Chapter 1.2: Computer Hardware

Computer Fundamentals and Introduction to Programming

System components | Input devices | Output devices | Storage | Compatibility

Sercan KÜLCÜ · Giresun University

© Sercan Külcü. All rights reserved. · Interactive HTML version of these slides:
https://sercankulcu.github.io/files/computer/slides/Chapter_01_02_Computer_Hardware.html

INTRODUCTION

Learning Outcomes

Components

Explain what each basic hardware component does.

Comparison

Distinguish RAM, ROM, HDD and SSD within the memory hierarchy.

Choosing devices

Choose input and output devices for a given purpose.

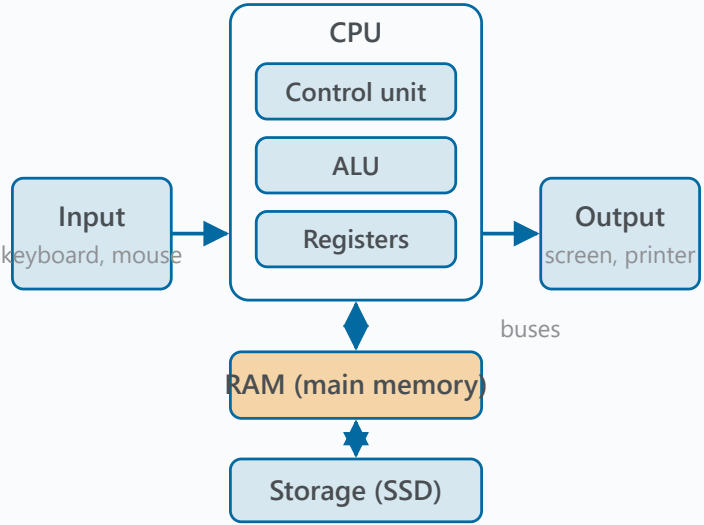
Compatibility

Assess whether components fit together in connections, power and size.

What Is Hardware?

Hardware is all the physical parts of a computer system. Software tells those parts what to do.

- **Input** devices bring data into the system.
- The **CPU** executes instructions.
- **RAM** holds the running program and its data.
- **Storage** keeps data permanently.
- **Output** devices present the result.



Hardware Categories

System components
CPU, motherboard, RAM, GPU, power supply and case.

Input devices
Keyboard, mouse, microphone, camera and sensors.

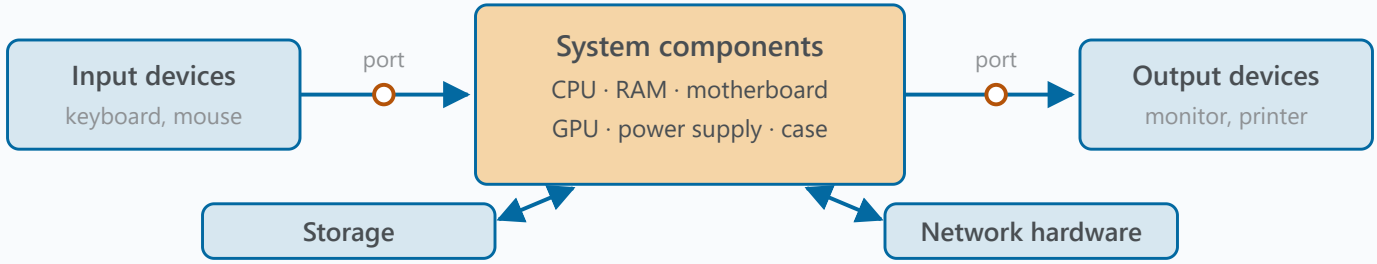
Output devices
Monitor, printer, speakers and projector.

Storage
HDD, SSD, USB drive, memory card and optical media.

Network hardware
Network card, switch, router and access point.

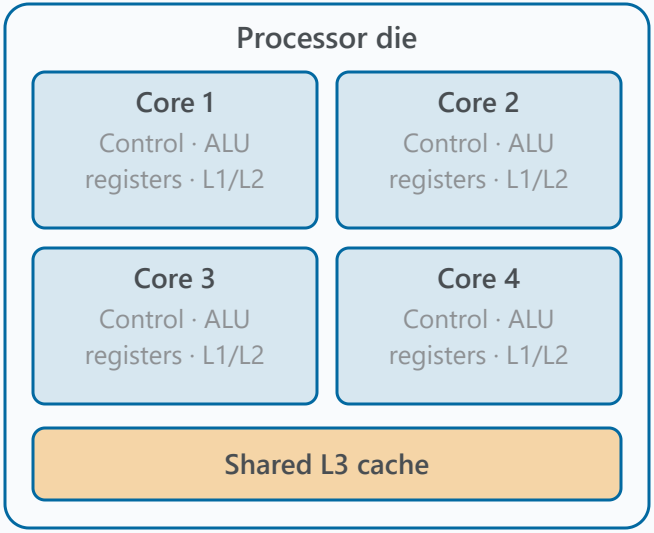
Connections
USB, HDMI, DisplayPort, audio and network ports.

Connections (USB, HDMI, network port) attach devices to the system



The Central Processing Unit (CPU)

- It **fetches, decodes and executes** program instructions.
- The ALU performs arithmetic and logical operations.
- The control unit manages the data flow and sequencing.
- Core count, clock speed (GHz), cache and architecture together determine performance.



Processor Ecosystems

x86-64

The dominant architecture in desktops, laptops and servers.

ARM

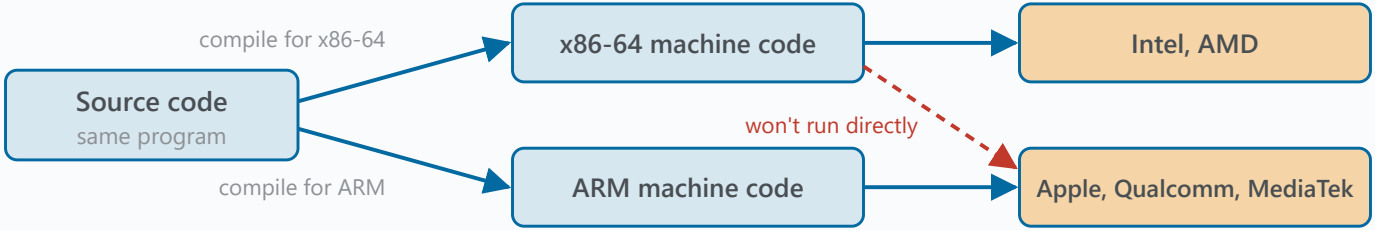
The dominant architecture in mobile and energy-efficient systems.

Intel and AMD

x86-64 processors for consumers and servers.

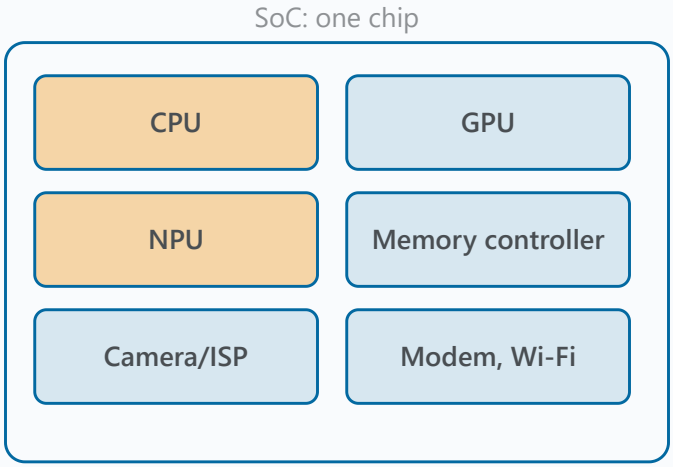
Apple, Qualcomm, MediaTek

Different designs and product families built on ARM.



A brand or model name alone does not determine performance. Generation, power limit, core design and workload all count.

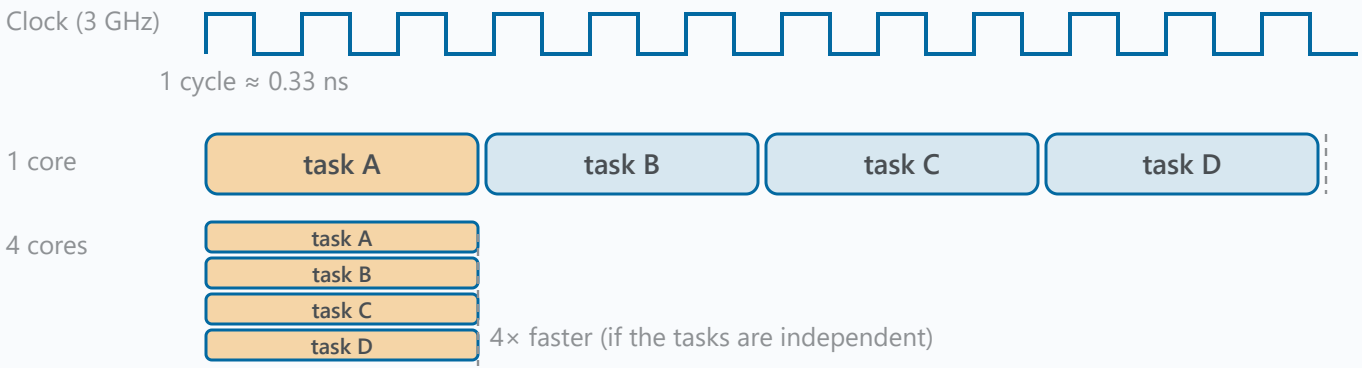
SoC, Mobile Processors and NPUs



- **SoC (system on a chip):** the CPU, GPU, memory controller and other units sit on one chip, saving space and power in phones and thin laptops.
- **NPU (neural processing unit):** speeds up AI workloads (speech recognition, photo enhancement, translation) at low power.
- In most SoCs, RAM is a separate chip placed very close to the SoC; details vary by manufacturer.

Instruction sets: x86-64 (Intel, AMD) dominates desktops and servers; ARM dominates mobile and increasingly laptops; **RISC-V** is an open, royalty-free instruction set. Efficiency differences come from the design, not only the instruction set.

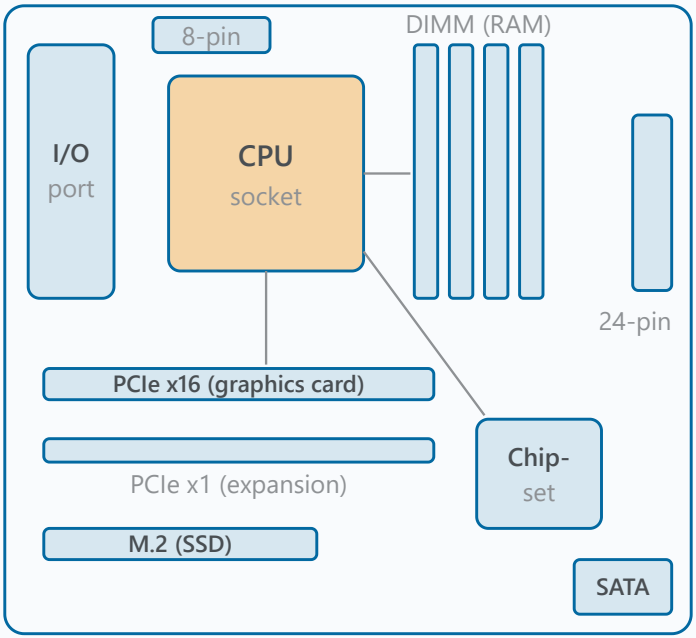
Clock Speed and Core Count



- **Clock speed:** 3 GHz = 3 billion cycles per second; each cycle advances one step.
- **Core:** a unit that runs its own instruction stream; more cores help only if the work can be split.
- A single-threaded program uses one core even on an 8-core CPU, so “8 cores = 8× speed” is false.

SYSTEM COMPONENTS

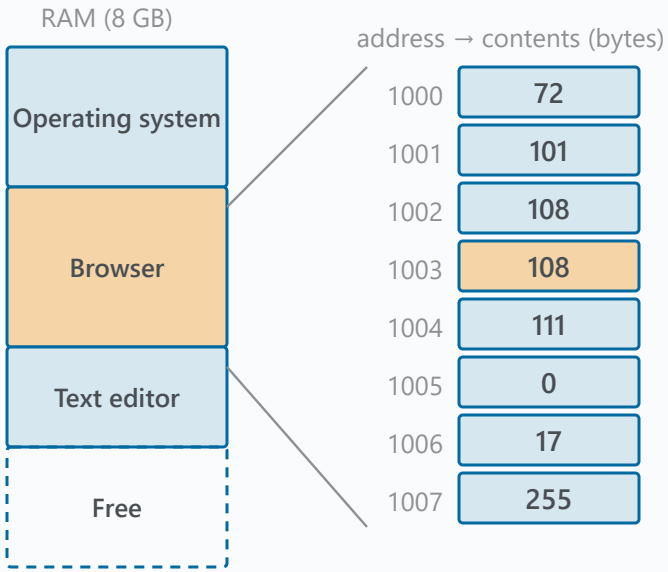
Motherboard



The motherboard ties the components together physically and electrically and lets them talk over buses.

- **CPU socket:** determines which processor family is supported.
- **DIMM slots:** hold the RAM modules.
- **PCIe:** graphics and expansion cards.
- **M.2 / SATA:** storage devices.
- **Chipset:** manages traffic between ports and slots.
- **24-pin / 8-pin:** motherboard and CPU power.

RAM (Random Access Memory)



- **Volatile:** contents are lost when power goes off.
- **Fast access:** the CPU can reach any address directly.
- **Capacity:** affects how many applications run comfortably at once.
- **Compatibility:** DDR generation, speed and motherboard support must match.

Even with free disk space, too little RAM slows programs down and forces the disk to be used as temporary memory.

SYSTEM COMPONENTS

ROM and Firmware

ROM

Memory that keeps its contents without power. Today firmware usually lives in rewritable flash memory.

Firmware

Low-level software that controls the basic behavior of the hardware.

BIOS / UEFI

Initializes the hardware at power-on and starts loading the operating system.

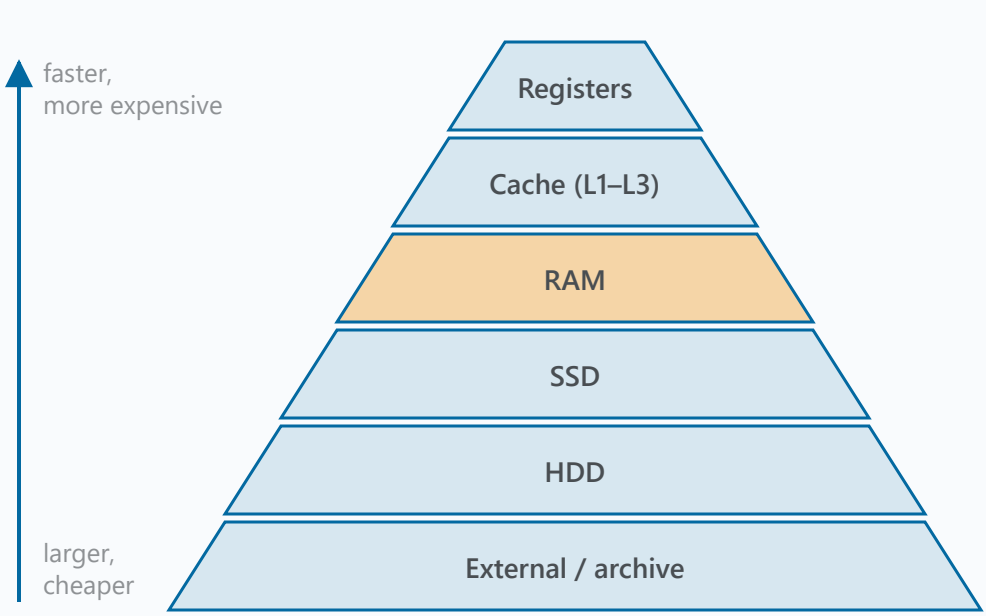
Updates

Fix compatibility or security issues; never interrupt one midway.

The first software to run at power-on is the firmware (UEFI) in the motherboard's flash chip. It finds the operating system on disk and starts it.

SYSTEM COMPONENTS

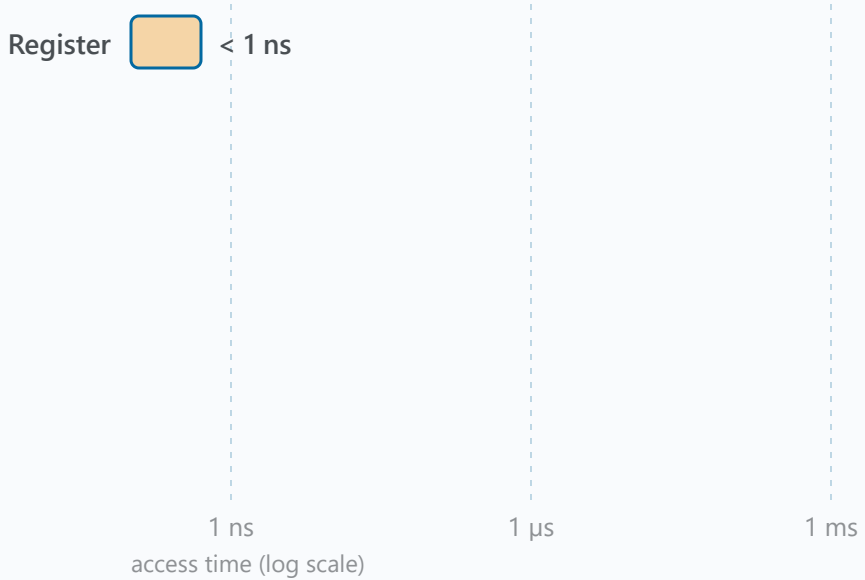
The Memory Hierarchy



	access time	typical capacity
Registers	< 1 ns	hundreds of bytes
Cache (L1–L3)	1–10 ns	KB – tens of MB
RAM	50–100 ns	8–64 GB
SSD	~0.1 ms	0.25–4 TB
HDD	5–10 ms	1–20 TB
External / archive	seconds	TB and up

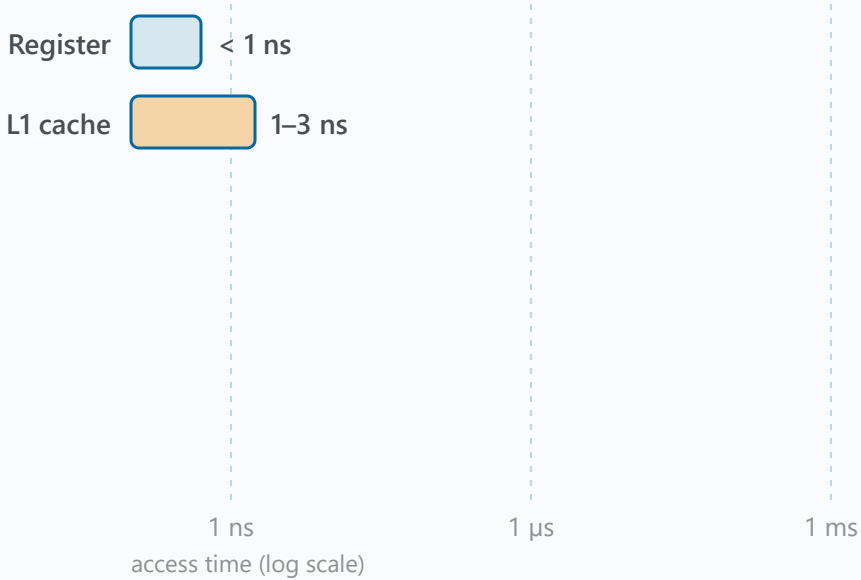
Values are approximate. The CPU keeps frequently used data in the upper levels to avoid slow trips down; RAM is volatile, SSD/HDD are persistent.

Access Time Across the Levels



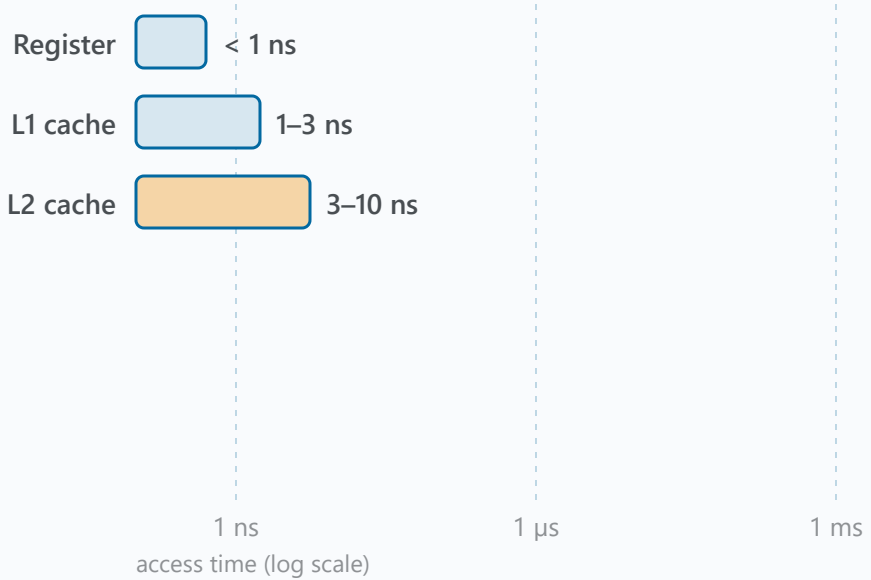
Registers are inside the processor; access takes less than one clock cycle (< 1 ns). Capacity: hundreds of bytes.

Access Time Across the Levels



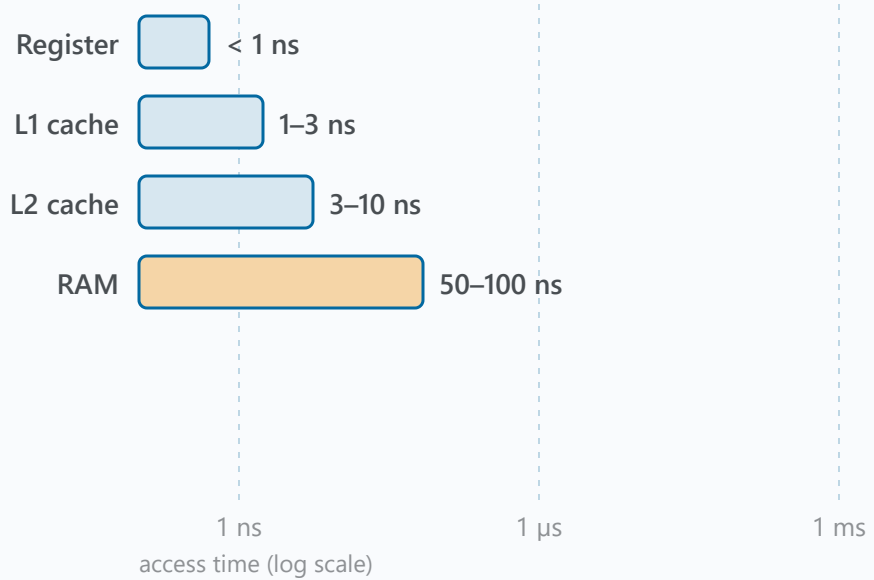
L1 cache is the smallest and fastest cache. L1–L3 range from 1 to 10 ns; L1 is at the low end.

Access Time Across the Levels



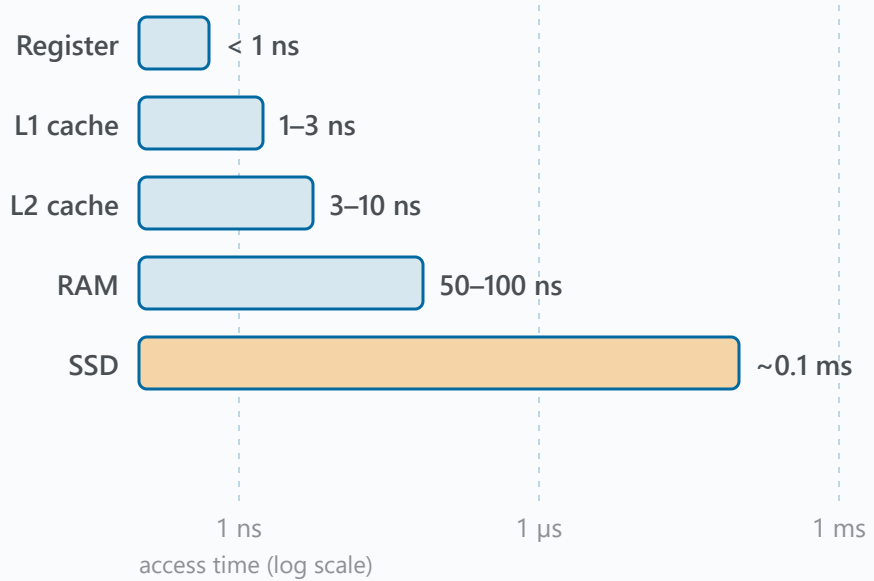
L2 cache is larger but a bit slower (upper part of the range). The scale is logarithmic: gridlines are 1000× apart (1 ns, 1 µs, 1 ms).

Access Time Across the Levels



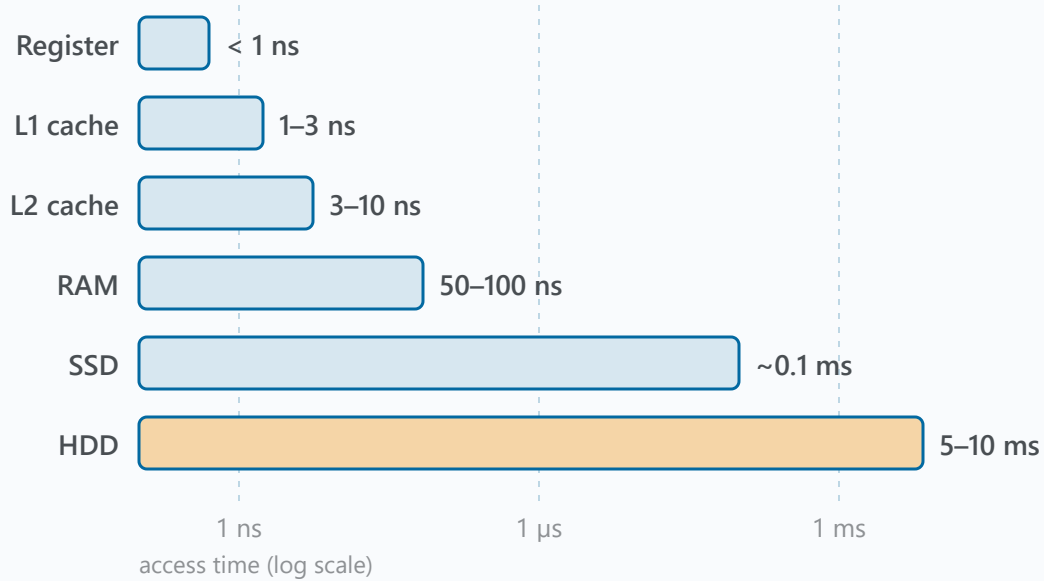
RAM 50–100 ns: tens of times slower than L1. The processor caches data to avoid waiting on RAM.

Access Time Across the Levels



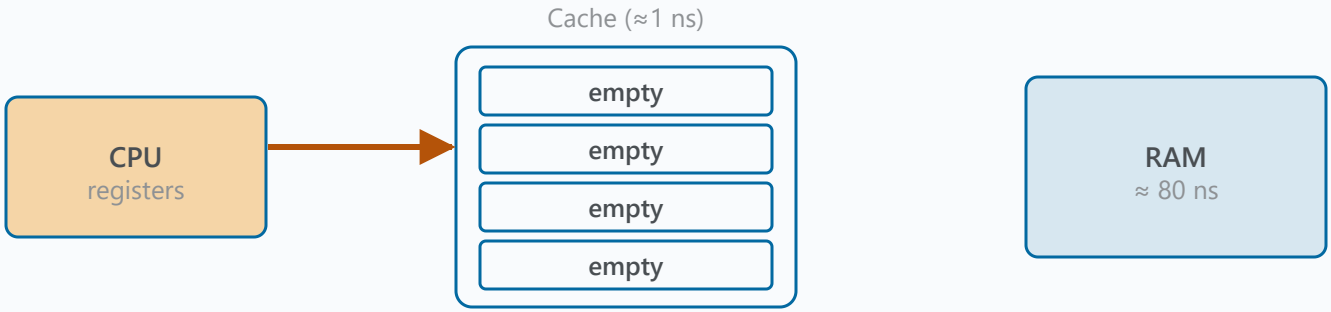
SSD ≈ 0.1 ms = 100,000 ns: about 1000–2000× slower than RAM, but persistent.

Access Time Across the Levels



HDD 5–10 ms: 50–100× slower than an SSD and 50,000–200,000× slower than RAM; it needs mechanical movement (seek + rotation).

How Does a Cache Work? (Hits and Misses)

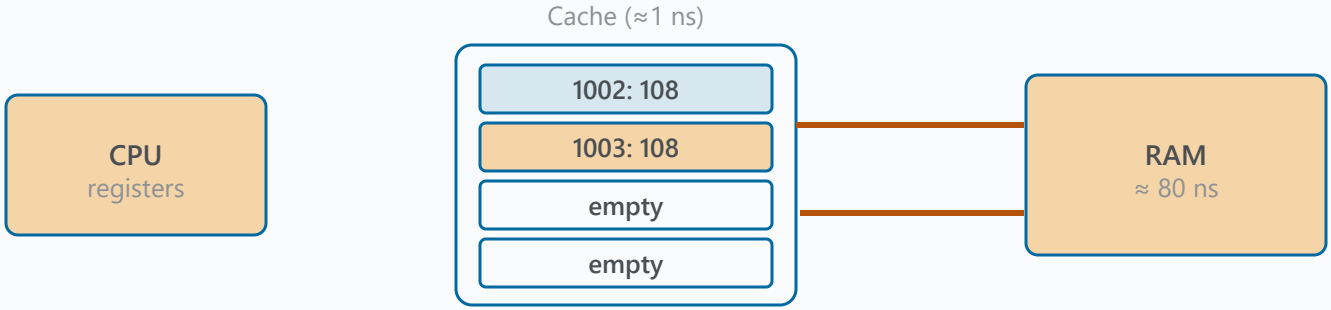


total time: 0 ns

The CPU requests address **1003** (first access). It checks the cache first.

Step 1 / 6

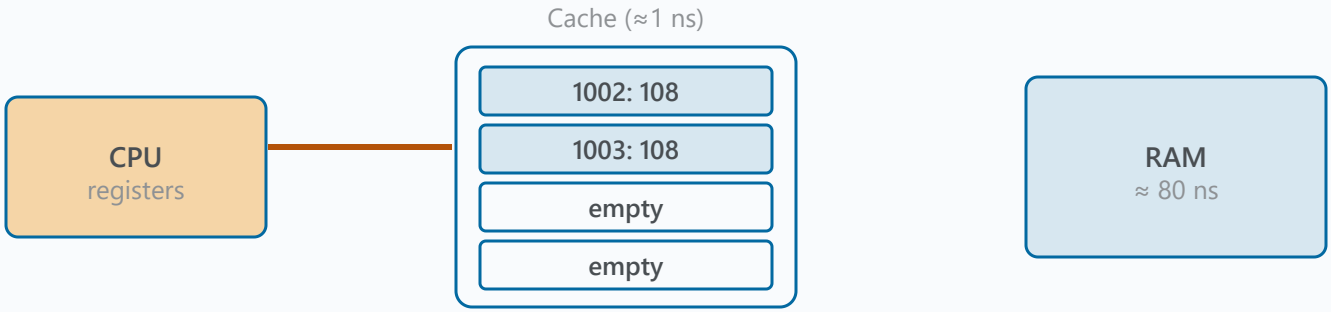
How Does a Cache Work? (Hits and Misses)



total time: 80 ns

Miss: 1003 is not in the cache → fetched from RAM (~80 ns) and copied into the cache together with its neighbor (1002–1003).

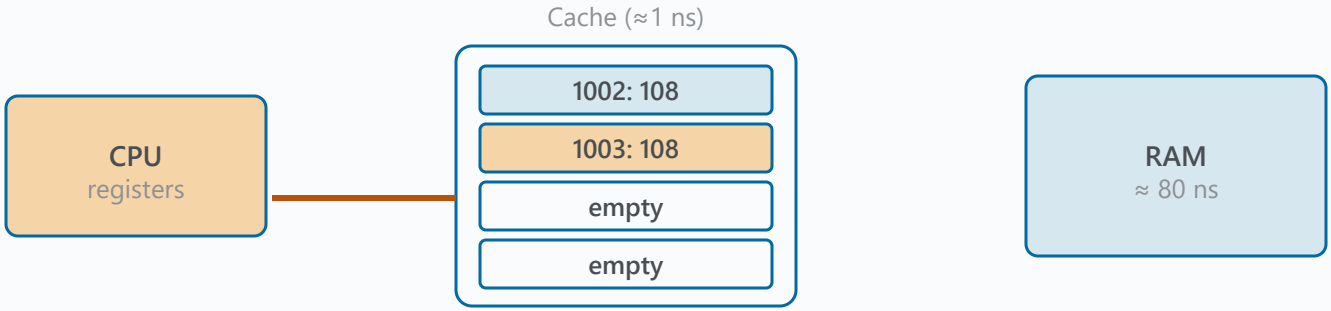
How Does a Cache Work? (Hits and Misses)



total time: 80 ns

The CPU requests address **1003** (again). It checks the cache first.

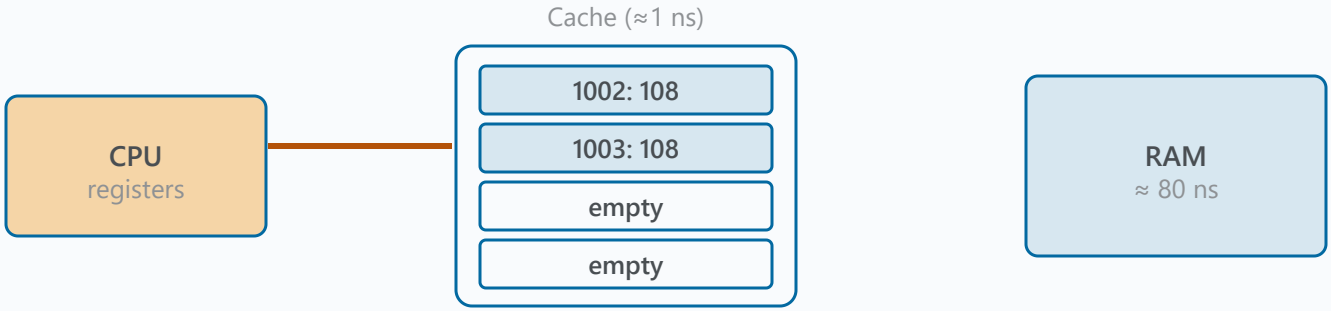
How Does a Cache Work? (Hits and Misses)



total time: 81 ns

Hit: 1003 is in the cache → the value 108 reaches the CPU in about 1 ns.

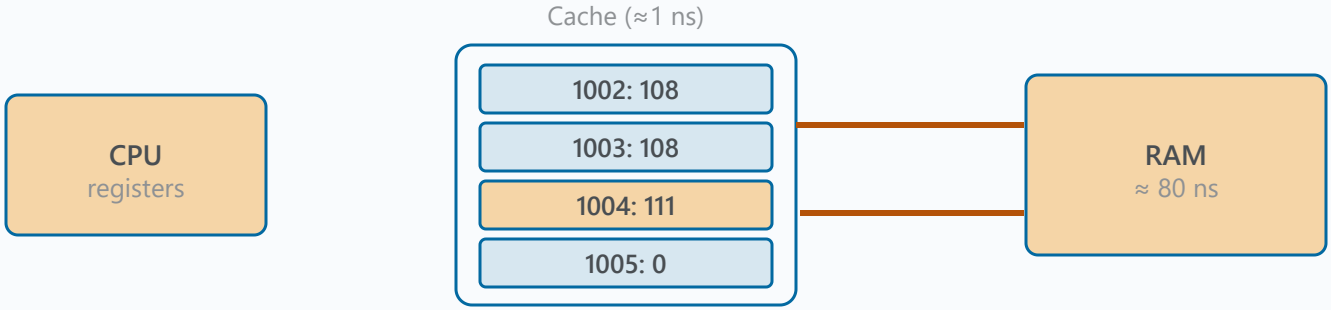
How Does a Cache Work? (Hits and Misses)



total time: 81 ns

The CPU requests address **1004** (neighboring address). It checks the cache first.

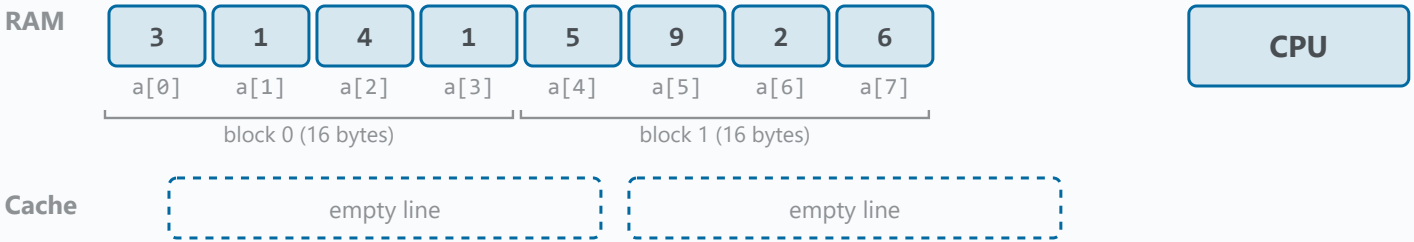
How Does a Cache Work? (Hits and Misses)



total time: 161 ns

Miss: 1004 is not in the cache → fetched from RAM (~80 ns) and copied into the cache together with its neighbor (1004–1005).

Caching an Array Sum: Counting Hits and Misses



Array a is in RAM; the cache is empty. Assume a 4-byte int and a 16-byte cache line (block), i.e. 4 elements; hit ≈ 1 ns, miss ≈ 80 ns.

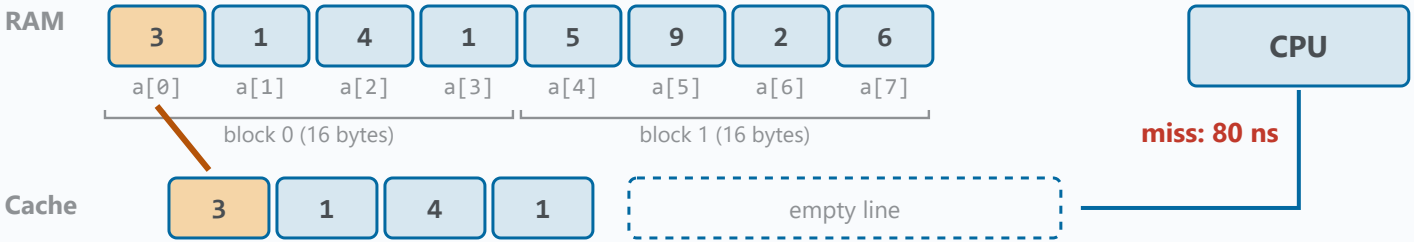
Step 1 / 10

```
▶ sum = 0
2 for i = 0, 1, …, 7 do
3     sum = sum + a[i]
4 // 4-byte int, 16-byte line
```

VARIABLES

`sum = 0` `hits = 0` `misses = 0` `time = 0 ns`

Caching an Array Sum: Counting Hits and Misses



a[0] requested: not in cache → **miss**. RAM sends the whole block (a[0]…a[3]), not just a[0]: 80 ns.

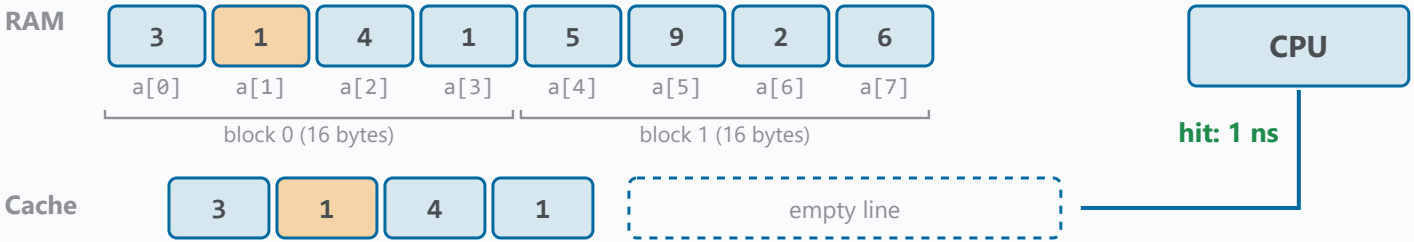
Step 2 / 10

```
1  sum = 0
2  for i = 0, 1, …, 7 do
▶      sum = sum + a[i]
4  // 4-byte int, 16-byte line
```

VARIABLES

`i = 0` `sum = 3` `hits = 0` `misses = 1` `time = 80 ns`

Caching an Array Sum: Counting Hits and Misses



a[1] requested: block 0 is cached → **hit**, 1 ns. It came along with the previous miss.

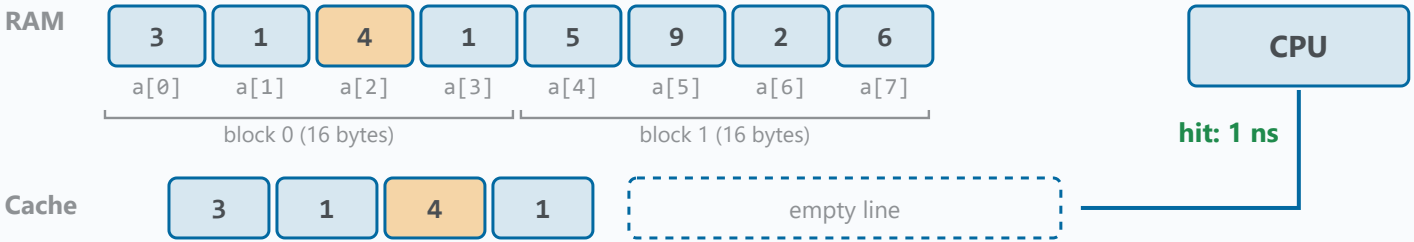
Step 3 / 10

```
1 sum = 0
2 for i = 0, 1, …, 7 do
▶     sum = sum + a[i]
4 // 4-byte int, 16-byte line
```

VARIABLES

`i = 1` `sum = 4` `hits = 1` `misses = 1` `time = 81 ns`

Caching an Array Sum: Counting Hits and Misses



a[2] requested: block 0 is cached → **hit**, 1 ns. It came along with the previous miss.

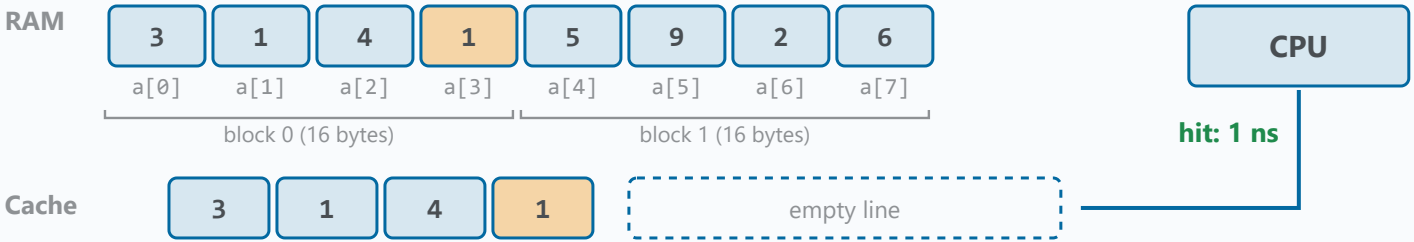
Step 4 / 10

```
1 sum = 0
2 for i = 0, 1, …, 7 do
▶     sum = sum + a[i]
4 // 4-byte int, 16-byte line
```

VARIABLES

`i = 2` `sum = 8` `hits = 2` `misses = 1` `time = 82 ns`

Caching an Array Sum: Counting Hits and Misses



a[3] requested: block 0 is cached → **hit**, 1 ns. It came along with the previous miss.

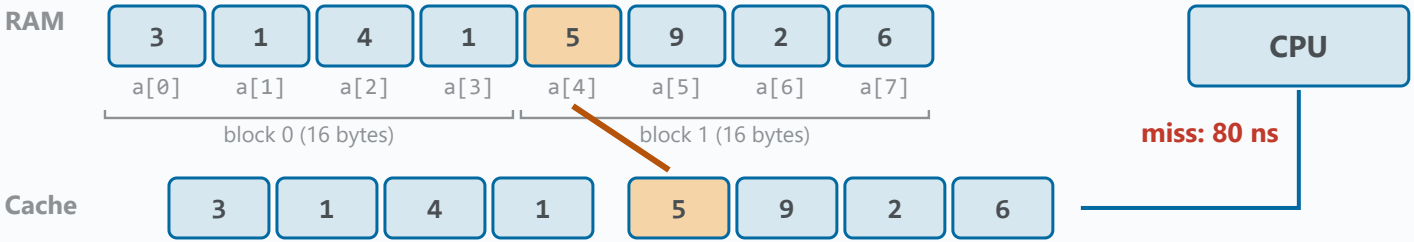
Step 5 / 10

```
1 sum = 0
2 for i = 0, 1, …, 7 do
▶     sum = sum + a[i]
4 // 4-byte int, 16-byte line
```

VARIABLES

`i = 3` `sum = 9` `hits = 3` `misses = 1` `time = 83 ns`

Caching an Array Sum: Counting Hits and Misses



a[4] requested: not in cache → **miss**. RAM sends the whole block (a[4]…a[7]), not just a[4]: 80 ns.

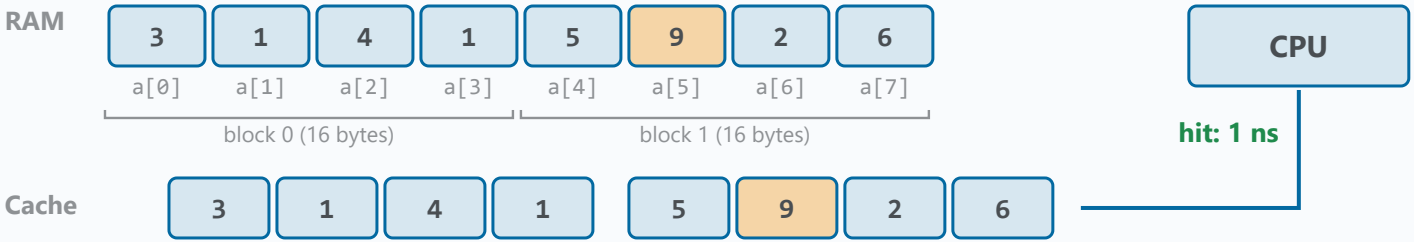
Step 6 / 10

```
1 sum = 0
2 for i = 0, 1, …, 7 do
▶     sum = sum + a[i]
4 // 4-byte int, 16-byte line
```

VARIABLES

`i = 4` `sum = 14` `hits = 3` `misses = 2` `time = 163 ns`

Caching an Array Sum: Counting Hits and Misses



a[5] requested: block 1 is cached → **hit**, 1 ns. It came along with the previous miss.

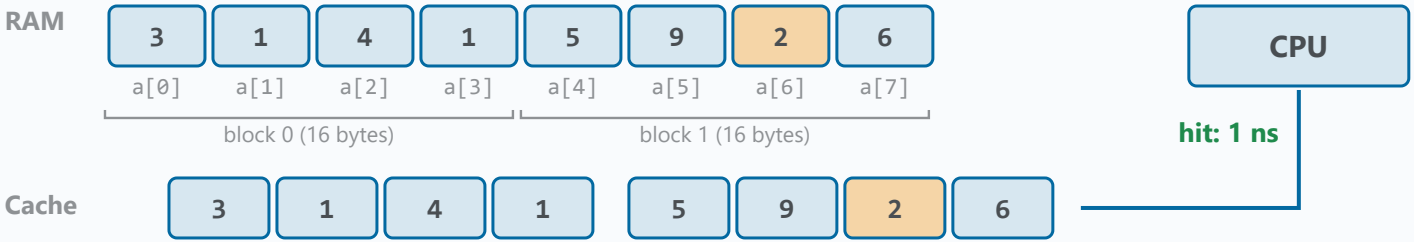
Step 7 / 10

```
1 sum = 0
2 for i = 0, 1, …, 7 do
▶     sum = sum + a[i]
4 // 4-byte int, 16-byte line
```

VARIABLES

`i = 5` `sum = 23` `hits = 4` `misses = 2` `time = 164 ns`

Caching an Array Sum: Counting Hits and Misses



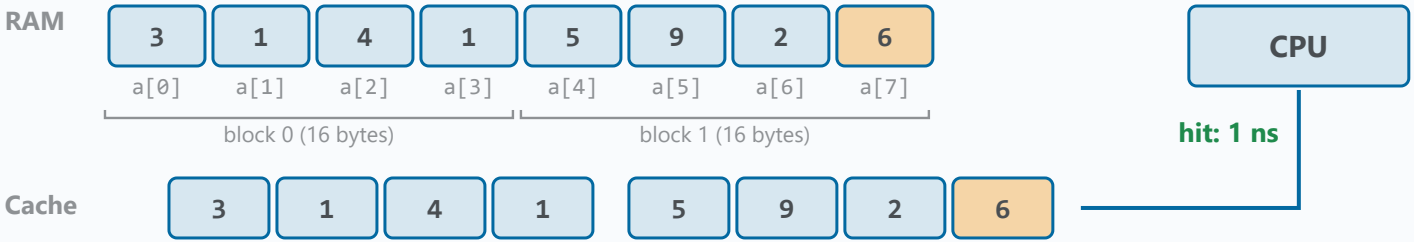
a[6] requested: block 1 is cached → **hit**, 1 ns. It came along with the previous miss.


```
1 sum = 0
2 for i = 0, 1, …, 7 do
▶     sum = sum + a[i]
4 // 4-byte int, 16-byte line
```

VARIABLES

`i = 6` `sum = 25` `hits = 5` `misses = 2` `time = 165 ns`

Caching an Array Sum: Counting Hits and Misses



a[7] requested: block 1 is cached → **hit**, 1 ns. It came along with the previous miss.

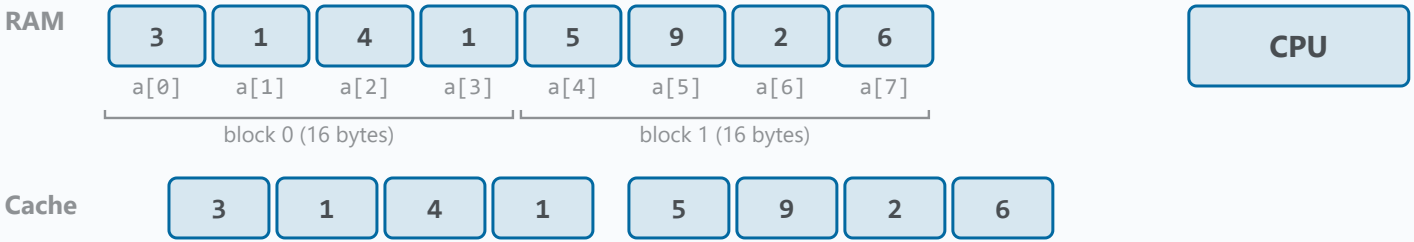

```
1 sum = 0
2 for i = 0, 1, …, 7 do
▶     sum = sum + a[i]
4 // 4-byte int, 16-byte line
```

VARIABLES

`i = 7` `sum = 31` `hits = 6`

`misses = 2` `time = 166 ns`

Caching an Array Sum: Counting Hits and Misses



Done: sum = 31. 6 of the 8 accesses hit; time 2 × 80 + 6 × 1 = **166 ns**. Without a cache it would be 8 × 80 = 640 ns. That is why sequential access is fast: **spatial locality**. Striding through the array (a[0], a[4], …) would miss every time.

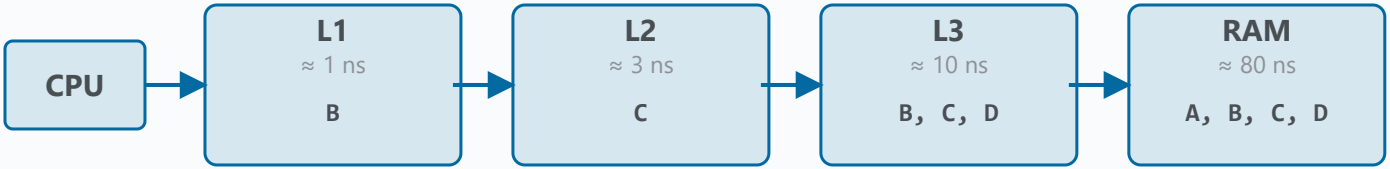
Step 10 / 10

```
1 sum = 0
2 for i = 0, 1, …, 7 do
▶     sum = sum + a[i]
4 // 4-byte int, 16-byte line
```

VARIABLES

`i = 7` `sum = 31` `hits = 6` `misses = 2` `time = 166 ns`

Multi-Level Cache Reads, Step by Step



each level: larger and slower

The CPU will read addresses **A**, **A**, **D** in turn. Initially L1 holds B, L2 holds C, L3 holds B, C, D; A is only in RAM. Times are approximate.

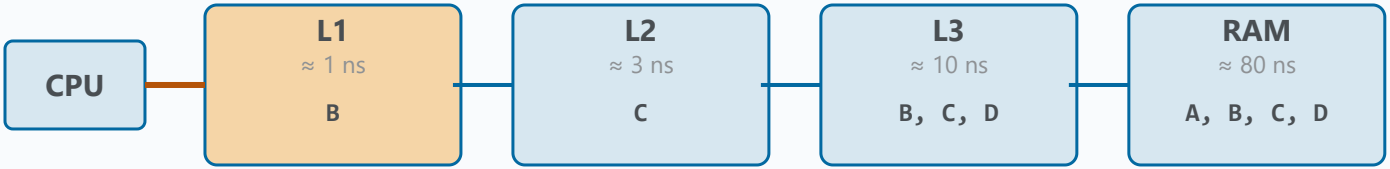
Step 1 / 10

```
1 In L1?  Yes → read, done
2 In L2?  Yes → copy to L1, read
3 In L3?  Yes → copy to L2 and L1, read
4 Fetch from RAM → copy to L3, L2, L1, read
```

VARIABLES

`address = –` `time = 0 ns`

Multi-Level Cache Reads, Step by Step



each level: larger and slower

A requested. Start with the fastest, smallest level: not in L1 (1 ns spent).

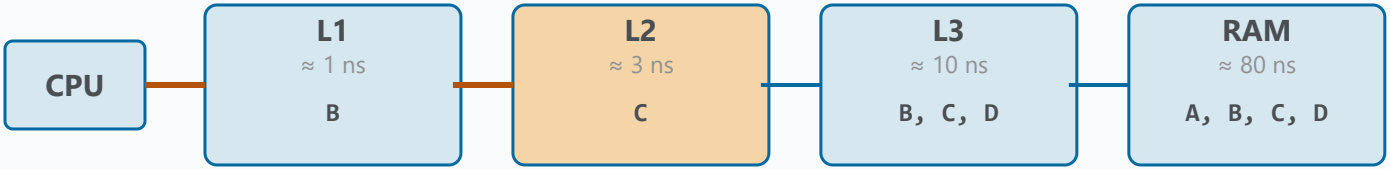
Step 2 / 10

```
▶ In L1?  Yes → read, done
2 In L2?  Yes → copy to L1, read
3 In L3?  Yes → copy to L2 and L1, read
4 Fetch from RAM → copy to L3, L2, L1, read
```

VARIABLES

`address = A` `time = 1 ns`

Multi-Level Cache Reads, Step by Step



each level: larger and slower

Not in L2 either: +3 ns.

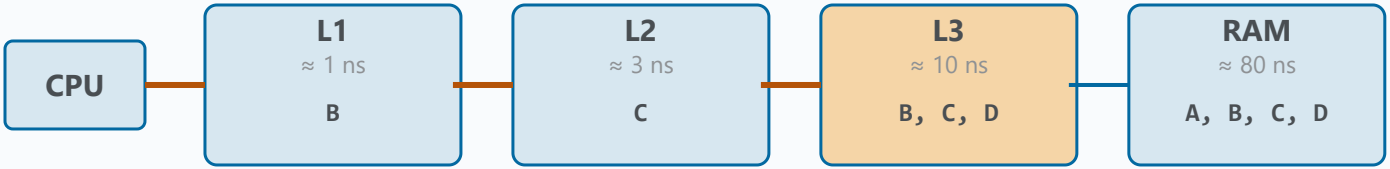
Step 3 / 10

```
1 In L1?  Yes → read, done
▶ In L2?  Yes → copy to L1, read
3 In L3?  Yes → copy to L2 and L1, read
4 Fetch from RAM → copy to L3, L2, L1, read
```

VARIABLES

`address = A` `time = 4 ns`

Multi-Level Cache Reads, Step by Step



each level: larger and slower

Not in L3 either: +10 ns.

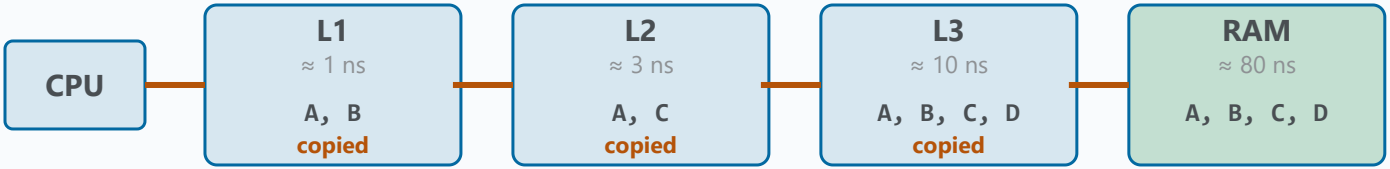
Step 4 / 10

```
1 In L1?  Yes → read, done
2 In L2?  Yes → copy to L1, read
▶ In L3?  Yes → copy to L2 and L1, read
4 Fetch from RAM → copy to L3, L2, L1, read
```

VARIABLES

`address = A` `time = 14 ns`

Multi-Level Cache Reads, Step by Step



each level: larger and slower

Fetched from RAM: +80 ns, total ≈ **94 ns**. On the way, the data is copied into every level so the next request is fast.

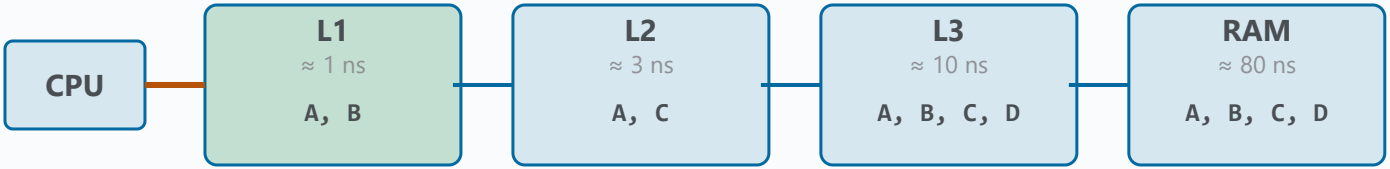
Step 5 / 10

```
1 In L1?  Yes → read, done
2 In L2?  Yes → copy to L1, read
3 In L3?  Yes → copy to L2 and L1, read
▶ Fetch from RAM → copy to L3, L2, L1, read
```

VARIABLES

`address = A` `time = 94 ns`

Multi-Level Cache Reads, Step by Step



each level: larger and slower

A requested again: in L1 → **hit**, ≈ 1 ns. Reusing the same data soon after is fast: **temporal locality**.

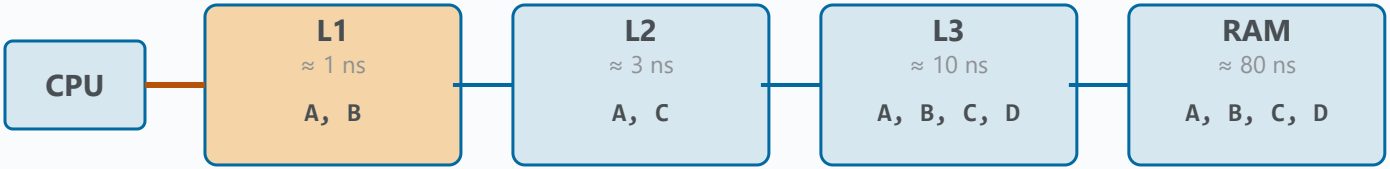
Step 6 / 10

```
▶ In L1?  Yes → read, done
2 In L2?  Yes → copy to L1, read
3 In L3?  Yes → copy to L2 and L1, read
4 Fetch from RAM → copy to L3, L2, L1, read
```

VARIABLES

`address = A` `time = 1 ns`

Multi-Level Cache Reads, Step by Step



each level: larger and slower

D requested: not in L1 (1 ns).

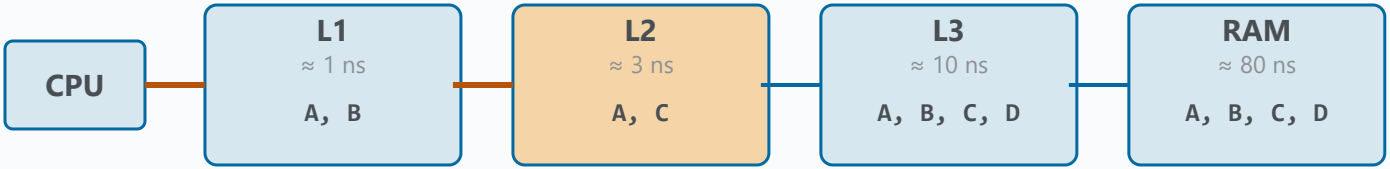
Step 7 / 10

```
▶ In L1?  Yes → read, done
2 In L2?  Yes → copy to L1, read
3 In L3?  Yes → copy to L2 and L1, read
4 Fetch from RAM → copy to L3, L2, L1, read
```

VARIABLES

`address = D` `time = 1 ns`

Multi-Level Cache Reads, Step by Step



each level: larger and slower

Not in L2 (+3 ns).

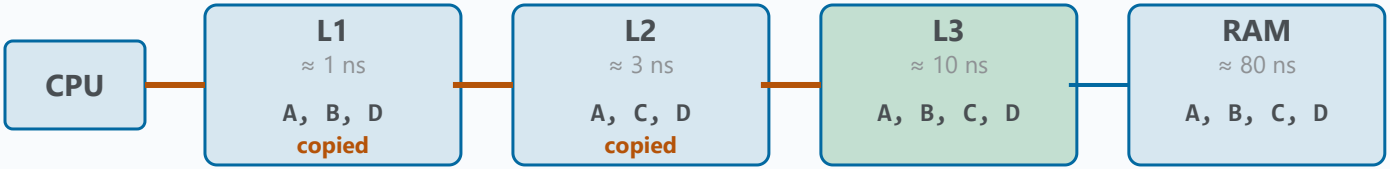
Step 8 / 10

```
1 In L1?  Yes → read, done
▶ In L2?  Yes → copy to L1, read
3 In L3?  Yes → copy to L2 and L1, read
4 Fetch from RAM → copy to L3, L2, L1, read
```

VARIABLES

`address = D` `time = 4 ns`

Multi-Level Cache Reads, Step by Step



each level: larger and slower

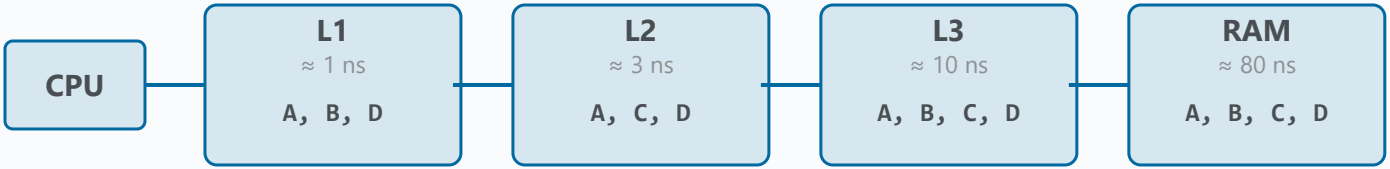
Found in L3: +10 ns, total ≈ **14 ns**; copied to L2 and L1. About 7× faster than going to RAM.


```
1 In L1?  Yes → read, done
2 In L2?  Yes → copy to L1, read
▶ In L3?  Yes → copy to L2 and L1, read
4 Fetch from RAM → copy to L3, L2, L1, read
```

VARIABLES

`address = D` `time = 14 ns`

Multi-Level Cache Reads, Step by Step



each level: larger and slower

Three reads: 94 + 1 + 14 = 109 ns. Reading all from RAM would take 3 × 80 = 240 ns. Real caches evict the least recently used line when full; this example ignores space limits.

Step 10 / 10

```
1 In L1?  Yes → read, done
2 In L2?  Yes → copy to L1, read
3 In L3?  Yes → copy to L2 and L1, read
4 Fetch from RAM → copy to L3, L2, L1, read
```

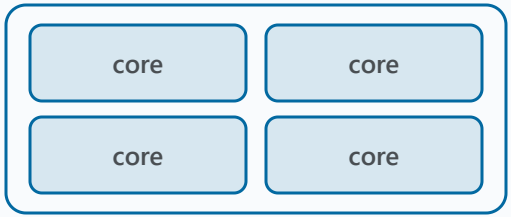
VARIABLES

`address = –` `time = 109 ns (total)`

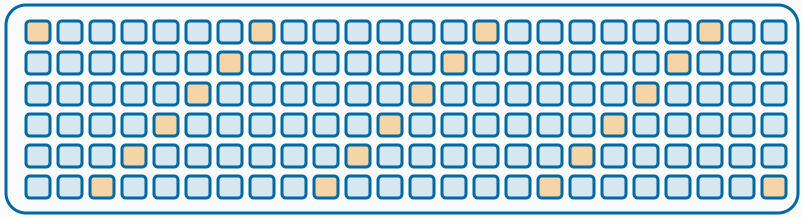
SYSTEM COMPONENTS

The Graphics Processing Unit (GPU)

A GPU is designed to run huge numbers of similar operations (e.g. the colors of millions of pixels) in **parallel**.



CPU: a few powerful cores



GPU: thousands of simple cores, one operation in parallel

Integrated GPU

In the same package as the processor; shares system memory.

Discrete GPU

A separate card with its own processor and video memory (VRAM).

Uses

Gaming, video processing, scientific computing, AI.

Selection criteria

Workload, VRAM, power draw, cooling, software support.

Display Connections

Connector	Signal	Common use
HDMI	Digital video and audio	Monitors, TVs, projectors
DisplayPort	Digital video and audio	Computer monitors
USB-C	Video, data and power, depending on model	Laptops and mobile devices
VGA	Analog video	Older displays and projectors

Connector faces (schematic)



HDMI
digital

DisplayPort
digital

USB-C
digital (depends on model)

VGA
analog, 15 pins

SYSTEM COMPONENTS

USB and Other Ports



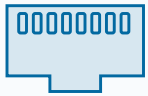
USB-A
fits one way only



USB-C
reversible, data+power+video



HDMI
video + audio



RJ-45
wired network (Ethernet)



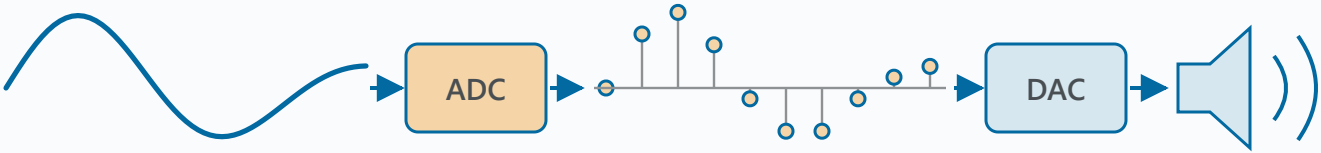
3.5 mm
analog audio

USB version	Max speed	Note
USB 2.0	480 Mbit/s	keyboard, mouse
USB 3.2 Gen 1 (formerly USB 3.0)	5 Gbit/s	external drive
USB 3.2 Gen 2	10 Gbit/s	fast SSD
USB4	20 or 40 Gbit/s	USB-C; displays, docks
Thunderbolt 4	40 Gbit/s	USB-C; displays, docks

Connector shape (A, C) and version (speed) are separate things: not every USB-C port supports 40 Gbit/s or video.

SYSTEM COMPONENTS

Sound Card



analog signal (microphone) samples: 0, 5, 7, 4, -1, -4… speaker

Analog to digital (ADC)

The microphone signal is measured thousands of times per second and turned into numbers (e.g. 44,100 samples/s).

Digital to analog (DAC)

The numbers are turned back into an electrical signal that drives the speaker.

Integrated

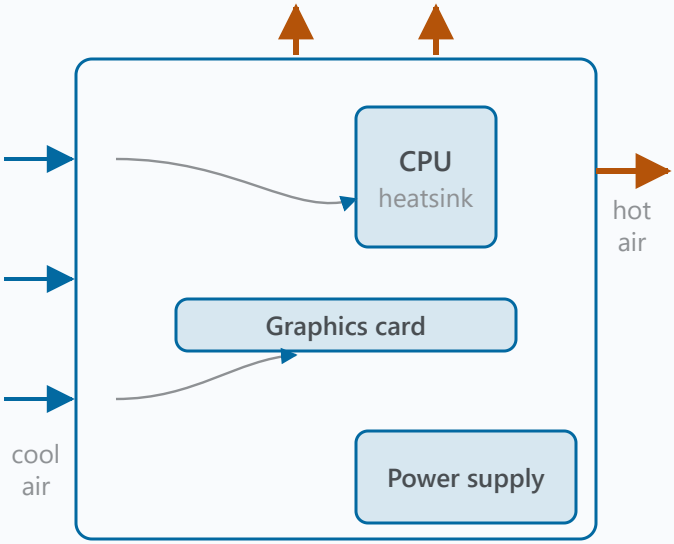
Most motherboards include audio good enough for everyday use.

External

For recording quality, more inputs or professional use.

SYSTEM COMPONENTS

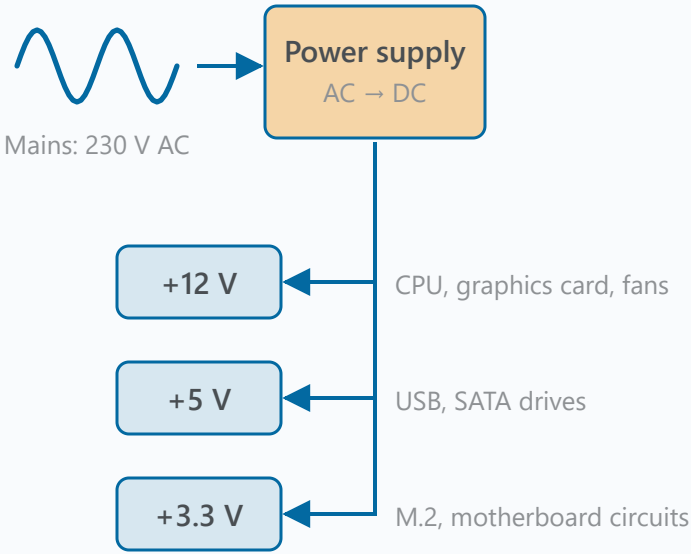
Case and Cooling



- **Protection:** shields components from physical damage.
- **Airflow:** cool air enters at the front, hot air leaves at the back/top.
- **Form factor:** determines which motherboard and power supply fit.
- **Cable management:** helps airflow and maintenance.

At high temperatures the processor slows itself down (thermal throttling), and component life can shorten.

The Power Supply (PSU)



Converts alternating current (AC) from the mains into the direct current (DC) voltages the components use.

- **Capacity (W):** must cover total demand with a safety margin.
- **Efficiency:** higher means less waste heat.
- **Connectors:** motherboard, CPU, GPU and storage cables must match.
- **Protection:** safeguards against overcurrent and voltage surges.

Input Devices

Keyboard
Text, numbers and commands.

Mouse / touchpad
Pointing, selecting and moving.

Microphone
Audio input.

Camera / scanner
Image and document input.

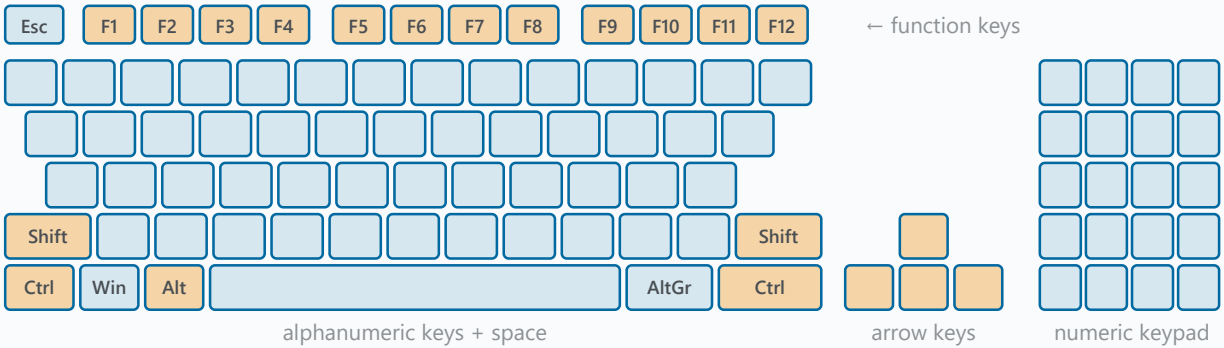
Game controller
Axis, button and motion data.

Sensor
Measures temperature, position, light or motion.

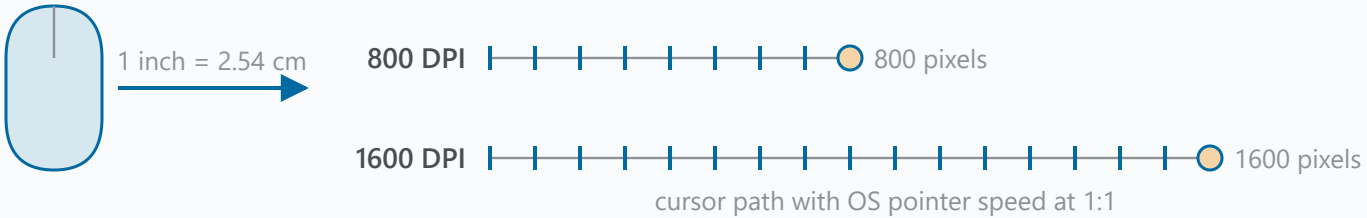
Keyboard

- Letter and number keys enter data.
- Function keys perform application-specific tasks.
- Ctrl, Alt and Shift combine with other keys to form shortcuts.
- Enter confirms; Backspace deletes the character left of the cursor, Delete the one to the right (or the selection).

```
Ctrl + C   Copy
Ctrl + V   Paste
Ctrl + Z   Undo
Alt + Tab  Switch apps
```



Mice and Pointing Devices



DPI

Steps reported per inch of movement; higher is not always better.

Polling rate

How many times per second the position is sent (e.g. 1000 Hz = every 1 ms).

Buttons and wheel

Selecting, context menus, scrolling and app commands.

Alternatives

Touchpad, stylus, trackball and touchscreen.

Output Devices

Monitor
Displays text, images and video.

Printer
Puts a digital document on paper.

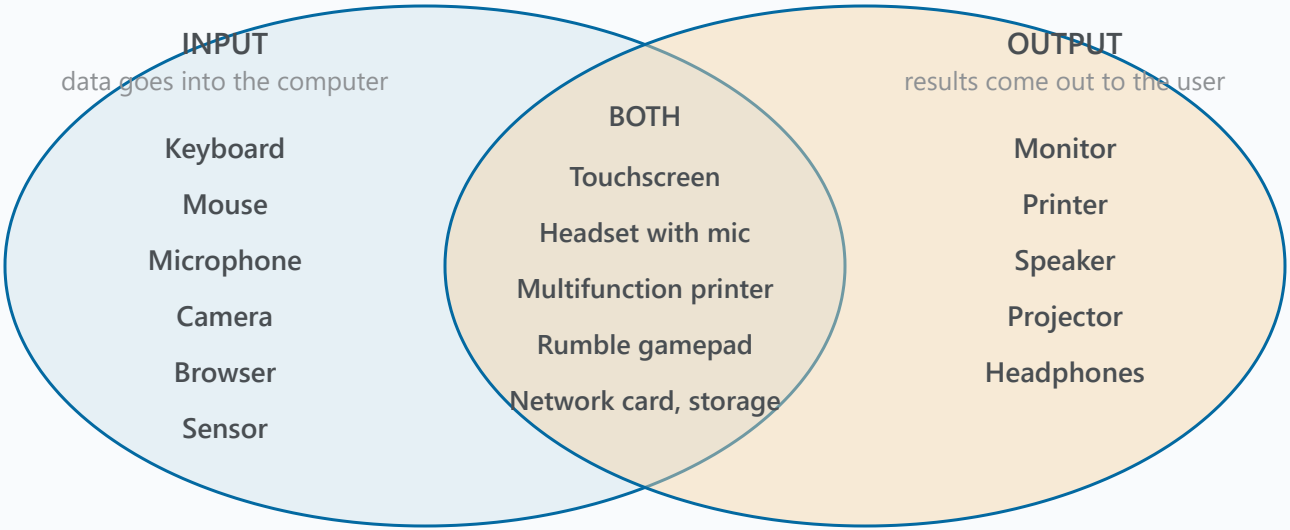
Speakers / headphones
Deliver audio to the user.

Projector
Projects an image onto a large surface.

Haptic feedback
Produces output through vibration or force.

Multifunction device
Can combine input and output in one device.

Classifying Input and Output Devices



The test is the direction of data flow: toward the computer is input, toward the user is output. Storage and network devices both read and write.

OUTPUT DEVICES

Monitors: Resolution and Size

Resolution

The number of pixels horizontally and vertically.

`1920 × 1080`

Screen size

The diagonal, measured in inches.

`24 inches`

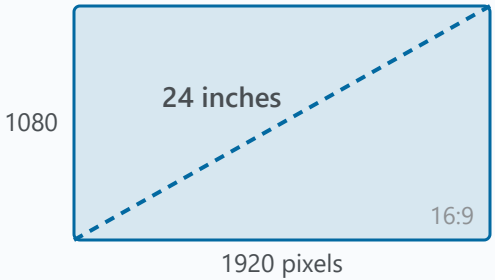
Pixel density

At the same resolution, a smaller screen packs pixels more tightly.

Aspect ratio

The ratio of screen width to height.

`16:9`



Pixels along the diagonal:

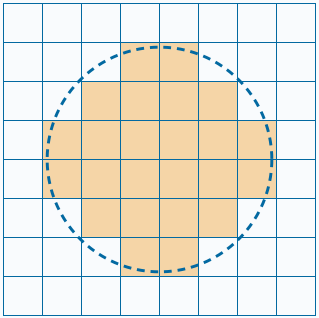
√(1920² + 1080²) ≈ 2203 pixels

Pixel density:

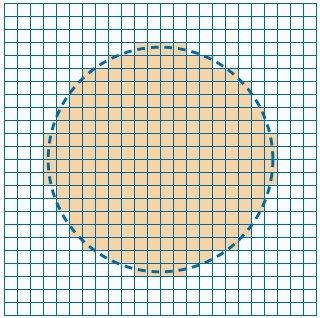
2203 ÷ 24 ≈ 92 PPI

The same 1920 × 1080 on a 15.6-inch laptop ≈ 141 PPI

The Pixel Grid and Pixel Density



8 × 8 = 64 pixels



24 × 24 = 576 pixels



1 pixel = red + green + blue

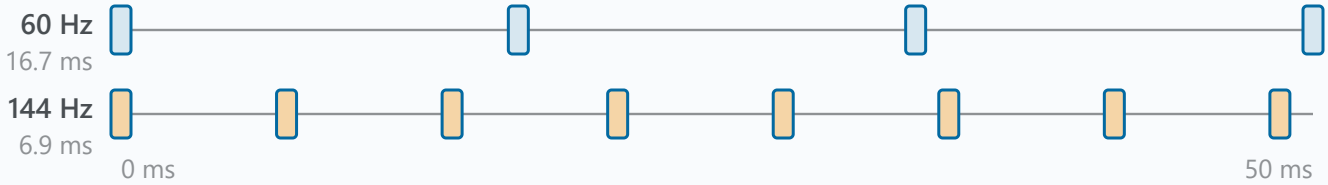
A screen is a grid of tiny square dots (pixels). **PPI** (pixels per inch):

```
diagonal pixels = √(1920² + 1080²)
                ≈ 2203
24 in   → 2203 / 24   ≈ 91.8 PPI
15.6 in → 2203 / 15.6 ≈ 141.2 PPI
```

The same resolution looks denser and sharper on a smaller screen.

Monitors: Motion and Image

Property	What it means
Refresh rate	How many times per second the screen redraws; measured in Hz.
Response time	How long a pixel takes to change from one state to another.
Brightness	Light output of the screen (cd/m^2).
Color accuracy	How close displayed colors are to their target values.
Viewing angle	How well the image holds up when viewed from the side.



Panel Technologies

IPS
Wide viewing angles and balanced colors.

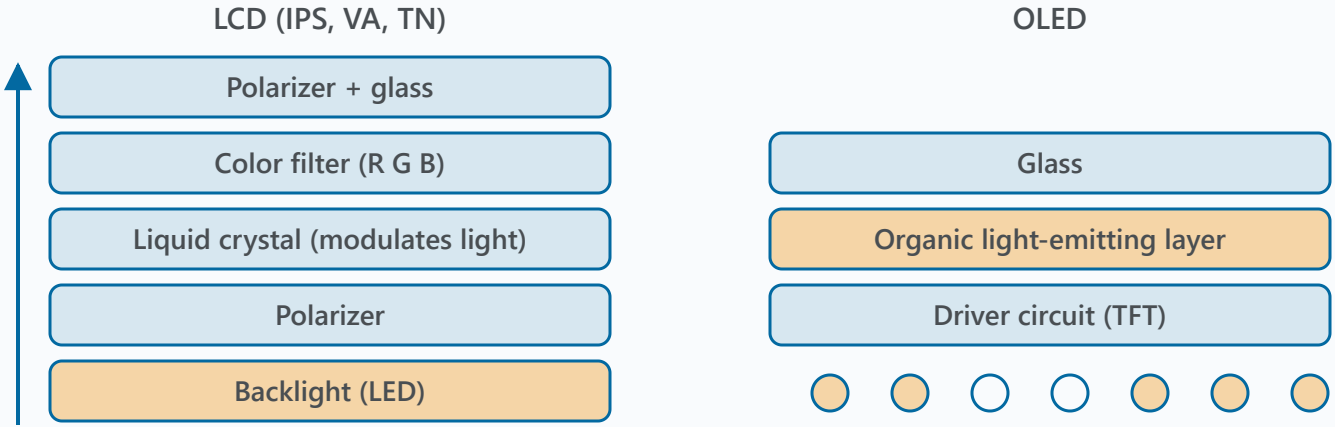
VA
High contrast and deep blacks.

TN
Fast response; viewing angles and colors can be more limited.

OLED
Each pixel emits its own light; very high contrast.

Choose a panel by use (office, design, gaming, movies) and budget.

How Do LCD and OLED Make Light?



black: light is blocked, but the backlight is always on

black pixel = pixel fully off

IPS, VA and TN are all LCD types; they differ in how the liquid crystals are arranged. OLED has no backlight, so black is truly black.

Printers, Audio and Projectors

Inkjet

Common for color documents and photos.

Laser printer

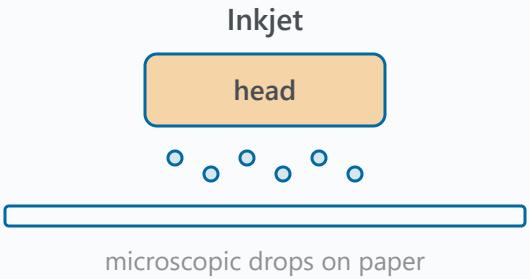
Can be fast and economical for high-volume text printing.

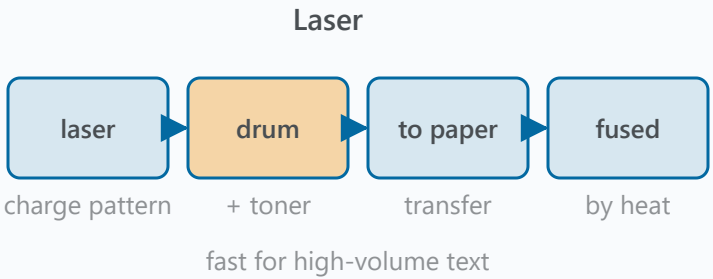
Speakers / headphones

Provide audio output over an analog or digital connection.

Projector

Chosen by brightness, resolution and throw distance.

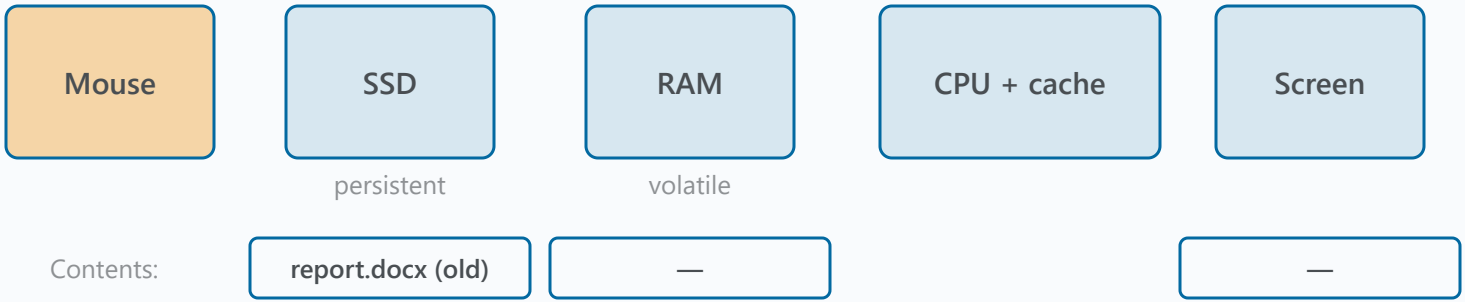




STORAGE

Storage Devices

Storage devices keep data when the power is off. When you open and save a document, the data travels like this:

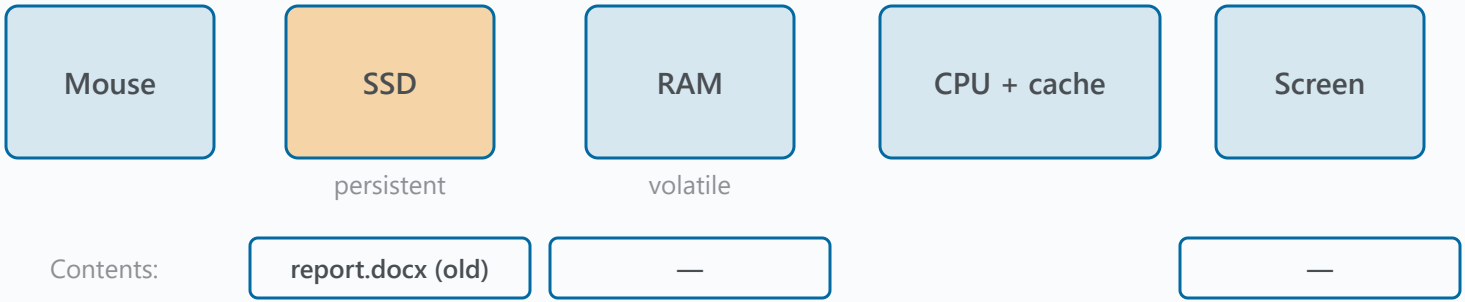


1. The user double-clicks **report.docx**; the mouse sends this event to the computer (input).

Step 1 / 6

Storage Devices

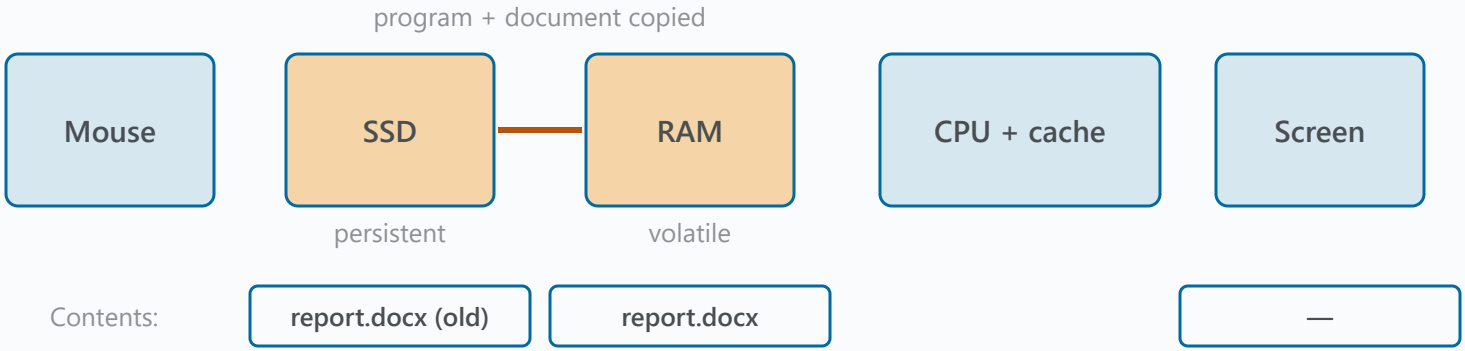
Storage devices keep data when the power is off. When you open and save a document, the data travels like this:



2. The operating system finds the program and document files in persistent storage (SSD).

Storage Devices

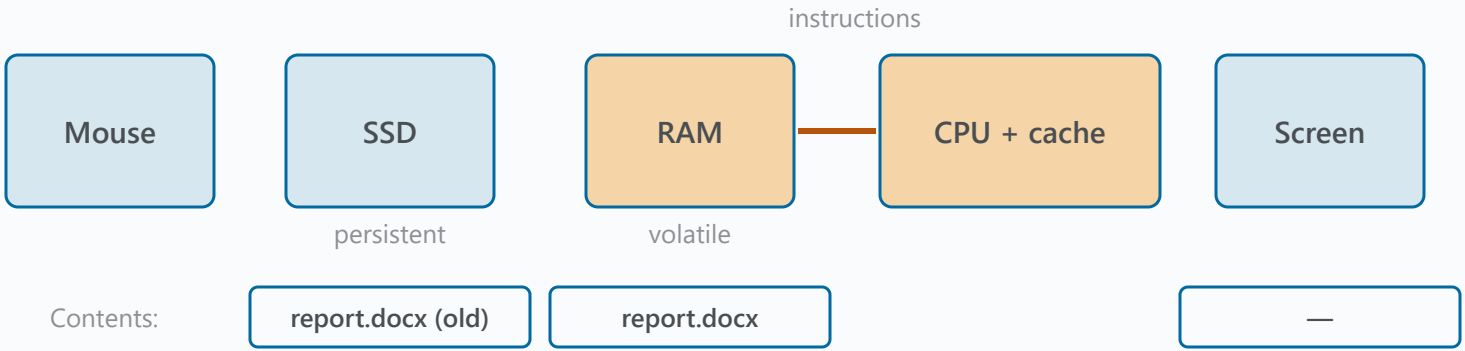
Storage devices keep data when the power is off. When you open and save a document, the data travels like this:



3. The program and document are copied from the SSD into RAM: the CPU runs from RAM, not directly from the SSD.

Storage Devices

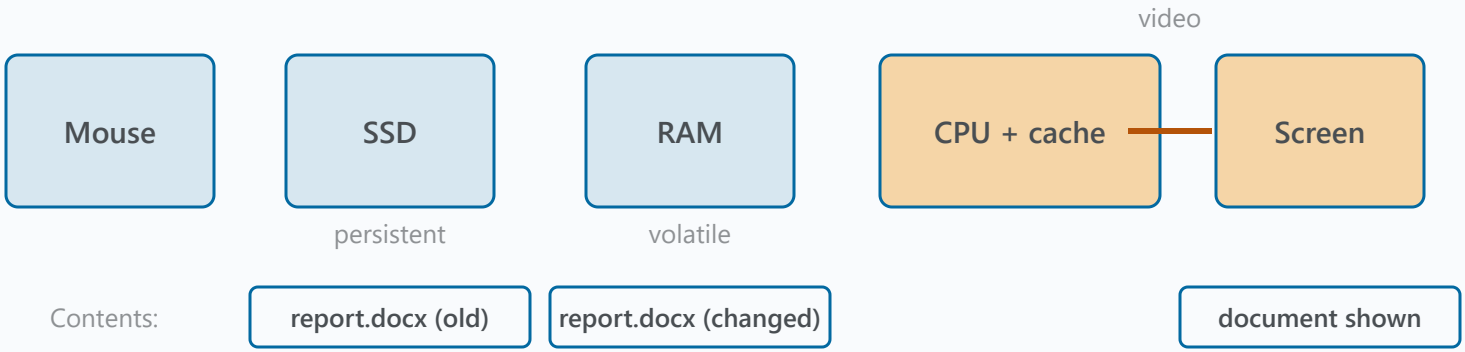
Storage devices keep data when the power is off. When you open and save a document, the data travels like this:



4. The CPU fetches instructions from RAM into the cache and registers and executes them (fetch–decode–execute).

Storage Devices

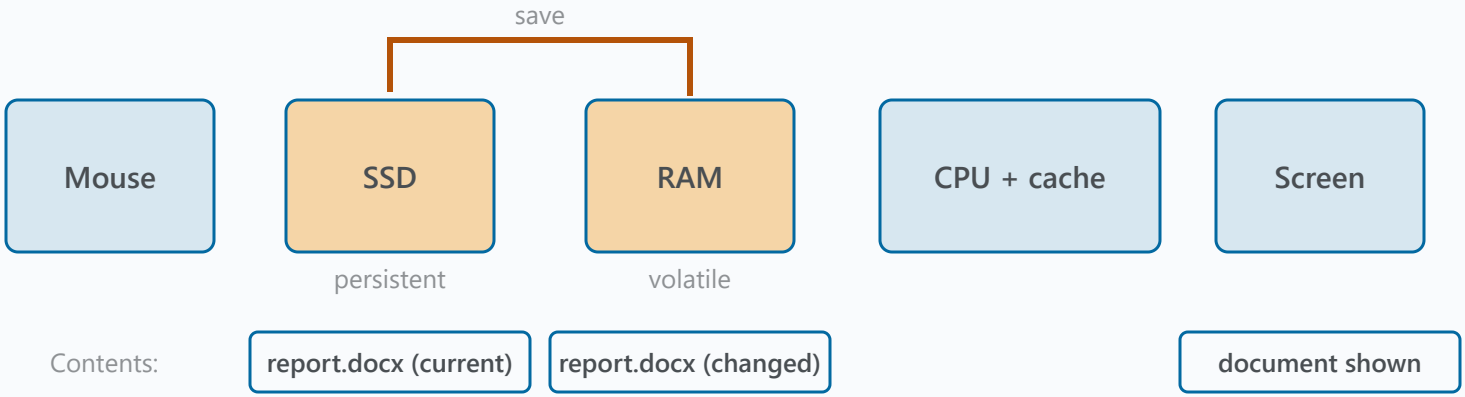
Storage devices keep data when the power is off. When you open and save a document, the data travels like this:



5. The result goes through the graphics card to the monitor; the user sees and edits the document (output).

Storage Devices

Storage devices keep data when the power is off. When you open and save a document, the data travels like this:



6. On “Save”, the changes are written from RAM to the SSD. Unsaved changes are lost if the power fails.

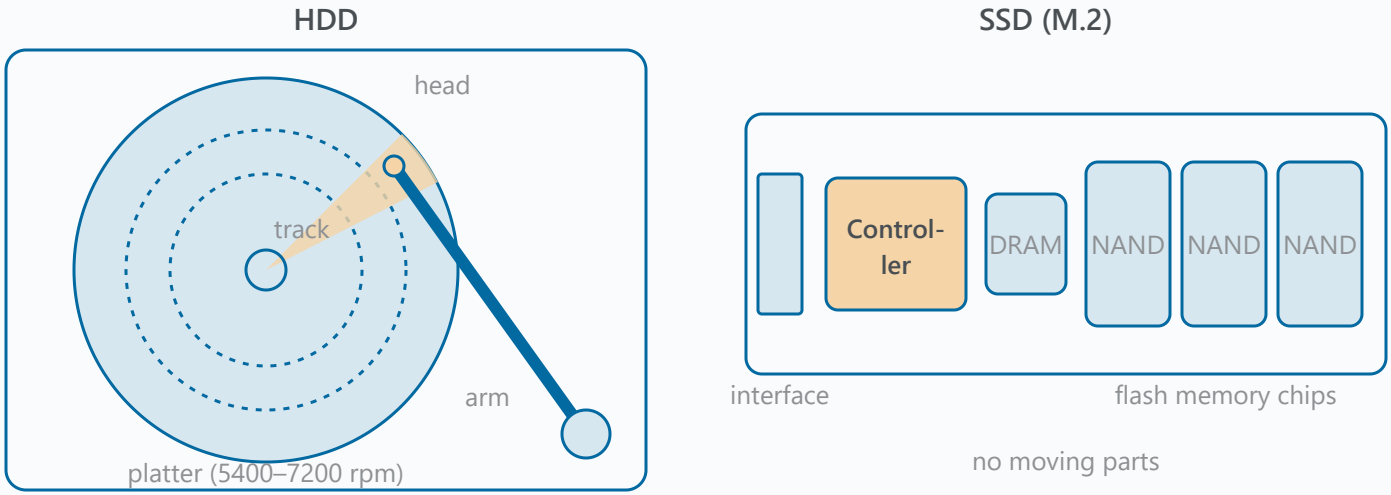
STORAGE

HDD and SSD

Property	HDD	SSD
Technology	Spinning magnetic platters	Flash memory
Speed	Lower	Higher
Shock resistance	More fragile because of moving parts	No moving parts
Cost per capacity	Usually lower	Usually higher
Common use	Archives and high capacity	Operating system and applications

STORAGE

Inside an HDD and an SSD

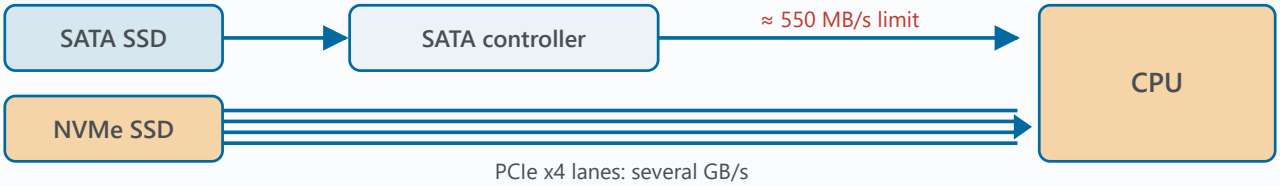


- **HDD:** the head must move to the right track and the sector must rotate under it → access takes milliseconds.
- **SSD:** the controller reads from several flash chips in parallel → access takes microseconds.

STORAGE

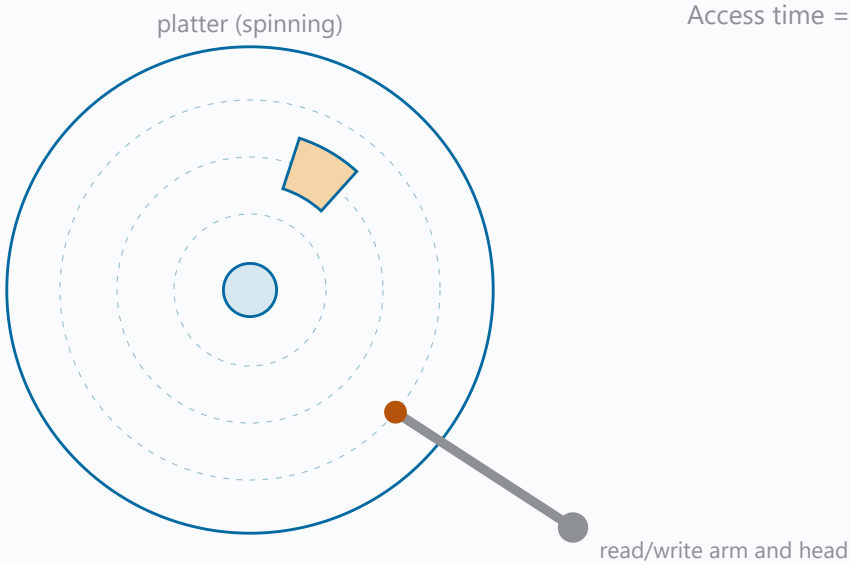
NVMe vs. SATA SSDs

Property	SATA SSD	NVMe SSD
Interface	SATA (the old hard-disk interface)	NVMe protocol over PCIe
Typical sequential read	about 550 MB/s (the SATA III limit)	several GB/s (depends on PCIe version and lane count)
Parallelism	A single command queue	Many parallel queues; low latency
Form factor	2.5-inch case or M.2	Mostly M.2 (U.2 in some servers)



M.2 is a form factor, not a protocol. An M.2 slot may support SATA, NVMe or both; the motherboard manual says which. Speeds are approximate; in real use, small files and heat can lower them.

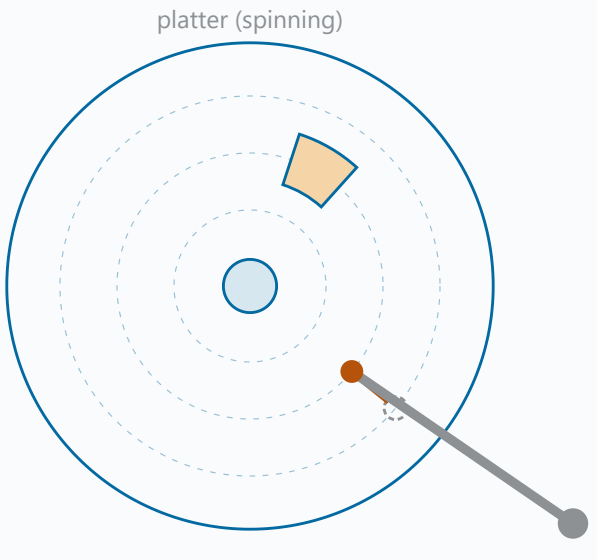
How an HDD Reads a Sector



Access time = seek + rotation + transfer

The platter spins thousands of times a minute (e.g. 7200 rpm). The **sector** to read (orange) and the head start in different places; access time has three parts.

How an HDD Reads a Sector



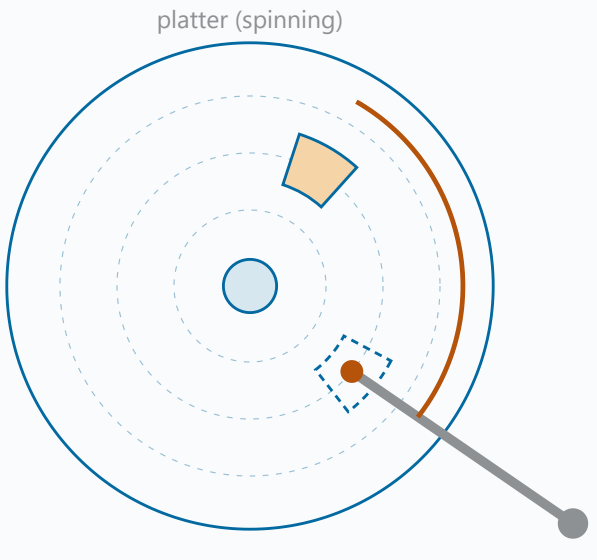
Access time = seek + rotation + transfer

Seek time ≈ 5 ms

head moves to target track

1) Seek time: the arm moves the head to the **track** holding the target sector. It is mechanical, so it takes a few milliseconds (5 ms here).

How an HDD Reads a Sector



Access time = seek + rotation + transfer

Seek time ≈ 5 ms

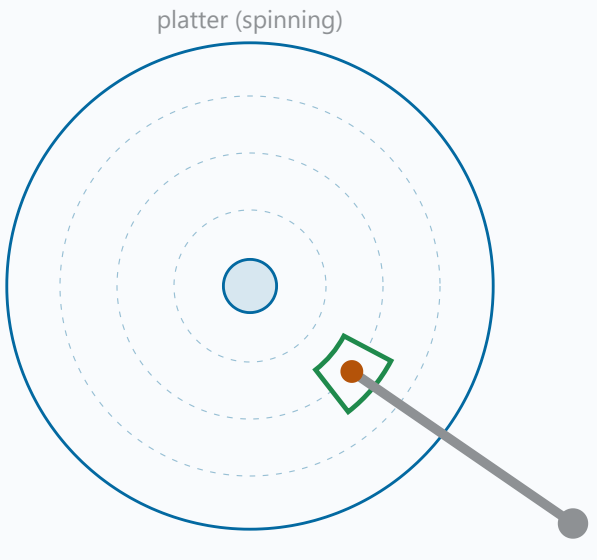
head moves to target track

Rotational latency ≈ 4.2 ms

7200 rpm: half a turn on average

2) Rotational latency: the head is on the right track, but the sector is not under it yet; we wait for the platter to turn. At 7200 rpm a full turn takes 8.3 ms, so the average wait (half a turn) is ≈ **4.2 ms**.

How an HDD Reads a Sector



Access time = seek + rotation + transfer

Seek time ≈ 5 ms

head moves to target track

Rotational latency ≈ 4.2 ms

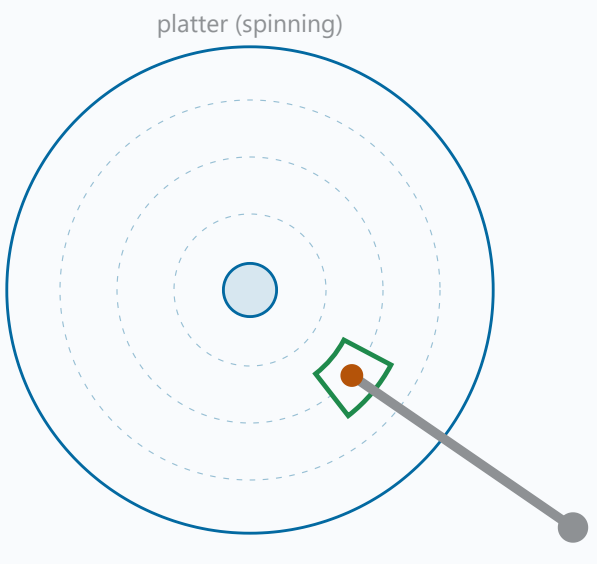
7200 rpm: half a turn on average

Transfer ≈ 0.03 ms

4 KB ÷ 150 MB/s

3) Transfer: data is read as the sector passes under the head. 4 KB ÷ 150 MB/s ≈ **0.03 ms**: the actual read is tiny compared with the waiting.

How an HDD Reads a Sector



Access time = seek + rotation + transfer

Seek time ≈ 5 ms

head moves to target track

Rotational latency ≈ 4.2 ms

7200 rpm: half a turn on average

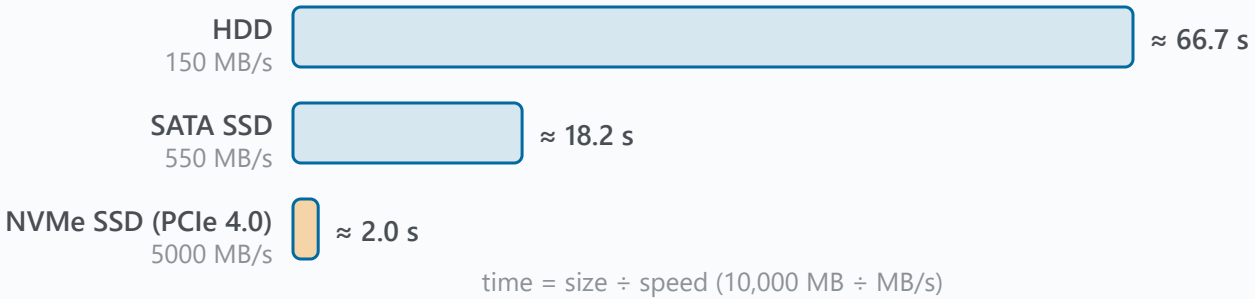
Transfer ≈ 0.03 ms

4 KB ÷ 150 MB/s

Total ≈ 9.2 ms

Total access ≈ 5 + 4.2 + 0.03 ≈ **9.2 ms**. Almost all of it is mechanical waiting, which is why many small files are slow on an HDD (an SSD has no such wait).

Worked Example: How Long Does a 10 GB Copy Take?



- These are typical sequential read speeds; with many small files an HDD slows down far more (head movement).
- Copying means reading and writing: the time depends on the slower of the source and target devices and on the link between them (USB, network).

A note on units: manufacturers use 1 GB = 1000 MB (SI); the operating system sometimes shows the 1024-based value (GiB) as “GB”, so the disk looks “smaller”.

STORAGE

Portable and Archival Storage

USB drive
Easy to carry; easy to lose or damage.

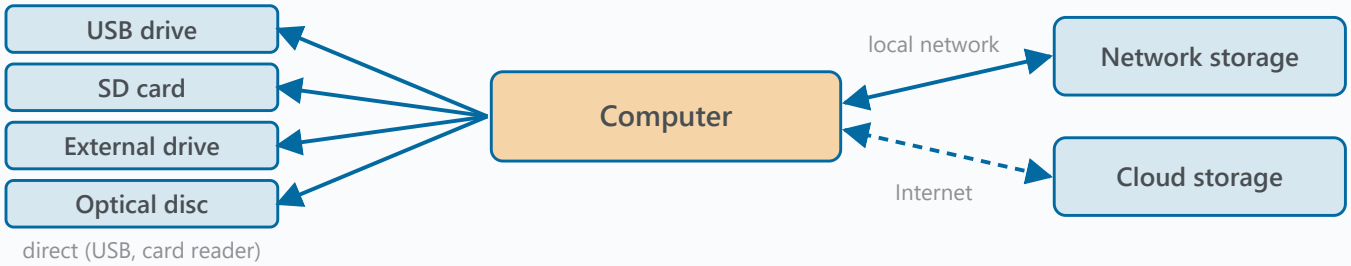
SD card
Common in cameras and mobile devices.

Optical disc
Used for distribution and some archiving.

External drive
Can offer high capacity for backups and moving data.

Network storage
Gives several users central access to files.

Cloud storage
Sync and sharing over the Internet.



SYSTEM DESIGN

Component Compatibility

Physical

Case, motherboard and card sizes must fit each other.

Electrical

The power supply must have enough capacity and the right connectors.

Logical

The CPU socket, memory generation and chipset must be supported.

Connections

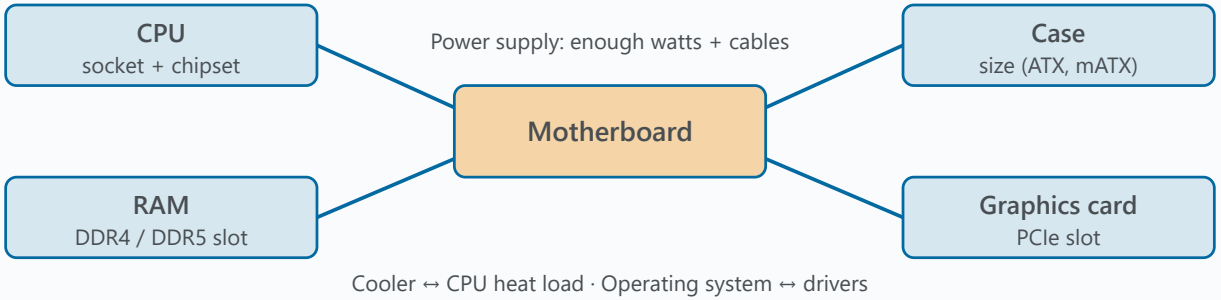
Ports, cables and protocol versions must be compatible.

Thermal

The cooling system must handle the CPU and GPU load.

Software

The operating system and drivers must support the hardware.



DDR5 and Current Standards

Standard	What changed?	Watch out
DDR4 → DDR5	Higher typical bandwidth and speed range (DDR5 modules start at higher speeds); on-module power management and two independent subchannels.	Physically different: a DDR4 module won't fit a DDR5 slot (or vice versa).
PCIe 3 → 4 → 5	Bandwidth per lane roughly doubles with each generation.	Slots are backward compatible; the speed is set by the **slower** of the slot and the device.
USB 3.2 → USB4	USB-C connector, 20/40 Gbit/s and Thunderbolt compatibility; USB4 Version 2.0 defines even higher speeds.	Not every USB-C port supports these speeds; check the port's specifications.
Wi-Fi 6 → 7	6 GHz band (with 6E), channels up to 320 MHz wide, multi-link operation (MLO).	Both the router and the device must support it.

Speeds vary by generation, product and configuration; compare motherboard and device specifications before buying.

Choosing by Use Case

Use	Priority components
Office and school	Balanced CPU, enough RAM, SSD, ergonomic display
Gaming	GPU, CPU, cooling, high-refresh-rate display
Content creation	Multi-core CPU, GPU, RAM, color accuracy, fast storage
Data analysis	RAM capacity, CPU/GPU and fast storage
Mobile use	Battery, weight, display, ports and energy efficiency

SUMMARY

Chapter Summary

- The CPU executes instructions; the motherboard connects the components.
- RAM is temporary working space; SSDs and HDDs provide persistent storage.
- The GPU handles parallel graphics and compute workloads.
- Input devices bring data into the system; output devices present results to the user.
- When choosing hardware, compatibility, power, cooling and purpose matter as much as performance.

Review Questions

1. What is the key difference between RAM and an SSD?
2. What kinds of compatibility does the motherboard determine?
3. When would you choose an integrated GPU, and when a discrete one?
4. Do refresh rate and response time measure the same thing?
5. What should you look at when choosing a power supply?
6. How do the priorities of an office PC and a gaming PC differ?

SUMMARY

Exercises

1. Compute the PPI of a 27-inch, 2560 × 1440 monitor.
2. How many seconds does it take to copy an 8 GB video to an SSD at 500 MB/s?
3. How many ms does each frame stay on screen at 144 Hz? Compare with 60 Hz.
4. CPU 125 W, GPU 220 W, everything else 75 W. With a 30% margin, what power supply do you choose?
5. Touchscreen, scanner, projector, headset with mic, SSD: input, output or both?
6. In what order does data move between the SSD, RAM, cache and CPU when a document is opened and saved?

▶ **Answers**

References

- Patterson and Hennessy, Computer Organization and Design.
- Stallings, Computer Organization and Architecture.
- PCI-SIG, USB-IF, VESA and JEDEC technical standards.

The End

Chapter 1.2: Computer Hardware

Previous: Chapter 1.1: Basic Concepts · Next: Chapter 1.3: Computers