clock frequence = how fast a processor can perform tasks

1 GHz = processor ticks 1 billion times per second

GFLOPS = 10^9 floating point operations per second } 1980s 1990s

TFLOPS = 10^{10} " "

But increase in speed => increase in heat & energy dissipation issues.

1 CPU => multiple cores, work simultaneously

sequence => multiple sequences

} workaround for

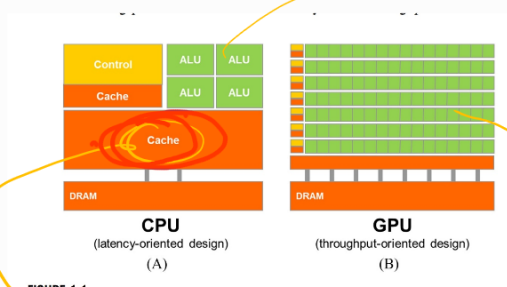
program counter (instruction pointer) — memory address of next instruction → sequence = thread of execution

parallel programs = multiple threads together = speed stonk → parallel over sequential = CONCURRENCY REVOLUTION

X86 = CPU design by Intel (8086 chip debut)

X86 instruction set = collection of machine language instructions that X86 compatible processors can understand & execute.
(Arithmetic calculations, data movement, control flow, I/O interactions)
Backward compatible.

hyperthreading = ONE physical CPU acting like TWO virtual CPU.
DOESNT double the speed, but makes multitasking smooth.



optimized for sequential code perf.
reduced latency at the cost of chip size & power per unit.
↓
latency oriented design.

maximize # of floating point operations & memory access throughput.

stores frequently accessed data.

GPU: lots of ops = better graphics.

Moore's law:- the # of transistors double every 2 years at the cost of computing being halved.

denard's law:-

as the size of transistor shrink:-
→ voltage reduces proportionally (V)
→ current " " (I)
→ capacitance " " (C)

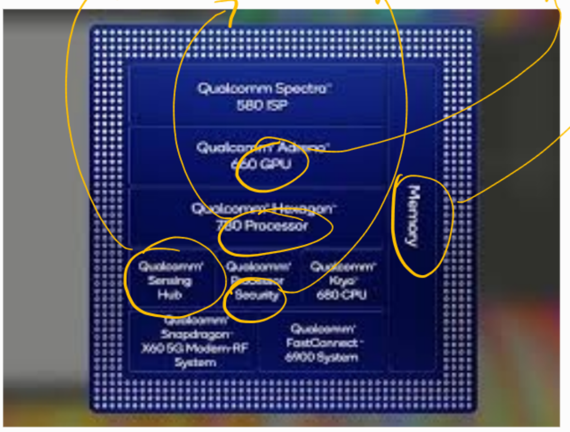
delay = $\frac{CV}{I}$

as delay ↓, performance/transistor should be boosted and scales up by α^2.

⇒ same 10W chip is more performant few years later.

but not anymore. → multicore introduced, threaded appns

processors are not just processors anymore, they are processor + memory + gpu + security ....



→ CPUs are latency driven
→ GPUs are throughput oriented; get as much done in the silicon as possible; cores as simple as possible but soooo many ALUs that are energy efficient. massive # of threads.

Amdahl's law:- % of speedup that can be achieved overall is limited by % of code that can be parallelizable.

→ Another limitation is how often memory needs to be accessed from DRAM. Usually done by using the memory in GPU.

→ Open MP & MPI (message passing interface) are other interfaces.

OpenCL (nvidia + AMD + intel etc.) is another interface but relies heavily on APIs.