

计算系统结构

Computer Architecture

计算机与大数据学院

林嘉雯

`ljw@fzu.edu.cn`

Q1: 购置电脑时, 除价格外, 考虑哪些性能?

Q2: 在性能与价格间如何抉择?

Q3: 在同等硬件成本下, 如何划分软硬件功能? 部件如何组织? 如何评价设计方案? 是否可以优化?

系统结构的简单理解: 系统功能的软硬件划分、部件组织的结构与技术

● 课程目标

- 计算机系统结构的研究;
- 并行处理技术的研究

● 课程任务:

1

掌握系统结构的基本概念、基本原理以及计算机系统的基本分析与设计方法

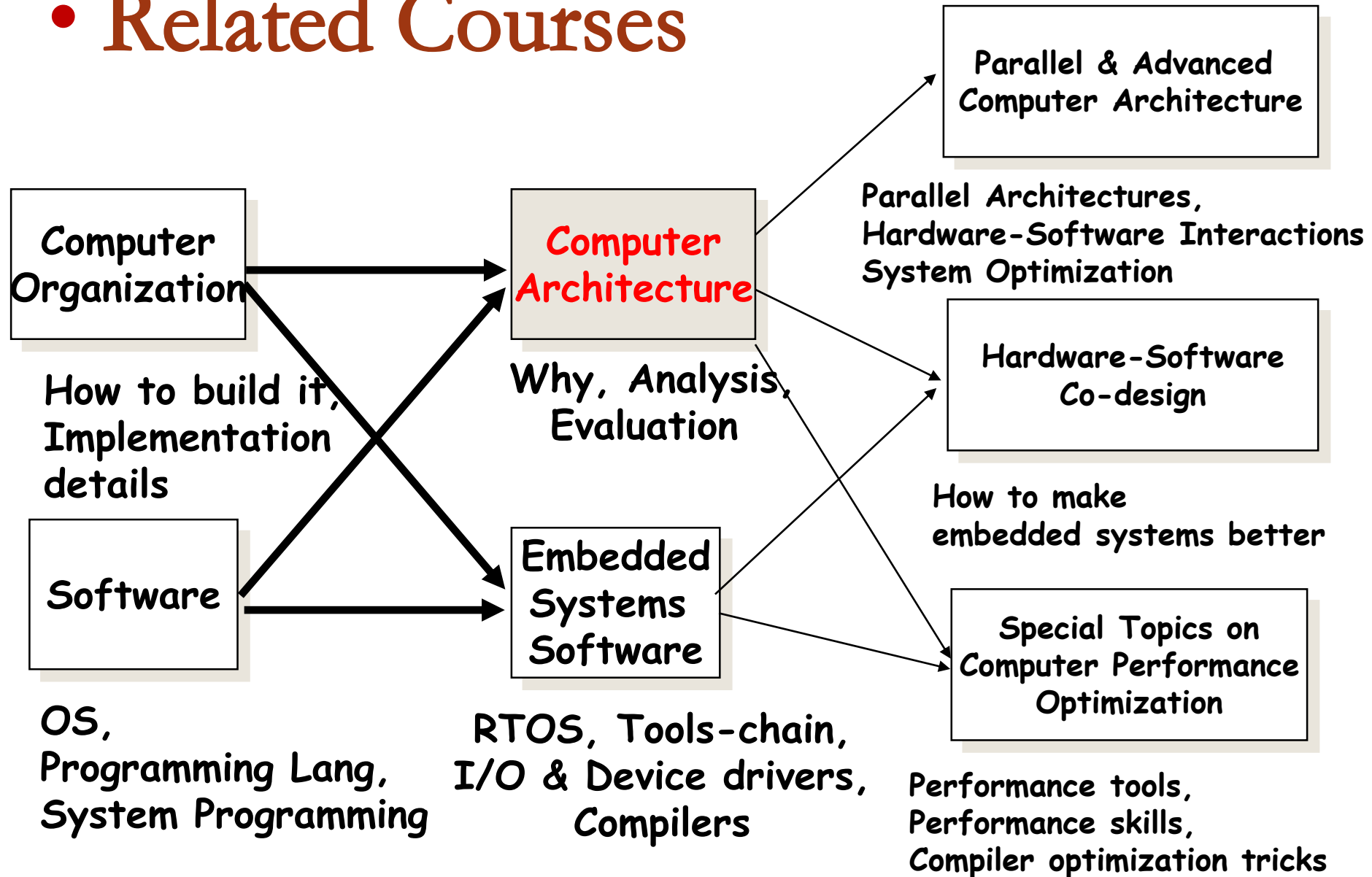
2

掌握并行处理的相关技术

3

了解计算机体系结构的新技术和新方法

• Related Courses



● 学习要求

- 课前预习，课后复习；
- 保证课堂效率，按时完成作业；
- 善用网络资源，积极讨论、交流。

● 学习资料

- 《计算机系统结构》（第5版），李学干编著，西安电子科技大学出版社
- 计算机系统结构，张晨曦等，高等教育出版社
- 《**Computer Systems**》（深入理解计算机系统-第3版），**Randal E.Bryant**，机械工业出版社

- 网络资源

- 福州大学课程中心平台：

(<http://met2.fzu.edu.cn/meol/index.do>)

手机app： 优慕课

- QQ交流群： 717821938

- 考核方式

期末考试（闭卷）70%

平时成绩30%， 包含：出勤、小测、作业

第1章 计算机系统结构概论

主要内容:

- 计算机系统的层次结构
- 计算机系统结构、组成与实现
- 计算机系统设计的一般原则
- 软件的可移植性
- 并行性发展与计算机系统的分类

1.1 计算机系统的多级层次结构

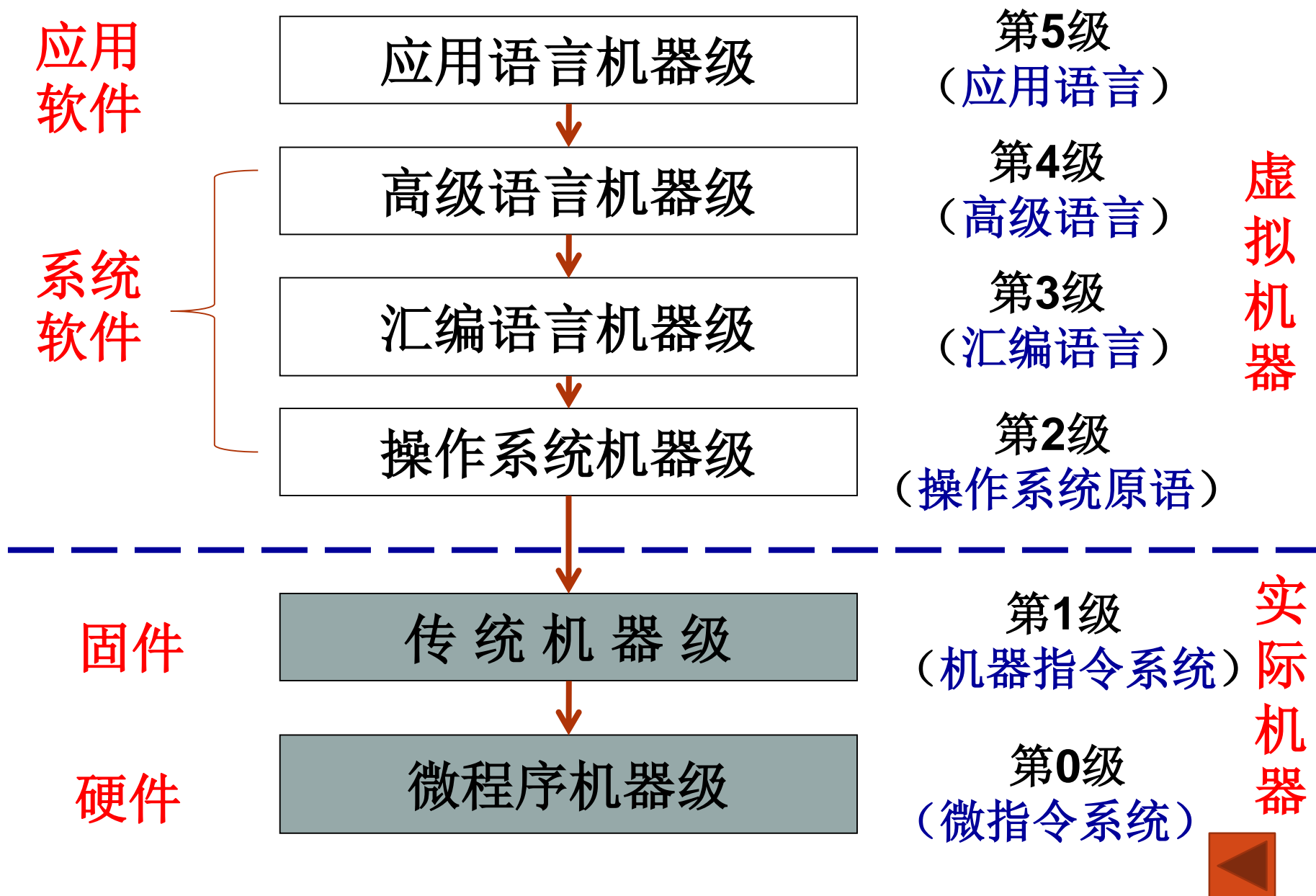
- 计算机系统=硬件/固件+软件
- 计算机语言从低级往高级发展电子数字计算机

高一级语言功能更强，但均以低级语言为基础。

- 从计算机语言的角度，可以将计算机系统看成是按功能划分的多层次结构。

每一层以一种语言为特征。

计算机系统多层次结构图



从设计人员看到的层次

- 应用程序级 ← 用户
- 高级语言级 ← 高级语言程序员
- 汇编语言级 ← 汇编语言程序员
- 操作系统级 ← 操作员
- 机器语言级 ← 机器语言程序员
- 微程序控制级 ← 逻辑程序员、微程序员
- 硬联逻辑级 ← 硬件设计员

□ 各机器级的实现主要靠翻译或解释，或两者结合。

- **翻译**：先用转换程序将高级机器级上的程序整个地变换成低级机器级上等效的程序，然后在低级机器级上实现的技术

编译程序实现将高级语言源程序转换为机器语言目标程序

- **解释**：在低级机器级上用它的一串语句或指令来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。

微指令程序解释实现机器指令



解释执行比翻译后再执行所花时间多，但占用存储空间较小

○ 翻译与解释对比：

翻译	解释
多占存储空间 (中间代码、目标代码)	节省存储空间
需要翻译程序	需要编译程序
不占运行时间	运行费时

○ 两种方法经常一起使用

高级语言源程序 $\xrightarrow{\text{翻译}}$ 机器语言程序 $\xrightarrow{\text{解释}}$ 微程序

\searrow 翻译
易于执行的中间代码 $\xrightarrow{\text{解释}}$ 机器指令

❓从概念和功能上将计算机系统看成多级层次结构有哪些优点？

1. 理解软件、硬件、固件的地位和作用
2. 理解各种语言的实质和实现途径
3. 探索虚拟机新的实现方法和新的系统设计
 由硬件固件实现——高级语言机器
 多处理机系统——由真正微处理机实现
4. 理解计算机体系结构的定义，合理进行软、硬件系统设计与开发

1.2 计算机系统结构、组成与实现

计算机系统的多级层次结构的划分是相对的，其中的每一层机器只对一定的用户或人员存在，其功能在广义的概念上是对该层语言提供解释手段，用该层的语言了解和使用计算机，而不必关心其他。

对每一层的使用者，他是如何观察计算机，并且看到了解了什么呢？

1.2 计算机系统结构、组成与实现

1.2.1 计算机系统结构

计算机系统的多级层次结构的划分是相对的，不同级别的程序员所了解到的**计算机属性**是不一样的，低层机器的属性对高层机器的程序员基本上是**透明**的（即**抽象使用**）。

计算机属性：计算机的概念性结构和功能特性。

透明性：本来存在的事物或属性，从某个角度去看却好象不存在，这种概念称为**透明性**。



计算机系统中常见的透明（抽象使用）

指令集架构、指令系统

无需了解底层复杂的硬件

应用程序接口（API）

无需了解内部具体编码

虚拟内存

无需了解程序存储器

文件

无需了解I/O设备的工作

虚拟机

无需了解机器的处理器、操作系统和程序

系统结构就是要研究对于某级，哪些属性应该透明，哪些属性不应该透明。更本质地说，**系统结构就是某一语言程序员在对应的机器级上能够编写正确运行的程序所必须了解的所有计算机属性的集合。**

计算机系统结构也称计算机系统的体系结构,它只是系统结构中的一部分，指的是**传统机器级的系统结构**，即机器级程序员所看到的计算机属性（包括了**概念性结构**和**功能特性**）

概念性结构和功能特性——外特性

例如

- 人需要有消化系统，胃→外特性
- 人需要有思维系统，大脑→外特性
- 动物需要运动系统，运动系统相关的器官→外特性

计算机的外特性有哪些？

- 计算机需要运算系统，需要运算器件。
- 计算机需要存储系统，需要存储器件。
- 计算机需要I/O系统，需要输入输出设备。

运算器件是怎么实现的，电路是如何构造的不是外特性

□透明性概念

➤ CPU类型、型号，主存容量

对应用程序员

透明

对系统程序员、硬件设计人员

不透明

➤ 浮点数表示、乘法指令

对汇编语言程序员、机器语言程序员

不透明

对高级语言程序员、应用程序员

透明

➤ 数据总线宽度、微程序

对硬件设计人员

透明

对计算机维修人员

透明

对汇编语言程序员

不透明

1、计算机系统结构的实质

完成硬件、软件的功能分配, 对计算机的机器级界面的确定。

界面之上, 是软件实现的功能; 界面之下是硬件和固件实现的功能。

软件硬件在逻辑功能上是等效的, 但在性能、价格、实现难易程度上不同。各级的实现应全面考虑系统的效率、速度、价格等, 对软、硬、固件综合取舍。



2、计算机系统结构的属性

①数据表示

硬件能直接识别和处理的数据类型、格式

②寻址方式

最小寻址单位，寻址方式的种类，地址的计算

③寄存器组织

数据寄存器、变址、控制寄存器、专用寄存器等数量，使用方法

④指令系统

指令的格式、类型，使用方法

⑤存储系统

最小编址单位，编址方式，存储容量，最大存储空间

⑥中断系统

中断类型，优先级别，入口地址的形成等

⑦I /O 接口的结构

I / O的连接方式、访问方式、编方式址

⑧信息的保护

保护方式，硬件对信息的保护和支持

⑨机器工作状态、定义和切换

1.2.2 计算机组成

1、计算机组成的定义

计算机系统结构的逻辑实现

包括机器级内的数据流、控制流的组成及逻辑设计，完成机器内控制机构、各部件的功能及连接

2、计算机组成的目的

解决在性能价格比的要求下，达到最佳、最合理地把各种部件、设备组成计算机，实现系统结构所确定的功能。

重点：提高速度、提高并行度、功能分散和专用部件设置

3、计算机组成设计所包含的内容

- ① 数据通路宽度
- ② 专用部件的设置
- ③ 各种操作对部件的共享程度
- ④ 功能部件的并行度
- ⑤ 控制机构的组成方式
- ⑥ 缓冲和排队技术
- ⑦ 可靠性技术

1.2.3 计算机实现

1、计算机实现的定义

计算机组成的物理实现

主要涉及器件技术和组装技术

2、计算机实现的内容

- ① 处理机、主存、I/O 接口等部件的物理结构
- ② 器件的集成度和速度的选择
- ③ 器件、模块、插件、底板的划分与连接
- ④ 专用器件的设计
- ⑤ 组装技术
- ⑥ 信号传输
- ⑦ 电源、冷却、整机的装配等

例如：计算机的存储系统

主存系统的确定
(如容量/编址方式等)

计算机系统结构

主存的逻辑组织结构
(如容量扩充, 存储器采用
交叉存取还是并行存取等)

计算机组成

元器件、组装技术的选择

计算机实现

* 指令执行采用串行、重叠还是流水控制方式，对系统结构来说是透明的，但对计算机组成来说不是透明的。

* 乘法指令采用专用乘法器实现，对系统结构来说是透明的，而对计算机组成来说不是透明的

* 存储器采用哪种芯片，对计算机系统结构和组成来说是透明的，而对计算机实现来说不是透明的。

1.2.4 三者的关系

计算机系统结构——系统设计
计算机组成——逻辑设计
计算机实现——物理设计

例如：计算机指令系统中

指令系统的确定

计算机系统结构

指令的实现

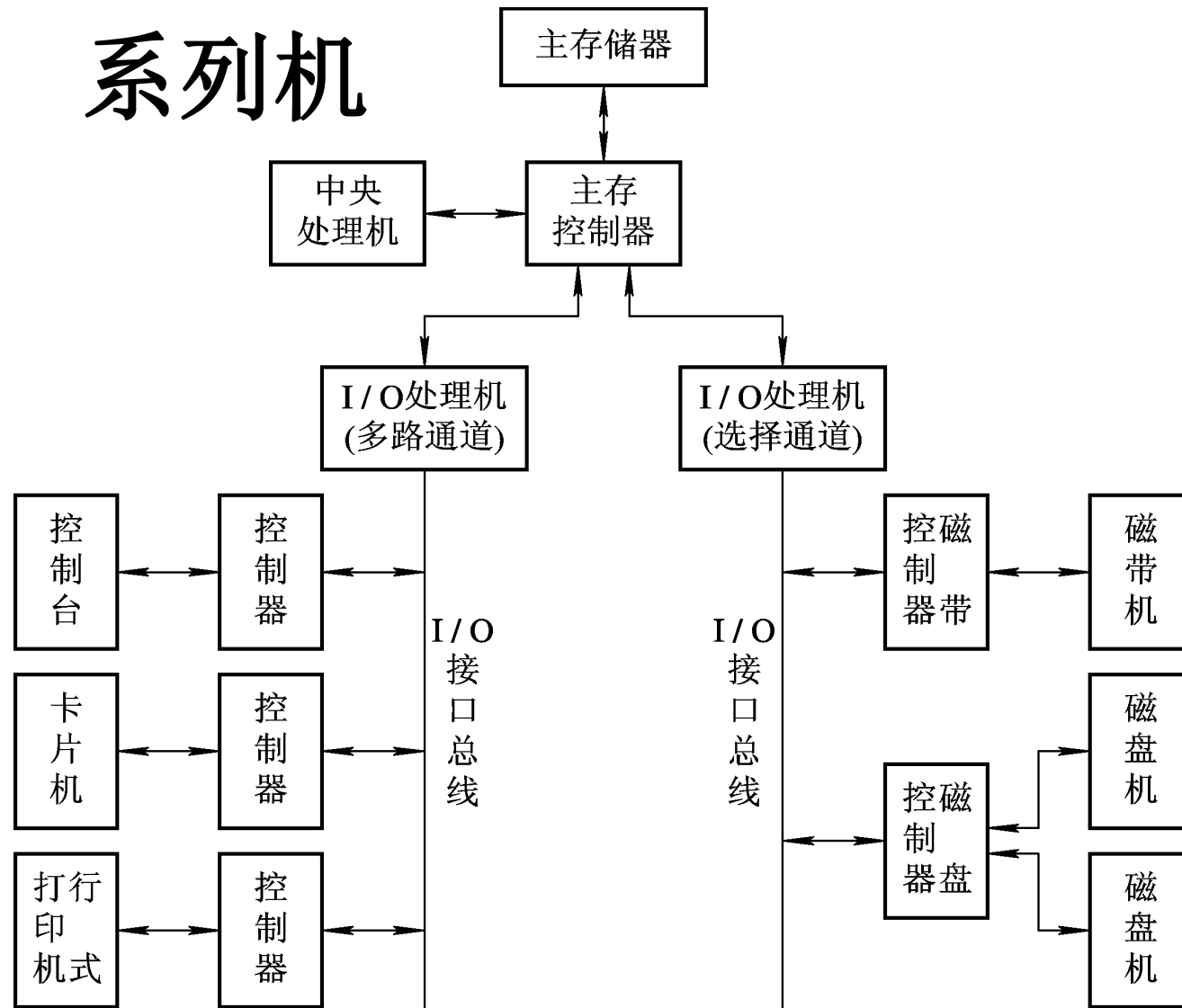
计算机组成

指令实现的具体电路

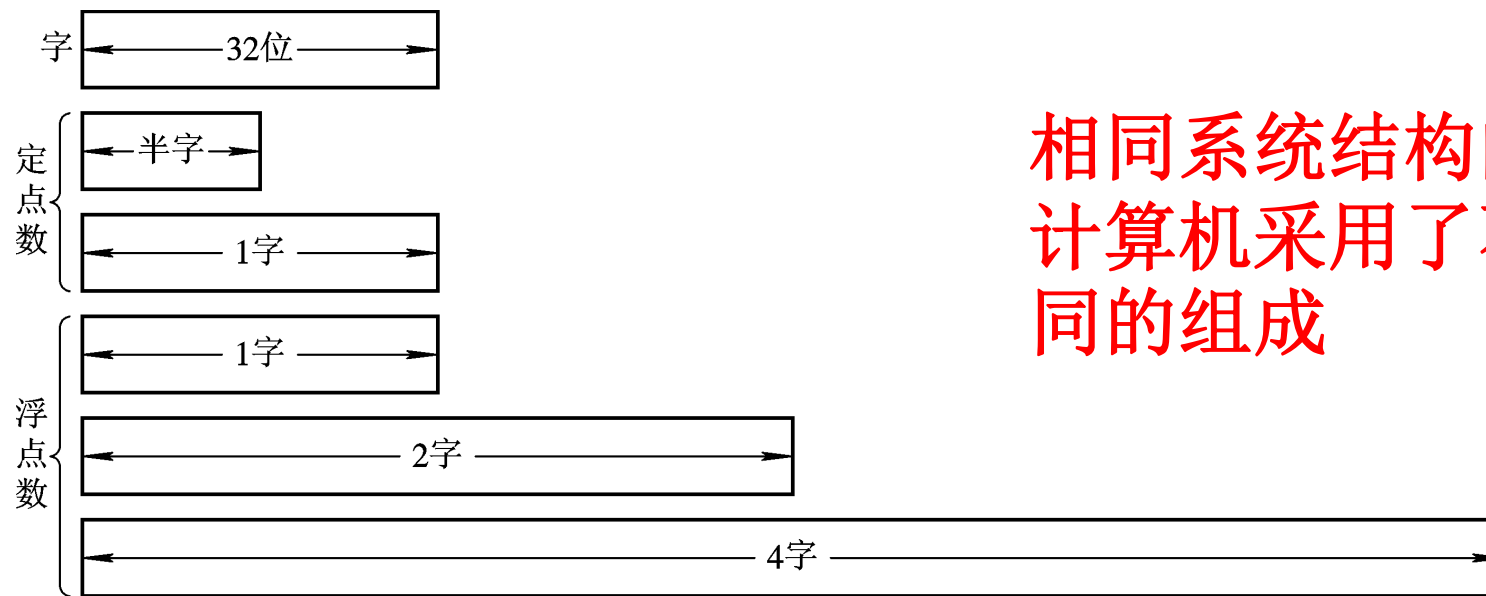
计算机实现

相同系统结构的计算机可以采用不同的组成；
相同组成的计算机可以采用不同的实现技术。

例如：系列机

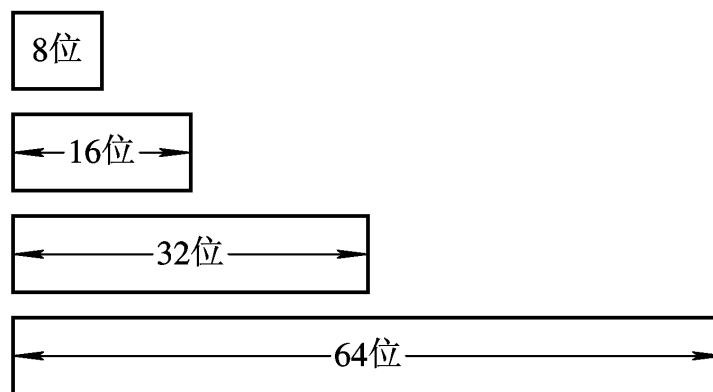


IBM370系列的概念性结构



相同系统结构的
计算机采用了不
同的组成

(a)



(b)

IBM370系列字长、数的表示和数据通路宽度
(a)统一的字长与定、浮点数表示；(b)不同的数据通路宽度

例如 系列机 Family Machine

计算机	时间	处理器	字宽	主要I/O总线	存储空间
PC和PC XT	1981	8088	16位	PC总线	20位
PC AT	1982	80286	16位	AT (ISA)	24位
80386 PC	1985	80386	32位	ISA/EISA	32位
80486 PC	1989	80486	32位	ISA+VL	32位
Pentium PC	1993	Pentium	32位	ISA+PCI	32位
Pentium II PC	1997	Pentium II	32位	ISA+PCI+AGP	32位
Pentium III PC	1999	Pentium III	32位	PCI+AGP +USB	32位
Pentium 4 PC	2000	Pentium 4	32位	PCI-X+AGP +USB	32位

□三者的相互影响

组成设计向上决定于结构，向下受限于实现技术

- 结构不同，采用的组成技术不同；
- 组成技术的进步推动结构的发展；
- 实现技术的发展使得三者的关系更加紧密



现在的计算机系统结构实际包含了系统结构和组成两方面内容，研究的是软、硬件的功能分配以及如何更好、更合理地实现分配给硬件的功能的计算机组成

举例：不同系统结构对组成的要求不同

A:= B+C D:=E*F

面向三地址寻址：

ADD B,C,A

MPY E,F,D

面向寄存器：

LOAD R1,B

ADD R1,C

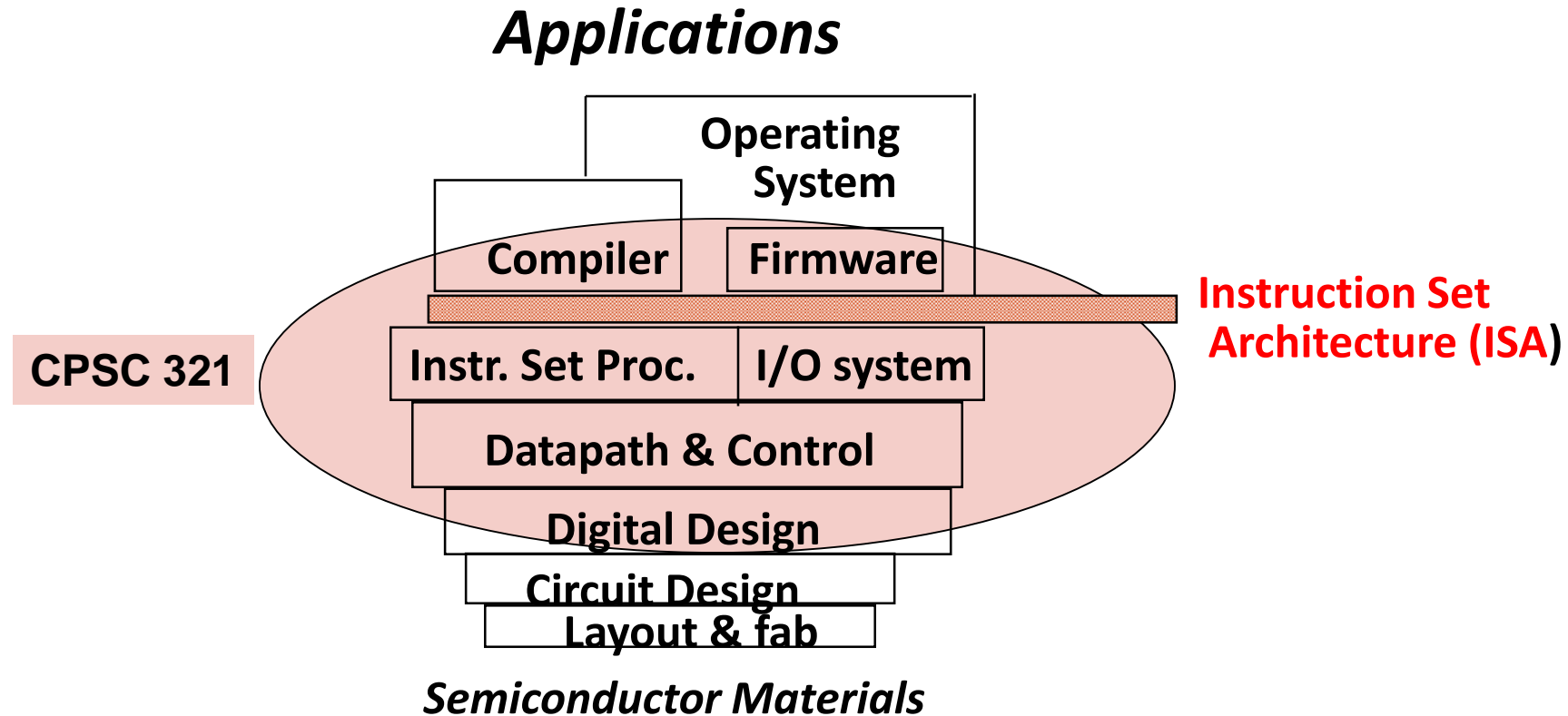
STORE R1,A

LOAD R2,E

MPY R2,F

STORE R2,D

What is “Computer Architecture”?

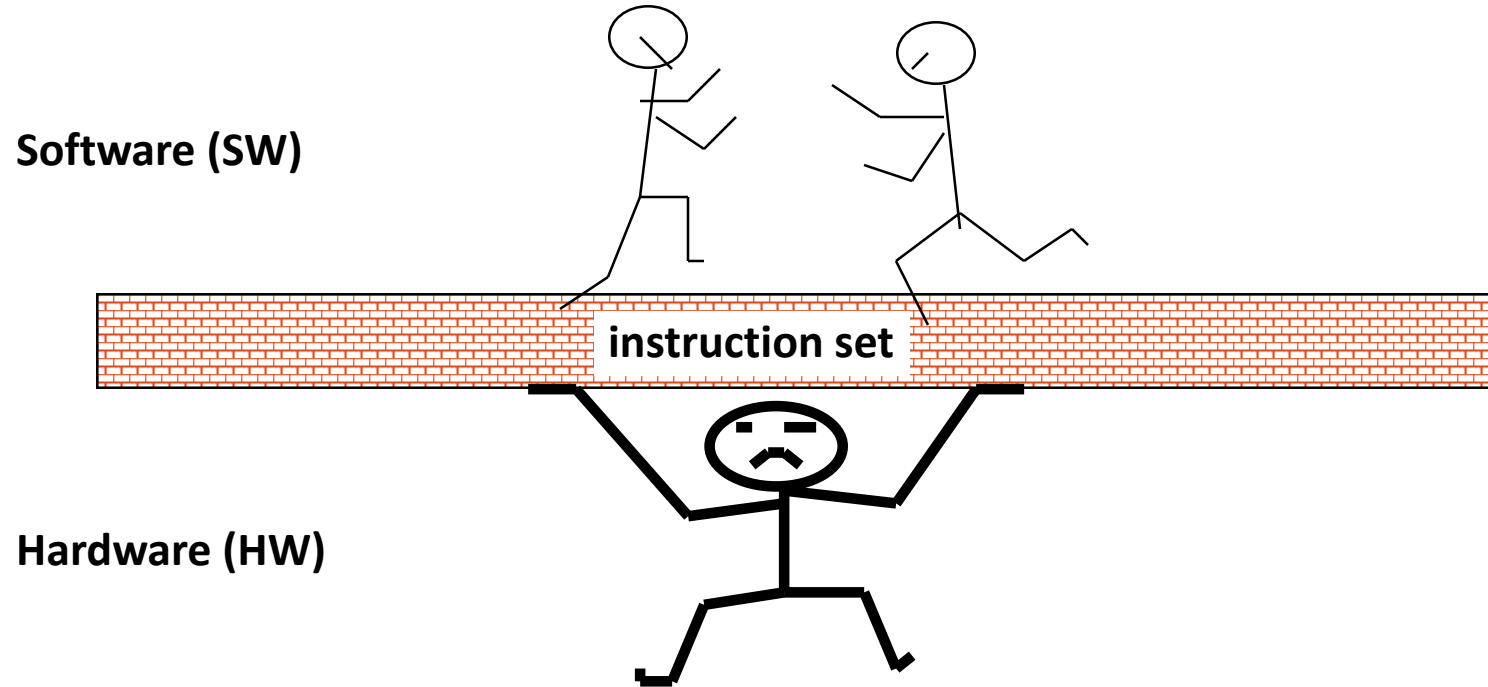


- Coordination of many *levels of abstraction*
- Under a rapidly *changing set of forces*
- Design, Measurement, *and* Evaluation

Computer Architecture - Definition

- **Computer Architecture = ISA + MO**
- **Instruction Set Architecture**
 - What the executable can “see” as underlying hardware
 - Logical View
- **Machine Organization**
 - How the hardware implements ISA ?
 - Physical View

The Instruction Set: a Critical Interface



- Properties of a good abstraction
 - Lasts through many generations (portability)
 - Used in many different ways (generality)
 - Provides **convenient** functionality to higher levels
 - Permits an **efficient** implementation at lower levels

第1章 计算机系统结构概论

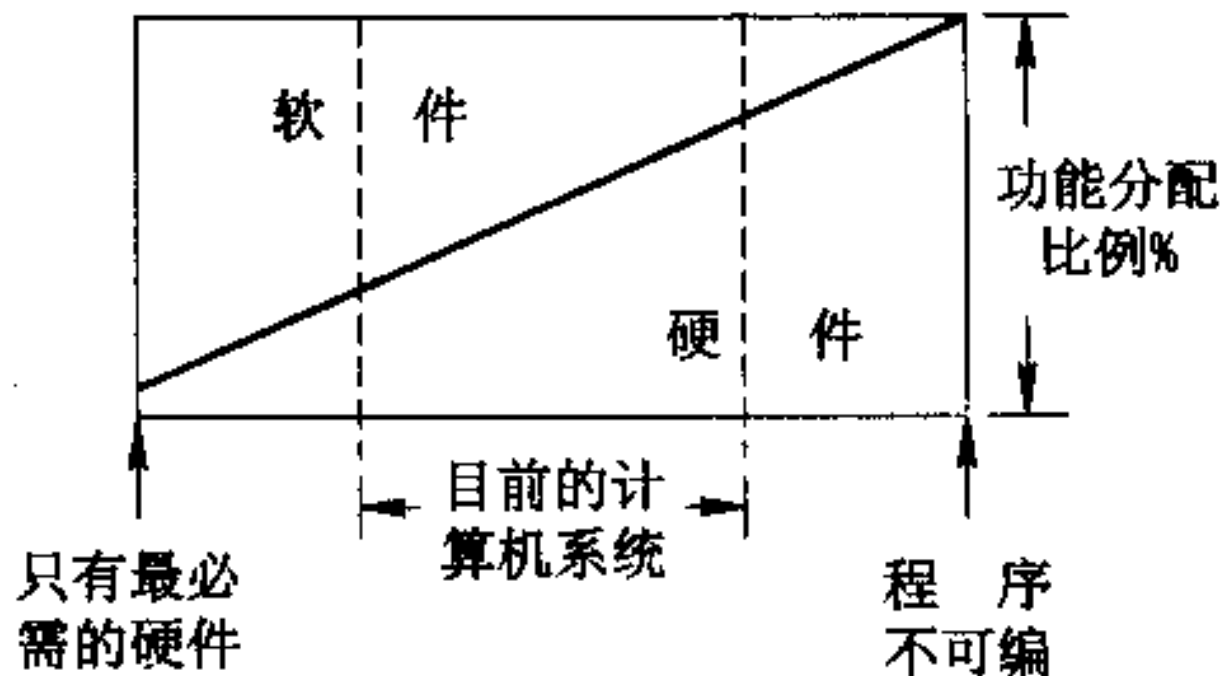
主要内容:

- 计算机系统的多级层次结构
- 计算机系统结构、组成与实现
- 计算机系统设计的一般原则
- 软件的可移植性
- 并行性发展与计算机系统的分类

1.3 计算机系统设计准则

1.3.1 软硬件取舍原则 (P8-9, 3点)

软件和硬件在**逻辑功能上是等效的**，只是性能、价格、实现的难易程度不同。



计算机系统功能分配的三种方法

○全硬件、软硬件结合、全软件

逻辑功能上等价

- 硬件实现：速度快、成本高；灵活性差、占用内存少
- 软件实现：速度慢、复制费用低；灵活性好、占用内存多、易设计、可改性强、适应性强、设计周期短；

○发展趋势

- 硬件实现的比例越来越高
- 硬件所占的成本越来越低

软硬件实现的优缺点

	软件	硬件
速度	慢	快
系统灵活性、 适应性	强	弱
成本	低	高

1.3.2 计算机系统的性能评价

计算机系统的性能指标体现于时间和空间两个方面。其中，在系统上程序实际运行的时间应该是衡量机器时间(速度)性能最可靠的标准。

程序的实际运行时间 = CPU时间 (用户CPU时间 + 系统CPU时间) + I/O 时间 + 访存时间...

对用户来说，更关心的是**用户的CPU时间**

计算CPU的程序执行时间 T_{CPU} 有3个因素:

- 程序执行的总指令条数 IC (Instruction Counter)
- 平均每条指令的时钟周期数 CPI (Cycles Per Instruction)
- 主时钟频率 f_c

$$T_{\text{CPU}} = \text{IC} \times \text{CPI} \times \frac{1}{f_c}$$

假设系统共有 n 种指令，第 i 种指令的时钟周期数为 CPI_i ，第 i 种指令在程序中出现的次数为 I_i ，则

$$T_{\text{CPU}} = \left[\sum_{i=1}^n (\text{CPI}_i \times I_i) \right] \times \frac{1}{f_c}$$

$$\text{CPI} = \frac{\sum_{i=1}^n (\text{CPI}_i \times I_i)}{\text{IC}} = \sum_{i=1}^n (\text{CPI}_i \times \frac{I_i}{\text{IC}})$$

其中， I_i/IC 为第 i 种指令在程序总指令数 IC 中所占的比值。

例：假设我们考虑条件分支指令的两种不同设计方法如下：

(1) CPU_A：通过比较指令设置条件码，然后测试条件码进行分支。

(2) CPU_B：在分支指令中包括比较过程

在两种**CPU**中，条件分支指令都占用**2**个时钟周期而所有其它指令占用**1**个时钟周期，对于**CPU_A**，执行的指令中**分支指令占20%**；由于每个分支指令之前都需要有比较指令，因此**比较指令也占20%**。

假设CPU_B的时钟周期时间是CPU_A的1.25倍。哪一个CPU更快？如果CPU_B的时钟周期时间是CPU_A的1.1倍，哪一个CPU更快呢？

解：我们不考虑所有系统问题，所以可用CPU性能公式。占用2个时钟周期的分支指令占总指令的20%，剩下的指令占用1个时钟周期。所以

$$CPI_A = 0.2 \times 2 + 0.80 \times 1 = 1.2$$

则CPU性能为：

$$\text{总CPU时间}_A = IC \times 1.2 \times \text{时钟周期}_A$$

根据假设，有：

$$\text{时钟周期}_B = 1.25 \times \text{时钟周期}_A$$

在CPU_B中没有独立的比较指令，所以CPU_B的程序量为CPU_A的80%，分支指令的比例为：

$$20\%/80\% = 25\%$$

这些分支指令占用2个时钟周期，而剩下的75%的指令占用1个时钟周期，因此：

$$\text{CPI}_B = 0.25 \times 2 + 0.75 \times 1 = 1.25$$

因为CPU_B不执行比较，故：

$$\text{IC}_B = 0.8 \times \text{IC}_A$$

因此**CPU_B**性能为:

$$\begin{aligned}\text{总CPU时间}_B &= IC_B \times CPI_B \times \text{时钟周期}_B \\ &= 0.8 \times IC_A \times 1.25 \times (1.25 \times \text{时钟周期}_A) \\ &= 1.25 \times IC_A \times \text{时钟周期}_A\end{aligned}$$

在这些假设之下, 尽管**CPU_B**执行指令条数较少,
CPU_A因为有着更短的时钟周期, 所以比**CPU_B**快。

如果**CPU_A**的时钟周期时间仅仅比**CPU_B**快**1.1**倍，
则

$$\text{时钟周期}_B = 1.10 \times \text{时钟周期}_A$$

CPU_B的性能为：

$$\begin{aligned} \text{总CPU时间}_B &= IC_B \times CPI_B \times \text{时钟周期}_B \\ &= 0.8 \times IC_A \times 1.25 \times (1.10 \times \text{时钟周期}_A) \\ &= 1.10 \times IC_A \times \text{时钟周期}_A \end{aligned}$$

因此**CPU_B**由于执行更少指令条数，比**CPU_A**运行更快。

为了反映程序的运行速度，通常引入如下一些定量指标：

- **MIPS(Million Instructions Per Second,每秒百万条指令数)。**

$$MIPS = \frac{IC}{T_{CPU} \times 10^6} = \frac{f_c}{CPI} \times 10^{-6}$$

这样，程序的执行时间为

$$T_{CPU} = \frac{IC}{MIPS} \times 10^{-6}$$

不足之处：

- 1.MIPS 依赖于指令系统，对指令系统不同的机器不准确；
- 2.同一机器上，MIPS因程序不同而变化，有时差距很大

● **MFLOPS**(Million Floating Point Operations Per Second,每秒百万次浮点运算)。

假设 I_{FN} 表示程序运行中的浮点运算总次数，有

$$\text{MFLOPS} = \frac{I_{\text{FN}}}{T_{\text{CPU}}} \times 10^{-6}$$

不足之处：

MFLOPS只能反映机器执行浮点操作的性能，并不能反映机器的整体性能。例如，在程序编译过程中，不管MFLOPS有多高，对编译速度都不会有影响。

其他常见的评价性能指标

Performance Metrics

- Response Time
 - Delay between start end end time of a task
- Throughput
 - Numbers of tasks per given time
- *New*: Power/Energy
 - Energy per task, power

一般利用测试程序对新设计的系统进行评价

● 测试程序

1. 采用实际的应用程序测试

如：C语言的编译程序，CAD应用：Spice

2. 采用核心程序测试

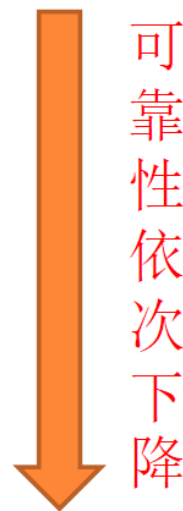
从实际程序中抽出关键部分组合而成

3. 合成测试程序

人为写的核心程序，规模小，结果预知

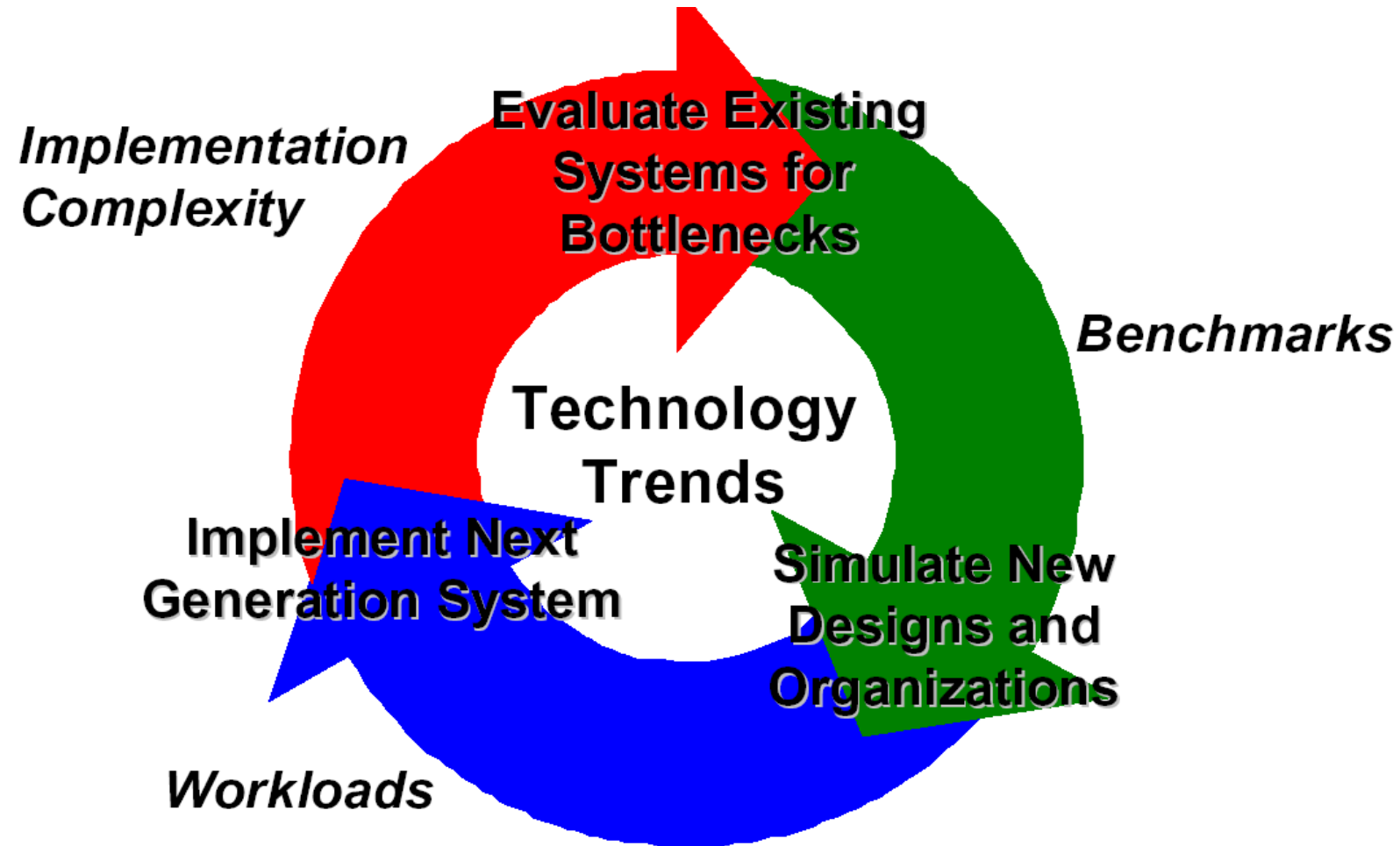
4. 综合基准测试程序

类似于核心程序，但考虑了各自操作和各种程序的比例，与核心程序相比，前者从真实程序中抽象出来的，而综合测试程序是为了体现平均执行而人为编制的



对新设计的系统评测与改进是不断循环的过程！！

The Role of Performance



1.3.3 计算机系统的定量设计原理

设计时，一般应遵循哈夫曼压缩原理、程序局部性原理、**Amdahl定律**

□ 哈夫曼压缩原理

在各种事件发生概率不相同的情况下，尽可能采用优化技术，对**高概率事件**用**最短位数（时间）**来表示（处理），而对**低概率事件**允许用**较长的位数（时间）**来表示（处理）。该原理有助于整体性能的提高。

□ 程序局部性原理

➤ 时间局部性：最近的将来要用到的信息可能就是现在正在使用的信息。

这与程序不断在循环运行有关。

➤ 空间局部性：最近的将来要用到的信息很可能与现在正在使用的信息在程序上的位置是临近的。

这与指令的顺序存放、数据的簇聚存放有关。

□Amdahl定律

该定律用于确定系统中某一部件采取措施提高速度后能得到系统性能改进的程度，即系统性能加速比 S_p 。

$$S_p = \frac{S_{\text{old}}}{S_{\text{new}}} = \frac{T_{\text{old}}}{T_{\text{new}}}$$

加速比依赖于两个因素

➤ **可改进比** f_{new} : 在改进前的系统中, 可改进部分的执行时间占系统总执行时间的比值

$$0 \leq f_{\text{new}} \leq 1$$

➤ **部件加速比** r_{new} : 可改进部分在改进后性能提高的倍数, 是改进前执行时间与改进后时间的比

$$r_{\text{new}} > 1$$

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

例1：将计算机系统中某一功能的处理速度加快**15**倍，但该功能的处理时间仅占整个系统运行时间的**40%**，则采用此增强功能方法后，能使整个系统的性能提高多少？

解：由题可知： $f_{\text{new}} = 0.4$ ， $r_{\text{new}} = 15$

根据**Amdahl**定律可知

$$\begin{aligned} S_p &= \frac{1}{(1 - f_{\text{new}}) + f_{\text{new}} / r_{\text{new}}} \\ &= \frac{1}{(1 - 0.4) + 0.4 / 15} \approx 1.6 \end{aligned}$$

例2：某计算机系统采用浮点运算部件后，使得浮点运算速度提高到原来的**25**倍，系统运行某一程序的整体性能提高到原来的**4**倍，求该程序中浮点操作所占的比例。

解：由题可知： $r_{\text{new}}=25$ ， $s_p=4$

根据**Amdahl**定律可知

$$4 = \frac{1}{(1 - f_{\text{new}}) + f_{\text{new}} / 25}$$

求得： $f_{\text{new}}=78.1\%$

第1次 作业

计算机系统中某一功能A经过改进增强后，其处理速度是未改进前的8倍。改进后，该功能的处理时间仅为整个系统运行时间的40%。试问改进增强功能A后，整个系统获得的加速比是多少？

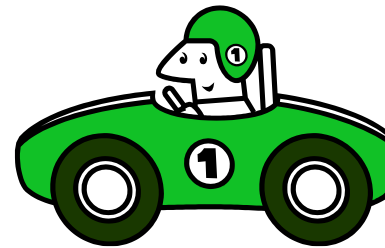
Amdahl定律表明了性能提高量的递减规律：

只对系统中的一部分进行性能改进，改进的越多，整体系统性能提高的增量却越小。

因此，Amdahl定律告诉我们，改进好的高性能系统应是一个各部分性能均能平衡地得到提高的系统，不能只是其中某一个功能部件性能的提高。

Amdahl's Law

- **Pitfall:** *Expecting the improvement of one aspect of a machine to increase performance by an amount proportional to the size of improvement*



Amdahl's Law

- Opportunity for improvement is affected by how much time the event consumes
- **Make the common case fast**
- Very high speedup requires making nearly every case fast
- *Focus on overall performance, not one aspect*

例3 某一计算机用于商业外贸的事务处理，有大量的字符串操作。由于这种事务处理很普遍，有较大的市场，故而设计人员决定在下一代此类计算机的**CPU**中加入字符串操作的功能。经测试应用软件调查发现，字符串操作的使用占整个程序运行时间的**50%**，而增加此功能如用软件（如微程序）实现，则快**5**倍，增加**CPU**成本**1/5**倍；如果用硬件实现，则快**100**倍，**CPU**成本增加到**5**倍。问设计人员提出增加此功能是否恰当？是否用软件还是硬件？

设**CPU**成本占整机成本的**1/3**。

□ 硬件实现

$$T_{new} = T_{old} (1 - 50\%) + \frac{50\% T_{old}}{100}$$

$$\text{性能变化} = \frac{T_{old}}{T_{new}} = \frac{T_{old}}{T_{old} (1 - 50\%) + \frac{50\% T_{old}}{100}} = \frac{1}{1 - 50\% + \frac{50\%}{100}} = 1.98 \text{倍}$$

$$\text{成本增加} = \frac{2}{3} * 1 + \frac{1}{3} * 5 = 2.33 \text{倍}$$

$$\text{成本性能比} = \frac{2.33}{1.98} = 1.18 \text{倍}$$

□ 软件实现

$$T_{new} = T_{old} (1 - 50\%) + \frac{50\%T_{old}}{5}$$

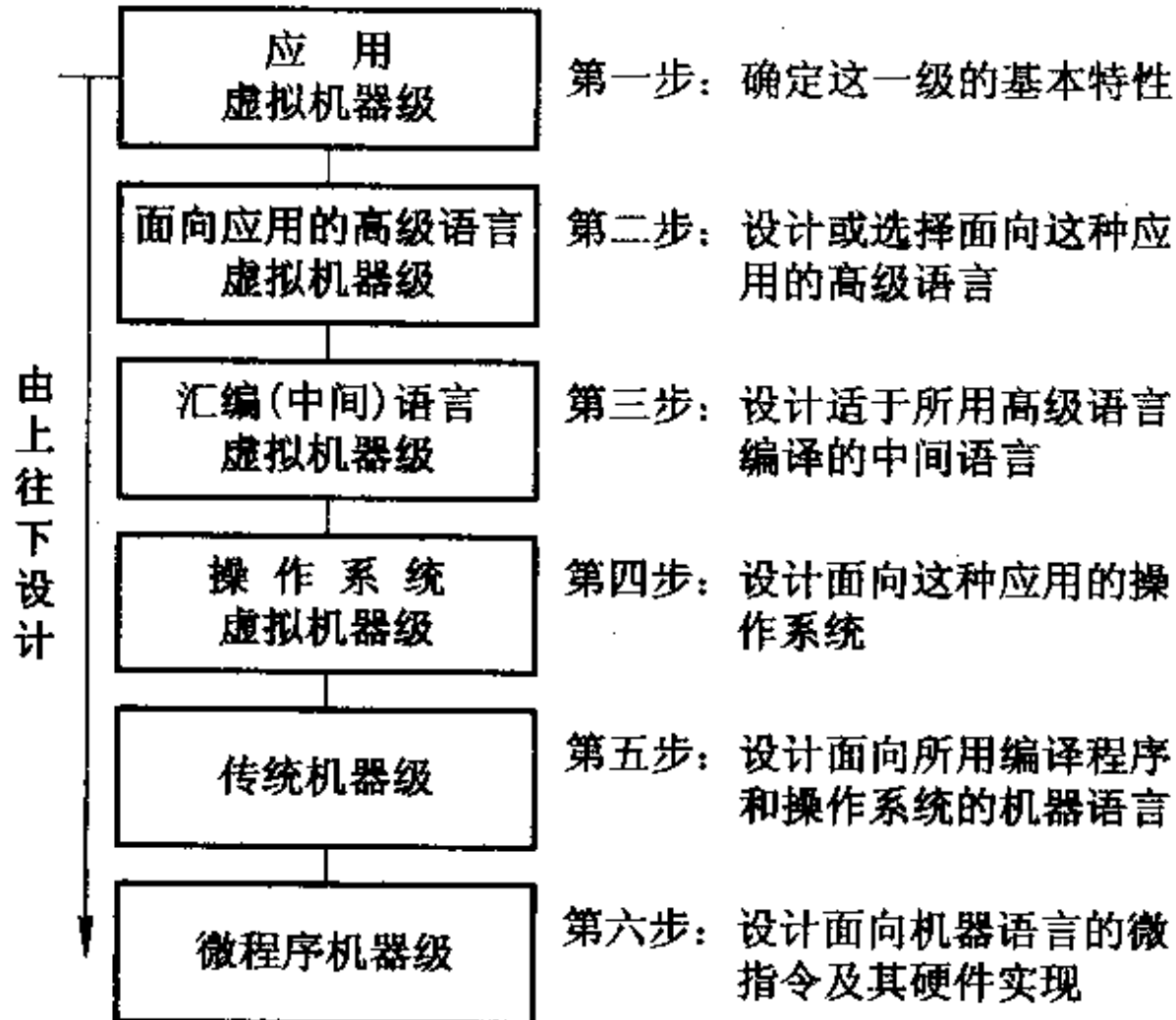
$$\text{性能变化} = \frac{T_{old}}{T_{new}} = \frac{T_{old}}{T_{old} (1 - 50\%) + \frac{50\%T_{old}}{5}} = \frac{1}{1 - 50\% + \frac{50\%}{5}} = 1.66\text{倍}$$

$$\text{成本增加} = \frac{2}{3} * 1 + \frac{1}{3} * (1 + \frac{1}{5}) = 1.07\text{倍}$$

$$\text{成本性能比} = \frac{1.07}{1.66} = 0.64\text{倍}$$

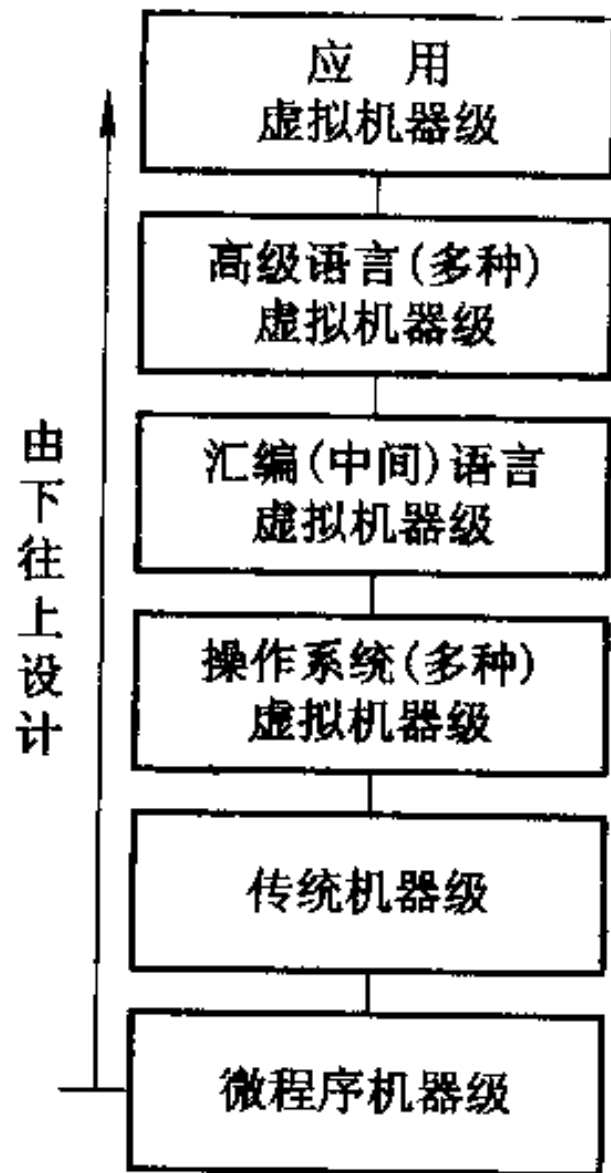
1.3.4 计算机系统设计的主要方法

1、由上往下 (top-down)



- 从使用者面向的机器级开始设计;
- 适用于专用机的设计

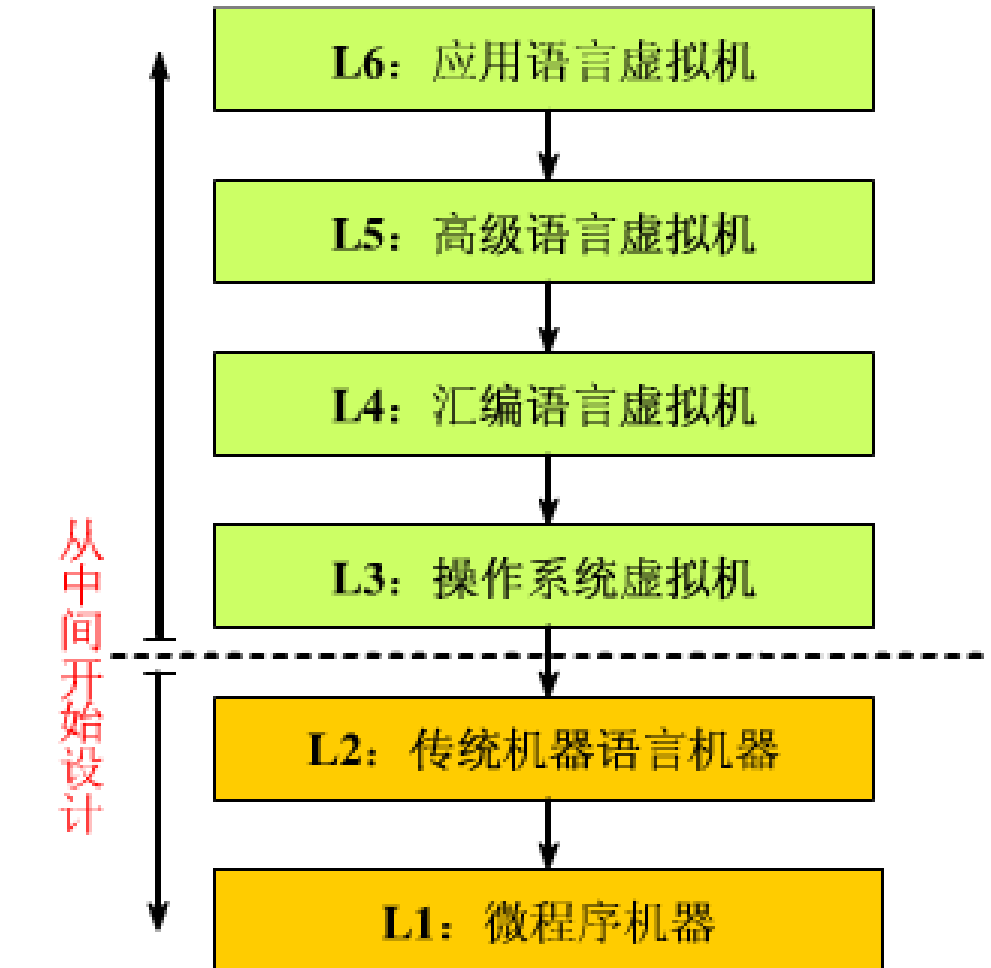
2、由下往上（bottom-top）



- 不管要求、应用需要，只根据现有能得到的器件，构成系统。再配合不同的应用需要，加入操作系统、高级语言等

- 硬件、软件设计会产生脱节，硬件会过于繁杂，造成浪费。

3、从中间开始（middle-out）



● 首先进行软硬件功能分工，确定交界面，再分别向上、下进行设计

1.4 软件等对系统结构的影响

1.4.1 软件对系统结构的影响

软件相对于硬件的成本越来越贵，产量和可靠性的提高越来越困难，软件的排错比编写难。

为此，在系统结构设计时，提出应在新的系统结构上解决好软件的可移植性问题。

□ 软件可移植性的定义

软件产品不用修改或只需经过少量加工，就能由一台机器搬到另一台机器上运行。

▣ 软件可移植性的基本技术

1、统一高级语言

软件的移植种类：系统软件和应用软件

目前有数百种高级语言，用于不同的应用场合，却没有一种能满足各种需要、通用的高级语言，因为：

- 不同的用途，语言的语法结构和语义结构不同
- 对语言的基本结构没有透彻的统一的认识
- 惯性作用

2、系列机思想

系列机应该是多功能的、通用的，按照**统一系统结构**设计的，根据结构生产**软件**，按现代技术**用各种器件实现的**，具有**不同速度、价格和应用场合**的。

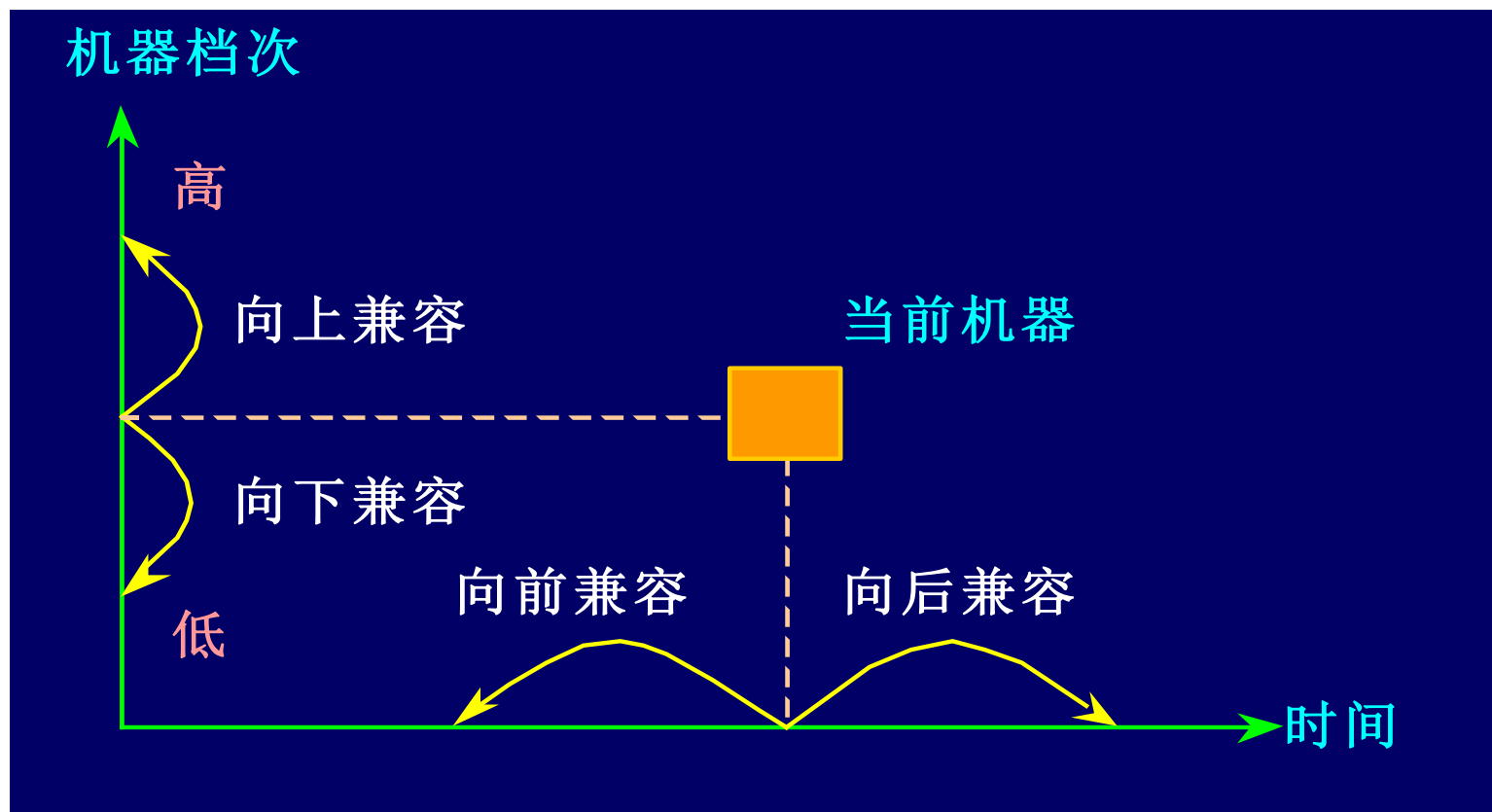
“从中间开始”设计

系列机的特性在于：

- 机器的属性相一致
- 软件的兼容性（应做到向后兼容，并力争向上兼容）

软件运行于**同系列不同型号**的机器上，仅有**运行时间不相同**

向后兼容是软件兼容的根本特征，
也是系列机的根本特征。



系列机思想的优点在于：

- 能够解决程序的兼容问题；
- 使用统一数据结构和指令，便于组成多机系统和网络；
- 扩大应用领域，在同一系列中选用适合的机型；
- 使总线和硬件的接插件通用兼容；
- 有利于机器的使用、维护和人员的培训；
- 有利于系统的更新、换代、升级；
- 有利于提高生产率，降低成本，增加产量。

3、模拟和仿真

为实现不同系统结构的机器之间的机器语言软件移植，就必须做到在一种机器的系统结构上实现另一种机器的系统结构。其中最重要的，就是要**在一种机器上（A）实现另一种机器（B）的指令系统，即另一种机器语言。**

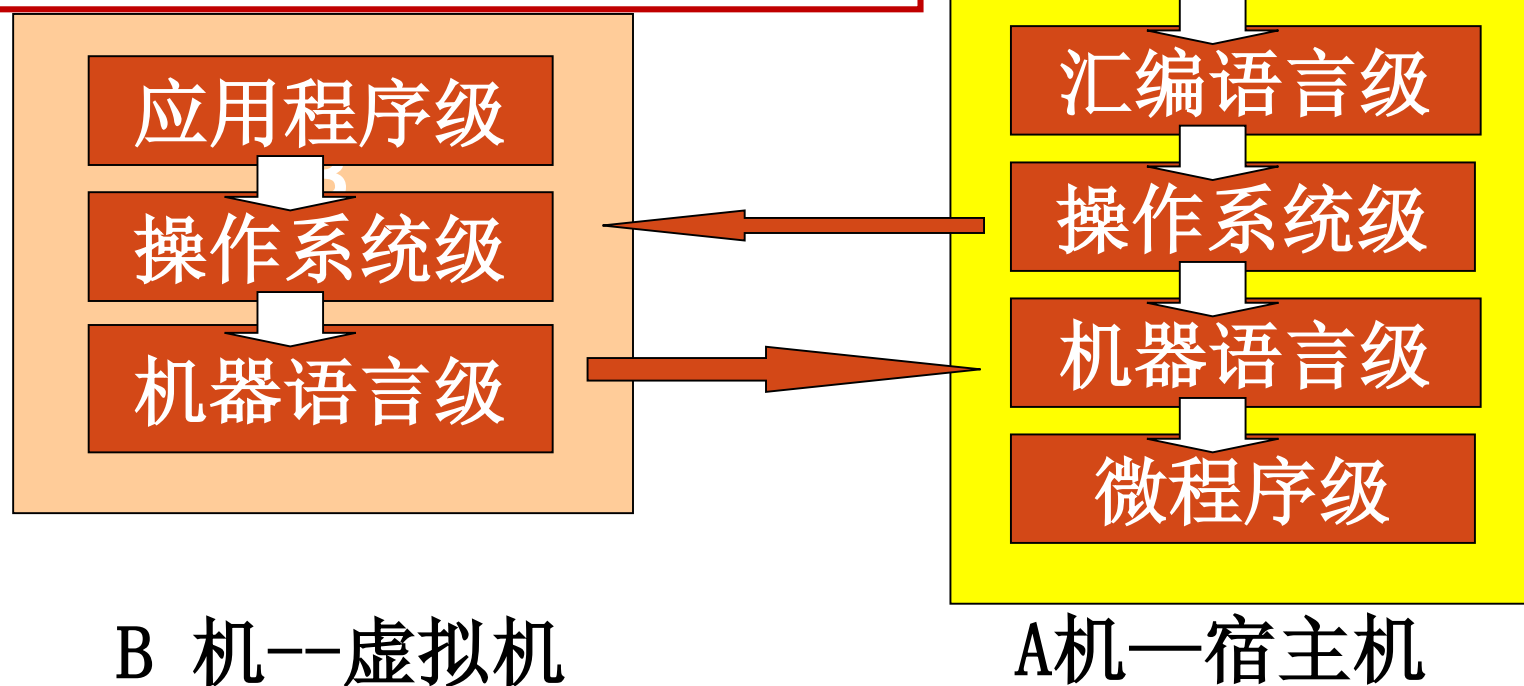


① 模拟

用**机器语言程序**解释，实现程序移植的方法。

即B机器的**每条机器指令**用A机器**一段机器语言程序**解释

将模拟虚拟机的过程看成是宿主机的一道应用程序——**模拟程序**，存于**主存中**



模拟的内容:

机器语言、存储体系、I / O 系统、操作系统

当宿主机中的系统结构和虚拟机系统结构不同，如果用相应的机器语言替代虚拟机的机器指令，完成相同的功能:

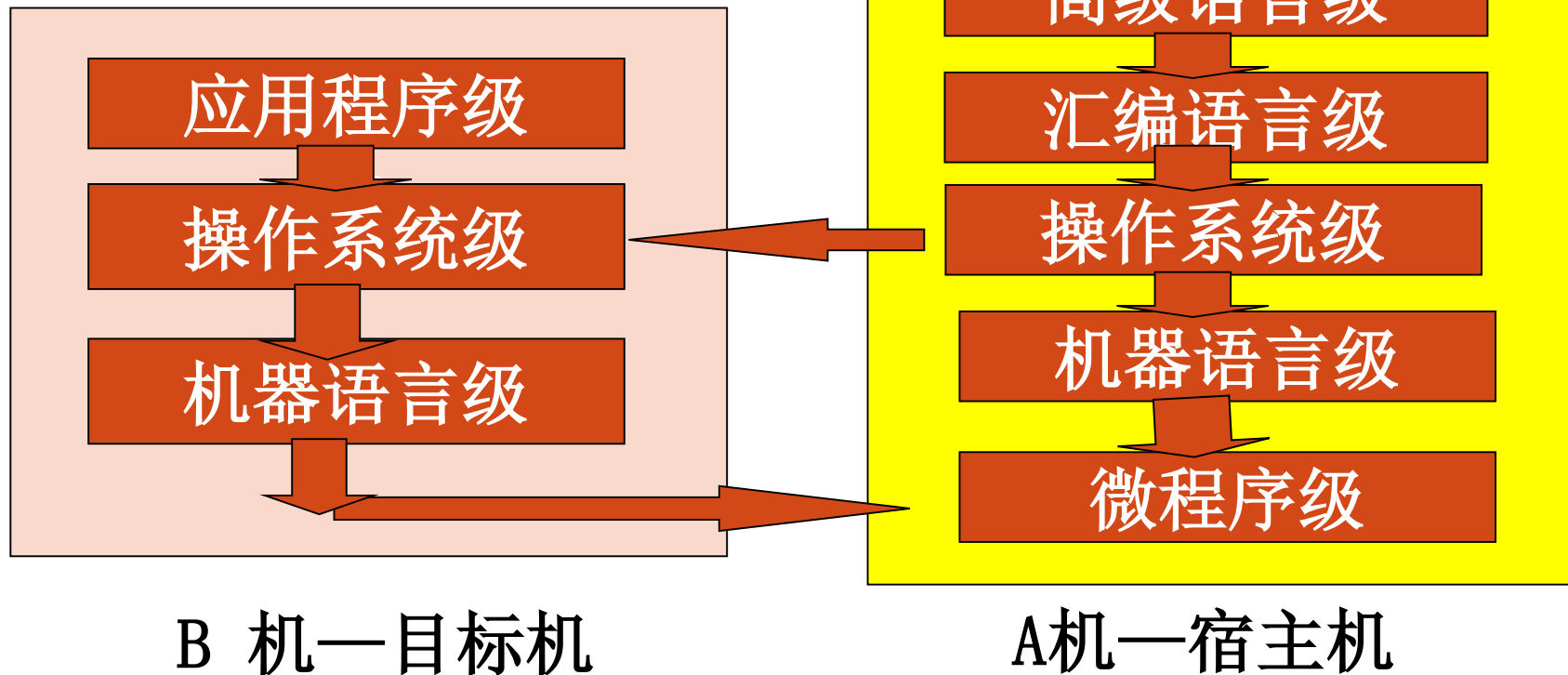
- 一般要用多条指令才能完成某一功能
- 有的功能无法完成的，甚至要很复杂

必须采用新的方法-----仿真

② 仿真

用机器中微程序控制的方法，解释另一台机器指令的系统，即B机器的**每条机器指令**用A机器**一段微程序**解释

为仿真所写的解释程序称为**仿真程序**，存放在**控制存储器**中



仿真的内容:

机器语言、存储体系、I / O 系统、操作系统

仿真的特点:

- 加快运行的速度
- 可完成一些难以实现的操作

仿真 VS 模拟

- 解释程序
- 执行速度
- 适用情况

1.4.2 器件对系统结构的影响

- 器件是推动计算机系统结构不断发展的最活跃的因素
- 摩尔定律
- 计算机的分代主要以器件作为划分标准

1.4.3 应用对系统结构的影响

- 应用需求是推动系统结构发展的原动力
- 不同应用提出了不同的要求
- 基本要求包括高的运算速度、大的存储容量和大得I/O吞吐率

1.5 并行性发展与计算机系统分类

1.5.1 并行性概念

1、并行性的定义与功能

把同时进行运算或操作的特性，称并行性。即：在**同一时刻**或**同一时间间隔内**完成两种或两种以上，性质相同或不同的工作。

开发并行性是为了**提高计算机处理的速度**。包括运算速度、执行速度、数据存取的速度。

并行性可分为：同时性、并发性

同时性： 并发事件在同一时刻发生

并发性： 并发事件在同一时间间隔里发生

2、并行性的等级

并行性可以有不同的等级，而且从不同的角度，等级的分法也不同。

□ 程序执行

指令内部 —— 一条指令内部各个微操作之间的并行

指令之间 —— 多条指令的并行执行

任务、进程之间 —— 多个任务或程序段的并行执行

作业、程序之间 —— 多个作业或多道程序的并行

□ 数据处理

位串字串----只能对一个字中的一位进行处理

指传统的 “串行” 概念

位并字串----同时对一个字中的全部位进行处理

指传统的 “并行” 概念

位串字并----同时对多个字中的同一位进行处理

位并字并----同时对多个字中的多位进行处理

也称 “全并行”

□ **信息加工**，是从各个加工的步骤和阶段来分

存储器操作并行 ----采用单体多字、多体单字、多体多字方式进行存储器访问、读写过程

处理器操作步骤并行 ----处理器内部操作，取指令、分析、执行操作的并行（流水线）

处理器操作并行 ----采用多个处理器，在同一控制器下同时一条指令的多个数据、多个向量并行处理

指令任务、作业并行 ----采用多个处理器，同时对多条指令的多个数据进行处理

3、并行性开发的途径

□ 时间重叠 引入时间因素

让多个处理过程在时间上相互错开，轮流重叠地使用同一套硬件设备的各个部件，以加快部件的周转提高速度。

不增加硬件设备（部件），可以提高性能价格比

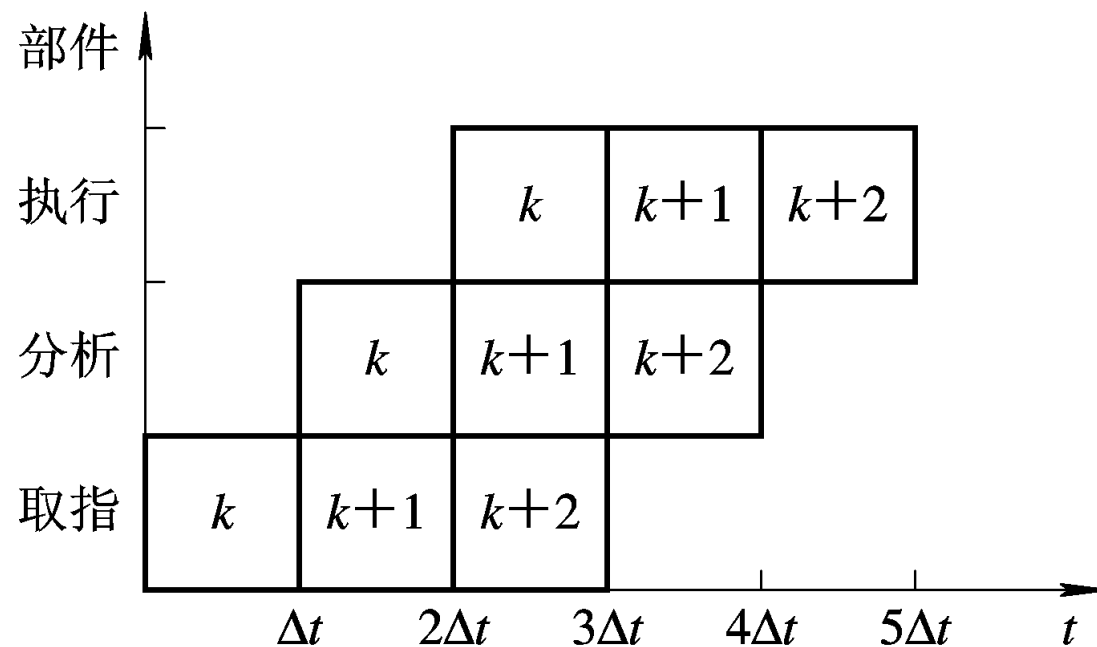
□ 资源重复 引入空间因素

通过重复设置硬件资源，提高性能和可靠性。由同一个控制器，控制多个处理器同时处理同一个数据，进行同个运算。

硬件价格下降了，有利于提高系统的速度和可靠性



(a)

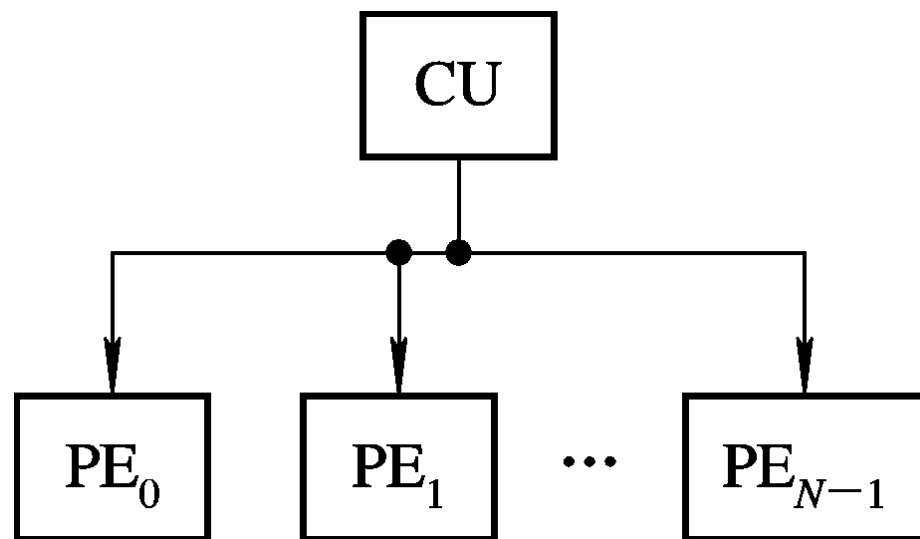


(b)

时间重叠

(a)指令流水线

(b)指令在流水线各部件中流过的时间关系



资源重复

□ 资源共享

用软件的方法，使**多个用户**按一定时间轮流地**分时**使用**同一资源**，以提高利用率，提高整个系统的性能。

资源共享可分为

硬件资源共享：CPU、主存、外设资源

软件资源共享： 软件、信息资源

4、并行的结构

□ 采用时间重叠技术

采用专用化功能部件完成专一功能，各执行过程在时间上重叠起来。把专门化功能部件从指令内部，扩展到指令之间的执行部件，构成专用功能段。采用一条指令对向量的多个元素（多个数据）进行处理，构成的流水线处理机。

由多个不同类型、不同功能的处理机，构成异构型多机系统

□ 采用资源重复技术

关键部件采用重复设置、冗余技术，提高系统速度。采用多操作部件、多存储体，用同样的资源结构，通过重复设置多个相同的处理单元，在一个控制器指挥下，同时对多个数据操作。并行处理机采用阵列结构形式，构成阵列机。

由多个相同类型、相同功能的处理机，形成同构型多机系统

□ 采用资源共享技术

采用多道程序、分时系统在单机上使用形成了虚拟机系统。分时系统适用于多终端、远程终端上使用。

将若干台具有独立功能的处理机（或计算机）相互连接起来，在操作系统控制下，统一协调地运行，最少依赖某一软件、硬件资源，称为分布式处理系统

分时系统 VS. 分布式系统

- 控制方式
- 构成方式
- 并发实质

4、多机系统的耦合度

多机系统包含**多处理机**系统和**多计算机**系统。

多处理机系统	多计算机系统
多个处理器构成	多个计算机构成
一个控制器，共享OS	各自有 独立的 OS
公共主存 ，以 单个数据 形式进行信息交换	有 通信通道 ，以 文件形式 交换信息
处理速度快	处理速度慢
处理 任务级 并行	处理 作业级 并行

多机系统的耦合度指各机间紧密的程度和相互作用的能力。一般可分为最低耦合、松散耦合、紧密耦合三种

最低耦合：仅通过中间存储介质相互通信。
各机器间并无物理连接，也无共享的联机硬件资源。

松散耦合：通过通道或通信线路实现互连。
共享某些外围设备，以较低频带在文件或数据集合一级进行相互通信

紧密耦合：通过总线或高速开关实现互连。
共享主存，有较高的信息传输速率，实现数据传输的吞吐量大，高效率

1.5.2 计算机系统的分类

1、弗林分类法：按指令流、数据流方式进行分类

指令流-----指机器执行的指令序列

数据流-----指指令流调用的数据序列

● **单指令流单数据流** (SISD)

指令部件每次只对一条指令进行译码，并只对一个操作部件分配数据

● **单指令流多数据流** (SIMD)

由单一指令部件控制，按照同一指令流的要求，给它们分配不同的数据。



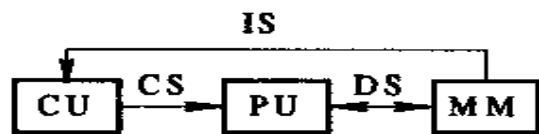
- 多指令流单数据流 (MISD)

有多个处理单元，按多条不同指令的要求对同一个数据流及中间结果进行不同的处理。一个处理单元的输出作为另一个处理单元的输入

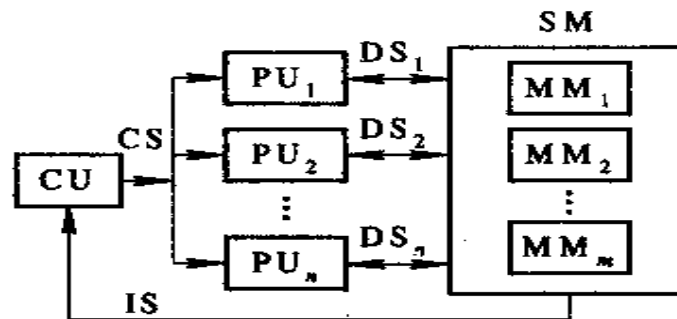
- 多指令流多数据流 (MIMD)

由多个单指令流单数据流的集合构成

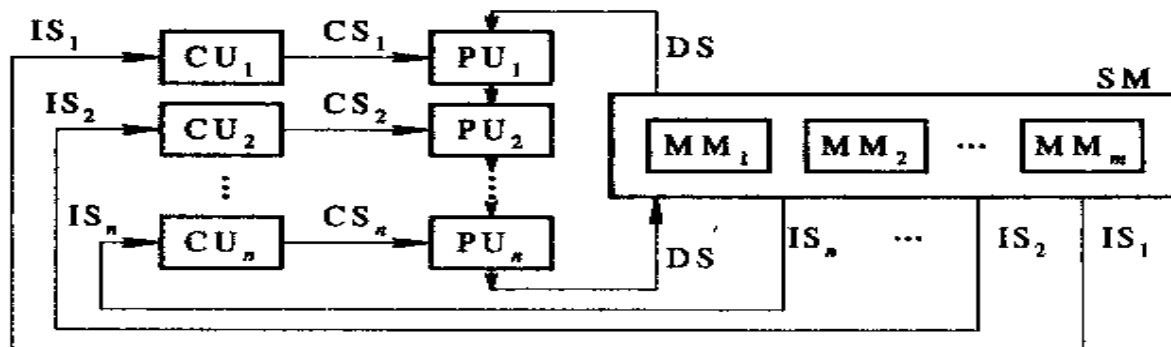




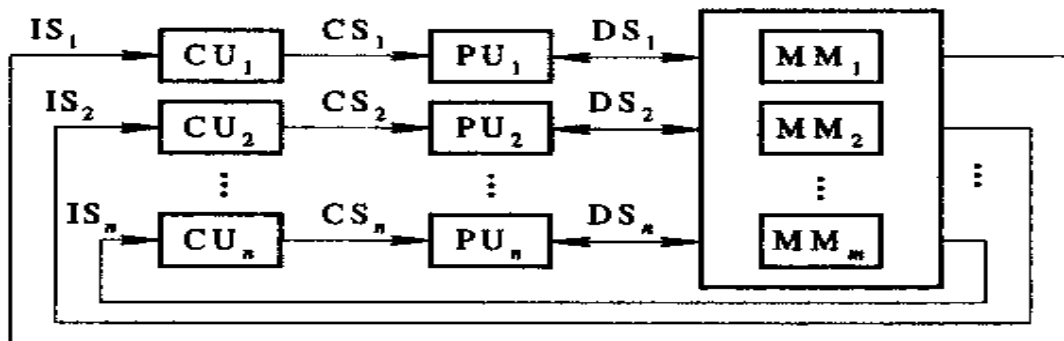
(a) SISD计算机



(b) SIMD计算机



(c) MISD计算机



(d) MIMD计算机



2、库克分类法：按指令流、执行流方式来描述计算机总控制器的结构特征

单指令流单执行流（ SISE ）

-----典型的单处理机系统

单指令流多执行流（ SIME ）

----- 带多操作部件的处理机

多指令流单执行流（ MISE ）

----- 带指令级多道程序的单处理机

多指令流多执行流（ MIME ）

---- 典型的多处理机系统

3、冯泽云分类法：以计算机在单位时间内处理的最大二进制位数

字串位串 (WSBS) ----位串处理方式，每次只处理一个字中的一位，如位串行计算机。

字串位并 (WSBP) -----字（字片）处理方式，每次处理一个字中的 n 位，如传统的位并行单处理机。

字并位串 (WPBS) -----位（位片）处理方式，一次处理 m 个字中的一位，如相联处理机、阵列处理机。

字并位并 (WPBP) -----全并行处理方式，一次处理 m 个字，每个字为 n 位，如阵列处理机、多处理机。

用一台40MHz处理机执行标准测试程序，它含的混合指令数和相应所需的时钟周期数如下：

指令类型	指令条数	时钟周期数
整数运算	45000	1
数据传送	32000	2
浮点运算	15000	2
控制转移	8000	2

求有效CPI、MIPS速率和程序的执行时间。

解：依题意可知 $I_N=10^5$ 条， $n=4$ ，

$$\begin{aligned}CPI &= \sum_{i=1}^n (CPI_i \times \frac{I_i}{IC}) \\&= \sum_{i=1}^4 (1 \times 0.45 + 2 \times 0.32 + 2 \times 0.15 + 2 \times 0.08) \\&= 1.55\end{aligned}$$

$$MIPS = \frac{f_c}{CPI \times 10^6} = \frac{40 \times 10^6}{1.55 \times 10^6} \approx 25.8$$

$$\begin{aligned}T_{CPU} &= IC \times CPI \times T_c \\&= 10^5 \times 1.55 \times \frac{1}{40} \times 10^{-6} = 3.875(ms)\end{aligned}$$

Example: Calculating CPI bottom up

Run benchmark and collect workload characterization (simulate, machine counters, or sampling)

Base Machine (Reg / Reg)

Op	Freq	Cycles	CPI(i)	(% Time)
ALU	50%	1	.5	(33%)
Load	20%	2	.4	(27%)
Store	10%	2	.2	(13%)
Branch	20%	2	.4	(27%)
			1.5	

Typical Mix of
instruction types
in program

Design guideline: Make the common case fast

MIPS 1% rule: only consider adding an instruction if it is shown to add 1% performance improvement on reasonable benchmarks.

Example: Branch Stall Impact

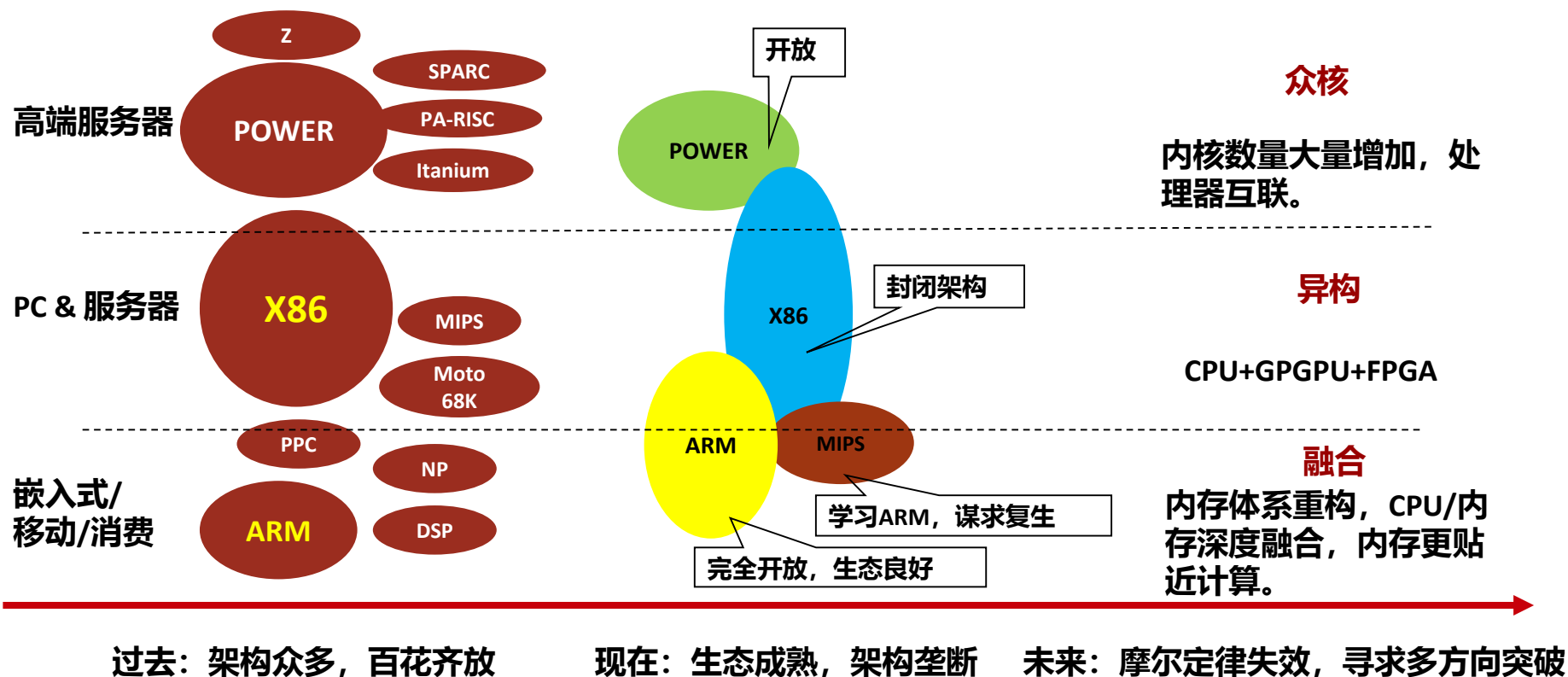
- Assume CPI = 1.0 ignoring branches (ideal)
- Assume solution was stalling for 3 cycles
- If 30% branch, Stall 3 cycles on 30%

Op	Freq	Cycles	CPI(i)	(%Time)
Other	70%	1	.7	(37%)
Branch	30%	4	1.2	(63%)

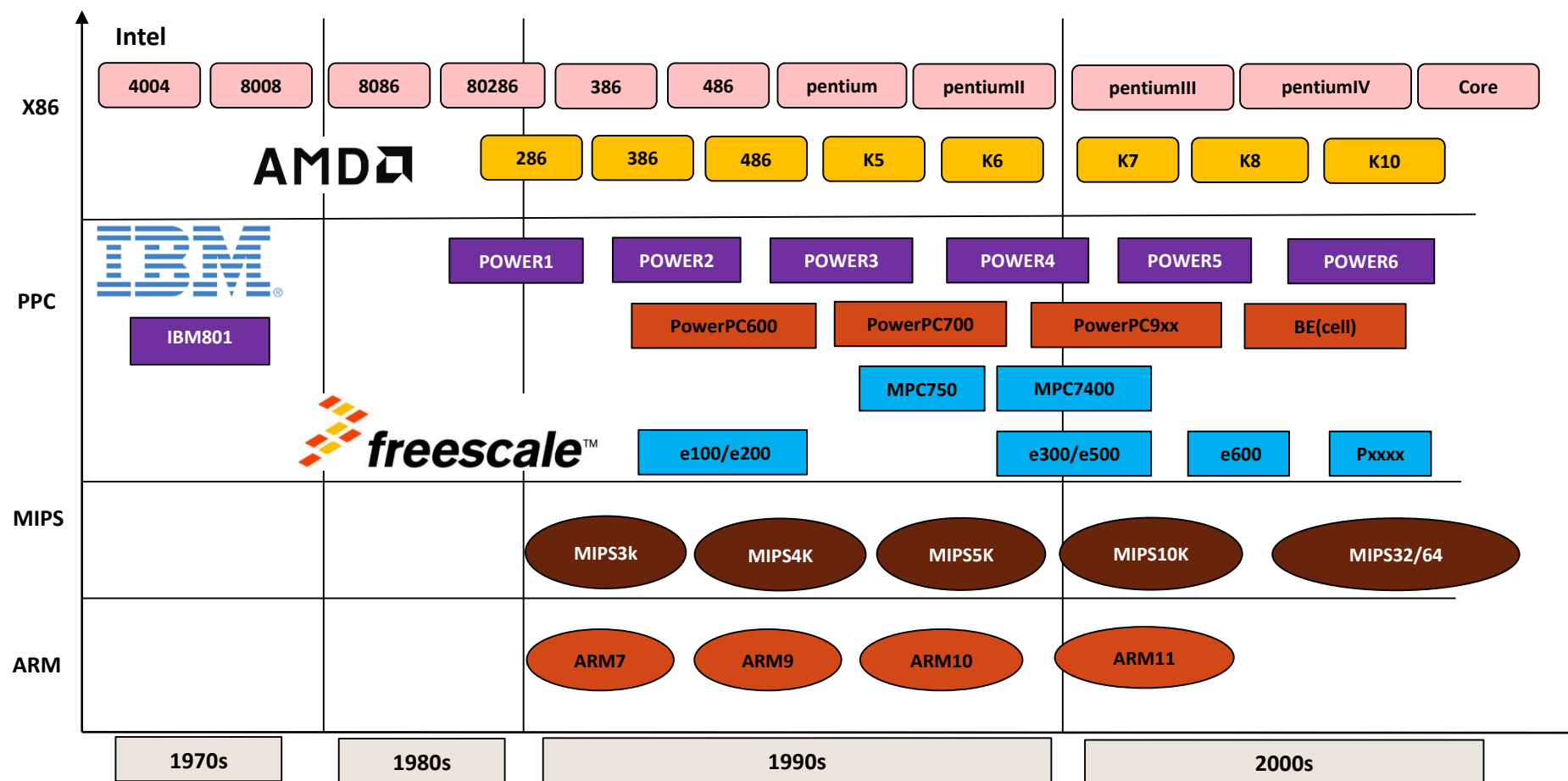
\Rightarrow new CPI = 1.9

- New machine is $1/1.9 = 0.52$ times faster (i.e. slow!)

处理器发展趋势



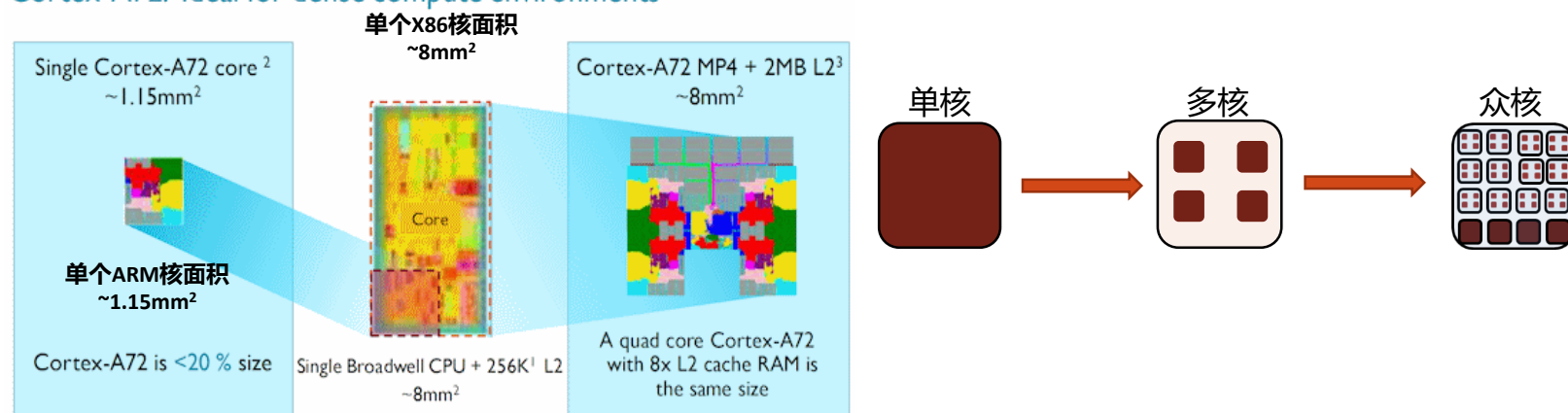
主流CPU发展路径












ARM提供更多计算核心

- 工艺、主频遇到瓶颈后，开始通过增加核数的方式来提升性能；
- 芯片的物理尺寸有限制，不能无限制的增加；
- ARM的众核横向扩展空间优势明显。

Cortex-A72: Ideal for dense compute environments



ARM服务器级别处理器一览

HISILICON		32C, 2.1GHz 16nm		32C,2.4GHz 16nm		48C,3.0GHz 7nm	
CAVIUM		48C,2.5GHz 28nm		32-54C,3.0GHz 14nm			
高通					48Cores,2.2-2.6GHz 14nm		
AMPERE							32C, 3.3GHz 16nm
飞腾		16 Cores,1.6GHz 28nm				64Cores,2.3GHz 16nm	

横轴代表性能

横轴代表性能

ARM公司授权体系

- ARM目前在全球拥有大约1000个授权合作方、320家伙伴，但是购买架构授权的厂家不超过20家，中国有华为、飞腾获得了架构授权。

