Ch1: Computer Abstractions

计算机系统概述

第1讲：计算机系统概述

第2讲：计算机性能评价

第一讲 计算机系统概述

主 要 内 容

- 计算机发展简史
  - IAS通用计算机模型机：冯.诺依曼结构
  - IBM360系列机：引入兼容性（系列机）概念
  - DEC PDP-8：引入总线结构
- 计算机系统的组成
  - 计算机硬件：CPU+MM+I/O
  - 计算机软件：系统软件+应用软件
- 计算机层次结构
  - 计算机硬件和软件的接口：指令系统
  - 计算机软件如何在硬件上执行
- 本课程主要内容

计算机的功能和特点

- 什么是计算机？
  - 计算机是一种能对数字化信息进行自动、高速运算的通用处理装置。
- 计算机的功能：
  - 数据运算、数据存储、数据传送、控制
- 计算机的特点：
  - 高速：高速元器件和“存储程序”工作方式带来高速性
  - 通用：体现在处理对象和应用领域没有限制
  - 准确：精度足够的算术运算带来准确性
  - 智能 ：逻辑推理能力带智能性

回顾：计算机发展简史

- 第一代：真空管（电子管Vacuum Tube ）1946～57年
  - 46年诞生第1台电子计算机 ENIAC
    - 体积庞大，重30吨，有18000多个真空管组成，5000次加法/秒。
    - 采用十进制表示/运算，其存储器由20个累加器组成，每个累加器可存10位十进制数，每一位数由10个真空管表示。
    - 采用手动编程，通过设置开关和插拔电缆来实现。
  - 冯·诺依曼机（Von Neumann Machine）
    - 45年冯·诺依曼（图灵也同时）提出“存储程序(Stored-program)”思想，并于46年开始设计“存储程序”计算机，被称为IAS计算机。
    - “存储程序”思想：

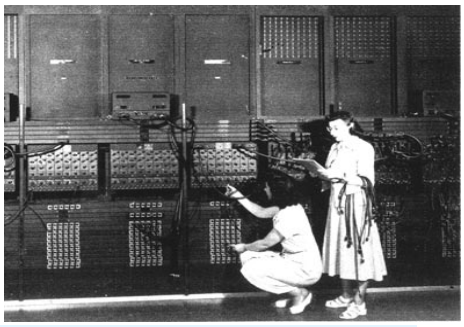
      将事先编好的程序和原始数据送入主存中，然后启动计算机工作。计算机应能在不需操作人员干预下，自动完成逐条取出指令和执行指令的任务。

SKIP

The First Generation: Vacuum Tube Computers (1946 - 1957)

The first *general-purpose computer* - **ENIAC**

ENIAC----Non von Neumann Model



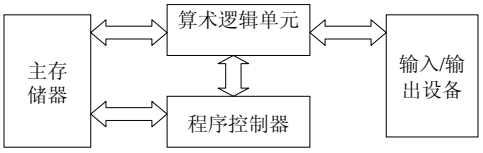
- Electronic Numerical Integrator and Computer (ENIAC)
- John Mauchly and J. Presper Eckert
- University of Pennsylvania, 1946

BACK

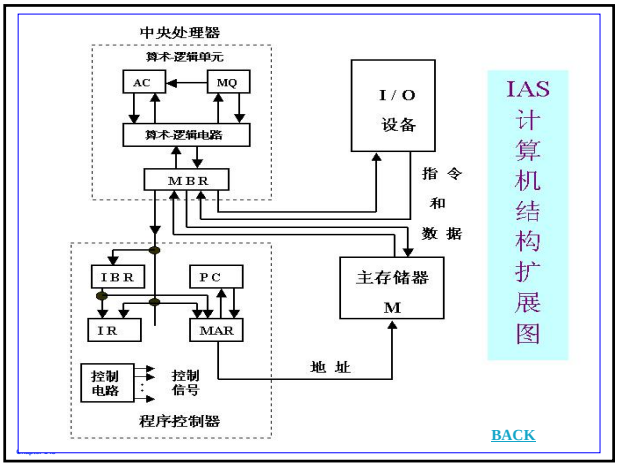
IAS计算机-通用计算机原型

1946年由冯·诺依曼及其同事在普林斯顿高级研究院IAS设计以“存储器”为中心，其后通用机原型

IAS计算机的一般结构，它包含5个部件:



IAS (the Institute for Advance Study at Princeton)



冯·诺依曼结构的主要思想

1. 计算机应由运算器、控制器、存储器、输入设备和输出设备五个基本部件组成。
2. 各基本部件的功能是：
   - 存储器不仅能存放数据，而且也能存放指令，形式上两者没有区别，但计算机应能区分数据还是指令；
   - 控制器应能自动执行指令；
   - 运算器应能进行加/减/乘/除四种基本算术运算，并且也能进行一些逻辑运算和附加运算；
   - 操作人员可以通过输入设备、输出设备和主机进行通信。
3. 内部以二进制表示指令和数据。每条指令由操作码和地址码两部分组成。操作码指出操作类型，地址码指出操作数的地址。由一串指令组成程序。
4. 采用“存储程序”工作方式。

计算机发展简史

- 第二代：晶体管 1958～64年
  - 元器件：逻辑元件采用晶体管，内存由磁芯构成，外存为磁鼓与磁带。
  - 特点：变址，浮点运算，多路存储器，I/O处理机，中央交换结构(非总线结构)。
  - 软件：使用高级语言，提供了系统软件。
  - 代表机种：IBM 7094 (scientific)、1401 (business)和 DEC PDP-1

晶体管：Transistor

DEC PDP-1

计算机发展简史

- 第三代：SSI/MSI 1965～71年
  - 元器件：逻辑元件与主存储器均由集成电路（Integrated Circuit）实现 。
  - 特点：微程序控制，Cache，虚拟存储器，流水线等。
  - 代表机种： IBM 360和DEC PDP-8（大/巨型机与小型机同时发展 ）
    - 巨型机(Supercomputer)：Cray-1
    - 大型机(Mainframe)：IBM360系列
    - 小型机(Minicomputer)：DEC PDP-8

IBM System/360系列计算机

- IBM公司于1964年研制成功
- 引入了“兼容机”（“系列机”）概念
  - 兼容机的特征 ：
    - 相同的或相似的指令集
    - 相同或相似的操作系统
    - 更高的速度
    - 更多的I/O端口数
    - 更大的内存容量
    - 更高的价格

IBM 360

低端机指令集是高端机的一个子集，称为“向后兼容”。功能相同，而性能不同。

问题1：引入“兼容机”有什么好处？

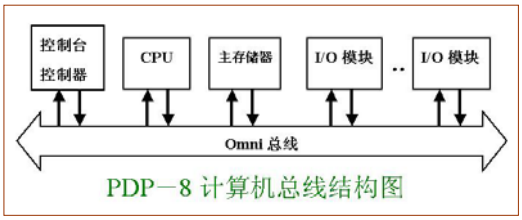
问题2：实现“系列机”的关键是什么？

DEC公司的PDP-8机

- 同在64年出现。与IBM 360相比，价格更低、更小巧，因而被称为小型机（Minicomputer）
- PDP-8“创造了小型机的概念，并使之成为数十亿美元的工业”，使DEC成为了最大的小型机制造商。
- 主要特点：首次采用总线结构。

Omnibus总线包含了96个独立的信号通道，用以传送控制、地址和数据信号。这种结构具有高度的灵活性，允许将模块插入总线以形成各种配置。

PDP–8/E计算机系统框图



PDP—8 计算机总线结构图

问题：“总线结构”有什么好处？

具有高度的灵活性，允许将模块插入总线以形成各种配置
节省器件，体积小，价格便宜

计算机发展简史

- 以后几代（标准、意见不一）
  （注：有称第四代是VLSI，从80年代开始；也有称第四代是LSI，从72年开始；有的又分成LSI时代和VLSI时代）
- （第四代：LSI/VLSI/ULSI 1972～至今）
  - 微处理器和半导体存储器的技术发展迅猛，微型计算机出现。使计算机以办公设备和个人电脑的方式走向普通用户。
    半导体存储器
    - 70年Fairchild公司生产出第一个相对大容量半导体存储器
    - 74年位价格低于磁芯的半导体存储器出现，并快速下跌
    - 从70年起，存储密度呈4倍提高（几乎是每3年）。

    微处理器
    - 微处理器芯片密度不断增加，使CPU中所有元件放在一块芯片上成为可能。71年开发出第一个微处理器芯片4004。
  - 特点：共享存储器，分布式存储器及大规模并行处理系统

计算机的五个发展阶段

代号	大致年份	技术标志	代表速度(每秒执行指令数)
1	1946—1957	真空管	40,000
2	1958—1964	晶体管	200,000
3	1965—1971	中小规模集成电路	1,000,000
4	1972—1977	大规模集成电路	10,000,000
5	1978—	超大规模集成电路	100,000,000

摘自《COMPUTER ORGANIZATION AND ARCHITECTURE Design for Performance》William Stallings

除了元器件外，系统结构的特点是换代的一个重要标志
1965-75期间，元器件延迟降为1/10，但指令时间却降为1/100。这说明了计算机结构改进的重要性。

Moore’s Law

Moore’s Law

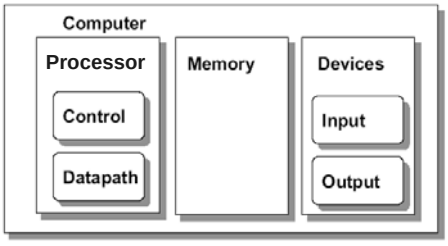
- The number of transistors on a microchip doubles about every 18-24 months,
- The speed of a microprocessor doubles about every 18-24 months,
- The price of a microchip drops about 48% every 18-24 months,
  - assuming the performance metric (processor speed or memory capacity) of the chip stays the same.
- **Official Definition of Moore’s Law:**

http://www.intel.com/intel/museum/25anniv/hof/moore.htm

Computer System

- Computer System = Hardware + Software
- Hardware (tangible objects 具体的物理器件)
  e.g integrated circuit, printed circuit board, cables,power supplies,memories,printer
- Software (encoded symbol 对抽象概念进行符号编码)
  Software=program+document
  program = algorithms+data
  program 由若干条指令或语句组成

Hardware



Processor：CPU , 处理器
Control：控制器，控制单元
DataPath： 数据通路，执行单元

Memory：内存（主存MM)
Device：外设，外围设备，外部设备
Input：输入设备
Output：输出设备

功能部件1--CPU

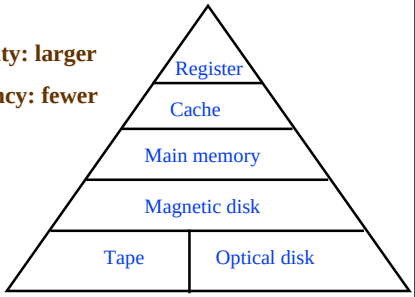
- Processor (CPU - Central Processing Unit)：是计算机的“大脑”
  - 功能: 执行程序（Execute programs）
  - 组成: Control Unit + Data path
    - Control Unit（控制单元）：
      功能：对指令进行译码，送出控制信号
    - Datapath （数据通路）：
      功能：完成指令的执行
      核心：ALU(Arithmetic Logic Unit-算术逻辑单元)+Register(寄存器)
      - ALU用来执行算术逻辑运算
      - 寄存器用来存储临时的数据或控制信息，常用寄存器有：
        GRS(General Register Set)：存放操作数和中间结果
        PC(Program Counter)：存放下条要执行指令的地址
        IR(Instruction Register)：存放当前指令

功能部件2--存储器

- 存储器相当于计算机中的“仓库”，有多种规格的“仓库”
  - 功能：存储程序和数据
  - 组成：(层次化结构 Hierarchies)： *Primary + Secondary*
    - 内存（Primary memory）: Cache + Main memory (MM)
      - Cache(高速缓存): 最近使用的数据和指令
      - MM(主存): 存放被启动的程序中的部分数据和指令
    - 外存（Secondary Storage）: Magnetic disk,Tape,Optical disk
      - Magnetic disk(磁盘): 存放系统中所有的软件、文档
      - Tape(磁带) or Optical disk(光盘): 脱机存档

Memory Hierarchies

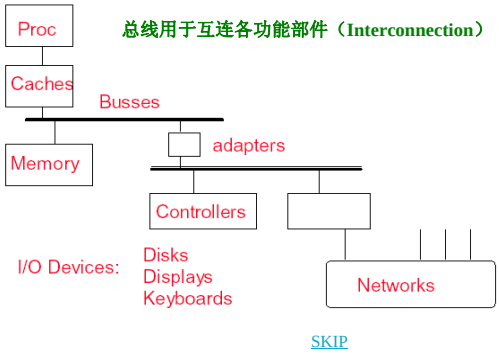
Storage capacity: larger
Access frequency: fewer
Speed: Slower
Cost: cheaper



功能部件3—输入/输出

- Input/Output (I/O)
  - 功能: 各种信息的输入/输出
  - 组成: I/O Controller + I/O Device
    - I/O Controller（I/O控制器）：
      – 控制外设，完成与主机侧、外设侧的通信
    - I/O device （I/O设备）：
      – 输入/输出信息
      - Keyboard
      - Printer
      - Display (CRT Monitor) etc

总线：互连(Interconnection)

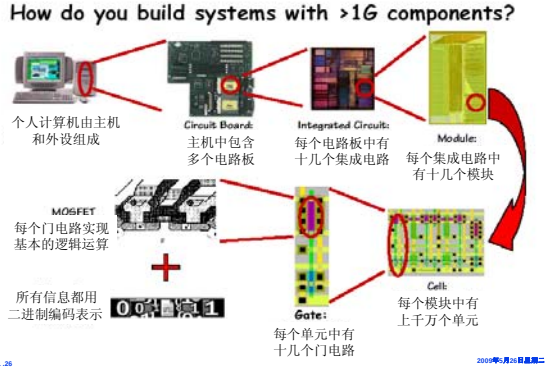


总线用于互连各功能部件（Interconnection）

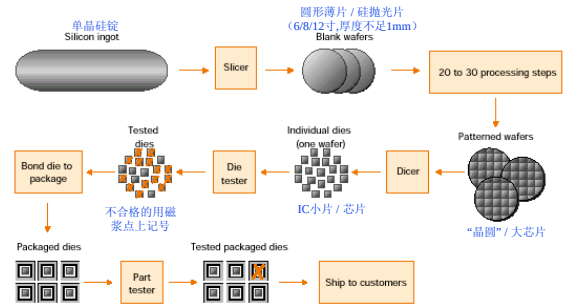
SKIP

打开计算机来看看

解剖计算机



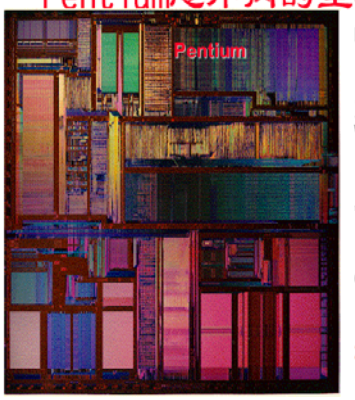
Integrated Circuits manufacturing process



封装：将芯片固定在塑胶或陶瓷基座上，把芯片上蚀刻出来的引线与基座底部伸出的引脚连接，盖上盖板并封焊成芯片

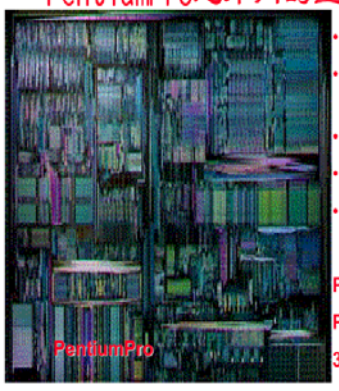
约需400多道工序！

Pentium芯片内的主要功能块



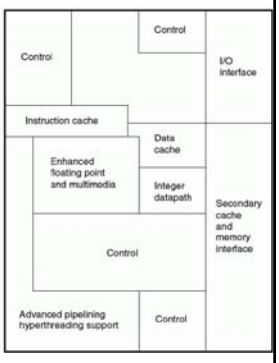
- Die Area: 91 mm²
- 直径8 inch(200mm)的Wafer最多可做196个Die
- ≈ 3,300,000 Transistors
- Cache: ≈1M Transistors
- 296 Pins

PentiumPro芯片内的主要功能块



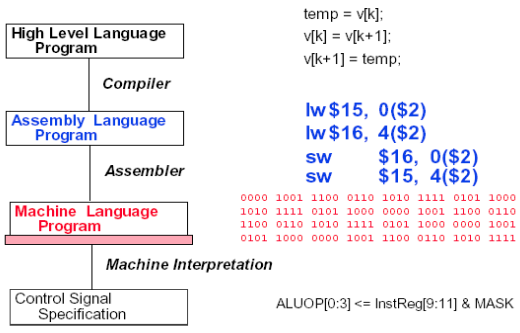
- Die Area: 306 mm²
- 直径8 inch(200mm)的Wafer最多可做78个Die
- ≈ 5,500,000 Transistors
- Cache: ≈1M Transistors
- External Cache: 31M Transistors
- PentiumPro Package = PentiumPro+ExternalCache
- 387 Pins

Pentium4处理器内部布局



Back

Hardware/Software Interface



Back

Software

- System software(系统软件) - 简化编程过程，使硬件资源被有效利用
  - 操作系统（Operating System）：硬件资源管理，用户接口
  - 语言处理系统：翻译程序+ Linker, Debug, Loader, etc …
    - 翻译程序(Translator)有三类：
      - ★ 汇编程序(Assembler)：将汇编语言源程序翻译为机器语言目标程序文件。
      - ★ 编译程序(Complier)：将高级语言源程序翻译为汇编语言或机器语言目标程序文件。
      - ★ 解释程序(Interpreter)：将高级语言语句逐条翻译成机器指令并立即执行。不生成目标文件。
  - 其他实用程序: 如：磁盘碎片整理程序、备份程序等
- Application software(应用软件) - 解决具体应用问题/完成具体应用任务
  - 各类媒体处理程序：Word/ Image/ Graphics/…
  - 管理信息系统 (MIS)
  - Game, …

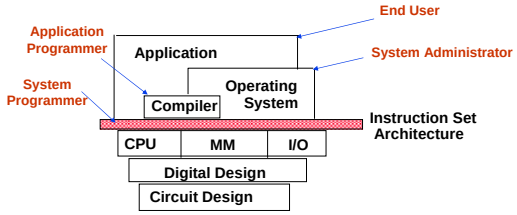
Hardware/Software Interface

The Instruction Set Architecture (ISA): 指令集体系结构

机器语言由指令代码构成，能被硬件直接执行。

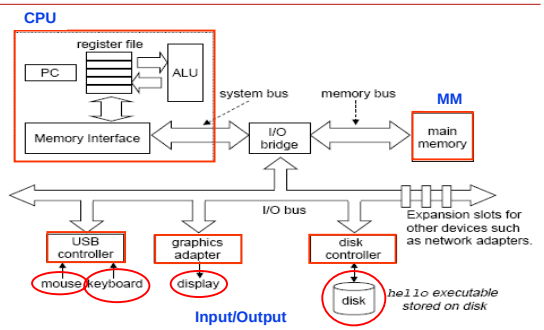
Computer Hierarchy

- Co-ordination of *levels of abstraction*



- 上图给出的是计算机系统的层次结构（指令系统是软/硬件的交界面）
- 不同用户工作在不同层次，所看到的计算机不一样

一个典型系统的硬件组成



PC：程序计数器；ALU：算术/逻辑单元；USB：通用串行总线

一个典型程序的转换处理过程

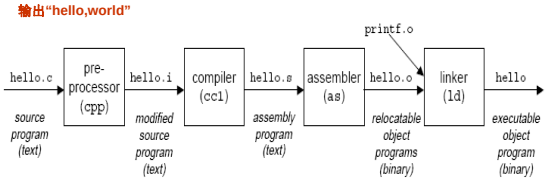
经典的“ hello.c ”C-源程序

```
1 #include <stdio.h>
2
3 int main()
4 {
5 printf("hello, world\n");
6 }
```

程序的功能是：
输出“hello,world”

hello.c的ASCII文本表示

```
# i n c l u d e <sp> < s t d i o .
35 105 110 99 108 117 100 101 32 60 115 116 100 105 111 46
h > \n \n i n t <sp> m a i n ( ) \n {
104 62 10 10 105 110 116 32 109 97 105 110 40 41 10 123
\n <sp> <sp> <sp> <sp> p r i n t f ( " h e l
10 32 32 32 32 112 114 105 110 116 102 40 34 104 101 108
l o , <sp> w o r l d \ n " ) ; \n }
108 111 44 32 119 111 114 108 100 92 110 34 41 59 10 125
```



Hello程序的执行过程

Unix系统启动可执行程序hello的shell命令行：

unix> ./hello [Enter]
hello, world
unix>

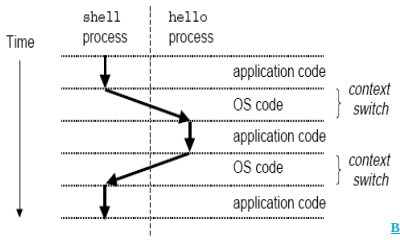
Hello程序被启动后，计算机的动作过程如下：

1. Shell程序读取字符串“./hello”中各字符到寄存器，然后存放到主存；
2. “Enter”键输入后，shell根据主存中的字符串“hello”到磁盘上找到特定的hello目标文件，将其包含的指令代码和数据（“hello, world\n”）从磁盘读到主存；
3. 处理器从hello主程序的指令代码开始执行；
4. Hello程序将“hello, world\n”串中的字节从主存读到寄存器，再从寄存器输出到显示器上

Hello程序执行过程中进程间的切换

unix> ./hello [Enter]
hello, world
unix>

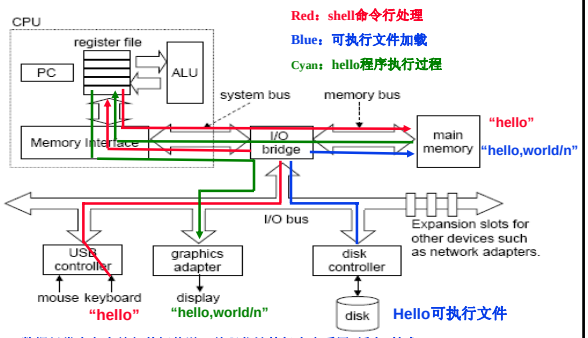
- 开始，Shell进程等待命令行输入
- 输入“Hello”后Shell进行系统调用
- OS保存shell上下文，创建并换入hello进程
- Hello进程中止时，进行系统调用
- OS恢复shell进程上下文，并换入shell进程



由于在一个进程的整个生命周期中，有不同进程在处理器交替运行，所以运行时间很难准确、重复测量。

BACK

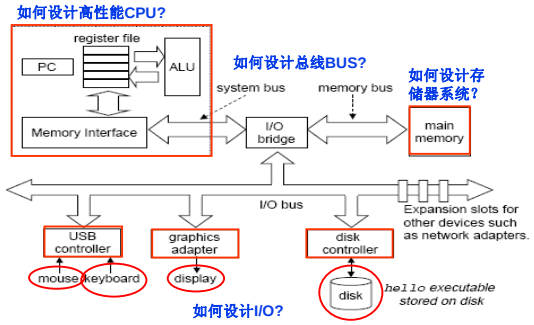
Hello程序的数据流动过程



Red：shell命令行处理
Blue：可执行文件加载
Cyan：hello程序执行过程

数据经常在各存储部件间传送。故现代计算机大多采用“缓存”技术！
所有过程都是在CPU执行指令所产生的控制信号的作用下进行的。

该课程的主要学习内容



信息（指令和数据）在计算机中如何表示?
指令系统如何设计?

第一讲小结

- 软件、硬件之间的接口：指令系统
  - ISA：屏蔽硬件细节，简化程序员的使用
- 层次化的好处
  - 隐藏低层层次的实现细节，简化各层次上用户的使用
- 由五部分组成（有些书称为三部分），总线互连
  - Processor: (1) datapath and (2) control
  - (3) Memory
  - I/O: (4) Input devices and (5) Output device
- 存储系统按照容量、速度、价格分层
  - 寄存器：最贵，最快，容量最小，离ALU最近
  - Cache：贵，快，容量小，离CPU近
  - Main memory：便宜， 慢，容量大，离CPU远
  - 磁盘\磁带\光盘：最便宜，最 慢，容量最大，离CPU最远
- I/O的组织呈多样性
  - 速度范围大: graphics vs. keyboard
  - 考虑因素多: speed, standard, cost ... etc.
  - 实现方式多样：机械、光电、磁性材料, etc.

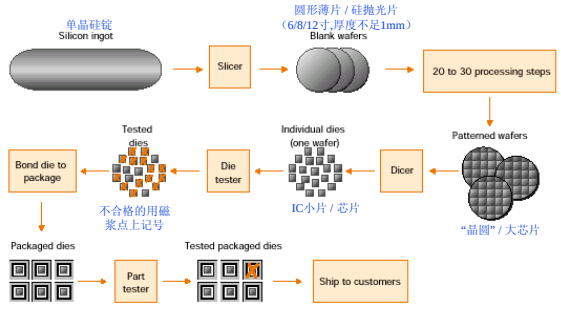
第二讲 计算机性能评价

主 要 内 容

- 制造成本（manufacturing cost）
- 衡量计算机性能的基本指标
  - 响应时间（response time）
    - 执行时间（execution Time）、等待时间（latency）
  - throughput （吞吐量）
    - 带宽（bandwidth）
- 计算机性能测量
  - CPU Time=cycle time x CPI x Instructions / program
- 指令执行速度（MIPS、MFLOPS）
- 基准程序（ Benchmark）

回顾: Integrated Circuits Costs --- manufacturing process

在考察性能前，先考察成本！



封装：将芯片固定在塑胶或陶瓷基座上，把芯片上蚀刻出来的引线与基座底部伸出的引脚连接，盖上盖板并封焊成芯片

约需400多道工序！

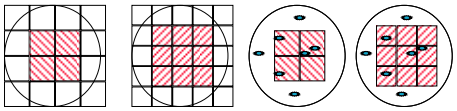
Integrated Circuits Costs 公式

$$\text{Die cost} = \frac{Cost\_per\_wafter}{Die\_per\_wafer \times Yield}$$

$$\text{Dies per wafer} = \frac{wafer\_area}{Die\_area}$$

芯片成本与以下因素有关：

- 圆晶价格
- 圆晶所含小片数
- 小片合格率



小片合格率

$$\text{Die Yield} = \frac{1}{(1+(Defect\_per\_area \times Die\_area))^2}$$

由此看出：每个圆晶片上的小片数、集成电路成本都与芯片面积有关！

Other Costs

$$\text{IC cost} = \frac{\text{Die cost} + \text{Testing cost} + \text{Packaging cost}}{\text{Final test yield}}$$

封装成本（Packaging cost）：取决于引脚数、散热性等

Chip	*Die cost*	*Package pins*	*Package type*	*Package cost*	*Test & Assembly*	*Total*
386DX	$4	132	QFP	$1	$4	$9
486DX2	$12	168	PGA	$11	$12	$35
PowerPC 601	$53	304	QFP	$3	$21	$77
HP PA 7100	$73	504	PGA	$35	$16	$124
DEC Alpha	$149	431	PGA	$30	$23	$202
SuperSPARC	$272	293	PGA	$20	$34	$326
Pentium	$417	273	PGA	$19	$37	$473

什么是计算机性能？

先考虑民航客机的“性能”：续航能力、速度、载客量、运输能力？

Airplane	DC to Paris	Range (英里)	Speed (m.p.h.)	Passengers	Throughput (p x m.p.h.)	Cost
Boeing 747	6.5 hours	4150	610	470	286,700	???
BAC/Sud Concorde	3 hours	4000	1350	132	178,200	???
Douglas DC-8	7.3 hours	8720	544	146	79,424	???

若考虑制造成本，则性能评价更复杂

Time of Concorde vs. Boeing 747?

从DC到巴黎，Concorde 比Boeing 747快：3.5小时

速度上Concorde 比Boeing 747快：1350/610=2.2倍

Concorde的性能更好！

Throughput of Boeing 747 vs. Concorde?

运载能力上Boeing 747 比Concorde大：286,700/178,200=1.6倍

Boeing 747的性能更好！

不同的性能评价标准导致不同的结论！

计算机性能的基本评价指标

- 计算机也有两种不同的性能
  - Time to do the task
    - 响应时间（response time）
    - 执行时间（execution time）
    - 等待时间或时延（latency）
  - Tasks per day, hour, sec, ns. ..
    - 吞吐率（throughput）
    - 带宽（bandwidth）

不同应用场合用户关心的性能不同：

要求吞吐率高的场合，例如：

多媒体应用（音/视频播放要流畅）

要求响应时间短的场合：例如：

事务处理系统（存/取款的速度要快）

要求吞吐率高且响应时间短的场合：

ATM、文件服务器、Web服务器等

- 基本的性能评价标准是：CPU的执行时间

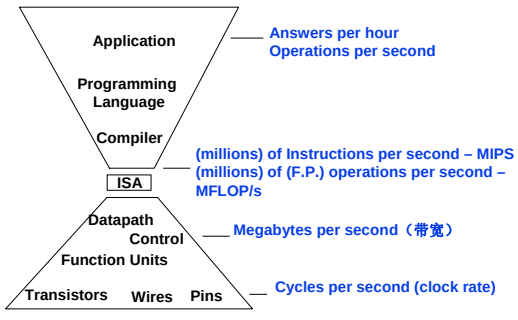
" X is n times faster than Y" means

$$\frac{\text{ExTime(Y)}}{\text{ExTime(X)}} = \frac{\text{Performance(X)}}{\text{Performance(Y)}}$$

相对性能用执行时间的倒数来表示！

SKIP

不同层次上对吞吐率性能的度量



BACK

计算机性能的测量

- 比较计算机的性能时，用执行时间来衡量
  - 完成同样工作量所需时间最短的那台计算机就是性能最好的
  - 处理器时间往往被多个程序共享使用，因此，用户感觉到的程序执行时间并不是程序真正的执行时间（从hello程序执行过程可知）
  - 通常把用户感觉到的响应时间分成：
    - CPU时间：指CPU真正花在程序执行上的时间。又包括两部分：
      - 用户CPU时间：用来运行用户代码的时间
      - 系统CPU时间：为了执行用户程序而需要运行操作系统程序的时间
    - 其他时间：指等待I/O操作完成或CPU花在其他用户程序的时间
  - 系统性能和CPU性能不等价，有一定的区别
    - 系统性能(System performance)：系统响应时间，与CPU外的其他部分也都有关系
    - CPU性能(CPU performance)：用户CPU时间
  - 本章主要讨论CPU性能，即：CPU真正用在用户程序执行上的时间

问题：用户CPU时间与系统响应时间哪个更长？

CPU执行时间的计算

- CPU execution time = CPU clock cycles/pgm ÷ clock rate
- = CPU clock cycles/pgm X clock cycle time
- = instrs/pgm X CPI X clock cycle time
- CPU clock cycles/pgm = Instrs/pgm X avg. clock cycles per instr.
- CPI = CPU clock cycles/pgm ÷ Instructions/pgm
- CPI 用来衡量以下各方面的综合结果
  - Instruction Set Architecture
  - Implementation of that architecture
  - program

Aspects of CPU Performance

$$\text{CPU time} = \frac{\text{Seconds}}{\text{Program}} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Cycle}}$$

	instr. count	CPI	clock rate
Program			
Compiler			
Instr. Set Arch.			
Organization			
Technology			

思考：三个因素与哪些方面有关？

Aspects of CPU Performance

$$\text{CPU time} = \frac{\text{Seconds}}{\text{Program}} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Cycle}}$$

	instr. count	CPI	clock rate
Program	X	X	
Compiler	X	(X)	
Instr. Set Arch.	X	X	
Organization		X	X
Technology			X

问题：ISA、计算机组织（Organization）、计算机实现技术（Technology）三者的关系是什么？

Architecture = Instruction Set Arch. + Organization

Computer Design

Instruction Set Design
- Machine Language
- Compiler View
- "Computer Architecture"
- "Instruction Set Processor"

"Building Architect"
"建筑设计师"
功能定义与设计

Computer Hardware Design
- Machine Implementation
- Logic Designer's View
- "Processor Architecture"
- " Computer Organization"

"Construction Engineer"
"建造工程师"
考虑用什么材料，如何布线等

例如，是否提供"乘法指令"是ISA设计要考虑的问题； 如何实现乘法指令（用专门的乘法器还是用一个加法器+移位器实现）是组成（Organization）考虑的问题； 如何布线、用什么材料、工艺设计等是计算机实现技术（Technology）考虑的问题。

Instruction Set Architecture

. . . the attributes of a [computing] system as seen by the programmer, i.e. the conceptual structure and functional behavior, as distinct from the organization of the data flows and controls the logic design, and the physical implementation.

Amdahl, Blaw, and Brooks, 1964

主要内容包括：

-- Organization of Programmable Storage（程序员可见存储的组织）
（如：寄存器的个数、名称、长度；内存单元的长度、主存地址长度等等）
-- Data Types & Data Structures（数据类型和结构）:
Encodings & Representations（编码和表示）
-- Instruction Formats（指令格式）
-- Instruction (or Operation Code) Set（操作码集合，即：指令功能）
-- Modes of Addressing and Accessing Data Items and Instructions
（寻址方式、数据 / 指令的存取方式）
-- Exceptional Conditions and handle（异常条件和处理）

Organization

Logic Designer's View → ISA Level / FUs & Interconnect

★ 为了实现ISA，该如何安排功能部件

★ 是关于寄存器传送级的描述

教材中除了“指令系统”和“数据的表示”外大都是“计算机组织”的内容

主要内容包括：

-- Capabilities & Performance Characteristics of Principal Functional Units
（主要功能部件的能力和工作特性）
(e.g., Registers, ALU, Shifters, Memory, Cache, etc.)

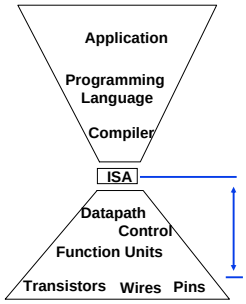
-- Ways in which these components are interconnected
（互连方式）

-- nature of information flows between components
（部件间的信息流动方式）

-- logic and means by which such information flow is controlled.
（部件间信息流动的控制逻辑和控制方法）

Organizational Trade-offs

CPU time = Instruction counts x CPI x Cycle Time



3 factors: How are they related?

- CPI的减少可能会增加时钟周期的长度
- 缩短时钟周期可能会增加指令的条数
- 改变IS以减少指令条数可能会使时钟周期变长
- 即使是在同一台机器上的同一个问题，最少指令条数的程序不一定执行的最快

Instruction Count (Mix)

CPI

Cycle Time

3 factors: Where are they?

因此，必须在各方面进行权衡！

如何计算CPI?

对于某一条特定的指令而言，其CPI是一个确定的值。但是，对于某一类指令、或一个程序、或一台机器而言，其CPI是一个平均值，表示该类指令或该程序或该机器的指令集中每条指令执行时平均需要多少时钟周期。

假定CPI i 和 C i分别为第 i类指令的CPI和指令条数，则程序的总时钟数为：

总时钟数 = $\sum_{i=1}^{n} CPI_i \times C_i$ 所以， CPU时间= 时钟周期 x $\sum_{i=1}^{n} CPI_i \times C_i$

假定CPIi、Fi是每类指令的CPI和在程序中的出现频率，则程序的综合CPI为：

CPI = $\sum_{i=1}^{n} CPI_i \times F_i$ where $F_i = \frac{C_i}{Instruction\_Count}$

假定已知CPU时间、时钟频率、总时钟数、指令条数，则程序的综合CPI为：

CPI = (CPU 时间×时钟频率) / 指令条数 = 总时钟周期数 / 指令条数

问题：指令的CPI、机器的CPI、程序的CPI各能反映哪方面的性能？
单靠CPI不能反映CPU的性能！为什么？如：单周期处理器CPI=1，但性能差！

Example1

Our favorite program runs in 10 sec on machine A, which has a 400MHz clock. We are trying to design a machine B with faster clock rate so as to reduce the execution time to 6 sec.

The increase of clock rate will affect the rest of the CPU design, causing B to require 1.2 times as many clock cycles as machine A for this program. What clock rate should be?

Answer:

CPU time A = CPU clock cycle A / clock rate A

==> CPU clock cycle A = 10 sec x 400 x 10^6

Clock rate B = CPU clock cycle B / CPU time B

= 1.2 x 400 x 10^6 / 6 = 800 MHz

机器B的频率是A的两倍，但机器B的速度并不是A的两倍！

Example2

Instruction type and Instruction frequencies in the execution of a program:

Op	Freq	Cycles
ALU	43%	1
Load	21%	2
Store	12%	2
Branch	24%	2

Question: What is the average CPI of the machine?

CPI = 1×43% + 2×21% + 2×12% +2×24% = 1.57

Example3

Suppose we have two implementations of the same instruction set. Machine A has a clock cycle time of 10 ns and an average CPI of 2.0 for some program.

Machine B has a clock cycle time of 20 ns and an average CPI of 1.2 for the same program.

Which is faster? And by how much?

相同IS对于同一个程序，其指令序列是一样的，当然条数相同！

Let n denote the number of instructions of the program

CPU time A = n x 2.0 x 10 = 20n（ns）

CPU time B = n x 1.2 x 20 = 24n（ns）

Machine A is 1.2 faster than B .

在此，又看到三个因素之间的相互影响。

Example 4

ISA has 3 kinds of instructions:

Instruction class	CPI for this instruction class
A	1
B	2
C	3

One program has 2 code sequences:

Code Sequence	Instruction counts for instruction class		
	A	B	C
1	2	1	2
2	4	1	1

Which code sequence has more instructions?
Which will be faster?
What is the CPI for each sequence?

虽然序列2的指令条数更多，但速度更快！

序列1 有5条指令； 序列2 有 6条.

序列1 需 2x1+1x2+2x3=10 cycles；序列2 需 4x1+1x2+1x3=9 cycles.

序列1 的 CPI=10/5=2； 序列2 的CPI=9/6=1.5

Marketing Metrics （产品宣称指标）

MIPS = Instruction Count / Time x10^6
= Clock Rate / CPI x 10^6
Million Instructions Per Seconds
因为每条指令执行时间不同，所以MIPS总是一个平均值。

- 不同机器的指令集不同
- 程序由不同的指令混合而成
- 指令使用的频度动态变化
- Peak MIPS: （不实用）

用MIPS数表示性能有没有局限?

所以MIPS数不能说明性能的好坏（用下页中的例子来说明）
MFLOPS = FP Operations / Time x10^6
Million Floating-point Operations Per Second

- 与机器相关性大
- 并不是程序中花时间的部分

用MFLOPS数表示性能也有局限！

Example: MIPS数不可靠！

Assume we build an optimizing compiler for the load/store machine. The compiler discards 50% of the ALU instructions.

1) What is the CPI ?

仅仅在软件上进行优化，没有涉及到任何硬件措施。

2) Assuming a 20 ns clock cycle time (50 MHz clock rate). What is the MIPS rating for optimized code versus unoptimized code? Does the MIPS rating agree with the rating of execution time?

Op	Freq	Cycle
ALU	43%	1
Load	21%	2
Store	12%	2
Branch	24%	2

Optimizing compiler

New Freq
21.5/ (21.5+21+12+24)=27%
21 / (21.5+21+12+24)=27%
12 / (21.5+21+12+24)=15%
24 / (21.5+21+12+24)= 31%

CPI	1.57	50M/1.57=31.8MIPS	1.73
MIPS	31.8	50M/1.73=28.9MIPS	28.9

结果：因为优化后减少了ALU指令（其他指令数没变），所以程序执行时间一定减少了，但优化后的MIPS数反而降低了。

选择性能评价程序（Benchmarks）

- 用基准程序来评测计算机的性能
  - 基准测试程序是专门用来进行性能评价的一组程序
  - 不同用户使用的计算机用不同的基准程序
  - 基准程序通过运行实际负载来反映计算机的性能
  - 最好的基准程序是用户实际使用的程序或典型的简单程序
- 基准程序的缺陷
  - 现象：基准程序的性能与某段短代码密切相关时，会被利用以得到不当的性能评测结果
  - 手段：硬件系统设计人员或编译器开发者针对这些代码片段进行特殊的优化，使得执行这段代码的速度非常快
    - 例1：Intel Pentium处理器运行SPECint时用了公司内部使用的特殊编译器，使其性能极高
    - 例2：矩阵乘法程序SPECmatrix300有99%的时间运行在一行语句上，有些厂商用特殊编译器优化该语句，使性能达VAX11/780的729.8倍！

用于性能估计的程序

- (Toy) Benchmarks（短小基准程序）
  - 程序短小容易编译，便于仿真或手工编译，因而可用于对新开发的机器进行性能评测.（因为新机器往往没有配编译器）。
  - 大小：10-100 line
  - 例：sieve, puzzle, quicksort
  - 缺陷：不是实际使用的程序，只用于新开发的计算机。
- Synthetic Benchmarks（综合基准程序）
  - 目的：试图用一个基准程序去涵盖一系列基准程序的特征。
  - 做法：使各种语句的执行频度与一系列基准程序中的频度一致。
  - 例： Whetstone(Algol60◦Fortran), Dhrystone(Ada◦C)
  - 缺陷：不是实际使用的程序，故可特殊优化使评测结果优，但实际并不如此。
- Kernels（核心程序段）
  - 实际程序中的耗时较多的关键片断
  - 通常用在科学计算领域测试某个特殊功能的性能
  - 例： Livermore loops(21 loops), Linpack(linear algebra)
- Real programs（频繁使用的实际程序）
  - e.g., gcc, spice

Successful Benchmark: SPEC

- 1988年，5家公司（ Sun, MIPS, HP, Apollo, DEC ）联合提出了Systems Performance Evaluation Committee (SPEC)
- SPEC给出了一组标准的测试程序、标准输入和测试报告。它们是一些实际的程序，包括 OS calls、 I/O等。
- 版本 89：10 programs = 4 for integer + 6 for FP, 用每个程序的执行时间求出一个综合性能指标
- 版本92：SPECInt92 (6 integer programs) and SPECfp92 (14 floating point programs)
  - 整数和浮点数单独提供衡量指标：SPECInt92和SPECfp92
  - 增加 SPECbase: 禁止使用任何与程序有关的编译优化开关
- 版本95： 8 int + 10fp
- 较新版本： include SPEC HPC96, SPEC JVM98, SPEC WEB99, SPEC OMP2001. SPEC CPU2000，See http://www.spec.org for more details
  - “benchmarks useful for 3 years”
  - Base machine is changed from VAX-11/780 to Sun SPARC 10/40

如何给出综合评价结果?

问题：如果用一组基准程序在不同的机器上测出了运行时间，那么如何综合评价机器的性能呢？

先看一个例子：

Program 1: 1 sec on machine A, 10 sec on machine B

Program 2: 1000 sec on A，100 sec on B

What are your conclusions?

- A is 10 times faster than B for program1.
- B is 10 times faster than A for Program2.

这个结论无法比较A和B的好坏

必须用一个综合的值来表示！

Total execution time是一个综合度量值，可以据此得出结论：

B is 1001/110=9.1 times faster than A

实际上，必须考虑每个程序在作业中的使用频度，即加权平均

综合性能评价的方法

- 可以用以下两种平均值来评价：
  - Arithmetic mean(算术平均)：求和后除n
  - Geometric mean(几何平均)：求积后开根号n
- 根据算术平均执行时间能得到总平均执行时间
- 根据几何平均执行时间不能得到程序总的执行时间
- 执行时间的规格化（测试机器相对于参考机器的性能）：
  - time on reference machine ÷ time on measured machine
- 平均规格化执行时间不能用算术平均来计算，而应该用几何平均
  - program A going from 2 seconds to 1 second as important as program B going from 2000 seconds to 1000 seconds.

（算术平均值不能反映这一点！）

综上所述，算术平均和几何平均各有长处，可灵活使用！

Impact of Means on SPECmark89 for IBM 550

	Ratio to VAX:		Time:		Weighted Time:	
Program	*Before*	*After*	*Before*	*After*	*Before*	*After*
gcc	30	29	49	51	8.91	9.22
espresso	35	34	65	67	7.64	7.86
spice	47	47	510	510	5.69	5.69
doduc	46	49	41	38	5.81	5.45
nasa7	78	144	258	140	3.43	1.86
li	34	34	183	183	7.86	7.86
eqntott	40	40	28	28	6.68	6.68
matrix300	78	730	58	6	3.43	0.37
fpppp	90	87	34	35	2.97	3.07
tomcatv	133	138	20	19	2.01	1.94
Mean	54	72	124	108	54.42	49.9
	Geometric		*Arithmetic*		*Weighted Arith.*	
	Ratio	1.33	Ratio	1.16	Ratio	1.09

注："After"表示加了特殊编译开关后的结果。

好的评价标准应该对特殊处理不敏感！

该表反映了不同的均值计算法得到的结论可能不同。

第二讲小结

- 性能的定义：一般用程序的响应时间或系统的吞吐率表示机器或系统整体性能。
- CPU性能的测量（用户程序的CPU执行时间）：
  - 一般把程序的响应时间划分成CPU时间和等待时间，CPU时间又分成用户CPU时间和系统CPU时间。
  - 因为操作系统对自己所花费的时间进行测量时，不十分准确，所以，对CPU性能的测算一般通过测算用户CPU时间来进行。
- 各种性能指标之间的关系：
  - CPU执行时间=CPU时钟周期数 x 时钟周期
  - 时钟周期和时钟频率互为倒数
  - CPU时钟周期数 = 程序指令数 x 每条指令的平均时钟周期数CPI
- MIPS数在有些情况下不能说明问题，不具有可比性！
- 性能评价程序的选择：
  - 采用一组基准测试程序进行综合（算术（加权）平均/几何平均）评测。
  - 有些制造商会针对评测程序中频繁出现的语句采用专门的编译器，使评测程序的运行效率大幅提高。因此有时基准评测程序也不能说明问题。
- 对于某种特定的指令集体系结构，提高计算机性能的主要途径有：
  - 提高时钟频率（第七章 流水线）
  - 优化处理器中数据通路的结构以降低CPI（第六章 单周期/多周期处理器）
  - 采用编译优化措施来减少指令条数或降低指令复杂度（第五章 指令系统）

本章作业

第一讲作业

1. 冯·诺依曼结构计算机由哪几部分组成？各部分的功能是什么？采用什么工作方式？
2. 计算机系统的层次结构如何划分？计算机系统的用户可分哪几类？每类用户工作在哪个层次？
3. 假定你的朋友不太懂计算机，请用简单通俗的语言给你的朋友介绍计算机系统是如何工作的。要求写一页纸左右。
4. 你对计算机系统的哪些部分最熟悉，哪些部分最不熟悉？最想进一步了解细节的是哪些部分的内容？

本章作业

第二讲作业

- 第5、6、7、8、9、10题

写在作业本上，可以用英文答题，也可以用中文答题

要求：书写工整、清楚，答题要完整

下星期二（2月24号）交作业！