

CSE 305

Computer Architecture

Computer Architecture

- Course Teachers
 - Rifat Shahriyar (rifat1816@gmail.com)
 - Johra Muhammad Moosa
- Textbook
 - Computer Organization and Design (The Hardware/Software Interface)
 - David A. Patterson and John L. Hennessy (5th Ed.)
- Homepage
 - <http://rifatshahriyar.github.io/CSE305.html>

Chapter 1

Computer Abstractions and Technology

The Computer Revolution

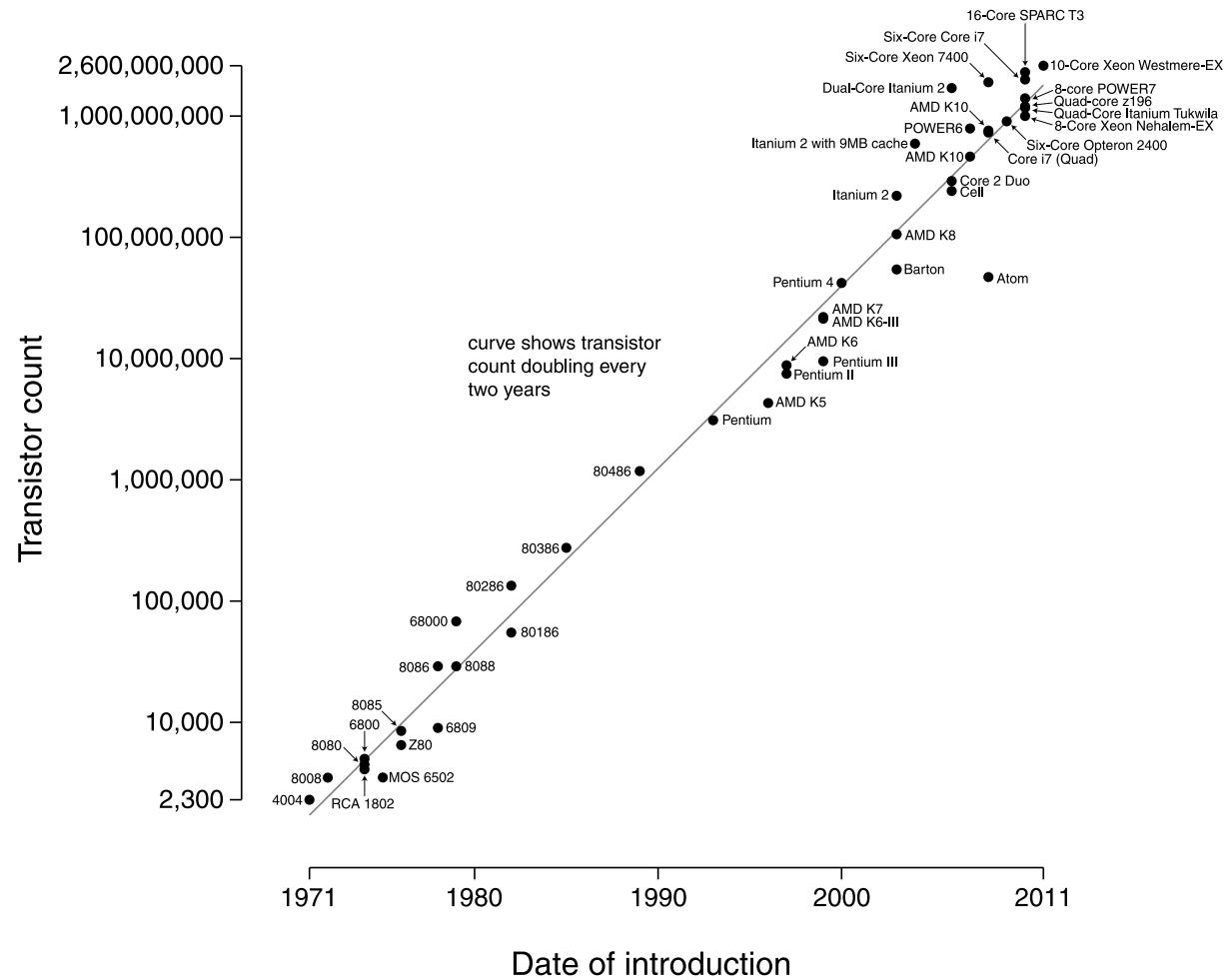
- Progress in computer technology
 - Underpinned by Moore's Law
- Makes novel applications feasible
 - Computers in automobiles
 - Cell phones
 - Human genome project
 - World Wide Web
 - Search Engines
- Computers are pervasive

Moore's Law

- The number of transistors in a dense integrated circuit doubles approximately every two years
- Named after Gordon Moore, the co-founder of Intel
 - 1965 - described a doubling every year
 - 1975 - revised the forecast to doubling every two years
 - The period is often quoted as 18 months

Moore's Law

Microprocessor Transistor Counts 1971-2011 & Moore's Law



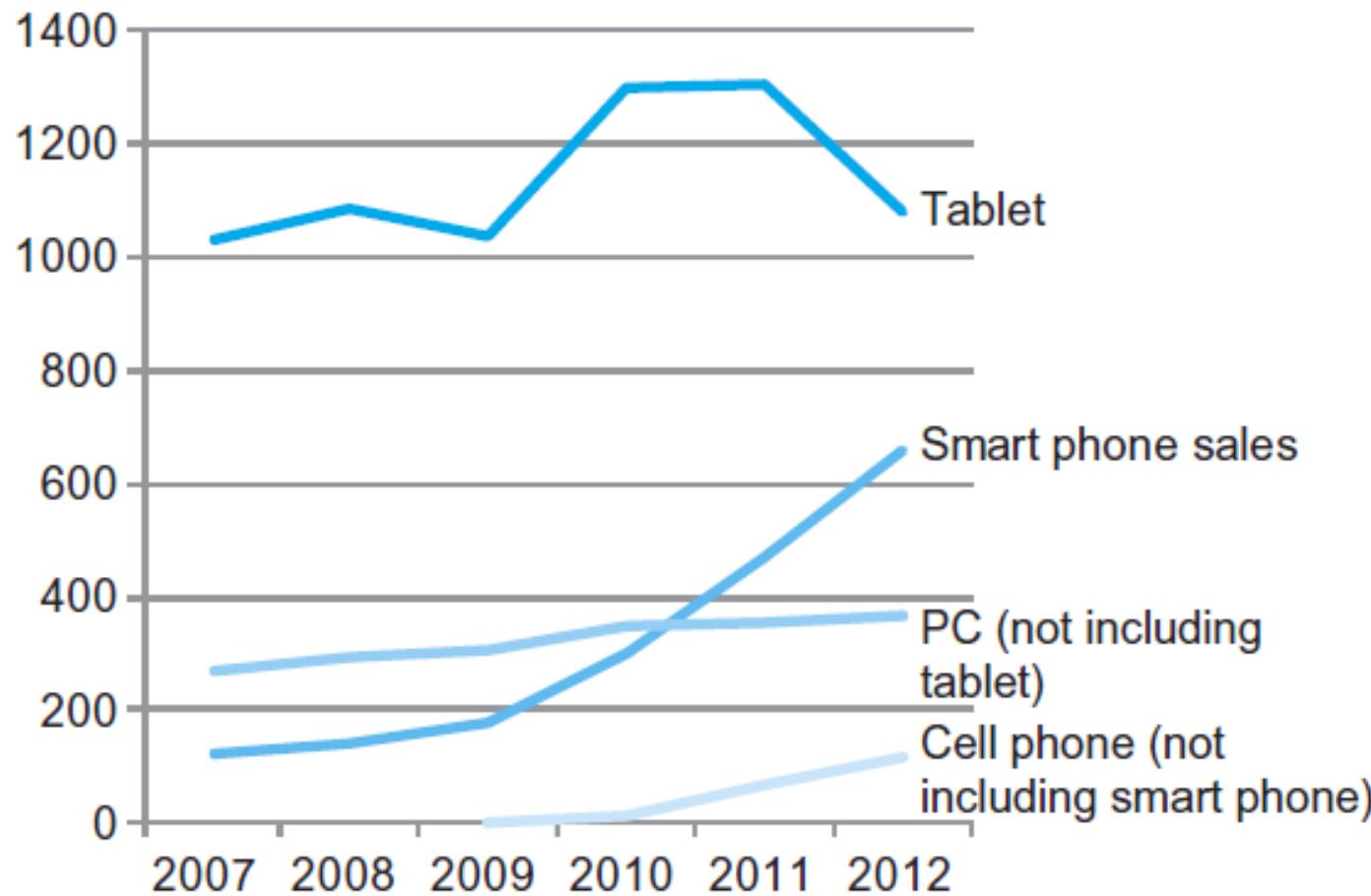
Classes of Computers

- Personal 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

Classes of Computers

- Supercomputers
 - High-end scientific and engineering calculations
 - Highest capability but represent a small fraction of the overall computer market
- 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

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

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

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*



MOORE'S LAW



ABSTRACTION



COMMON CASE FAST



PARALLELISM



PIPELINING



PREDICTION

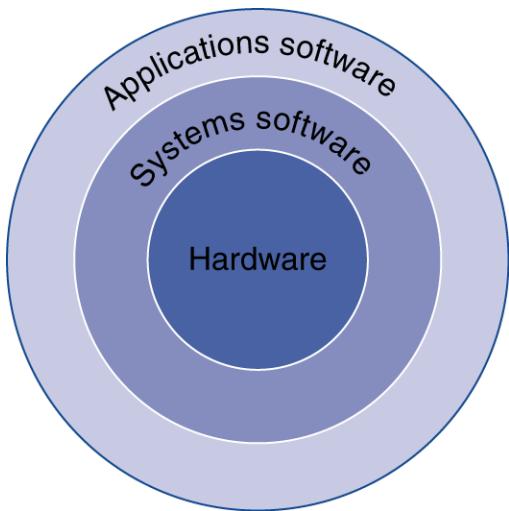


HIERARCHY



DEPENDABILITY

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
```

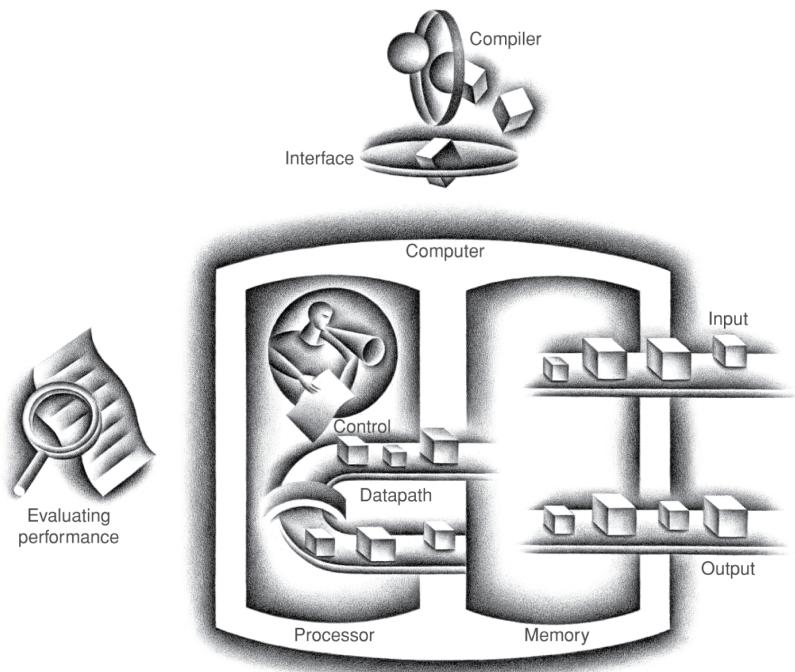
Assembler

Binary machine language program (for MIPS)

```
000000001010000100000000000011000
000000000000110000001100000100001
1000110001100010000000000000000000
1000110011110010000000000000000000
1010110011110010000000000000000000
1010110001100010000000000000000000
0000001111000000000000000000000000
```

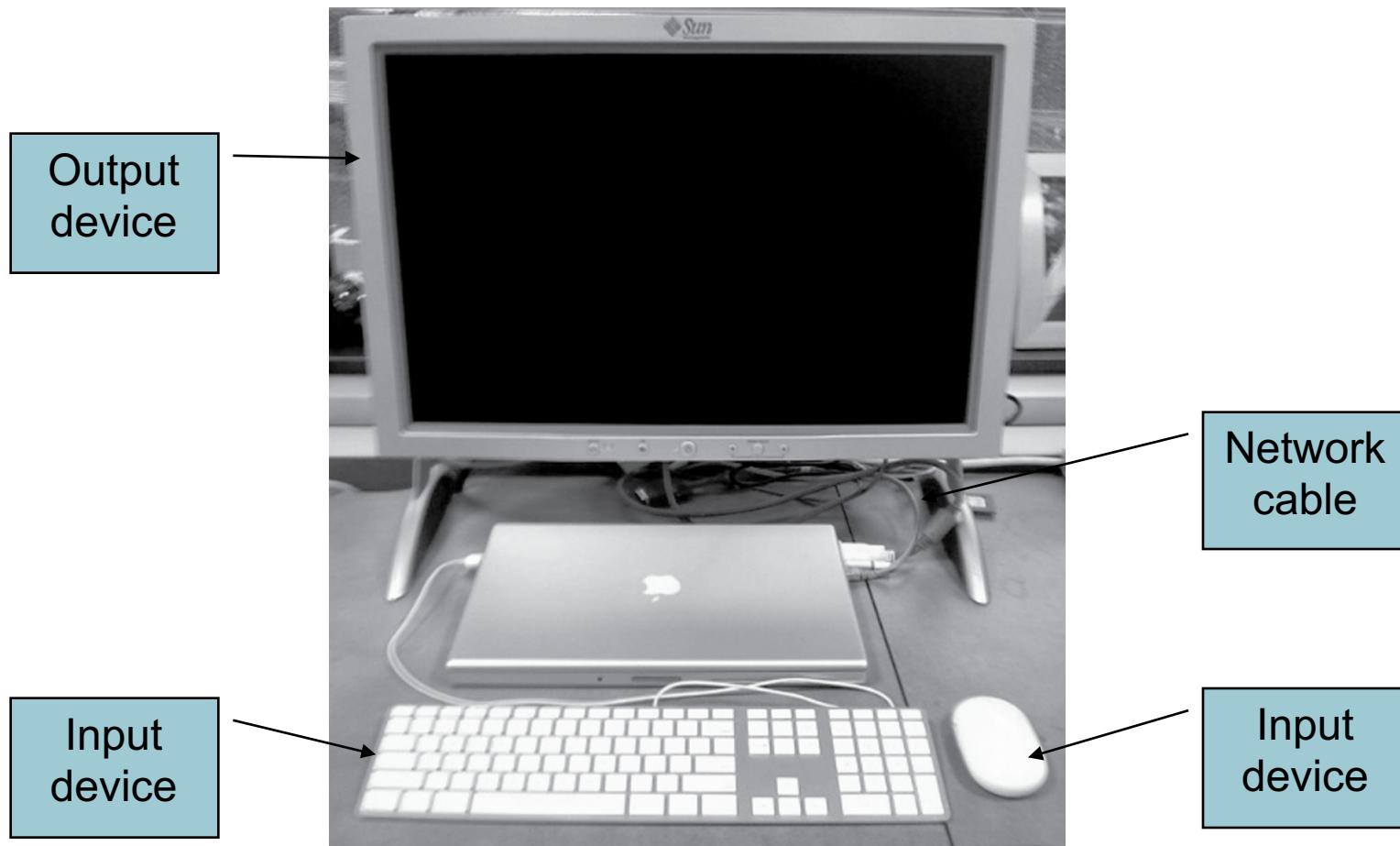
Components of a Computer

The BIG Picture



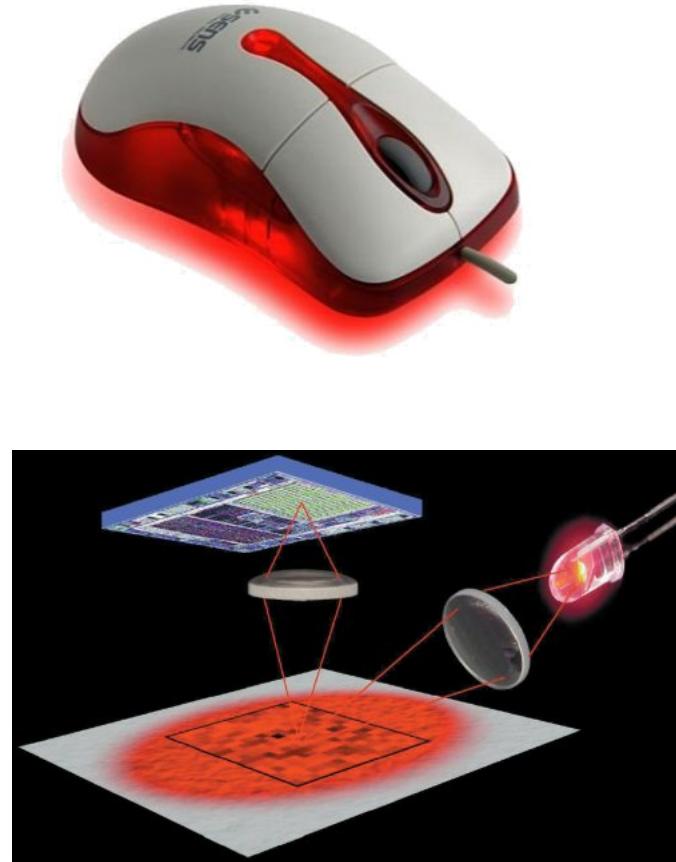
- Same components for all kinds of computer
 - Desktop, server, embedded
- Input/output includes
 - User-interface devices
 - Display, keyboard, mouse
 - Storage devices
 - Hard disk, CD/DVD, flash
 - Network adapters
 - For communicating with other computers

Anatomy of a Computer



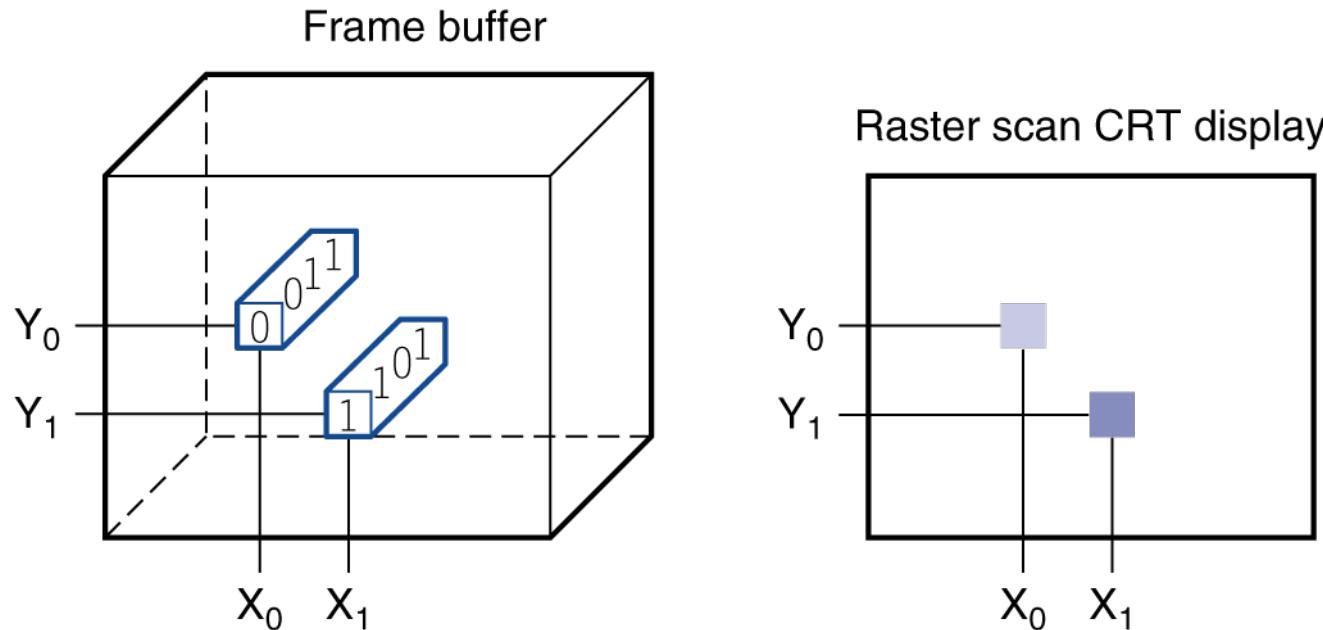
Anatomy of a Mouse

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



Through the Looking Glass

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

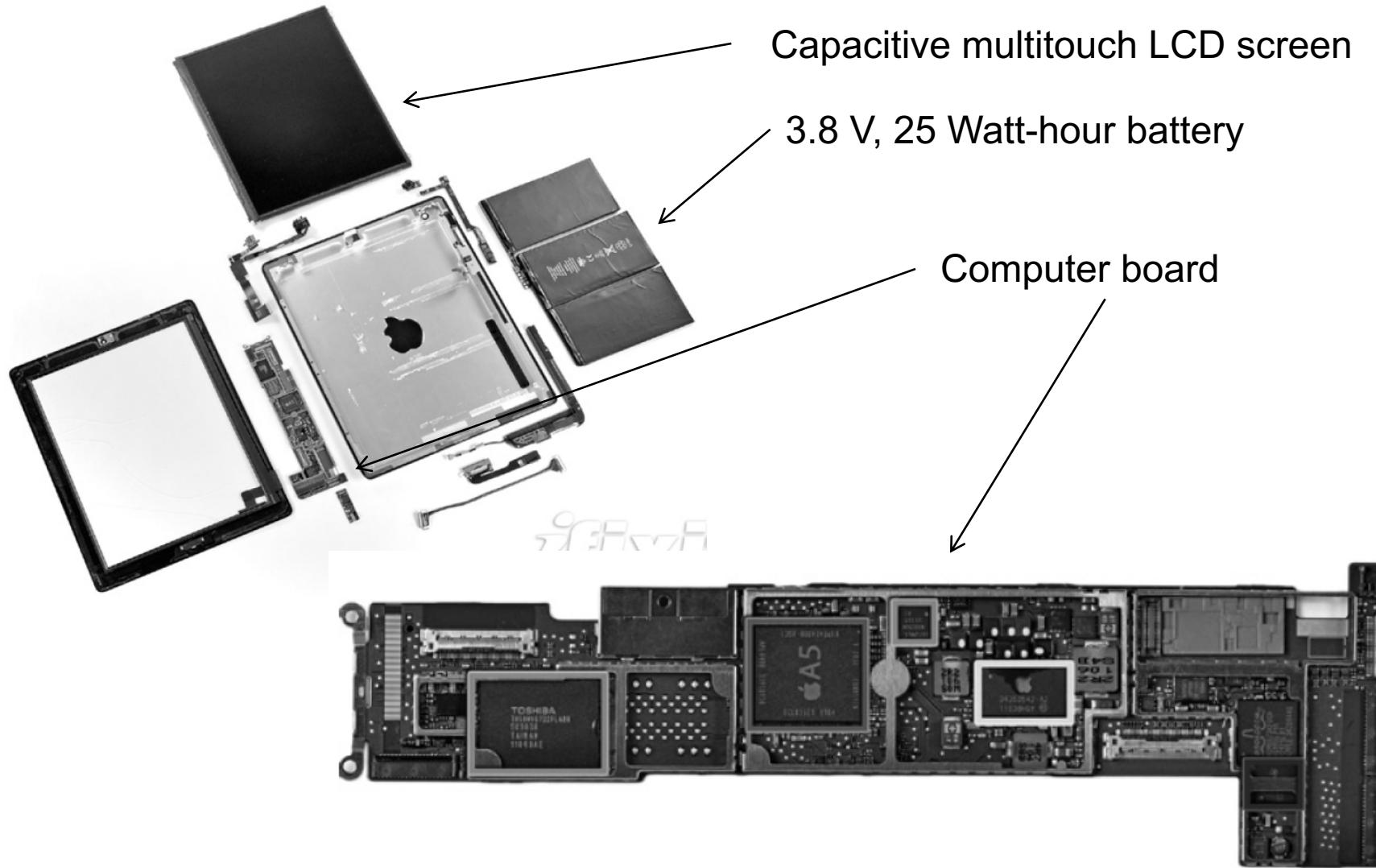


Touchscreen

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



Opening the Box



Inside the Processor (CPU)

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

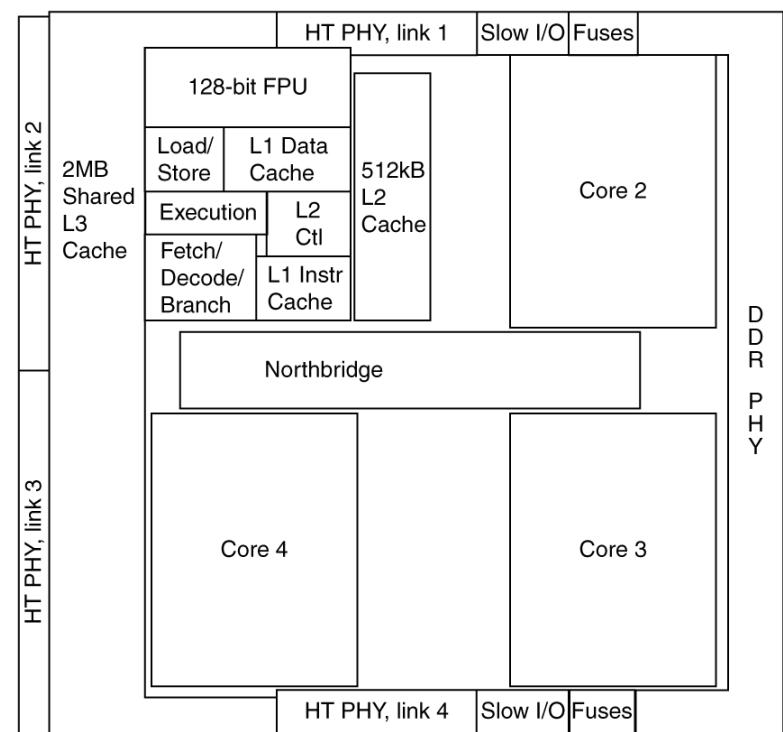
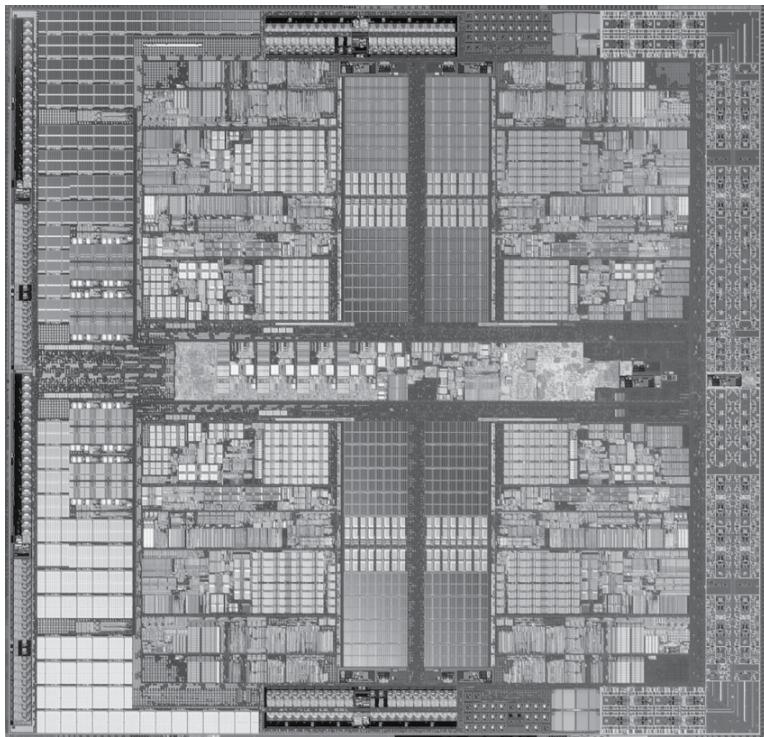
Inside the Processor

■ Apple A5



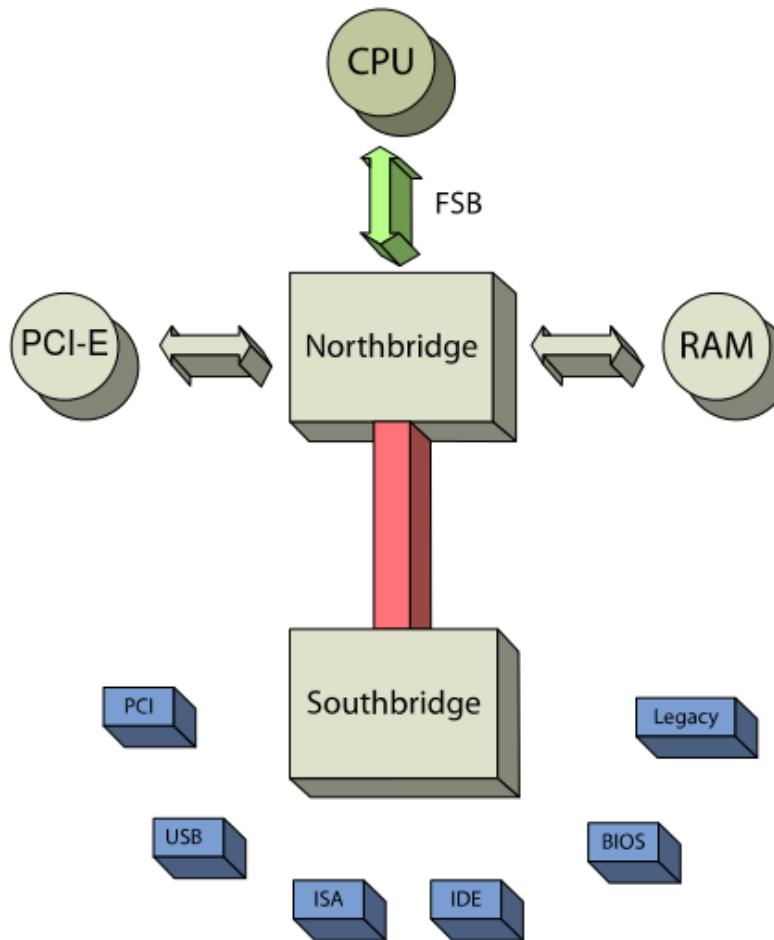
Inside the Processor

- AMD Barcelona: 4 processor cores



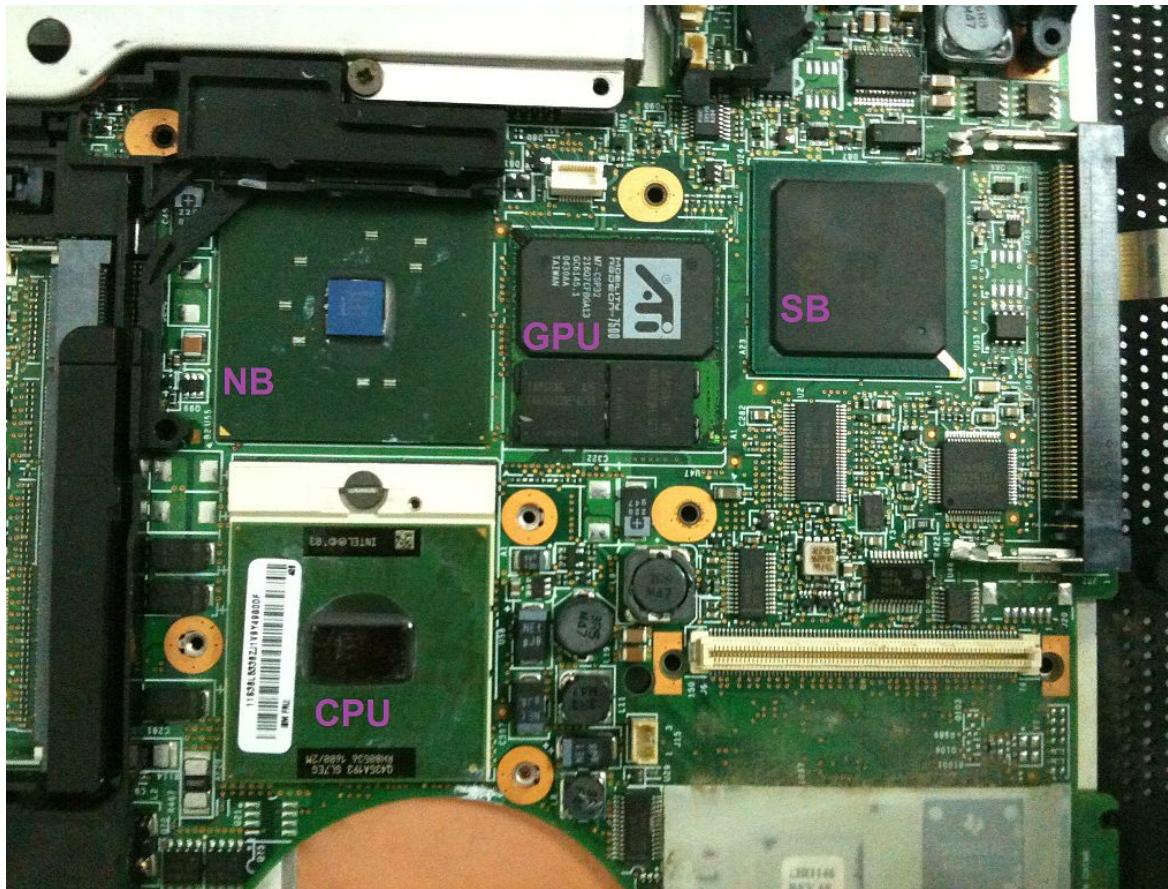
Inside the Processor

■ Northbridge and Southbridge



Inside the Processor

■ Northbridge and Southbridge



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

Abstractions

Problem



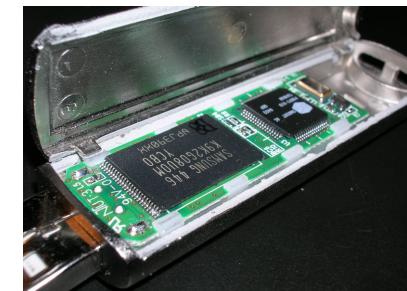
Operating System

*Interface between
Architecture
Software and Hardware*

Circuits

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)



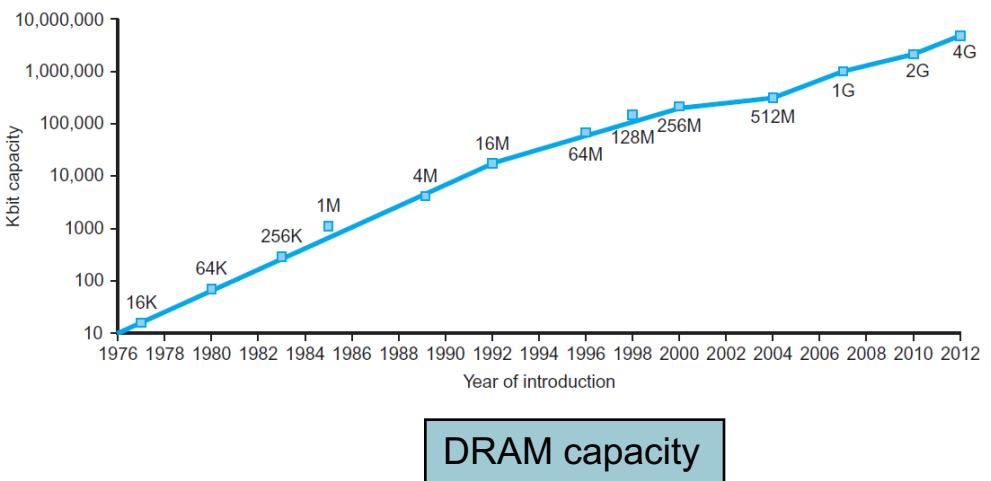
Networks

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



Technology Trends

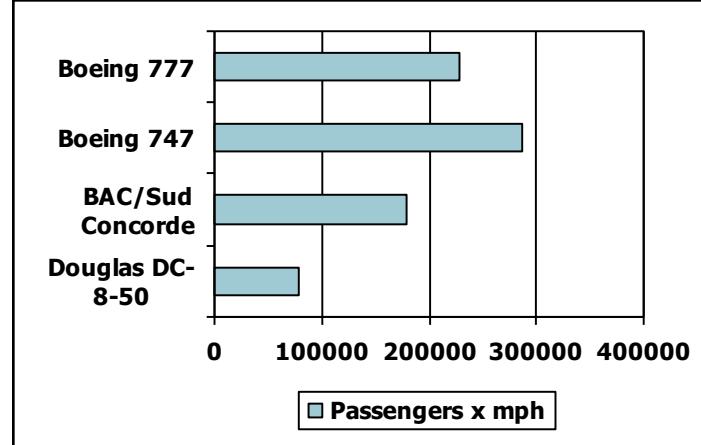
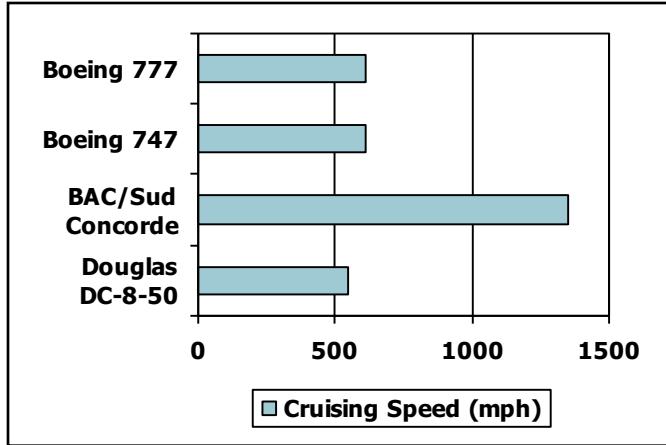
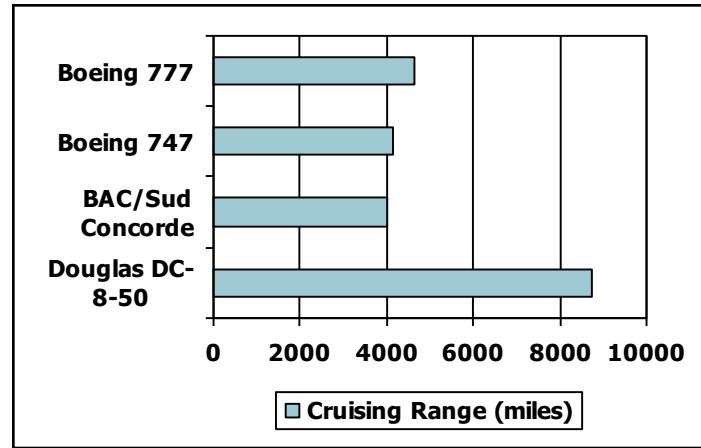
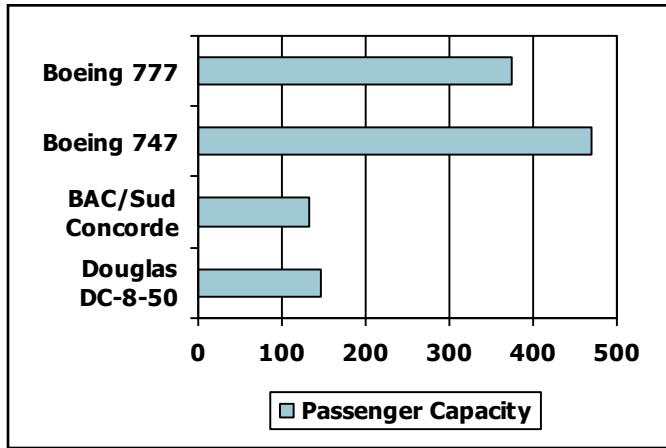
- Electronics technology continues to evolve
 - Increased capacity and performance
 - Reduced cost



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
2013	Ultra large scale IC	250,000,000,000

Defining Performance

- Which airplane has the best performance?



Response Time and Throughput

- Response time
 - How long it takes to do a task
 - Individual computer users interest
- Throughput
 - Total work done per unit time (tasks per hour)
 - Datacenter managers interest
- How are response time and throughput affected by
 - Replacing the processor with a faster version
 - Adding additional processors to a system that uses multiple processors for separate tasks

Relative Performance

- Define Performance = 1/Execution Time
- “X is n time faster than Y”

$$\begin{aligned}\text{Performance}_x / \text{Performance}_y \\ = \text{Execution time}_y / \text{Execution time}_x = n\end{aligned}$$

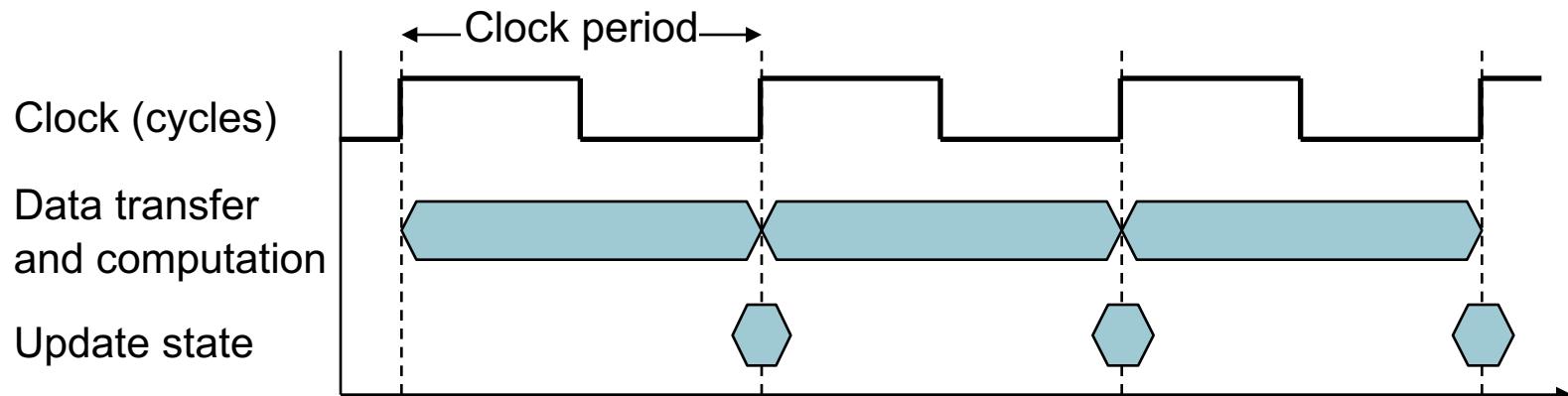
- Example: time taken to run a program
 - 10s on A, 15s on B
 - $\text{Execution Time}_B / \text{Execution Time}_A$
 $= 15s / 10s = 1.5$
 - So A is 1.5 times faster than B

Measuring Execution Time

- Wall Clock 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

CPU Time = CPU Clock Cycles \times Clock Cycle Time

$$= \frac{\text{CPU Clock Cycles}}{\text{Clock Rate}}$$

- Performance improved by
 - Reducing number of clock cycles
 - Increasing clock rate
 - Hardware designer must often trade off clock rate against cycle count
 - Many techniques that decrease the number of clock cycles may also increase the clock cycle time

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

Clock Cycles = Instruction Count \times Cycles per Instruction

CPU Time = Instruction Count \times CPI \times 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?

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

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

A is faster...

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

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

$$\frac{\text{CPU Time}_B}{\text{CPU Time}_A} = \frac{I \times 600\text{ps}}{I \times 500\text{ps}} = 1.2$$

...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

A compiler designer is trying to decide between two code sequences for a particular computer. The hardware designers have supplied the following facts:

	CPI for each Instruction class		
	A	B	C
CPI	1	2	3

For a particular high-level language statement, the compiler writer is considering two code sequences that require the following instruction counts:

Code sequence	Instruction counts for each Instruction class		
	A	B	C
1	2	1	2
2	4	1	1

Which code sequence executes the most instructions? Which will be faster? What is the CPI for each sequence?

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 = $9/6 = 1.5$

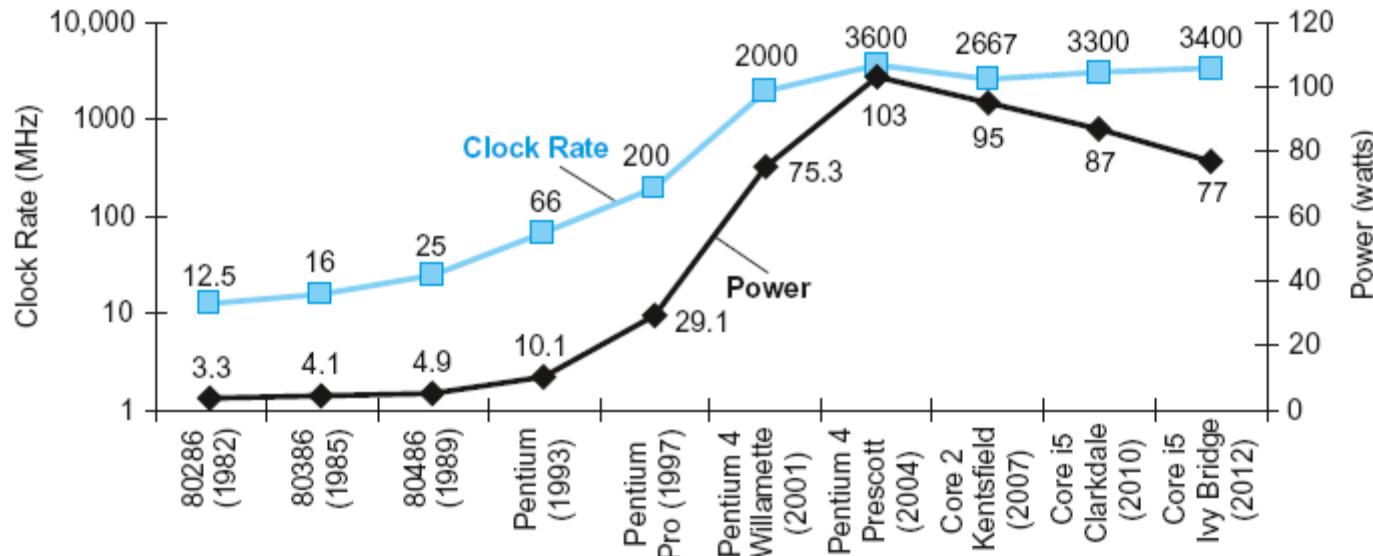
Performance Summary

The BIG Picture

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

- 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

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

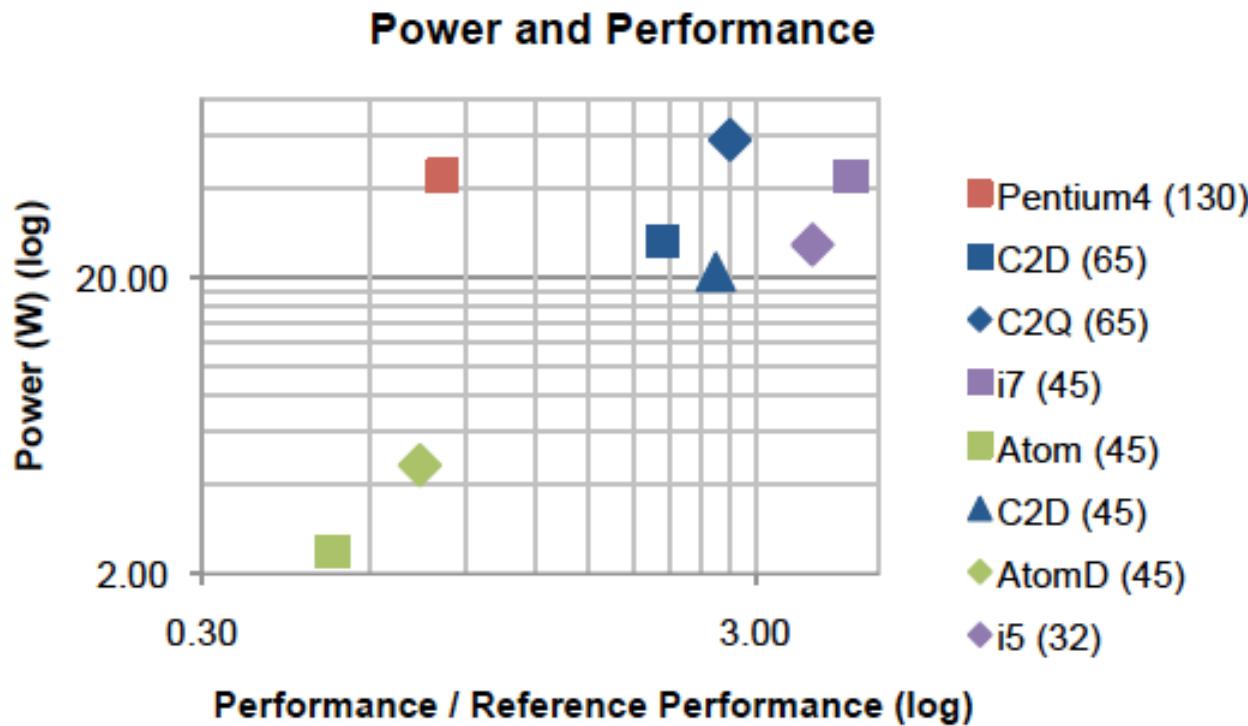
Reducing Power

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

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

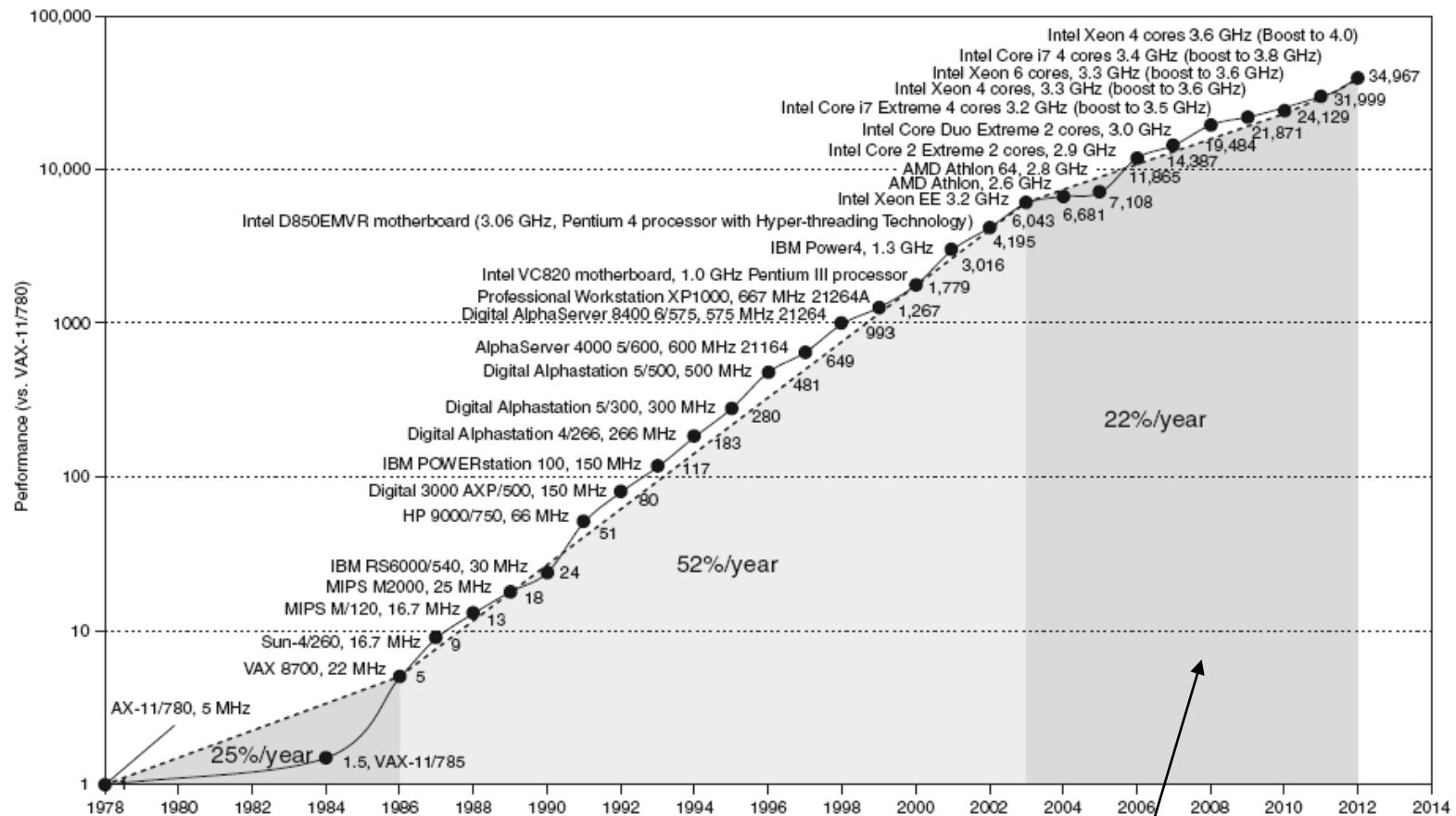
- The power wall
 - We can't reduce voltage further
 - We can't remove more heat
- How else can we improve performance?

Power vs. Performance



(a) Power / performance tradeoff by processor.

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 Intel Core i7 920

Description	Name	Instruction Count x 10 ⁹	CPI	Clock cycle time (seconds x 10 ⁻⁹)	Execution Time (seconds)	Reference Time (seconds)	SPECratio
Interpreted string processing	perl	2252	0.60	0.376	508	9770	19.2
Block-sorting compression	bzip2	2390	0.70	0.376	629	9650	15.4
GNU C compiler	gcc	794	1.20	0.376	358	8050	22.5
Combinatorial optimization	mcf	221	2.66	0.376	221	9120	41.2
Go game (AI)	go	1274	1.10	0.376	527	10490	19.9
Search gene sequence	hmmer	2616	0.60	0.376	590	9330	15.8
Chess game (AI)	sjeng	1948	0.80	0.376	586	12100	20.7
Quantum computer simulation	libquantum	659	0.44	0.376	109	20720	190.0
Video compression	h264avc	3793	0.50	0.376	713	22130	31.0
Discrete event simulation library	omnetpp	367	2.10	0.376	290	6250	21.5
Games/path finding	astar	1250	1.00	0.376	470	7020	14.9
XML parsing	xalancbmk	1045	0.70	0.376	275	6900	25.1
Geometric mean	-	-	-	-	-	-	25.7

Pitfall: Amdahl's Law

- Improving an aspect of a computer and expecting a proportional improvement in overall performance

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

- Example: multiply accounts for 80s/100s
 - How much improvement in multiply performance to get 5 × overall?

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

- Can't be done!

- Corollary: make the common case fast

Fallacy: Low Power at Idle

- Look back at i7 power benchmark
 - At 100% load: 258W
 - At 50% load: 170W (66%)
 - At 10% load: 121W (47%)
- 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

Pitfall: MIPS as a Performance Metric

- MIPS: Millions of Instructions Per Second
 - Doesn't account for
 - Differences in ISAs between computers
 - Differences in complexity between instructions

$$\begin{aligned}\text{MIPS} &= \frac{\text{Instruction count}}{\text{Execution time} \times 10^6} \\ &= \frac{\text{Instruction count}}{\frac{\text{Instruction count} \times \text{CPI}}{\text{Clock rate}} \times 10^6} = \frac{\text{Clock rate}}{\text{CPI} \times 10^6}\end{aligned}$$

- CPI varies between programs on a given CPU

Pitfall: MIPS as a Performance Metric

- **Computer A**
 - 10 Billion instructions, 4GHz Clock rate
 - CPI = 1
- **Computer B**
 - 8 Billion instructions, 4GHz Clock rate
 - CPI = 1.1
- 1) Which has the highest MIPS rating?
- 2) Which is faster?

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