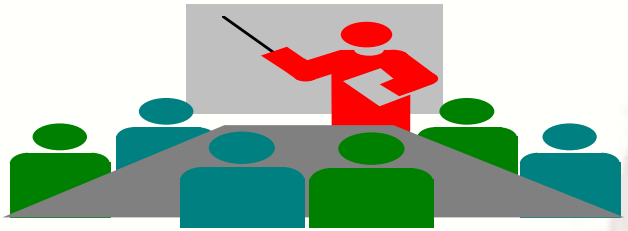
浙江大学 ZHEJIANG UNIVERSITY



计算机组成与设计

Chapter 1

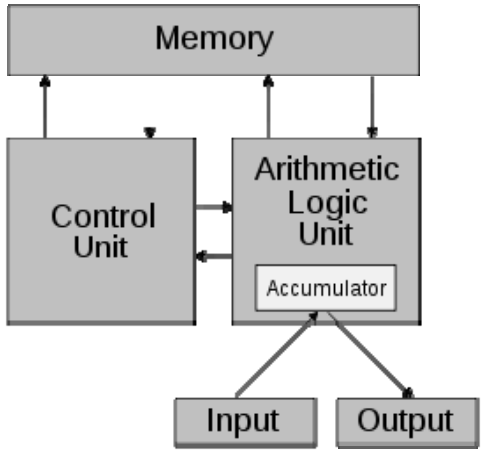
Computer Abstractions and Technology

刘海风
Zhejiang University
haifengliu@zju.edu.cn

1.0 History of Computer Development

- **The first electronic computers**
  - ENIAC (Electronic Numerical Integrator and Calculator)
    - J. Presper Eckert and John Mauchly
    - Publicly known in 1946
    - 30 tons, 80 feet long, 8.5 feet high, several feet wide
    - 18,000 vacuum tubes
    - Decimal (not binary)
    - 20 accumulators of 10 digits
    - Programmed manually by switches

- EDVAC (Electronic Discrete Variable Automatic Computer)
  - John von Neumann's memo about stored-program computer
  - von Neumann Computer
- EDSAC (Electronic Delay Storage Automatic Calculator)
  - Operational in 1949
  - First full-scale, operational, stored-program computer in the world
- Other computers(omitted)
- Harvard architecture:
  Program memory and data memory are independent.



Generations of Computer

- **Generation 1: 1946-1957**
  - Vacuum tube
- **Generation 2: 1958-1964**
  - Transistor -
- **Generation 3: 1965-1970**
  - Small scale integration, Up to 100 devices on a chip
  - Medium scale integration , 100-3,000 devices on a chip
- **Generation 4: 1971-**
  - Large scale integration , 3000 - 100000 devices on a chip
  - Very large scale integration , 100,000 - 100,000,000 devices on a chip, 1978-
  - Ultra large scale integration, Over 100,000,000 devices on a chip

Electronic Computers -- Generation 5

- **Dominant technologies**
  - Processors: Mass-produced microprocessors
  - Memroy: SRAM, DRAM
  - Compilers
- **RISC processors**
  - MIPS (Silicon Graphics)
  - PA-RISC (Hewlett Packard)
  - SPARC (Sun Microsystems)
  - Alpha (Digital Equipment)
  - PowerPC(Apple, Motorala, IBM)
  - Motorala 88000
  - I860, i960 (Intel)
- **CISC processors:**
  - 80x86, Pentium, Penium Pro, Pentium II & III (Intel)

Contents of Chapter 1

1.1 Introduction

- **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**

Present

- **Present science fiction computer applications**
  - Cashless society
  - Automated intelligent highways
  - Genuinely ubiquitous computing:

    No one carries computers because they are available everywhere.

Tomorrow

- **Tomorrow's science fiction computer applications**
  - self-driving car
  - artificial intelligent
  - brain-computer interface (BCI)
  - Human Brain Project (HBP)
  -

Classes of Computers

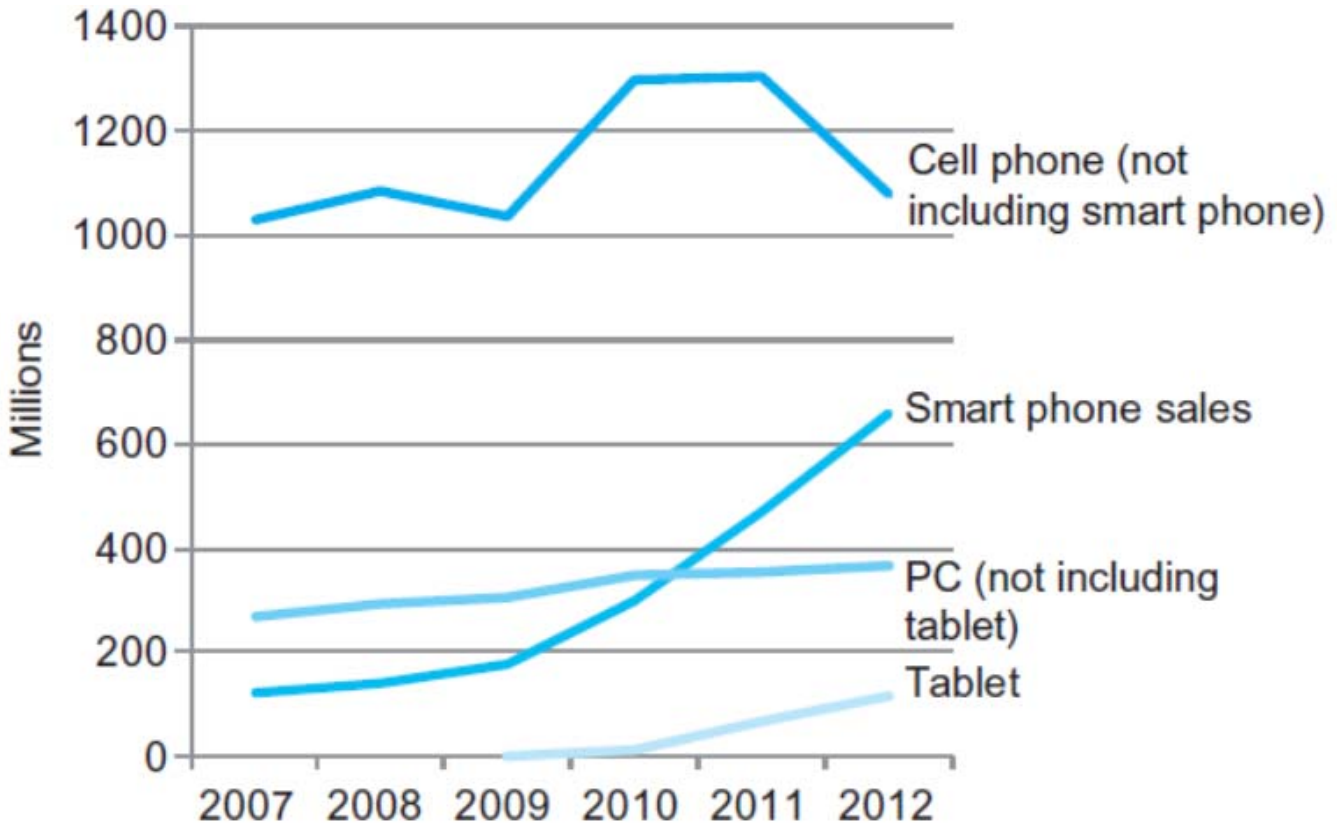
- **Personal computers**
  - General purpose, variety of software
  - Subject to cost/performance tradeoff
- **Server computers**
  - Network based
  - High capacity, performance, reliability
  - Range from small servers to building sized

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

Hardware or software component	How this component affects performance	Where is this topic covered?
Algorithm	Determines both the number of source-level statements and the number of I/O operations executed	Other books!
Programming language, compiler, and architecture	Determines the number of machine instructions for each source-level statements	Chapter 2 and 3
Processor and memory system	Determines how fast instructions can be executed	Chapter 4，5，6
I/O system(hardware and operating system)	Determines how fast I/O operations may be executed	Chapter 4，5，6

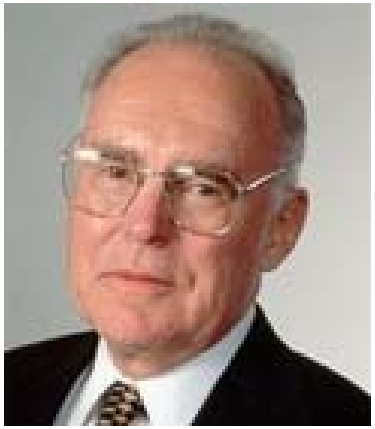
1.2 Eight Great Ideas

- 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

Eight Great Ideas --1 in Computer Architecture

Design for Moore'Law 1965

- The integrate circuit resource double every 18-24 months.

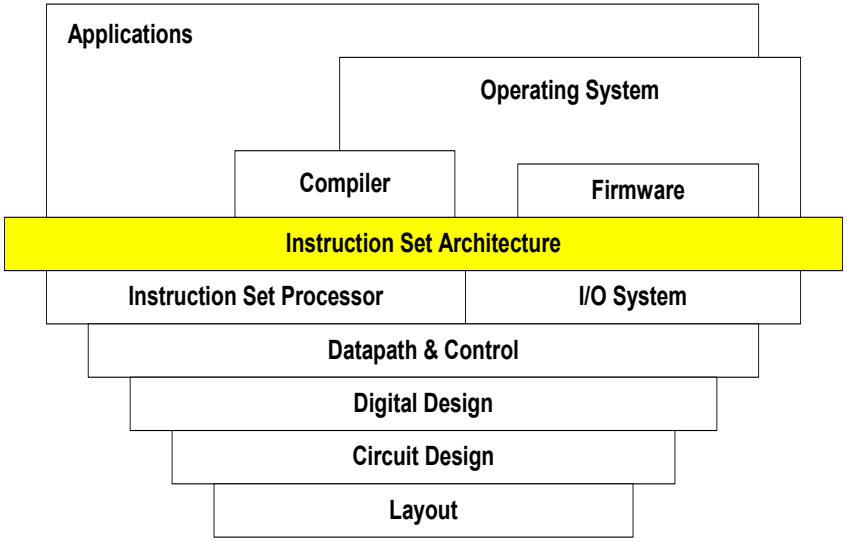


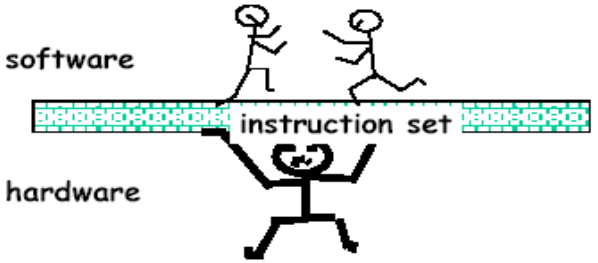
Gordon Moore

Eight Great Ideas –2 in Computer Architecture

Use Abstraction to Simplify Design

- Lower-level details are hidden to higher levels
- **Instruction set architecture** ---- the interface between hardware and lowest-level software
- Many implementations of varying cost and performance can run identical software





Assembly Language
|||
Instruction Set Architecture
|||
Machine Language

Eight Great Ideas –3 in Computer Architecture

- **Make the common case fast**
  - Common case is often simple

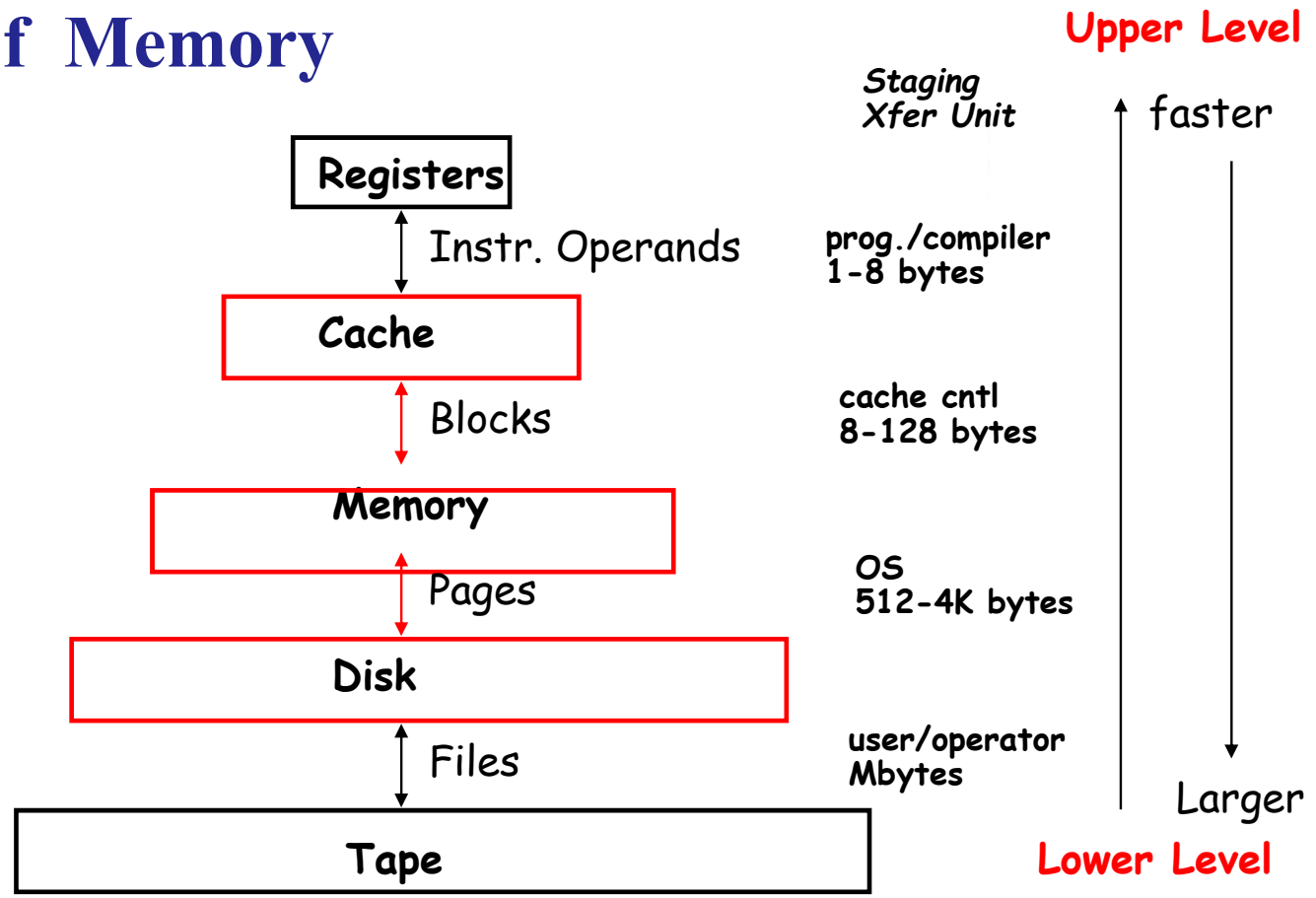
vs

Eight Great Ideas –4.5.6 in Computer Architecture

- Performance via Parallelism
- Performance via Pipelining
- Performance via Prediction

Eight Great Ideas –7 in Computer Architecture

Hierarchy of Memory

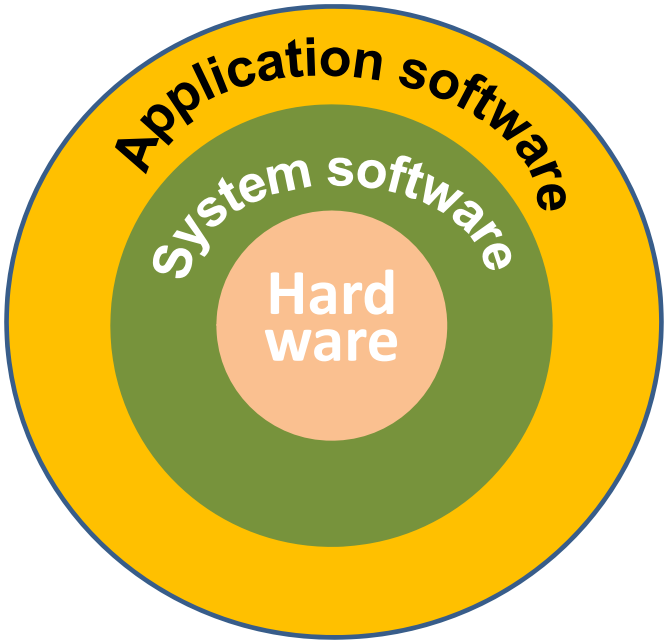


Eight Great Ideas –8 in Computer Architecture

- Dependability via Redundancy

1.3 Below Your Program

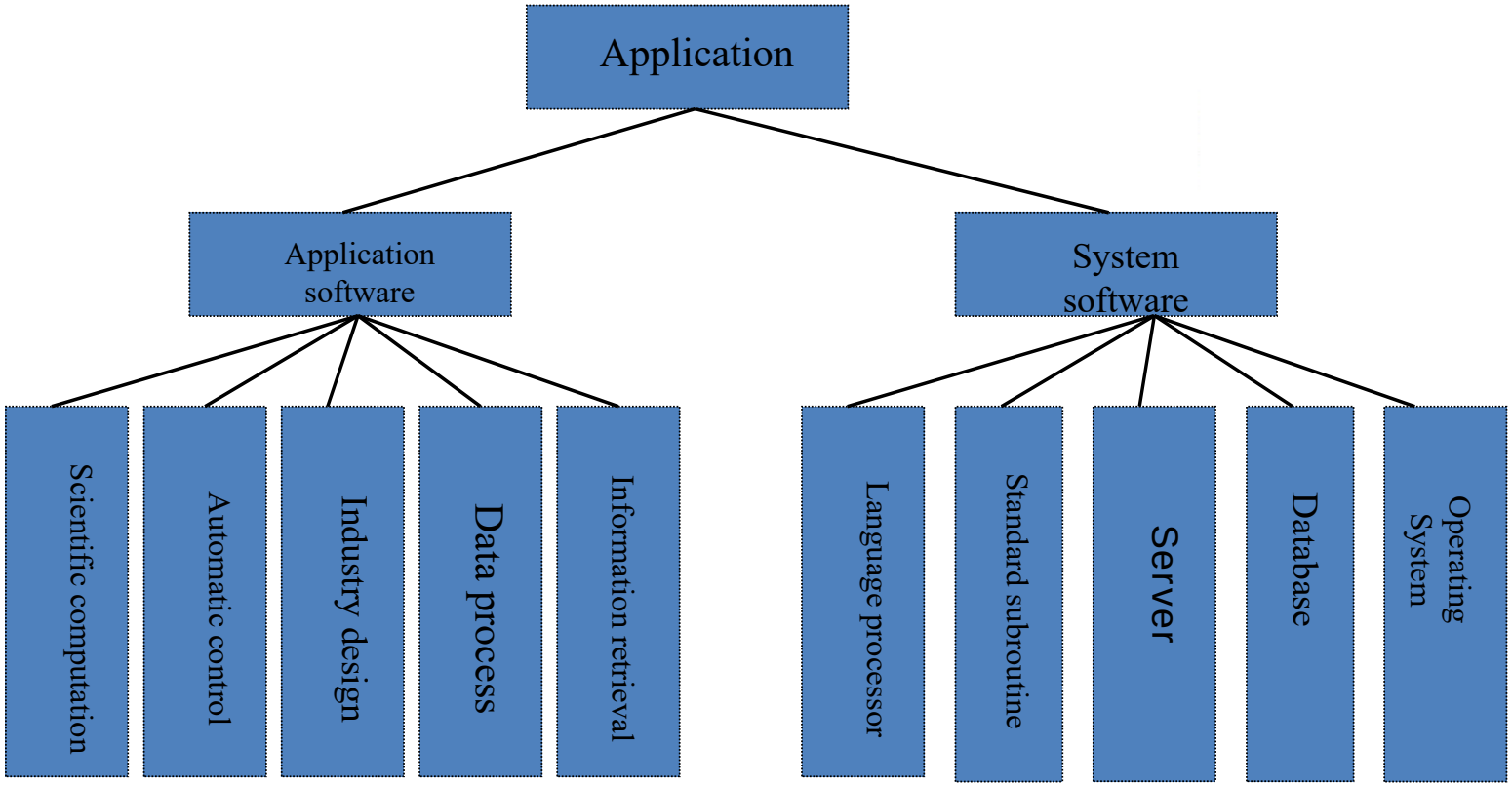
- A simplified view of Hardware and software as hierarchical layers



Hierarchical layers

- Application software
  - aimed at users
- System software
  - aimed at programmers
  - Includes:
    - Operation System
      - Virtual memory, File system, IO device drivers
    - Compiler
    - Assembler

The software classification



Computer Language

Machine language

- Computers only understands electrical signals
- Easiest signals: *on* and *off*
- Binary numbers express machine instructions ex. 1000110010100000 means to add two numbers
- Very tedious to write

Assembly language

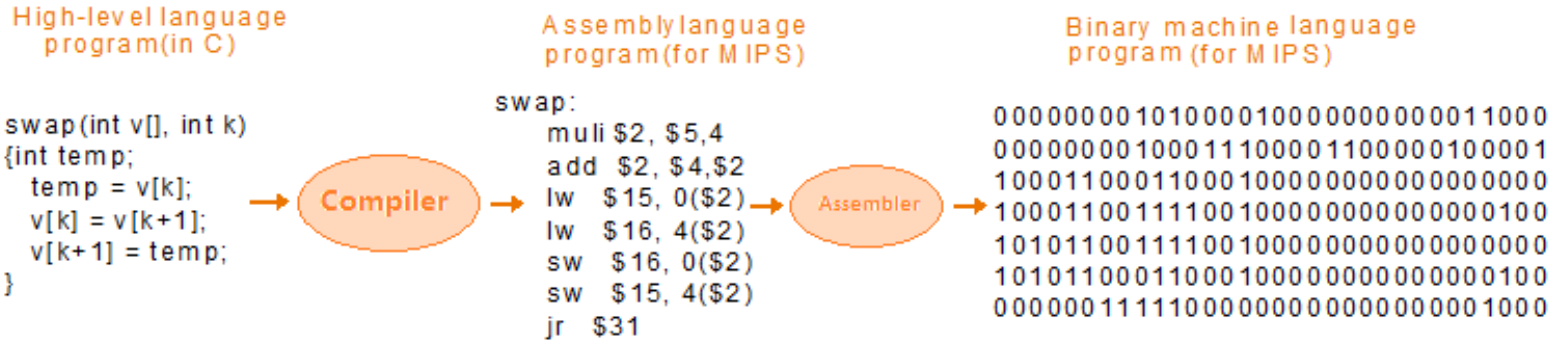
- Symbolic notations ex. add A, B
- The assembler translates them into machine instruction
- Programmers have to think like the machine

High-level programming language

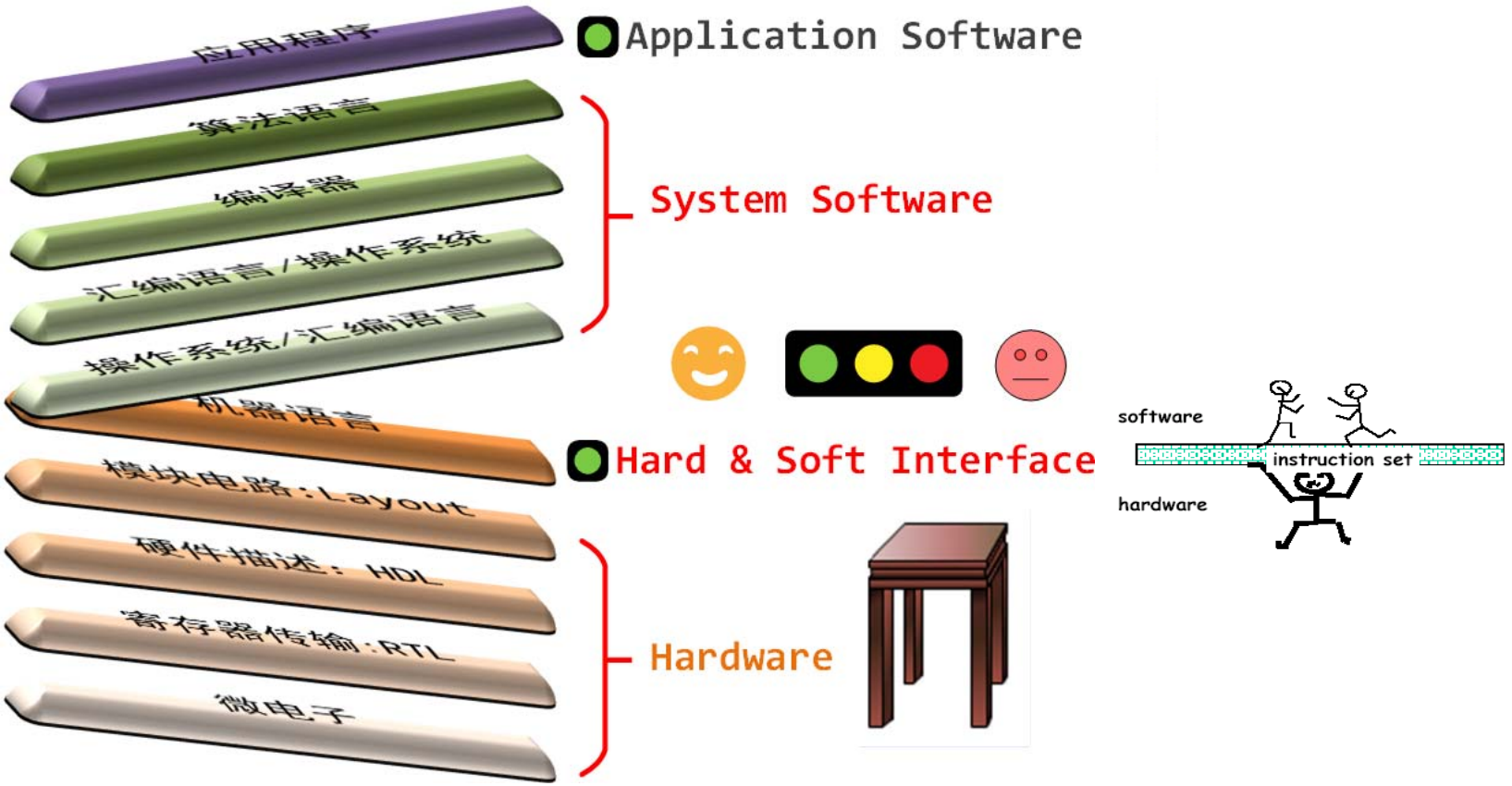
- Notations more closer to the natural language ex. A + B
- The compiler translates them into assembly language statements
- Subroutine library -- reusing programs
- Advantages over assembly language
  - Programmers can think in a more natural language
  - Improved programming productivity
  - Programs can be independent of hardware

From a High-Level Language to the Language of Hardware

- The process of compiling and assembling

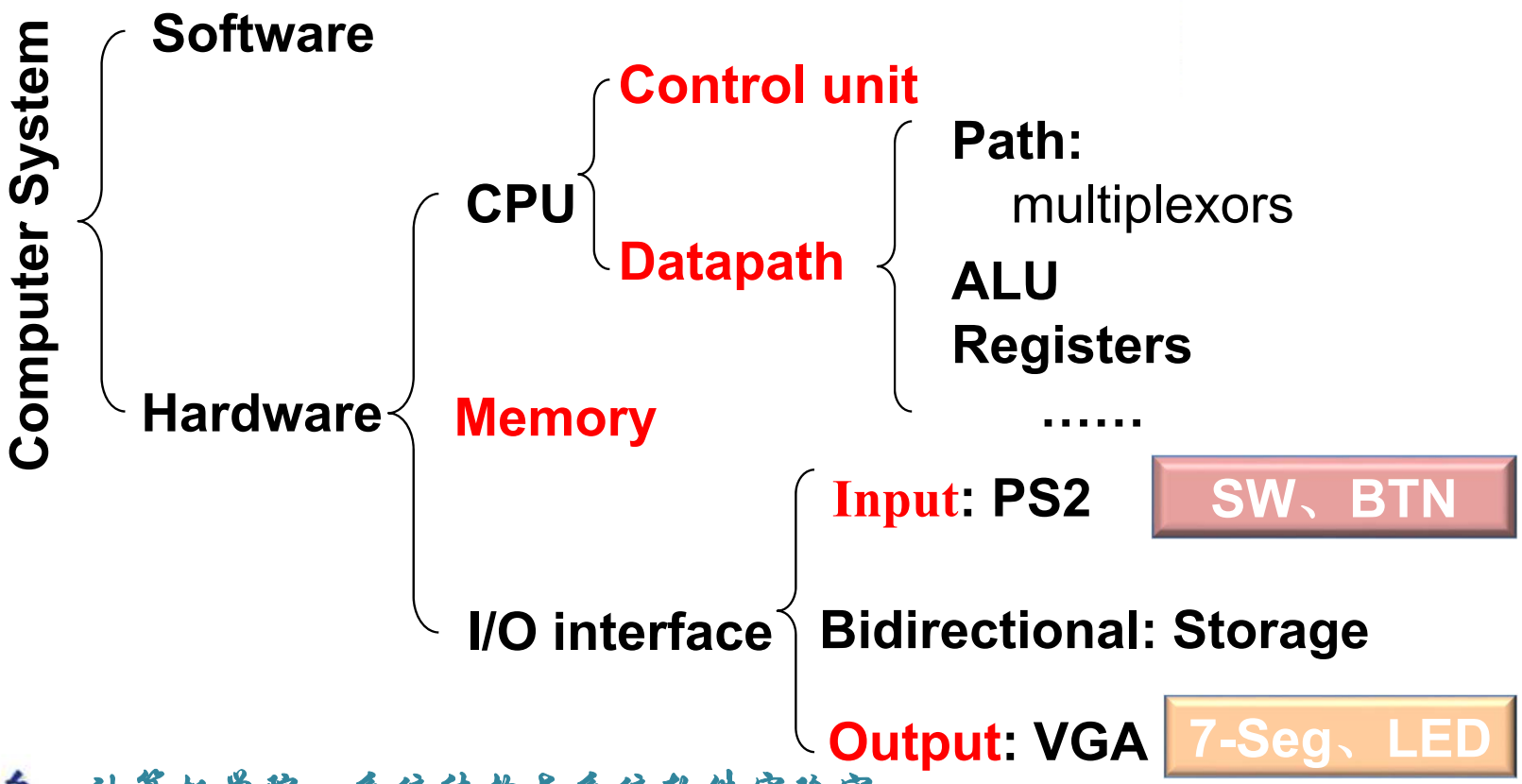


Computer System

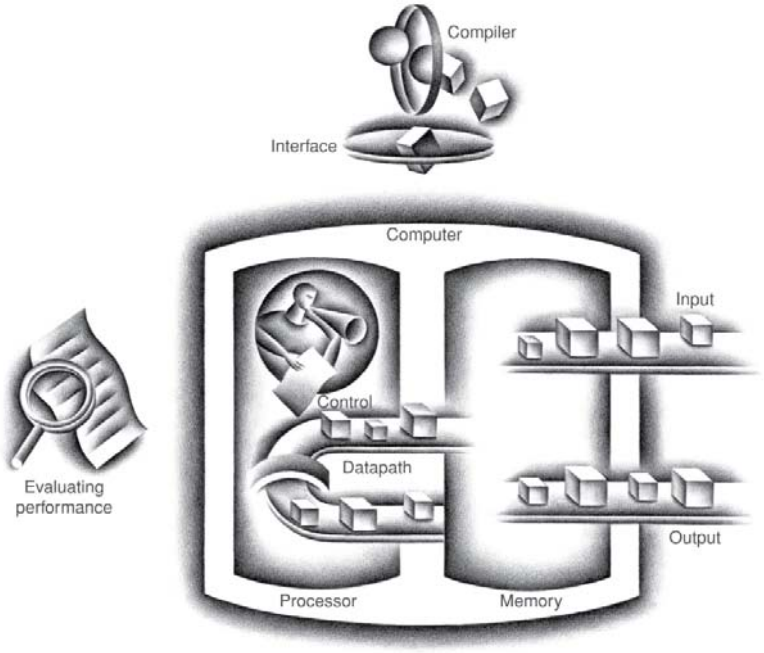


1.4 Computer Organization and Hardware System

Decomposability of computer systems



Components of a Computer



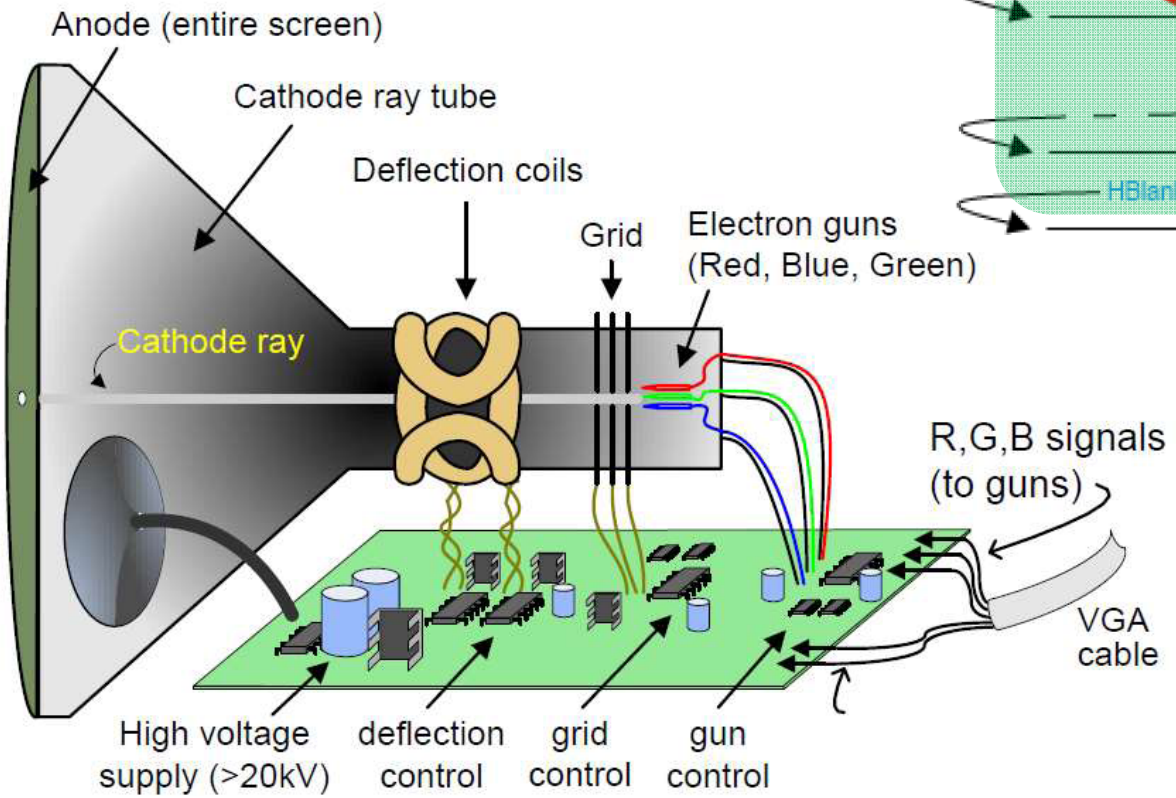
- Same components for all kinds of computer
  - Desktop, server, embedded
- Input/output includes
  - User-interface devices
    - Display, keyboard, mouse
  - Storage devices
    - Hard disk, CD/DVD, flash
  - Network adapters
    - For communicating with other computers

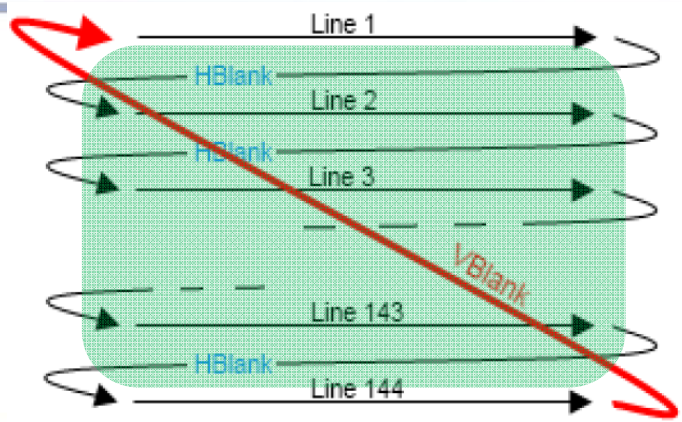
Touchscreen

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

Display

Cathode Ray Tube

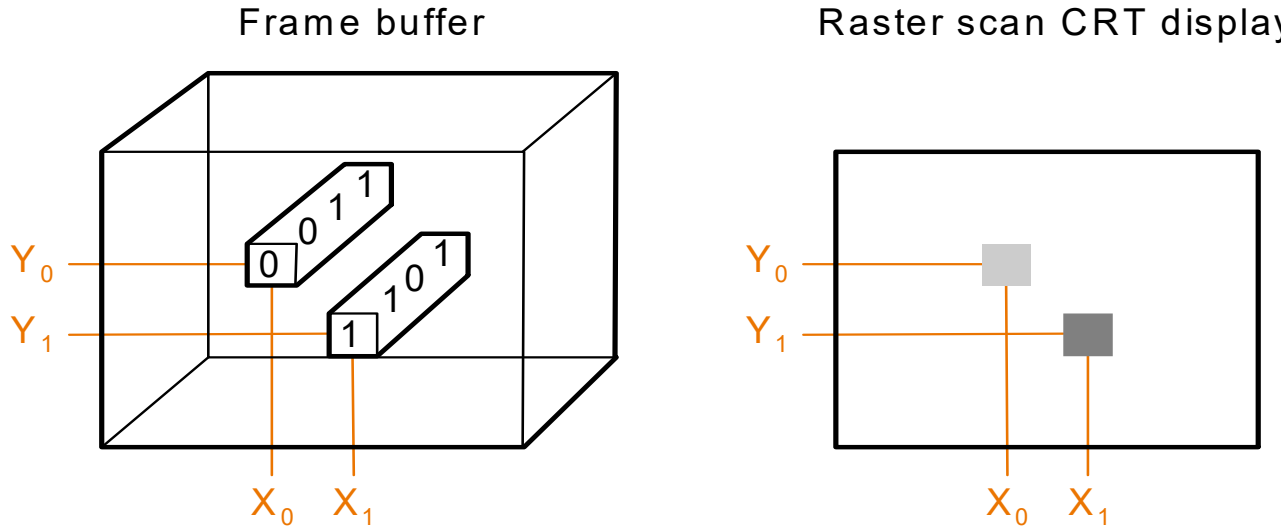




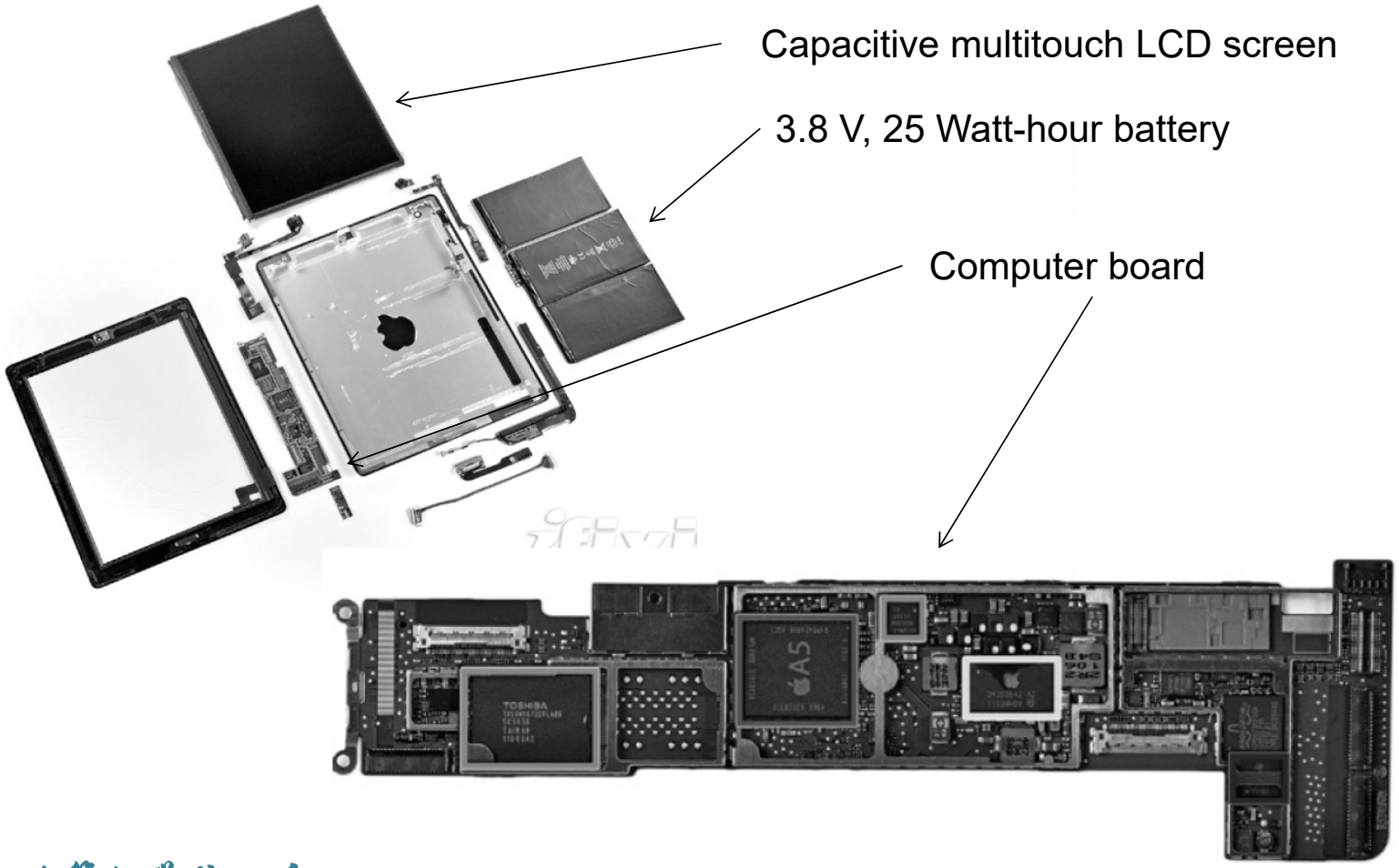
- Display
  - CRT (raster cathode ray tube) display
    - Scan an image one line at a time, 30 to 75 times / s
    - Pixels and the bit map, 512×340 to 1560×1280
    - The more bits per pixel, the more colors to be displayed
  - LCD (liquid crystal display)
    - Thin and low-power
    - The LCD pixel is not the source of light
    - Rod-shaped molecules in a liquid that form a twisting helix that bends light entering the display

The display principle

- Hardware support for graphics -- raster refresh buffer (**frame buffer**) to store bit map
- Goal of bit map -- to faithfully represent what is on the screen



Open the Box

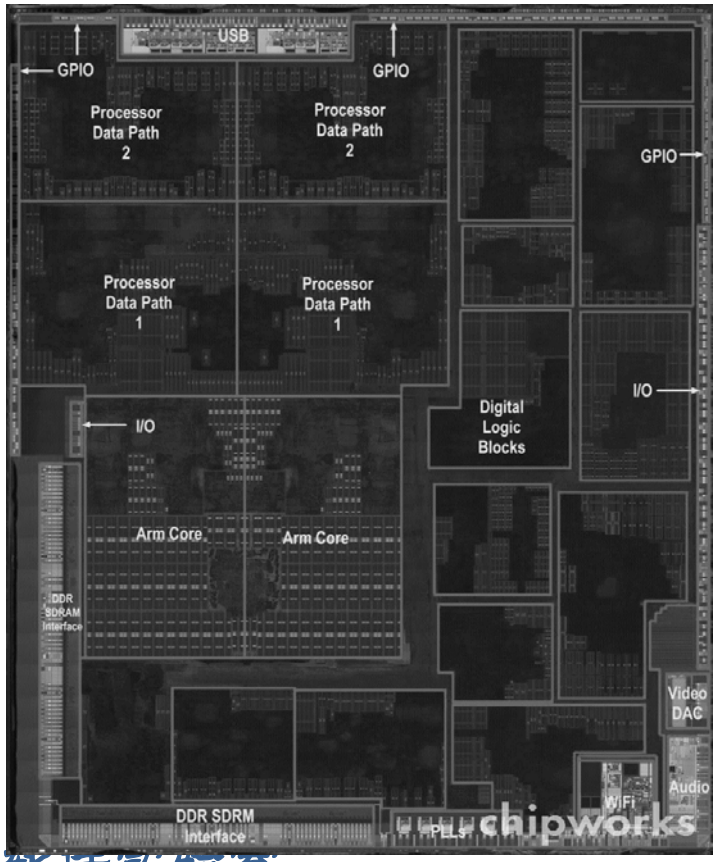

Capacitive multitouch LCD screen
3.8 V, 25 Watt-hour battery
Computer board

Motherboard

- **Motherboard and the hardware on it**
  - Motherboard
    - Thin, green, plastic, covered with dozens of small rectangles which contain **integrated circuits (chips)**
    - Three pieces: the piece connecting to the I/O devices, memory, and processor
  - Memory
    - Place to keep running programs and data needed
    - Each memory board contains some integrated circuits
    - DRAM and cache
  - Central Processor unit --CPU
    - Information processing center
    - Central processor unit

Inside the Processor

◎Apple A5



Inside the Processor (CPU)

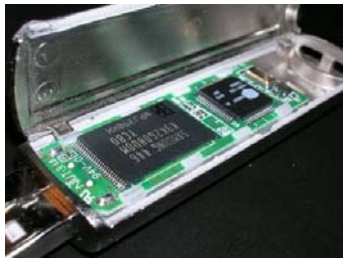
- Datapath: performs operations on data
- Control: sequences datapath, memory, ...
- Cache memory
  - Small fast SRAM memory for immediate access to data

Memory

- **A safe place for data -- secondary memory**
  - Main memory is volatile
  - Secondary memory is nonvolatile
    - Magnetic disk
      - Rotating platter coated with a magnetic material
      - Floppy disk
        - » Flexible mylar substance, 1.44~100MB, Removable
      - Hard disk
        - » Metal, Mostly not removable, Rotate on a spindle at 3600 to 7200 r.p.m., Read/write head and movable arm
        - » Slower than DRAM, but cheaper for a given storage unit
    - CD (optical compact disk)
    - Magnetic tape
    - Solid state memory

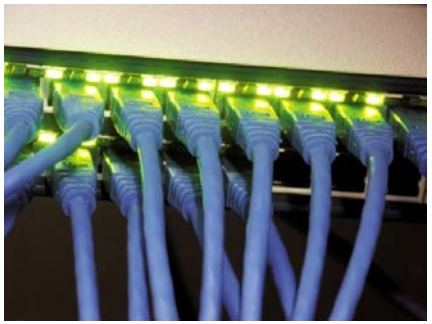
A Safe Place for Data

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



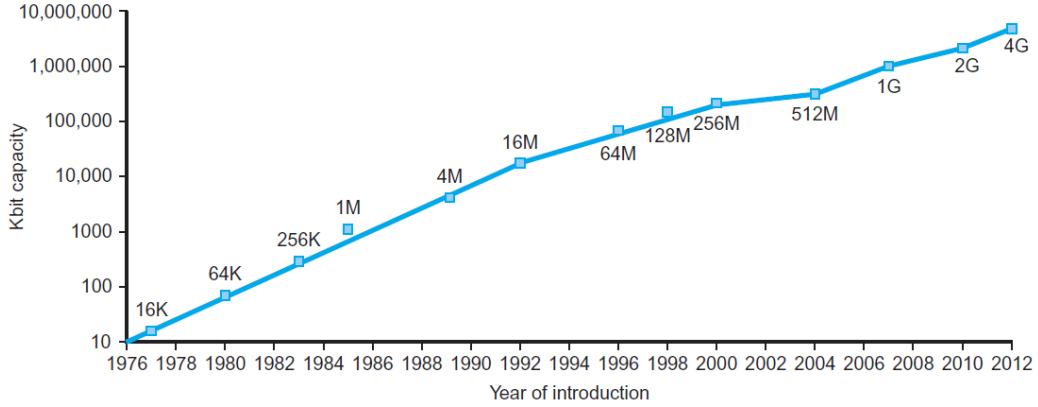
Networks

- Communication, resource sharing, nonlocal access
- Local area network (LAN): Ethernet
- Wide area network (WAN): the Internet
- Wireless network: WiFi, Bluetooth



1.5 Integrated Circuits

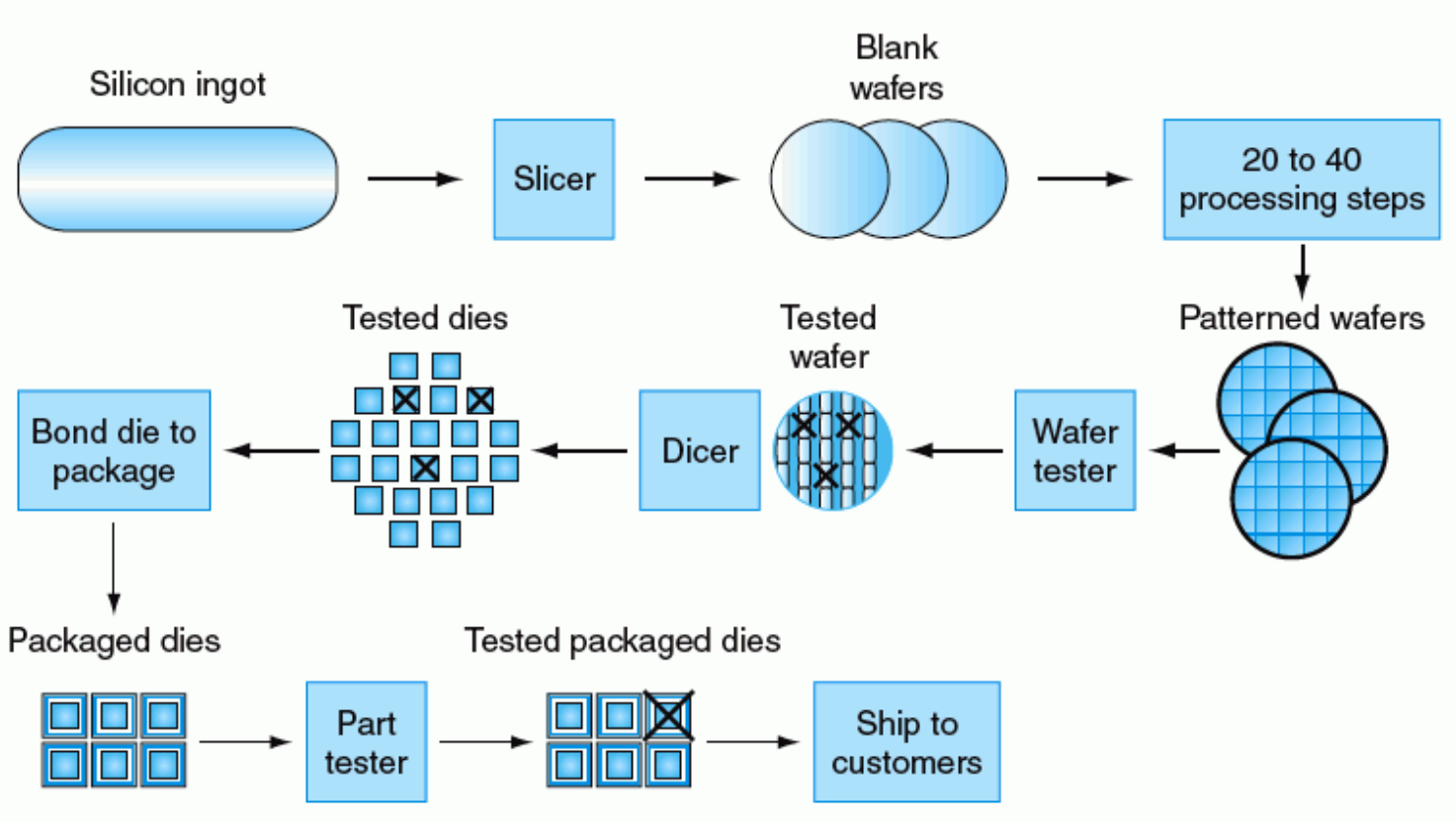
- **Electronics technology continues to evolve**
  - Increased capacity and performance
  - Reduced cost



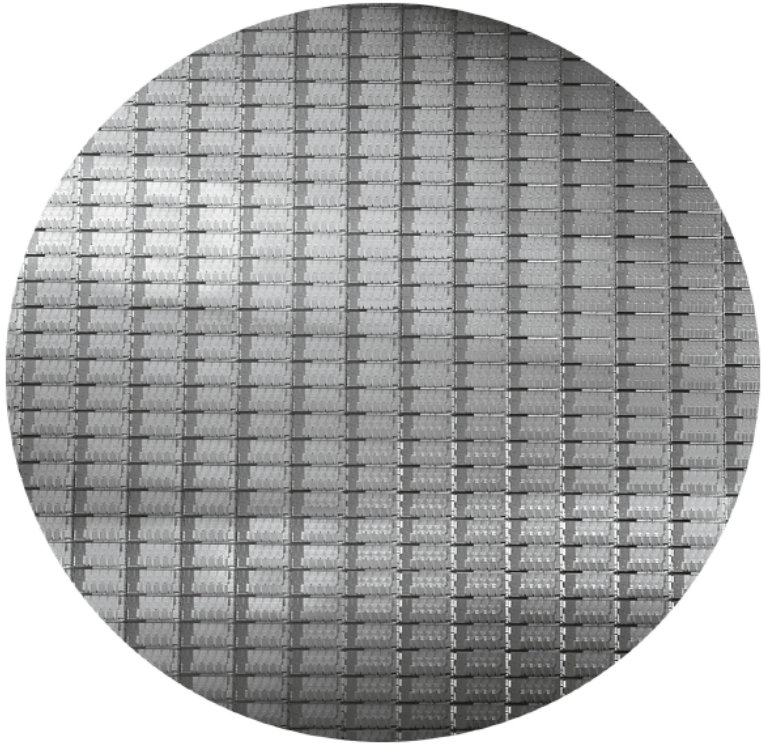
DRAM capacity

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

The semiconductor silicon and the chip manufacturing process



Intel Core i7 Wafer



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

Integrated Circuit Cost

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

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

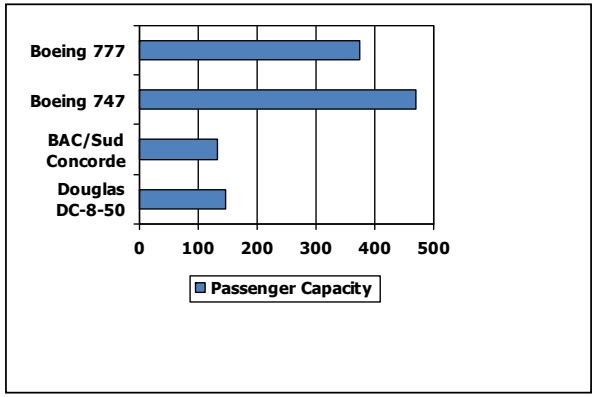
$$\text{Yield} = \frac{1}{(1 + (\text{Defects per area} \times \text{Die area}/2))^2}$$

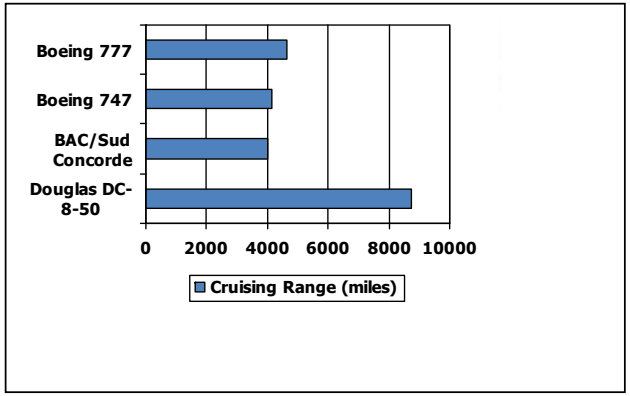
◎ Nonlinear relation to area and defect rate

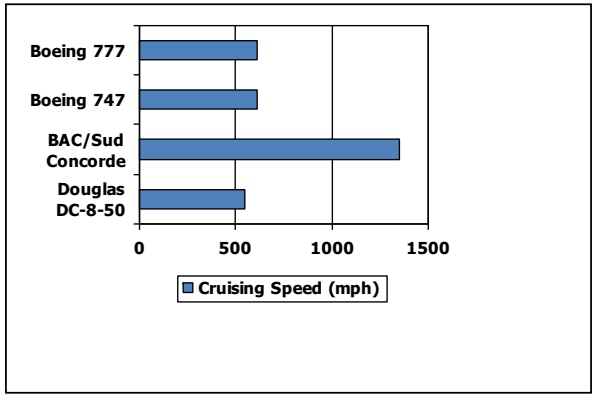
- Wafer cost and area are fixed
- Defect rate determined by manufacturing process
- Die area determined by architecture and circuit design

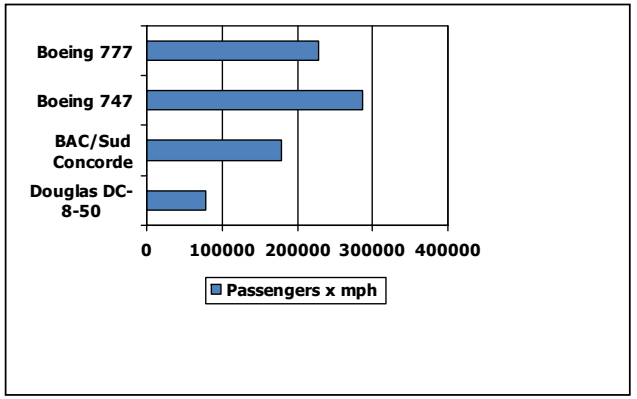
1.6 Performance

◎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
- 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/Execution Time

◎ "X is n time faster than Y"

$$\text{Performance}_X / \text{Performance}_Y = \text{Execution time}_Y / \text{Execution time}_X = n$$

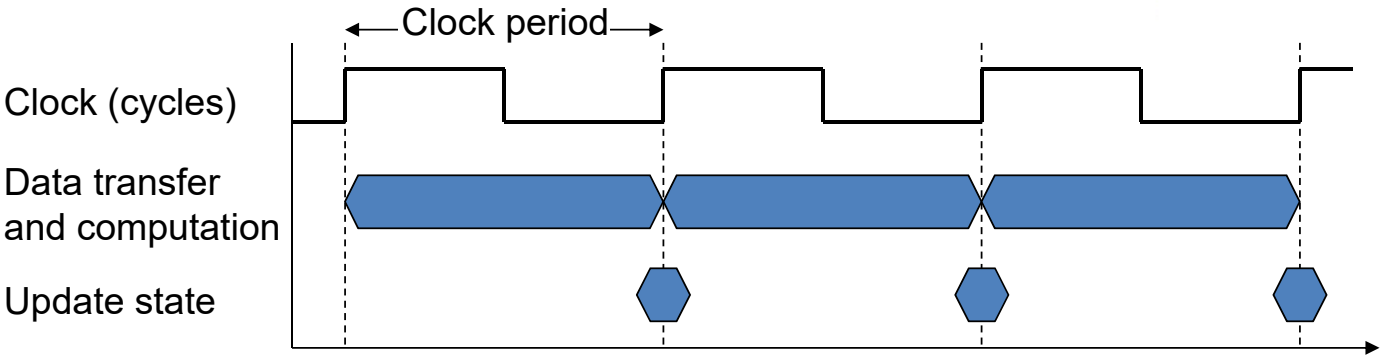
- Example: time taken to run a program
  - 10s on A, 15s on B
  - Execution Time_B / Execution Time_A = 15s / 10s = 1.5
  - So A is 1.5 times faster than B

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., 250ps = 0.25ns = 250×10^{-12}s
- Clock frequency (rate): cycles per second
  - e.g., 4.0GHz = 4000MHz = 4.0×10^{9}Hz

CPU Time

$$\text{CPU Time} = \text{CPU Clock Cycles} \times \text{Clock Cycle Time}$$

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

◎Performance improved by

- Reducing number of clock cycles
- Increasing clock rate
- Hardware designer must often trade off clock rate against cycle count

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 × clock cycles
- How fast must Computer B clock be?

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

$$\text{Clock Cycles}_A = \text{CPU Time}_A \times \text{Clock Rate}_A$$

$$= 10\text{s} \times 2\text{GHz} = 20 \times 10^9$$

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

Instruction Count and CPI

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

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

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

- ◎ 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?

$$\text{CPU Time}_A = \text{Instruction Count} \times \text{CPI}_A \times \text{Cycle Time}_A$$
$$= I \times 2.0 \times 250\text{ps} = I \times 500\text{ps}$$

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

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

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
    = 2×1 + 1×2 + 2×3
    = 10
  - Avg. CPI = 10/5 = 2.0
- Sequence 2: IC = 6
  - Clock Cycles
    = 4×1 + 1×2 + 1×3
    = 9
  - Avg. CPI = 9/6 = 1.5

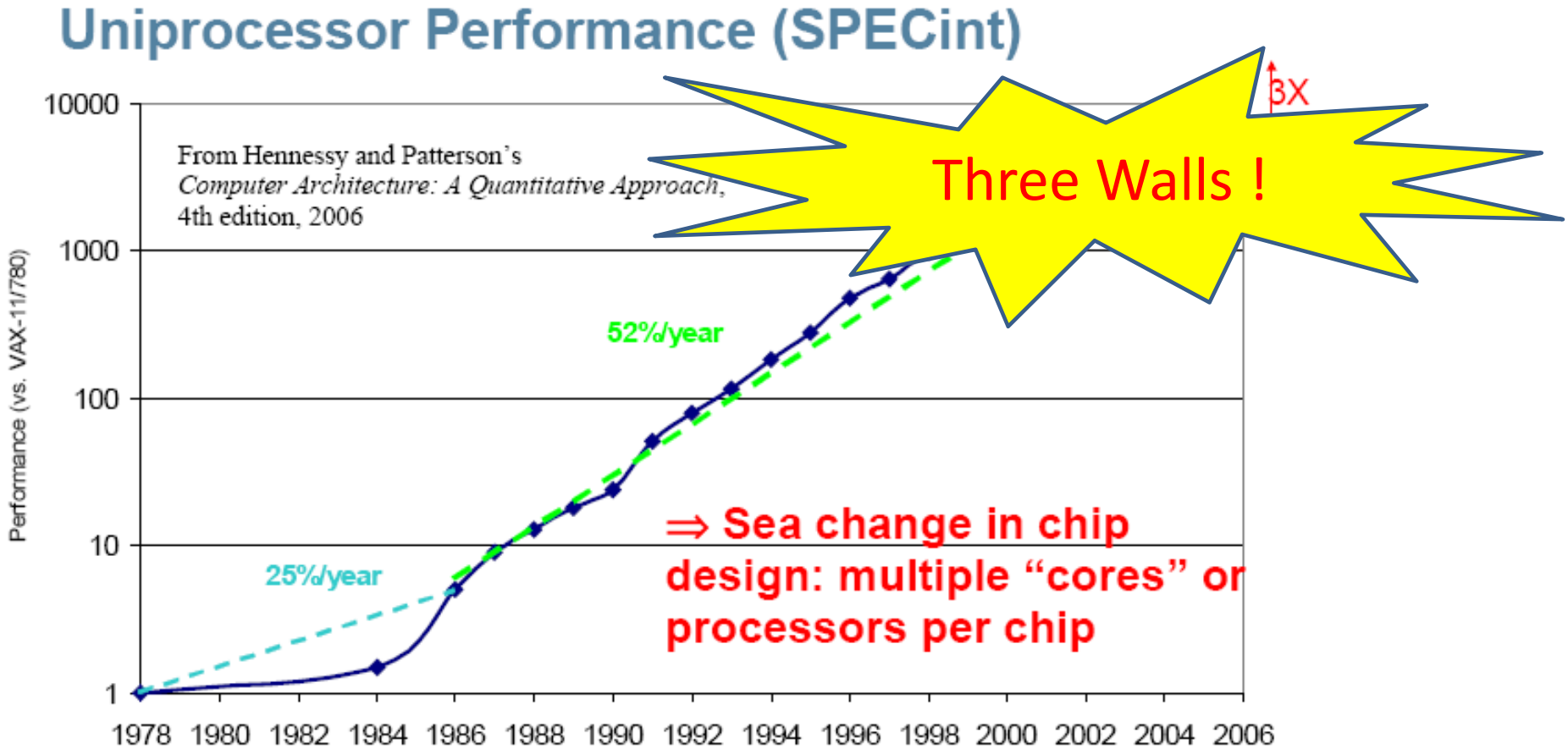
Performance Summary

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

◎Performance depends on

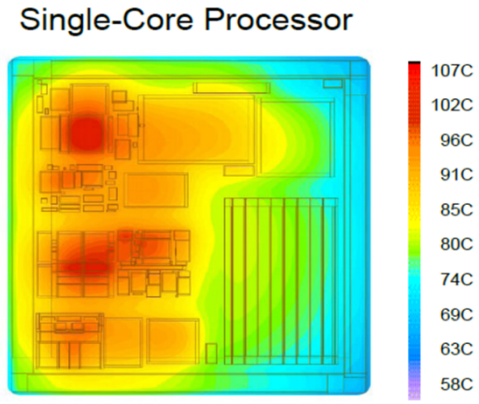
- Algorithm: affects IC, possibly CPI
- Programming language: affects IC, CPI
- Compiler: affects IC, CPI
- Instruction set architecture: affects IC, CPI, T_c

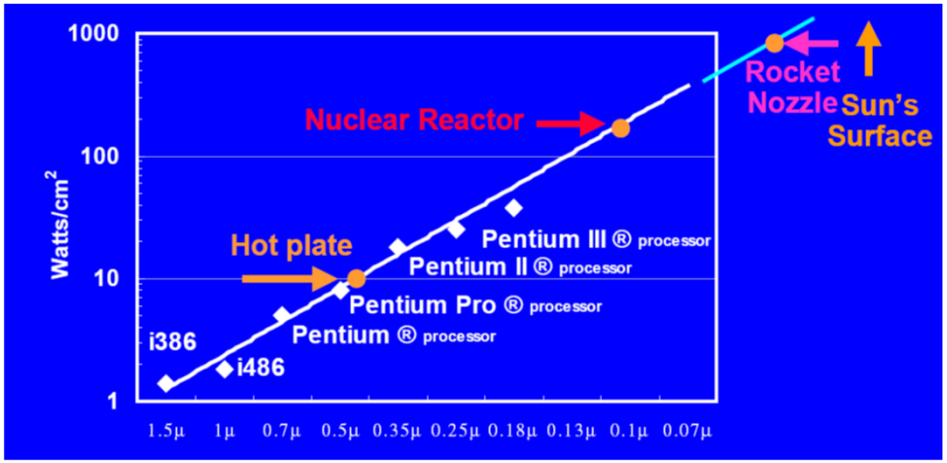
1.7 Incredible performance improvement



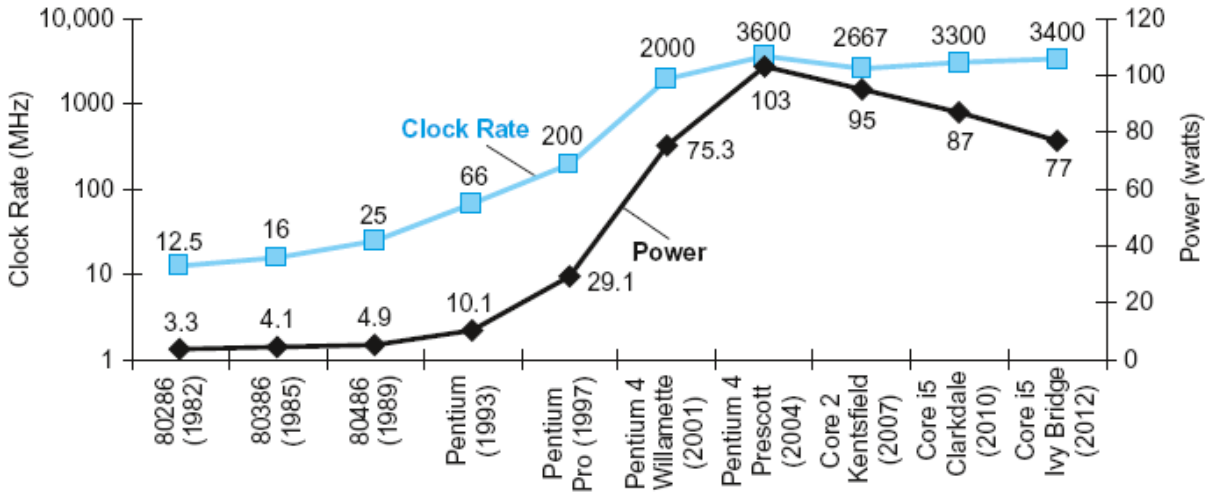
Power Wall

- Power efficiency decrease
  - the trend of consuming double the power with each doubling of operating frequency
- Hot-spot
- Power leakage





Power Trends



◎In CMOS IC technology

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

×30　　5V → 1V　　×1000

Reducing Power

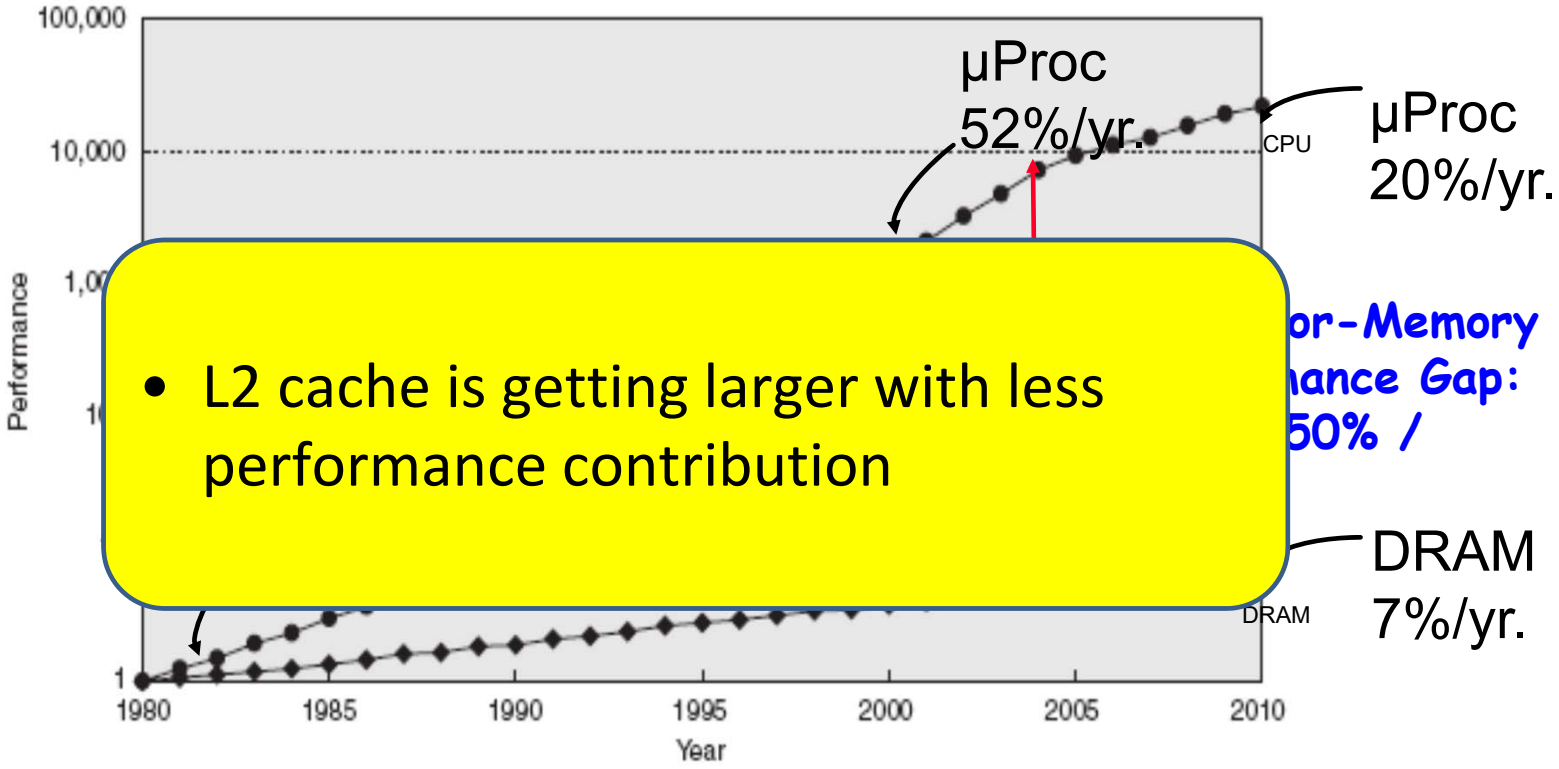
◎Suppose a new CPU has

- 85% of capacitive load of old CPU
- 15% voltage and 15% frequency reduction

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

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

Memory Wall



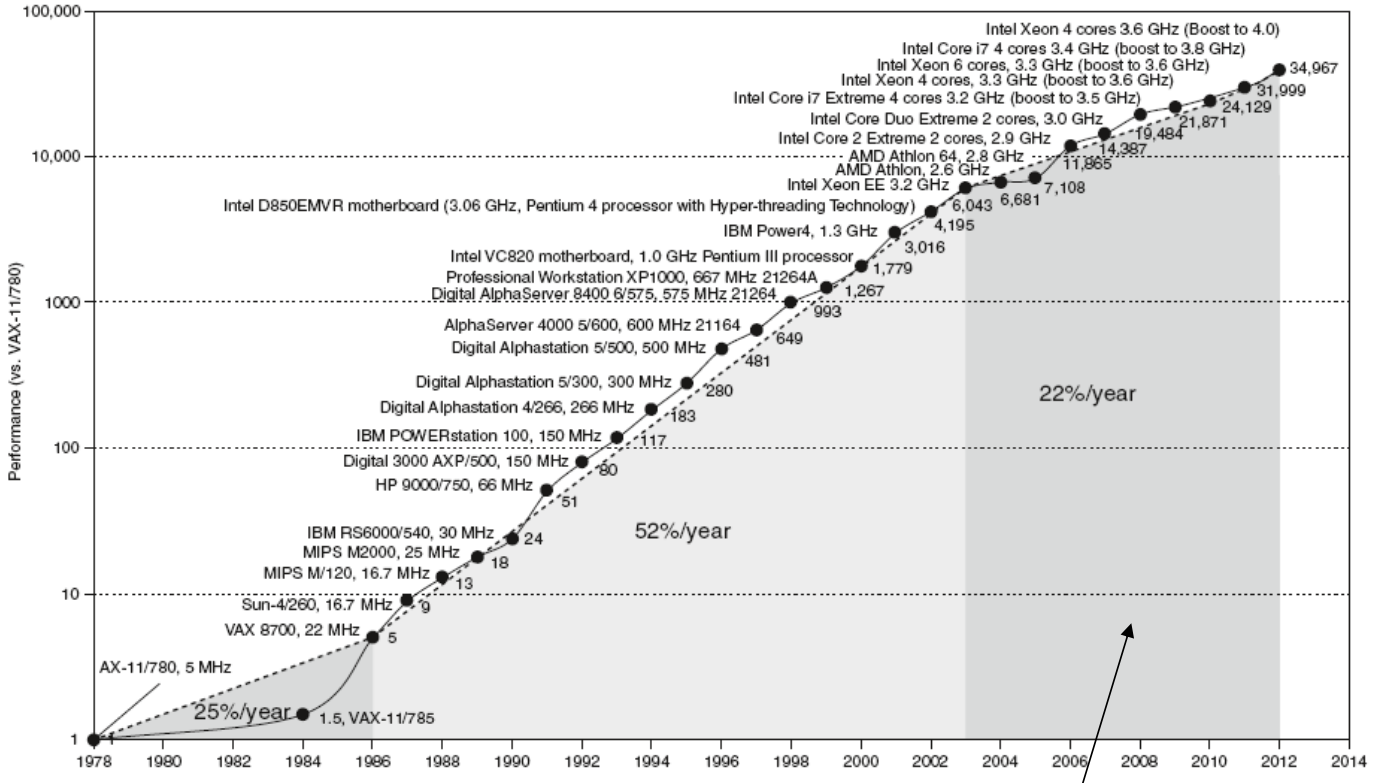
❑1980: no cache in µproc; 2001: 2-level cache on chip (1989 first Intel µproc with a cache on chip)

ILP Wall

- *"ILP wall"* refers to increasing difficulty to find enough parallelism in the instructions stream of a single process to keep higher performance processor cores busy.

- **20%: little ILP left to exploit due to power dissipation and memory gap**
  - ILP => TLP and DLP

Uniprocessor Performance



Constrained by power, instruction-level parallelism, memory latency

§ 1.8 The Sea Change: The Switch to Multiprocessors

1.8 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

SPEC Power Benchmark

◎Power consumption of server at different workload levels

- Performance: ssj_ops/sec
- Power: Watts (Joules/sec)

$$\text{Overall ssj\_ops per Watt} = \left(\sum_{i=0}^{10} \text{ssj\_ops}_i\right) \Big/ \left(\sum_{i=0}^{10} \text{power}_i\right)$$

SPECpower_ssj2008 for Xeon X5650

Target Load %	Performance (ssj_ops)	Average Power (Watts)
100%	865,618	258
90%	786,688	242
80%	698,051	224
70%	607,826	204
60%	521,391	185
50%	436,757	170
40%	345,919	157
30%	262,071	146
20%	176,061	135
10%	86,784	121
0%	0	80
Overall Sum	4,787,166	1,922
Σssj_ops/Σpower =		2,490

Pitfall: Amdahl's Law

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

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

- 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

$$\text{MIPS} = \frac{\text{Instruction count}}{\text{Execution time} \times 10^6}$$

$$= \frac{\text{Instruction count}}{\frac{\text{Instruction count} \times \text{CPI}}{\text{Clock rate}} \times 10^6} = \frac{\text{Clock rate}}{\text{CPI} \times 10^6}$$

- CPI, MIPS varies between programs on a given CPU

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

Assignment

Reading: chapter 1

Problems:

1-1, 1-2, 1-4, 1-6, 1-7, 1-14


END