BRAC UNIVERSITY
Inspiring Excellence

CSE340: Computer Architecture

Chapter - 1: Computer Abstractions and Technology

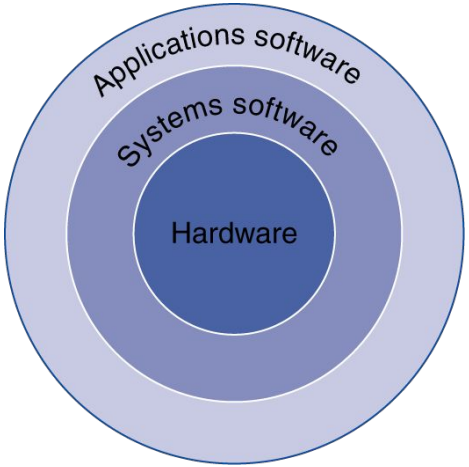
Understanding Performance

Hardware or Software Component	How this component affects performance?	Where to learn?
Algorithm	Determines both the number of **source-level statements** and the number of I/O operations executed.	Covered in CSE221
Programming Language, Compiler and architecture	Determines the number of computer instructions for each source-level statement.	Chapter 2 & 3
Processor and memory system	Determines how fast instructions can be executed	Chapter 4, 5 & 6
I/O system (Hardware and OS)	Determines how fast I/O operations may be executed	Chapter 4, 5 & 6

Seven Great Ideas in Computer Architecture

Ideas	Explanation
Use ***abstraction*** to simplify design	Divide systems into layers to manage **complexity** and hiding the lower level details.
Make the ***common case fast***	Optimizing frequently performed operations.
Performance *via* ***parallelism***	Running tasks **simultaneously** to boost speed.
Performance *via* ***pipelining***	A way to achieve parallelism. More will be discussed in Chapter - 4
Performance *via* ***prediction***	**Anticipate** future data or values to avoid delays.
Hierarchy of memories	**Organizing memory** into **levels** (registers, cache, RAM, storage) based on speed and size.
Dependability *via* redundancy	Adding extra components ensures reliability by compensating for failures in critical hardware or data.

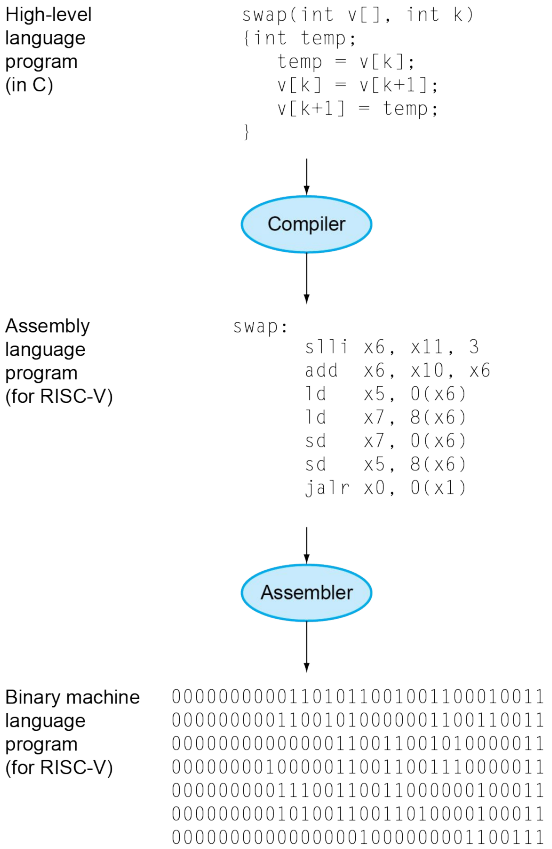
Below Your Program



- Application software
  - Written in high-level language
- System software
  - Compiler: translates HLL code to machine code
  - Operating System: service code
    - Handling input/output
    - Managing memory and storage
    - Scheduling tasks & sharing resources
- Hardware
  - Processor, memory, I/O controllers

Levels of Program Code

- High-level language
  - Level of abstraction closer to problem domain
  - Provides for productivity and portability
- Assembly language
  - Textual representation of instructions
- Hardware representation
  - Binary digits (bits)
  - Encoded instructions and data



Components of a Computer

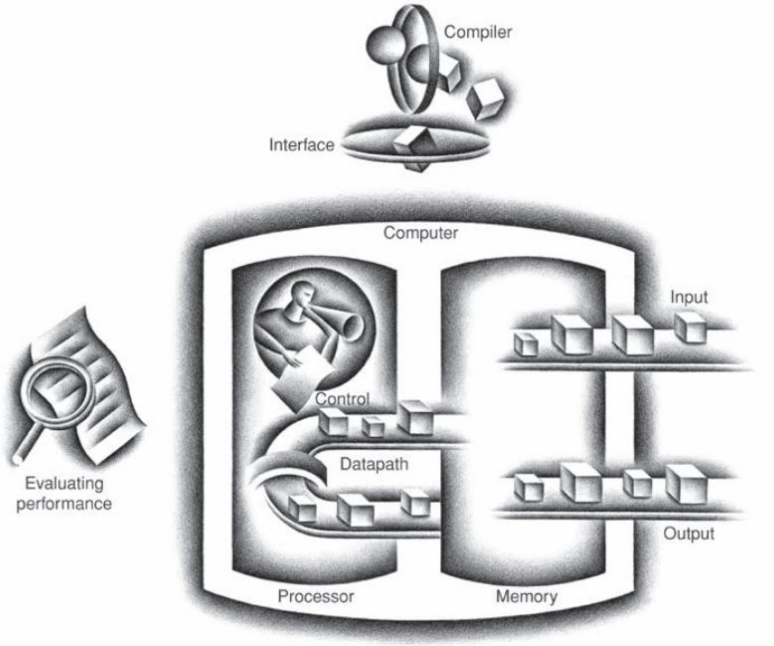


Figure: The standard organization of a computer

The five classic components of a computer are:

- Input
- Output
- Memory
- Datapath
- Control

- Datapath and control are sometimes combined and called the processor.

Opening the Box

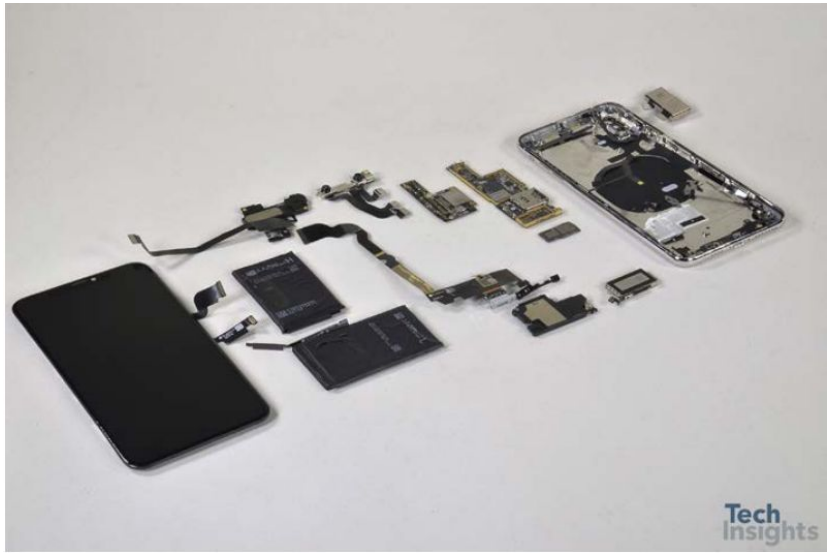


Figure: Components of the Apple iPhone XS Max

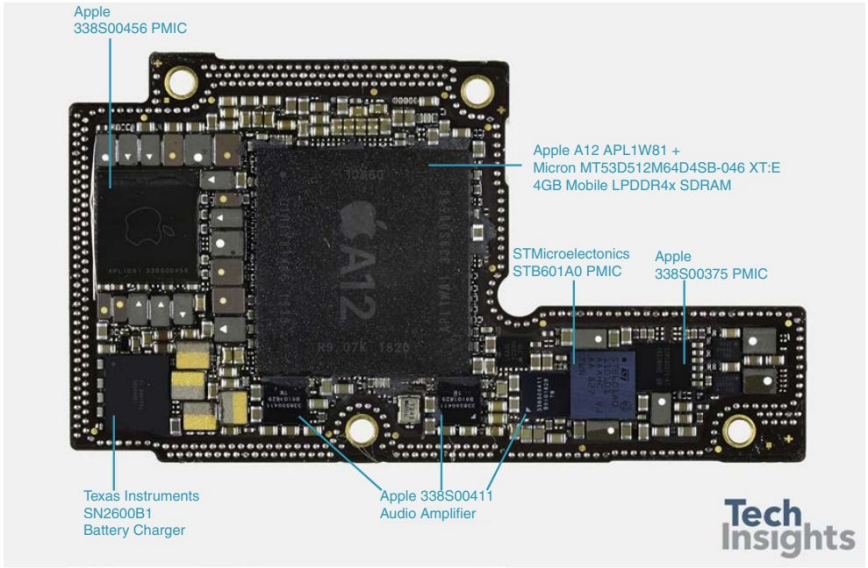


Figure: Logic board of the Apple iPhone XS Max

Opening the Box

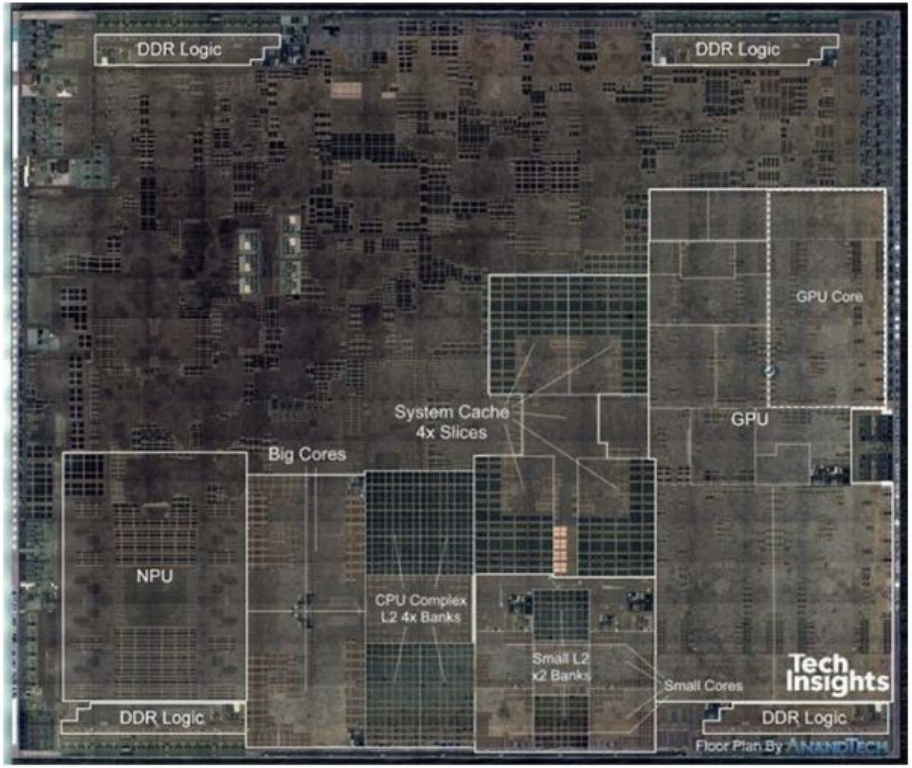


Figure: The processor integrated circuit inside the A12 package

Technologies for Building Processors

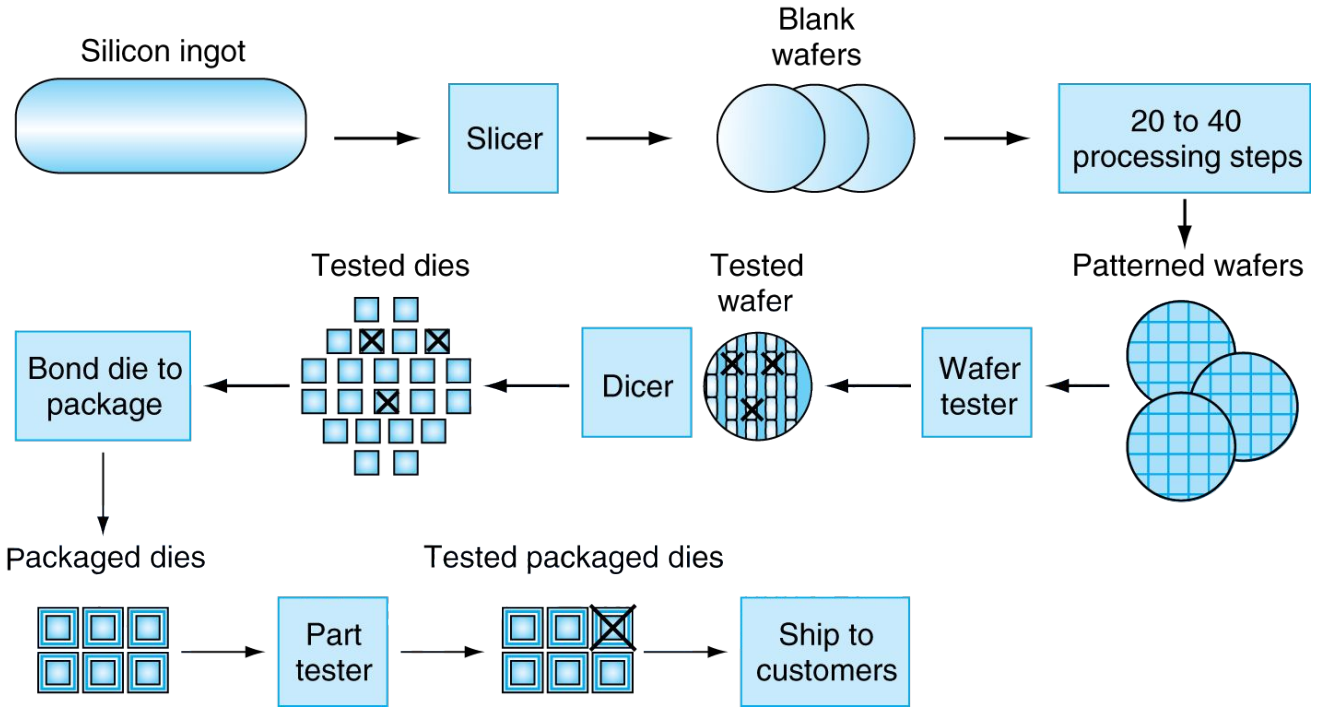


Figure: Manufacturing process of an IC

Intel 10th Gen. Wafer

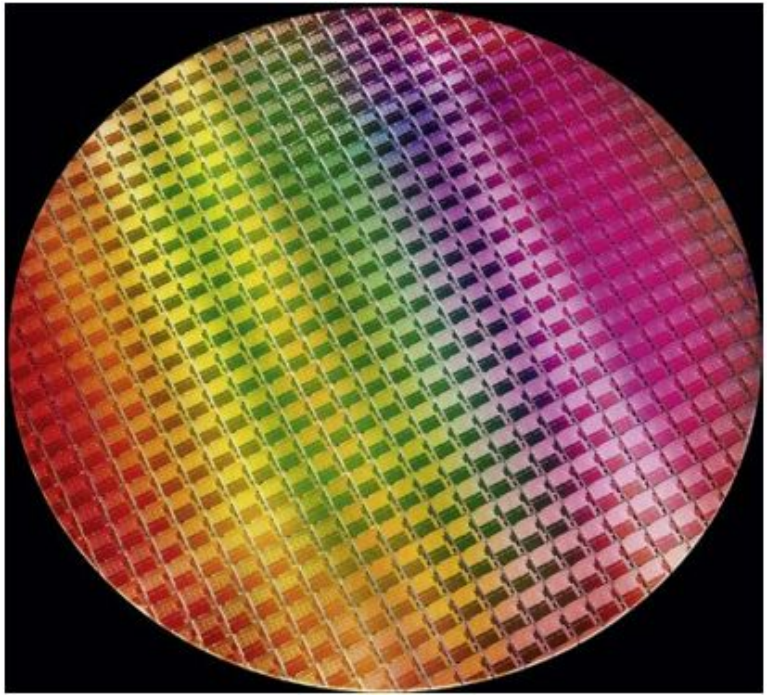


Figure: A 12-inch (300mm) wafer

This 10nm wafer contains 10th Gen Intel® Core™ processors, code-named "**Ice Lake**" (Courtesy Intel).

The number of dies on this 300 mm (12 inch) wafer at 100% yield is 506.

Each die or chip size = 11.4mm by 10.7mm

Yield: The percentage of good dies from the total number of dies on the wafer.

Die: The individual rectangular sections that are cut from a wafer, more informally known as **chips**.

IC Cost Calculation

$$\text{Cost per die} = \frac{\text{Cost per wafer}}{\text{Dies per wafer} \times \text{yield}}$$

$$\text{Dies per wafer} \approx \frac{\text{Wafer area}}{\text{Die area}}$$

$$Yield = \frac{1}{(1 + (Defects\ per\ area \times \frac{Die\ area}{2}))^{N}}$$

- Nonlinear relation to area and defect rate
  - Wafer cost and area are fixed
  - Defect rate determined by manufacturing process
  - Die area determined by architecture and circuit design

Note:
N in the yield formula is a model parameter that determines how defects and die size impact yield. It is typically derived from real-world data and can range between 1 and 2, though it may vary outside this range depending on the technology and defect behavior.

Defining Performance

Based on the table below which airplane has the best performance?

Airplane	Passenger capacity	Cruising range (miles)	Cruising speed (m.p.h.)	Passenger throughput (passengers × m.p.h.)
Boeing 737	240	3000	564	135,360
BAC/Sud Concorde	132	4000	1350	178,200
Boeing 777-200LR	301	9395	554	166,761
Airbus A380-800	853	8477	587	500,711

Figure: The capacity, range, and speed for a number of commercial airplanes

Response Time and Throughput

Response time: The time between the start and completion of a task. Also known as **execution time**.

Throughput: Total work done per unit time.
e.g., tasks/transactions/… per hour

We'll focus on response time for now…

Answer yourself:

How are response time and throughput affected by,

1. Replacing the processor with a faster version?
2. Adding additional processors to a system that uses multiple processors for separate tasks?

Relative Performance

To maximize performance, we want to minimize the **response/execution** time for a task.
The relation between performance and execution time for computer X:

$$\text{Performance}_X = \frac{1}{\text{Execution time}_X}$$

"X is *n* time faster than Y" $\text{Performance}_X / \text{Performance}_Y = \text{Execution time}_Y / \text{Execution time}_X = n$

- Example: time taken to run a program 10s on A, 15s on B
  - Execution Time$_B$ / Execution Time$_A$
    = 15s / 10s = 1.5
  - So A is 1.5 times faster than B

Measuring Execution Time

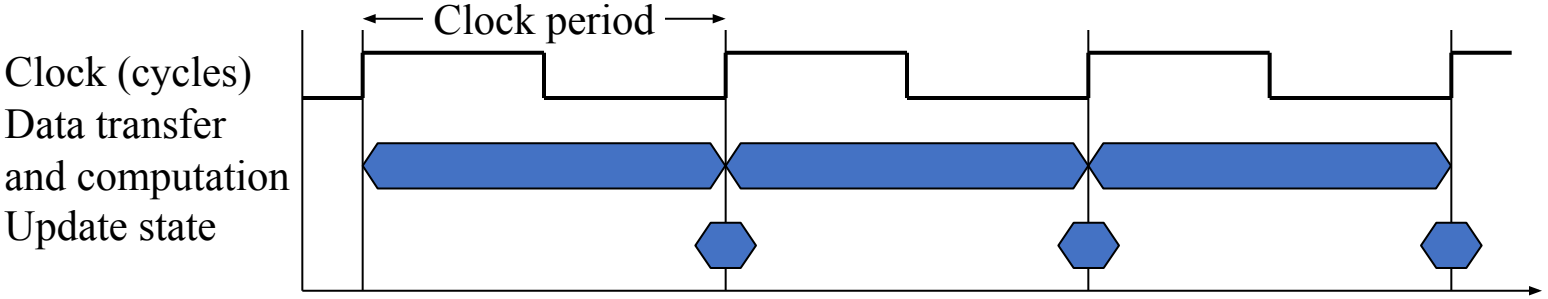
Elapsed time: Total response time, including all aspects. E.g Processing, I/O, OS overhead, idle time etc.
Determines system performance

CPU time: The actual time the CPU spends computing for a specific task.
Discounts I/O time, other jobs' shares
Comprises **user CPU time** and **system CPU time**

Different programs are affected differently by CPU and system performance

CPU Clocking

Operation of digital hardware governed by a constant-rate clock.



- **Clock period**: Time taken to complete a clock cycle
  - e.g., 250ps = 0.25ns = 250×10^{-12}s
- **Clock frequency (rate)**: Clock cycles completed per second
  - e.g., 4.0GHz = 4000MHz = 4.0×10^{9}Hz

CPU Performance and Its Factors

$$\text{CPU execution time for a program} = \text{CPU clock cycles for a program} \times \text{Clock cycle time}$$

Alternatively, because clock rate and clock cycle time are inverses,

$$\text{CPU execution time for a program} = \frac{\text{CPU clock cycles for a program}}{\text{Clock rate}}$$

Answer yourself:

Analyze the formula mentioned above and determine how can you improve the performance of a program?

CPU Performance Example

Problem statement:

A program runs in 10 seconds on computer A, which has a 2GHz clock. We are trying to help a computer designer build a computer, B, which will run this program in 6 seconds. The designer has determined that a substantial increase in the clock rate is possible, but this increase will affect the rest of the CPU design, causing computer B to require 1.2 times as many clock cycles as computer A for this program. What clock rate should we tell the designer to target?

Solution:

Let's first find the number of clock cycles required for the program on A:

$$\text{CPU time}_A = \frac{\text{CPU clock cycles}_A}{\text{Clock rate}_A}$$

$$10\,\text{seconds} = \frac{\text{CPU clock cycles}_A}{2 \times 10^9 \frac{\text{cycles}}{\text{second}}}$$

$$\text{CPU clock cycles}_A = 10\,\text{seconds} \times 2 \times 10^9 \frac{\text{cycles}}{\text{second}} = 20 \times 10^9 \text{ cycles}$$

CPU time for B can be found using this equation:

$$\text{CPU time}_B = \frac{1.2 \times \text{CPU clock cycles}_A}{\text{Clock rate}_B}$$

$$6 \text{ seconds} = \frac{1.2 \times 20 \times 10^9 \text{ cycles}}{\text{Clock rate}_B}$$

$$\text{Clock rate}_B = \frac{1.2 \times 20 \times 10^9 \text{ cycles}}{6 \text{ seconds}} = \frac{0.2 \times 20 \times 10^9 \text{ cycles}}{\text{second}} = \frac{4 \times 10^9 \text{ cycles}}{\text{second}} = 4 \text{ GHz}$$

To run the program in 6 seconds, B must have twice the clock rate of A.

Instruction Count and CPI

$$\text{CPU clock cycles} = \text{Instructions for a program} \times \frac{\text{Average clock cycles}}{\text{per instruction}}$$

$$\text{CPU time} = \text{Instruction count} \times \text{CPI} \times \text{Clock cycle time}$$

$$\text{CPU time} = \frac{\text{Instruction count} \times \text{CPI}}{\text{Clock rate}}$$

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

Instruction Count & CPI Example

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

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

So, A is 1.2 times faster

Answer yourself:

Why we assumed that the instruction count is same in both cases?

Average CPI with Example

- If different instruction classes take different numbers of cycles

$$\text{Clock Cycles} = \sum_{i=1}^{n} (\text{CPI}_i \times \text{Instruction Count}_i)$$

- Weighted average CPI

$$\text{CPI} = \frac{\text{Clock Cycles}}{\text{Instruction Count}} = \sum_{i=1}^{n} \left(\text{CPI}_i \times \frac{\text{Instruction Count}_i}{\text{Instruction Count}} \right)$$

Problem Statement:

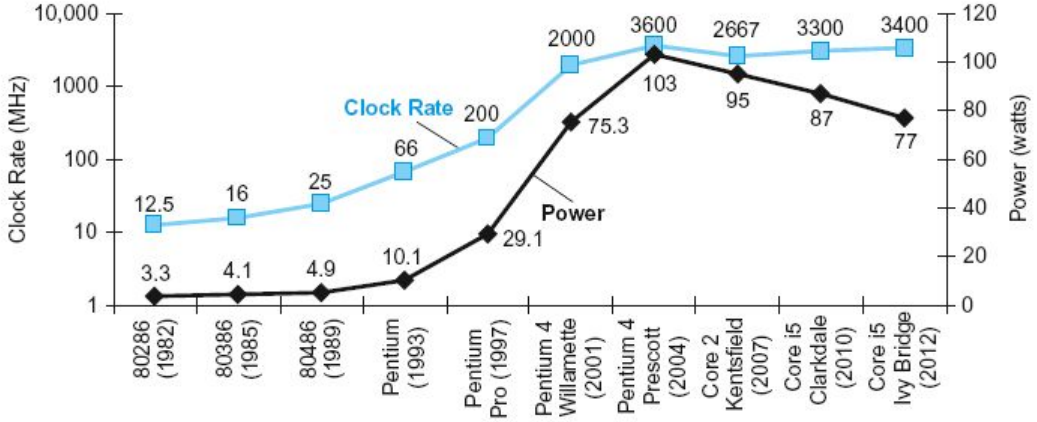
Alternative compiled code sequences using instructions in classes A, B, C

Solution:

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

Power Trends



- In CMOS IC technology

$$\text{Power} = \text{Capacitive load} \times \text{Voltage}^2 \times \text{Frequency}$$

Reducing Power

- Suppose a new CPU has
  - 85% of capacitive load of old CPU
  - 15% voltage and 15% frequency reduction

$$\frac{P_{new}}{P_{old}} = \frac{C_{old} \times 0.85 \times (V_{old} \times 0.85)^2 \times F_{old} \times 0.85}{C_{old} \times V_{old}^{\ 2} \times F_{old}} = 0.85^4 = 0.52$$

- The power wall
  - We can't reduce voltage further
  - We can't remove more heat

Answer yourself:

How else can we improve performance?

Uniprocessor Performance

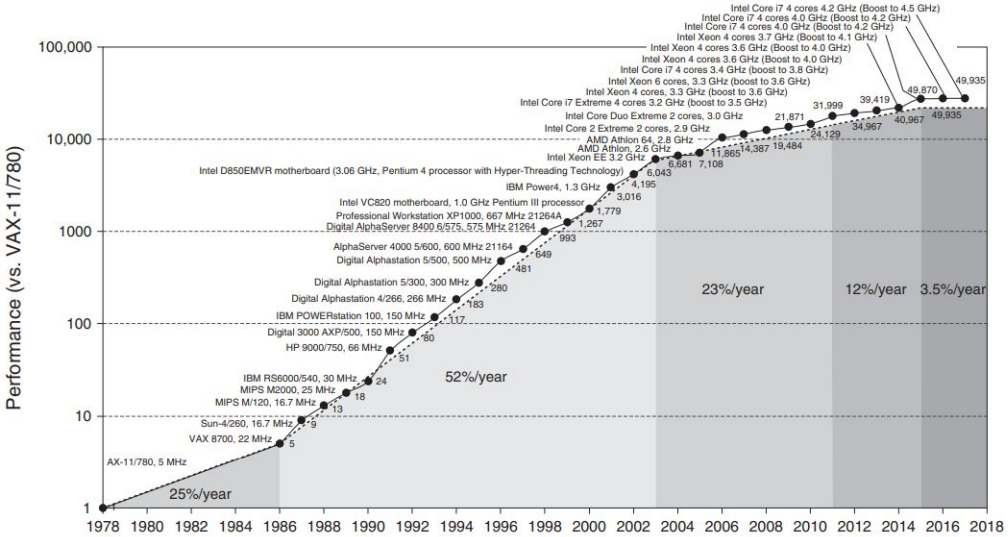


Figure: Growth in processor performance since the mid-1980s

Constrained by power, instruction-level parallelism, memory latency

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

SPEC CPU Benchmark

- What is Benchmark?
  Programs used to measure performance
- Standard Performance Evaluation Corp (**SPEC**)
  - Develops benchmarks for CPU, I/O, Web, …
- SPEC CPU2017
  - Elapsed time to execute a selection of programs
    - Negligible I/O, so focuses on CPU performance
  - Normalize relative to reference machine
  - Summarize as geometric mean of performance ratios
    - SPECspeed2017 (integer) and CFP2006 (floating-point)

$$\sqrt[n]{\prod_{i=1}^{n} \text{Execution time ratio}_i}$$

SPEC Ratio & Geometric Mean Calculation

Description	Name	Instruction Count x 10^9	CPI	Clock cycle time (seconds x 10^{-9})	Execution Time (seconds)	Reference Time (seconds)	SPECratio
Perl interpreter	perlbench	2684	0.42	0.556	627	1774	2.83
GNU C compiler	gcc	2322	0.67	0.556	863	3976	4.61
Route planning	mcf	1786	1.22	0.556	1215	4721	3.89
Discrete Event simulation - computer network	omnetpp	1107	0.82	0.556	507	1630	3.21
XML to HTML conversion via XSLT	xalancbmk	1314	0.75	0.556	549	1417	2.58
Video compression	x264	4488	0.32	0.556	813	1763	2.17
Artificial Intelligence: alpha-beta tree search (Chess)	deepsjeng	2216	0.57	0.556	698	1432	2.05
Artificial Intelligence: Monte Carlo tree search (Go)	leela	2236	0.79	0.556	987	1703	1.73
Artificial Intelligence: recursive solution generator (Sudoku)	exchange2	6683	0.46	0.556	1718	2939	1.71
General data compression	xz	8533	1.32	0.556	6290	6182	0.98
Geometric mean	–	–	–	–	–	–	2.36

Note:

Reference time is supplied by SPEC

Amdahl's Law

- Improving an aspect of a computer and expecting a proportional improvement in overall performance

$$T_{improved} = \frac{T_{affected}}{improvement\ factor} + T_{unaffected}$$

- Example: multiply accounts for 80s/100s
  - How much improvement in multiply performance to get 5× overall?

$$20 = \frac{80}{n} + 20$$

  - Can't be done!

- Corollary: make the common case fast

MIPS as a Performance Metric

- MIPS: Millions of Instructions Per Second
  - Doesn't account for
    - Differences in ISAs between computers
    - Differences in complexity between instructions

$$\text{MIPS} = \frac{\text{Instruction count}}{\text{Execution time} \times 10^6}$$

$$= \frac{\text{Instruction count}}{\frac{\text{Instruction count} \times \text{CPI}}{\text{Clock rate}} \times 10^6} = \frac{\text{Clock rate}}{\text{CPI} \times 10^6}$$

  - CPI varies between programs on a given CPU

Concluding Remarks

- Cost/performance is improving
  - Due to underlying technology development
- Hierarchical layers of abstraction
  - In both hardware and software
- Instruction set architecture
  - The hardware/software interface
- Execution time: the best performance measure
- Power is a limiting factor
  - Use parallelism to improve performance