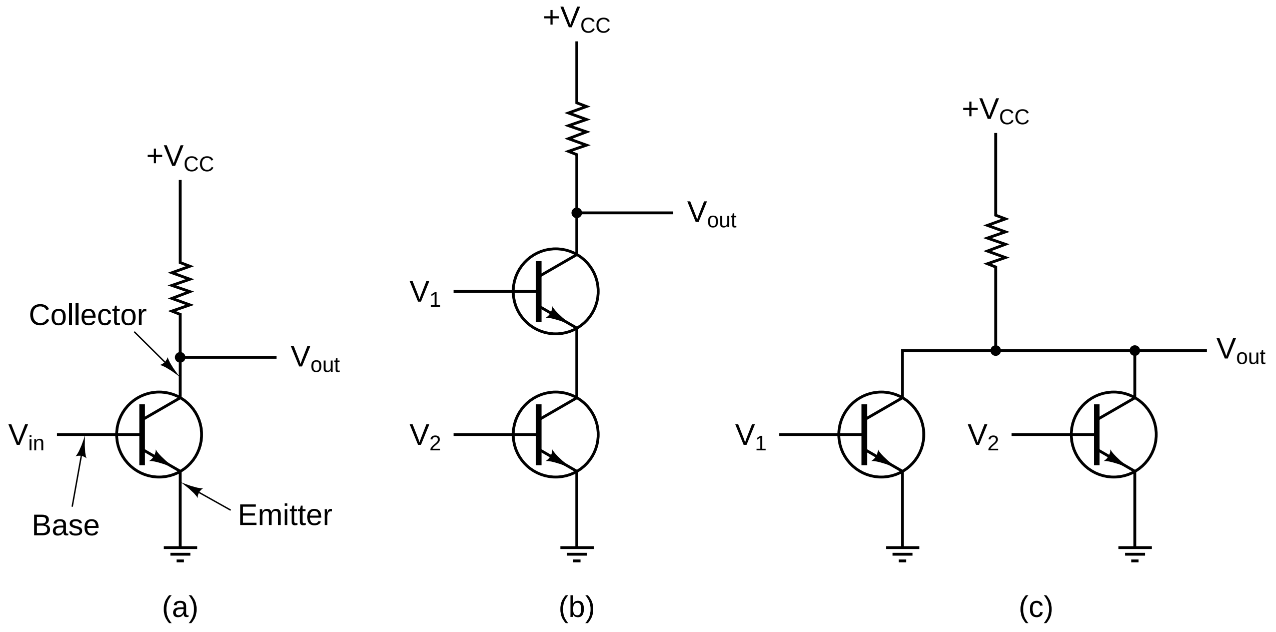
Today's Topics

Today we are going to focus on processor implementation. So we will need a bit of revision then we will will try and understand what is going in processor design.

- Digital logic (revision), combinatorial circuits
- Latches
- Clock cycles, sequential circuits
- Data paths and simple CPU design
- Clock speed and the longest path.
- Microcode
- Risc vs. Cisc.

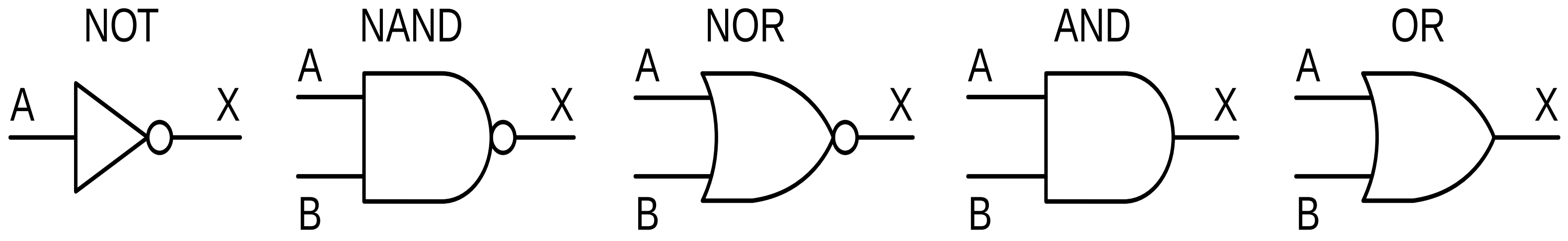
Boolean Logic

- A digital circuit is where only two voltage levels (or ranges) matter. Typically between 0 and 1 volts represents level 0 and between 2 and 5 volts represents 1.



Transistors, Switches and Gates

- A transistor (within digital electronics) acts as swtich. If V_{in} is high then there is a circuit between the collector and the emitter.
- Using transistors the following gates can be built



A	X
0	1
1	0

(a)

A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

(b)

A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

(c)

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

(d)

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

(e)

Combinatorial functions

- Any combinatorial function can be realised by combinations of the simple logical gates on the previous slide.
- In fact you can do every thing with a NAND gate.

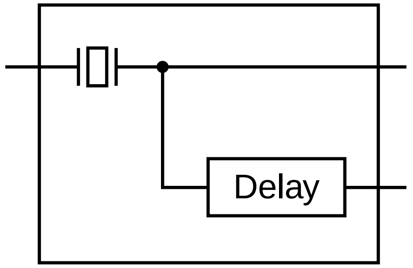
Additional reading Implentation of Boolean functions 3.1.3., 3.1.4, circuit equivalence

Clocks

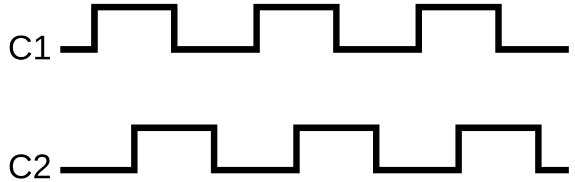
- Just building things out of gates makes things very difficult. You can't (unless you are very clever) make things happen in certain orders.
- A *clock* is simply a circuit that emits a series of plues with a precise width and interval.
- The time interval between the corresponding edges of two consective pules is known as the *clock cycle time*
- Often many things may happen during a clock cycle.
- A common way of providing finer resolution than the basic clock is to tap the primary clock and instert a delay circuit thus generating a phase-shifted secondary clock.

Clocks

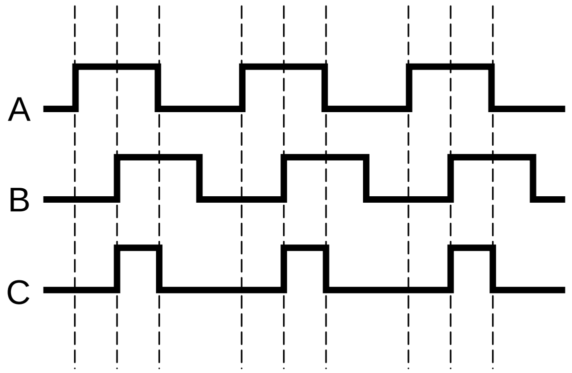
For example 4 events can be provided: rising and falling edge of C1 and the rising and falling edge of C2.



(a)



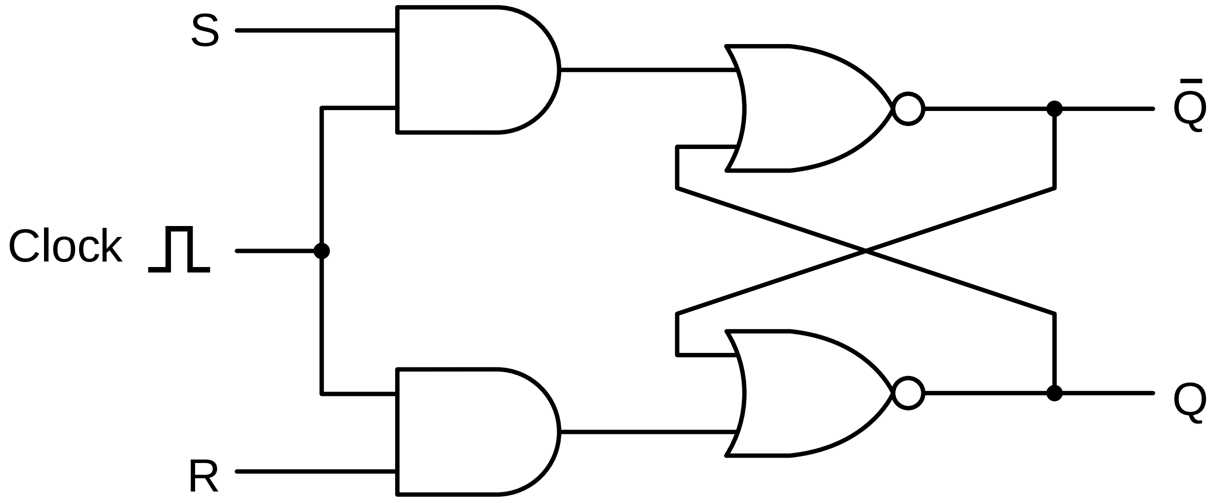
(b)



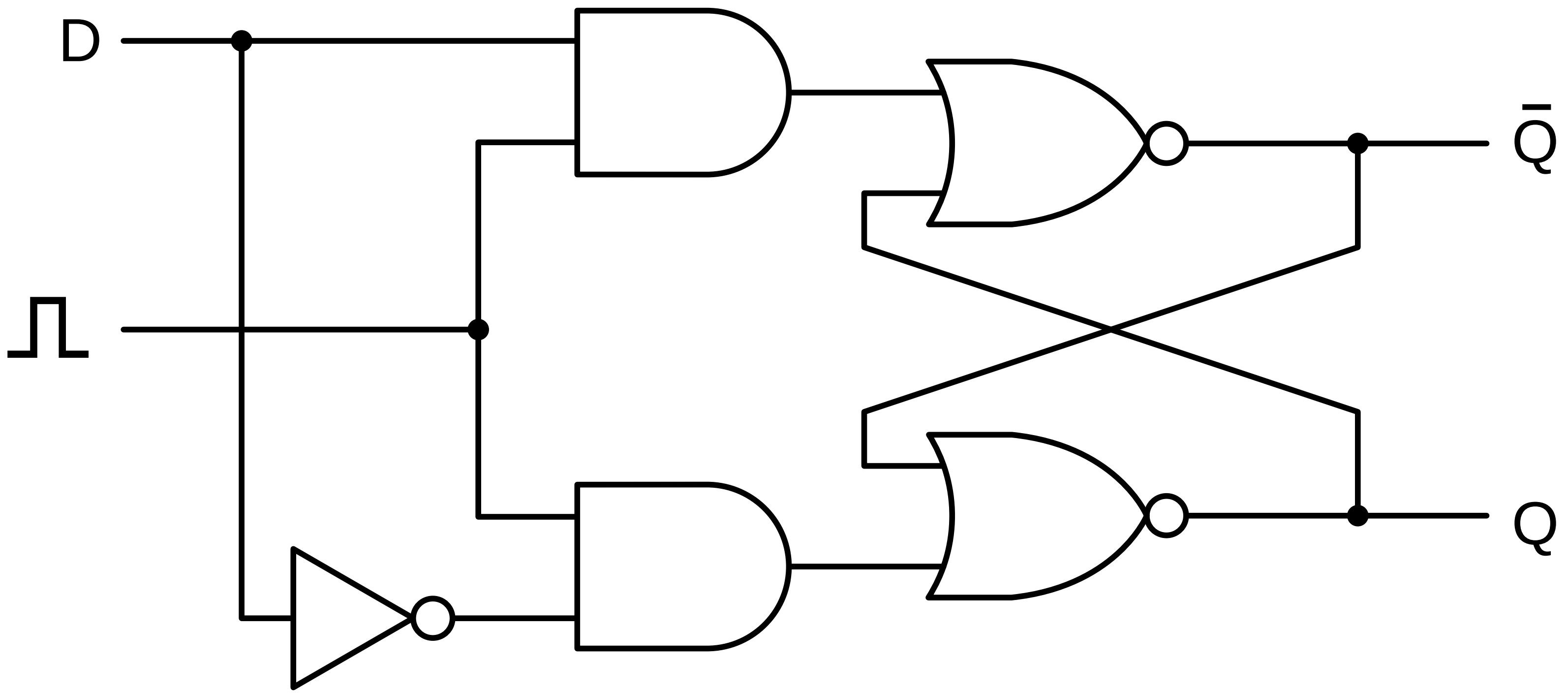
(c)

Latches

- When designing a sequential circuit, split the computation up into phases. Store the result of each phase in a latch of or a flip-flop.
- There are various flavours of latches.
- Essentially latches store data via feedback.



D-latch



If the clock is high then $Q = D$.

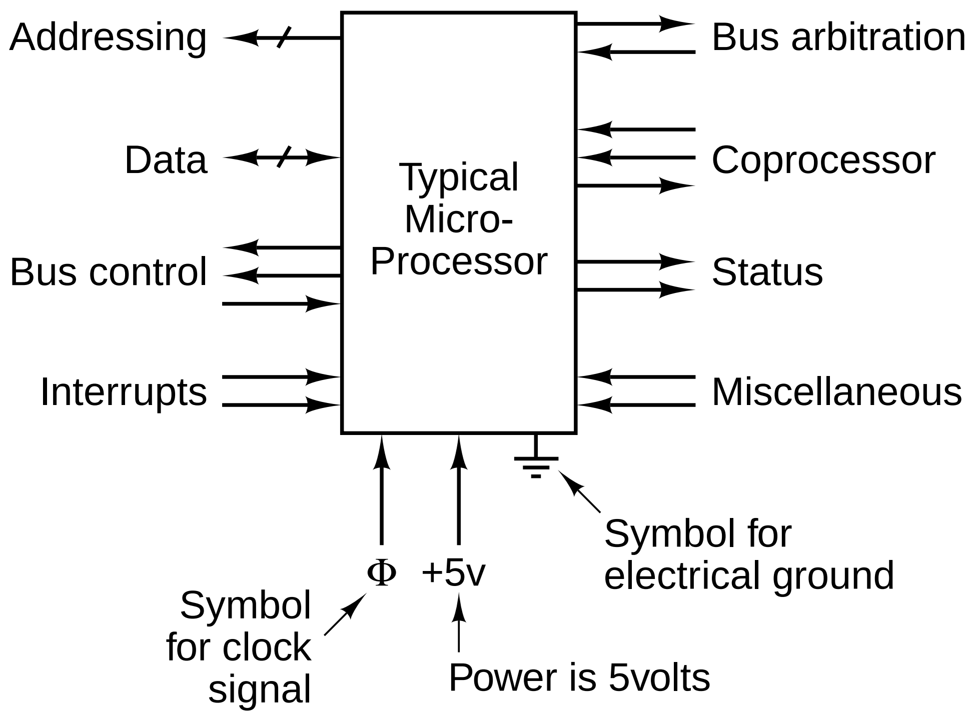
Flip-flops and Latches

- Latches are *level triggered* output chages when the level goes high.
- Flip-flops are edge triggered.
- Registers and memory can be built using flip-flops.

Longest path and clock speed

- Suppose the output of a combintorial circuit (such as an adder) is connected to a latch or a flip-flop.
- The result of the computation had to be completed before the end of the clock cycle. Otherwise it won't get stored in a latch.
- The time a circuit takes to complete a computation is proporational to the longest path between an input and an output.
- Hence we have to have the clock cycle longer the longest path in the circuit.
- Typically some tolerance is built in. Which is why you can overclock a bit, but why you can't over clock too much.

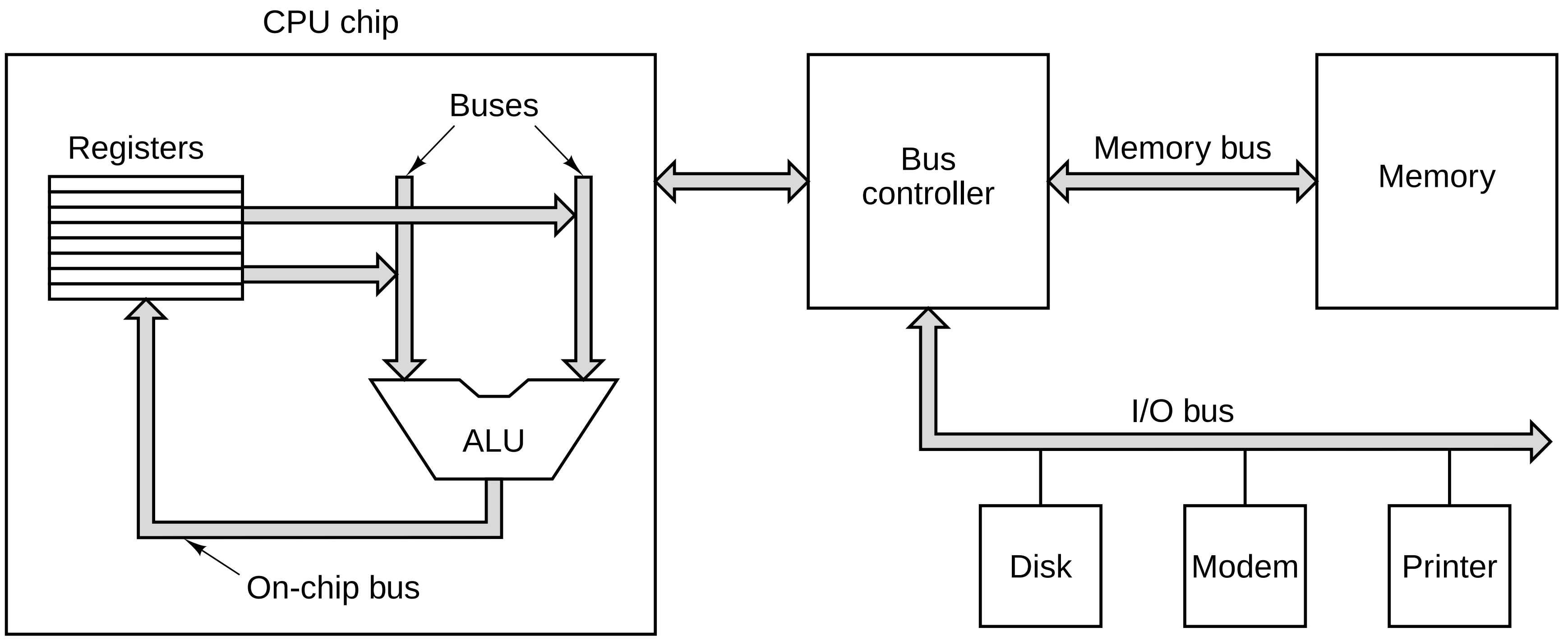
CPU Chips and Buses



Typical processor pin-out.

Computer Buses

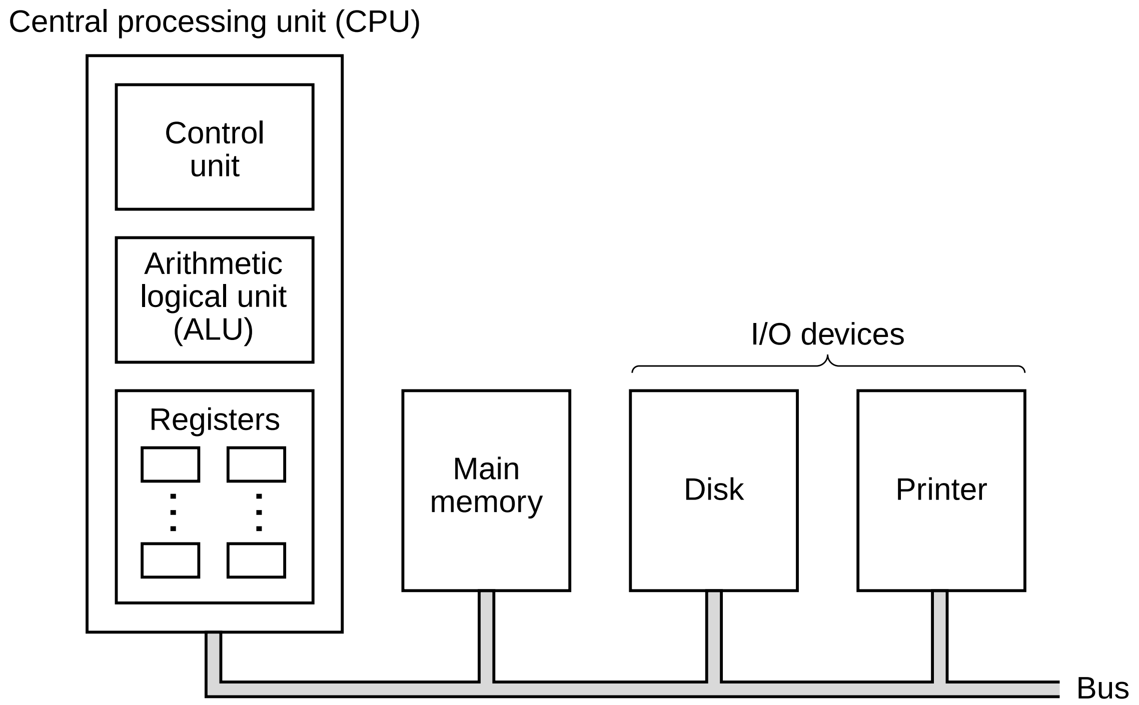
- A *bus* is a common electrical pathway between multiple devices.



Computer Buses

- If more than one devices tries to use the bus at the same time then chaos can happen.
- Hence rules have be defined refered to as the *bus protocol*.
- Active devices on buses which can initiate bus transfers referred to as *masters* the passive ones referred to as *slaves*.
- When the CPU instructs the disk controller to read from the disk, the CPU is the master and the disk controller is the slave. Later when the disk controller instructs the memory to receive some date then the memory is the slave and the disk controller is the master.
- Memory can never be the master.

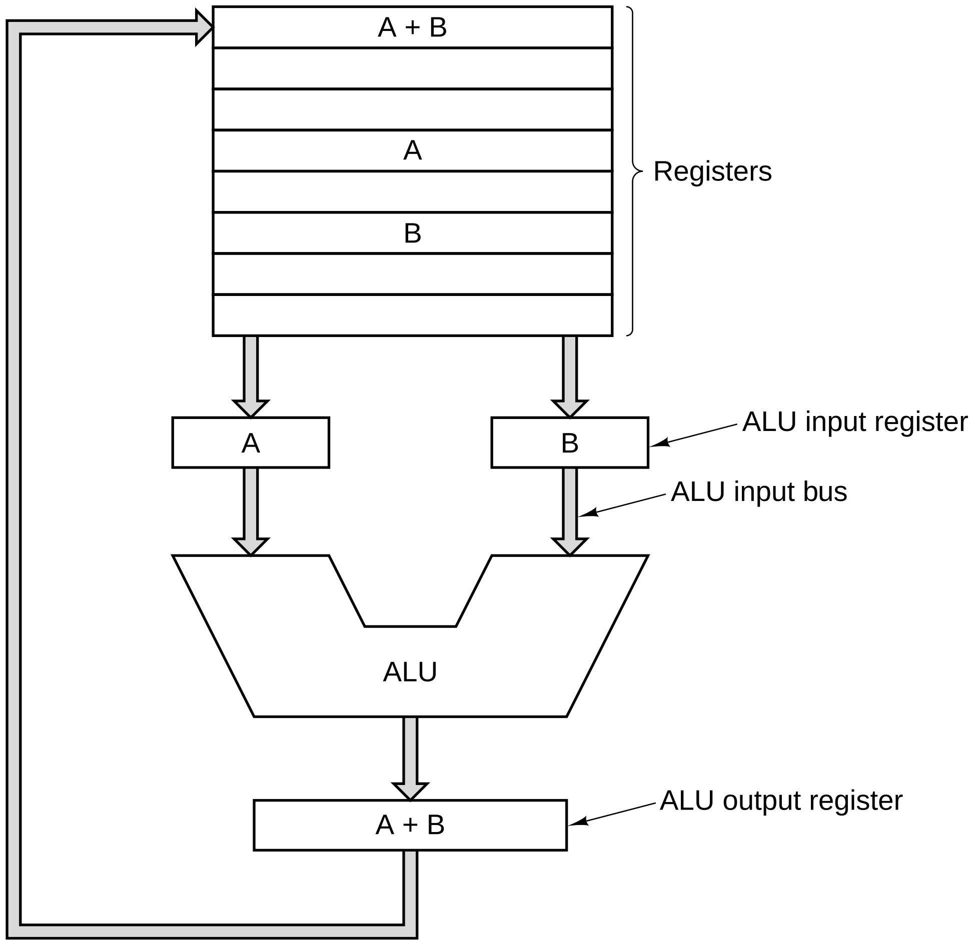
CPU Organization



Types of instructions

- Register $\leftrightarrow$ register
- Register $\leftrightarrow$ memory

Register ↔ register datapath cycle



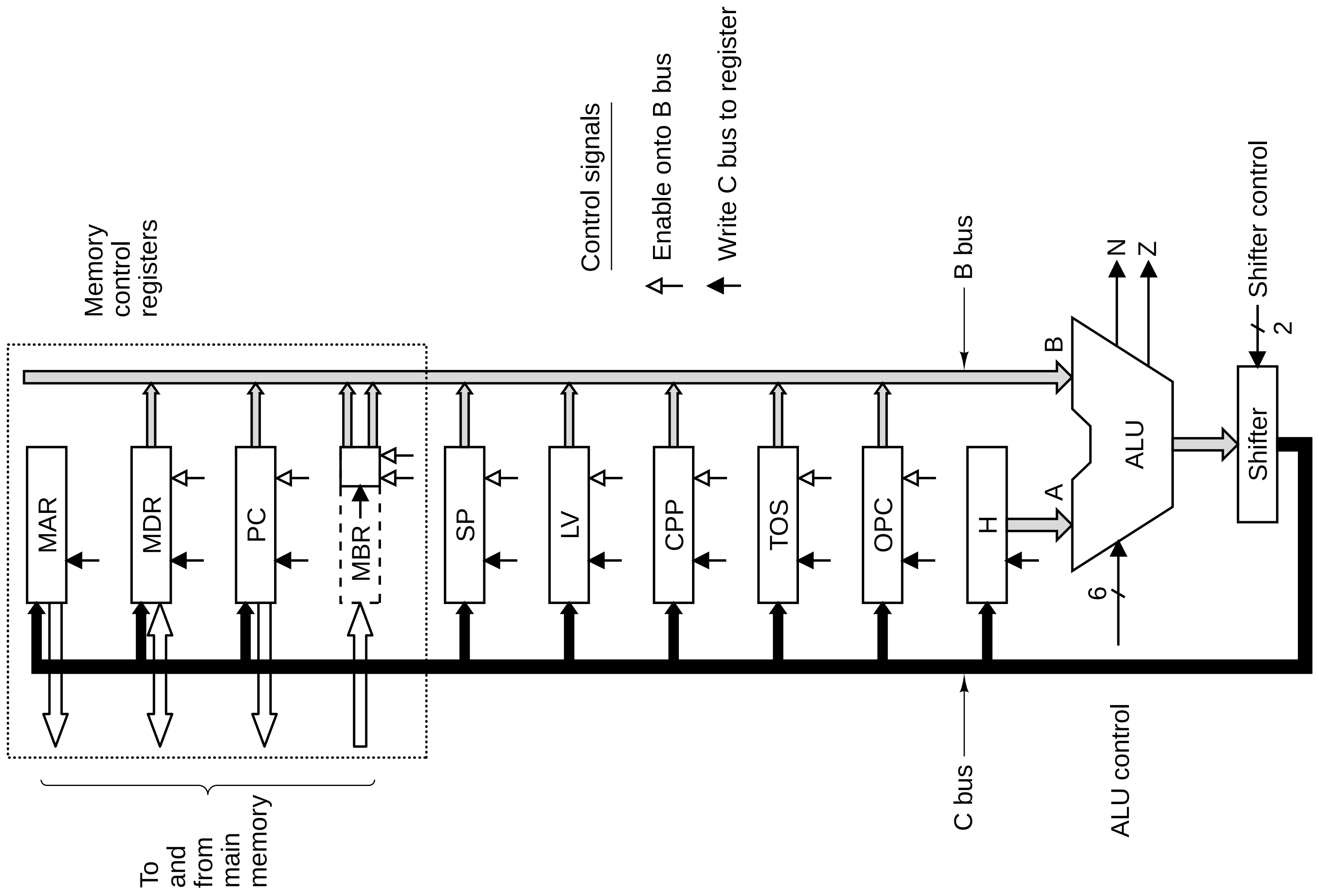
Instruction Execution

Fetch-execute decode cycle.

- Fetch the next instruction from memory (use the program counter register)
- Increment the program counter
- Determine the type of the instruction just fetched.
- If the insruction uses data from memory determine where it is.
- Fetch data from memory
- Execute the instructions
- Goback to step 1.

Various posssibilities, each step takes a clock cycle, or (less efficient why?) do everyhthing in one clock cycle.

A more complicated datapath



Datapath control

- Key idea :- Each unit is either enabled or disabled via a control signal.
- For example the ALU takes two agruuments to add togther two registers and but the result into another register then enable the adder unit in the ALU.
- The instruction decoder then simply works out which units to enable on the datapath.

Datapath Control - Strategies

- Microcode :-
  - Have lots of simpler instructions (micro instructions). Split each instruction up into microinstructions.
  - Decoder then finds a list of micro-instructions for each high-level instruction.
  - Requires micro-code memory, a micro-program counter.
- Advantages
  - Same instructions can be implemented on different architectures.
  - Can change the micro-code. Can be easier to debug.
- Disadvantage
  - Extra layer between instructions and hardware.

RISC philosophy

- Decode instructions directly to control signals. No microcode.
- Faster simpler instructions, closer to the hardware.
- In contrast with CISC no extra internal decoding is needed.