Computer Architecture

Computer Abstractions and Technology

An course overview

Instructors

- **Instructor:** Xianbin Gu (Bing) (xianbin.gu@nyu.edu)
    - Assistant professor of practice in computer science
    - Teaching a few courses, including ICDS
    - Running a programming club for XCPC
    - Office hours: 1 pm - 2 pm Wednesday in RM S713; if you need office hours at other time (in-person, or Zoom meeting), please send me emails.
- **LA:** ???
    - Undering hiring but we will have one soon

Agenda

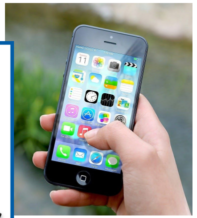
- Classes of computers
- Below your program
  - Course content
- Computer abstractions
  - Components of a computer
  - Abstractions
- Technologies for building Processors and Memory
- Accessing performance

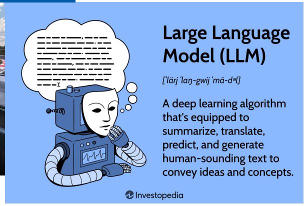
The computer revolution

Computers have led to a third revolution for civilization. It makes novel applications feasible,

- Cell phones
- World Wide Web
- Search Engines
- LLMs
- Computers in automobiles (auto pilot)

Computers are pervasive!




Large Language Model (LLM)

[ˈlärj ˈlaŋ-gwij ˈmä-dᵊl]

A deep learning algorithm that's equipped to summarize, translate, predict, and generate human-sounding text to convey ideas and concepts.

Investopedia

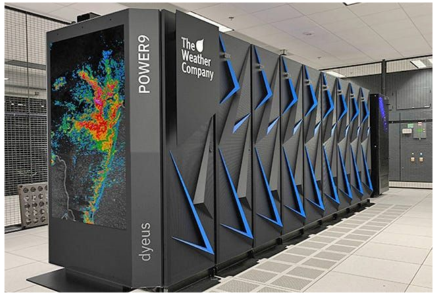
Classes of Computers

Although a common set of hardware technologies is used in all computers, different applications have their own characteristics.

- Personal computers (PC)
  - 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

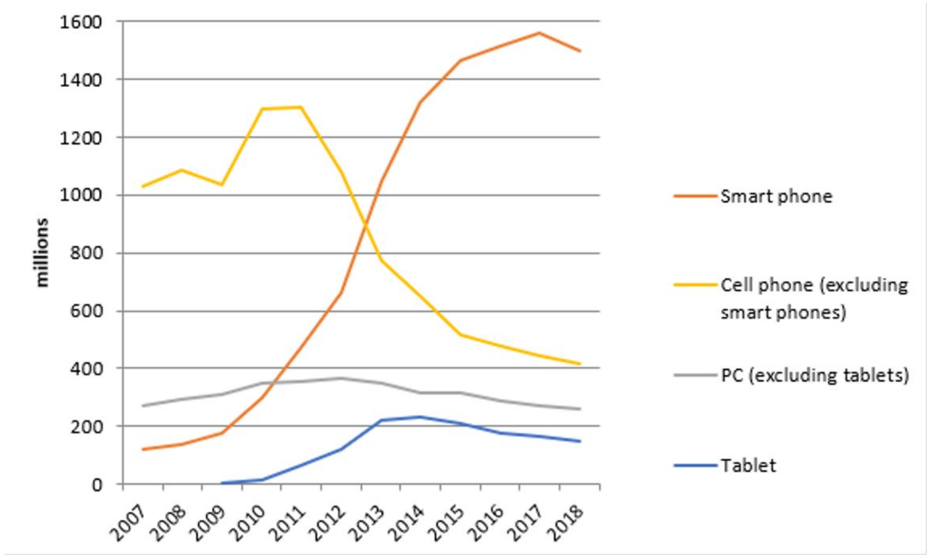
Classes of Computers

- Supercomputers
  - A type of server
  - 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 (e.g., microprocessors in the cars, and TVs)
  - Stringent power/performance/cost constraints



Classes of Computers

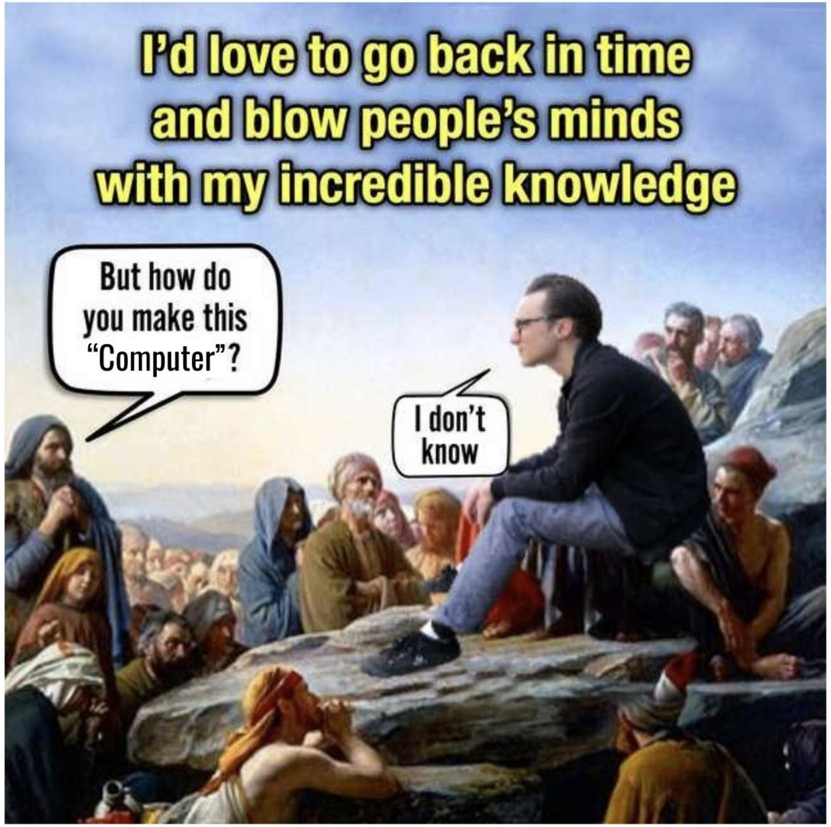
- Personal mobile device (PMD) is replacing PC:
  - They are battery operated
  - Wireless connectivity to the Internet
  - Affordable



We are people living in the Era of Computers

- Computers multiply our intellectual strength and change the way we search for new knowledge.
- There are great times in history when people built wonders.
- We should be proud of ours because we have computers.

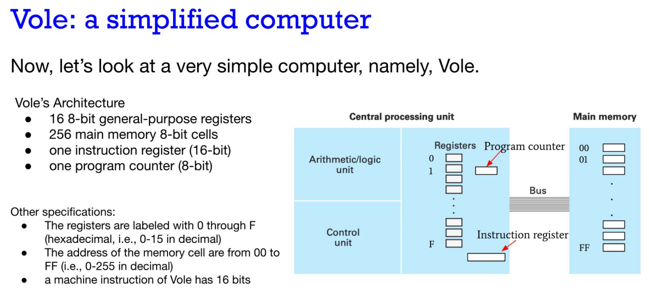
But if you travel back in time, and …


I'd love to go back in time and blow people's minds with my incredible knowledge
But how do you make this "Computer"?
I don't know

What you will learn in this course

- Programming in C
- How your programs are translated into the machine language
- How the hardware executes your programs
- How to design a processor

In short, we will go deeper into the computer and explore the details of how it executes your instructions.



Vole: a simplified computer

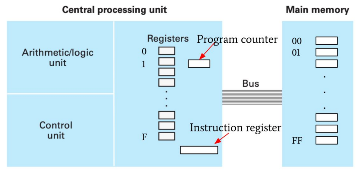
Now, let’s look at a very simple computer, namely, Vole.

Vole’s Architecture

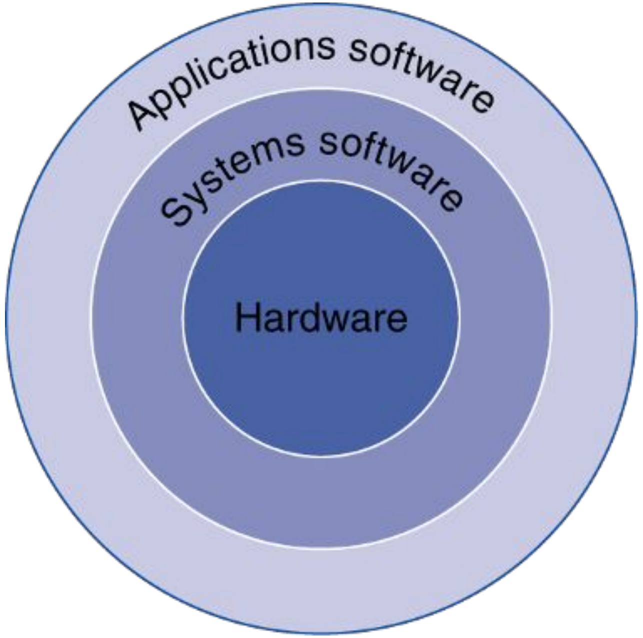
- 16 8-bit general-purpose registers
- 256 main memory 8-bit cells
- one instruction register (16-bit)
- one program counter (8-bit)

Other specifications:

- The registers are labeled with 0 through F (hexadecimal, i.e., 0-15 in decimal)
- The address of the memory cell are from 00 to FF (i.e., 0-255 in decimal)
- a machine instruction of Vole has 16 bits



Below your program



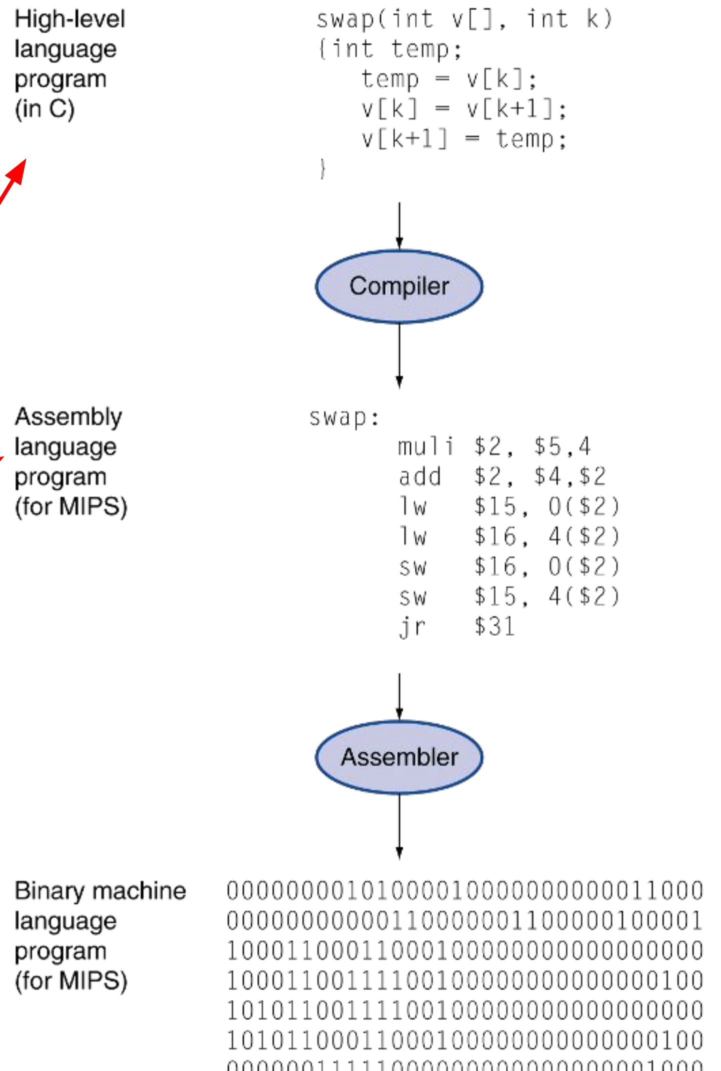
A program may have millions of line of code, and they finally are translated into machine language and executed by the hardware. This process is organized by layers.

- 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

We can divide the program code into three levels corresponding to the previous mentioned layers.

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



High-level language program (in C)
Compiler
Assembly language program (for MIPS)
Assembler
Binary machine language program (for MIPS)

```
swap(int v[], int k)
{int temp;
   temp = v[k];
   v[k] = v[k+1];
   v[k+1] = temp;
}
```

```
swap:
      muli $2, $5,4
      add  $2, $4,$2
      lw   $15, 0($2)
      lw   $16, 4($2)
      sw   $16, 0($2)
      sw   $15, 4($2)
      jr   $31
```

```
00000000101000010000000000011000
00000000000110000001100000100001
10001100011000100000000000000000
10001100111100100000000000000100
10101100111100100000000000000000
10101100011000100000000000000100
00000011111000000000000000001000
```

Course contents

- We will follow the layers and go from the top to the ground. (like landing)

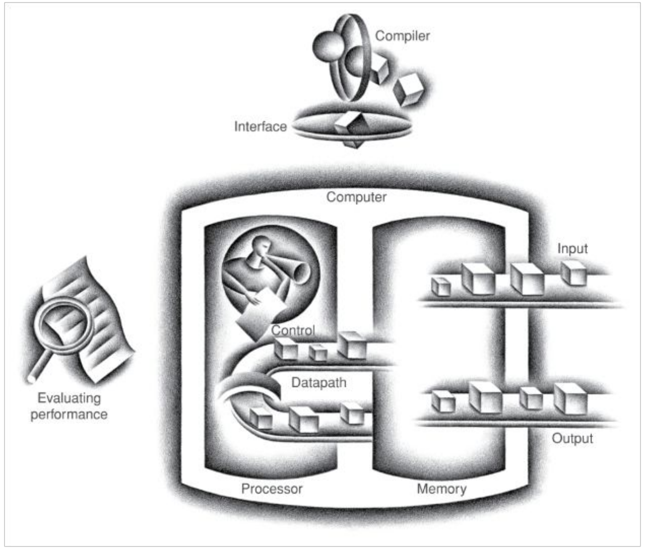
Seg 1: a high level program language		
Programming in C	~4 weeks	
Seg2: an assembly language		
MIPS instructions	~4 weeks	Midterm
Seg 3: processor		
Processor designing	~5 weeks	Final

- We have 8 assignments and 2 exams.

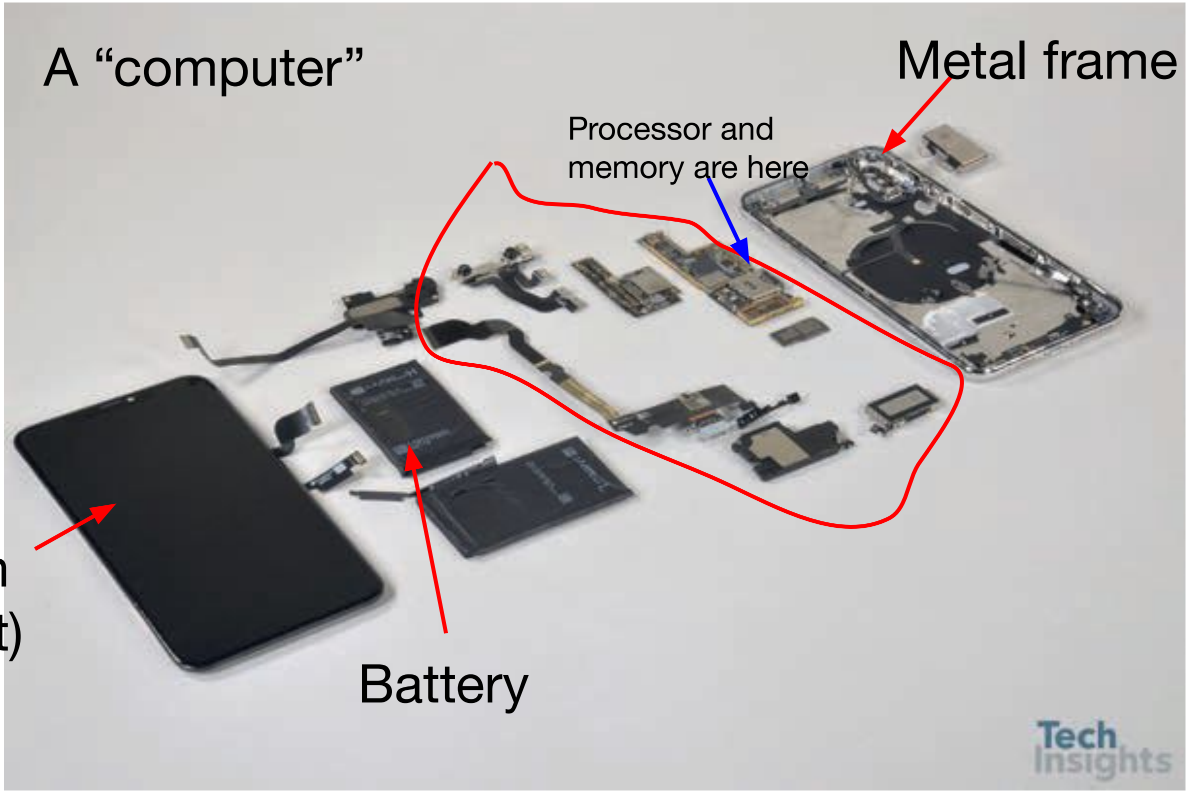
Computer Abstractions

Under the cover: Components of a Computer

- Input/Output
  - User-interface devices
    - Display, keyboard, mouse
  - Storage devices
    - Hard disk, CD/DVD, flash
  - Network adapters
    - For communicating with other computers
- Memory
  - Keeping programs and data when they are running
- Processor
  - Datapath
  - Control

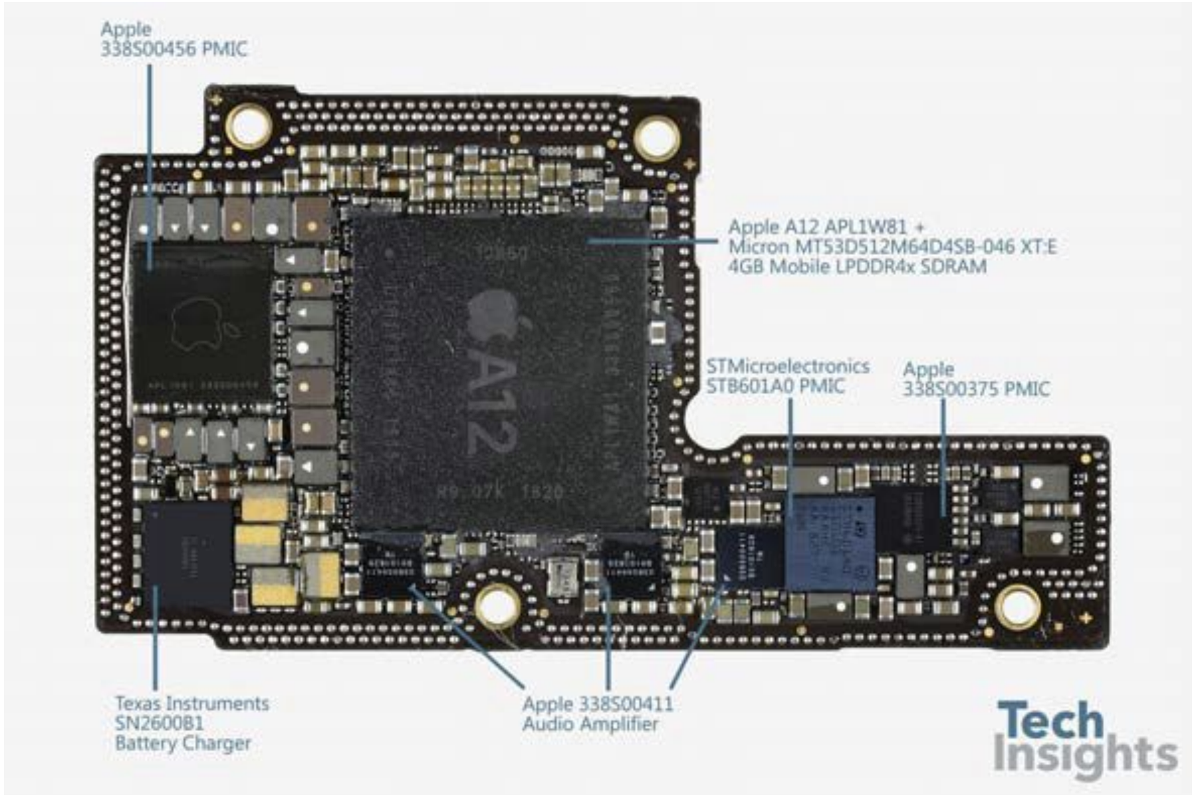


The Apple iPhone Xs Max

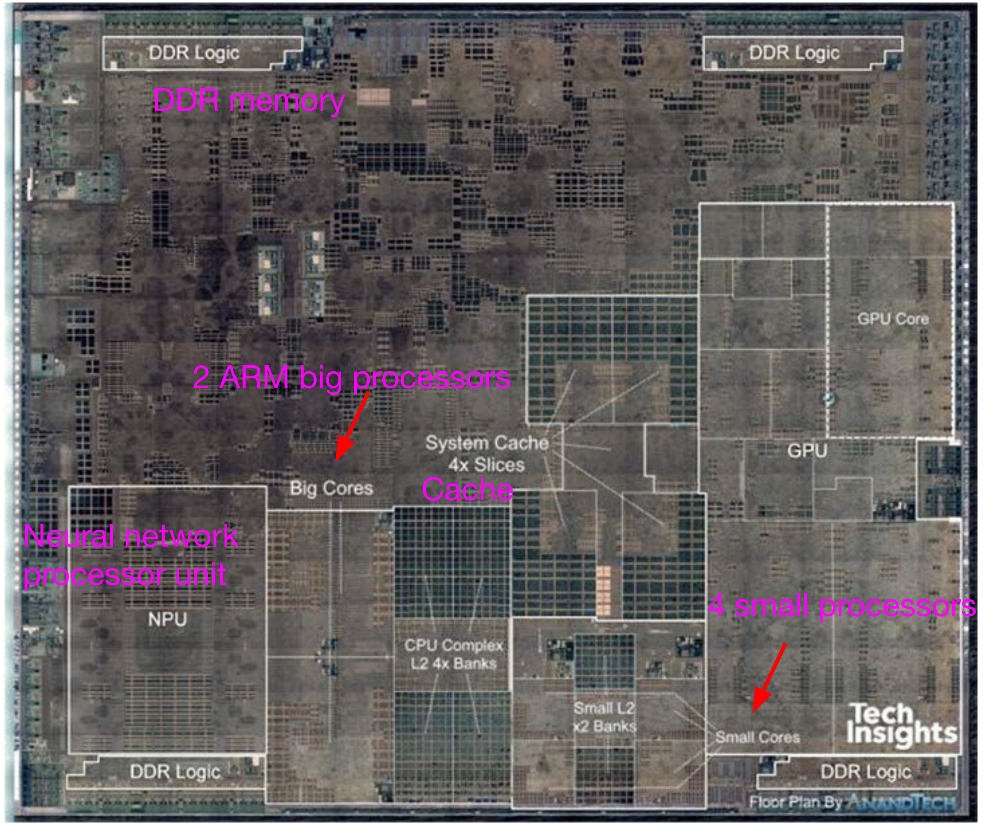


Touch screen (Input/Output)

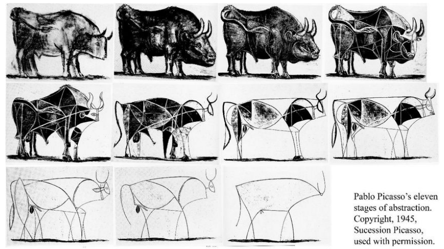
Details on the logic printed circuit board



Inside the processor: the incredible complexity



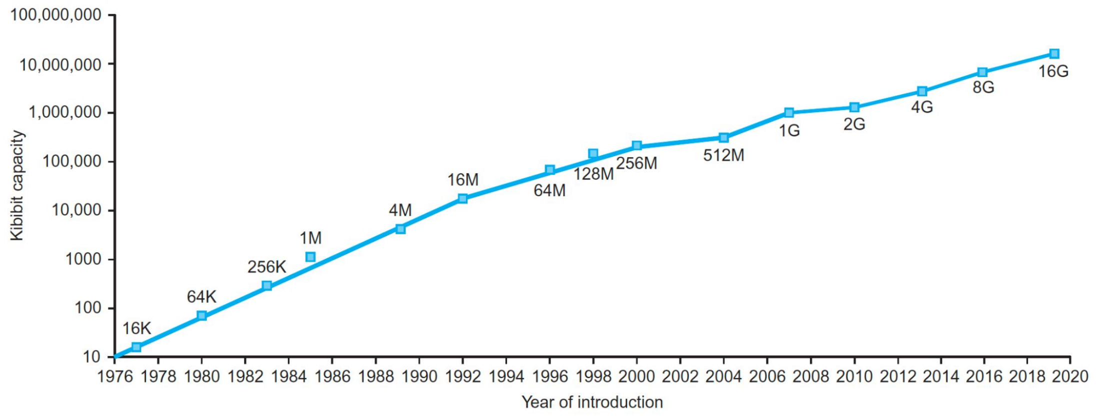
Abstractions: a way to manage complexity


Pablo Picasso's eleven stages of abstraction. Copyright, 1945, Sucession Picasso, used with permission.

- Abstraction: Lower-level details are hidden to offer a simpler model at higher levels. It helps us deal with complexity.
- Examples in practice:
  - Instruction set architecture (ISA): the hardware/software interface; it is a vocabulary by which the software communicates to the hardware.
    - The words of the vocabulary are called instructions.
  - Implementation: hardware that obeys the architecture abstraction.
    - Abstract interface enables many implementations of varying cost and performance to run identical software. E.g., Mac and PC both can run Python.
    - Google flutter ⇒ you can use it to develop apps for web, ios, and android

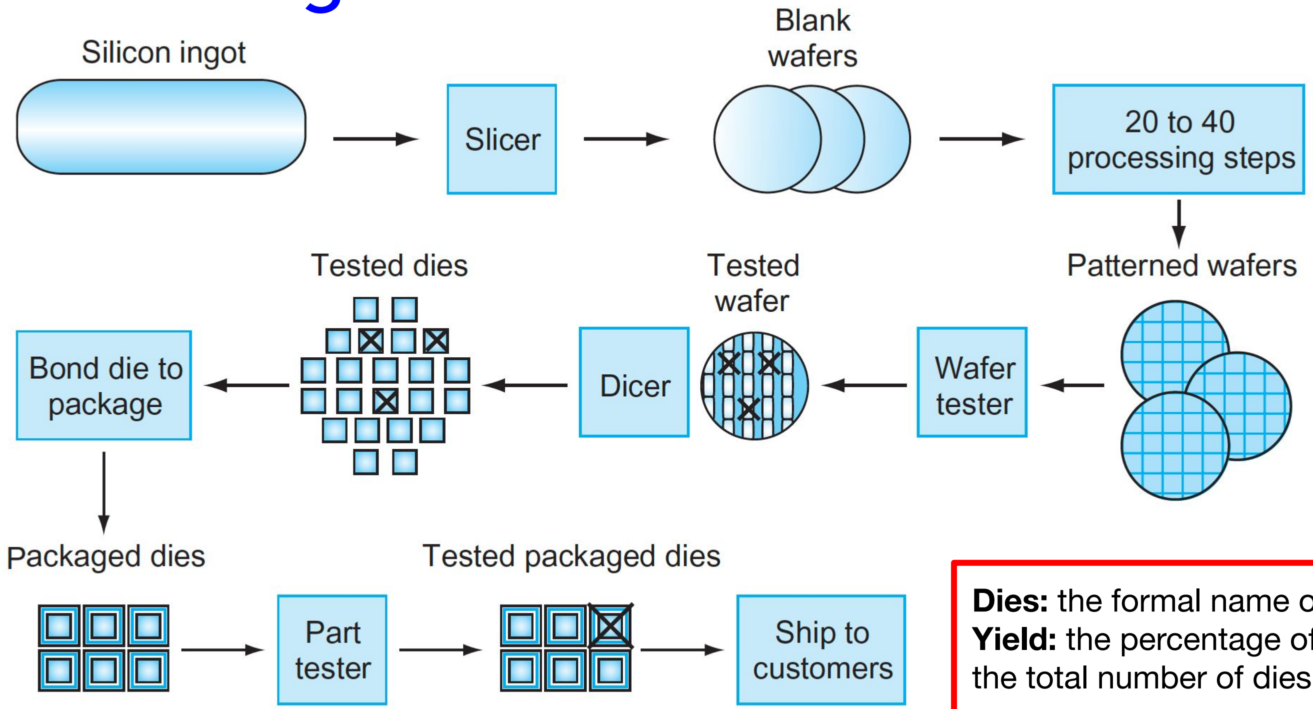
On the other hand, electronic technology continues envolve

- Processors and memory become more and more complex:
  - capacity and performance are increasing
  - cost is reducing



Moore's Law: the number of transistors on a microchip doubles approximately every two years, leading to an exponential increase in computing power and a decrease in relative cost.

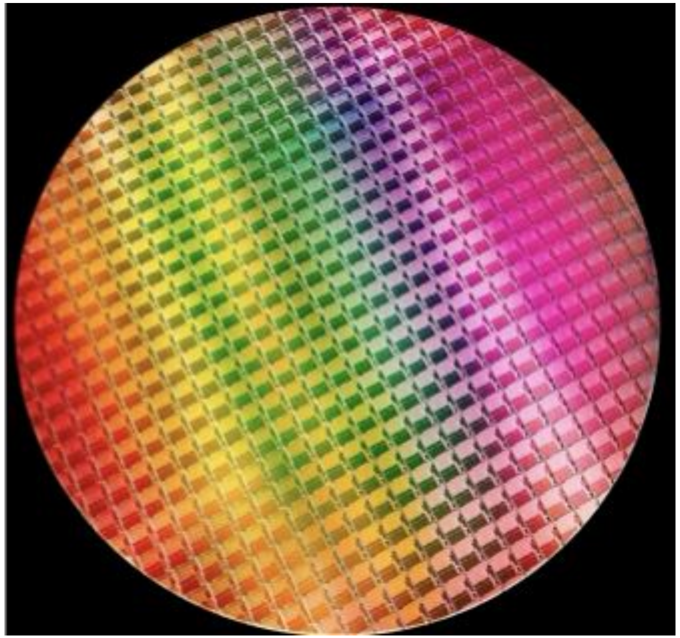
Manufacturing IC



Dies: the formal name of chips
Yield: the percentage of good dies from the total number of dies on the wafer

After being sliced from the silicon ingot, blank wafers are put through 20 to 40 steps to create patterned wafers. These patterned wafers are then tested with wafer tester, and a map of the good parts is made. Then, the wafers are diced into dies. In this figure, one wafer produced 20 dies, of which 17 passed testing. The yield was 17/20 (i.e., 85%). These good dies are then bonded into packages and tested one more time before shipping the packaged parts to customers. One bad packaged part was found in this final test.

Intel Core 10th Gen



- 300mm wafer, 506 chips, 10nm technology
- Each chip is 11.4 x 10.7 mm

Integrated Circuit Cost

- The cost of an IC rises quickly as the die size increases, due to the lower yield and the smaller number of dies that fit on a wafer.
- Using smaller size transistors and wires to **shrink** the size of a die.
    - A 7-nanometer (nm) processor means that the smallest feature size on the die is 7nm.
    - Recently, 4nm processor has been released.

Integrated Circuit Cost

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

Straightforwardly derived

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

Approximation, not subtracting the area near the border of the round wafer

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

Based on empirical observations of yields at IC factories; being nonlinear

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

Accessing the Performance

Performance is an important attribute for computer users and designers.

- When we say one computer has better performance than another, what do we mean?
- There are different ways in which performance can be determined.

Response Time and Throughput

- Response time: How long it takes to do a task, also called **execution time**
- Throughput: Total work done per unit time
  - e.g., tasks/transactions/… per hour
- We will focus on **execution time** for now…
  - To maximize the performance, we want to minimize the execution time for some task ⇒ so, we can relate performance and execution time for a computer.

Relative Performance

For computer X, the performance can be defined as,

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

So, for computer X and Y, if we have

$$\text{Performance}_X > \text{Performance}_Y$$

$$\frac{1}{\text{Execution time}_X} > \frac{1}{\text{Execution time}_Y}$$

$$\text{Execution time}_Y > \text{Execution time}_X$$

We know X is faster than Y because of the execution time on Y is longer than that on X.

Relative Performance

- “X is n times faster than Y” means

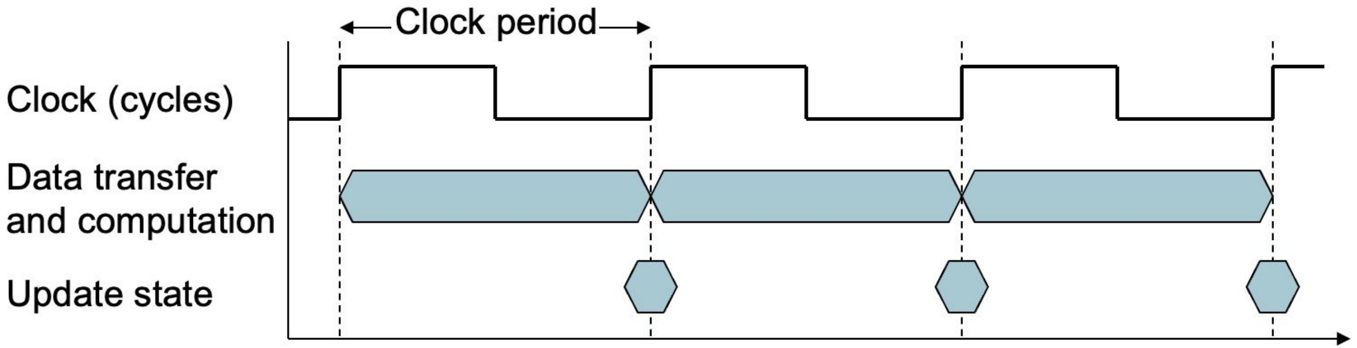
$$\frac{\text{Performance}_X}{\text{Performance}_Y} = \frac{\text{Execution time}_Y}{\text{Execution time}_X} = n$$

Measuring Execution Time

- Elapsed time:
  - the total time to complete a task, also named as response time, wall clock time; it includes disk accesses, memory access, input/output activities, operating system overhead.
  - **Determines system performance**
- CPU time:
  - Time spent processing a given job (only the cpu, discount I/O time, and other jobs' share)
  - It comprises User CPU time and system CPU time.
    - User CPU time: the CPU time spent in the program;
    - System CPU time: the CPU time spent in the operating system performing tasks on behalf of the program.
    - It is hard to differentiate between system and user CPU time, because
      - it is hard to assign responsibility for operating system activities to one user program rather than another.
      - The functionality differences among operating systems.
  - **Determines CPU performance**

CPU Clocking

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



- Clock period: duration of a clock cycle
  - e.g., 250 picosecond = 0.25 ns = 250 x 10^{-12} second
- Clock rate (frequency): cycles per second
  - e.g., 4.0 GHz = 4000MHz = 4.0 x 10^{9} Hz

CPU Time for a program

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

Performance improved by

- Reducing number of clock cycles
- Increasing clock rate
- Hardware designer must often trade off clock rate against cycle count

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 x clock cycles
- How fast must Computer B clock be?

$$\text{Clock Rate}_B = \frac{\text{Clock Cycles}_B}{\text{CPU Time}_B} = \frac{1.2 \times \text{Clock Cycles}_A}{6\text{s}}$$

$$\text{Clock Cycles}_A = \text{CPU Time}_A \times \text{Clock Rate}_A$$

$$= 10\text{s} \times 2\text{GHz} = 20 \times 10^9$$

$$\text{Clock Rate}_B = \frac{1.2 \times 20 \times 10^9}{6\text{s}} = \frac{24 \times 10^9}{6\text{s}} = 4\text{GHz}$$

Instruction Performance

Clock cycles per instruction (CPI): Average number of clock cycles per instruction for a program or program fragment.

- Different instruction may take different amounts of time depending on what they do, CPI is an average of all the instructions executed in the program.

The number of clock cycles required for a program can be written as

CPU clock cycles = Instructions for a program x CPI

Instruction Count and CPI

$$\text{Clock Cycles} = \text{Instruction Count} \times \text{Cycles per Instruction}$$

$$\text{CPU Time} = \text{Instruction Count} \times \text{CPI} \times \text{Clock Cycle Time}$$

$$= \frac{\text{Instruction Count} \times \text{CPI}}{\text{Clock Rate}}$$

- 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

CPI Example

- Computer A: Cycle Time = 250ps, CPI = 2.0
- Computer B: Cycle Time = 500ps, CPI = 1.2
- Same ISA (which means the instruction counts are the same)
- Which is faster, and by how much?

$$\text{CPU Time}_A = \text{Instruction Count} \times \text{CPI}_A \times \text{Cycle Time}_A$$

$$= \text{I} \times 2.0 \times 250\text{ps} = \text{I} \times 500\text{ps} \longleftarrow \boxed{\text{A is faster…}}$$

$$\text{CPU Time}_B = \text{Instruction Count} \times \text{CPI}_B \times \text{Cycle Time}_B$$

$$= \text{I} \times 1.2 \times 500\text{ps} = \text{I} \times 600\text{ps}$$

$$\frac{\text{CPU Time}_B}{\text{CPU Time}_A} = \frac{\text{I} \times 600\text{ps}}{\text{I} \times 500\text{ps}} = 1.2 \longleftarrow \boxed{\text{…by this much}}$$

CPI in More Details

If different instruction classes take different number of cycles, the total clock cycles is,

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

We can calculate the weighted average CPI,

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

CPI Example

Alternative compiled code sequences (sequence 1 and 2) 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 = 2+1+2 = 5

- Clock cycles = 2*1+1*2+2*3 = 10
- Aver. CPI = 10/5 = 2

Sequence 2: IC = 4+1+1 = 6

- Clock cycles = 4*1+1*2+1*3 = 9
- Aver. CPI = 9/6 = 1.5

Power Trends

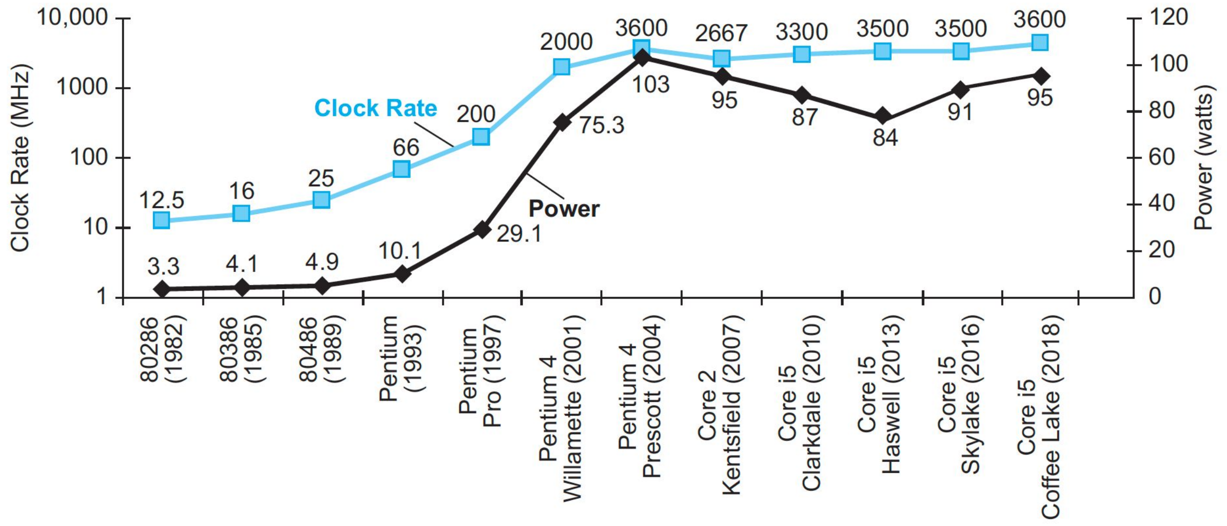


FIGURE 1.16 Clock rate and Power for Intel x86 microprocessors over nine generations and 36 years. The Pentium 4 made a dramatic jump in clock rate and power but less so in performance. The Prescott thermal problems led to the abandonment of the Pentium 4 line. The Core 2 line reverts to a simpler pipeline with lower clock rates and multiple processors per chip. The Core i5 pipelines follow in its footsteps.

- Both clock rate and power increased rapidly for decades, and then, flattened or drop off recently.
- We have run into the practical power limit for cooling commodity microprocessors.

Dynamic Energy

- **CMOS** (Complementary metal oxide semiconductor) is the dominant technology for IC.
- The primary source of energy consumption of CMOS is called dynamic energy, which is the energy consumed when transistors switch states from 0 to 1 and vice versa. We have the following equation for the power consumption,

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

If we want to increase the frequency by 1000 while the power only grew by 30, what can we do? ⇒ reducing the voltage

Note: Capacitive loads include energy stored in materials and devices, such as capacitors, and cause changes in voltage to lag behind changes in current.

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 the performance?

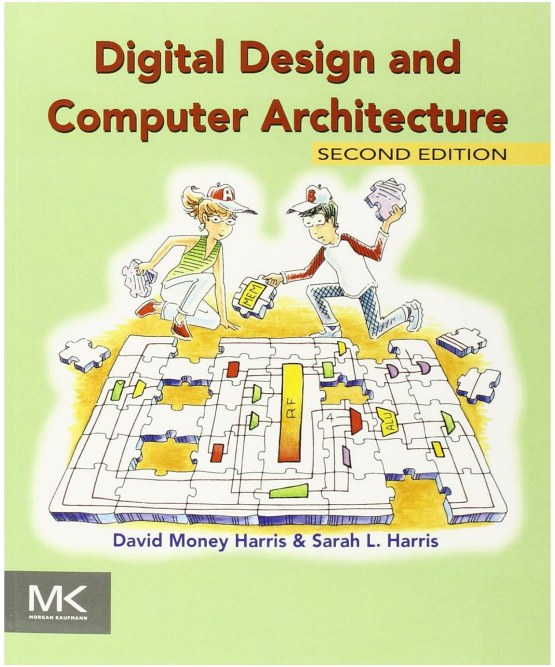
The switch from uniprocessor to multiprocessors

- Increasing the throughput rather than continuing to decrease the response time on the single processor.
- Multiplecore 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 between synchronization

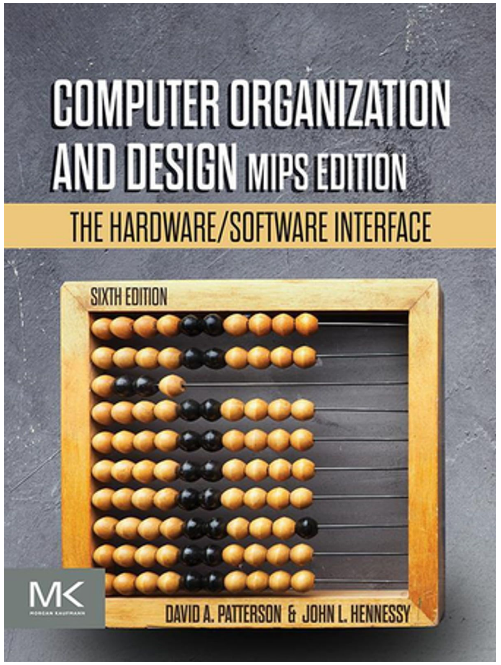
Summary

- Hierarchical layers of abstraction
  - In both hardware and software
- Instruction set architecture
  - The hardware/software interface (e.g., the machine language)
- Cost evaluation
- Performance evaluation
- Power is a limiting factor
  - Using parallelism to improve performance

Reference



Digital Design and Computer Architecture
SECOND EDITION
David Money Harris & Sarah L. Harris
MK



COMPUTER ORGANIZATION AND DESIGN MIPS EDITION
THE HARDWARE/SOFTWARE INTERFACE
SIXTH EDITION
DAVID A. PATTERSON & JOHN L. HENNESSY
MK