北京郵電大學
Beijing University of Posts and Telecommunications

Computer Architecture

计算机系统结构

北京邮电大学
邝 坚

计算机系统结构

参考教材：

1. **计算机系统结构教程(第2版)** 张晨曦 清华大学出版社
2. **Computer Architecture – A Quantitative Approach (Sixth Edition or later)** John L.Hennessy, David A.Patterson, Morgan Kaufmann Publish, 2007-2019
3. **计算机系统结构(第三版)** 白中英 科学出版社

网课资源：

- **NAS + 爱课堂**

课程介绍

- **学习目的**：
  建立计算机系统的完整概念、学习计算机系统的分析方法和设计方法、掌握新型计算机系统的基本结构及其工作原理

- **研究内容**：
  从系统外部研究使用者所关心到的物理计算机的抽象、软硬件功能分配及分界面的确定。

系统结构的相关概念	**流水线技术**
向量处理机	**指令级并行**
互连网络	**多处理机（多核*）**
阵列处理机	**机群系统***

课程介绍

本课程所要具备的知识：

计算机组成原理、计算机操作系统、汇编语言、数据结构、微机原理、高级语言

本课程教学形式、考核方式和考核结构

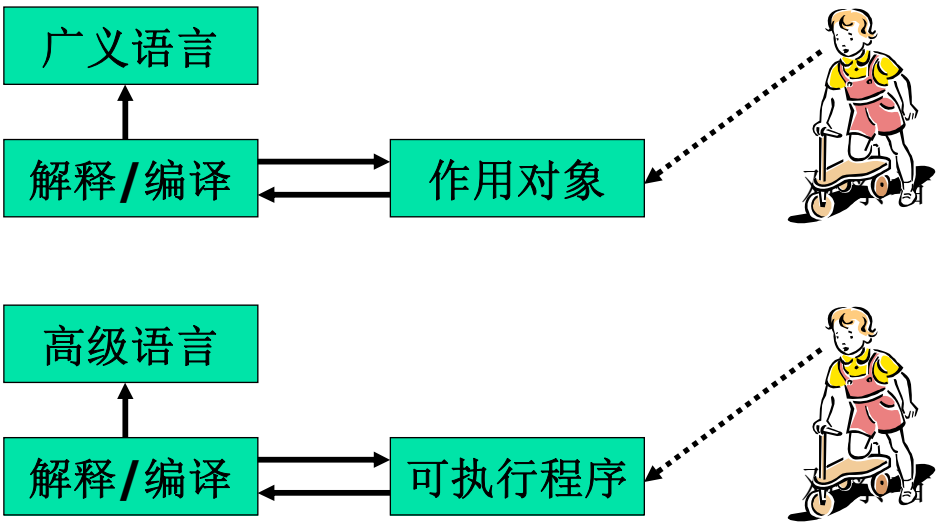
计算机是全国重点学科，是如何设置的？

- 一级学科：计算机科学与技术
- 二级学科：计算机系统结构、计算机软件与理论、计算机应用技术

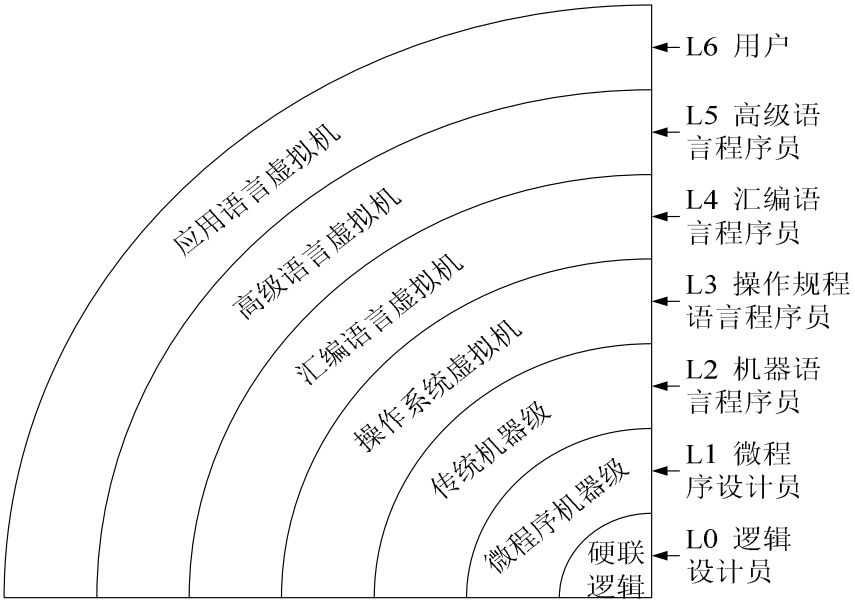
Concepts

虚拟机(Virtual Machine)概念

虚拟机是一个抽象的计算机，由软件实现，并与实际机器一样，都具有指令集并使用不同的存储区域



计算机系统的层次结构



L0和L1级属于计算机组织与实现。

什么是计算机系统结构?

Chapter1- 引言（定义）

计算机体系结构的定义之一:

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, Blaaw, and Brooks, 1964 (IBM 360)

对计算机系统而言，是指那些由程序员可见的系统属性。

所指的程序员:

汇编语言、机器语言、编译程序、操作系统

程序员所看到的:

为了程序能在机器上正确运行所必须了解到的内容

Know what ?

- **数据表示**：硬件能够直接认别和处理的数据类型和格式
- **寻址方式**：最小寻址单位、寻址方式的种类和地址运算等
- **寄存器组织**：操作数寄存器、变址寄存器、控制寄存器及专用寄存器的定义、数量和使用规则等
- **指令系统**：机器指令的操作类型、格式，指令间的排序和控制机制等
- **中断系统**：中断类型、中断级别和中断响应方式等
- **存储系统**：最小编址单位、编址方式、主存容量、最大寻址空间等
- **处理机工作状态**：定义和切换方式，如管态和目态等
- **输入输出系统**：连接方式、数据交换方式、数据交换过程的控制等
- **信息保护**: 包括信息保护方式和硬件对信息保护的支持等

透明性概念

本来存在的事物或属性，从某种角度看似乎不存在

在计算机技术中，把这种本来存在的事物或属性，但从某种角度看又好像不存在的概念称为透明性。

浮点数表示、乘法指令

对高级语言程序员、应用程序员　　透明

对汇编语言程序员、机器语言程序员　不透明

数据总线宽度、微程序

对汇编语言程序员、机器语言程序员　透明

对硬件设计者、计算机维修人员　　不透明

定义

计算机体系结构的定义之二：

计算机系统的软、硬件的界面，即机器语言程序员所看到的传统机器级所具有的属性

—— 计算机系统结构主要研究软硬件功能分配和对软硬件界面的确定

- 计算机系统由软件、硬件和固器组成，它们在功能上是同等的
- 同一种功能可以用硬件实现，也可以用软件或固件实现
- 不同的组成只是性能和价格不同

Computer Organization

计算机组成:指计算机系统结构的逻辑实现

主要包括：

- 确定数据通路的宽度
- 确定各种操作对功能部件的共享程度
- 确定专用的功能部件
- 确定功能部件的并行度
- 设计缓冲和排队策略
- 设计控制机构
- 确定采用何种可靠性技术

Computer Implementation

计算机实现：指计算机组成的物理实现

- 处理机、主存储器等部件的物理结构
- 器件的集成度和速度
- 专用器件的设计
- 器件、模块、插件、底版的划分与连接
- 信号传输技术
- 电源、冷却及装配技术，相关制造工艺及技术等

一种体系结构可以有多种组成。

一种组成可以有多种物理实现。

Are you clear ?

- The difference between **Architecture** with **Organization**
  - refer to the operational units and their interconnections that realize the architecture specifications.

  Example?

- 计算机系统结构、计算机组成和计算机实现是三个不同的概念，但随着技术、器件和应用的发展，三者之间的界限越来越模糊。

计算机设计者所面临的问题

- Determine what attributes are important for a new machine, then design a machine to maximize performance while staying within cost and power constraints.
- 确定用户对计算机系统的功能、价格和性能需求
  - 应用领域
  - 软件兼容
  - OS要求
  - 标准
- 软硬件功能分配
- 生命周期

计算机系统结构的分类

常见的分类方法：

1. 按大小 — 巨型机、大型机、中型机、小型机、微型机等

以性能为表征，按价格划分

划分的标准随时间变化，随时间推移会降低等级

设计方法：

最高性能	特殊用途
最佳性能价格比	一般商用计算机
最低价格	家用计算机

计算机系统结构的分类

2. **按用途** — 科学计算、事务处理、实时控制、工作站、服务器、家用计算机，以及特殊应用的：**嵌入式处理机**、**单片机**等

3. **按数据类型** — 定点机、浮点机、向量机、堆栈机等

4. **按处理机个数和种类** — 单处理机、并行处理机、多处理机、分布处理机、关联处理机、超标量处理机、超流水线处理机、SMP（对称多处理机）、MPP（大规模并行处理机）、机群（Cluster）系统等

5. **按所使用的器件** — 第一代电子管（Valve）、第二代晶体管（Transistor）、第三代集成电路（LSI）、第四代大规模集成电路（VLSI）、第五代（智能计算机）等

按并行度的分类方法

Flynn（佛林）分类法：

主要.

按照指令流和数据流的多倍性特征对计算机系统进行分类

指令流：机器执行的指令序列

数据流：由指令流调用的数据序列，含输入数据和中间结果

多倍性（multiplicity）：在系统性能瓶颈部件上同时处于同一执行阶段的指令或数据的最大可能个数

Flynn 分类法..

按并行度的分类方法

单指令流单数据流 SISD(Single Instruction stream Single Data stream)

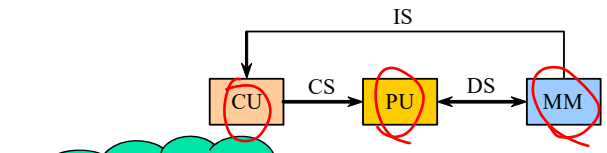
单指令流多数据流 SIMD(Single Instruction stream Multiple Data stream)

多指令流单数据流 MISD(Multiple Instruction stream Single Data stream)

多指令流多数据流 MIMD(Multiple Instruction stream Multiple Data stream)

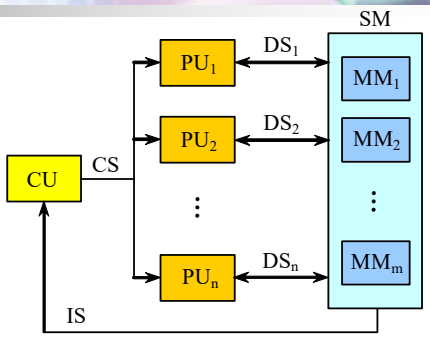
按并行度的分类方法

IS：指令流，DS：数据流，CS：控制流，
CU：控制部件，PU：处理部件，MM和SM：存储器



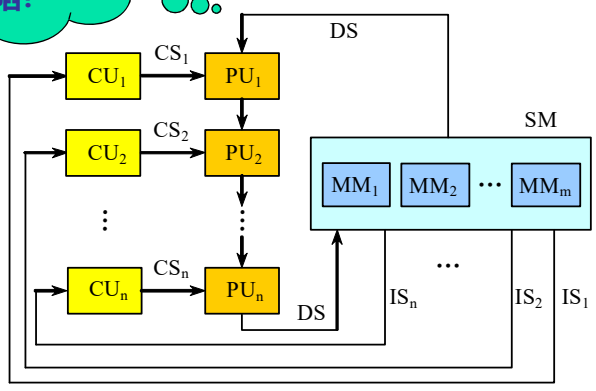
是否都存在?
分类是否有缺陷?

（a）SISD 计算机



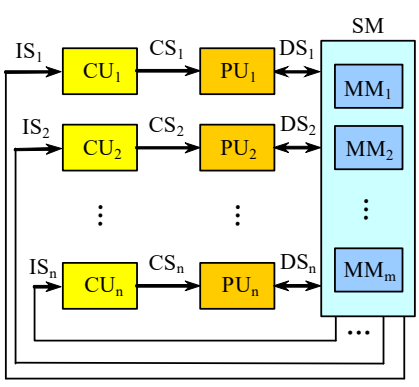
（b）SIMD 计算机

单指令多数据



（c）MISD 计算机

多指令单数据流



（d）MIMD 计算机

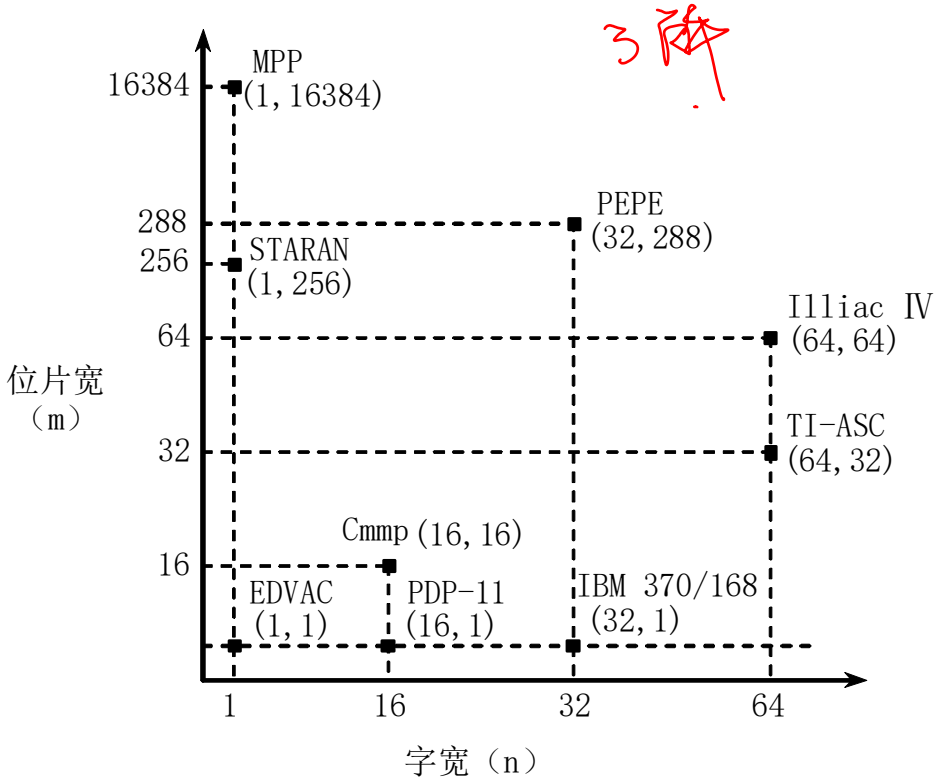
多指令多数据流

按并行度的分类方法

- 冯氏分类法（冯泽云，1972）
  - 用系统的最大并行度对计算机进行分类。
  - 最大并行度：计算机系统在单位时间内能够处理的最大的二进制位数。
    - 用平面直角坐标系中的一个点代表一个计算机系统
    - 其横坐标表示字宽（n位），纵坐标表示一次能同时处理的字数（m字）。m×n就表示了其最大并行度。

按并行度的分类方法

- 字串位串
- 字串位并
- 字并位串
- 字并位并



按并行度的分类方法

Handler分类法

根据并行度和流水线分类，计算机的硬件结构分成三个层次，并分别考虑它们的可并行性和流水处理程度

PCU 程序控制部件的个数k

ALU 算术逻辑部件 (或运算部件PE)的个数d

ELC 每个算数逻辑部件包含基本逻辑线路电路的套数ω

t(系统型号) = (k, d, ω) 或

t(系统型号) = (k x k', d x d', ω x ω')

k':宏流水线中程序控制部件的个数

d':指令流水线中算数逻辑部件的个数

ω':操作流水线中基本逻辑线路的套数

例: Cray-1

1个CPU；12个ALU/PE部件，可最多实现8级流水；字长64位，可实现1~14位流水线处理；

t(Cray-1) = [1, 12 x 8, 64 x (1~14)]

Embedded Immanence

嵌入式微处理器

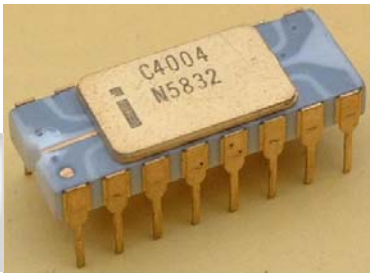
What is Embedded Microprocessor?

A programmable processor whose programming interface is not accessible to end-user of the product.

The only user-interaction is through the actual application.

嵌入式微处理器

- Intel 4004 (1970, 4-bit, 46 instructions)的最初目的是为了应用于计算器 —— 一个嵌入式应用



- 当今的处理器中有95%是用于嵌入式应用(以百亿计/年)
- 通常面向特殊应用
  - Microcontrollers
  - Network & Communication Processors
  - Media Processors
  - DSPs
  - Graphics Processors
  - …

不应该只知道
或只了解X86！

定量分析

Make the Common Case Fast

In making a design trade-off, favor the frequent case over the infrequent case.

大概率事件优先原理（以经常性事件为重点）

- 对于大概率事件(经常性事件)，赋予它优先的处理权和资源使用权，以获得全局的最优结果。

  对经常发生的情况采用优化方法的原则进行选择，以得到更多的总体上的改进。

  优化是指分配更多的资源、达到更高的性能或者分配更多的电能等。

—— 系统结构设计中最常用的思想方法

Example?

❖大概率事件往往比小概率事件简单?

Amdahl's Law（Amdahl定律）

- The performance improvement to be gained from using some faster mode of execution is limited by the fraction of the time the faster mode can be used.

系统中某一部件由于采用某种更快的执行方式后整个系统性能的提高，与这种执行方式的使用频率或占总执行时间的比例有关。

Amdahl定律

$$\text{系统加速比} = \frac{\text{改进后系统性能}}{\text{改进前系统性能}} = \frac{\text{改进前总执行时间}}{\text{改进后总执行时间}}$$

$$Fe = \frac{\text{可改进部分占用的时间}}{\text{改进前整个任务的执行时间}} < 1 \qquad Se = \frac{\text{改进前改进部分的执行时间}}{\text{改进后改进部分的执行时间}} > 1$$

改进后整个任务的执行时间为：$T_n = T_0\left(1 - F_e + \frac{F_e}{S_e}\right)$

其中T_0为改进前的整个任务的执行时间。

未改进部分

改进后

Amdahl定律

改进后整个系统的加速比为： $S_n = \frac{T_0}{T_n} = \frac{1}{(1-F_e)+\frac{F_e}{S_e}}$

Se → ∞时，则 $S_n = \frac{1}{1-Fe}$ ，可改近极限受Fe的约束。

例：将某一部件的处理速度加快10倍，该部件的原处理时间占整个运行时间的40%，整个系统的性能提高多少?

解：Fe=0.4，Se=10，根据Amdahl定律，

$$S_n = \frac{1}{(1-0.4)+\frac{0.4}{10}} = \frac{1}{0.64} \approx 1.56$$

Amdahl定律

- 一种性能改进的递减规则
  - 如果仅仅对计算任务中的一部分做性能改进，则改进得越多，所得到的总体性能的提升就越有限。
- 如果只针对整个任务的一部分进行改进和优化，那么所获得的加速比不超过

  1/（1 - 可改进比例）

CPU性能公式

一个程序所花的CPU时间 $T_{CPU} = N_C \times t = N_C / f$

其中N_C表示CPU时钟周期数

每条指令的平均时钟周期数CPI $CPI = N_C / I_N$

$\therefore T_{CPU} = I_N \times CPI \times t = (I_N \times CPI) / f$

IPC=1/CPI

- 结论：CPU的性能与3个要素有关：

①时钟频率f；② 每条指令所花的时钟周期数 N_C；③ 指令条数I_N。其中时钟频率取决于硬件技术和组织，CPI (Cycles Per Instruction)取决于系统结构组织和指令集，指令数目取决于系统结构的指令集和编译技术。

CPU性能公式

假设：计算机系统有n种指令，计算程序的CPU时钟周期总数：

$$N_C = \sum_{i=1}^{n} (CPI_i \times IC_i)$$

其中IC_i为i指令在程序中执行的的次数；CPI_i为i指令平均时钟周期数；n为指令种类数。

因而有：

$$T_{CPU} = \left[\sum_{i=1}^{n} (CPI_i \times IC_i) \right] \times t$$

$$CPI = \sum_{i=1}^{n} (CPI_i \times IC_i) / IC = \sum_{i=1}^{n} \left(CPI_i \times \frac{IC_i}{IC} \right)$$

其中IC_i/IC为i指令在程序中所占的比例。

CPU性能公式

- 考虑条件分支指令的两种不同设计方法：
  - （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。
  - 在这两种CPU中，条件分支指令都占用2个时钟周期，而所有其他指令占用1个时钟周期。对于CPU$_A$，执行的指令中分支指令占20%；由于每条分支指令之前都需要有比较指令，因此比较指令也占20%。由于CPU$_B$在分支时不需要比较，因此CPU$_B$的时钟周期时间是CPU$_A$的1.25倍。问：哪一个CPU更快？
  - 问：如果CPU$_B$的时钟周期时间只是CPU$_A$的1.1倍，哪一个CPU更快呢？

CPU性能公式

解：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$_A$性能为

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

根据假设，有

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

CPU性能公式

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

$$CPI_B = 0.25 \times 2 + 0.75 \times 1 = 1.25$$

因为CPU_B不执行比较，故

$$IC_B = 0.8 \times IC_A$$

什么鬼题.

因此CPU_B性能为

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

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。

CPU性能公式

如果CPU_B的时钟周期时间仅仅是CPU_A的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

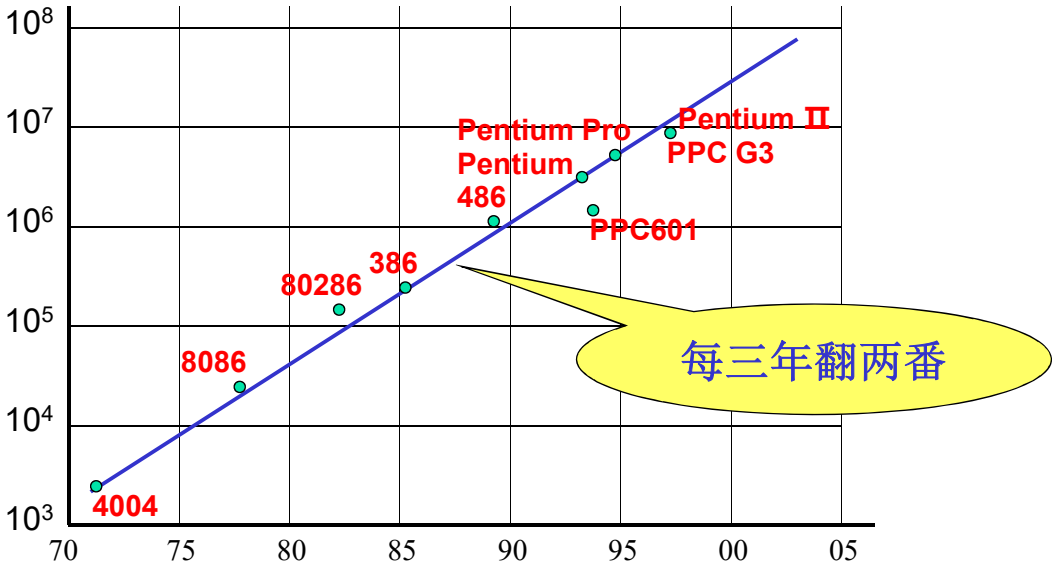
$$\text{总CPU时间B} = IC_B \times CPI_B \times \text{时钟周期}_B$$
$$= 0.8 \times IC_A \times 1.25 \times (1.10 \times \text{时钟周期}_A)$$
$$= 1.10 \times IC_A \times \text{时钟周期}_A$$

因此CPU_B由于执行更少指令条数，比CPU_A运行更快。

计算机性能的发展——摩尔定律

高登·摩尔(Gordon Moore) —— Intel的创始人之一

断言芯片的晶体管数量每年翻一番，而且这种事态将在不远的将来仍继续下去（1965年）



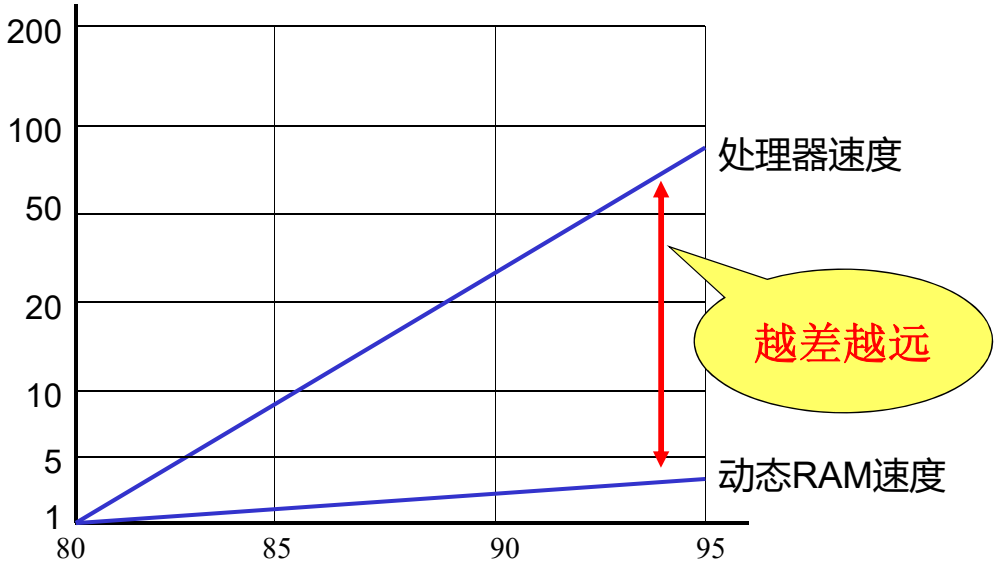
摩尔定律

技术工艺发展趋势：

项目	速度(响应时间)
Logic	每三年两倍
DRAM	每十年两倍
DISK	每十年两倍

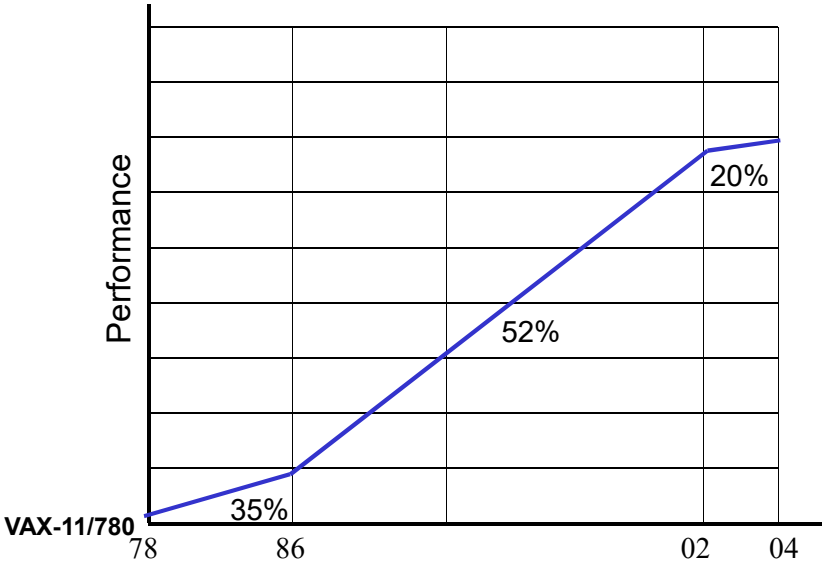
这中间会出什么问题?

性能平衡问题



当处理器的性能以惊人的速度向前发展的时候，其他部件（包括I/O系统）的速度没有跟上——寻求**性能平衡**的需要：调整组织与结构

CPU Performance



CPU Performance

时　间	原　因	每年的性能增长
1946年起的25年	两种因素都起着主要的作用	25%
20世纪70年代末—80年代初	大规模集成电路和微处理器出现，以集成电路为代表的制造技术的发展	约35%
20世纪80年代中开始	RISC结构的出现，系统结构不断更新和变革，制造技术不断发展	50%以上 维持了约16年
2002年以来	功耗问题（已经很大）。 可以进一步有效开发的指令级并行性已经很少。 存储器访问速度的提高缓慢。	约20%

性能评测(Performance Evaluation)

性能评测的一个重点问题是对性能**量度**的选择。
对系统设计者来说，**执行时间**和**吞吐率**是两个重要的性能量度值。

1. **响应时间（response time）**，也可称为**执行时间（execution time）**，即事件从发生到结束所花费的时间。

“甲机器比乙机器快n倍”的含义应该是：

$$\frac{\text{机器性能（甲）}}{\text{机器性能（乙）}} = \frac{\text{执行时间（乙）}}{\text{执行时间（甲）}} = n$$

性能评测

- 执行时间可以有多种定义：
  - 计算机完成某一任务所花费的全部时间，包括磁盘访问、存储器访问、输入/输出、操作系统开销等。
  - CPU时间：CPU执行所给定的程序所花费的时间，不包含I/O等待时间以及运行其他程序的时间。
    - 用户CPU时间：用户程序所耗费的CPU时间。
    - 系统CPU时间：用户程序运行期间操作系统耗费的CPU时间。

性能评测

2. 对用于事务处理的计算机系统进行性能评价时，有一个常用的词是“**吞吐率（throughput）**”。如果我们说“甲的吞吐率是乙的2倍”，那么则意味着在每个时间单元内，甲完成任务的数量是乙的2倍。

通常用每分钟的事务**处理量（Transactions Per Minute - tpm）**来表述事务处理系统的性能。

性能评测

MIPS和MFLOPS的不足之处?

指令执行速度：

MIPS(Million Instructions Per Second), GIPS、TIPS

$$MIPS = \frac{\text{指令条数}}{\text{执行时间} \times 10^6} = \frac{Fz}{CPI} = IPC \times Fz$$

其中：Fz为处理机的工作主频。

MFLOPS(Million Floating Point Operations Per Second),

每秒百万次浮点运算。(GFLOPS、TFLOPS）

$$MFLOPS = \frac{\text{浮点运算次数}}{\text{执行时间} \times 10^6}$$

性能评测

- 基准测试程序（Benchmark）
  - 用于测试和比较性能的基准测试程序的最佳选择是真实应用程序（例如编译器）。
  - 以前常采用简化了的程序，例如：
    - 核心测试程序：从真实程序中选出的关键代码段构成的小程序。
    - 小测试程序：简单的只有几十行的小程序。
    - 合成的测试程序：人工合成出来的程序。
      Whetstone与Dhrystone是最流行的合成测试程序

性能评测

Whetstone

用FORTRAN语言编写的综合性测试程序，主要包括浮点运算、整数算术运算、功能调用、数组变址、条件转移、超越函数。结果用KMIPS表示

Dhrystone - 整数测试程序

用C语言编写，100条语句。包括各种赋值语句、数据类型和数据区、控制语句、过程调用和参数传送、整数运算和逻辑操作。

Linpack -浮点测试程序

用FORTRAN语言编写，主要是浮点加法和浮点乘法操作。用MFLOPS、 GFLOPS 及TFLOPS表示。-

性能评测

- 从测试性能的角度来看，上述测试程序就不可信了。原因：
  - 这些程序比较小，具有片面性；
  - 系统结构设计者和编译器的设计者可以“合谋”把他们的计算机面向这些测试程序进行优化设计，使得该计算机显得性能更高。
- 性能测试的结果除了和采用什么测试程序有关以外，还和在什么条件下进行测试有关。
- 基准测试程序设计者对制造商的要求
  - 采用同一种编译器；
  - 对同一种语言的程序都采用相同的一组编译标志。

性能评测

- 基准测试程序套件：
  - 由各种不同的真实应用程序构成，能比较全面地反映计算机在各个方面的处理性能
- SPEC系列：最成功和最常见的测试程序套件（美国的标准性能评估公司开发，http://www.spec.org）
  - SPEC89：　　10个程序
  - SPEC92：　　20个程序
  - SPEC95：　　18个程序
  - SPEC2000：　　26个程序
  - SPEC CPU2006：　　29个程序
  - SPEC CPU2017：　　43个程序

性能评测

- SPEC CPU2006
  - 整数程序12个（CINT2006）
    - 9个是用C写的，3个是用C++写的
  - 浮点程序17个（CFP2006）
    - 6个是用FORTRAN写的，4个是用C++写的，3个是用C写的，4个是用C和FORTRAN混合编写的。
- SPEC CPU2017（4 suites）
  - SPECspeed®2017 Integer(10)
  - SPECspeed®2017 Floating Point(10)
  - SPECrate®2017 Integer(10)
  - SPECrate®2017 Floating Point(13)

性能评测

- SPEC测试程序套件中的其他一系列测试程序组件
  - SPECSFS：用于NFS（网络文件系统）文件服务器的测试程序。它不仅测试处理器的性能，而且测试I/O系统的性能。它重点测试吞吐率
  - SPECWeb：Web服务器测试程序
  - SPECviewperf：用于测试图形系统支持OpenGL库的性能
  - SPECapc：用于测试图形密集型应用的性能

性能评测

- **事务处理（TP）性能测试基准程序：**
  - 用于测试计算机在事务处理方面的能力，包括数据库访问和更新等。
  - 20世纪80年代中期，一些工程师成立了称为TPC的独立组织。目的是开发用于TP性能测试的真实而又公平的基准程序。
  - 先后发布了多个版本，主要是用于测试服务器的性能：
    - TPC-A、TPC-C、TPC-H、TPC-W、TPC-App等

性能评测

EEMBC系列- EEMBC（EDN Embedded Microprocessor Benchmark Consortium / EDN嵌入式微处理器基准测试联盟）

建立于1997年4月。EEMBC系列基准测试程序面向嵌入式处理器。由于联盟主要由半导体工业中的主流企业组成，因此发布的相关内容自然成为实际的工业标准。其内容涵盖汽车工业、消费电子、数字娱乐、JAVA、网络、办公室自动化和电信市场，应用包括引擎控制、数码相机、打印机、蜂窝电话、调制解调器等包含嵌入式处理器的系统，以数十个独立的算法程序组成的套件。

http://www.eembc.org/

性能评测

- 性能比较
  - 总执行时间/平均执行时间/加权执行时间
  - 调和平均
  - 几何平均

	A机	B机	C机	W（1）	W（2）	W（3）
程序1	1.00	10.00	20.00	0.50	0.909	0.999
程序2	1000.00	10.00	20.00	0.50	0.091	0.001
加权算术平均值A_m（1）	500.50	10.00	20.00			
加权算术平均值A_m（2）	91.91	10.00	20.00			
加权算术平均值A_m（3）	2.00	10.00	20.00			

系统结构的发展

- 1950年以来，在计算机结构和组织领域中的几个真正变革：
  - Family Concept(系列计概念)
  - Microprogrammed Control Unit(微程序控制单元)
  - Cache Memory
  - Pipelining(流水)
  - Multiple Processors(多处理机)
  - Reduced Instruction Set Computer(RISC)
  - Multi-Core
  - …

系统结构的发展

问题：

设计Cache机制的原因和依据是什么？

Principle of Locality

Programs tend to reuse data and instructions they have used recently.

程序访存的局部性

•实验统计表明：一个程序用90%的执行时间去执行仅占10%的程序代码。

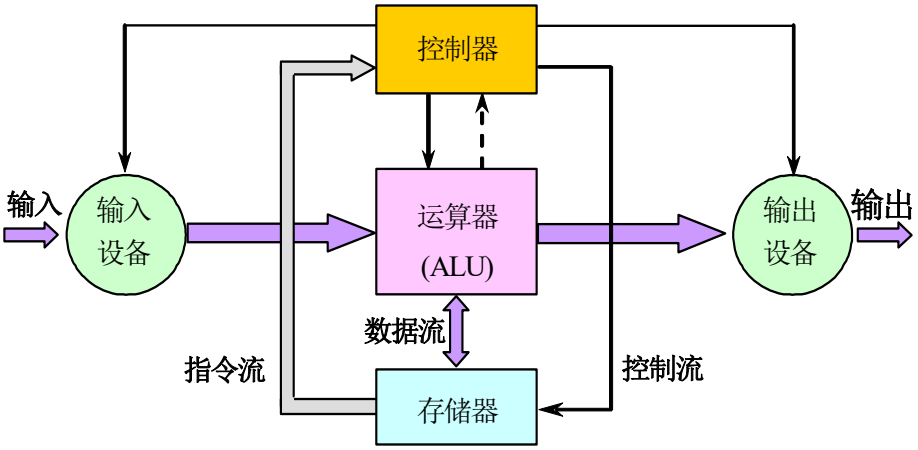
数据访问的局部性

- 空间上的局部性(Spatial locality)
- 时间上的局部性(Temporal locality)

如何利用?

计算机系统结构的发展

- 冯·诺依曼（Van Nenmann）结构 — 1936～46年形成，冯·诺依曼等人于1946年提出
- 存储程序计算机的结构 -指令驱动



冯·诺依曼结构特点

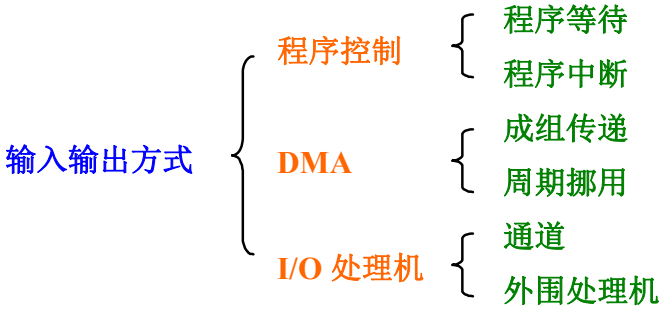
- 以运算器为中心
- 在存储器中，指令和数据同等对待
  - 指令和数据一样可以进行运算，即由指令组成的程序是可以修改的。
- 存储器是按地址访问、按顺序线性编址的一维结构，每个单元的位数是固定的
- 指令的执行是顺序的
  - 一般是按照指令在存储器中存放的顺序执行。
  - 程序的分支由转移指令实现。
  - 由指令计数器PC指明当前正在执行的指令在存储器中的地址。
- 指令由操作码和地址码组成
- 指令和数据均以二进制编码表示，采用二进制运算

对系统结构进行的改进

- 改进：存储程序，存储器为中心，分散控制
- 非冯计算机
  - 数据驱动型，出现了数据流机计算机
  - 需求驱动型，出现各种图归约计算机
  - 处理非数值化信息的智能计算机，例如自然语言、声音、图形和图象处理、虚拟现实处理等。
  - 第五代计算机，由推理机和知识库机等组成
  - 神经网络计算机，脉动式计算机，仿生计算机，……

对系统结构进行的改进

- 输入/输出方式的改进



- 采用并行处理技术
  - 如何挖掘传统机器中的并行性？
  - 在不同的级别采用并行技术。
    - 例如，微操作级、指令级、线程级、进程级、任务级等

对系统结构进行的改进

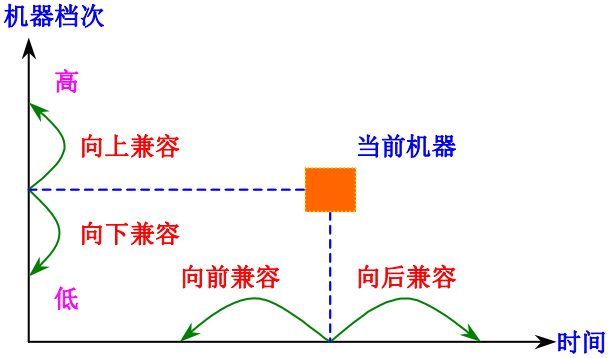
- 存储器组织结构的发展
  - 相联存储器与相联处理机
  - 通用寄存器组
  - 高速缓冲存储器Cache
- 指令集的发展
  - 复杂指令集计算机（CISC）
  - 精减指令集计算机（RISC）

软件对系统结构的影响

- 软件的可移植性
  - 一个软件可以不经修改或者只需少量修改就可以由一台计算机移植到另一台计算机上正确地运行。差别只是执行时间的不同。我们称这两台计算机是软件兼容的。
- 实现可移植性的常用方法
  - 系列机
  - 模拟与仿真
  - 统一高级语言 。

软件对系统结构的影响

- 系列机
  - 具有相同的系统结构，但具有不同组成和实现的一系列不同型号的机器。
    - 较好地解决软件开发要求系统结构相对稳定与器件、硬件技术迅速发展的矛盾。
  - 系列机的软件兼容 - 向后兼容是系列机的根本特征



软件对系统结构的影响

- 模拟和仿真
  - 使软件能在具有不同系统结构的机器之间相互移植。
    - 在一种系统结构上实现另一种系统结构。
    - 从指令集的角度来看，就是要在一种机器上实现另一种机器的指令集。
  - 模拟：用软件的方法在一台现有的机器（宿主机）上实现另一台机器（虚拟机）的指令集。
    - 通常用解释的方法来实现。
    - 运行速度较慢，性能较差。
  - 仿真：用一台现有机器（宿主机）上的微程序去解释实现另一台机器（目标机）的指令集。
    - 运行速度比模拟方法的快
    - 仿真只能在系统结构差距不大的机器之间使用

软件对系统结构的影响

- 统一高级语言
  - 实现软件移植的一种理想的方法(JAVA?)
  - 较难实现

器件发展对系统结构的影响

- 推动计算机系统结构发展最活跃的因素
  - 系统结构和组成技术的新发展和新思维是否能用的上，最终取决于器件
    - 可靠性在数量级的提高 – 流水线技术
    - 高速+廉价存储器 – Cache/虚存
    - PROM/EPROM – 微程序技术的广泛使用
  - 摩尔定律
  - 量子计算

应用对系统结构的影响

- 不同的应用对计算机系统结构的设计提出了不同的要求
- 应用需求是促使计算机系统结构发展的最根本的动力
- 一些特殊领域：需要高性能的系统结构
  - 高结构化的数值计算
    - 气象模型、流体动力学、有限元分析
  - 非结构化的数值计算
    - 蒙特卡洛模拟、稀疏矩阵
  - 实时多因素问题
    - 语音识别、图像处理、计算机视觉
  - 大存储容量和输入输出密集的问题
    - 数据库系统、事务处理系统
  - 图形学和设计问题
    - 计算机辅助设计
  - 人工智能
    - 面向知识的系统、推理系统、自然语言处理等

X86 review

1. 8080：世界上第一个通用微处理器。8位。曾用于第一台个人计算机Altair。
2. 8086：16位。8086 开辟了指令高速缓存或称队列，在指令被实际执行之前能预取几条指令。变形形式是8088，曾用于最初的IBMPC机，并确保了Intel的成功。
3. 80286：8086扩展产品，可寻址16MB存储区间，不再受IMB存储区间的限制。
4. 80386：Intel的第一个32位处理器，一个有重大改进的产品。32位的结构，可以与几年前的小型机和大型机相抗衡。也是Intel的第一个支持多任务的处理器。
5. 80486：采用更为复杂、功能更强的Cache和指令流水线技术。内置数字协处理器，减轻了主CPU复杂算术运算的负担。

X86 review

6. Pentium：Intel引入了超标量(Superscalar)技术，允许更多的指令并行执行。
7. PentiumPro：继续推进由Pentium开始的超标量结构，极富进取性地使用了包括寄存器重命名、转移预测、数据流分析、推测执行等技术。
8. Pentium II：融入了专门用于有效处理视频、音频和图形数据的Intel MMX技术。
9. Pentium III：融人了新的浮点指令，以支持三维图形软件。
10. Pentium IV：….。

“新型”系统结构

- 超标量处理机、超流水线处理机、向量处理机
- 并行处理机
  - ✓ SIMD
  - ✓ 并行向量处理机PVP(Parallel Vector Processor)
  - ✓ 对称多处理机SMP(Symmetric Multiprocessor)
  - ✓ 大规模并行处理机MPP(Massively Parallel Processor)
  - ✓ 工作站机群COW(Cluster of Workstation)
  - ✓ 分布共享DSM(Distributed Shered Memory)多处理机

并行处理机

并行处理 - 高等计算机的核心技术

所涉及的主要问题：

互联网络与通信

粒度划分与调度

并行存储器系统

Cache一致性问题

存储器一致性问题

指令的并行性

计算机系统结构中并行性的发展

并行性概念：计算机系统所具有的能在同一时刻或同一时间间隔进行多种运算或操作的特性

- **同时性(simultaneity)**：两个或两个以上的事件在同一时刻发生
- **并发性(concurrency)**：两个或两个以上的事件在同一时间间隔发生

并行处理是系统结构、硬件、软件、算法、语言等多方面综合研究的领域

数据处理的并行性 - 从低到高：

字串位串（无并行） **字串位并（初步并行）**
字并位串（较高并行） **全并行（更高并行）**

程序执行的并行性 - 等级从低到高：

指令内部并行（各微操作之间） **指令级并行**（ILP，流水线）
线程级并行（TLP，结合多核，通常以进程内派生的多个线程为调度单位）
任务级/过程级并行（以子程序/进程为单元）
作业/程序级并行（单机系统通过软件，多机系统大多通过硬件）

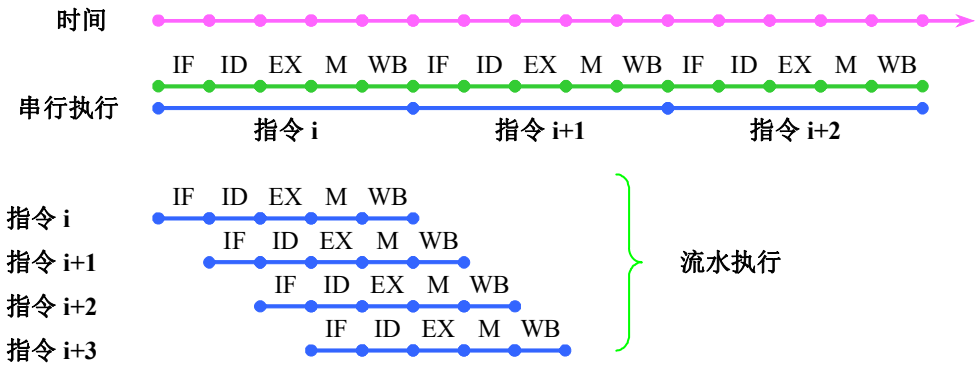
并行性

提高并行性的技术途径：

- **时间重叠(Time Interleaving)** - 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分，以加快硬件周转而赢得速度
- **资源重复(Resource Replication)** - 引入空间因素，以数量取胜。通过重复设置硬件资源，大幅度地提高计算机系统的性能
- **资源共享(Resource Sharing)** - 一种软件方法，它使多个任务按一定时间顺序轮流使用同一套硬件设备

单机系统中并行性的发展

- 起主导作用的是时间重叠原理。
  - 实现时间重叠的基础：部件功能专用化
    - 把一件工作按功能分割为若干相互联系的部分；
    - 把每一部分指定给专门的部件完成；
    - 按时间重叠原理把各部分的执行过程在时间上重叠起来，使所有部件依次分工完成一组同样的工作。
  - 如指令流水线



单机系统中并行性的发展

- 资源重复原理的运用也已经十分普遍
  - 多体存储器
  - 多操作部件
    - 通用部件被分解成若干个专用部件，如加法部件、乘法部件、除法部件、逻辑运算部件等，而且同一种部件也可以重复设置多个。
    - 只要指令所需的操作部件空闲，就可以开始执行这条指令（如果操作数已准备好的话）。
    - 这实现了指令级并行。
  - 阵列处理机
    - 更进一步，设置许多相同的处理单元，让它们在同一个控制器的指挥下，按照同一条指令的要求，对向量或数组的各元素同时进行同一操作，就形成了阵列处理机 。
- 在单处理机中，资源共享的概念实质上是用单处理机模拟多处理机的功能，形成所谓虚拟机的概念。

多机系统中并行性的发展

- 多机系统遵循时间重叠、资源重复、资源共享原理，发展为3种不同的多处理机：
  - 异构(Heterogeneous)型多处理机
  - 同构(Homogeneous)型多处理机
  - 分布式系统

多机系统中并行性的发展

- 耦合度

  反映多机系统中各机器之间物理连接的紧密程度和交互作用能力的强弱。
  - **紧耦合(Tightly Coupled)/直接耦合(Directly Coupled)**：物理连接频带高，总线/高速开关互联，可共享主存；
  - **松耦合(Loosely Coupled)/间接耦合(Indirectly Coupled)**：通道/通信线互联，可共享外存/外设。机器之间的相互作用是在文件或数据集一级上进行的。

多机系统中并行性的发展

- 功能专用化（实现时间重叠 ）
  - 专用外围处理机
    - 例如，输入/输出功能的分离。
  - 专用处理机
    - 如数组运算、高级语言翻译、数据库管理等，分离出来。
  - 异构型多处理机系统
    - 由多个不同类型、至少担负不同功能的处理机组成，它们按照作业要求的顺序，利用时间重叠原理，依次对它们的多个任务进行加工，各自完成规定的功能动作。

多机系统中并行性的发展

- 机间互连
  - 容错系统(Fault Tolerant)
  - 可重构系统(Reconfigurable)
    - 对计算机之间互连网络的性能提出了更高的要求。
    - 高带宽、低延迟、低开销的机间互连网络是高效实现程序或任务一级并行处理的前提条件。
  - 同构型多处理机系统
    - 由多个同类型或至少担负同等功能的处理机组成，它们同时处理同一作业中能并行执行的多个任务。

并行

- 系统结构的重大转折：
  - 从单纯依靠指令级并行转向开发线程级并行和数据级并行。
- 计算机系统结构在计算机的发展中有着极其重要的作用。