Computer Architecture

Ngo Lam Trung
Department of Computer Engineering
School of Information and Communication Technology (SoICT)
Hanoi University of Science and Technology
E-mail: trungnl@soict.hust.edu.vn

Course administration

- Instructor: Ngo Lam Trung
  803 B1, SoICT, HUST
- Text: [Required] Computer Organization and Design, 5th edition revised printing Patterson & Hennessy 2014.

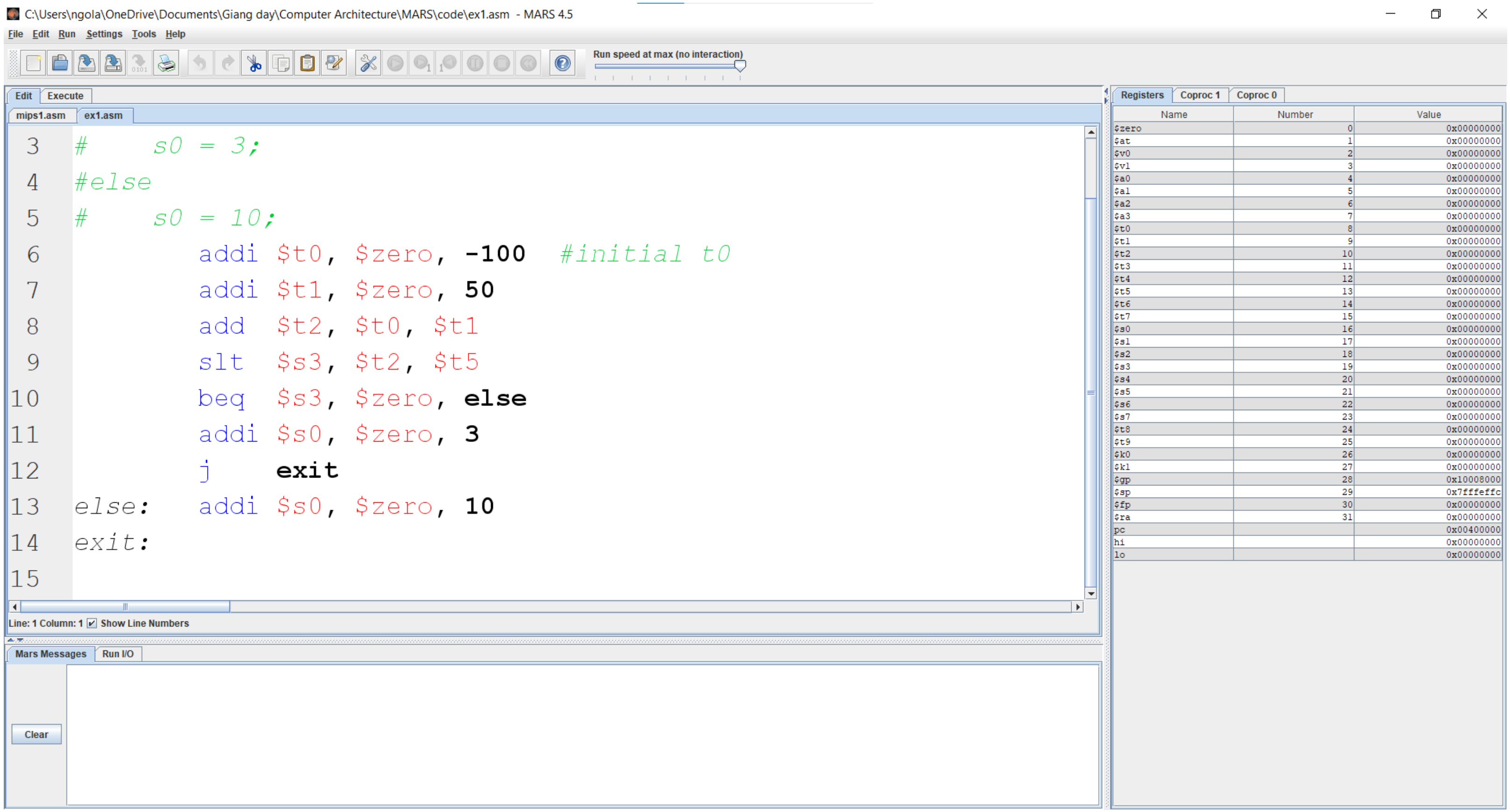
  [Optional] Computer Organization and Architecture, 10th Edition, William Stalling
- Slides: pdf
- Schedule: as in timetable

Course content

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

Homework/exercises

- MIPS assembly programming
- MARS simulator


C:\Users\ngola\OneDrive\Documents\Giang day\Computer Architecture\MARS\code\ex1.asm - MARS 4.5
File Edit Run Settings Tools Help
Run speed at max (no interaction)
Edit Execute
mips1.asm ex1.asm
Mars Messages Run I/O
Clear
Registers Coproc 1 Coproc 0

```
#     s0 = 3;
#else
#     s0 = 10;
        addi $t0, $zero, -100  #initial t0
        addi $t1, $zero, 50
        add  $t2, $t0, $t1
        slt  $s3, $t2, $t5
        beq  $s3, $zero, else
        addi $s0, $zero, 3
        j    exit
else:   addi $s0, $zero, 10
exit:
```

Line: 1 Column: 1 ☑ Show Line Numbers

Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x00000000
$v1	3	0x00000000
$a0	4	0x00000000
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000000
$t1	9	0x00000000
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00000000
pc		0x00400000
hi		0x00000000
lo		0x00000000

Computers are so important

- Current modern life
  - Modern automobiles
  - Cellphones
  - WWW
  - Search engines
- Future
  - Tailored medical care based on individual genome
  - Super-human: transfer human's brain (14 billion of neural) to a mechanical body (robot) for interstellar traveling

Outcomes from this course

- Understanding of how high-level language programs (C, Java…) translate into computer language programs, and how hardware execute the latter programs.

- Hardware/software interface, and how software instructs hardware to perform functions.

- The factors that determine computer performance.

- What techniques can be used to improve computer performance.

- Why computer moves from sequential to parallel processing.

Prerequisites - What You Should Already Knew

- What is computer
- Look and feel of computer
  - How computer parts look like?
- Basic logic design
  - Combinational, sequential circuit design
- Create, compile, and run C/C++ programs

Chapter 1: Introduction

1. Computer Abstraction and Technology
2. Performance Evaluation

[with materials from *Computer Organization and Design, 5th Edition*,
Patterson & Hennessy, ©2014, 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?

✓

✓

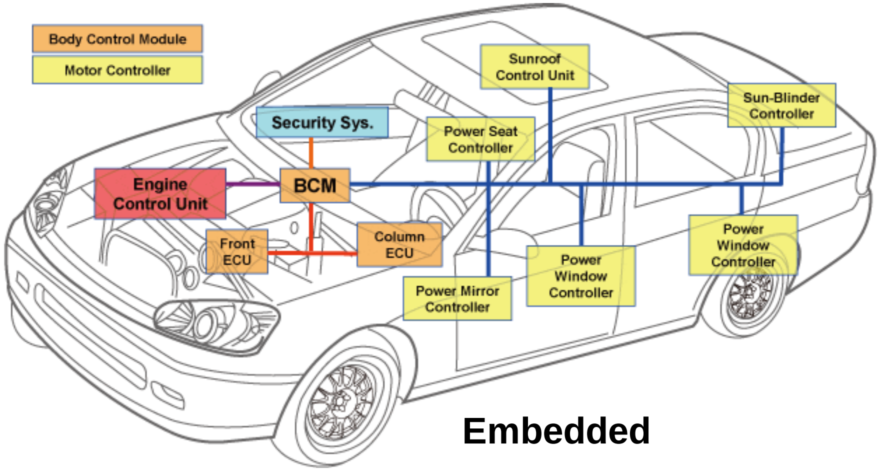
✓

✗

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



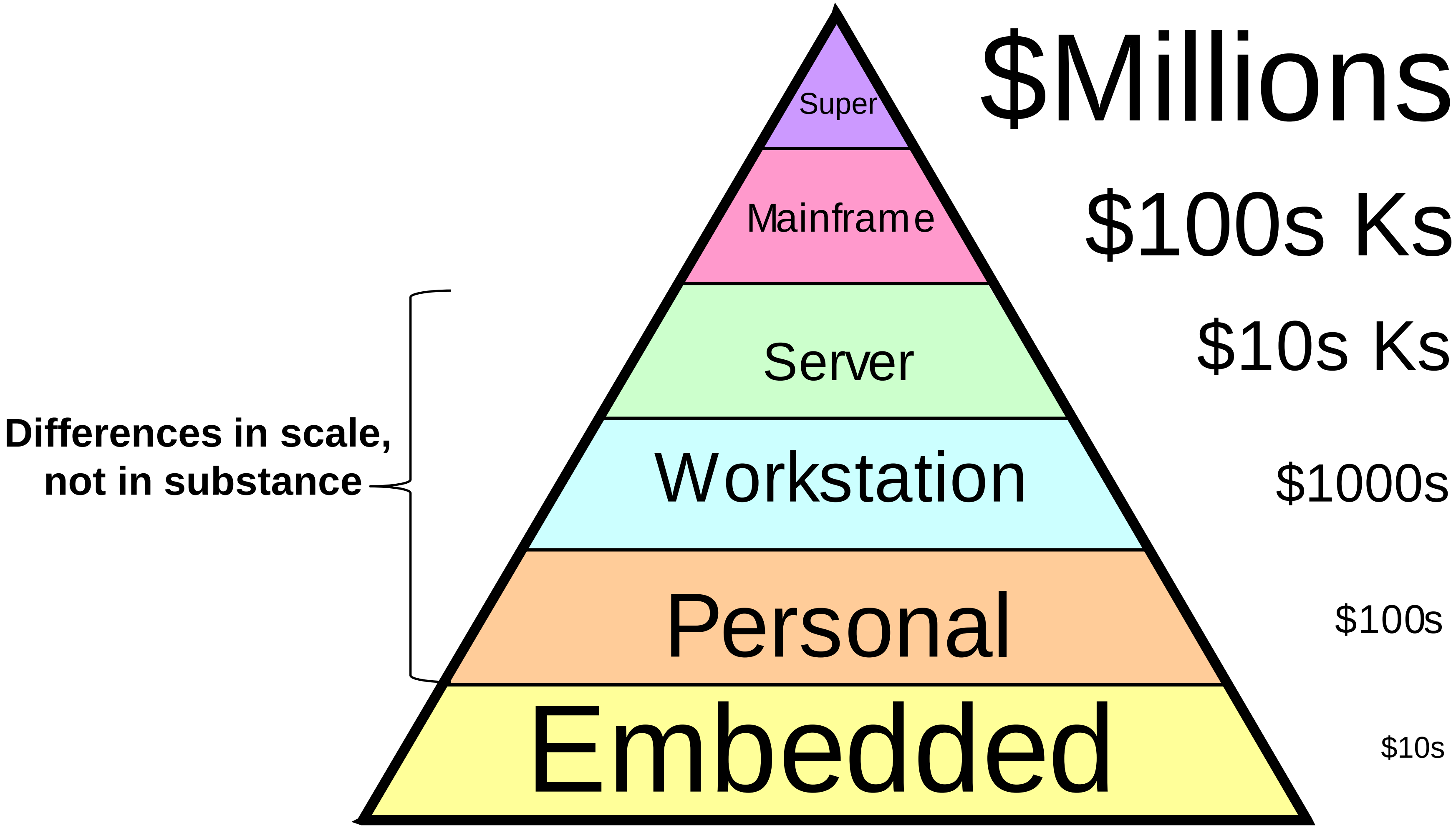
Embedded

PC

Server

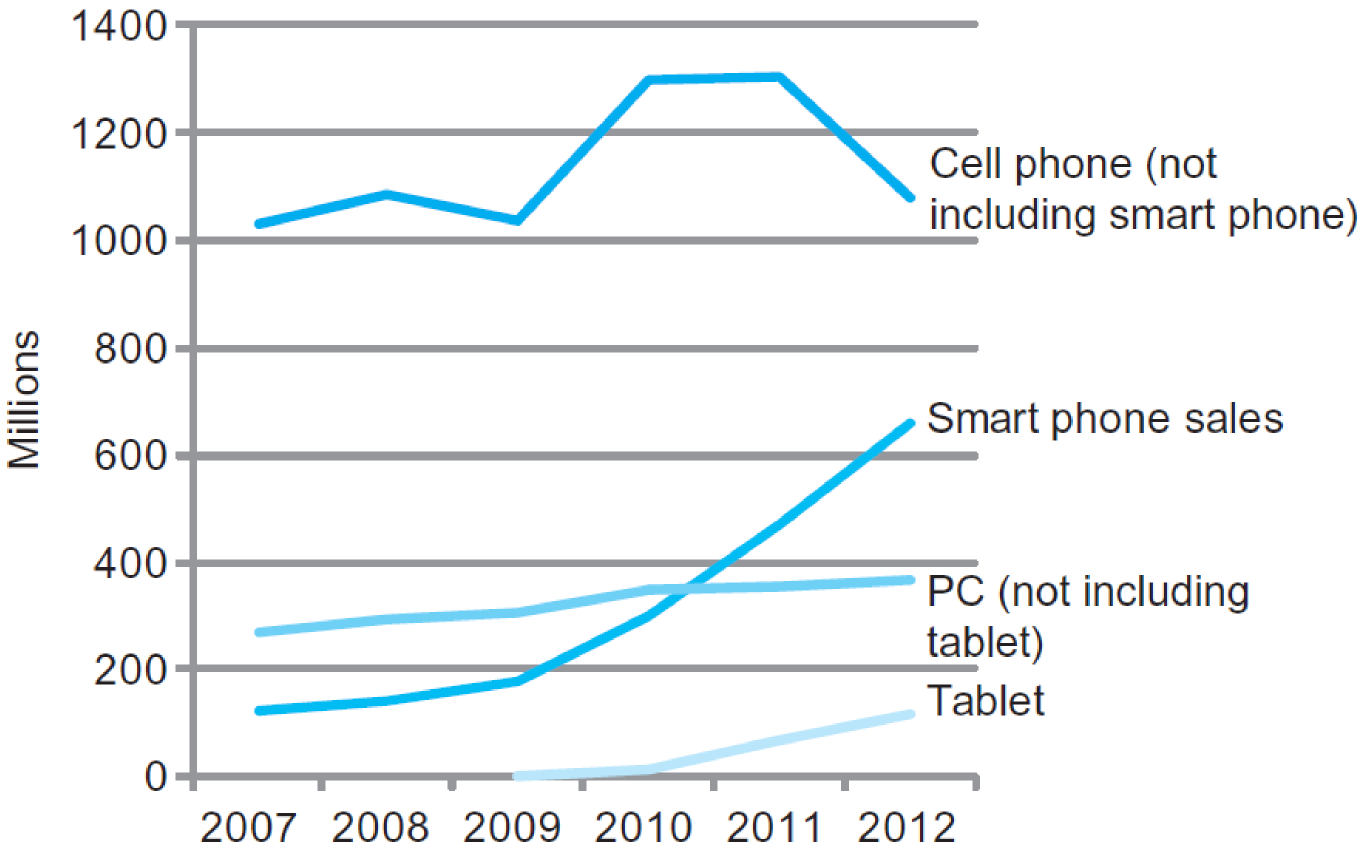
Super computer

Price/performance of computer classes



Post-PC era

- PDA, smart phone, tablet…
- Smart TV, set top box…



Eight important ideas

Design for Moore's law

Simplification via abstraction

Make common cases fast

Performance via Parallelism

Performance via Pipelining

Performance via Prediction

Memory hierarchy

Dependability via redundancy

What’s below your program?

- High-level language program (in C)

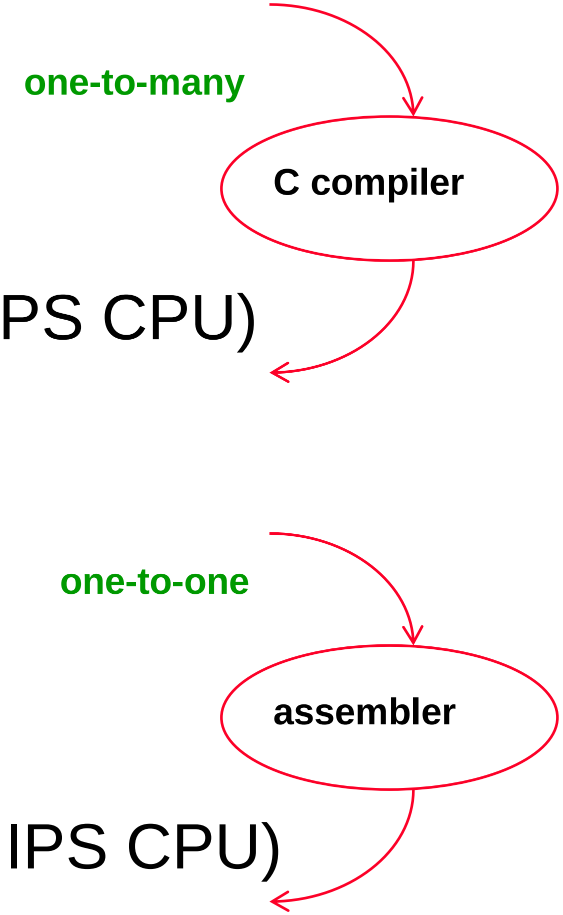
```
swap (int v[], int k)
{       int temp;
        temp = v[k];
        v[k] = v[k+1];
        v[k+1] = temp;
}
```

- Assembly language program (for MIPS CPU)

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

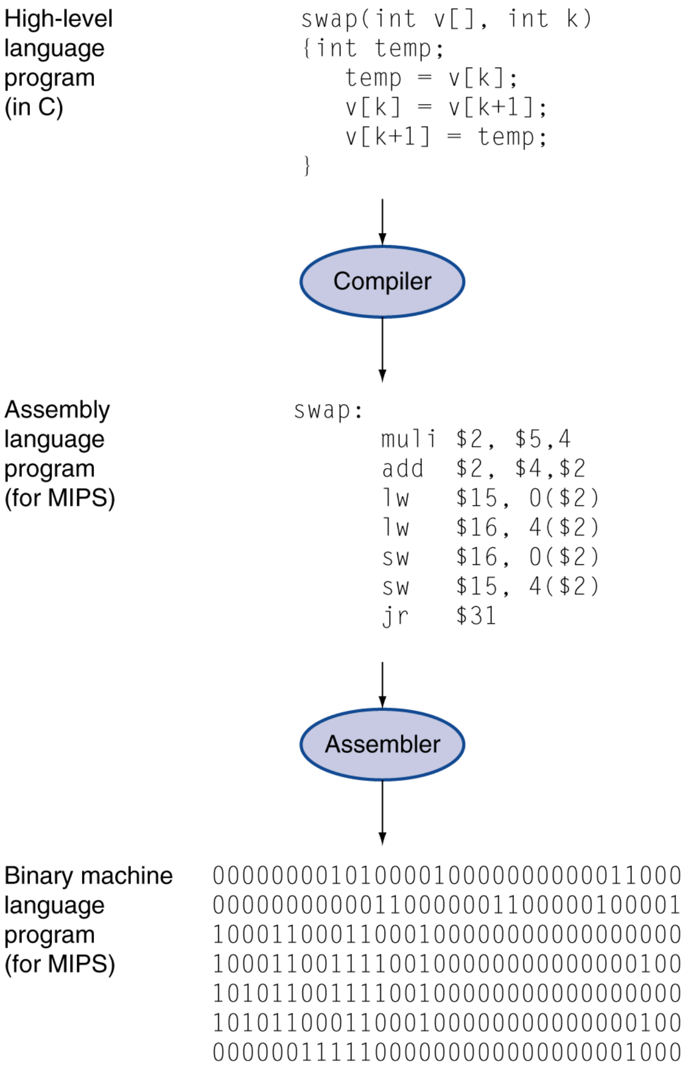
- Machine (object, binary) code (for MIPS CPU)

```
000000 00000 00101 0001000010000000
000000 00100 00010 0001000000100000
 . . .
```



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



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;
}
```

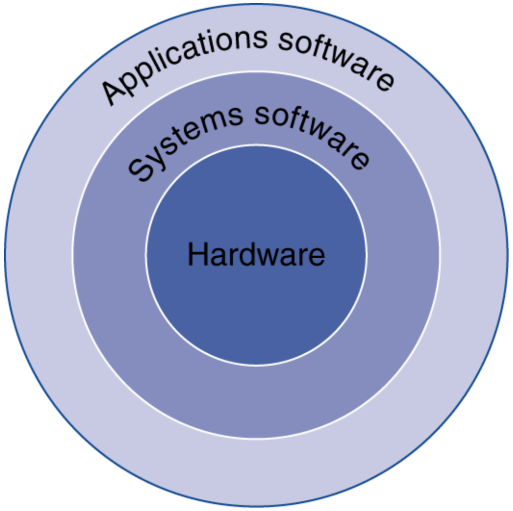
Assembly language program (for MIPS)

```
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
```

Binary machine language program (for MIPS)

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

Hardware/software interface: below your program



- 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

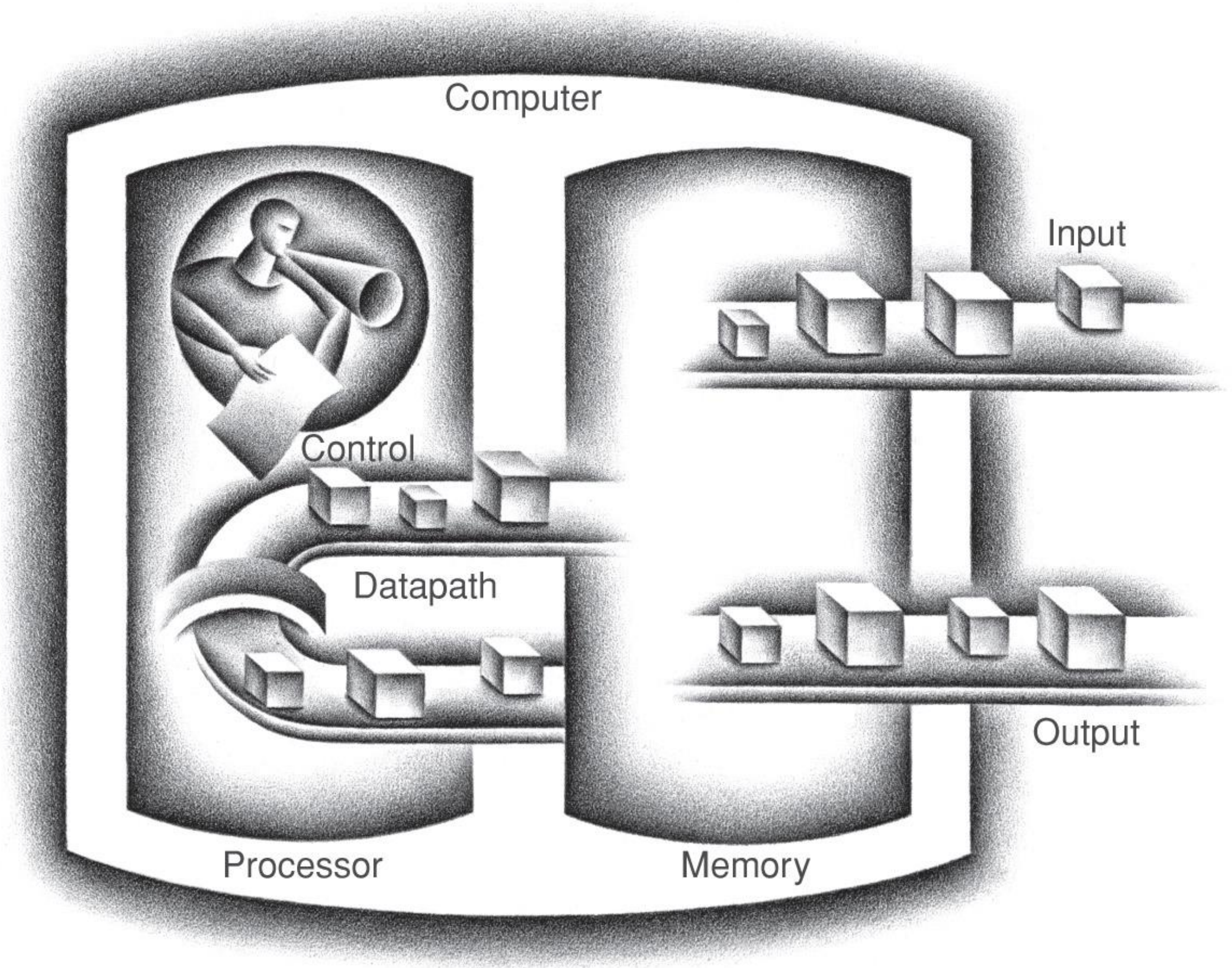
Computer Organization

- 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

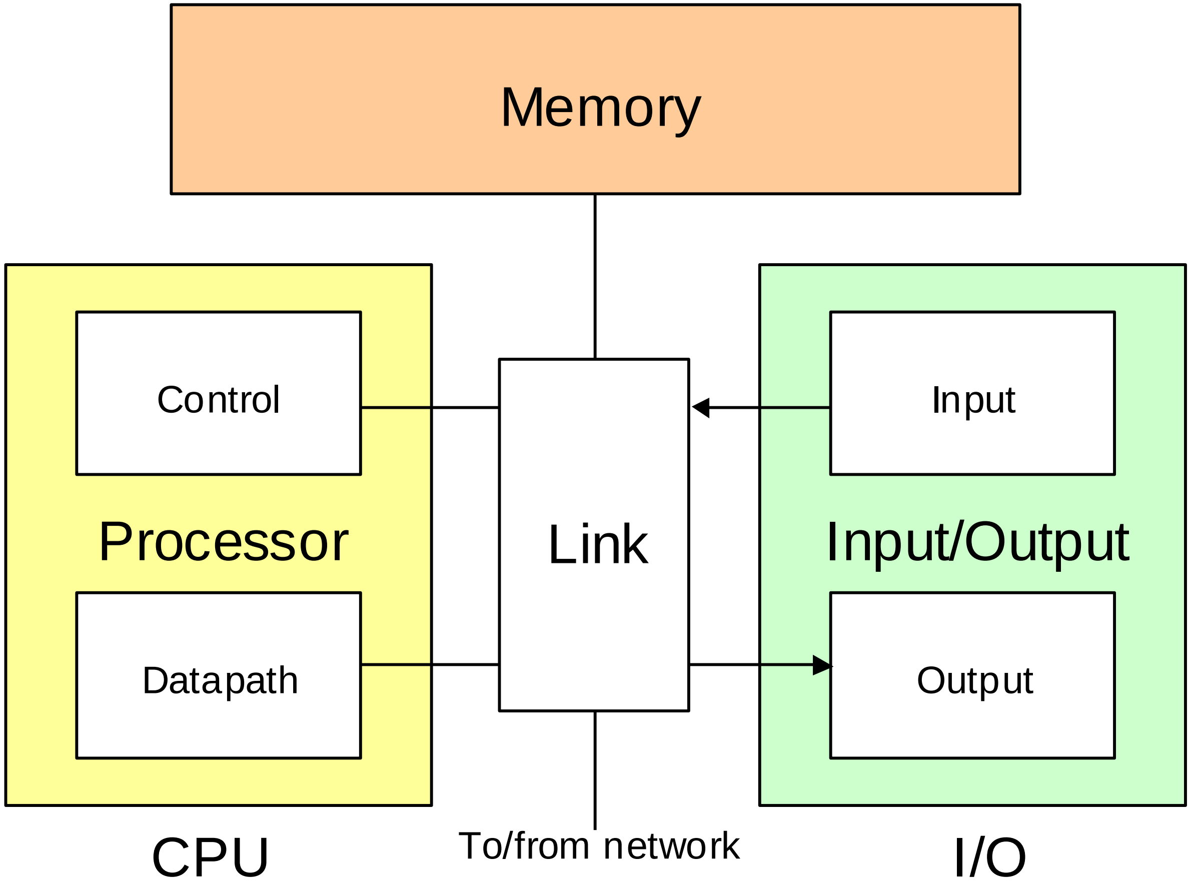
- Five classic components of a computer – input, output, memory, datapath, and control

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

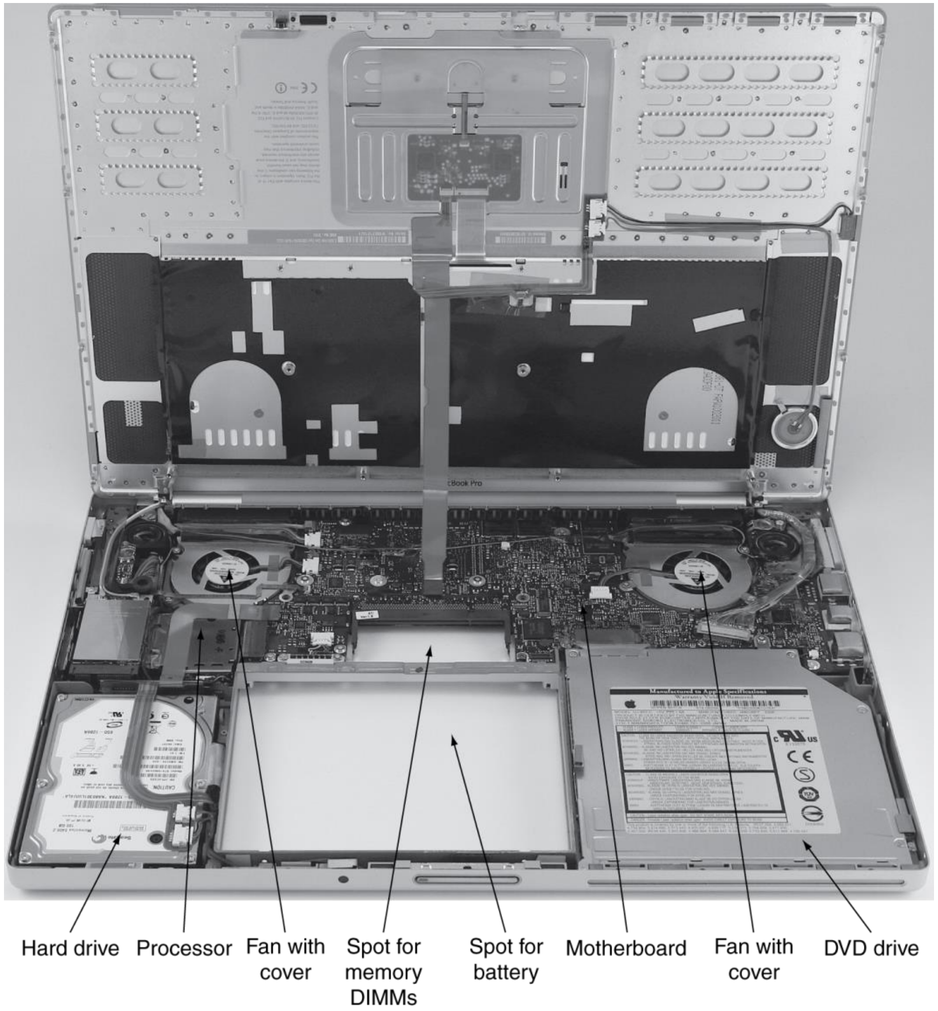


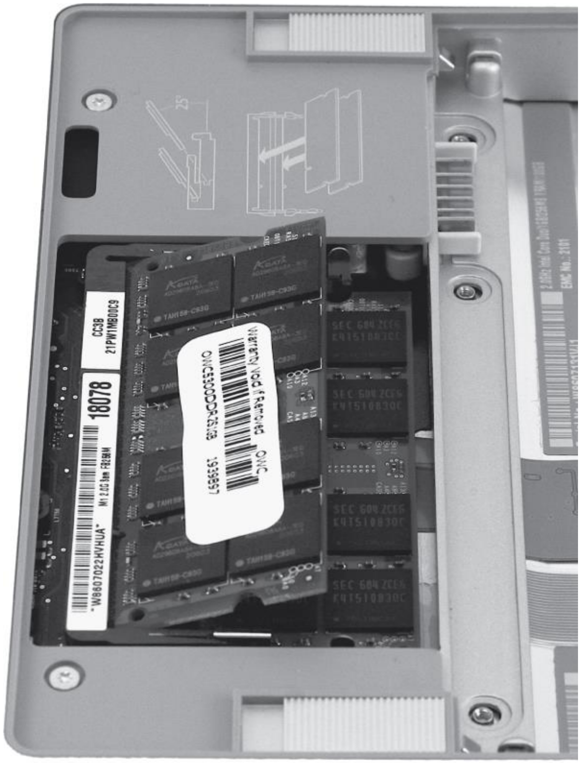
A similar view

- Usually, the Link unit is hidden

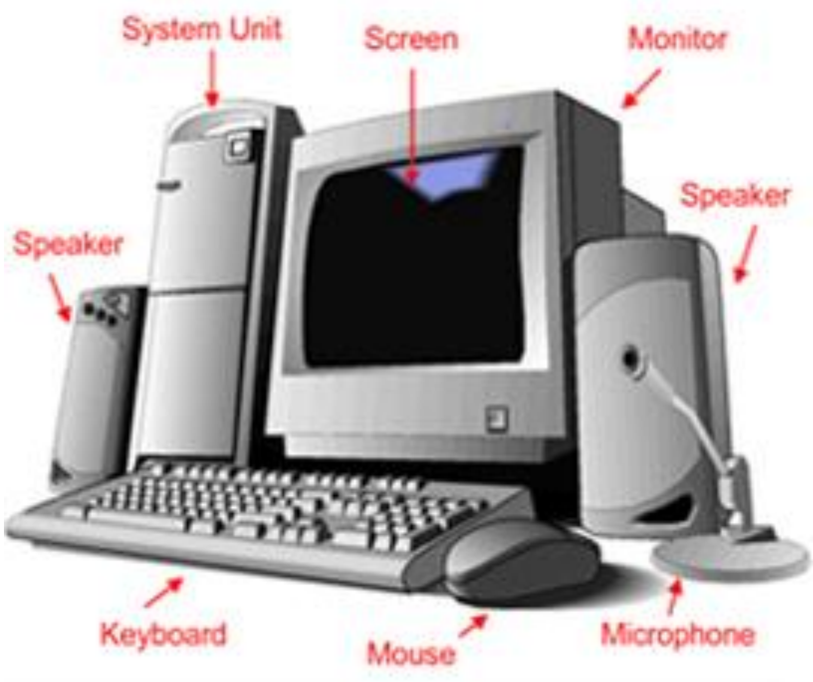


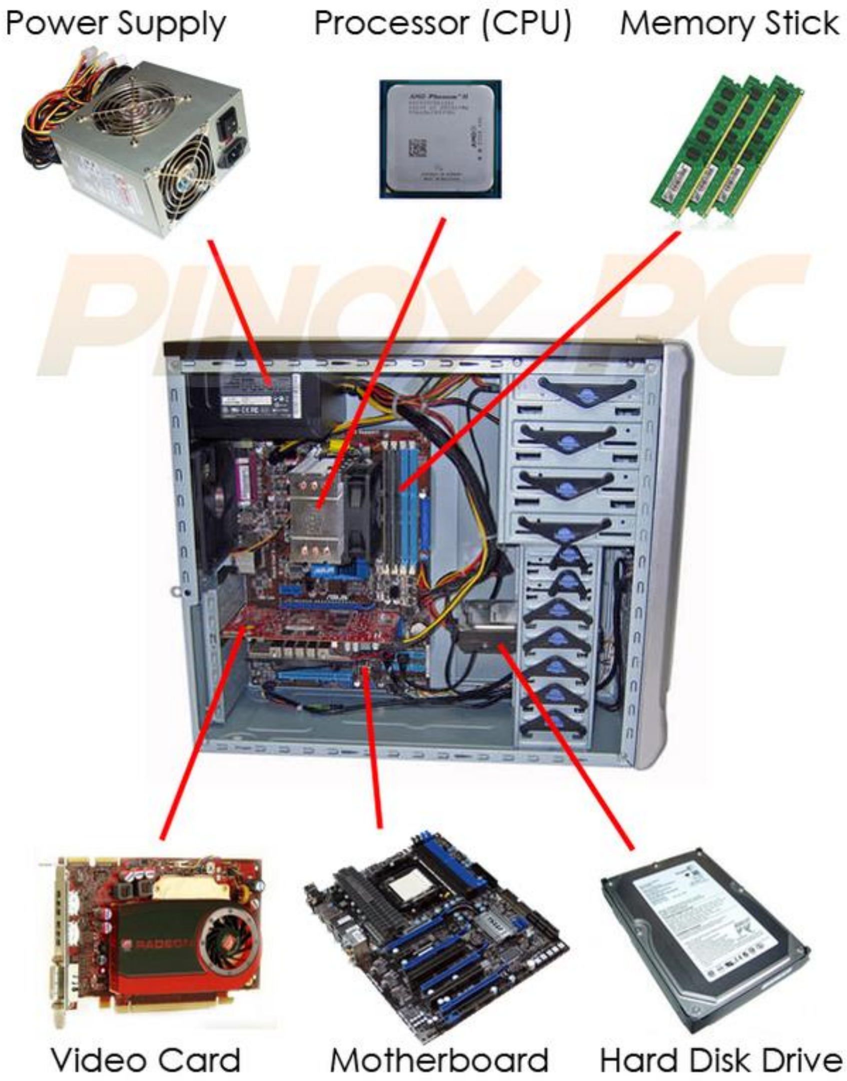
Opening the box: anatomy of computer





Opening the box: anatomy of computer





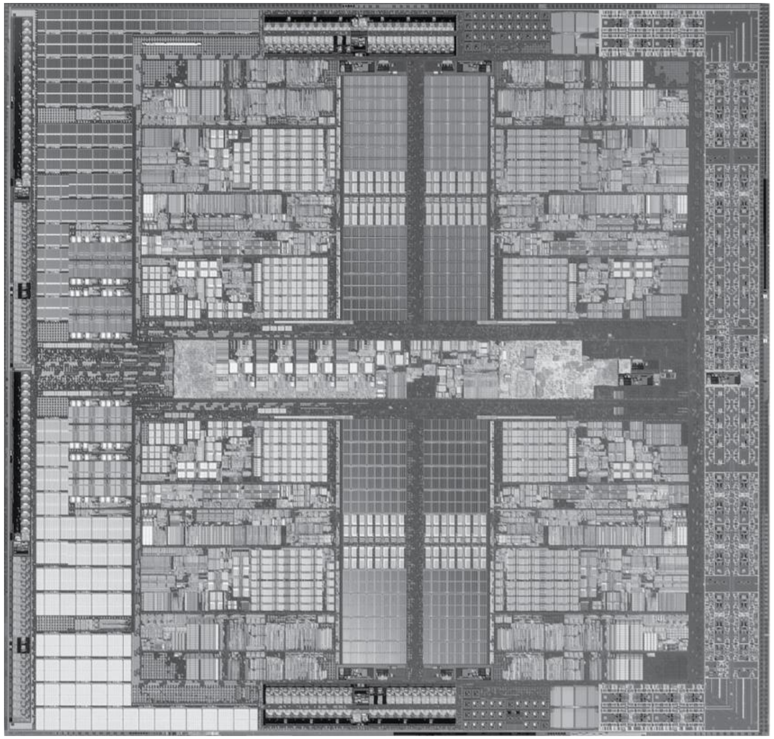
The story of each component worth a separate course!

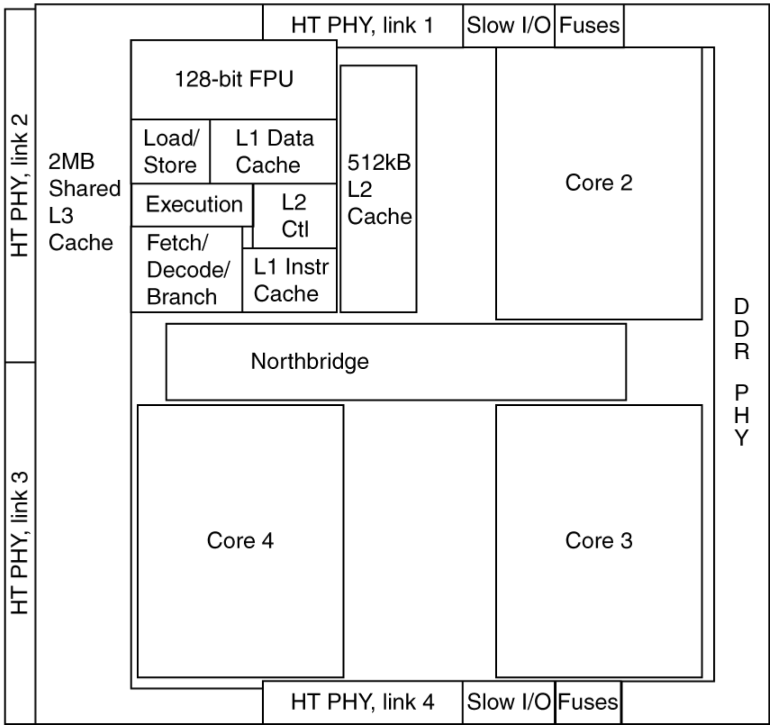
Inside the Processor (CPU)

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

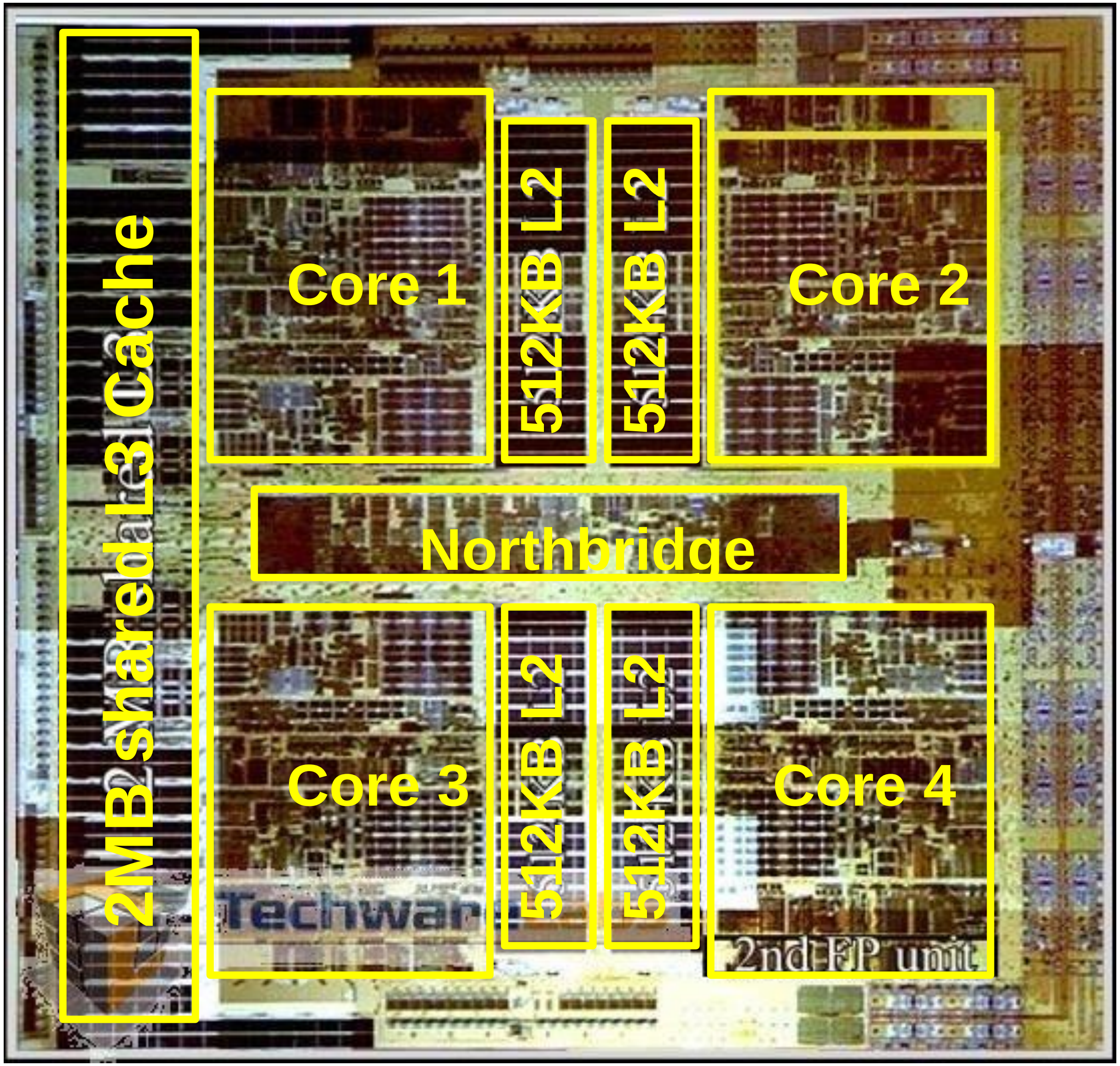
Inside the Processor

- AMD Barcelona: 4 processor cores





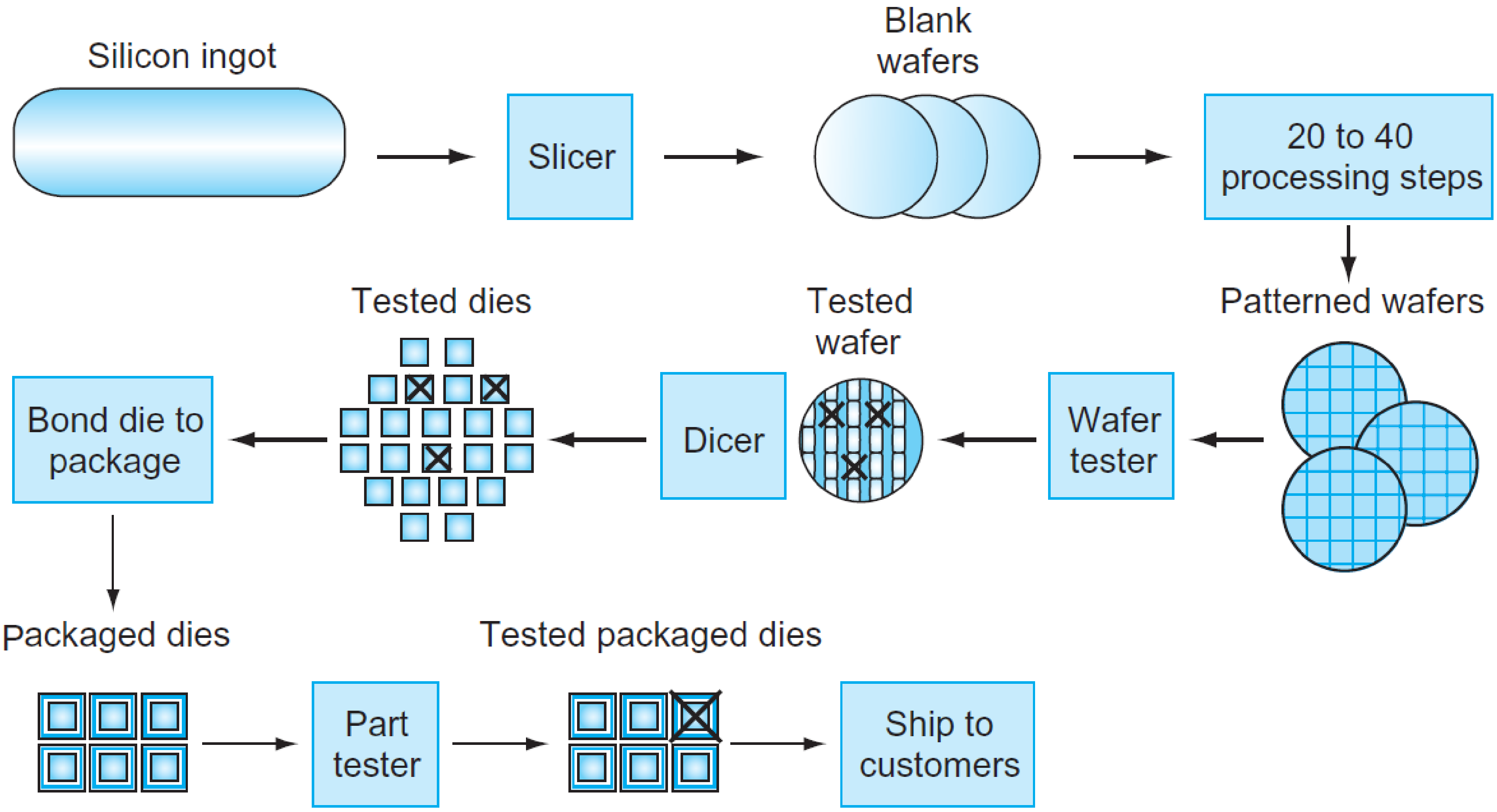
AMD's Barcelona Multicore Chip



- ❑ Four out-of-order cores on one chip
- ❑ 1.9 GHz clock rate
- ❑ 65nm technology
- ❑ Three levels of caches (L1, L2, L3) on chip
- ❑ Integrated Northbridge

http://www.techwarelabs.com/reviews/processors/barcelona/

Key to computer evolution: IC making technology



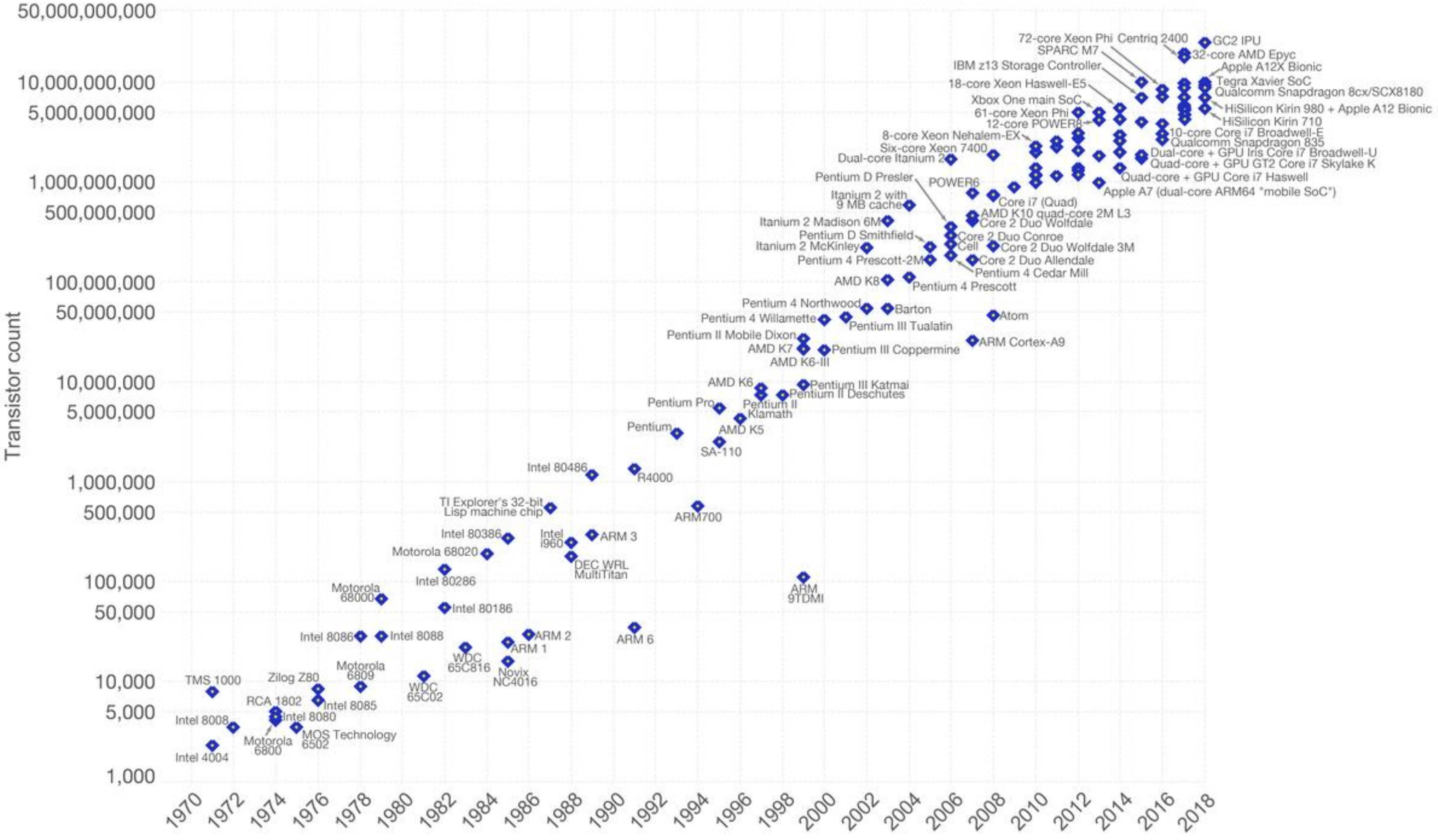
The chip manufacturing process

Video: How an IC is made

Moore’s Law

Moore’s Law – The number of transistors on integrated circuit chips (1971-2018)

Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years. This advancement is important as other aspects of technological progress – such as processing speed or the price of electronic products – are linked to Moore's law.



Data source: Wikipedia (https://en.wikipedia.org/wiki/Transistor_count)
The data visualization is available at OurWorldinData.org. There you find more visualizations and research on this topic.

Licensed under CC-BY-SA by the author Max Roser.

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	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
2005	Ultra large scale IC	6,200,000,000

[Textbook]

2. Computer performance evaluation

- To maximize performance, need to minimize execution time

$$\text{performance}_X = 1 / \text{execution\_time}_X$$

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 $$\frac{15}{10} = 1.5$$

So A is 1.5 times faster than B

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

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

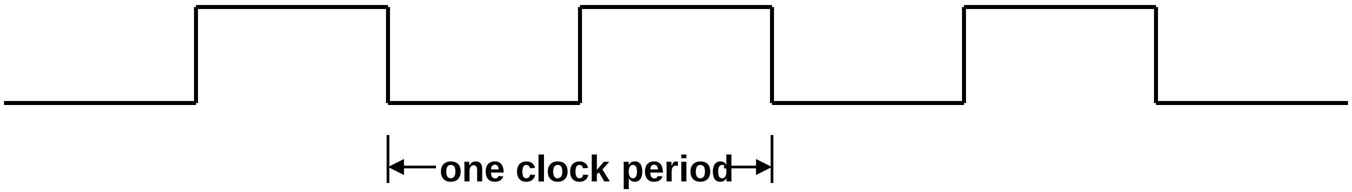
$$= \frac{\text{\# CPU clock cycles for a program}}{\text{clock rate}}$$

- 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)

$$CC = 1 / CR$$



10 nsec clock cycle	=>	100 MHz clock rate
5 nsec clock cycle	=>	200 MHz clock rate
2 nsec clock cycle	=>	500 MHz clock rate
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

# CPU clock cycles for a program = **# Instructions for a program** x **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

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

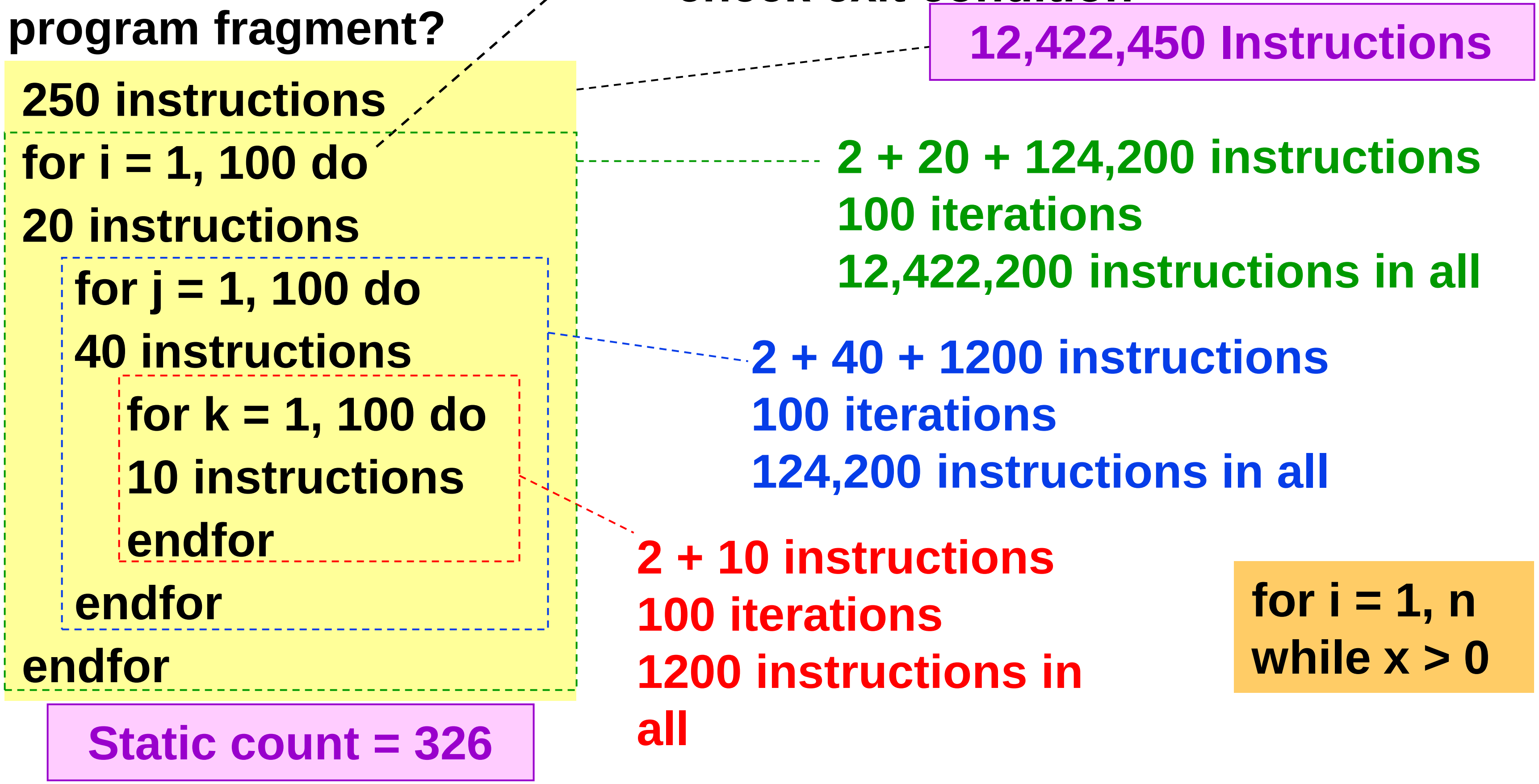
$$= \frac{\text{Instruction\_count} \times \text{CPI}}{\text{clock\_rate}}$$

- 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

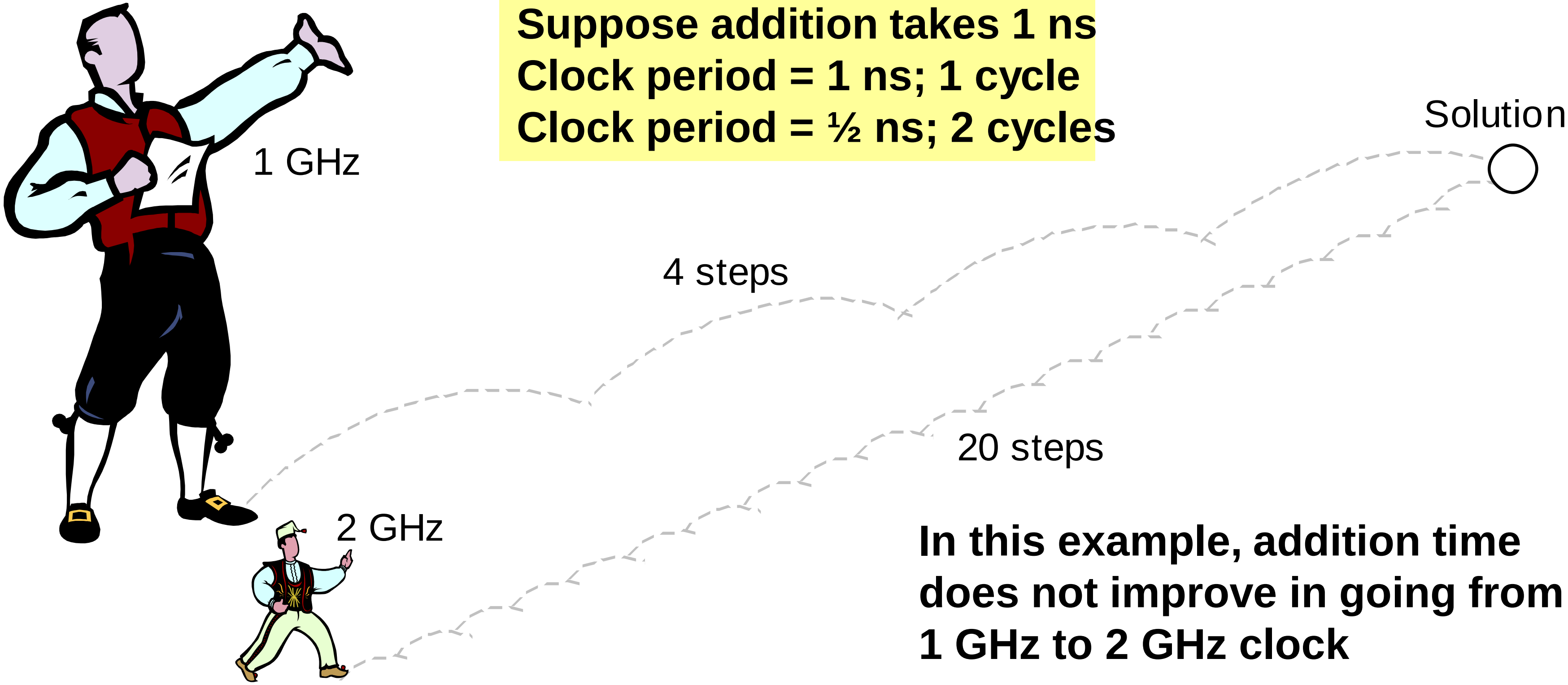
  → optimizing algorithm and compiler

- 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

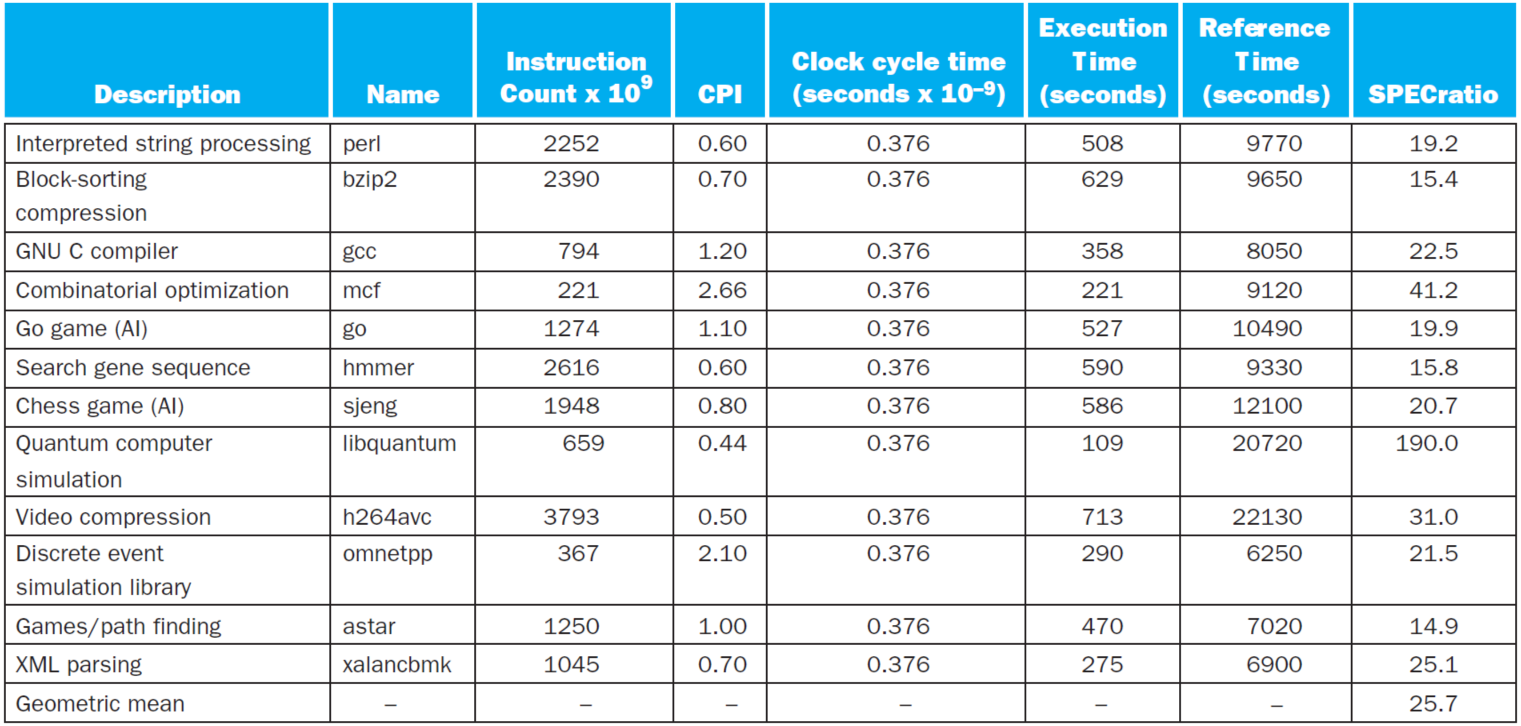


In this example, addition time does not improve in going from 1 GHz to 2 GHz clock

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.