Computer Architecture

Ngo Lam Trung, Pham Ngoc Hung, Hoang Van Hiep
Department of Computer Engineering
School of Information and Communication Technology
Hanoi University of Science and Technology
E-mail: [trungnl, hungpn, hiephv]@soict.hust.edu.vn

Course administration

- Textbook: [Required] Computer Organization and Design RISC-V 2nd edition, Patterson & Hennessy 2021.

  [Optional] Computer Organization and Architecture, 10^{th} Edition, William Stalling
- Slides: pdf
- Schedule: as in timetable

Course syllabus

- Chapter 1: Introduction
- Chapter 2: Computer System and Interconnection
- Chapter 3: Instruction Set Architecture
- Chapter 4: Computer Arithmetic
- Chapter 5: CPU
- Chapter 6: Memory
- Chapter 7: I/O system
- Chapter 8: Multicores and multiprocessors

Computers are so important

- Current modern life
  - Industrial revolutions, the 3rd (Automation) and the 4th (Digital revolution).
  - Cell phones, the Internet, Grab, Google Maps...
  - WWW, search engines, social networks, e-commerce…
  - Robotics, EV, UAV, self-driving cars,…
- Future
  - Tailored medical care based on individual genome.
  - Super-human: transfer human’s brain to a mechanical body (robot) for interstellar traveling (The Matrix movies franchise, 1999, Michio Kaku, *Physics of the Future 2011* and *The Future of the Mind 2015).*
  - *…and many more*

Outcomes from this course

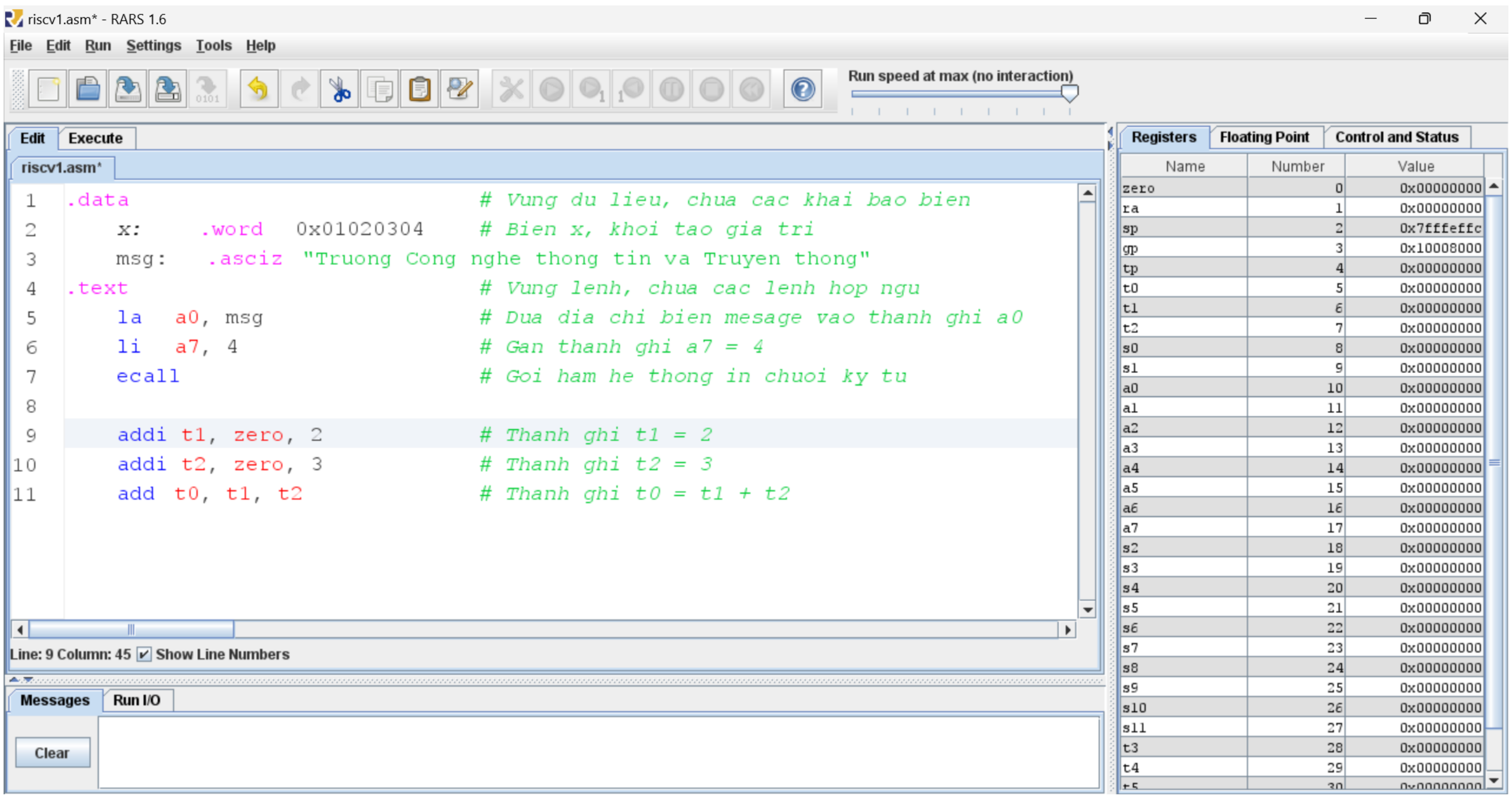
- Computer Architecture and Organization
  - Understanding of basic computer system organization.
  - Abstraction and instruction set architecture: how high-level language programs translate into computer language programs, and how hardware execute the latter programs.
  - Hardware/software interface, and how software instructs hardware to perform functions.
- Computer performance
  - How to evaluate performance
  - Basic techniques to improve computer performance.

Study guide

- Do read the textbook and APPENDIX A!
- Attend class regularly, stay focused.
- Comprehend all exercises and homework.
- Old-school approach: pen and paper for doing exercise and taking notes.
- Experience in C/C++ will be useful.
- Code of conduct:
  - No web surfing, music, video, game in class.
  - Food is not allowed (water/soft drink OK).
- Mid-term and Final exam will be online quiz, with topics from exercises and homework.

Homework/exercises

- RISC-V assembly programming
- RISC-V simulator (RARS 1.6)



Chapter 1: Introduction

1. Computer Abstraction and Technology
2. Performance Evaluation

[with materials from *Computer Organization and Design RISC-V Edition*,
Patterson & Hennessy, ©2021, MK
and M.J. Irwin's presentation, PSU 2008]

1. Computer Abstraction and Technology

- What is a computer?
- Computer classification
- Computer generations
- The key of computer evolution: IC making technology
- Computer organization

1. Computer Abstraction and Technology

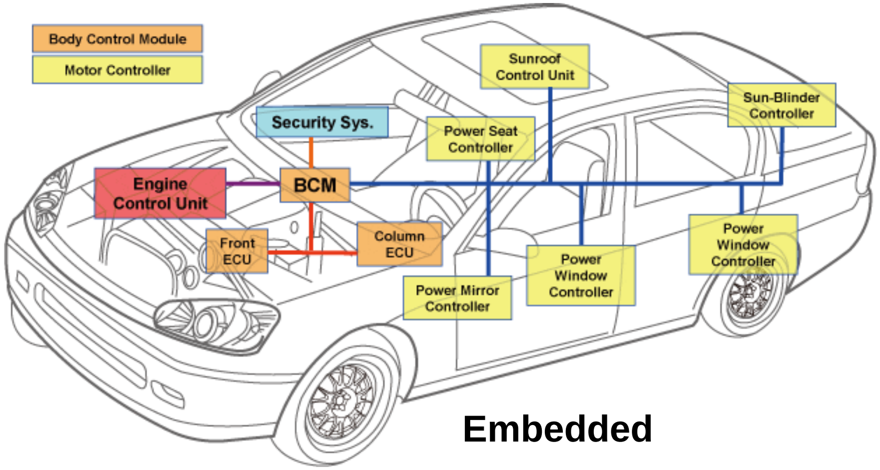
- **What is a computer?**
- A machine that
  - Accepts input data
  - Processes data by executing a stored program
  - Produces output
- Which one is computer?



Classes of Computers

- Supercomputers
  - Super fast + expensive for high-end applications
- Server
  - Network based
  - High capacity, performance, reliability
  - Range from small servers to building sized
- Desktop computers
  - General purpose, variety of software
  - Subject to cost/performance tradeoff
- Embedded computers
  - Hidden as components of systems
  - Stringent power/performance/cost constraints

Dominant look and feel of computer classes



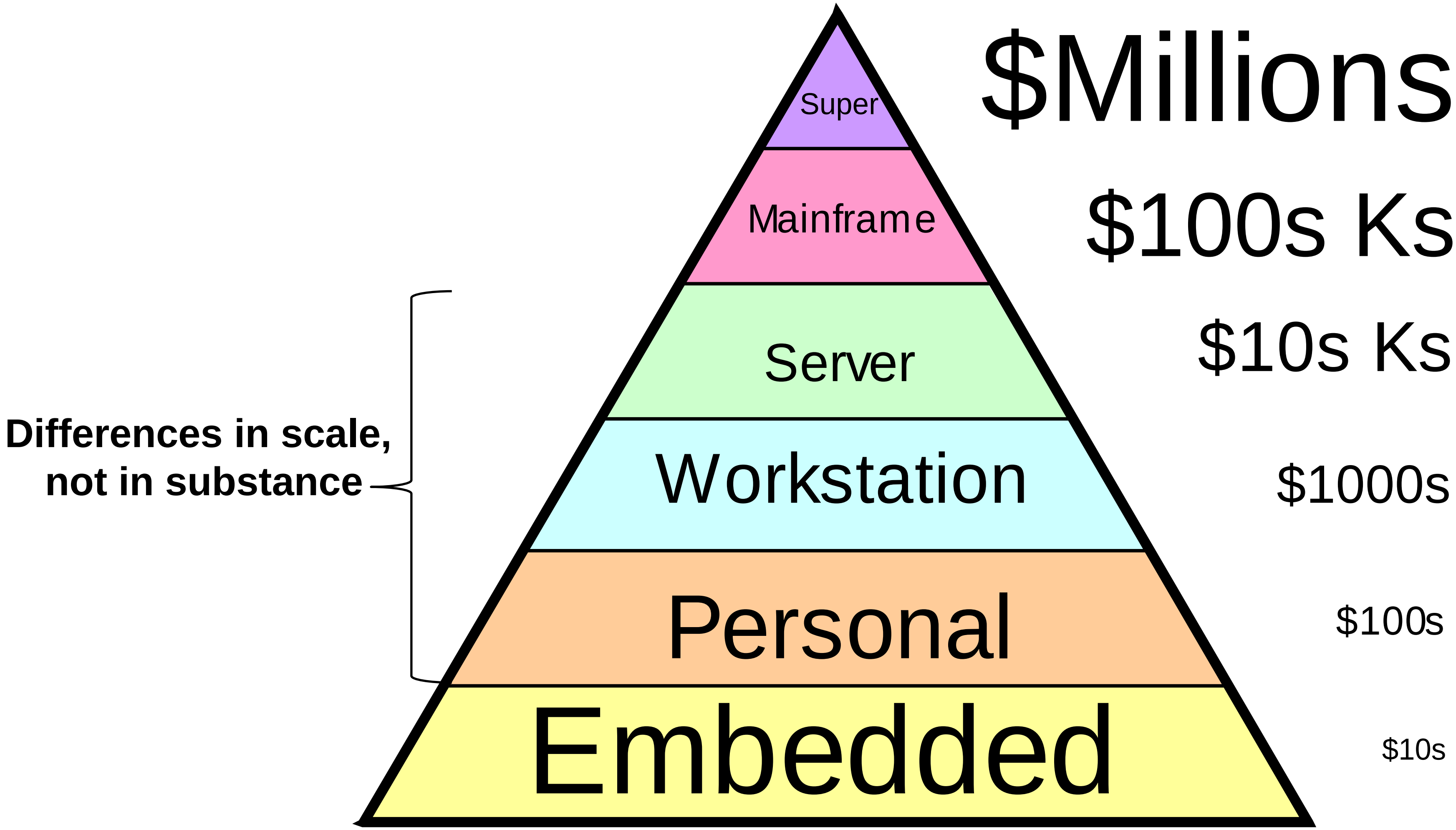
Embedded

PC

Server

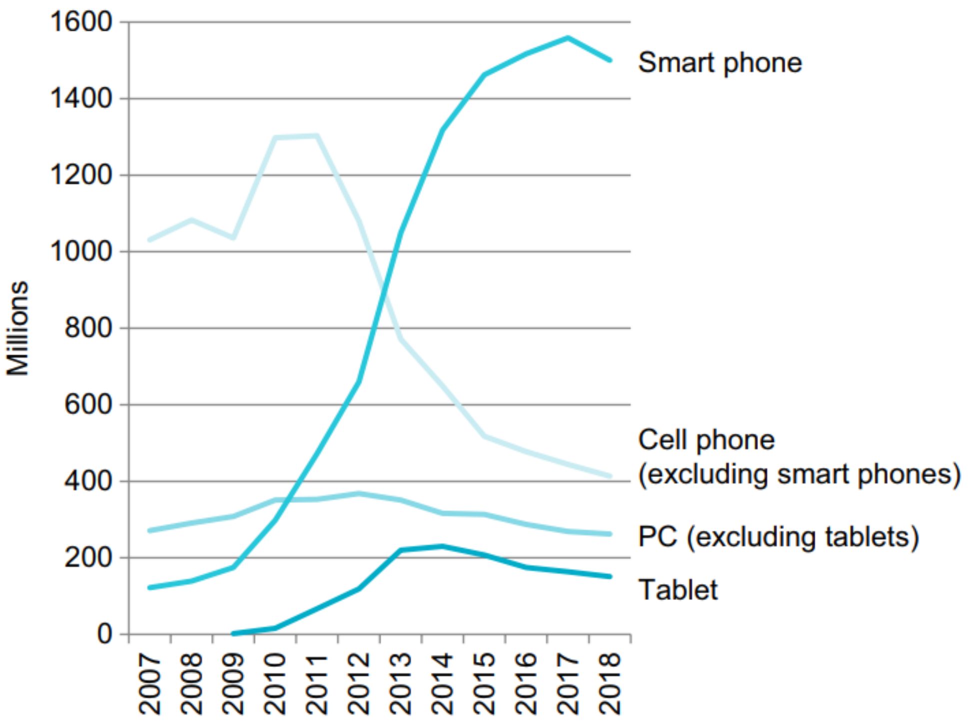
Super computer

Price/performance of computer classes



Post-PC era

- PDA, smart phone, tablet…
- Smart TV, set top box…
- Cloud computing (AMZ EC2, cloud gaming…)



The number manufactured per year of tablets and smart phones

[Smartphone sales worldwide 2007-2023 | Statista]

Seven important ideas in computer architecture

Simplification via abstraction

Make common cases fast

Performance via Parallelism

Performance via Pipelining

Performance via Prediction

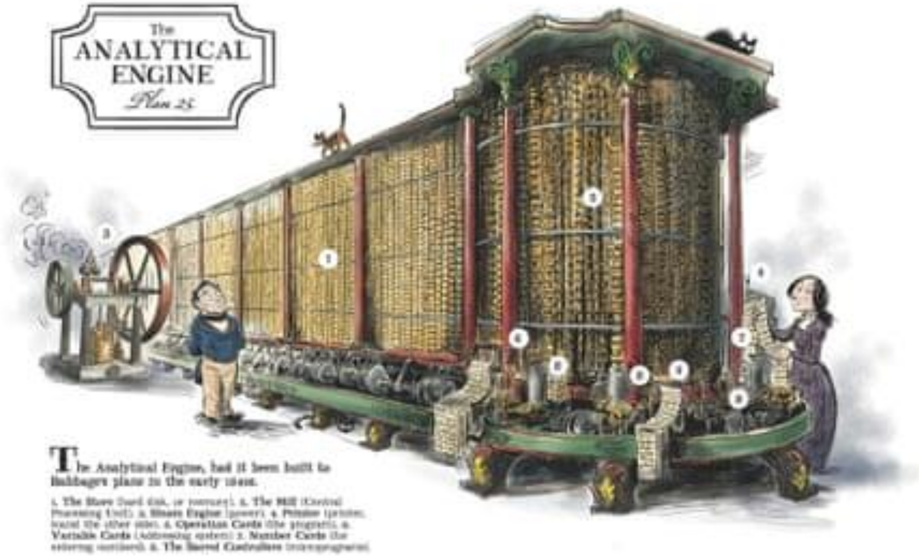
Memory hierarchy

Dependability via redundancy

A brief history of computers

- 0th generation: mechanical/analog calculators
  - Jacquard’s punch card: for textile factories, later used for the first computers
  - Pascalite machine
  - Babage’s Analytical Engine
  - Ada Lovelace: first computer program!!!

Pascalite machine

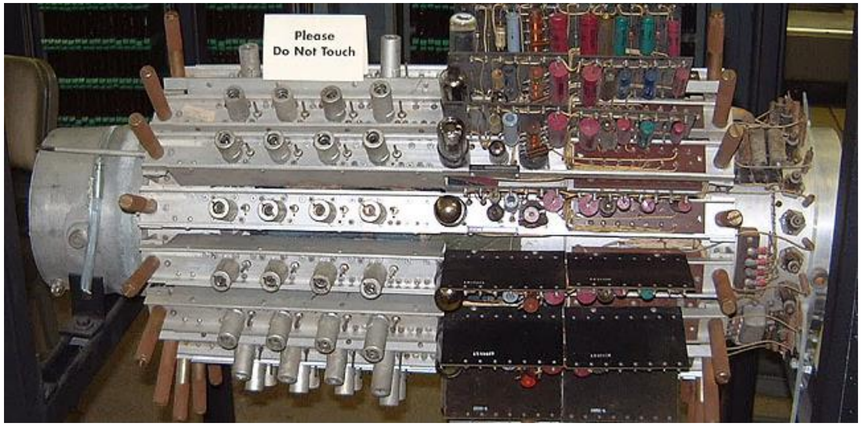


Babbage’s Analytical Engine (plan 25)

[Curiosity Stream - Calculating Ada: The Countess of Computing]

A brief history of computers

- 1^{st} generation: Vacuum tubes
  - ENIAC: 1^{st} general purpose computer
    - Computing artillery-firing tables
    - Enormous in size and energy consumption
  - IAS: computer with Von Newman architecture
    - Memory, ALU, Control, Input/Output, stored-program concept
  - UNIVAC: 1^{st} commercial computer



A brief history of computers

- 2nd generation: transistor
- Computer became smaller and faster

IBM System/360

A brief history of computers

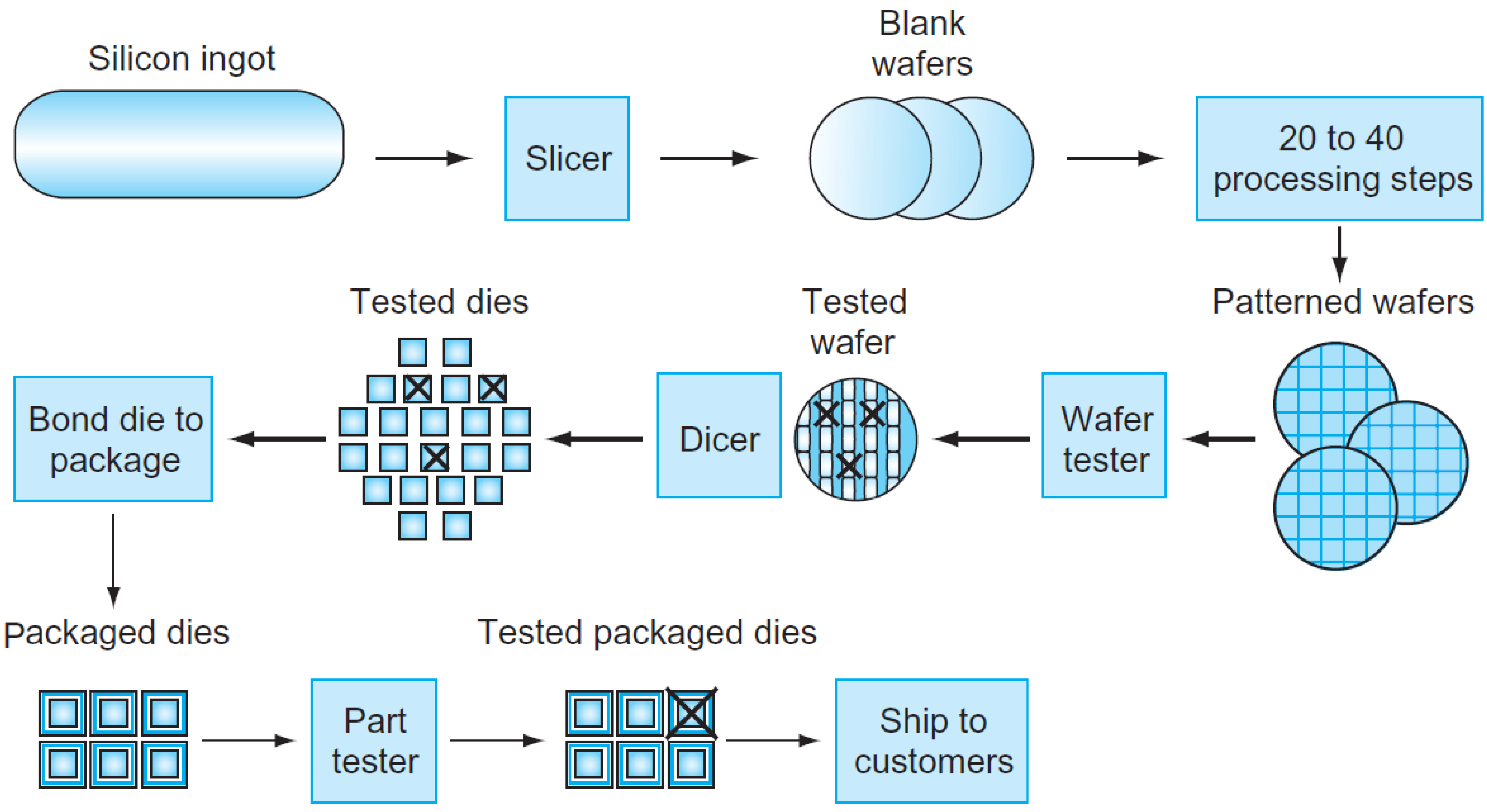
- Later generations: IC and VLSI
- Increasing price/performance
- Moore's law

Table 1.2 Computer Generations

Generation	Approximate Dates	Technology	Typical Speed (operations per second)
1	1946–1957	Vacuum tube	40,000
2	1957–1964	Transistor	200,000
3	1965–1971	Small- and medium-scale integration	1,000,000
4	1972–1977	Large scale integration	10,000,000
5	1978–1991	Very large scale integration	100,000,000
6	1991–	Ultra large scale integration	>1,000,000,000

W.Stallings, COA, 10th edition

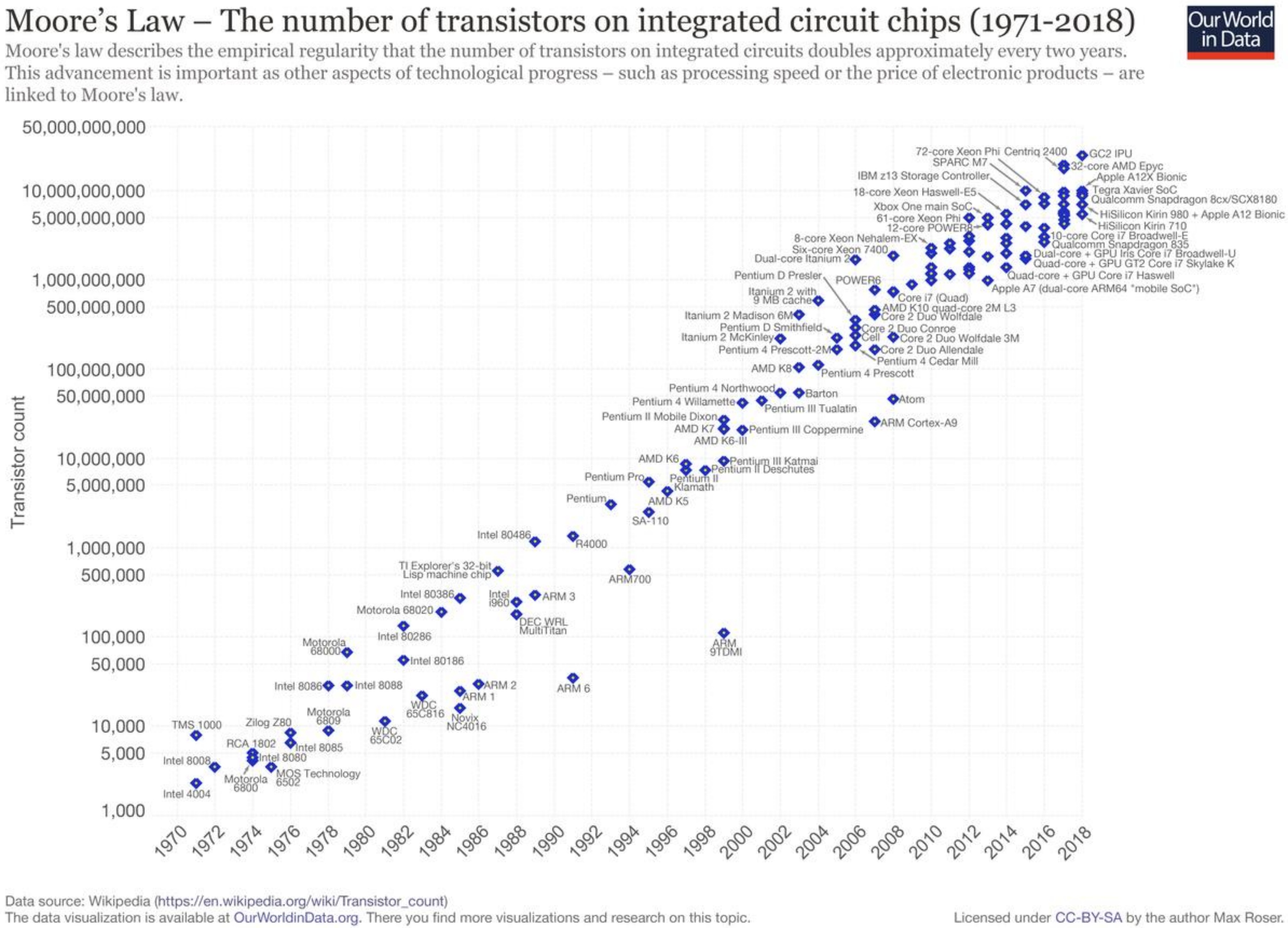
Key to computer evolution: IC making technology



The chip manufacturing process

Video: How an IC is made

Moore’s Law



How do we benefit from this?

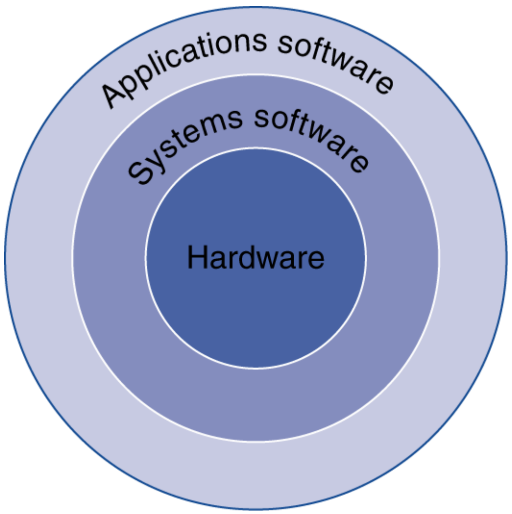
Key to computer evolution: IC making technology

- Electronics technology continues to evolve
  - Increased capacity and performance
  - Reduced cost

Year	Name	Size (cu. ft.)	Power (watts)	Performance (adds/sec)	Memory (KB)	Price	Price/ performance vs. UNIVAC	Adjusted price (2007 $)	Adjusted price/ performance vs. UNIVAC
1951	UNIVAC I	1,000	125,000	2,000	48	$1,000,000	1	$7,670,724	1
1964	IBM S/360 model 50	60	10,000	500,000	64	$1,000,000	263	$6,018,798	319
1965	PDP-8	8	500	330,000	4	$16,000	10,855	$94,685	13,367
1976	Cray-1	58	60,000	166,000,000	32,000	$4,000,000	21,842	$13,509,798	47,127
1981	IBM PC	1	150	240,000	256	$3,000	42,105	$6,859	134,208
1991	HP 9000/ model 750	2	500	50,000,000	16,384	$7,400	3,556,188	$11,807	16,241,889
1996	Intel PPro PC (200 MHz)	2	500	400,000,000	16,384	$4,400	47,846,890	$6,211	247,021,234
2003	Intel Pentium 4 PC (3.0 GHz)	2	500	6,000,000,000	262,144	$1,600	1,875,000,000	$2,009	11,451,750,000
2007	AMD Barcelona PC (2.5 GHz)	2	250	20,000,000,000	2,097,152	$800	12,500,000,000	$800	95,884,051,042

[Textbook COD 5th edition]

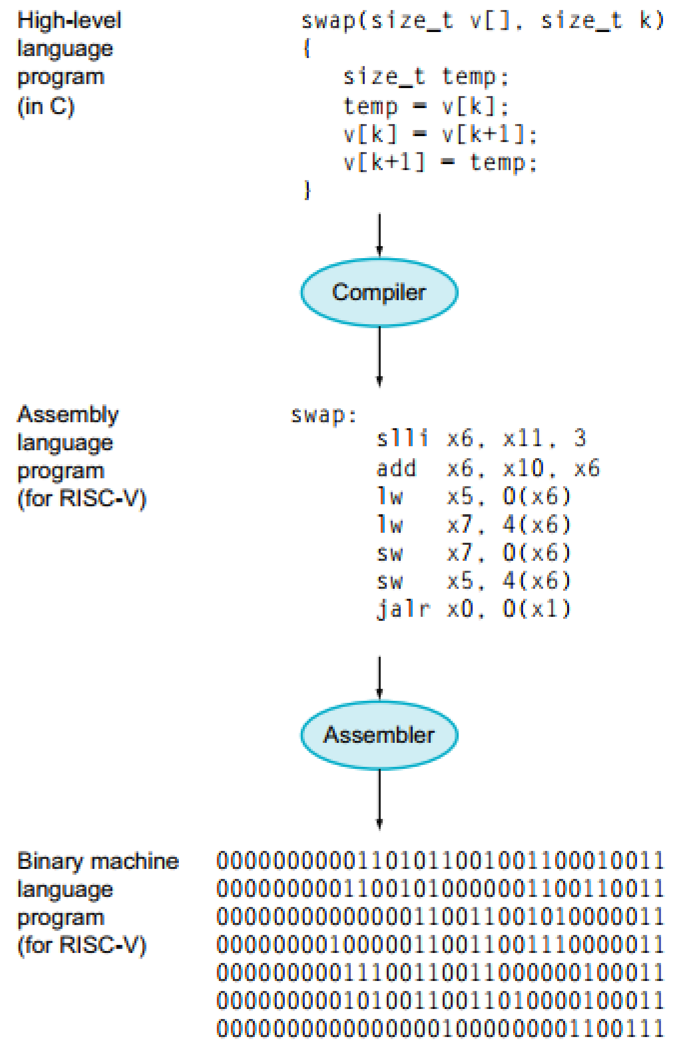
Hardware/software interface: abstraction



- Application software
  - Written in high-level language (HLL)
- 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



What's below your program?

- High-level language program (in C)

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

one-to-many

C compiler

- Assembly language program (for RISC-V CPU)

```
swap:  slli x6, x11, 3
       add x6, x10, x6
       lw x5, 0(x6)
       lw x7, 4(x6)
       sw x7, 0(x6)
       sw x5, 4(x6)
       jalr x0, 0(x1)
```

one-to-one

assembler

- Machine (object, binary) code (for RISC-V CPU)

```
00000000001101011001001100010011
00000000011001010000001100110011
00000000000000110011001010000011
. . .
```

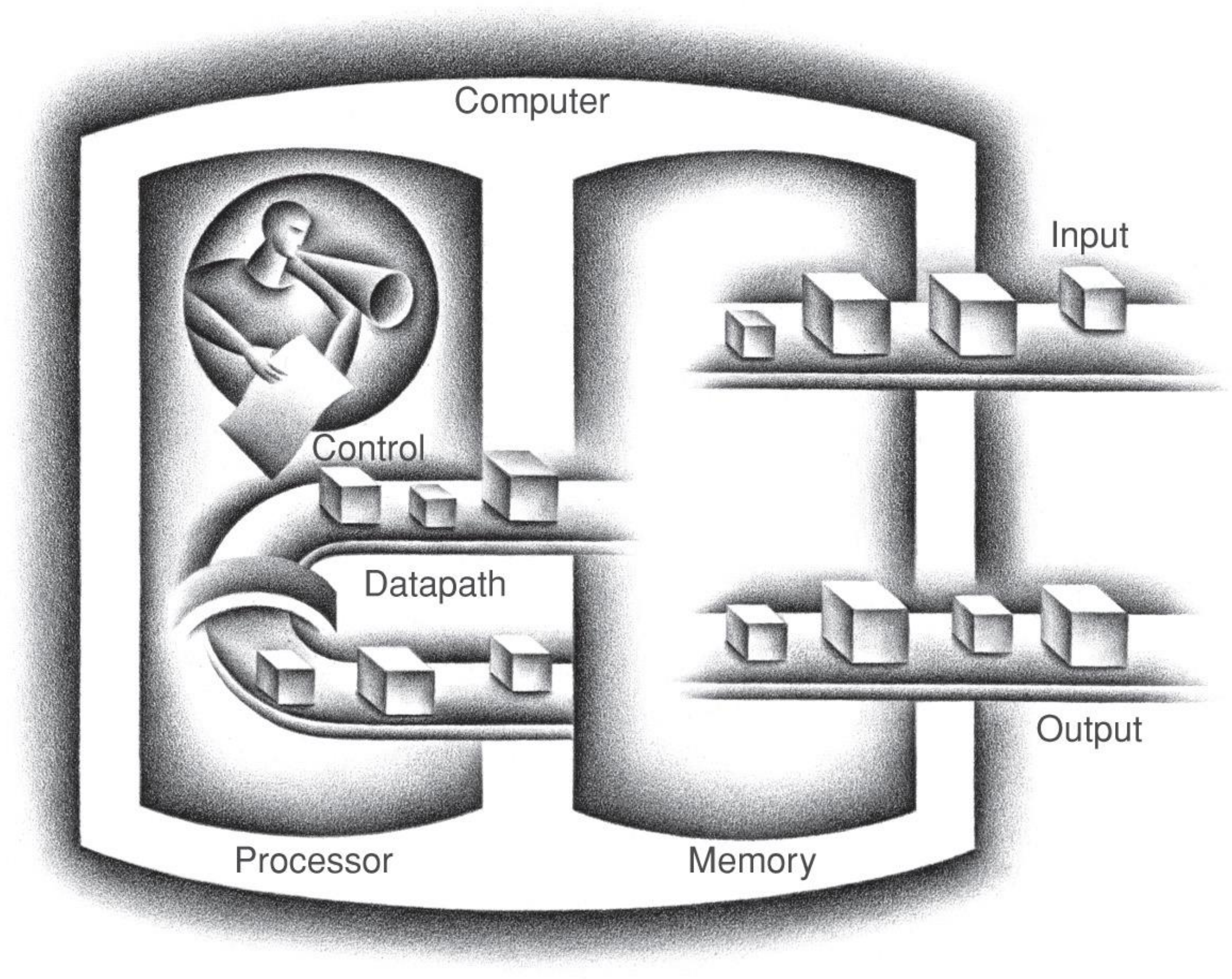
Computer Organization

- Computer's basic operation
  - Input data
  - Process data by executing stored program
  - Output data
- What are required components of computer?
  - For data input:
  - For storing information:
  - For program execution and data processing:
  - For data output:

Computer Organization

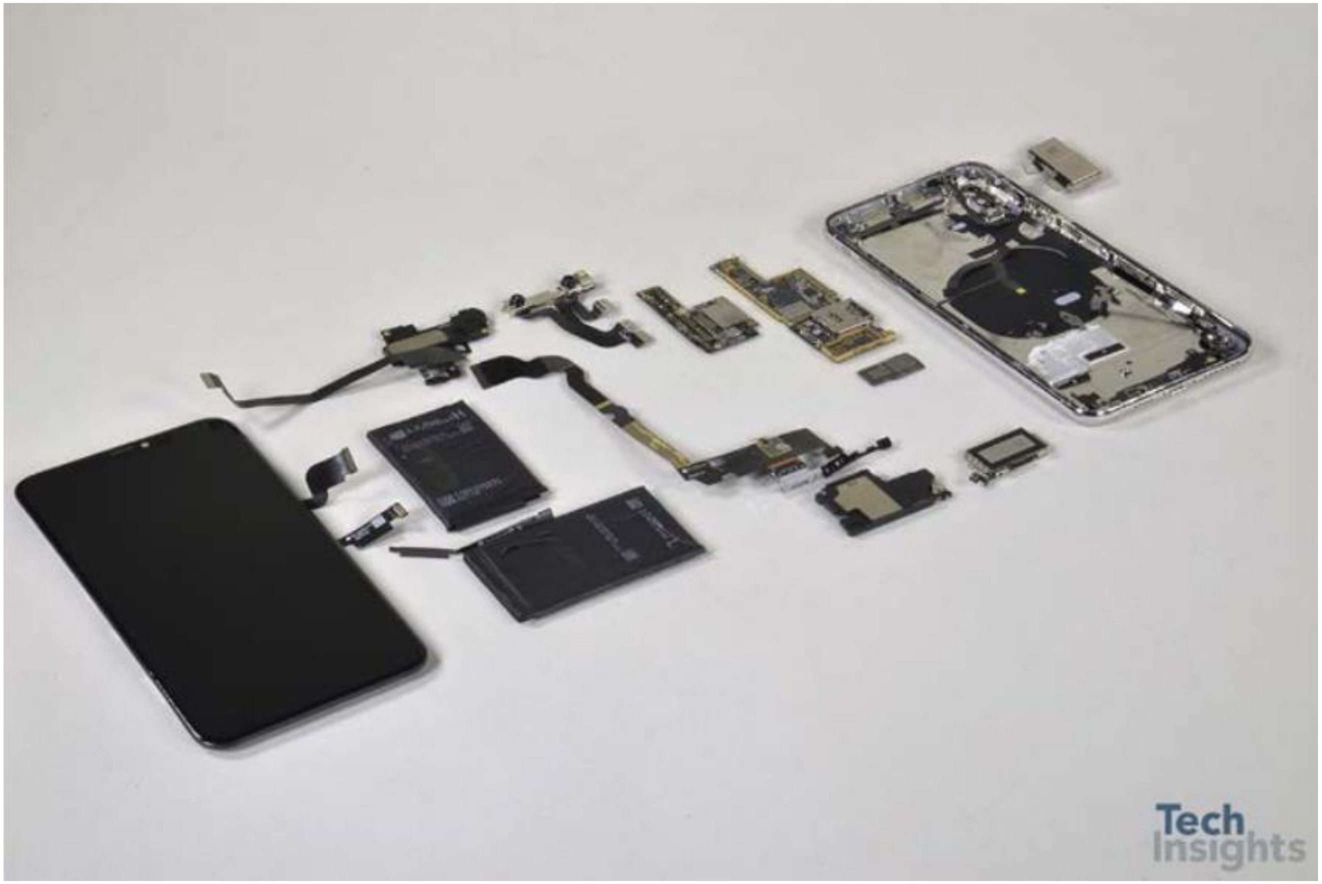
- Five classic components of a computer – input, output, memory, datapath, and control

- **datapath + control = processor (CPU)**



Computer Organization (in real-world products)

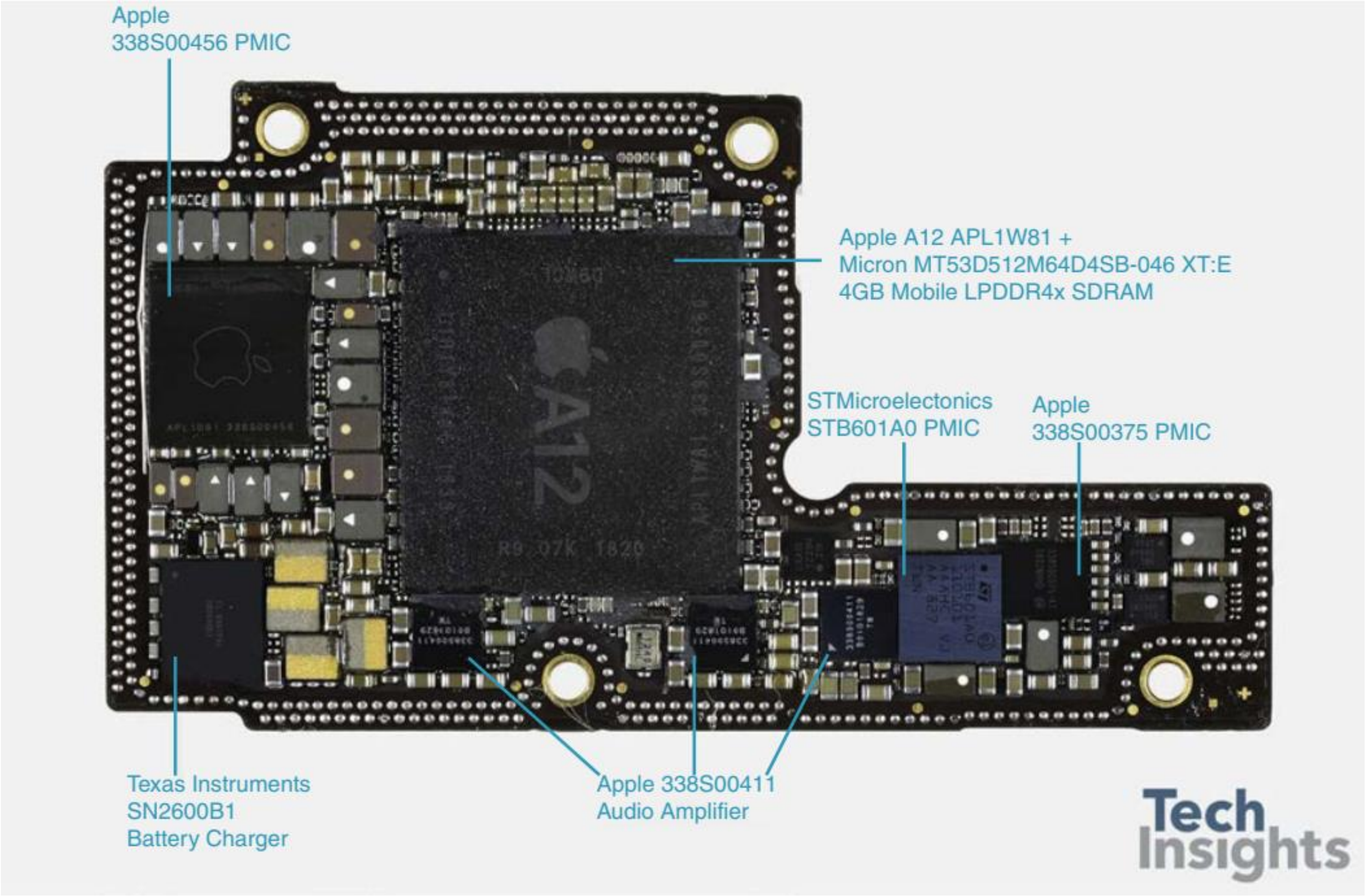
- Where are CPU, mem, I/O?



iPhone XS Max teardown

Computer Organization (in real-world products)

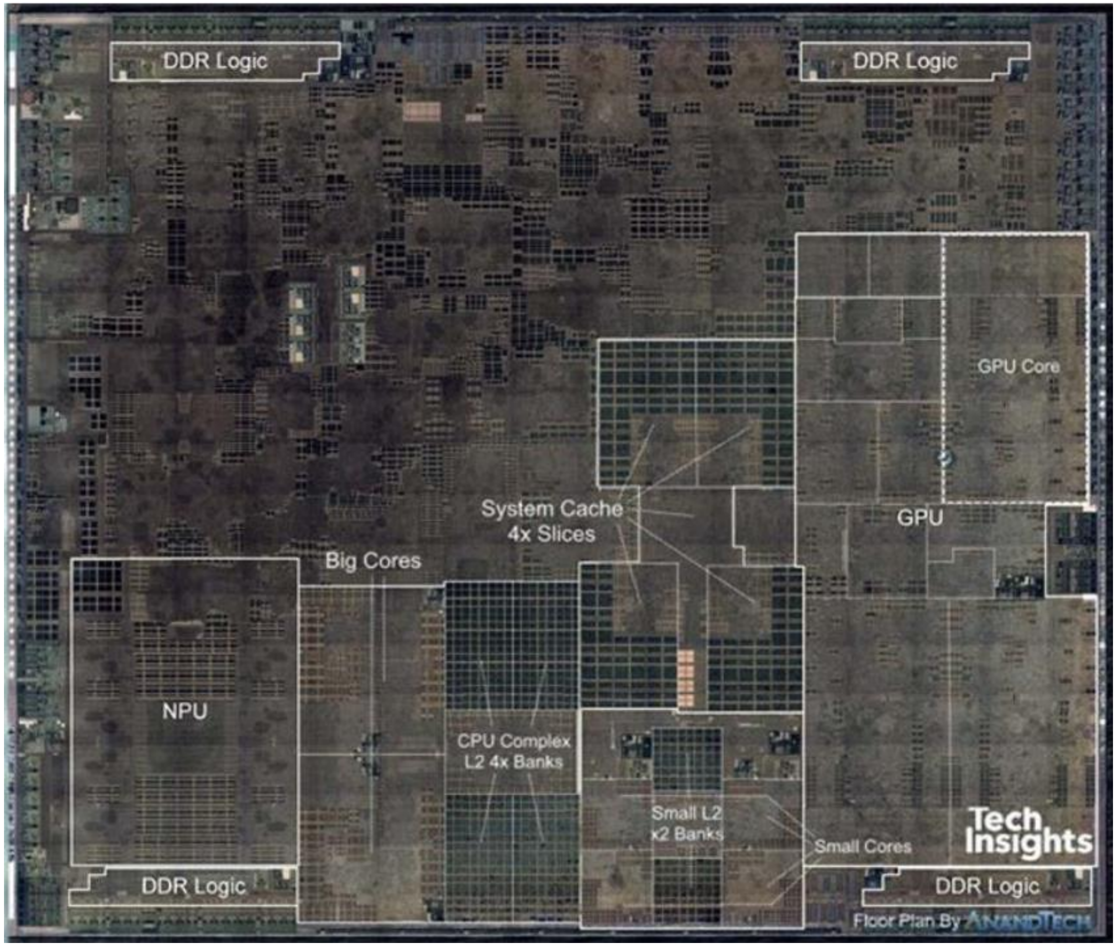
- Where are CPU, mem, I/O?



iPhone XS Max mainboard

Computer Organization (in real-world products)

- Is the CPU only a "CPU"?



Apple A12 package

2. Computer performance evaluation

- What is performance?
- A storage system
  - How much time to find a file/object?
  - How much time to transfer a file?
  - How many files can be served simultaneously?
- A web server
  - How fast a request can be served?
  - How many request can be served per second?
- Performance is multi-dimensional with many criteria
  - Throughput
  - **Response time**
  - …
- We focus on response time

2. Computer performance evaluation

- Response time:
  - System performance: elapsed time on unload system
  - **CPU performance**: user CPU time, the time that CPU actually spent on executing user program.
- To maximize performance, need to minimize execution time

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

If computer X is n times faster than Y, then

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

Relative Performance Example

- If computer A runs a program in 10 seconds and computer B runs the same program in 15 seconds, how much faster is A than B?

We know that A is n times faster than B if

$$\frac{\text{performance}_A}{\text{performance}_B} = \frac{\text{execution\_time}_B}{\text{execution\_time}_A} = n$$

The performance ratio is $$\frac{15}{10} = 1.5$$

Assume performance of B is 1, then performance of A is 1.5

Performance Factors

- CPU execution time (CPU time) – time the CPU spends working on a task
  - Does not include time waiting for I/O or running other programs

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

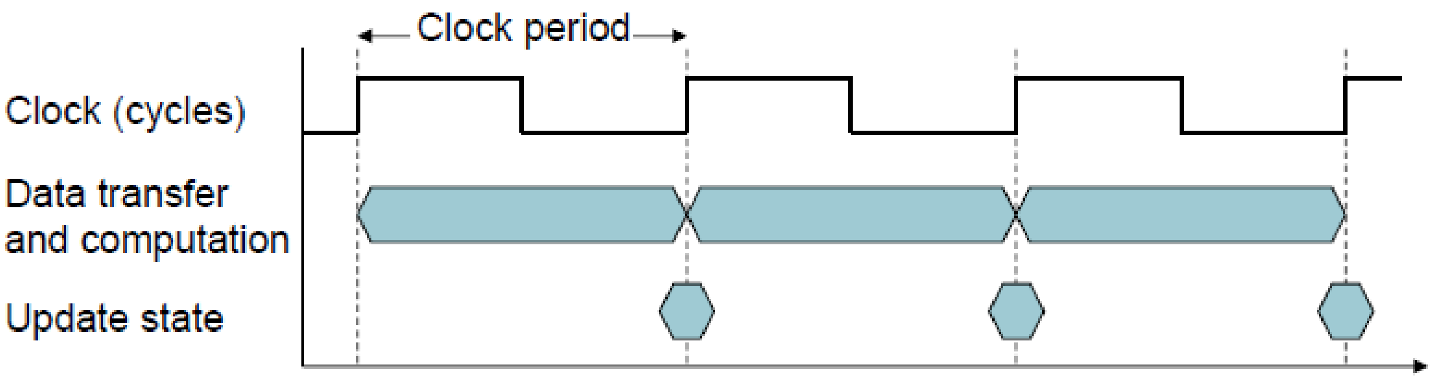
$$= \frac{\text{\# CPU clock cycles for a program}}{\text{clock rate}}$$

- Can improve performance by reducing either the length of the clock cycle or the number of clock cycles required for a program

Review: Machine Clock Rate

- Clock rate (clock cycles per second in MHz or GHz) is inverse of clock cycle time (clock period)

$$CC \;=\; 1 / CR$$



1 nsec (10^{-9}) clock cycle => 1 GHz (10^{9}) clock rate

500 psec clock cycle => 2 GHz clock rate

250 psec clock cycle => 4 GHz clock rate

200 psec clock cycle => 5 GHz clock rate

Improving Performance Example

- A program runs on computer A with a 2 GHz clock in 10 seconds. What clock rate must computer B run at to run this program in 6 seconds? Assume that, computer B will require 1.2 times as many clock cycles as computer A to run the program.

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

$$\textbf{CPU clock cycles}_A = \textbf{10 sec x 2 x } 10^9 \textbf{ cycles/sec}$$
$$= \textbf{20 x } 10^9 \textbf{ cycles}$$

$$\textbf{CPU time}_B = \frac{\textbf{1.2 x 20 x } 10^9 \textbf{ cycles}}{\textbf{clock rate}_B}$$

$$\textbf{clock rate}_B = \frac{\textbf{1.2 x 20 x } 10^9 \textbf{ cycles}}{\textbf{6 seconds}} = \textbf{4 GHz}$$

Clock Cycles per Instruction

- Not all instructions take the same amount of time to execute
  - Average execution time ~ average clock cycles per instruction

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

- **Clock cycles per instruction (CPI) – the average number of clock cycles each instruction takes to execute**
  - **A way to compare two different implementations of the same ISA**

	CPI for this instruction class		
	A	B	C
CPI	1	2	3

Using the Performance Equation

- Computers A and B implement the same ISA. Computer A has a clock cycle time of 250 ps and an effective CPI of 2.0 for some program and computer B has a clock cycle time of 500 ps and an effective CPI of 1.2 for the same program. Which computer is faster and by how much?

Each computer executes the same number of instructions, *I*, so

$$\textbf{CPU time}_A = I \times 2.0 \times 250 \text{ ps} = 500 \times I \text{ ps}$$

$$\textbf{CPU time}_B = I \times 1.2 \times 500 \text{ ps} = 600 \times I \text{ ps}$$

Clearly, A is faster … by the ratio of execution times

$$\frac{\text{performance}_A}{\text{performance}_B} = \frac{\text{execution\_time}_B}{\text{execution\_time}_A} = \frac{600 \times I \text{ ps}}{500 \times I \text{ ps}} = 1.2$$

The Performance Equation

- Our basic performance equation is then calculated

$$\text{CPU time} = \text{Instruction\_count} \times \text{CPI} \times \text{clock\_cycle}$$

$$= \frac{\text{Instruction\_count} \times \text{CPI}}{\text{clock\_rate}}$$

- Key factors that affect performance (CPU execution time)
  - The clock rate: CPU specification
  - CPI: varies by instruction type and ISA implementation
  - Instruction count: measure by using profilers/ simulators

Dynamic Instruction Count

How many instructions are executed in this program fragment?

Each "for" consists of two instructions: increment index, check exit condition

```
250 instructions
for i = 1, 100 do
20 instructions
   for j = 1, 100 do
   40 instructions
      for k = 1, 100 do
      10 instructions
      endfor
   endfor
endfor
```

Static count = 326

12,422,450 Instructions

2 + 20 + 124,200 instructions
100 iterations
12,422,200 instructions in all

2 + 40 + 1200 instructions
100 iterations
124,200 instructions in all

2 + 10 instructions
100 iterations
1200 instructions in all

for i = 1, n
while x > 0

Improving performance by CPI

Op	Freq	CPI_i	Freq x CPI_i
ALU	50%	1	
Load	20%	5	
Store	10%	3	
Branch	20%	2	
$Avg\ CPI = \sum freq_i * CPIi$			=

- How much faster would the machine be if a better data cache reduced the average load time to 2 cycles?
- What if branch instruction is only one cycle?
- What if two ALU instructions could be executed at once?

Improving performance by CPI

Op	Freq	CPI_i	Freq x CPI_i			
ALU	50%	1	.5	.5	.5	.25
Load	20%	5	1.0	.4	1.0	1.0
Store	10%	3	.3	.3	.3	.3
Branch	20%	2	.4	.4	.2	.4
$Avg\ CPI = \sum freq_i * CPIi$			= 2.2	1.6	2.0	1.95

- How much faster would the machine be if a better data cache reduced the average load time to 2 cycles?

  CPU time new = 1.6 x IC x CC so 2.2/1.6 means 37.5% faster

- What if branch instruction is only one cycle?

  CPU time new = 2.0 x IC x CC so 2.2/2.0 means 10% faster

- What if two ALU instructions could be executed at once?

  CPU time new = 1.95 x IC x CC so 2.2/1.95 means 12.8% faster

How to improve performance?

- Shorter clock cycle = faster clock rate

  → latest CPU technology

- Smaller CPI

  → optimizing Instruction Set Architecture

- Smaller instruction count

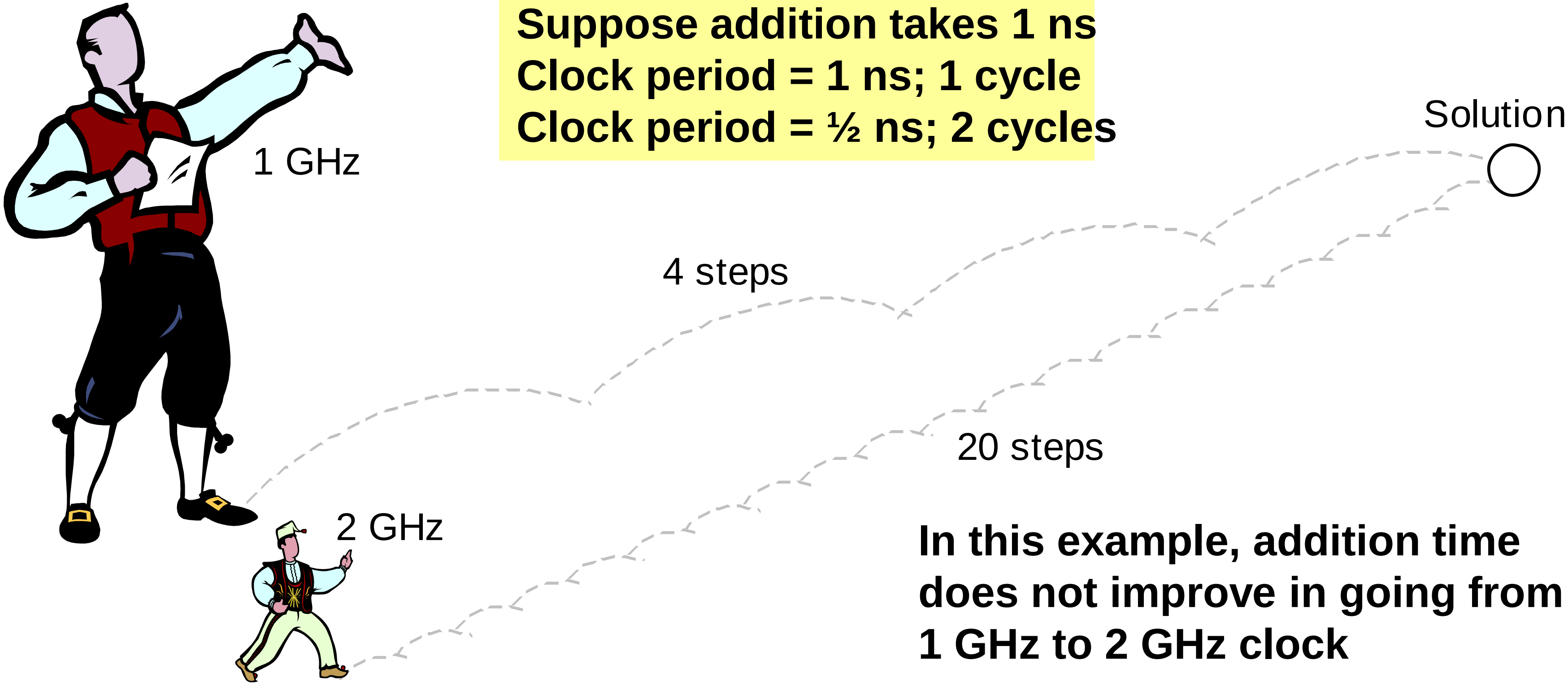
  → optimizing algorithm and compiler

- To get best performance, multiple criteria are combined and considered at design time

→ specific CPU for specific class computation problem

Faster Clock ≠ Shorter Running Time

Suppose addition takes 1 ns
Clock period = 1 ns; 1 cycle
Clock period = ½ ns; 2 cycles



In this example, addition time does not improve in going from 1 GHz to 2 GHz clock

Faster steps do not necessarily mean shorter travel time.

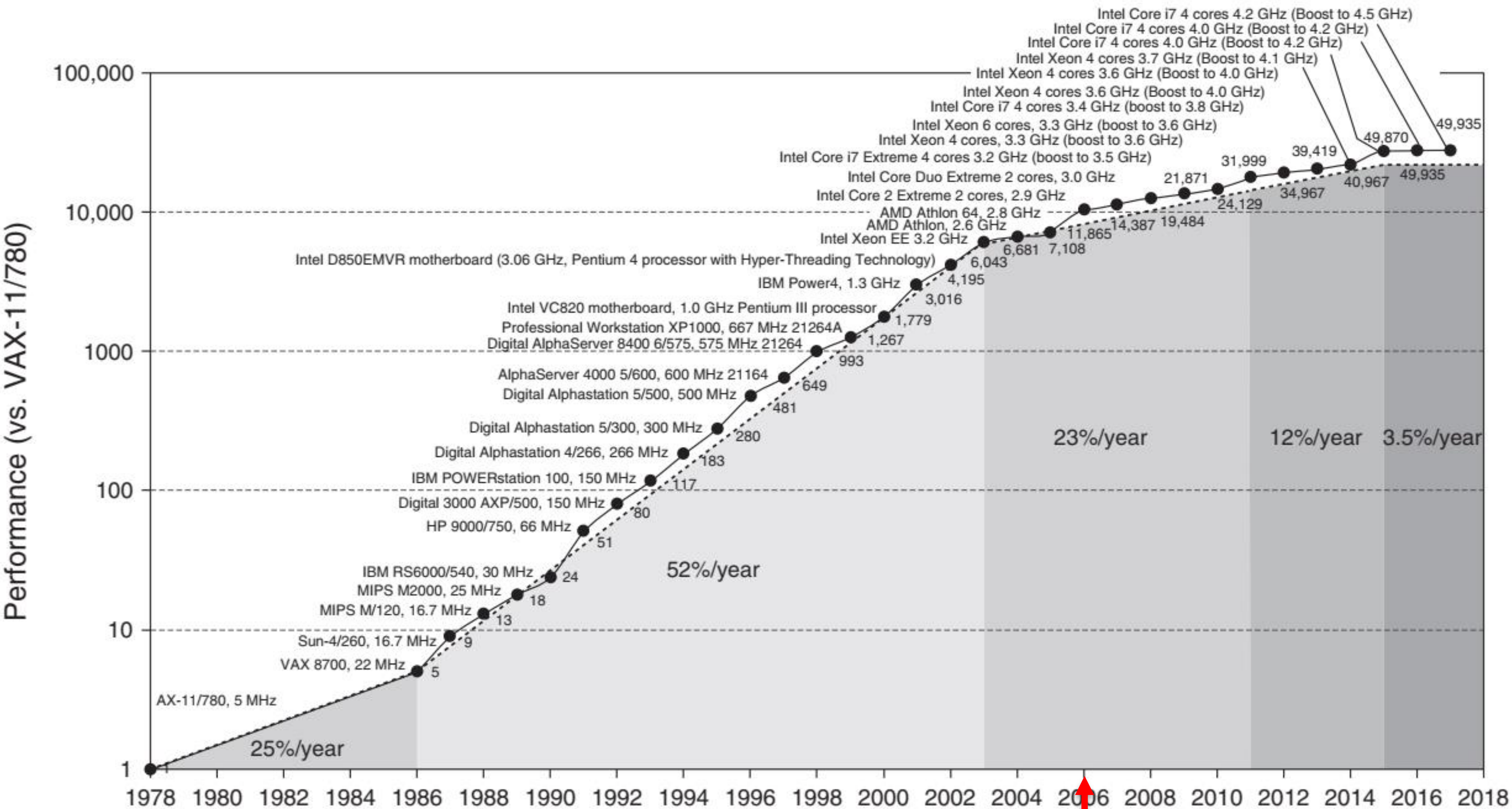
Measuring/benchmarking PC performance

- SPEC CPU benchmark
  - Started in 1989
  - SPEC CPU2006: 12 integer, 17 floating point benchmarks
  - Reference machine: Sun Ultra Enterprise 2 (1997) running on a 296 MHz UltraSPARC II CPU.

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

FIGURE 1.18 SPECINTC2006 benchmarks running on a 2.66 GHz Intel Core i7 920.

(Uni)processor performance since 1908



Reference machine: VAX 11/780 (5MHz)

Change to multi-processors

End of chapter 1