Memories

- Memories hold binary values. These can be:

  Data (e.g. Integers, Reals, Characters)

  CPU Instructions (i.e. Computer Programs)

  Memory Addresses (“Pointers” to (position of) data or instructions)

- The **contents** of a memory remain unchanged unless overwritten with a new binary value. For some memories the contents are "lost" when power to the memory is turned off.

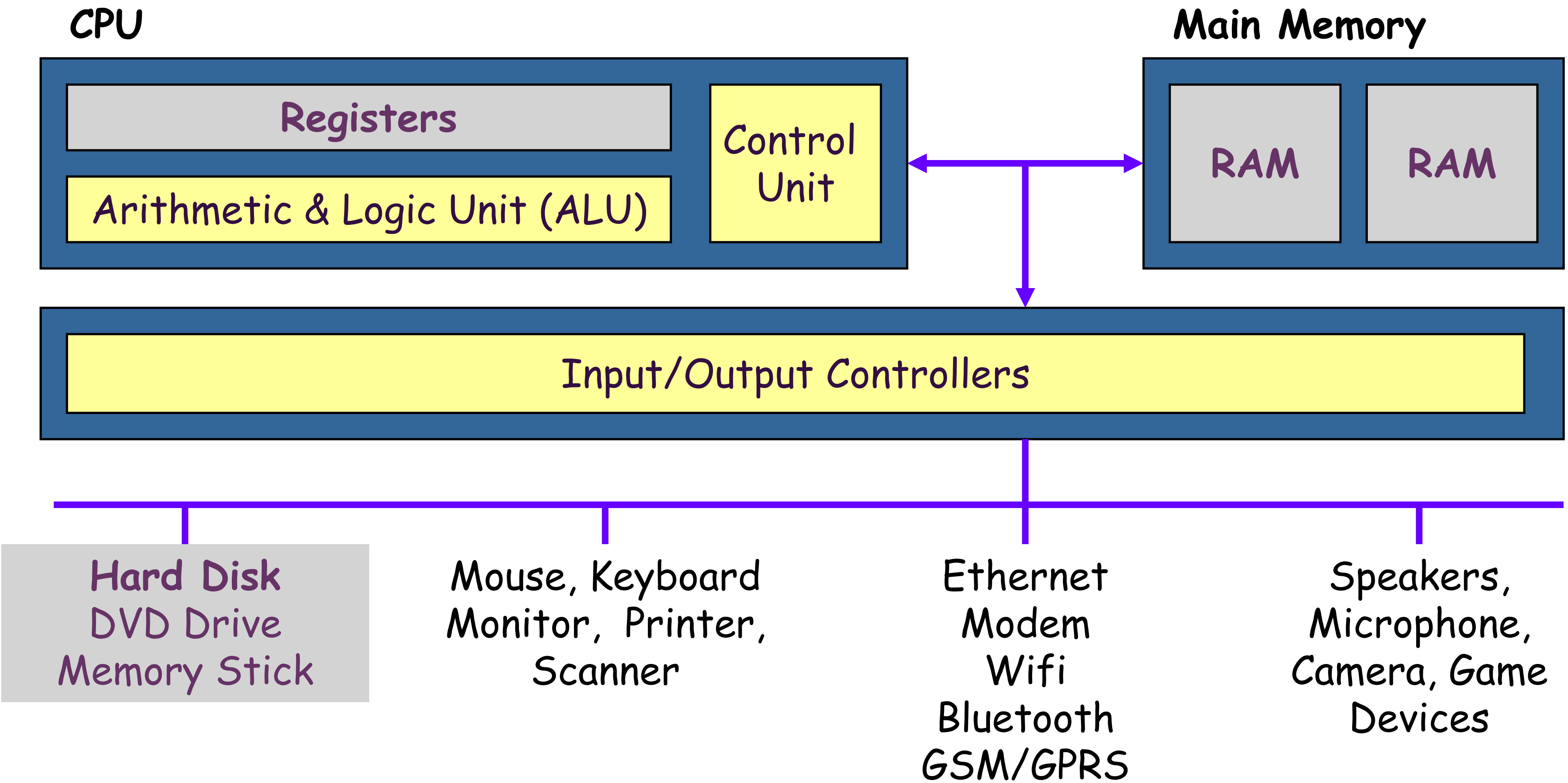
Examples of Memories

- **CPU**
  Registers
  Caches

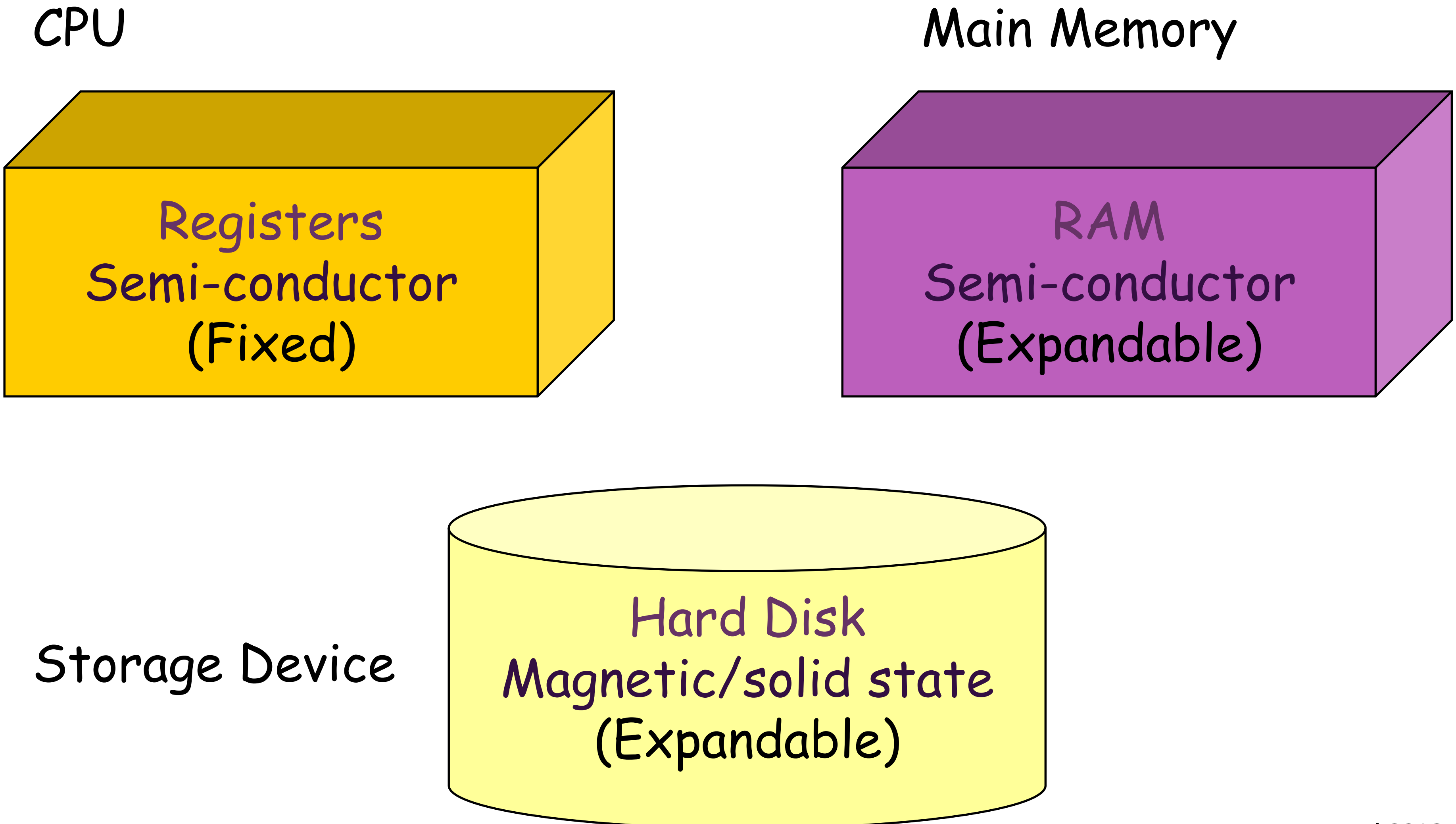
- **Motherboard**
  RAM (Random Access Memory)
  ROM (Read Only Memory)
  Caches
  I/O Registers & Buffers
  Videocard Memory

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

CPU Organisation

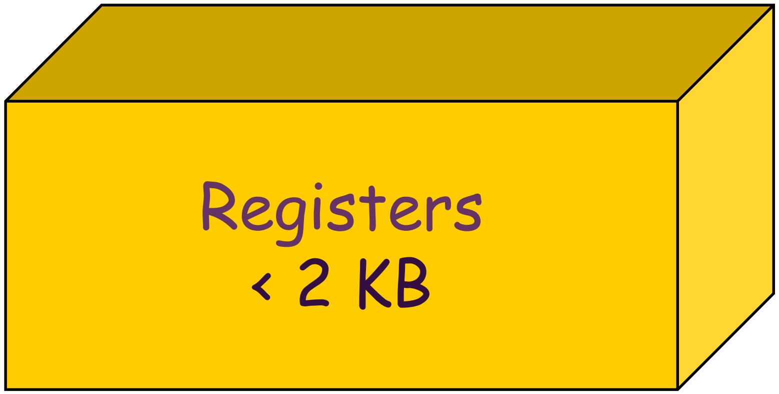


3 Types of Memory

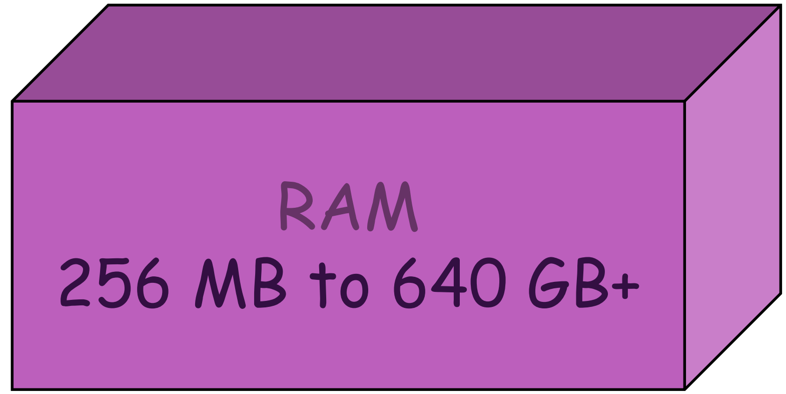


Capacity

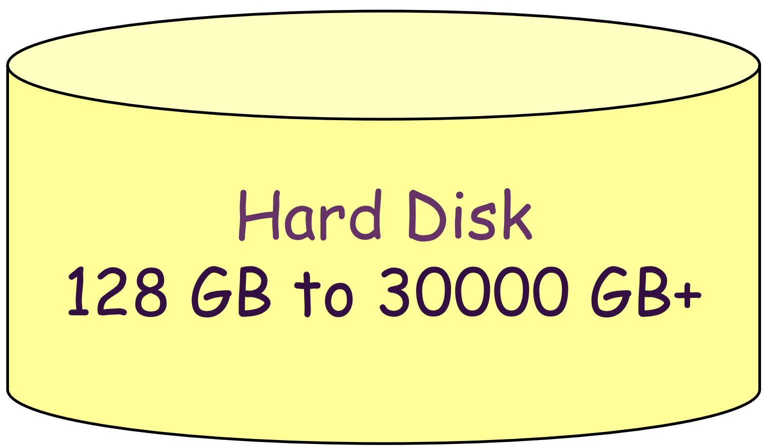
CPU



Main Memory



Storage Device



1 KB = 2^{10} bytes

1 MB = 2^{20} bytes

1 GB = 2^{30} bytes

Speed: Memory Access Time

CPU

Registers
< 0.5 nanosecs

Main Memory

RAM
1.5 nanosecs

Storage Device

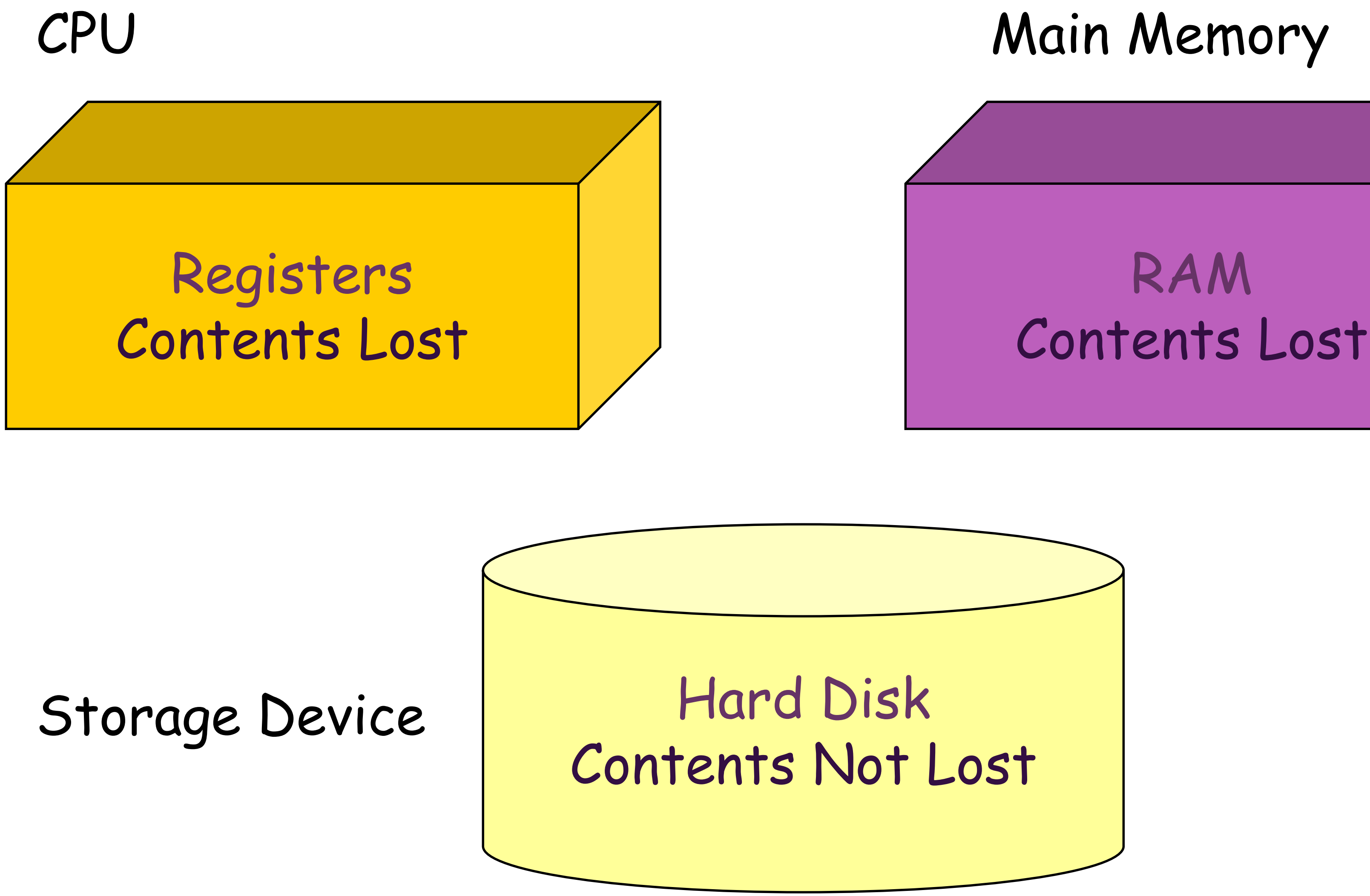
Hard Disk
5 - 10 millisecs
(SSD: 0.1 millisec)

milli = 10^{-3}

micro = 10^{-6}

nano = 10^{-9}

Volatility


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

Cost

Main Memory

RAM
1 GB ~ £6-£12

CPU

Registers
CPU ~ £19-£2636

Sempron 2650 1.45GHz Dual Core 1 MB cache
Xeon E5 2.7GHz 12-core 24 threads 30 MB cache...

Samsung 500GB T3: £172.81

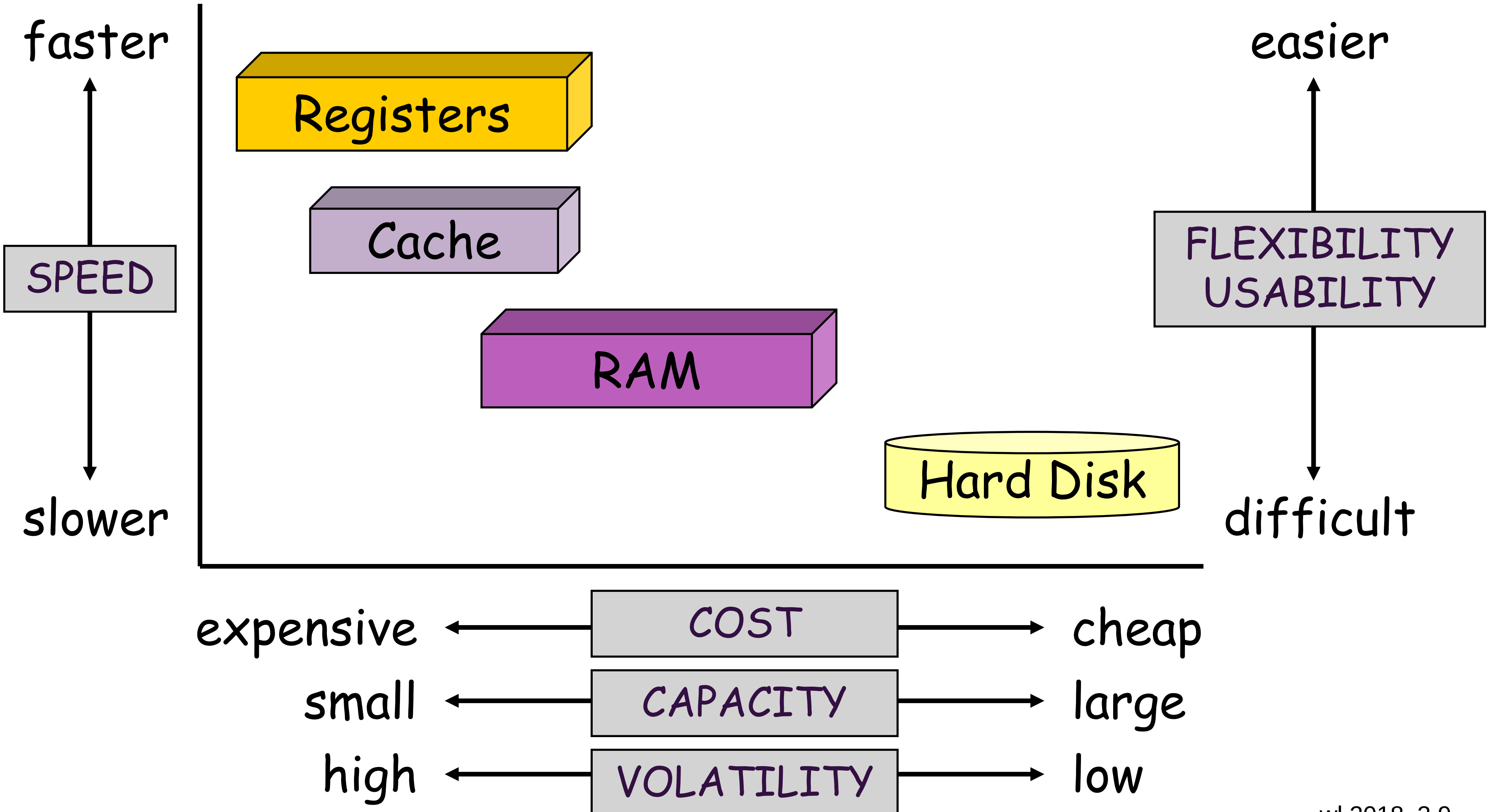
Storage Device

Hard Disk: 1TB ~ £33
SSD: 1TB ~ £344!

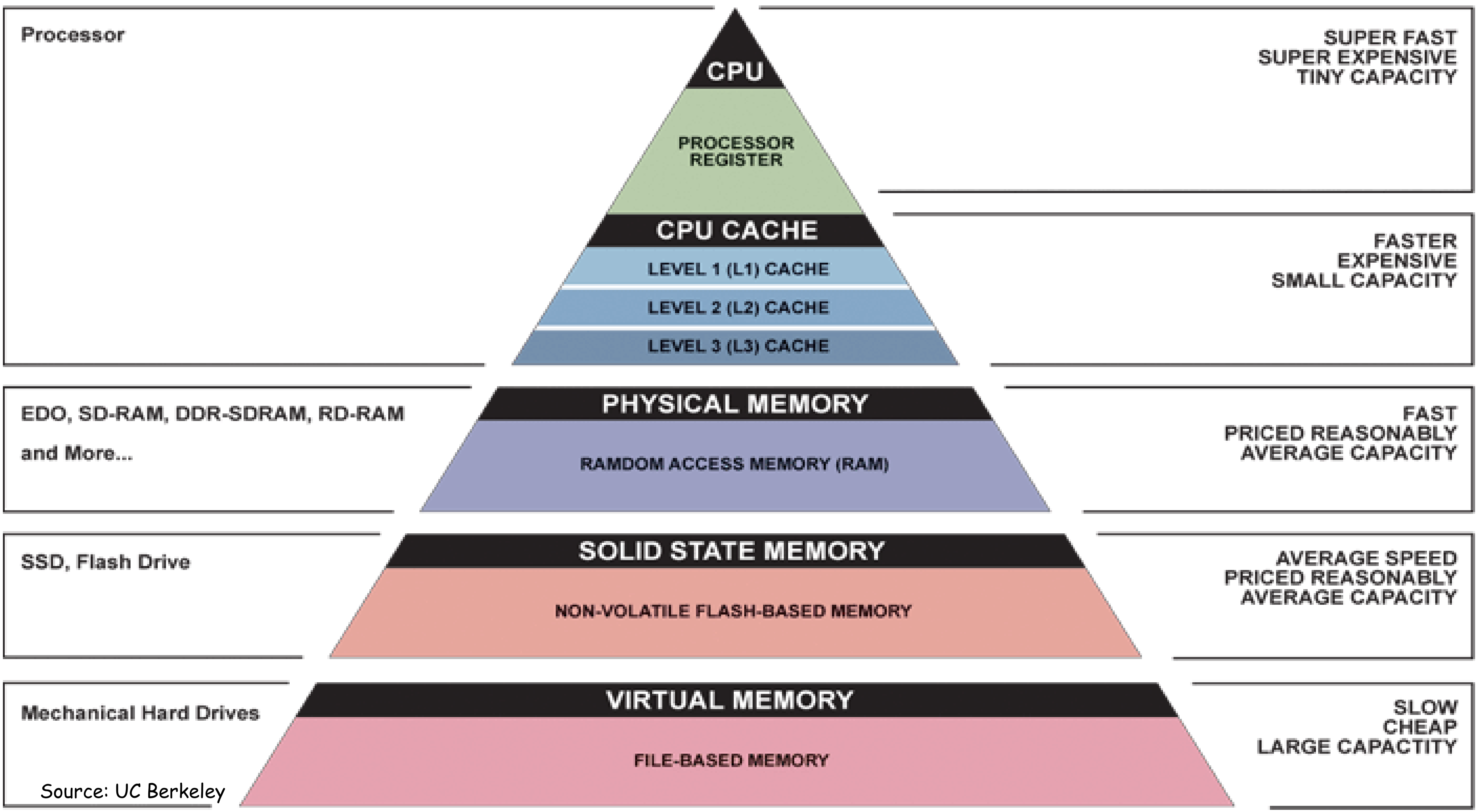
Maxtor 2TB M3: £64.98

Price from shop.bt.com

Summary

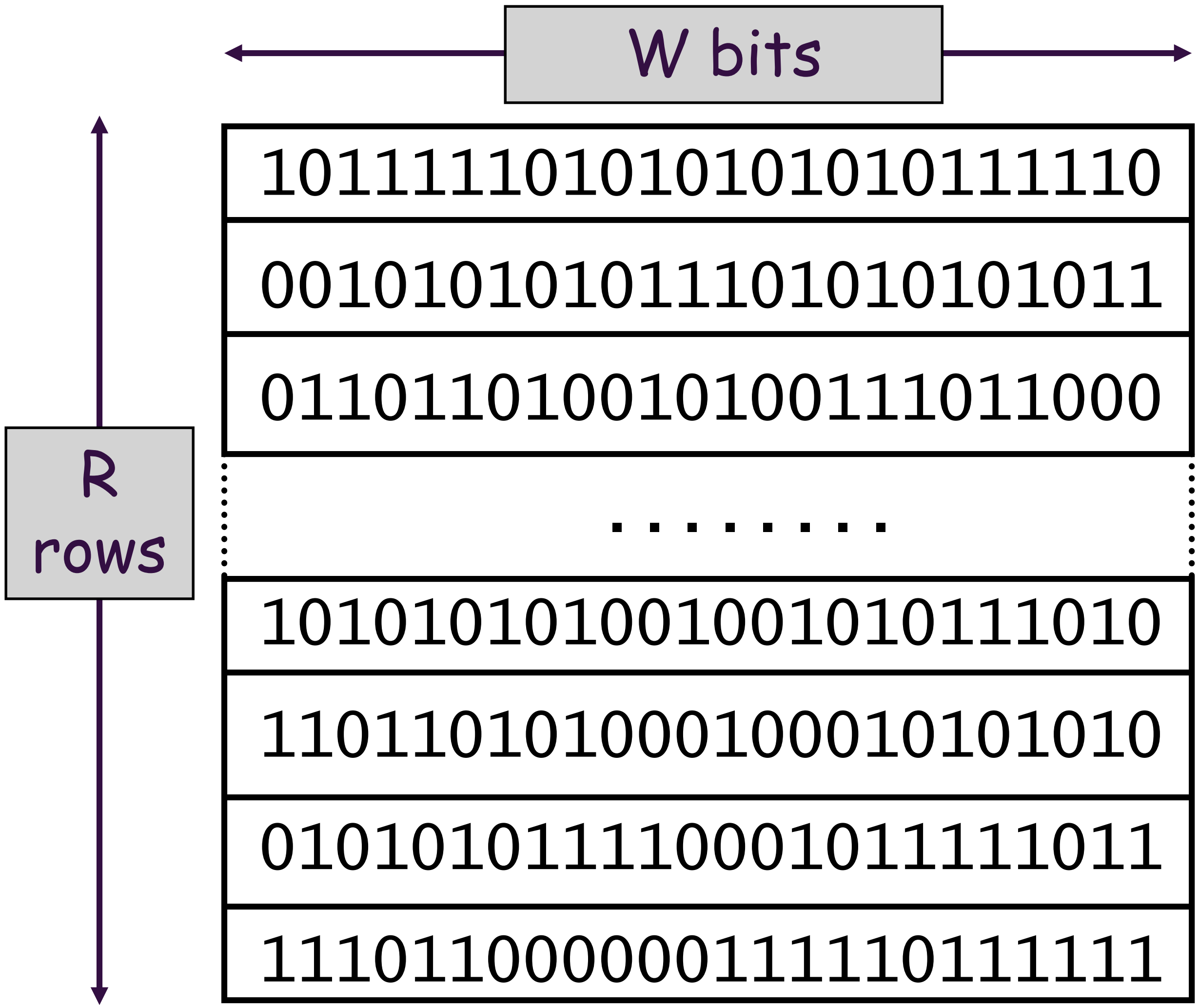

faster
SPEED
slower
Registers
Cache
RAM
Hard Disk
easier
FLEXIBILITY
USABILITY
difficult
expensive
COST
cheap
small
CAPACITY
large
high
VOLATILITY
low

Principle of Locality/Memory Hierarchy



Source: UC Berkeley

Main Memory (RAM)



- **Memory Location** (Memory Word) = W bits. Normally a byte-multiple, e.g. 16 bits, 32 bits.
- **Memory Size** R x W *bits*
- **Access**: *** In our model we will always Read/Write a whole Memory Word at a time ***

Addressing

Main Memory

0110 1101 1010 1101
0000 0000 0000 0011
0000 0000 0000 0000
1111 1111 1111 1111
0000 0000 0000 0000
1001 1010 1010 0010
0000 0000 0000 0000
1111 1111 1111 1110

- Where in memory is the 16-bit two's Complement value 3?
- We need a scheme for uniquely identifying every memory location
- **ADDRESSING**
  Identify memory locations with a positive number called the (memory) **address**

Memory Word Addressing

Main Memory	Address	Address (binary)
0110 1101 1010 1101	0	0000
0000 0000 0000 0100	1	0001
0000 0000 0000 0000	2	0010
1111 1111 1111 1111	3	0011
0000 0000 0000 0000	4	0100
1001 1010 1010 0010	5	0101
0000 0000 0000 0000	6	0110
1111 1111 1111 1110	7	0111

Memory[1] = ?

Memory[?] = -1

Memory[?] = 0

Mem[Mem[1]]= ?

Byte Addressing

Main Memory		Word Address
0110 1101	1010 1101	← 0
0000 0000	0000 0011	← 2
0000 0000	0000 0000	← 4
1111 1111	1111 1111	← 6
0000 0000	0000 0000	← 8
1001 1010	1010 0010	← 10
0000 0000	0000 0000	← 12
1111 1111	1111 1110	← 14

- With byte addressing, every byte in main memory has an address
- In this example which is byte 0 and which is byte 1?

Byte Ordering (Big Endian)

Byte Address	Main Memory		Byte Address
0	0110 1101	1010 1101	1
2	0000 0000	0000 0011	3
4	0000 0000	0000 0000	5
6	1111 1111	1111 1111	7
8	0000 0000	0000 0000	9
10	1001 1010	1010 0010	11
12	0000 0000	0000 0000	13
14	1111 1111	1111 1110	15

Byte Ordering (Little Endian)

Byte Address	Main Memory		Byte Address
1 →	0110 1101	1010 1101	← 0
3 →	0000 0000	0000 0011	← 2
5 →	0000 0000	0000 0000	← 4
7 →	1111 1111	1111 1111	← 6
9 →	0000 0000	0000 0000	← 8
11 →	1001 1010	1010 0010	← 10
13 →	0000 0000	0000 0000	← 12
15 →	1111 1111	1111 1110	← 14