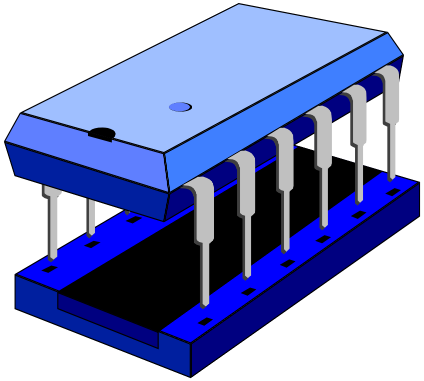
BASIC CIRCUITS AND MEMORY



Anandha Gopalan (with thanks to **N. Dulay** and **E. Edwards**)

axgopala@imperial.ac.uk

Digital Circuits

- Basic Circuits
  - Half Adder
  - Full Adder
  - Latches

Adders

- A digital circuit that performs addition of numbers
- Not only used in arithmetic logic unit(s), but also in other parts of the processor, where they are used to calculate addresses, table indices, and similar operations
- Most common adders operate on binary numbers

Half Adder

- Consider adding two 1-bit binary numbers together:

	0	0	1	1
+	0	1	0	1
	00	01	01	10

- Input – 2 separate lines
- Output – two bits – how do we represent this?
  - Use two separate lines (Sum and Carry)

Half Adder

- Can we now draw the circuit?
  - What do we need? – Truth Tables
    - One each for sum and carry

Half Adder

- Recall

	0	0	1	1
+	0	1	0	1
	00	01	01	10

- Truth Table

A	B	A + B	Sum	Carry
0	0	0	0	0
0	1	1	1	0
1	0	1	1	0
1	1	2	0	1

What do these correspond to?

Half Adder

- Selecting Gates

Sum	Carry
0	0
1	0
1	0
0	1

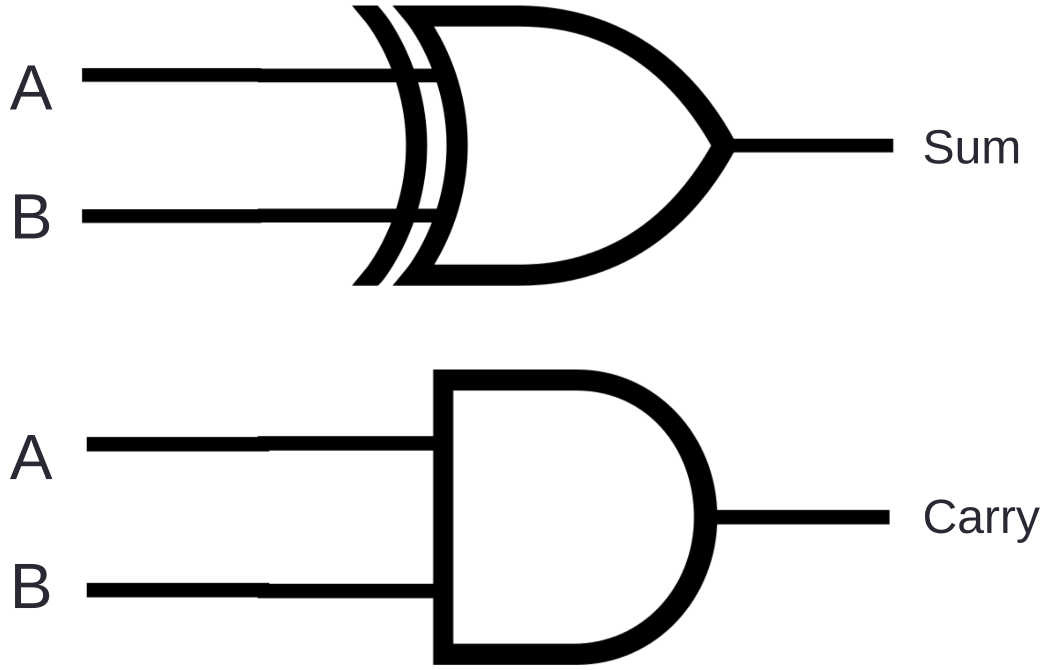
$\Leftrightarrow$

XOR	And
0	0
1	0
1	0
0	1

- Hence, we can build the expressions as:
  - Sum = $A \oplus B$
  - Carry = $A \bullet B$

Half Adder

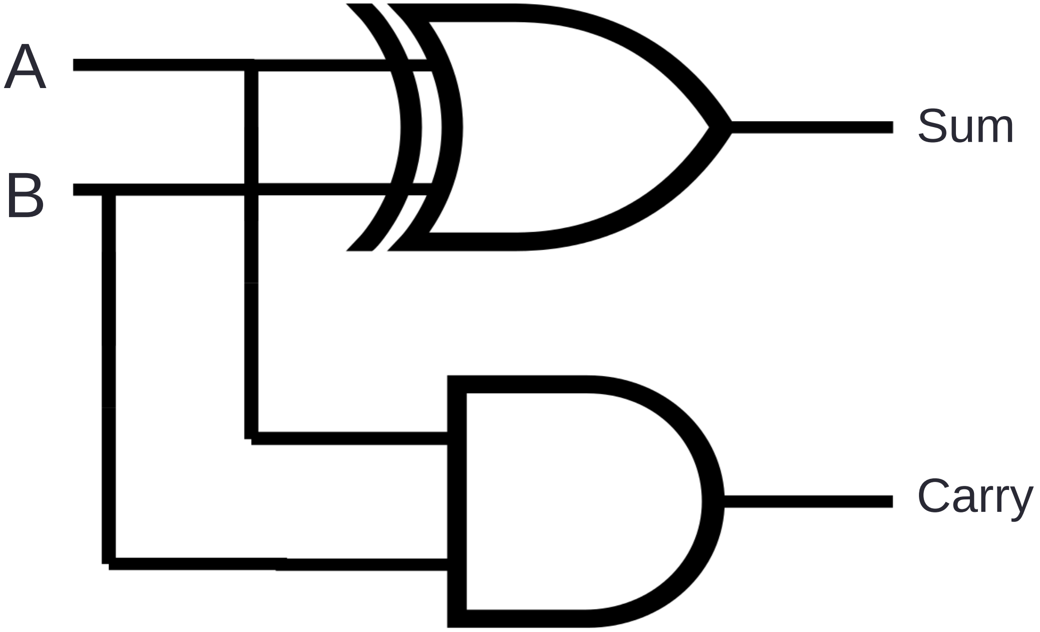
- Circuit



Is this Correct?

Half Adder

- A more concise and better version ☺



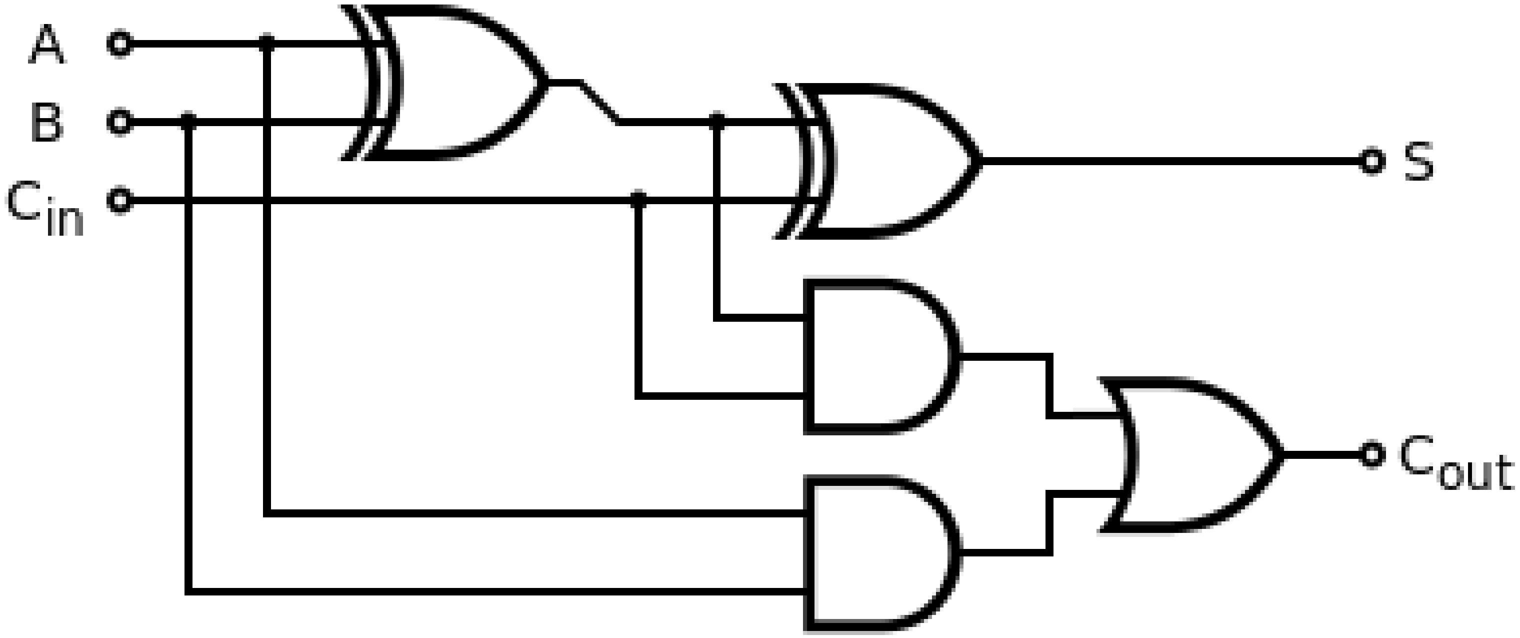
Full Adder

- Half-adders have a major limitation
  - Cannot accept a carry bit from a previous stage ➔ they cannot be chained together to add multi-bit numbers

- Full-adders can accept three bits as input
  - Third bit is the carry-in bit

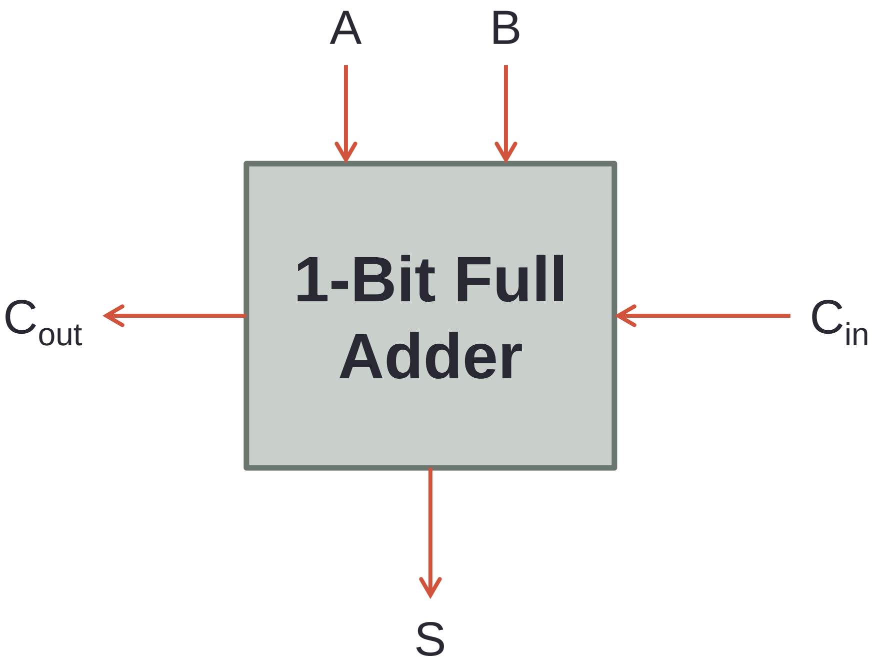
- Can be cascaded to produce adders of any number of bits by daisy-chaining the carry of one output to the input of the next

Full Adder



Full Adder

- Conceptually

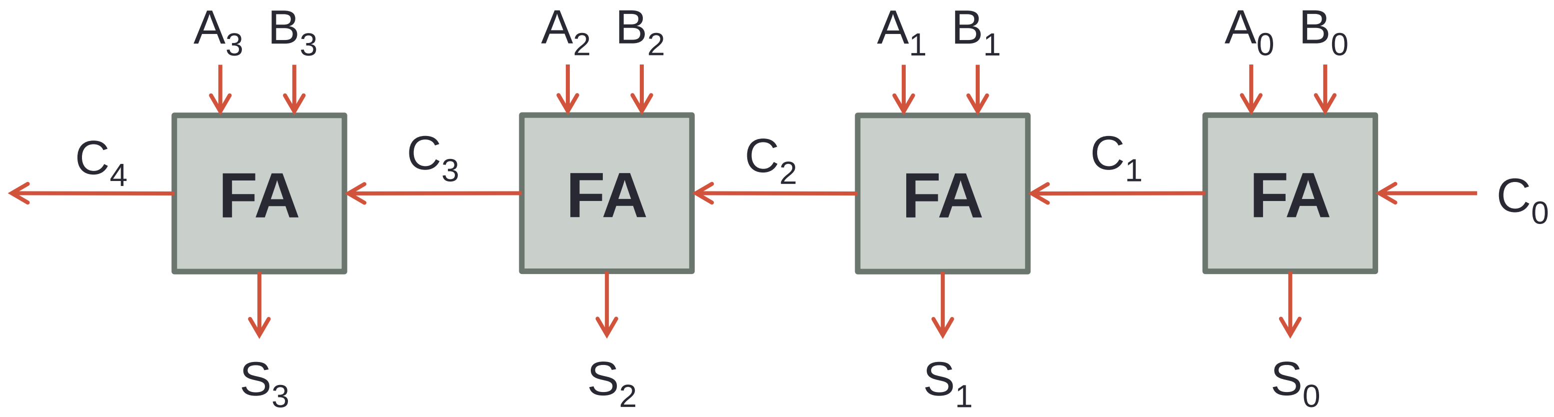


Ripple-Carry Adder

- Consists of several full adders connected in a series so that the carry must propagate through every full adder before the addition is complete
- Require the least amount of hardware of all adders, but they are the slowest
  - Carry-Lookahead Adder

Ripple-Carry Adder

- The following diagram shows a four-bit adder, which adds the numbers A and B, as well as a carry input, together to produce S and the carry output



Gates

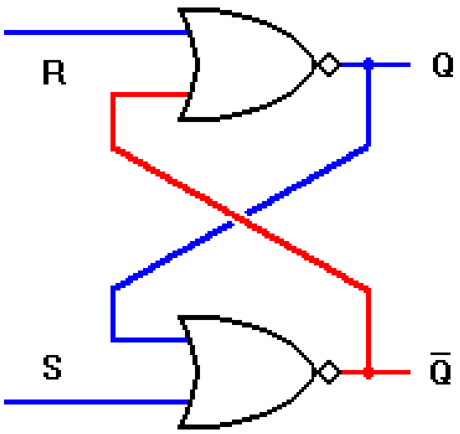
- Building blocks for combinatorial circuits
  - Output depends only on current Input
- All gates can be built out of NAND and NOR gates
- What if we would like to store values?
  - Use a *feedback* mechanism where the output values depend indirectly, on themselves

Latches

- Building blocks to sequential circuits
- Can be built from gates
- Able to remember 1-bit of information ☺
- Useful web-page
  - http://www.play-hookey.com/digital/sequential/

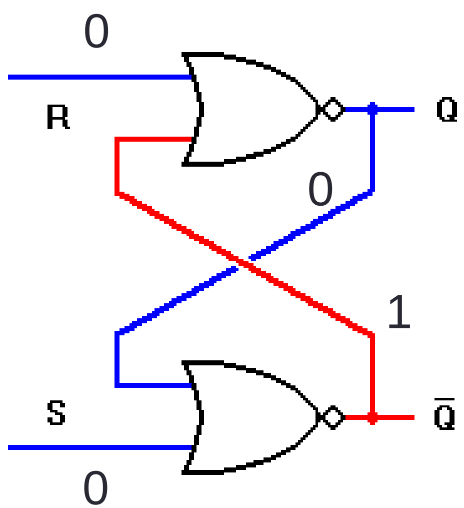
Latches

- SR-latch
  - S = Set
  - R = Reset



Latches

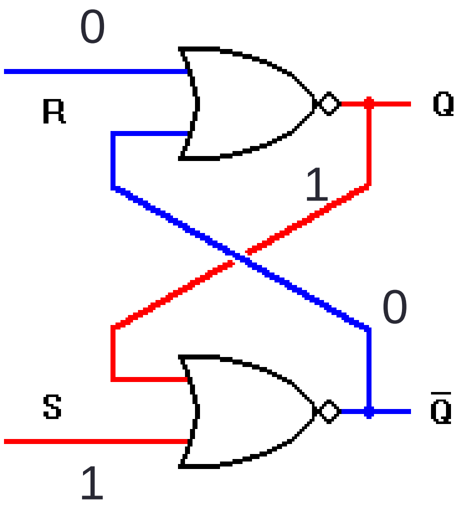
- S = 0, R = 0



- Value of Q does not change → value is ‘remembered’
  - Sometimes called the *latch* state

Latches

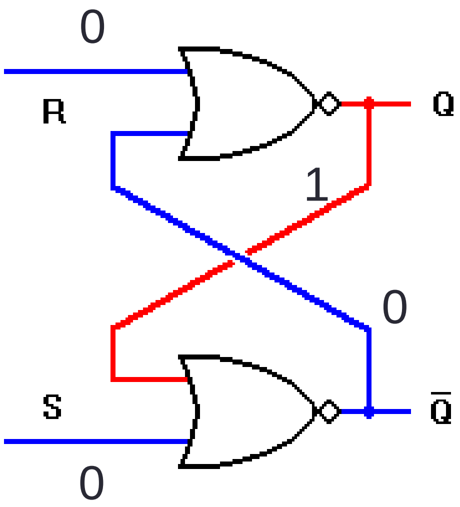
- S = 1, R = 0



- Set the value of Q

Latches

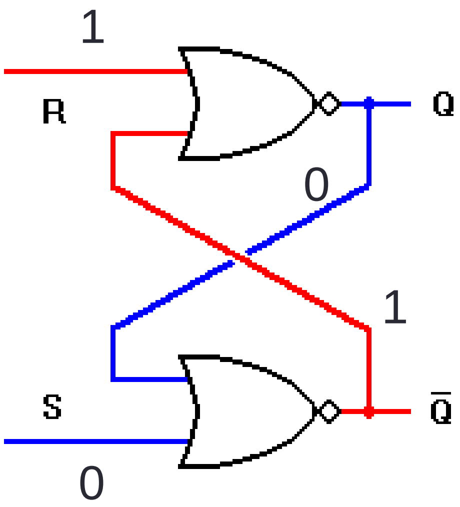
- S = 0, R = 0



- Value of Q stays the same – it ‘remembers’ ☺

Latches

- S = 0, R = 1



- Reset the value of Q to 0

- S = 1, R = 1 leads to undefined state

Latches

- SR-Latch: Truth table

S	R	Q	Q'
0	0	Latch	
0	1	0	1
1	0	1	0
1	1	Undefined	

Flip-Flops

- Latches are *asynchronous* → output changes very soon after the input changes

- Most computers today, are *synchronous*
  - Outputs of all the sequential circuits change simultaneously to the rhythm of a global *clock signal*

- A *flip-flop* is a synchronous version of the latch

Memory

- Memories hold binary values
  - Data (e.g. Integers, Reals, Characters)
  - CPU Instructions (i.e. Computer Programs)
  - Memory Addresses (“Pointers” to data or instructions)
- Contents remain unchanged unless overwritten with a new binary value
  - Some of them *lose* the content when power is turned off

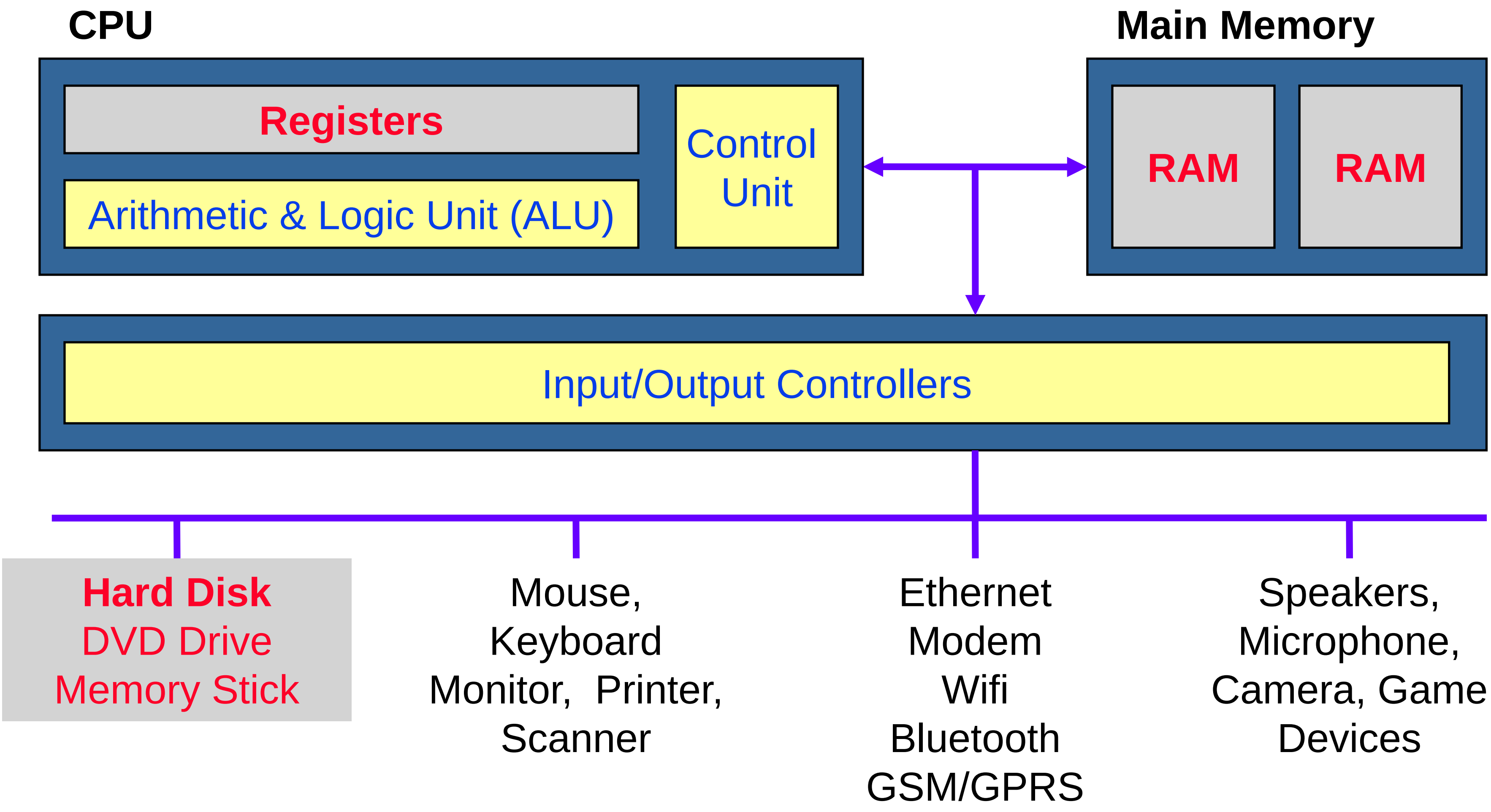
Memory – Examples

- CPU, Registers, Caches – L1, L2 [L3]

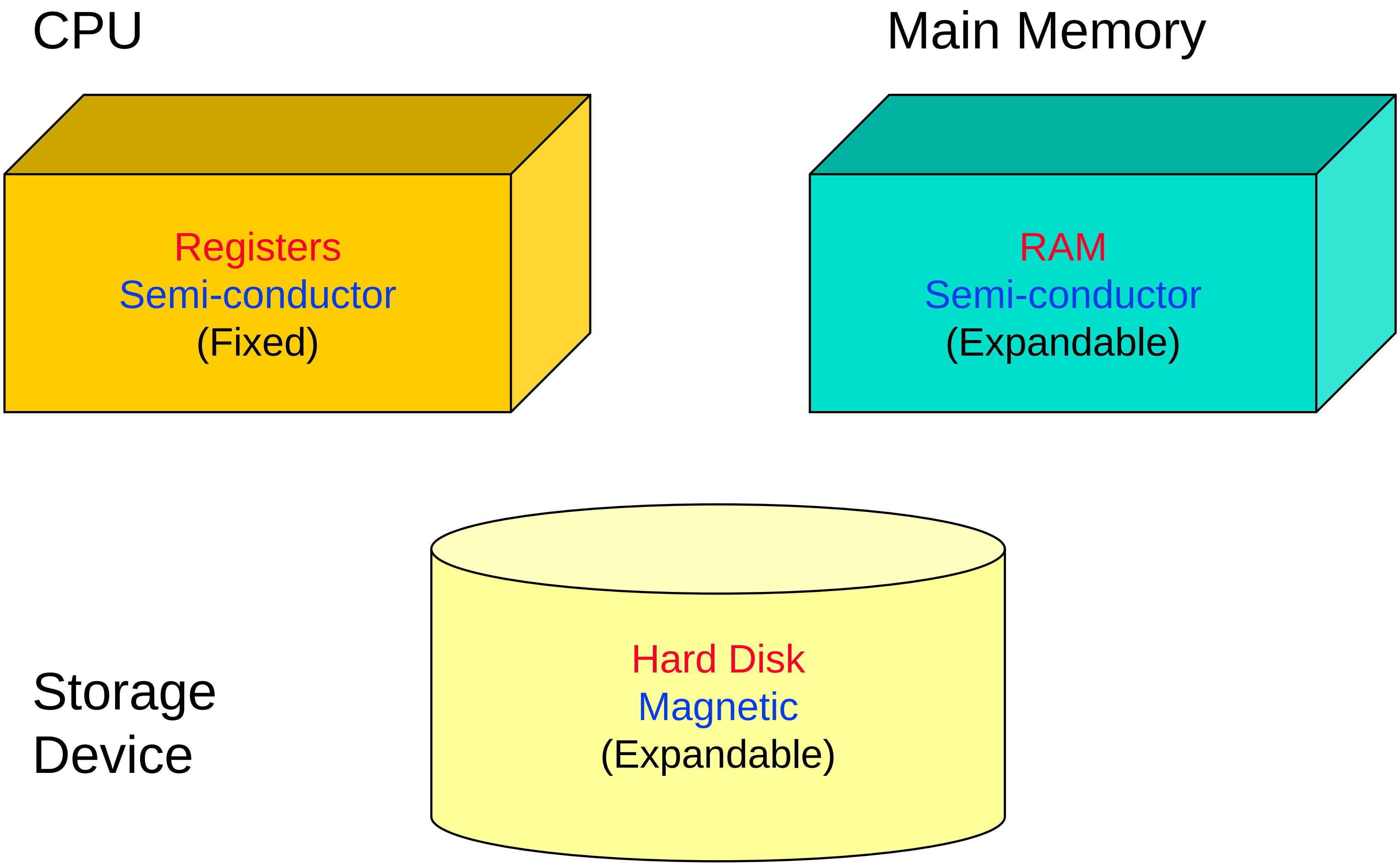
- Motherboard
  - RAM (Random Access Memory)
  - Caches
  - I/O Registers & Buffers
  - Video-card Memory

- Storage Devices
  - Hard Disks, CDs, DVDs, Tapes, Memory Sticks, Flashcards

CPU Organisation



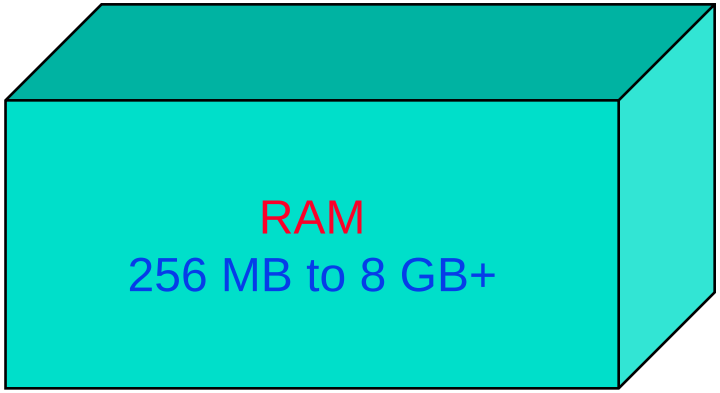
3 Types of Memory



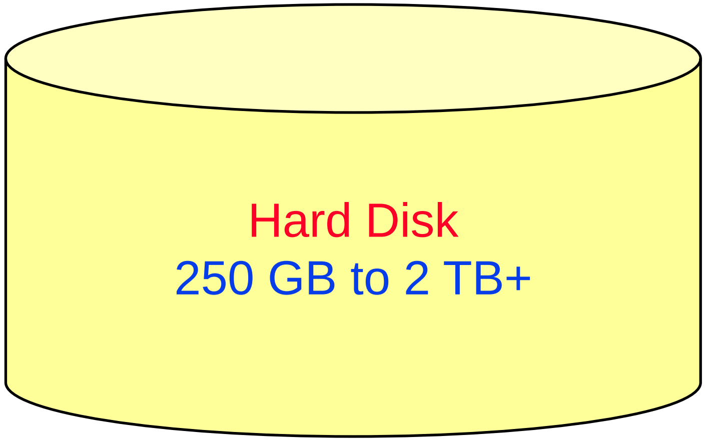
Capacity

CPU

Main Memory



Storage Device



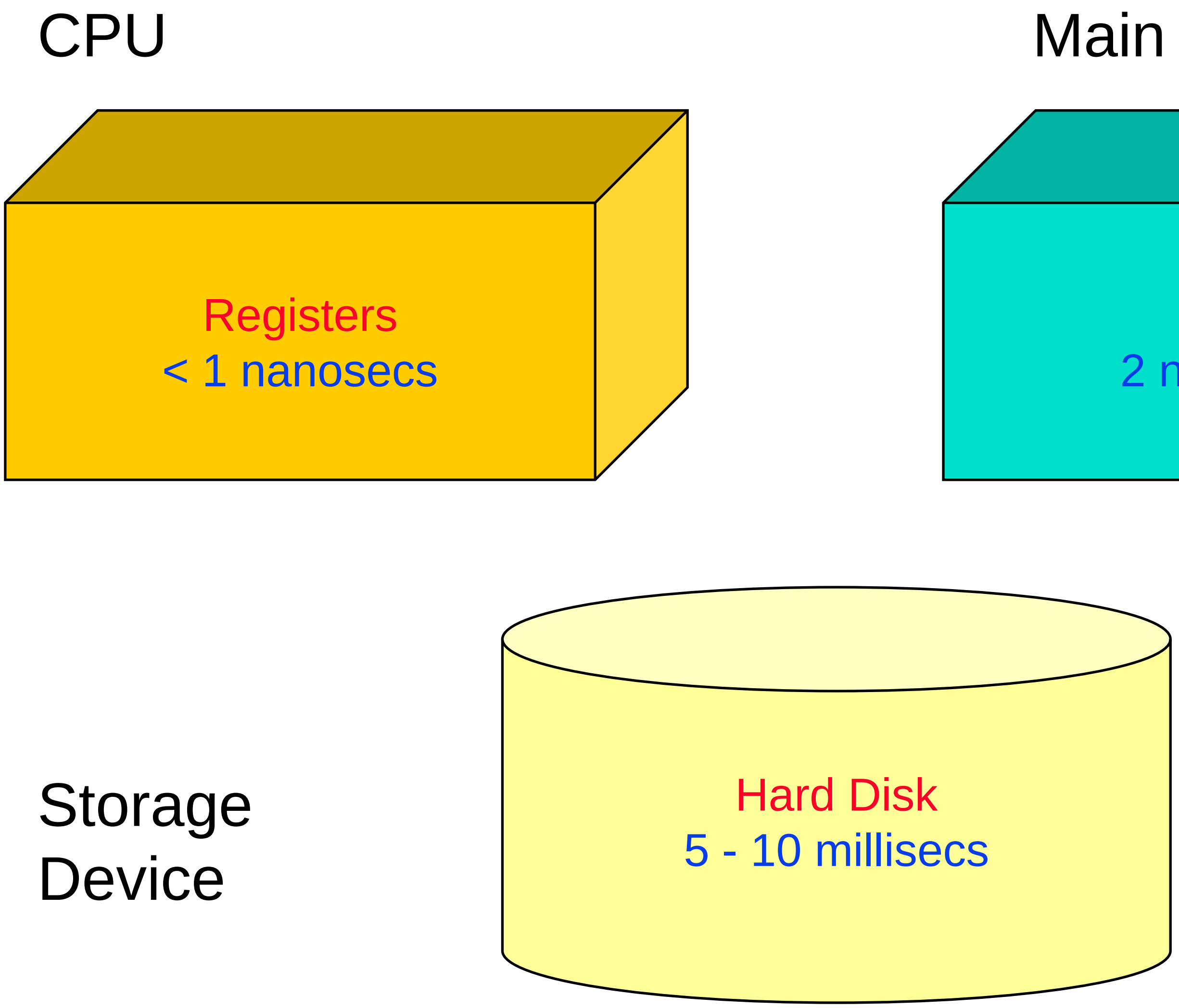
1 KB = 2^{10} bytes

1 MB = 2^{20} bytes

1 GB = 2^{30} bytes

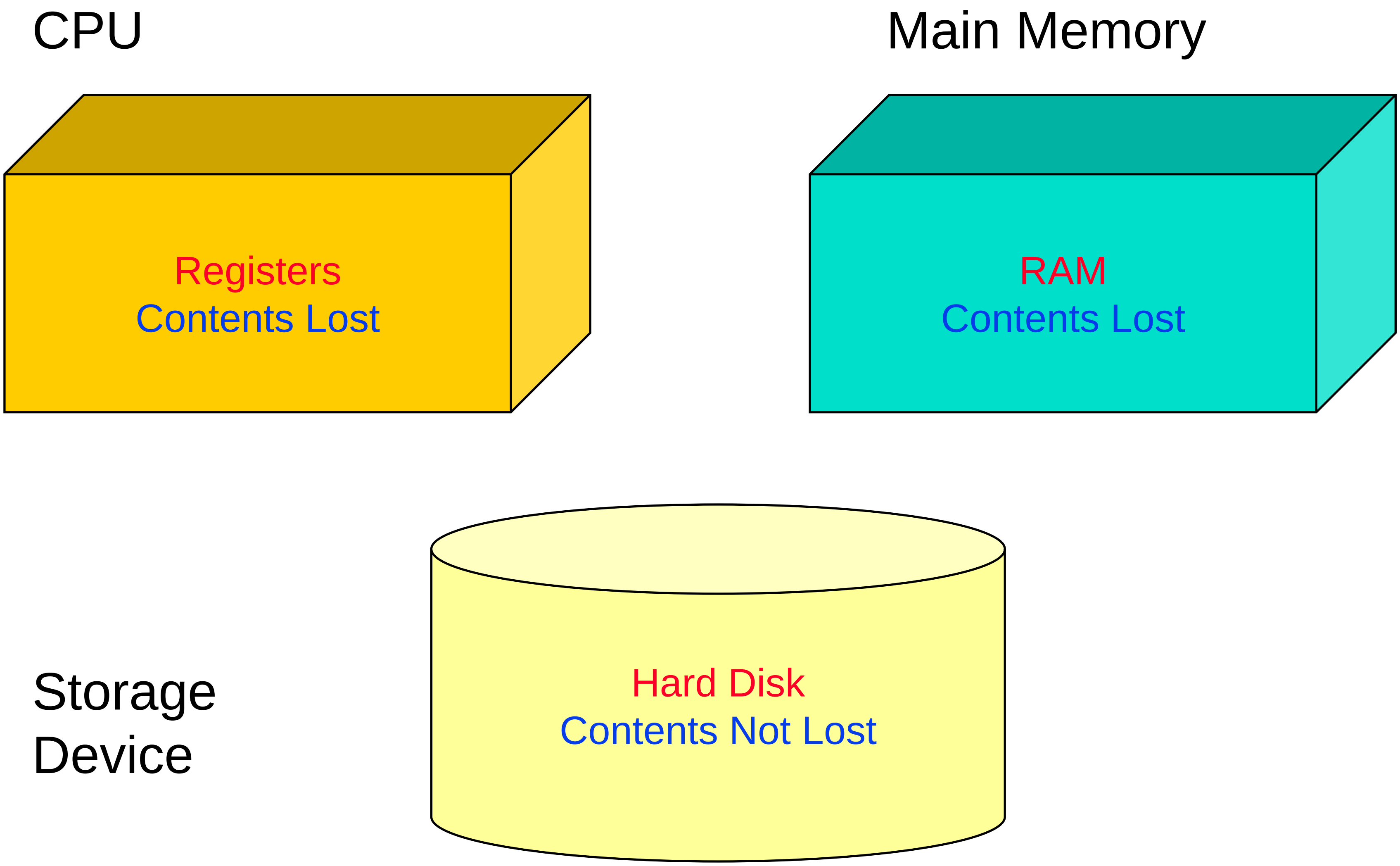
1TB = 2^{40} bytes

Speed (Access Time)

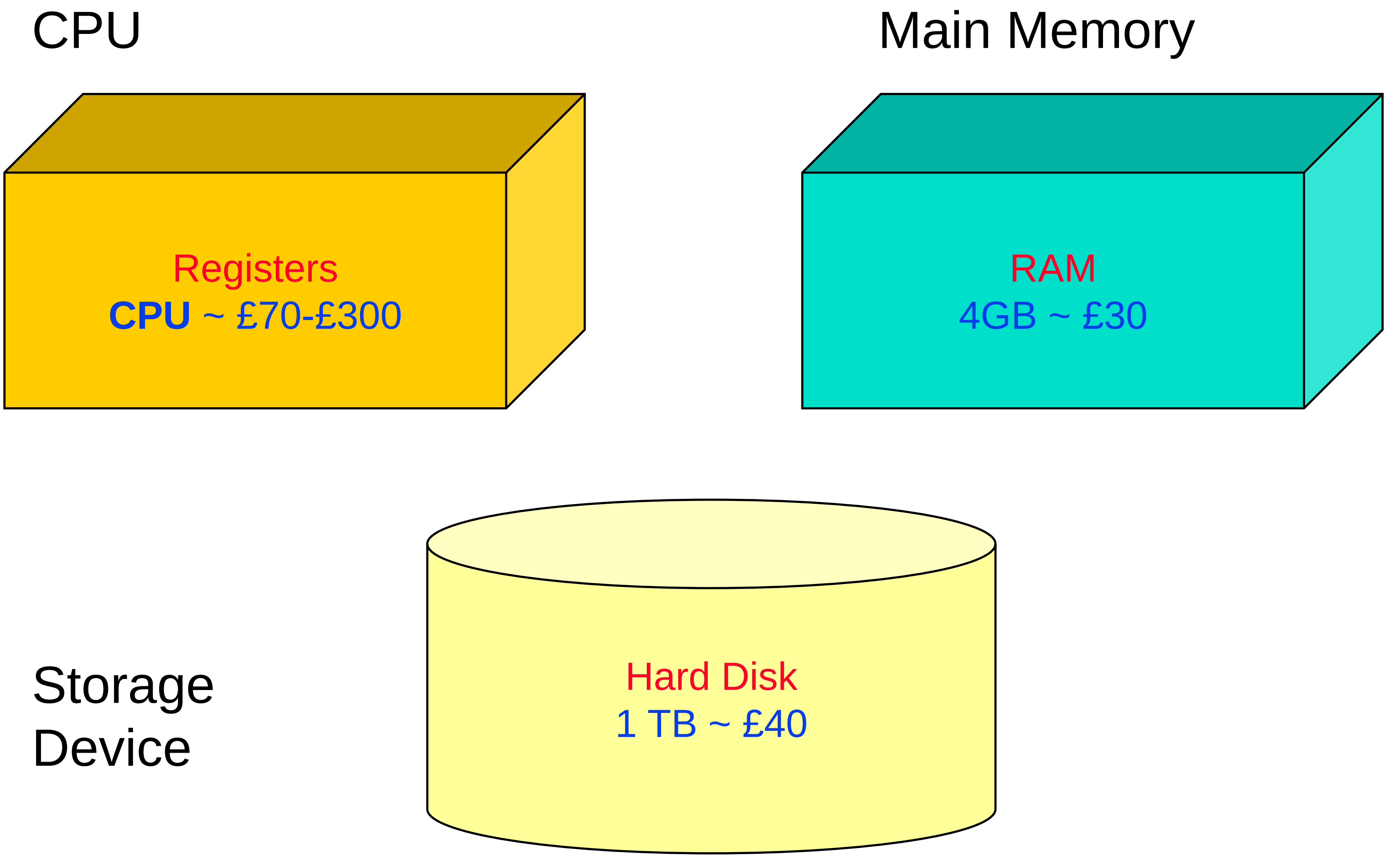


milli $= 10^{-3}$
micro $= 10^{-6}$
nano $= 10^{-9}$

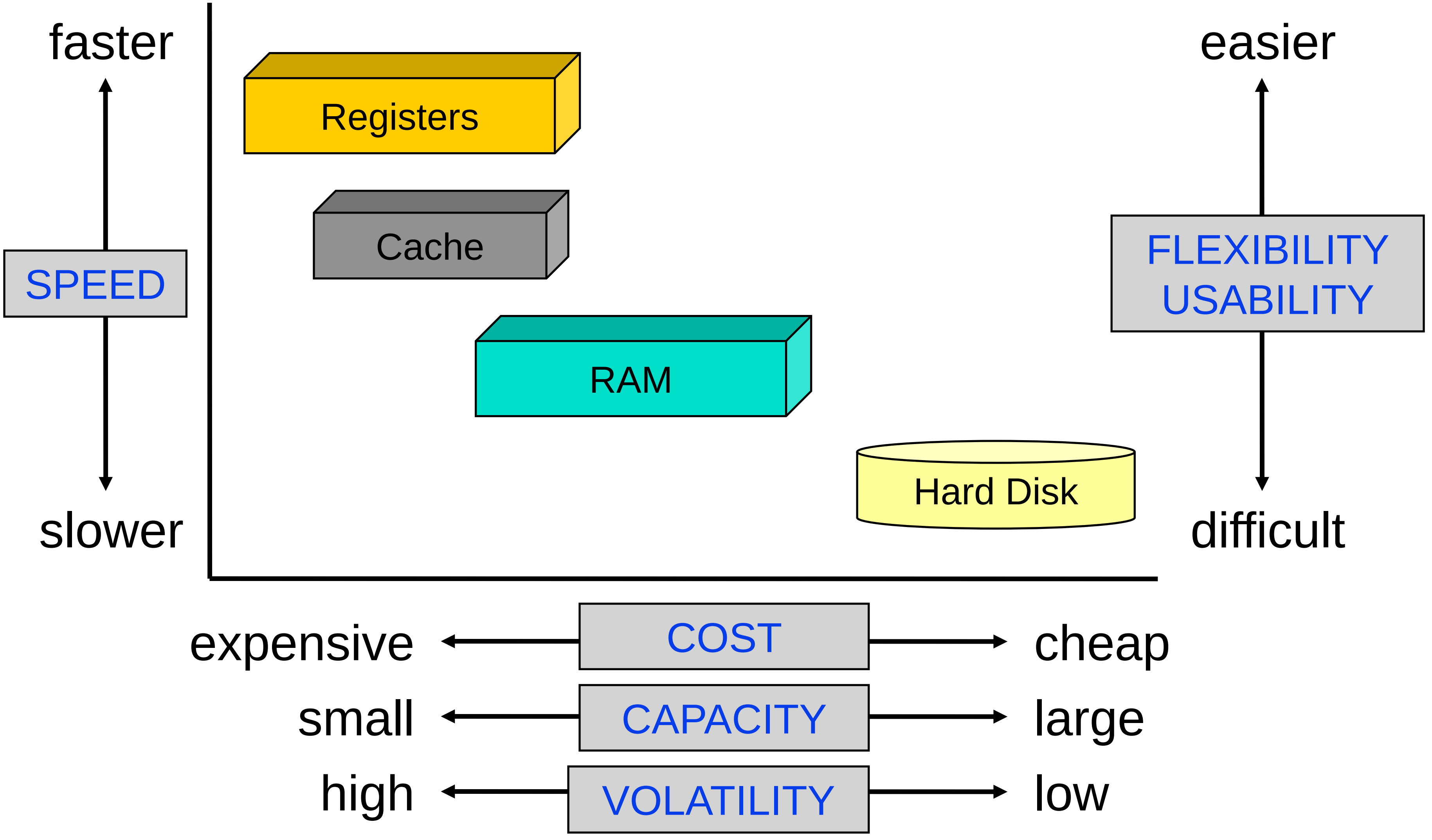
Volatility


CPU
Registers
Contents Lost
Main Memory
RAM
Contents Lost
Storage
Device
Hard Disk
Contents Not Lost

Cost


CPU
Main Memory
Registers
CPU ~ £70-£300
RAM
4GB ~ £30
Storage
Device
Hard Disk
1 TB ~ £40

Summary



faster
SPEED
slower
Registers
Cache
RAM
Hard Disk
easier
FLEXIBILITY
USABILITY
difficult
expensive
COST
cheap
small
CAPACITY
large
high
VOLATILITY
low