7: MEMORY

COMP110 PRINCIPLES OF COMPUTING

RESEARCH JOURNAL

Presentations this week

Check your timetable

Peer review next week

Have a draft ready!

WORKSHEETS

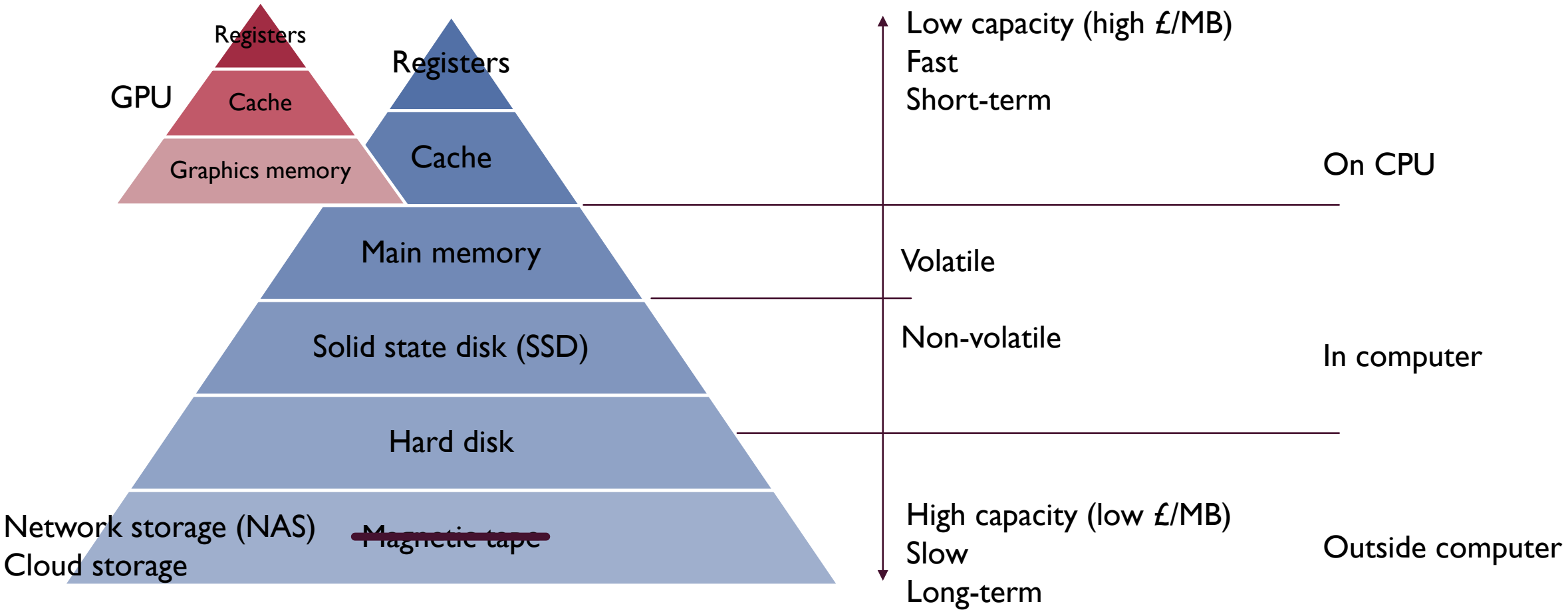
No worksheet this week

If you still have worksheets to complete, there is still time!

WHAT IS MEMORY?

- Allows the computer to **store data**
- Usually when we say "memory" we mean **Random Access Memory (RAM)**
  - **Readable** and **writable**
  - **Volatile** (loses its contents when the machine is powered off)
- Also encompasses
  - **Read Only Memory (ROM)**
  - **Cache**
  - **Non-volatile storage** (SSD, hard disk, flash memory, …)

MEMORY HIERARCHY



REGISTERS

- Memory locations on the CPU itself
- **Very fast** to read and write
- Typically hold **intermediate results** of calculations, and values about to be used by other instructions
- Intel x64 processor has ~40 registers, each 64-bit, so a total of **~320 bytes** of register memory

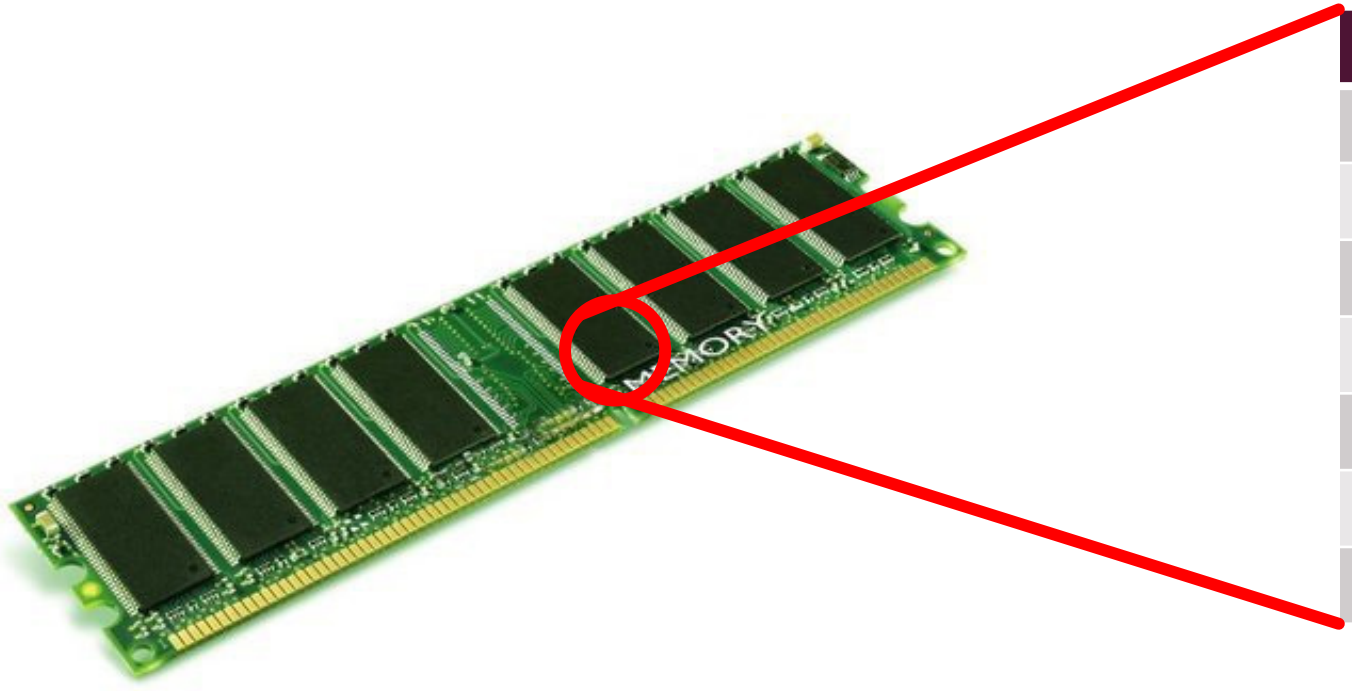
CACHE

- A small amount of RAM on the CPU itself
- Very fast, but not quite as fast as registers
- Can be further divided into **levels**, lower levels being **smaller** and **faster**
- Typically hold **frequently or recently used data** to avoid having to fetch it from main RAM
  - Clever algorithms to determine what to store
- On a recent CPU (Intel Core i9 Comet Lake):
  - Level 2 cache: 10 x 256KB (divided between 10 cores) = 2.5MB
  - Level 3 cache: 20MB (shared between all cores)

BUT WHAT IS MEMORY?

- All types of memory are just ways of storing **binary digits (bits – 0s and 1s)**
- These are organised into **bytes**
- Conceptually, memory is a **sequence of bytes**, each with a numerical **address**

PHYSICAL ADDRESSING



Address	Contents
00000000	01101011
00000001	10001011
00000002	10111000
00000003	11000110
00000004	11000010
00000005	00000001
…	…

VIRTUAL ADDRESSING

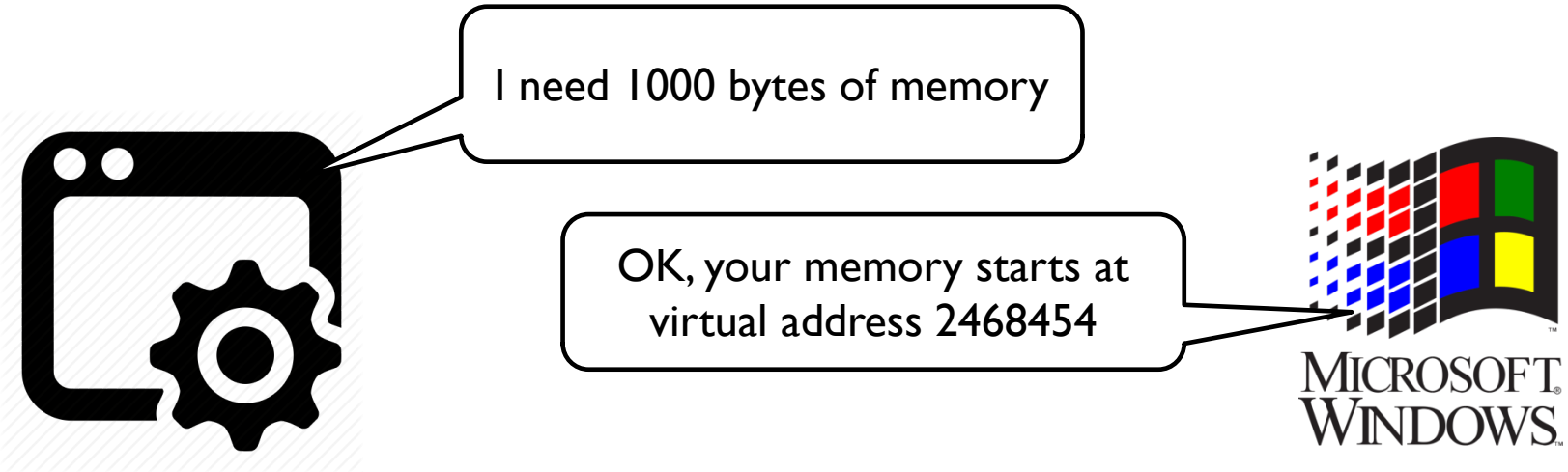
- Historically, programs used physical addressing
- Nowadays, the operating system hides the physical addressing of memory and gives programs a **virtual address space**
- This allows for **multitasking** (programs can coexist)
- Means the OS can decide how to map virtual memory to physical memory
  - Keep frequently accessed data in **cache**
  - Move infrequently accessed data to a **swap file** on disk
  - Allocate memory **lazily** rather than all at once
  - Keeping memory spaces of programs separate helps **stability** and **security**

VIRTUAL ADDRESSING

- Virtual addressing is an **abstraction** of the underlying physical memory
- We can't get away from this particular abstraction without hacking the OS
- Most of the time we can **take for granted** that the abstraction works
- **Understanding** the abstraction can be useful for **optimisation**
  - **Data-oriented design** is one example of this

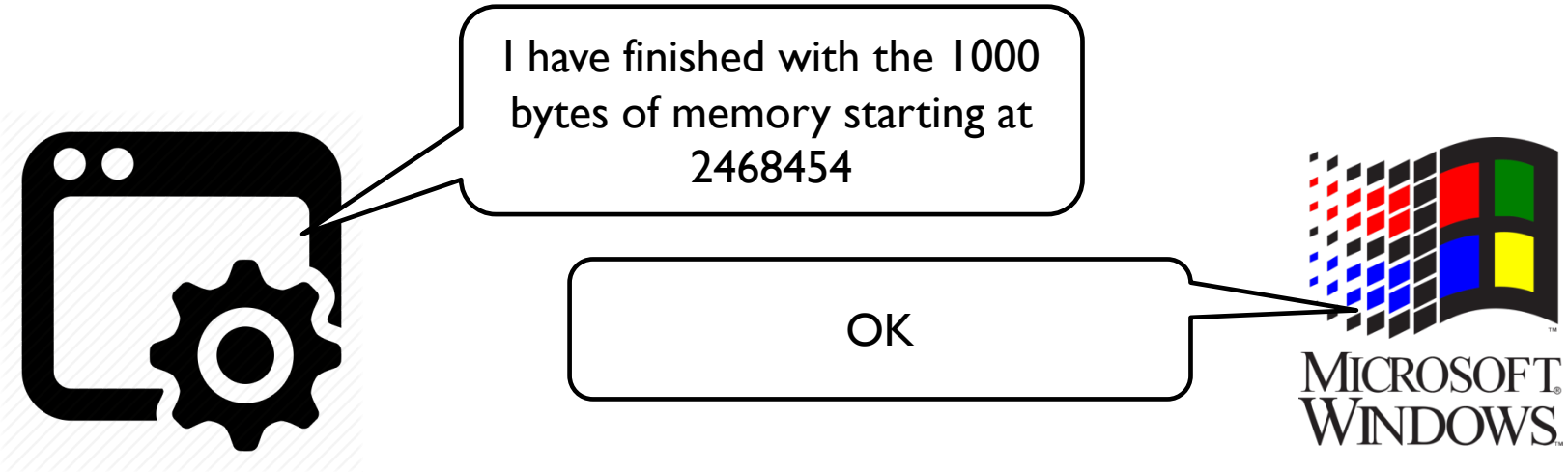
MEMORY ALLOCATION

- Programs can **allocate** memory in **contiguous blocks**
- E.g. `malloc` in C, `new` in C++



MEMORY DEALLOCATION

- Programs can **deallocate** a previously allocated memory block
- E.g. `free` in C, `delete` in C++



MEMORY LEAKS

- Forgetting to deallocate memory that is not needed any more is called a **memory leak**
- A common source of bugs in C/C++ programs
- Generally leads to program's memory usage increasing over time, eventually exceeding physical memory capacity

MANAGED MEMORY

- Many high-level languages use a **managed memory** model
  - Including C#, Python, Java, JavaScript, …
- A **garbage collector (GC)** detects when a block of memory is not needed any more, and deallocates it automatically
- GC takes some CPU resources, but eliminates risk of memory leaks

STORING NUMBERS

- Integer numbers are stored in memory using **binary** notation
- 8-bit numbers stored in 1 byte
- Larger numbers stored in multiple consecutive bytes, in one of two ways…

ENDIANNESS

- E.g. storing the 16-bit number 1234
- Binary: 0000010011010010
- **Big endian** format:

First byte	Second byte
00000100	11010010

- **Little endian** format:

First byte	Second byte
11010010	00000100

WHICH ENDIAN?

- Modern PCs (Intel x64) use **little endian**
- Main advantage is ease of converting between different sizes of integer value

	1st byte	2nd byte	3rd byte	4th byte	5th byte	6th byte	7th byte	8th byte
16-bit	11010010	00000100						
32-bit	11010010	00000100	00000000	00000000				
64-bit	11010010	00000100	00000000	00000000	00000000	00000000	00000000	00000000

	1st byte	2nd byte	3rd byte	4th byte	5th byte	6th byte	7th byte	8th byte
16-bit	00000100	11010010						
32-bit	00000000	00000000	00000100	11010010				
64-bit	00000000	00000000	00000000	00000000	00000000	00000000	00000100	11010010

STORING OTHER DATA

- Any data stored by a computer can be thought of as a sequence of numbers
- **Text**: sequence of character codes in ASCII, UTF-8 etc (see week 3)
- **Graphics**: sequence of RGB pixel values (see COMP120)
- **Audio**: sequence of sample values representing sound waves (see COMP120)
- **Video**: sequence of image frames and accompanying audio
- **3D mesh**: sequence of vertex coordinates
- **Executable**: sequence of CPU instruction opcodes and parameters (see week 12)

SHORT-TERM VS LONG-TERM STORAGE

- Previous slide describes **raw forms** of data storage
- Typically used in RAM to allow for efficient computation
- Volatile storage (SSD, HDD etc) generally uses other data formats
  - **Compression** – especially where data access speed > time to decompress
  - **Headers** and **metadata** for file interchange
  - Standardised file formats vs application-specific memory layout

DATA STORAGE FORMATS

- **Text**: usually still raw text in ASCII / UTF-8 etc
- **Graphics**: JPEG, PNG, BMP, TIFF,…
- **Audio**: WAV, FLAC, OGG, MP3, …
- **Video**: MP4, AVI, …
- **3D mesh**: OBJ, FBX, …
- **Executable**: EXE (contains machine code and metadata)

SUMMARY

- **Memory** refers to how computers store data
- Memory stores **bits**, or sequences of **numbers in binary** – all other data reduces down to this
- There is a **hierarchy** of memory – fast to slow, small to large, temporary to permanent
- **Virtual memory** and **garbage collection** are useful abstractions for us as programmers