Levels in processor design

Circuit design: the key words here are transistors, resistors etc. The result is in terms of gates, flip-flops and so on.

Logical design: putting gates (AND, OR, NAND, NOR, etc.) and flip-flops together to build blocks such as registers, adders, memory etc. (cf. CSE 370)

Register transfer level: describes the execution of instructions by showing how information is transferred and manipulated between adders, registers etc.

Processor description: the ISA (as we have seen it)

System description: includes memory hierarchy, I/O, number of processors etc.

Processor design or register transfer level

Two types of components:

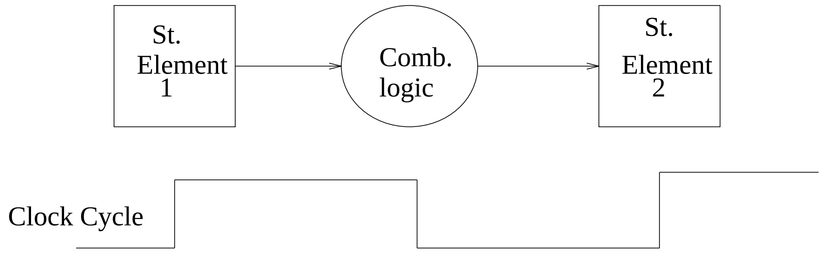
- Combinational: the output is a function of the inputs (e.g., adder)
- Sequential: state is remembered (e.g., register)

Synchronous design

Use of a periodic clock. *Edge-triggered* clocking methodology determines when signals can be read and when they can be written.

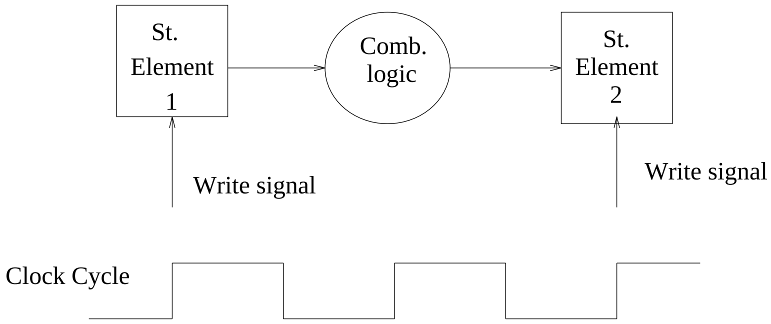
Values in storage elements can only be updated at clock edges.

The clock tells when events can occur, i.e., when signals sent by the control unit are obeyed in the data path.



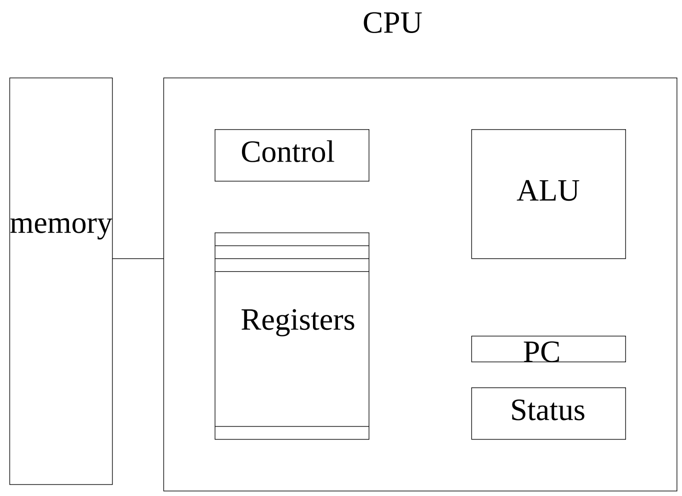
Events occur in one cycle.

The same storage element can be read and then written in the same cycle



Logic needs several cycles to propagate values

Processor design: data path and control unit



Data path:

- Combinational (ALU) + Sequential (Registers + PC + Status)
- How does data flow between these components
- What operations can be performed when data flows
- What work can be done in one cycle

Control unit:

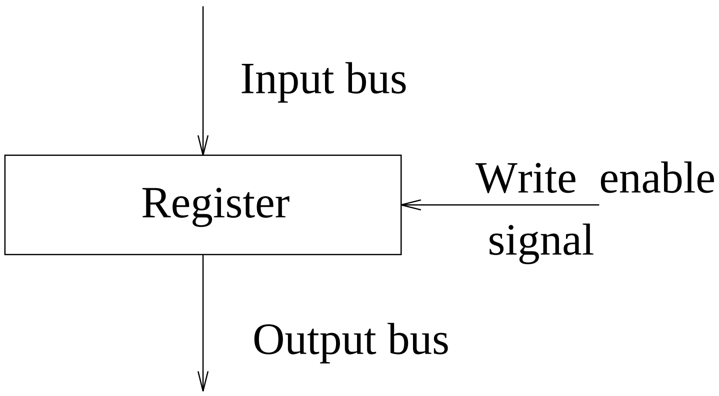
- Sends signals to data path elements
- tells what data to move, where to move it, what operations to be performed

Data path building blocks. Storage elements

The basic building block is a register.

In our mini-MIPS implementation, a register will store 32 bits.

A register can be read or written.

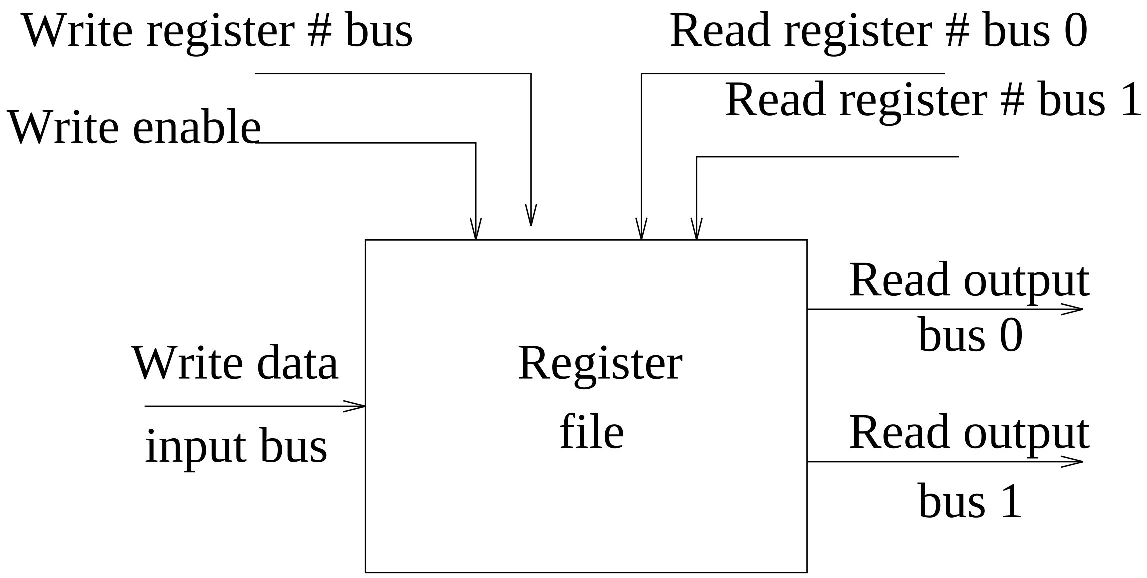


Data path building blocks. Register file.

Register file is an array of registers (32 in mini-MIPS)

ISA tells us that we should be able to read 2 registers and write 1 register in a given instruction (and right now we want one instruction per cycle).

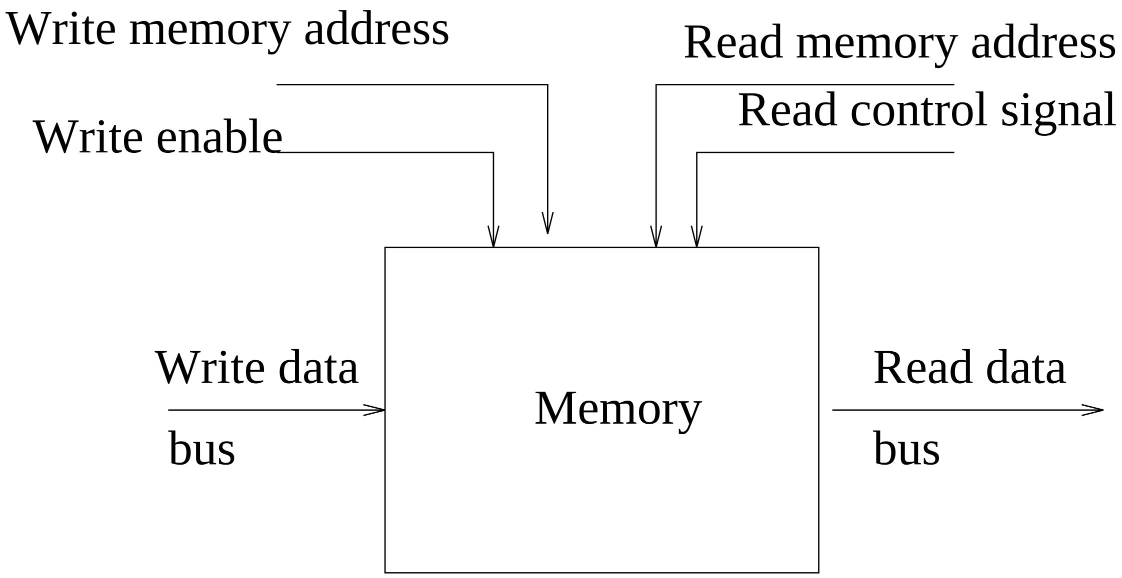
We need to know which registers to read/write



Memory

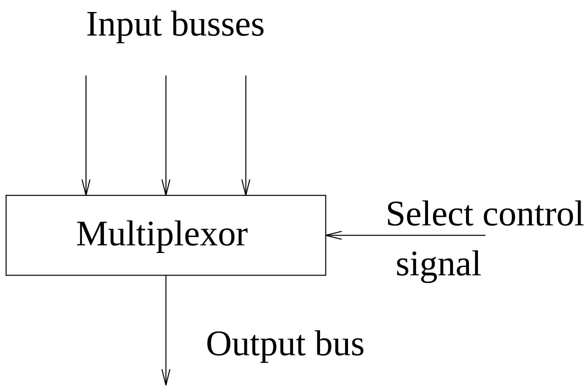
Memory is like a register file but much larger.

Can only read one location or write one location per cycle.



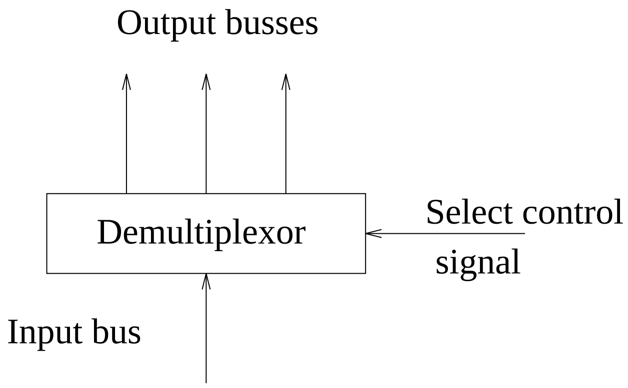
Combinational elements – Multiplexor

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



Combinational elements – Demultiplexor

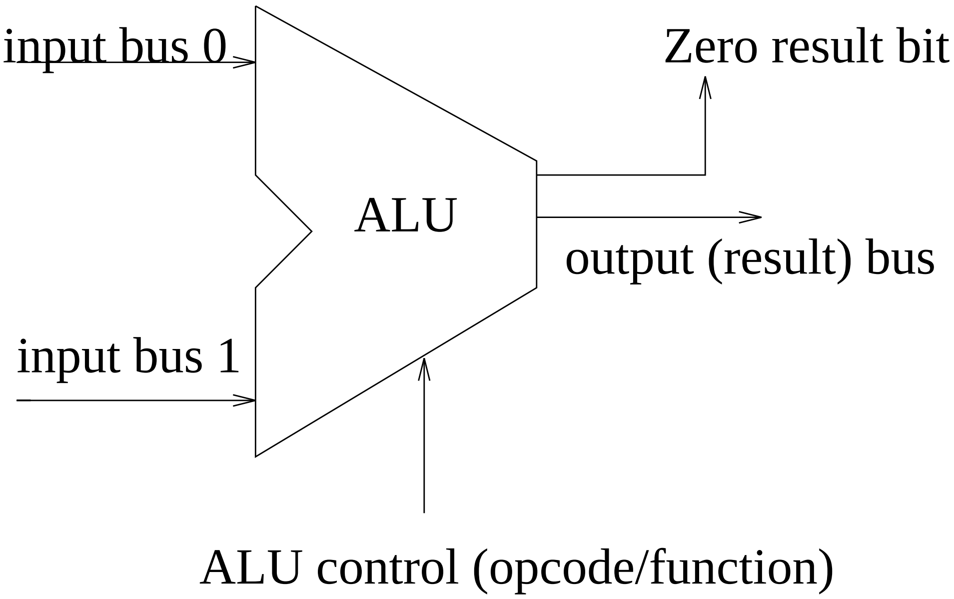
Demultiplexor (or selector) routes its input to one of its outputs



Arithmetic and Logic Unit (ALU)

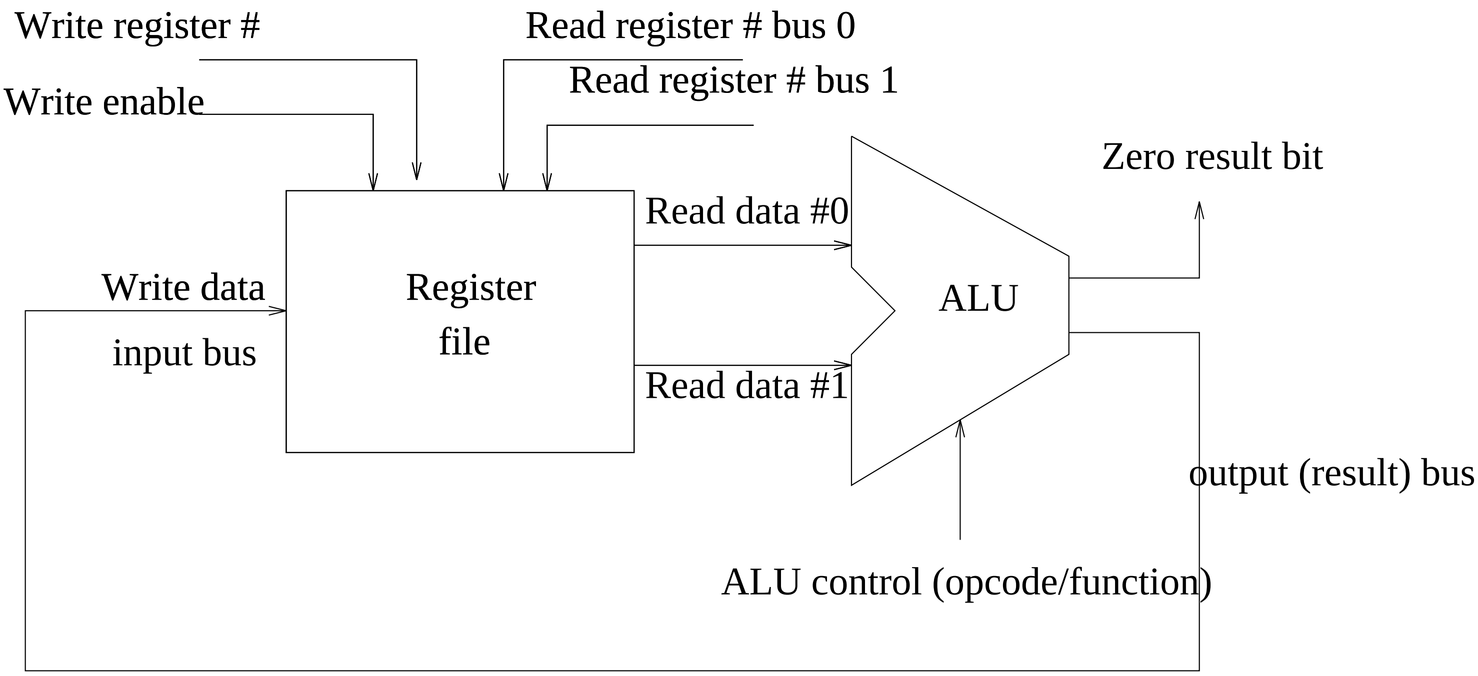
ALU computes (combinational) output from its two inputs

As its name implies does arithmetic and logic functions

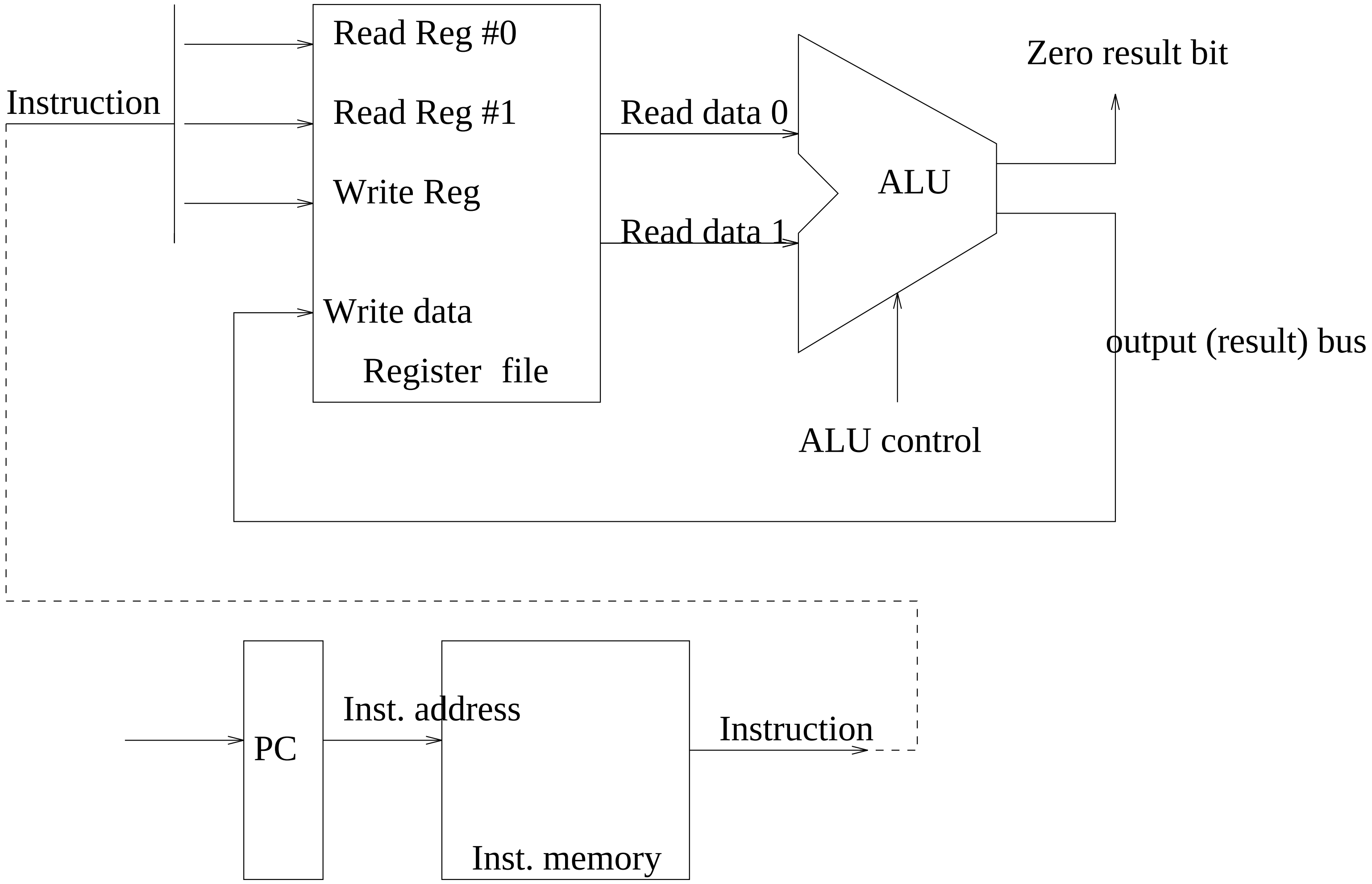


Putting basic elements together

Skeleton of data path for arithmetic operations



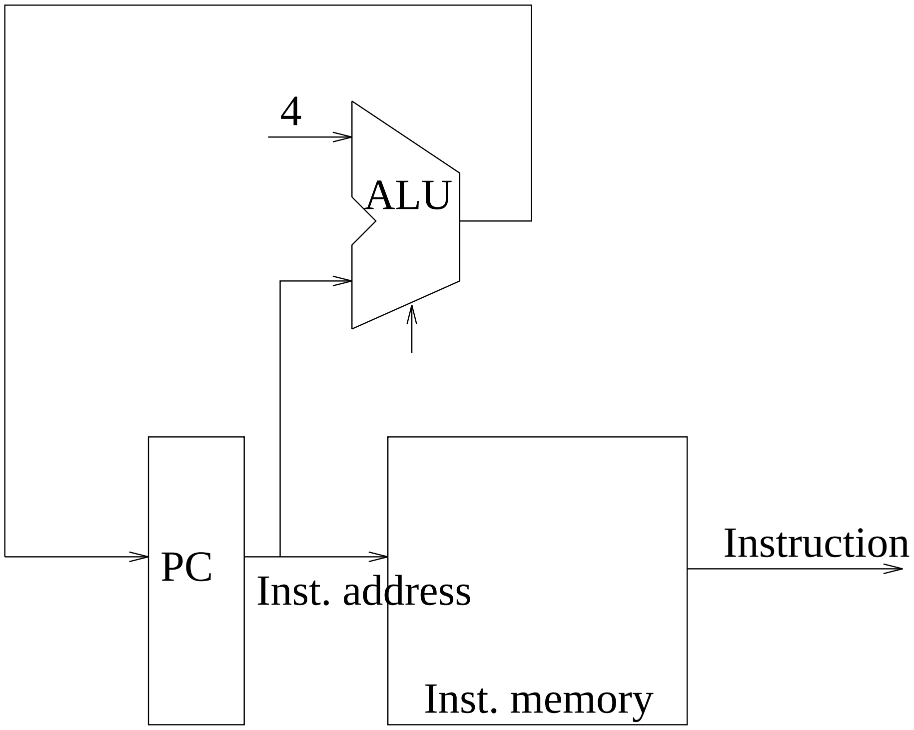
Introducing the instruction fetch



(Control signals omitted)

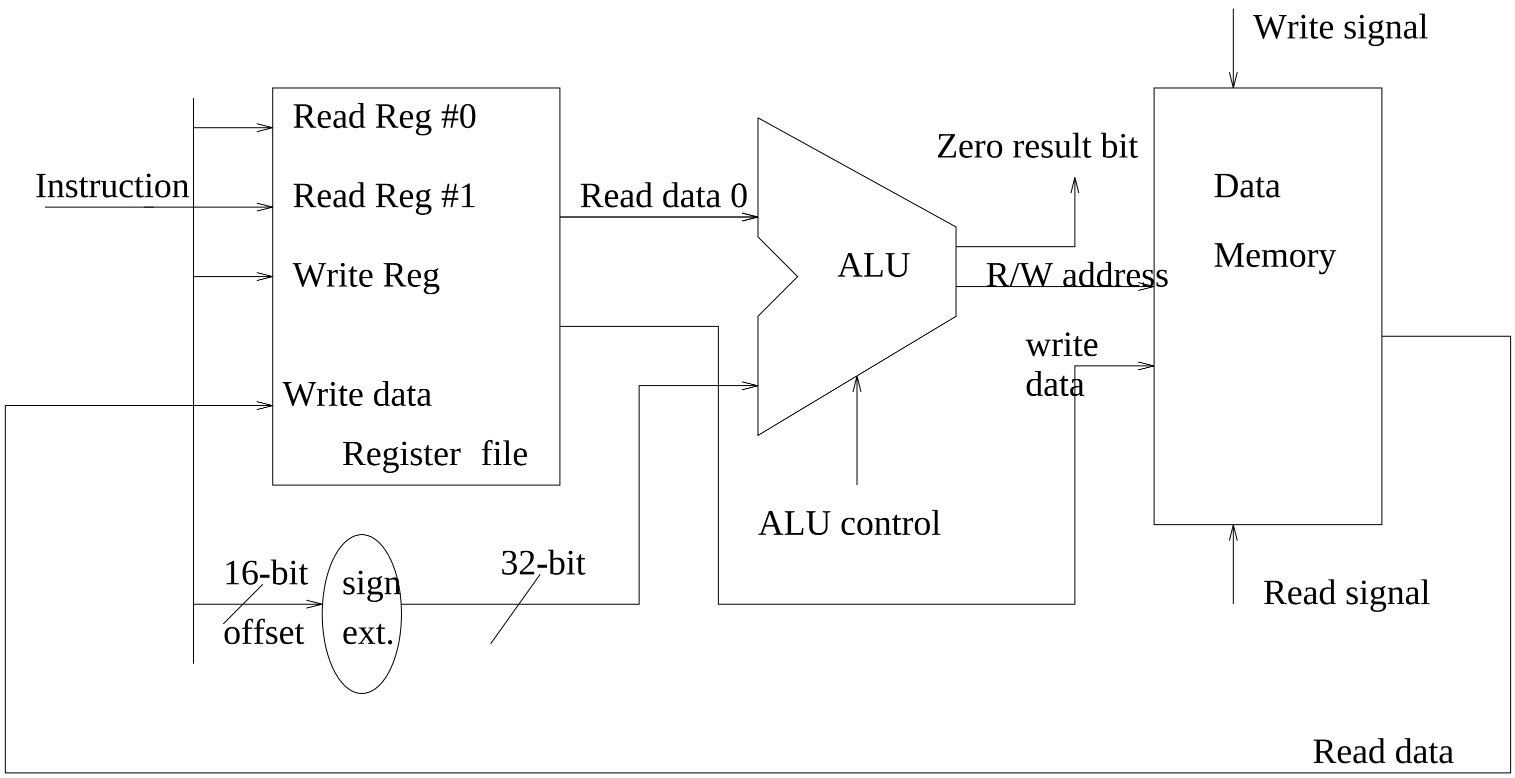
PC has to be incremented

Incrementing the PC (assume no branch)



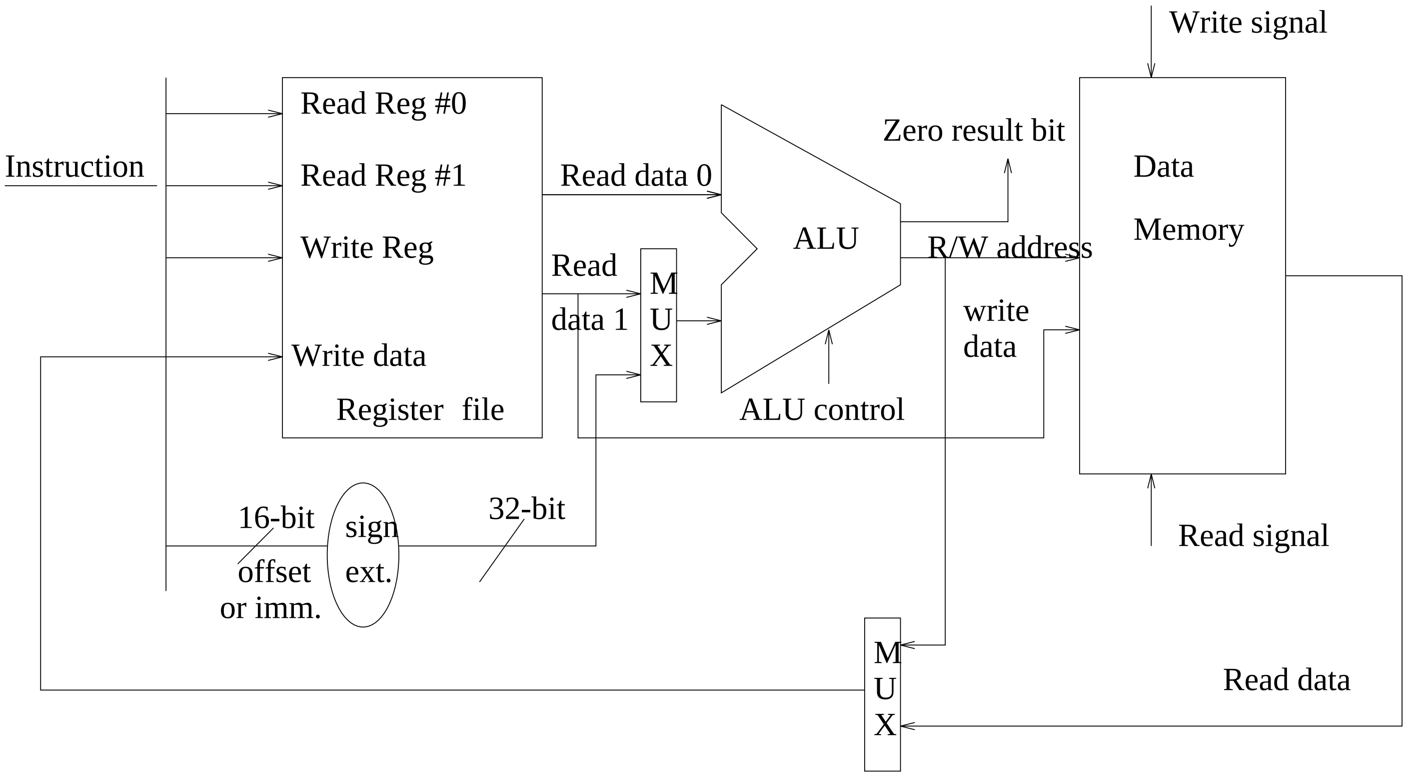
(Control signals omitted)

Load-Store instructions



Datapath for straight code

Register-register, immediate and load-store



Branch datapath