COMPUTER ARCHITECTURE
CSE

Faculty of Computer Science and Engineering
Department of Computer Engineering

Vo Tan Phuong
http://www.cse.hcmut.edu.vn/~vtphuong

Chapter 1

Introduction

Presentation Outline

- Welcome to CA CSE
- Computer Architectures and Trends
- High-Level, Assembly-, and Machine-Languages
- Components of a Computer System
- Chip Manufacturing Process
- Programmer's View of a Computer System

Welcome to CA CSE

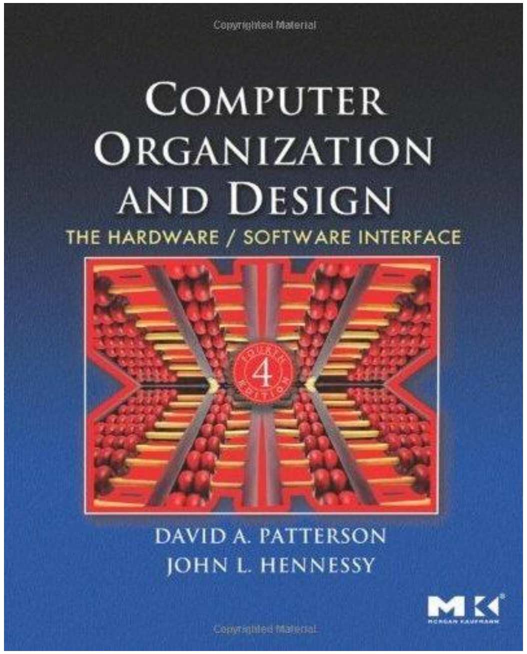
- Instructor: Võ Tấn Phương
  - ➢ Email: vtphuong@cse.hcmut.edu.vn
- TA: Trần Thanh Bình
  - ➢ Email: thanhbinh.hcmut@gmail.com
- Course Web Page:
  - – http://www.cse.hcmut.edu.vn/~vtphuong/KTMT

Which Textbook will be Used?

- Computer Organization & Design:

  The Hardware/Software Interface

  - Fourth Edition
  - David Patterson and John Hennessy
  - Morgan Kaufmann Publishers, 2009



- Read the textbook in addition to slides

Estimated Schedule

- Introduction, Performance (2 week)
- Integer arithmetic, Floating Point Numbers (1 week)
- MIPS Instruction Set Architecture (3 weeks)
- MIPS Assembly Programming (1 weeks)
- Basic Digital Function Block, ALU (1 week)
- Single Cycle MIPS Processor (2 weeks)
- Pipelined MIPS Processor (2 weeks)
- Memory System (1 week)
- Cache Memory System (2 week)

Course Learning Outcomes

- Towards the end of this course, you should be able to …
  - Describe the instruction set architecture of a MIPS processor
  - Analyze, write, and test MIPS assembly language programs
  - Design the datapath and control of a single-cycle CPU
  - Design the datapath/control of a pipelined CPU & handle hazards
  - Describe the organization/operation of memory and caches
  - Analyze the performance of processors and caches
- Required Background
  - Ability to program confidently in Java or C
  - Ability to design a combinational and sequential circuit

Tentative Grading Policy

• Labs & Assignment	40%
– 2 Assignments	30%
– Exercises	10%
• Mid Exam	20%
– Quiz questions, closed book	
• Final Exam	40%
– Quiz questions, closed book	

- Bonus by white board quick exercises (max + 2)

Software Tools

- MIPS Simulators
  - MARS: MIPS Assembly and Runtime Simulator
    - Runs MIPS-32 assembly language programs
    - Website: http://courses.missouristate.edu/KenVollmar/MARS/
  - SPIM
    - Also Runs MIPS-32 assembly language programs
    - Website: http://www.cs.wisc.edu/~larus/spim.html
- Design simple CPU
  - NandToTetris
    - Link: http://www.nand2tetris.org/course.php

Presentation Outline

- Welcome to CA CSE
- Computer Architectures and Trends
- High-Level, Assembly-, and Machine-Languages
- Components of a Computer System
- Chip Manufacturing Process
- Programmer's View of a Computer System

What is “Computer Architecture“ ?

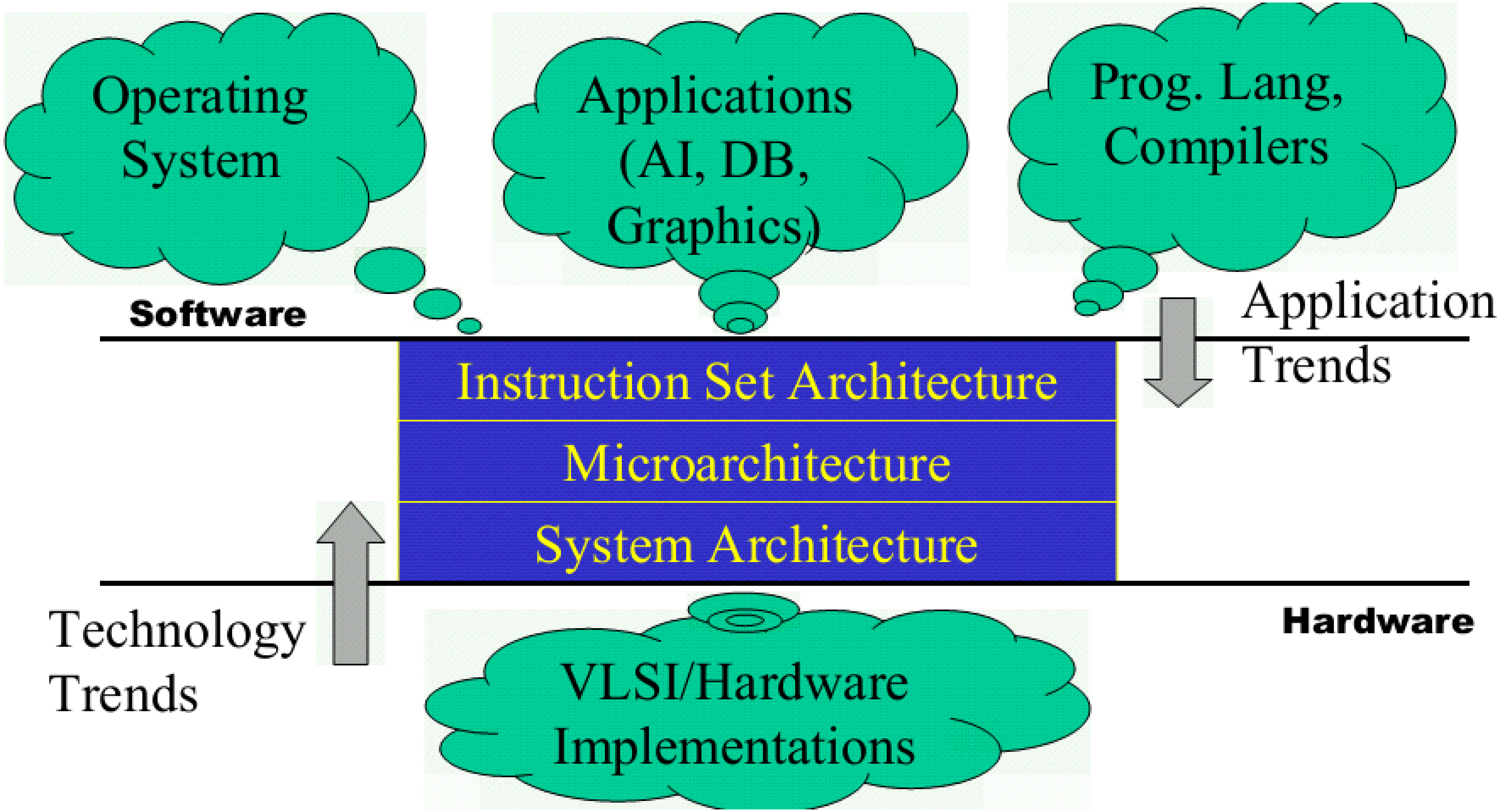
- Computer Architecture =

  Instruction Set Architecture +

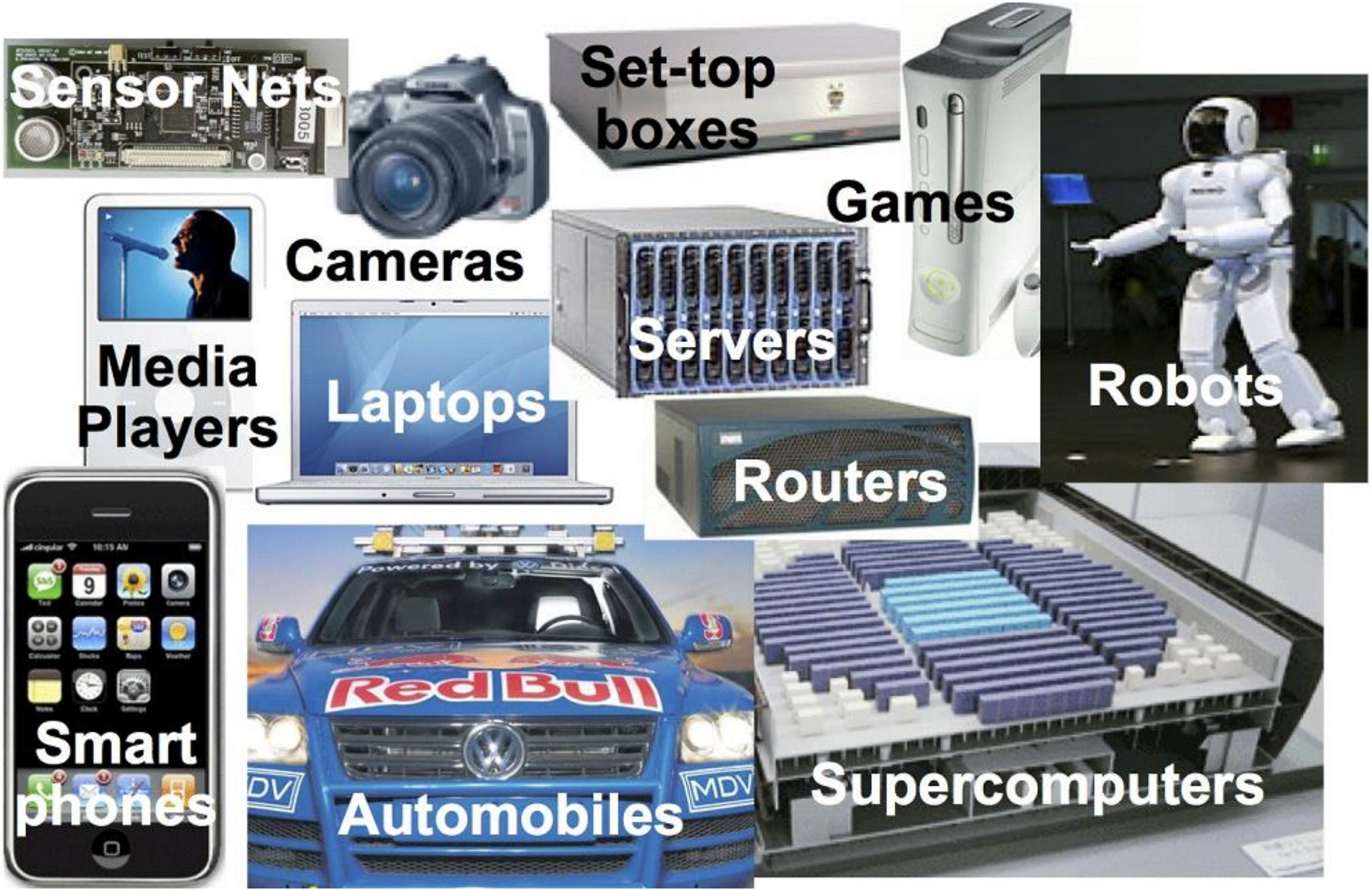
  Computer Organization

- Instruction Set Architecture (ISA)
  - WHAT the computer does (logical view)
- Computer Organization
  - HOW the ISA is implemented (physical view)
- We will study both in this course

Computer Architecture In Context



Trend 1: Growing Diversity In Apps & Systems



Trend 2: Software trend

- No longer just executing C/FORTRAN code
- Object Oriented Programming
- Java
- Architectural features to assist security
- Middleware
  - Layer(s) between client and server applications
  - Hides complexity of client/server communications

Trend 3: Energy/Power Constrain all Modern Systems

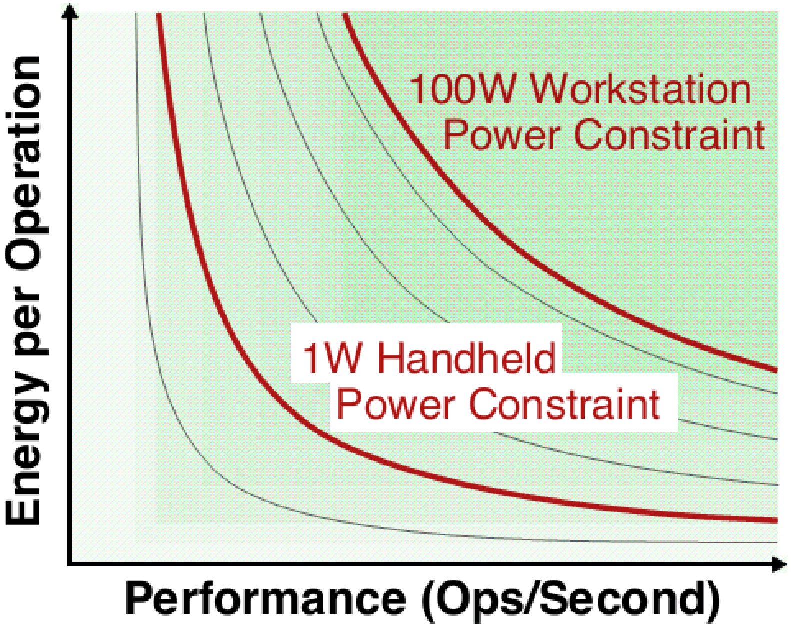
$$\text{Power} = \frac{\text{Energy}}{\text{Second}} = \frac{\text{Energy}}{\text{Op}} \times \frac{\text{Ops}}{\text{Second}}$$

Power

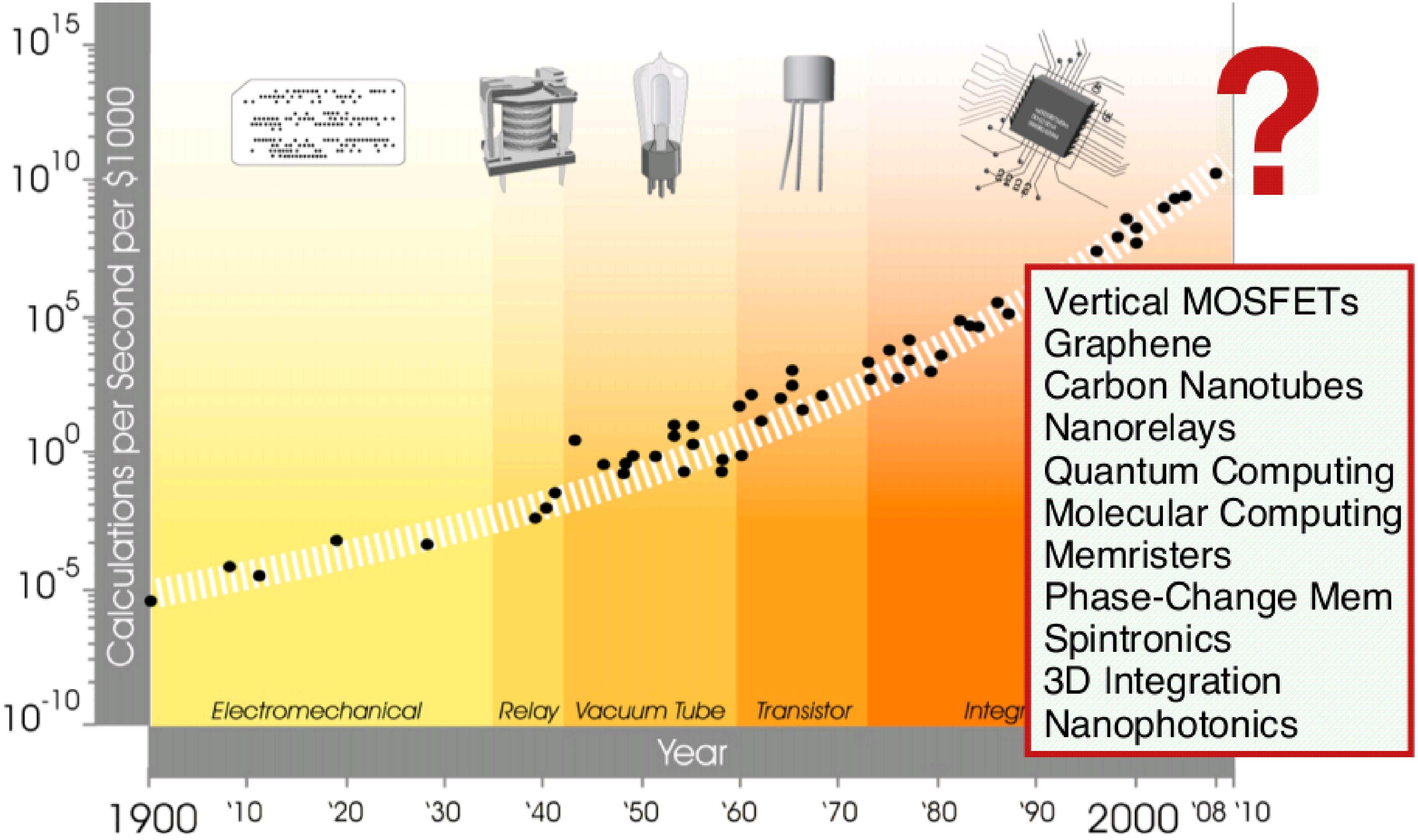
Chip Packaging
Chip Cooling
System Noise
Case Temperature
Data-Center Air
Conditioning

Energy

Battery Life
Electricity Bill
Mobile Device
Weight

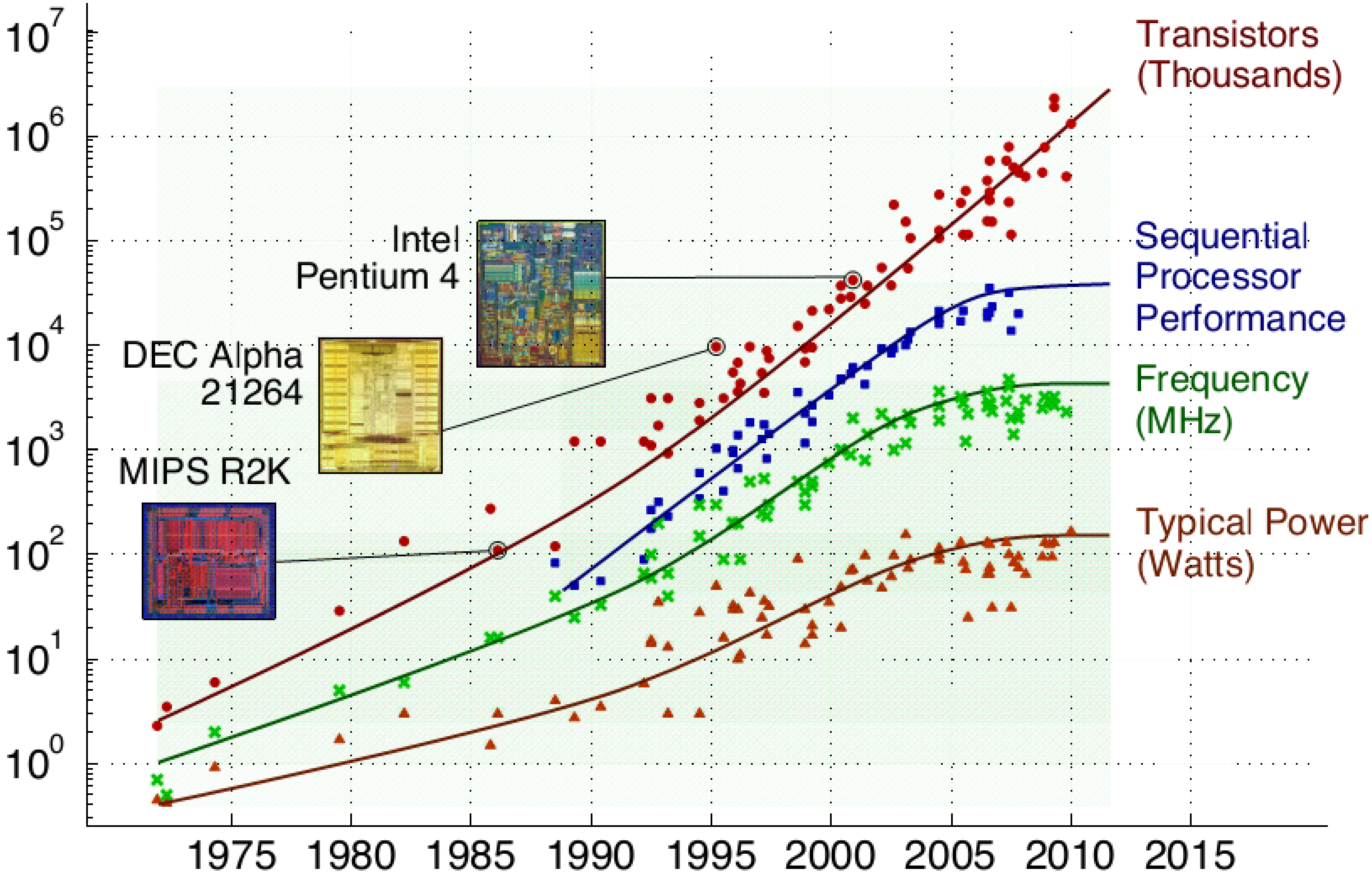


Emerging Device Technologies



- Vertical MOSFETs
- Graphene
- Carbon Nanotubes
- Nanorelays
- Quantum Computing
- Molecular Computing
- Memristers
- Phase-Change Mem
- Spintronics
- 3D Integration
- Nanophotonics

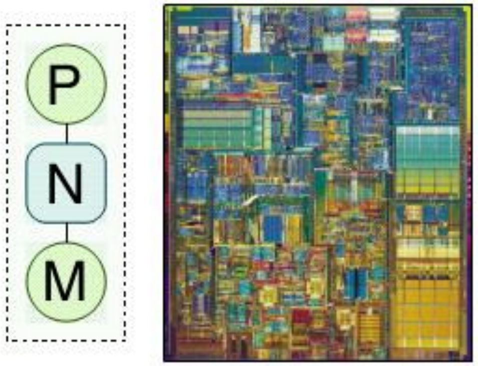
Power Constrains Single-Processor Scaling



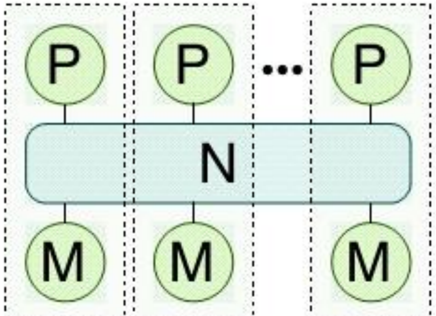
Data partially collected by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond

Transition to Multicore Processors

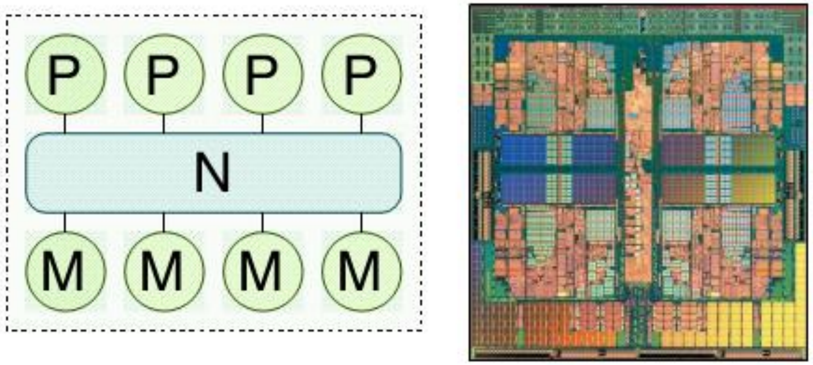
Intel Pentium 4
Single monolithic processor

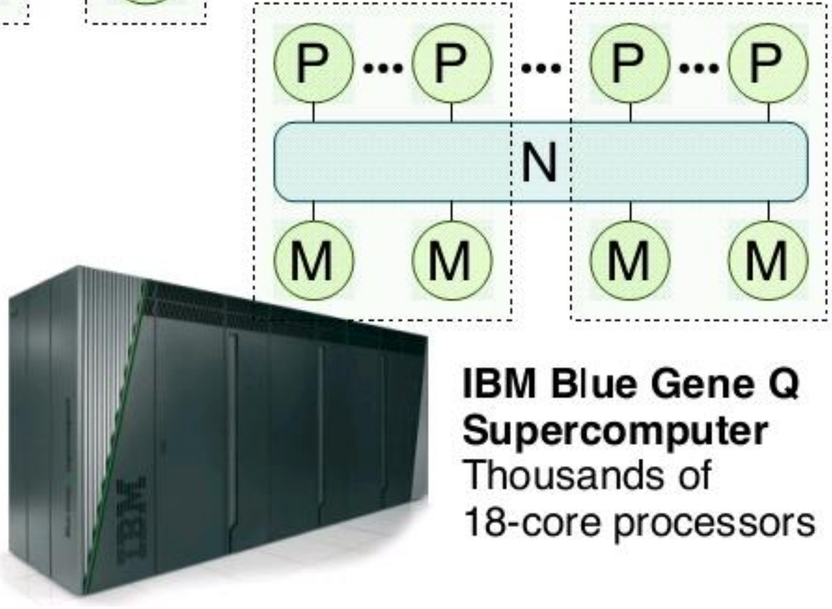


Cray XT3 Supercomputer
1024 single-core processors



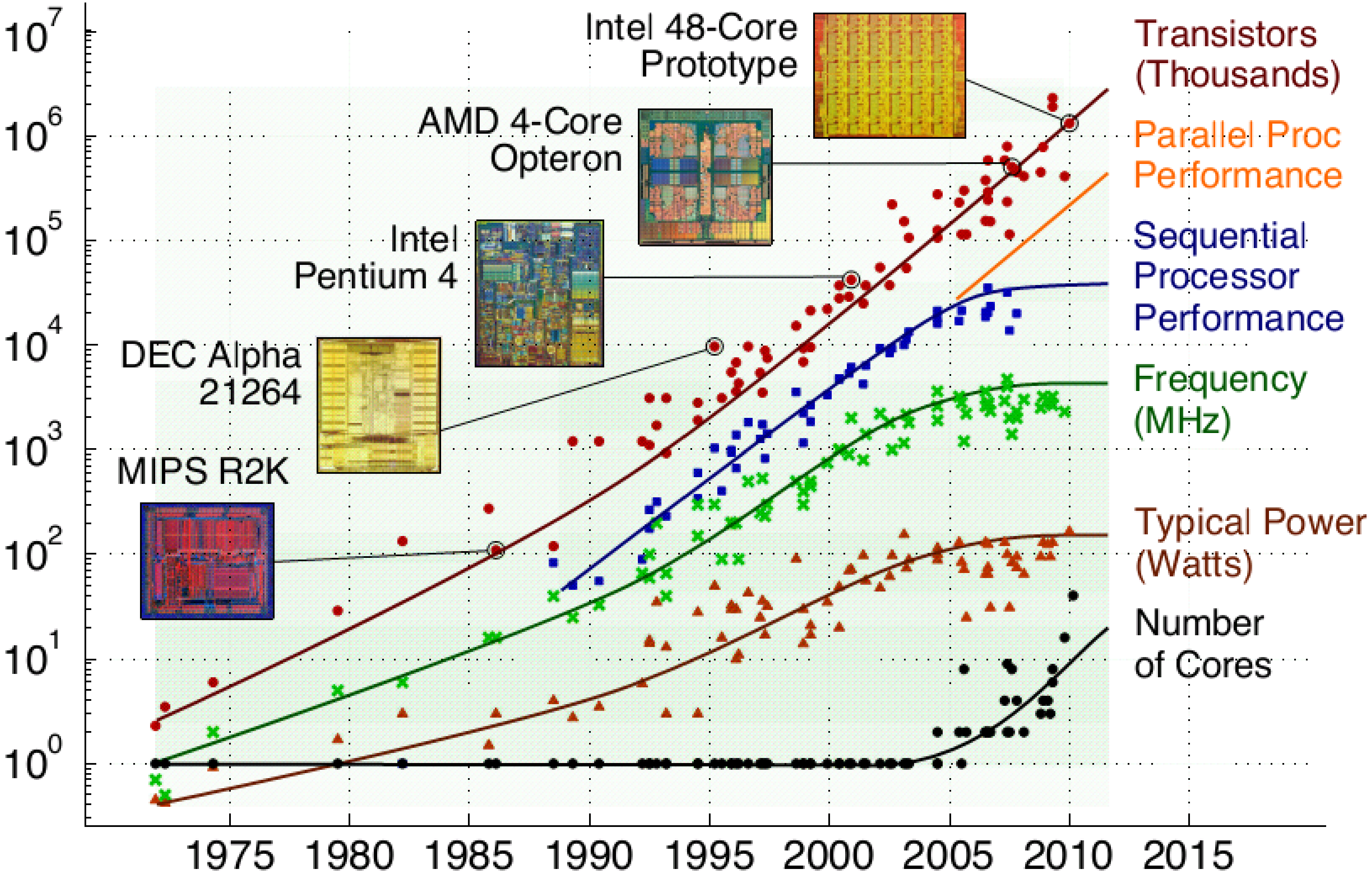
AMD Quad-Core Opteron
Four cores on the same die





IBM Blue Gene Q Supercomputer
Thousands of
18-core processors

Multicore Performance Scaling

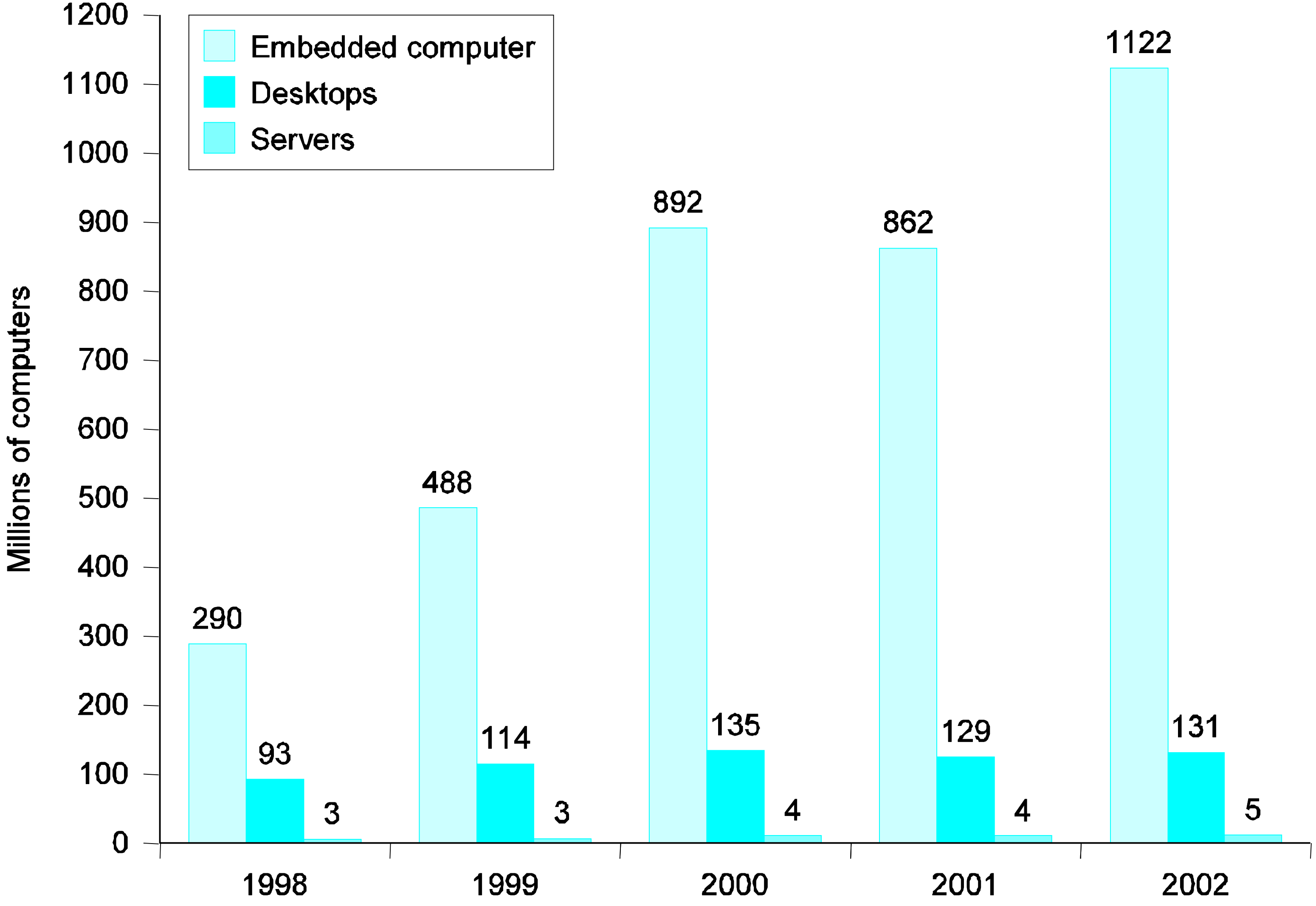


Data partially collected by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond

Classes of Computers

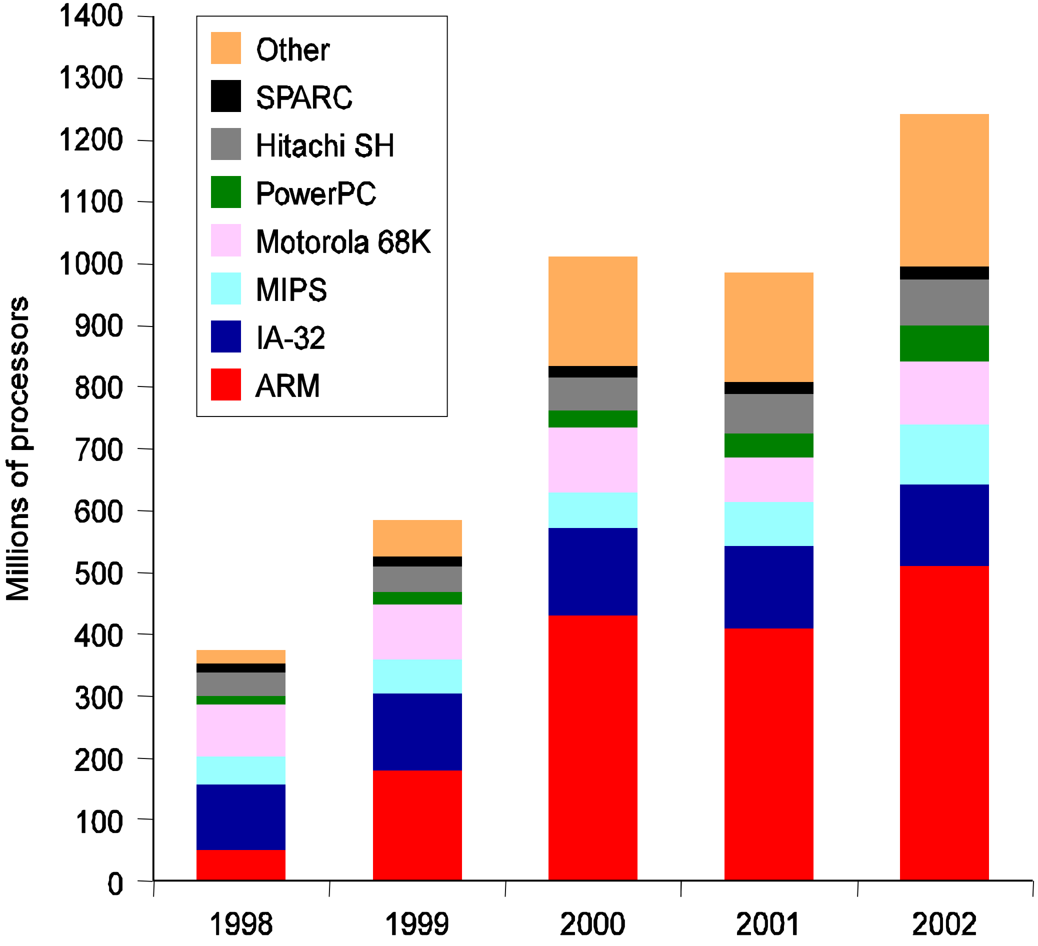
- Desktop / Notebook Computers
  - General purpose, variety of software
  - Subject to cost/performance tradeoff
- Server Computers
  - Network based
  - High capacity, performance, reliability
  - Range from small servers to building sized
- Embedded Computers
  - Hidden as components of systems
  - Stringent power/performance/cost constraints

Computer Sales



Microprocessor Sales

- ARM processor sales exceeded Intel IA-32 processors, which came second
- ARM processors are used mostly in cellular phones
- Most processors today are embedded in cell phones, digital TVs, video games, and a variety of consumer devices



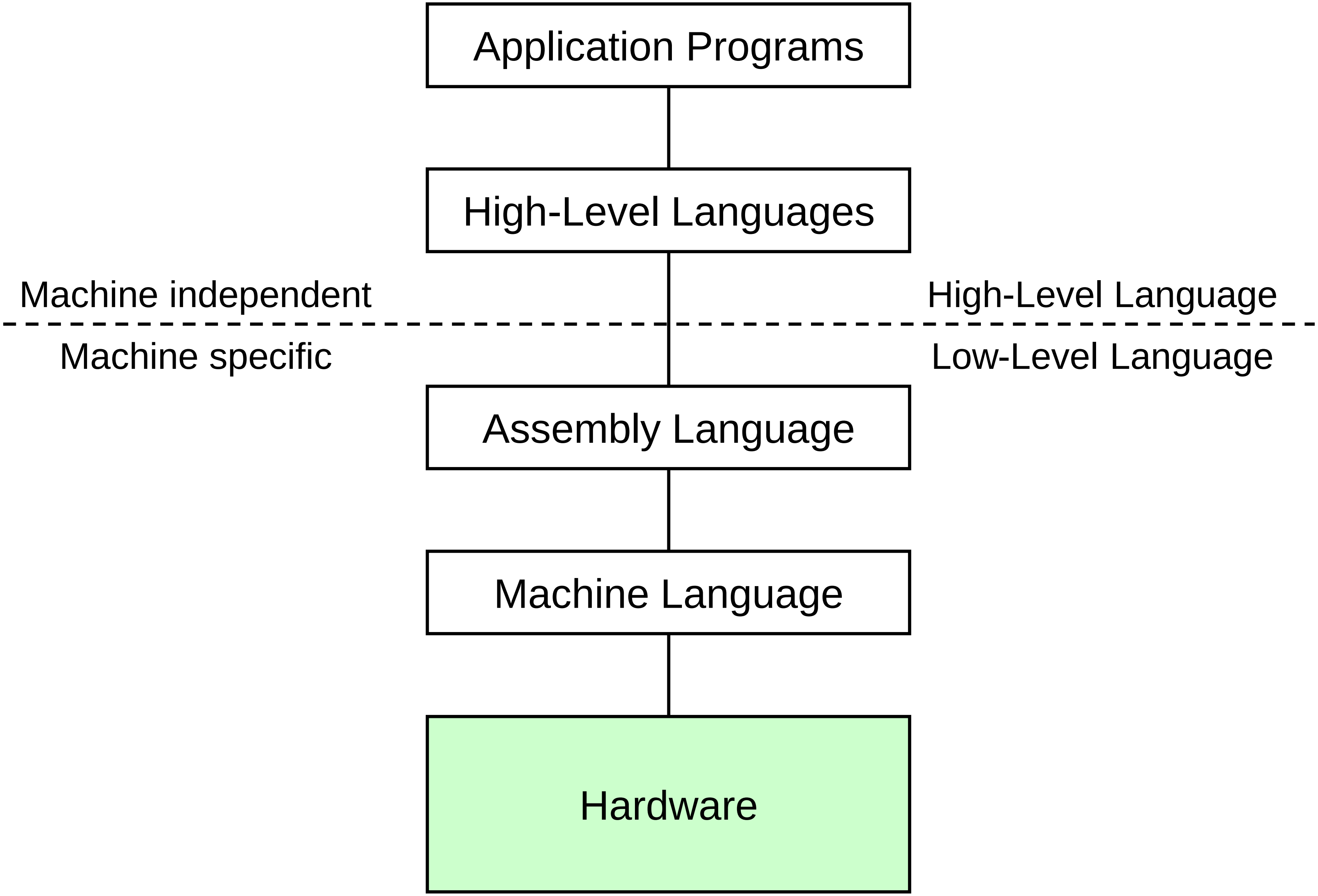
Presentation Outline

- Welcome to CA CSE
- Computer Architectures and Trends
- High-Level, Assembly-, and Machine-Languages
- Components of a Computer System
- Chip Manufacturing Process
- Programmer's View of a Computer System

Some Important Questions to Ask

- What is Assembly Language?
- What is Machine Language?
- How is Assembly related to a high-level language?
- Why Learn Assembly Language?
- What is an Assembler, Linker, and Debugger?

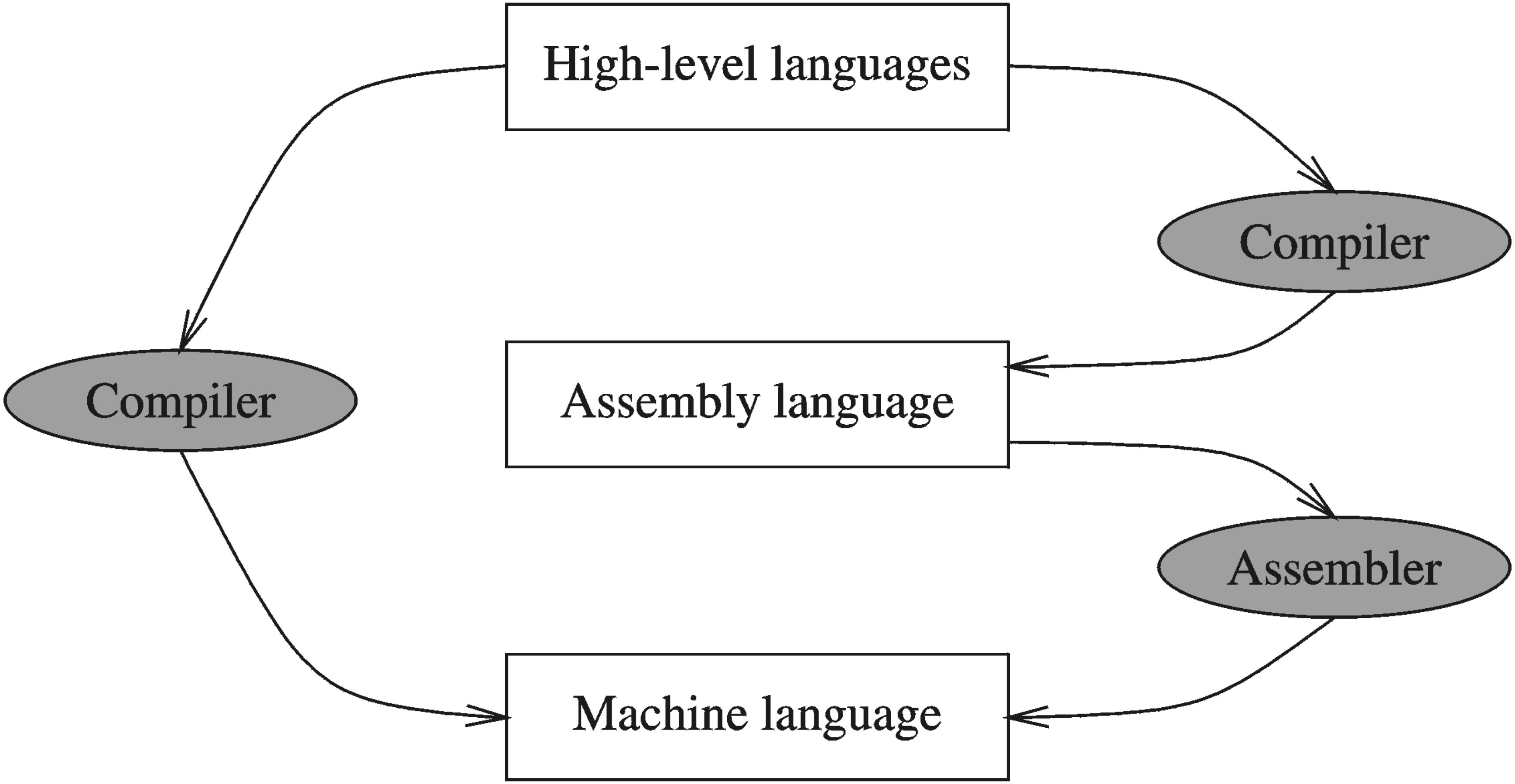
A Hierarchy of Languages



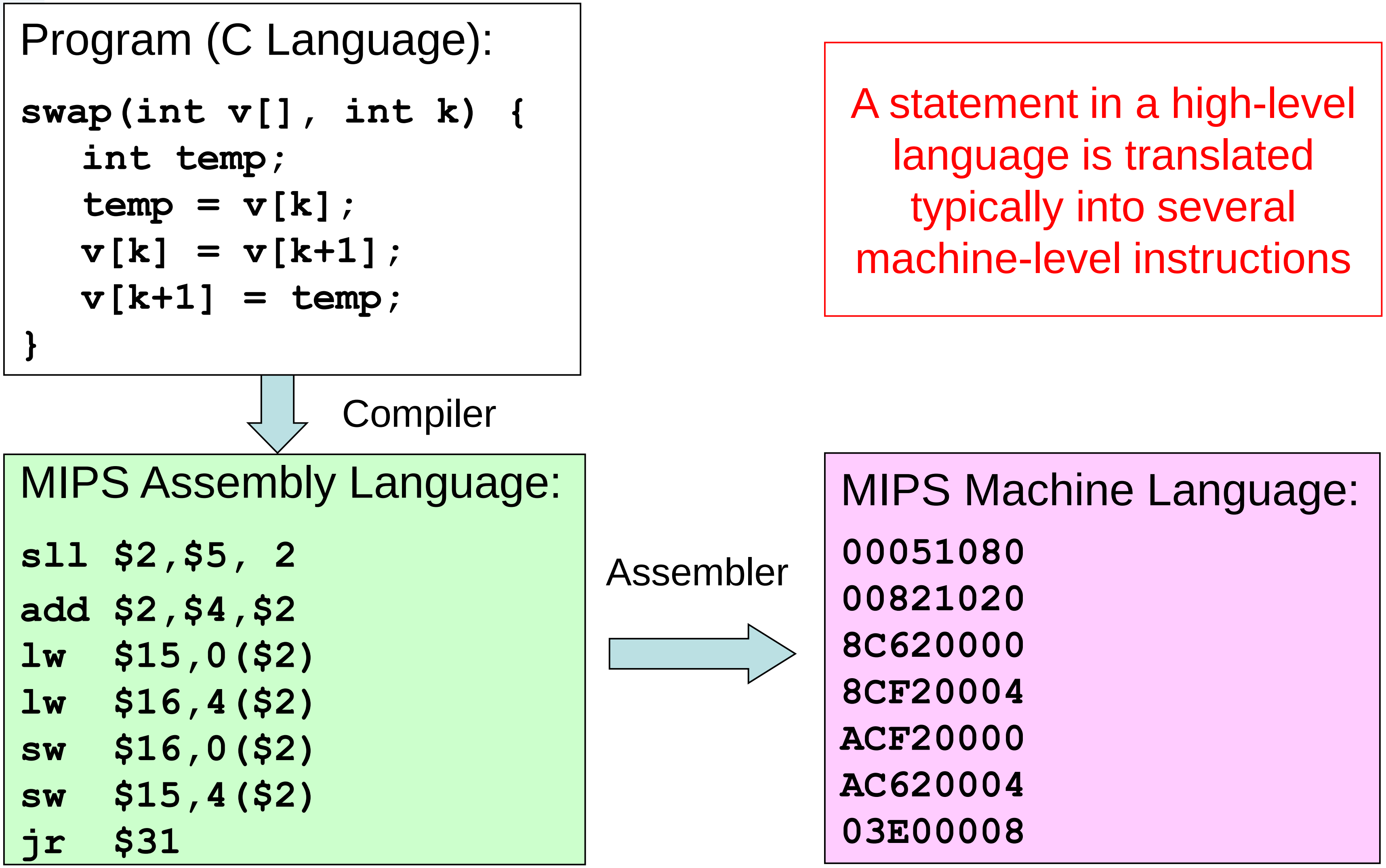
Assembly and Machine Language

- Machine language
  - Native to a processor: executed directly by hardware
  - Instructions consist of binary code: 1s and 0s
- Assembly language
  - Slightly higher-level language
  - Readability of instructions is better than machine language
  - One-to-one correspondence with machine language instructions
- Assemblers translate assembly to machine code
- Compilers translate high-level programs to machine code
  - Either directly, or
  - Indirectly via an assembler

Compiler and Assembler



Translating Languages



Program (C Language):

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

Compiler

A statement in a high-level language is translated typically into several machine-level instructions

MIPS Assembly Language:

```
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
```

Assembler

MIPS Machine Language:

```
00051080
00821020
8C620000
8CF20004
ACF20000
AC620004
03E00008
```

Advantages of High-Level Languages

- Program development is faster
  - High-level statements: fewer instructions to code
- Program maintenance is easier
  - For the same above reasons
- Programs are portable
  - Contain few machine-dependent details
    - Can be used with little or no modifications on different machines
  - Compiler translates to the target machine language
  - However, Assembly language programs are not portable

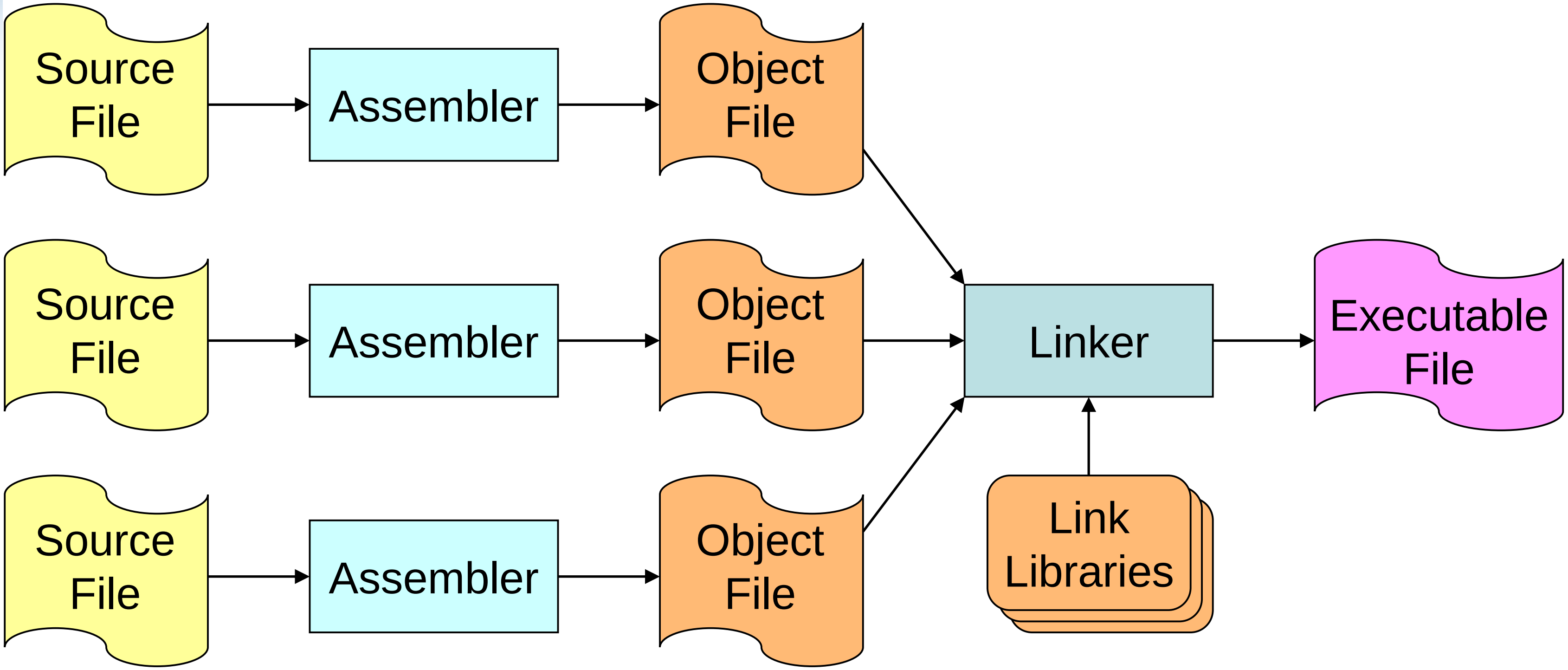
Why Learn Assembly Language?

- Many reasons:
  - Accessibility to system hardware
  - Space and time efficiency
  - Writing a compiler for a high-level language
- Accessibility to system hardware
  - Assembly Language is useful for implementing system software
  - Also useful for small embedded system applications
- Space and Time efficiency
  - Understanding sources of program inefficiency
  - Tuning program performance
  - Writing compact code

Assembly Language Programming Tools

- Editor
  - Allows you to create and edit assembly language source files
- Assembler
  - Converts assembly language programs into object files
  - Object files contain the machine instructions
- Linker
  - Combines object files created by the assembler with link libraries
  - Produces a single executable program
- Debugger
  - Allows you to trace the execution of a program
  - Allows you to view machine instructions, memory, and registers

Assemble and Link Process

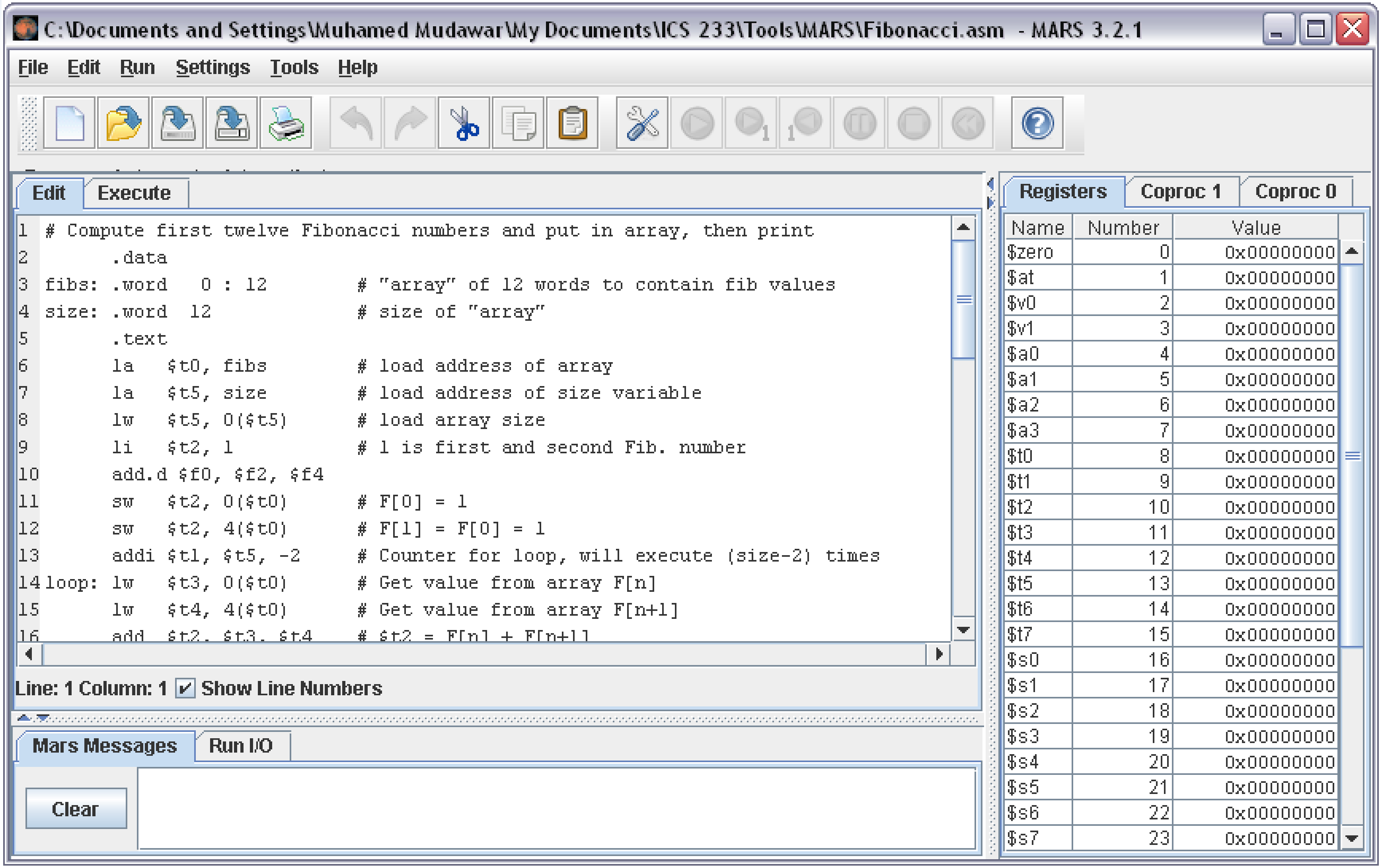


A program may consist of multiple source files

Assembler translates each source file separately into an object file

Linker links all object files together with link libraries

MARS Assembler and Simulator Tool


C:\Documents and Settings\Muhamed Mudawar\My Documents\ICS 233\Tools\MARS\Fibonacci.asm - MARS 3.2.1
File Edit Run Settings Tools Help
Edit
Execute
Registers
Coproc 1
Coproc 0
Mars Messages
Run I/O
Clear

```
# Compute first twelve Fibonacci numbers and put in array, then print
      .data
fibs: .word   0 : 12        # "array" of 12 words to contain fib values
size: .word  12             # size of "array"
      .text
      la   $t0, fibs        # load address of array
      la   $t5, size        # load address of size variable
      lw   $t5, 0($t5)      # load array size
      li   $t2, 1           # 1 is first and second Fib. number
      add.d $f0, $f2, $f4
      sw   $t2, 0($t0)      # F[0] = 1
      sw   $t2, 4($t0)      # F[1] = F[0] = 1
      addi $t1, $t5, -2     # Counter for loop, will execute (size-2) times
loop: lw   $t3, 0($t0)      # Get value from array F[n]
      lw   $t4, 4($t0)      # Get value from array F[n+1]
      add  $t2, $t3, $t4    # $t2 = F[n] + F[n+1]
```

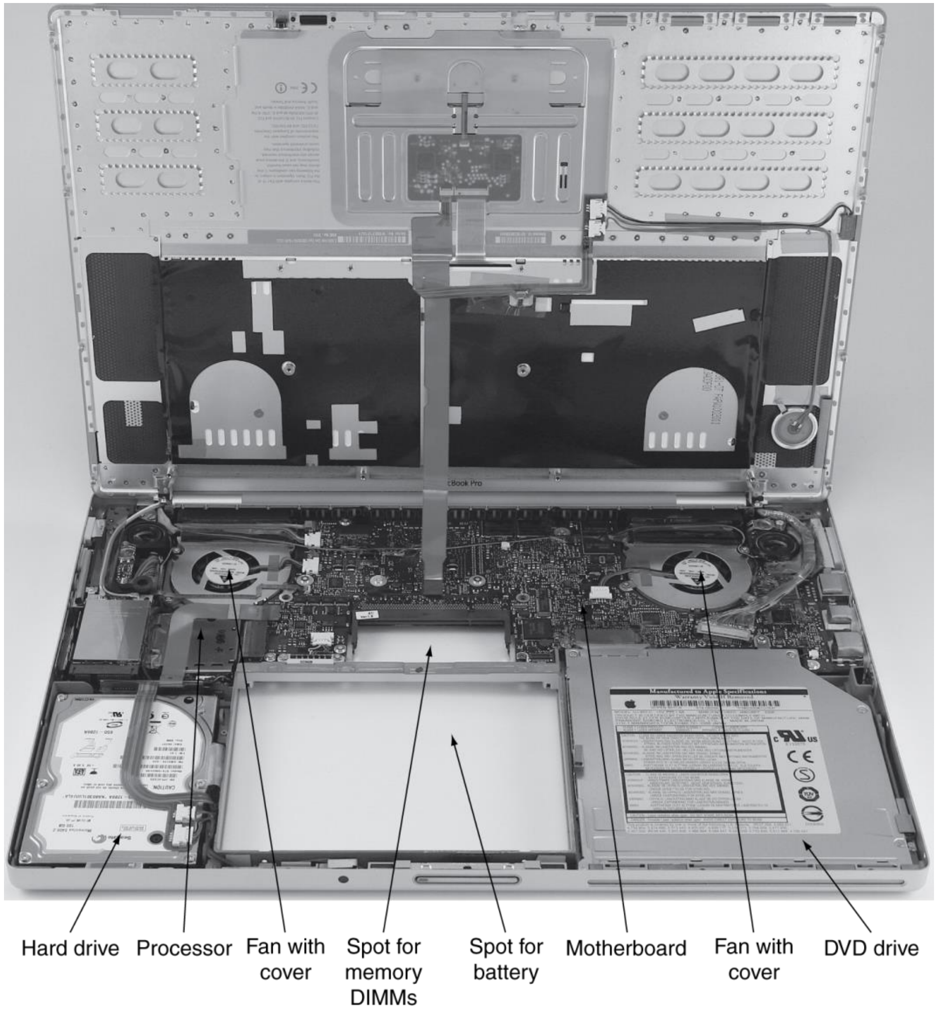
Line: 1 Column: 1 ☑ Show Line Numbers

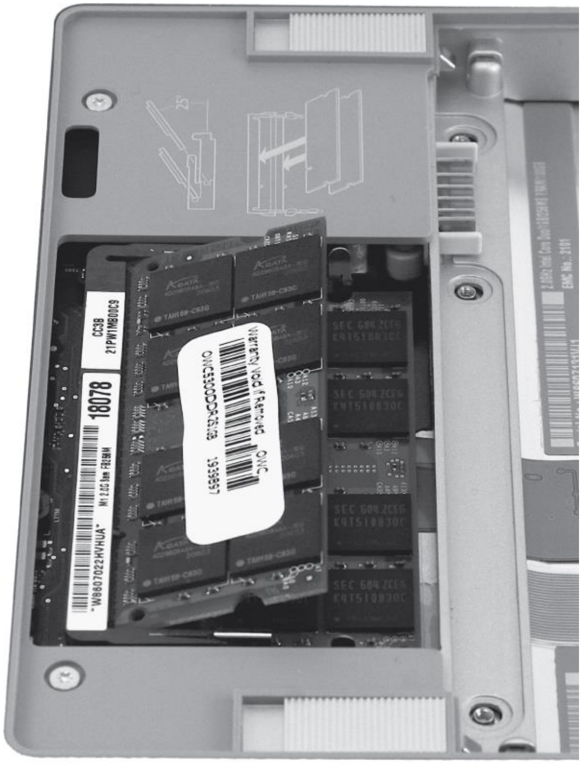
Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x00000000
$v1	3	0x00000000
$a0	4	0x00000000
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000000
$t1	9	0x00000000
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000

Presentation Outline

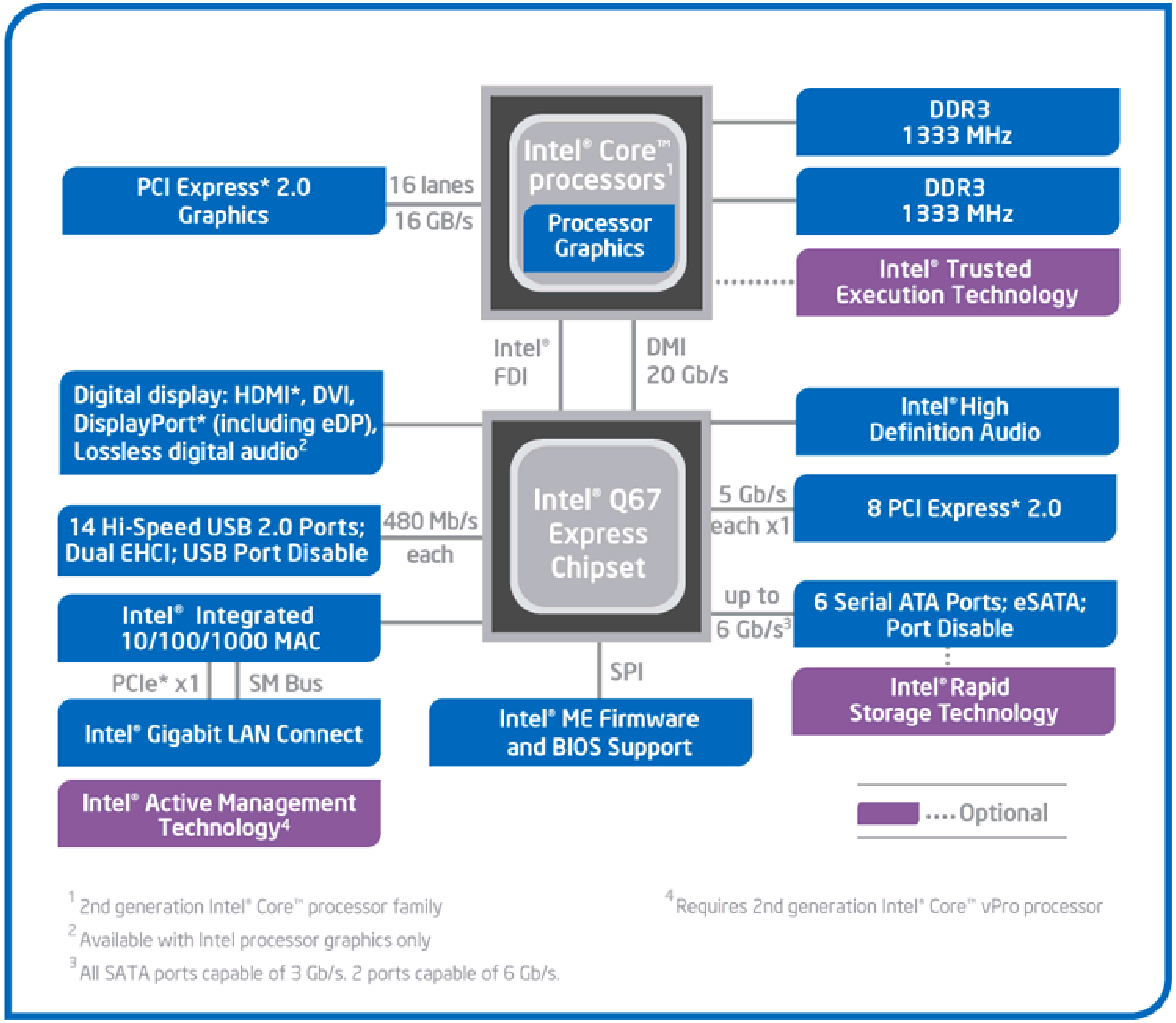
- Welcome to CA CSE
- Computer Architectures and Trends
- High-Level, Assembly-, and Machine-Languages
- Components of a Computer System
- Chip Manufacturing Process
- Programmer's View of a Computer System

Opening the Box





How do Components Connect


PCI Express* 2.0 Graphics
16 lanes
16 GB/s
Intel® Core™ processors[1]
Processor Graphics
DDR3 1333 MHz
DDR3 1333 MHz
Intel® Trusted Execution Technology
Intel® FDI
DMI 20 Gb/s
Digital display: HDMI*, DVI, DisplayPort* (including eDP), Lossless digital audio[2]
Intel® High Definition Audio
Intel® Q67 Express Chipset
5 Gb/s each x1
8 PCI Express* 2.0
14 Hi-Speed USB 2.0 Ports; Dual EHCI; USB Port Disable
480 Mb/s each
up to 6 Gb/s[3]
6 Serial ATA Ports; eSATA; Port Disable
Intel® Integrated 10/100/1000 MAC
PCIe* x1
SM Bus
SPI
Intel® Rapid Storage Technology
Intel® Gigabit LAN Connect
Intel® ME Firmware and BIOS Support
Intel® Active Management Technology[4]
▬ .... Optional

[1] 2nd generation Intel® Core™ processor family

[2] Available with Intel processor graphics only

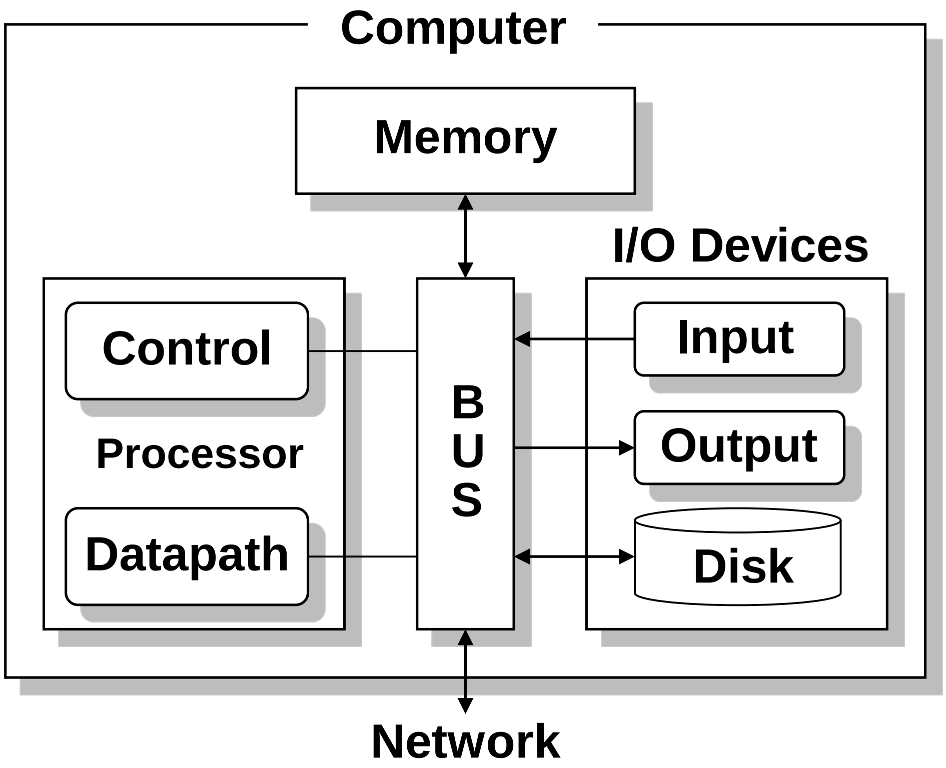
[3] All SATA ports capable of 3 Gb/s. 2 ports capable of 6 Gb/s.

[4] Requires 2nd generation Intel® Core™ vPro processor

Intel® Q67 Express Chipset Platform Block Diagram

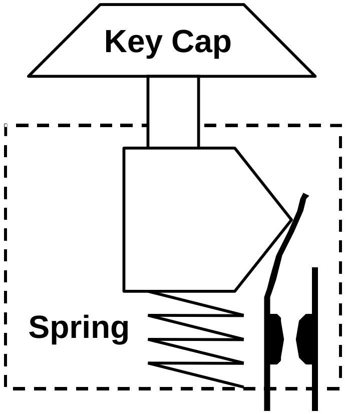
Components of a Computer System

- Processor
  - Datapath
  - Control
- Memory & Storage
  - Main Memory
  - Disk Storage
- Input devices
- Output devices

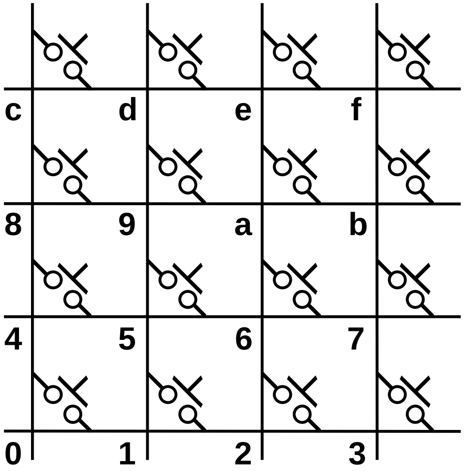


- Bus: Interconnects processor to memory and I/O
- Network: newly added component for communication

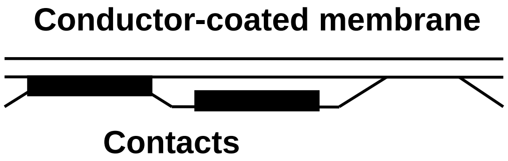
Input Devices



Mechanical switch

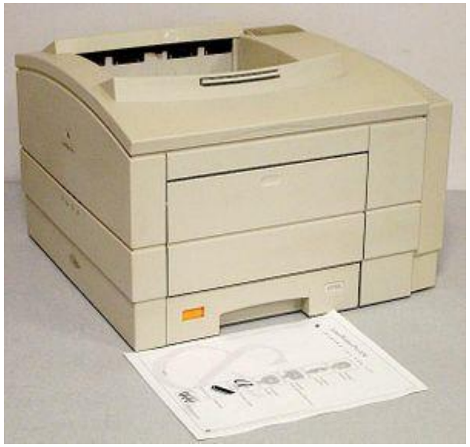


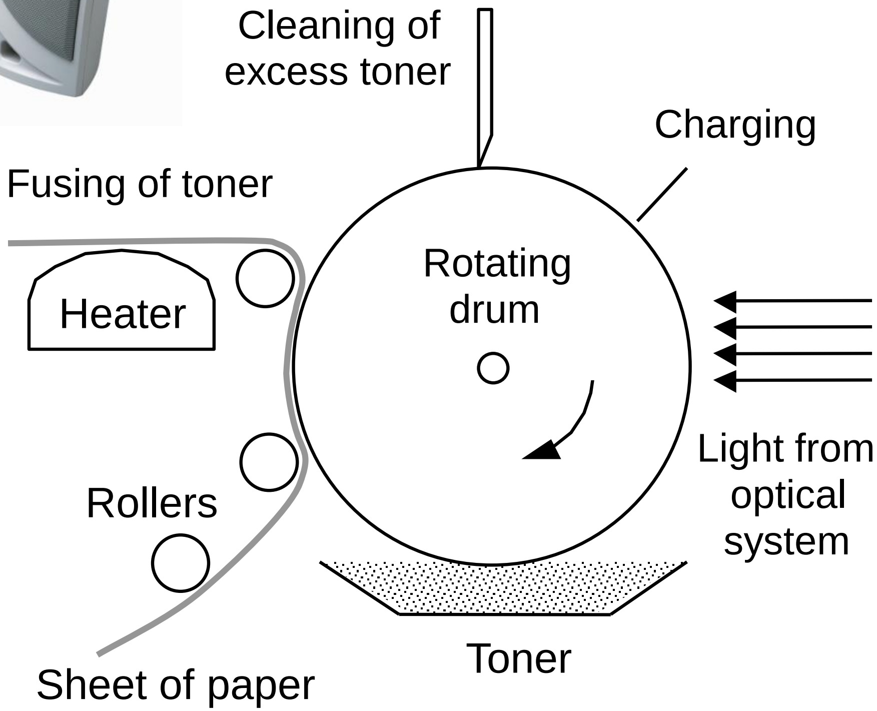
Logical arrangement of keys



Membrane switch

Output Devices





Laser printing

Memory Devices

- Volatile Memory Devices
  - RAM = Random Access Memory
  - DRAM = Dynamic RAM
    - 1-Transistor cell + capacitor
    - Dense but slow, must be refreshed
    - Typical choice for main memory
  - SRAM: Static RAM
    - 6-Transistor cell, faster but less dense than DRAM
    - Typical choice for cache memory
- Non-Volatile Memory Devices
  - ROM = Read Only Memory
  - Flash Memory

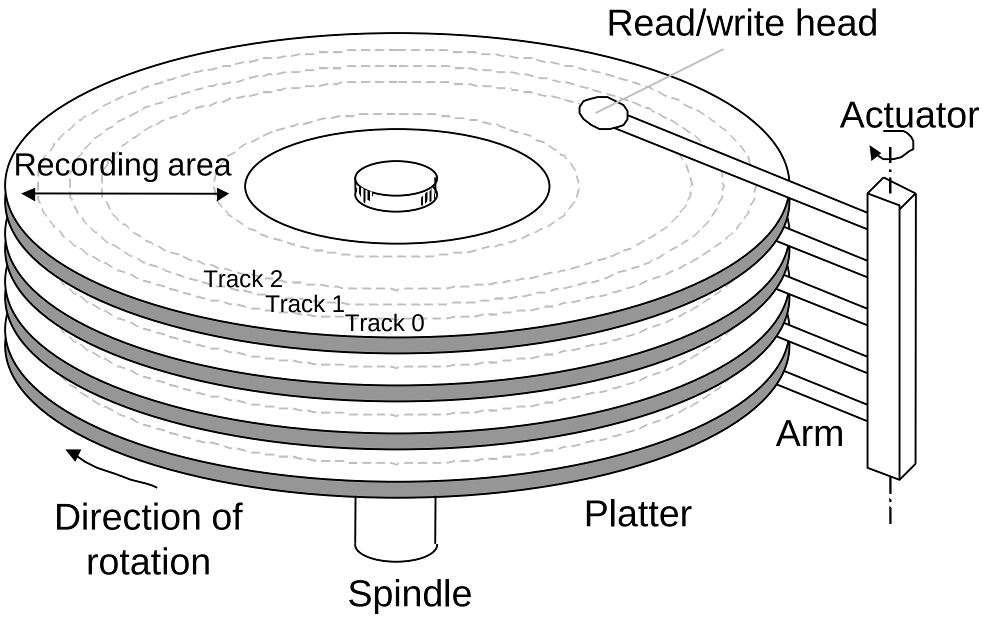
Magnetic Disk Storage

A Magnetic disk consists of a collection of platters

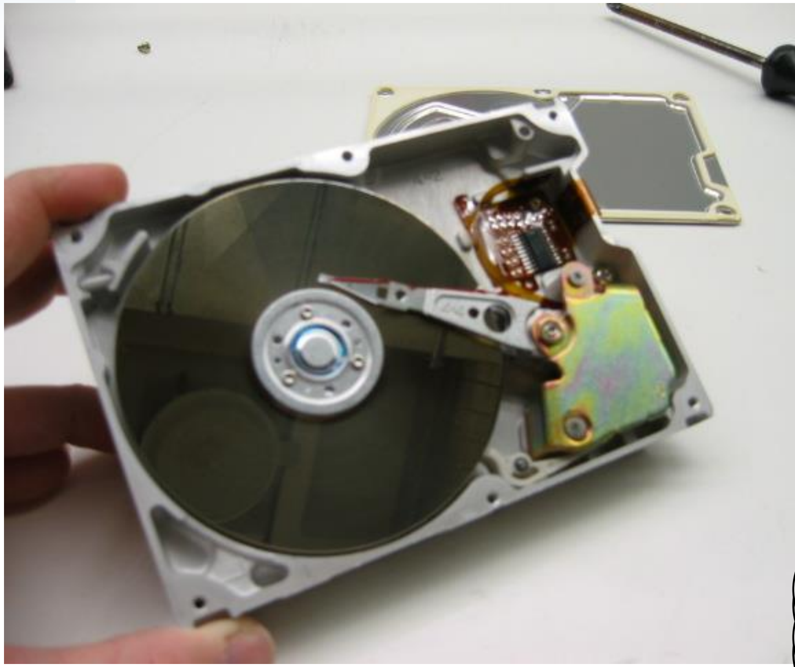
Provides a number of recording surfaces

Arm provides read/write heads for all surfaces

The disk heads are connected together and move in conjunction



Magnetic Disk Storage

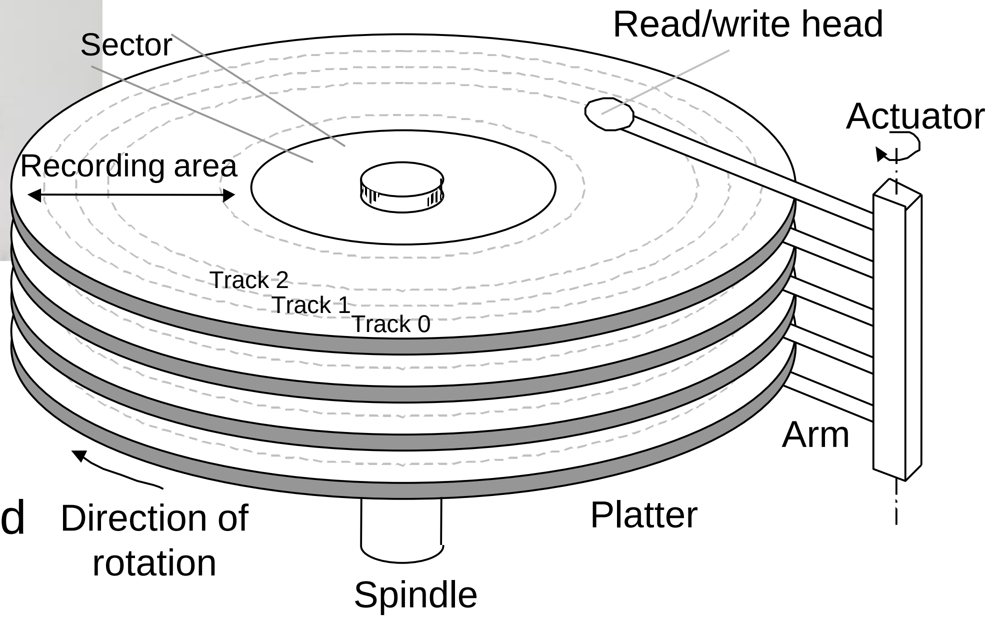


Disk Access Time =

Seek Time +

Rotation Latency +

Transfer Time

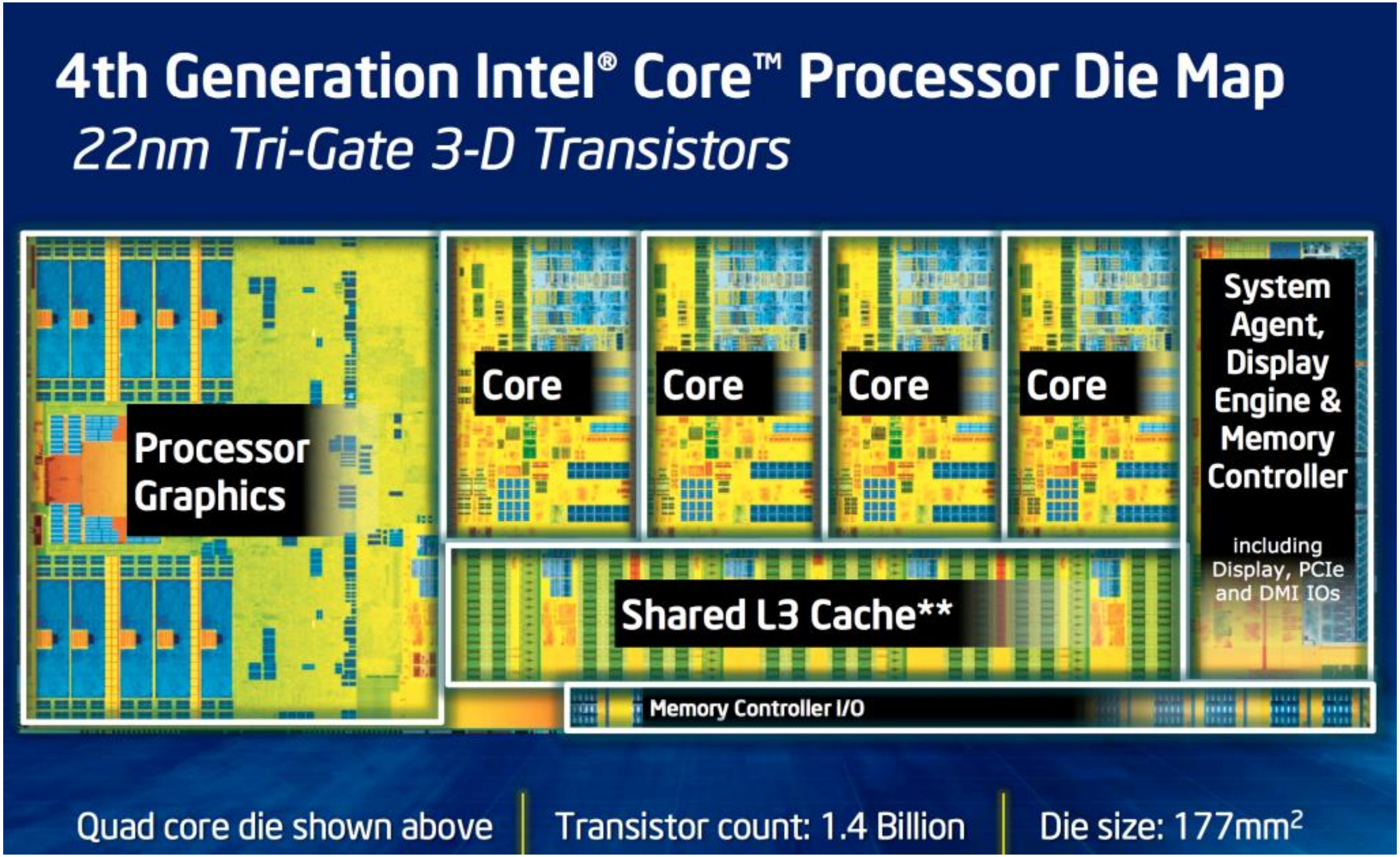


Seek Time: head movement to the desired track (milliseconds)

Rotation Latency: disk rotation until desired sector arrives under the head

Transfer Time: to transfer data

Inside the Processor (CPU)



4th Generation Intel® Core™ Processor Die Map

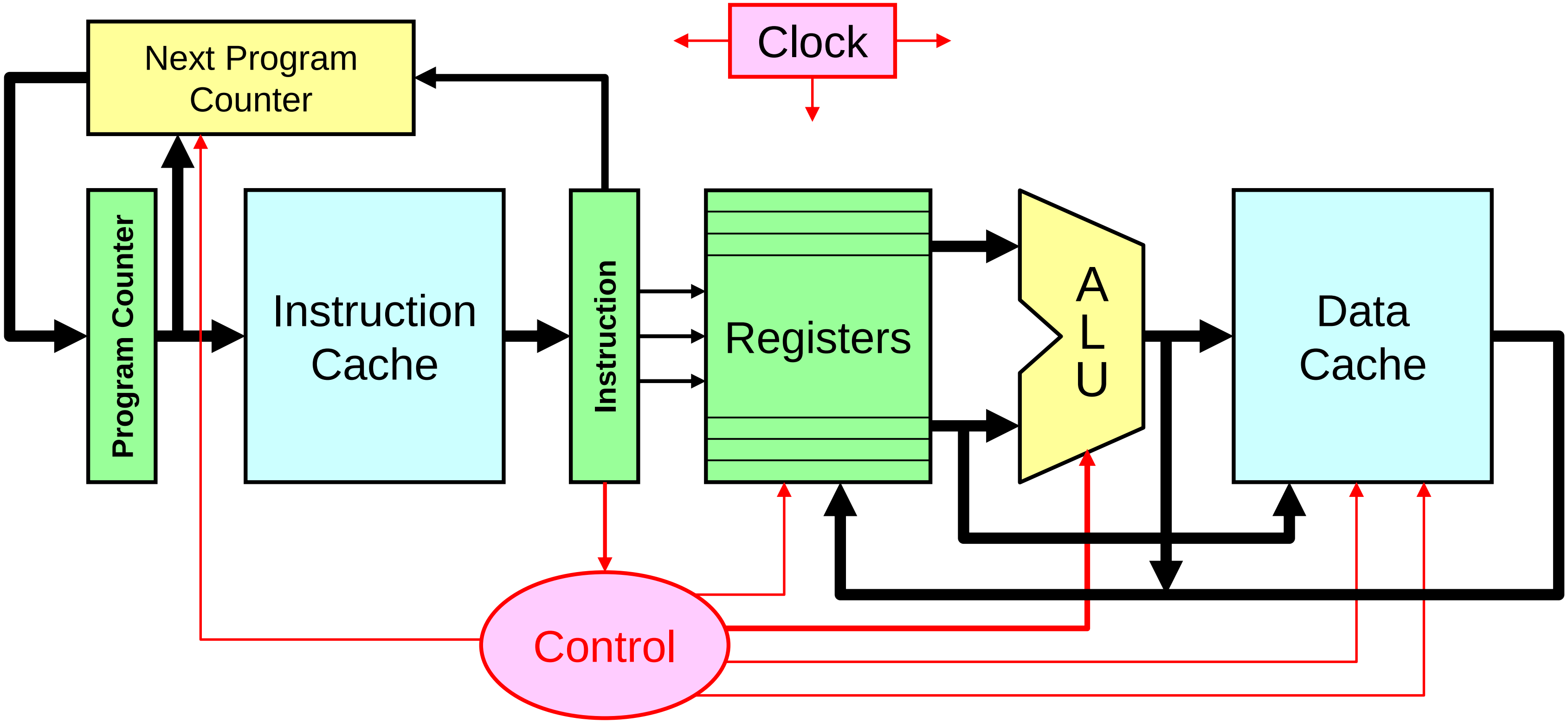
22nm Tri-Gate 3-D Transistors

Processor Graphics
Core
Core
Core
Core
System Agent, Display Engine & Memory Controller
including Display, PCIe and DMI IOs
Shared L3 Cache**
Memory Controller I/O

Quad core die shown above | Transistor count: 1.4 Billion | Die size: 177mm²

Inside the Processor (CPU)

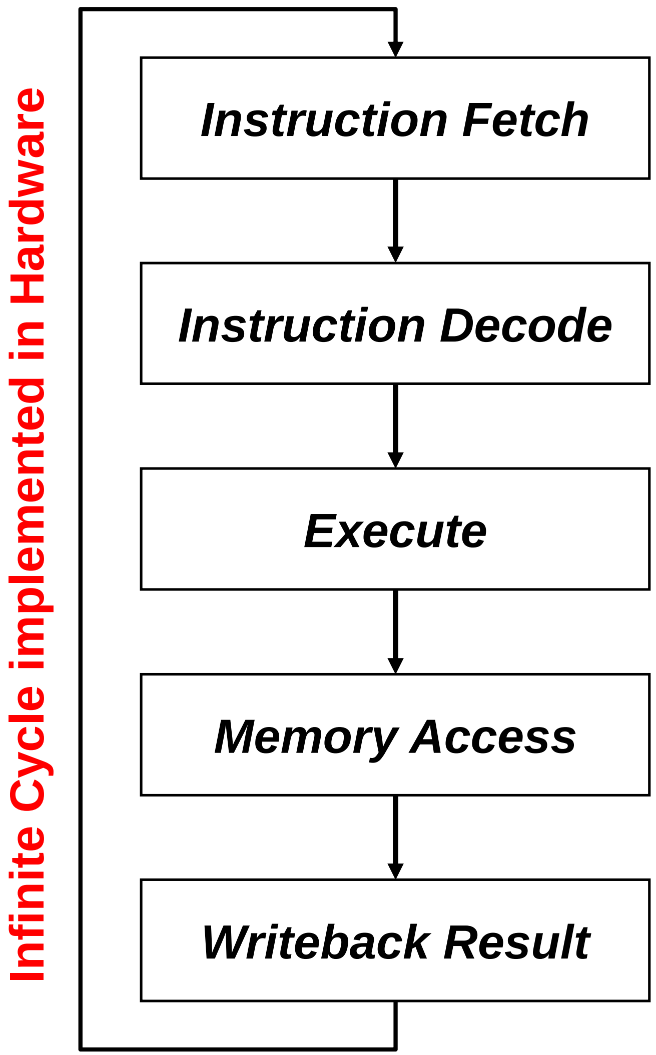
- Datapath: part of a processor that executes instructions
- Control: generates control signals for each instruction



Datapath Components

- Program Counter (PC)
  - Contains address of instruction to be fetched
  - Next Program Counter: computes address of next instruction
- Instruction and Data Caches
  - Small and fast memory containing most recent instructions/data
- Register File
  - General-purpose registers used for intermediate computations
- ALU = Arithmetic and Logic Unit
  - Executes arithmetic and logic instructions
- Buses
  - Used to wire and interconnect the various components

Fetch - Execute Cycle



Infinite Cycle implemented in Hardware

Instruction Fetch

Instruction Decode

Execute

Memory Access

Writeback Result

Fetch instruction

Compute address of next instruction

Generate control signals for instruction

Read operands from registers

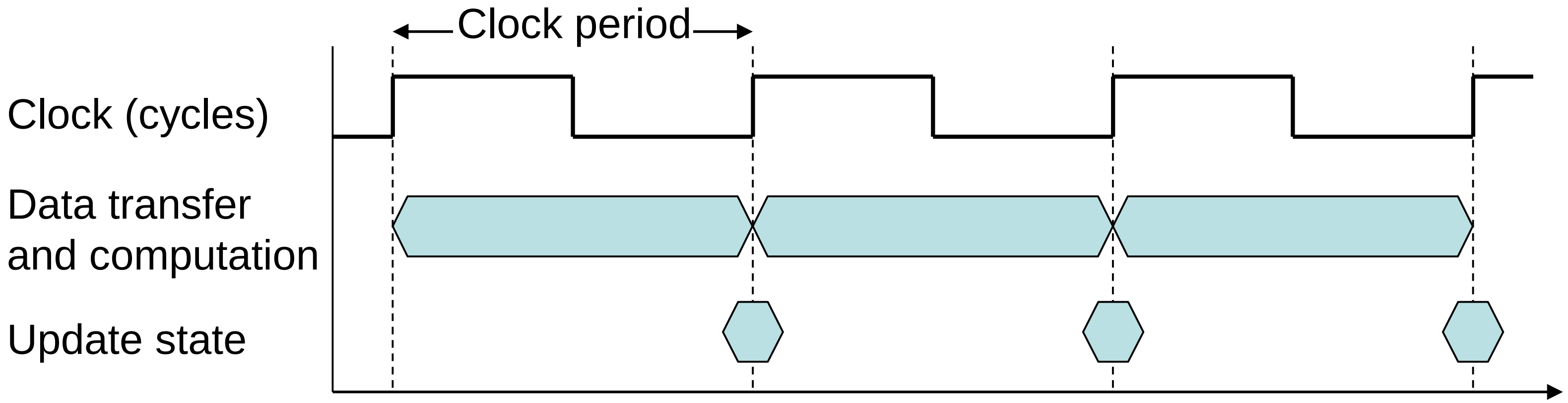
Compute result value

Read or write memory (load/store)

Writeback result in a register

Clocking

Operation of digital hardware is governed by a clock

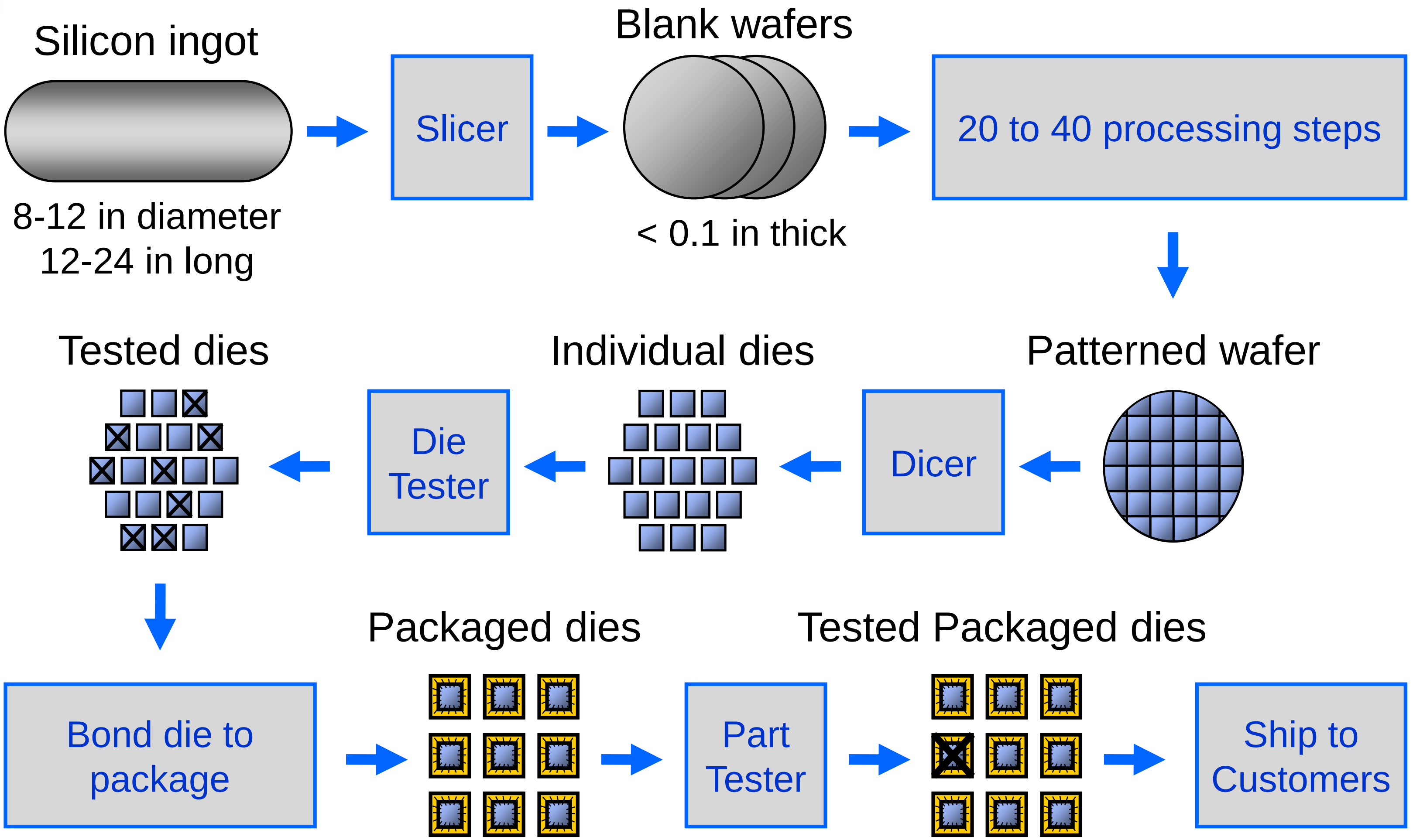


- Clock period: duration of a clock cycle
  - e.g., 250 ps = 0.25 ns = 0.25 $\times 10^{-9}$ sec
- Clock frequency (rate) = 1 / clock period
  - e.g., 1/ 0.25 $\times 10^{-9}$ sec = 4.0$\times 10^{9}$ Hz = 4.0 GHz

Presentation Outline

- Welcome to CA CSE
- Computer Architectures and Trends
- High-Level, Assembly-, and Machine-Languages
- Components of a Computer System
- Chip Manufacturing Process
- Programmer's View of a Computer System

Chip Manufacturing Process



Silicon ingot

8-12 in diameter
12-24 in long

Slicer

Blank wafers

< 0.1 in thick

20 to 40 processing steps

Patterned wafer

Dicer

Individual dies

Die Tester

Tested dies

Bond die to package

Packaged dies

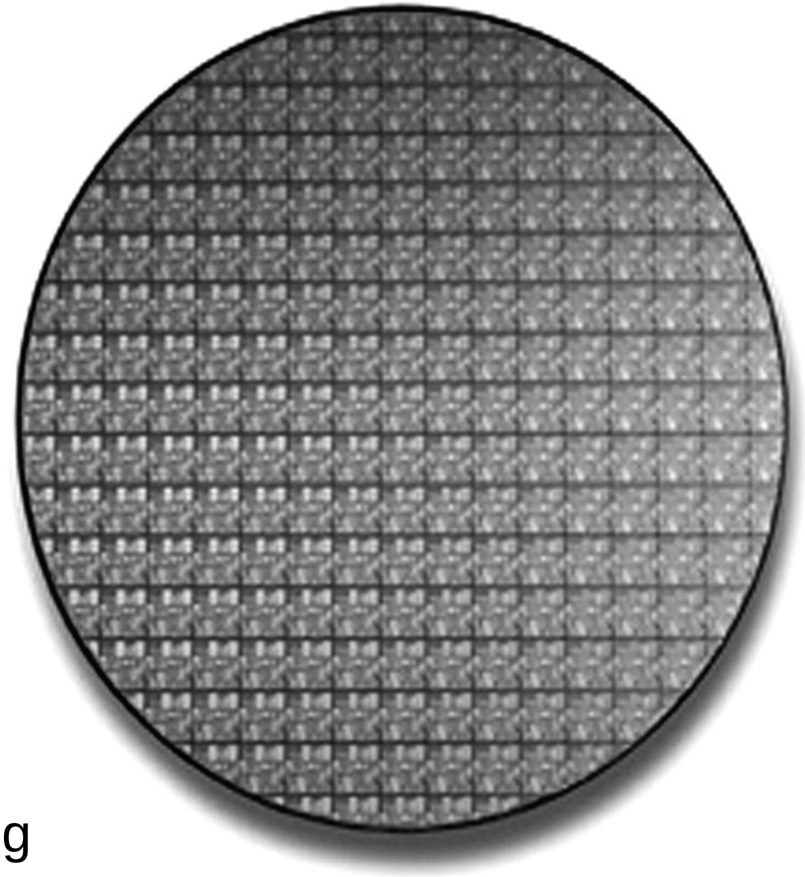
Part Tester

Tested Packaged dies

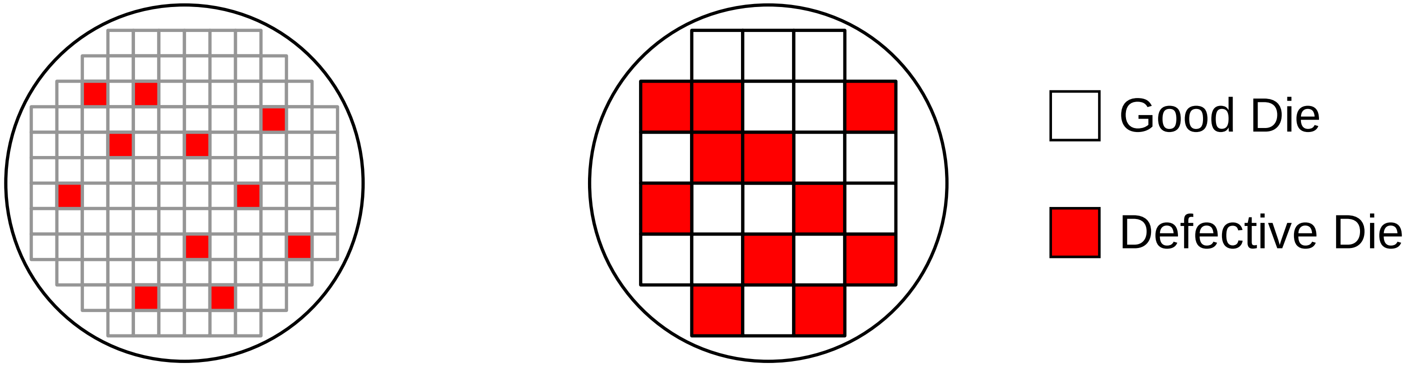
Ship to Customers

Wafer of Pentium 4 Processors

- 8 inches (20 cm) in diameter
- Die area is 250 mm^2
  - About 16 mm per side
- 55 million transistors per die
  - 0.18 µm technology
  - Size of smallest transistor
  - Improved technology uses
    - 0.13 µm and 0.09 µm
- Dies per wafer = 169
  - When yield = 100%
  - Number is reduced after testing
  - Rounded dies at boundary are useless



Effect of Die Size on Yield



120 dies, 109 good

26 dies, 15 good

Dramatic decrease in yield with larger dies

Yield = (Number of Good Dies) / (Total Number of Dies)

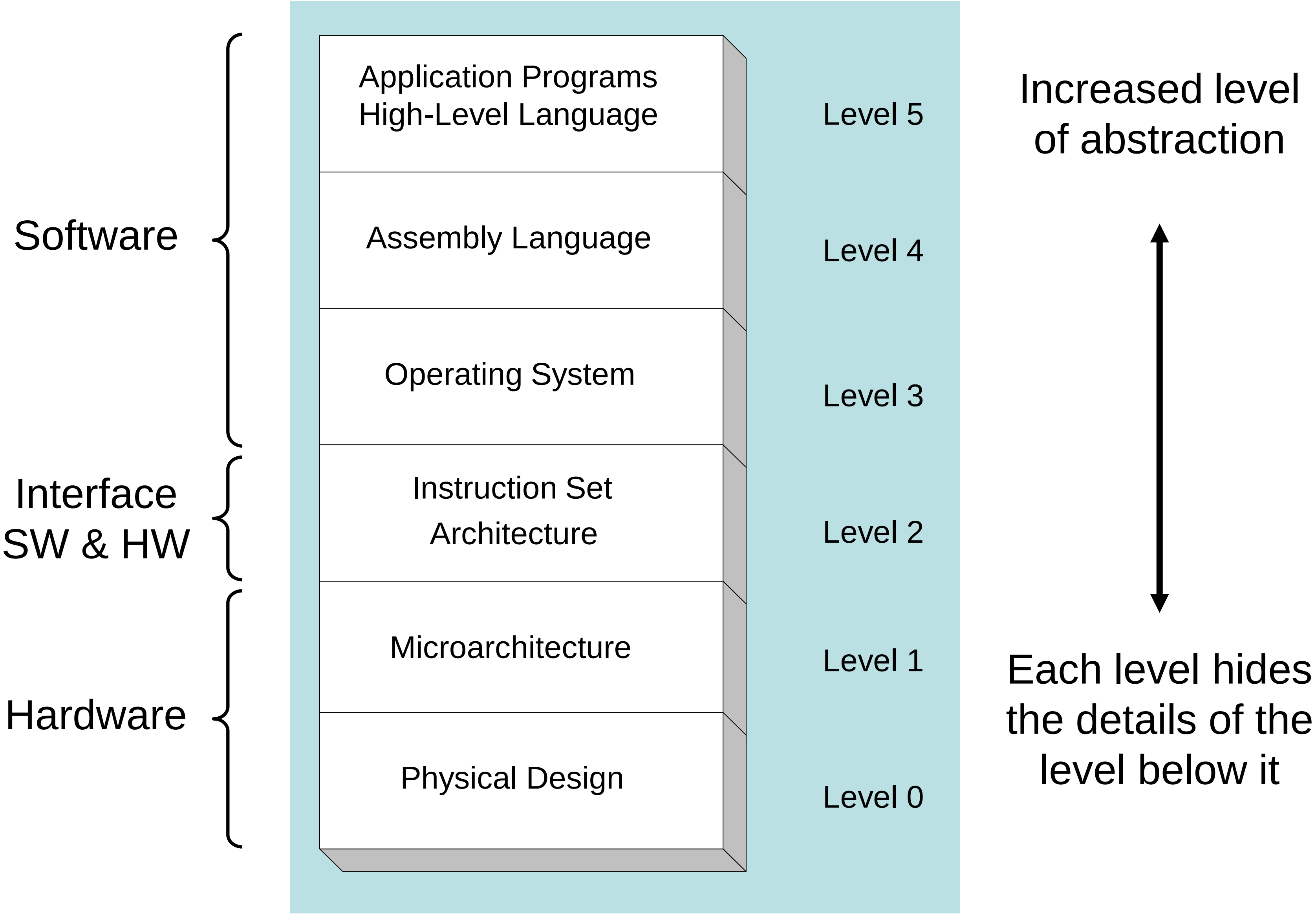
$$\text{Yield} = \frac{1}{(1 + (\text{Defect per area} \times \text{Die area} / 2))^2}$$

Die Cost = (Wafer Cost) / (Dies per Wafer × Yield)

Presentation Outline

- Welcome to CA CSE
- Computer Architectures and Trends
- High-Level, Assembly-, and Machine-Languages
- Components of a Computer System
- Chip Manufacturing Process
- Programmer's View of a Computer System

Programmer's View of a Computer System



Software

Interface
SW & HW

Hardware

Application Programs
High-Level Language
Level 5

Assembly Language
Level 4

Operating System
Level 3

Instruction Set
Architecture
Level 2

Microarchitecture
Level 1

Physical Design
Level 0

Increased level
of abstraction

Each level hides
the details of the
level below it

Programmer's View - 2

- Application Programs (Level 5)
  - Written in high-level programming languages
  - Such as Java, C++, Pascal, Visual Basic . . .
  - Programs compile into assembly language level (Level 4)
- Assembly Language (Level 4)
  - Instruction mnemonics are used
  - Have one-to-one correspondence to machine language
  - Calls functions written at the operating system level (Level 3)
  - Programs are translated into machine language (Level 2)
- Operating System (Level 3)
  - Provides services to level 4 and 5 programs
  - Translated to run at the machine instruction level (Level 2)

Programmer's View - 3

- Instruction Set Architecture (Level 2)
  - Interface between software and hardware
  - Specifies how a processor functions
  - Machine instructions, registers, and memory are exposed
  - Machine language is executed by Level 1 (microarchitecture)
- Microarchitecture (Level 1)
  - Controls the execution of machine instructions (Level 2)
  - Implemented by digital logic
- Physical Design (Level 0)
  - Implements the microarchitecture
  - Physical layout of circuits on a chip