Computer fundamentals

International Baccalaureate · IB Diploma · Computer Science · HL

Scope and prerequisites

This handout teaches Theme A1, Computer fundamentals, for the course first assessed in 2027: hardware, data representation, logic and operating systems. Sections marked **AHL** are for Higher Level only.

Prerequisites: powers of 2 ($2^{10} = 1024$) and simple arithmetic.

Computer hardware

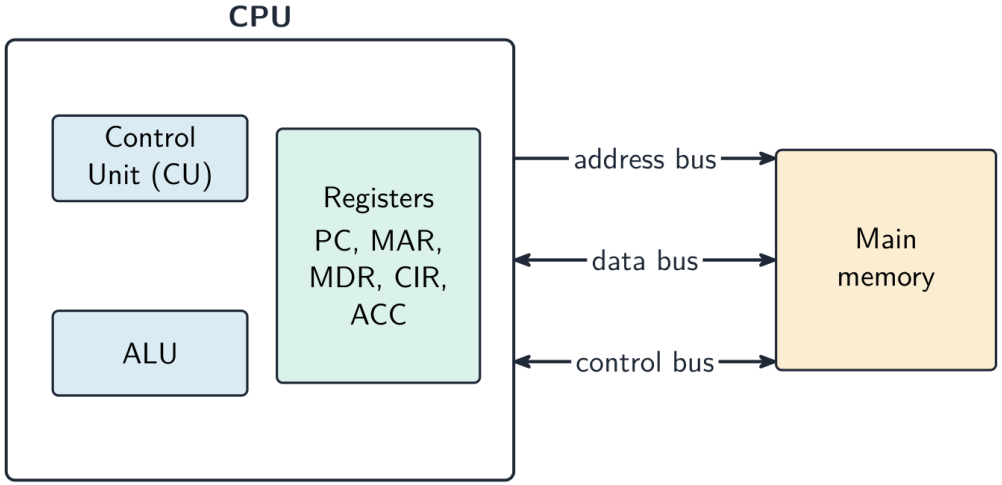
What happens when you tap an app?

You tap a photo app. Within a second its code is copied from storage into memory, the processor runs millions of instructions, the graphics chip draws the picture, and the operating system shares the processor with the music still playing. This handout follows that path.

The central processing unit

The **central processing unit** (CPU) runs program instructions. Its main parts are:

Part	Job
control unit (CU)	fetches and decodes instructions, and sends control signals to the other parts
arithmetic logic unit (ALU)	does arithmetic (add, subtract) and logic (AND, compare)
registers	tiny, very fast stores inside the CPU

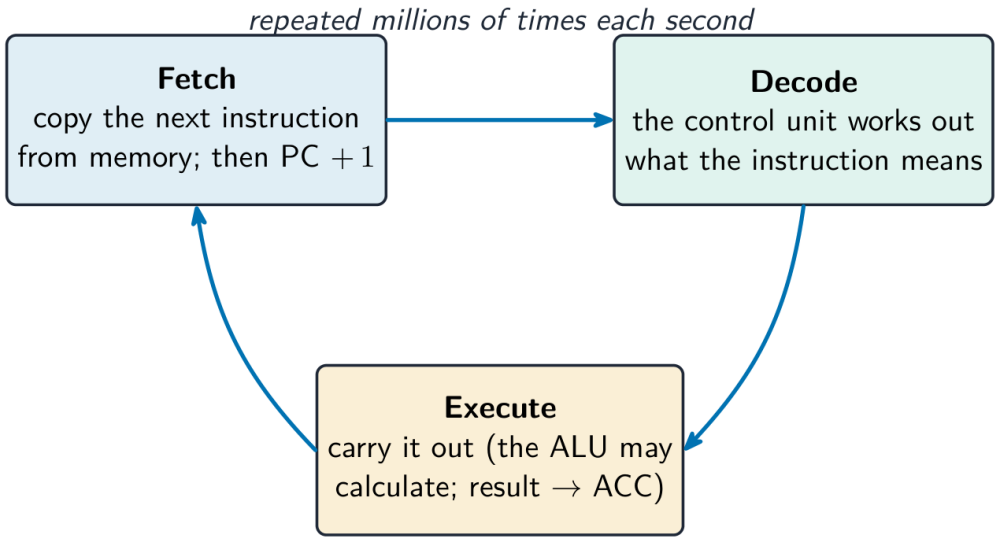


The CPU and main memory, joined by three buses

Register	Holds
program counter (PC)	the address of the next instruction
memory address register (MAR)	the address being read or written
memory data register (MDR)	the data or instruction just read, or about to be written
current instruction register (CIR)	the instruction being decoded and executed
accumulator (ACC)	the result of the last ALU operation

The CPU is joined to memory by **buses**, sets of parallel wires: the **address bus** carries addresses from the CPU to memory, the **data bus** carries data both ways, and the **control bus** carries signals such as “read” and “write”.

The fetch–decode–execute cycle



The cycle every CPU repeats

1. **Fetch**: the address in the PC is copied to the MAR; the instruction at that address travels on the data bus to the MDR, then to the CIR. The PC is increased by 1.
2. **Decode**: the CU works out what the instruction in the CIR means.
3. **Execute**: the instruction is carried out, for example the ALU adds and the result goes into the ACC.

The **clock speed** is the number of cycles per second: a 3 GHz CPU has 3×10^9 clock ticks per second. Performance also depends on the number of **cores** (each core runs its own instructions at the same time) and on the size of the cache.

CPU and GPU

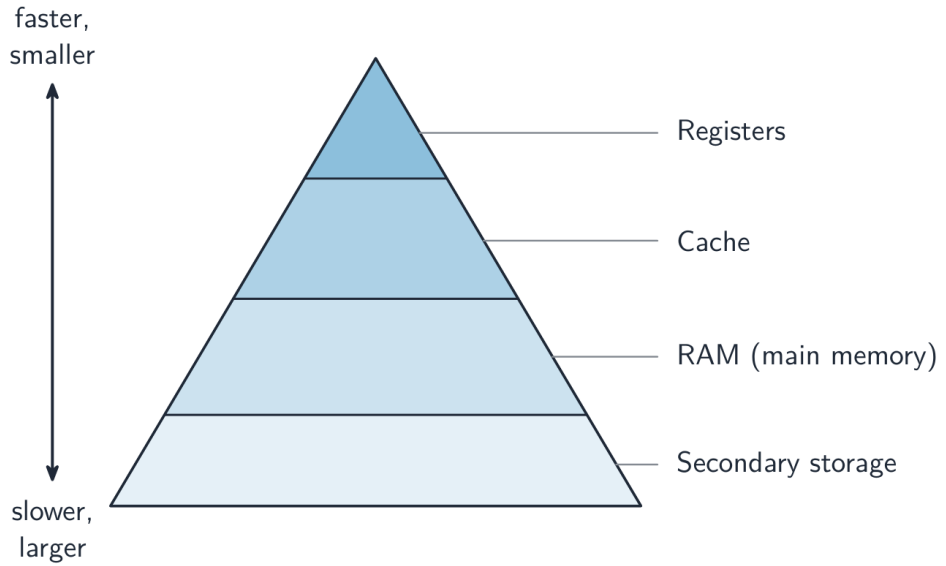
A **graphics processing unit** (GPU) has thousands of simple cores that do the same operation on many pieces of data at once.

	CPU	GPU
cores	few, powerful	thousands, simple
good at	varied tasks with many decisions	the same calculation on lots of data
uses	running the OS and apps	graphics, video, machine-learning training

The CPU and GPU work together: the CPU runs the program and hands the parallel work, such as drawing each pixel, to the GPU.

Memory and storage

Type	Volatile?	Use
RAM	yes: loses its data without power	the programs and data in use now
ROM	no	the start-up instructions (firmware)
cache	yes	copies of the data the CPU uses most, inside or next to the CPU



The memory hierarchy: each level trades speed for size and cost

Cache works because programs reuse the same data and instructions again and again. Level 1 cache is the smallest and fastest; levels 2 and 3 are larger and slower, but all are faster than RAM. A **cache hit** finds the data in cache; a **cache miss** must fetch it from RAM.

Secondary storage keeps data without power:

Type	How it stores bits	Strengths	Weaknesses
hard disk (HDD)	magnetised spots on spinning platters	cheap per gigabyte	moving parts, slower
solid state (SSD)	electric charge in flash memory	fast, silent, robust	dearer per gigabyte
optical (DVD)	pits read by a laser	cheap to distribute	small, slow

Compression

Compression makes files smaller, so they take less storage and transfer faster.

- **Lossless** : the original can be rebuilt exactly. Needed for text and program files. Example, **run-length encoding** (RLE): `WWWWWBBBWW` becomes `5W3B2W`.
- **Lossy** : removes detail people are unlikely to notice; the original cannot be rebuilt. Used for photos (JPEG), music (MP3) and video.

Cloud computing

Cloud computing provides computing resources over the internet, paid for as they are used.

Service	The provider supplies	Example
SaaS (software as a service)	a finished application	web e-mail, online office apps
PaaS (platform as a service)	a platform to build and run your own apps	a hosted database and web server
IaaS (infrastructure as a service)	virtual machines, storage and networks	renting servers by the hour

Advantages: no hardware to buy, scales up and down, accessible anywhere. Disadvantages: needs a reliable connection, ongoing costs, and the data is held by another company.

Data representation

Binary and hexadecimal

Computers store everything as **binary** digits, **bits** , each 0 or 1. Eight bits make a **byte** .

Place value	128	64	32	16	8	4	2	1
bits of 182	1	0	1	1	0	1	1	0

$128 + 32 + 16 + 4 + 2 = 182$. With n bits there are 2^n different values: 8 bits store 0 to 255.

Hexadecimal (base 16) uses the digits 0–9 and A–F (10–15). Each hex digit stands for exactly 4 bits, so hex is a short, readable way to write binary: `1011 0110` is `B6`, and $11 \times 16 + 6 = 182$. Hex is used for colour codes (`#FF8800`) and memory addresses.

Unit	Size
kibibyte (KiB)	$2^{10} = 1024$ bytes
kilobyte (kB)	1000 bytes
megabyte (MB)	10^6 bytes
gigabyte (GB)	10^9 bytes

Negative numbers

Two's complement stores negative integers: the leftmost bit has a negative place value (-128 in 8 bits). To write -42: write 42 as `00101010`, flip every bit to `11010101`, then add 1: `11010110`. Check: $-128 + 64 + 16 + 4 + 2 = -42$. Eight bits store -128 to 127.

Text, images and sound

- **Text**: each character has a code. **ASCII** uses 7 bits (128 characters, English only); **Unicode** assigns code points to characters and symbols, including Chinese; encodings such as UTF-8, UTF-16 and UTF-32 store those code points using different code-unit sequences.
- **Images**: a bitmap is a grid of **pixels** ; the **colour depth** is the number of bits per pixel. File size in bits = width × height × colour depth.
- **Sound**: the wave is measured at regular intervals. The **sample rate** is samples per second; the **bit depth** is bits per sample. Size in bits = sample rate × bit depth × channels × seconds.

Worked examples:

$$\text{image size} = \text{width} \times \text{height} \times \text{colour depth}$$

$$1920 \times 1080 \times 24 = 49\,766\,400 \text{ bits} = 6\,220\,800 \text{ bytes} \approx 6.2 \text{ MB}$$

$$\text{sound size} = \text{sample rate} \times \text{bit depth} \times \text{channels} \times \text{time}$$

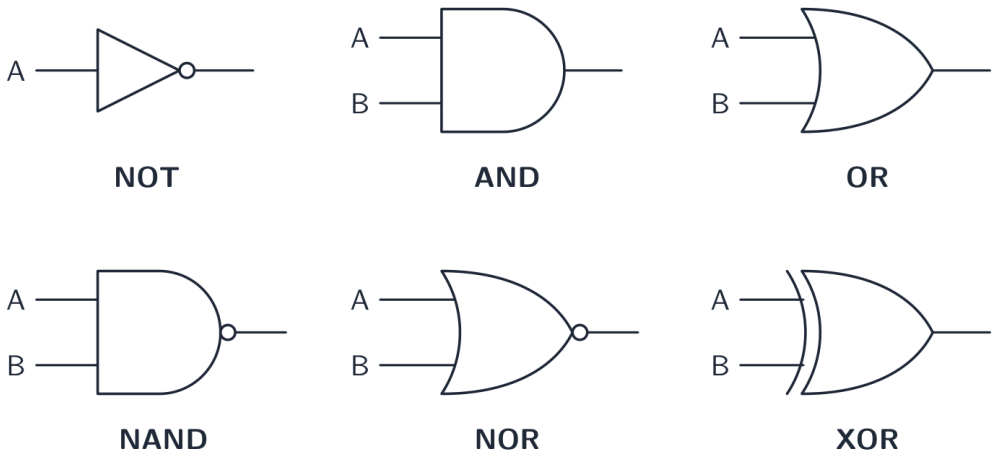
$$44\,100 \times 16 \times 2 \times 60 = 84\,672\,000 \text{ bits} = 10\,584\,000 \text{ bytes} \approx 10.6 \text{ MB}$$

A higher sample rate, bit depth or colour depth gives better quality and a bigger file.

Computer logic

Logic gates

A **logic gate** takes one or two binary inputs and gives one output. A **truth table** lists the output for every combination of inputs.



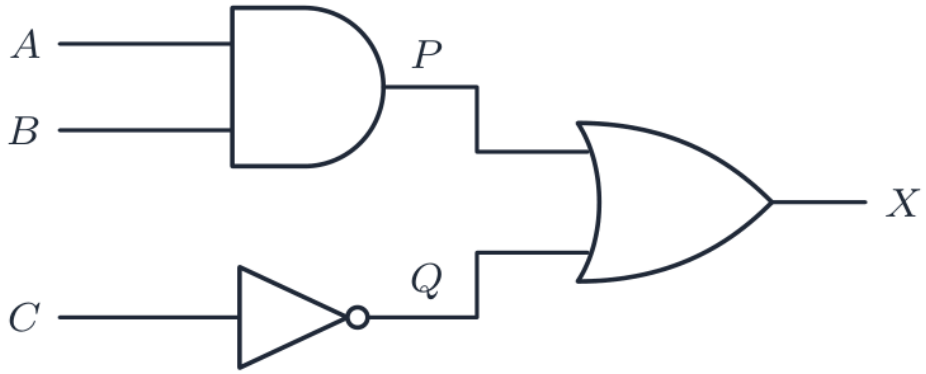
The six gates; a small circle on the output means NOT

A	B	**AND**	**OR**	**XOR**	**NAND**	**NOR**
0	0	0	0	0	1	1
0	1	0	1	1	1	0
1	0	0	1	1	1	0
1	1	1	1	0	0	0

NOT has one input and flips it: NOT $0 = 1$. XOR is 1 when the inputs **differ**; XNOR is 1 when they are the same.

Logic circuits

Gates are joined into circuits. Work out a circuit's truth table by labelling each gate's output and filling in one column per label.



$X = (A \ AND \ B) \ OR \ (NOT \ C)$

A	B	C	$P = A$ **AND** B	$Q =$ **NOT** C	$X = P$ **OR** Q
0	0	0	0	1	1
0	0	1	0	0	0
0	1	0	0	1	1
0	1	1	0	0	0
1	0	0	0	1	1
1	0	1	0	0	0
1	1	0	1	1	1
1	1	1	1	0	1

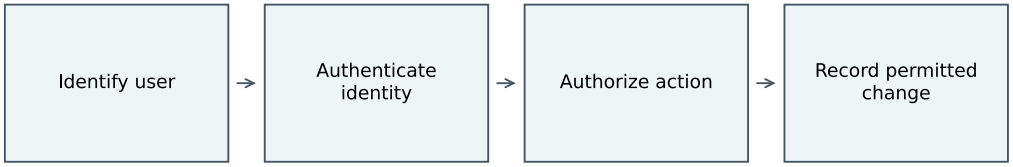
Three inputs give $2^3 = 8$ rows; list them counting up in binary so none is missed.

Operating systems

What an operating system does

An **operating system** (OS) manages the hardware and provides services to programs and users.

Function	What it does
process management	starts and stops programs and shares the CPU between them
memory management	gives each program its own area of RAM and protects it from others
file management	organises files in folders and controls who may read or write them
device management	controls hardware through **drivers**
security	checks users' identity and their permissions
user interface	a graphical or command-line interface



The OS checks who you are (authentication) before what you may do (authorization)

Scheduling

One CPU core runs one process at a time; the OS **scheduler** switches between processes so fast that they seem to run together.

Method	Rule	Weakness
first come, first served	run each process to the end, in arrival order	a long job makes short ones wait
round robin	each process gets a fixed **time slice** in turn	many switches if the slice is small
priority	the highest-priority ready process runs	low-priority processes may never run

Round robin, time slice 2 ms, processes P1 (5 ms), P2 (3 ms), P3 (1 ms) arriving together:

Time (ms)	0–2	2–4	4–5	5–7	7–8	8–9
runs	P1	P2	P3 (ends)	P1	P2 (ends)	P1 (ends)

P3 finishes at 5 ms, P2 at 8 ms, P1 at 9 ms.

Polling and interrupts

The CPU must know when a device needs attention.

- **Polling** : the CPU checks each device again and again. Simple, but wastes CPU time when nothing has happened.
- **Interrupt** : the device sends a signal; the CPU finishes its current instruction, saves its state, runs the interrupt's handler, then carries on. Efficient for events that are rare or unpredictable, such as a key press.

AHL: Translation

Compilers and interpreters

The CPU runs only **machine code** . A program in a high-level language such as Python or Java must be translated.

	Compiler	Interpreter
when	translates the whole program before it runs	translates and runs one statement at a time
output	an executable file	no separate file
speed of running	fast	slower: translated every time
errors	all reported after compiling; nothing runs until they are fixed	stops at the first error it reaches
source code	not needed by users	needed on every machine

Java uses both: the compiler turns source code into **bytecode** , which a virtual machine interprets, or compiles **just in time** (JIT) while the program runs. The same bytecode then runs on any machine with the virtual machine.

Practice

1. State the role of (a) the program counter; (b) the ALU; (c) the address bus.
2. Describe the fetch stage of the fetch–decode–execute cycle, naming the registers used.
3. Explain why a GPU trains a machine-learning model faster than a CPU.
4. Explain why cache memory speeds up a computer.

5. Convert (a) 10011101 to denary; (b) 200 to binary; (c) 2F to denary.
6. Write -19 as an 8-bit two's complement number.
7. Calculate the file size, in bytes, of an 800×600 image with a colour depth of 16 bits.
8. Give the truth table for $X = \text{NOT}(A \text{ OR } B)$, and name the single gate that does the same.
9. State **two** functions of an operating system.
10. Processes P1 (4 ms) and P2 (3 ms) arrive together; round robin with a 2 ms slice. State when each finishes.
11. Explain why a keyboard uses interrupts rather than polling.
12. **AHL** Give **one** advantage of a compiler and **one** of an interpreter for a programmer.

Answers.

1. (a) holds the address of the next instruction; (b) does arithmetic and logic operations; (c) carries addresses from the CPU to memory.
2. The address in the PC is copied to the MAR; the instruction at that address is copied along the data bus into the MDR, then to the CIR; the PC is increased by 1.
3. Training repeats the same calculation on huge amounts of data; the GPU's thousands of cores do these calculations in parallel.
4. It holds copies of the most-used data and instructions next to the CPU, so they are read far faster than from RAM.
5. (a) $128+16+8+4+1 = 157$; (b) $200 = 128+64+8$: `11001000`; (c) $2\times16+15 = 47$.
6. $19 =$ `00010011`; flip: `11101100`; add 1: `11101101`.
7. $800 \times 600 \times 16 = 7\,680\,000$ bits $= 960\,000$ bytes.
8. Rows $00 \to 1$, $01 \to 0$, $10 \to 0$, $11 \to 0$: a NOR gate.
9. E.g. memory management, process management, file management, security.
10. P1 0–2, P2 2–4, P1 4–6 (ends), P2 6–7 (ends): P1 at 6 ms, P2 at 7 ms.
11. Key presses are rare and unpredictable; polling would waste CPU time checking, while an interrupt uses the CPU only when a key is pressed.
12. Compiler: e.g. the finished program runs faster, and the source code need not be shared. Interpreter: e.g. errors are found line by line while testing, and there is no wait to compile.

Exam tips

- Name registers in full the first time (program counter, PC) and give the direction of each bus.
- In the fetch stage, follow the order PC $\to$ MAR $\to$ memory $\to$ MDR $\to$ CIR, and say that the PC increments.
- For conversions, show the place values; for two's complement, show flip then add 1.
- File-size answers: show the formula, the numbers and the unit, and divide by 8 for bytes.
- Truth tables for n inputs need 2^n rows in binary order, with a column for every intermediate output.
- Compare CPU and GPU, polling and interrupts, or compiler and interpreter point by point, not in two separate lists.