Computer Abstractions and Technology

Lecture 1
September 6th, 2023

Jae W. Lee (jaewlee@snu.ac.kr)
Computer Science and Engineering
Seoul National University

***Slide credits**: [CS:APP3e] slides from CMU; [COD:RV2e] slides from Elsevier Inc.*

Outline

Textbook: [COD] 1.1-1.8, 1.10

- **Introduction**
- **Eight Great Ideas in Computer Architecture**
- **Below Your Program**
- **Under the Cover**
- **Technologies for Building Processors and Memory**
- **Performance**
- **Power Wall**
- **Switch from Uniprocessors to Multiprocessors**
- **Amdahl's Law**

The Computer Revolution

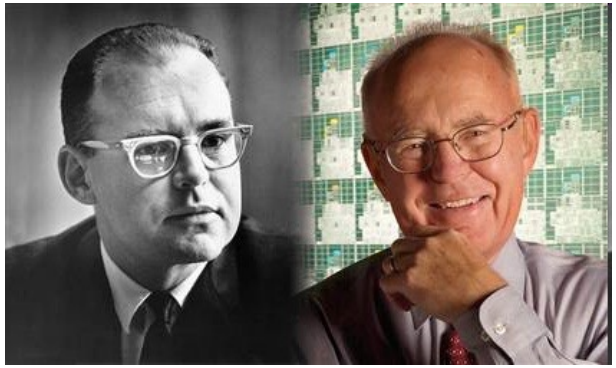
- **Progress in computer technology**
  - Underpinned by Moore's Law
- **Makes novel applications feasible**
  - Computers in automobiles
  - Cell phones
  - Human genome project
  - World Wide Web
  - Search Engines
- **Computers are pervasive**

Overview: Moore's Law (1965)

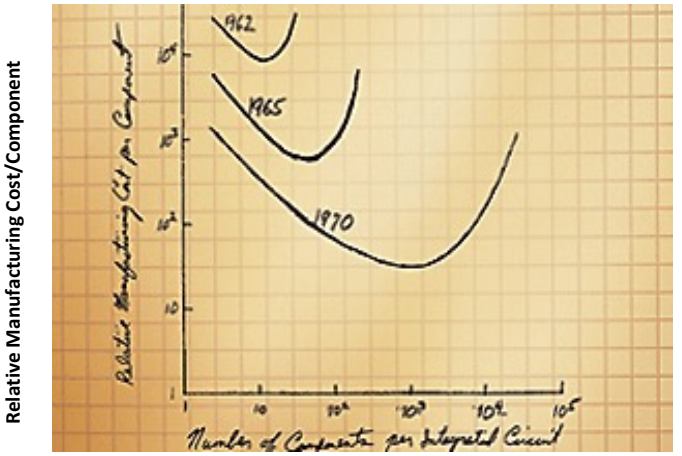
- **The single most important guideline in microprocessor fabrication and architecture**

  "the number of transistors per chip will double every 18 months"

- **Main driving force for the phenomenal growth of IT industry**



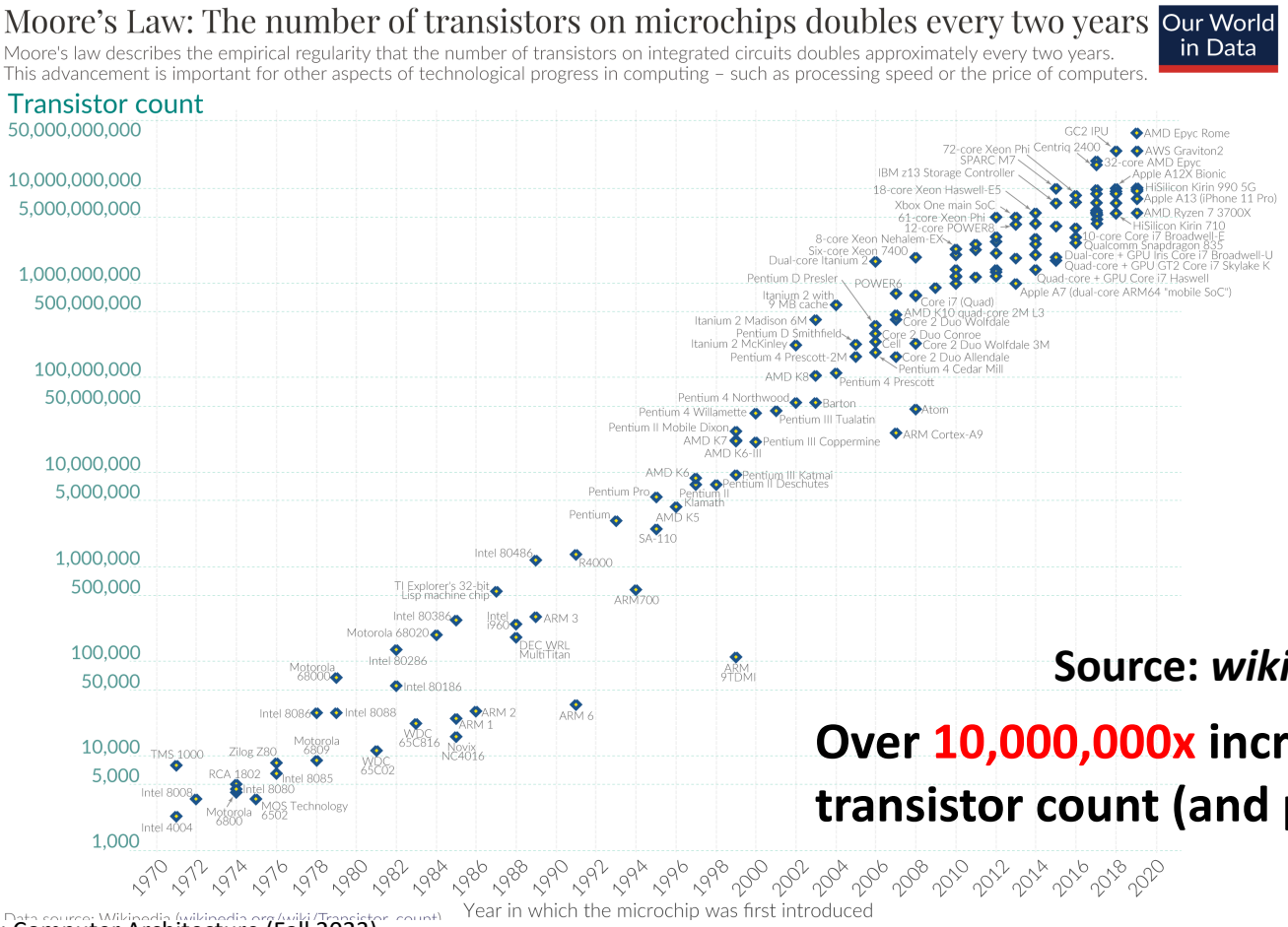
(http://download.intel.com/museum/research/arc_collect/history_docs/pix/hoff1.jpg)



Gordon Moore Original graph from 1965 *(source: www.intel.com)*

Overview: Moore's Law (1965)

- **Glorious 50 years of Moore's Law**
  - Recently being slowed down



Source: *wikipedia.org*

Over 10,000,000x increase of transistor count (and performance)!

Classes of Computers (1)

- **Personal computers**
  - General purpose, variety of software
  - Subject to cost/performance tradeoff
- **Server computers**
  - Network based
  - High capacity, performance, reliability
  - Range from small servers to building sized

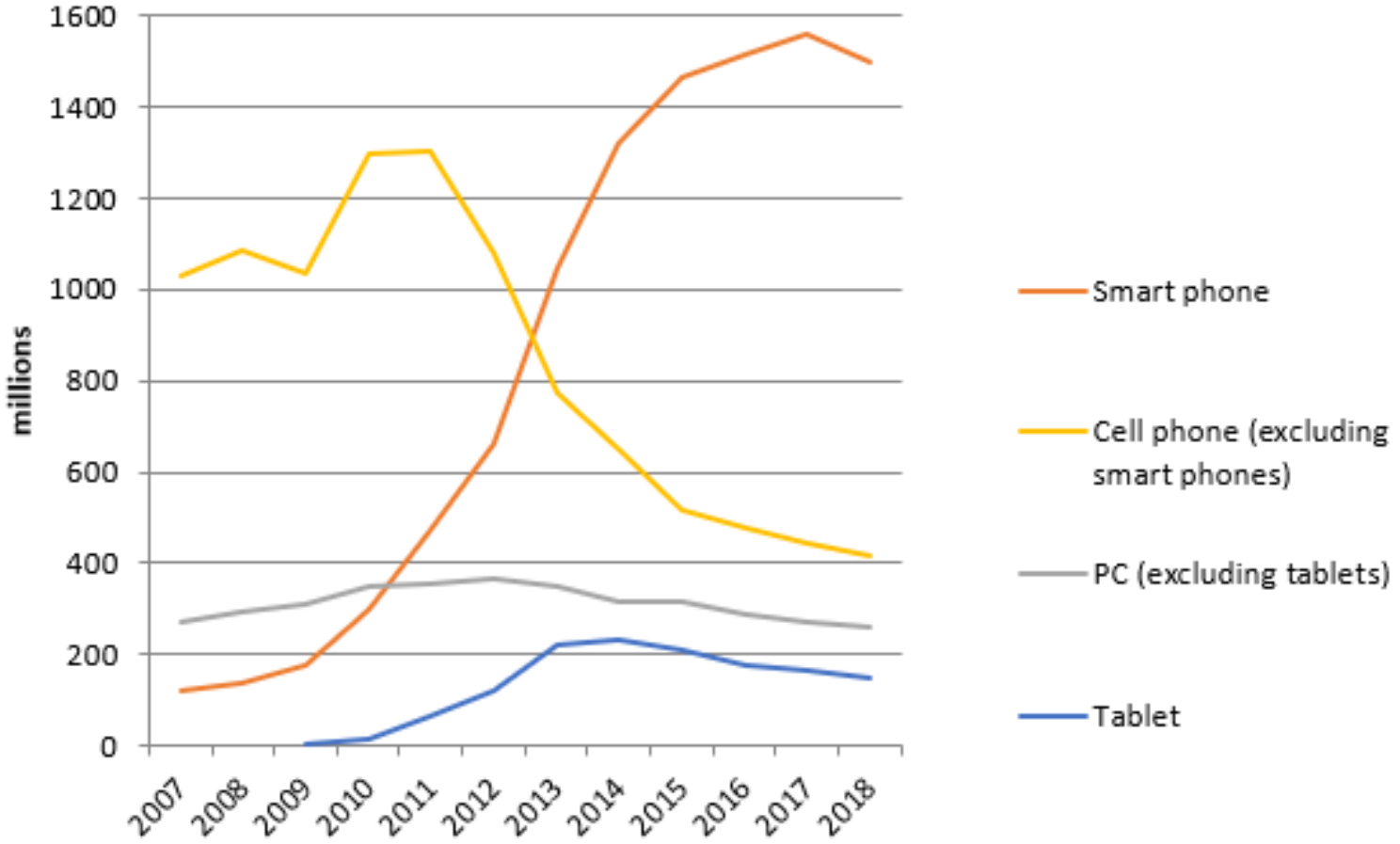
Classes of Computers (2)

- **Supercomputers**
  - High-end scientific and engineering calculations
  - Highest capability but represent a small fraction of the overall computer market

- **Embedded computers**
  - Hidden as components of systems
  - Stringent power/performance/cost constraints

Classes of Computers (3)

- **Post-PC era**

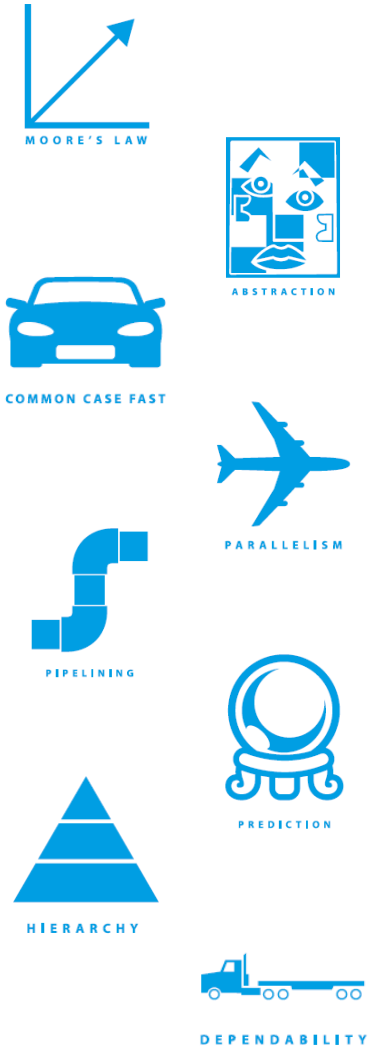


Classes of Computers (4): The PostPC Era

- **Personal Mobile Device (PMD)**
  - Battery operated
  - Connects to the Internet
  - Hundreds of dollars
  - Smart phones, tablets, electronic glasses
- **Cloud computing**
  - Warehouse Scale Computers (WSC)
  - Software as a Service (SaaS)
  - Portion of software run on a PMD and a portion run in the Cloud
  - Amazon, Google, Microsoft, Naver, Kakao, KT, etc.

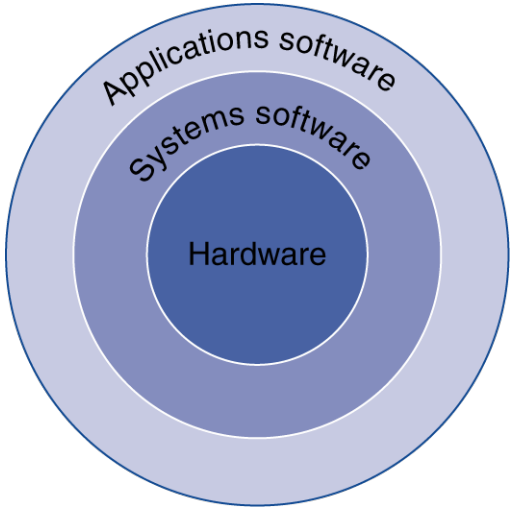
Eight Great Ideas

- Design for *Moore's Law*
- Use *abstraction* to simplify design
- Make the *common case fast*
- Performance *via parallelism*
- Performance *via pipelining*
- Performance *via prediction*
- *Hierarchy* of memories
- *Dependability via* redundancy



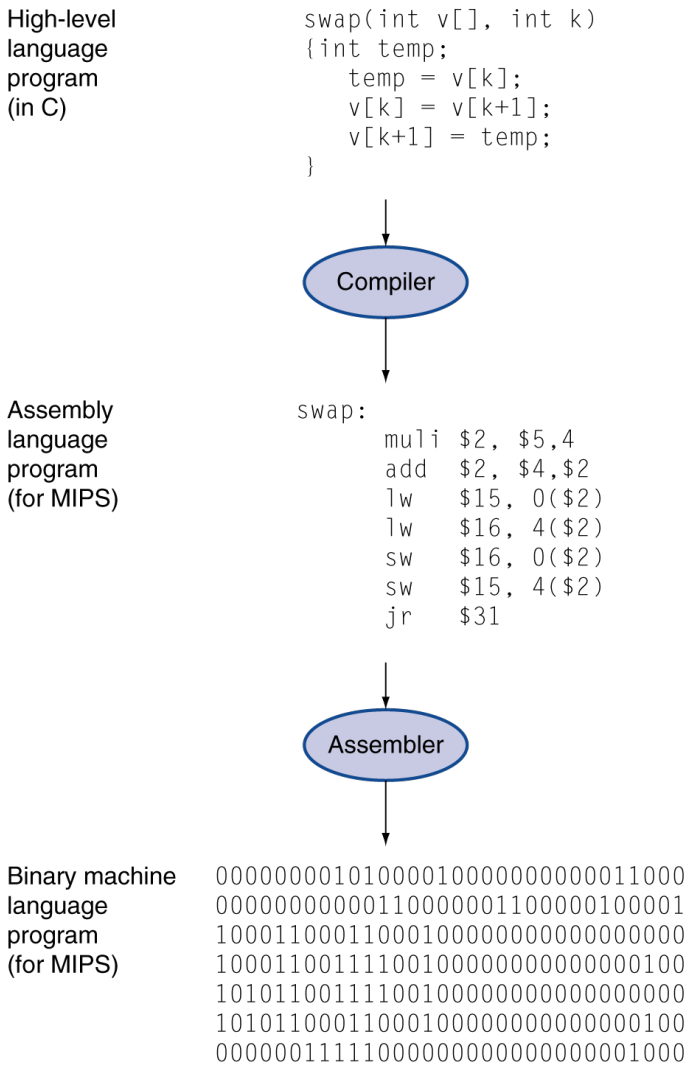
Below your program

- **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



Understanding Performance

- **Algorithm**
  - Determines number of operations executed
- **Programming language, compiler, architecture**
  - Determine number of machine instructions executed per operation
- **Processor and memory system**
  - Determine how fast instructions are executed
- **I/O system (including OS)**
  - Determines how fast I/O operations are executed

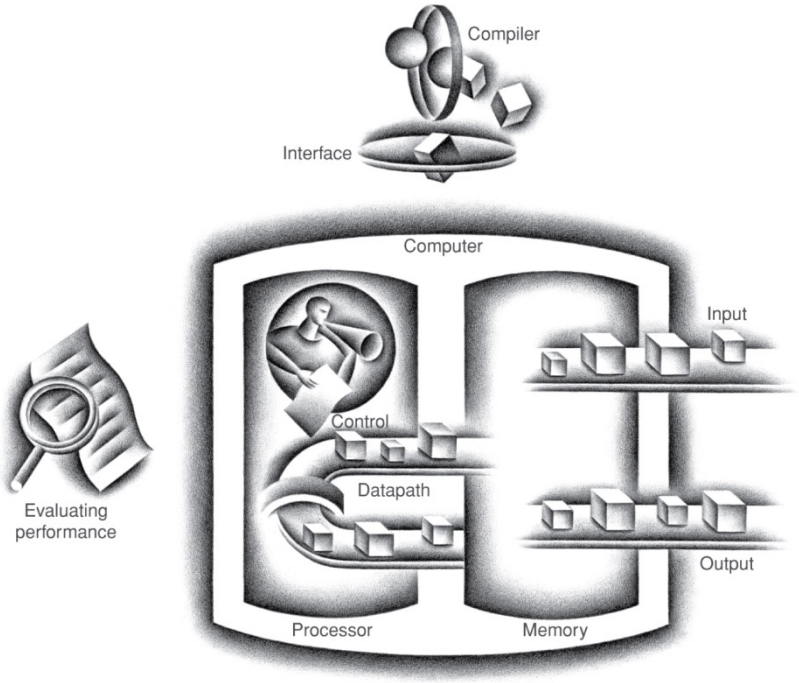
Outline

Textbook: [COD] 1.1-1.8, 1.10

- Introduction
- Eight Great Ideas in Computer Architecture
- Below Your Program
- **Under the Cover**
- Technologies for Building Processors and Memory
- Performance
- Power Wall
- Switch from Uniprocessors to Multiprocessors
- Amdahl's Law

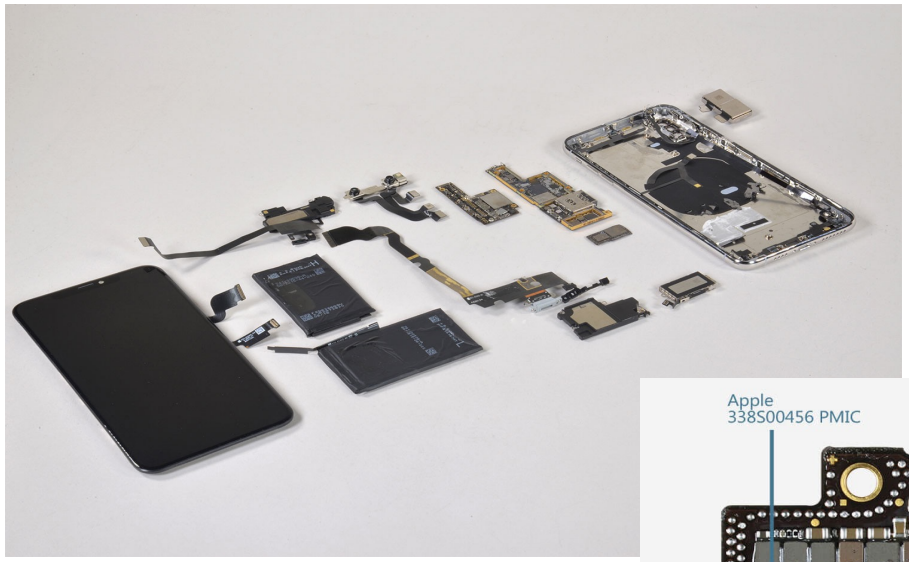
Under the Cover: Components of a computer

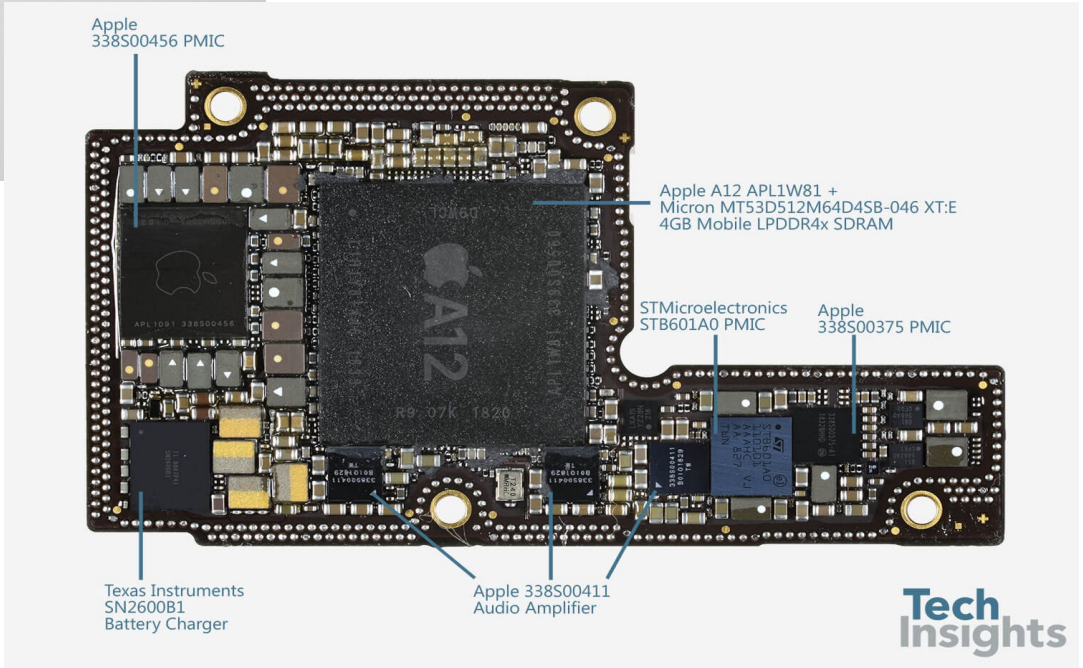
The BIG Picture



- **Common for all kinds of computer**
  - Desktop, server, embedded
- **Processor (Datapath/Control), Memory, and I/O**
- **Input/output (I/O) includes**
  - User-interface devices
    - Display, keyboard, mouse
  - Storage devices
    - Hard disk, CD/DVD, flash
  - Network adapters
    - For communicating with other computers

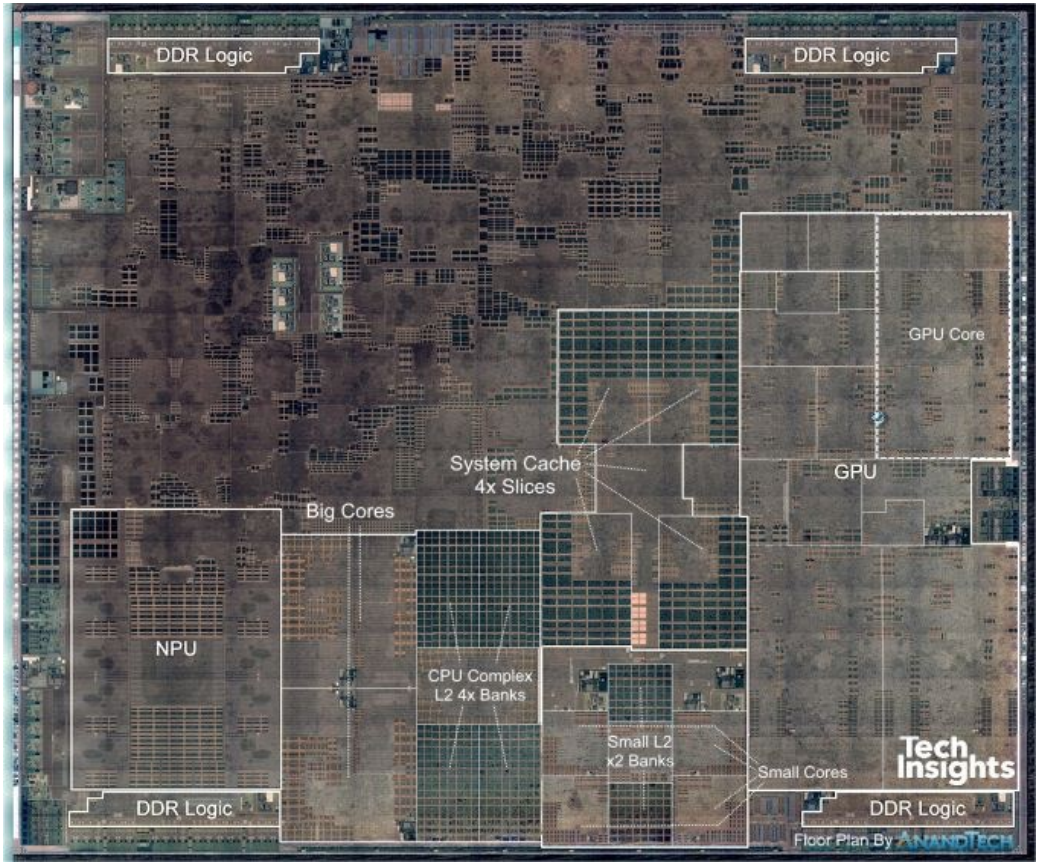
Under the Cover: Opening the box





Under the Cover: Inside the processor (CPU)

- **Apple A12**



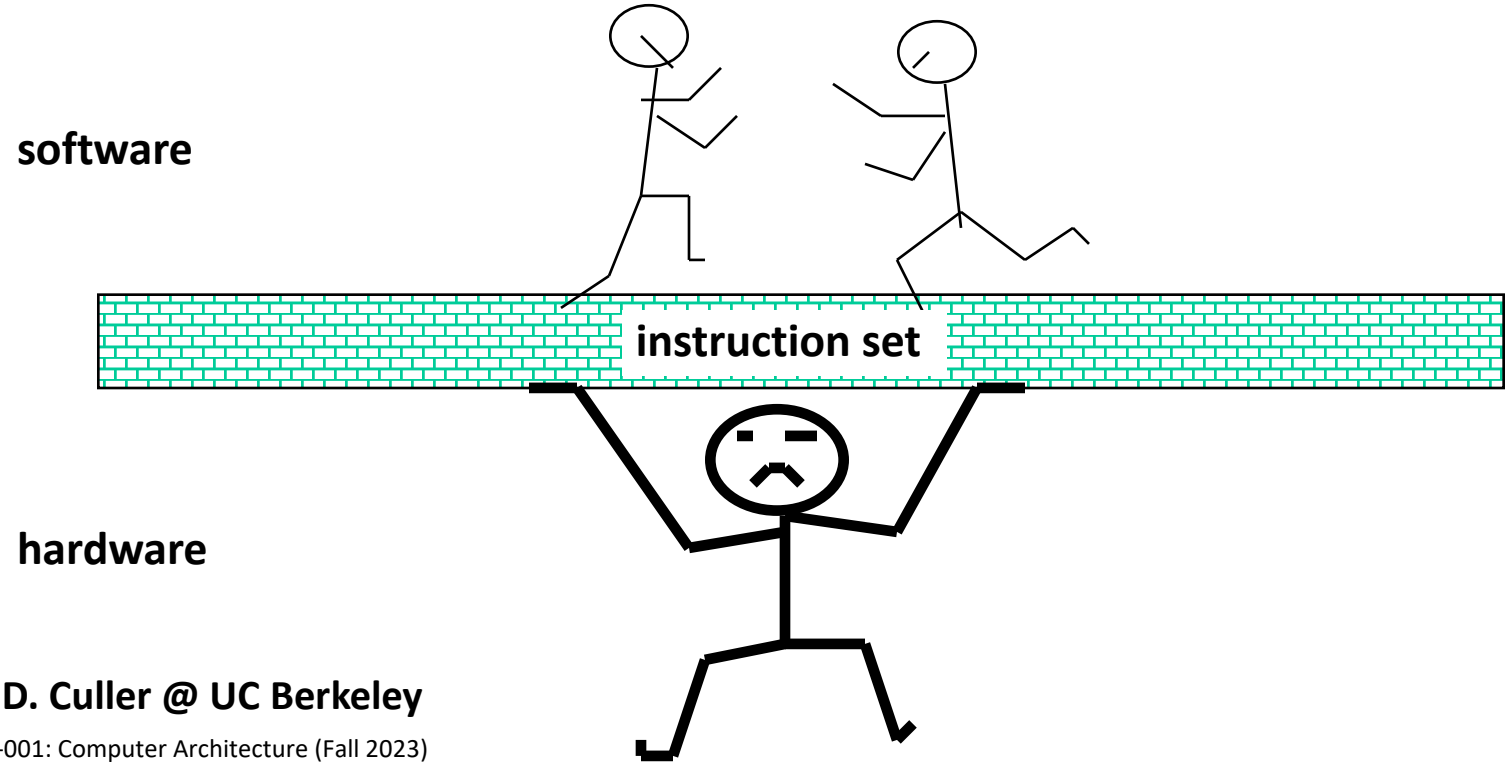
You will understand what each block does by the end of this course!

Under the Cover: Inside the Processor (CPU)

- **Datapath: performs operations on data**
- **Control: sequences datapath, memory, ...**
- **Cache memory**
  - Small fast SRAM memory for immediate access to data

Under the Cover: Abstractions

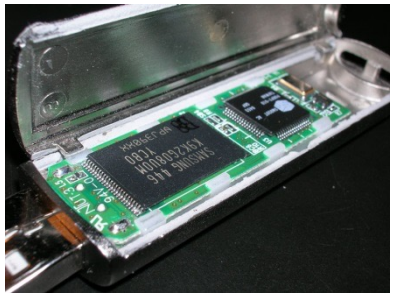
- **Abstraction helps us deal with complexity**
  - Hide lower-level detail
- **Instruction set architecture (ISA)**
  - Hardware abstraction visible to software (compiler or programmer)
  - The hardware/software interface



Source: D. Culler @ UC Berkeley

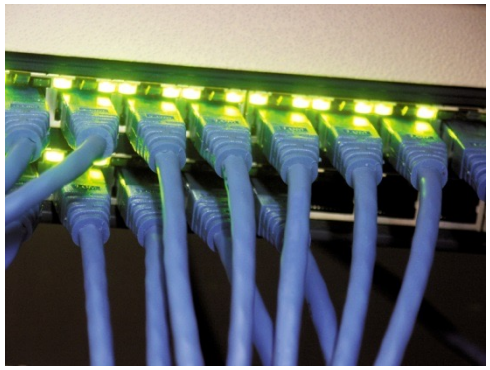
Under the Cover: Safe place for data

- **Volatile main memory**
  - Loses instructions and data when power off
  - E.g., DRAM
- **Non-volatile secondary memory**
  - Magnetic disk
  - Flash memory
  - Optical disk (CDROM, DVD)



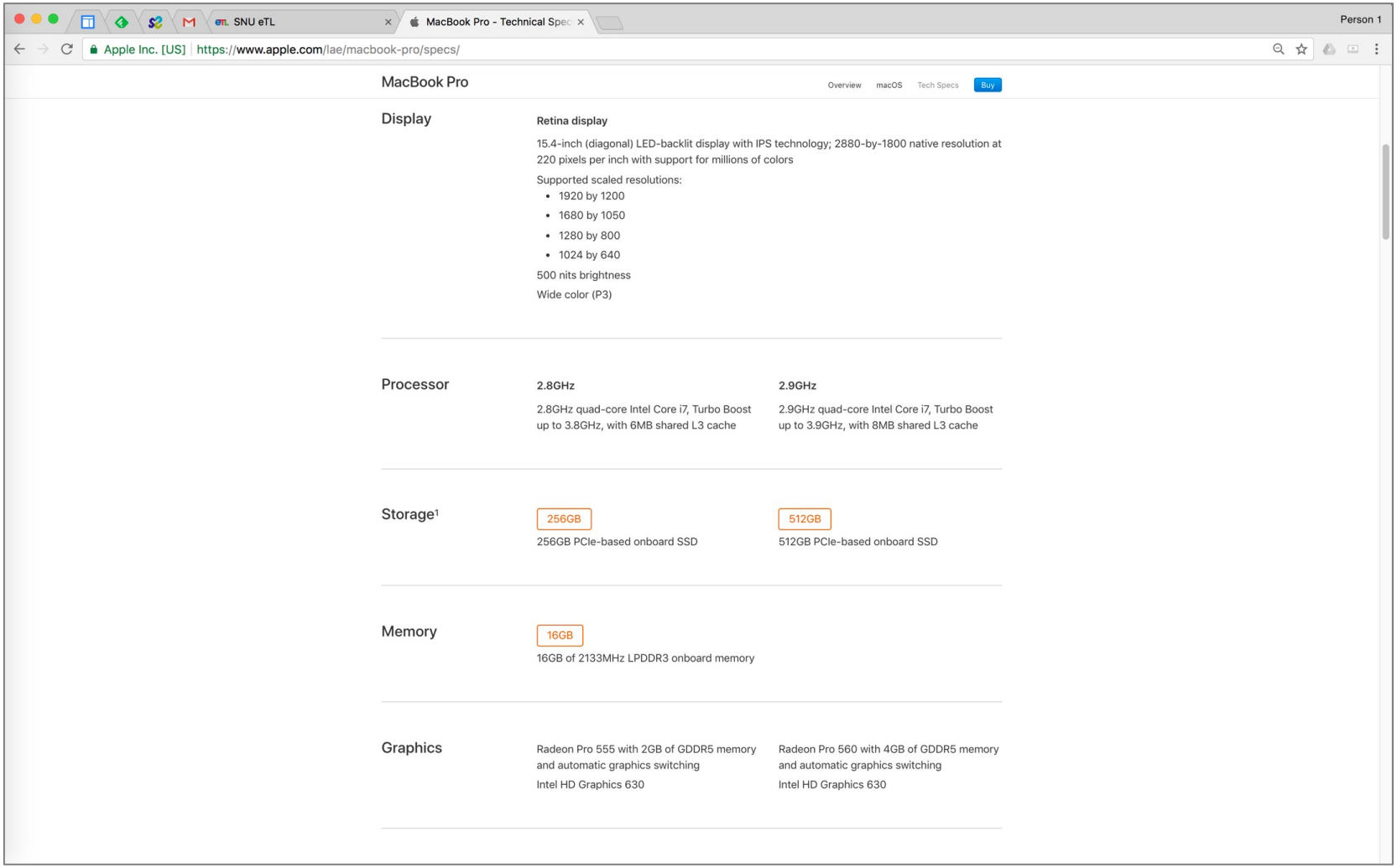
Under the Cover: Networks

- **Communication and resource sharing**
- **Local area network (LAN): Ethernet**
  - Within a building
- **Wide area network (WAN): the Internet**
- **Wireless network: WiFi, Bluetooth**



Under the Cover

- **Example: Tech specs of Macbook Pro (2017)**



Under the Cover: Technology trends

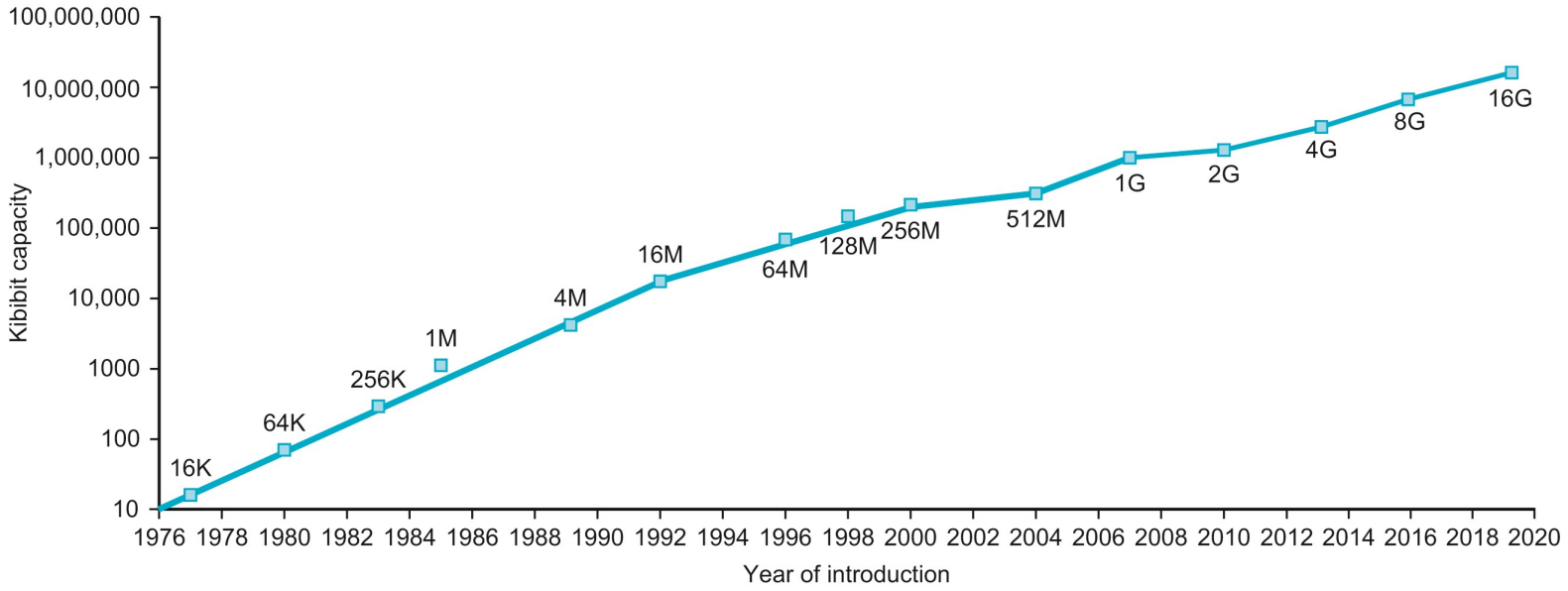
- **Processor**
  - Logic capacity: about 30%/year
  - Clock rate: about 20%/year (and slowing down)

- **Disk**
  - Capacity: about 60%/year

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

Under the Cover: Technology trends

- **DRAM (Memory)**
  - Capacity: about 60%/year (4x every 3 years)
  - Speed: about 10%/year



Outline

Textbook: [COD] 1.1-1.8, 1.10

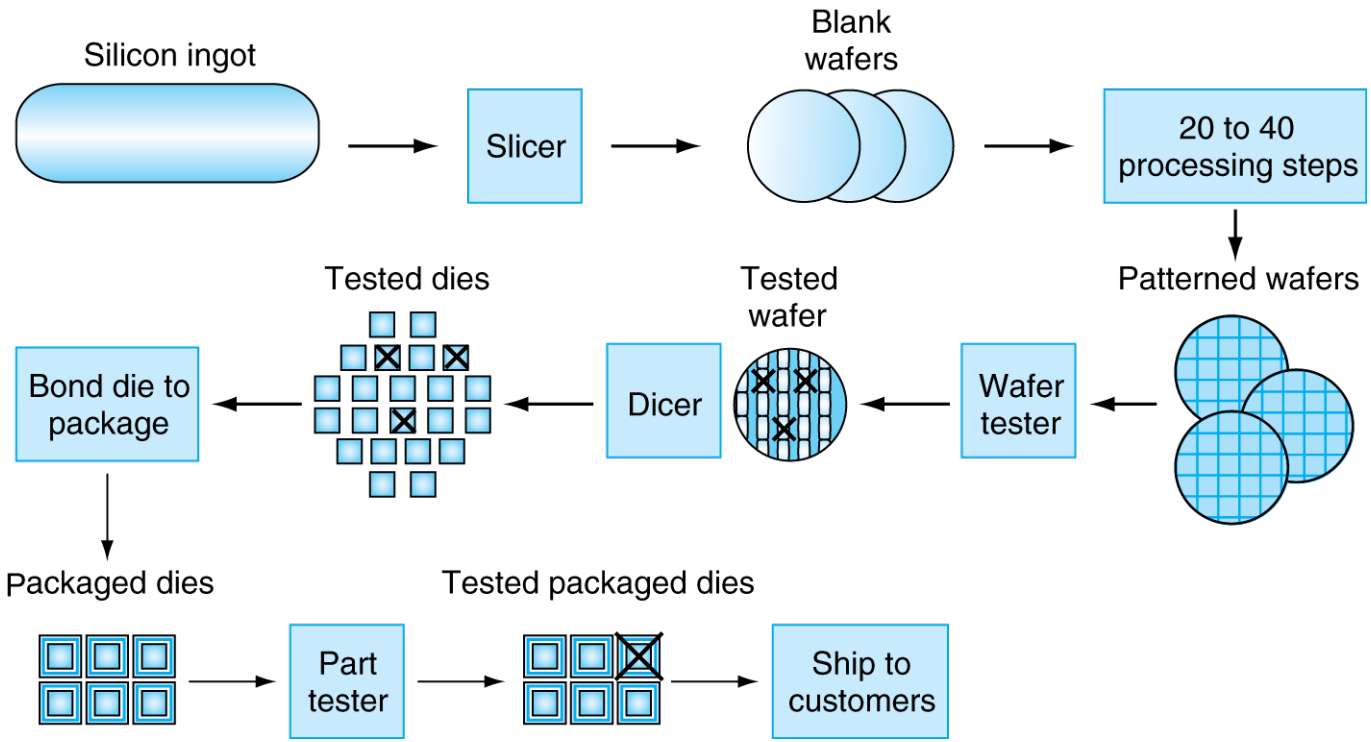
- Introduction
- Eight Great Ideas in Computer Architecture
- Below Your Program
- Under the Cover
- **Technologies for Building Processors and Memory**
- Performance
- Power Wall
- Switch from Uniprocessors to Multiprocessors
- Amdahl's Law

Semiconductor Technology (1)

- **Silicon: semiconductor**
- **Add materials to transform properties:**
  - Conductors
  - Insulators
  - Switch

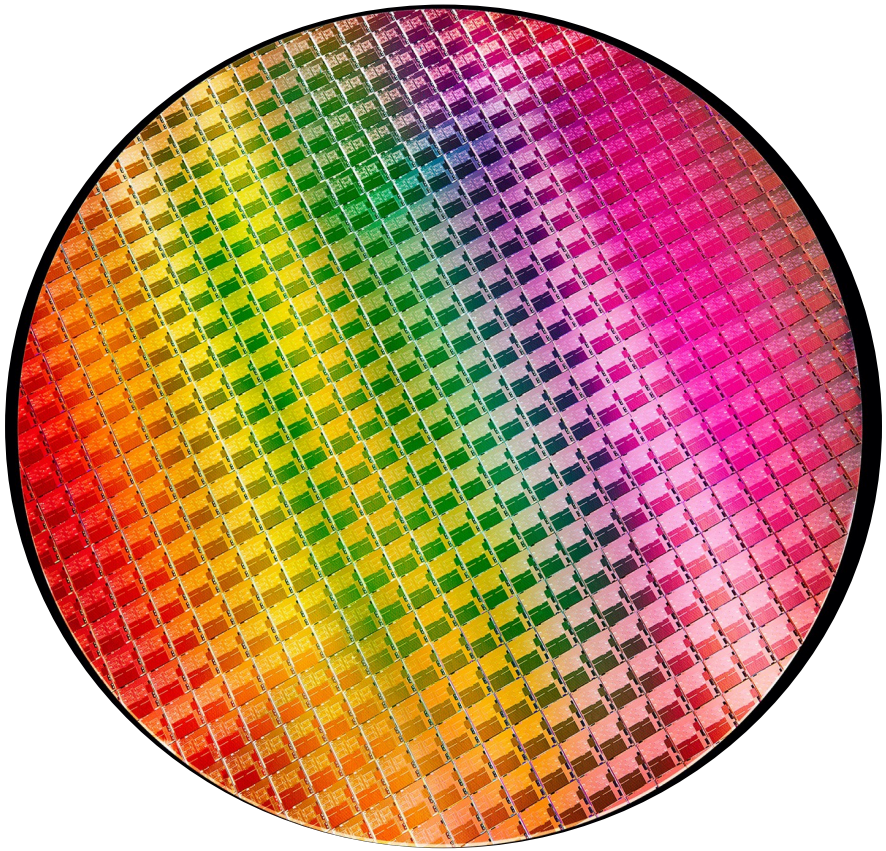
Semiconductor Technology (2)

- **Manufacturing ICs**
  - Yield: proportion of working dies per wafer



Semiconductor Technology (3)

- **Intel® Core 10th Gen**
  - 300mm wafer, 506 chips, 10nm technology
  - Each chip is 11.4 x 10.7 mm^2



Semiconductor Technology (4)

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

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

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

- **Integrated circuit cost: 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

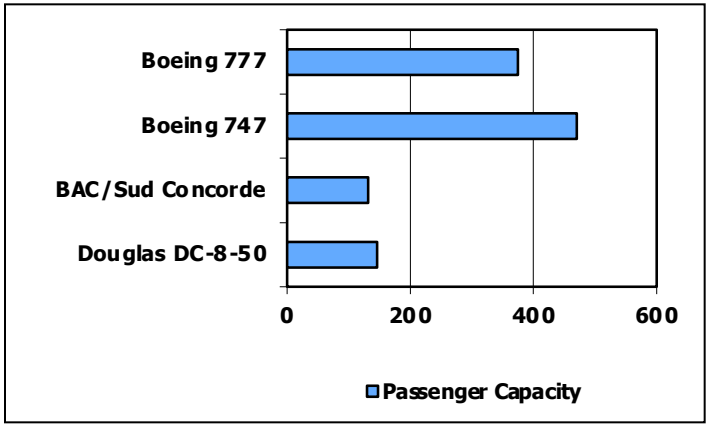
Outline

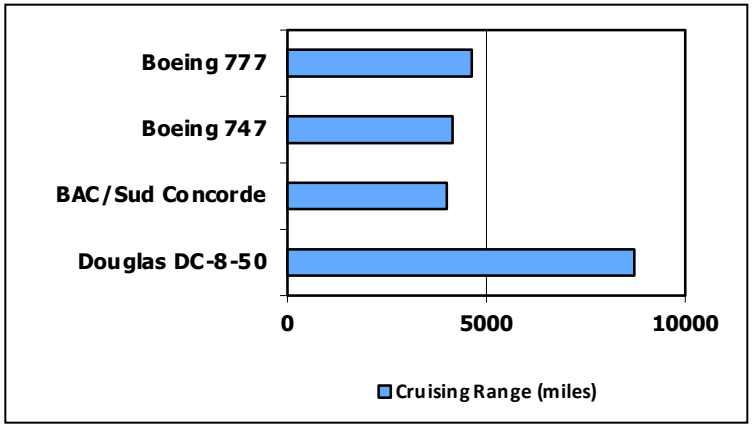
Textbook: [COD] 1.1-1.8, 1.10

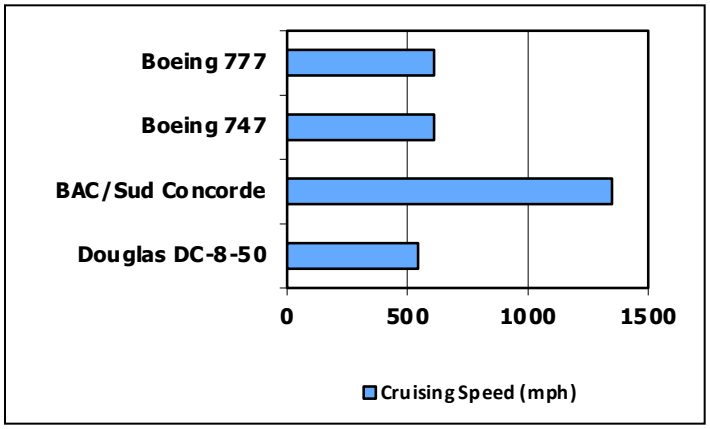
- Introduction
- Eight Great Ideas in Computer Architecture
- Below Your Program
- Under the Cover
- Technologies for Building Processors and Memory
- **Performance**
- Power Wall
- Switch from Uniprocessors to Multiprocessors
- Amdahl's Law

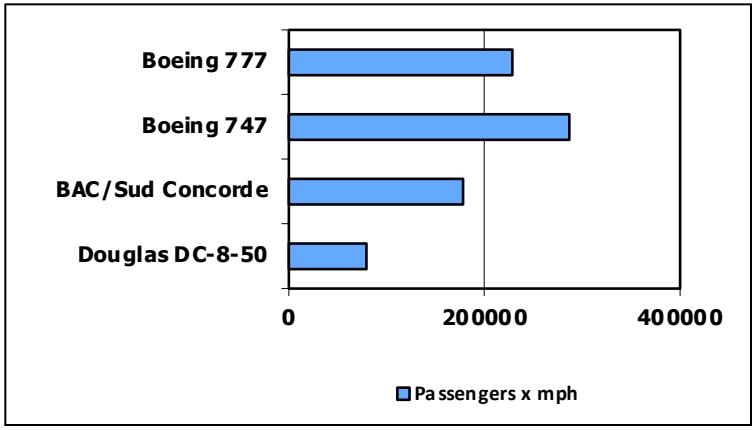
Defining Performance

- Question: Which aircraft performs the best??









Performance Metrics #1: Time

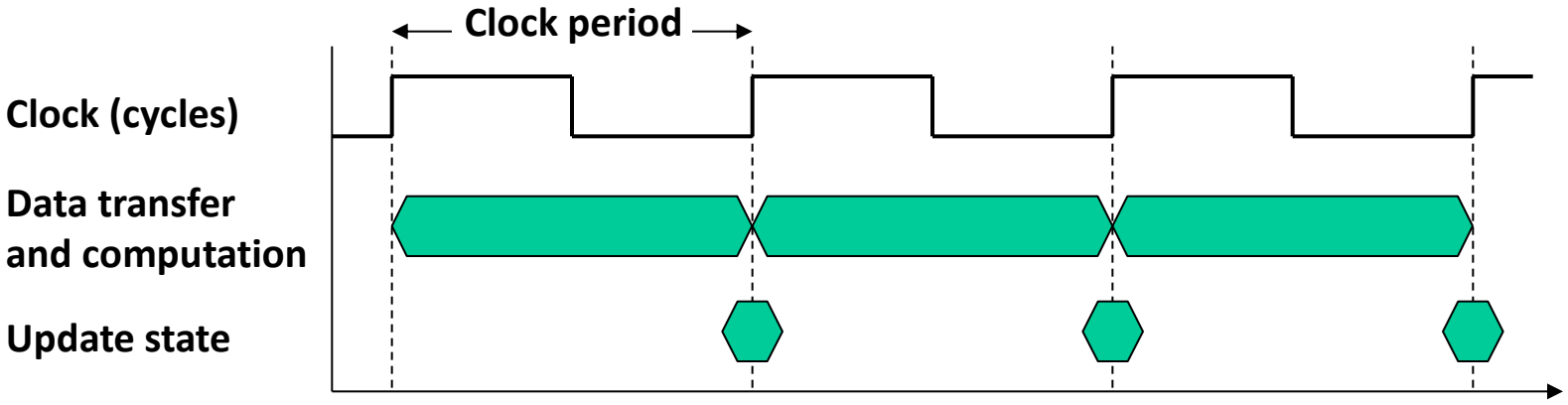
- **Wall-clock time, response time, or elapsed time**
  - Actual time from start to completion
  - Includes everything: CPU time for other programs as well as for itself, I/O, operating system overheads, etc

- **CPU (execution) time**
  - CPU time spent for a given program
  - user CPU time + system CPU time
  - e.g., results of UNIX `time` command

```
90.7u  12.9s  2:39  65%
```

Performance Metrics #1: Time

- **CPU Clocking**
  - Operation of digital hardware governed by a constant-rate clock
  - Clock period: duration of a clock cycle
    - e.g., 250ps = 0.25ns = 250×10^{-12}s
  - Clock frequency (rate): cycles per second
    - e.g., 4.0GHz = 4000MHz = 4.0×10^{9}Hz



Performance Metrics #1: Time

- **Decomposition of CPU (Execution) Time**

$$\text{CPU time} = \frac{\text{Seconds}}{\text{Program}}$$

$$= \frac{\text{Cycles}}{\text{Program}} \times \frac{\text{Seconds}}{\text{Cycle}}$$

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

This equation is called "Iron Law of CPU Performance."

- 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

Performance Metrics #1: Time

- **More on CPI (Clocks or Cycles Per Instruction)**
  - $CPI = \dfrac{\sum_{i=1}^{n} (CPI_i \times I_i)}{\text{Instruction Count}}$
  - CPI Example

Instruction Class	Frequency	CPI_i
ALU operations	43%	1
Loads	21%	2
Stores	12%	2
Branches	24%	2

$$CPI = 0.43 \times 1 + 0.21 \times 2 + 0.12 \times 2 + 0.24 \times 2$$

Performance Metrics #1: Time

- **Comparing CPIs of two CPUs**
  - Example question: What is the CPI of CPU_S and CPU_Q?

Instruction Type	Instr. count (millions)	Cycles per Instr. (CPI)	
		CPU_S	CPU_Q
Arithmetic & Logic	10	1	1
Load & Store	5	4	2
Branch	4	2	3
Miscellaneous (기타)	1	4	4

$CPI_S = (10 \times 1 + 5 \times 4 + 4 \times 2 + 1 \times 4) / (10 + 5 + 4 + 1) = 2.1$

$CPI_Q = (10 \times 1 + 5 \times 2 + 4 \times 3 + 1 \times 4) / (10 + 5 + 4 + 1) = 1.8$

Question: So, CPU_Q always performs better?

Performance Metrics #1: Time

- **Factors involved in the CPU Time**

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

	Instructions / Program	Cycles / Instruction	Seconds / Cycle
Program	V		
Compiler	V		
ISA	V	V	
Organization		V	V
Technology			V

Performance Metrics #2: Rate

- **MIPS (million instructions per second)**
  - $\text{MIPS} = \dfrac{\text{Instruction count}}{\text{Execution time} \times 10^6}$
  - Specifies performance (roughly) inversely to execution time
  - Easy to understand; faster machines means bigger MIPS
  - Problems
    - It does not take into account the capabilities of the instructions.
    - It varies between programs on the same computer.
    - It can even vary inversely with performance!!

- **MFLOPS (million floating-point operations per second)**

Performance Metrics: Ratio

- "X is *n* times faster than Y" means:

$$\frac{\text{Execution Time}_Y}{\text{Execution Time}_X} = n$$

- "X is *n*% faster than Y" means:

$$\frac{\text{Execution Time}_Y}{\text{Execution Time}_X} = 1 + \frac{n}{100}$$

- "X is *n* order of magnitude faster than Y" means:

$$\frac{\text{Execution Time}_Y}{\text{Execution Time}_X} = 10^n$$

Summarizing Performance

- **Arithmetic mean**
  (Time)

$$\frac{1}{n}\sum_{i=1}^{n} T_i$$

- **Harmonic mean**
  (Rate)

$$\frac{n}{\sum_{i=1}^{n} \frac{1}{R_i}}$$

- **Geometric mean**
  (Ratio)

$$\sqrt[n]{\prod_{i=1}^{n} Ratio_i}$$

Summarizing Performance: Arithmetic Mean

- **Used to summarize performance given in times**
  - Average Execution Time= ($\sum_{i=1}^{n}$ Execution Times) / n
  - Assumes each benchmark is run an equal no. of times

- **Weighted Arithmetic Mean**
  - Weighted Average Execution Time = $\sum_{i=1}^{n}$ (W_i X Execution Times) / $\sum_{i=1}^{n}$ W_i
  - One possible weight assignment: equal (weighted) execution time on some machine

Summarizing Performance: Harmonic Mean

- **Used to summarize performance in rates (e.g., MIPS, FLOPS):**
  - Harmonic Mean = $n \; / \; \sum_{i=1}^{n} (1 / R_i)$
  - Example
    - Four programs execute at 10, 100, 50 and 20 MFLOPS, respectively
    - Harmonic mean is 4 / (1/10 + 1/100 + 1/50 + 1/20) = 22.2 MFLOPS

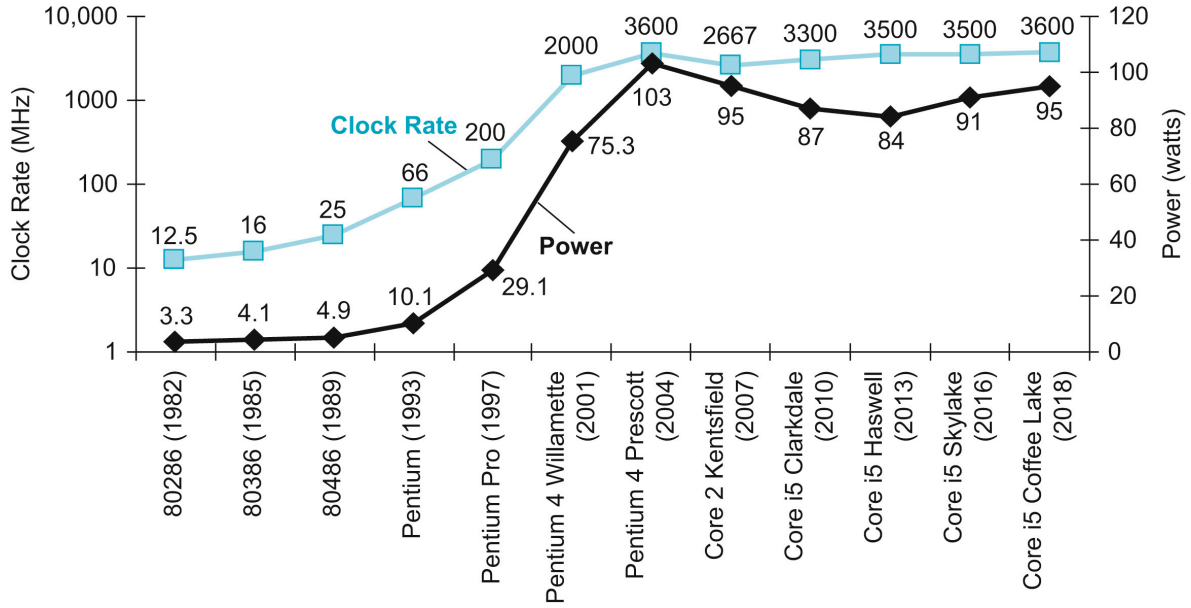
- **Weighted Harmonic Mean**
  - Weighted Harmonic Mean = $\sum_{i=1}^{n} W_i \; / \; \sum_{i=1}^{n} (W_i \; / \; R_i)$

Outline

Textbook: [COD] 1.1-1.8, 1.10

- Introduction
- Eight Great Ideas in Computer Architecture
- Below Your Program
- Under the Cover
- Technologies for Building Processors and Memory
- Performance
- **Power Wall**
- **Switch from Uniprocessors to Multiprocessors**
- **Pitfall: Amdahl's Law**

Power Trends



- **In CMOS IC technology**

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

×30 **5V → 1V** **×1000**

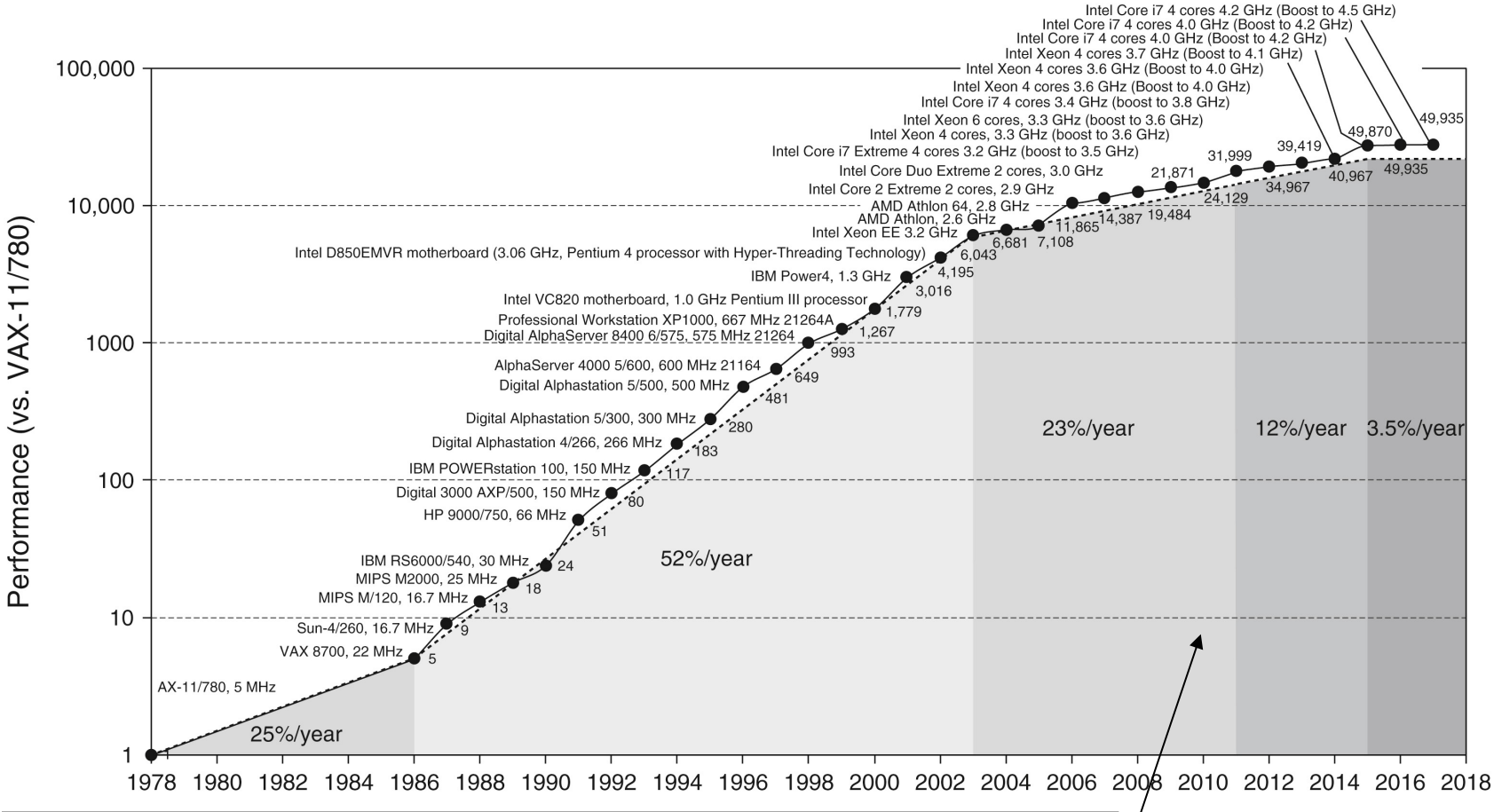
Reducing Power

- **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
  - How else can we improve performance?

Uniprocessor Performance



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

Pitfall: 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!**

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

SPEC CPU Benchmark

- **Programs used to measure performance**
  - Supposedly typical of actual workload
- **Standard Performance Evaluation Corp (SPEC)**
  - Develops benchmarks for CPU, I/O, Web, …
- **SPEC CPU2006**
  - 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
    - CINT2006 (integer) and CFP2006 (floating-point)

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

SPEC CPU Benchmark

- CINT2006 for Intel Core i7 920

Description	Name	Instruction Count x 10^9	CPI	Clock cycle time (seconds x 10^{-9})	Execution Time (seconds)	Reference Time (seconds)	SPECratio
Interpreted string processing	perl	2252	0.60	0.376	508	9770	19.2
Block-sorting compression	bzip2	2390	0.70	0.376	629	9650	15.4
GNU C compiler	gcc	794	1.20	0.376	358	8050	22.5
Combinatorial optimization	mcf	221	2.66	0.376	221	9120	41.2
Go game (AI)	go	1274	1.10	0.376	527	10490	19.9
Search gene sequence	hmmer	2616	0.60	0.376	590	9330	15.8
Chess game (AI)	sjeng	1948	0.80	0.376	586	12100	20.7
Quantum computer simulation	libquantum	659	0.44	0.376	109	20720	190.0
Video compression	h264avc	3793	0.50	0.376	713	22130	31.0
Discrete event simulation library	omnetpp	367	2.10	0.376	290	6250	21.5
Games/path finding	astar	1250	1.00	0.376	470	7020	14.9
XML parsing	xalancbmk	1045	0.70	0.376	275	6900	25.1
Geometric mean	–	–	–	–	–	–	25.7

SPEC Power Benchmark

- **Power consumption of server at different workload levels**
  - Performance: ssj_ops/sec
  - Power: Watts (Joules/sec)

$$\text{Overall ssj\_ops per Watt} = \left(\sum_{i=0}^{10} \text{ssj\_ops}_i\right) \Big/ \left(\sum_{i=0}^{10} \text{power}_i\right)$$

SPEC Power Benchmark

- **SPECpower_ssj2008 for Xeon X5650**

Target Load %	Performance (ssj_ops)	Average Power (Watts)
100%	865,618	258
90%	786,688	242
80%	698,051	224
70%	607,826	204
60%	521,391	185
50%	436,757	170
40%	345,919	157
30%	262,071	146
20%	176,061	135
10%	86,784	121
0%	0	80
Overall Sum	4,787,166	1,922
Σssj_ops/Σpower =		2,490