哈尔滨工业大学远程教育学院

计 算 机 系 统 结 构

第 1 学时

哈尔滨工业大学远程教育学院

计 算 机 系 统 结 构

欢 迎 观 看 !!

第 1 章
计算机系统结构的基本概念

- 主要内容：
  计算机系统结构的定义
  (已学:概念性的结构,功能特性)
  结构， 组成， 实现的含义及关系
  计算机系统软硬件功能分配的基本原则，设计思路
  软件，应用，器件对系统结构的影响
  并行性的概念
  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：软件　硬件

系统的功能划分层次结构：

高　à　低

L5 (M5)	L4(M4)	L3(M3)	L2(M2)	L1(M1)	L0(M0)
应用语言	高级语言	汇编语言	操作系统	传统机器	微程序

M0　硬件实现

M1　微程序（固件）实现

M2~M5　软件实现

虚拟机器：以软件为主实现的机器。(特例)

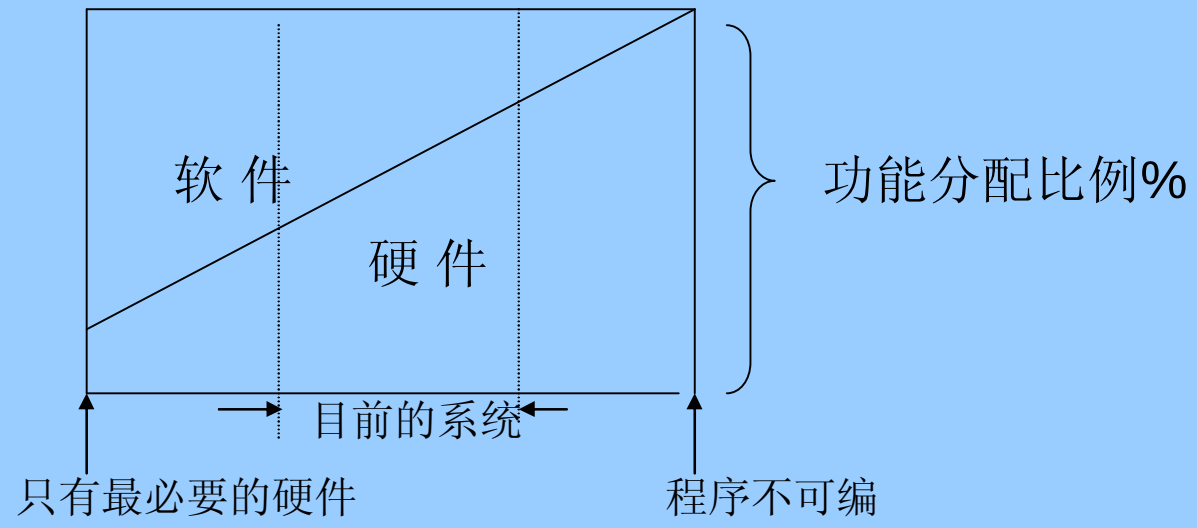
实际机器：以硬件或固件实现的机器。

n 采用方式原则：

效率 速度 造价 资源状况

软件 硬件 固件综合评定

软件硬件在逻辑功能上是有效的。



结论：计算机系统层次结构的划分有利于理解软,硬件,固件的地位和作用,推动计算机系统结构的发展。

(提供更好的硬件

发展高级语言机器或操作系统计算机结构)

1.2 计算机系统结构，组成与实现

1.2.1 结构，组成和实现

从计算机系统的层次结构上定义，系统结构（System Architecture）是对计算机系统中各机器级之间界面的划分和定义，以及对各级界面上下的功能进行分配。(IBM PC与VAX-11

传统机器级与高级语言级　)

计算机体系结构（计算机系统结构）只是系统结构的一部分，指的是层次结构中传统机器级的系统结构。

- 界面之上功能：
  操作系统级 汇编语言级 高级语言级
  应用语言级

- 界面之下功能：所有硬件和固件的功能

软件，硬件的交界处

研究目的：软件，硬件功能分配以及传统机器级界面的确定，提供机器语言，汇编语言程序设计者或编译程序生成系统为使其所设计或生成的程序能在机器上正确运行，应看到和遵循的计算机属性。

数据流和控制流的组成逻辑设计，器件设计不在计算机系统结构中。

计算机系统结构属性包括（见书）

- 计算机组成（Computer Organization ）

  指的是计算机系统的逻辑实现。计算机组成设计是围绕提高速度，着重从提高操作的并行度，重叠度，以及分散功能和设置专用功能部件来进行。包括内容见书。

- 计算机实现（ Computer Implementation ）

  指的是计算机组成的物理实现，包括处理机，主存等部件的物理结构，器件的集成度和速度，器件，模块，插件，底板的划分与连接，专用器件的设计，微组装技术，信号传输，电源，冷却及整机装配技术等。包括内容见书。

1.2.2 计算机系统结构，组成和实现的相 互影响

相同系统结构的计算机可以采用不同的组成。

相同的计算机组成可采用不同的计算机实现。

不同系统结构会使可以采用的组成技术产生差异。举例见书。

反过来，计算机组成也会影响系统结构。

系统结构设计　计算机组成设计

计算机实现---器件技术的发展

计算机系统结构这门学科研究：软硬件的功能分配以及如何最佳最合理实现分配给硬件的功能

1.3 软硬取舍与计算机系统的设计思想

1.3.1 软硬取舍的原则

* 提高硬件功能的比例

正面：可以提高解题速度，减少程序所需的存储空间

负面：会提高硬件成本，降低硬件利用率和计算机系统的灵活性，适应性。

*提高软件功能的比例：

正面：可以降低硬件的造价，提高系统的灵活性，适应性

负面：解题速度下降，软件设计费用和所需的存储器用量增加

- 原则：

1.在现有硬件和器件（逻辑器件和存储器件）的条件下，系统要有高的性能价格比。性能(实现费用,速度等)是综合指标。

实现费用= 研制费用+重复生产费用

Ds表示软件设计费用
Dh表示硬件设计费用

Dh 约等于100Ds 。 Mh 约等于100Ms

假定该计算机系统共生产了V台，
硬件实现的费用：Dh / V + Mh
软件实现费用： C * Ds / V+R*Ms
只有：Dh / V + Mh < C * Ds / V+R*Ms
用硬件实现才适宜。

$100D_s / V + 100 M_s < C * D_s / V + R * M_s$

只有C，R较大时，即只有某个功能是经常用的基本单元功能。

D_s约等于10000 M_s。

只有计算机系统的产量大，增大硬件实现是适宜的。

2. 要考虑到准备采用和可能采用的组成技术，使它尽可能不要过多或不合理限制各种组成，实现技术的采用。

3. 不能仅从“硬”的角度去考虑如何便于应用组成技术的成果，和发挥期间技术的进展，还应从“软”的角度把为编译和操作系统的实现，以至高级语言程序设计提供更好的硬件支持放在首位。

1.3.2 计算机系统的设计思路

- 计算机系统的设计

  由上往下　由下往上 从中间开始

  由上往下设计的方法是先考虑如何满足用户的要求，定好面对使用者那级虚拟机器应具有什么基本特性和环境。，然后逐级向下设计，每级都考虑怎样优化上一级来实现。适合专用机。

- 由下往上设计方法：先不管应用要求，只根据当时能拿到的器件参照或吸收已有的各种机器的特点，把微程序机器级及传统机器级研制出来。然后再配适合于不同应用领域的多种操作系统和编译系统软件。适合通用机
- 缺点：软硬设计脱节。
- 从中间设计

  中间取在传统机器级与操作系统机器级之间

1.4 软件,应用,器件对系统结构的影响

1.4.1软件的可移植问题

软件的可移植性(Portability)

是指软件不用修改或只需经少量加工就能由一台机器搬到另一台机器上运行,即同一软件可以应用于不同的环境.

基本技术:

统一高级语言　采用系列机　模拟和仿真

- 统一高级语言

软件的移植包括:

应用软件和系统软件的移植

至今未有真正通用的高级语言.

原因: * 不同用途要求语言有不同的语法结构和语义结构.

*人们对语言的基本结构看法不一.

*即使同一种高级语言在各个不同厂家的机器上也不能完全通用.

*习惯势力的影响.

- 采用系列机

系列机思想

软件兼容

*向上,向下：是指按某档机器编制的软件，不加修改就能运行于比它高（低）档的 机器上。

*向前，向后：在按某个时期投入市场的该型号的机器上编制的软件，不加修改就能运行在它之（前）后投入生产的机器上。

低档机：不高的速度；低价格；高档机：成本高；先进的器件和复杂的组成实现技术提高速度；

中档机：性能价格比高。

性能价格比：在满足某种性能条件下，降低价格
在某种价格情况下，提高性能。

兼容机：不同公司厂家生产的具有同一系统结构的计算机。

n 模拟和仿真

目的：为实现软件在不同系统结构的机器之间相互移植。

模拟：用机器语言程序解释实现软件移植的方法。

宿主机：进行模拟工作的机器。被模拟的机器称为虚拟机。

模拟程序：所有为各种模拟所编写的解释程序。复杂，费时

模拟的运行速度显著降低，实时性差。

n 仿真

用微程序直接解释另一种机器指令系统的方法称为仿真。

宿主机　目标机　仿真微程序

模拟和仿真的区别：解释所用的语言。

仿真用微程序解释，其解释程序存在控制存储器中；而模拟是用机器语言程序解释，其解释程序存在主存中。

总结：统一高级语言解决结构相同或不同的机器上的软件移植。采用系列机只能实现同一系列结构内的软件兼容，而软件兼容反过来影响制约系统结构的发展。

仿真模拟系统结构差别不大时采用。

1.4.2 应用对系统结构的影响

- 应用对系统结构发展有很重要的影响，系统结构的设计要适应各种应用。

  *20世纪40年代末出现的冯.诺依曼机器是为计算弹道，解偏微分方程而设计的。

  *50年代中，末期以后，计算机应用范围扩展到商业，事务处理。

  *60年代中期以后，IBM 360等一批同时具有科学计算，事务处理，实时控制的应用

- 计算机性能包括：硬件（如主频，CPU运算速度，字长，数据类型，主存容量，寻址空间大小,存储体系,I/O处理能力,I/O设备量,指令系统），软件（高级语言状况,操作系统功能,用户程序包），可靠性，可用性等多种指标的综合。
- 计算机应用归纳为向上升级的4类:

数据处理,信息处理,知识处理,智能处理

1.4.3 器件发展对系统结构的影响

- 计算机器件发展:

  电子管 晶体管

  小规模集成电路 大规模集成电路
- 砷化镓优良的高频特性，它被广泛用于制造无线通信和光通信器件，半绝缘砷化镓单晶已经成为制造大功率微波、毫米波通信器件和集成电路的主要材料。
- 非用户片 **à** 现场片,用户片

对系统结构和组成的发展起重要的推动作用.

- 非用户片(通用片)

  其功能是在器件厂生产时确定下来的,器件的用户(机器的设计者)只能使用.不能改变器件内部的功能

- 现场片

  使用户可根据需要改变器件内部的功能或内容,以适应结构和组成变化的需要.

器件的发展改变了逻辑设计的传统方法,使系统结构下移速度加快,还促进了算法,语言和软件的发展.

1.5 系统结构中的并行性发展及计算机系统的分类

1.5.1 并行性概念

1.并行性的含义与并行性级别

- 提高计算机系统性能的有效途径是开发并行性,挖掘潜在的并行能力,提高其并行处理和操作的程度.
- 并行性:解题中具有可以同时进行运算或操作的特性.
- 只要在同一时刻或是在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,它们在时间上能互相重叠,都体现了并行性.

- 并行性包括同时性和并发性

同时性(Simultaneity)指的是两个或多个事件在同一时刻发生.

并发性(Concurrency)指的是两个或多个事件在同一时间间隔内发生.

从计算机系统中执行程序的角度来看,并行性等级从低到高分为四级:

指令内部　指令之间　　任务或进程之间

作业或程序之间

从计算机系统处理数据的并行性从低到高等级分为:

位串字串:只对一个字的一位进行处理

位并字串:同时对一个字的全部位进行处理

位并串字并:同时对许多字的同一位(位片)进行处理

全并行:同时对许多字的全部或部分位组进行处理.

- 并行性是贯穿于计算机信息加工的各个步骤和阶段的.

存储器操作并行

(并行存储器系统,相联处理机)

处理器操作步骤并行(流水线处理机)

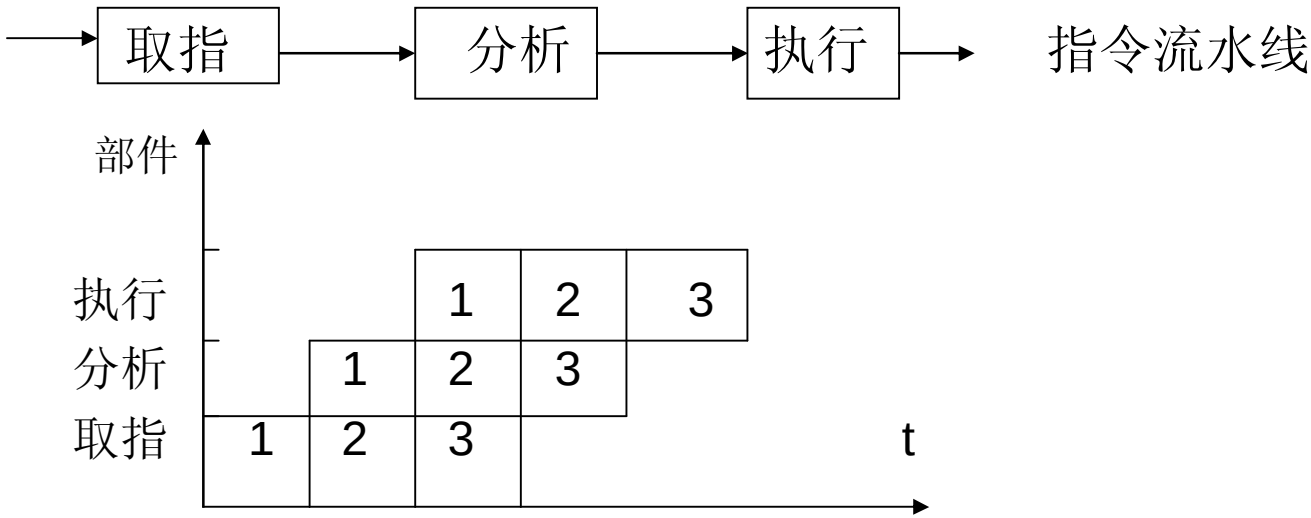
处理器操作并行(阵列处理机)

指令,任务,作业并行(多处理机)

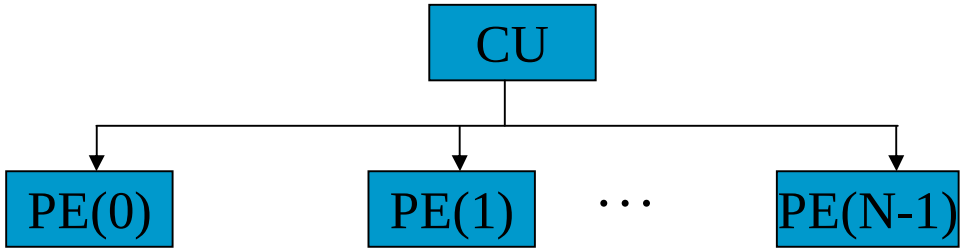
- 2.并行性开发的途径

时间重叠 资源重复 资源共享

时间重叠:是在并行性中概念引入时间因素,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.



- 资源重复是在并行性概念中引入空间因素.通过重复设置硬件资源来提高可靠性或性能.再如:



- 资源共享:是利用软件的方法让多个用户按一定时间顺序轮流地使用同一套资源,以提高其利用率,提高整个系统的性能.

(如计算机网络，分布处理系统等。

软件资源,硬件资源)

- 并行处理是信息处理的一种有效形式，它着重发掘解体过程中的并行事件，使并行性达到较高级别。

1.5.2计算机系统的并行性发展

- 3T性能指标

  1TFLOPS的计算能力

  1TBYTE 的主存

  1TBYTE/S的I/O带宽

1.5.3并行处理系统结构与多机系统的耦合度

- 1.并行处理计算机的结构

按基本结构特征分为:

流水线计算机　阵列处理机

多处理机系统　数据流计算机

- 2.多机系统的耦合度

多机系统指的是多处理机系统和多计算机系统.

二者区别:

为了反映多机系统中各机器之间物理连接的紧密程度和交叉作用的能力的强弱,引入耦合度概念.

多机系统的耦合度可以分为:

最低耦合 松散耦合 紧密耦合

最低耦合除通过某种中间存储介质之外,各计算机之间并无物理连接,也无共享的联机硬件资源.

松散(间接)耦合

紧密耦合结合

1.5.4计算机系统的分类

- Flynn分类法

SISD SIMD MISD MIMD

- Kuck分类法

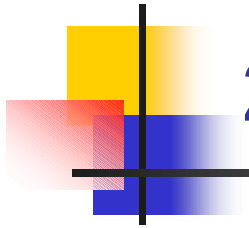
SISE SIME MISE MIME

- Feng氏分类法

WSBS WSBP WPBS WPBP

第2章 数据表示与指令系统

- 本章从数据表示,寻址方式,指令系统来分析应向程序设计者提供什么样的机器级界面,合理进行软硬件功能分配.分析改进计算机体系结构,同时讨论缩小语义差距的某些途径.



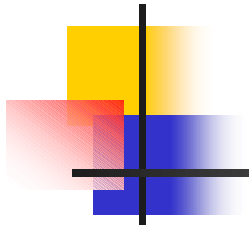
2.1 数据表示

2.1.1数据表示与数据结构

数据表示指的是机器硬件直接识别和引用的数据类型.

串,队,栈,向量,阵列,链表,树,图是软件系统所要处理的各种数据结构.

数据表示的确定实质上是软硬件的取舍问题.



- 早期的机器只有定点数据表示
- 50年代初提出的变址操作,为向量,阵列数据结构的实现提供了直接支持.
- 可变长字符串数据的引入,有力地支持了串数据结构的实现.
- 机器的运算类指令和运算器的结构主要是按机器有什么样的数据表示来确定.怎样为数据结构的实现进一步提供支持,引入高级数据表示

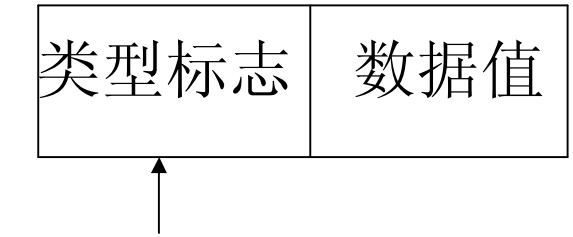
2.1.2 高级数据表示

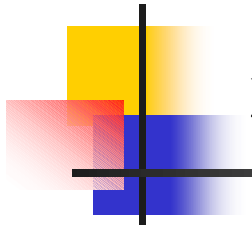
- 自定义数据表示

1)带标志符的数据表示

引入原因:

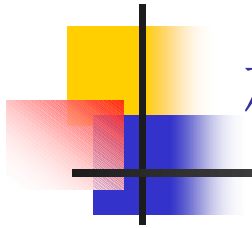
缩短高级语言与机器语言的差距





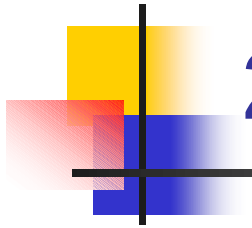
标志符数据表示的主要优点:

- 简化了指令系统和程序设计
- 简化了编译程序
- 便于实现一致性校验
- 能由硬件自动完成数据类型的交换
- 支持了数据库系统的实现与数据类型无关要求
- 为软件调试和应用软件开发提供支持



标志符数据表示的问题

- 每个数据字因增设标志符,会使程序所占用的主存空间增加
- 采用标志符会降低指令的执行速度



2) 数据描述符

- 引入原因
- 数据描述符和标志符的区别
- 优点:有利于简化编译中的代码生成

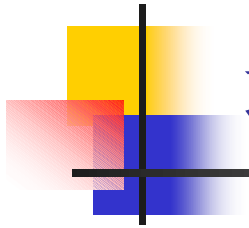
向量数组数据表示

```
  Do 40 I=10,1000
40 C(I)=A(I+5)+B(I)
```

- 没有向量,数组数据表示的机器上,采用变址操作实现
- 在具有向量,数组数据表示的向量处理机上只需一条指令:

向量加 A向量参数 B向量参数 C向量参数

- 对参加运算的每个源向量都应指出其基地址 ,位移量,向量长度和向量元素步距等参数.



堆栈数据表示

- 堆栈数据结构在编译和子程序调用中很有用,不少大型机乃至微型机都设有堆栈数据表示.具有堆栈数据表示的机器称为堆栈机器.
- 堆栈机器特点:

*有若干高速寄存器组成的硬件堆栈

*有丰富的操作类指令

*有力地支持高级语言程序的编译

*有力地支持子程序的嵌套和递归调用。

2.1.3 引入数据表示的原则

存储器　　一维顺序存储的线性结构

数据结构　　多维离散结构

原则：

*系统效率有否提高。即是否减少了实现时间和所需的存储空间。

*通用性和利用率是否高。

综上，数据结构的发展先于机器的数据表示。

2.1.4 浮点数尾数基值大小和下溢处理方法的选择

- 1.浮点数尾数基值的选择

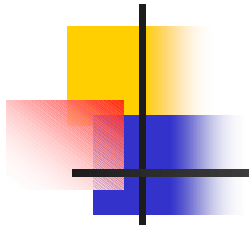
在计算机机器字长相同时，用浮点数表示数学中的实数，比用定点数表示时能有更大的表示数范围。

阶码包含：阶符　阶码值

p+1位阶码部分中影响阶码值的仅有p位

在这种浮点数表示中，阶码的位数p主要影响两个可表示区的大小即可表示数的范围大小；而尾数的位数主要影响可表示区中能表示值的精度。当阶码位数一定时，尾数采用什么进制还会影响到数的可表示范围，精度及数在数轴上分布的离散程度。

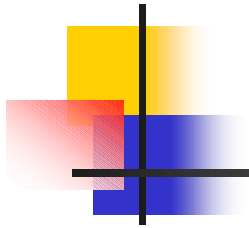
阶码采用二进制。尾数的基值呢？



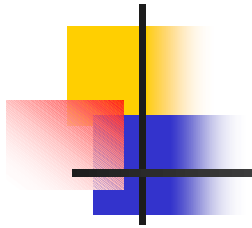
- r_m表示浮点数尾数的基值
- 在机器中，一个r_m进制的数位是用$\lceil \log_2 r_m \rceil$个机器位数来表示。因此，尾数的机器位数为m时,相当于r_m进制的尾数共有m’个数位

$$m' = m / \lceil \log_2 r_m \rceil$$

所谓以r_m为尾数基值的浮点数就是当其尾数右移一个r_m进制数位时，为保持不变，阶码才增1。



- 为了简化问题的讨论，采取只比较非负阶 正尾数，且都是规格化数（尾数小数点后第一个数位不为0的数）
- 见书中表
- 讨论尾基r_m取不同值的影响，相同的机器位数（相同的阶码位数，机器尾数位数）不同r_m对特性参量的影响。



n 结论

*可表示数的范围

*可表示数的个数

*数在实数轴上的分布

表示比e：是在相同p,m 位数时，在$r_m=2$的可表示最大值以内，采用r_m〉2的可表示浮点数个数与$r_m=2$的可表示浮点数个数之比 。

*可表示数的精度

*运算中的精度损失

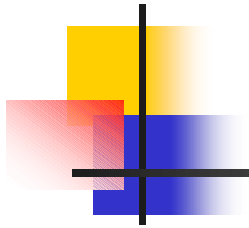
*运算速度

2.浮点数尾数的下溢处理方法

- 截断法:将尾数超出机器字长的部分简单截去。

优点：下溢处理实现最简单，不增加硬件，不需要额外的处理时间。

缺点：最大误差较大，平均误差大且无法调节，因而很少用。

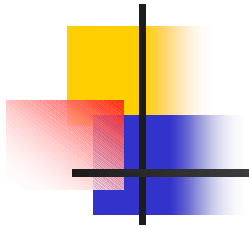


- 舍入法：

是在机器运算部分的规定字长之外增设一位附加位，存放溢出部分的最高位， 每当进行尾数下溢处理时，将此附加位加1

优点：实现简单，增加的硬件很少，最大误差小，平均误差接近于0。

缺点：处理速度慢，需要花费在数的附加位加1以及因此产生进位的时间，最坏的情况可能需要从尾数最低位进位至最高位



- 恒置“1”法：是在机器运算部分的规定字长之最低位恒置“1”状态

优点：实现简单，不增加硬件和处理时间，平均误差接近于0。

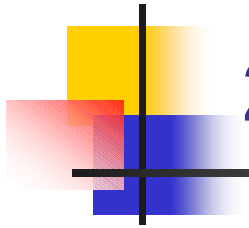
缺点：最大误差最大，适合于中高速机器

- 查表舍入法

其方法基于存储逻辑思想，用ROM或PLA存放下溢处理表。

优点：ROM法速度快，平均误差可调节到趋于0

缺点：硬件设备量增多。



2.2 寻址方式

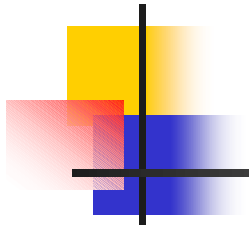
- 寻址方式指的是按什么方式寻找到所需的操作数或信息的。
- 2.2.1 寻址方式分析

按部件分类

各自从0编址,构成多个一维线性地址空间

统一编址

采用隐式地址



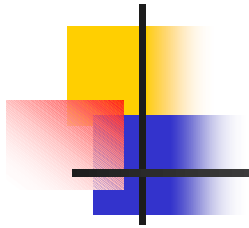
- 大多数计算机将主存 通用寄存器 堆栈分类编址。

面向寄存器的寻址方式

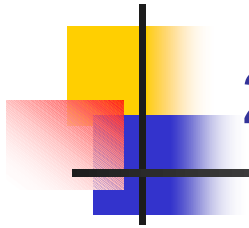
面向主存的寻址方式

面向堆栈的寻址方式

三种寻址方式不应互相排斥



- 寻址方式在指令中的两种指明方式：
* 占用操作码中的某些位指明
* 不占用操作码，而是在地址码部分专门设置寻址方式位字段予以指明。



2.2.2 逻辑地址与主存物理地址

- 逻辑地址
- 主存地址

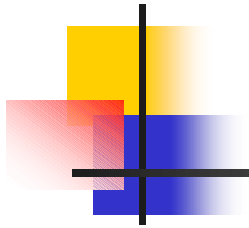
二者是一致的。程序的定位

程序定位方法：

*静态再定位

*动态再定位（基址寻址法）

变址寻址与 基址寻址是两种不同的概念



- 寻址方式中有关物理地址空间的信息分布

信息在存储器内按整数边界存储

今昔状况

2.3 指令系统的设计和改进

- 2.3.1 指令格式的优化

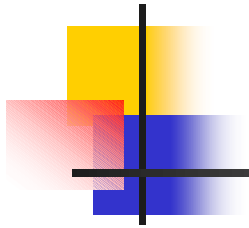
是指如何用最短的位数来表示指令的操作信息和地址信息，是程序中的指令的平均字长最短。

哈夫曼压缩概念

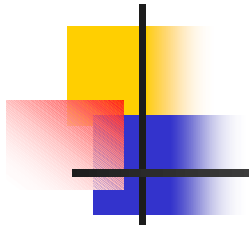
1.操作码的优化

目的:缩短指令字的长度,减少程序总位数及增加指令字所能表示的操作信息和地址信息

操作码的信息熵



- 利用哈夫曼算法，构造哈夫曼树
- 操作码不规整　　扩展操作码
- 举例



2.指令字格式的优化

只对操作码表示进行优化，而没有在地址码表示和寻址方式上采取相应的措施，程序所需总位数难以减少。

一方面,从指令字所能表明的访存操作数地址的寻址范围考虑，希望越大越好。

另一方面，在满足很大寻址范围的前提下，缩短指令字中的地址码位数。

- 举例：IBM 370指令中的地址码形式：

B	D

4位　12位

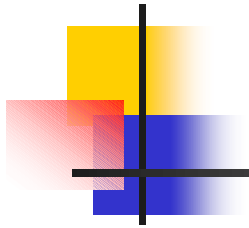
B为基址寄存器，存放24位基地址

- 基址变址寻址

X	B	D
4位	4位	12位

将访存地址空间分成若干个段

段号	段内地址



- 多种地址制
- 同种地址制的多种地址形式和长度
- 让最常用的操作码最短，其地址码字段个数越多，越能使指令的功能增强，从宏观上减少所需的指令条数。
- IBM 370指令格式

*采用8位操作码（5种指令格式）

RR　RX　RS　SI　SS

2.3.2 按增强指令功能的方向发展与改进指令系统

目的：

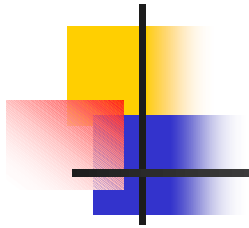
为了使计算机系统具有更强的功能，更高的性能和更好的性能价格比

途径：

* 一种途径是如何进一步增强原有指令的功能以及设置更为复杂的新指令取代原先由软件子程序完成的功能，实现软件功能的硬化。(CISC)

*另一种途径和方法是如何通过减少指令总数和简化指令的功能来降低硬件设计的复杂度，提高指令的执行速度。(RISC)

- 面向目标程序的优化实现改进
- 面向高级语言的优化实现改进
- 面向操作系统的优化实现改进



- 面向目标程序的优化实现改进

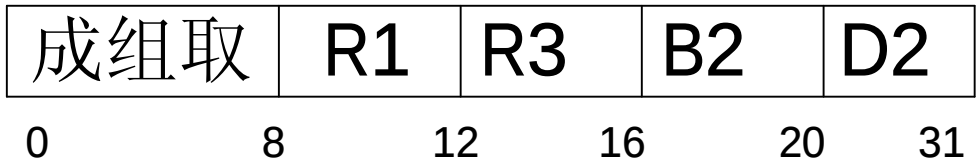
*一种改进和发展指令系统的思路是对大量已有的机器的机器语言程序及其执行情况进行统计，按统计出的指令和指令串的使用频度来分析改进。

对程序中出现的各种指令及指令串进行统计得出的百分比，称为静态使用频度。按静态使用频度来改进指令系统是着眼于减少目标程序所占用的存储空间。

在目标程序执行过程中对出现的各种指令和指令串进行统计得出的百分比,称为动态使用频度.

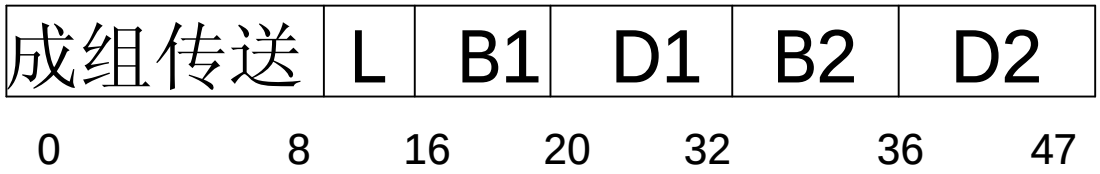
动态使用频度----〉减少目标程序的执行时间

- IBM 370机上增设了用单条指令完成多个数据传送的功能。如成组取指令形式：



将从B2,D2 指明的主存地址起始的一个子向量读到号为R1到R3的一组顺序的通用寄存器中。

如成组传送指令形式：



- 条件转移指令形式（两种）：

条件转移	M1	R2

条件转移	M1	X2	B2	D2

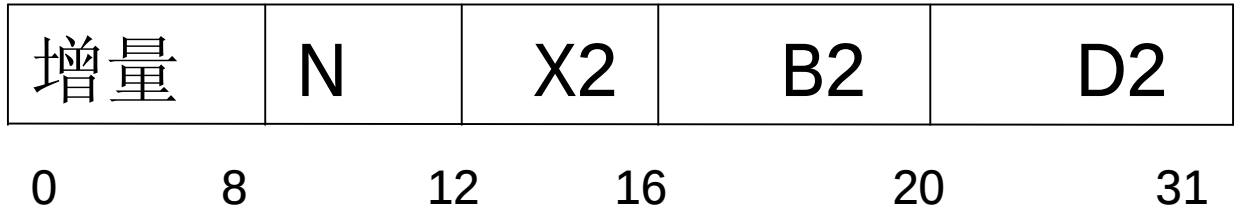
例如加法指令： 条件码00 01 10 11

M1字段共有4位.

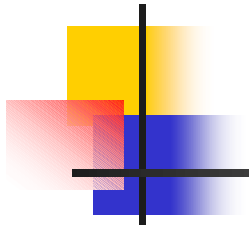
- 指令串

（VARIABLE）+N -> VARIABLE

用“增量”新指令替代，形式为：



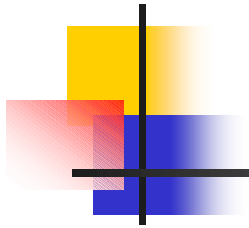
完成 N+((X2)+(B2)+D2) -> (X2)+(B2)+D2



*第二种改进和发展指令系统的思路是，通过增设强功能复合指令来取代原先由常用宏指令或子程序实现的功能。

在IBM 370上增设“翻译”指令，形式：

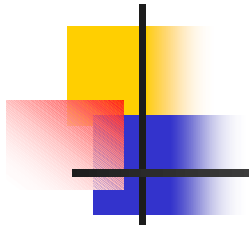
指令系统的上述改进，是以不删改原有指令系统为前提的，通过增加少量强功能新指令来替代常用指令串，以满足软件向后兼容，从而使用新的指令编制的程序具有更高的效率。



- 面向高级语言的优化实现改进

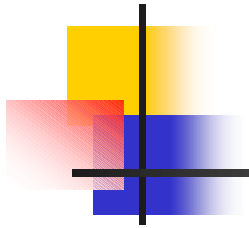
目的：

尽可能缩短高级语言和机器语言的语义差距。有利于支持高级语言编译系统，缩短编译程序的长度和编译所需的时间。



- 途径

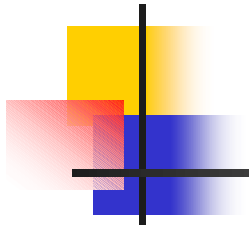
*一种改进思路基于对源程序中各种高级语言语句的使用频度进行统计分析。(见书表)



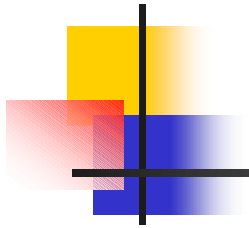
*第二种重要的思路是如何面向编译，优化代码生成来改进。

从优化代码生成上考虑,应当增强系统结构的规整性，尽可能减少甚至不出现例外情况和特殊的用法，让所有运算都能对称均匀地在存储单元间进行。

指令系统设计成简单，对称，规整的作用很重大。



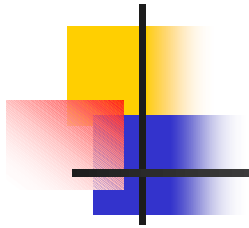
*第三种思路是设法改进指令系统，使它与各种语言间的语义差距都有共同的缩小。



- 第四种思路是让机器具有分别面向各种高级语言的多种指令系统，多种系统结构，并能动态的切换

微程序的发展，特别是可读写控存的采用，给这种动态结构及其的实现提供了可能。

让系统结构优化多种高级语言实现，也就是让系统结构是动态的，而不是静态的，由“以指令系统为主，高级语言为从”演变成“高级语言为主，指令系统为从”方式。



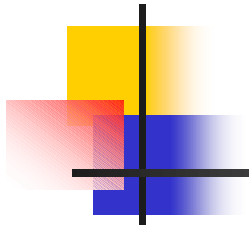
*第五种思路是发展高级语言计算机。

高级语言机器的基本特点是不需要编译，有两种形式：

一种是让高级语言直接成为机器的汇编语言，让高级语言和机器语言基本上一一对应。称这种高级语言机器为间接执行高级语言机器。

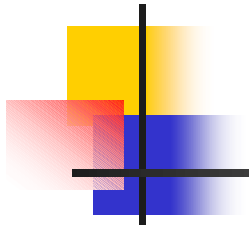
另一种是让高级语言本身作为机器语言。由硬件或固件对高级语言源程序的语句逐条进行解释执行，既不编译，也不用汇编，称这种高级语言机器为直接执行高级语言机器。

编译程序只是系统支持软件中的一个很小的部分，设计人员时熟练的软件人员，几年才设计一次，应用开发是天天在进行，系统结构要重视应用软件的开发支持。



- 面向操作系统的优化实现改进

目的：缩短操作系统与计算机系统结构之间的语义差距，以利于进一步减少运行操作系统所需要的辅助操作时间和节省操作系统软件所占用的存储空间。



- 途径：

*第一种通过对操作系统中常用的指令和指令串的使用频度进行统计和分析来改进。

*第二种改进思路是考虑如何增设专用于操作系统的新指令。

举例,“测试与置定”指令

测试与指定	未用	B2	D2

问题:出现死锁

所谓死锁就是有一组进程，其中每个进程都只占有为完成该进程所必需的部分资源，并未获得全部资源，从而都无法进行下去。

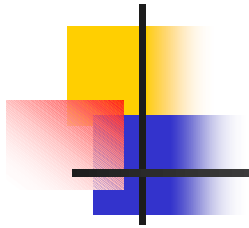
“比较与交换”指令

比较与交换	R1	R3	B2	D2

B2,D2形成公用单元地址

R1 暂存本进程增值前的原始值

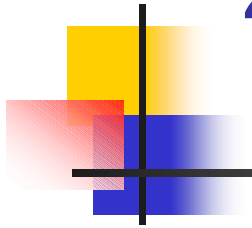
R3 用于本进程增值的工作寄存器



*第三种思路是把操作系统由软件子程序实现的某些功能进行硬化或固化，改用硬件和固件实现。

第四种思路是发展让操作系统由专门的处理机来完成的功能分布处理系统结构。

2.3.3 按简化指令功能的方向发展与改进指令系统

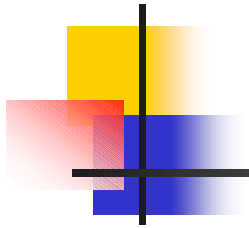


- 精简指令系统思想的提出
- RISC结构采用的基本技术
- RISC技术的发展

1.精简指令系统思想的提出

- 提出的意义：

  为适应对计算机的越来越广泛的应用需要，增强计算机系统的功能，也为减少系统辅助开销，提高机器的运行速度和效率，计算机结构设计者一直在致力于研究进一步缩短高级语言，操作系统，程序设计环境及应用等与机器语言和系统结构的语义差距，加强软硬结合，为系统结构提供更多更好的硬件支持。



n 过去的CISC

问题：

1）指令系统庞大

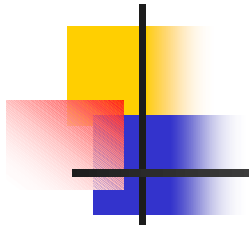
会使完成指令的译码，分析，和执行的控制器硬件非常复杂。

2）由于上述原因，指令执行速度很低，甚至比用几条简单的指令组合实现还慢。

3）使得高级语言编译程序选择目标指令的范围很大，从而难以优化编译生成真正有效的机器语言程序，也使

编译程序本身太长太复杂。

4）各种指令的使用频度都不会太高，且相互差别很大。



- 基于上述问题，Patterson等人提出了精简指令系统计算机的设想原则：

1）确定指令系统时，只选择使用频度很高的那些指令，在此基础上增加少量能有效支持操作系统和高级语言实现及其它功能的最有用的指令，让指令的条数减少

2）减少指令系统可采用的寻址方式种类，简化指令的格式，并让全部指令都具有相同的长度

3）让所有指令都在一个机器周期内完成

4）扩大通用寄存器的个数，减少访存操作，

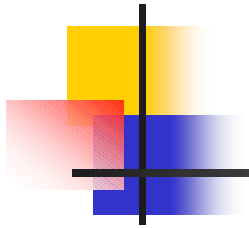
5）为提高指令执行速度，大多数指令都采用硬联控制实现，少数指令采用微程序实现

6）通过精简指令和优化设计编译程序，以简单有效的方式来支持高级语言的实现

2. RISC 结构采用的基本技术

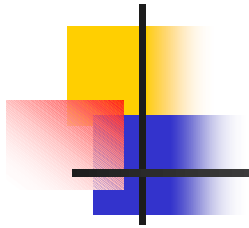
- 遵循前面一般原则设计的技术
- 在逻辑上采用硬联实现和微程序固件实现相结合的技术
- 在CPU中设置数量较大的寄存器组，并采用重叠寄存器窗口的技术

意义：为减少访存次数，尽可能让指令的操作在寄存器间进行，以提高执行速度，缩短指令周期，简化寻址方式和指令方式，同时为了能更简单有效的支持高级语言中大量出现的过程调用，减少过程调用中为保存主调过程的现场，建立被调过程的新现场，以及返回时恢复主调过程现场等所需的辅助操作，也为了更简单直接实现过程与过程之间的参数传递，大多数RISC机器的CPU中都设置有数量较大的寄存器组，让每个过程是用一个有限数量的寄存器窗口，并让各个过程的寄存器窗口部分重叠



举例

结论：采用让相邻过程的低区和高区共用同一组物理寄存器的重叠技术，不需要花费任何附加的操作时间就可以实现这两个过程直接交换参数，因而减少了过程调用和返回时执行的时间，执行的字令条数及访问存储器的次数



- 指令的执行采用流水和延迟转移技术

  一般采用本指令的执行和下条指令的预取在时间上重叠---流水线

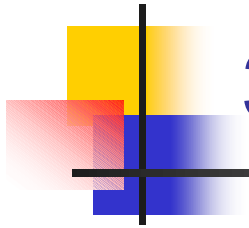
一旦正在执行的是一条条件转移指令且转移成功，或者是一条无条件转移指令，则重叠方式与取得下一条指令就应作废，以保证程序的正确运行，这样浪费了存储器的访问时间-----提出延迟转移的思想

方法：将转移指令与前面的一条指令对换位置，让成功转移总是在紧跟的指令被执行之后发生，从而使预取的指令不作废，就可节省了一个机器周期

举例

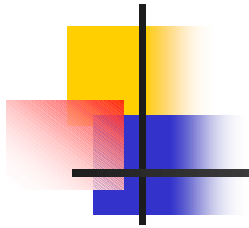
- 采用认真设计和优化编译系统设计的技术

设计出好的编译程序，能够提高机器性能是RISC系统设计的关键



3.RISC技术的发展

- 简化指令系统设计，适合超大规模集成电路实现
- 提高机器的执行速度和效率
- 降低设计成本，提高了系统的可靠性
- 可以提供支持高级语言的能力，简化编译程序的设计



- 问题：

1　由于指令少，使原在CISC上由单一指令完成某些功能现在需要多条指令才能完成

2 对浮点运算和虚拟存储器的支持虽有很大加强，但不够理想

3　编译程序比在CISC机器上难写

- 发展方向

如何减少指令系统的指令条数，每条指令执行的平均周期数及缩短程序执行时所需花费的时间，包括采用高效率的编码技术和更优化的算法，提高CPU的主频，增大结构内部的操作并行性

结论：

设计CPU时，将RISC和 CISC概念技术密切结合

第3章　总线，中断与输入输出系统

- 3.1 输入输出系统概述
- 输入输出系统包括输入输出设备，设备控制器，与输入输出有关的软硬件

早期低性能　程序员直接安排　其I/O系统设计主要考虑及绝好CPU，主存，和输入输出设备在速度上差距

大多数计算机　配备了较多的I/O设备　多用户分享CPU，主存和外部设备资源，每个用户程序的输入输出不能由各个用户自己安排，由用户向系统发出要求进行输入输出请求，经操作系统来分配调度设备进行输入输出

- 应用程序员---输入输出系统硬件功能是透明的
- 计算机输入输出系统的系统结构设计应是面向操作系统，即在操作系统与输入输出系统之间进行合理的软硬件功能分配
- 功能
- 输入输出系统的系统结构设计的重要性

- 输入输出系统的发展经历三个阶段（三种方式）

程序控制输入输出

直接存储器访问（DMA）

I/O处理机(通道方式，外围处理机方式)

- 1）通道方式

通道实际上可以看作是一台处理机，有自己的指令系统。每条指令为输入输出设备规定一定的动作，通道程序存放在驻村乡应该通道的缓冲区中，通道通过执行通道程序来控制输入输出，这又是可以与CPU的程序并行执行的

缺点：指令功能简单 只具有面向外设控制和数据传送的指令，通道程序又是存在和CPU公用的主存内，不能看作是独立的处理机

- 2）外围处理机

  独立性 通用性 功能较强的处理机

I/O系统的复杂由分时系统的发展导致

通道总线，输入输出总线----总线

中断系统

3.2 总线设计

- 重要性：

输入输出系统的总线既要传送数据信息，地址信息，控制信息，还要传送状态信息，并使多台外设与CPU 或主存交叉经这些总线传送信息，对 I/O系统性能有较大影响

3.2.1 总线类型

按允许信息传送方向：

单向传输

双向传输（半双向，全双向区别）

按用法：专用（只连接一对物理部件的总线）

非专用（可以被多种功能或多个部件所分时共享，同一时侯只有一对部件可使用总线进行通信）

专用总线的特点

优点：系统的流量高多个部件可以同时发送或接收信息，几乎不争用总线，控制简单，不用指明源和目的，系统可靠

缺点：总线数目多 N*(N-1)/2

非专用总线的特点

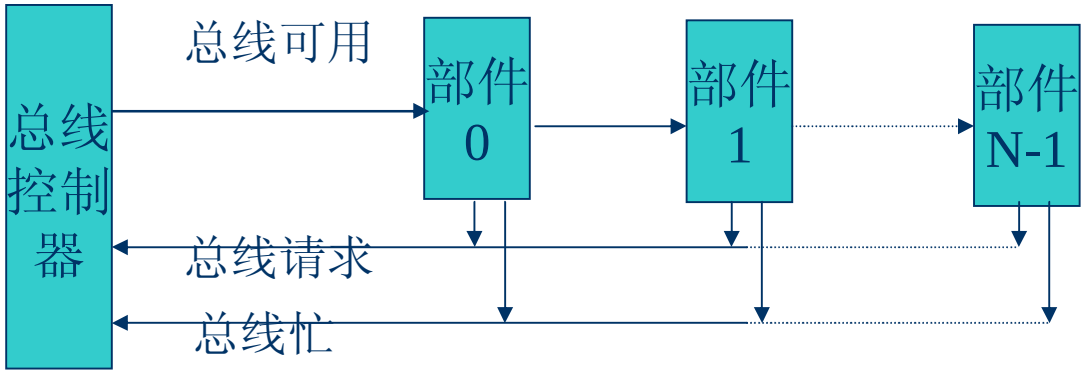
优点：总线数少，造价低，总线接口标准化，可扩充性强

缺点：系统流量小，经常出现总线争用

- 结论：I/O系统适用于使用非专用总线

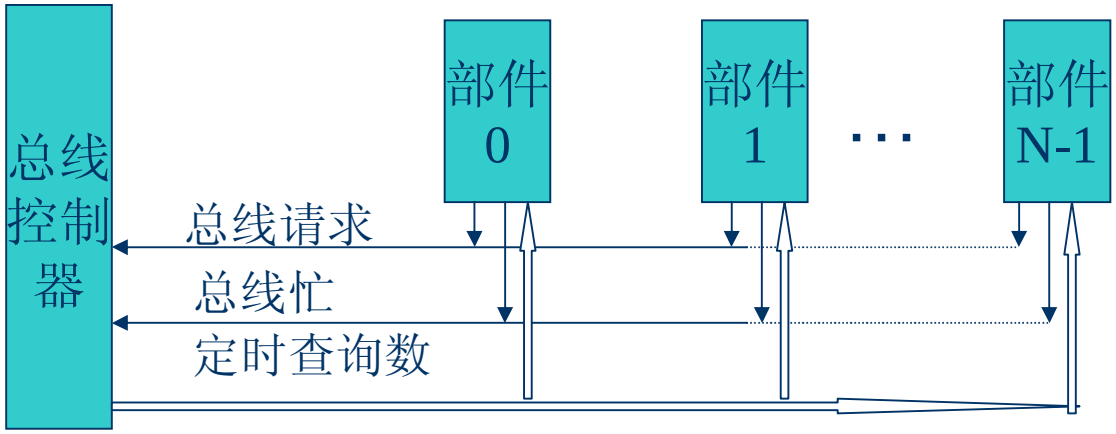
3.2.2 总线的控制方式

- 采用非专用总线---------需总线控制机构来按照某种优先次序裁决
- 集中式总线控制　分布式总线控制
- 确定优先次序的方式：
  串行链接，定时查询，独立请求



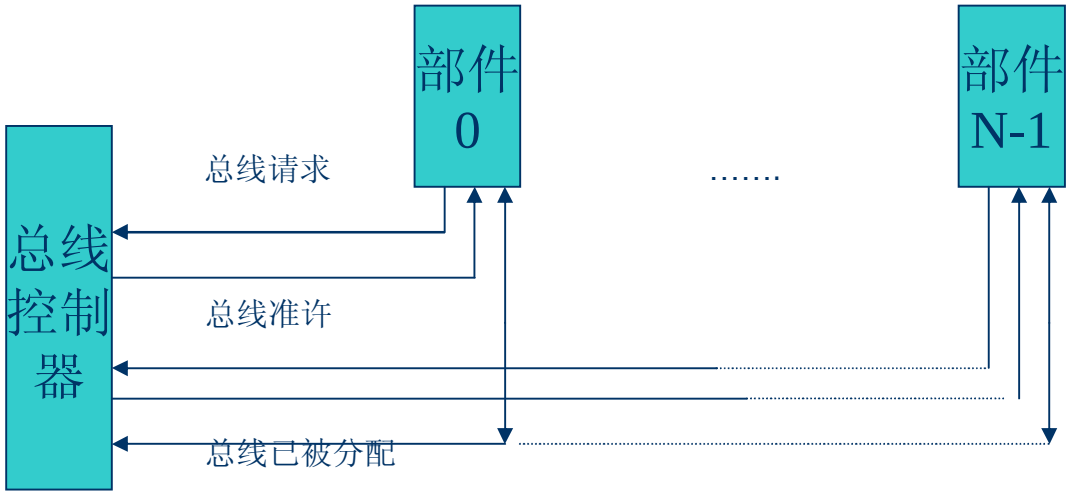
集中式串行链接

特点：



集中式定时查询

特点：



集中式独立请求

特点：

3.2.3 总线的通讯技术

- 信息在总线上的传送方式：

  同步 异步

1. 同步通信：两个部件之间的信息传送是通过定宽，定距的系统时标进行同步的

特点:

信息传送效率高，受总线的长度影响小，但因在总线上的时滞而造成同步误差，且时钟线上的干扰信号易引起误同步

2. 异步通信（单向控制　双向控制）

原因：

I/O总线为具有不同速度的许多I/O设备所共享

单向控制指的是通信过程只有目的或源部件中的一个控制。

双向控制指的是通信过程由目的和源部件双方控制

3.2.4 数据宽度与总线线数

- 数据宽度

  指的是I/O设备取得I/O总线使用权后所传送数据的总量 数据通路宽度

  数据宽度的种类有单字，定长块，可变长块，单字加定长块和单字加可变长块

- 总线的线数

  总线越多越长，成本越高，干扰越大，可靠性越低

  在满足性能要求以及所用通信类型和速率适配的情况下，应尽量减少总线的线数

  （ 采用线的组合 并/串　串/并转换　编码）

  所谓I/O接口或I/O控制器的接口，除了接口编号外还需包括能满足I/O总线要求的申请，使用规范，因此I/O总线接口的标准化非常重要

流量的确定

3.3 中断系统

中断系统不只是I/O系统，也是整个计算机系统必不可少的重要组成部分

3.1.1 中断的分类和分级

中断源：引起中断的各种事件

中断请求：中断源向中断系统发出请求中断的申请

多种中断请求 优先次序

中断响应：允许其中断CPU现行程序的运行，转去对该请求进行预处理，包括保存好断点调出有关处理该中断的中断处理程序，准备运行

- 为处理一个中断请求，必须调出相应的中断处理程序

  中断源少----硬件

  中断源多----将中断源归类，对每一类给定一个中断处理程序入口，再由软件转入对相应的中断源进行处理

以IBM 370为例

- 机器校验中断
- 访管中断
- 程序性中断
- 外中断
- 输入输出中断

- 不少计算机把中断现行进程的事件进一步分成中断和异常

  中断

  异常：自陷 故障 失败

中断处理次序可以不同中断响应次序

中断响应的次序由于使用排队器硬件实现的，所以响应次序是固定的

设置中断级屏蔽位寄存器硬件，决定是否让某级中断请求进入中断响应排队器排队

由操作系统控制改变实际的中断处理次序，操作系统对每一类中断处理程序的现行程序状态字中的中断级屏蔽位设置成不同状态

3.3.2 中断系统的软硬件功能分配

- 中断系统的功能包括中断请求的保存和清除，优先级的确定，中断断点及现场的保存，对中断请求的分析和处理以及中断返回等，这些全由中断响应硬件和中断处理程序共同完成。
- 中断系统的软硬件功能分配实质上是中断处理程序软件和中断硬件的功能分配
- 中断现场包括软件状态，硬件状态
- 中断系统的重要性能指标：

  从发出中断请求到进入中断处理程序的中断响应时间　取决于交换程序状态字的时间

中断系统应具有较大的灵活性

实际上并不是所有的中断处理都需要把通用寄存器的内容或全部通用寄存器的内容都保护起来

一种是整个任务的切换

一种是某道程序调用某个管理程序的切换（部分保存）

设置通用寄存器组与主存或堆栈之间的成组传送指令，减少取指时间

3.4通道处理机

通道处理机是IBM公司首先提出来的一种I/O处理机方式

- 工作原理

管态指令　中央处理机用来控制外部设备操作用的输入输出指令，以使用户不得在目态程序中使用这些指令

用户只有通过在目态程序中安排一条要求输入输出的广义指令来使用外部设备，经广义指令（访管指令和参数）进入相应的管理程序来解释执行

访管指令是目态指令，当目态程序执行到要求输入输出的访管指令后，产生自愿访管中断。

CPU相应此中断请求后，转向该管理程序入口，进入管态

- 根据通道数据传送期中信息传送的方式不同，分为：

  字节多路通道　选择通道　数组多路通道

字节多路通道适用于连接字符类低速设备

数组多路通道　适合于多台高速设备服务

选择通道适合于连接优先级高的高速设备

- 通道流量分析

通道流量是指通道在数据传送期内，单位时间内传送的字节数

通道极限流量是它能达到的最大流量

一个通道能达到的极限流量与其工作方式，数据传送期内选择一次设备的时间（Ts）和传送一个字节的时间(Td)的长短有关

通道极限流量

- 字节多路通道　　1/(Ts +Td)
- 选择通道　　N/(Ts +NTd)= 1/(Ts/N +Td)
- 数组多路通道　k/(Ts +kTd)= 1/(Ts/k +Td)

结论： Ts，Td一定时，当N>K时

字节多路方式通道极限流量最小

数组多路方式 居中

选择方式最大

- 实际最大流量
- 实际最大流量不超过通道所能达到的极限流量
- 举例

3.5 外围处理机

- 通道处理机实际上不能看成是独立的处理机（指令功能简单，只有面向外设控制和数据传送的功能，没有大容量的存储器。就是在输入输出的过程中，也还需CPU承担）
- 外围处理机接近于一般的处理机，指令功能丰富，有利于简化设备控制器
- 优点：可以自由选择通道和设备进行通信

  主存，PPU，通道和设备控制器相对独立，可以按需要用程序动态控制它们之间的连接，更具有灵活性；由于PPU是独立的处理机，具有一定的运算能力，可以承担一般的外围运算处理和控制任务，还可以让各台外设不必通过主存就可以直接交换信息，提高了整个系统的效率

- 外围处理机方式就其硬件利用率和成本来讲不如通道处理机好
- 结论：

在设计I/O系统时，硬件的利用率和成本不再是强调的问题，而是应当考虑怎样才能进一步减少CPU对I/O系统的介入充分提高整个系统的功能和性能。

第4章　存贮体系

- 4.1 存贮体系的形成与性能
- 4.1.1　发展存贮体系的必要性

性能的基本要求：　大容量　高速度　低价格

存贮器容量=w*l*m

w---存贮体的字长

l ----每个存贮体的字数

m---并行工作的存贮体个数

- 存贮器的速度可以用访问时间Ta,存贮周期Tm 和频宽Bm来描述
- 访问时间
- 存贮周期
- 频宽（最大频宽　实际频宽）

单体　　Bm=W/Tm

m个存贮体 Bm=W*m/Tm

实际频宽往往低于最大频宽

存贮器的价格可以用总价格或每位价格来表示

- 计算机系统希望存贮器

  价格低，高速度，高容量

采用多种不同工艺的存贮器组成存贮器系统，使所有的信息以各种方式分布于不同的存贮器

主存 辅存

发展存贮体系的必要性

原因　主存速度与CPU速度的差别

采用技术：

在组成上的并行和重叠技术—并行主存系统

使主存的频宽提高

4.1.2 并行主存系统频宽的分析

- 举例
- 同时启动 分时启动
- 主存用多体单字方式组成，所花费的器件和总价格并不比采用单体多字方式的多多少，但实际频宽却比较高。（原因）
- 得出提高m 的值能提高贮存系统的最大频宽 但主存实际频宽并不随m值的增大而线性提高
- （原因：在工程实现上由于m 越高，存贮器数据总线越长，总线并联的负载越重，有时不得不增加门的级数，这样会使传输延迟增加。其次是系统效率问题，指令不总是顺序执行）

- 模型分析

处理机发出的一串地址为A1, A2,...Aq的访存申请队。

申请序列

申请序列的长度k是个随机变量，系统效率取决于k的平均值

k越接近于m，效率越高

设p(k)表示申请序列长度为k的概率密度函数

K的平均值用B表示

$$B=\sum_{k=1}^{m} k*p(k)$$

实际上是每个主存周期所能访问的平均字数

p(k)与程序的状态相关

如果访存申请队都是指令，那么影响最大的是转移概率

- 转移概率的定义
- 归纳出公式
- 结论：

若每条指令都是转移指令且转移成功（转移概率为1），B=1。就是说此时并行多体交叉存取的实际频宽降低到和单体单字得以样

若每条指令都不转移（转移概率为0），B=m。就是说此时多体交叉存贮的效率最高

单纯靠增大m 来提高并行主存系统的频宽是有限的，而且性能价格比还会随m的增大而下降，就比必须改进系统结构，采用存贮体系

- 4.1.3 存贮体系的形成与分支

操作系统和硬件技术的完善，增设辅助软硬件自动完成

从应用程序员来看，主辅存构成一个完整的整体。从整体上看，速度是主存的；容量是辅存的，形成了一个存贮体系

从容量上 引出虚拟存贮系统

虚地址 实地址

存贮层次对满足对应用程序设计者透明

从速度要求分析

- 主存和CPU的速度差距达1个数量级
- 解决方法：

  在CPU中设置通用寄存器

  采用存贮器的多体交叉并行存取来提高主存的等效速度

  采用存贮层次 --- Cache存贮器

能否预知出下步所要访问的程序块，对存贮体系的构成非常重要

程序的局部性（时间 空间）

- 预判的准确性是存贮层次设计好坏的主要标志，取决于所用的算法和地址映像，变换方式

- 4.1.4 存贮体系的性能参数

为评价存贮层次的性能，引入存贮层次的每位价格c，命中率H，等效访问时间Ta

每位平均价格c

$c=(c1*Sm1+ c2*Sm2)/(Sm1+ Sm2)$

希望存贮层次的每位平均价格能接近于c2，为此应使 $Sm1<< Sm2$

命中率H:CPU产生的逻辑地址能在M1中访问到的概率

$H=R1/(R1+R2)$

R1 为逻辑地址流的信息能在M1中访问到的次数

R2 在M2中还未调到M1的次数

H越大越好 接近1越好

失败率（不命中率）CPU产生的逻辑地址不能在M1中访问到的概率

- 存贮层次的等效访问时间

  Ta=HTa1+(1-H)Ta2

希望Ta接近于Ta1

存贮层次的访问效率= Ta1/Ta 越接近1越好

CPU对存贮层次相邻二级的访问时间比

r= Ta2/Ta1

e=1/(H+(1-H)*r)

要想e越接近1越好，要求能在选择具有高命中率的算法，相邻二级的容量差和速度差及增加的辅助软硬件的代价大小等因素之间加以权衡，优化设计

4.2 虚拟存贮器

- 虚拟存贮器是主存—辅存存贮层次的进一步发展和完善　为克服高速的实际贮存容量满足不了要求而提出来的
- 4.2.1　不同的虚拟存贮管理方式

虚拟存贮器通过增设地址映像表机构来实现程序在主存中的定位

由于采用的存贮映像算法不同，形成多种不同的存贮管理方式的虚拟存贮器

段式　页式　段页式

- 段式管理

  把主存按段分配的存贮管理方式称为段式管理

  为了进行段式管理，每道程序在系统中都有一个段表（映像表）用以存放该程序各程序段装入主存的状况信息

  见书上的示意图

分段方法能使大程序分模块编制，并行编程

分段便于几道程序共用已在主存内的程序和数据

易以段为单位实现存贮保护

为了进行段式管理，还得由操作系统为整个主存系统建立一个实主存管理表（占用区域表　可用区域表）

段的长度不同，一般采用

首先分配法，特点

最佳分配法，特点

- 页式管理

提出原因

系统设置相应的页（映像）表，保存好虚页装入实页时的页面对应关系，就可以由给定的程序地址通过查页表，变换成相应的实存地址访存

- 段页式管理

  提出的意义

  段页式存贮把实存机械地等分成固定大小的页，程序按模块分段，每个段又分成与主存页面大小相同的页

- 4.2.2 页式虚拟存贮器构成

1.地址的映像和变换

页式虚拟存贮器是采用页式存贮和管理的主存—辅存存贮层次。将主存空间和程序空间都机械地等分成页，并让程序的起点总是处在页的起点上

引题：虚拟存贮器工作过程中存在着如何把大得多用户虚存空间压缩到小的主存空间中的问题，在程序运行过程中又要考虑如何将多用户虚地址变换成主存地址，再让CPU去访问主存中该单元

- 地址的映像

  将每个虚存单元按某种规则（算法）装入（定位）实存，即建立多用户虚地址与实存地址之间的对应关系

- 地址的变换

  程序按照这种映像关系装入实存后，在执行时，多用户虚地址如何变换成对应的式地址

  对页式而言就是多用户虚页号如何变换成实页号

- 由于是把大的虚存空间压缩到小的主存空间去，主存中的每一个页面位置必须能与多个虚页相对应，能对应多个虚页与采用的映像方式有关。
- 出现的问题：发生页面争用或实页冲突
- 一旦发生实页冲突，只能先装入其中的一个虚页，待其退出主存后方可再装入

全相联映像：让每道程序的任何虚页可以映像装入到任何实页位置

全相联映像的实页冲突概率最低

采用页表作为地址映像表---页表法

页表中绝大部分行中实页号字段及其他字段是无用的，降低页表的空间利用率

解决办法：

一种解决方法是可以将页表中装入位为0的行，用实页号字段存放该程序此虚页在辅存中的实地址，以便调页时实现用户虚页号到辅存实地址的变换

另一种方法是把页表压缩成只存放已装入主存的那些虚页与实页位置的对应关系

内页表　　外页表

2. 替換算法

3.虚拟存贮器工作的全过程

4.2.3 页式虚拟存贮器实现中的问题

1.页面失效的处理

页面的划分只是机械地对程序空间和主存空间进行等分，与程序的逻辑结构毫无联系。

出现的问题：

一条指令，一个操作数跨页存贮

采用间接寻址时，在寻址过程中，出现跨页情况，每当当前一页已在主存。而跨页存放的另一页不在主存中时就会发生页面失效

页面失效会在一条指令的分析或执行的过程中发生

- 问题解决：

  页面失效不能按一般的中断对待，应作一种故障，处理机必须立即予以响应和处理

  如何保护故障点现场，以及故障处理完如何将当前所需的页面调入主存，恢复故障现场的问题，以便继续执行这条指令

（采用后援寄存器技术　　预判技术）

替换算法的重要性

一道程序分配的主存页数应由某个下限

页面的大小也不能过大，降低了虚拟存贮器的访问效率和速度

2.提高虚拟存贮器等效访问速度的措施

要想使虚拟存贮器的等效访问速度提高到接近与主存的访问速度是不容易的

要求能有很高的主存命中率，尽可能短的访主存时间

缩短访主存时间的措施

加快多用户虚地址到主存实地址的变换，仿佛虚拟存贮器不存在

问题：如何从逻辑结构上来提高内部地址变换的速度

- 需改进的原因
- 寄存器组存放页表　快表（用快速硬件构成比全表小得多的部分目录表存放当前正用的虚实地址映像关系，其相联查找的速度将会很快，称为快表）
- 书中地址变换图
- 快表--慢表存贮层次的替换算法一般采用LRU法
- 提高快表查找速度的办法：

减少相联比较位

增设一位用户位

存在的问题

- 散列方法

基本思想：是让内容Nv与存放该内容的地址A之间有某种散列函数的关系

3.影响主存命中率和CPU效率的某些因素

H Sp S1

主存命中率与采用的页面调度策略有关

4.3 高速缓冲存贮器

- 高速缓冲存贮器用来弥补主存速度的不足，在处理机与主存之间设置一个高速，小容量的缓冲存贮器（Cache)，构成Cache—主存存贮层次；从CPU来看，速度接近Cache ，容量是主存的
- 4.3.1 基本结构

在高速缓冲存贮器中，把Cache和主存机械等分成相同大小的块。每一块由若干个字组成。

实际上，块的大小比页的大小小得多。

每当给出一个主存地址进行访存时，都必须通过主存– Cache地址映像变换机构判定该访问字所在的块是否已在Cache 中。（三种情况）

- 目前访问Cache的时间一般可以是访主存时间的1/4到/10
- Cache—主存存贮层次和主存—辅存存贮层次基本原理是相同的
- Cache与主存的速度差距，比主存与辅存之间的速度差距小两个数量级
- Cache存贮器一般采用与CPU相同类型的半导体工艺构成，有关地址映像和变换的方式及替换调度算法全部采用专门的硬件实现

- 地址变换和真正访Cache两部分工作，采用在时间上重叠的流水工作方式
- 为了更好发挥Cache的高速性，减小CPU与Cache之间的传输延迟，需让Cache在物理位置上尽量靠近处理机或放在处理机中，而不放在主存中，对共用主存的多处理机系统，如果每个处理机都有它自己的Cache，让处理机主要与Cache交往，就能减少主存使用的冲突，提高整个系统的吞吐量

- 在处理机和Cache，主存的联系不同于虚拟存贮器的处理机和主存，辅存之间的联系方式

虚拟存贮器的处理机和辅存之间没有直接的通路，因为辅存的速度相对主存的差距很大。一旦发生页面失效，由辅存调页的时间是毫秒级。为使处理机在这段时间内不致白等，一般采用换到其他程序的办法

对Cache存贮器发生块失效时，由于主存调块的时间是微秒级，显然不能采用程序换道。 Cache到处理机设有通路，主存和处理机之间设有通路。（优点）

为了加速调块，一般让每块的容量等于一个主存周期内由主存所能访问到的字数，因此在有Cache存贮器的主存系统都采用多体交叉存贮器

提高Cache的访主存优先级

- 4.3.2 地址的映像与变换

1.全相联映像和变换

让主存中任意的一块均可映像装入到Cache内的任意一块位置上

为了加快主存– Cache地址变换速度，采用目录表硬件实现

优点：块冲突概率最低，只有当Cache中全部装满后，才有可能出现块冲突。空间利用率高

缺点：相联存贮器的容量大，其代价相对较大，查表速度难以提高

2.直接映像及其变换

主存中每一块只能映像到Cache中唯一一个特定块位置

这就相当于把主存空间按Cache得空间分成区，每区内的各块均只能按位置一一对应到Cache相应位置上

优点：硬件简单，只需容量较小的按地址访问的区号表指表存贮器和少量外比较电路。访问Cache与访问区号表，比较区号是否相符的操作是同时进行的。

缺点： Cache块冲突率高，只要有两个或两个以上经常使用的块被映像到Cache的同一块中，会使Cache的命中率急剧下降。

3. 组相联映像及其变换

将Cache空间和主存空间都分成组，每组为S块,整个Cache是一区。主存分成与Cache同样大小的2的nd次幂个区

组相联映像指的是各组之间是直接映像，但组内各块间则是全相联映像。

规则分析

结论：直接映像和全相联映像是组相联映像的两个极端.当S值大到等于Cache的块数时就变成了全相联映像，当S值小到只有1块时，就变成了直接映像

组相联映像比全相联映像在成本上要低得多，性能上仍接近全相联映像

各组间是直接映像，组号照搬。每组都需有目录表，目录表的行数，宽度均比全相联的少，查表速度提高

- 组相联的地址变换原理

一种方案： 通过S套外比较电路与主存地址的区号，组内块号比较

另一种方案：把主存每2的q次幂做一节，每节中的第i块只能直接映像到第i组，但可以全相联映像到该组的任一块

4. 段相联映像

在组相联映像，直接映像，全相联映像的基础上可以有各种变形。段相联映像实质是组相联映像的特例。

它采用各组之间是全相联映像，组内是直接映像，称之为段相联。

目的：减小相联目录表的容量，降低成本，提高地址变换的速度

- 4.3.3 替换算法的实现

Cache的调块时间是微秒级的，不能采用程序换道，只能全部采用硬件途径实现

堆栈法

LRU法是堆栈型替换算法，对于LRU 法，堆栈中由栈顶到栈底的各行反映出到某时刻，实存中各页被访问过的近远次序，以及每访问一页，堆栈的各项变化

硬件堆栈（全相联　组相联：每组都要有一个能反映该组内各块使用情况的堆栈）

各块被访问的先后次序由该项在堆栈中距离栈底是近还是远来反映，为了避免堆栈中各行存放的内容经常同时进行下移，以节省成本采用另外的变形。

比较对法

堆栈法需要硬件有相联比较的功能，因此速度较低，比较贵，，能否不用相联比较，只用一般的门，触发器来实现LRU替换算法呢？

基本思想：让各个块成对组合，用一个触发器的状态来表示该比较对内两块访问的远近次序，再经门电路就可找到LRU块（书中例题）

替换算法实现的设计是围绕下述两点考虑的：一是如何对每次访问进行记录的

二是如何根据所记录的信息来判定近期内哪一块是最久没有被访问过的

实现方法和所用的映像方法密切相关

- 4.3.4 Cache的透明性及性能分析

1. Cache 的透明性分析

采取措施解决读写过程中的产生的Cache和主存对应内容不一致的问题

* 主存内容跟不上Cache变化

更新主存方法：写回法　写直达法

共享主存的多处理机系统 多采用写直达法

大多数共享主存的多处理机系统，每个CPU都有自己的Cache与共享主存连接

问题：不能保证同一主存单元在各个Cache中的对应内容都一致

解决：

播写法：

控制方法：

目录表法：

*Cache内容跟不上主存内容的变化

解决方法：专用指令　专用硬件

2．Cache的取算法

由于Cache的命中率对机器速度性能影响很大，提高Cache的命中率的算法是设计中的重要问题

Cache的取算法基本上是按需取进法（在出现Cache块失效时，才将要访问的字所在的块取进）

提高Cache的命中率：

选择好Cache的容量，块的大小，组相联的租数，组内块数

预取法（恒预取　不命中时预取）

影响因素：块的大小　预取开销

D_c * 不命中率（按需取进）〉

[D_c * 不命中率（预取）+P_a*预取率 + A_c*（访问率-1）]

3. 任务切换对失效率的影响

任务切换频度是失效率高低的重要原因

热启动失效率　冷启动失效率

任务切换频度与工作负荷有关　Cache容量大小有关

解决办法：增大Cache容量；修改调度算法，使任务切换回来之前，有用的信息仍保留在Cache中不被破坏；设置多个Cache

4. 影响Cache存贮器性能的因素

评价Cache存贮器主要看命中率的高低，而命中率与块的大小，块的总数（ Cache的总容量），采用组相联时的组的大小（组内块数），替换策略和地址流情况有关

推导Cache—主存存贮层次的等效速度与命中率的关系

Cache的容量对机器速度的影响

4.3.5 Cache—主存---辅存 存贮层次

4.4 主存保护

- 目的：为使系统能正常工作，应防止由一个用户程序出错而破坏其他用户程序或系统软件，还要防止一个用户程序不合法地访问不是分配给它的主存区域

存贮区域的保护

对虚拟存贮器的主存区域保护就需要采用页表保护和键式保护

- 页表保护

每个程序有它自己的页表，其行数等于该程序的虚页数

若地址形成环节由于软硬件方面的故障而形成了不属于本程序的错误主存地址时，这种保护无能为力

- 键式保护

键方式由操作系统按当时主存的使用分配状况给主存的每页配一个键，称为存贮键（锁）

访问键（钥匙）

- 如何保护正在执行的程序

程序的重要关键部分的保护

环状保护：这种保护把系统程序和用户程序按其重要性及对整个系统能否正常工作的影响程度分层

这种环式保护既能保证由于用户程序的出错不致侵犯系统程序，也能保证由于同一用户程序内的低级部分的出错而不至于破坏其高级部分

上面讲的区域保护是对允许访问的区域可以进行任何形式的访问，而对允许访问的区域之外，不允许任何形式的访问，只是这种限制适应不了各种应用的要求，因此还得加上访问方式的限制

（见书）

至于环式保护和页表保护，可以把R,W,E等访问方式位设在各个程序的段页表的各行内，使得同一环内或同一段内的各页可以有上述种种不同的访问保护，增强灵活性

第5章 重叠，流水和向量处理机

- 加快机器语言的解释是计算机组成设计的基本任务　一方面，通过选用更高速的器件，采用更好的运算方法，提高指令内各微操作的并行程度，减少解释过程所需要的拍数等多项措施来加快每条机器指令的解释
- 另一方面通过控制机构采用同时解释两条，多条以至整段程序的控制方式来加快整个机器语言程序的解释
- 引出控制方式：重叠，流水

n 5.1 重叠解释方式

1．基本思想和一次重叠

指令的顺序解释方式(优点　缺点)

指令的重叠解释方式

不能加快一条指令的实现,但能加快两条或

一段程序的解释

对计算机组成的要求

取址和分析在时间上的重叠

（操作数和指令分别存放在两个独立编址且可同时访问的存贮器中

操作数和指令混存，但采用多体交叉主存结构

增设指令缓冲寄存器）

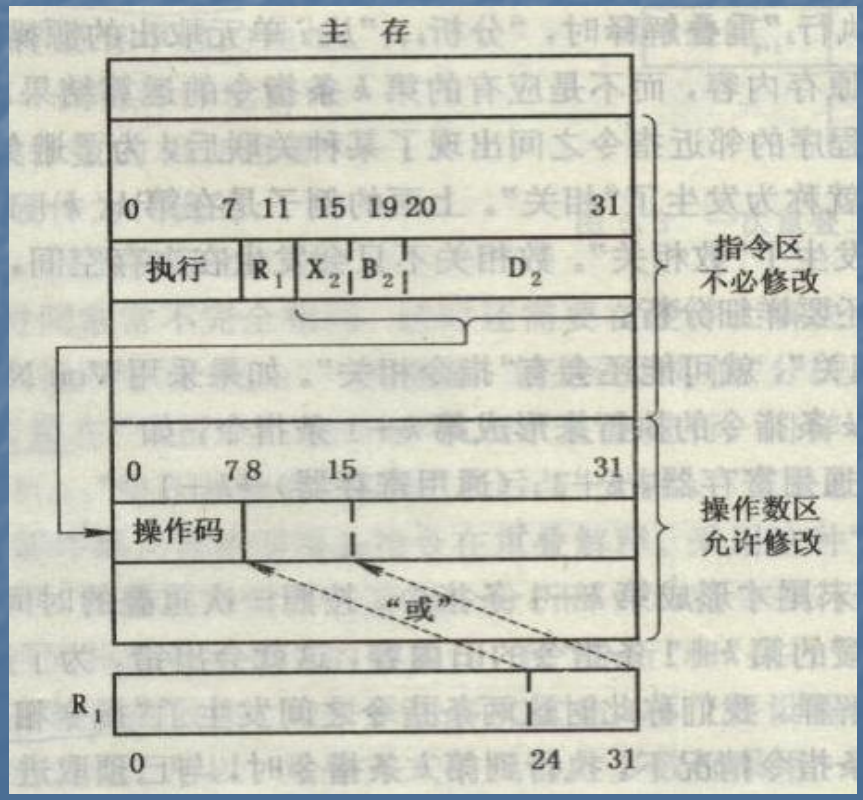
取址和分析在时间上的重叠

（独立的指令分析部件和指令的执行部件　同步的控制）

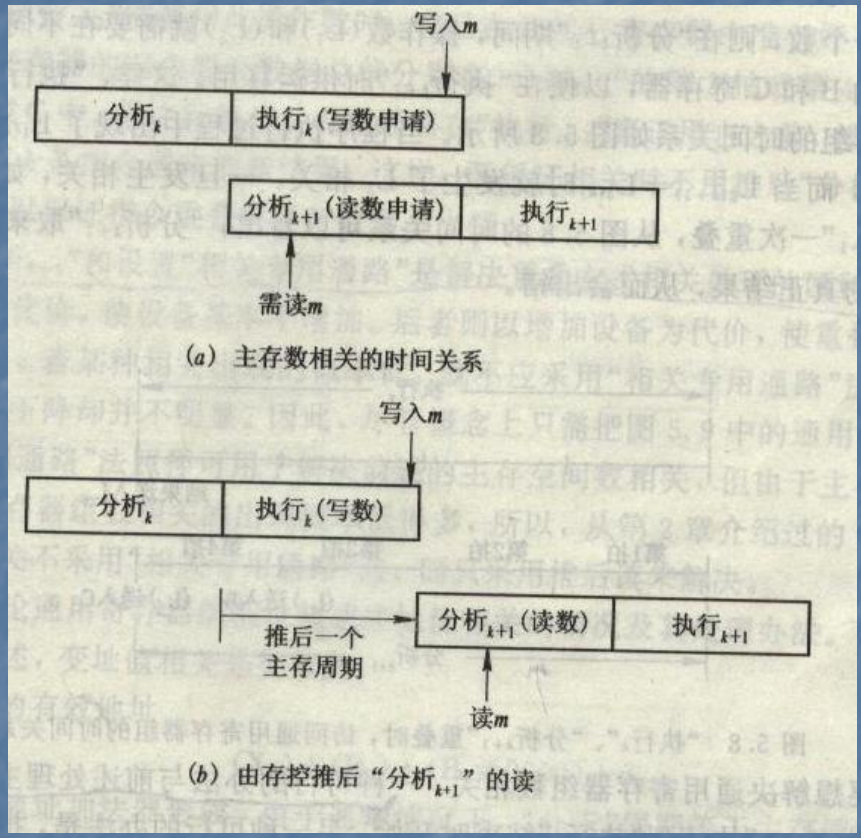
需解决的控制问题（条件转移　数相关 令相关）

5.1.2 相关处理

1．指令相关的处理



2.主存空间数相关的处理



3.通用寄存器组相关的处理（解决方法）

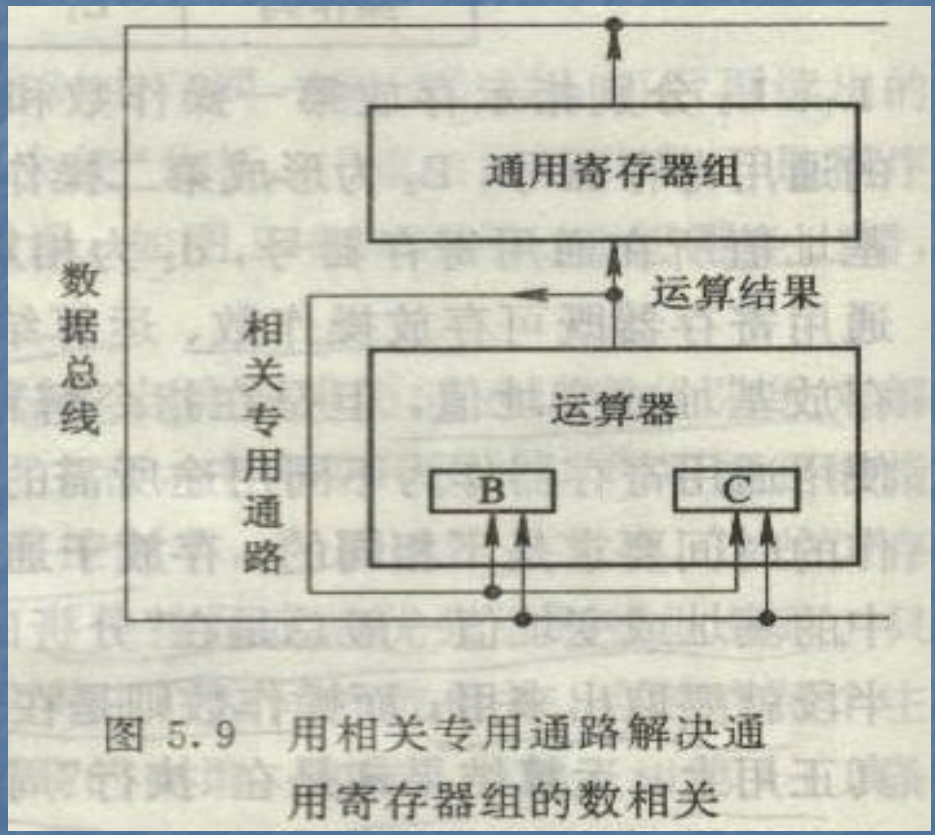


图 5.9　用相关专用通路解决通用寄存器组的数相关

- 推后“分析$_{k+1}$”和设置“相关专用通路”是解决重叠方式相关处理的两种基本方法

前者降低速度为代价　后者增加设备为代价

5.2 流水方式

- 5.2.1 基本概念

1.流水是重叠的引申

差别：一次重叠只是把一条指令的解释分解为两个子过程，而流水则是分解成更多个子过程，前者同时解释两条指令，后者可同时解释多条指令　　提高了吞吐率

流水的最大吞吐率是指当流水线正常满负荷流动时，才会每隔⊿t流出一个结果所达到的吞吐率

流水线的建立期间没有任何结果

- 2.流水线的分类

不同角度的不同分类

1）按流水处理的级别不同

向下扩展和向上扩展思路

部件级　处理级　系统级

2）按流水线具有功能的多少不同

单功能流水线　多功能流水线

3）按多功能流水线的各段能否允许同时用于多种不同功能联接流水

静态流水线　动态流水线

4）机器具有的数据表示

标量流水处理机　　向量流水处理机

5）流水线中各段之间是否有反馈回路

线性流水　　非线性流水

- 5.2.2　流水线处理机的主要性能
- 吞吐率

是流水线单位时间里能流出的任务数或结果数

$TP_{max}=1/\max\{\Delta t_1, \Delta t_2, \Delta t_3, \Delta t_4\}$

瓶颈子过程

例：流水线m段，各段经过时间均为Δt_0

流水建立时间：　$T_0=m\Delta t_0$

完成n个任务的解释时间：

$$T=m\Delta t_0+(n-1)t_0$$

流水方式工作的加速比：

$$Sp=n*m*\Delta t_0/(m\Delta t_0+(n-1)t_0)$$

$$=m/(1+(m-1)/n)$$

特例：各段经过的时间Δt不等时的实际吞吐率，加速比。

- 效率

是指流水线中的设备实际使用时间占整个运行时间之比

流水线各段经过时间均为Δt_0时，流水线的吞吐率正比于效率

- 流水线工作举例

- 5.2.3　流水机器的相关处理和控制机构

局部性相关的处理

全局性相关的处理

1. 猜测法
2. 加快和提前形成条件码
3. 采取延迟转移
4. 加快短循环程序的处理

流水机器的中断处理(不精确断点　精确断点)

流水线调度

线性流水线的每个任务不会争用同一个流水段

非线性流水线因为段间设置有反馈回路，一个任务在流水的全过程中，可能会多次通过同一段或越过某些段，会发生几个任务争用同一段的功能使用冲突现象

单功能非线性流水的调度

预约表 --〉 禁止表 --〉（N-1位）冲突向量

- 例：

若第二个任务在间隔2拍时流入，冲突向量（ 10110001 ）

第一个任务的冲突向量由（10110001）右移2位变为（00101100），

两个任务的冲突向量按位或，结果（10111101）

则第三个任务只能在第二个任务流入流水线后，隔2拍或7拍流入

流水线状态转移图 最佳调度方案

- 多功能流水线

对于一个多功能流水线，只需要将对应每种功能的预约表都重叠在一起

$M_A = \begin{pmatrix} 0110 \\ 1010 \end{pmatrix}$　　$M_B = \begin{pmatrix} 1011 \\ 0110 \end{pmatrix}$

按A功能流入一个任务后，根据V_{AA}的(0110)知道每隔1拍或4拍流入一个A功能任务，形成新的冲突矩阵：

$M_A = \begin{pmatrix} 0111 \\ 1111 \end{pmatrix}$

5.3 向量的流水处理与向量流水处理机

- 向量中的各个元素之间很少相关，容易发挥流水的效能
- 向量的数据表示与流水技术结合构成向量流水处理机

5.3.1向量的流水处理

- 5.3.2向量流水处理机
- 向量处理机的指令系统
- 向量处理机的结构
- 超级向量流水处理机举例

5.4指令级高度并行的超级处理机

- 超标量处理机
- 超常指令字处理机
- 超流水线处理机

第6章 并行处理机和相联处理机

并行处理机

相联处理机

6.1 并行处理机原理

6.1.1并行处理机的构形与特点

- 构形：分布式存贮器的并行处理机构形
  集中式共享存贮器的并行处理机构形
- 特点

- 6.1.2 并行处理机的算法
- ILLIAC IV的处理单元阵列结构
- 阵列处理机的算法举例

1 有限差分问题

2 矩阵加

3 矩阵乘

4 累加和

- 6.1.3 SIMD计算机的互连网络

1 互联网络的设计目标及互连函数

2 基本的单级互连网络

（立方体单级网络

PM2I单级网络

混洗交换单级网络）

3 多级互连网络

（多级立方体网络

多级PM2I网络

多级混洗交换网络）

4 全排列网络

- 6.1.4 并行存贮器的无冲突访问

6.2 并行处理机举例

- ILLIAC IV阵列处理机
- BSP 科学处理机
- MPP位平面阵列处理机
- CM连接机

6.3 相联处理机

- 相联处理机和相联存贮器的组成

1.相联处理机的特点和组成

2.相联存贮器的组成及相联处理机的结构类型

- 相联检索算法

1.全等查找算法

2.最大值查找算法

3. 幅值比较查找算法

4.其他算法

相联处理机结构举例