CSC 3210 Computer organization and programming

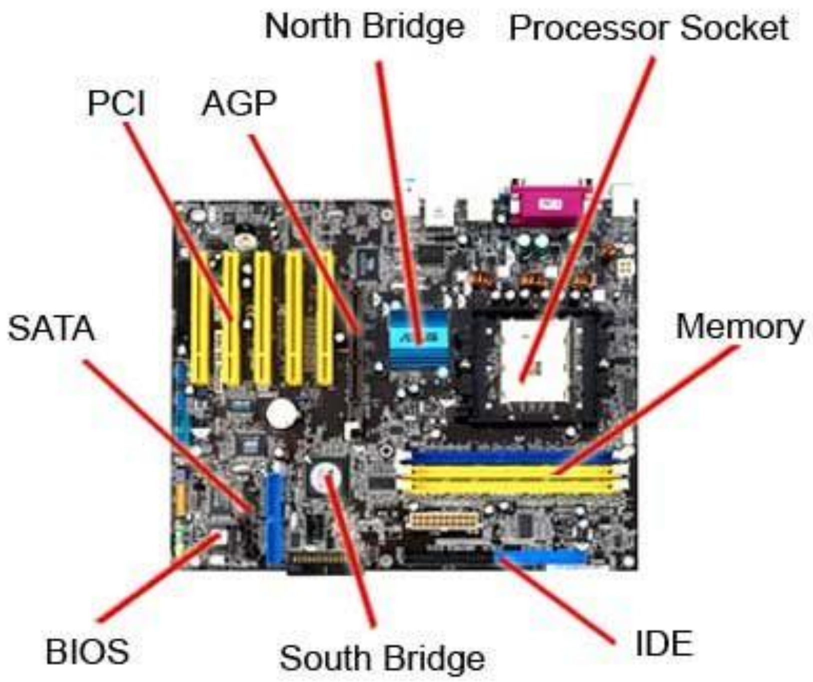
Chunlan Gao

Georgia State University

Chapter1

Computer Abstractions and Technology

In a PC ?



1.Processor Socket: This is where the **CPU (Central Processing Unit)** is installed. The socket type determines the kind of CPU the motherboard can support.

2. Memory (RAM) Slots: These are long slots where **RAM (Random Access Memory)** modules are installed. They provide temporary storage for the CPU to access data quickly during operation.

3. North Bridge: The **North Bridge** is a chipset component that handles high-speed communication between the CPU, RAM, and graphics card， It's located near the CPU and RAM slots because it needs to provide fast access.

4.South Bridge: The **South Bridge** is another chipset component that manages lower-speed peripherals such as SATA devices, USB, and audio. It handles communication between these devices and the CPU through the North Bridge.

5. PCI Slots: **PCI (Peripheral Component Interconnect)** slots are used to install expansion cards, such as network cards, sound cards, and older graphics cards.

6. AGP Slot: **AGP (Accelerated Graphics Port)** is a high-speed slot used for graphics cards. This slot was common in older motherboards before PCI Express became the standard.

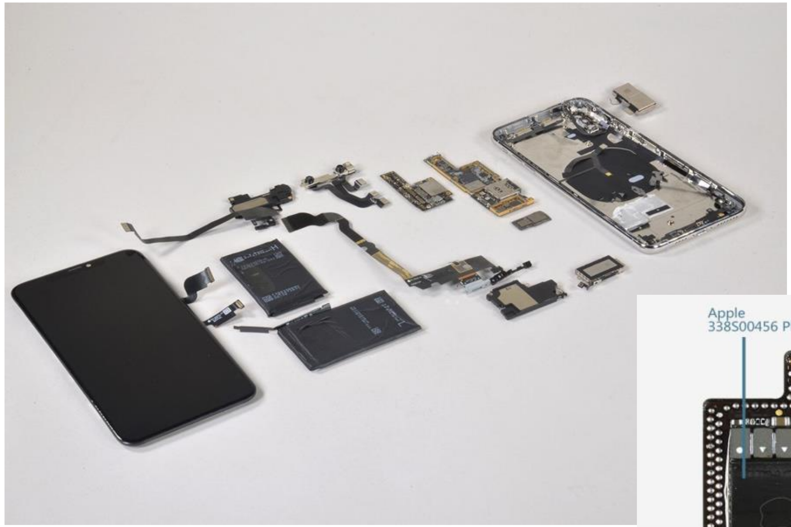
7. SATA Connectors: These are used to connect **SATA (Serial ATA)** devices, such as hard drives and solid-state drives (SSDs), to the motherboard.

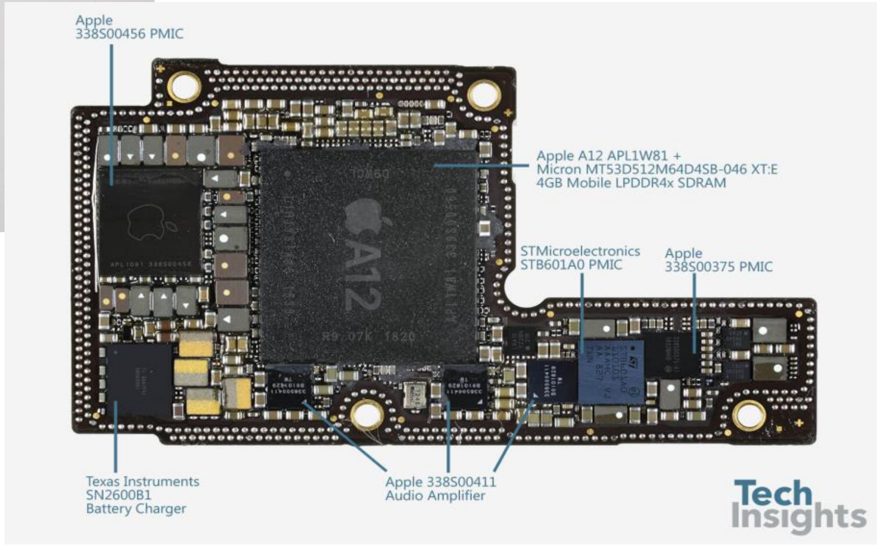
8. IDE Connector: The **IDE (Integrated Drive Electronics)** connector is used to connect older hard drives and optical drives. It has largely been replaced by SATA in modern systems.

9. BIOS Chip: the **BIOS (Basic Input/Output System)** chip contains the firmware needed to initialize hardware during booting and provides basic control over system settings.

10. I/O Ports:These are located on the rear of the motherboard and provide connectivity for external devices (e.g., USB, audio, video, Ethernet).

Teardown of a smartphone



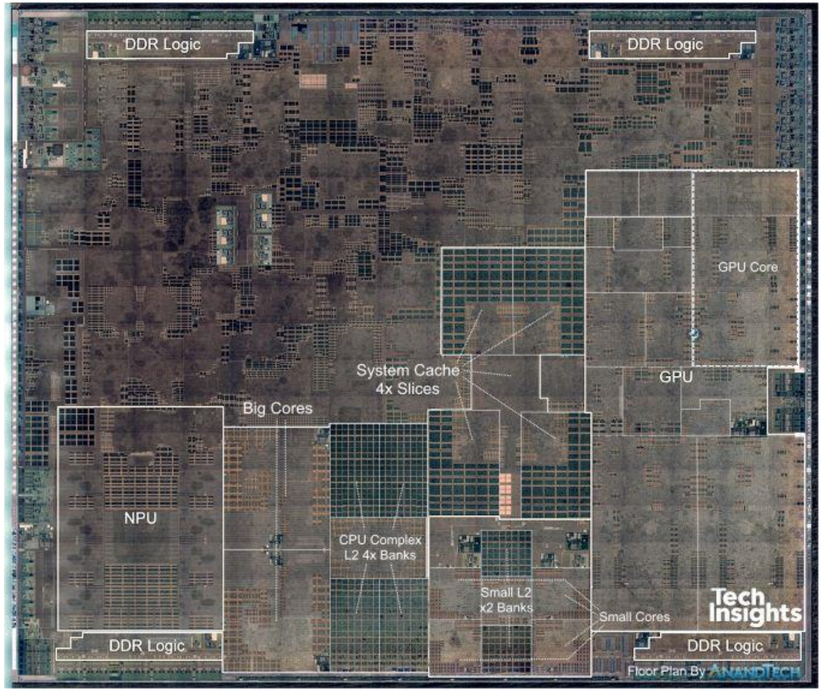


Teardown of a smartphone

- **Apple A12** is a custom-designed system-on-a-chip (SoC) used in products like the iPhone XS, XS Max, XR, and certain iPads.
- This image shows a **printed circuit board (PCB)** with key components labeled, likely from a mobile or embedded device, such as a smartphone or tablet. Based on the central component labeled **"A12"**, this PCB appears to belong to an **Apple device**, as the **Apple A12** is a custom-designed system-on-a-chip (SoC) used in products like the iPhone XS, XS Max, XR, and certain iPads.
- **Key components identified:**

1. **Apple A12 SoC**: The central processor chip, integrating the CPU, GPU, Neural Engine, and other components critical for the device's operation. Fabricated using a 7nm process, offering high performance and power efficiency.
2. **Micron LPDDR4X SDRAM**: The memory chip, providing 4GB of mobile DRAM, used by the A12 SoC for high-speed data processing.
3. **Power Management ICs (PMIC)**: Several power management chips from **Apple** and **Texas Instruments**, responsible for efficiently distributing power to different parts of the board.
4. **Audio Amplifier**: A component from **STMicroelectronics**, likely used to drive the speakers.
5. **Battery Charger IC**: A battery management chip from **Texas Instruments**, responsible for managing the charging process and battery health.

A12 microprocessor



1. **Big Cores**: These are high-performance CPU cores designed for tasks requiring significant computational power. They handle demanding workloads and ensure smooth performance for tasks like gaming, video editing, and multitasking.
2. **Small Cores**: These are low-power CPU cores optimized for energy efficiency. They handle less demanding tasks, such as background processes, to conserve battery life.
3. **CPU Complex (L2 4x Banks)**: This refers to the CPU cluster, which includes shared **L2 cache** banks. The L2 cache stores frequently accessed data to speed up processing and reduce latency.
4. **Small L2 Banks**: These smaller L2 cache banks may be dedicated to specific small cores or serve as additional cache for less demanding operations.
5. **System Cache (4x Slices)**: A large shared cache that all cores can access. It helps reduce memory bottlenecks by storing frequently used data closer to the cores, improving overall system performance.
6. **GPU Core**: This is the **graphics processing unit (GPU)**, responsible for rendering images, video, and animations. It's critical for tasks like gaming, video playback, and graphical user interfaces.
7. **GPU**: This section includes additional GPU components or execution units. These handle parallel workloads required for modern graphical and compute-intensive applications.
8. **NPU (Neural Processing Unit)**: A dedicated unit for **machine learning and AI tasks**. It accelerates processes like image recognition, natural language processing, and other AI-driven features.
9. **DDR Logic**: These sections interface with **DDR memory (RAM)**. They handle data transfer between the SoC and the memory, ensuring fast communication and efficient memory usage.

Computer performance？

- **Response time**
- **Throughput**

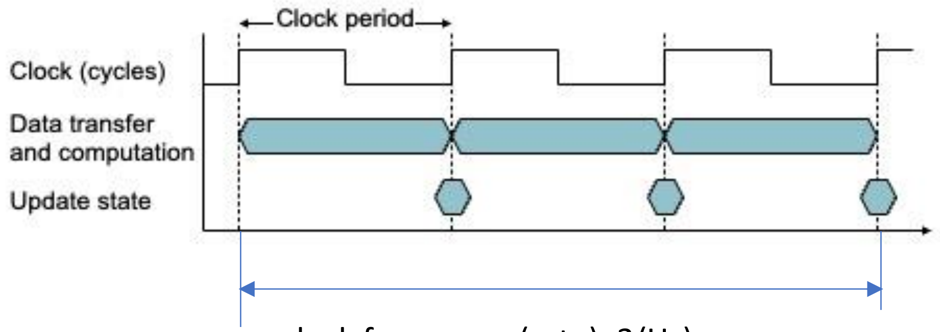
Performance compare:

"X is n time faster than Y" means:

P_X/P_Y = Execution Time $_Y$/ Execution Time $_X$ = n

CPU locking:

clock cycle time(ps, ns, s)



clock frequency (rate)=3(Hz)

$$\text{Clock Cycle Time} = \frac{1}{\text{Clock Frequency}}$$

1. **Kilohertz (kHz)**
   - **Definition**: 1 kHz = 1,000 Hz
   - **Scientific Notation**: $1\,\text{kHz} = 1 \times 10^3\,\text{Hz}$
2. **Megahertz (MHz)**
   - **Definition**: 1 MHz = 1,000,000 Hz
   - **Scientific Notation**: $1\,\text{MHz} = 1 \times 10^6\,\text{Hz}$
3. **Gigahertz (GHz)**
   - **Definition**: 1 GHz = 1,000,000,000 Hz
   - **Scientific Notation**: $1\,\text{GHz} = 1 \times 10^9\,\text{Hz}$

CPU Time

$$\text{CPU Time} = \text{CPU Clock Cycles} \times \text{Clock Cycle Time}$$

$$= \frac{\text{CPU } clock\ cycles}{Clock\ Rate}$$

CPU Time Example

- Computer A: 2GHz clock, 10s CPU time
- Designing Computer B
  - Aim for 6s CPU time
  - Can do faster clock, but causes 1.2 × clock cycles$_A$
- How fast must Computer B clock be?

Computer	CPU Time	CPU rate	Clock Cycles used	Put data in formula
A	12	2 GHz	Clock Cycles for A	12= $\frac{\text{Clock Cycles for A}}{2GHz}$
B	6	?	1.2 *Clock Cycles for A	6= $\frac{1.2*\text{Clock Cycles for A}}{?}$

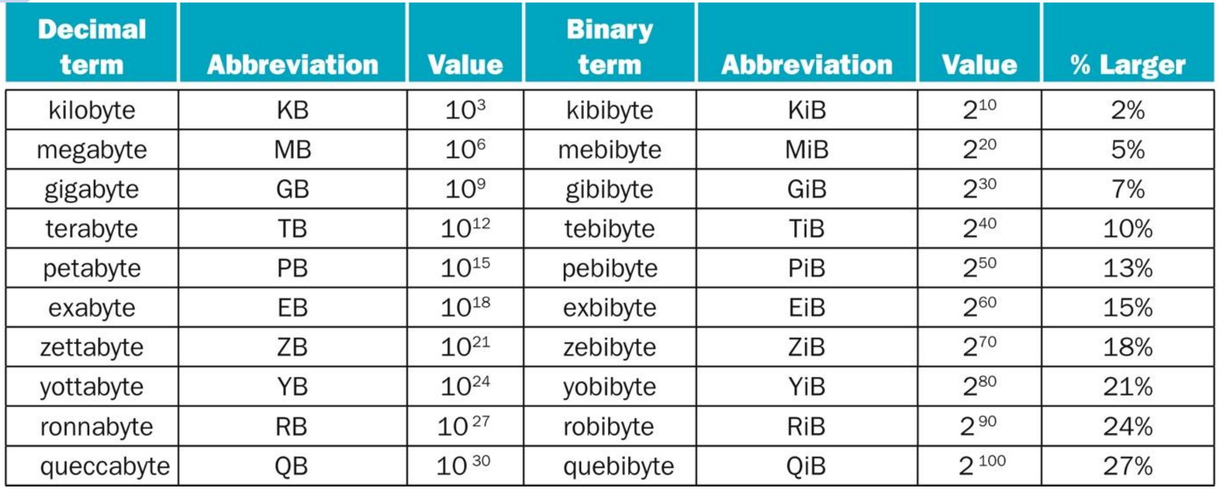
Electronics technology trends

Year	Technology	Relative performance/cost
1951	Vacuum tube	1
1965	Transistor	35
1975	Integrated circuit (IC)	900
1995	Very large scale IC (VLSI)	2,400,000
2013	Ultra large scale IC	250,000,000,000

Electronics technology trends

- https://www.youtube.com/watch?v=dX9CGRZwD-w&t=154s

- **Bit:** most basic unit of information in computing and digital communication. it is a contraction of "binary digit" and represents a logical state with one of two possible values: **0** or **1**.
- **Byte:** A group of eight bits is called a **byte**, which is the standard unit for measuring data storage. (00001110)



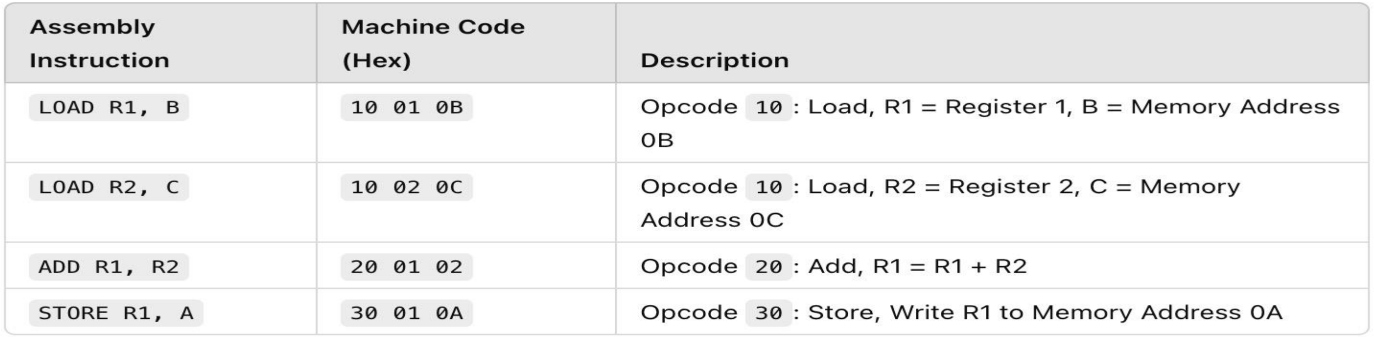
Decimal term	Abbreviation	Value	Binary term	Abbreviation	Value	% Larger
kilobyte	KB	10^3	kibibyte	KiB	2^{10}	2%
megabyte	MB	10^6	mebibyte	MiB	2^{20}	5%
gigabyte	GB	10^9	gibibyte	GiB	2^{30}	7%
terabyte	TB	10^{12}	tebibyte	TiB	2^{40}	10%
petabyte	PB	10^{15}	pebibyte	PiB	2^{50}	13%
exabyte	EB	10^{18}	exbibyte	EiB	2^{60}	15%
zettabyte	ZB	10^{21}	zebibyte	ZiB	2^{70}	18%
yottabyte	YB	10^{24}	yobibyte	YiB	2^{80}	21%
ronnabyte	RB	10^{27}	robibyte	RiB	2^{90}	24%
queccabyte	QB	10^{30}	quebibyte	QiB	2^{100}	27%

Figure 1.1 The 2^X vs. 10^Y bytes ambiguity was resolved by adding a binary notation for all the common size terms. In the last column we note how much larger the binary term is than its corresponding decimal term, which is compounded as we head down the chart. These prefixes work for bits as well as bytes, so *gigabit* (Gb) is 109 bits while *gibibits* (Gib) is 230 bits. The society that runs the metric system created the decimal prefixes, with the last two proposed only in 2019 in anticipation of the global capacity of storage systems. All the names are derived from the entymology in Latin of the powers of 1000 that they represent.

Instruction& Instruction set

An **instruction** is a single operation that a CPU can perform. It is the smallest unit of execution in machine-level programming.

A=C+B



Assembly Instruction	Machine Code (Hex)	Description
`LOAD R1, B`	`10 01 0B`	Opcode `10` : Load, R1 = Register 1, B = Memory Address 0B
`LOAD R2, C`	`10 02 0C`	Opcode `10` : Load, R2 = Register 2, C = Memory Address 0C
`ADD R1, R2`	`20 01 02`	Opcode `20` : Add, R1 = R1 + R2
`STORE R1, A`	`30 01 0A`	Opcode `30` : Store, Write R1 to Memory Address 0A

Instruction Set: The list of operations the CPU can perform, such as:

1. Arithmetic operations (ADD, SUB, MUL)
2. Data movement (LOAD, STORE)
3. Control flow (JUMP, CALL)

Instruction Count and CPI

- Clock Cycles = Instruction Count * Cycles per Instruction(CPI)
- CPU time = Instruction Count * CPI* Clock Cycle Time

  = Instruction Count * CPI / Clock Rate

- Instruction Count for a program
  - Determined by program, ISA (Instruction Set Architecture)and compiler
- Average cycles per instruction
  - Determined by CPU hardware
  - If different instructions have different CPI
    - Average CPI affected by instruction mix

CPI Example

- Computer A: Cycle Time = 250ps, CPI = 2.0
- Computer B: Cycle Time = 500ps, CPI = 1.2
- Same instruction amount I.
- Which is faster, and by how much?

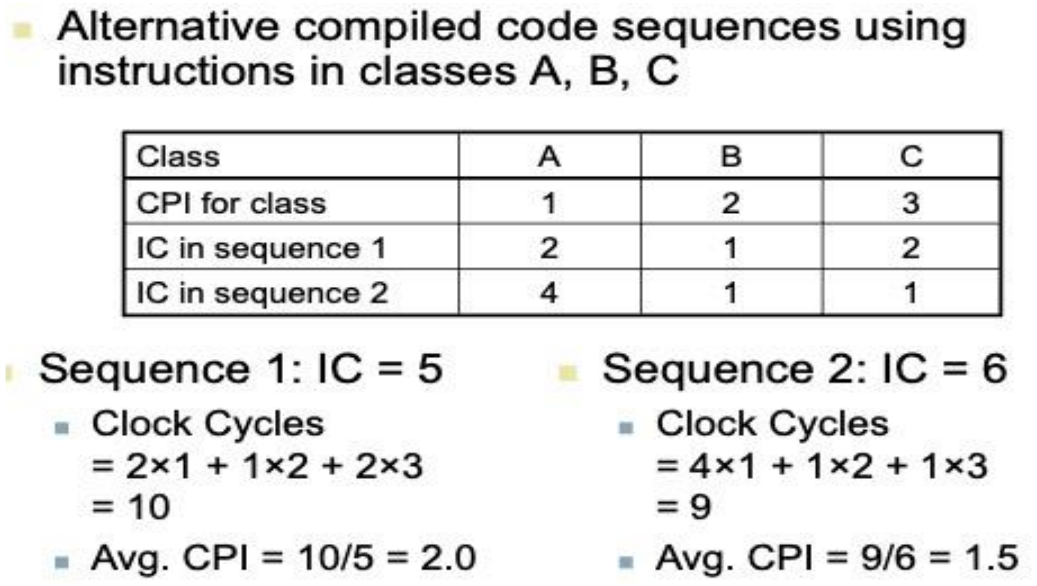
$$\begin{aligned}
\text{CPU Time}_A &= \text{Instruction Count} \times \text{CPI}_A \times \text{Cycle Time}_A \\
&= \text{I} \times 2.0 \times 250\text{ps} = \text{I} \times 500\text{ps} \\
\text{CPU Time}_B &= \text{Instruction Count} \times \text{CPI}_B \times \text{Cycle Time}_B \\
&= \text{I} \times 1.2 \times 500\text{ps} = \text{I} \times 600\text{ps} \\
\frac{\text{CPU Time}_B}{\text{CPU Time}_A} &= \frac{\text{I} \times 600\text{ps}}{\text{I} \times 500\text{ps}} = 1.2
\end{aligned}$$

CPI details

- If different instruction classes take different numbers of cycles
- Clock Cycles = $\sum_{i=1}^{n} CPI_i \ * \text{Instruction Count}_{\ i}$
- Weighted average CPI
- CPI = $\frac{Clock\ Cycles}{Insrtuction\ Count}$ =$\sum_{i=1}^{n} CPI_i \ * \frac{Instruction\ Count_{\ i}}{Instrution\ Count}$

CPI Example

- Alternative compiled code sequences using instructions in classes A, B, C



- Alternative compiled code sequences using instructions in classes A, B, C

Class	A	B	C
CPI for class	1	2	3
IC in sequence 1	2	1	2
IC in sequence 2	4	1	1

- Sequence 1: IC = 5
  - Clock Cycles
    = 2×1 + 1×2 + 2×3
    = 10
  - Avg. CPI = 10/5 = 2.0
- Sequence 2: IC = 6
  - Clock Cycles
    = 4×1 + 1×2 + 1×3
    = 9
  - Avg. CPI = 9/6 = 1.5

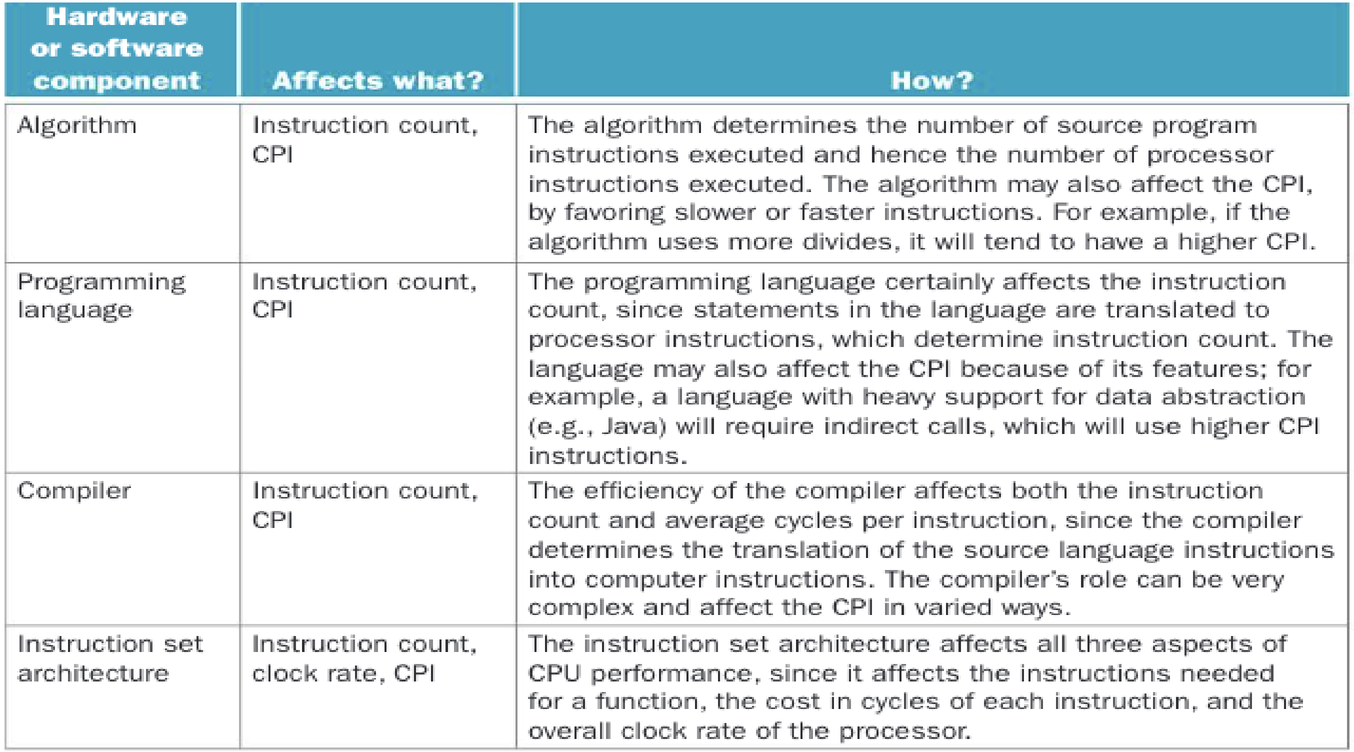
Summery of Performance

$$\text{CPU Time} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock Cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clock Cycle}}$$

This formula represents the total CPU execution time for a program and breaks it down into three key components:

1. **Instructions per program**: The total number of instructions executed in the program.
2. **Clock cycles per instruction (CPI)**: The average number of clock cycles required for each instruction.
3. **Seconds per clock cycle**: The duration of one clock cycle, which is the reciprocal of the CPU clock frequency.

What key factors affect performance？

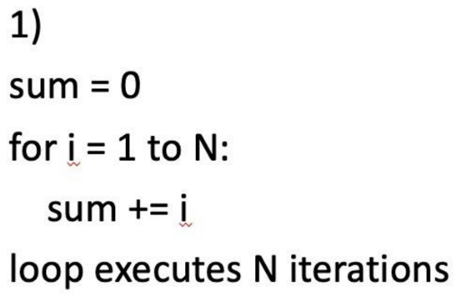


Hardware or software component	Affects what?	How?
Algorithm	Instruction count, CPI	The algorithm determines the number of source program instructions executed and hence the number of processor instructions executed. The algorithm may also affect the CPI, by favoring slower or faster instructions. For example, if the algorithm uses more divides, it will tend to have a higher CPI.
Programming language	Instruction count, CPI	The programming language certainly affects the instruction count, since statements in the language are translated to processor instructions, which determine instruction count. The language may also affect the CPI because of its features; for example, a language with heavy support for data abstraction (e.g., Java) will require indirect calls, which will use higher CPI instructions.
Compiler	Instruction count, CPI	The efficiency of the compiler affects both the instruction count and average cycles per instruction, since the compiler determines the translation of the source language instructions into computer instructions. The compiler's role can be very complex and affect the CPI in varied ways.
Instruction set architecture	Instruction count, clock rate, CPI	The instruction set architecture affects all three aspects of CPU performance, since it affects the instructions needed for a function, the cost in cycles of each instruction, and the overall clock rate of the processor.

Algorithm

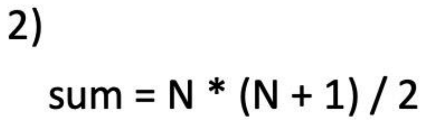
- **Algorithm**:
- Refers to the logical steps or method used to solve a problem or perform a task.

Sum of 1~N



1)

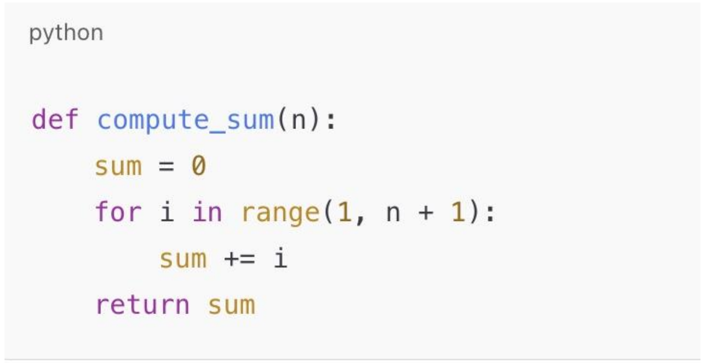
```
sum = 0
for i = 1 to N:
    sum += i
loop executes N iterations
```



2)

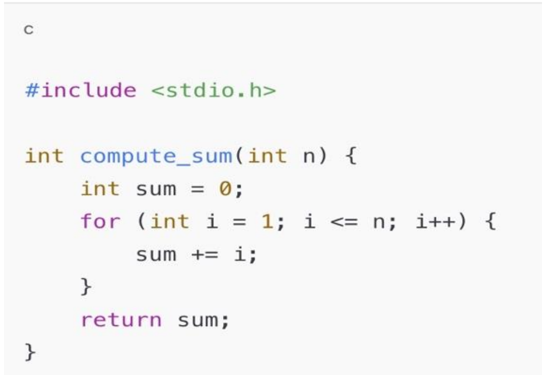
```
    sum = N * (N + 1) / 2
```

Programming Language



python

```python
def compute_sum(n):
    sum = 0
    for i in range(1, n + 1):
        sum += i
    return sum
```



c

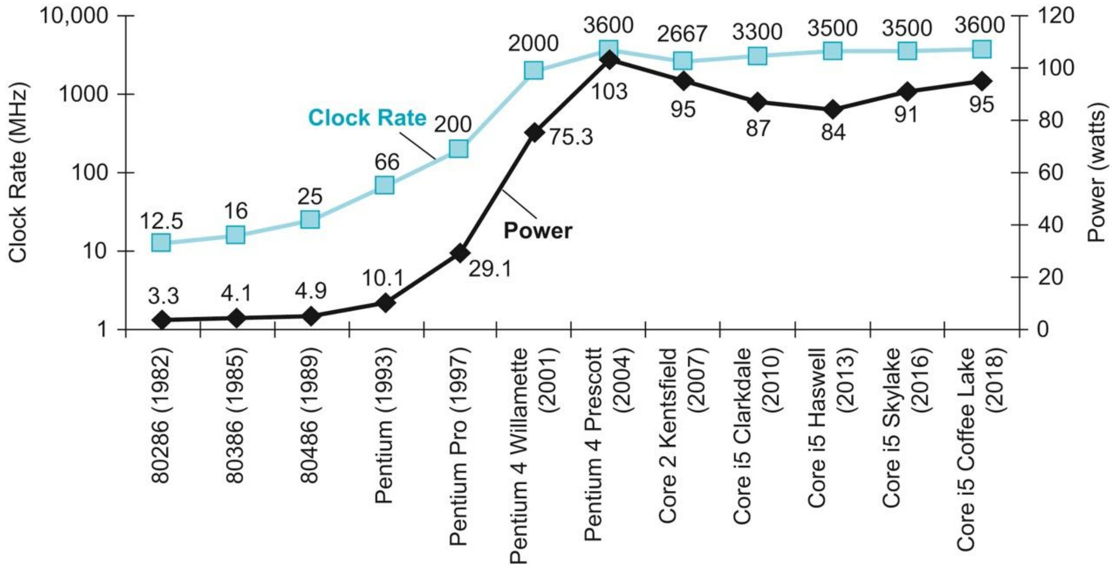
```c
#include <stdio.h>

int compute_sum(int n) {
    int sum = 0;
    for (int i = 1; i <= n; i++) {
        sum += i;
    }
    return sum;
}
```

- Python is an interpreted language, meaning each line of code is executed by the Python interpreter.
- The for loop involves: Interpreting each instruction (for, sum += i, etc.).
- Performing type checks (e.g., i is an integer, sum is updated correctly).
- Multiple function calls within the interpreter for managing the loop

C is a compiled language, so the code is directly translated into machine instructions.

The compiled code uses low-level instructions like ADD and JUMP for the loop, which execute directly on the hardware.

The Power Trends

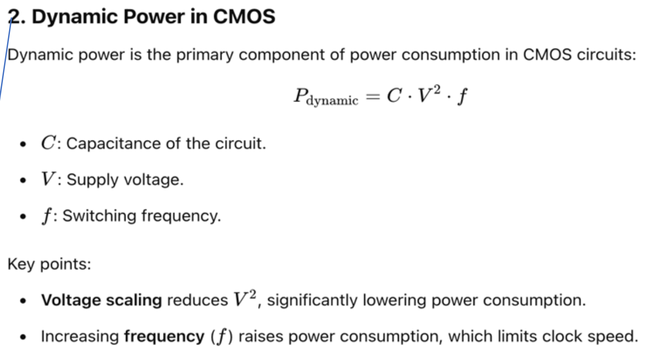


Why is recently slowing?
We run into the practical power limit for cooling commodity microprocessors

CMOS Technology

1. CMOS Technology Basics （Complementary Metal-Oxide-Semiconductor Integrated Circuit）

- **CMOS** is widely used in modern integrated circuits (ICs) because of its low power consumption and high noise immunity.
- **How it works**:
  - CMOS circuits use complementary pairs of **p-type** and **n-type** transistors.
  - When one transistor in the pair is on, the other is off, minimizing current flow and reducing static power consumption.



2. Dynamic Power in CMOS

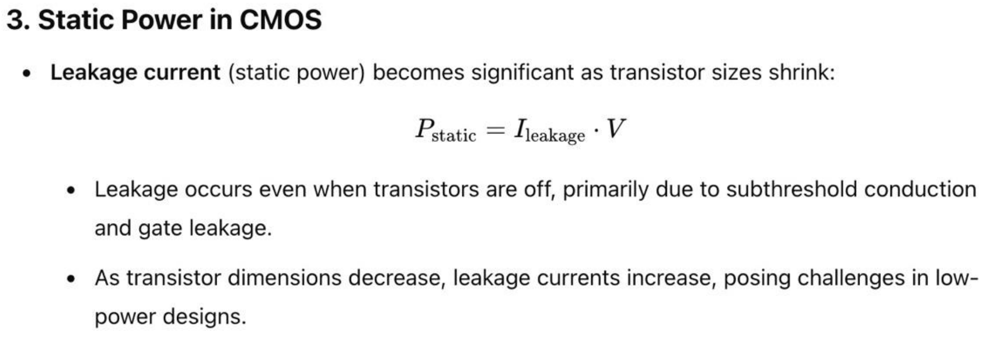
Dynamic power is the primary component of power consumption in CMOS circuits:

$$P_{\text{dynamic}} = C \cdot V^2 \cdot f$$

- C: Capacitance of the circuit.
- V: Supply voltage.
- f: Switching frequency.

Key points:

- **Voltage scaling** reduces V^2, significantly lowering power consumption.
- Increasing **frequency** (f) raises power consumption, which limits clock speed.



3. Static Power in CMOS

- **Leakage current** (static power) becomes significant as transistor sizes shrink:

$$P_{\text{static}} = I_{\text{leakage}} \cdot V$$

  - Leakage occurs even when transistors are off, primarily due to subthreshold conduction and gate leakage.
  - As transistor dimensions decrease, leakage currents increase, posing challenges in low-power designs.

Primary source of energy consumption, when transistors switch from state from 0 to 1 and vice versa!

Example

Suppose we developed a new, simpler processor that has 85% of the capacitive load of the more complex older processor. Further, assume that it can adjust voltage so that it can reduce voltage by 15% compared to processor B, which results in a 15% shrink in frequency. What is the impact on dynamic power?

$$\frac{\text{Power}_{\text{new}}}{\text{Power}_{\text{old}}} = \frac{\text{Capacitive load} \times 0.85 \ \times \ \text{Voltage} \times 0.85^{\,2} \times \ \text{Frequency switched} \times 0.85}{\text{Capacitive load} \times \text{Voltage}^2 \times \text{Frequency switched}}$$

Thus the power ratio is

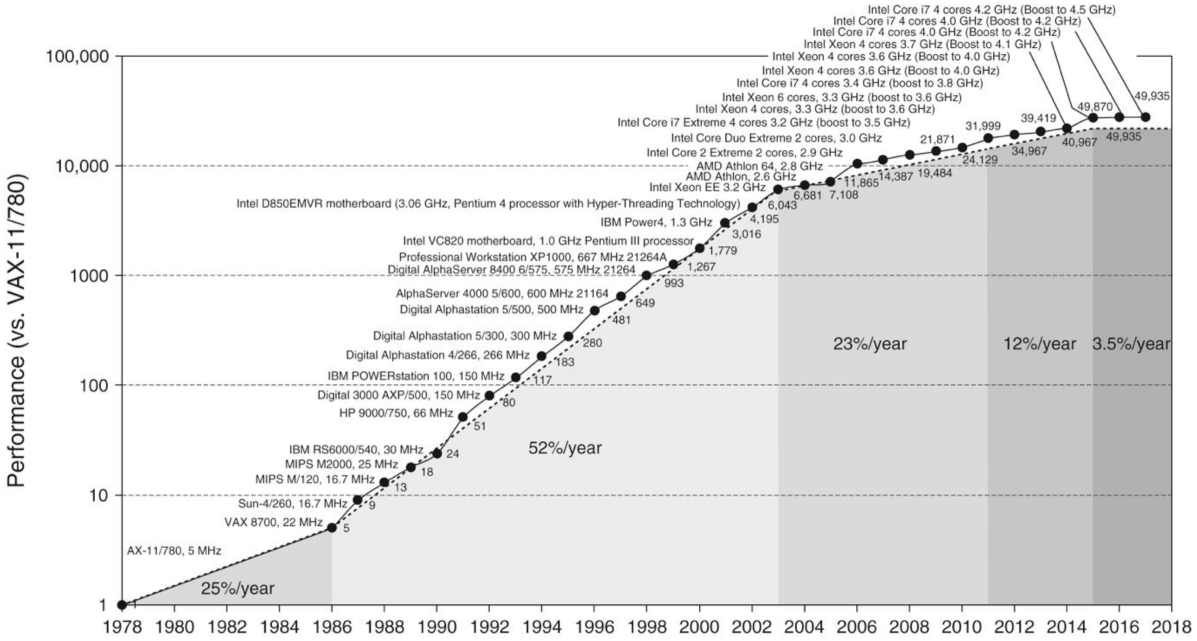
$$0.85^4 = 0.52$$

Hence, the new processor uses about half the power of the old processor.

Reducing Power

- The power wall
  - We can’t reduce voltage further
  - We can’t remove more heat
- How else can we improve performance?

Uniprocessor Performance



Multiprocessors

- Multicore microprocessors
  - More than one processor per chip
- Requires explicitly parallel programming
  - Compare with instruction level parallelism
    - Hardware executes multiple instructions at once
    - Hidden from the programmer
  - Hard to do
    - Programming for performance
    - Load balancing
    - Optimizing communication and synchronization

Attendance number

2