CPU, Cores, Cache, and Prefetching - Complete OS Concept Guide

CPU (Central Processing Unit) Definition

The CPU is the main processing unit of a computer that:

- Executes instructions
- Performs calculations
- Controls data movement between memory and peripherals

Its often called the brain of the computer.

CPU Core What is it?

A core is an independent execution unit within the CPU.

Each core has:

- Its own registers
- Its own ALU
- Usually its own L1 and L2 cache

Types of CPUs

CPU Type	Description
Single-core	One core executes one task at a time
Multi-core	Multiple cores can run multiple tasks/threads in parallel

Example:

If a CPU has 4 cores, it can truly run 4 threads simultaneously (if the OS schedules it that way).

CPU Cache (L1, L2, L3)

Caches are small, very fast memory units located inside or very close to the CPU cores.

They store frequently used data and instructions to avoid slow RAM access.

Cache Level	Where It Is	Size	Speed	Who Uses It
L1	Inside each core	3264 KB	Fastest (~1ns)	Only that core
L2	Inside/near core	256 KB1 MB	Fast (~3ns)	Only that core
L3	Shared among cores	230 MB	Slower (~10ns)	Shared by all cores

L1 is closest to the execution unit.
L3 is farther but still much faster than RAM.

What is Prefetching in Caching?

Prefetching is a technique used by the CPU to guess what data or instructions will be needed soon and load them into cache before they are actually requested.

Why Prefetch?

- RAM access is slow
- CPU doesnt want to wait when data is needed

So it says:
If I just accessed memory at address A, I will probably need A+1, A+2, etc. next so let me fetch them in advance.

Types of Prefetching

Type	Description
Hardware prefetching	CPU logic automatically detects patterns and prefetches nearby data
Software prefetching	Compilers or programmers add special instructions (e.g., __builtin_prefetch() in

C/C++)

Instruction prefetching | CPU fetches next instructions into cache as part of pipelining

Real Example (Array Access)

```
for (int i = 0; i < 1000; i++) {
    sum += arr[i];  // CPU may prefetch arr[i+1], arr[i+2], ...
}
```

Since the access is sequential, the CPU prefetches the next few elements.

This reduces cache misses.

How Prefetching Helps in Cache Hierarchy

- Data is fetched into L1 cache if there's room.
- If L1 is full, it may go to L2 or L3.
- This way, when the CPU needs data:
  - Its already in a nearby cache (not in slow RAM).

Result: Faster execution

Cache Miss (One-line Definition)

A cache miss occurs when the CPU looks for data in the cache but doesn't find it, so it must fetch it from slower memory like RAM.

CPU Affinity Related OS Concept

CPU affinity is the ability to bind a thread or process to a specific core.

Benefits:

- Better use of L1/L2 cache (if a thread stays on the same core)
- Fewer cache misses
- More predictable performance in real-time systems

Example:

In Linux, you can set CPU affinity using:

```
taskset -c 2 ./my_program
```

This runs my_program only on core 2.

All these concepts are key to writing efficient multithreaded programs, understanding OS behavior, and optimizing for modern CPUs.