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
  505 B1, SoICT, HUST
- Text: [Required] Computer Organization and Design, 4th edition revised printing Patterson & Hennessy 2012.

  [Optional] Computer Organization and Architecture, 8th Edition, William Stalling
- Slides: hard copy (how to print?)
  pdf on class website (URL TBA)
- Schedule: as in timetable

Self introduction

- Ngo Lam Trung, PhD
- Current position
  - 2004~: Lecturer, Department of Computer Engineering, School of Information and Communication Technology, Hanoi University of Science and Technology.
  - 2012~: Visiting researcher, Shibaura Institute of Technology, Tokyo, Japan.
  - 2013~: Head of Computer Systems Laboratory, School of Information and Communication Technology, Hanoi University of Science and Technology.
- Educational background
  - MSc in Information Processing and Communication, Hanoi University of Technology, 2006.
  - PhD in Functional Control Systems, Shibaura Institute of Technology, Japan, 2012.

Grading information

- Grading criteria
  - Attendance/Attitude 10%
  - Assignment(s) 20%
  - Mid-term test 20%
  - Final exam (TBA) 50%
- Assignments (TBA)
- Other requirements
  - No music/video/surfing/eating in class
  - Copy n paste from unreferenced sources

Why you need this course?

- Of course, it's in our curriculum!!!

…More than that, this course is helpful for you

  - Software developer: to write better programs
  - Hardware designer: to make better computer
  - User: to know which computer is best suitable for your work

- Expected outcomes: understanding of
  - Organization and architecture of modern computer.
  - Relationship between hardware and software.
  - Factors for computer performance evaluation.

Course content

- Chapter 1: Introduction
  - 1.1 Background
  - 1.2 Computer Abstraction and Technology
  - 1.3 Performance Evaluation
- Chapter 2: Instruction Set Architecture
  - 2.1 Overview
  - 2.2 MIPS instruction set
  - 2.3 MIPS organization

Course content

- Chapter 3: Computer Arithmetic
  - 3.1 Integer arithmetic
  - 3.2 Floating point arithmetic
- Chapter 4: CPU
  - 4.1 Introduction
  - 4.2 Simple CPU implementation
  - 4.3 Enhancing performance with pipelining

Prerequisites - What You Should Know

- Basic PC organization
  - How computer parts look like?
- Basic logic design & machine organization
  - logical minimization, FSMs, component design
- Create, compile, and run C/C++ programs
- Create, run, debug programs in an assembly language

- *What if all you know is MS Word, VLC, NFS, FB?*
  - *Try harder!*

Chapter 1: Introduction

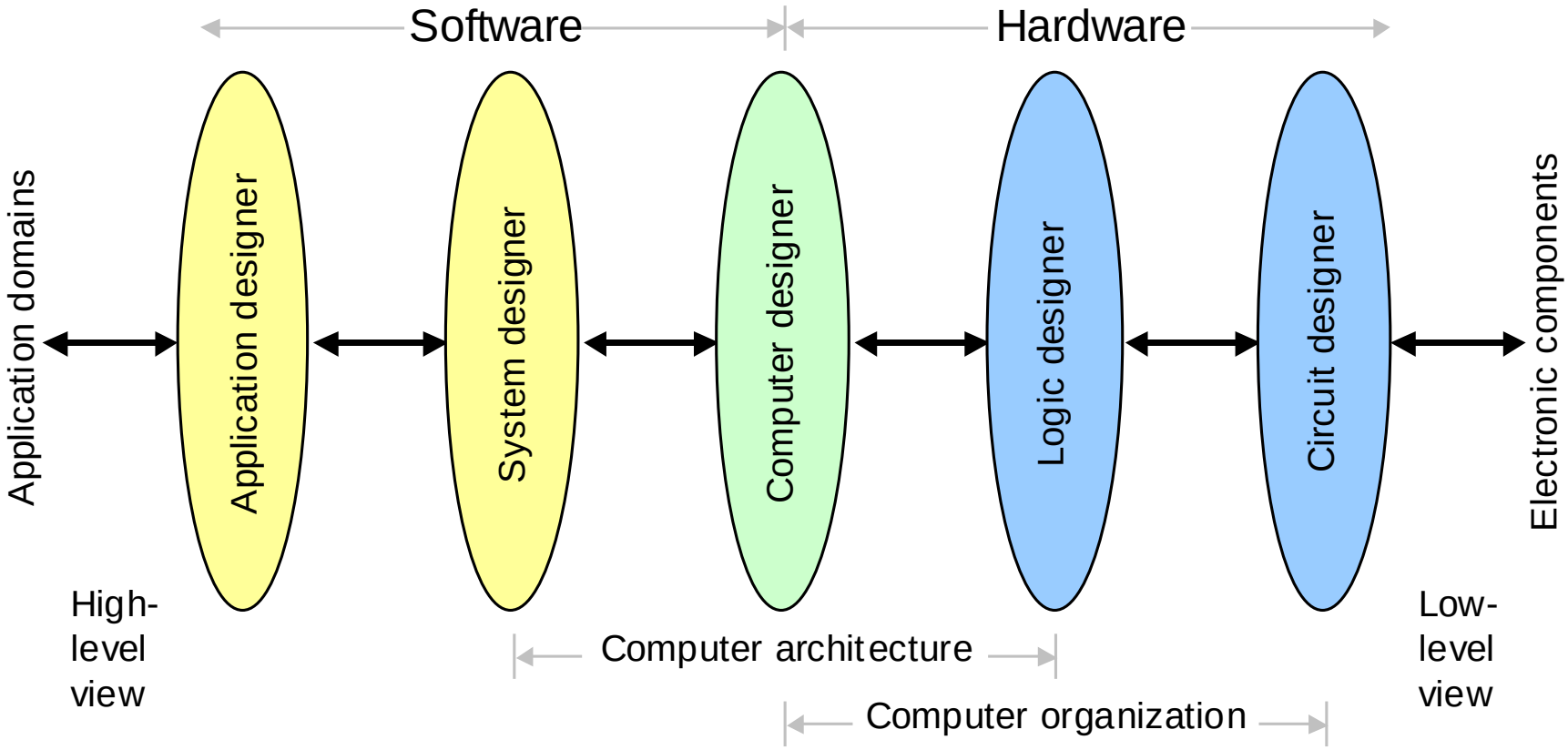
1. Computer Abstraction and Technology
2. Performance Evaluation

1. Computer Abstraction and Technology

- **Computer architecture:** attributes of a system
  - Visible to a programmer
  - Or have a direct impact on the logical execution of a program.
- **Computer organization**
  - The operational units and their interconnections that realize the architectural specifications.
  - include those hardware details transparent to the programmer,
    - control signals;
    - interfaces between the computer and peripherals;
    - and the memory technology used.
- Which is more hardware/software related?

Who we are?

- What do you want to do in the future?

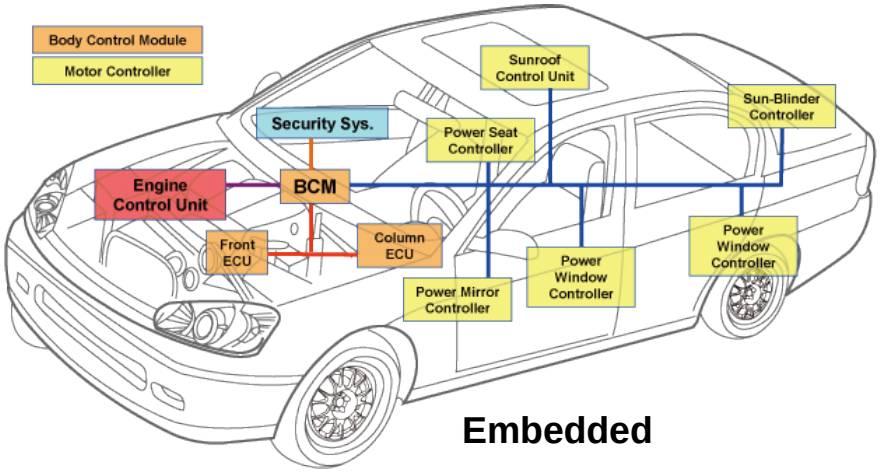


Subfields or views in computer system engineering

Classes of Computers

- Supercomputers
  - Super fast + expensive for high-end applications
- Server computers
  - 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

Dominan look and feel of computer classes



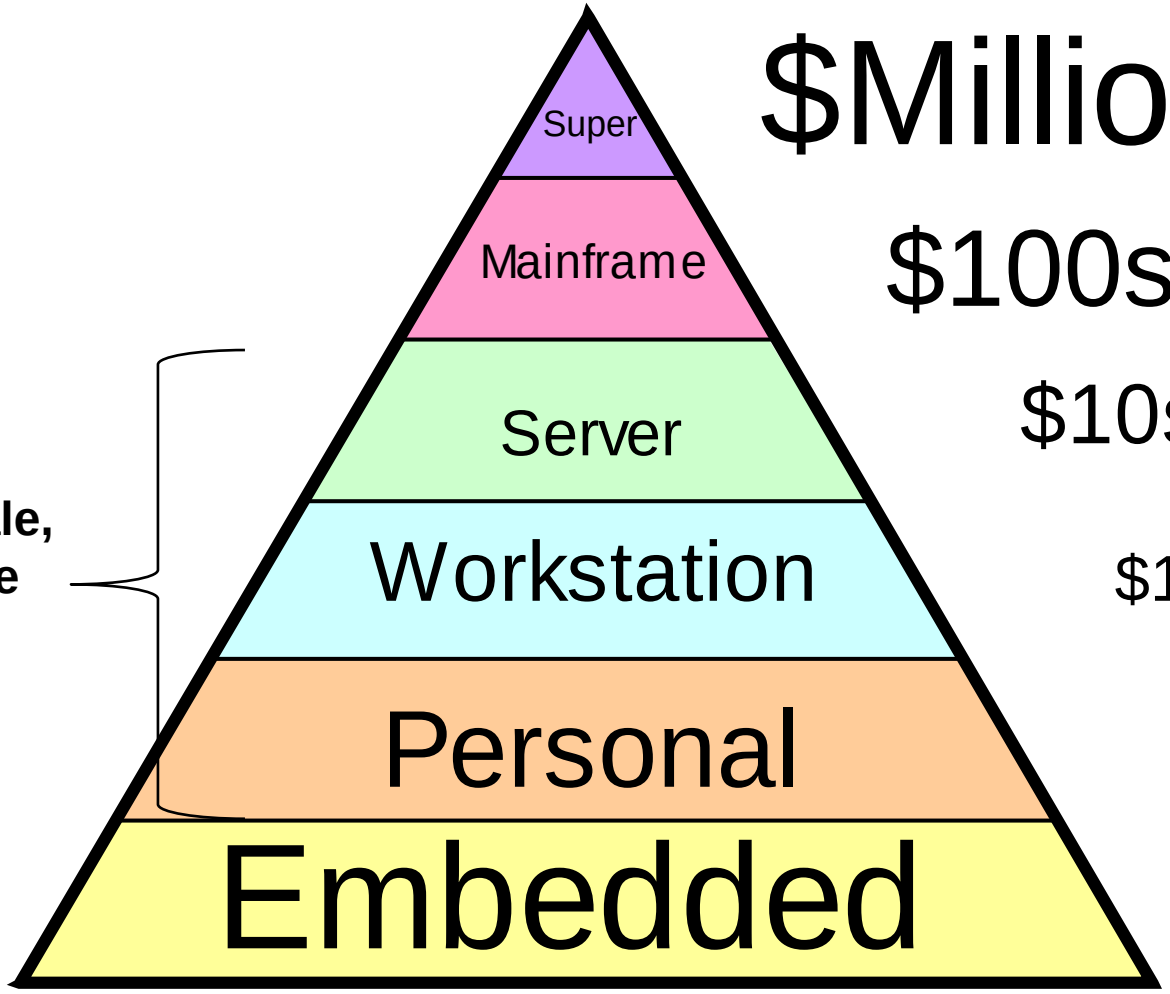
Embedded

PC

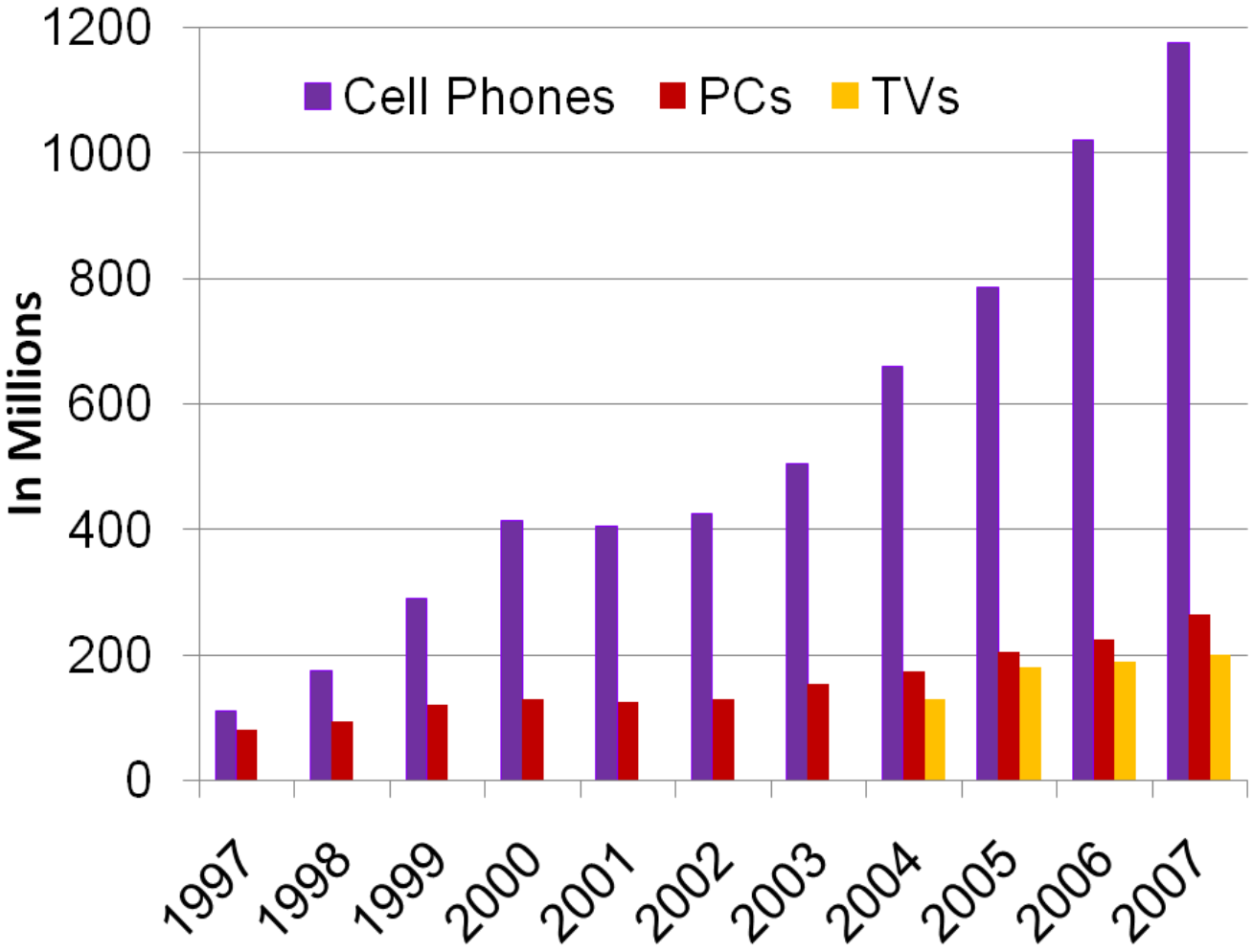
Server

Super computer

Price/performance of computer classes



The Processor Market



embedded growth >> desktop growth

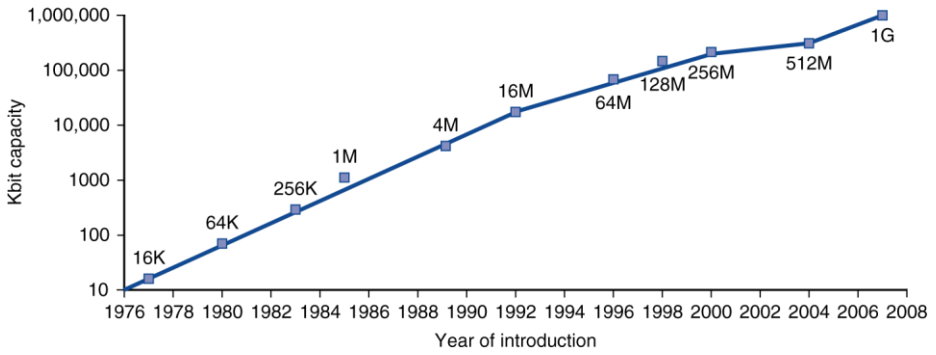
Generations of Progress

Generation (begun)	Processor technology	Memory innovations	I/O devices introduced	Dominant look & fell
0 (1600s)	(Electro-) mechanical	Wheel, card	Lever, dial, punched card	Factory equipment
1 (1950s)	Vacuum tube	Magnetic drum	Paper tape, magnetic tape	Hall-size cabinet
2 (1960s)	Transistor	Magnetic core	Drum, printer, text terminal	Room-size mainframe
3 (1970s)	SSI/MSI	RAM/ROM chip	Disk, keyboard, video monitor	Desk-size mini
4 (1980s)	LSI/VLSI	SRAM/DRAM	Network, CD, mouse,sound	Desktop/ laptop micro
5 (1990s)	ULSI/GSI/ WSI, SOC	SDRAM, flash	Sensor/actuator, point/click	Invisible, embedded

[Ref. 2]

Technology Trends

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

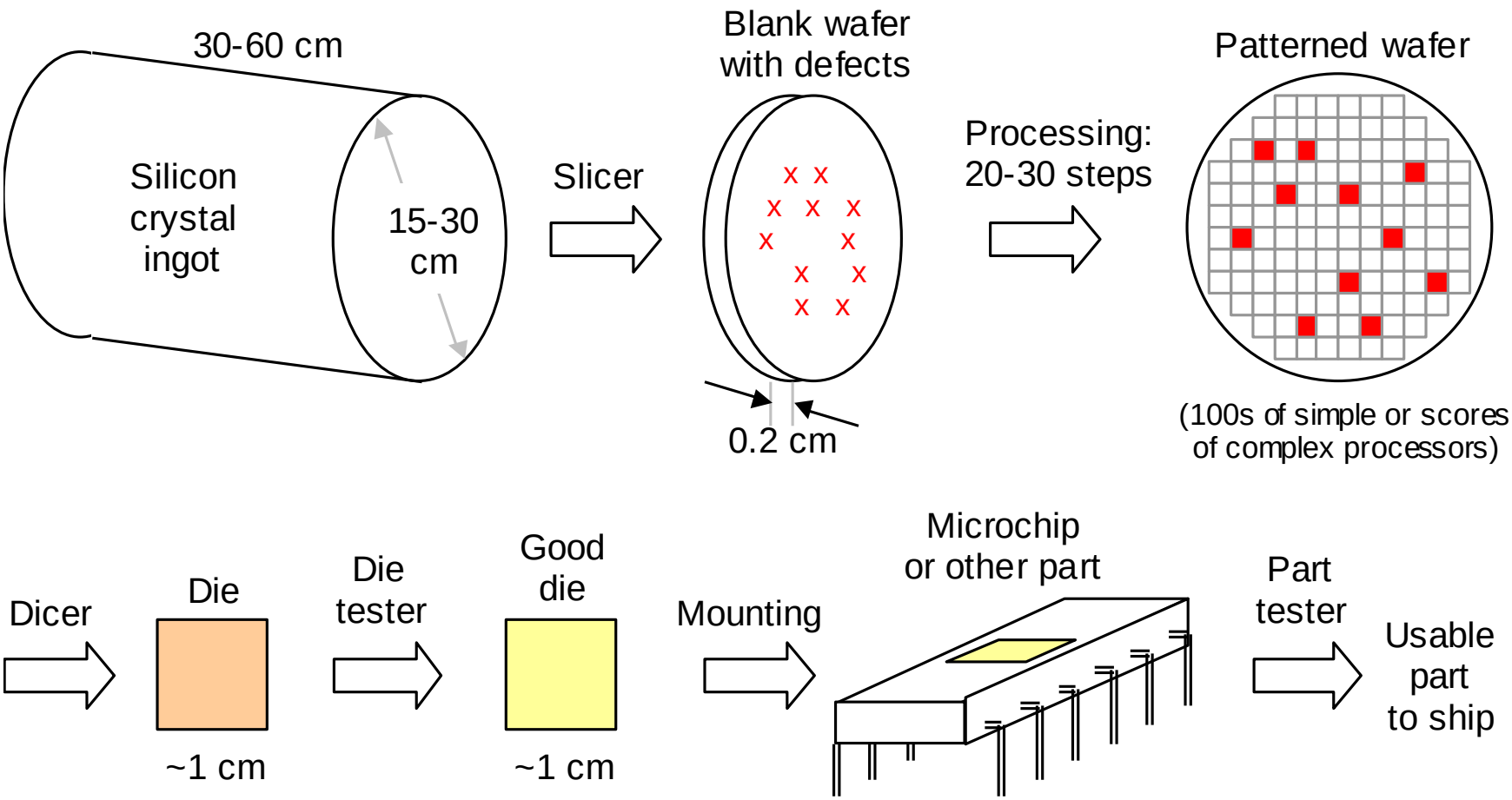


DRAM capacity

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]

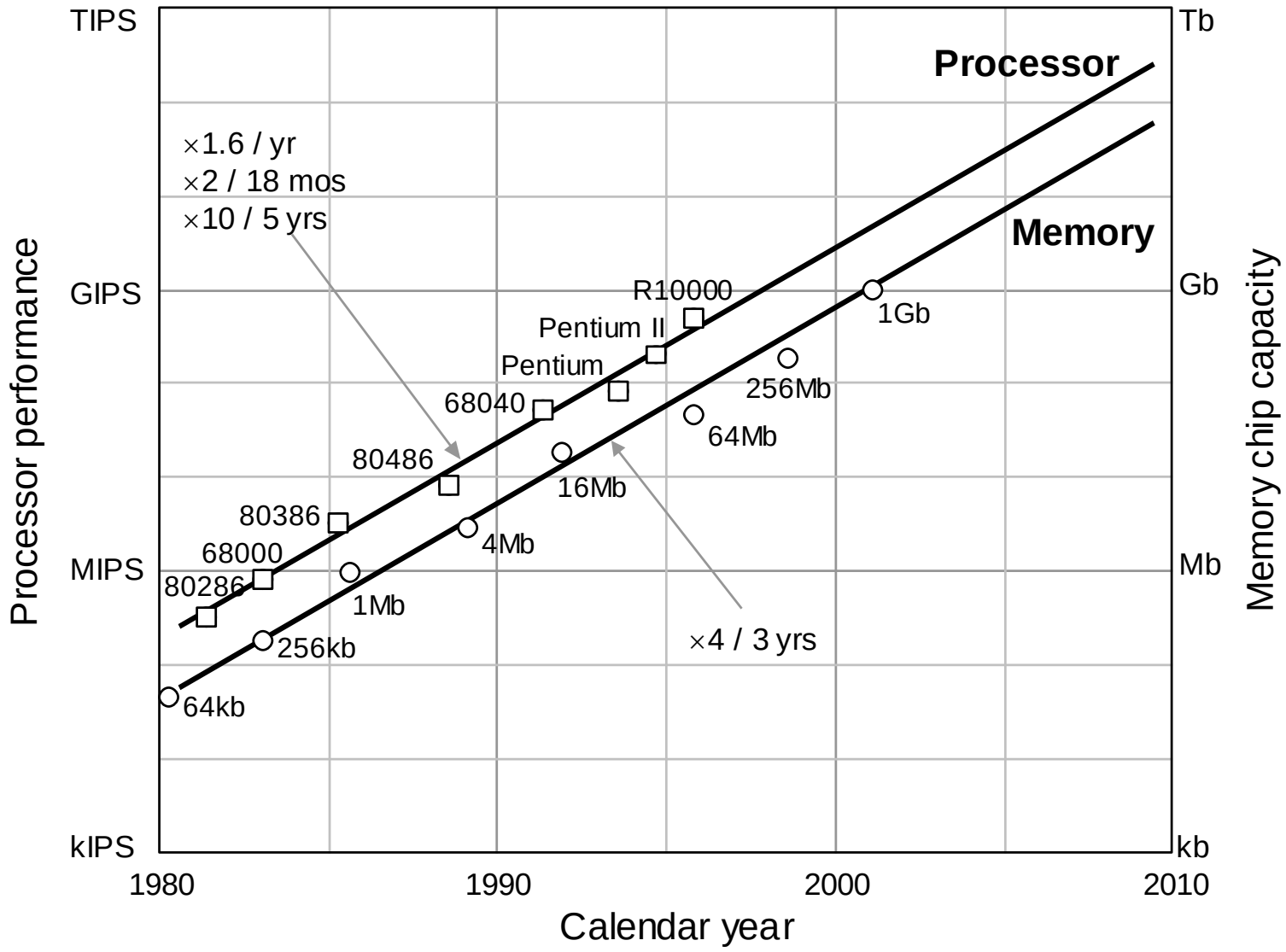
IC making



The manufacturing process for an IC part

Video: How an IC is made

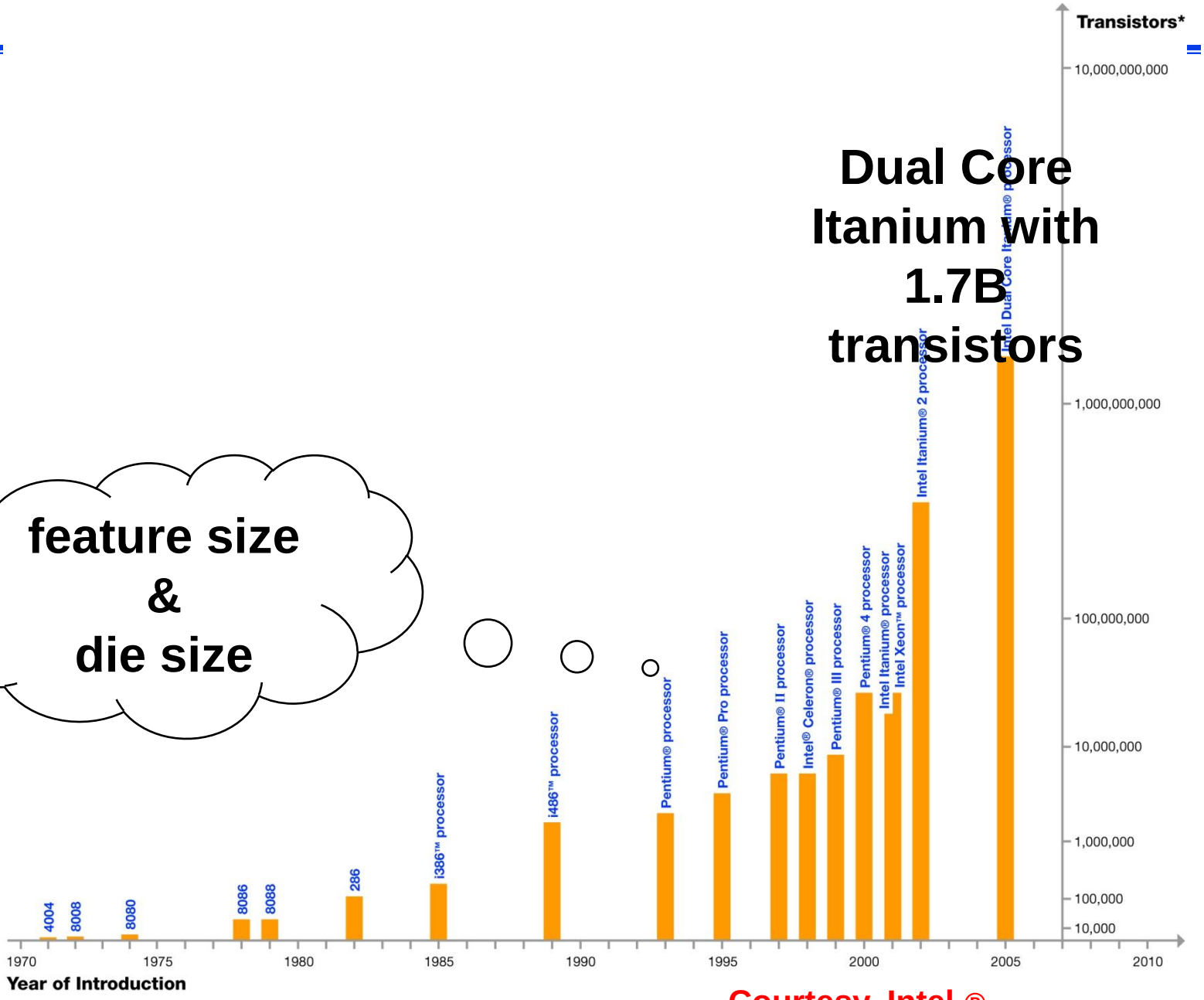
Moore's Law



How do we benefit from this?

Dual Core Itanium with 1.7B transistors

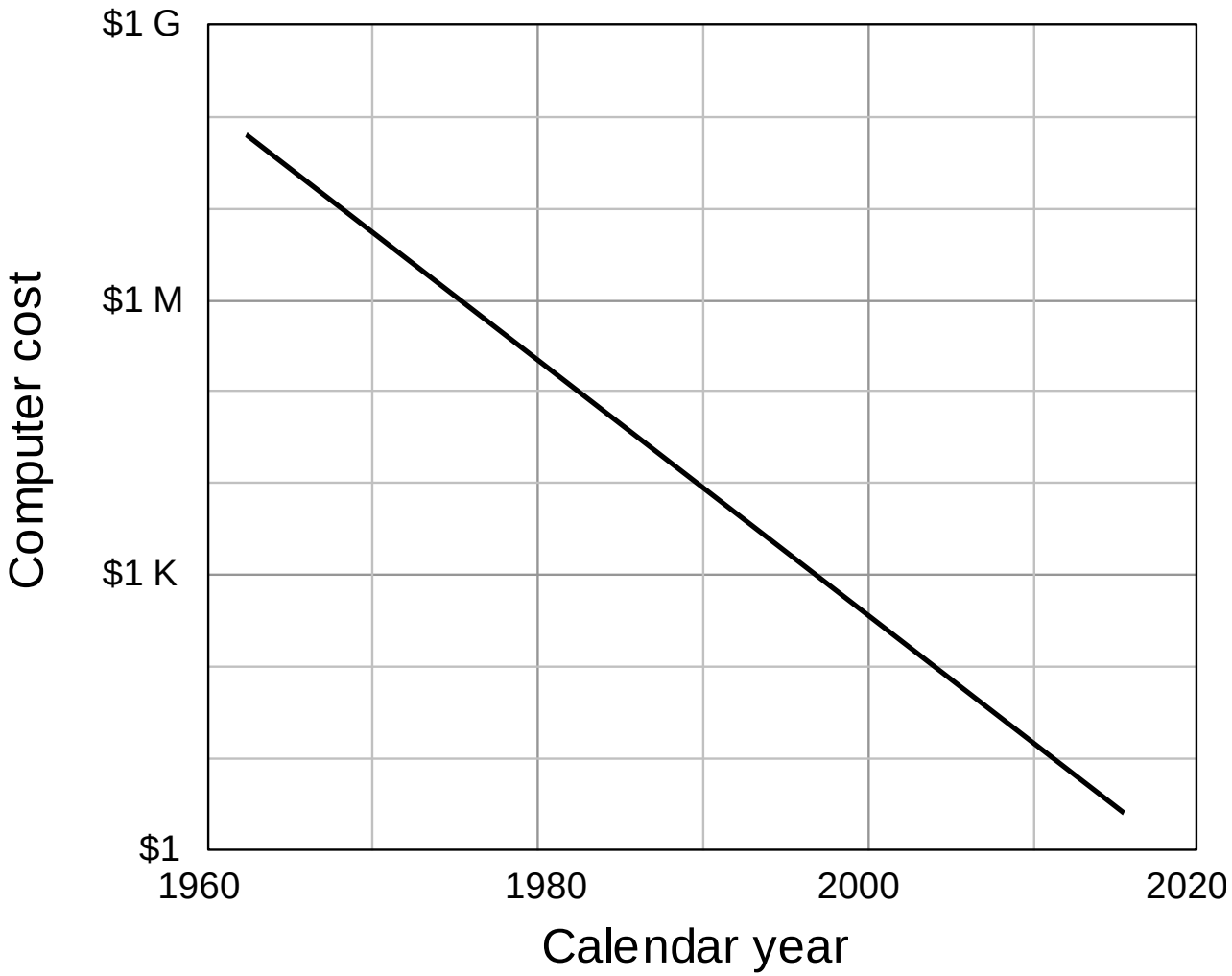
feature size & die size



*Note: Vertical scale of chart not proportional to actual Transistor count.

Courtesy, Intel ®

Cost/performance



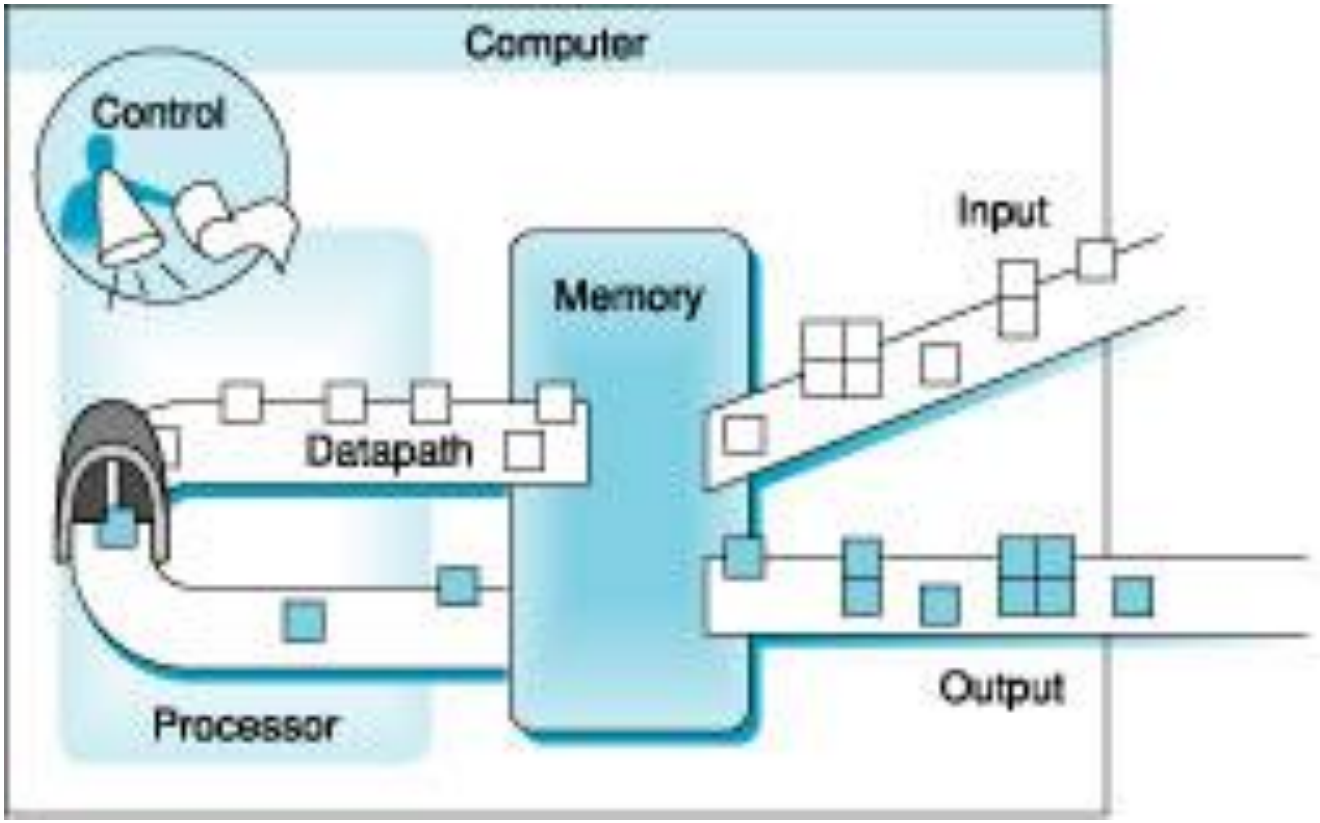
Homework

- Why CPU speed does not exceed 3.5-4GHz?

Computer Organization

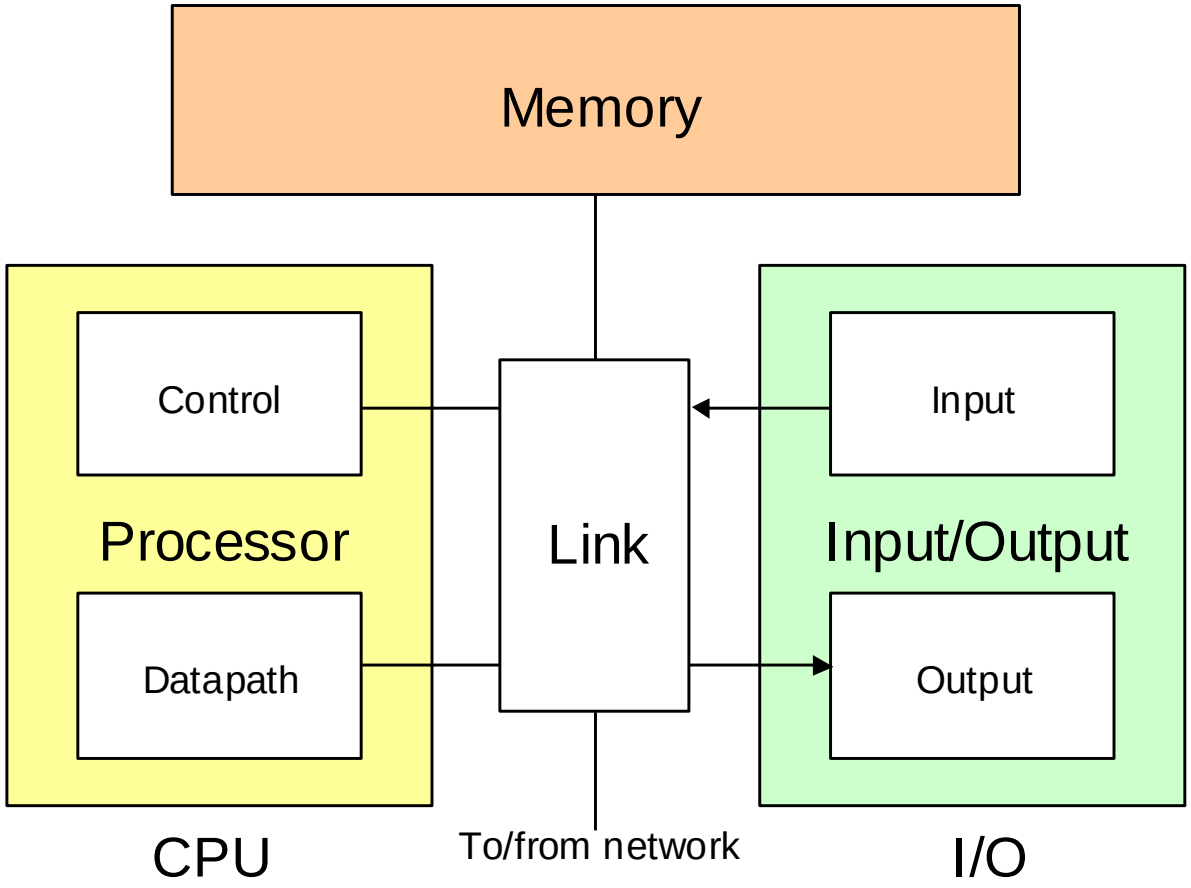
- Five classic components of a computer – input, output, memory, datapath, and control

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

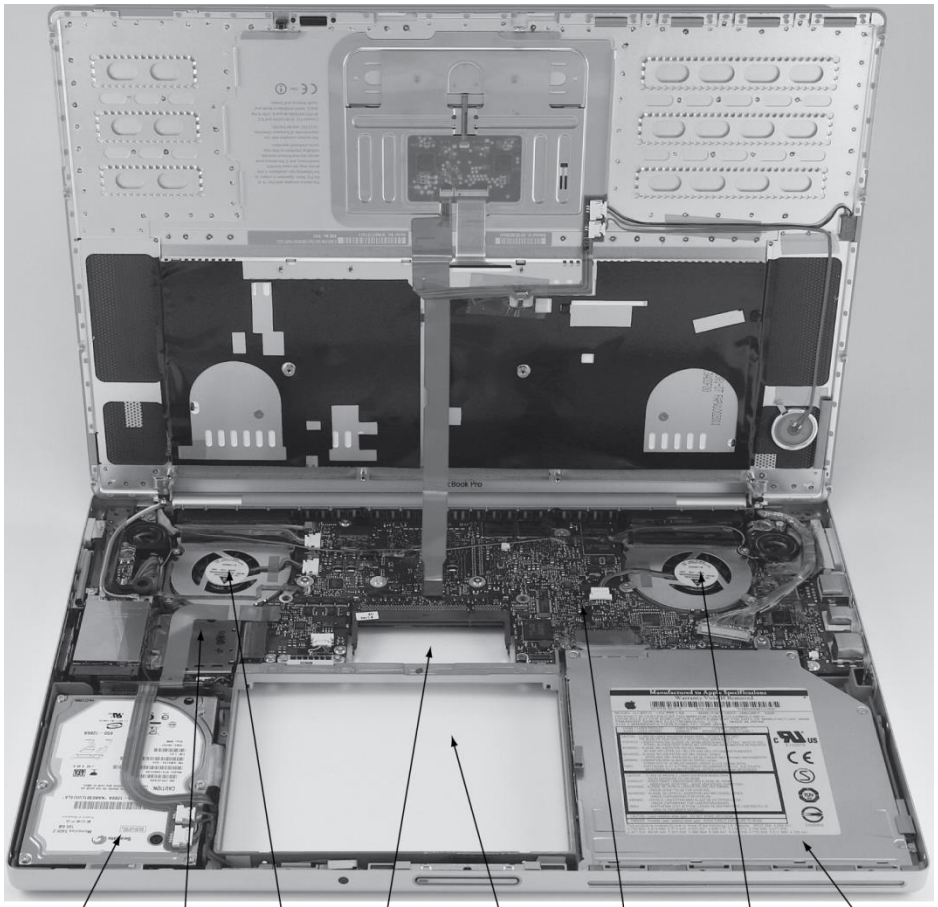


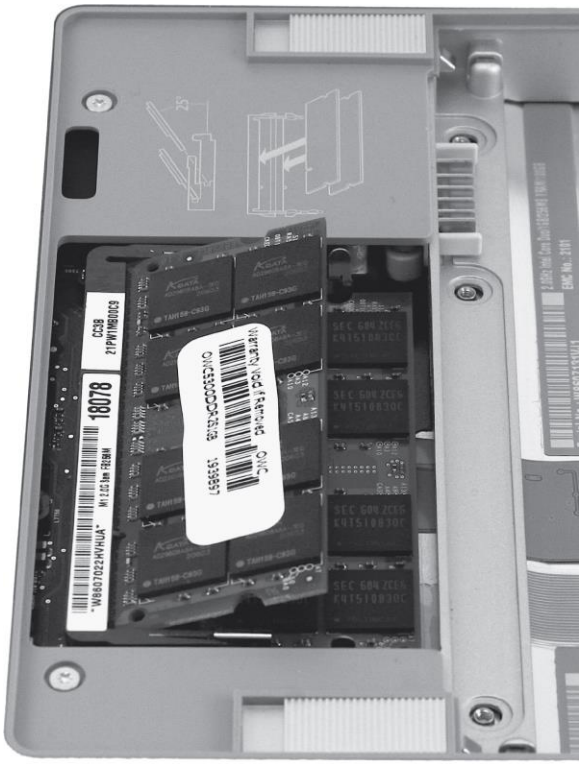
A similar view

- Usually, the Link unit is hidden



Opening the box: anatomy of computer





Opening the box: anatomy of computer

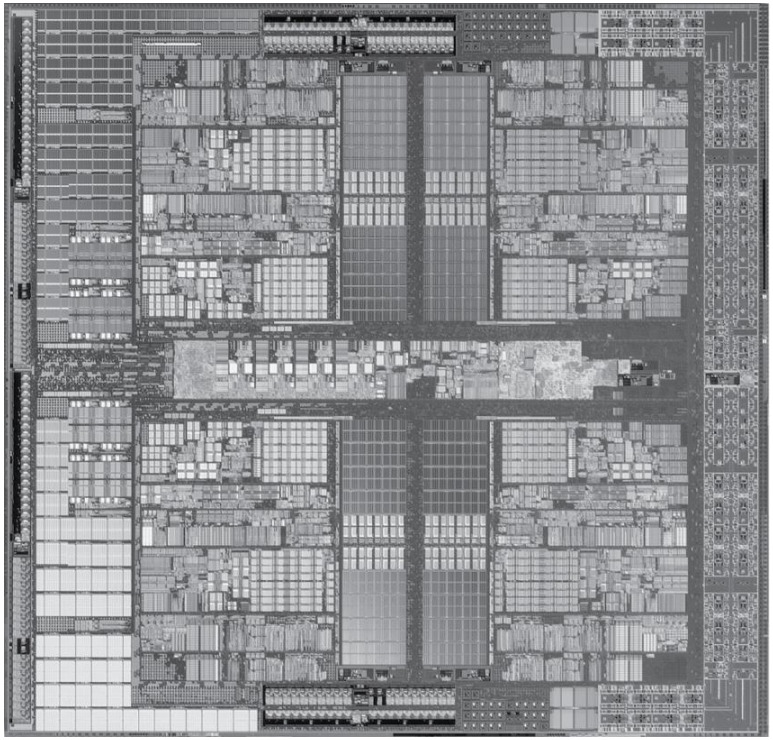
The story of each component worth a separate course!

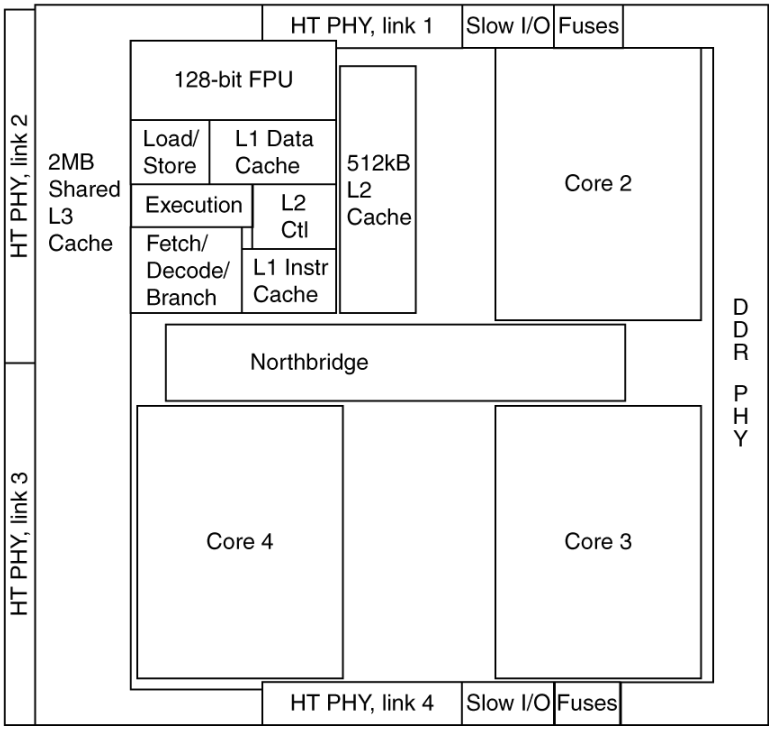
Inside the Processor (CPU)

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

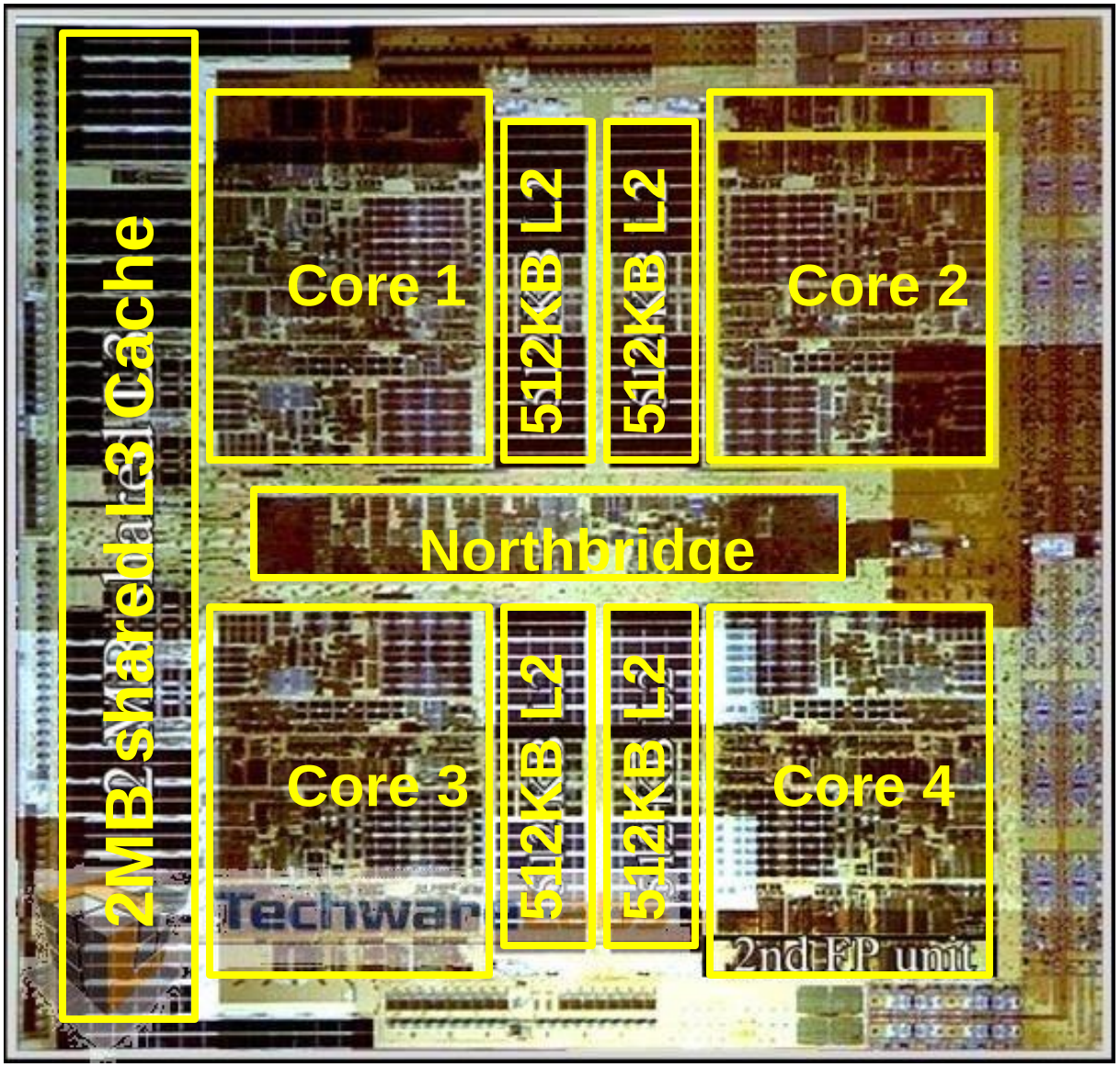
Inside the Processor

- AMD Barcelona: 4 processor cores





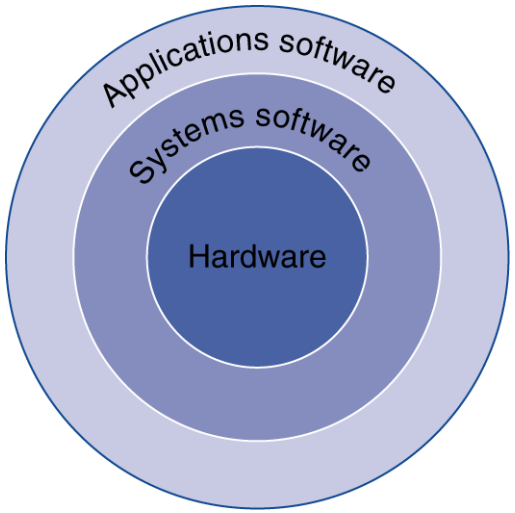
AMD's Barcelona Multicore Chip



- 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/

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

Below your program

- High-level language program (in C)

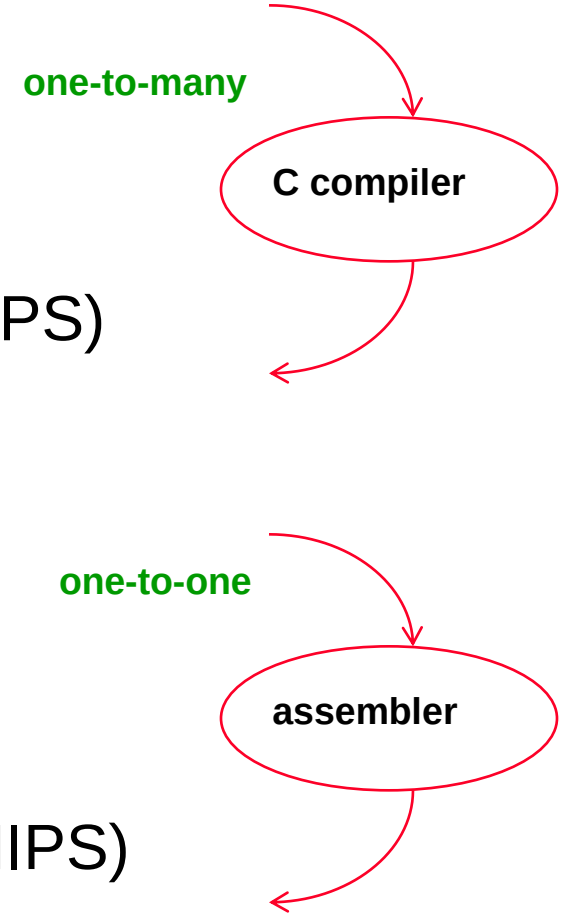
```
swap (int v[], int k)
(int temp;
        temp = v[k];
        v[k] = v[k+1];
        v[k+1] = temp;
)
```

- Assembly language program (for MIPS)

```
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
```

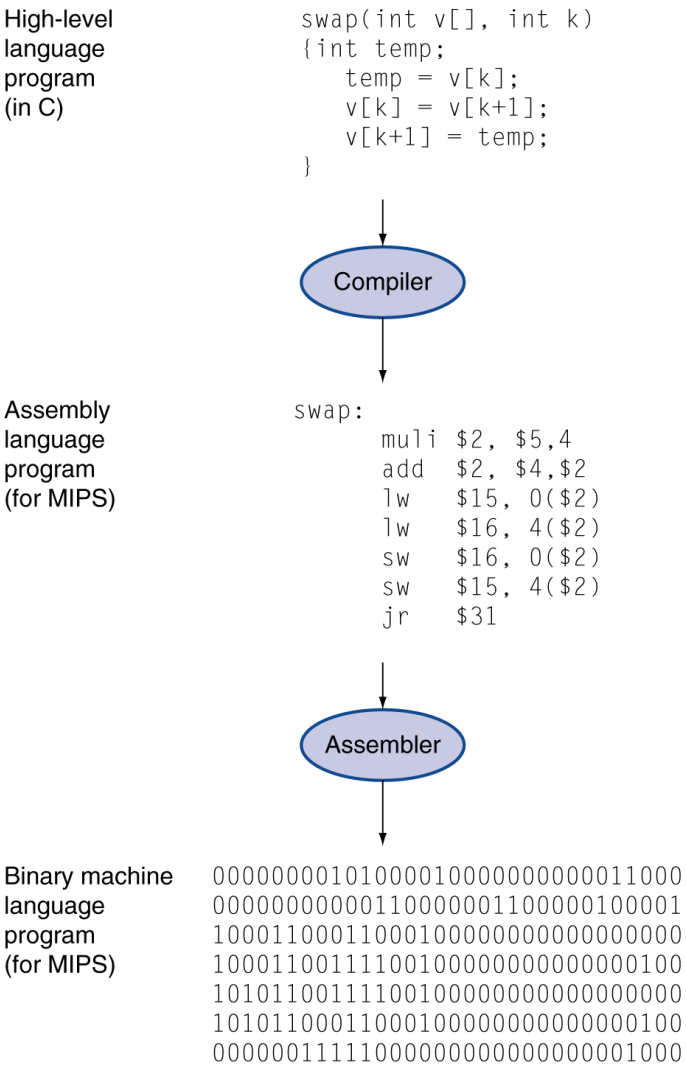
- Machine (object, binary) code (for MIPS)

```
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



Advantages of Higher-Level Languages ?

- Higher-level languages
  - Allow the programmer to think in a more natural language and for their intended use (Fortran for scientific computation, Cobol for business programming, Lisp for symbol manipulation, Java for web programming, …)
  - Improve programmer productivity – more understandable code that is easier to debug and validate
  - Improve program maintainability
  - Allow programs to be independent of the computer on which they are developed (compilers and assemblers can translate high-level language programs to the binary instructions of any machine)
  - Emergence of optimizing compilers that produce very efficient assembly code optimized for the target machine
- As a result, very little programming is done today at the assembler level

Problem

- Computer organization evolves fast
  - New processor
  - New hardware

  …every year/day/month
- How a program/software can run on different computers?
- Instruction Set Architecture
- Abstraction

Computer performance

- 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

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

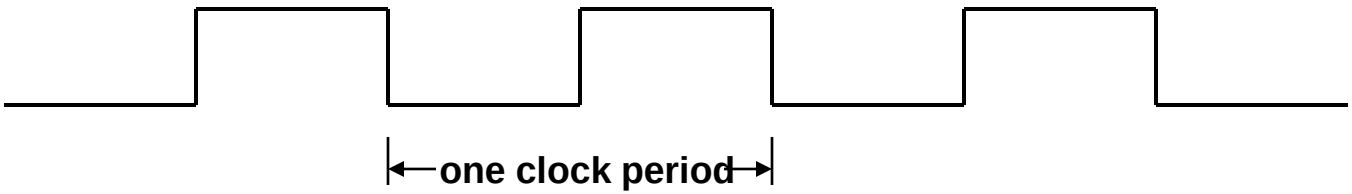
$$= \frac{\text{\# CPU clock cycles for a program}}{\text{clock rate}}$$

- 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

- CPU requires clock signal to operate
- 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.

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

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

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

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

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

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

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

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

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

Dynamic Instruction Count

How many instructions are executed in this program fragment?

```
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

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

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

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

Solution

4 steps

20 steps

2 GHz

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.

Review

- Computer technology and abstraction
- Computer performance evaluation

- Next chapter: MIPS's instruction set