

Chapter 1

Computer Abstractions and Technology

Dr. Yonshik Choi
Illinois Institute of Technology

Introduction

- This course is all about how computers work

- We know how a computer work...

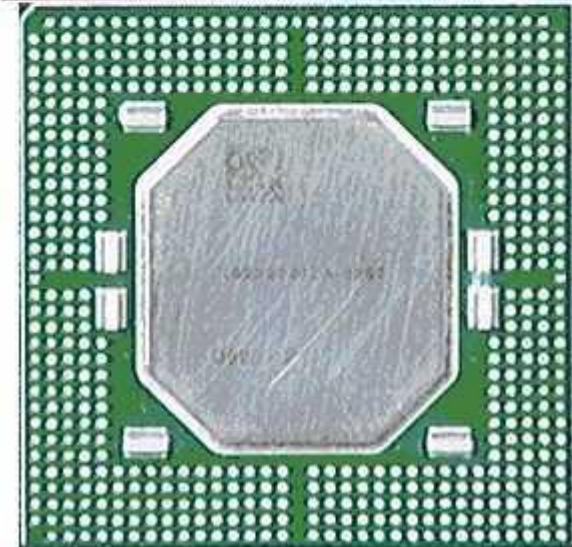
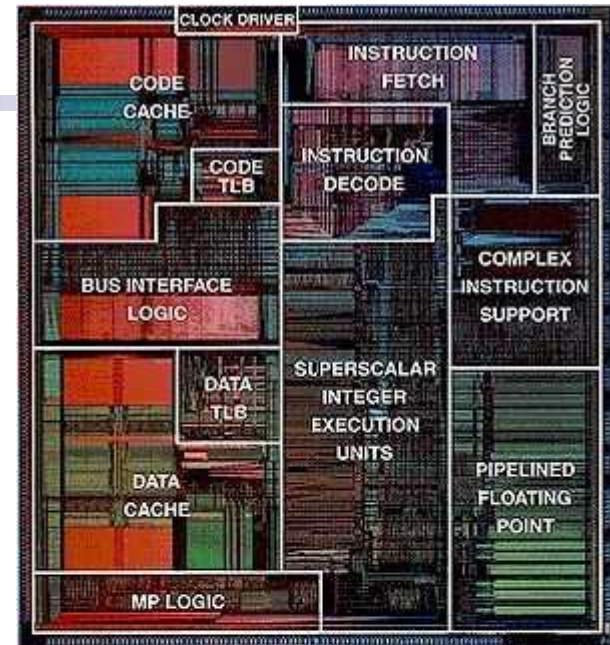
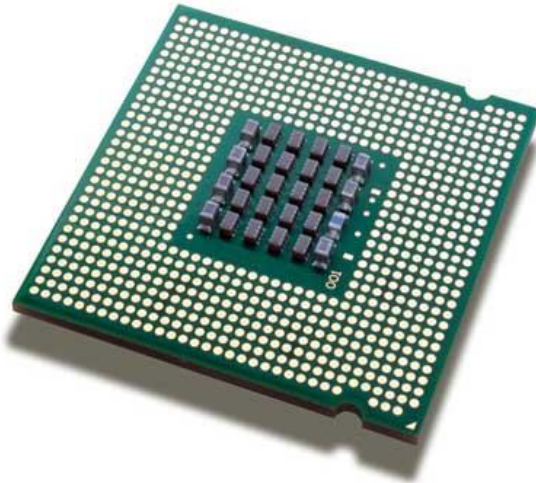
1. Power on
2. System checking
3. Load/start operating system (MS Window, Linux, Unix, etc...)
4. Load/start an application software
5. Application software works with OS and OS works with HW
6. Job is done!

Introduction

- When you say a computer, what do you need?
 - Hardware - computer
 - Software – applications
 - Computing system – HW & SW
- HW – items you can touch
- SW – programs to control CPU/HW and get result
- Program – rules to command CPU
- CPU – Central Processing Unit

Introduction

CPU



Introduction

- All computers are not the same.
- Differences of computers?
 - **Different types:** desktop, servers, mainframes, minicomputers, supercomputers, embedded computers/devices, ...
 - **Different uses:** automobiles, graphics, finance, genomics, web, smart phones...

Introduction

- (...continued)
 - **Different manufacturers:** Intel, Apple, IBM, Microsoft, Sun, DEC, Cray, SGI, HP, Samsung, Fusitsu, Hitachi...
 - **Different underlying technologies and different costs!**
Single core, Dual cores, Triple cores,
Quad cores, Six cores...

Introduction

- Analogy: Consider a course on “automotive vehicles”
 - Many similarities from vehicle to vehicle (e.g., wheels)
 - Huge differences from vehicle to vehicle (e.g., gas vs. electric)

Introduction

- Best way to learn:
 - Focus on a specific instance and learn how it works
 - While learning general principles and historical perspectives

Why learn this stuff?

- You want to call yourself a “computer scientist”
- You want to build software people use (need performance)
- Understand the fundamental concept how a general computing system works
- Advancing Software

Why learn this stuff?

- You want to call yourself a “computer scientist”
- You want to build software people use (need performance)
- Understand **the fundamental concept how a general computing system works**
- Advancing Software

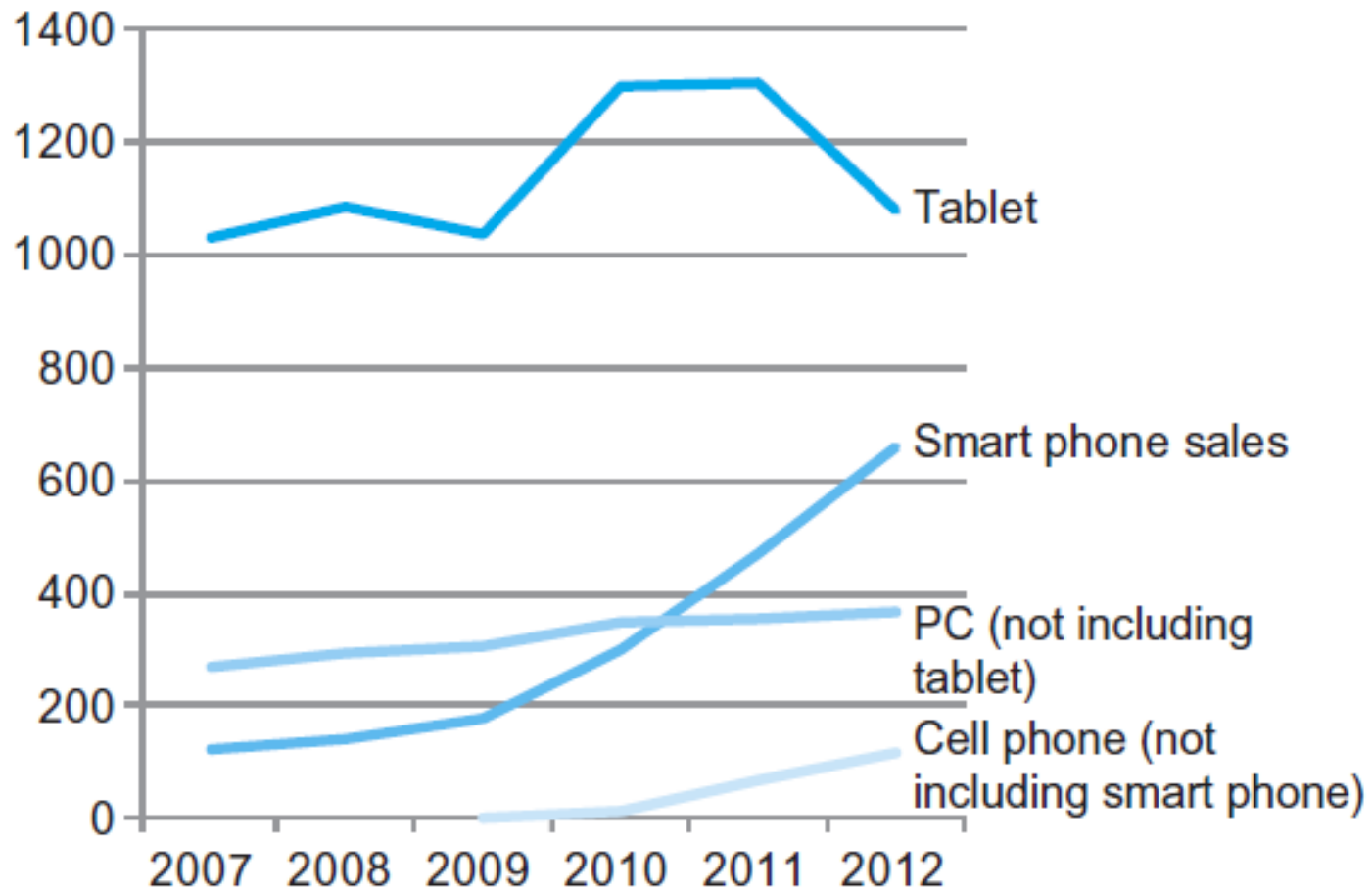
The Computer Revolution

- Progress in computer technology
 - Underpinned by Moore's Law
- Makes novel applications feasible
 - Computers in automobiles
 - Cell phones
 - Human genome project
 - World Wide Web
 - Search Engines
 - AI secretaries
- Computers are pervasive

Classes of Computers

- Desktop 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

The PostPC Era



The PostPC Era

- Personal Mobile Device (PMD)
 - Battery operated
 - Connects to the Internet
 - Hundreds of dollars
 - Smart phones, tablets, electronic glasses
- Cloud computing
 - Warehouse Scale Computers (WSC)
 - Software as a Service (SaaS)
 - Portion of software run on a PMD and a portion run in the Cloud
 - Amazon and Google

Eight Great Ideas

- Design for **Moore's Law**
- Use **abstraction** to simplify design
- Make the **common case fast**
- Performance via **parallelism**
- Performance via **pipelining**
- Performance via **prediction**
- **Hierarchy** of memories
- **Dependability** via redundancy



What You Will Learn

- How programs are translated into the machine language
 - And how the hardware executes them
- The hardware/software interface
- What determines program performance
 - And how it can be improved
- How hardware designers improve performance
- What is parallel processing

CPU Operation

- Fundamental operations of a processor:
 - Fetch
 - Decode
 - Execute
 - Writeback

CPU Operation

- Fetch – Involves retrieving an instruction (program code) from program memory
 - Modern computers are stored-program computer
 - The location in program memory is determined by a PC (program counter)
 - The PC keeps track of CPU's place in the current program

CPU Operation

- Decode – An instruction from program memory tells CPU what to do... In decode step, a part of instruction, opcode, tells what type of instruction is.. And other parts is used for operands

e.g. Human understand $A + B$, $A - C$, ...

CPU understands,

{Add opcode} A, B

{Subtrace opcode} A, C

CPU Operation

- Decode :

CPU understands,

{Add opcode} A, B

{Subtrace opcode} A, C

But still these examples are human way.

Machine only understands binary format.

Instruction is binary format language.

CPU Operation

- Execute – various desired operations are performed by after fetch step

e.g. if an addition operation is requested, an ALU (arithmetic Logical Unit) works inputs and output

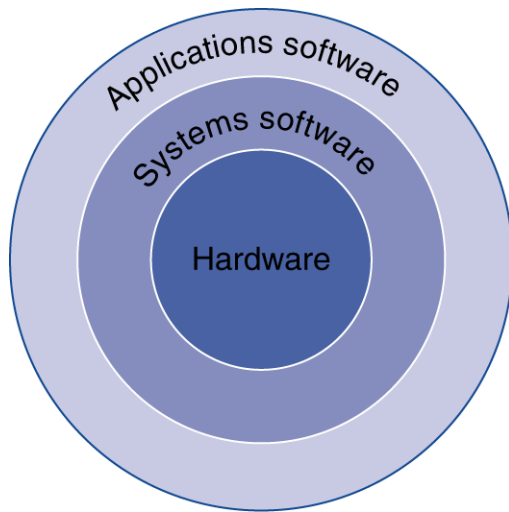
CPU Operation

- Writeback – store executed result to some form of memory
- To CPU internal memory (register) so next instruction uses the result
- Or to slow but cheaper storage, main memory, etc., so later use

Understanding Performance

- Algorithm
 - Determines number of operations executed
- Programming language, compiler, architecture
 - Determine number of machine instructions executed per operation
- Processor and memory system
 - Determine how fast instructions are executed
- I/O system (including OS)
 - Determines how fast I/O operations are executed

Below Your Program



- 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

- 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;
}
```

Compiler

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
```

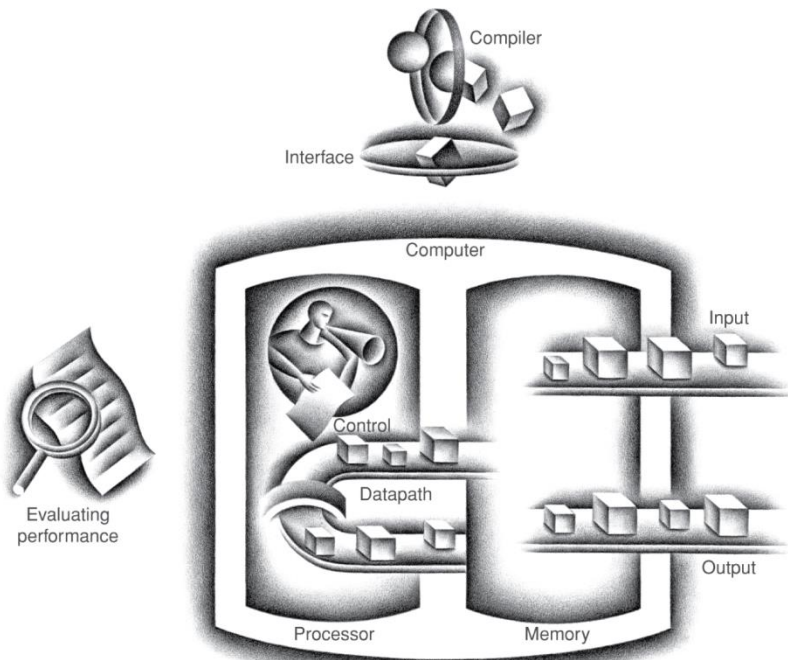
Assembler

Binary machine
language
program
(for MIPS)

```
000000001010000100000000000011000
000000000000110000001100000100001
100011000110001000000000000000000
100011001111001000000000000000100
101011001111001000000000000000000
101011000110001000000000000000100
00000011111000000000000000001000
```

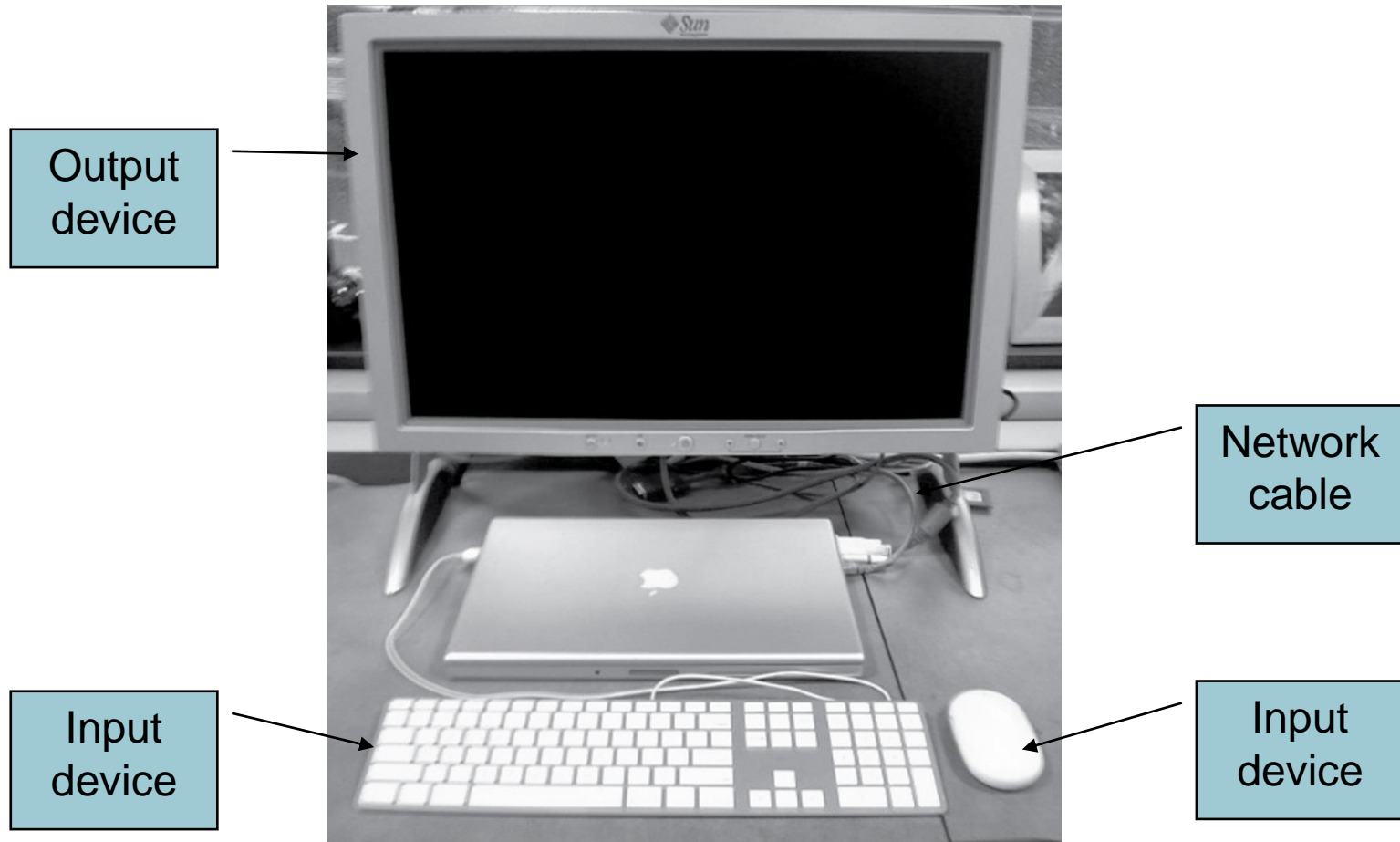
Components of a Computer

The BIG Picture



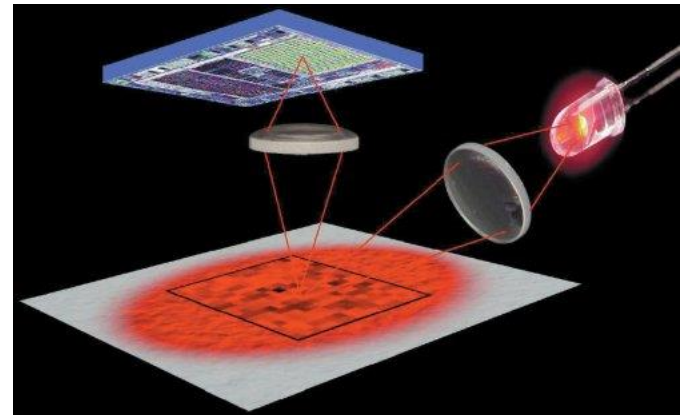
- Same components for all kinds of computer
 - Desktop, server, embedded
- Input/output includes
 - User-interface devices
 - Display, keyboard, mouse
 - Storage devices
 - Hard disk, CD/DVD, flash
 - Network adapters
 - For communicating with other computers

Anatomy of a Computer



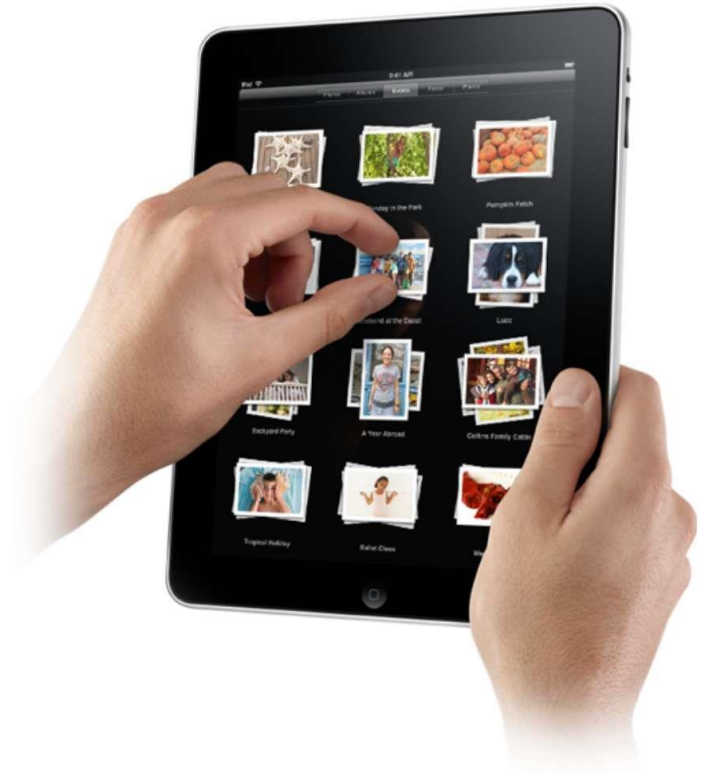
Anatomy of a Mouse

- Optical mouse
 - LED illuminates desktop
 - Small low-res camera
 - Basic image processor
 - Looks for x, y movement
 - Buttons & wheel
- Supersedes roller-ball mechanical mouse



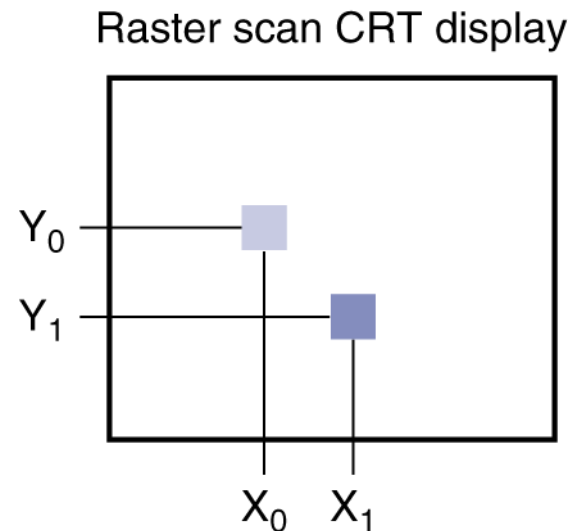
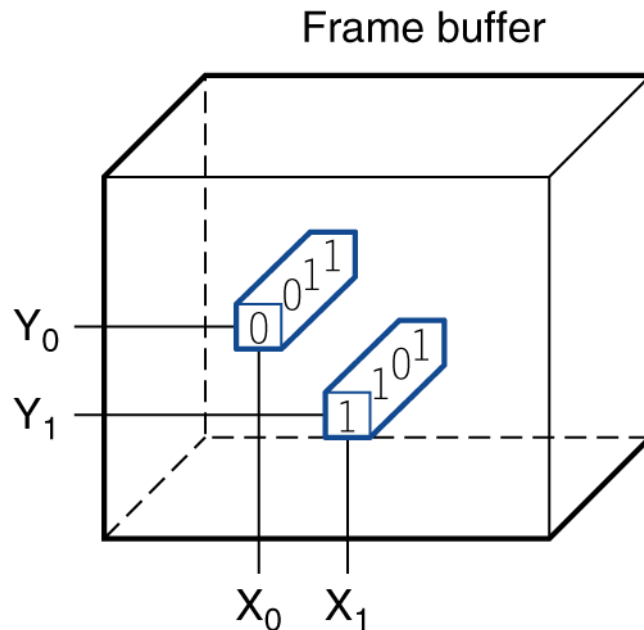
Touchscreen

- PostPC device
- Supersedes keyboard and mouse
- Resistive and Capacitive types
 - Most tablets, smart phones use capacitive
 - Capacitive allows multiple touches simultaneously

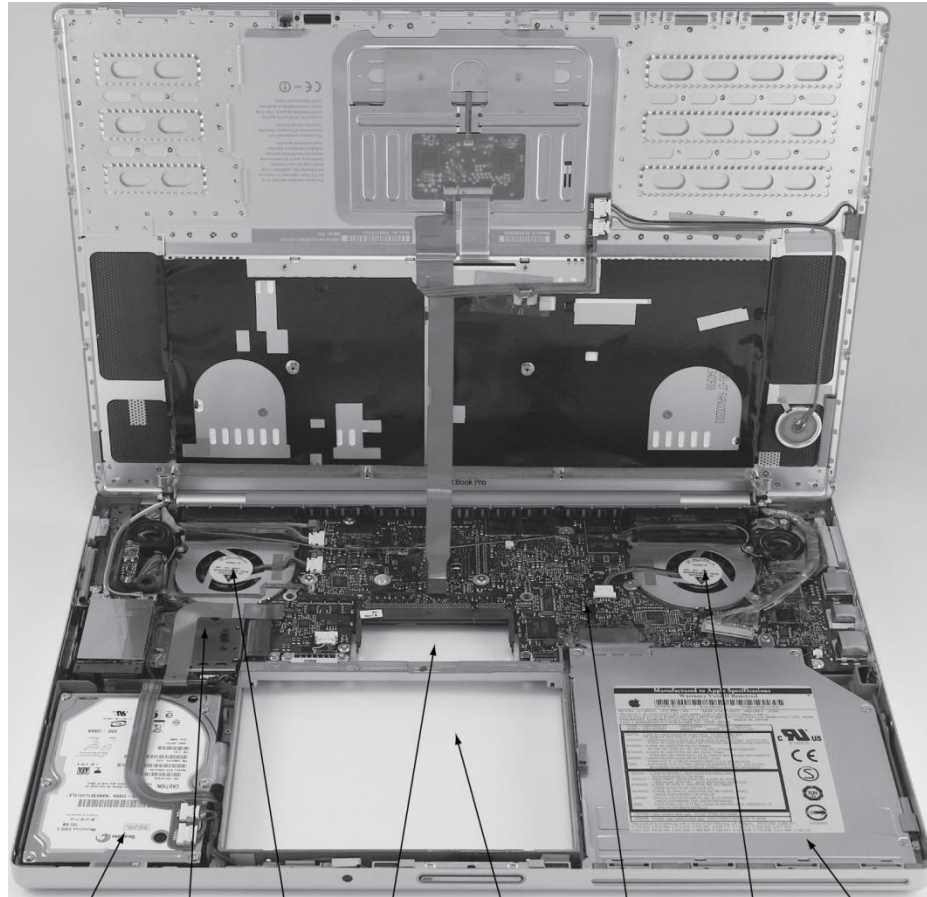


Through the Looking Glass

- LCD screen: picture elements (pixels)
 - Mirrors content of frame buffer memory



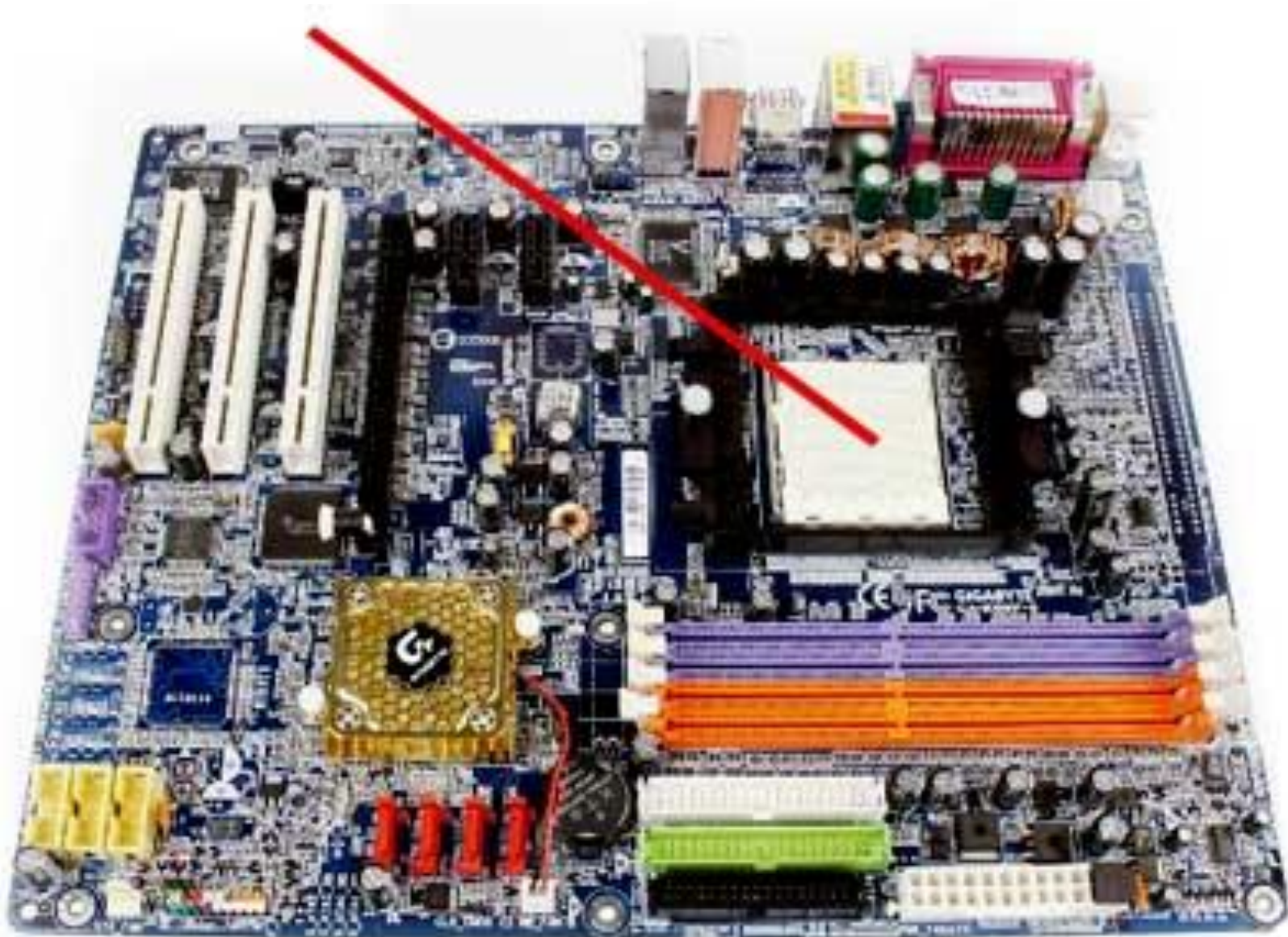
Opening the Box



Hard drive Processor Fan with cover Spot for memory DIMMs Spot for battery Motherboard Fan with cover DVD drive cover



CPU



Socket A Connector

Processor Heat Sensor

Back Panel
Connector

DDR
Memory Slots

VIA KT400
Northbridge
Chipset

ATX Power
Connector

CD-In Header

Front Panel

Audio Header

AGP 8X Slot

32-bit PCI Slots

CNR Slot

IDE

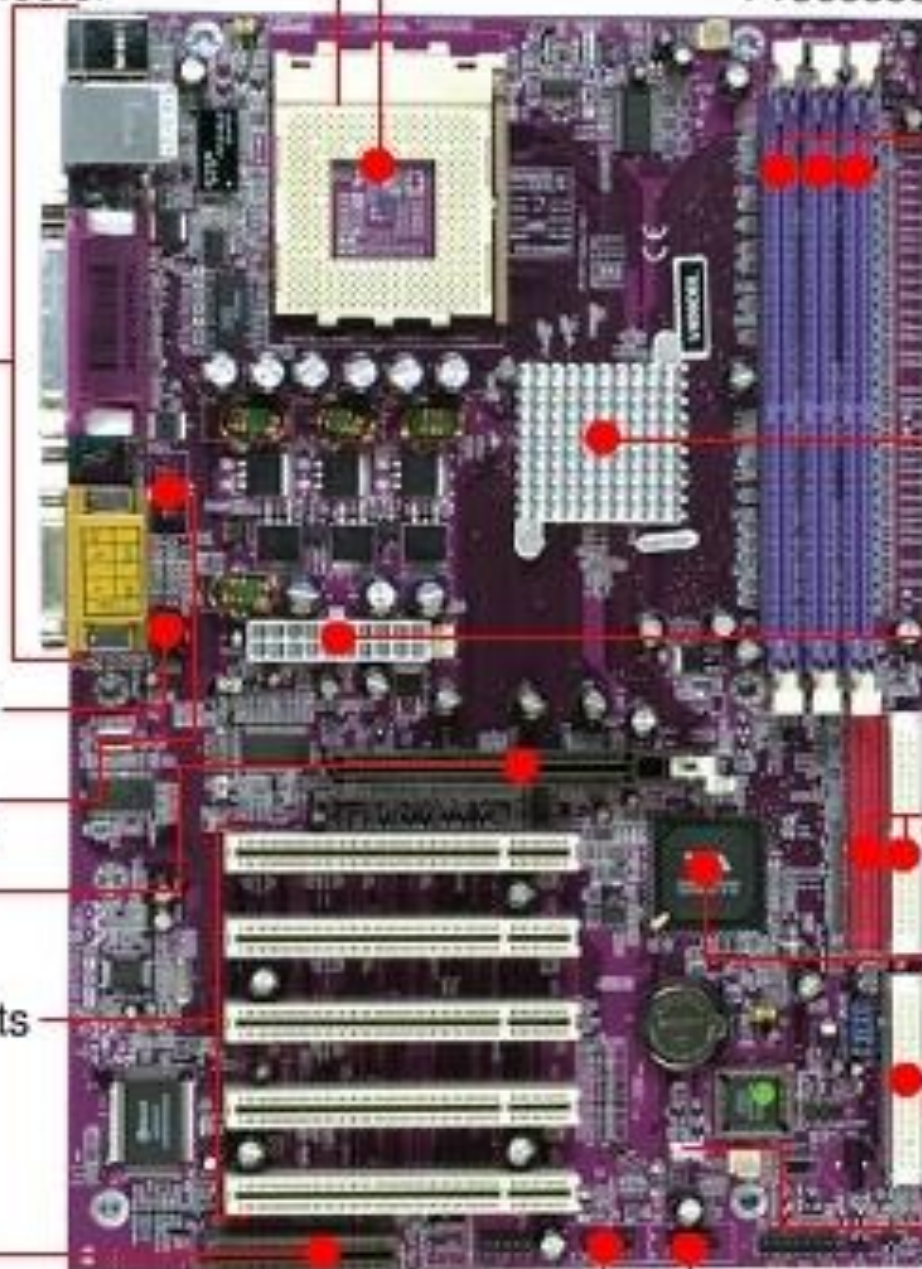
Connectors
VIA VT8235

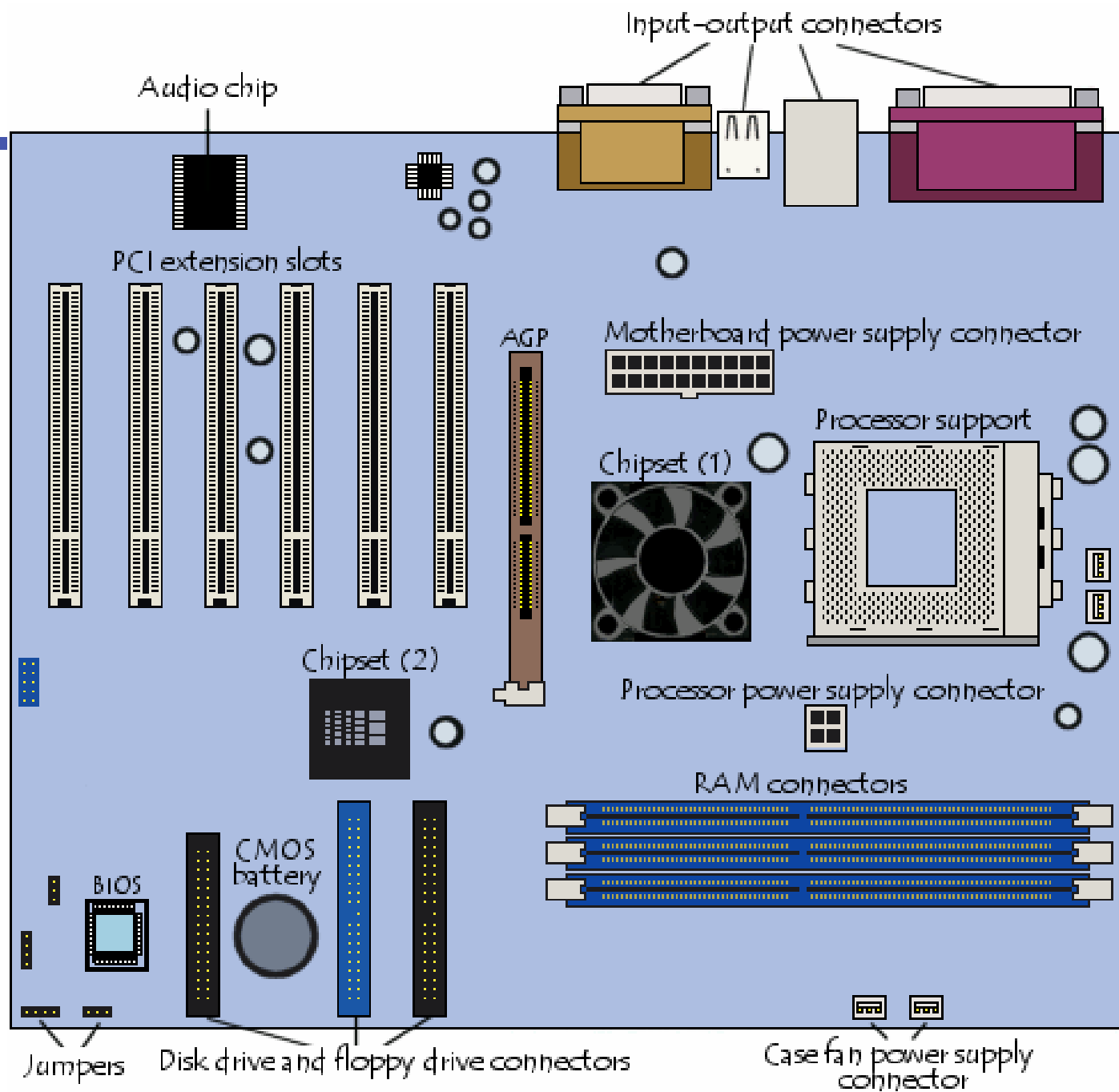
Southbridge
Chipset

Floppy Disk
Connector

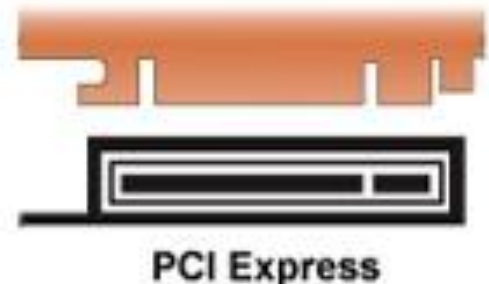
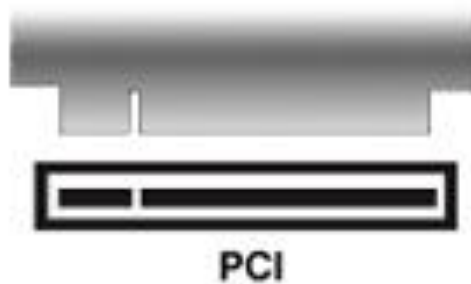
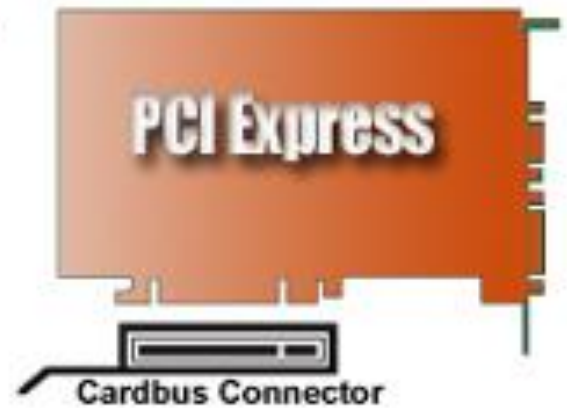
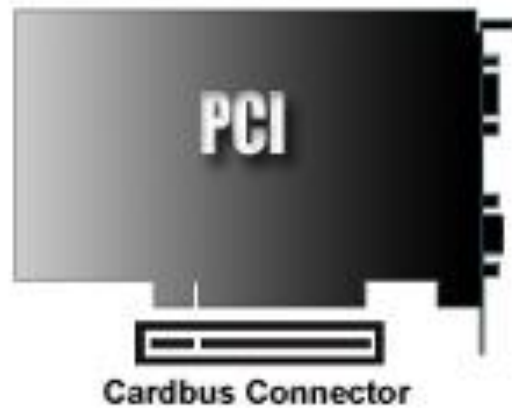
WOL Header

USB Headers

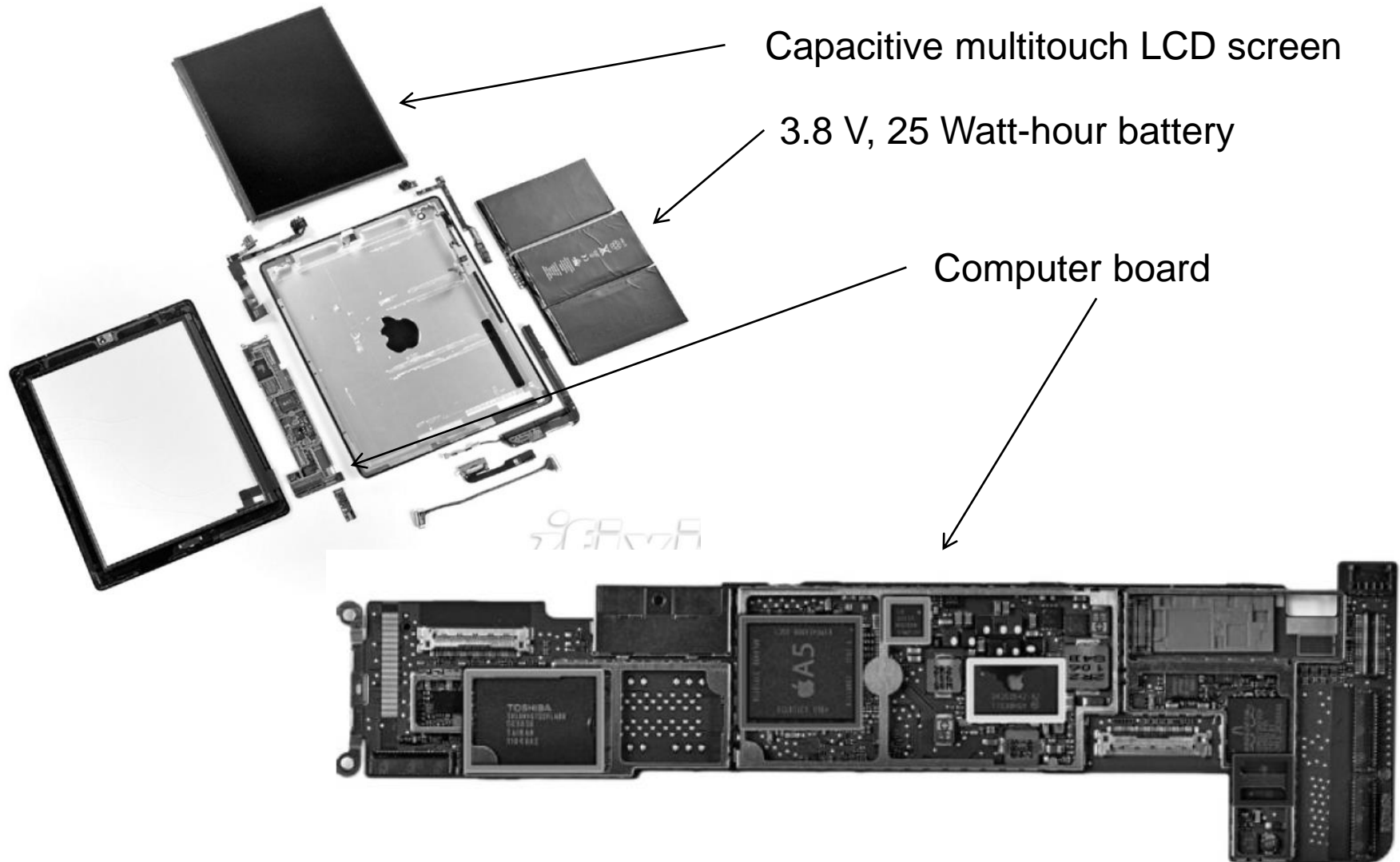




Extension interfaces



Opening the Box



Inside the Processor (CPU)

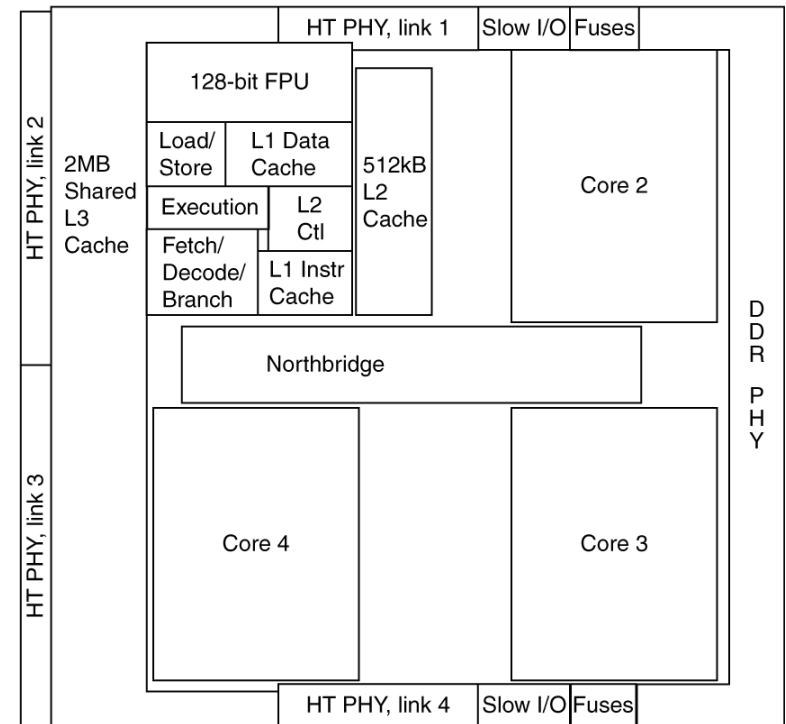
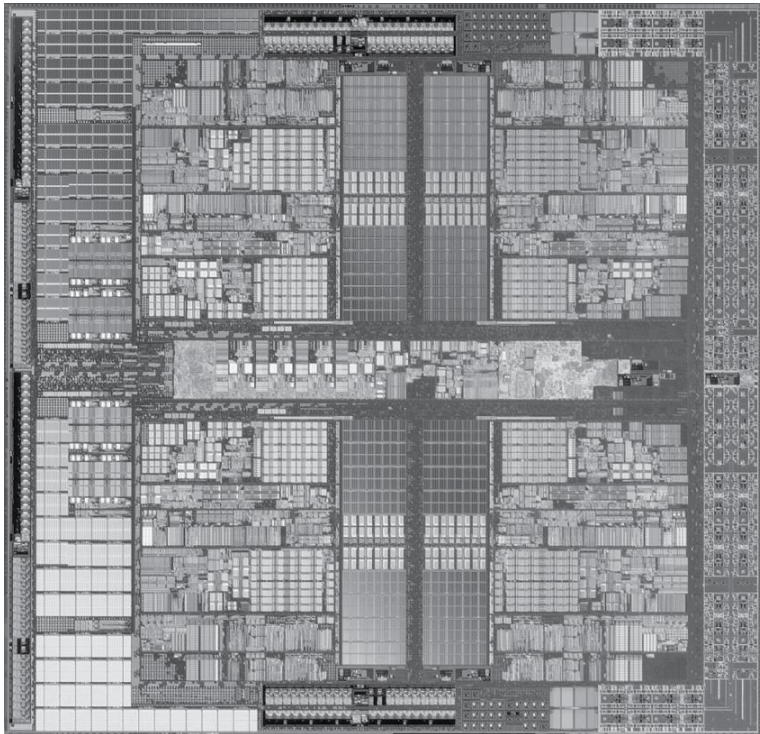
- Datapath: “performs arithmetic operations on data”
- Control: “sequences datapath, memory, ...”
- Cache memory
 - Small fast SRAM memory for immediate access to data

General CPU components

- **ALU (Arithmetic Logic Unit)** – a set of electronic circuits that carries out basic arithmetic operations
- Registers – memory in CPU
- **CU (Control Unit)** – a set of circuits that take care of fetching data and instructions from memory. Also controlling flow of data back and forth between registers and ALU

Inside the Processor

- AMD Barcelona: 4 processor cores



Inside the Processor

- Apple A5



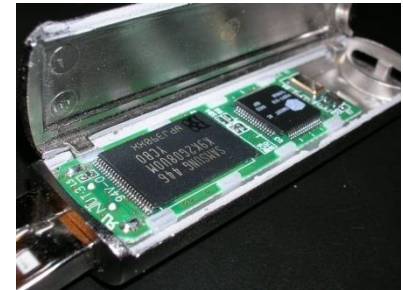
Abstractions

The BIG Picture

- Abstraction helps us deal with complexity
 - Hide lower-level detail
- Instruction set architecture (ISA)
 - The hardware/software interface
- Application binary interface
 - The ISA plus system software interface
- Implementation
 - The details underlying and interface

A Safe Place for Data

- Volatile main memory
 - Loses instructions and data when power off
- Non-volatile secondary memory
 - Magnetic disk
 - Flash memory
 - Optical disk (CDROM, DVD)



PC memory

DDR PC Memory

A dual inline memory module (DIMM) consists of a number of memory components (usually black) that are attached to a printed circuit board (usually green). The gold pins on the bottom of the DIMM provide a connection between the module and a socket on a larger printed circuit board. The pins on the front and back of a DIMM are not connected to each other.

184-pin DIMMs are used to provide DDR memory for desktop computers. Standard 184-pin DIMMs are available in PC2100 DDR, PC2700 DDR, and PC3200 DDR.

PC memory

DDR PC Memory

(continued)

To use DDR memory, your system motherboard must have 184-pin DIMM slots and a DDR-enabled chipset. A DDR DIMM will not fit into a standard SDRAM DIMM socket.

The number of black components on a 184-pin DIMM can vary, but it always has 92 pins on the front and 92 pins on the back, for a total of 184. 184-pin DIMMs are approximately 5.25 inches long and 1.25 inches high, though the heights can vary. While 184-pin DIMMs and 168-pin DIMMs are approximately the same size, 184-pin DIMMs have only one notch within the row of pins.

PC memory

DDR2 PC Memory:

A dual inline memory module (DIMM) consists of a number of memory components (usually black) that are attached to a printed circuit board (usually green). The gold pins on the bottom of the DIMM provide a connection between the module and a socket on a larger printed circuit board. The pins on the front and back of a DIMM are not connected to each other.

240-pin DIMMs are used to provide DDR2 memory for desktop computers. DDR2 is a leading-edge generation of memory with an improved architecture that allows it to transmit data very fast. Each 240-pin DIMM provides a 64-bit data path (72-bit for ECC or registered or Fully Buffered modules).

PC memory

DDR2 PC Memory:

(continued)

To use DDR2 memory, your system motherboard must have 240-pin DIMM slots and a DDR2-enabled chipset. A DDR2 DIMM will not fit into a standard SDRAM DIMM socket or a DDR DIMM socket.

The number of black components on a 240-pin DIMM can vary, but it always has 120 pins on the front and 120 pins on the back, for a total of 240. 240-pin DIMMs are approximately 5.25 inches long and 1.18 inches high, though the heights can vary. While 240-pin DDR2 DIMMs, 184-pin DDR DIMMs, and 168-pin DIMMs are approximately the same size, 240-pin DIMMs and 184-pin DIMMs have only one notch within the row of pins. The notch in a 240-pin DDR2 DIMM is closer toward the center of the module.

PC memory

DDR and DDR2 Notebook Memory:

A small outline dual inline memory module (SODIMM) consists of a number of memory components (usually black) that are attached to a printed circuit board (usually green). SODIMMs get their name because they are smaller and thinner than regular DIMMs. The gold pins on the bottom of the SODIMM provide a connection between the module and a socket on a larger printed circuit board. The pins on the front and back of a SODIMM are not connected. 200-pin SODIMMs are used to provide DDR and DDR2 memory for notebook computers. 200-pin SODIMMs are available in PC2700 DDR, PC3200 DDR, DDR2 PC6400, DDR2 PC5400, DDR2 PC4200 and DDR2 PC3200 . To use DDR or DDR2 memory, your system motherboard must have 200-pin SODIMM slots and a DDR- or DDR2-enabled chipset. A DDR or DDR2 SODIMM will not fit into a standard SDRAM SODIMM socket.

The number of black components on a 200-pin SODIMM can vary, but it always has 100 pins on the front and 100 pins on the back, for a total of 200. 200-pin SODIMMs are approximately 2.625 inches long and 1.25 inches high, though the heights can vary. Like 144-pin SODIMMs, 200-pin SODIMMs have one small notch within the row of pins; however, the notch on the 200-pin SODIMMs is closer to the left side of the module.

PC memory

SDRAM Memory:

A dual inline memory module (DIMM) consists of a number of memory components (usually black) that are attached to a printed circuit board (usually green). The gold pins on the bottom of the DIMM provide a connection between the module and a socket on a larger printed circuit board. The pins on the front and back of a DIMM are not connected.

168-pin DIMMs are commonly found in Pentium® and Athlon® systems. 168-pin DIMMs are available in 66MHz SDRAM, PC100 SDRAM, and PC133 SDRAM. When upgrading, be sure to match the memory technology that is already in your system.

The number of black components on a 168-pin DIMM can vary, but it always has 84 pins on the front and 84 pins on the back, for a total of 168. 168-pin DIMMs are approximately 5.25 inches long and 1.375 inches high, though the heights can vary. They have two small notches within the row of pins along the bottom of the module.

PC memory

SDRAM Notebook Memory:

A small outline dual inline memory module (SODIMM) consists of a number of memory components (usually black) that are attached to a printed circuit board (usually green). SODIMMs get their name because they are smaller and thinner than regular DIMMs. The gold pins on the bottom of the SODIMM provide a connection between the module and a socket on a larger printed circuit board. The pins on the front and back of a SODIMM are not connected. 144-pin SODIMMs are commonly found in notebook computers. 144-pin SODIMMs are available in 66MHz SDRAM, PC100 SDRAM, and PC133 SDRAM. When upgrading, be sure to match the memory technology that is already in your system.

The number of black components on a 144-pin SODIMM can vary, but it always has 72 pins on the front and 72 pins on the back, for a total of 144. 144-pin SODIMMs are approximately 2.625 inches long and 1.25 inches high, though the heights can vary. They have one small notch within the row of pins along the bottom of the module.

PC memory

SIMM Memory:

single inline memory module (SIMM) consists of a number of memory components (usually black) that are attached to a printed circuit board (usually green). The gold or tin pins on the bottom of the SIMM provide a connection between the module and a socket on a larger printed circuit board. The pins on the front and back of a SIMM are connected.

72-pin SIMMs are commonly found in older desktop computers, such as the 486 and early Pentium® models. 72-pin SIMMs are available in FPM or EDO. When upgrading, be sure to match the memory technology that is already in your system.

The number of black components on a 72-pin SIMM can vary. 72-pin SIMMs are approximately 4.25 inches long and 1 inch high, though the heights can vary. They have one notch on the bottom left and one notch in the center of the module.

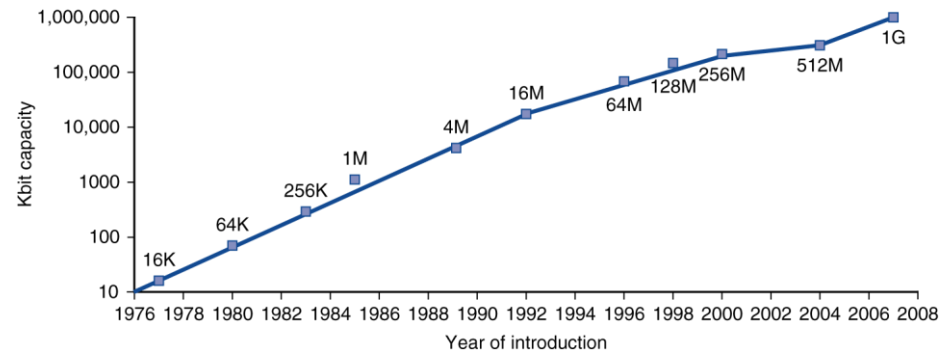
Networks

- Communication and resource sharing
- Local area network (LAN): Ethernet
 - Within a building
- Wide area network (WAN: the Internet)
- Wireless network: WiFi, Bluetooth



Technology Trends

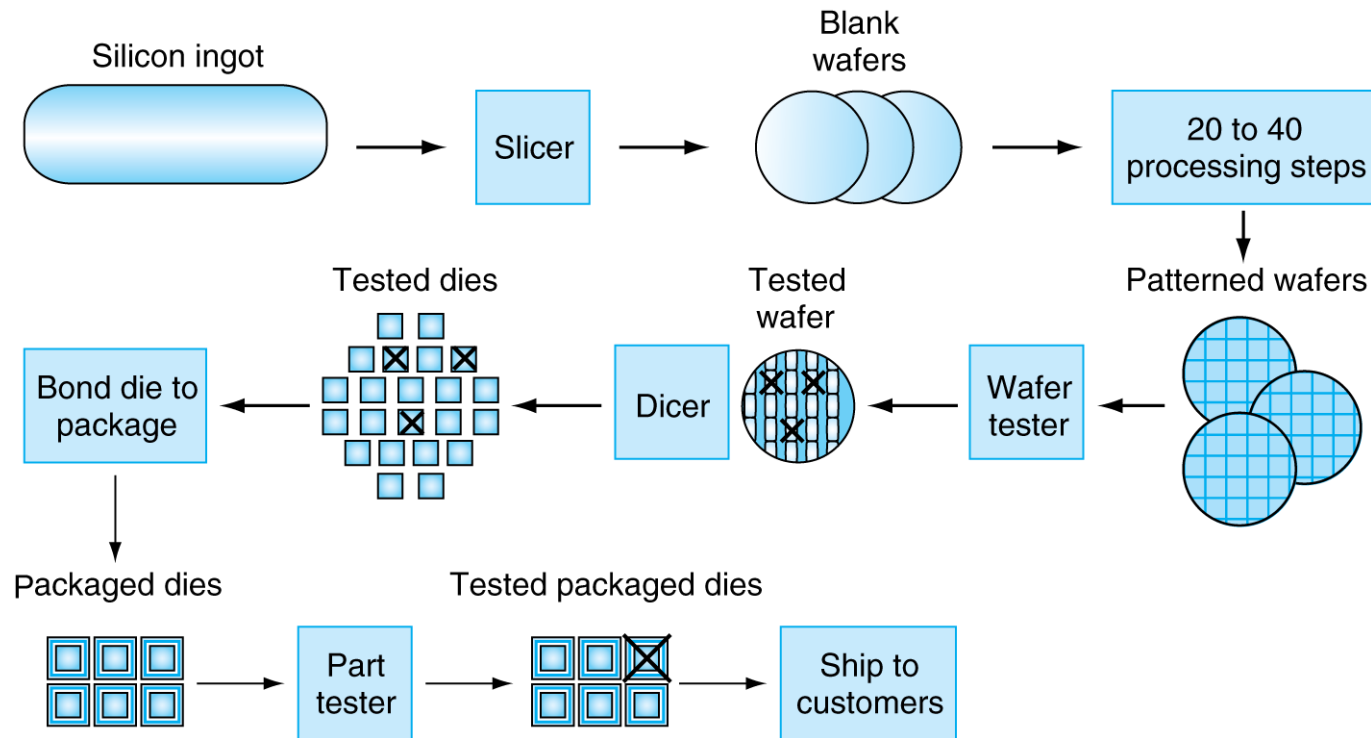
- Electronics technology continues to evolve
 - Increased capacity and performance
 - Reduced cost



DRAM capacity

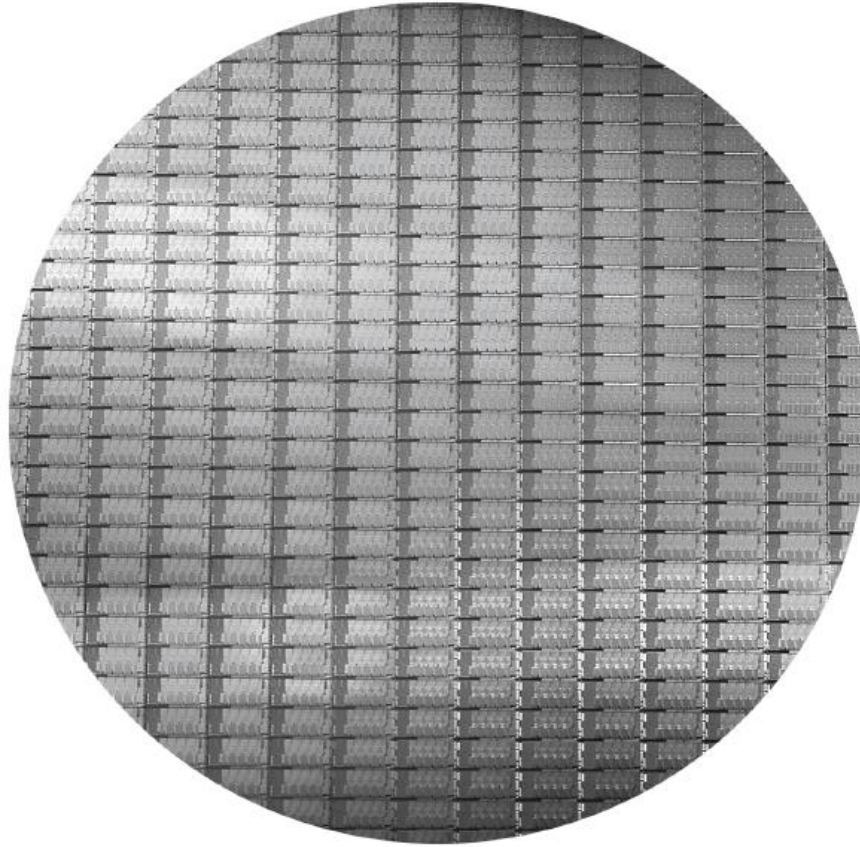
Year	Technology	Relative performance/cost
1951	Vacuum tube	1
1965	Transistor	35
1975	Integrated circuit (IC)	900
1995	Very large scale IC (VLSI)	2,400,000
2005	Ultra large scale IC	6,200,000,000

Manufacturing ICs



- Yield: proportion of working dies per wafer

Intel Core i7 Wafer



- 300mm wafer, 280 chips, 32nm technology
- Each chip is 20.7 x 10.5 mm

Integrated Circuit Cost

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

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

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

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

Summary: what is computer

- Components:
 - CPU
 - input (mouse, keyboard) / output (display, printer)
 - memory (disk drives, DRAM, SRAM, CD)
 - network

Summary: what is computer

- Our primary focus: the processor (datapath and control)
 - implemented using millions of transistors
 - Impossible to understand by looking at each transistor
 - We need abstraction

Summary: how computers work

- Need to understand abstractions such as:
 - Applications software
 - Systems software
 - Operating system
 - Compiler
 - Assembly Language
 - Machine Language
 - Architectural Issues: i.e., Caches, Virtual Memory, Pipelining

Summary: how computers work

- Need to understand abstractions such as:
 - Sequential logic, finite state machines
 - Combinational logic, arithmetic circuits
 - Boolean logic, 1s and 0s
 - Transistors used to build logic gates (CMOS)
 - Semiconductors/Silicon used to build transistors
 - Properties of atoms, electrons, and quantum dynamics

Summary: components (I)

- CPU
- I/O Devices
- Memory
- Motherboard
- (continue to next slides)

Summary: 3 main components

- CPU
- I/O Devices
- Memory

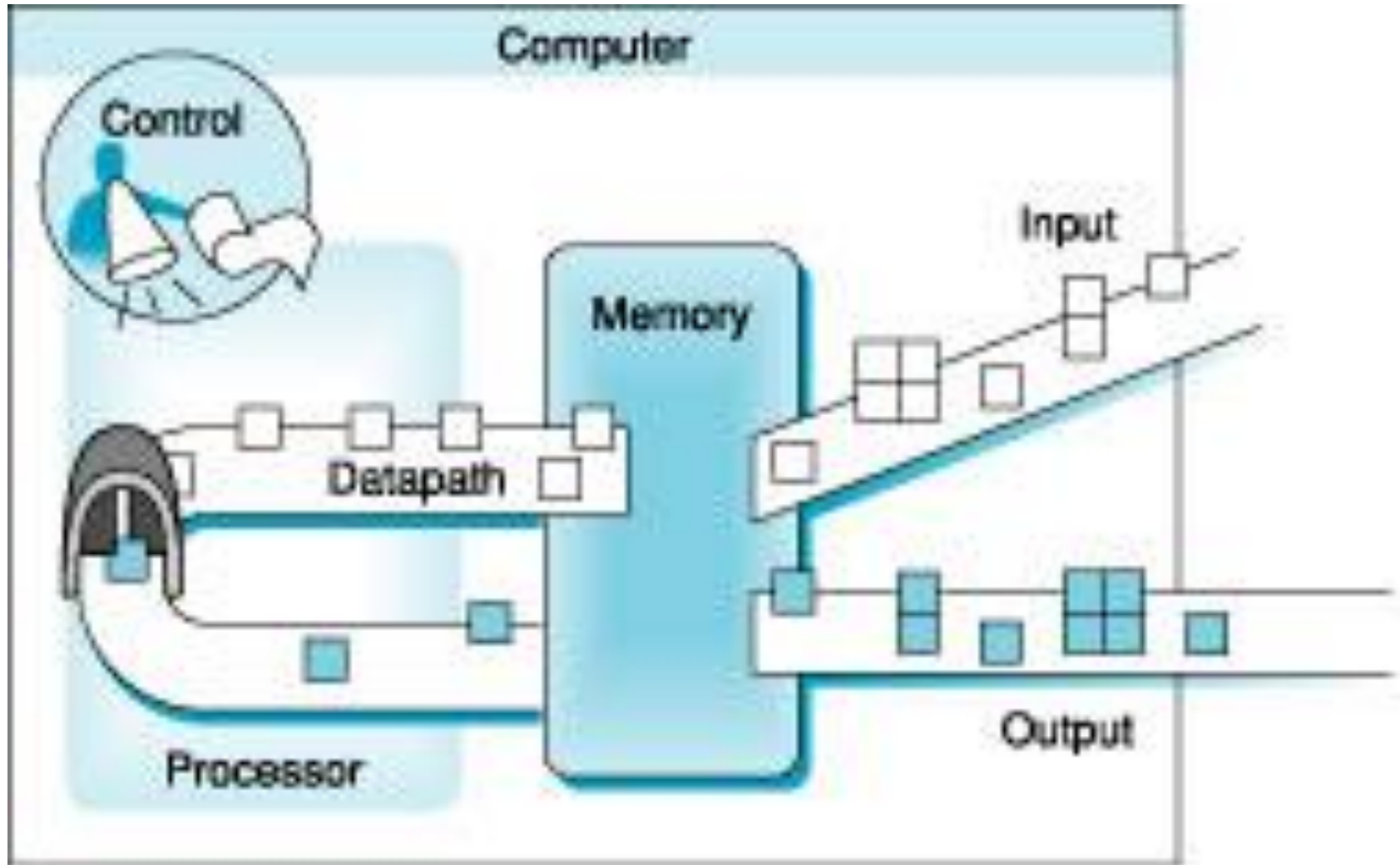


- on a Motherboard

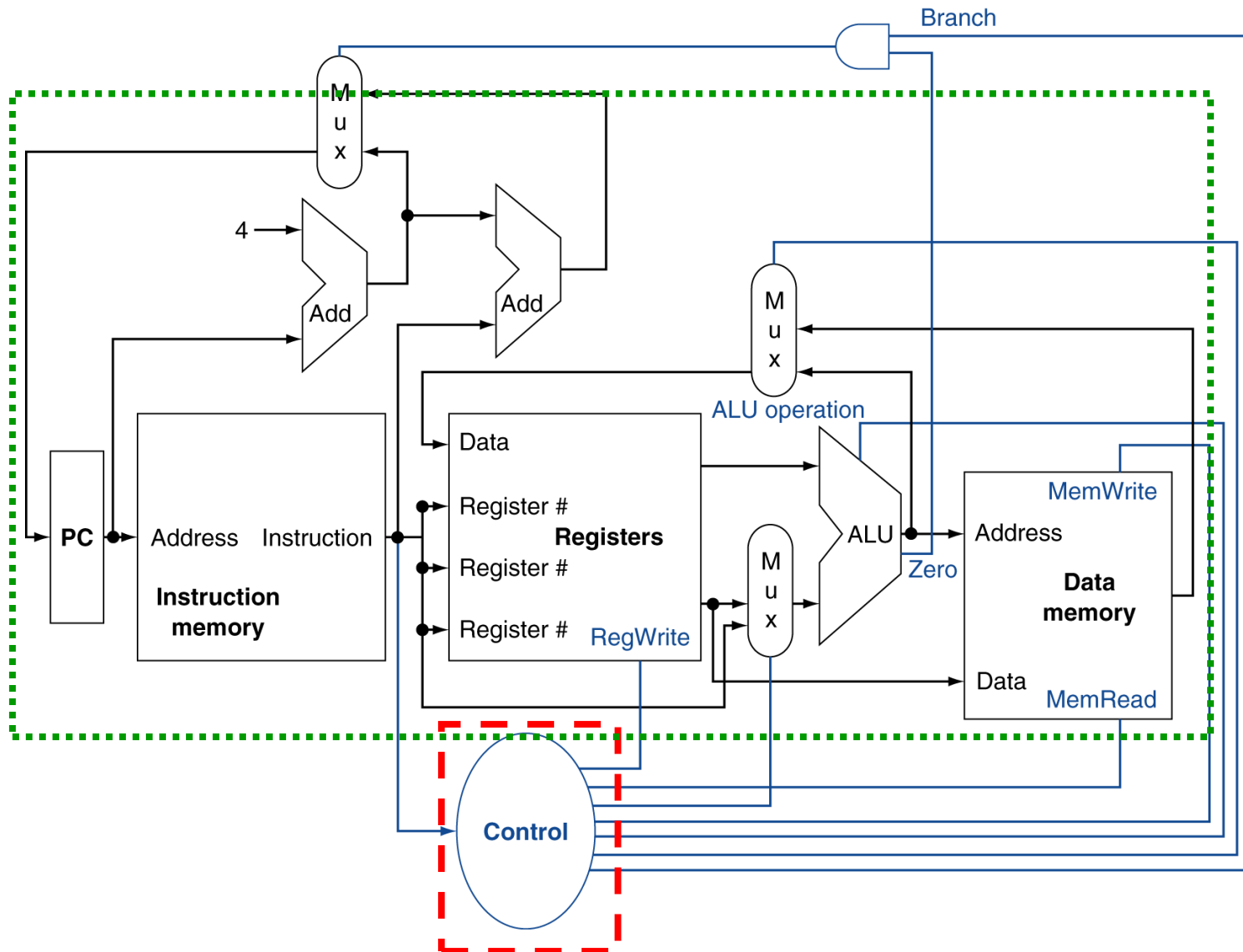
Summary: components (II)

- **Datapath:** the component of the processor that performs arithmetic operations
- **Control:** The component of the processor that commands the datapath, memory, and I/O devices according to the instructions of the program
- DRAM
- Cache Memory

Summary: components (II)



CPU basic design example

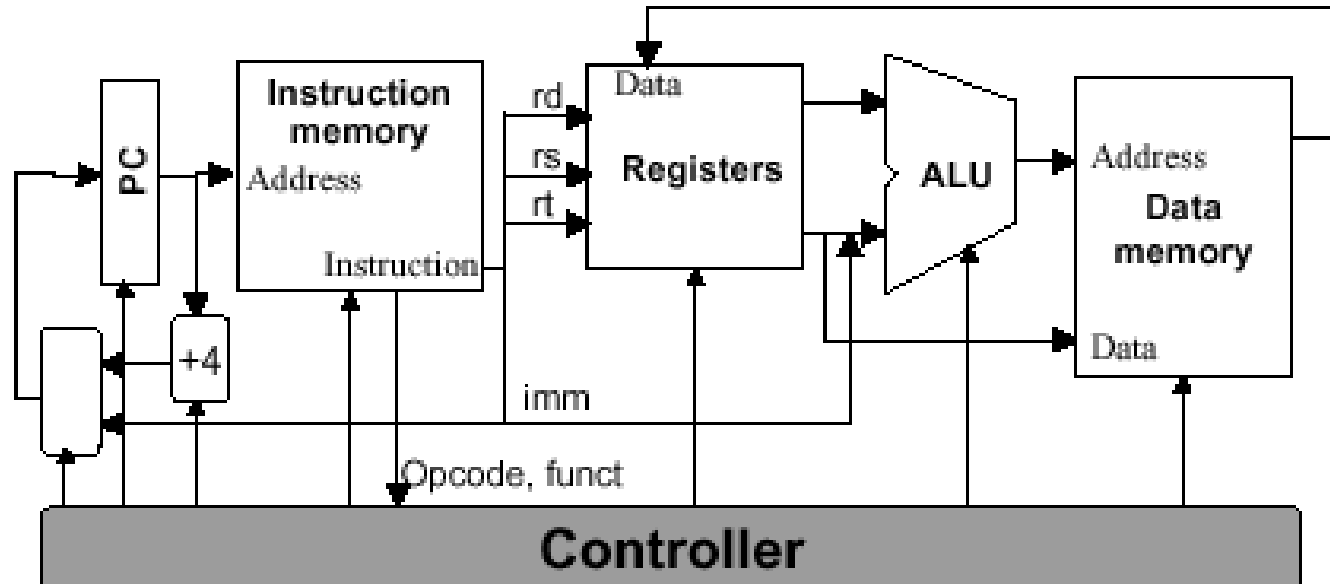


What is Datapath?

- Datapath: is the hardware that performs all the required operations
- for example, ALU, multipliers, registers, and internal buses.

What is Control?

- Control: is the hardware that tells the datapath what to do, in terms of switching, operation selection, data movement between ALU components, etc.



Terminologies (I)

- DIMM (dual inline memory module)
- ISA (Instruction Set Architecture)
- ABI (Application Binary Interface)
- Volatile Memory: storage, such as DRAM, that retains data only if it is receiving power
- Nonvolatile memory: A form of memory that retains data even in the absence of a power source and that is used to store programs between runs. E.g. HDD

Terminologies (II)

- Primary memory: main memory, e.g. DRAM
- Secondary memory: e.g. magnetic disk
- Flash memory
-

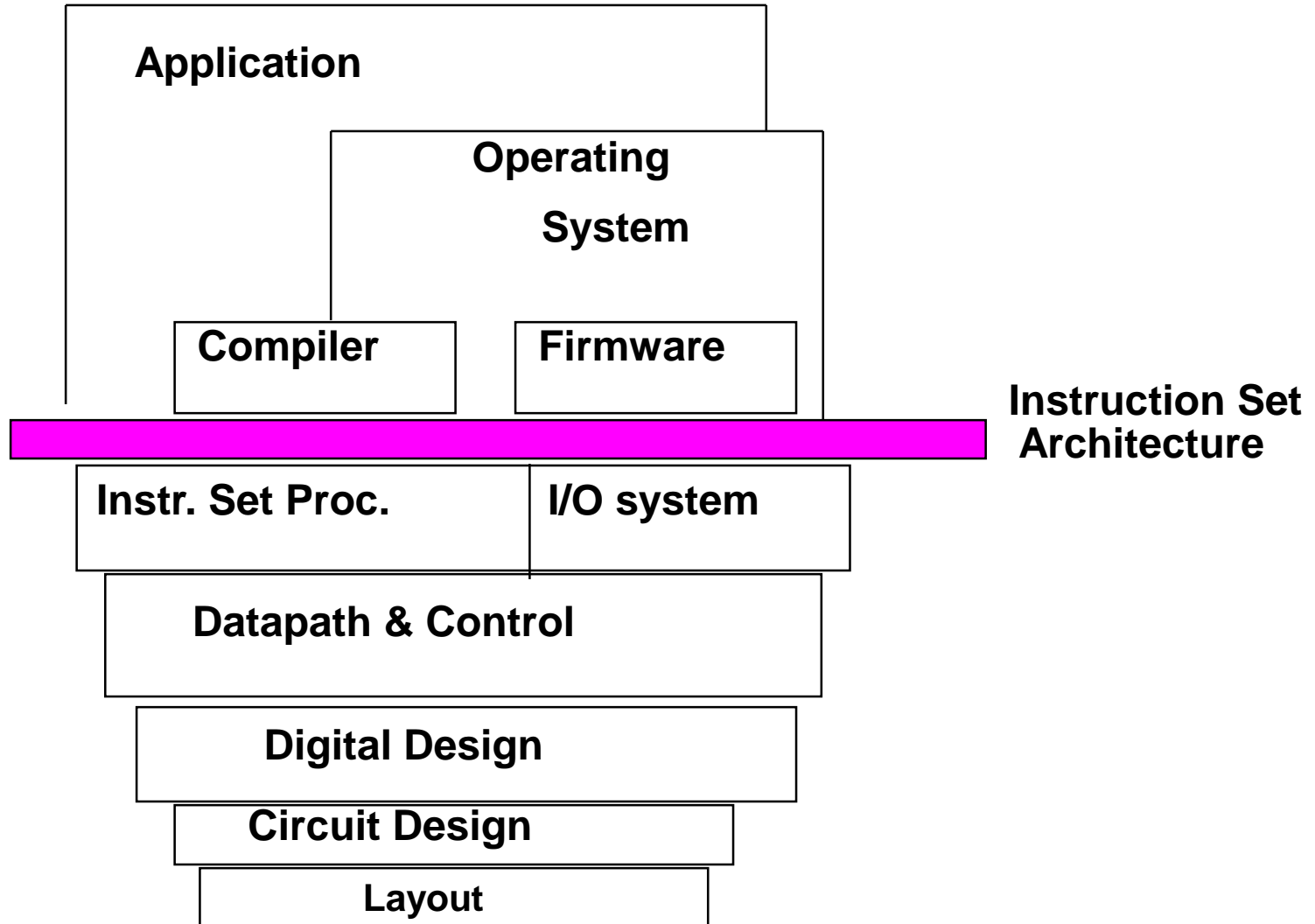
ISA (Instruction Set Architecture)

- A very important abstraction
 - interface between hardware and low-level software
 - standardizes instructions, machine language bit patterns, etc.
 - advantage: *different implementations of the same architecture*
 - disadvantage: *sometimes prevents using new innovations*

ISA

- Modern instruction set architectures:
 - IA-32, PowerPC, MIPS, SPARC, ARM, and others

ISA

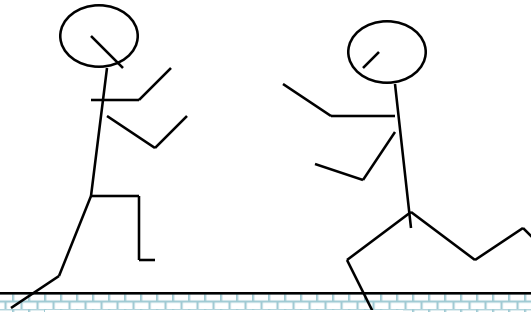


ISA

- Coordination of many *levels of abstraction*
- Under a rapidly changing set of forces
- Design, Measurement, *and* Evaluation

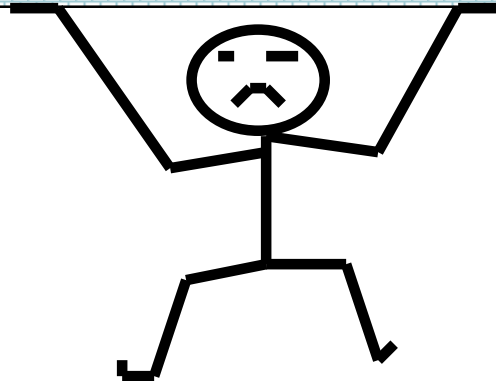
ISA

software



instruction set

hardware



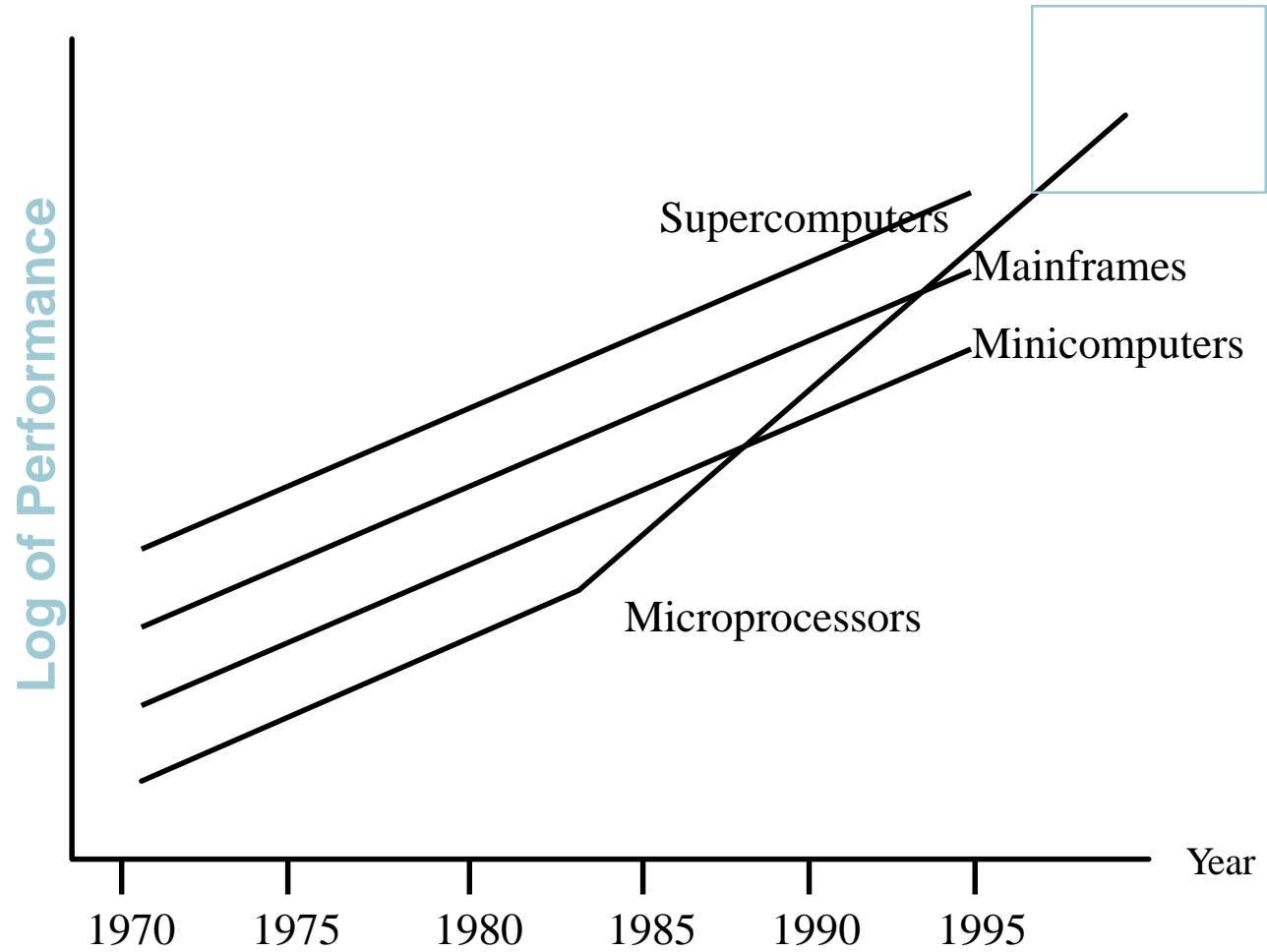
Examples of ISA

- Digital Alpha (v1, v3) 1992-97
- HP PA-RISC (v1.1, v2.0) 1986-96
- Sun Sparc (v8, v9) 1987-95
- SGI MIPS (MIPS I, II, III, IV, V) 1986-96
- Intel (8086, 80286, 80386, 1978-96
80486, Pentium, MMX, ...)
- AMD (Athlon, ...)

Instruction set	Bits	Version	Introduced	Max #operands	Type	Design	Registers(excluding FP/vector)	Instruction encoding	Branch evaluation	Endianness	Extensions	Open	Royalty free
6502	8		1975	1	Register Memory	CISC	3	Variable (8- to 32-bit)	Condition register	Little			
65k	64 (8→64) ^[1]		2006?	1	Memory Memory ^[citation needed]	CISC	1	Variable (8-bit to 256 bytes)	Compare and branch ^[citation needed]	Little			
68000 / 680x0	32		1979	2	Register Memory	CISC	8 data and 8 address	Variable	Condition register	Big		Unknown	Unknown
8080	8		1974	2	Register Memory	CISC	8	Variable (8 to 24 bits)	Condition register	Little			
8051	32 (8→32)		1977?	1	Register Register	CISC	<ul style="list-style-type: none"> •32 in 4-bit •16 in 8-bit •8 in 16-bit •4 in 32-bit 	Variable (8-bit to 128 bytes)	Compare and branch	Little			
8086 / x86	16, 32, 64 (16→32→64)		1978	2 (integer) 3 (AVX) ^[2]	Register Memory	CISC	<ul style="list-style-type: none"> •8 (+ 4 or 6 segment registers) in 16/32-bit •16 (+ 2 segment 	Variable (8086: 8- to 48-bit)	Condition code	Little	x87 , IA-32 , MMX , 3DNow! , SSE , SSE2 , PAE , x86-64 , SSE3 , SSE4 ,	No	No

Itanium (IA-64)	64		2001		Register Register	EPIC	128		Condition register	Bi (selectable)	Intel Virtualization Technology	No	No
M32R	32		1997			RISC	16	Fixed (16- or 32-bit)		Bi		Unknown	Unknown
Mico32	32		2006	3	Register Register	RISC	32 ^[11]	Fixed (32-bit)	Compare and branch	Big	User-defined instructions	Yes ^[12]	Yes
MIPS	64 (32→64)	5	1981	1–3	Register Register	RISC	4–32 (including "zero")	Fixed (32-bit)	Condition register	Bi	MDMX , MIPS-3D	Unknown	No
MMIX	64		1999	3	Register Register	RISC	256	Fixed (32-bit)		Big		Yes	Yes
NS320xx	32		1982	5	Memory Memory	CISC	8	Variable Huffman coded, up to 23 bytes long	Condition code	Little	BitBlit instructions	Unknown	Unknown
OpenRISC	32, 64		2010	3	Register Register	RISC	16 or 32	Fixed				Yes	Yes
PA-RISC (HP/PA)	64 (32→64)	2.0	1986	3	Register Register	RISC	32	Fixed (32-bit)	Compare and branch	Big → Bi	Multimedia Acceleration Extensions (MAX), MAX-2	No	Unknown
							8 (includes stack pointer,				Floating Point,		

Trends



Ref. History

- First was mechanical analog computers (all special-purpose)
- Antikythera mechanism is the first known example (205BC)
- used for calculating astronomical tables (for calendars, astrology, Olympiad cycles)
- used rotating gears with fine-tuned gear ratios (go find a recreation!)
- Su Song's Clock tower in China (1088)
- Babbage's Analytical -> Difference Engines (1840s)
- Later differential analyzers (late 1800s -> 1930s), solving differential equations
- Then moved to electromechanical devices (relays) 1930s
- Digital Devices -> Konrad Zuse Z2->3
- Moved to Vacuum Tubes (ENIAC) for US Army
- First semiconductor in 1947
- First Semiconductor Computer 1950 (NIST SEAC)
- First Transistor Computer 1953 (Univ. of Manchester)
- First IC 1958, Jack Kilby
- IBM System/360 uses ICs and discrete components (1964)
- Silicon Transistors in the CDC 6600 (first supercomputer, (1965)
- First Microprocessor, Intel 4004 (1971)
- Intel 8008 in 1972, used for Datapoint 2200, first microcomputer

Ref. History

- Altair 8800 in 1974, used Intel 8080
- MOS 6502 in 1975, still used today
- Apple I in 1976, designed by Steve Wozniak
- Intel 8086 in 1978, great great great grandfather of today's chips
- First IBM PC, 1981, based on Intel 8088, ran MS-DOS
- Intel Pentium, 1993
- Pentium 4, 2000 (End of clock speed scaling)
- IBM POWER 4, first commercial multicore (2001)
- Reference slides – [lec1 slides.pdf](#)

History

- ENIAC built in World War II was the first general purpose computer
 - Used for computing artillery firing tables
 - 80 feet long by 8.5 feet high and several feet wide
 - Each of the twenty 10 digit registers was 2 feet long
 - Used 18,000 vacuum tubes
 - Performed 1900 additions per second

History



History

- **Since then, Moore's Law:
Transistor capacity doubles
every 18-24 months**
- <http://www.intel.com/about/companyinfo/museum/exhibits/moore.htm?wapkw=moore>

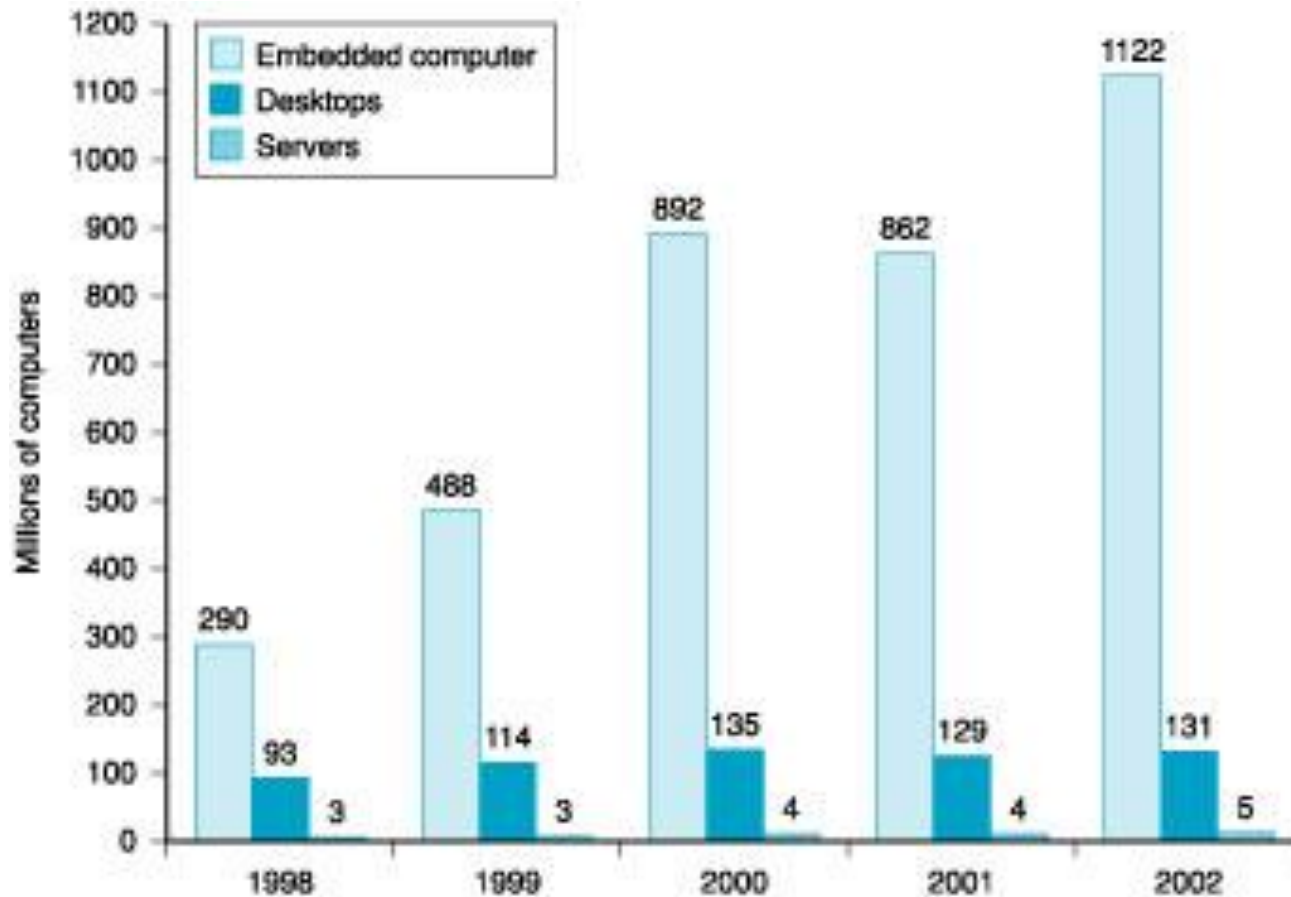
History

- ** Samsung Electronics beats the Moore's law: doubles at every 12 months:
- <http://yvettewohn.com/2008/10/06/hwangs-law/>
- [ZDnet](#) (2004)

But now this law is phasing out.... Back to Moore's law after proving nine straight years

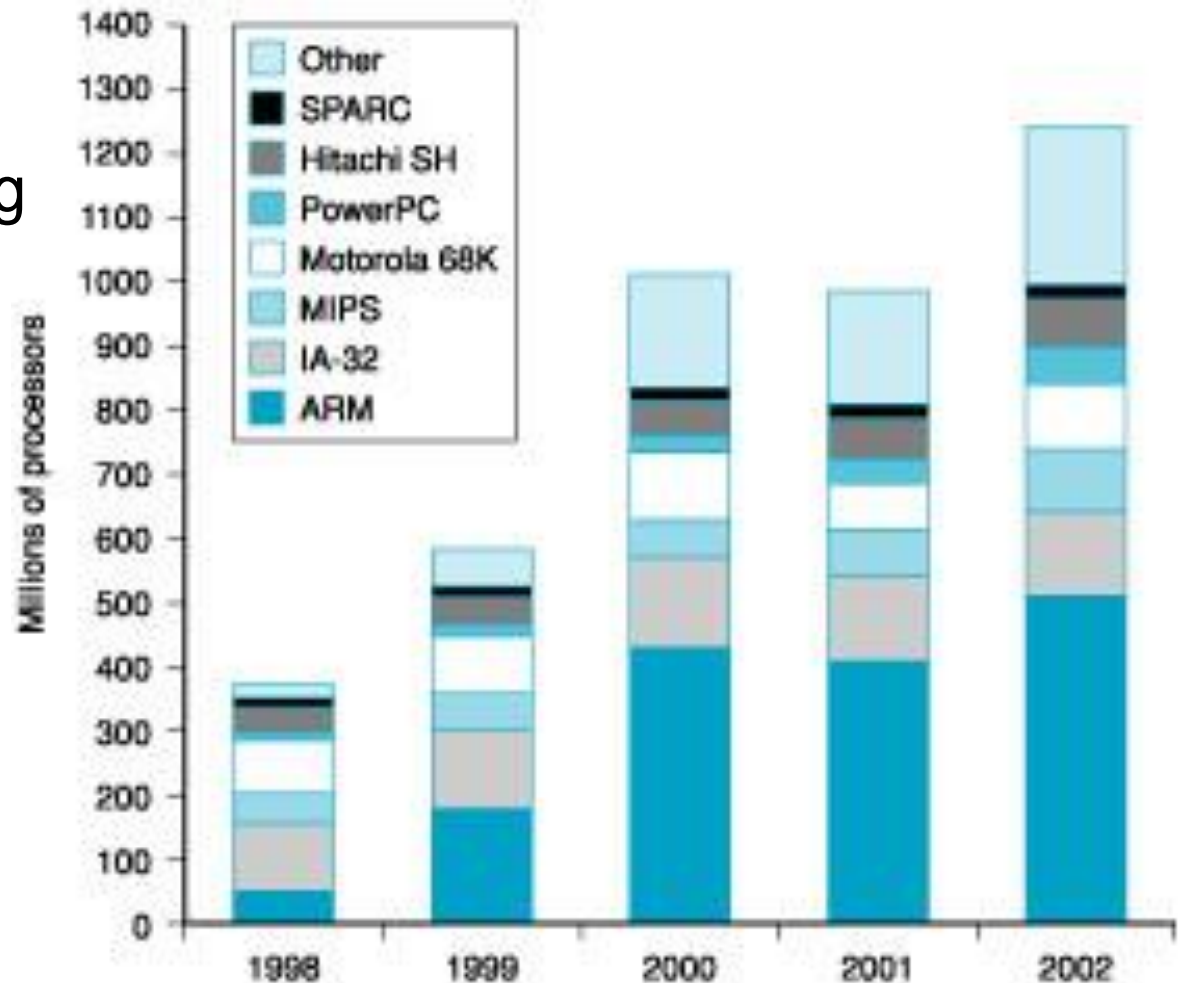
System Growth

The number of distinct processors sold between 1998 and 2002.



Processor Growth

Sales of microprocessors between 1998 and 2002 by ISA combining all uses



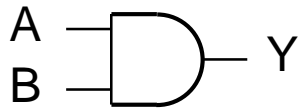
High level vs. low level langs..

- Electronic machine understands electrical signals.
- Easiest way to send ES is ON and OFF: binary digit (or bit)
- Computers work with instructions (machine language of bits)
- Programmers need to write machine understanding language.

Circuit Design example

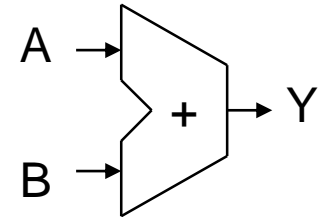
- AND-gate

- $Y = A \& B$



- Adder

- $Y = A + B$



- A: 1101

- B: 0001

- $Y = 0001$

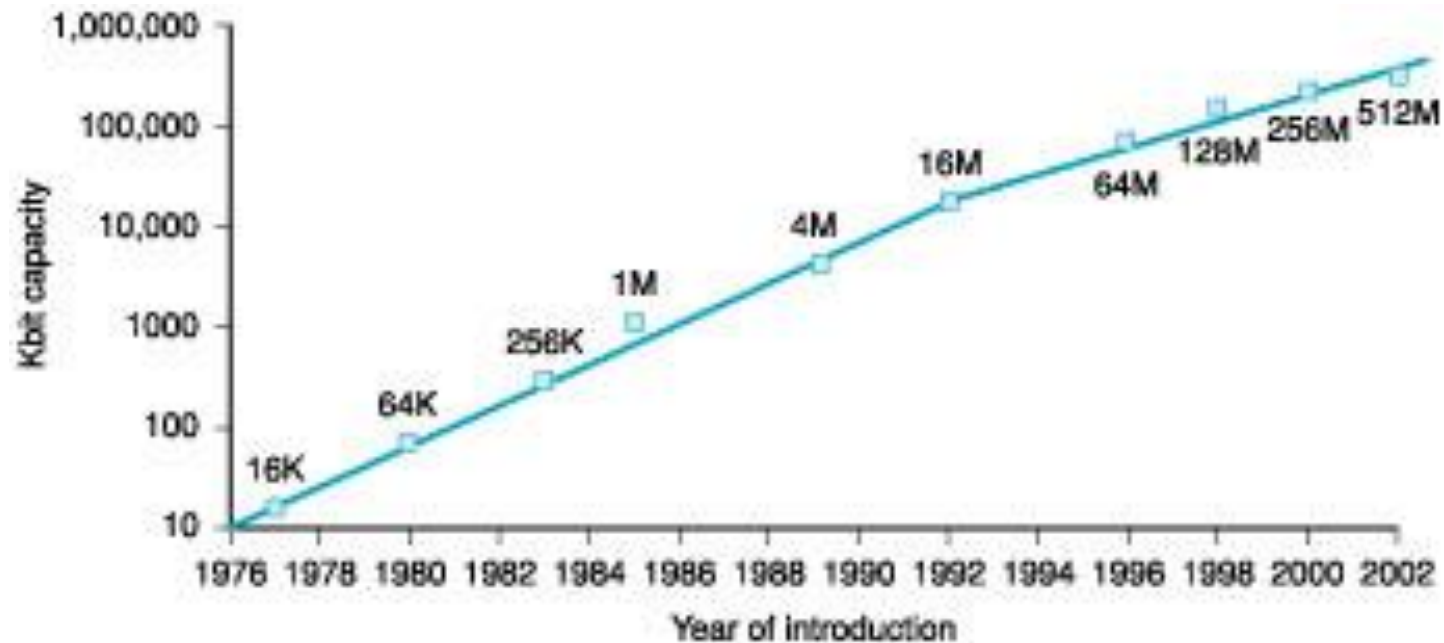
High level vs. low level langs..

- High level language: $A + B$
- Low level language: Add A, B
- Machine language: 1000110010100000
- Assembler: first programming language to translate symbolic notation to binary version

High level vs. low level langs..

- Assembly language: A symbolic representation of machine instructions.
- Benefit of HL: more natural language like thinking, to be designed to their intended use (Java/C, Fortran, Cobol, Lisp, ...),
Improve productivity for SE, easier to debug, programs to be independent of the computer

Assignment research

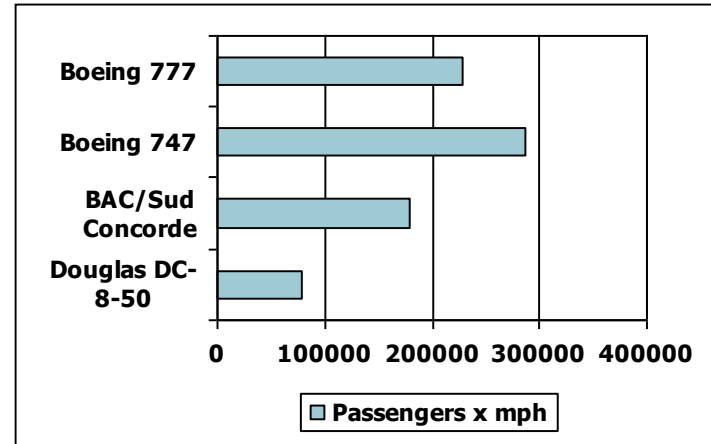
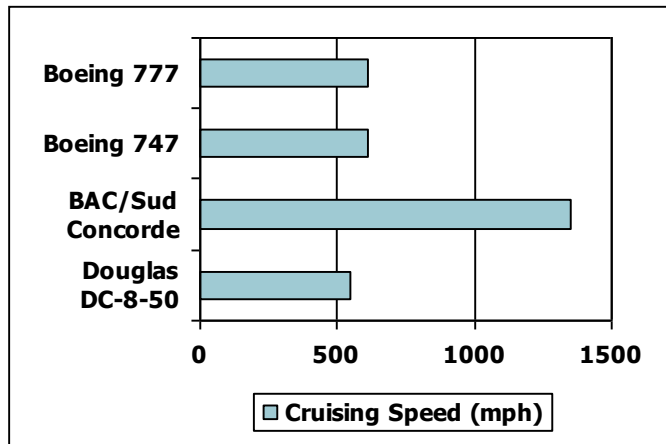
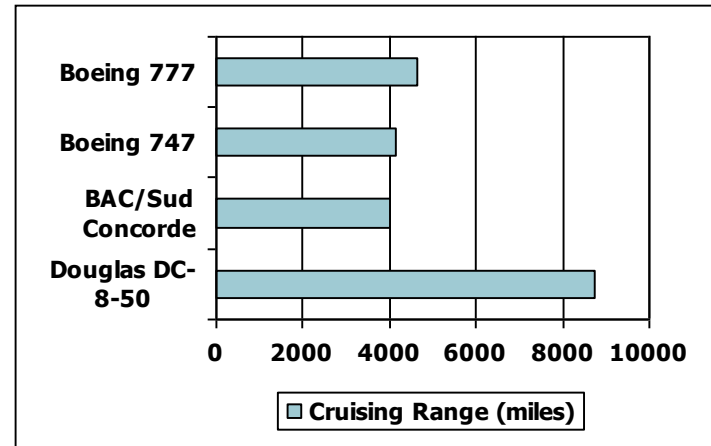
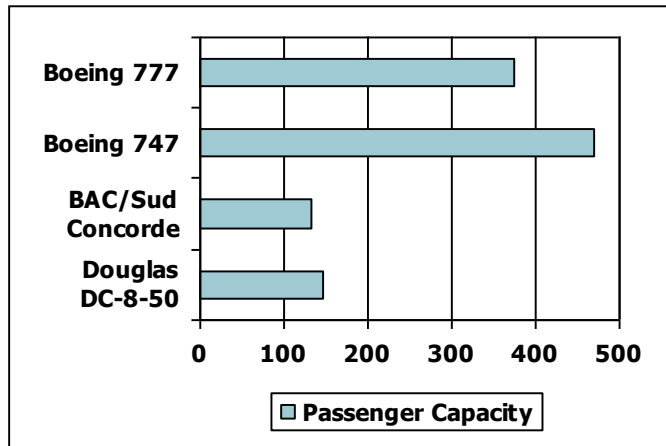


Assignment research

- Research different types of CPUs today: RISC, CISC, MIPS, SPARCs, Intel family, AMD family, others, etc.
- Watch Intel web site for video, e.g.
- Research what different memory types today: DDR, DDR2, DDR3, DRAM, SDRAM, SRAM, Cache in CPU, etc.

Defining Performance

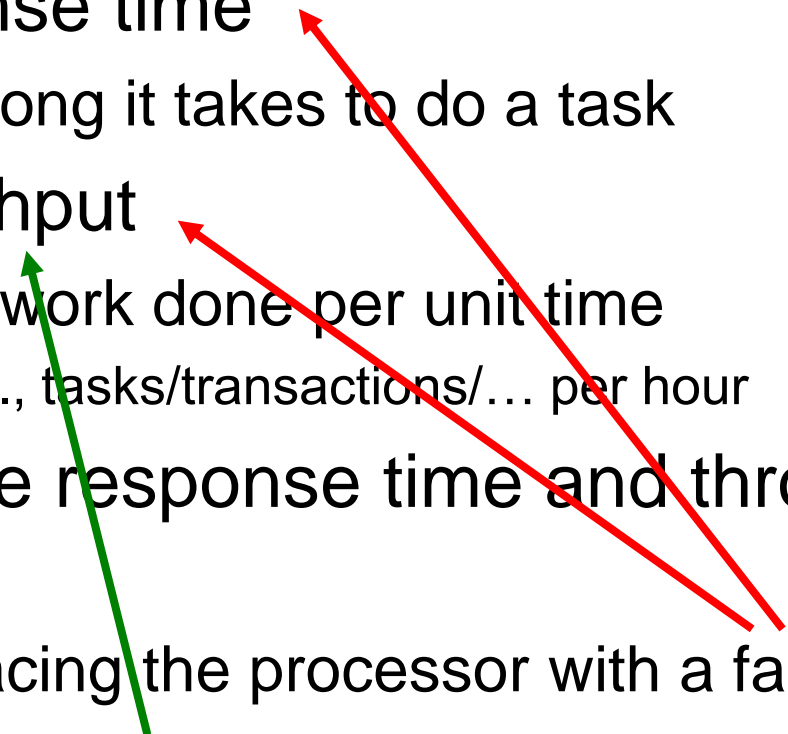
- Which airplane has the best performance?



Response Time and Throughput

- Response time
 - How long it takes to do a task
- Throughput
 - Total work done per unit time
 - e.g., tasks/transactions/... per hour
- PC focuses on how fast a job done
- Datacenter server focuses on throughput or bandwidth that the total amount of work done in a given time.. Why? Multiple users, multiple tasks..

Response Time and Throughput

- Response time
 - How long it takes to do a task
 - Throughput
 - Total work done per unit time
 - e.g., tasks/transactions/... per hour
 - How are response time and throughput affected by
 - Replacing the processor with a faster version?
 - Adding more processors?
 - We'll focus on response time for now...
- 

Relative Performance

- Define Performance = $1/\text{Execution Time}$
- “X is n time faster than Y”

$$\begin{aligned} & \text{Performance}_X / \text{Performance}_Y \\ &= \text{Execution time}_Y / \text{Execution time}_X = n \end{aligned}$$

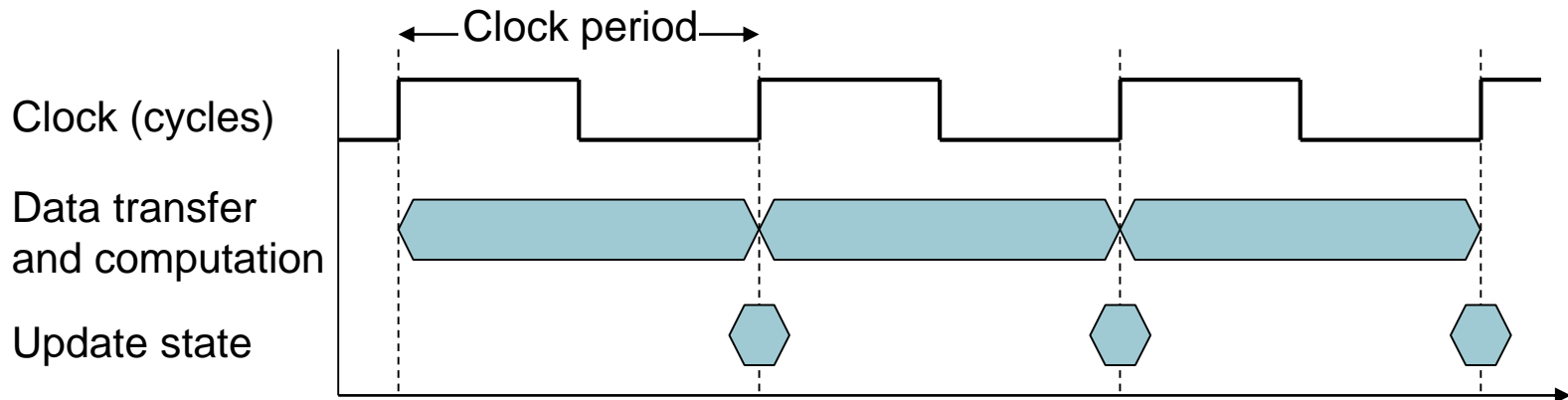
- Example: time taken to run a program
 - 10s on A, 15s on B
 - $\text{Execution Time}_B / \text{Execution Time}_A$
 $= 15\text{s} / 10\text{s} = 1.5$
 - So A is 1.5 times faster than B

Measuring Execution Time

- Elapsed time
 - Total response time, including all aspects
 - Processing, I/O, OS overhead, idle time
 - Determines system performance
- CPU time
 - Time spent processing a given job
 - Discounts I/O time, other jobs' shares
 - Comprises user CPU time and system CPU time
 - Different programs are affected differently by CPU and system performance

CPU Clocking

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



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

CPU Time

$$\begin{aligned}\text{CPU Time} &= \text{CPU Clock Cycles} \times \text{Clock Cycle Time} \\ &= \frac{\text{CPU Clock Cycles}}{\text{Clock Rate}}\end{aligned}$$

- 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 $1.2 \times$ clock cycles
- How fast must Computer B clock be?

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

$$\begin{aligned}\text{Clock Cycles}_A &= \text{CPU Time}_A \times \text{Clock Rate}_A \\ &= 10s \times 2\text{GHz} = 20 \times 10^9\end{aligned}$$

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

Instruction Count and CPI

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

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

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

- 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 is faster, and by how much?

$$\begin{aligned}\text{CPUTime}_A &= \text{Instruction Count} \times \text{CPI}_A \times \text{Cycle Time}_A \\ &= 1 \times 2.0 \times 250\text{ps} = 1 \times 500\text{ps} \leftarrow \text{A is faster...}\end{aligned}$$

$$\begin{aligned}\text{CPUTime}_B &= \text{Instruction Count} \times \text{CPI}_B \times \text{Cycle Time}_B \\ &= 1 \times 1.2 \times 500\text{ps} = 1 \times 600\text{ps}\end{aligned}$$

$$\frac{\text{CPUTime}_B}{\text{CPUTime}_A} = \frac{1 \times 600\text{ps}}{1 \times 500\text{ps}} = 1.2 \leftarrow \text{...by this much}$$

CPI in More Detail

- If different instruction classes take different numbers of cycles

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

- Weighted average CPI

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

Relative frequency

CPI Example

- Alternative compiled code sequences 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 = 5

- Clock Cycles
 $= 2 \times 1 + 1 \times 2 + 2 \times 3$
 $= 10$
- Avg. CPI = $10/5 = 2.0$

- Sequence 2: IC = 6

- Clock Cycles
 $= 4 \times 1 + 1 \times 2 + 1 \times 3$
 $= 9$
- Avg. CPI = $9/6 = 1.5$

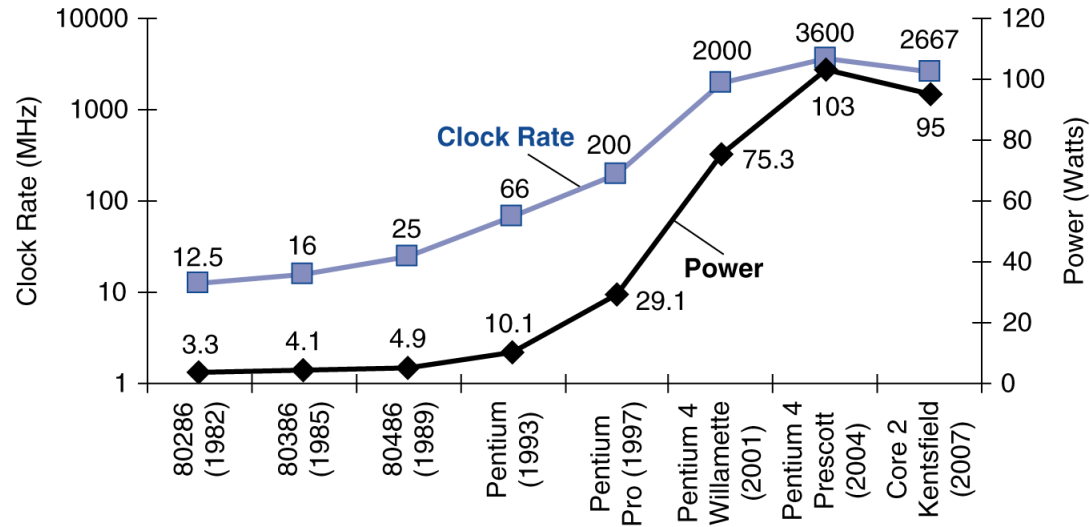
Performance Summary

The BIG Picture

$$\text{CPU Time} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clock cycle}}$$

- Performance depends on
 - Algorithm: affects IC, possibly CPI
 - Programming language: affects IC, CPI
 - Compiler: affects IC, CPI
 - Instruction set architecture: affects IC, CPI, T_c

Power Trends



- In CMOS IC technology

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

×30

5V → 1V

×1000

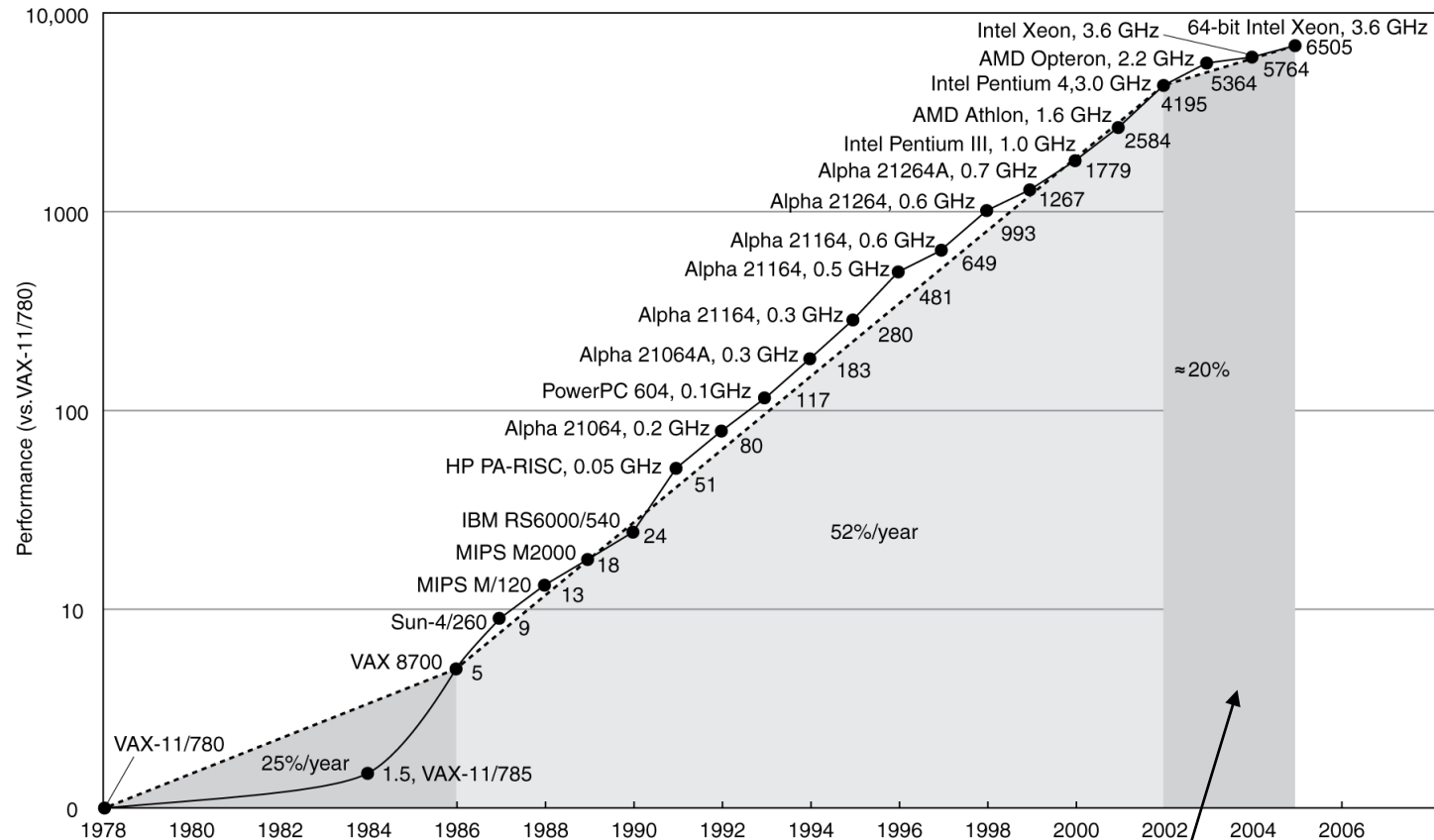
Reducing Power

- Suppose a new CPU has
 - 85% of capacitive load of old CPU
 - 15% voltage and 15% frequency reduction

$$\frac{P_{\text{new}}}{P_{\text{old}}} = \frac{[C_{\text{old}} \times 0.85] \times (V_{\text{old}} \times 0.85)^2 \times [F_{\text{old}} \times 0.85]}{C_{\text{old}} \times V_{\text{old}}^2 \times F_{\text{old}}} = 0.85^4 = 0.52$$

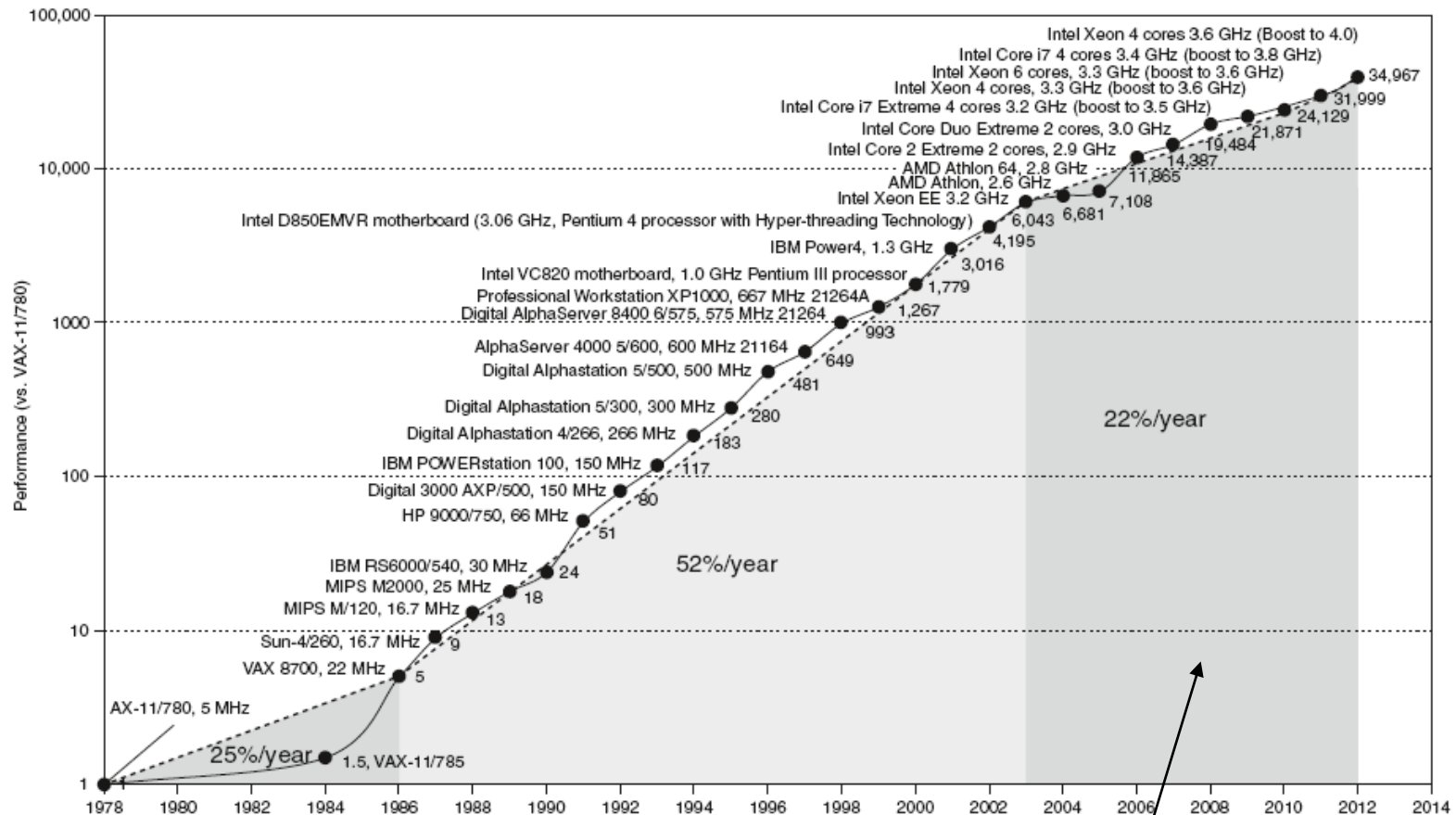
- The power wall
 - We can't reduce voltage further
 - We can't remove more heat
- How else can we improve performance?

Uniprocessor Performance



Constrained by power, instruction-level parallelism, memory latency

Uniprocessor Performance



Constrained by power, instruction-level parallelism, memory latency

Multiprocessors

- Multicore 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 and synchronization

SPEC CPU Benchmark

- Programs used to measure performance
 - Supposedly typical of actual workload
- Standard Performance Evaluation Corp (SPEC)
 - Develops benchmarks for CPU, I/O, Web, ...
- SPEC CPU2006
 - Elapsed time to execute a selection of programs
 - Negligible I/O, so focuses on CPU performance
 - Normalize relative to reference machine
 - Summarize as geometric mean of performance ratios
 - CINT2006 (integer) and CFP2006 (floating-point)

$$\sqrt[n]{\prod_{i=1}^n \text{Execution time ratio}_i}$$

CINT2006 for Opteron X4 2356

Name	Description	IC×10 ⁹	CPI	Tc (ns)	Exec time	Ref time	SPECratio
perl	Interpreted string processing	2,118	0.75	0.40	637	9,777	15.3
bzip2	Block-sorting compression	2,389	0.85	0.40	817	9,650	11.8
gcc	GNU C Compiler	1,050	1.72	0.47	24	8,050	11.1
mcf	Combinatorial optimization	336	10.00	0.40	1,345	9,120	6.8
go	Go game (AI)	1,658	1.09	0.40	721	10,490	14.6
hmmer	Search gene sequence	2,783	0.80	0.40	890	9,330	10.5
sjeng	Chess game (AI)	2,176	0.96	0.48	37	12,100	14.5
libquantum	Quantum computer simulation	1,623	1.61	0.40	1,047	20,720	19.8
h264avc	Video compression	3,102	0.80	0.40	993	22,130	22.3
omnetpp	Discrete event simulation	587	2.94	0.40	690	6,250	9.1
astar	Games/path finding	1,082	1.79	0.40	773	7,020	9.1
xalancbmk	XML parsing	1,058	2.70	0.40	1,143	6,900	6.0
Geometric mean							11.7

High cache miss rates



SPEC Power Benchmark

- Power consumption of server at different workload levels
 - Performance: ssj_ops/sec
 - Power: Watts (Joules/sec)

$$\text{Overall ssj_opsper Watt} = \left(\sum_{i=0}^{10} \text{ssj_ops}_i \right) / \left(\sum_{i=0}^{10} \text{power}_i \right)$$

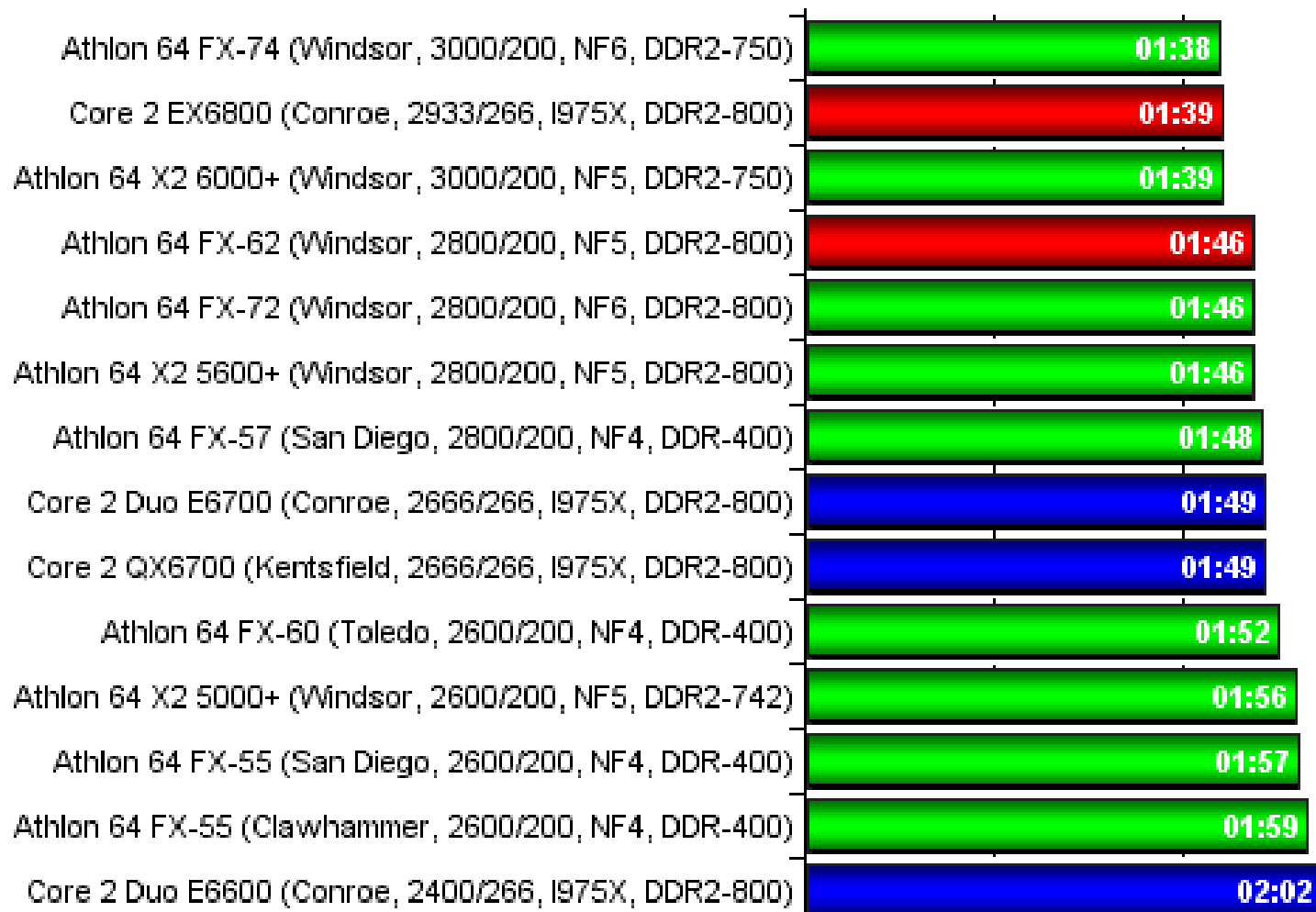
SPECpower_ssj2008 for X4

Target Load %	Performance (ssj_ops/sec)	Average Power (Watts)
100%	231,867	295
90%	211,282	286
80%	185,803	275
70%	163,427	265
60%	140,160	256
50%	118,324	246
40%	920,35	233
30%	70,500	222
20%	47,126	206
10%	23,066	180
0%	0	141
Overall sum	1,283,590	2,605
$\Sigma \text{ssj_ops} / \Sigma \text{power}$		493

Benchmark

- Performance best determined by running a real application
 - Use programs typical of expected workload
 - Or, typical of expected class of applications
e.g., compilers/editors, scientific applications, graphics, etc.
- Small benchmarks
 - nice for architects and designers
 - easy to standardize
 - can be abused
- SPEC (System Performance Evaluation Cooperative)
 - companies have agreed on a set of real program and inputs
 - valuable indicator of performance (and compiler technology)
 - can still be abused

Benchmark



Review

- Performance is specific to a particular program/s
 - Total execution time is a consistent summary of performance
- For a given architecture performance increases come from:
 - increases in clock rate (without adverse CPI affects)
 - improvements in processor organization that lower CPI
 - compiler enhancements that lower CPI and/or instruction count
 - Algorithm/Language choices that affect instruction count
- Pitfall: expecting improvement in one aspect of a machine's performance to affect the total performance

Amdahl's Law

The execution time after making an improvement to the system is given by

$$T_{\text{improved}} = \frac{T_{\text{affected}}}{\text{improvement factor}} + T_{\text{unaffected}}$$

Amdahl's Law

Suppose that program runs 100 seconds on a machine and multiplication instructions take 80% of the total time. How much do I have to improve the speed of multiplication if I want my program to run 5 times faster?

$$20 = \frac{80}{n} + 20$$

- Can't be done!

Fallacy: Low Power at Idle

- Look back at X4 power benchmark
 - At 100% load: 295W
 - At 50% load: 246W (83%)
 - At 10% load: 180W (61%)
- Google data center
 - Mostly operates at 10% – 50% load
 - At 100% load less than 1% of the time
- Consider designing processors to make power proportional to load

Concluding Remarks

- Cost/performance is improving
 - Due to underlying technology development
- Hierarchical layers of abstraction
 - In both hardware and software
- Instruction set architecture
 - The hardware/software interface
- Execution time: the best performance measure
- Power is a limiting factor
 - Use parallelism to improve performance