# 1.2 计算机体系结构

## 1.2.1 计算机体系发展结构

**1.计算机系统结构概述**

1964年，阿姆达尔在介绍IBM360系统时指出：计算机体系结构是站在程序员的角度所看到的计算机属性，即程序员要能编写出可在机器上正确运行的程序所必须要了解的概念性结构和功能特性。

1982年，梅尔斯在其所著的《计算机体系结构的进展》一书定义了组成计算机系统的若干层次，每一层都提供一定的功能支持它上面的一层，并把不同层之间的界面定义为某种类型的体系结构。

1984年，拜尔在一篇题为“计算机体系结构”的文章中给出了一个含义更加广泛的定义：体系结构由结构、组织、实现、性能四个基本方面组成。其中，结构指计算机系统各种硬件的互连，组织指各种部件的动态联系与管理，实现指各模块设计的组装完成，性能指计算机系统的行为表现。

计算机体系结构、计算机组织和计算机实现三者的关系如下：

1. 计算机体系结构是指计算机的概念性结构和功能属性。
2. 计算机组织是指计算机体系结构的逻辑实现。
3. 计算机实现是指计算机组织的物理实现。

**2.计算机体系结构分类**

1）从宏观上按处理机的数量进行分类：

·单处理系统。利用一个处理单元和其他外部设备结合起来，实现存储、计算、通信、输入与输出等功能的系统。

·并行处理与多处理系统。为了充分发挥问题求解过程中处理的并发性，将两个以上的处理机互连起来，彼此进行通信协调，以便共同求解一个大问题的计算机系统。

·分布式处理系统。指物理上远距离而松耦合的多计算机系统。其中，物理上的远距离意味着通信时间与处理时间相比已不可忽略，在通信线路上的数据传输速率要比在处理机内部内部总线上传输慢得多，这也正是松耦合的含义。

1. 从微观上按并行程度分：

·Flynn分类法。1966年，Flynn提出了按指令流和数据流的多少进行分类。指令流为机器执行的指令序列，数据流是由指令调用的数据序列，Flynn把计算机系统的结构分为单指令流单数据流，单指令流多数据流，多指令流单数据流和多指令流多数据流4类。

·冯泽云分类法。按并行度对各种计算机系统进行结构分类。所谓最大并行度Pm是指计算机系统在单位时间内能够处理的最大二进制位数。分为字串行位串行计算机、字并行位串行计算机、字串行位并行计算机和字并行位计算机。

·Handler分类法。把计算机的硬件结构分为3个层次：处理机级，每个处理机的算逻单元级、每个算逻单元中的逻辑门电路级。

·Kuck分类法。用指令流和执行流及其多重性来描述计算机系统控制结构的特征。把系统结构分为单指令流单执行流、单指令流多执行流、多指令流单执行流和多指令流多执行流4类。

补充：目前常用的是1966年出现的Flynn法进行分类。

1. 单指令流单数据流（SISD）

通常由一个处理器和一个寄存器组成。

1. 单指令流多数据流（SIMD）

通常由一个指令控制部件、多个处理器和多个存储器组成。各处理器和各存储器之间通过系统内部的互连逻辑电路进行通信。在程序运行时由指令控制部件向各个处理器传送同一条指令，处理器执行指令时所需的数据是从存储器中取的，各处理器所处理的数据是各不相同的，这就是多数据流。

1. 多指令流单数据流（MISD）

MISD计算机系统在同一时刻执行多条指令，但处理同一个数据。大多数人认为能列在这一系列的计算机很少或根本不存在。

1. 多指令流多数据流（MIMD）

由多台处理器和多个存储器组成，并有一个系统内部的互连逻辑电路实现各处理器和各存储器之间的通信。每台处理器执行各自的指令，存取各自的数据。

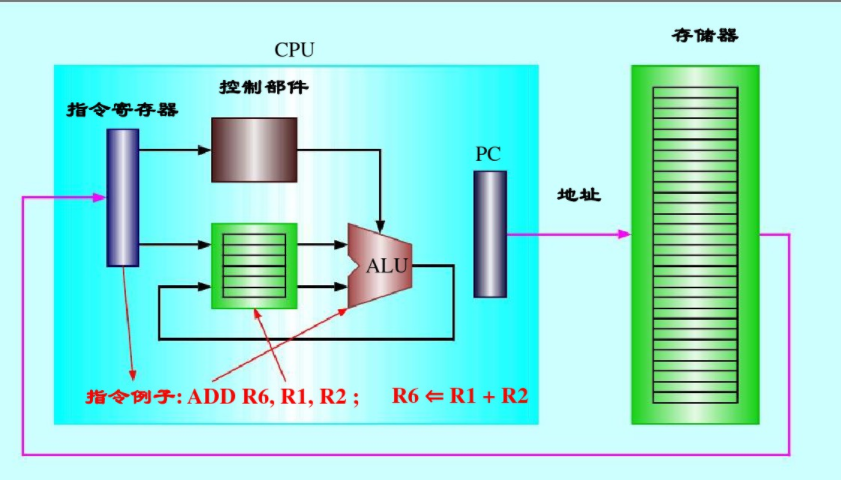
# 指令系统

## 5.1 指令系统的发展

计算机的性能与它所设置的指令系统有很大的关系，而指令系统的设置又与机器的硬件结构密切相关。

系列计算机：基本指令系统相同、基本体系结构相同的一系列计算机。

每一条机器语言的语句称为机器指令，而又将全部机器指令的集合称为机器的指令系统



## 5.2 指令格式

### 5.2.1 指令格式

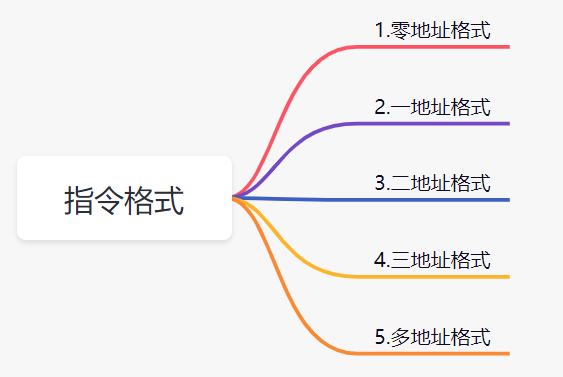
计算机是通过执行指令来处理各种数据的。为了指出数据的来源、操作结果的去向及所执行的操作，一条指令一般包含下列信息。

1. 操作码。具体说明操作的性质及功能。一台计算机可能有几十条至几百条指令，每一条指令都有一个对应的操作码，计算机通过识别该操作码来完成不同操作。
2. 操作数的地址。CPU通过该地址就可以取得所需的操作数。
3. 操作结果的存储地址。把对操作数的处理所产生的结果保存在该地址中。
4. 下一条指令的地址。当程序顺序执行时，下一条指令的地址由程序计数器（PC）给出，仅当改变程序的运行顺序（如转移、调用子程序）时，下一条指令的地址才由指令给出。

从上述分析可知，一条指令包括两种信息即操作码和地址码。

操作码表示该指令所要完成的操作（如加、减、乘、除、数据传送等），其长度反应了指令系统中的指令条数；

地址码描述该指令的源操作数的地址。这个地址可以是主存的地址，也可以是寄存器的地址，甚至可以是I/O设备的地址。



根据地址码部分所给出的地址的个数，指令可分为如下格式：

|  |
| --- |
| OPCODE |

·**零地址指令**

格式：

OPCODE-操作码。

指令中只有操作码，而没有操作数或没有操作数地址。这种指令有两种可能:

1. 无须任何操作数。如空操作数、停机命令等。
2. 操作数地址是默认的。如堆栈结构计算机的运算命令，所需的操作数默认在堆栈中，有的计算机指令将操作数或操作数地址安排在默认的寄存器中。

·**一地址指令**

格式：

|  |  |
| --- | --- |
| OPCODE | A |

OPCODE-操作码

1. 操作数的存储器地址或寄存器地址

指令只给出一个地址，该地址既是操作数的地址，又是操作结果的存储地址。如加1，减1和移位等单操作数指令均采用这种格式。

·**二地址指令**

格式：

|  |  |  |
| --- | --- | --- |
| OPCODE | A1 | A2 |

OPCODE-操作码

A1-第一个源操作数的存储器地址或寄存器地址

A2-第二个源操作数和存放结果的存储器地址或寄存器地址

这是最常见的指令格式，两个地址指出两个源操作数地址，其中一个还是存放结果的目的地址。对两个源操作数进行操作码所规定的操作后，将结果存入目的地址。

·**三地址命令**

格式：

|  |  |  |  |
| --- | --- | --- | --- |
| OPCODE | A1 | A2 | A3 |

OPCODE-操作码

对A1，A2地址指出的两个源操作数进行操作，结果存入A3中。

·**多地址指令**

在某些性能较强的计算机中，往往设置处理成批数据的指令。

计算机中，指令和数据一样都是以二进制码的形式存储的，从表面来看两者没有区别，但是，指令的地址是由程序计数器（PC）规定的，而数据的地址是由指令规定的，在CPU控制下访存绝对不会将指令和数据混淆。

### 5.2.2 指令操作码的扩展技术

指令操作码的长度决定了指令系统中完成不同操作的指令条数。若某机器的操作码长度固定为K位，则它最多只能有2^K条不同指令。指令操作码通常有两种格式，一种是**固定格式**，即操作码的长度固定，且集中放在指令字的一个字段中。在字长较长的计算机以及RISC上广泛使用。另一种是**可变格式**，即操作码的长度可变，且分散地放在指令字的不同字段中。在字长较短的计算机。

### 5.2.3 指令长度与数据字长的关系

数据字长决定了计算机的运算精度，字长越长，计算机的运算精度越高，因此，高性能计算机字长较长，其次，地址码长度决定了指令直接寻址能力，若为n位，则给出的n位直接地址可寻址2^n个字或字节。

数据字长：计算机数据存储所占用的位数。

## 5.3寻址方式

寻址方式分为**指令寻址**和**数据寻址**两大类。

1. **指令寻址**

指令寻址分为**顺序寻址**和**跳跃寻址**两种。

顺序寻址可通过程序计数器PC加1，自动形成下一条指令的地址；

跳跃寻址则通过转移类指令实现。

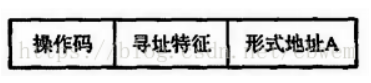
1. **数据寻址**

指令的地址码字段通常都不代表操作数的真实地址，把它称为形式地址，记作A。操作数的真实地址称为有效地址，记作EA，它是由寻址方式和形式地址共同来确定的。

|  |  |  |
| --- | --- | --- |
| 操作码 | 寻址特征 | 形式地址A |

**立即寻址**

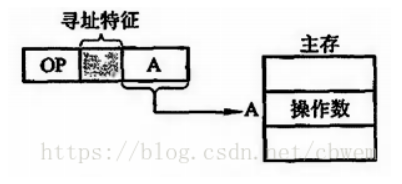
立即寻址的特点是操作数本身设在指令字内，即形式地址A不是操作数的地址，而是操作本身，又称为立即数。数据是采用补码的形式存放。



优点在于只要取出指令，便可立即获得操作数，这种指令在执行阶段不必再访问存储器。A的位数限制了这类指令所能表述的立即数的范围。

**直接寻址**

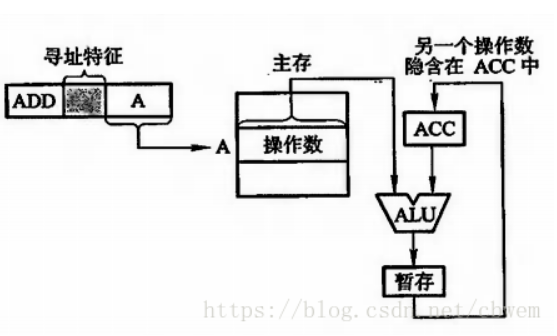
直接寻址的特点是指令字中的形式地址A就是操作数的真实地址EA，即EA=A。



它的优点是寻找操作数比较简单，也不需要专门计算操作数的地址，在指令执行阶段对主存只访问一次。缺点在于A的位数限制了操作数的寻址范围，而且必须修改A的值，才能修改操作数的地址。

**隐含寻址**

隐含寻址是指指令字中不明显地给出操作数的地址，其操作数的地址隐含在操作码或某个寄存器中。



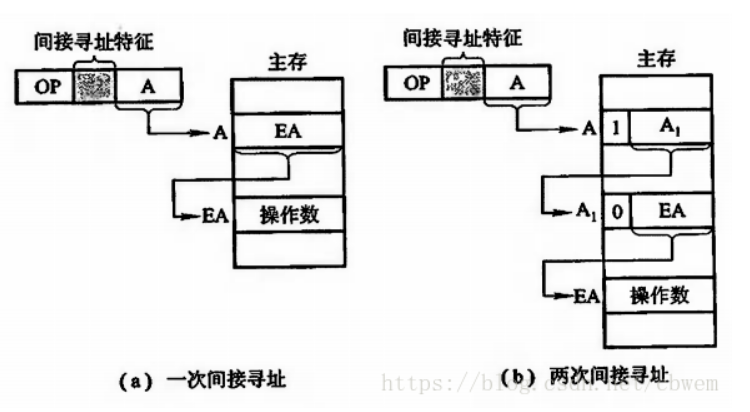
由于隐含寻址在指令字中少了一个地址，因此这种寻址方式的指令有利于缩短指令字长。

指令字长：计算机指令字的位数

**间接寻址**

倘若指令字中的形式地址不直接指出操作数的地址，而是指出操作数有效地址所在的存储单元地址，也就是有效地址是由形式地址间接提供的，即为间接寻址，即EA=(A)。

当多次间接寻址时，可用存储字的首位来标志间接寻址是否结束。当存储字首位“1”时，表明还需继续访存寻址；当存储字首位为“0”时，表明该存储字即为EA。



优点：这种寻址方式与直接寻址相比，扩大了操作数的寻址范围，因为A的位数通常小于指令字长，而存储字长可与指令字长相等。另外它便于编制程序。只要在调用前先将返回地址存入子程序最末条指令的形式地址A的存储单元内，便可准确返回到原程序断点。

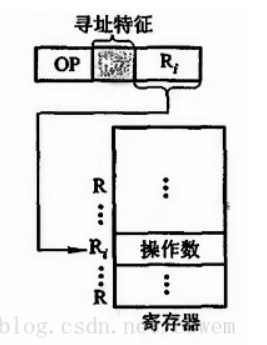
缺点：指令的执行阶段需要访存两次（一次间接寻址）或多次（多次间接寻址），致使指令执行时间延长。

存储字长：存储器中一个存储单元（存储地址）所存储的二进制代码的位数，即存储器中的MDR的位数。

MDR：存储器数据寄存器

**寄存器寻址**

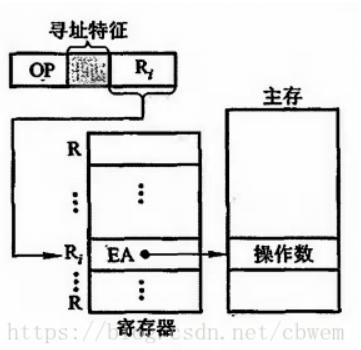
在寄存器寻址的指令字中，地址码字段直接指出了寄存器编号，其操作数在由Ri所指的寄存器内，即EA=Ri。



由于操作数不在主存中，故寄存器寻址在指令执行阶段无须访存，减少了执行时间。由于地址字段只需指明寄存器编号（计算机中寄存器数有限），故指令字较短，节省了存储空间。

**寄存器间接寻址**

Ri中的内容不是操作数，而是操作数所在主存单元的地址号，即有效地址EA=(Ri)

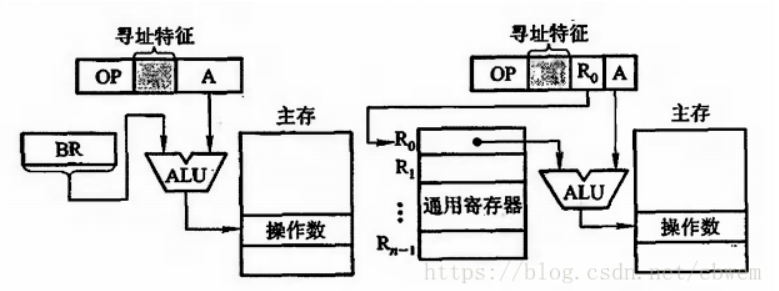


与寄存器寻址相比，指令的执行阶段还需要访问主存。因有效地址不是存放在存储单元中，故称其为寄存器间接寻址，它比间接寻址少访存一次。

**基址寻址**

基址寻址需设有基址寄存器BR，其操作数的有效地址EA等于指令字中的形式地址与基址寄存器中的内容（称为基地址）相加，即EA=A+(BR).

基址寄存器可采用隐式的和显示的两种。所谓隐式，是在计算机内专门设有一个基址寄存器BR，使用时用户不必明显指出该基址寄存器，只需由指令的寻址特征位反映出基址寻址即可。显式在一组通用寄存器里，由用户名明确指出哪个寄存器用作基址寄存器，存放基地址。



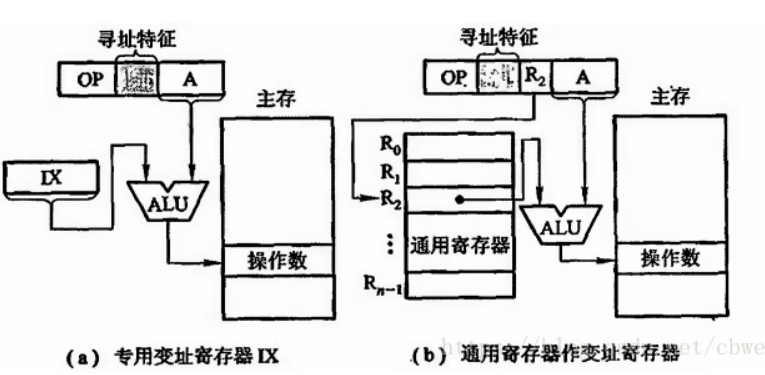
基址寻址可以扩大操作数的寻址范围，因基址寄存器的位数可以大于形式地址A的位数。当主存容量较大时，若采用直接寻址，因受A的位数限制，无法对主存所有单元进行访问，但采用基址寻址便可实现对主存空间的更大范围寻访。

用户可不必考虑自己的程序存于主存的哪一空间区域，完全可由操作系统或管理程序根据主存的使用状况，赋予基址寄存器内一个初始值（即基地址），便可将用户程序的逻辑地址转化为主存的物理地址（实际地址），把用户程序安置于主存的某一空间区域。

**变址寻址**

其有效地址EA等于指令字中的形式地址A与变址寄存器IX的内容相加之和，

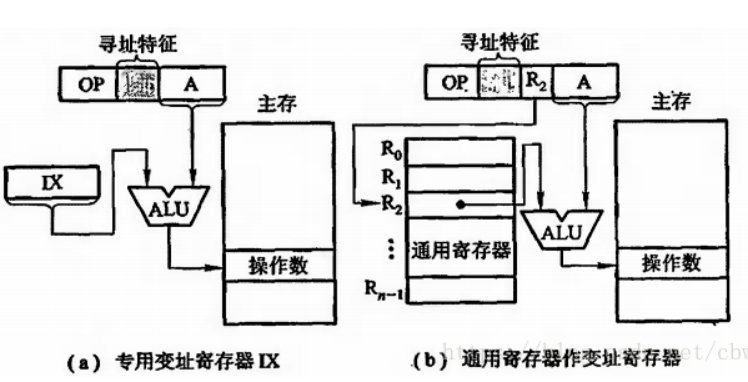
即E A = A + ( I X ) 。



只要变址寄存器位数足够，也可扩大操作数的寻址范围。

**相对寻址**

相对寻址的有效地址是将程序计数器PC的内容与指令字中的形式地址A相加而成，即E A = ( P C ) + A 。



**堆栈寻址**

堆栈寻址要求计算机中设有堆栈。堆栈既可用寄存器组（称为硬堆栈）来实现，也可利用主存的一部分空间作堆栈（称为软堆栈）。堆栈的运行方式为先进后出或先进先出两种，先进后出型堆栈的操作数只能从一个口进行读或写。堆栈寻址也可视为一种隐含寻址，其操作数的地址总被隐含在SP中。堆栈寻址就其本质也可视为寄存器间接寻址，因SP可视为寄存器，它存放着操作数的有效地址。

## 5.4 指令类型

### 5.4.1 指令的分类及功能

**1.算术逻辑运算指令**

对整型数据的处理

通常根据算术运算的结果置状态位，一般有Z（结果为0）、N（结果为负）、V（结果溢出）、C（产生仅为或错位）4个状态位。

**2.移位操作指令**

移位操作指令分为算术移位、逻辑移位和循环移位3种，可以将操作数左移或右移若干位。

**3.浮点数运算指令**

高级语言中的实数（real）经常是先转换成浮点数的形式而后再进行处理。

**4.十进制运算指令**

在人机交互作用时，输入输出的数据都是以十进制形式表示的。

**5.字符串处理指令**

字符串处理指令是一种非数值处理指令，一般包括字符串传送、字符串比较、字符串查询和字符串转换等指令。

**6.数据传送指令**

这类指令用以实现寄存器与寄存器、寄存器与存储器（主存）单元、存储器单元与存储器单元之间的数据传送。一次可以传送一个数据或一批数据。数据传送时，数据从源地址传送到目的地址，而源地址中的数据保持不变，因此是数据复制。

**7.转移指令**

这类指令用以控制程序流的转移。在大多数情况下，计算机是按顺序方式执行程序的，但是也经常会遇到离开原来的顺序转移到另一段程序或循环执行某一段程序的情况。

按转移的性质。转移指令分为：

1）无条件转移指令与条件转移指令

无条件转移指令不受任何条件约束，直接把程序转移到指令所规定的目的地，在那里继续执行程序。条件转移指令则根据计算机处理结果来决定程序如何执行。它先测试根据处理结果设置的条件码。

1. 调用指令与返回指令
2. 陷阱与陷阱指令

**8.堆栈及堆栈操作指令**

堆栈是由若干个连续的存储单元组成的先进后出（FILO）存储区，第一个送入堆栈中的数据存放在栈底，最近送入堆栈中的数据存放在栈顶。栈底是固定不变的，而栈顶却是随着数据的入栈和出栈在不断变化。为了表示栈顶的位置，有一个寄存器或存储器单元用于指出栈顶的地址，称为堆栈指针（SP）。堆栈操作的对象与堆栈指针有关。

**9.输入输出（I/O）指令**

计算机所处理的一切原始数据和所执行的程序（除了固化在ROM中的以外）均来自外部设备的输入，处理结果需通过外部设备输出。输入输出指令的一般格式如下：

|  |  |  |
| --- | --- | --- |
| OP | REG | A |

其中OP是操作码，表示I/O指令。REG是寄存器地址，用于指定CPU与外部设备交换数据的寄存器。A是外部设备中的寄存器地址或设备码，其长度一般为8-16位。输入就是从A所指向的寄存器取出放入REG指向的寄存器。输出则相反。

**10.特权指令**

某些指令使用不当会剖坏系统或其他用户信息，因此为了安全起见，这类指令只能用于操作系统或其它系统软件，而不提供给用户使用，称为特权软件。

特权指令主要用于系统资源的分配和管理，包括改变系统的工作方式、检测用户的访问权限、修改虚拟存储器管理的段表，页表和完成任务的创建和切换等。

**11.其他指令**

（1）向量指令

对向量或矩阵数据求和、求积的指令。

（2）多处理机指令。

在多处理机或多处理器系统中为了管理共享的公共资源和相互通信，一般设置“测试与设定”或“数据交换”指令，这些指令最主要的特点是在执行的过程中不能被打断。

（3）控制指令

包括等待命令、停机命令、空操作命令、开中断、关中断和置条件码指令等。

当用户程序执行完毕时，可安排一条停机指令，此时机器不再继续执行程序。但在多用户情况下，则不允许停机，因为其他用户程序可能正在等待，此时通常让机器处于动态停机态：执行等待指令或执行只有1-2条指令的小循环程序。

### 5.4.2 双字长运算（子程序举例）

### 5.4.3 指令系统的兼容

## 5.5 精简指令系统计算机（RISC）和复杂指令计算机（CISC）

CISC和RISC是指令集发展的两种途径。

1. CISC的基本思想是进一步增强原有指令的功能，用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，导致机器的指令系统越来越庞大复杂。

CISC的主要弊端：

①指令集过分庞杂

②微程序技术是核心，每条复杂指令都需要一段解释性微程序。

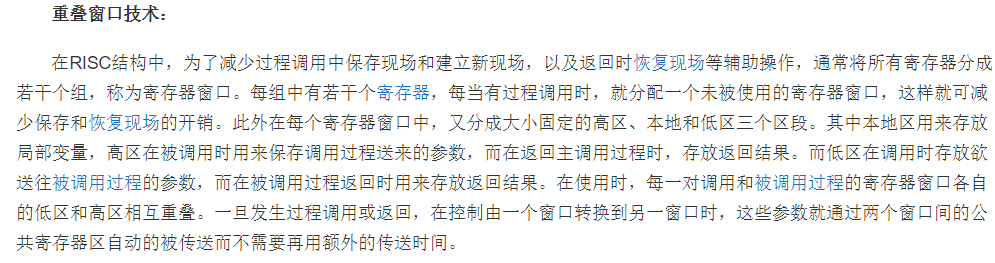
③CISC强调中断控制

④芯片设计困难

1. RISC的基本思想是通过减少指令总数和简化指令功能降低硬件设计的复杂度，使指令能单周期执行，并通过优化编译提高指令的执行速度，采用硬布线控制逻辑优化编译程序。

RISC的关键技术如下：

①重叠寄存器窗口技术（关键技术）



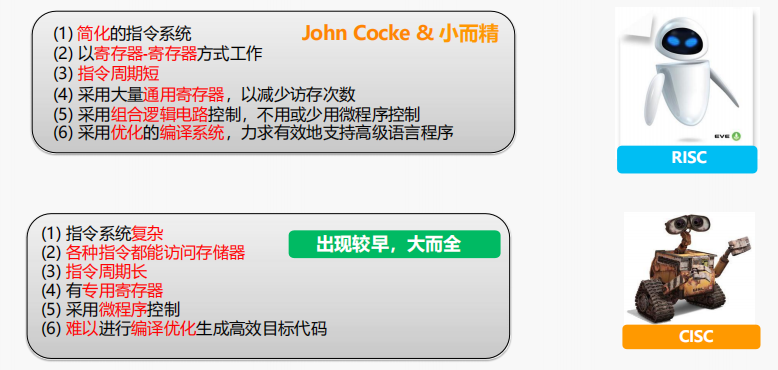
②优化编译技术

③超流水及超标量技术

④硬布线逻辑与微程序相结合在微程序技术中。

微程序控制技术：

<https://blog.csdn.net/iostream992/article/details/83869906/>



## 5.6 指令系统举例

## 5.7 机器语言、汇编语言和高级语言

本章讨论的指令系统以二进制码来表示操作码、操作数或地址，即为机器语言。当用助记符来取代二进制，就构成了汇编语言的基本部分。

用汇编语言编写程序的三个缺陷：

1. 汇编语言基本与机器语言对应，程序员使用它需要十分熟悉计算机硬件结构的配置，指令系统和寻址方式。
2. 汇编语言编写的程序与问题的描述相差的远，其可读性差。
3. 汇编语言依赖于计算机的硬件结构和指令系统，而不同的机器有不同的结构和指令，可移植性差

C语言等高级语言就是为了克服汇编语言的这些缺陷发展而来的，但高级语言也存在两个缺陷：

1. 用高级语言编写的程序，必须翻译成机器语言才能执行，这一工作通常由计算机执行编译程序来完成。由于编译过程既复杂又死板，翻译出的机器语言非常冗长，效率不如直接输入机器语言。
2. 由于高级语言程序“看不见”机器的硬件结构，因而很难用它来编写需访问机器硬件资源的系统软件或设备控制软件。

# 指令系统软考内容补充：

一个处理器支持的指令和指令的字节级编码称为其指令集体系结构，不同的处理器族支持不同的指令集体系结构（ISA），不同的处理器族支持不同的指令集系结构。

1. 指令集体系结构的分类

从体系结构的观点对指令集进行分类，可以根据下述5个方面。

1. 操作数在CPU的存储方式。
2. 显式操作数的数量
3. 操作数的位置
4. 指令的操作
5. 操作数的类型和大小

2）按暂存机制分类，根据在CPU内部存储操作数的区别，可以把指令集体系分为3类：堆栈，累加器和寄存器组。

通用寄存器（GPR）的关键性优点是编译程序能有效地使用寄存器，无论是计算表达式的值，还是从全局变量的角度使用寄存器来保存变量的值。

在求解表达式时，寄存器比堆栈或累加器能提供更灵活的次序。

当变量分配给寄存器时，访存流量（Memory Traffic）就会减少。

## 指令的流水处理

**1.指令的控制方式**。指令的控制方式有顺序方式、重叠方式和流水方式3种。

1）顺序方式。

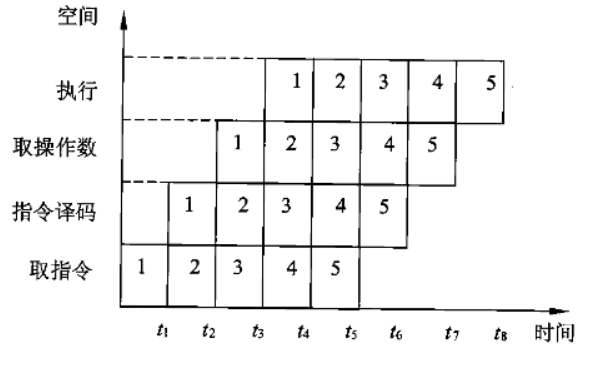
顺序方式是指各条机器指令之间顺序串行地执行，执行完一条指令后才取下一条指令，而且每条机器指令内部的各个微操作也是顺序串行地执行。这种方式的优点是控制简单。缺点就是速度慢，机器各部件的利用率低。

2）重叠方式

重叠方式是指在解释第K条指令的操作完成之前就可以开始解释第K+1条指令。通常采用的是一次重叠，即在任何时候，指令分析部件和指令执行部件都只有相邻两条指令在重叠解释。

3）流水方式

流水方式是模仿工业生产过程的流水线而提出的一种指令控制方式。流水技术是把并行性或并发性嵌入到计算机系统里的一种形式，它把复杂的顺序处理过程分解为若干子过程，每个子过程能在专用的独立模块上有效地并发工作。



在概念上，“流水”可以看作是“重叠”的延申。差别仅在于“一次重叠”是把一条指令解释分解为两个子过程，而“流水”则是分解为更多的子过程。

**2.流水线的种类**

①从流水的级别上，可分为部件级、处理机级以及系统级的流水。

②从流水的功能上，可分为单功能流水和多功能流水。

③从流水的连接上，可分为静态流水和动态流水。

④从流水是否有反馈回路，可分为线性流水线和非线性流水线。

⑤从流水的流动顺序上，可分为同步流水线和异步流水线。

⑥从流水线的数据表示上，可分为标量流水线和向量流水线。

**3.流水的相关处理**

由于流水时机器同时解释多条指令，这些指令可能有对同一主存单元或同一寄存器的“先写后读”的要求，这时就出现了相关。这些相关包括指令相关、访存操作数相关以及通用寄存器相关等，它只影响相关的几条指令，而且最多影响流水线的某些段推后工作，并不会改动指令缓冲器中预取到的指令内容，影响是局部的，所以称为局部性相关。

解决局部性相关的两种方法：**推后法**和**通路法**。

推后法是推后对相关单元的读，直至写入完成。

通路法设置相关专用道路，使得不必先把运算结果写入相关存储单元再读出后才能使用，而是经过相关专用通路直接使用运算结果，以加快速度。

转移指令（尤其是条件转移指令）与它后面的指令之间存在关联，使之不能同时解释。执行转移指令时，可能会改动指令缓冲器中预读取到的指令内容，从而会造成流水线吞吐率和效率下降，称为全局性相关。

解决全局性相关有三种方法：**猜测转移分支**，**加快和提前形成条件码**、**加快短循环程序的处理**。

条件转移指令有两个分支，一个分支是按原来的顺序继续执行下去，称为转移不成功分支；另一个分支是按转移后的新指令序列执行，称为转移成功分支。许多流水机器都猜选转移不成功分支，若猜对的几率很大，流水线的吞吐率和效率就会比不采用猜测法时高得多。

尽早获得条件码以便对流水线简化条件转移的处理。例如，一个乘法运算所需的时间较长，但在运算之前就能知道其结果为正或为负，或者是否为0，因此，加快单条指令内部条件码的形成，或者在一段程序内提前形成条件码，对转移问题的顺利解决是很有好处的。

由于程序中广泛采用循环结构，因此流水线大多采用特殊措施以加快循环程序的处理。例如，使整个循环程序都放入指令缓冲存储器中，对提高流水效率和吞吐率均有明显效果。中断和转移一样，也会引起流水线断流。好在中断出现的概率要比条件转移出现的概率低得多，因此只要处理好断点现场保护及中断后的恢复，尽量缩短断流即可。

RISC中采用的流水技术有3种，**超流水线**、**超标量**以及**超长指令字**。

①超流水线技术。它通过细化流水、增加级数和提高主频，使得在每个机器周期内能完成一个甚至两个浮点操作。其实质是以时间换空间。超流水机器的特征是在所有的功能单元都采用流水，并有更高的时钟频率和更深的流水深度。由于它只限于指令级的并行，所有超流水的CPI（每个指令所需的机器周期数）值稍高。

②超标量技术。它通过内装多条流水线来同时执行多个处理，其时钟频率虽然与一般流水接近，却有更小的CPI。其实质是以空间换时间。

③超长指令字技术（VLIW）。VLIW和超标量都是20世纪80年代出现的概念，其共同点是要同时执行多条指令，其不同在于超标量依靠硬件来实现并行处理的调度，VLIW则充分发挥软件的作用，而使硬件简化，性能提高。VLIW有更小的CPI值，但需要有足够高的时钟频率。

4.吞吐率和流水建立时间。

吞吐率是指单位时间内流水线处理机流出的结果数。对指令而言，就是单位时间内执行的指令数。如果流水线的子过程所用时间不一样，则吞吐率p应为最长子过程的倒数，即

p = 1/max{t1,t2,...,tm}

流水线开始工作，需经过一定时间才能达到最大吞吐率，这就是建立时间。若m个子过程所用时间一样，均为t0，则建立时间T0 = m\*t0 。

## 阵列处理机、并行处理机和多处理机

并发性包括同时性和并发性。其中，同时性是指两个或两个以上的事件在同一时刻发生，并发性是指两个或两个以上的事件在同一时间间隔内连续发生。

从计算机信息处理的步骤和阶段的角度看，并行处理可分为如下几类。

1. 存储器并行操作
2. 处理器操作步骤并行（流水线处理机）
3. 处理器操作并行（阵列处理机）
4. 指令、任务、作业并行（多处理机、分布处理系统、计算机网络）
5. 阵列处理机

阵列处理机将重复设置的多个处理单元（PU）按一定方式连成阵列，在单个控制部件（CU）控制下，对分配给自己的数据进行处理，并行地完成一条指令所规定地操作。这是一种单指令流多数据流计算机，通过资源重复实现并行性。

1. 并行处理机

SIMD和MIMD是典型的并行计算机，SIMD有共享存储器和分布式存储器两种形式。

在具有共享存储器的SIMD结构，将若干个存储器构成统一的并行处理机存储器，通过互联网络ICN为整个并行系统的所有处理单元共享。其中，PE是处理单元，CU是控制部件，M为共享存储器，ICN为互联网络。

分布式存储器的SIMD结构。其中，PE是处理单元，CU是控制部件，PEM为局部存储器，ICN为互联网络。

1. 多处理机

多处理机系统就是由多台处理机组成的系统，每台处理机都有属于自己的控制部件，可以执行独立的程序，共享一个主存储器和所有的外部设备。它是多指令流多数据流计算机。在多处理机系统中，机间的互连技术决定了多处理机的性能。多处理机之间的互连要满足高频带、低成本、连接方式的多样性以及在不规则通信情况下连接的无冲突性。

1. 其他计算机

集群一般是指连接在一起的两个或多个计算机（结点）。集群计算机是一种并行或分布式处理系统，由很多连接在一起的独立计算机组成，像一个单集成的计算机资源一样协同工作，主要用来解决大型计算问题。