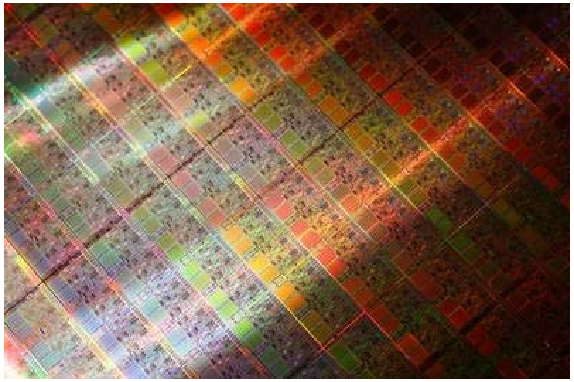
Computer Organization and Design
No.18001140 (Fall 2014)

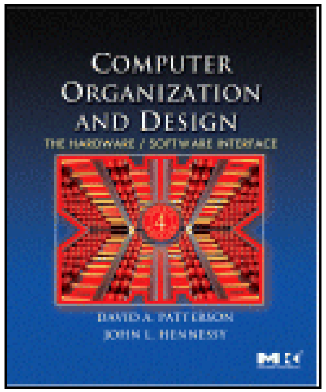
Introduction




Prof. Jiang Zhong

Course Logistics

- Instructor
  - Jiang Zhong (zhongjiangjx@163.com, zhongjiang@cqu.edu.cn, 13983650069)
  - Website: http://www.cs.cqu.edu.cn/public/tindex/20196
  - Office Hours: Wed.: 1:30-2:30 pm (Main Building 1709)
    - or by appointment (send email)
- Class Meets
  - Tuesday 10:00-11:55 pm in Room A.5207
  - Thursday 10:00-11:55 pm in Room A5207
- Textbook (Required)
  - David A. Patterson and John L. Hennessy, *Computer Organization and Design: The Hardware/Software Interface*, **4th Edition**, Morgan Kaufmann, October 2008



What You Should Know

- Prerequisite
  - Intro to Microprocessors
- Basic digital logic design
  - FSM, synchronous design
- Basic structure of a microprocessor
  - including memory subsystem, I/O
- Addressing modes
  - for operands in instructions
- Some experience with assembly language programming, debugging

课程定位

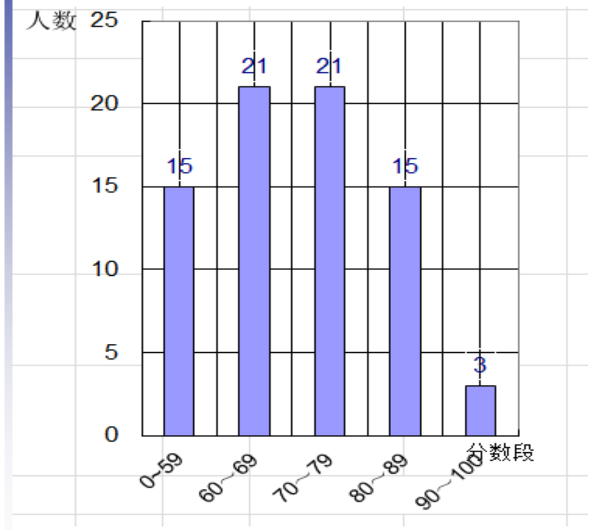

专业方向课程
可视计算方向
计算机图形学研究进展
计算机视觉
信息可视化
图象信息处理
计算机图形学
计算机理论方向
人工智能
数据挖掘导论
应用运筹学基础
数值分析
算法设计与分析
计算机软件方向
软件工程实践
多媒体技术
智能终端软件开发
Java应用技术
软件体系结构
计算机系统方向
计算机安全
流计算与GPGPU软件开发
片上系统设计与实现
并行计算与多核编程
现代存储技术基础
专业核心课程
计算机软件设计课程群
数据库系统b
计算机网络b
软件工程
面向对象程序设计
高级数据结构与算法分析
编译系统a
操作系统a
计算机网络a
数据库系统a
计算机软件设计探索
计算机系统课程群
计算机软硬件系统综合
操作系统b
编译系统b
汇编与接口
计算机系统硬件综合
计算机体系结构
计算机组成
逻辑与计算机设计基础
计算机系统探索
专业基础课程
数据结构基础
程序设计基础
计算理论
离散数学及应用
计算机基础和理论课程群

Evaluation and Grading

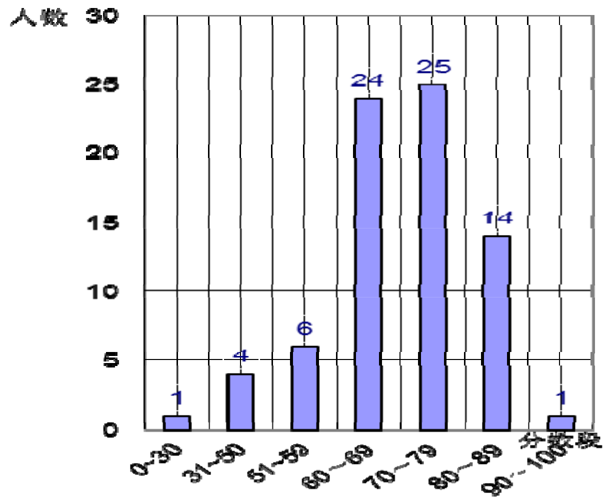
- Homework Assignments (5): 10%
- Class Participation: 10%
  - Quizzes (4-5): 10%
- Examinations (closed book/notes): 40%
  - Comprehensive Final: 40%
  - Experiments: 20%
  - Project: 20%
- Grading scale

>95% A+	*80-84% B+*	*65-69% C+*	*<40% F*
90-94% A	*75-79% B*	*55-64% C*	
85-89% A-	*70-74% B-*	*40-55% D*	

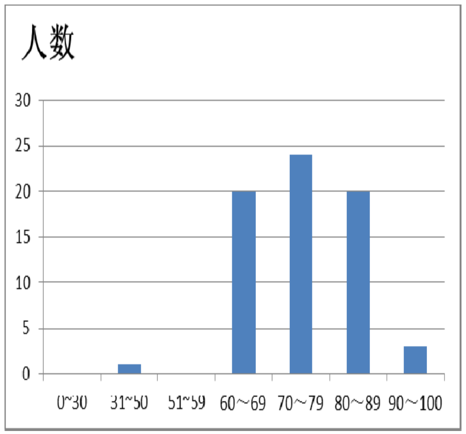
Grade Distributions in Recent Years



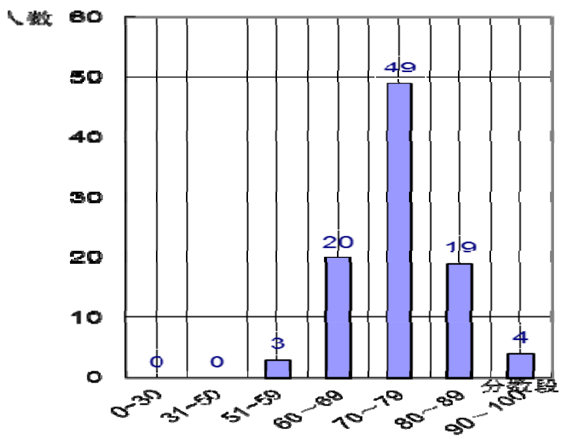
2007



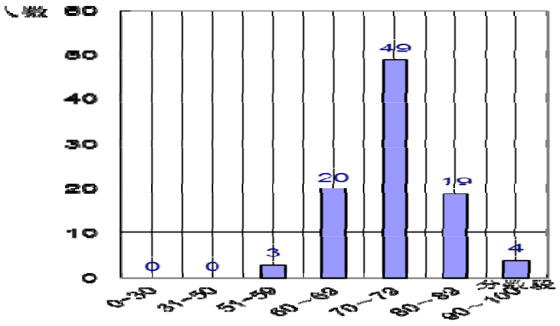
2009



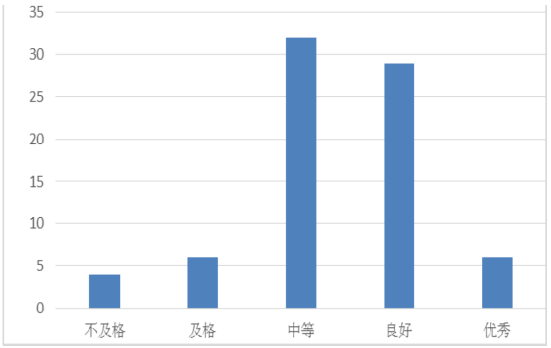
2010



2011



2012



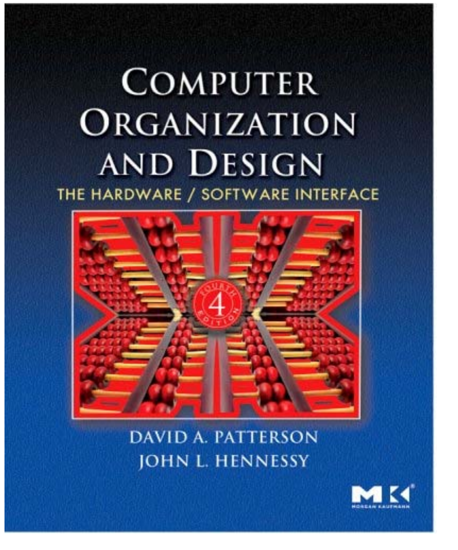
2013

Evaluation and Grading

- **Homework Policy**
  - No late submissions accepted unless you have a valid excuse
- **Attendance**
  - Your responsibility to keep track of what you missed if absent
  - Keep track of assignments and due dates
- **Academic Honesty**
  - All submitted work should be your own. Plagiarism/cheating will result in all students involved getting a zero on the assignment/exam and potentially a failing grade. Refer to the *CQU Academic Integrity Guidelines* for more information.
- **Appointment**
  - I encourage you to make at least one appointment with me during the semester for advice or to discuss research opportunities, research ideas, course suggestions, concerns etc.

Scope of Course

- Lecture Topics:
  - Instruction Set Architectures (MIPS/ARM)
  - Computer Arithmetic
  - VLIW and Superscalar Processor Design
  - Memory Hierarchy, Storage and I/O
  - Multicores and multiprocessors
  - Interconnection Networks



What you will Learn

- How are programs written in high level languages (C or Java) translated into the language of the hardware, and how does hardware execute the resulting program?
- What is the interface between software and hardware, and how does software instruct hardware to perform needed functions? ISA (instruction set architecture)
- What determines the performance of a program, and how can software programmers and hardware designers improve performance?
- What are the reasons for and consequences of the recent switch from sequential to parallel processing?

What you will Learn

- The implementation of a machine has two components: organization and hardware. The term organization includes the high-level aspects of a computer's design, such as the memory system, the bus structure, and the internal CPU (central processing unit—where arithmetic, logic, branching, and data transfer are implemented) design

Don’t Forget …

Ask Questions in class!

Chapter 1

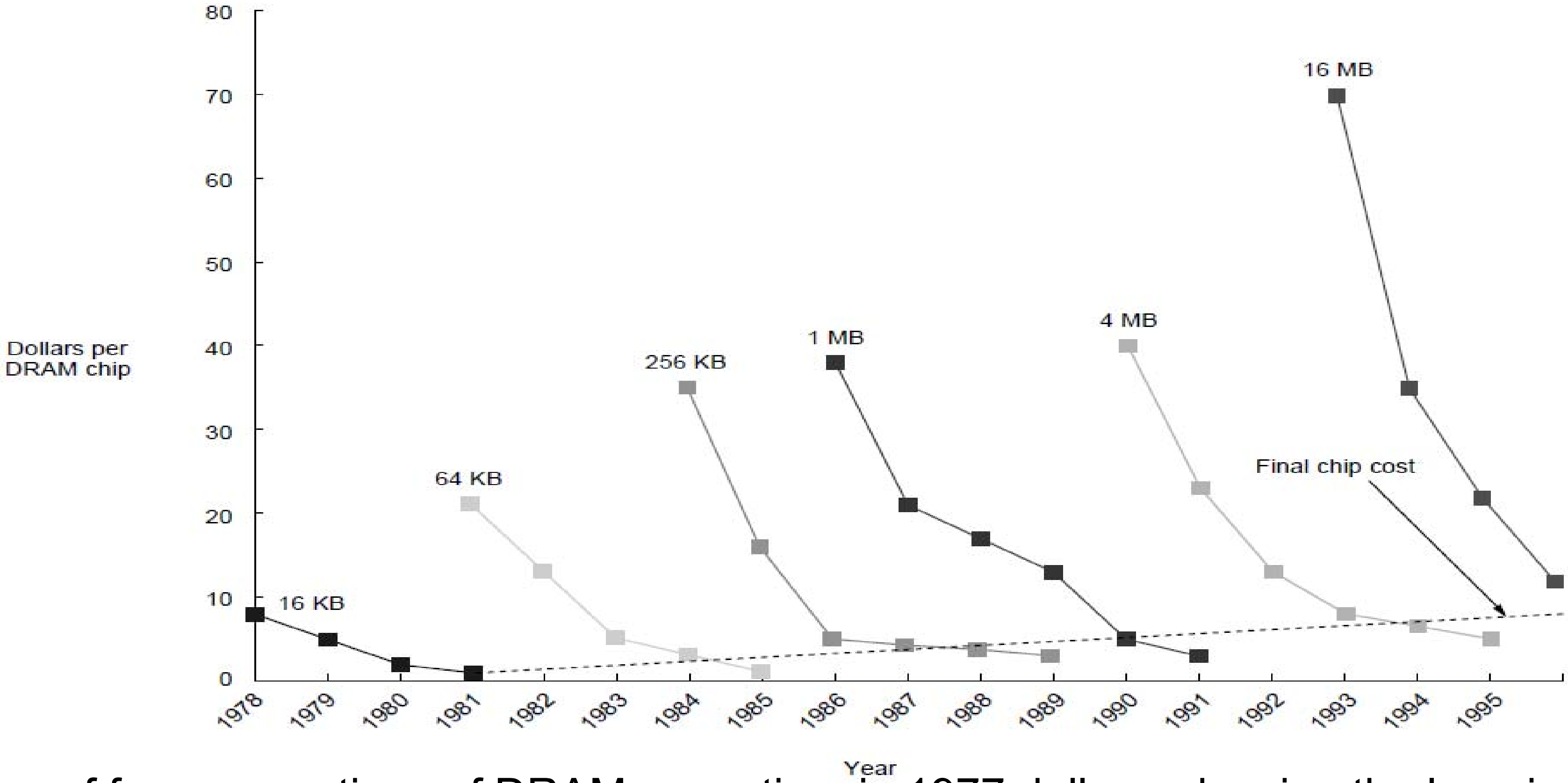
Computer Abstractions and Technology

The Computer Revolution

- Progress in computer technology
  - Underpinned by Moore's Law
- Makes novel applications feasible
  - Computers in automobiles 嵌入式计算机
  - Cell phones 智能手机
  - Human genome project 基因工程
  - World Wide Web
  - Search Engines 搜索引擎
- Computers are pervasive

IOT 物联网

Prices of DRAMs over time in 1977 dollars



Prices of four generations of DRAMs over time in 1977 dollars, showing the learning curve at work. A1977 dollar is worth about $2.44 in 1995; most of this inflation occurred in the period of 1977–82, during which the value changed to $1.61. The cost of a **megabyte** of memory has dropped incredibly during this period, from over **$5000** in 1977 to just over **$6** in 1995 (in 1977 dollars)

Classes of Computers

❑ Desktop and laptop computers

Designed to deliver good performance to a single user at low cost usually executing 3rd party software, usually incorporating a graphics display, a keyboard, and a mouse

❑ Servers

Used to run larger programs for multiple, simultaneous users typically accessed only via a network and that places a greater emphasis on dependability and (often) security

❑ Supercomputers

A high performance, high cost class of servers with hundreds to thousands of processors, terabytes of memory and petabytes of storage that are used for high-end scientific and engineering applications

❑ Embedded computers (processors)

A computer inside another device used for running one predetermined application

Review: Some Basic Definitions

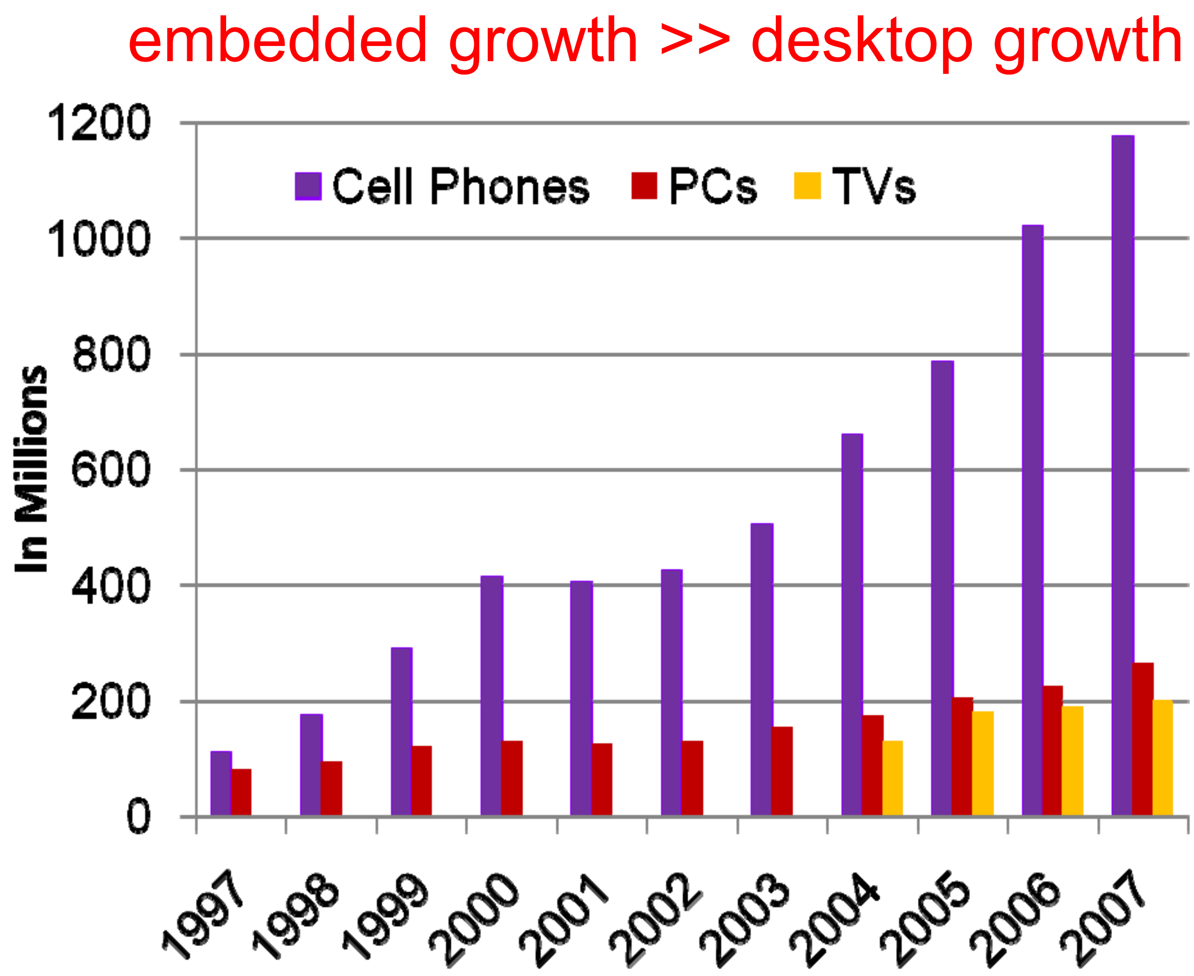
- Kilobyte – 2^{10} or 1,024 bytes

KB MB GB TB PB EB

- Megabyte– 2^{20} or 1,048,576 bytes
  - sometimes “rounded” to 10^6 or 1,000,000 bytes
- Gigabyte – 2^{30} or 1,073,741,824 bytes 海量数据
  - sometimes rounded to 10^9 or 1,000,000,000 bytes
- Terabyte – 2^{40} or 1,099,511,627,776 bytes
  - sometimes rounded to 10^{12} or 1,000,000,000,000 bytes
- Petabyte – 2^{50} or 1024 terabytes 大数据
  - sometimes rounded to 10^{15} or 1,000,000,000,000,000 bytes
- Exabyte – 2^{60} or 1024 petabytes 超算 super computer
  - Sometimes rounded to 10^{18} or 1,000,000,000,000,000,000 bytes

ZB YB

The Processor Market



- Where else are embedded processors found?

Embedded Processor Characteristics

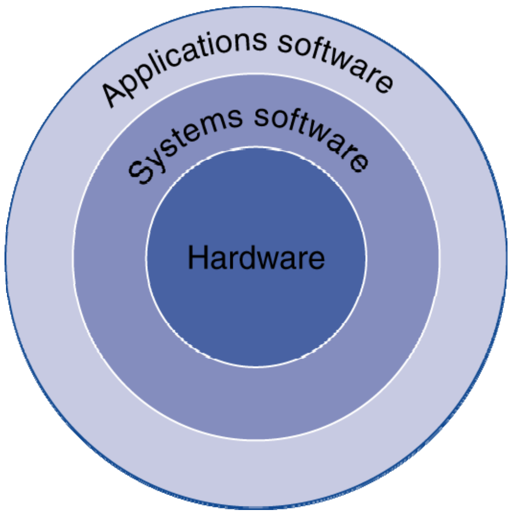
The largest class of computers spanning the widest range of applications and performance

- Often have minimum performance requirements. Example?
- Often have stringent limitations on cost. Example?
- Often have stringent limitations on power consumption. Example?
- Often have low tolerance for failure. Example?

Understanding Performance

- Algorithm
  - Determines number of operations executed
- Programming language, compiler, ISA
  - 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 program (in C)

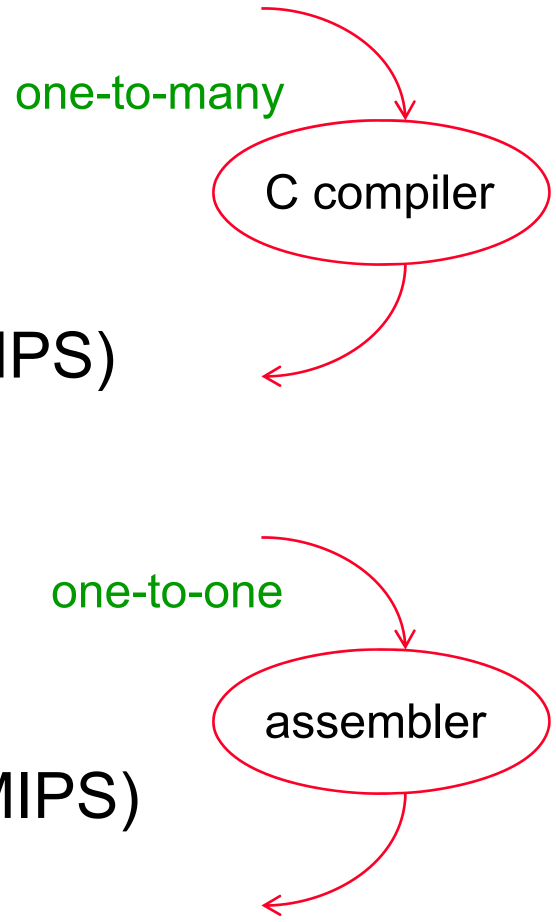
```
void swap (int v[], int k)
{      int temp;
       temp = v[k];
       v[k] = v[k+1];
       v[k+1] = temp;
}
```

- Assembly language program (for MIPS)

```
swap:  sll    $2, $5, 2
       add    $2, $4, $2
       lw     $15, 0($2)
       lw     $16, 4($2)
       sw     $16, 0($2)
       sw     $15, 4($2)
       jr     $31
```

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

```
000000 00000 00101 0001000010000000
000000 00100 00010 0001000000100000
 . . .
```



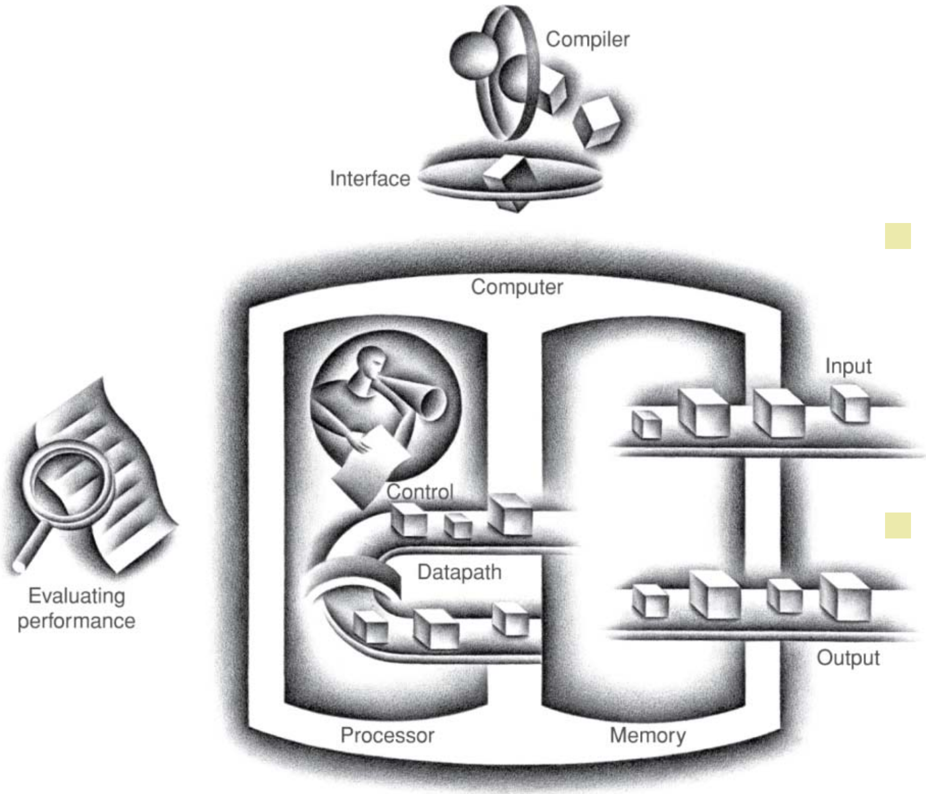
Advantages of Higher-Level Languages ?

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

- As a result, very little programming is done today at the assembler level

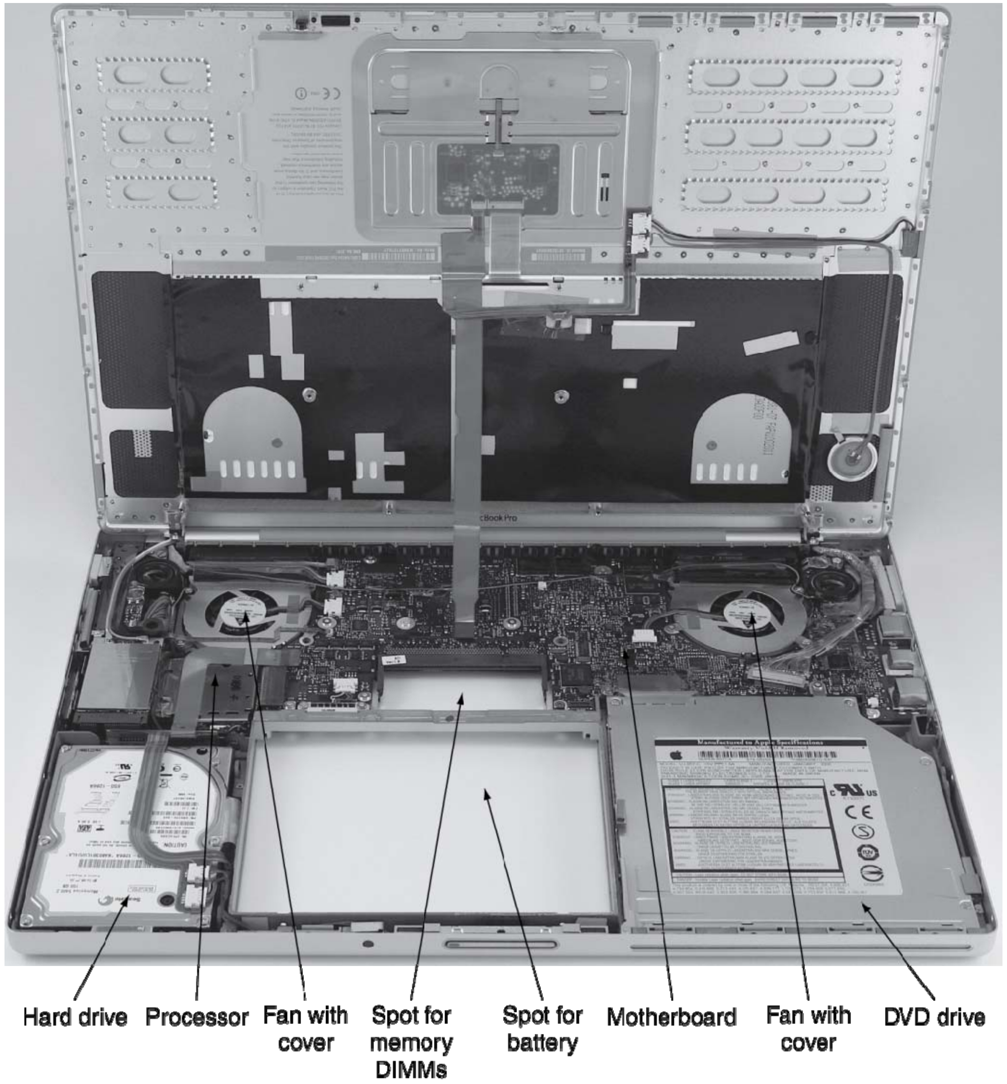
Components of a Computer

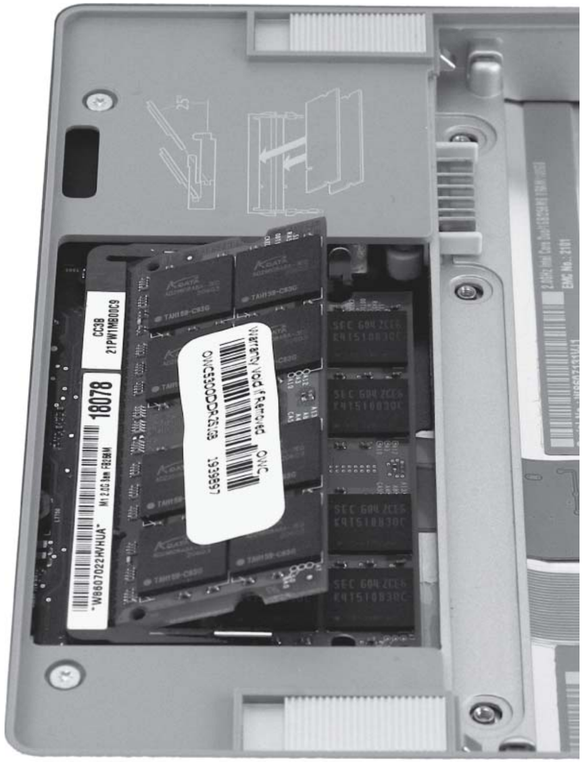
The BIG Picture



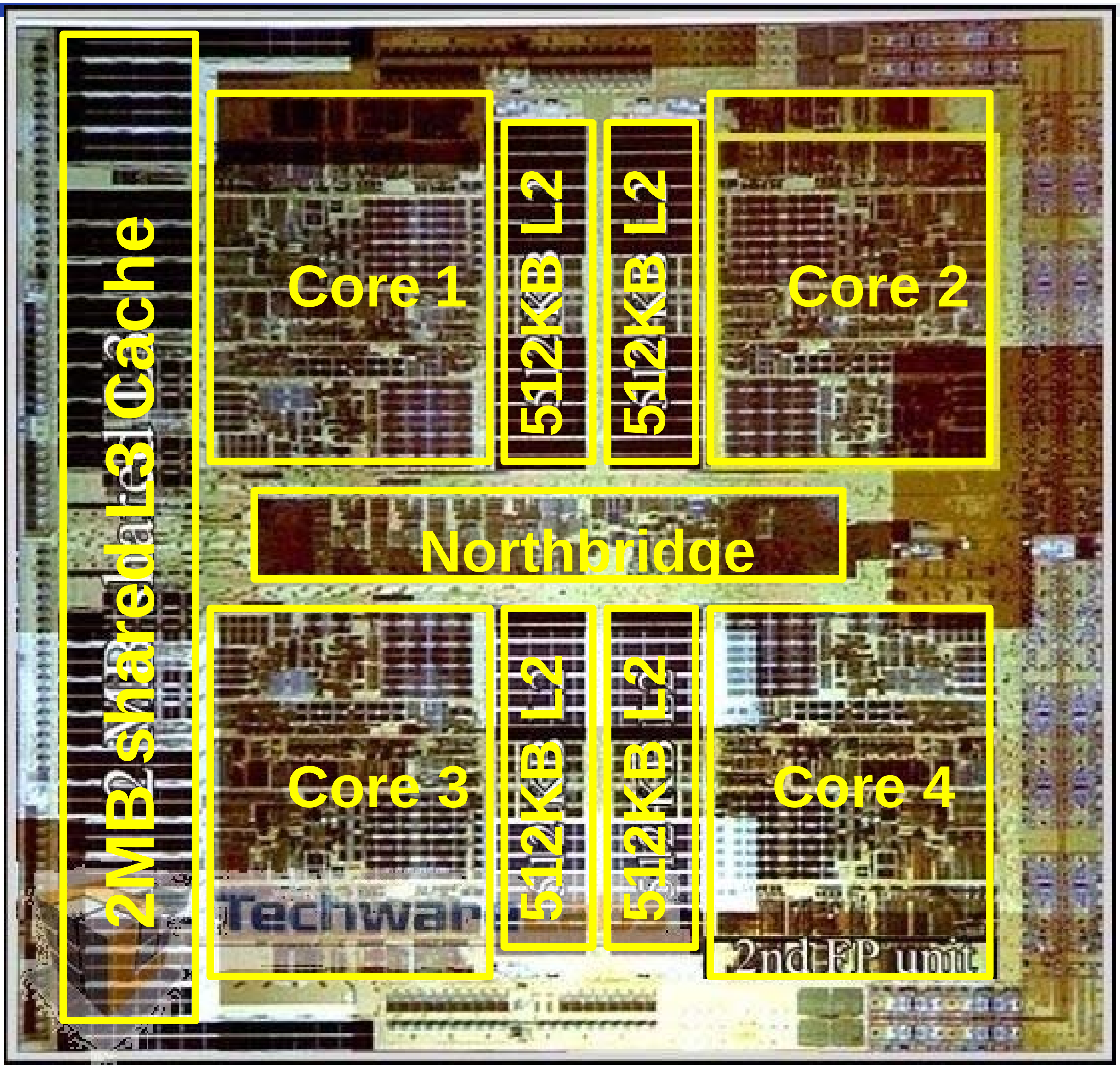
- Five main components
  - Input, output, memory, 数据通路（处理器、运算器）datapath, control
  - Datapath + Control = Processor
- Same components for all kinds of computers
  - Desktop, server, embedded
- Input/output includes
  - User-interface devices, storage devices, network adapters

Opening the Box





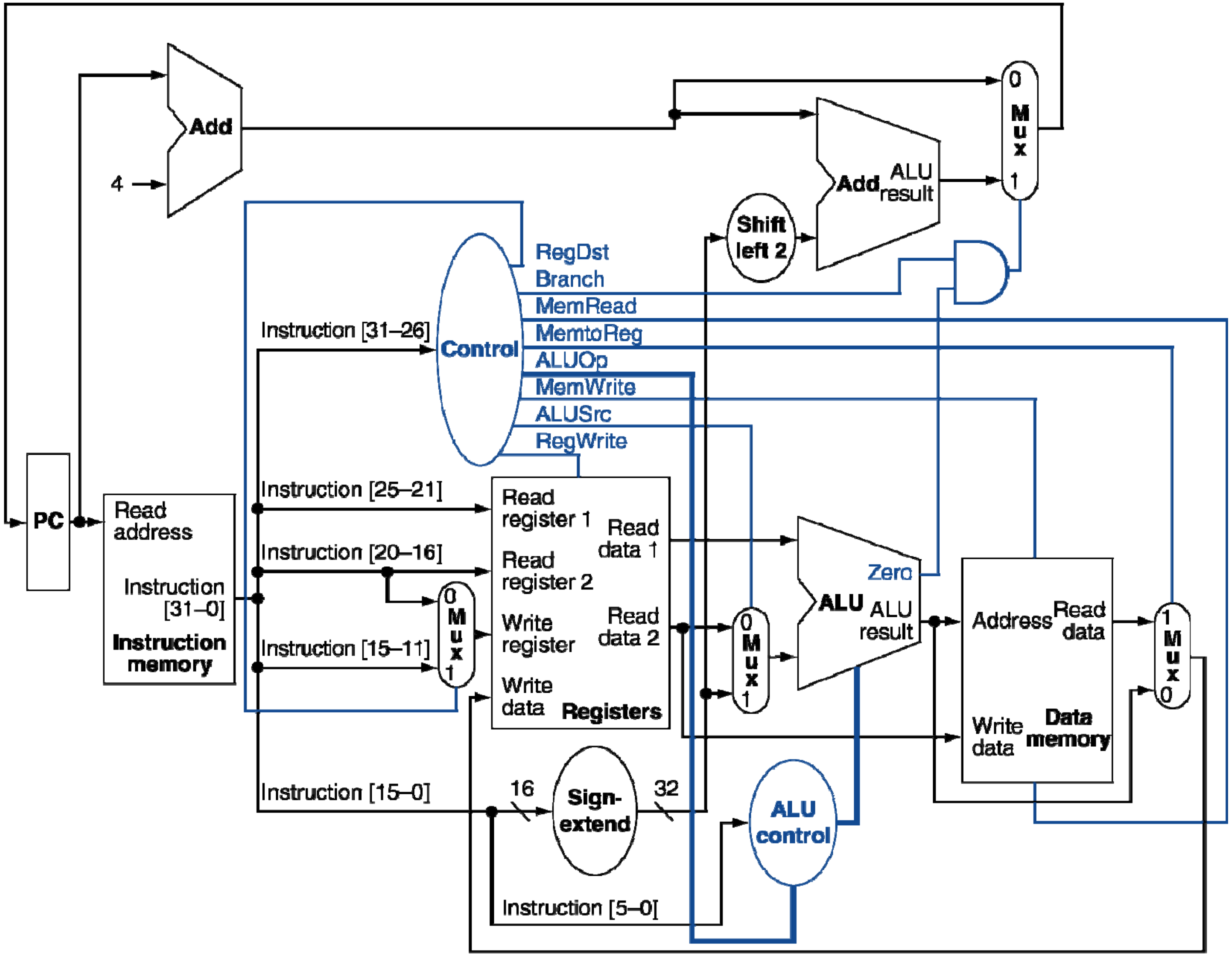
Inside the Processor 处理器 AMD多核



- AMD's Barcelona Multicore Chip
- Four out-of-order cores on one chip
- 1.9 GHz clock rate
- 65nm technology
- Three levels of caches (L1, L2, L3) on chip
- Integrated Northbridge

Inside the Processor

寄存器级别的抽象



设计全新芯片必须设计一个新的指令集架构
类似抽象类

指令集架构

Instruction Set Architecture (ISA)

硬件向上提供的硬件接口

- ISA, or simply architecture – the abstract interface between the hardware and the lowest level software that encompasses all the information necessary to write a machine language program, including instructions, registers, memory access, I/O, ...
  - Enables implementations of varying cost and performance to run identical software

保证程序不用重新编译

- The combination of the basic instruction set (the ISA) and the operating system interface is called the application binary interface (ABI) 实现了二进制代码级别的兼容性
  - ABI – The user portion of the instruction set plus the operating system interfaces used by application programmers.
    - Defines a standard for binary portability across computers.

不同Inter的芯片运行相同程序也不行的原因是因为ABI (by operating system)的不兼容性

Moore's Law

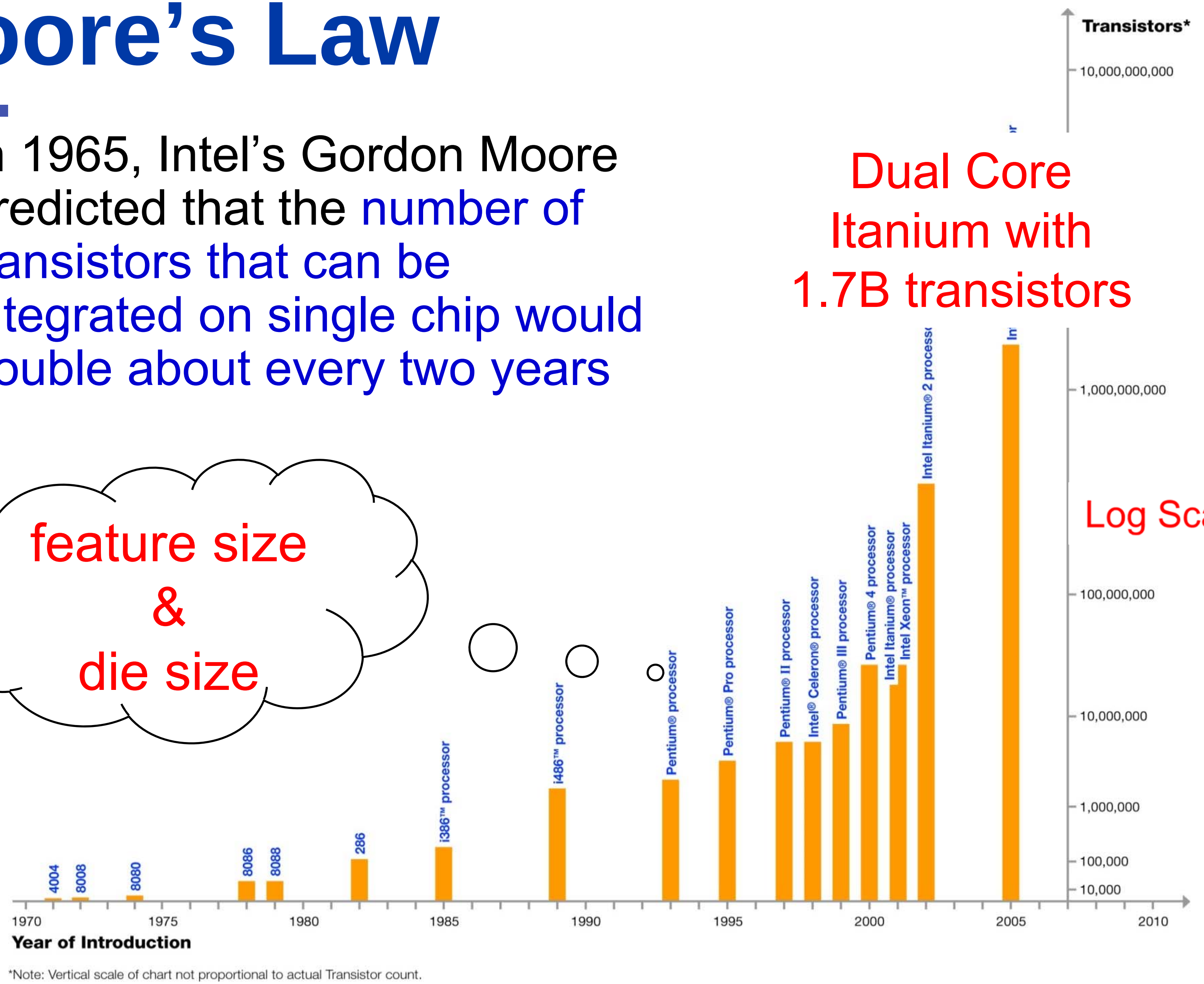
- In 1965, Intel's Gordon Moore predicted that the number of transistors that can be integrated on single chip would double about every two years

Dual Core Itanium with 1.7B transistors

feature size
&
die size

Log Scale

Intel ®



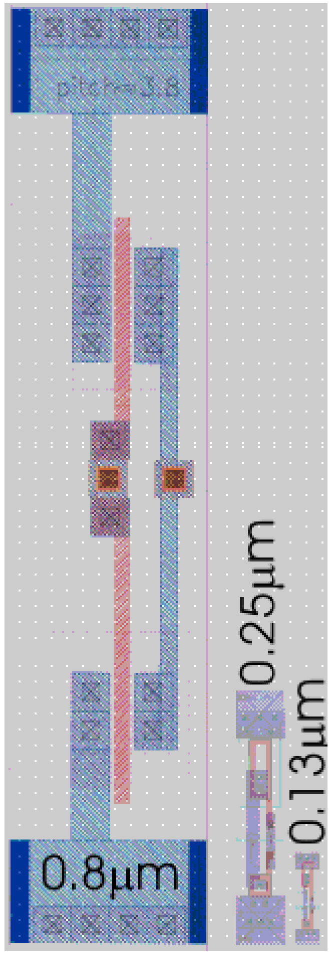
Technology Scaling

- Key enabler for smaller, faster and more power efficient computing systems

- Technology scaling has a threefold objective:
  - Increase the transistor density
  - Reduce the gate delay
  - Reduce the power consumption

  - Device dimensions (lateral and vertical) and voltages are reduced by $1/\alpha$ (~0.7)

- Technology generation spans 2-3 years



Technology Scaling Roadmap (ITRS)

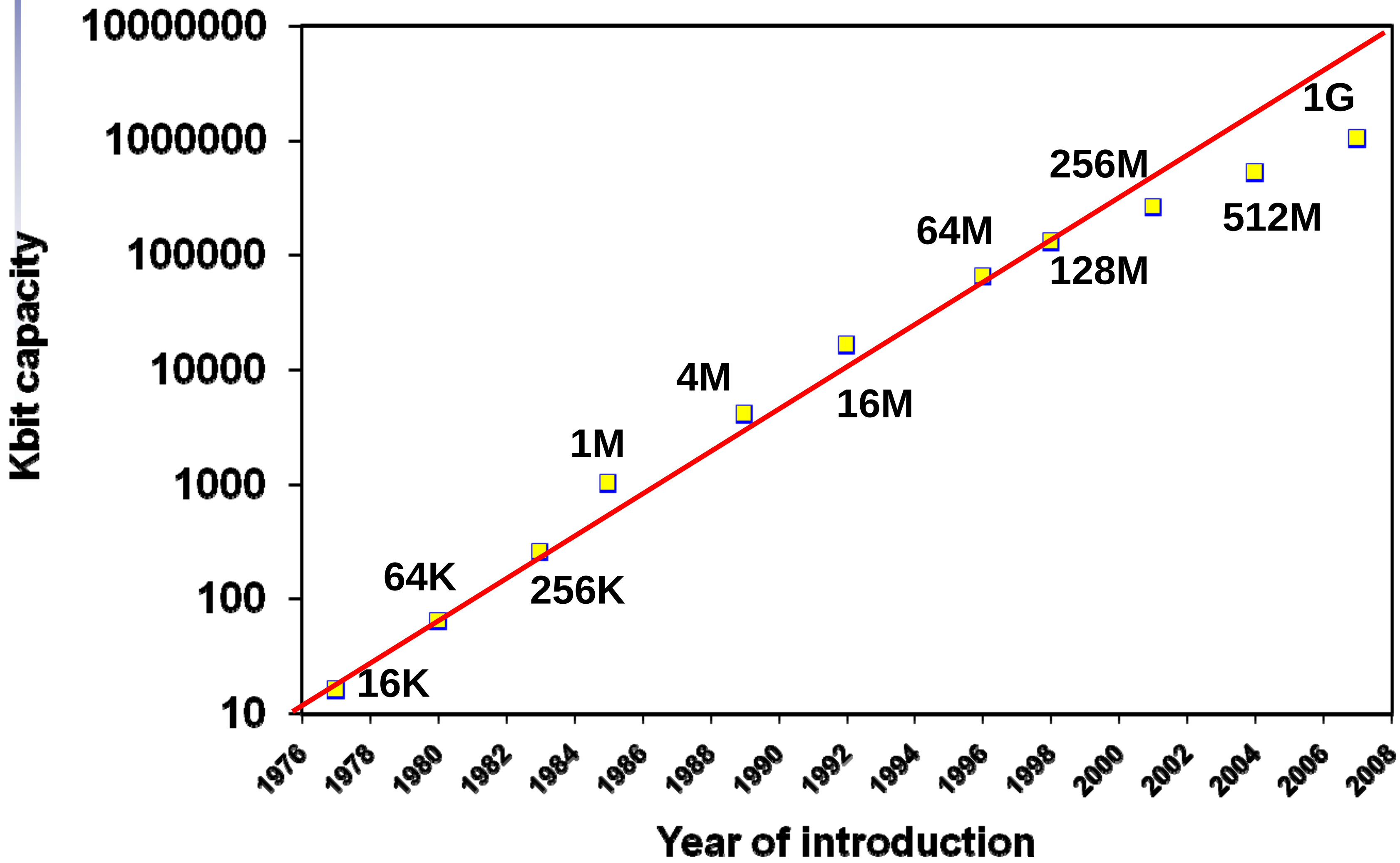
- Electronics technology continues to evolve
  - Increased capacity and performance; reduced cost

Year	2006	2008	2010	2012	2014
Feature size (nm)	90	65	45	32	22
Intg. Capacity (BT)	2	4	6	16	32

- Fun facts about 45nm transistors
  - 30 million can fit on the head of a pin
  - You could fit more than 2,000 across the width of a human hair
  - If car prices had fallen at the same rate as the price of a single transistor has since 1968, a new car today would cost about ?
  - 1 cent

Another Example of Moore's Law Impact

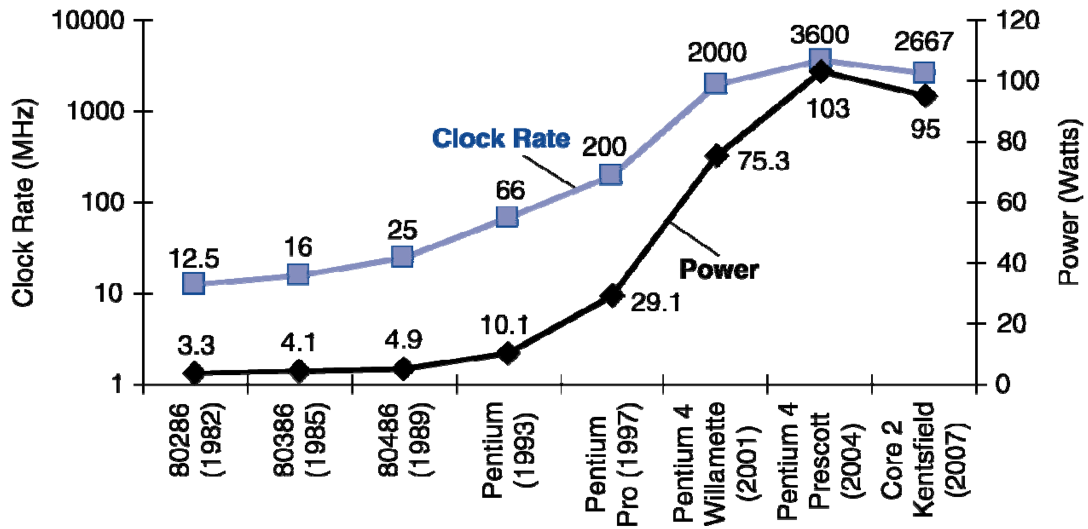
DRAM capacity growth over 3 decades



Power Trends

功耗强

为什么频率增加了300，power增加了30，所以是由于电压下降了



- In CMOS IC technology

电压　频率

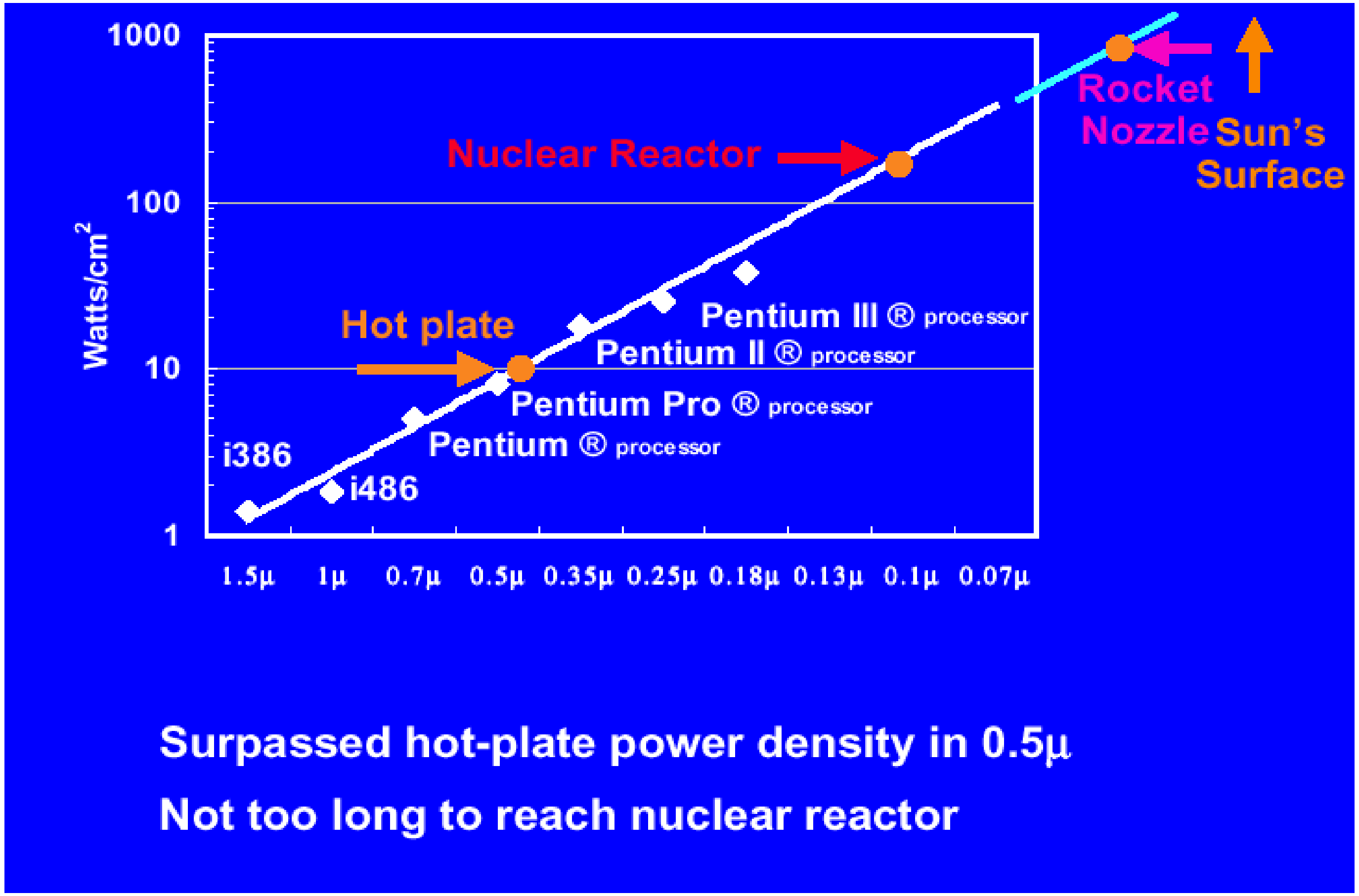
$$\text{Power} = \text{Capacitive load} \times \text{Voltage}^2 \times \text{Frequency}$$

电容

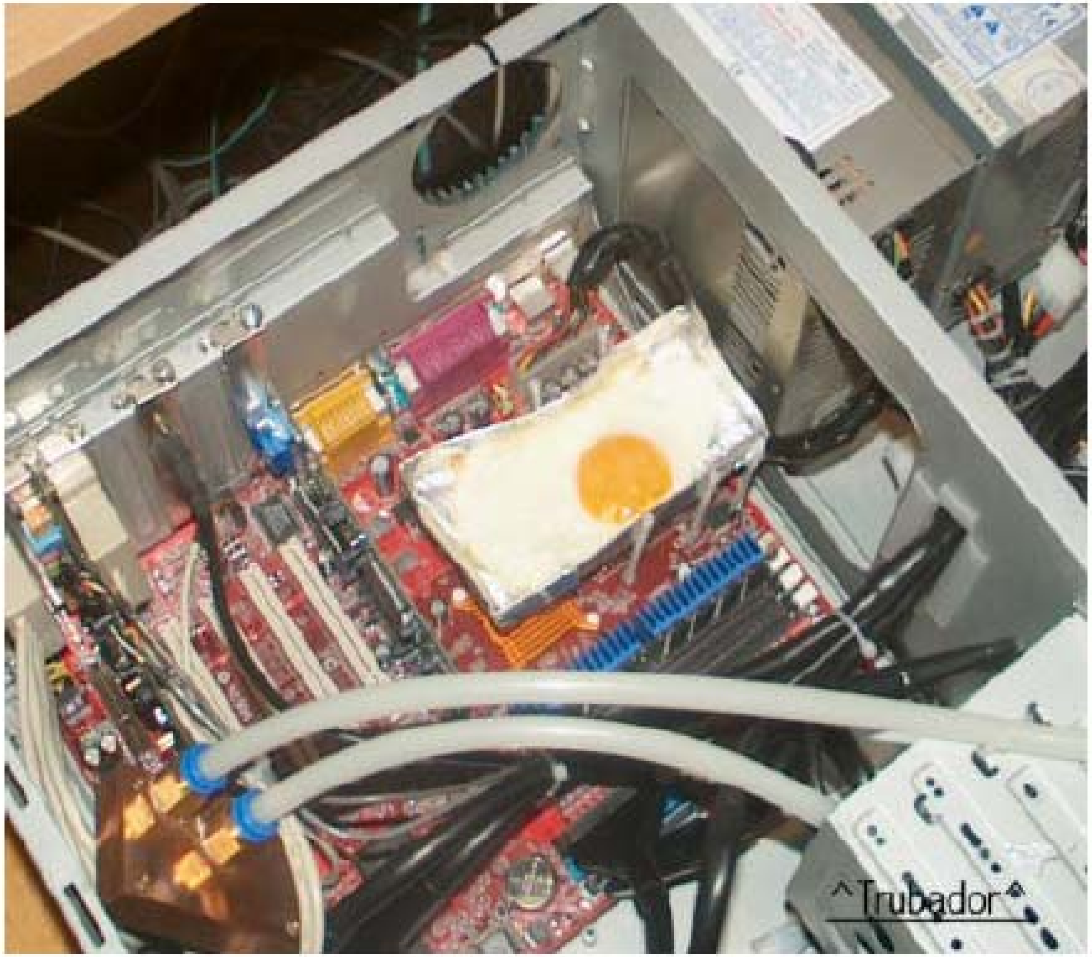
未来控制功耗，设置了动态调频功能，
负载低的时候，工作电压和频率就降低；
当负载高，都增加

$\times 30$　$5\text{V} \rightarrow 1\text{V}$　$\times 1000$

Power Density Getting Worse



Surpassed hot (kitchen) plate … why not use it?



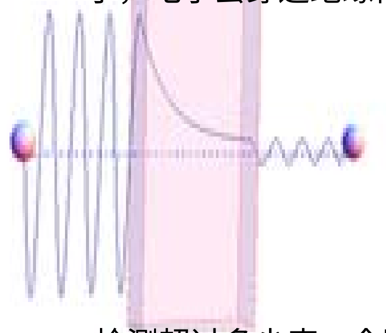
散热\隧道效应

风寒

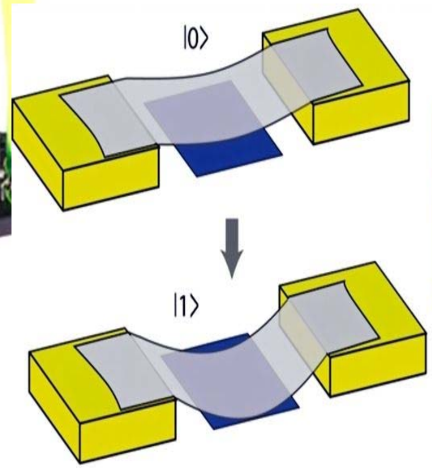
水冷

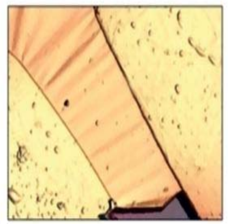
其他：液氮

温度越高，电子运动越快，越不规律，但14nm级别的数据通路可能约束不了电子，电子会穿透绝缘体



cpu检测超过多少度，会降频，否则就会烧掉

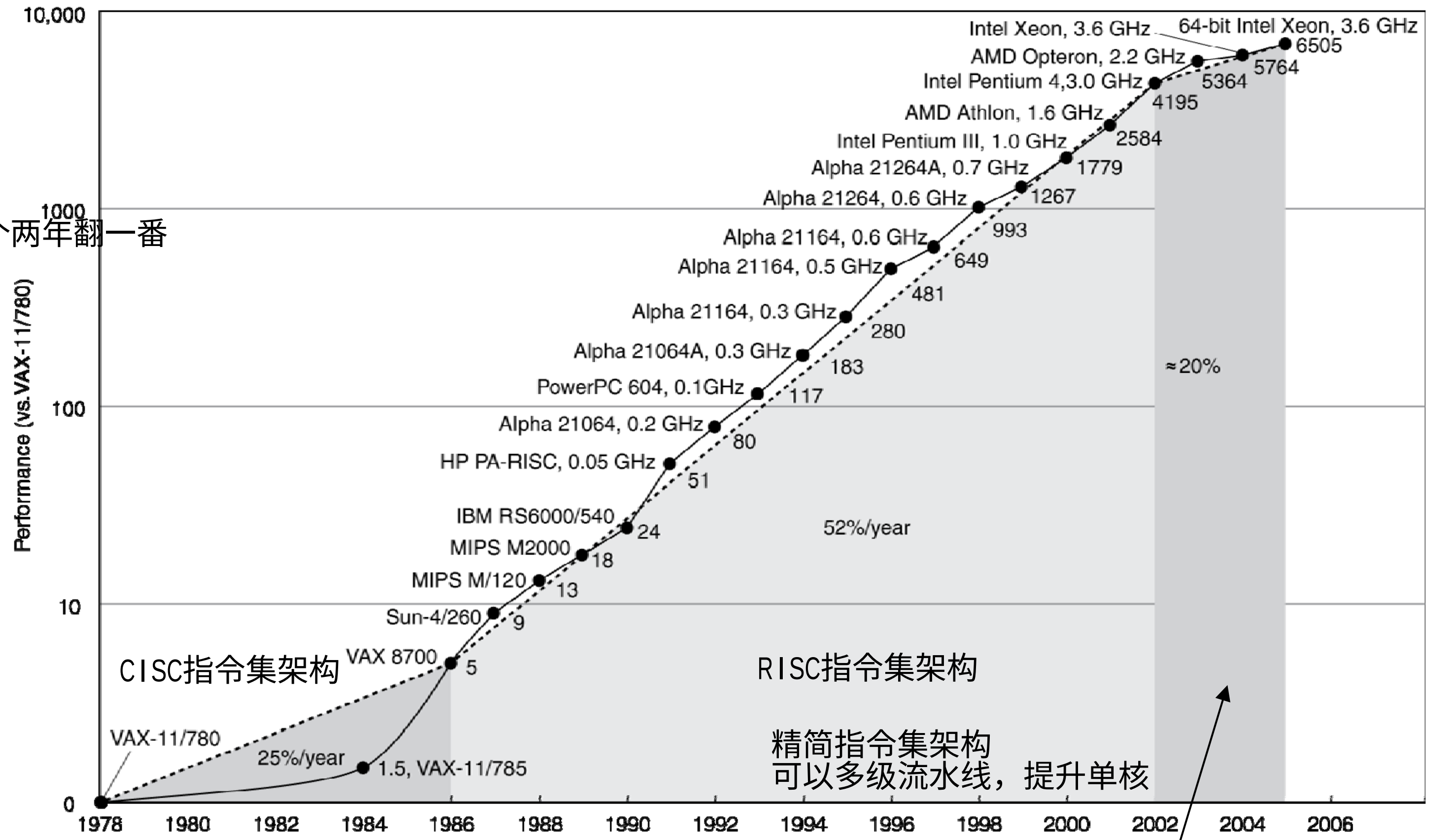




单核cpu 性能变化

Uniprocessor Performance

1.遵循摩尔定律，每个两年翻一番



复杂结构

Constrained by power, instruction-level parallelism, memory latency

The Sea Change

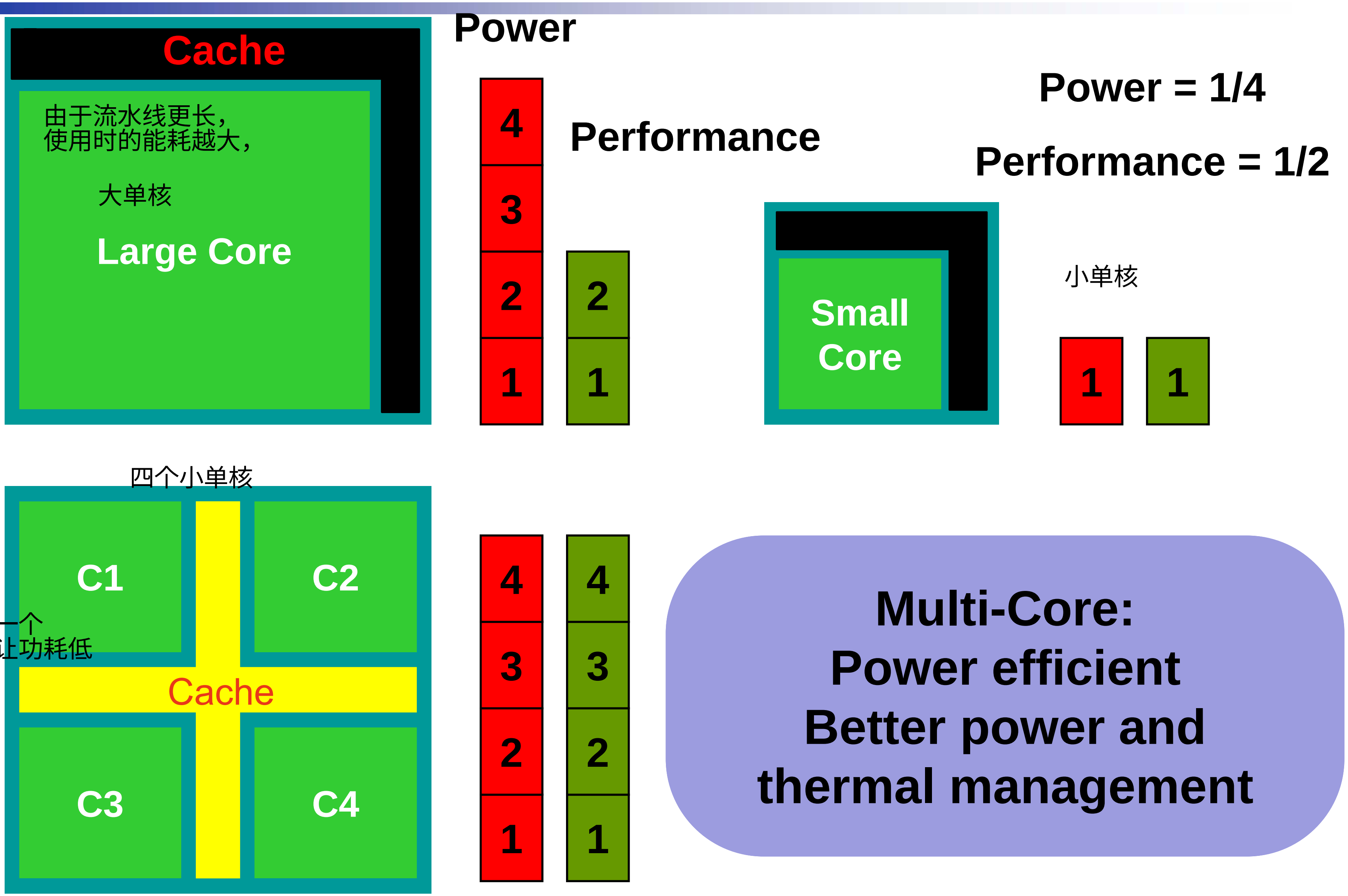
- The power challenge has forced a change in the design of microprocessors
  - Since 2002 the rate of improvement in the response time of programs on desktop computers has slowed from a factor of 1.5 per year to less than a factor of 1.2 per year
- As of 2006 all desktop and server companies are shipping microprocessors with **multiple** processors – cores – per chip

Product	AMD Barcelona	Intel Nehalem	IBM Power 6	Sun Niagara 2
Cores per chip	4	4	2	8
Clock rate	2.5 GHz	~2.5 GHz?	4.7 GHz	1.4 GHz
Power	120 W	~100 W?	~100 W?	94 W

- Plan of record is to double the number of cores per chip per generation (about every two years)

多核获得更高的能效

Why Multiple Cores on a Chip?



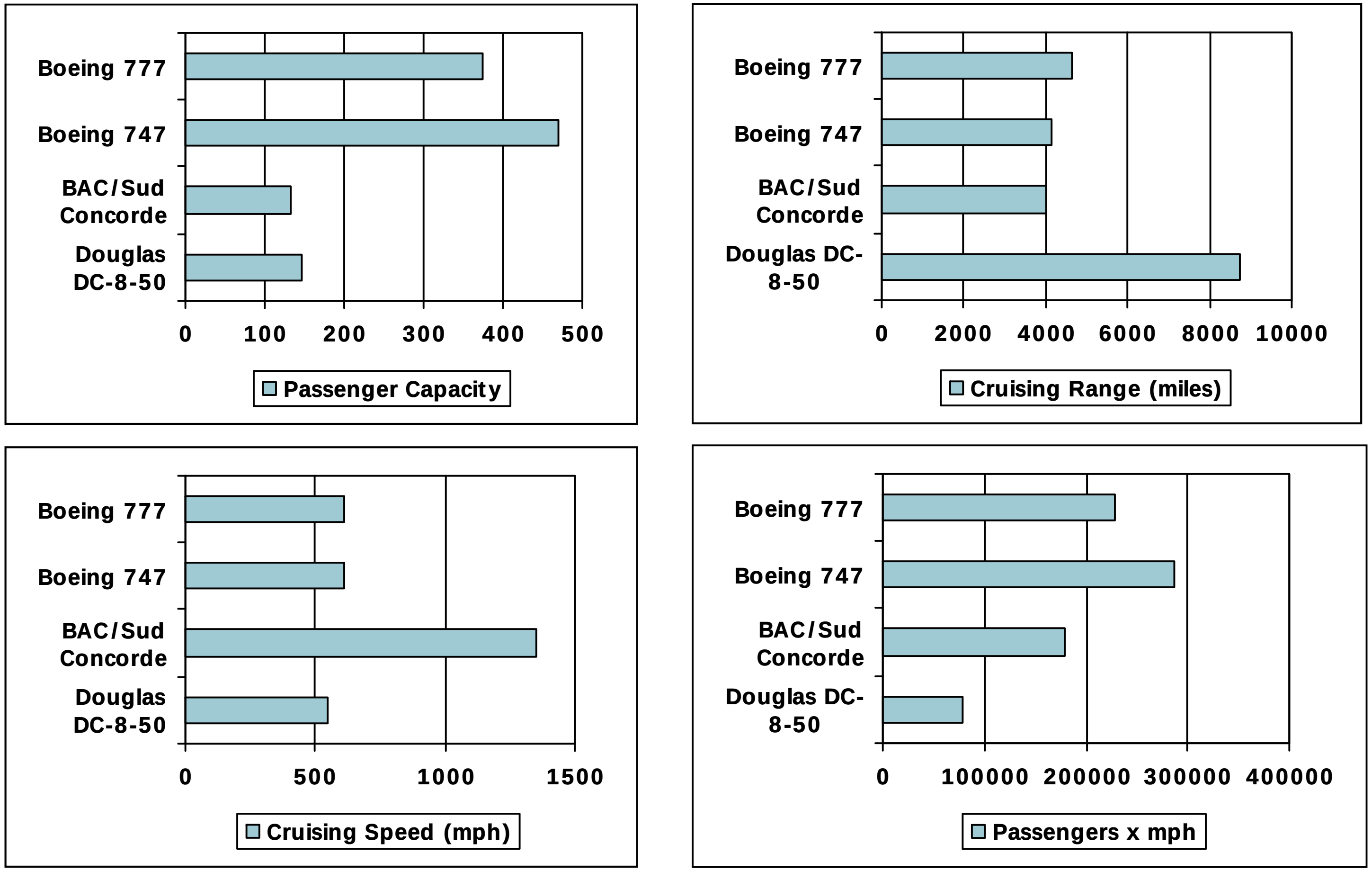
当负载低，只开一个
或者轮流工作，让功耗低

Performance Metrics

- Purchasing perspective
  - given a collection of machines, which has the
    - best performance ?
    - least cost ?
    - best cost/performance?
- Design perspective
  - faced with design options, which has the
    - best performance improvement ?
    - least cost ?
    - best cost/performance?
- Both require
  - basis for comparison
  - metric for evaluation
- Our goal is to understand what factors in the architecture contribute to overall system performance and the relative importance (and cost) of these factors

Defining Performance

- Which airplane has the best performance?



Response Time and Throughput

- Response time
  - How long it takes to do a task
    - Important to individual users
- Throughput
  - Total work done per unit time
    - e.g., tasks/transactions/… per hour
    - Important to data center managers
- How are response time & 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"

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

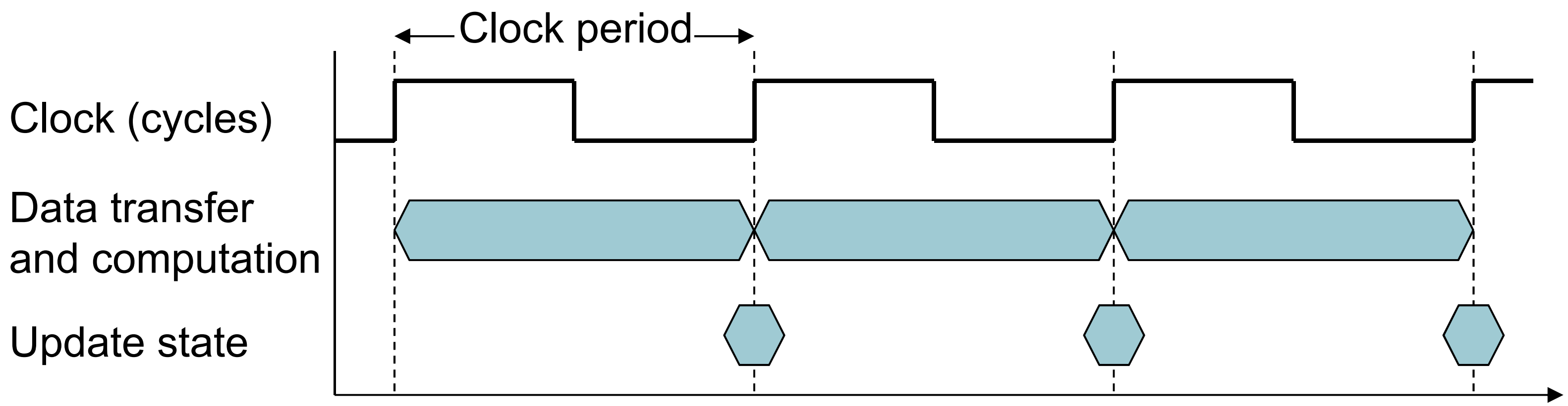
- Example: time taken to run a program
  - 10s on A, 15s on B
  - Execution Time_B / Execution Time_A
    = 15s / 10s = 1.5
  - So A is 1.5 times faster than B

Measuring Execution Time

- Execution time: seconds/program
- Elapsed time (wall clock 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: Review

- 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 Clocking: Review

- Clock rate (clock cycles per second in MHz or GHz) is inverse of clock cycle time (clock period)

$$CC = 1 / CR$$

10 nsec clock cycle => 100 MHz clock rate

5 nsec clock cycle => 200 MHz clock rate

2 nsec clock cycle => 500 MHz clock rate

1 nsec (10^{-9}) clock cycle => 1 GHz (10^{9}) clock rate

500 psec clock cycle => 2 GHz clock rate

250 psec clock cycle => 4 GHz clock rate

200 psec clock cycle => 5 GHz clock rate

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
  - Many techniques that decrease the number of clock cycles also increase the clock cycle time

CPU Time Example

- A program runs on computer A with a 2 GHz clock in 10 seconds. What clock rate must computer B run at to run this program in 6 seconds? Unfortunately, to accomplish this, computer B will require 1.2 times as many clock cycles as computer A to run the program.

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

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

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

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

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?

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

A is faster…

$$\text{CPU Time}_B = \text{Instruction Count} \times \text{CPI}_B \times \text{Cycle Time}_B$$
$$= \text{I} \times 1.2 \times 500\text{ps} = \text{I} \times 600\text{ps}$$

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

…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. What is avg. CPI?

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

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

	Instruction_ count	CPI	clock_cycle
Algorithm	X	X	
Programming language	X	X	
Compiler	X	X	
ISA	X	X	X
Core organization		X	X
Technology			X

A Simple Example

Op	Freq	CPI_i	Freq x CPI_i			
ALU	50%	1	.5	.5	.5	.25
Load	20%	5	1.0	.4	1.0	1.0
Store	10%	3	.3	.3	.3	.3
Branch	20%	2	.4	.4	.2	.4
			Σ = 2.2	1.6	2.0	1.95

- How much faster would the machine be if a better data cache reduced the average load time to 2 cycles?

  CPU time new = 1.6 x IC x CC so 2.2/1.6 means 37.5% faster

- How does this compare with using branch prediction to shave a cycle off the branch time?

  CPU time new = 2.0 x IC x CC so 2.2/2.0 means 10% faster

- What if two ALU instructions could be executed at once?

  CPU time new = 1.95 x IC x CC so 2.2/1.95 means 12.8% faster

Workloads and Benchmarks

- Benchmarks – a set of programs that form a “workload” specifically chosen to measure performance
- SPEC (System Performance Evaluation Cooperative) creates standard sets of benchmarks starting with SPEC89. The SPEC CPU2006 which consists of 12 integer benchmarks (CINT2006) and 17 floating-point benchmarks (CFP2006).

www.spec.org

- There are also benchmark collections for power workloads (SPECpower_ssj2008), for mail workloads (SPECmail2008), for multimedia workloads (mediabench), …

CINT2006 for Opteron X4 2356

Name	Description	IC×10⁹	CPI	Tc (ns)	Exec time	Ref time	SPECratio
perl	Interpreted string processing	2,118	0.75	0.4	637	9,777	15.3
bzip2	Block-sorting compression	2,389	0.85	0.4	817	9,650	11.8
gcc	GNU C Compiler	1,050	1.72	0.4	24	8,050	11.1
mcf	Combinatorial optimization	336	10.00	0.4	1,345	9,120	6.8
go	Go game (AI)	1,658	1.09	0.4	721	10,490	14.6
hmmer	Search gene sequence	2,783	0.80	0.4	890	9,330	10.5
sjeng	Chess game (AI)	2,176	0.96	0.4	37	12,100	14.5
libquantum	Quantum computer simulation	1,623	1.61	0.4	1,047	20,720	19.8
h264avc	Video compression	3,102	0.80	0.4	993	22,130	22.3
omnetpp	Discrete event simulation	587	2.94	0.4	690	6,250	9.1
astar	Games/path finding	1,082	1.79	0.4	773	7,020	9.1
xalancbmk	XML parsing	1,058	2.70	0.4	1,143	6,900	6.0
Geometric mean							11.7

High cache miss rates

Comparing and Summarizing Performance

- How do we summarize the performance for benchmark set with a single number?
  - First the execution times are normalized giving the “SPEC ratio” (bigger is faster, i.e., SPEC ratio is the inverse of execution time)
  - The SPEC ratios are then “averaged” using the geometric mean (GM)

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

- Guiding principle in reporting performance measurements is reproducibility – list everything another experimenter would need to duplicate the experiment (version of the operating system, compiler settings, input set used, specific computer configuration (clock rate, cache sizes and speed, memory size and speed, etc.))

SPEC Power Benchmark

- Power consumption of server at different workload levels
  - Performance: ssj_ops/sec (server side java 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 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
∑ssj_ops/ ∑power		493

Pitfall: Amdahl's Law

- Amdahl's law: performance enhancement possible with a given improvement is limited by the amount that the improved feature is used
- Pitfall: 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 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

Pitfall: MIPS as a Performance Metric

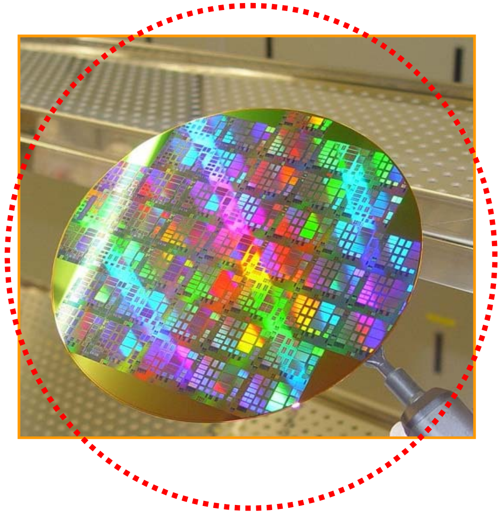
- MIPS: Millions of Instructions Per Second
  - If used as a metric to compare computers:
    - Doesn't account for differences in instruction complexity
      - Different ISAs may lead to different instruction counts for same program
    - MIPS varies between programs on the same computer!
      - Computer cannot have a single MIPS rating

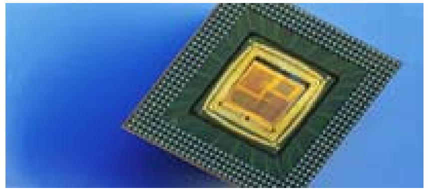
$$\text{MIPS} = \frac{\text{Instruction count}}{\text{Execution time} \times 10^6}$$

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

- e.g. CPI varied by 13x for SPEC2006 on AMD Opteron X4, so MIPS does as well

芯片的产生与应用示意



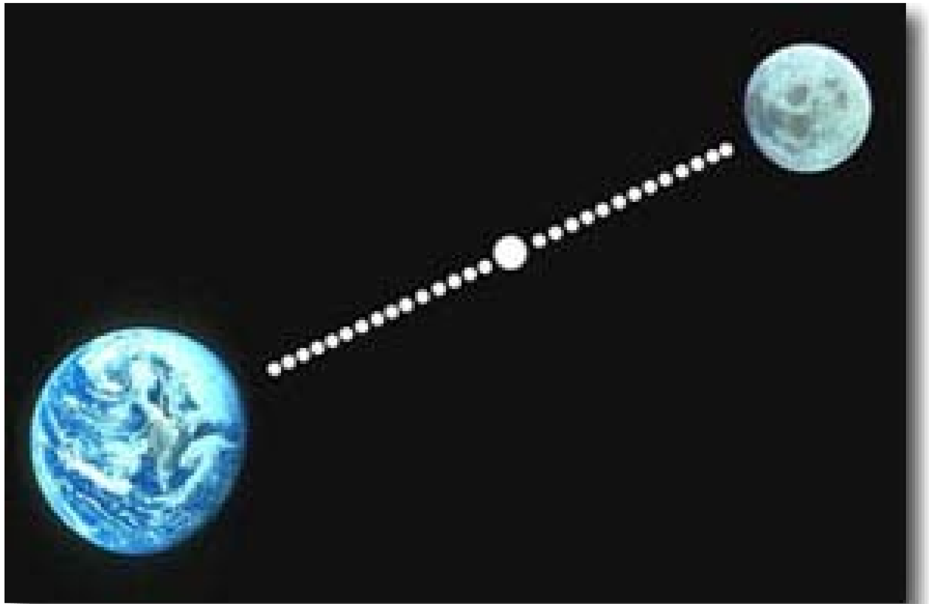


硅: 来源于沙子

Purity of Silicon of chips

- 硅体材料纯度达到 99.9999999%

  Only allow one purity in 10B atoms —— equivalent to one defect in the Pin-Pon balls filled in between Earth and Moon.



Purity of Manufacturing enviroments

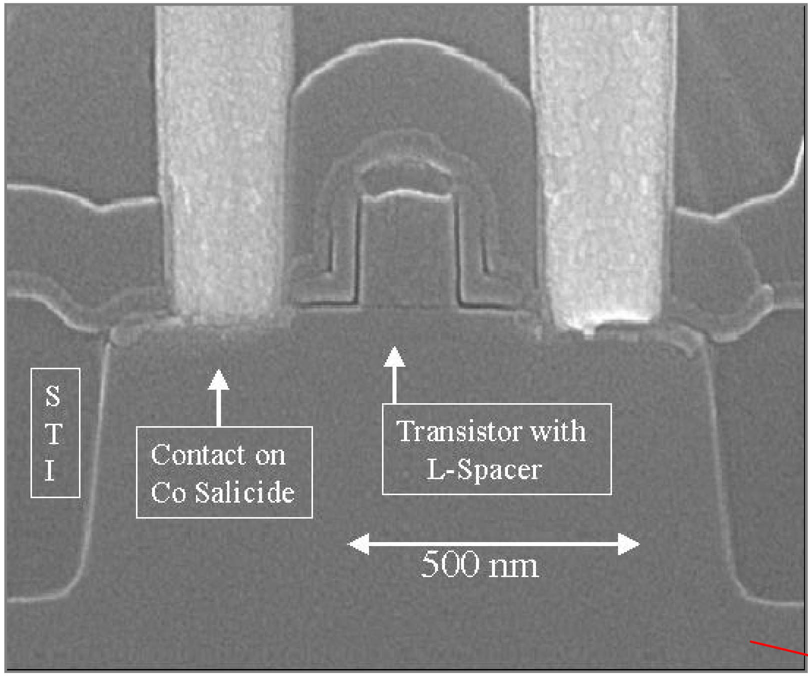
- 洁净度是药品制造的1000倍。

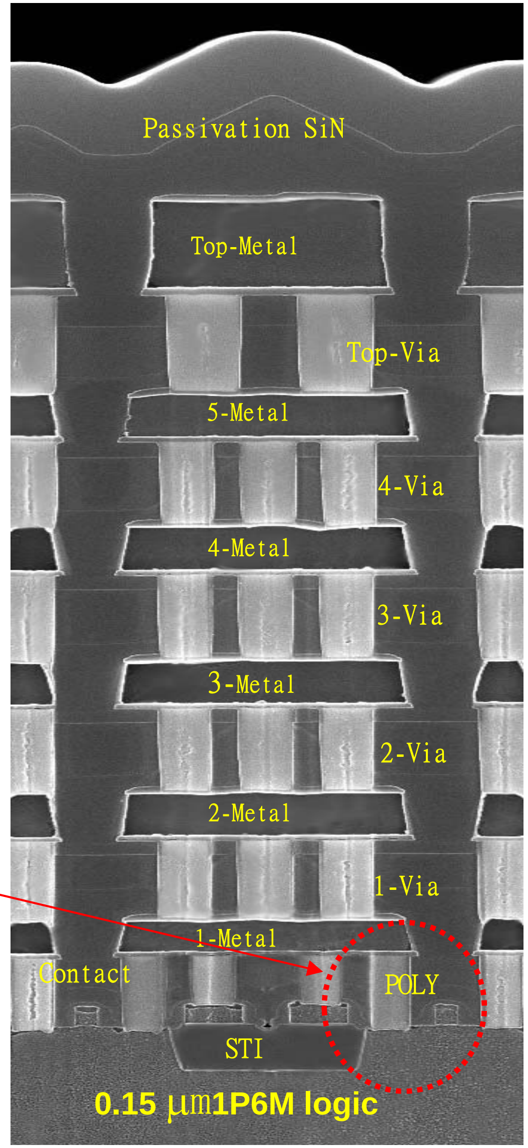
晶圆: 不能用手直接拿取

洁净空气

操作者—— 必须穿洁净工作服

分层结构



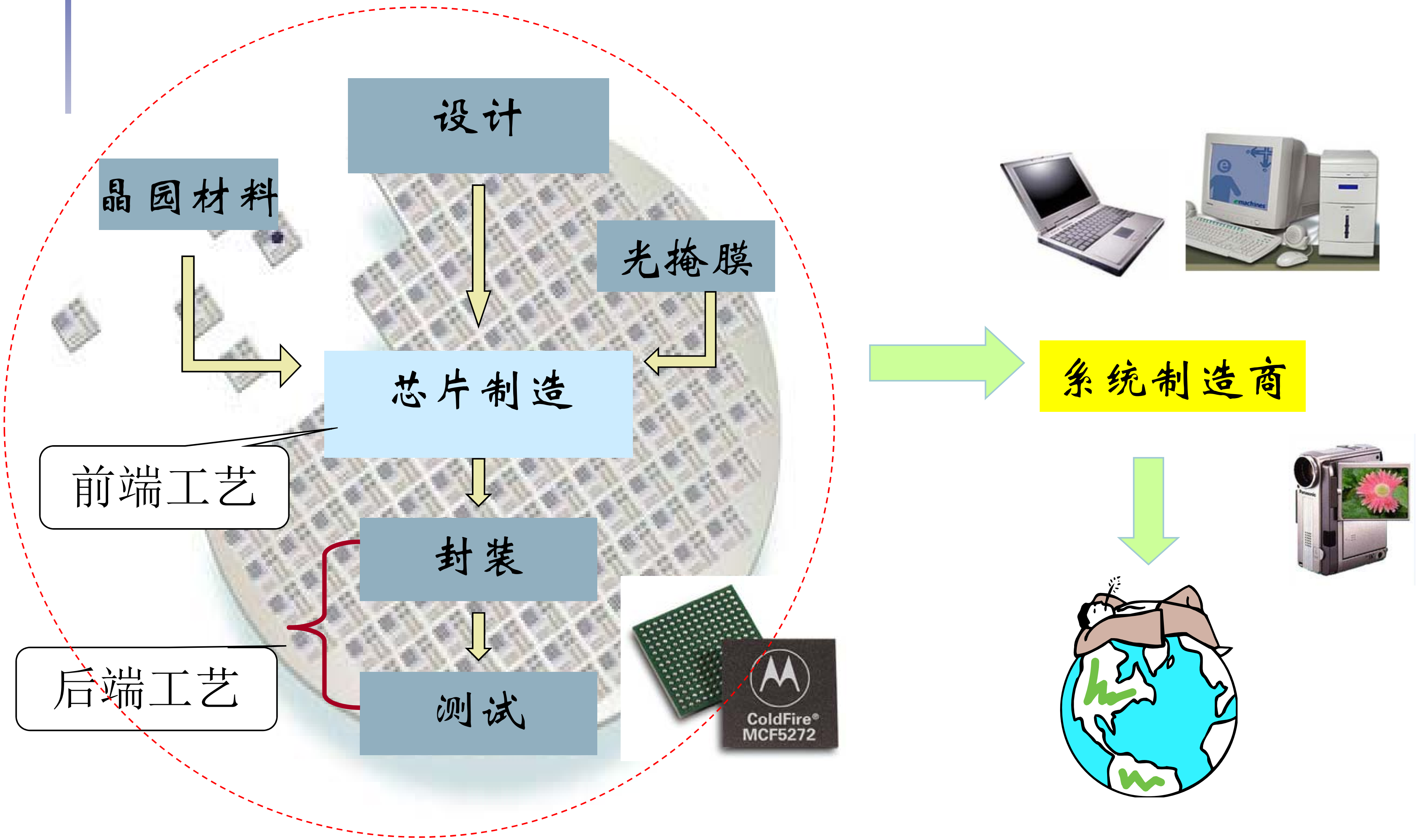


0.15 µm 逻辑电路分层结构:

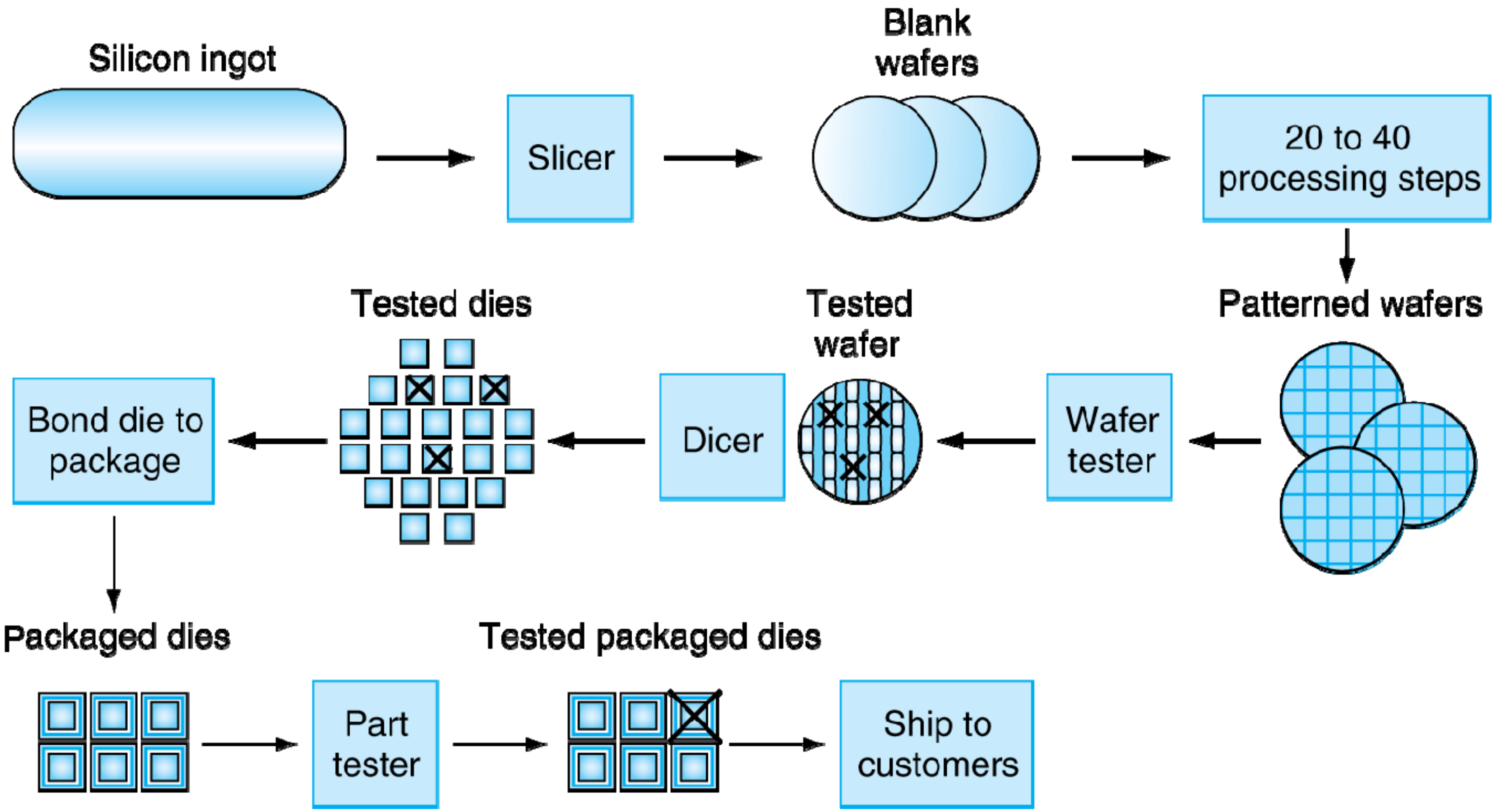
共30层光掩膜

总共需600多步骤制造

IC Industrial Chain

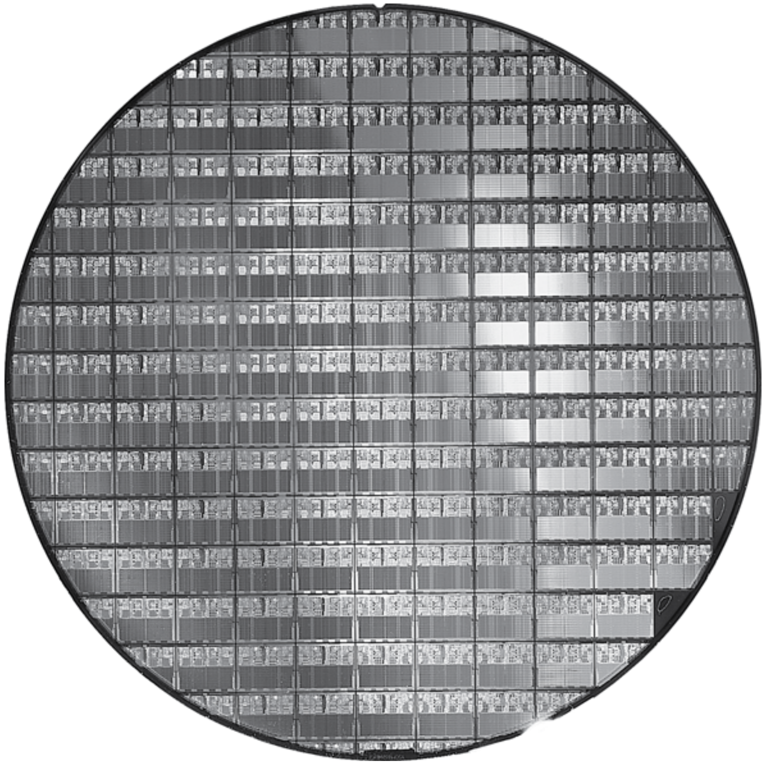


Manufacturing ICs



- Yield: proportion of working dies per wafer

AMD Opteron X2 Wafer



- X2: 300mm wafer, 117 chips, 90nm technology
- X4: 45nm technology

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

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 1

- Homework assignment

  1.1 ~1.4 , 1.6 ~1.8 , 1.13
- To be submitted to E-mail

  zhongjiangjx@163.com

  File Format： Word or PDF
- Read

  Chapter 1 of P&H

  The First Draft Report on the EDVAC