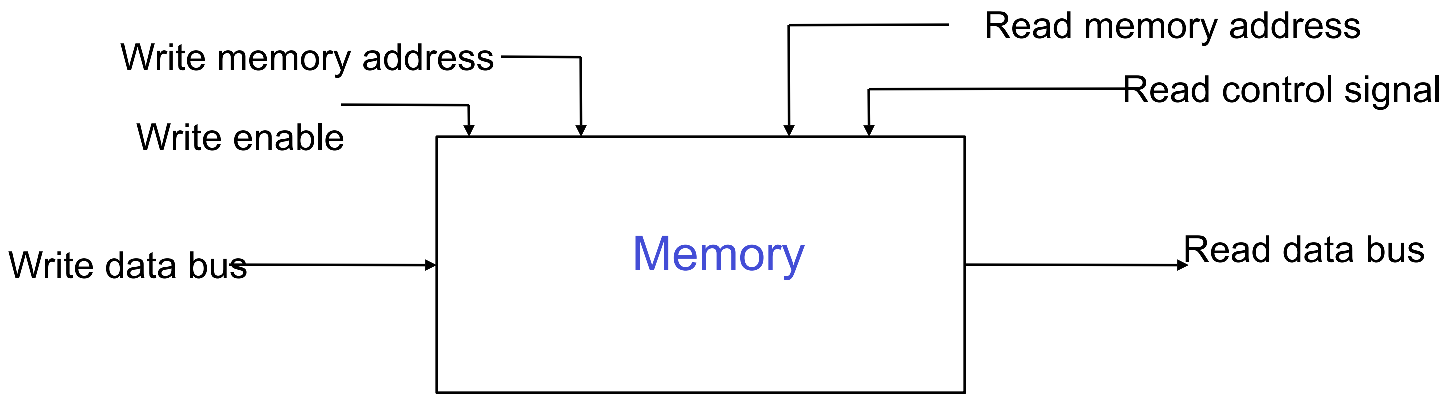
Register file

- Array of registers (32 for the integer registers in MIPS)
- ISA tells us that we should be able to:
  - read 2 registers, write one register in a given instruction (at this point we want one instruction per cycle)
  - Register file needs to know which registers to read/write



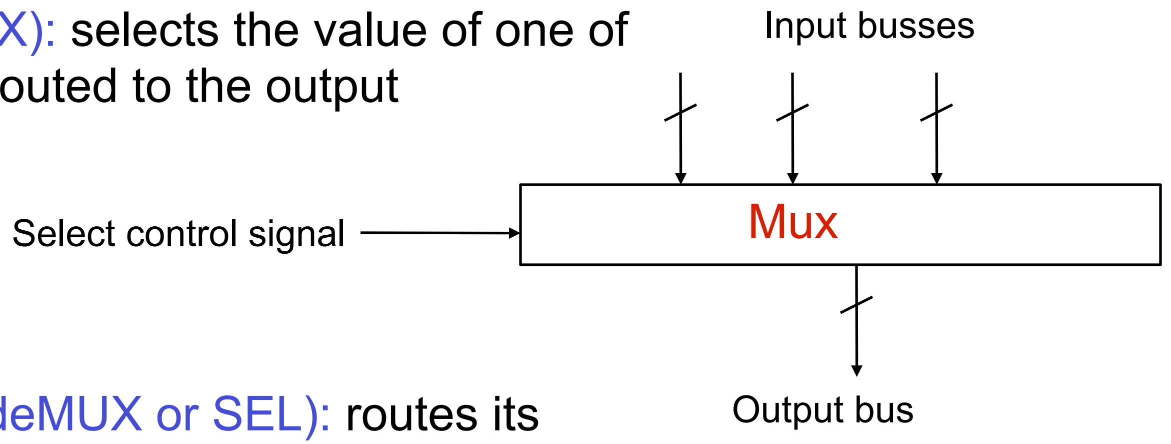
Memory

- Conceptually, like register file but much larger
- Can only read one location or write to one location per cycle

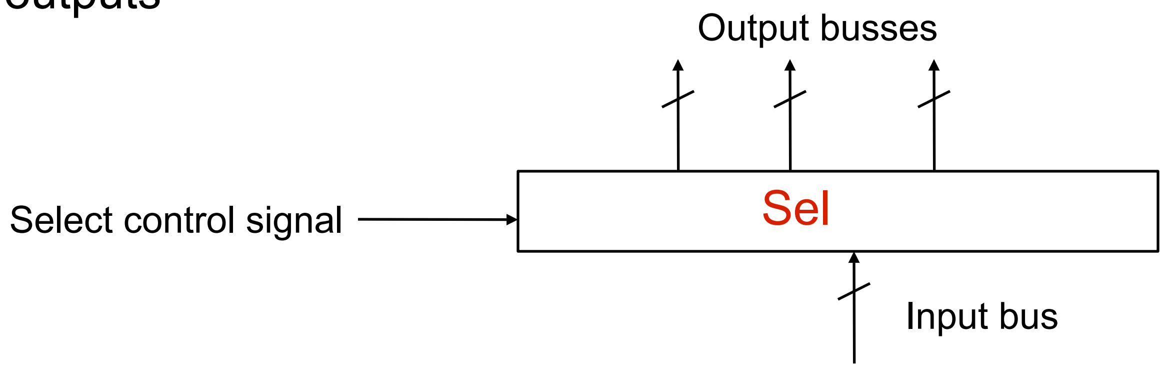


Combinational elements

Multiplexor (MUX): selects the value of one of its inputs to be routed to the output

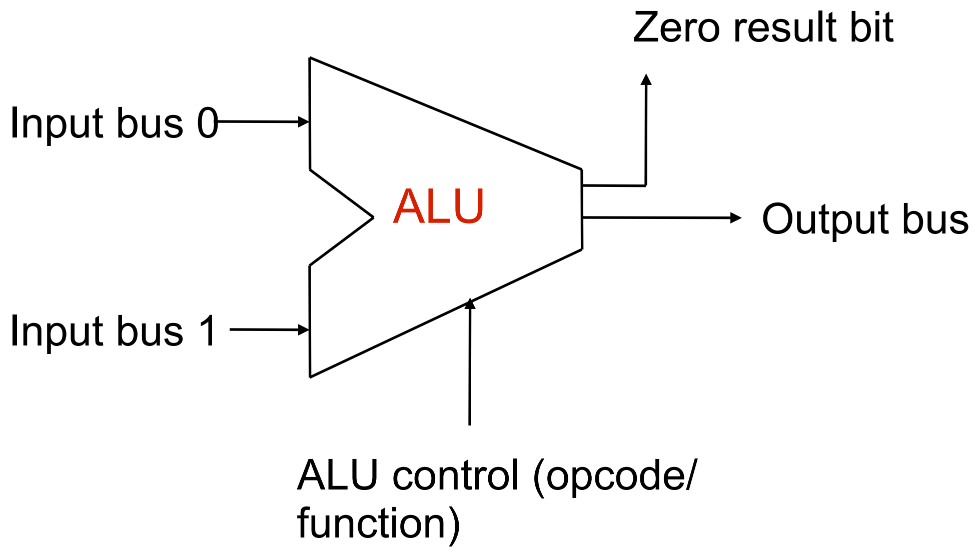


Demultiplexor (deMUX or SEL): routes its input to one of its outputs

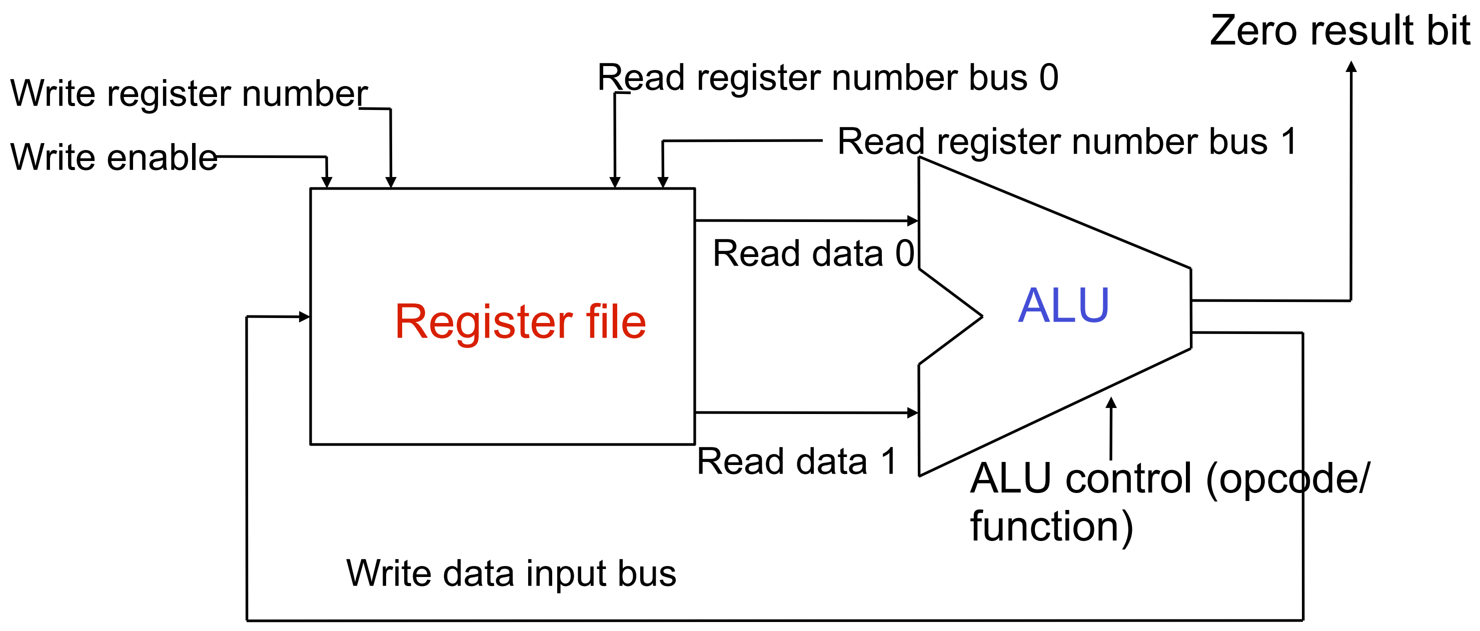


Arithmetic and Logic Unit (ALU - combinational)

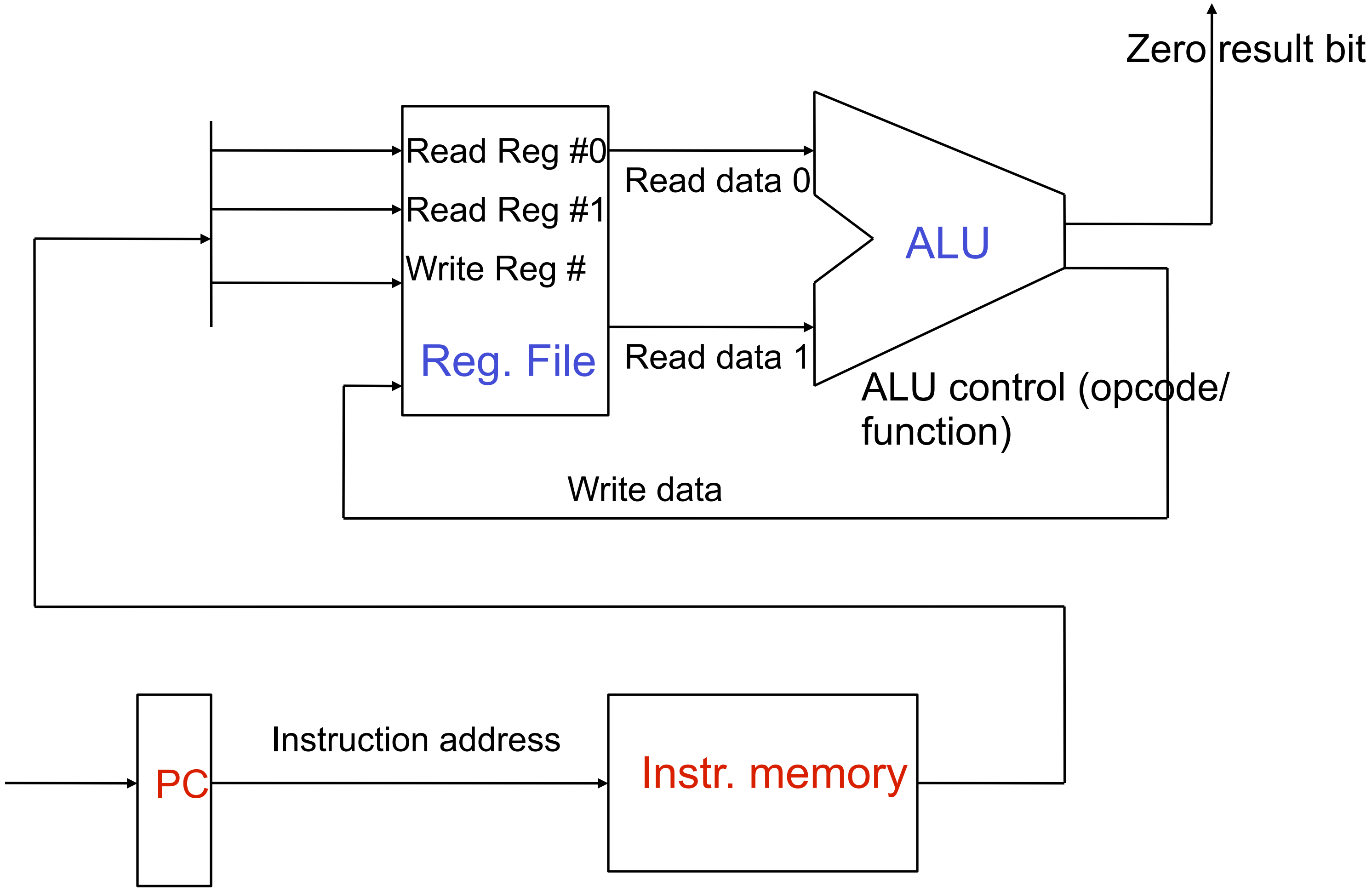
- Computes (arithmetic or logical operation) output from its two inputs



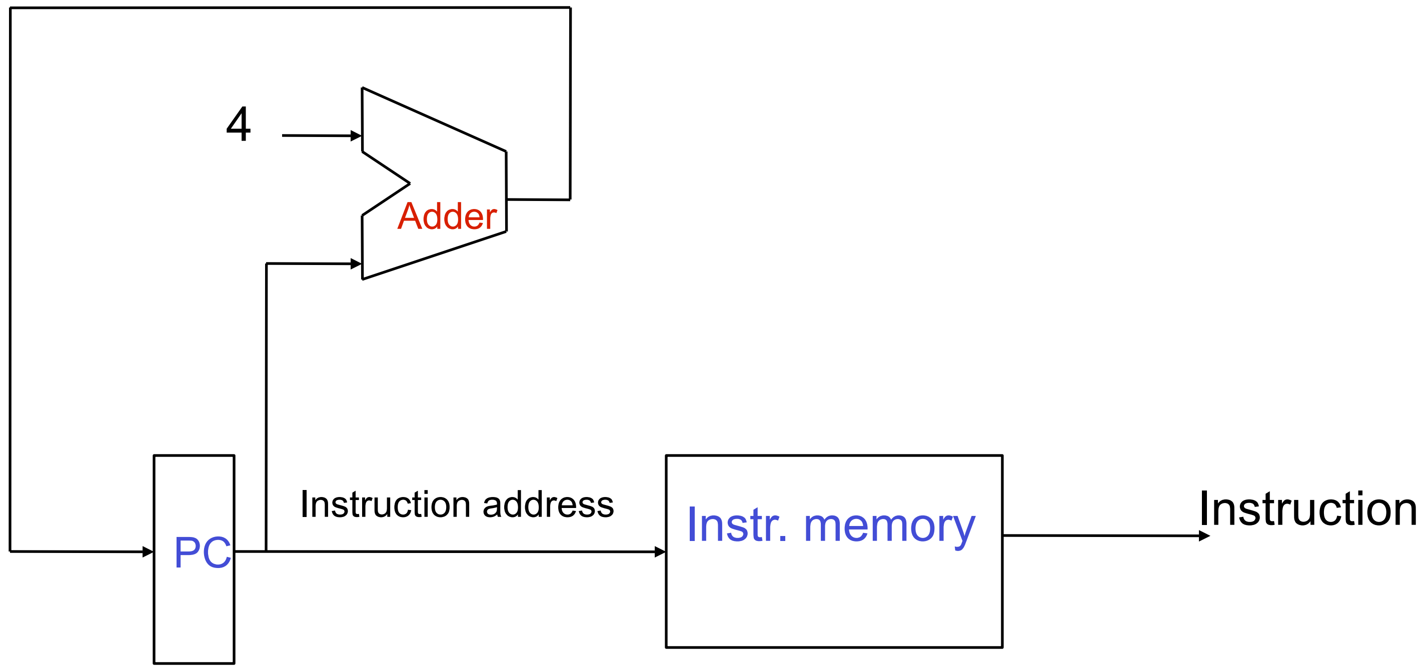
Putting basic blocks together (skeleton of data path for arith/logical operations)



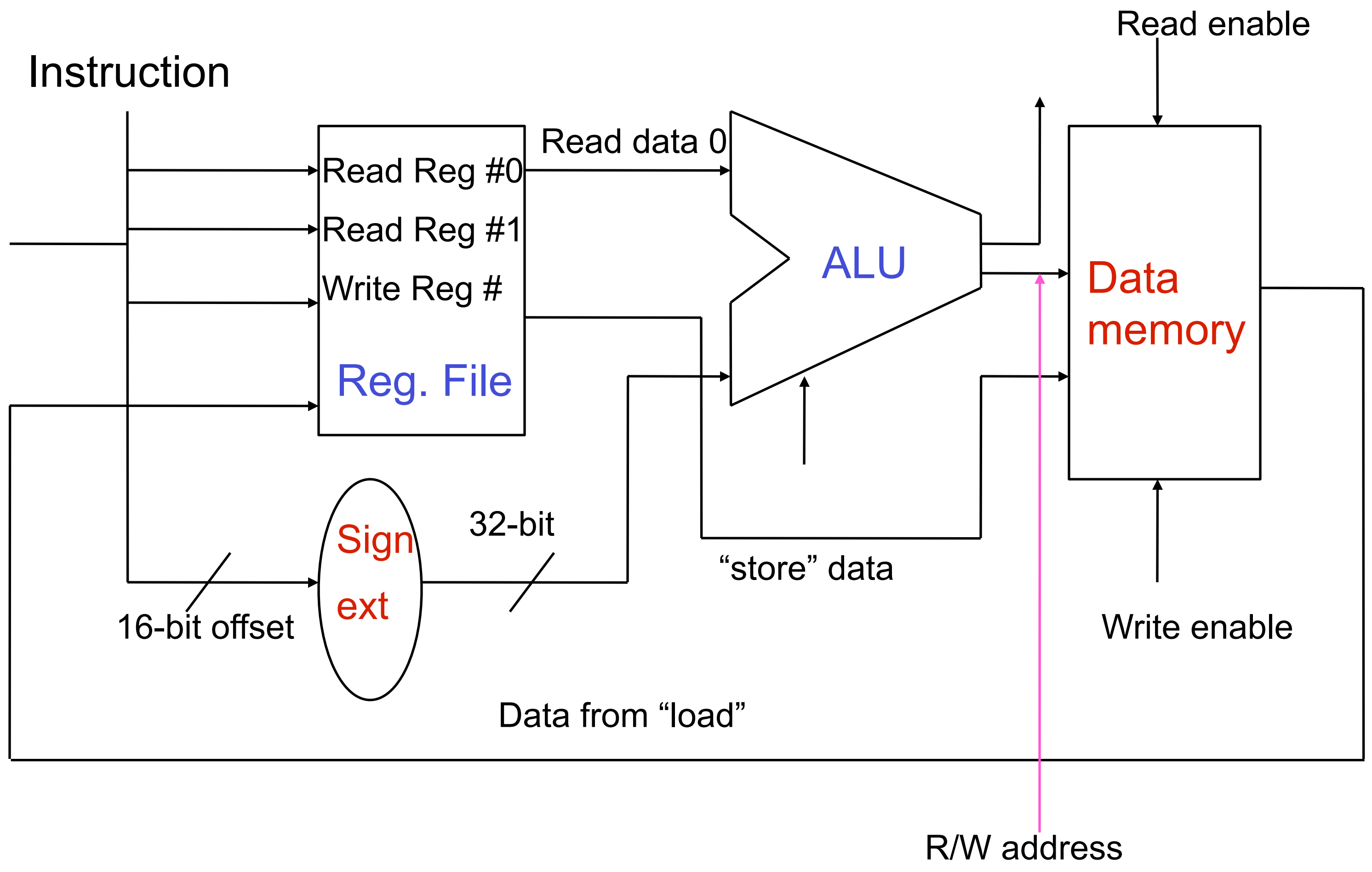
Introducing instruction fetch



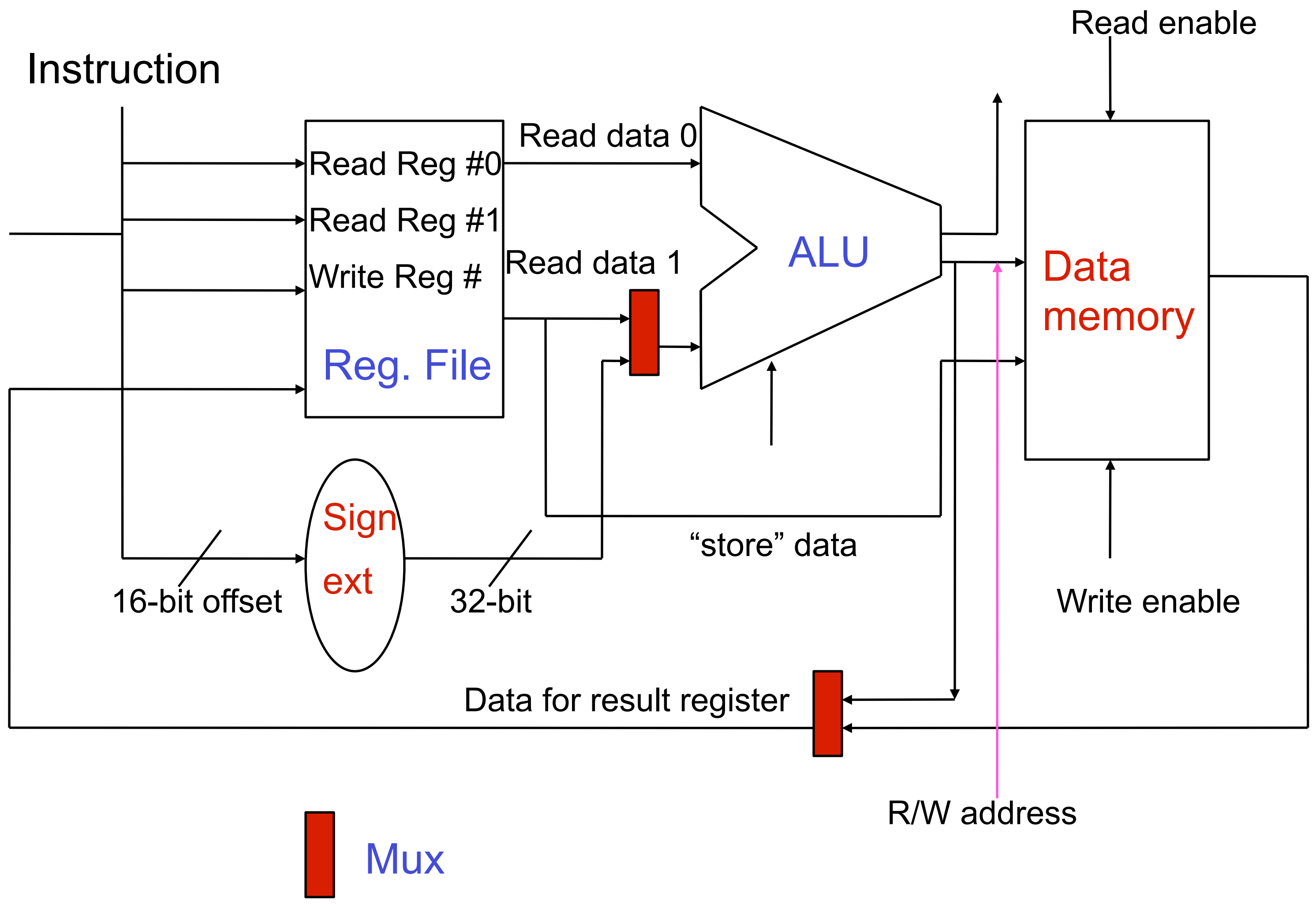
PC has to be incremented (assume no branch)



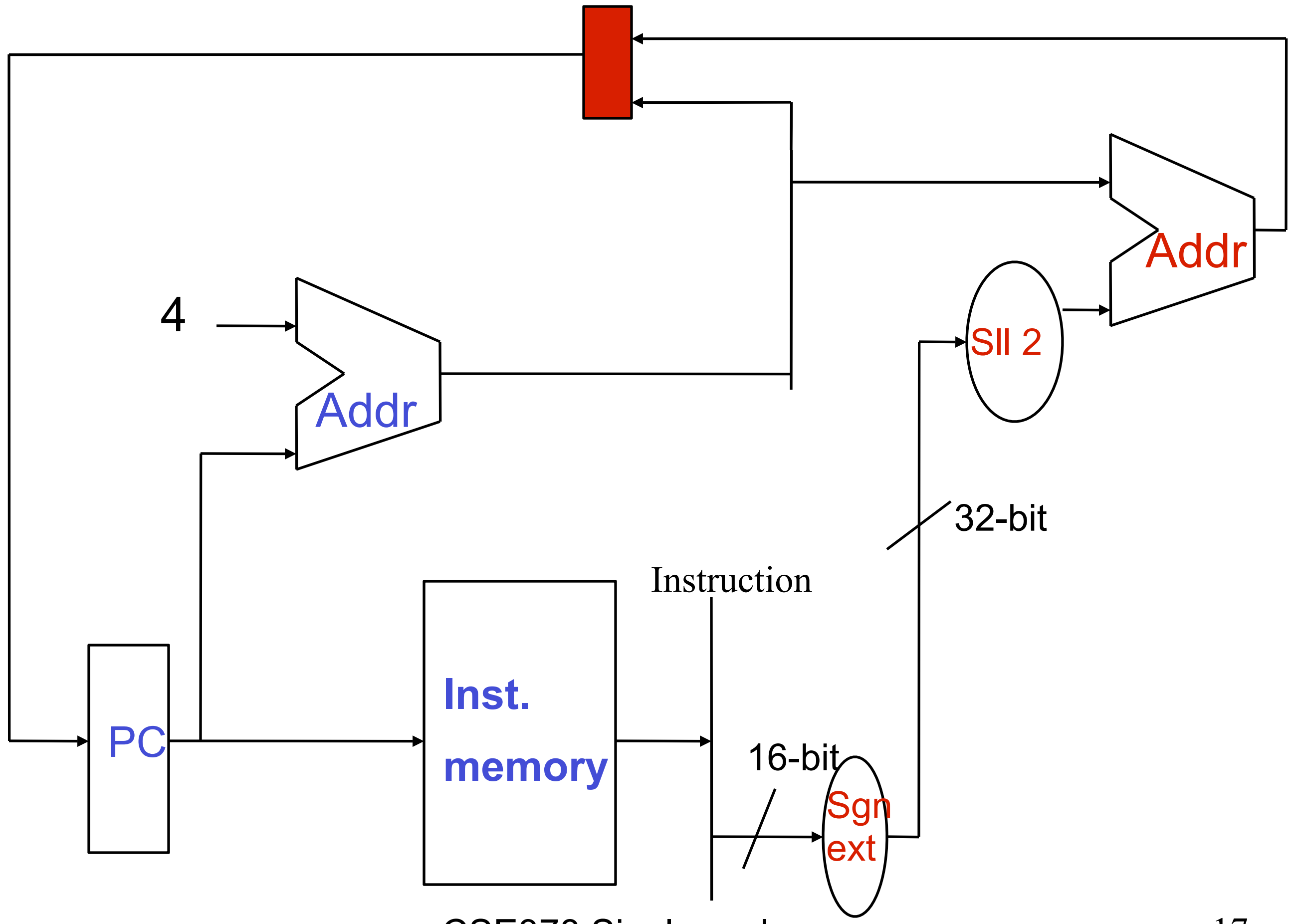
Load-Store instructions



Load-Store instructions



Branch data path


4
Addr
Addr
Sll 2
32-bit
Instruction
PC
Inst.
memory
16-bit
Sgn
ext