计算机组成与系统结构

计算机科学与技术学院

王也

wangye2020@hrbeu.edu.cn

课程学习目的

- In depth understanding of modern computer architecture, fundamental issues, engineering tradeoffs
- How to design a computer
- Understanding how a system works and why it performs as it does

Reference Books

- 《Computer Organization and Design》hardware and software interface, Patterson and Hennessy,3nd Edition, Morgan Kaufmann Pub.
  - 英文版：机械工业出版社影印
  - 中文版：机械工业出版社 郑纬民 等 翻译

- 《Computer Systems A Programmer's Perspective》Randal E. Bryant Devid O'Hallaron, Prentice-Hall Internation Inc. 2003 中文译本：《深入理解计算机系统》中国电力出版社 龚奕利 等，2004

教材后面给出了参考文献，特别建议看国外网站

教学安排

- 理论部分：28学时，多媒体授课，每周留有作业，讲授1-8章，第9章自学；

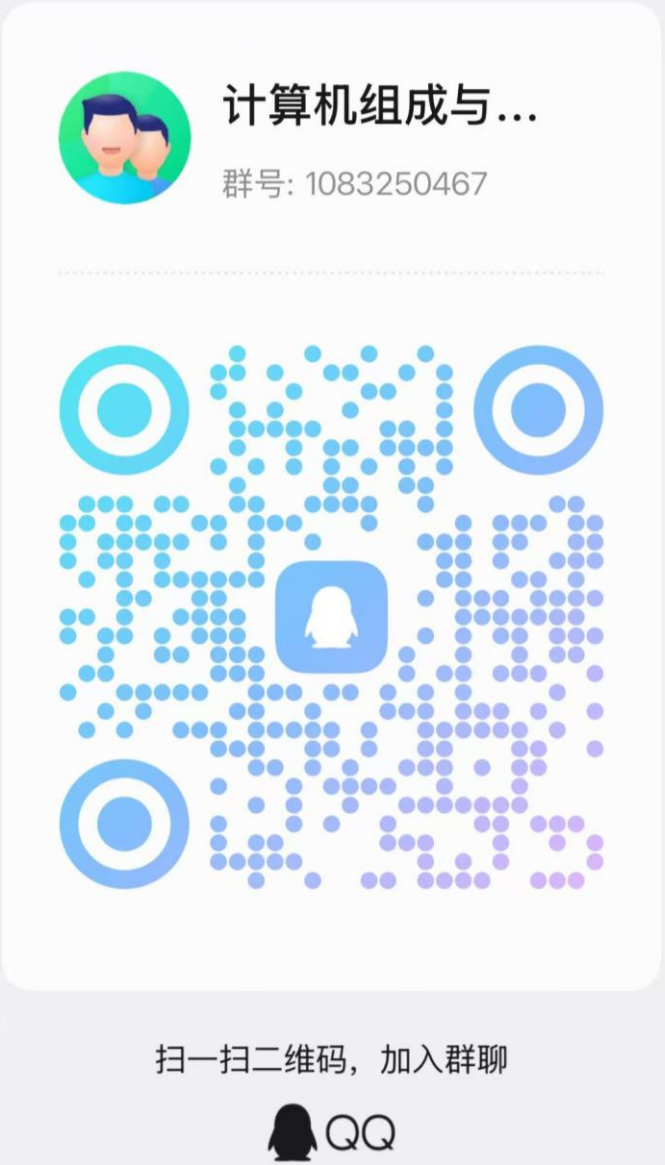
- 实践部分：8学时，两个实验，需要提交实验报告电子版，具体如下：

实验项目名称	实验内容	关联的课程章节	实验要求
流水线性能分析与优化实验	通过流水线模拟器，分析流水线中的数据相关、结构相关和控制相关，并对流水线的性能进行优化	六	1. 一人一题； 2. 能够分析各类相关并优化； 3. 现场演示； 4. 撰写实验报告，有模板。
指令高速缓冲存储器实验（选做）	通过模拟器分析指令高速缓冲存储器的操作，研究未命中和命中的处理方式，并分析高速缓存替换策略	七	1. 一人一题； 2. 能够针对某个问题分析并优化； 3. 正确统计和分析实验结果；并录屏讲解； 4. 撰写实验报告，模板同上。

评价机制

- 平时成绩作业：10分；
- 实验部分：每个实验15分，共30分；
- 期末闭卷考试：60分。

交流平台



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

- 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、高速算术和逻辑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

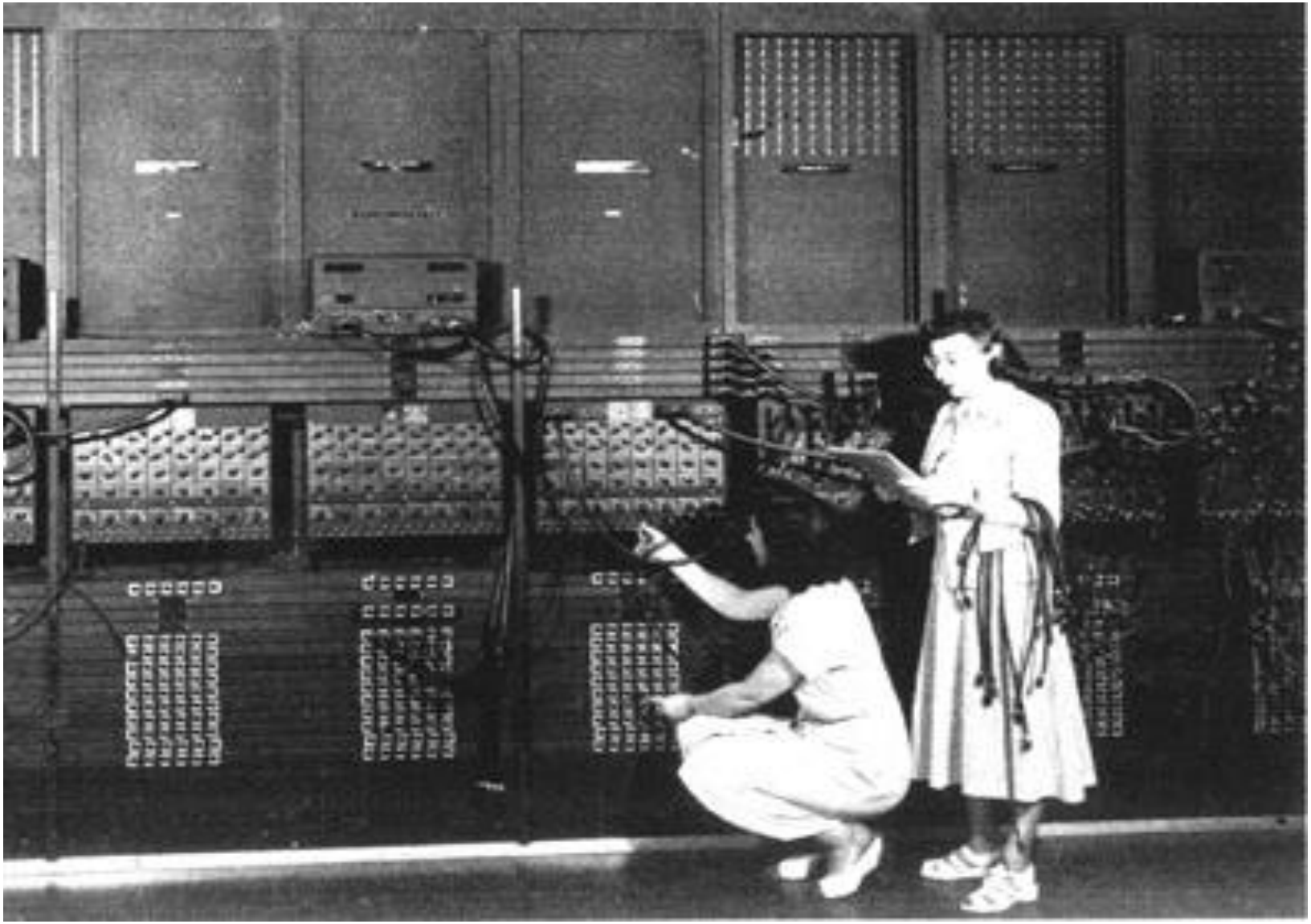
- 第一代：真空管（电子管Vacuum Tube ）1946～57年
  - 46年诞生第1台电子计算机 ENIAC
    - 体积大，重30吨，有18000多个真空管，5000次加法/s
    - 十进制表示/运算，存储器由20个累加器组成，每个累加器存10位十进制数，每一位由10个真空管表示。
    - 采用手动编程，通过设置开关和插拔电缆来实现。
  - 冯·诺依曼机（Von Neumann Machine）
    - 45年冯·诺依曼提出“存储程序(Stored-program)”思想，并于46年开始设计“存储程序”计算机。
    - “存储程序”思想：

      将事先编好的程序和原始数据送入主存中，然后启动执行。计算机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。

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

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

ENIAC----Non von Neumann Model



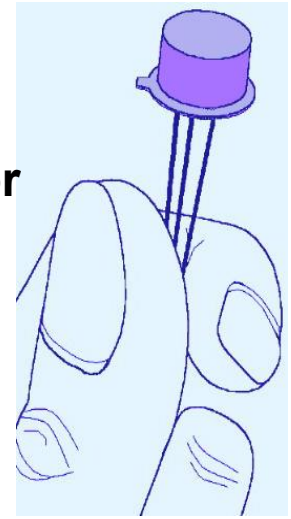
冯·诺依曼结构的主要思想

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；
   - 控制器用来指令译码并送出操作控制信号；
   - 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（IC）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？

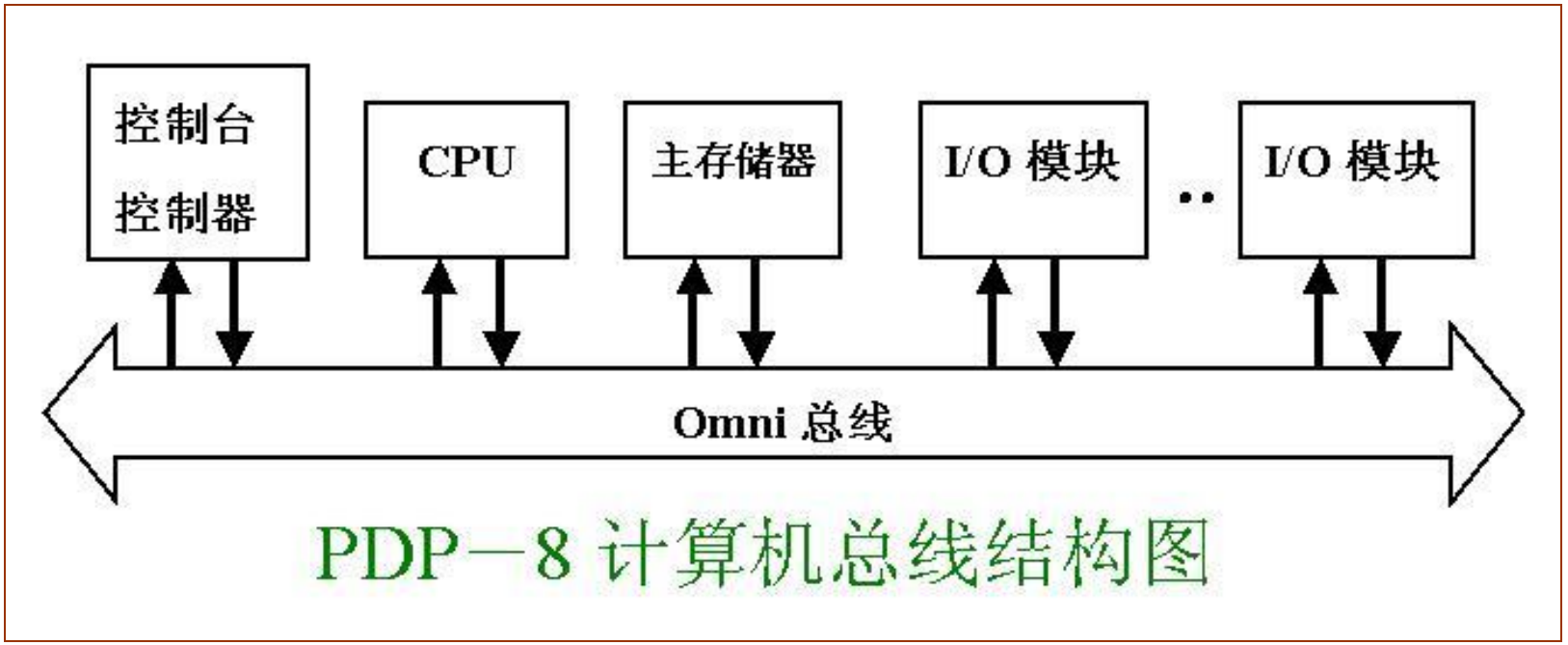
问题2：实现“系列机”的关键是什么？

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



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

- （第四代：LSI/VLSI/ULSI 1972～至今）
  - 微处理器和半导体存储器技术发展迅猛，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

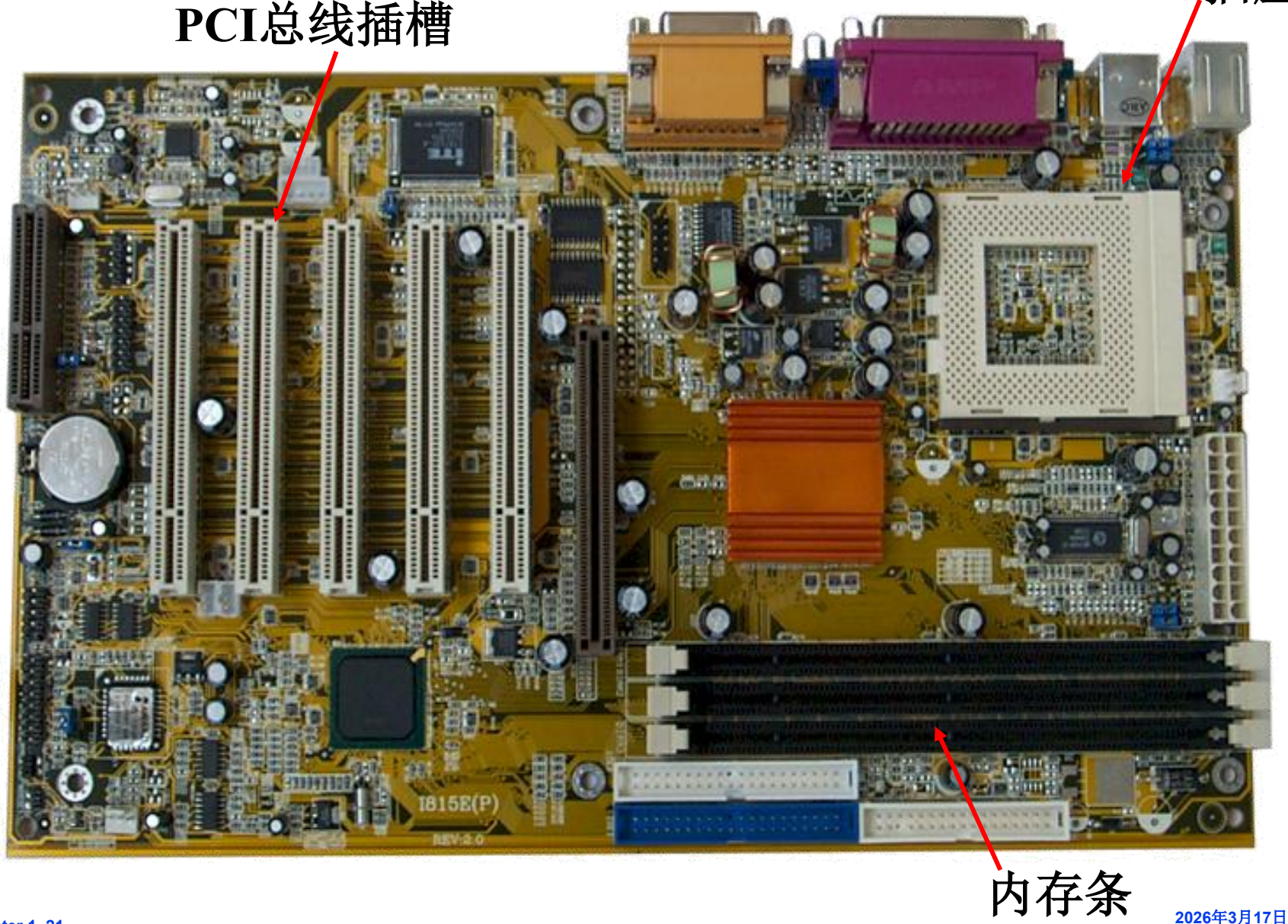
    - 微处理器芯片密度不断增加，使CPU中所有元件放在一块芯片上成为可能。71年开发出第一个微处理器芯片4004。
  - 特点：共享存储器，分布式存储器及大规模并行处理系统

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）

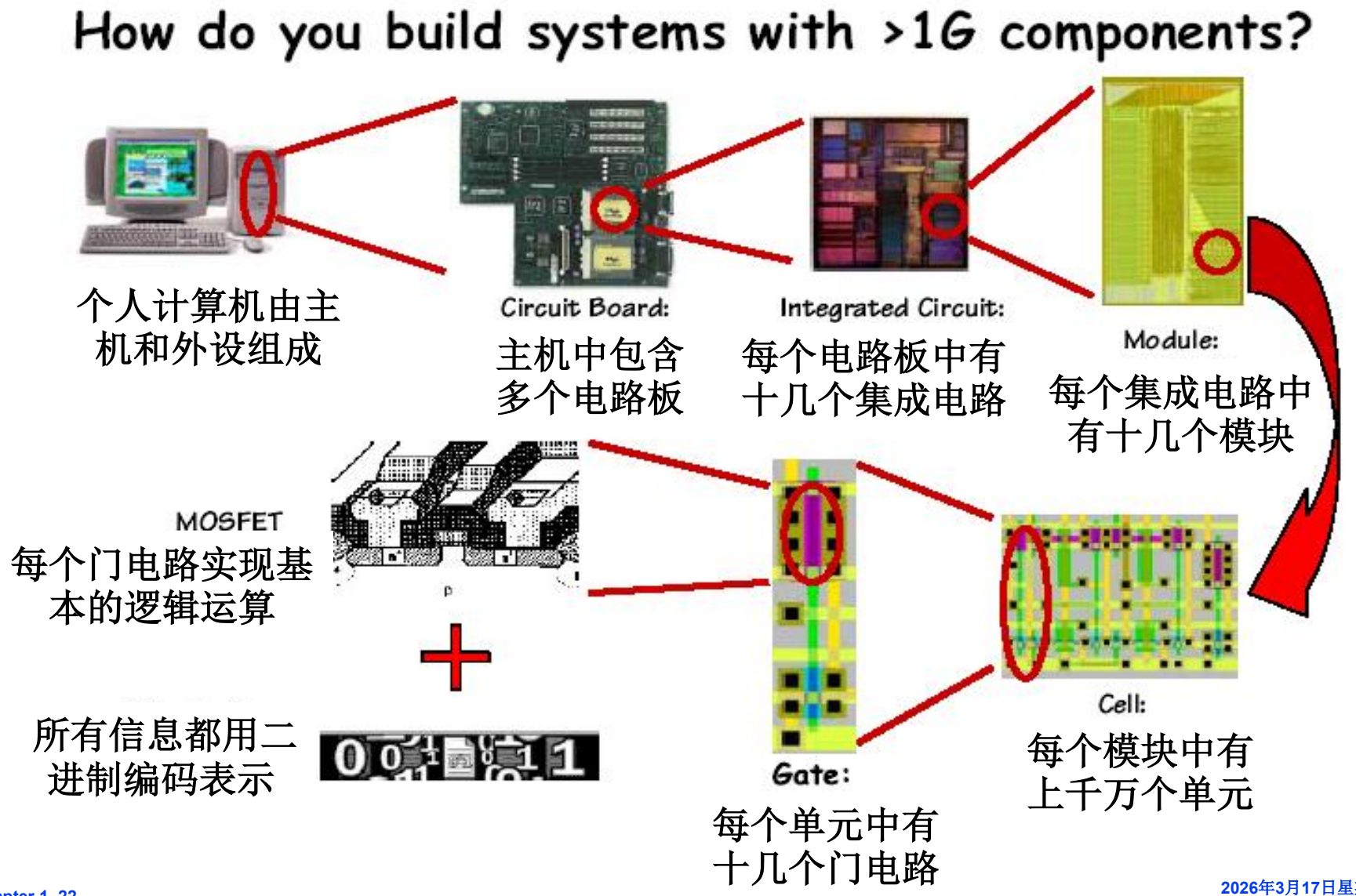
计算机硬件：打开计算机来看看

c

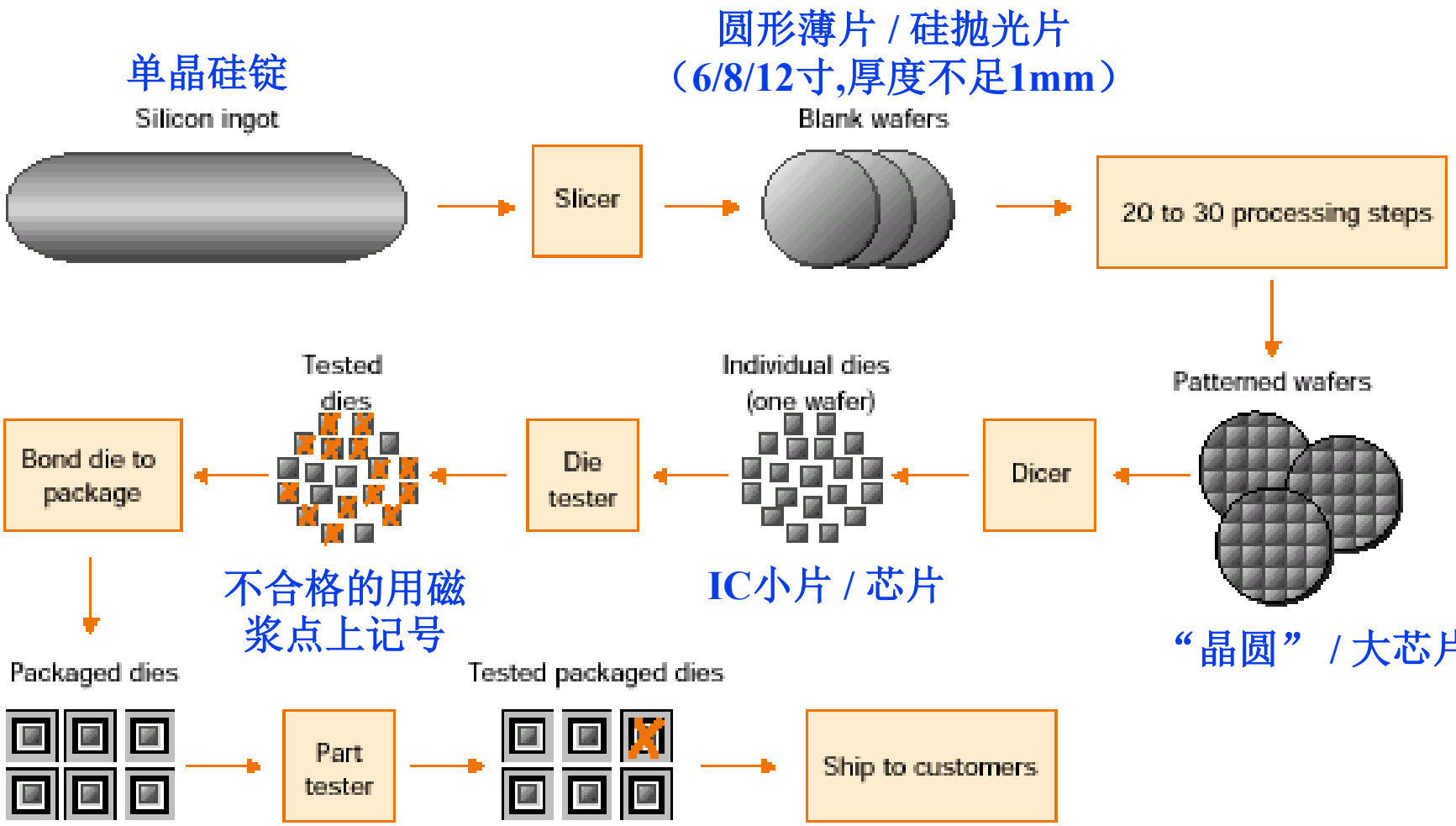
PC主板



解剖一台计算机


How do you build systems with >1G components?
个人计算机由主机和外设组成
Circuit Board:
主机中包含多个电路板
Integrated Circuit:
每个电路板中有十几个集成电路
Module:
每个集成电路中有十几个模块
MOSFET
每个门电路实现基本的逻辑运算
p
所有信息都用二进制编码表示
0 0 1 1
Gate:
每个单元中有十几个门电路
Cell:
每个模块中有上千万个单元

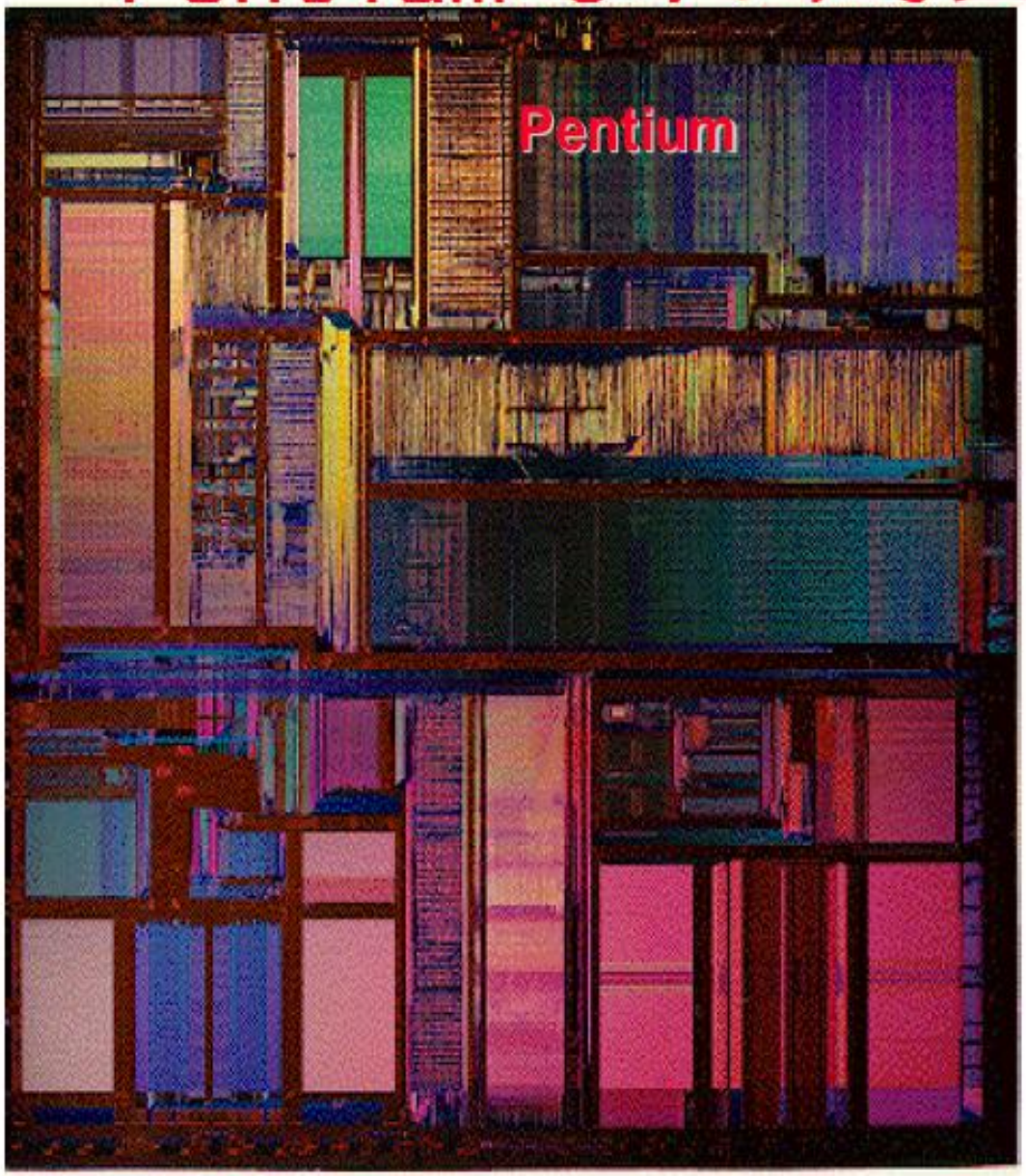
Integrated Circuits manufacturing process



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 mm^2

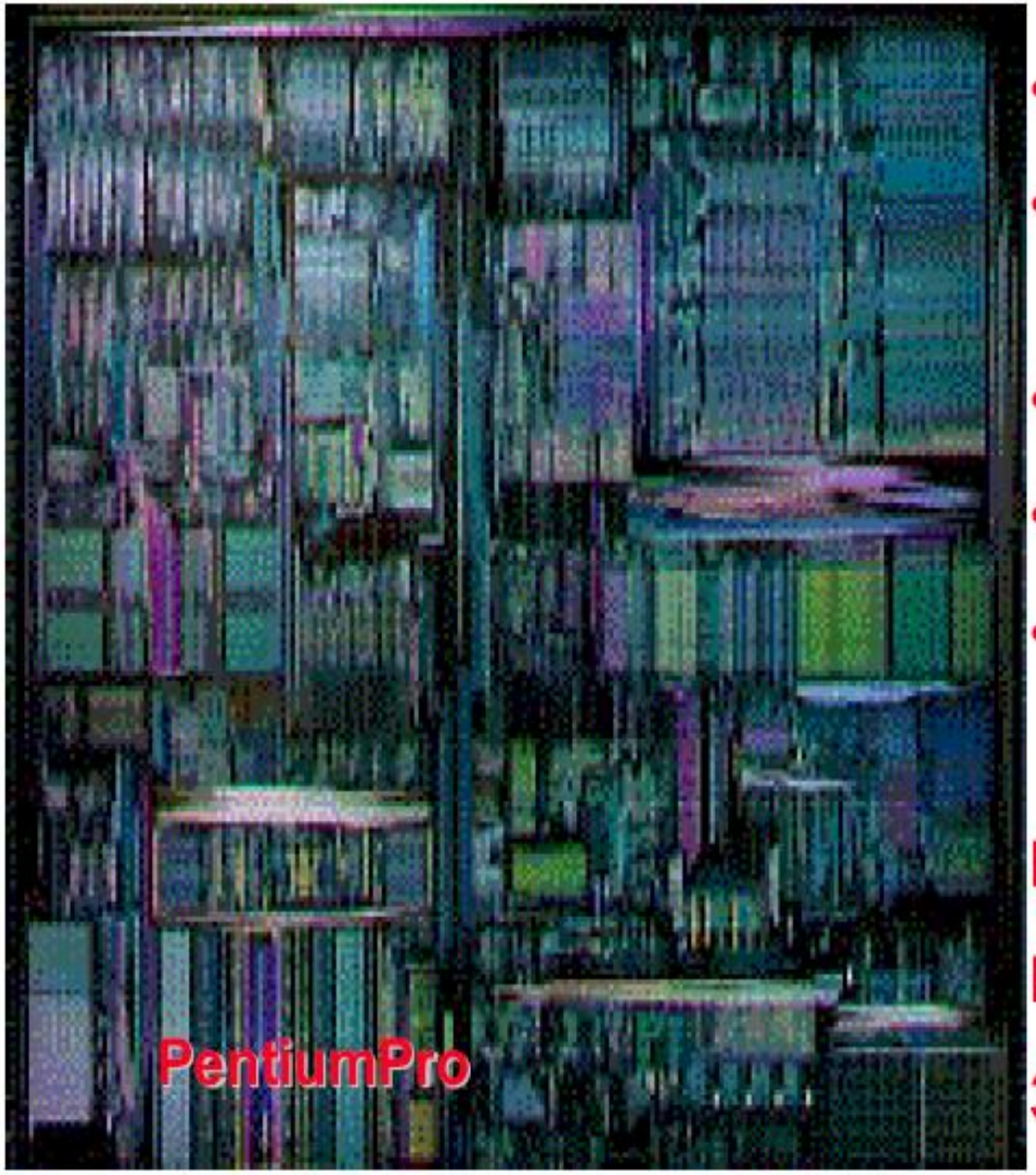
直径8 inch(200mm)的
Wafer最多可做 196个Die

≈ 3,300,000 Transistors

Cache: ≈1M Transistors

296 Pins

PentiumPro芯片内的主要功能块



- Die Area: 306 mm^2
- 直径8 inch(200mm)的 Wafer最多可做 78个Die
- ≈ 5,500,000 Transistors
- Cache: ≈1M Transistors
- External Cache:

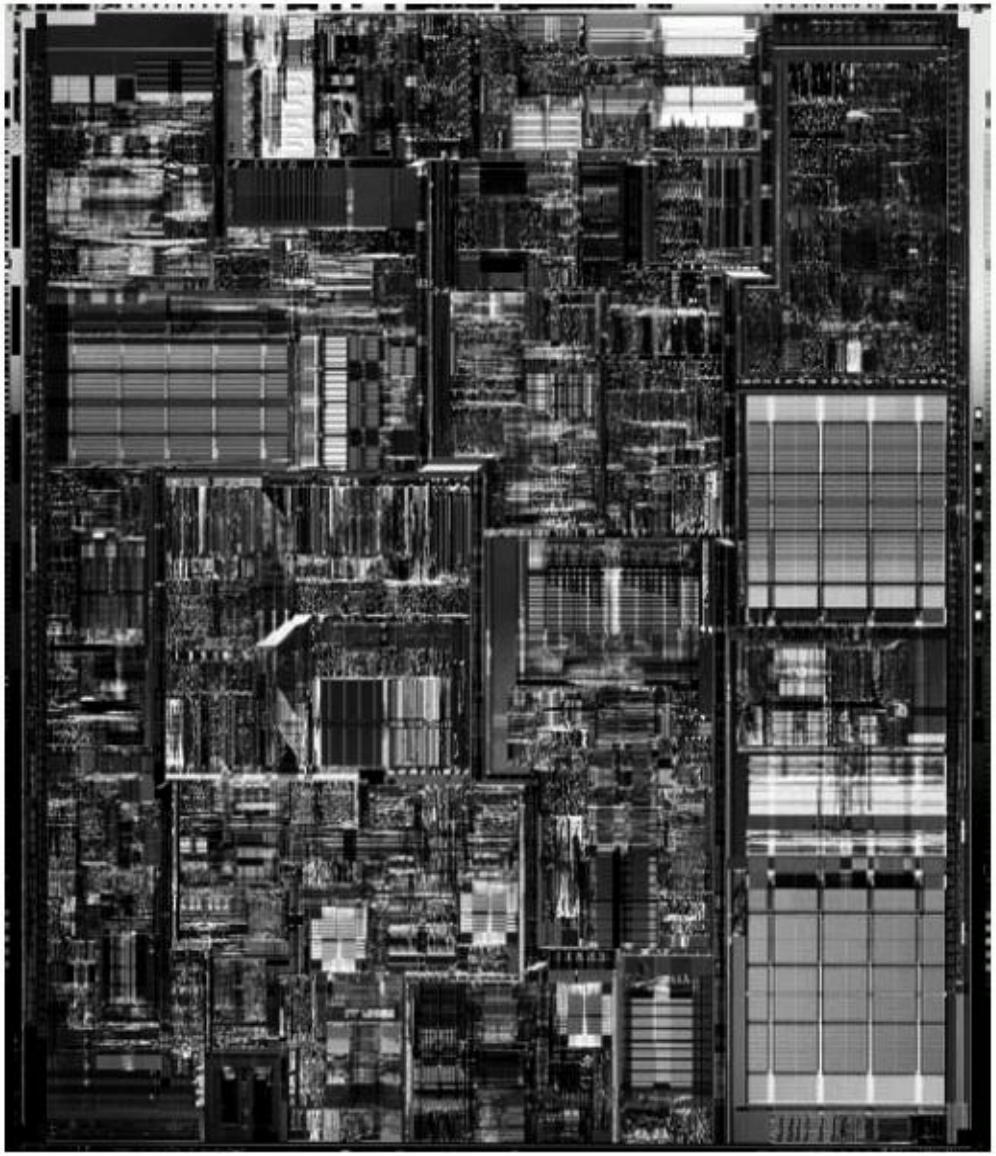
  31M Transistors

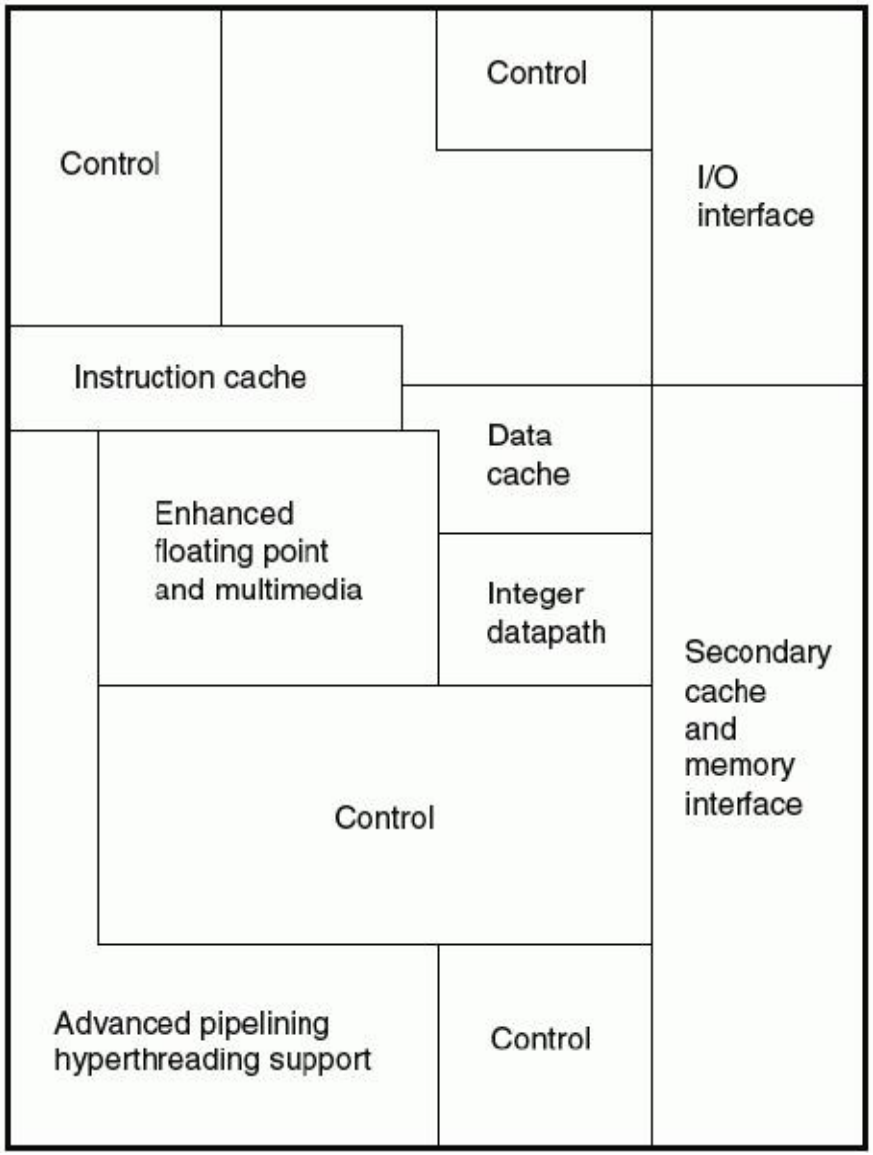
PentiumPro Package =

PentiumPro+ExternalCache

387 Pins

Pentium4处理器内部布局





计算机系统中的层次概念

- 计算机体系结构概念的提出
  - **1964年C. M. Amdahl在介绍IBM 360系统时提出经典的计算机体系结构（Computer Architecture）定义。**

计算机先驱奖

计算机系统中的层次概念

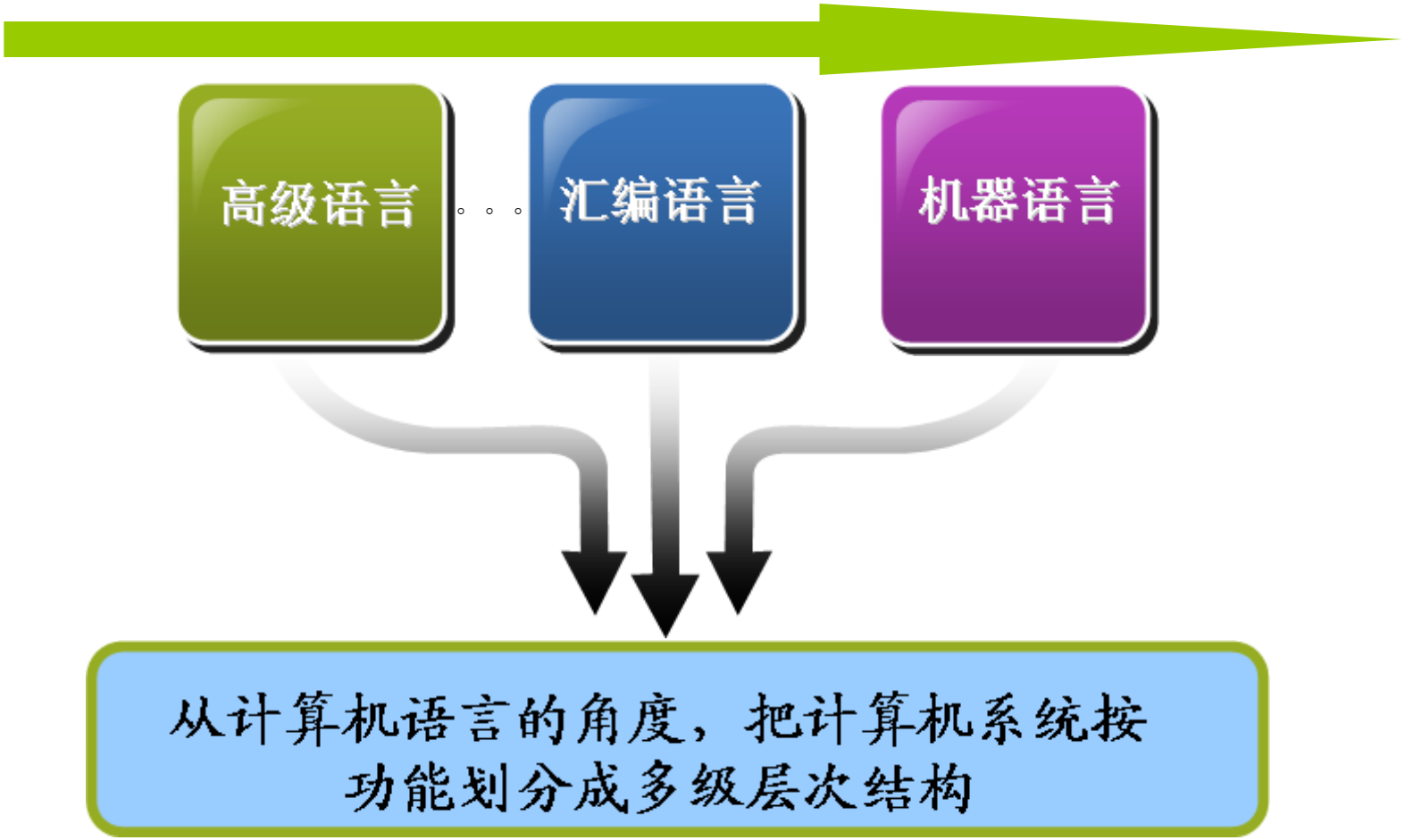
- 计算机体系结构概念的提出
  - 1964年C. M. Amdahl在介绍IBM 360系统时提出经典的计算机体系结构（Computer Architecture）定义。

程序员所看到的计算机属性，即程序员编写出的能够在机器上正确运行的程序所需要了解的概念性结构和功能特性。

…the attributes of a [computer] system as seen by the programmer, i.e. the conceptual structure and function behaviour, as distinct from the organization of the data flows and controls， the logic design, and the physical implementation.
Amdahl, Blaaw and Brooks (1964)

计算机系统中的层次概念

从计算机语言的使用者角度分析



计算机系统中的层次概念

第 5 级：应用语言虚拟机

第 4 级：高级语言虚拟机

第 3 级：汇编语言虚拟机

第 2 级：操作系统虚拟机

软件

存储在控制存储器中完成指令功能的程

可以被机器直接执行

硬件或固件

第 0 级：微程序机器级

操作系统虚拟机

(1) 是否需要

(2) 如果需要，应放在哪个层次上

计算机系统中的层次概念

- 各机器级的实现

翻译

先用转换程序把N+1级上的程序整个变换成N级上的等效程序，然后再在N级上实现的技术，在执行过程中N+1级程序不再被访问。

解释

在低级机器级上用它的一串语句或指令仿真高级机器级上的一条语句或指令的功能，是通过对高级的机器级语言程序中的每条语句或指令逐条解释来实现的技术。

解释执行比翻译花的时间多，但存储空间占用较少。

计算机系统中的层次概念

- 虚拟机
  - 由软件实现的机器。虚拟机的功能也不一定全由软件实现，也可以是固件或硬件

    如：操作系统的某些命令可由微程序直接解释实现

- 选择什么样的软、硬件比例，是系统结构研究的核心内容之一。
  - 软件实现的功能可以用硬件或固件实现，硬件实现的功能可以由软件模拟完成。

透明性

系统结构是对计算机系统中各级界面的定义及其上下的功能分配

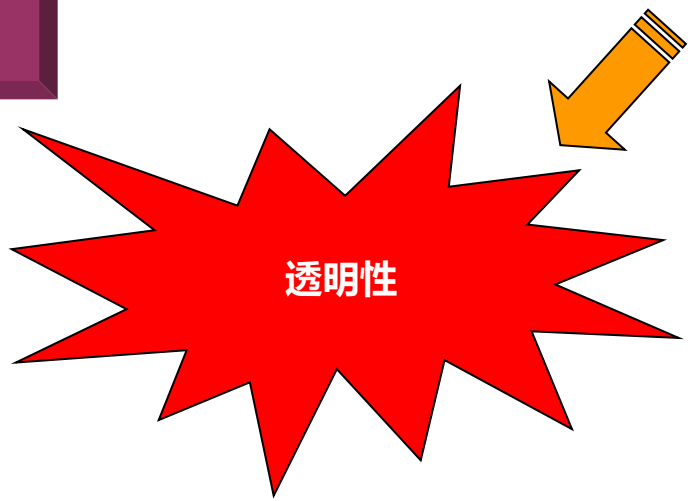
从计算机系统的层次结构

每级都有自己的用户、实现方法、指令系统和完成的功能

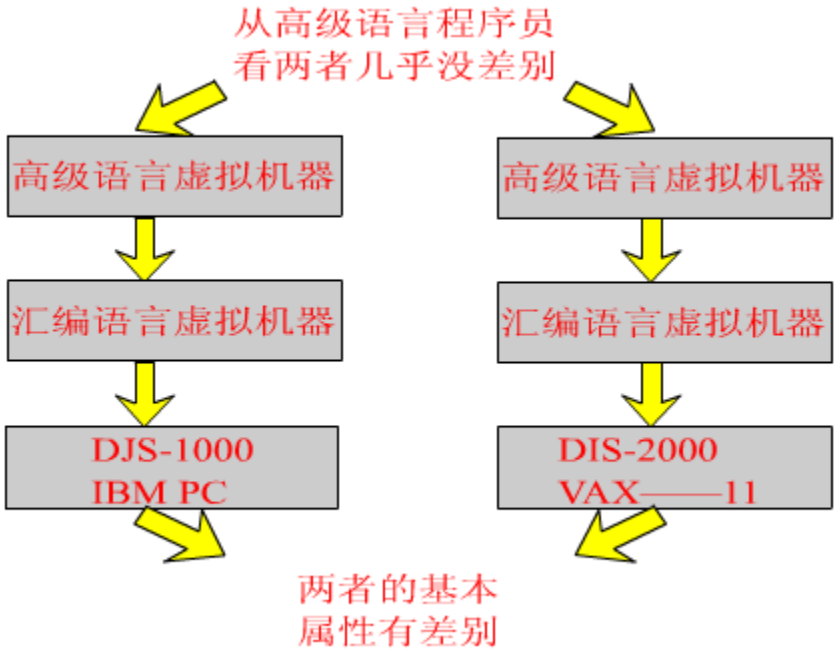
从不同级看到的计算机属性是不同的

客观存在的属性从某种角度看是看不到的

计算机系统的层次结构具有的特征



透明性



⑩指令系统、寻址方式、寄存器组织、I/O设备的连接方式等

透明性

- 透明性定义
  - 在计算机技术中，对这种本来是存在的事物或属性，但从某种角度看又好象不存在的概念称为**透明性**
- 优点
  - **可以不用管理它，简化设计**
- 缺点
  - **看不到而无法加以控制，会带来不利**

透明性

- 例题
  - 从汇编语言程序员的角度来看，以下哪些是透明的?

指令地址寄存器	否
指令缓冲器	是
时标发生器	是
条件码寄存器（状态寄存器）	否
乘法器	是
主存地址寄存器	是
磁盘外设	否
先行进位链	是
移位器	是
通用寄存器	否
中断字寄存器	否

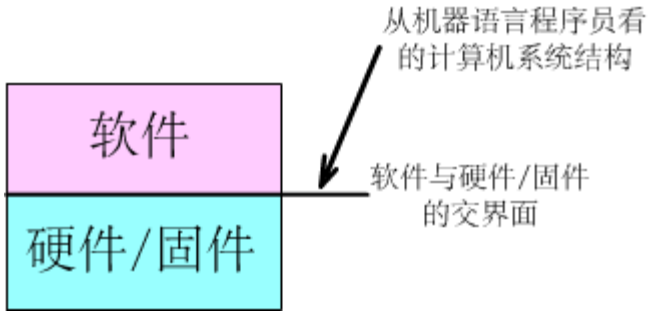
计算机系统结构、组成与实现

狭义的计算机系统结构定义

- **由机器语言、汇编语言设计者，或编译程序设计者所看到的计算系统的属性,是硬件子系统的概念性结构和功能特性。**

狭义的计算机系统结构概念的实质

计算机系统中软硬件界面的确定，其界面之上的是软件的功能，界面之下的是硬件和固件的功能。

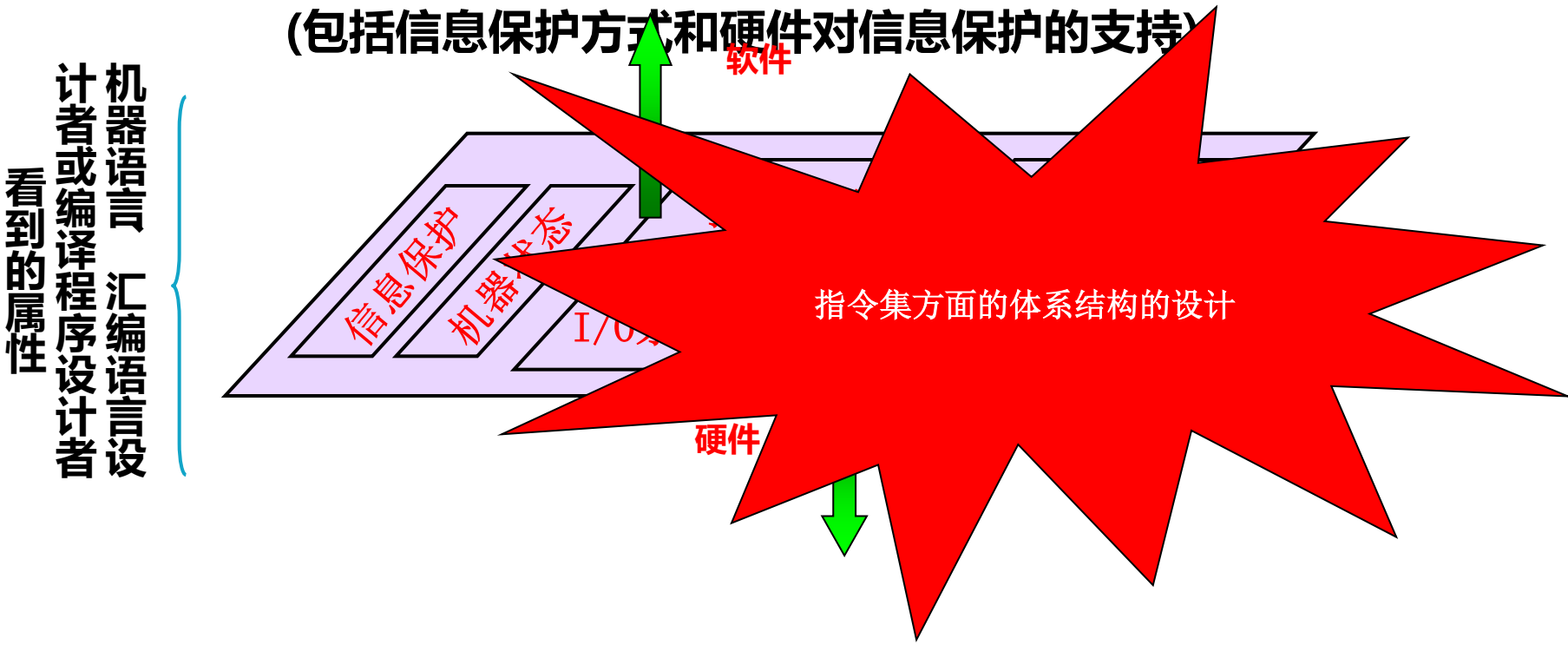


计算机系统结构、组成与实现

- 狭义的计算机系统结构概念包含内容
  - 对于通用寄存器机器，这些属性主要是指：
    - 数据表示
      (硬件能直接辩认和处理的数据类型)
    - 指令集
      (包括机器指令的操作类型和格式、指令间的排序和控制机构等)
    - 寻址方式
      (包括最小可寻址单位、寻址种类、地址计算等)
    - 寄存器组织
      (包括各种寄存器的设置、数量、字长和使用方式等)
    - 存储系统
      (最大可编址空间、最小编址单位、编址方式和主存容量等)
    - 中断系统
      (中断分类与分级、中断处理程序功能及入口地址等)
    - 机器工作状态的定义和切换
      (如管态和目态等)

计算机系统结构、组成与实现

- I/O结构
  (包括I/O联结方式、处理机/存储器与I/O设备间数据传送的方式和格式以及I/O操作的状态等)
- 信息保护
  (包括信息保护方式和硬件对信息保护的支持)



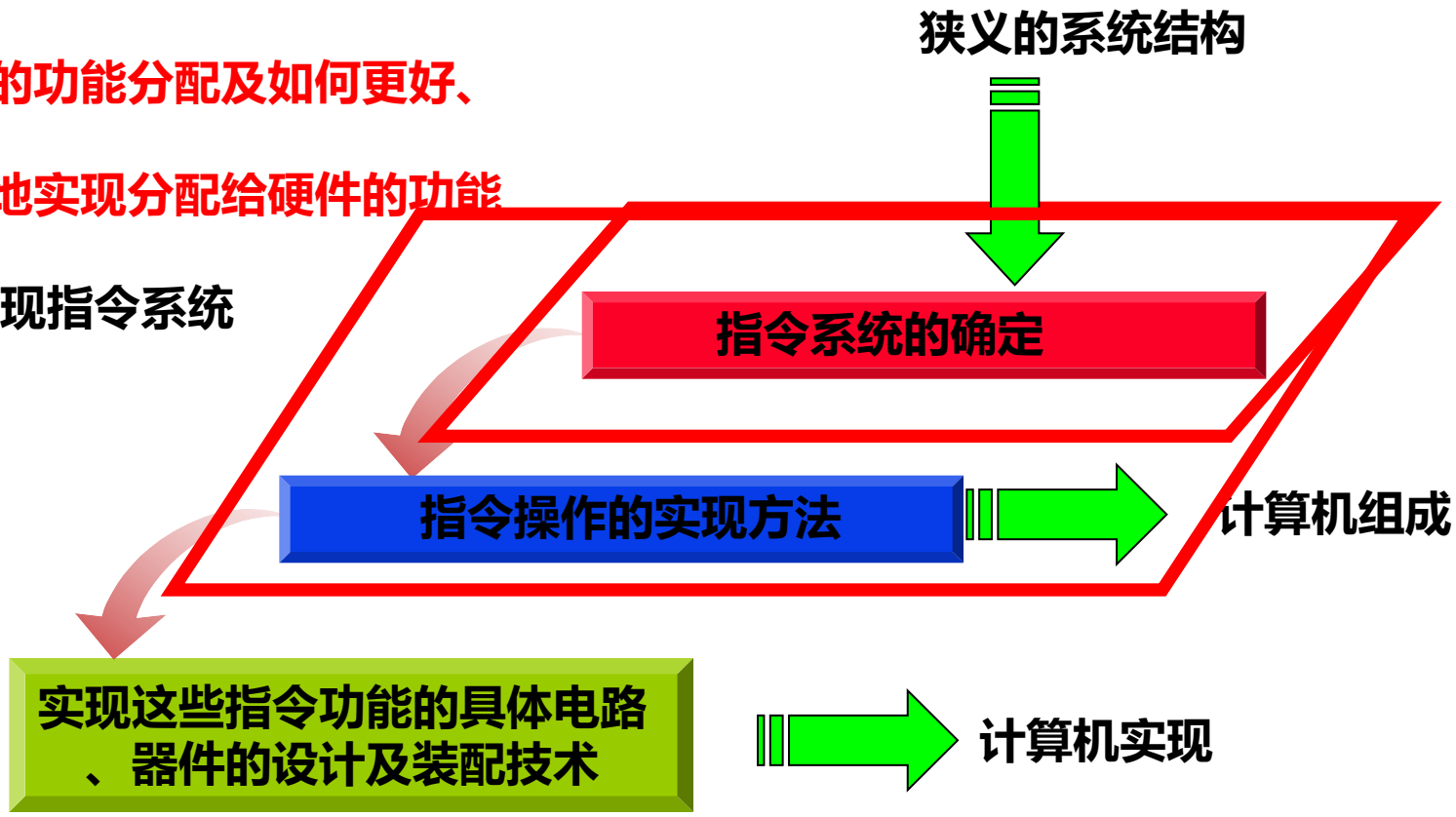
计算机系统结构、组成与实现

广义的计算机体系结构定义?

软硬件的功能分配及如何更好、

更合理地实现分配给硬件的功能

- 例：实现指令系统



计算机系统结构、组成与实现

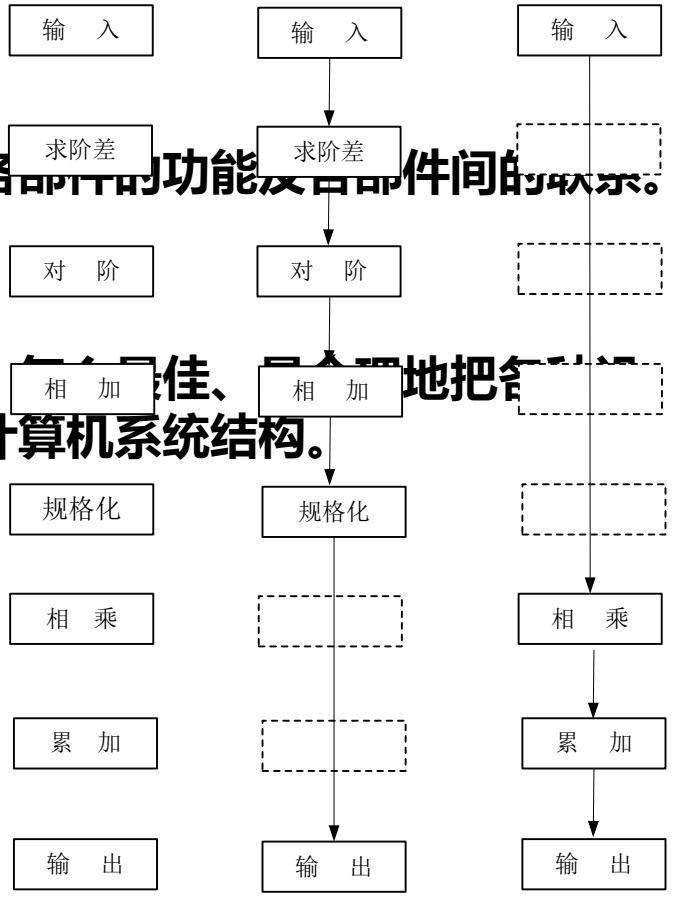
- 计算机组成
  - 是计算机系统结构的逻辑实现，包括机器内部的数据流和控制流的组成以及逻辑设计等。

着眼点：

- 机器内部各事件的排序方式与控制机构，各部件的功能及各部件间的联系。

欲解决：

- 在合理或满足要求的性能和价格的条件下，[illegible]佳、[illegible]地把各种设备和部件组织成计算机，以实现所确定的计算机系统结构。



计算机系统结构、组成与实现

- 计算机组成包括
  - **数据通路宽度**：数据总线上一次并行传送的信息位数
  - **专用功能部件的设置**：选用哪些专用部件。如乘法专用部件、浮点运算部件、字符处理部件、地址运算部件等。每种专用部件的数量等等。这些都取决于所需达到的机器速度、专用部件的使用频率及能承受的价格。
  - **各种操作对功能部件的共享程度**：尽管许多操作在逻辑上互不相关，只能分时使用，但共享程度高，速度相应也下降。可以设置多个部件降低共享程度，用提高操作并行度来提高速度，但价格也相应提高。

计算机系统结构、组成与实现

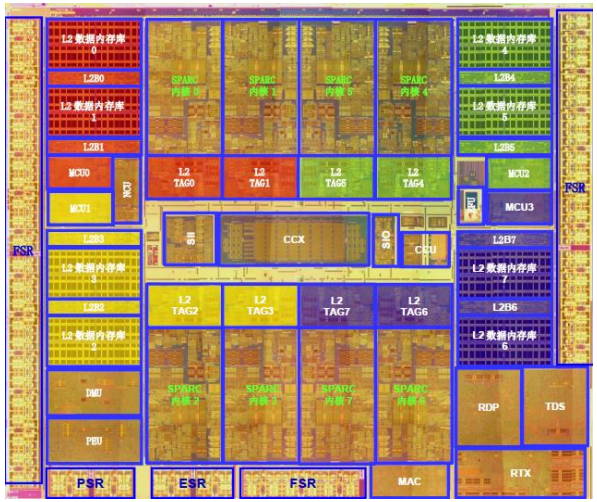
- 计算机组成包括
  - **功能部件的并行性**：用顺序串行，还是用重叠、流水或分布式控制和处理。
  - **控制机构的组成方式**：由硬件还是微程序控制，是单机处理还是多机处理或功能分布处理。
  - **缓冲和排队技术**：部件间如何设置及设置多大容量的缓冲器来协调它们的速度差；是用随机、先进先出、先进后出、优先级，还是循环方式来安排事件处理的顺序。
  - **预测技术**：为优化性能，用什么原则预测未来的行为。
  - **可靠性技术**：用何种冗余和容错技术来提高可靠性。

计算机系统结构、组成与实现

- 计算机实现
  - 是计算机组成的物理实现

着眼点

  - 器件技术（起主导作用）、微组装技术



计算机系统结构、组成与实现

- 计算机实现
  - 处理器、主存的物理结构
  - 器件的集成度和速度
  - 信号传输
  - 器件、模块、插件、底板的划分与连接
  - 涉及的专用器件
  - 电源、冷却、微组装技术、整机装配技术，等等

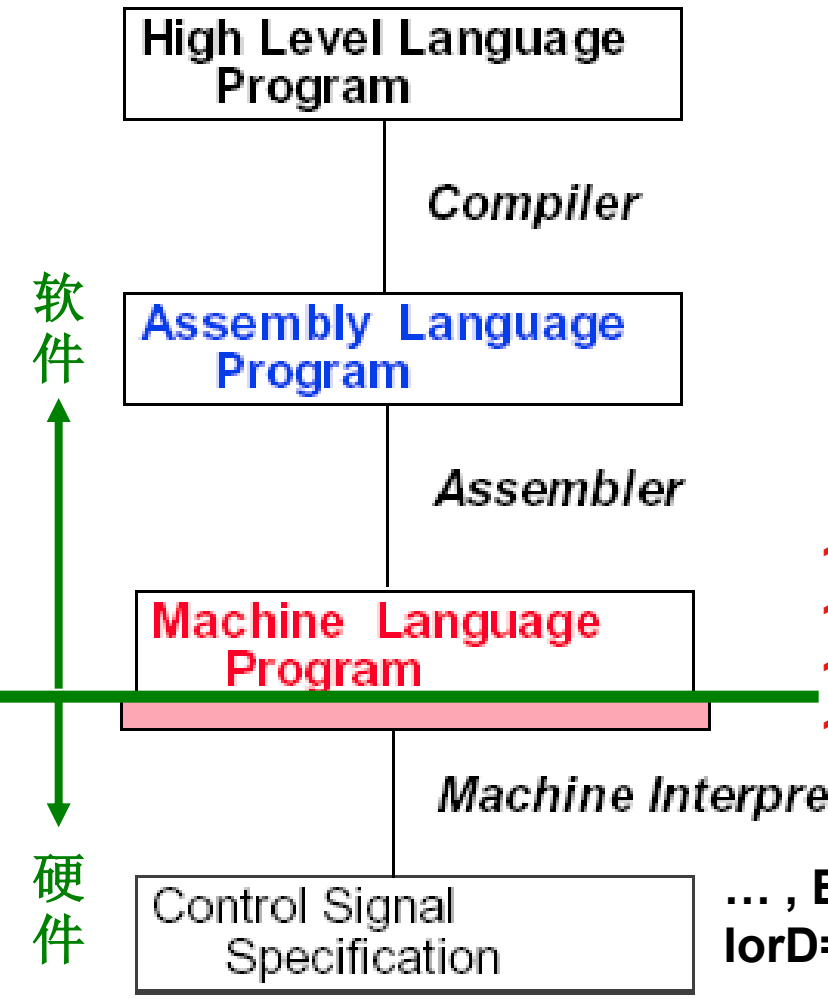
是数字电路等课程主要研究的内容

计算机系统结构、组成与实现

- 以指令系统、乘法指令系统及主存系统为例，说明系统结构、组成和实现各自所研究的内容

	系统结构	组成	实现
指令系统	指令系统的确定	指令的实现，如取指，译码、求AE取操作数等设计与排序	实现各指令的电路设计、器件的设计及装配
乘法指令	确定是否有乘法指令	指令是由加法器连加实现还是用专门的乘法器实现	加法器或乘法器的类型、集成度、数量、价格、组装技术
主存系统	主存的容量与编址方式	主存的速度Tm、单体多字或多体多字等的逻辑结构	器件的选定、电路的设计、微组装技术（双极型、MOS型）

Hardware/Software Interface



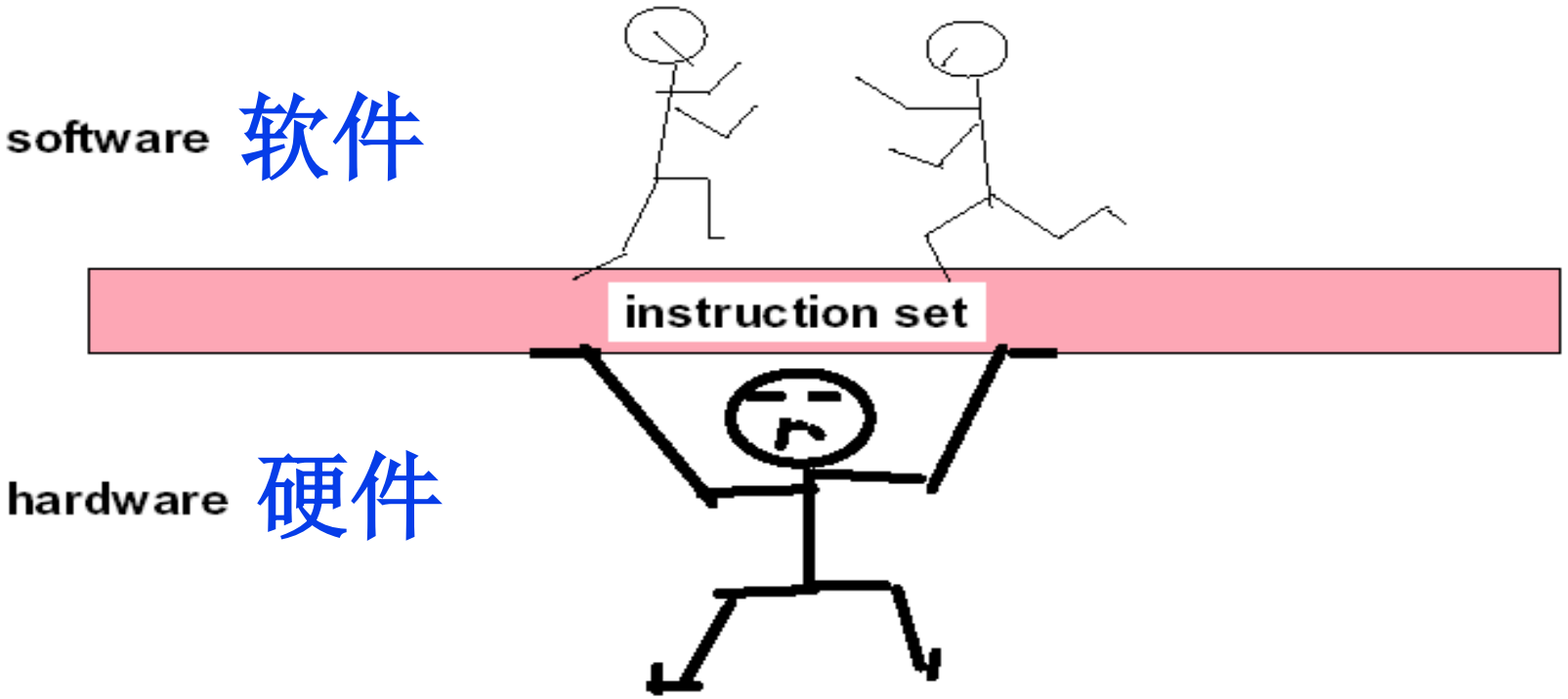

```
temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;
```

```
lw $15, 0($2)
lw $16, 4($2)
sw $16, 0($2)
sw $15, 4($2)
```

```
1000 1100 0100 1111 0000 0000 0000 0000
1000 1100 0101 0000 0000 0000 0000 0100
1010 1100 0101 0000 0000 0000 0000 0000
1010 1100 0100 1111 0000 0000 0000 0100
```

```
… , EXTop=1,ALUSelA=1,ALUSelB=11,ALUop=add,
IorD=1,Read,MemtoReg=1,RegWr=1,.....
```

Hardware/Software Interface（界面）



软件和硬件的界面： ISA（Instruction Set Architecture）

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。

软件

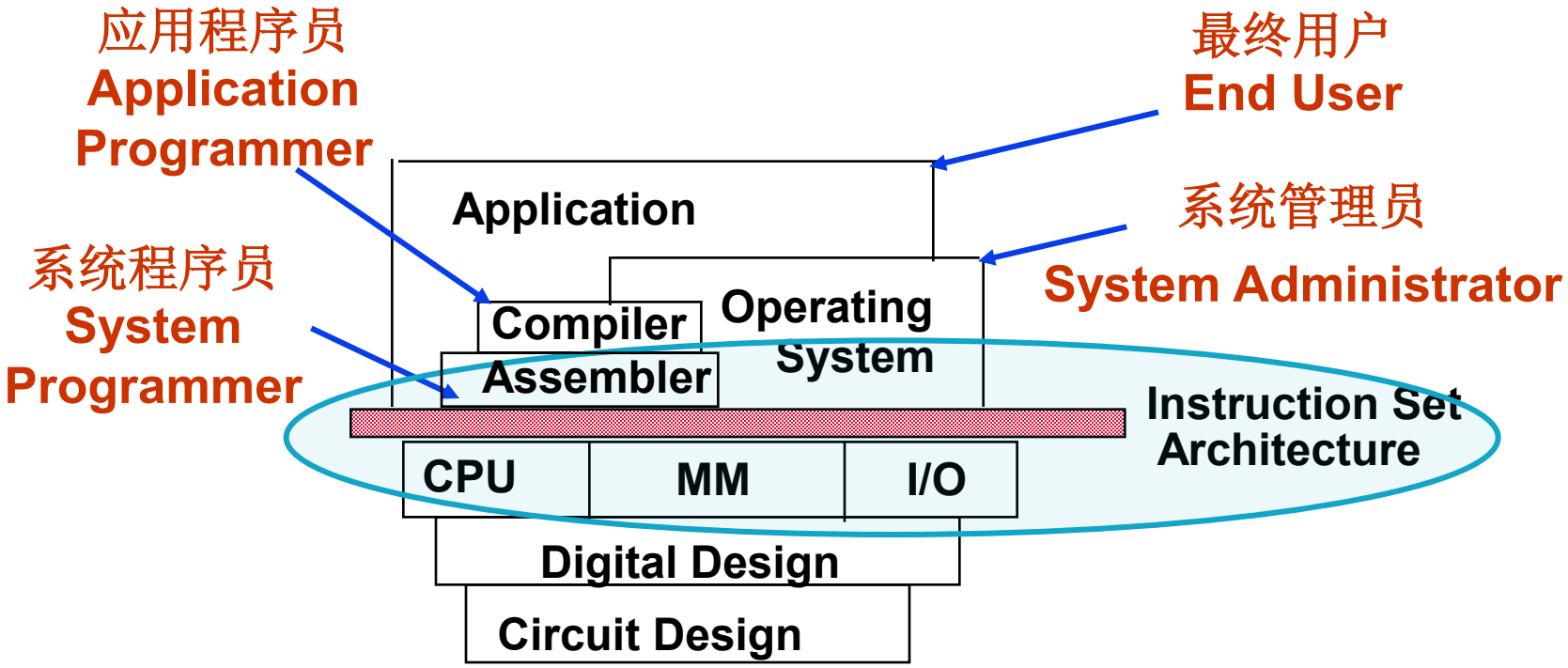
- System software(系统软件) - 简化编程过程，使硬件资源被有效利用
  - 操作系统（Operating System）：硬件资源管理，用户接口
  - 语言处理系统：翻译程序+ Linker, Debug, Loader, etc …
    - 翻译程序(Translator)有三类：

      汇编程序(Assembler)：汇编语言源程序→机器语言目标程序

      编译程序(Complier)：高级语言源程序→汇编/机器语言目标程序

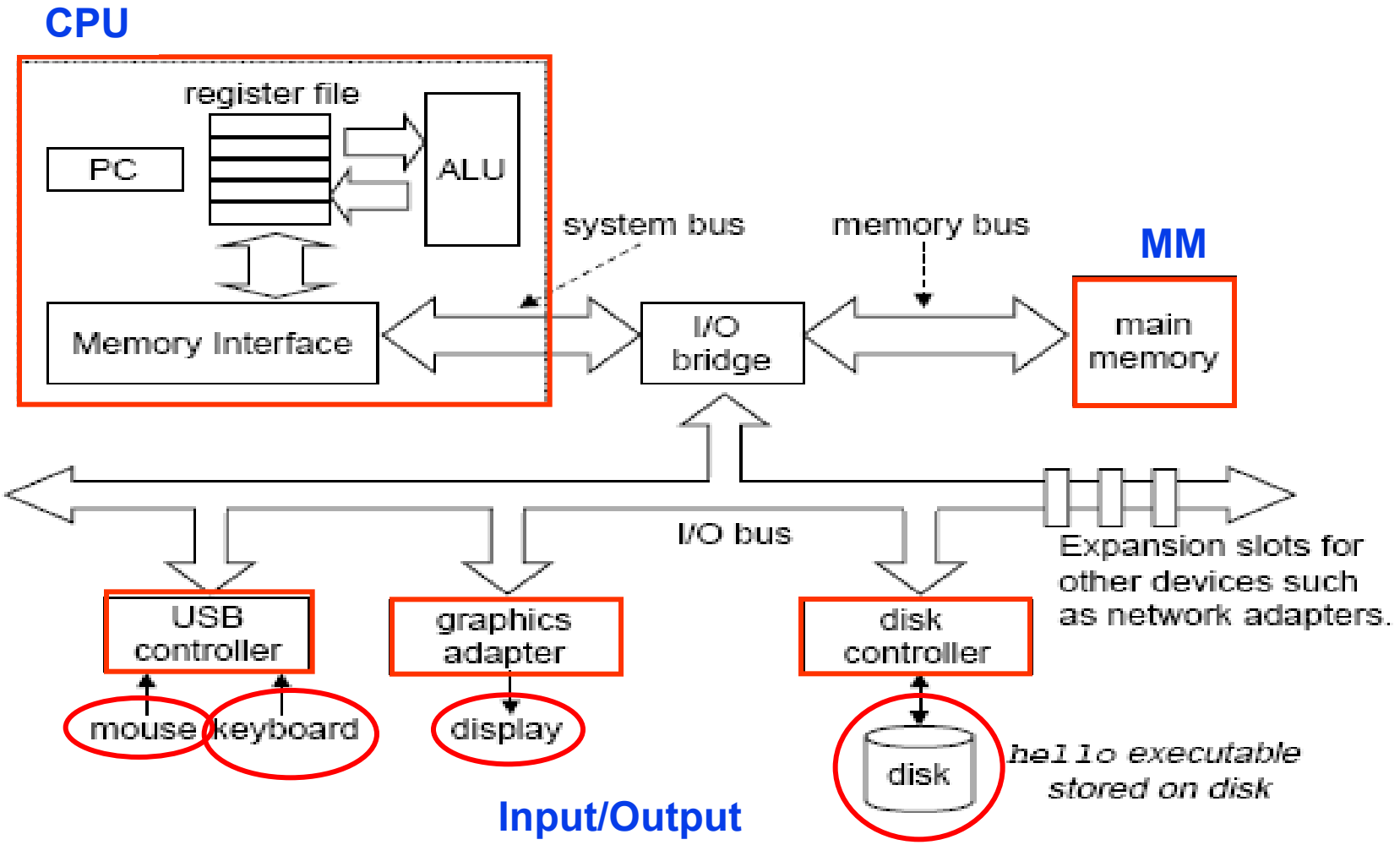
      解释程序(Interpreter)：将高级语言语句逐条翻译成机器指令并立即执行,不生成目标文件。
  - 其他实用程序: 如：磁盘碎片整理程序、备份程序等
- Application software(应用软件) - 解决具体应用问题/完成具体应用任务
  - 各类媒体处理程序：Word/ Image/ Graphics/…
  - 管理信息系统 (MIS)
  - Game，…

计算机系统层次的不同用户



- 上图给出的是计算机系统的层次结构
  指令系统（即ISA）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
- 中间阴影部分就是本课程主要内容，处于最核心的部分！

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

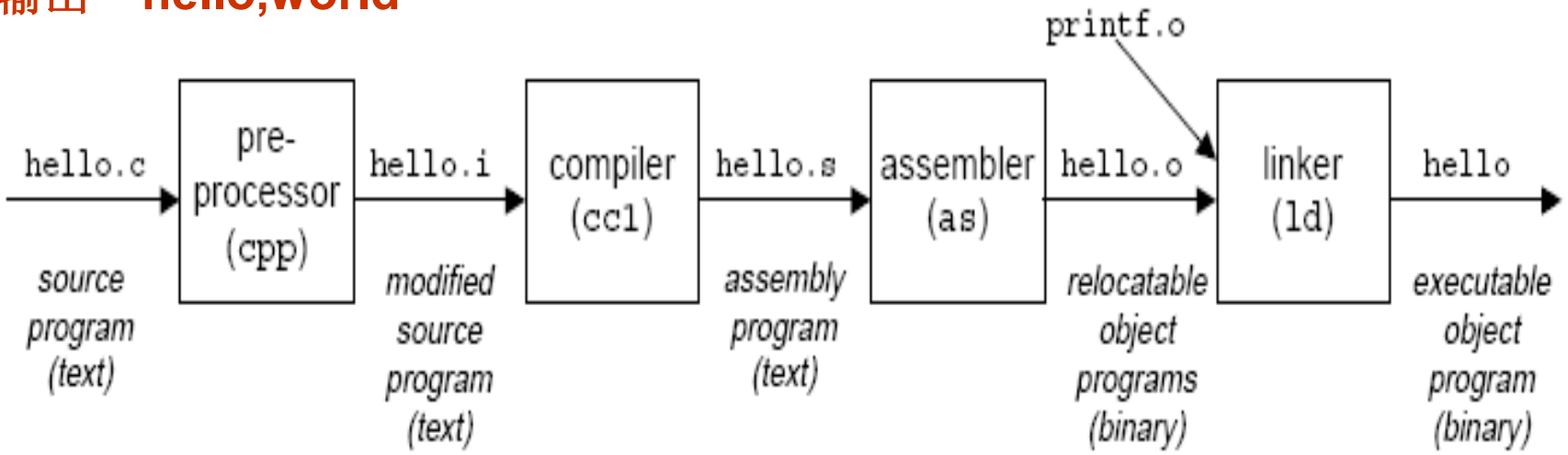
```
1 #include <stdio.h>
2
3 int main()
4 {
5 printf("hello, world\n");
6 }
```

程序的功能是：

输出“hello,world”

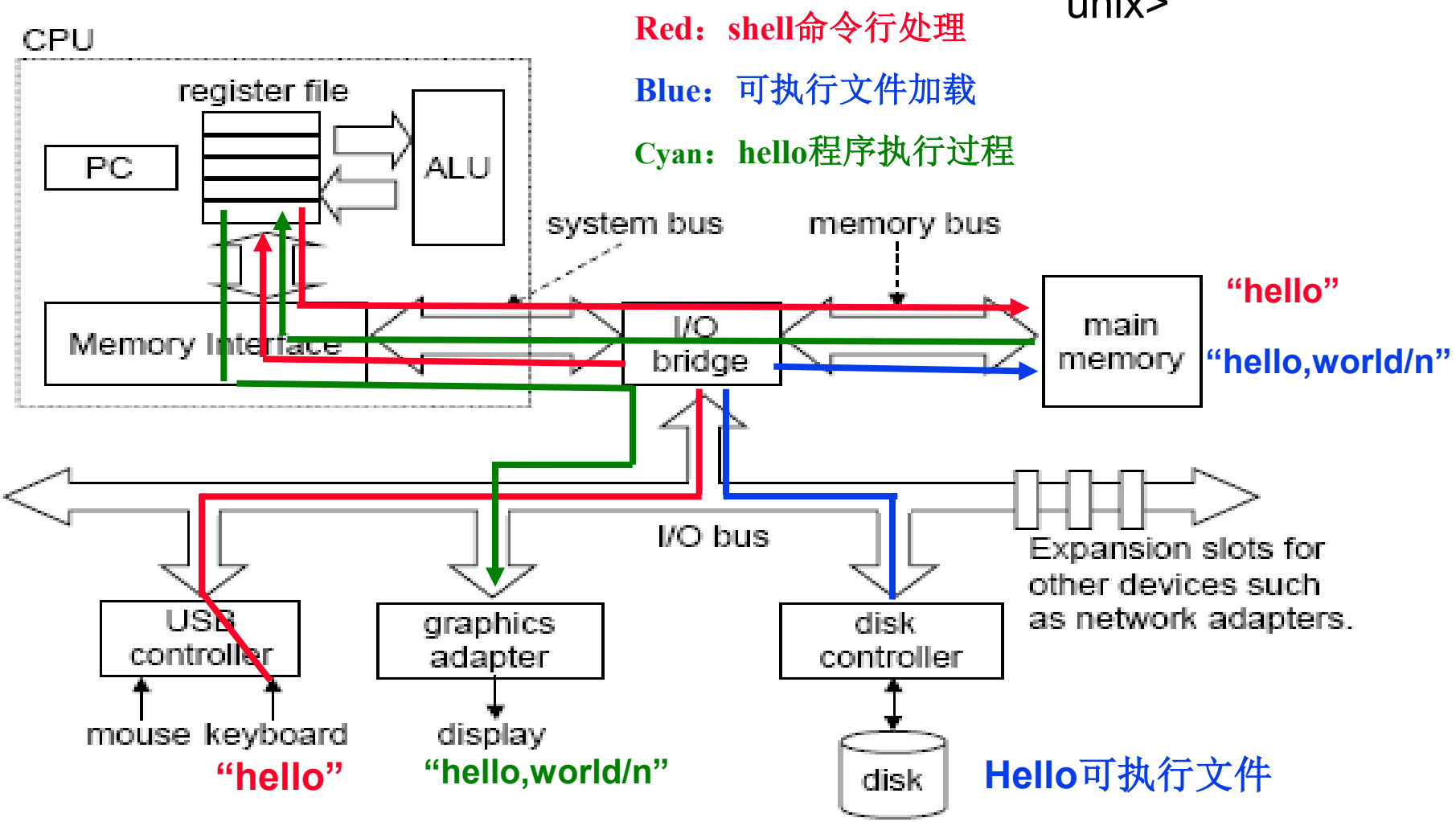
hello.c的ASCII文本表示

```
# 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程序的数据流动过程

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

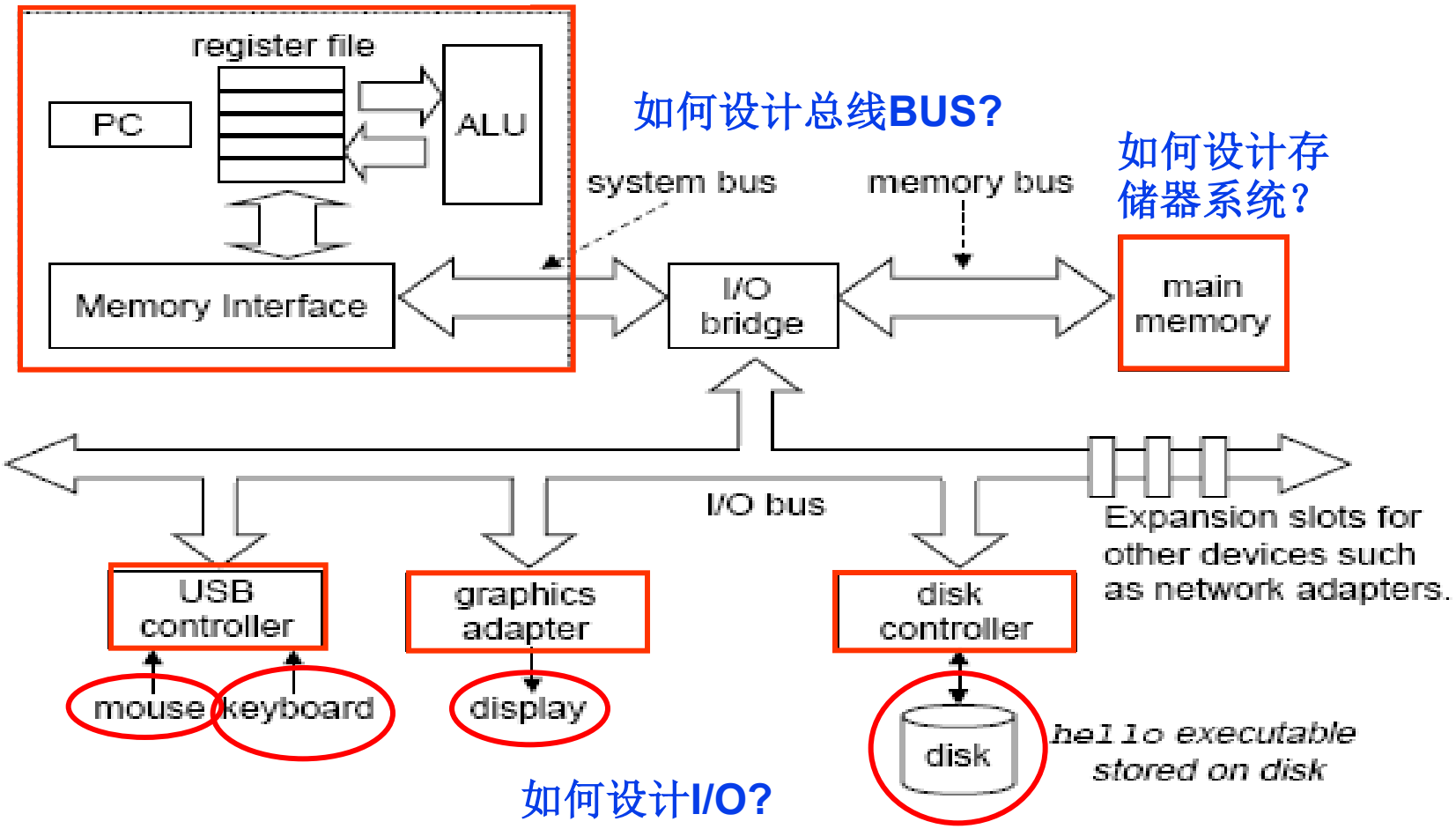


数据经常在各存储部件间传送。故现代计算机大多采用“缓存”技术！

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

如何设计高性能CPU?



如何设计总线BUS?

如何设计存储器系统?

如何设计I/O?

信息（指令和数据）在计算机中如何表示?

指令系统如何设计?

Course Outline

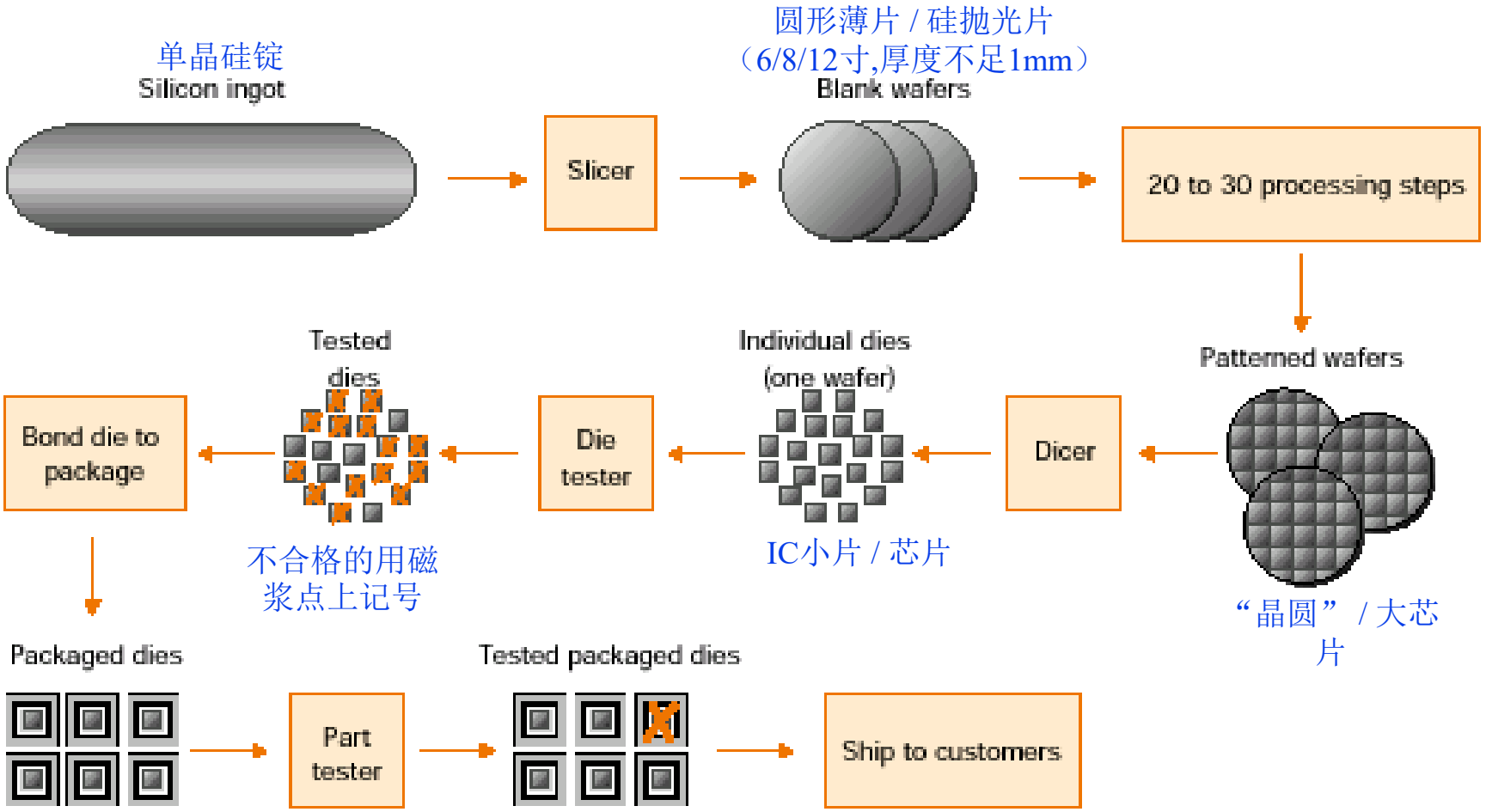
- 性能评价（Performance measurement）
- 计算机算术（Arithmetic for Computer）
  - 数据的表示和运算
- 存储器层次结构（Memory Hierarchies ）
- 指令集体系结构（Instruction Set Architecture）
- CPU设计
  - 数据通路 (Data path) 和控制器(Control Unit)
- 流水线技术 (Pipelining)
- 系统总线 （System Buses）
- 输入/输出系统（Input / Output system）

第二讲 计算机性能评价

- 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
- 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！

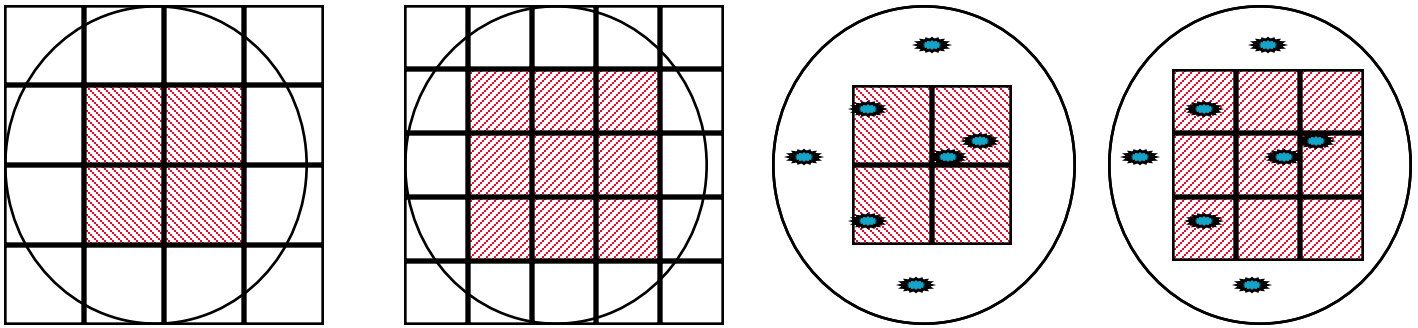
Integrated Circuits Costs 公式

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

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

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！

计算机性能的基本评价指标

- 计算机有两种不同的性能
  - 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！

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以下两个时间：
    - 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

CPI：Cycles Per Instruction

CPU 执行时间 = CPU时钟周期数 / 程序 X 时钟周期

= CPU时钟周期数 / 程序 ÷ 时钟频率

= 指令条数 / 程序 X CPI X 时钟周期

CPU时钟周期数 / 程序 = 指令条数 / 程序 X CPI

CPI = CPU时钟周期数 / 程序 ÷指令条数 / 程序

CPI 用来衡量以下各方面的综合结果

- Instruction Set Architecture（ISA）
- Implementation of that architecture
- Program（Compiler、Algorithm）

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。

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：

$$\text{总时钟数} = \sum_{i=1}^{n} CPI_i \times C_i$$

所以，

$$\text{CPU时间} = \text{时钟周期} \times \sum_{i=1}^{n} CPI_i \times C_i$$

假定CPIi 、 Fi是各指令CPI和在程序中的出现频率，则程序综合CPI为:

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

已知CPU时间、时钟频率、总时钟数、指令条数，则程序综合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

程序P在机器A上运行需10 s， 机器A的时钟频率为400MHz。现在要设计一台机器B，希望该程序在B上运行只需6 s.

机器B时钟频率的提高导致了其CPI的增加，使得程序P在机器B上时钟周期数是在机器A上的1.2倍。机器B的时钟频率达到A的多少倍才能使程序P在B上执行速度是A上的10/6=1.67倍？

Answer:

CPU时间A = 时钟周期数A / 时钟频率A

时钟周期数A = 10 sec x 400MHz = 4000M个

时钟频率B = 时钟周期数B / CPU时间B

= 1.2 x 4000M / 6 sec = 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！

Marketing Metrics （产品宣称指标）

MIPS = Instruction Count / Time x10^6

= Clock Rate / CPI x 10^6

Million Instructions Per Second

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！

问题：TFLOPS、PFLOPS等的含义是什么？

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

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%

New	Freq
27%	
27%	
15%	
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！

Successful Benchmark: SPEC

- 1988年，5家公司（ Sun, MIPS, HP, Apollo, DEC ）联合提出了SPEC (Systems Performance Evaluation Committee)
- 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
  - “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 exec. 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，可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

- 可用以下两种平均值来评价：
  - **Arithmetic mean(算术平均)：求和后除n**

$$A_m = \frac{1}{n}\sum_{i=1}^{n} R_i = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{T_i} = \frac{1}{n}\left(\frac{1}{T_1} + \frac{1}{T_2} + \ldots + \frac{1}{T_n}\right)$$

  - **Geometric mean(几何平均)：求积后开根号n**

$$G_m = \sqrt[n]{(\prod_{i=1}^{n} R_i)} = \sqrt[n]{(\prod_{i=1}^{n} \frac{1}{T_i})}$$

- 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）：

$$\frac{G_m(X_i)}{G_m(Y_i)} = G_m(\frac{X_i}{Y_i})$$

- 平均规格化执行时间不能用算术平均计算，而应该用几何平均
  - **program A going from 2s to 1s as important as program B going from 2000s to 1000s.**

（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！

计算机系统的性能评测与定量设计原理

- 计算机定量设计的原理

  - 哈夫曼压缩原理
  - Amdahl定律
  - 程序访问的局部性规律

计算机系统的性能评测与定量设计原理

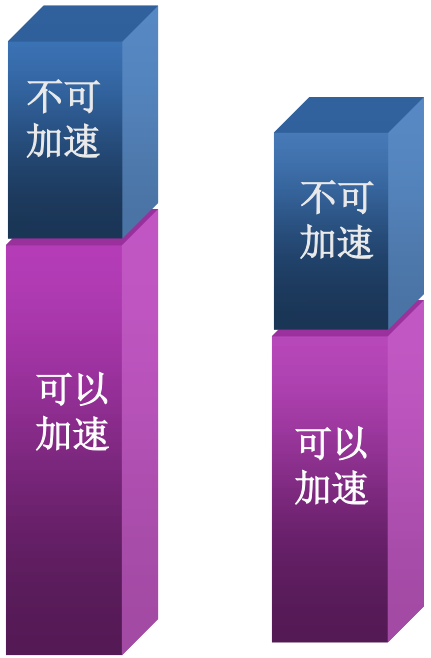
- 哈夫曼压缩原理
  - 尽可能加速处理高概率事件远比加速低概率事件对性能得提高要显著。
    - CPU在运算中出现溢出得概率很低，设计加快不溢出时得速度。
    - 缩短高频指令长度来减少占用主存空间。

计算机系统的性能评测与定量设计原理

- Amdahl定律
  - 该定律将“关注经常性事件原则”进行了量化
  - 用于确定对系统中性能瓶颈部件采取措施提高速度后能得到系统性能改进的程度，即系统加速比S_p
  - S_p定义为系统改进后的性能与未改进时的性能的比值，或者定义为系统未改进时的程序执行时间T_{old}与改进后程序执行时间T_{new}的比值
  - S_p与两个因素有关，即性能可改进比f_{new}和部件加速比r_{new}

计算机系统的性能评测与定量设计原理

- 性能可改进比f_{new}是系统性能可改进部件占用的时间与未改进时系统总执行时间的比值
  - $0 \leq f_{new} \leq 1$
- 部件加速比r_{new}是系统性能可改进部分在改进后性能提高的比值
  - $r_{new} > 1$
  - $r_{new} = T_{rold} / T_{rnew}$



计算机系统的性能评测与定量设计原理

$$S_p = \frac{\text{没有采用改进措施前执行某任务的时间}}{\text{采用改进措施后执行某任务的时间}}$$

$$= \frac{T_{old}}{T_{new}} = \frac{T_{old}}{(1-f_{new})*T_{old} + T_{rnew}}$$

$$= \frac{T_{old}}{(1-f_{new})*T_{old} + T_{rold}/r_{new}}$$

$$= \frac{T_{old}}{(1-f_{new})*T_{old} + T_{old}*f_{new}/r_{new}}$$

$$= \frac{1}{(1-f_{new}) + f_{new}/r_{new}}$$

- 当f_{new}为0时，S_p=1
- 当r_{new}趋于无穷大时，

$$S_p = \frac{1}{(1-f_{new})}$$

- 通过使用某种较快的执行方式所获得的性能提高，受限于该部件占用系统执行时间的百分比
- 它是一个悲观定律

计算机系统的性能评测与定量设计原理

假设将某系统的某一部件的处理速度加快到10倍,但该部件的原处理时间仅为整个运行时间的40%，则采用加快措施后能使整个系统的性能提高多少?

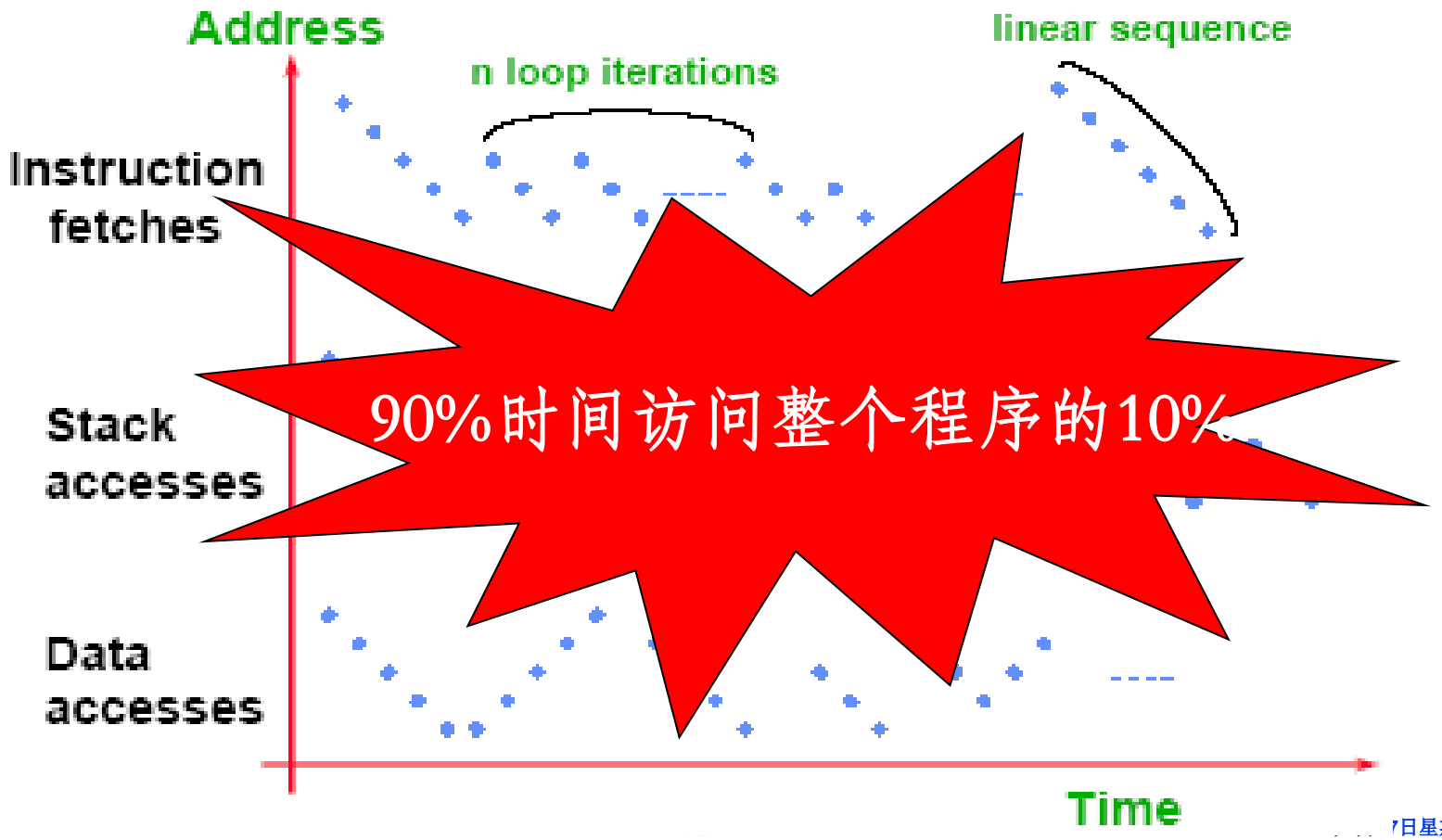
解：**由题意可知：** $f_{new}=0.4$, $r_{new}=10$，**根据Amdahl定律**

$$S_n = \frac{1}{0.6+\frac{0.4}{10}} = \frac{1}{0.64} \approx 1.56$$

如果r_{new}=100，S_p=1.66，**性能提高的增量仅为10%，所以应该平衡的提高各个部件性能来提高性能，不能只是其中某一功能部件的性能得到提高。**

计算机系统的性能评测与定量设计原理

- 程序访问的局部性定律
  - 时间局部性: 一个存储项被访问,可能很快再访问。
  - 空间局部性: 存储项被访问,它的邻近项可能很快被访问。



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（单周期/多周期/流水线）
  - 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

- 习题1中2（4）、3、4、5、8、10

第1题需要掌握不需写作业提交；

其它写在A4纸上，要求书写工整、清楚，答题要完整！