

机内代码及运算



第7章 计算机的体系结构和主要部件

也许有人认为系统设计师似乎不需要什么硬件的知识，计算机的硬件理论知识并非是空中楼阁，不是好看或者用来考试的，它是确实确实的每台计算机设计和制造的基础，而且在学习的过程中，我们能够发现许多在硬件上使用的原则在软件上使用同样有益。在很我们都能发现，计算机软件和硬件并非截然分开，而是有个此消彼长、相互促进的发展过程。的中央处理器是没有浮点运算功能，浮点运算需要使用软件实现，而后来，许多处理器有了浮点运算功能。另外，嵌入式系统的软件设计师必须考虑硬件的问题。

7.1 机内代码及运算

人人都知道计算机只处理二进制数据，二进制是最简单的进制方式，只有0和1两个基数，也就是说，计算机底层硬件只要能保持两个状态即可，这样使得计算机的底层设计变得简单，出错的概率也大为减小。当然二进制数据使得表示和保存数据的长度大大增加，但是大规模和超大规模的集成电路使得这成为次要的问题，人们能在越来越小的芯片空间里容纳越来越多的电路。

另一方面计算机为了使得处理方便，其内部存储数据的格式和我们看见的是不同的。

版权方授权希赛网发布，侵权必究

上一节 本书简介 下一节

数的进制

7.1.1 数的进制

1.进制的表示法

R进制，通常说法就是逢R进1.可以用的数为R个，分别是0,1,2,...，R-1.例如十进制数的基数为10,即可以用到的数码个数为10,它们是0,1,2,3,4,5,6,7,8,9.二进制数的基数为2,可用的数码个数为2,它们是0和1。

为了把不同的进制数分开表示，避免造成混淆，采用下标的方式来表示一个数的进制，如十进制数56表示为：（ 56 ）<sub>10</sub>,八进制数42表示为：（ 42 ）<sub>8</sub>。

对于任意一个R进制数，它的每一位数值等于该位的数码乘以该位的权数。权数由一个幂表示，即幂的底数是R,指数为k,k与该位和小数点之间的距离有关。当该位位于小数点左边，k值是该位和小数点之间数码的个数，而当该位位于小数点右边，k值是负值，其绝对值是该位和小数点之间数码的个数加1.

例如十进制数1234.56,其数值可计算如下：

$$1234.56=1\times10^3+2\times10^2+3\times10^1+4\times10^0+5\times10^{-1}+6\times10^{-2}$$

例如二进制数10100.01的值可计算如下：

$$10100.01=1\times2^4+1\times2^2+1\times2^{-2}$$

## 2.进制的转换

### 1) R进制数转换成十进制数

按照上面的表示法，即可计算出R进制数十进制的值。

### 2) 十进制数转换为R进制数

最常用的是"除以R取余法".例如将十进制数94转换为二进制数：

2   94	余 0
2   47	1
2   23	1
2   11	1
2   5	1
2   2	0
1	1

将所得的余数从低位到高位排列 ( 1011110 )<sub>2</sub>就是94的二进制数。

### 3) 二进制数与八进制数、十六进制数之间的转换

将二进制数转换为八进制数，只有将每3个二进制数转换为八进制数即可，将二进制数转换为十六进制数，只要将每4个二进制数转换为八进制数即可。将八进制数转换为二进制数，只要将每个八进制数转换为3位二进制数即可，将十六进制数转换为二进制数，只要将每个十六进制数转换为4位二进制数即可。上面的转换都是以小数点作为计算数码个数的起点。八进制数和十六进制数转换可先转换为二进制数，然后再转换为目标进制。

版权方授权希赛网发布，侵权必究

[上一节](#)

[本书简介](#)

[下一节](#)

## 原码、反码、补码、移码

### 7.1.2 原码、反码、补码、移码

一个正数的原码、补码、反码是相同的，负数则不同。先提一个问题，为什么在计算机中要使用这些编码方式呢？

#### 1.原码

将最高位用做符号位（0表示正数，1表示负数），其余各位代表数值本身的绝对值的表示形式。这种方式是最容易理解的。

例如，+11的原码是00001011,-11 的原码是10001011。

但是直接使用原码在计算时却会有麻烦，比如 ( 1 )<sub>10</sub> + ( -1 )<sub>10</sub> = 0,如果直接使用原码则：

$$( 00000001 )_2 + ( 10000001 )_2 = ( 10000010 )_2$$

这样计算的结果是-2,也就是说，使用原码直接参与计算可能会出现错误的结果。所以，原码的符号位不能直接参与计算，必须和其他位分开，这样会增加硬件的开销和复杂性。

#### 2.反码

正数的反码与原码相同。负数的反码符号位为1,其余各位为该数绝对值的原码按位取反。这个取

反的过程使得这种编码称为"反码"。

例如，-11的反码：11110100

同样对上面的加法，使用反码的结果是：

$$(00000001)_2 + (11111110)_2 = (11111111)_2$$

这样的结果是负0,而在人们普遍的观念中，0是不分正负的。反码的符号位可以直接参与计算，而且减法也可以转换为加法计算。

### 3.补码

正数的补码与原码相同。负数的补码是该数的反码加1,这个加1就是"补"。

例如，-11的补码：11110100+1 = 11110101

再次做加法是这样的：

$$(00000001)_2 + (11111111)_2 = (00000000)_2$$

直接使用补码进行计算的结果是正确的。注意到我们这里只是举例，并非证明。

对一个补码表示的数，要计算其原码，只要对它再次求补，可得该数的原码。

由于补码能使符号位与有效值部分一起参加运算，从而简化运算规则，同时它也使减法运算转换为加法运算，进一步简化计算机中运算器的电路，这使得在大部分计算机系统中，数据都使用补码表示。

版权方授权希赛网发布，侵权必究

[上一节](#)

[本书简介](#)

[下一节](#)

## 定点数和浮点数

### 7.1.3 定点数和浮点数

定点数和浮动数的区别在于如何对待小数点，在运算方式上也不相同，衡量一个计算机系统，定点运算和浮点运算是两个重要的指标。定点数的小数点是隐含的，固定在某个位置。如果该位置是在数的最低位之后，就是定点整数。定点数表示比较简单，运算规则也比较容易实现，但是当数值范围变化大时，使用定点数表示和运算就比较困难。

为了表示更大范围的数值，可以使用浮点数表示法。

在表示一个很大的数时，我们常常使用一种称为科学计数法的方式：

$$N = M \times R^e$$

其中M称为尾数，e是指数，R为基数。

浮点数就是使用这种方法来表示大范围的数，其中指数一般是2,8,16.而且对于特定机器而言，指数是固定不变的，所以在浮点数中指数并不出现。从这个表达式可以看出：浮点数表示的精度取决于尾数的宽度，范围取决于基数的大小和指数的宽度。

#### 1.格式化数

使用格式化数是提高浮点数有效位的方法。格式化的意思是把尾数前面加0,同时修改指数，这样在尾数位数固定的情况下，能提供最多的有效位来表示尾数。当指数小于能够表示的最小值时，这

个数称为机器零，此时会把尾数和指数同时清零。看到这里，应该能回答指数为什么常使用移码来表示问题了。

## 2. 定点数的算术运算和溢出处理

如前叙述，计算机中通常使用补码进行计算。两个正数相加，如果结果的符号位变成了1,则表示有溢出，同样两个负数相加，如果结果的符号位变成了0,那么也意味着溢出。如果是正数和负数相加，则不会出现溢出的情况。

判断处理的方法可以再增加一个符号位，称之为第一符号位，原来那个符号位变成了第二符号位。计算时两个符号位都参与计算，如果计算结果的两个符号位相同，表示没有溢出，如果不同，就表示出现了溢出。而第一符号位才是真正的符号。

也可以通过进位信号来判断，当结果的最高位和符号位的进位信号一致时（都有进位信号或都没有进位信号），则没有溢出，否则表示有溢出。

## 3. 定点数的逻辑运算

逻辑运算意味着各位的运算不产生进位，操作数的对应位独立计算。逻辑加实际就是按位"或"的计算，逻辑乘实际上是按位"与"的操作，逻辑非是按位"取反"。在校验码中，我们将接触到逻辑运算。

## 4. 浮点数的运算

浮动数运算过程比定点数复杂，包括以下过程。

### 1) 对阶

首先计算两个数的指数差，把指数小的向指数大的对齐，并将尾数右移指数差的位数，这样两个浮点数就完成了对阶的操作。可以看出，对阶的过程可能使得指数小的浮点数失去一些有效位。如果两个浮点数阶数相差很大，大于指数小的浮点数的尾数宽度，那么对阶后那个浮点数的尾数就变成了0,即当做机器零处理了。

### 2) 尾数计算

对阶完成后，两个浮点数尾数就如同定点数，计算过程同定点数计算。

### 3) 结果格式化

尾数计算后，可能会产生溢出，此时将尾数右移，同时指数加1,如果指数加1后发生了溢出，则表示两个浮点数的运算发生了溢出。

如果尾数计算没有溢出，则尾数不断左移，同时指数减1,直到尾数为格式化数。如果这个过程中，指数小于机器能表达的最小值，则将结果置"机器零",这种情况称为下溢。

版权方授权希赛网发布，侵权必究

[上一节](#)

[本书简介](#)

[下一节](#)

## BCD码及其运算

### 7.1.4 BCD码及其运算

人们习惯于十进制数，不习惯于二进制数。计算机只用二进制数。有一个折中的方法，这就是8421码。8421码使用4个二进制数来表示一位十进制数。比如十进制数的138的8421码表示为：

0001 0011 1000,所以这种表示方法也称为：二进制编码的十进制数，简称BCD码。这样比较容易阅读。在许多通信程序中，系统分析师都愿意采用BCD码来表示数据，这样在调试的过程中，比较容易读出其中的内容。

由于BCD码的特性，如果使用BCD码进行计算时，可能需要进行修正。

如 $3+5:0011 + 0101 = 1000$ ,正确。

而 $6+7:0110 + 0111 = 1101$ ,这已经不在BCD码的范围里，此时需要进行修正。修正的方法是本位加6,并向高位进1。

还有一种编码是将BCD码加3作为数字表示法，称为余3码，在使用余3码进行计算时，从最高位产生的进位是真正的进位，对每4位的结果，如果是小于等于9的，则减3,如果大于9,则加3。

版权方授权希赛网发布，侵权必究

上一节      本书简介      下一节

文字符号的编码

7.1.5 文字符号的编码

1.ASCII码

为了表示英文字母和其他一些符号、控制符，计算机中普遍采用的是ASCII码。它使用7位代表一个字符，包括了字母的大小写、数字、标点、控制符等。计算机通常使用8位一个字节来存储，其高位为0,如表7-1所示。

表7-1 ASCII码表

<div>0000</div> <div>0001</div> <div>0010</div> <div>0011</div> <div>0100</div> <div>0101</div> <div>0110</div> <div>0111</div> <div>1000</div> <div>1001</div> <div>1010</div> <div>1011</div> <div>1100</div> <div>1101</div> <div>1110</div> <div>1111</div>	000	001	010	011	100	101	110	111
	NUL	DLE	SP	0	@	P	'	P
	SOH	DC1	!	1	A	Q	A	Q
	STX	DC2	”	2	B	R	B	R
	ETX	DC3	#	3	C	S	C	S
	EOT	DC4	\$	4	D	T	D	T
	ENO	NAK	%	5	E	U	E	U
	ACK	SYN	&	6	F	V	F	V
	BEL	ETB	'	7	G	W	G	W
	BS	CAN	(	8	H	X	H	X
	HT	EM	)	9	I	Y	I	Y
	LF	SUB	*	:	J	Z	J	Z
	VT	ESC	+	;	K	[	K	{
	FF	FS	,	<	L	\	L	
	CR	GS	-	=	M	}	M	]
	SO	RS	.	>	N	!	N	~
	SI	US	/	?	O	~	o	DEL

汉字和拼音文字不同，拼音文字只需要定义少量的字母和符号的编码即可完成所有文字的保存、显示任务。而汉字存在大量的单字，为了让计算机能够处理汉字，必须对汉字进行单独的编码。

2.GB2312-80

GB2312-80全称是信息交换用汉字编码字符集基本集，于1980年发布。这种国家编码标准，已经得到了广泛的应用。它以两个字节表示一个汉字或符号，取值范围是A1A1~FEFE,共定义了682个符号，6 763个汉字。其中一级汉字3 755个，以拼音排序，二级汉字3 008个，以偏旁排序。这个编

码的缺点是收录的汉字太少，有许多次常用字没有收录，在处理人名、地名时非常不方便。

### 3.BIG5编码

俗称"大五码",是普遍使用的繁体汉字的编码标准，包括440个符号，一级汉字5 401个，二级汉字7 652个，共计13 060个汉字。

### 4.GBK编码 ( Chinese Internal Code Specification )

GBK编码是中国大陆制定的中文编码扩展国家标准。GBK工作小组于1995年12月完成GBK规范。该编码标准兼容GB2312,共收录汉字21 003个、符号883个，并提供1894个造字码位，其特点是简、繁体字融于一库。

### 5.GB18030-2000

这是在原来的GB2312-80 编码标准和 GBK 编码标准的基础上进行扩充，增加了四字节部分的编码。向上则兼容ISO10646,共有150多万个码位。它在原来的2万多汉字的基础上增加了7 000 多个汉字的码位和字型，从而汉字达到27 000多个。它能有效地解决一些生、偏、难字的问题，适用于需要的人名、地名用字的系统。支持GB13000.1-1993的全部中日韩（CJK）统一汉字字符和全部中日韩统一汉字Extension A和Extension B的字符。

### 6.Unicode编码 ( Universal Multiple Octet Coded Character Set )

这是国际标准组织对各国文字、符号进行的统一性编码。目前Unicode采用16位编码体系，其字符集内容与ISO10646的BMP（ Basic Multilingual Plane ）相同。版本V2.0中包含了符号6 811个，汉字20 902个，韩文拼音11 172个，造字区6 400个，保留20 249个，共计65 534个。

### 7.ISO10646 / Unicode字符集

全球可以共享的编码字符集。规定用4个字节表示世界各国语言文字的代码，其中汉字字符集可以扩大到6万字。但考虑到即使包括某些古籍汉字在内，现代一般使用的汉字，有2万字也已足够，这样ISO 10646的字符编码就可以减缩成两个字节，和Unicode相同，从而达到兼容性。

[版权方授权希赛网发布，侵权必究](#)

[上一节](#)      [本书简介](#)      [下一节](#)

## 校验码概述

### 7.1.6 校验码概述

#### 1.编码体系

体系这个词总是比较高深，但在这里有些大材小用之嫌，编码体系指一种编码方式中所有合法码字的集合。合法码字占有所有码字的比率就是编码效率。请读者计算一下BCD编码的编码效率。

#### 2.码距

码距是恒量一种编码方式的抗错误能力的一个指标。数字信息在传输和存取的过程中，由于各种意外情况的发生，数据可能会发生错误，即所谓误码。

一种编码，如果所有可能的码字都是合法码字，如ASCII,当码字中的一位发生错误时，这个错误的码仍然在编码体系中，这样我们称这种编码的码距小，如果我们把编码体系变得稀疏一点，使得很多的信号值不在编码体系之内，这样，合法的码字如果出现错误，可能就变成了不合法的编码，

这样的编码的码距就大。

定义：一个编码系统中任意两个合法的编码之间的不同的二进制位称为这两个码字的码距。该编码系统的任意两个编码之间的距离的最小值称为该编码系统的码距。

显然，码距越大，编码系统的抗偶然错误能力越强，甚至可以纠错（纠错详见各种编码的介绍）。同时，码距的增加，使得必须提供更多的空间来存放码字，数据冗余增加，编码效率则降低了，系统设计师需要综合考虑系统效率和系统健壮性两个方面，在众多的编码体系中选择适合特定目标系统的编码。

版权方授权希赛网发布，侵权必究

[上一节](#)      [本书简介](#)      [下一节](#)

奇偶校验

7.1.7 奇偶校验

奇偶校验较为简单，被广泛地采用，常见的串口通信中基本都使用奇偶校验作为数据校验的方法。

一个码距为1的编码系统加上一位奇偶校验码后，码距就成为2.产生奇偶校验时将信息数据的各位进行模二加法，直接使用这个加法的结果的称为奇校验。把这个加法值取反后作为校验码的称为偶校验。从直观的角度而言，奇校验的规则是：信息数据中各位中1的个数为奇数，校验码为1,否则校验码为0,偶校验则相反。

使用一位奇偶校验的方法能够检测出一位错误，但无法判断是哪一位出错。当两位同时出错时，奇偶校验也无法检测出来。所以奇偶校验通常用于对少量数据的校验，如一个字节。在串口通信中，通常是一个字节带上起始位、结束位和校验位共11位来传送。

如果对一位奇偶校验进行扩充，在若干个带有奇偶校验码的数据之后，再附上一个纵向的奇偶校验数据，如表7-2所示。

表7-2 奇偶校验位组成

信 息 位				校 验 位
$a_1$	$a_2$	...	$a_n$	$Hp_1$
$\beta_1$	$\beta_2$		$\beta_n$	$Hp_2$
...				
$x_1$	$x_2$		$x_n$	$Hp_m$
$\forall p_1$	$\forall p_2$		$\forall p_n$	$Hp_{(n+1)}$

这样，在出现一个错误的情况下，就能找到这个错误。而如果出现两个以上的错误，则可能无法判断误码的位置。这种方式在移动通信领域中被广泛采用。

版权方授权希赛网发布，侵权必究

[上一节](#)      [本书简介](#)      [下一节](#)

海明码和恒比码



### 7.1.8 海明码和恒比码

海明码是奇偶校验的另一种扩充。和上面提到的奇偶校验不同之处在于海明码采用多位校验码的方式，在这些校验位中的每一位都对不同的信息数据位进行奇偶校验，通过合理地安排每个校验位对原始数据进行校验位组合，可以达到发现错误，纠正错误的目的。

假设数据位有 $m$ 位，如何设定校验位 $k$ 的长度才能满足纠正一位错误的要求呢？我们这里做一个简单的推导。

$k$ 位的校验码可以有 $2^k$ 值。显然，其中一个值表示数据是正确的，而剩下的 $2^k-1$ 个值意味着数据中存在错误，如果能够满足： $2^k-1 > m + K$ （ $m + k$ 为编码后的总长度），在理论上 $k$ 个校验码就可以判断是哪一位（包括信息码和校验码）出现问题。

#### 1. 校验方程

校验方程是指示每个校验位对哪些信息位进行校验的等式。

确定了 $k$ 的值后，如何确定每 $k$ 位中的每一位对哪些数据进行校验呢？上面的推导只是说能够做的，那么如何达到纠错的目的呢？但是幸好考试中都会列出海明校验方程。例如：

$$b_1 \oplus b_3 \oplus b_5 \oplus b_7 = 0 \quad \text{①}$$

$$b_2 \oplus b_3 \oplus b_6 \oplus b_7 = 0 \quad \text{②}$$

$$b_4 \oplus b_5 \oplus b_6 \oplus b_7 = 0 \quad \text{③}$$

$\oplus$ 表示逻辑加。

在一般情况下，校验码会被插入到数据的1,2,4,8, ...位置，那么，在数据生成时，按照提供的海明校验方程计算出 $b_1, b_2, b_4, \dots$ 各位，在数据校验时，按照海明检验方程进行计算，如果所有的方程式计算都为0，则表示数据是正确的。如果出现1位错误，则至少有一个方程不为0。海明码的特殊之处在于，只要将①②③3个方程左边计算数据按③②①排列，得到的二进制数值就是该数据中出错的位，例如第6位出错，则③②①为110等于二进制的6。

当出现两位错误时，这种海明码能够查错，但无法纠错。

#### 2. 恒比码

采用恒比码编码体系中，所有有效的编码中为1的位都相同，所以被称为恒比。在邮电部门的电传、电报及条形码中就广泛地使用恒比码。这种编码生成时是查表，接收检验时是检查每个编码中1出现的次数是否正确。

版权方授权希赛网发布，侵权必究

[上一节](#)

[本书简介](#)

[下一节](#)

## 循环冗余校验码

### 7.1.9 循环冗余校验码

这种方式已经被广泛地在网络通信及磁盘存储时采用，所以在历年考试中出现概率也比较大。先看几个基本概念。

#### 1. 多项式



在循环冗余校验码中，无一例外地要提到多项式的概念。一个二进制数可以以一个多项式来表示。如1011表示为多项式 $x^3+x^2+x+1$ ，在这里， $x$ 并不表示未知数这个概念，如果把这里的 $x$ 替换为2，这个多项式的值就是该数的值。从这个转换我们可以看出多项式最高幂次为 $n$ ，则转换为二进制数有 $n+1$ 位。

## 2. 编码的组成

循环冗余校验码校验由 $k$ 位信息码，加上 $R$ 位的校验码。

## 3. 生成多项式

和海明码的校验方程一样，生成多项式非常重要，以至于考试中总是直接给出。

由 $K$ 位信息码如何生成 $R$ 位的校验码的关键在于生成多项式。这个多项式是编码方程和解码方程共同约定的，编码方程将信息码的多项式除以生成多项式，将得到余数多项式作为校验码，解码方程将收到的信息除以生成多项式，如果余数为0，则认为没有错误。如果不为0，余数则作为确定错误位置的依据。

生成多项式并非任意指定，它必须具备以下条件：最高位和最低位为1，2.数据发生错误时，余数不为0，对余数补0后，继续做按位除，余数循环出现，这也是冗余循环校验中循环一词的来源。

## 4. 校验码的生成

将 $k$ 位数据 $C(x)$ 左移 $R$ 位，给校验位留下空间，得到移位后的多项式： $C(x) \times x^R$ 。

将移位后的信息多项式除以生成多项式，得到 $R$ 位的余数多项式。

将余数嵌入信息位左移后的空间。

例：信息位为10100110 生成多项式： $a(x) = x^5 + x^4 + x + 1$

则： $C(x) = x^7 + x^5 + x^2 + x$

$C(x) \times x^R = x^5(x^7 + x^5 + x^2 + x)$

$= x^{12} + x^{10} + x^7 + x^6$

求余式：

$$\begin{array}{r}
 x^5 + x^4 + x + 1 \overline{) x^{12} + x^{10} + x^7 + x^6} \\
 \underline{x^{12} + x^{11} + x^8 + x^7} \phantom{+ x^6} \\
 x^{11} + x^{10} + x^8 + x^6 \\
 \underline{x^{11} + x^{10} + x^7 + x^6} \\
 x^8 + x^7 \\
 \underline{x^8 + x^7 + x^4 + x^3} \\
 x^4 + x^3
 \end{array}$$

得到余式为 $x^4 + x^3$ 即校验码为11000，所以得到CRC码是：1010011011000。

循环冗余校验码的纠错能力取决于 $k$ 值和 $R$ 值。在实践中， $k$ 取值往往取得非常大，远远大于 $R$ 的值，提高了编码效率。在这种情况下，循环冗余校验就只能检错不能纠错。一般来说， $R$ 位生成多项式可检测出所有双错、奇数位错和突发位错小于等于 $R$ 的突发错误。使用循环冗余校验码能用很少的校验码检测出大多数的错误，检错能力是非常强的，这使得它得到了广泛的应用。

## 多处理机系统

### 7.8 多处理机系统

并行处理机只有一个控制器，由这个控制器控制所有的运算器，而多处理机则有多个控制器，分别控制对应的运算器。

多处理机属于MIMD,即有多个处理单元，同时对不同的数据进行不同的处理，指令流和数据流都存在并行，这决定了多处理机就是为解决复杂问题而设计的。这些复杂问题包括我们熟悉的天气预报，由于影响天气变化的因素太多，所以想做预报就必须给计算机系统加入大量的运算过程，而尴尬的情况是，当计算的时间比要预报的时间还要长时，就变成了验证而不是预报了。我们已经发现天气预报已经比原来准多了，国家已经能够开始提前发布"黄金周"的天气预报了，这也是计算机发展的一个佐证吧。

从多个处理器之间的关系而言，多处理机又分为对称的多处理机和非对称的多处理机。如果所有的处理机都能同样访问所在的外围设备，就是对称的多处理机，相反，只有部分处理机（称主处理机）能执行操作系统，而附属处理机在主处理机的监督和控制，就是非对称的多处理机。并行处理机不需要这样的分类，它的控制单元只有一个，而属于被控制的处理单元有很多，所有的处理单元的地位是一样的。

版权方授权希赛网发布，侵权必究

## 访问存储器方式

### 7.8.1 访问存储器方式

多处理机有多个处理单元，就产生了这些处理单元如何访问内存的问题，有两种方式：共享存储方式和分布式存储方式，来访问内存。

共享存储方式的多处理机有公共的共享存储器。多个处理机能够访问共享的存储器，并使用共享的存储器传递共享公共信息和参数等。特点是容易管理和利用资源，常常适用于多用户的一般应用和分时应用，也能加快大程序的运行速度，但处理机数目有限，不易扩充。

各个处理机对共享存储器的访问方式有：均匀存储器存取、非均匀存储器存取和只有高速缓存的多处理机。

#### 1.均匀存储器存取（Uniform-Memory Access UMA）

采用这种方式访问共享存储器的多处理机，所有处理机对所有存储单元有相同的存取时间，所以称为"均匀",如图7-21所示。

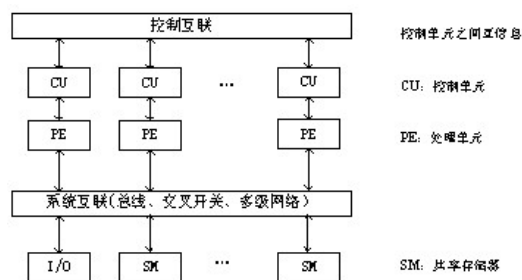


图7-21 均匀存储器存取

## 2.非均匀存储器存取的多处理机

和均匀存储器存取不同，这种多处理机中不同处理机访问不同存储单元的存取时间是不同的。

这种方式也有多种实现的方式，典型的有共享本地存储式，如图7-22所示。

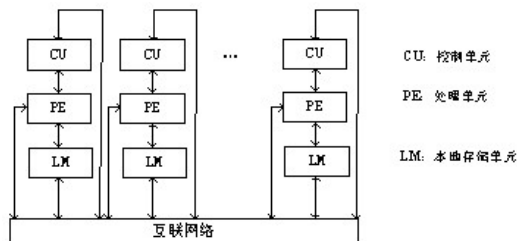


图7-22 非均匀存储器存取

显然多个处理机访问本地存储很快，而访问其他处理机的存储器则需要经过互连网络。还有层次机群，层次机群扩展了上面的结构，增加了全局共享存储器，如图7-23所示。

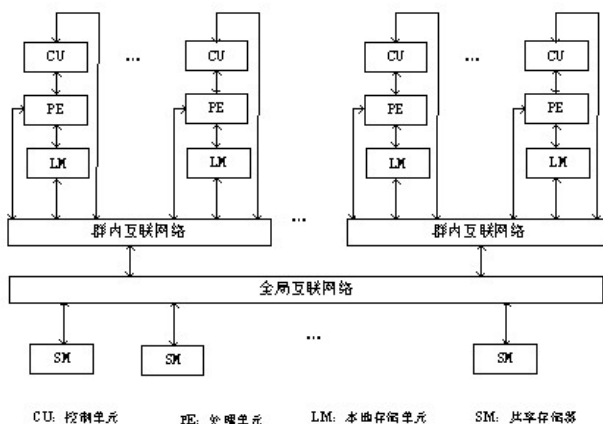


图7-23 层次机群

非均匀存储多处理机的特点是：容易扩充，但复杂的数据结构难以在处理机之间传递。

## 3.只有高速缓存的多处理机

这是非均匀存储访问处理机一个特例，处理机内没有本地或共享的存储器，只有高速缓存，数据从哪里来是由目录模块决定的，如图7-24所示。

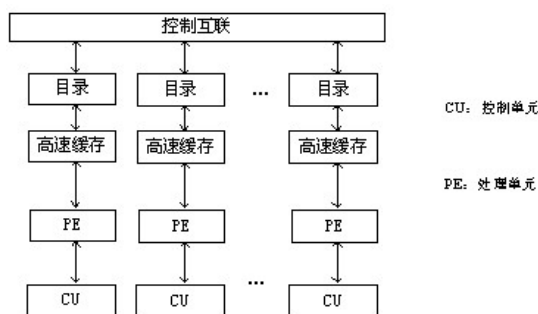


图7-24 高速缓存的多处理机

## 4.分布式多处理机

分布式多处理机如图7-25所示。

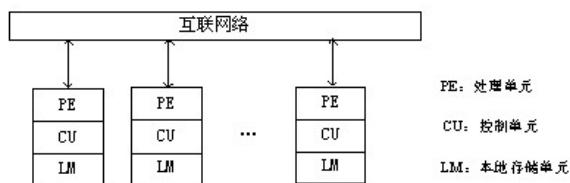


图7-25 分布式多处理器

和共享存储器访问方式不同，这种方式是每个处理机独占本地存储器，各处理机通过信息网络相联，这更像计算机网络的结构。

版权方授权希赛网发布，侵权必究

上一节 本书简介 下一节

第 7 章：计算机的体系结构和主要部件

作者：希赛教育软考学院 来源：希赛网 2014年01月27日

## 互联方式

### 7.8.2 互联方式

上节讲述的“互连网络”只是一个方框，其实多处理机和共享存储器、输入/输出设备等之间的通信性能是多处理机性能的重要一环，互联方式有多种，介绍如下。

#### 1.总线

总线方式是最简单的方法。总线按照层次可分为：处理机板上的本地总线，为所有插入板提供信息通路的底板总线和同输入/输出外设交换信息的输入/输出总线，总线方式适用于多处理机个数少于100的系统，主要是受到了组装技术的限制，而数据的传输率受到总线带宽和速度的限制。

#### 2.开关网络和交叉开关

开关网络为处理机和共享存储器之间提供了一个动态互联通路，如图7-26所示。

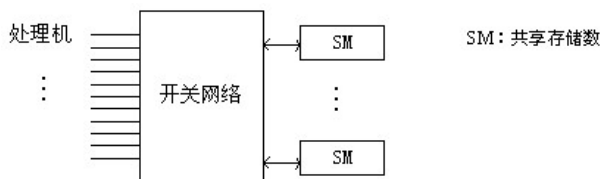


图7-26 开关网络

如果中间的开关网同时连接到达一定的数量后，不能再容纳新的连接，这种网络是阻塞网络。

交叉开关提供了非阻塞网络的连接，它使用一组开关阵列将处理机和共享存储器连接，如图7-27所示。

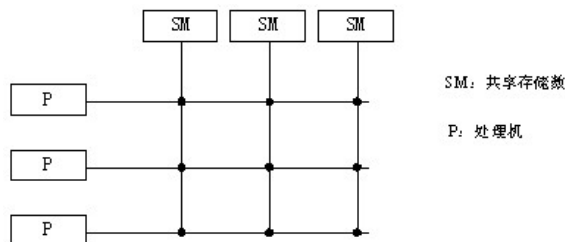


图7-27 交叉开关

每个交叉的结点都是开关，可以连接或断开，这样以空间矩阵来换时间的方式需要使用大量的连线和交叉点开关，所以只有处理机和共享存储器数目较少时才可能使用这种方式，而且一旦建成，很难扩充。

和交叉开关不同的是，开关枢纽不使用交叉矩阵这样庞大的硬件结构。它由仲裁单元和开关单元组成，仲裁单元完成冲突处理，开关单元完成连接，其方式和通信中的交换机有些相似。

### 3.多端口存储器

这事实上是将交叉点仲裁逻辑移动到存储器去控制的方法，如图7-28所示。

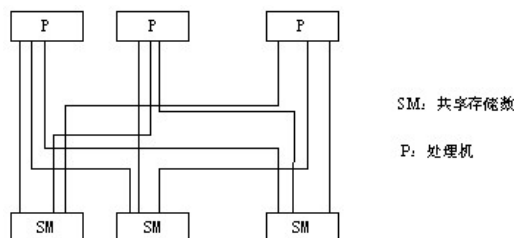


图7-28 多端口存储器

每个存储器模块有多个存取端口，由存储器负责分解多个处理机的冲突请求。这增加了存储器模式的成本，和交叉开关类似，需要大量的连线。而且存储器端口数目是固定的，一旦开始生成之后就无法增加。

版权方授权希赛网发布，侵权必究

[上一节](#)

[本书简介](#)

[下一节](#)

## 多处理机的特点

### 7.8.3 多处理机的特点

多处理机的特点，就是复杂，这个复杂体现在以下方面。

#### 1.结构的复杂性

书中提到的多处理机结构只是有代表性的几种，为了适应复杂的和各种各样需要解决的问题，多处理机之间的互连网络、数据传递的种类相当多。由于这个复杂性，所以多处理机的数量还无法大量增加。

#### 2.程序并行性识别

和流水线及并行处理机不同，多处理机的并行性不再是细粒度的并行，而是中、粗粒度的并行，是在过程和子程序级别以上的并行，如何识别程序的并行性也是非常困难。

#### 3.需要专门的并行指令

在程序中需要专门的一种控制并发的指令，这种指令用来表示可以并发执行的多个程序段之间的并发关系，并且可以控制他们的并发执行。这样，在一个任务执行的时候，可以导致一些其他任务和这个任务并发执行，这称为并行任务的派生。这种派生同样复杂，和程序的性能及程序的流程不同而有不同的变化。

#### 4.同步措施

在并发的各个处理机执行的不同程序或程序段中，经常会存在数据依赖或者是控制依赖，这种相互关系需要专门的同步措施，来调节不同的处理机执行过程和执行内容。当然最好是能尽量减少不同处理机运行的程序依赖性，还记得软件中的"高内聚，低耦合"吗？

#### 5.资源分配和任务调度

版权方授权希赛网发布，侵权必究

第 7 章：计算机的体系结构和主要部件

作者：希赛教育软考学院 来源：希赛网 2014年01月26日

## 7.2 中央处理器、存储器

版权方授权希赛网发布，侵权必究

第 7 章：计算机的体系结构和主要部件

作者：希赛教育软考学院 来源：希赛网 2014年03月13日

### 7.2.1 中央处理器

中央处理器的类型和品种异常丰富，各种中央处理器的性能也差别很大，有不同的内部结构、不同的指令系统。但由于基于冯·诺依曼结构，基本部分组成相似。

运算器的主要功能是在控制器的控制下完成各称算术运算逻辑运算和其他操作。一个计算过程需要用到由加法器/累加器、数据寄存器或其他寄存器、状态寄存器等。

其中：

$X_i$ 、 $Y_i$ 是加数和被加数；

$C_{i+1}$ 是低位进位；

$C_i$ 是进位；

$Z_i$ 是和。

为完成多位数据加法，可以通过增加电路和部件，简单的加法器能够变为串行、并行加法器，超前进位加法器等。

运算器的位数，即运算器一次能对多少位的数据做加法是衡量中央处理器的一个重要指标。

## 2.控制器

控制器是中央处理器的核心，它控制和协调整个计算机的动作，控制通常需要程序计数器（PC），指令寄存器（IR），指令译码器（ID），定时与控制电路，以及脉冲源、中断等共同完成，如图7-2所示。

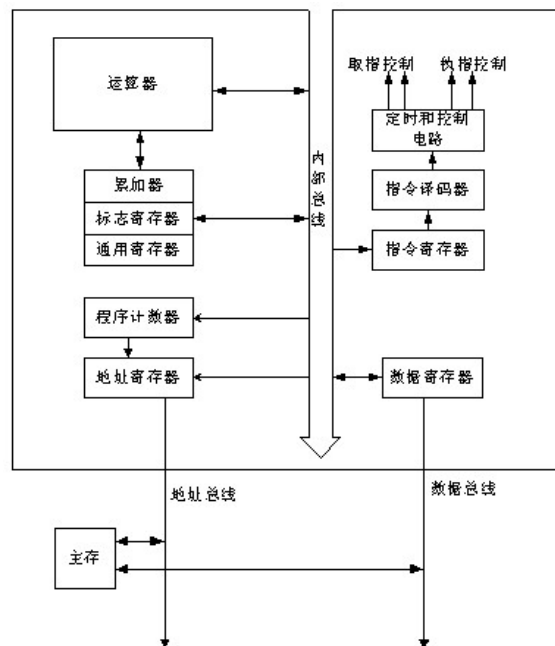


图7-2 控制器组成

中央处理器中各组件介绍如下。

指令寄存器（Instruction Register）：显然，中央处理器即将执行的操作码表在这里。

指令译码器（Instruction Decoder）：将操作码解码，告诉中央处理器该做什么。

定时与控制电路（Programmable Logic Array）：用来产生各种微操作控制信号。

程序计数器（Program Counter）：程序计数器中存放的是下一条指令的地址。由于多数情况下程序是顺序执行的，所以程序计数器设计成能自动加1的装置。当出现转移指令时，就需重填程序计数器。

程序计数器可能是下一条指令的绝对地址，也可能是相对地址，即地址偏移量。

标志寄存器（Flags Register）：这个寄存器通常记录运算器的重要状态或特征，典型的是是否溢出，结果为0、被0除等。这个寄存器的每一位表示一个特征。标志寄存器的典型应用是作为跳转指令的判断条件。

堆栈和堆栈指针（Stack Pointer）：堆栈可以是一组寄存器或在存储器内的特定区域。由于寄存器数量总是有限，所以大多数系统采用了使用存储器的软件堆栈。堆栈顶部的指针称为堆栈指针。



### 3.寄存器组

上面提及的程序计数器、标志寄存器等为专用寄存器，它们有特定的功能和用途。

通用寄存器的功能由程序指令决定，最常见的应用是放置计算的中间结果，减少对存储器的访问次数。通常寄存器的宽度是和运算器的位数相一致的。

版权方授权希赛网发布，侵权必究

[上一节](#)

[本书简介](#)

[下一节](#)

第 7 章：计算机的体系结构和主要部件

作者：希赛教育软考学院 来源：希赛网 2014年01月26日

## 内存

### 7.2.2 内存

这里的存储器是中央处理器通过总线与直接能访问的存储器，通常称为内存。硬盘等需通过I/O接口访问的存储器常称为外存或者辅存。

存储器作用显然是存储数据包括指令、指令所带的数据（注意到了吗？这正是冯·诺依曼结构的特点之一）和中央处理器处理后的结果，包括中间结果。

中央处理器对存储器的访问须通过控制地址、数据总线方式进行。存储器的数组组织是线性的，所存储的数据都有整齐地"编号",即访问地址。存储器一般每个存储单元中有8位数据，其容量是其存储单元的总和。

#### 1.存储器的性能指标

存取时间：指的是从中央处理器发出指令到操作完成的时间。

传输率：或称为数据传输带宽，指单位时间内写入或读取的数据的多少，显然，存取时间越小，则传输率越高。

存储密度：单位面积的存储容量。

#### 2.存储器分类

传统意义上存储器分为RAM（Random Address Memory）和ROM（Read Only Memory），RAM是随机存储器，数据可读可写，一旦掉电，数据将消失。和ROM的命名方式不同，随机存储器强调的并非是其的可读可写特性，而是它的"随机"特性，也就是通过地址和数据总线，中央处理器可以"随机"读写存储器的任何某个字节的内容。ROM是只读存储器，掉电后数据依然保存。在个人计算机中，典型的ROM是BIOS,它里面是个人计算机的硬件检测和引导程序。事实上，有许多种不同类型的存储器，新的产品不断出现，正在打破通常意义上的RAM和ROM的分界线。既可读可写，又能掉电后仍保持数据的存储器已经出现，有的资料称为"混合"类型。

RAM有静态和动态两种。最常用的动态RAM需要定时刷新电路才能保持数据，而静态RAM上电后信息不丢失，无须刷新过程，所以速度上有优势，而且不需要额外的刷新电路，常作为芯片中的Cache使用。而动态RAM集成度高成本低，得到了广泛的应用。

PROM（可编程ROM）：这种ROM可写一次，写完后就不能再更改了。

EPROM（可擦除可编程ROM）：这种ROM可多次擦除和重写，擦除工作由紫外线照射来完成。

EEPROM（可擦除可编程ROM）：使用电信号即可擦除的ROM,可多次重写。

普通的ROM最便宜，适合于已经成型和经过严格测试的大量生产的情况。而在开发的阶段，使用EEPROM则可以减少程序的烧录时间，提高开发效率，节约开发成本。

Fash（闪存）：在嵌入式系统中，还会经常利用Flash,它的特征和EEPROM相同，又有EPROM的高集成性，低廉的成本，广泛应用于掌上电脑和手机等嵌入式系统。它的特点是擦除和写信息必须是按照数据块来操作的。这和普通的随机存储内存不同。

NVRAM为非挥发性RAM,这实际上是静态RAM加上电池，加电时，它的表现和静态RAM一样，失去电力时，由电池供电以保持其中的内容不丢失，由于成本高昂，所以其容量通常非常小，通常只有几百字节。如表7-3所示。

表7-3 几种存储器之间的比较

类 型	保 持 数 据	可 写	擦 除 大 小	擦 除 次 数	价 格	读 写 速 度
静态 RAM	√	√	字节	无限	昂贵	快
动态 RAM	√	√	字节	无限	中等	中等
ROM	×	×			便宜	快
PROM	×	一次使用编程器			中等	快
EPROM	×	√ 使用编程器	整个芯片	有限	中等	快
EEPROM	×	√	字节	有限	昂贵	读快 写慢
Flash	×	√	区	有限	中等	读快 写慢
NVRAM	×	√	字节	无限	昂贵	快

版权方授权希赛网发布，侵权必究

[上一节](#)      [本书简介](#)      [下一节](#)

外存

7.2.3 外存

和内存相比，外存或辅存有以下特点：中央处理器必须通过I/O接口和外存联系，容量大，速度较慢。

硬盘是最常见的外存，硬盘的容量越来越大，平均寻道时间越来越短，是大多数计算机不可缺少的组成部分。

软盘已经在很长的时间内没有发展了，现在仍然在使用3.5英寸规格，容量为1.44MB,它曾经辉煌的地位已经让位给现在的闪存盘。

光盘、CD-ROM已经是软件发行最常见的载体，DVD的驱动器也很常见，光盘刻录也是备份的优先选择之一。

磁带的特点是顺序读写，容量很大，这使得磁带是重要数据备份的主要选择。

闪存盘以容量大（相对软盘而言）体积小而成为流行的便携式存储产品，闪存盘一般以USB接口和计算机相连，也常称为U盘。

版权方授权希赛网发布，侵权必究

[上一节](#)      [本书简介](#)      [下一节](#)

## 时序产生器和控制方式

### 7.2.4 时序产生器和控制方式

为了使得计算机各部件同步工作，计算机中都有一个脉冲源，通常是晶振。这个脉冲源产生主振脉冲，主振脉冲的时间间隔为主振周期，即时钟周期。中央处理器执行指令的时间（包括取址）为指令周期，由于指令可能有不同的复杂度，所以，每种指令的指令周期可能不同。CPU周期也称机器周期，一般是从内存中读一个指令的最短时间。CPU周期又由若干个时钟周期组成。如图7-3所示。

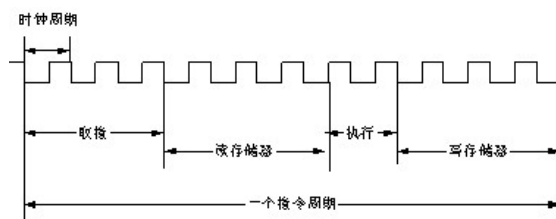


图7-3 一个指令周期

通常把CPU执行指令的各个微操作遵循的时间顺序叫时序。时序图是形象地表示信号线上信息变化的时间序列的图形。

组合逻辑控制和微过程控制是两种基本的控制方式。

#### 1. 组合逻辑控制

组合逻辑控制器是使用专用门的逻辑电路。它的实现有硬件接线控制和可编程逻辑阵列两种，硬件连线法最直接，可以用较少的组件实现最快的速度，但是如果要更改，只有重新设计。可编程逻辑阵列采用低成本大规模集成电路的方式。组合逻辑控制灵活性很差，在复杂指令系统计算机中难以处理不断增加的复杂指令，但是它使用电子组件少，在采用精简指令集的计算机中发挥了很大作用。

#### 2. 微过程控制

为提高控制的灵活性，许多中央处理器采用的微过程控制的控制方法，先看几个概念。

微程序：微程序对应一条机器指令，若干个微指令序列形成一段微程序。而微指令又可细分为若干微操作，控制内存是存放微程序的地方。

微操作是最基本的操作，可分为兼容性微操作和不兼容性微操作，这两种微操作的区别在于该微操作是否能在一个CPU周期内并行执行。

微指令格式：

操作控制字段	顺序控制字段
--------	--------

前半部分，存放对各种控制门进行激活或关闭的控制信息，后半部分是后续微指令的地址，这实现了数据结构中单相关列表。

操作控制字段的格式有两种。

水平型微指令：操作控制字段的每一位控制不同的控制门，可以在一个微指令中定义，执行多个并行的微操作优点是效率高、灵活，执行时间短。

垂直型微指令：和水平型相比，其格式要短，一条微指令中包括的微操作少，只有1~2个，由于其指令字短，所以比较容易掌握。

在实践中也常常使用混合型微指令，即水平型微指令和垂直型微指令的混合。

## 输入/输出及其控制

---

### 7.3 输入/输出及其控制

曾经在科幻作品中有大胆的设想：玻璃瓶中的大脑，如果科技发达到能给玻璃瓶中的大脑施加视觉、听觉、触觉等信号，这个大脑能够意识到自己是在玻璃瓶中，而不是在现实中吗？

借助于输入、输出、设备，我们才能"操纵计算机"和"知道它在工作"。

## 输入/输出设备一览

---

### 7.3.1 输入/输出设备一览

计算机的输入/输出设备品种繁多，主要有以下几种。

#### 1.纸带机、卡片机

这都是"古代"大型计算机的输入设备，机器指令被以打孔等方式存在于纸带、卡片上，由纸带机和卡片机输入计算机，以后人们可能只能从博物馆见到这些设备。

#### 2.键盘、鼠标

嵌入式系统中键盘往往简化成少数的几个键，和鼠标在本质上并没有变化，键盘负责输入字符，鼠标负责指示位置进行选择或点取等操作。手写笔是鼠标的扩展，需要识别软件的支持。

#### 3.显示器

嵌入式系统中的显示器往往简化为小屏幕CD或数码管，显示器向着越来越大、越来越清晰的方面进步，现在CRT显示器仍占主流，但统治地位已经受到了液晶显示器的挑战。触摸屏实际上是显示器和鼠标的结合。

#### 4.外存

外存是主存的辅助和延伸，有软盘、硬盘、光存储器、磁带机、闪存等。

#### 5.打印机

将人们需要的结果在纸面上输出。打印机有针打、喷墨、激光打印机。绘图仪是一种特殊的打印机，专门用于大幅的图形精确输出。

#### 6.图形图像摄影输入设备

这些是新兴的输入设备，包括扫描仪、数码相机、数字摄像机等。这些设备能够把图像摄影等信息输入计算机，极大地丰富了个人计算机在普通家庭的作用。图形图像已经形成了TWAIN标准接口，这样软件通过这个标准接口，能同各种不同的图像输入设备进行交互。数字摄像机是通过

输入/输出控制器

7.3.2 输入/输出控制器

由于外设的复杂多样性（控制方式、数据传输速率、数据格式也各不相同），同时不断有新的类型的外设出现，这使得很难使用中央处理器来和它们直接打交道。同时这些外设的数据传送速度往往远远低于系统总线的速度，不适合把它们直接挂在系统总线上。

输入/输出控制器来协调和控制数据的输入/输出，具体功能有：缓冲锁存数据、地址译码、传递命令、码制转换、电平转换等如图7-4所示。

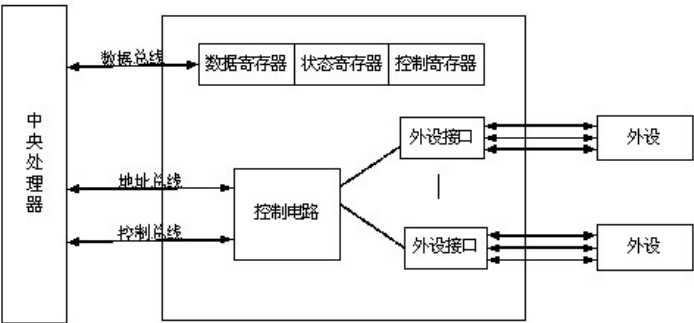


图7-4 输入/输出控制器组成

- 数据寄存器：CPU或外设数据的缓冲。
- 状态寄存器：向中央处理器指示控制器和外设的状态。
- 控制寄存器：由中央处理器写入该输入/输出控制器的识别指令，指示输入/输出控制器去和外设交互，将数据寄存器的内容写入外设，或者将外设的数据写入数据寄存器，并将此过程的状态写入状态寄存器。
- 控制电路是输入/输出控制的核心，负责输入/输出控制器的行为。

外设的识别

7.3.3 外设的识别

计算机程序要访问外设，从输入/输出控制器组成图中可以看出，中央处理器需要访问输入/输出控制器的各种寄存器，这个识别是通过地址总线和接口电路的地址译码器来实现的，可以有不同的外设编址方式。

1.独立编址方式

在这种方式下，输入/输出地址和主存地址是分开的，从电路上说，有分离的输入/输出读写控制线，从指令上说，有专门的输入/输出控制指令。这些指令常常以汇编指令的方式提供，往往是和具体的中央处理器密切相关的，这对程序的可移植性造成了影响，如表7-4所示。

表7-4 个人计算机部分输入/输出地址表

输入/输出设备	输入/输出地址
硬盘控制器	0x320~0x32F
彩色显示器控制	0x3D0~0x3DF
串口控制器	0x3F8~0x3FF

2.统一编址方式

和独立编址不同，统一编址方式中输入/输出地址是主存地址中的一部分，访问输入/输出寄存器和访问主存的方法是一样的，不需要单独的指令。这种方式能有效地降低软件编程的复杂度，在许多计算机，特别是嵌入式系统中得到了广泛的认可和应用，也称为地址映像。

在统一编址方式编程过程中，需要注意的是程序优化带来的负面影响，对外设的操作往往需要一个过程，可能会对同一个地址多次写不同的值，这在编译器看来，这种多次对一个变量的赋值是属于优化范围的，编译器通常会忽略掉前面的赋值语句，只保留它认为"有效的"赋值语句。这时候需要选择合适的编译选项告诉编译器停止这样的优化。

统一编址方式不但会将内存地址和I/O地址统一编址，也会将RAM、ROM、Flash等统一编址。

版权方授权希赛网发布，侵权必究

[上一节](#)      [本书简介](#)      [下一节](#)

外设的访问

7.3.4 外设的访问

1.程序查询方式

这是最简单的方式，也是简单系统中（外设种类和数目有限，数据传输速度较低的系统）常用的方式。这种方式是中央处理器定时查询外设的状态，如果发现某个外设就绪，就开始和该外设进行输入/输出操作和处理，如图7-5所示。

当存在多个外设时，中央处理器有串行和并行两种查询方式，串行查询是每次查询一个外设，并行则是将多个外设的状态位集中成一个专用端口，这样中央处理器一次查询即可得到多个外设的状态。

程序查询方式的缺点是：当输入/输出控制器和外设交换数据时的过程中，中央处理器必须等待。这种等待对于许多系统而言是无法容忍的。

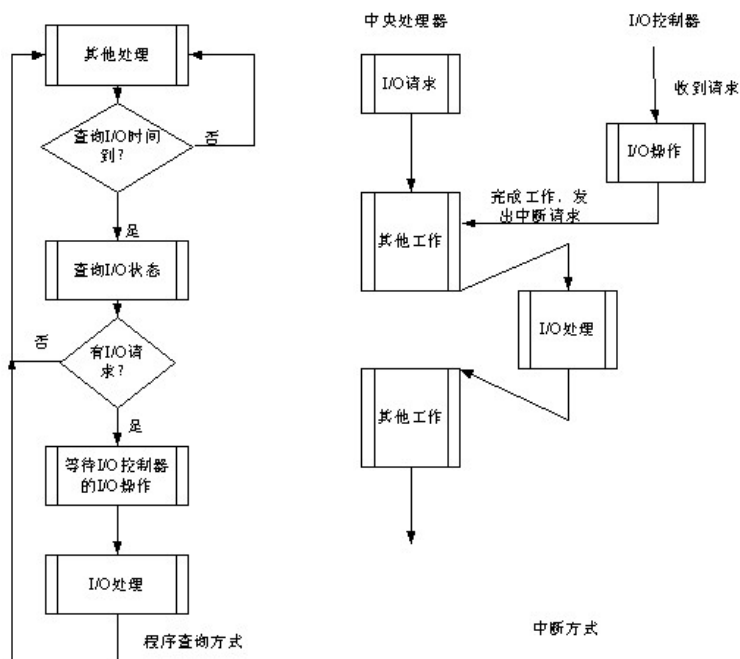


图7-5 程序查询和中断方式

## 2.中断方式

使用中断方式，可以克服查询方式的低效问题。

当中央处理器执行到I/O请求指令时，向输入/输出控制器发出相应指令后，中央处理器并不等待，而是继续执行其他操作。此时，输入/输出控制器负责和外设进行通信，当数据从其数据寄存器写到外设后或者外设的数据写入其数据寄存器后，输入/输出控制器向中央处理器发出中断请求，中央处理器响应中断，并进行相应的处理。注意到，由于输入/输出控制器的数据寄存器大小有限，一次输入/输出请求往往要经过多次的中断过程才能够完成。由于中央处理器无须等待输入/输出控制器和外设的数据交换，提高了整个系统的效率。中断方式已经得到了普遍的应用。

### 1) 中断的基本概念

中断并不只用于输入/输出系统中，中断系统是计算机的基本结构，中断系统的出现，是现代计算机功能强大的标志。顾名思义，中断就是打断中央处理器正在执行的工作，让中央处理器去处理其他更加重要或者更为紧迫的任务。发起中断的事务称为中断源，中断源包括I/O设备、实时时钟、故障源、软件中断等。中断系统使得中央处理器摆脱了只能按照指令顺序执行的束缚，让计算在并行性、分时操作、故障处理等方面更加强大。

按照中断源来区分中断，可以分为内部中断和外部中断。内部中断是中央处理器内部产生的中断，在个人计算机中，内部中断又分为溢出中断、除法错中断、断点中断、软中断及单步中断，其中可以使用软件中断实现DOS功能调用和基本BIOS调用，也可以使用单步中断实现程序的调试，与之相对应的是外部中断，中断源来自于中央处理器之外。而外部中断按照中央处理器的响应可以分为可屏蔽中断和非屏蔽中断。非屏蔽中断是中央处理器一定要响应的中断，通常是计算机发生了紧急情况，如掉电等。可屏蔽中断大多数是外设和时钟中断，在计算机处理一些不应该打断的任务时，可以通过屏蔽位来禁止响应这些中断。

### 2) 中断处理过程

中央处理器收到中断请求后，如果是当前允许的中断，那么要停止正在执行的代码，并把内部寄存器入栈，这个过程不能被再次打断，所以在保护现场的开始要先关中断，保护完后再开中断。这个过程应该尽量短，以避免错过了其他中断。这个过程消耗的时间称为中断响应时间。然后开始执行中断处理程序，中断处理程序常常比较简单，通常是设置一些标志位，做一些简单的数据处



理，而让其他更耗时的处理在非中断程序中完成。中断处理程序完成后，需要将刚才保存的现场恢复，把入栈的寄存器出栈，继续执行被中断的程序。整个过程消耗的时间称为中断处理时间，当然对于这个时间，不同的中断，不同的应用差别比较大，而且也不是一味求短，实际编写时要考虑中断处理的重要程度。现在大多数中央处理器都支持多级中断，即在进行中断处理程序时，还可以响应其他中断，形成中断嵌套。

3) 中断的判断

当有多个中断源时，常用的处理方式有以下几种。

每个中断源使用自己的中断请求信号线和中央处理器相连，这种方式适用于中断源不是很多的情况，而中央处理器的外部中断引脚是有限的。

统一的中断请求，由中央处理器使用专门程序依次判断是哪个中断源的请求，通过查询的次序，可以实现中断的优先级控制。

硬件查询法：有一条中断确认信号链和输入/输出设备相连，某个外设发出中断请求后，中断确认信号开始在各外设间传递，发出中断请求的外设响应这个信号，如图7-6所示。

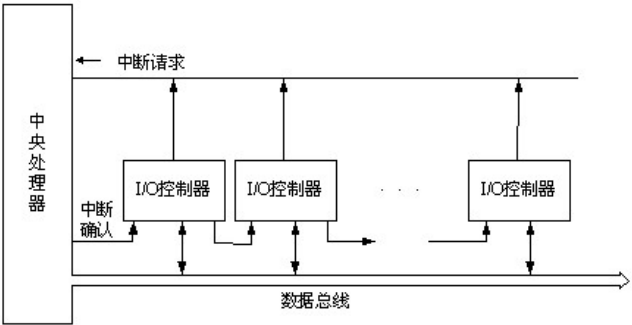


图7-6 中断方式的判断

总线仲裁：在这种方式中，外设须先得到总线控制权，发出中断请求，最后将设备号通过数据总线发给中央处理器。由总线仲裁机制决定可以发信号的外设。

中断向量表：中断向量表是一张不同中断处理程序入口地址的表格，用这种机制，每个中断源有不同的"中断号",即中断向量中央处理器收到中断信号，并根据中断号查中断向量表，以得到该中断的处理程序的入口地址。

3.DMA方式

DMA ( Direct Memory Access ) 直接存储器存取。这种方式可以使得数据从输入/输出模块到主存的传输过程中，无须中央处理器的中转，这个工作转移给了DMA控制器（DMAC）来完成，这种方式可以达到高速的数据传输。

1) DMAC控制器

DMAC也能访问系统总线，能够独立访问主存（这两个特点使得DMAC完成主存和输入/输出设备之间的数据交换）如图7-7所示。

DMA中断控制示意图如图7-8所示。

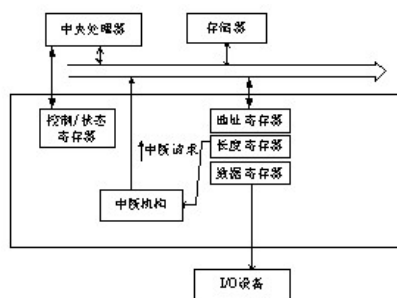


图7-7 DMAC示意图

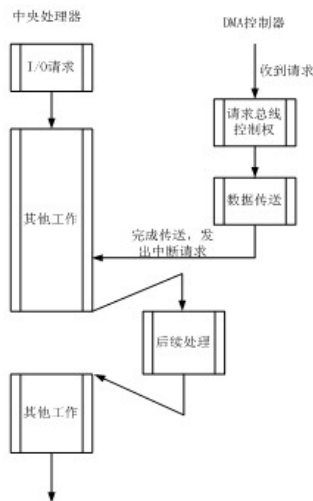


图7-8 DMA中断控制

地址寄存器是内存地址，每传递一个数据，将这个寄存器加1,长度寄存器减1,当长度寄存器为0时，发给中断机构完成信号，通知中央处理器进行后续处理。

当中央处理器执行到输入/输出请求时，向DMA控制器发出相应指令，DMA控制器首先判断外设是否可用，如果可用，填充地址寄存器、长度寄存器等，向中央处理器发出总线请求信号，申请总线的处理权。中央处理器收到总线请求信号，让出总线控制权，然后DMA控制器将数据在外设和内存指定区域之间进行传送，而长度寄存器保存的值随着数据的传送不断减少，当减少到0时，通过中断机构向中央处理器发出中断请求，中央处理器响应中断，对内存中的数据进行后续的处理。

## 2) DMA传送过程的总线占有方式

在DMA传输过程中，中央处理器停止访问主存，只进行了一些与总线无关的内部操作。这种方法常用于高速的输入/输出设备。

优点是减少系统总线控制权的交换次数，实现简单；缺点在于这样的结果往往使中央处理器在DMA过程中无所事事。

时间轮转片法：这种方法按照一定时间间隔，将总控制权分别轮换着交给中央处理器和DMA.这样中央处理器不会停止工作，但往往外设的速度低，可能使得DMAC的某些时间空转。从效率而言，仍然不高。

借用周期法：这是时间轮转片的改进，即当有DMA操作时，DMAC控制总线访问内存，其他时间总线的控制权在中央处理器，还适合于外设速度远低于总线速度的高速主机。这种方式由于要判断DMAC是否需要使用总线，所以实现起来要比前面两者要复杂。

## 3) DMA方式和中断方式的区别

DMA方式中使用到了中断，但是DMA和中断的输入/输出方式是有很大的区别的。最根本的区别在于，使用中断方式时，主存和输入/输出控制器之间的数据传送仍然需要用中央处理器来操作，需要使用中央处理器的寄存器等资源，如图7-9所示。

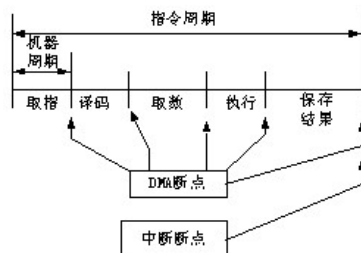


图7-9 DMA与中断方式的区别

而且，由于输入/输出控制器的数据寄存器大小有限，所以一个数据传送过程往往需要多次，这

样中断发生也就很频繁。由于中断调用过程使用了中央处理器的资源，所以中央处理器必须保护现场，使得在相当程度上增加了处理时间。而在DMA传送过程中，虽然DMA控制器需要暂停中央处理器的执行，来达到控制总线的目的，但是这种暂停是机器周期的中断，而且这个暂停中央处理器不需要保护现场，没有切换任务的操作。当数据传送完成后，才有一个中断，通知中央处理器进行数据传送的后续工作。DMA方式提供了比中断方式更好的并行性，如表7-5所示。

表7-5 DMA方式和中断方式的比较

	中 断 方 式	DMA 方式
I/O 和主存数据传送	需要 CPU 处理	不需要 CPU 处理
保护执行现场	需要	不需要
相应时间	一条指令结束	CPU 周期结束
并行性	有	更好的并行性
处理异常能力	强	比中断差

4.信道方式

信道方式比DMA方式更进一步的是信道方式，和DMA相比，通常控制器是一个有自身指令结构的处理器，有自己的简单的指令系统，通过执行程序，能够有更强的处理能力，可以同时控制多种外设。

版权方授权希赛网发布，侵权必究

[上一节](#)      [本书简介](#)      [下一节](#)

常见输入/输出接口

7.3.5 常见输入/输出接口

1.磁盘接口

IDE ( Integrated drive eectronics ) 接口，这个接口是个人计算机必备的，由于普通IDE容量不超过528MB,现在用的是增强型EIDE,数据传输率有UDMA-33、UDMA-66、UDMA-133等，32位数据带宽可联4个IDE设备。

SCSI ( Sma Computer System Interface ) 接口，以名称而言，是小型计算机系统接口，显然它的设计并非只为外存储器而设计，放在这里介绍是因为它最常用于小型机，作为服务器和 workstation 上的外存储器接口，事实上，也有一些其他外设（如某些扫描仪等，也使用SCSI接口）SCSI有多个版本，带宽和数据传输率也不断上升，其中Fast SCSI-II带宽为16位，传输速率为20Mb/s,而Fast/Wide SCSI II的性能指标是它的两倍。

SCSI更可以称为一种总线，可以以雏菊链（在中断中见过这个词，不是吗？）形式接入多个外设，这些外设被分配唯一的标号，既可以相互交换数据，也可以和主存交换数据。

2.串行接口和并行接口

串行接口是一次发送一位信息，而并行接口则每次发送多位信息。并行接口必须使用多条数据线，适用于短距离的高速连接，而串行接口则实现比较复杂。

串行接口有两种方式，同步通信和异步通信。

异步通信接口传送数据时有确定的起始位、数据位、停止位，必要时还有检验位。可以有长度位，也可以只用停止位；同步传输只有数据位，没有起始位等额外开销，但接口两方的时钟要求一

致。

### 3.PC Card接口

原来的PCMCIA ( Persona Computer Memory Card Internationa Association ) 是一种笔记本电脑上常见的接口，特点是体积小、扩展灵活。现常见于笔记本电脑外接存储器、传真/调制解调器卡、网卡等扩展的接口。

### 4.USB接口

个人计算机对外标准接口曾经只有串口和并口，这两种接口的速度及性能都难以适应新设备的发展，于是有了USB和1394两种接口。

USB系统永远是以个人计算机为主站，而连接到主站的都称为设备，这是和串口不同的地方。

USB接口的特点是：

通过HUB可以扩展连接到127的外设。

速度快、USB 1.0为1.2Mb/s,USB2.0的达到480Mb/s.

设备安装配制容易：不必开机箱，也不必考虑中断等因素。

支持热拔插。

由于以上特点，USB已经成为个人计算机标准接口之一。

### 5.1394接口

1394接口广泛用于娱乐性外设，如数字摄像机等，是一种串行接口、速度可达400Mb/s新的标准是800Mb/s1394使用维菊链，一个端口可支持63个外设口，而且使用桥互联的方式，以树型结构配制，可支持高达6 022个设备。

在许多方面，如支持热拔插、外设容易安装配置等，USB和1394有相似的地方，这也使得这两种接口有一定的竞争关系。由于USB的实现简单，在PC上发展势头很猛，而1394接口制定本就是以"高速"为核心的，能出色实现人们对影音方面大容量、高速度的需求，这两种接口在未来的一段时间内仍然会在各种应用领域相互竞争。

版权方授权希赛网发布，侵权必究

[上一节](#)

[本书简介](#)

[下一节](#)

## 联机、脱机和假脱机

### 7.3.6 联机、脱机和假脱机

联机操作是输入/输出在计算机直接控制下进行的。联机时，操作者"正在"使用计算机资源，脱机操作是输入/输出操作在要进行操作的计算机以外的设备上进行，在需要时再送计算机处理。

假脱机 ( Spooing ) 是指输入/输出不直接送往输入/输出设备或计算机，而是先送到外存储器，典型的例子是打印。联机、脱机和假脱机的行为，不仅仅用于输入/输出系统，事实上，许多大型的高负荷的软件系统设计都会利用这些技术。

版权方授权希赛网发布，侵权必究

总线结构、指令流和数据流

7.4 总线结构、指令流和数据流

本节将介绍总线结构、指令流和数据流。

版权方授权希赛网发布，侵权必究

三态门

7.4.1 三态门

三态门的结构如图7-10所示。

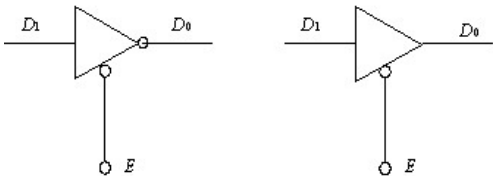


图7-10 三态门结构

和0,1两个状态的门电路相比，三状态增加了一个高阻状态，又称为"悬空"、"断开"等。这个状态实际上意味着什么也得不到，实现三状态的逻辑电路称为三态门。

如图7-10所示，当 $E=1$ 时，(a)图中 $D_0 = D_1$ ，而(b)图则 $D_0 = \sim D_1$ 为反相。 $E=0$ 时， $D_0$ 为高阻状态。在电路设计中，也可把 $E=1$ 作为高阻状态。

这里的E对于能否输出显得非常重要，在作为总线结构的关键原件时，就称为"总线有效信号"。

版权方授权希赛网发布，侵权必究

总线

7.4.2 总线

总线在前面已经略有提及，它是计算机中各部件相连的通信线，通过总线，各部件之间可以相互通信，而不是每两个部件之间相互直连，减少了计算机体系结构的设计成本，有利于新模块的扩展，如图7-11所示。

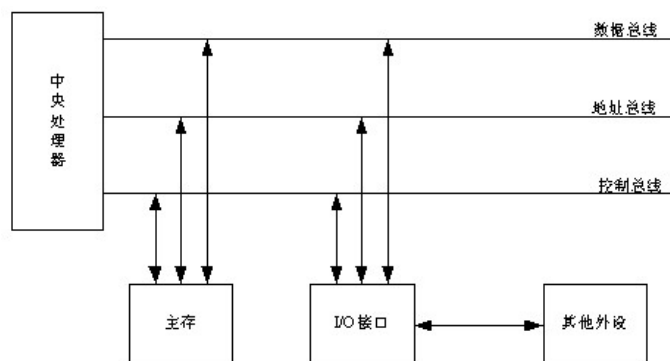


图7-11 总线图

从中央处理器到外设，总线可以分为4类。

#### 1) 中央处理器内部总线

这是内部各部件之间的信息传送线，寄存器与寄存器之间、各寄存器与运算器之间的信息传送线。

#### 2) 部件内总线

一块插卡的内部总线，又称片级总线，像显卡、多功能卡等插卡都使用了部件内总线实现本卡上的各种芯片的互联。

#### 3) 系统总线

系统总线是计算机系统内各功能部件（中央处理器、存储器、外设等）之间相互联接的总线。

从位置上来说，一般位于计算机系统的底板上。它从功能上说，可以分为以下几种。

数据总线：一般是由三态门控制的双向数据信道，中央处理器通过数据总线和主存、外设交换数据。

地址总线：常常由三态门控制的单向数据信道，由中央处理器“点名”取数的位置。

控制总线：用来传递控制信号。如读/写信号、中断请求、复位等信号。

#### 4) 外总线

它是计算机系统之间，或者是和计算机和其他设备通信的总线。

版权方授权希赛网发布，侵权必究

[上一节](#)

[本书简介](#)

[下一节](#)

## 常见的个人计算机系统总线的比较

### 7.4.3 常见的个人计算机系统总线的比较

最早的是PC/XT 电脑采用的系统总线，它基于8位的8088 处理器，被称为PC总线或者PC/XT总线。1984年，出现了基于16位Intel 80286处理器的PC/AT 电脑，系统总线也相应地扩展为16位，并被称呼为PC/AT 总线。以上两种总线规范为基础的ISA（工业标准架构：Industry Standard Architecture）总线，方便了开发与个人计算机兼容的外围设备ISA是8/16位的系统总线，最大传输速率为8MB/s,允许多个CPU 共享系统资源。由于兼容性好，它在20世纪80年代得到了广泛采用，其缺点是传输速率过低、CPU占用率高、占用硬件中断资源等。

当出现了32位外部总线的386DX处理器之后，ISA的宽度就已经成为严重的瓶颈，并影响到处理

器性能的发挥。新的EISA ( Extended ISA,扩展ISA ) 总线是为适应32位中央处理器而制定的。EISA总线的工作频率仍旧是8MHz,并且与8/16位的ISA总线完全兼容,带宽提高了一倍,达到了32MB/s.

为了解决ISA和EISA速度慢的问题,出现了32位的PCI ( 周边组件互联 ) 总线。最初PCI 总线工作在33MHz 频率之下,传输带宽达到了133MB/s ( 33MHz×32位/8 ),后来又提出了64位的PCI总线,并把PCI 总线的频率提升到66MHz~100MHz,甚至更高。PCI不像ISA总线那样把地址寻址和数据读写控制信号都交给微处理器来处理,而是独立于处理器,采用了独特的中间缓冲器设计,可将显示卡、声卡、网卡、硬盘控制器等高速的外围设备直接挂在CPU 总线上,打破了瓶颈,使得CPU 的性能得到充分的发挥。PCI总线还具有能自动识别外设,与处理器和存储器子系统完全并行操作的能力,具有隐含的中央仲裁系统,采用多路复用方式 ( 地址线 and 数据线 ) 减少了引脚数,支持64位寻址,完全的多总线主控能力,提供地址和数据的奇偶校验等特点。

PCI的缺点是存在中断共享冲突,只能支持有限数量设备等问题,可支持10台外设  
后来3D显示卡的迅猛发展,让PCI总线吃紧,就出现了作为PCI 总线的补充--AGP 总线。  
AGP ( 加速图形接口, Accelerated Graphics Port ) 接口,为显示卡专用,工作频率为66MHz,是PCI的两倍,带宽相应地增加到266MB/s.后来的AGP 2X、AGP4X,以及AGP8X ,传输速度达到了2.1GB/s.

三大总线的比较如表7-6所示。

	PC-XT 总线	ISA 总线	PCI 总线
数据总线宽度	8	16	32/64
地址总线宽度	20	24	32/64
时钟频率 ( MHz )	4.77	8.33	33/66

表7-6 三大总线对比

版权方授权希赛网发布，侵权必究

[上一节](#)      [本书简介](#)      [下一节](#)

总线仲裁

7.4.4 总线仲裁

如前所述,总线由多个部件和设备共享,必须有一个总线控制机构,对总线的使用进行合理分配和管理,以避免使用过程中发生冲突。

在DMA方式中,我们已经提到过DMA控制器和中央处理器之间需要交换控制权,显然当有多个设备请求总线时,总线管理机构必须对这些设备访问总线的优先级进行判断,并把总线的控制权交给合适的设备,而同时请求总线的其他设备,则必须等待。

从总线仲裁控制机构的位置可以把总线控制方式分为集中式控制和分布式控制。集中式和分布式这两个词我们能够在许多地方看到,分布式数据库、分布式网络、分布式计算等在软件方面也大量在研究集中和分布的问题。从辩证的观点来说,两者各有千秋,不过是在不同的发展阶段,不同的软硬件成本、不同的开发成本上的不断比较和更新。不过这个真理对考试而言基本没有什么帮助,分布式总线控制常用于称为现场总线的工业控制应用中,其监视、控制设备相对简单。

集中式控制的总线控制逻辑常常在于中央s处理器,所有需要使用总线的设备必须向总线控制器



提出请求，等待总线控制器将总线的控制权交给自己，才开始相应的工作。当多个设备请求总线控制权时，总线控制器必须采用某种方式来进行仲裁，常见的有3种仲裁方式。

1.链式查询方式

链式查询方式如图7-12所示。

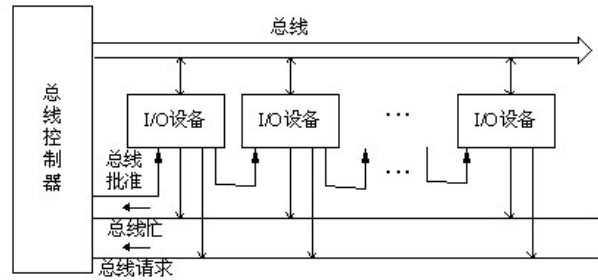


图7-12 链式查询

这个图形是不是很熟，它和中断中的"硬件查询法"比较像吧？

在图7-12中，总线控制器和设备之间有3根信号线。

总线请求：当总线请求有效时，表示至少有一个设备在请求使用总线。

总线忙：当总线忙有效时，表示总线正在被系统中某个设备使用。在这种情况下，总线请求的信号是会被总线控制器响应的。

总线批准：当总线批准有效，表示总线控制器响应总线请求，总线可以被使用。

当某个设备需要使用总线时，发出总线请求信号，如果此时总线空闲（总线忙信号为无效的情况），总线控制器响应该总线请求信号，并发出总线批准信号，这个信号按照次序经过每一个设备。对于某个设备而言，收到总线批准信号，如果自身没有发出总线请求信号，就把总线批准信号向下一个设备传递，直到一个发出总线请求信号的设备。这个设备停止传递总线请求信号，并获得总线的使用权。如果这个设备后面的设备也发出过总线请求信号，那个设备得不到总线批准信号，当然也就无法使用总线。这样在硬件结构上实现了总线使用的优先级。

链式查询方法的优点是实现简单，容易扩充。但是优先级固定，不能更改，而且可能造成低优先级的设备长时间无法得到总线控制权的情况。

2.计数器定时查询方式

这是一种可以灵活改变设备优先级的方式，如图7-13所示。

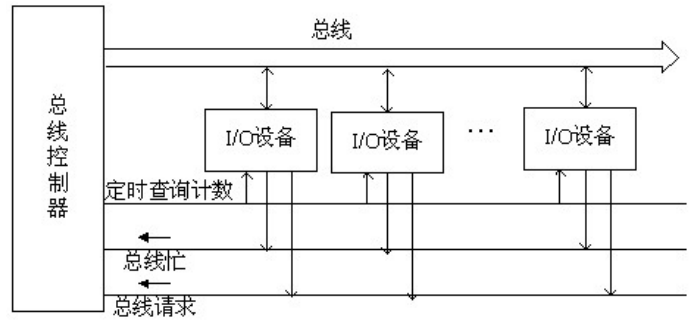


图7-13 计数器定时查询方式

和链式查询方式比较，没有了总线批准信号线，取而代之的是定时器查询计数。当总线控制器收到总线请求时，如果总线空闲，则启动计数器，开始计数，当定时查询计数信号线上的计数值和请求总线的设备一致的时候，该设备就置总线忙，并获得总线控制权。总线控制器收到总线忙，则计数停止。

这种方式利用计数器的起始值来获得设备的优先级，如果每一次计数都从0开始，那么优先级划分和链式是相同的，如果每次从上次停止的位置开始计数，则各设备的优先级相同。如果由程序控

制计数器的初值，就能动态改变各设备的优先级，从而提高了系统的灵活性。

### 3.独立请求方式

和上面两种方式不同，独立请求方式每个设备有专门的信号线，而不是所有的设备共享信号线，如图7-14所示。

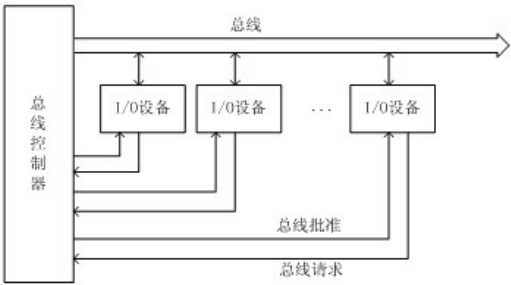


图7-14 独立请求方式

这个和中断中的第一种是不是很像？独立请求方式以增加控制线数和硬件电路为代价来得到响应时间快的优点，以硬件的增加来提高速度的方法读者应该见过很多了。同时总线控制器可以由程序控制响应这些请求，可以灵活地制定优先级，能够动态地改变优先级。

取得控制权的设备称为主设备，总线的控制权决定之后，主设备就开始和另外一个设备开始通过总线进行数据通信，这个设备称为从设备。一个设备在这次是主设备，下一次可能是从设备。

版权方授权希赛网发布，侵权必究

[上一节](#)      [本书简介](#)      [下一节](#)

## 指令流、数据流和计算机的分类

### 7.4.5 指令流、数据流和计算机的分类

有关定义如下。

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，包括输入数据和中间结果。

#### 1.根据多倍性的分类

按照计算机同时处于一个执行阶段的指令或数据的最大可能个数，人们把计算机分成4种，如表7-7所示。

数据流 \ 指令流	指令流	
	单（single）	多（multiple）
单（single）	SISD	MISD
多（multiple）	SIMD	MIMD

表7-7 指令流的分类

SISD:这是最简单的方式，计算机每次处理一条指令，并只对一个操作部件分配数据。一般认为流水线技术的计算机仍然属于SISD.

SIMD:具备SIMD点的常常是并行处理机，这种处理机具备多个处理单元，每次都执行同样的指令，对不同的数据单元进行处理。这种计算机非常适合处理矩阵计算等。

MISD:这种处理方式比较难以想像，有多个处理单元，同时执行不同的指令，针对的是单一数

据。但有资料认为，流水线处理机中每个数据由不同操作部件对之进行处理。

MIMD:这是一种全面的并行处理机，典型的机型是多处理机。这种计算机的设计和控制都很复杂。

## 2.按照程序流程机制的分类

### 1) 控制流计算机

这是通常见到的计算机，使用程序计数器（PC）来确定下一条指令的地址。指令程序流由程序员直接控制，其主存是共享的，存储区可以被多指令修改，容易产生数据相关性，对并行性不利。

### 2) 数据流计算机

在冯·诺依曼体系中，是指令流驱动的，而数据流则是处于被动地位的，这看起来合理，但在某些时候也不尽然。相对的是数据流驱动，即一旦数据准备好，则立即开始执行相关的指令，非冯·诺依曼体系仍在探索中，但对冯·诺依曼体系的改良也有相当好的成果，即流水线技术和并行计算机。

在数据流计算机中，数据不在共享的存储器中，而是在指令间传送，成为令牌。当需要使用该数据的指令收到令牌，开始执行之后，该令牌即消失，执行的指令将执行的结果数据当做新的令牌发送，这种方式不再需要程序计数器、共享的存储器，但是需要检测数据可用性的专门部件，建立、识别、处理数据令牌标记，需要时间和空间开销。

在其他一些方面，数据流计算机还有一些困难需要克服，在数据流计算机中由于没有程序计数器，使得程序的调试和诊断变得困难，没有共享的存储器，也就无法控制其分配，无法支持数组、递归等操作。

### 3) 归约机（Reductions Machine）

又称为需求驱动，是由对一个操作结果的需求而起动的。归约机采用一种“惰性计算”的方式，操作只有在另一条指令需要这个操作的结果时才执行。比如在计算 $5 + (6 \times 2 - 10)$ 时，归约机并非先去计算 $6 \times 2$ ，而是先计算整个算式，碰到 $(6 \times 2 - 10)$ 再启动一个过程去计算它，最后需要计算 $6 \times 2$ ，计算后一层层退回，得到整个算术的值。由于需求驱动可减少那些不必要的求值操作，可以提高系统效率。

归约机是一种面向函数式的语言，或以函数式语言为机器语言的机器，要有函数定义存储器和表达式存储；操作和数据合并存储。需要大容量物理存储器并采用大虚拟存储容量的虚拟存储器，来满足对动态存储分配和大容量的存储空间的需求。

版权方授权希赛网发布，侵权必究

[上一节](#)

[本书简介](#)

[下一节](#)

## 流水线技术

### 7.5 流水线技术

本节将介绍流水线技术。

流水线

7.5.1 流水线

还记得美国人泰勒吗？他了不起地发现了工人的机械劳动可细分为若干个环节，当然他还有其他了不起的发现。这样，如果所有工人都遵循固定的、优化了的劳动程序进行劳动，劳动效率就大为提高，资本家们进一步发挥，发展到每个工人只执行全部劳动细节的一小部分，一个产品由多个工人共同完成，这使得劳动效率飞升，这种技术称为流水线。

在中央处理器处理指令时，为提高效率，人们也采用了这种技术，对中央处理器而言，这实际是一种以硬件增加来换取性能提升的方式，由于硬件成本的持续下降，越来越多的中央处理器采用了这个技术：把一个指令分解成多个更小的指令，由不同的处理单元来处理，这样就形成了流水线，在理想的满负荷运行的状态下，执行一条指令的时间虽然没有减少，但是多个处理单元同时工作，在同一时间可以执行不同的指令的不同部分，从而使得总体的执行时间大大减少，减少到最慢的那一步的时间，如果各步骤处理时间相等，则一条指令分解成多少步，则处理速度就能提高多少，如图7-15所示。

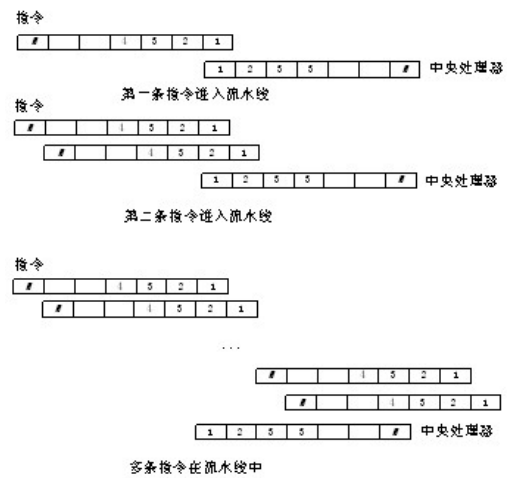


图7-15 流水线的进行方式

例如，如果一个中央处理器把执行指令分成7步，如表7-8所示。

表7-8 执行指令的七大步骤

步 骤	简 述	CPU 周期数
1.PC(程序计算器)	PC 自动加 1	1
2.取址	从 PC 指示的地址中取得完整指令	4
3.译码	可是硬件译码，也可能是微程序译码	1
4.取操作数地址	如果操作数是立即数可省略	1
5.取操作数		4
6.执行		1
7.保存结果		4

这种情况下，理想的满负荷执行的多个指令执行时间为4个CPU周期。

更进一步讲，中央处理器进行运算操作也可以利用流水线计数，例如：一个浮点加法运算，通常分解为3个阶段，如图7-16所示。

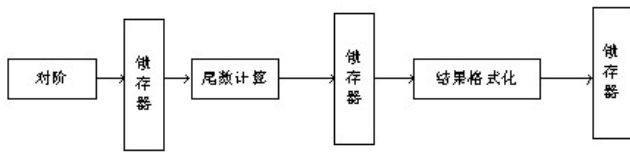


图7-16 浮点运算的分解

锁存器的作用是在各步骤之间保存中间结果，这样理想状态下中央处理器把浮点加法的运算速度提高3倍。

上面多次提到理想状态和负荷两个词，显然在中央处理器流水线工作时，后面部分的处理单元处于无所事事的等待状态，只有在所有的工作单元都开始工作时，流水线才处于负荷的状态下。

理想状态是指没有阻塞的情况，现实中的流水线如某一环节出了问题，流水线的速度就会大为降低，中央处理器也是这样，那么影响流水线的因素有哪些呢？

版权方授权希赛网发布，侵权必究

[上一节](#)
[本书简介](#)
[下一节](#)

## 影响流水线效率的因素

### 7.5.2 影响流水线效率的因素

#### 1. 条件转移指令

最常见的就是条件转移指令了，在存在转移指令的情况下，下一条需要执行的指令未必是程序计数器所指定的指令。只有在这条转移指令执行完成后，才能判断下一条指令是什么。

如果在遇到转移指令时，关闭流水线的进入端口，防止错误发生，这种方法无疑降低流水线的效率，而且程序中的条件转移是大量存在的，这势必使得流水线在很多时间内闲置，影响计算机的性能。

有的计算机采用猜测法，当发现条件转移指令时则系统猜测可能会跳转到的语句，如果猜测正确，则流水线正常运行，如果猜测错误，则需要清空当前流水线的内容。如图7-17所示。

还有一种需要编译系统的支持，如图7-17所示，方法是这样的，即将必须执行的D指令提前执行，在D指令执行之后，条件转移指令的结果出来后，再判断是B或者C进入流水线。同样也可以把A指令前的指令滞后到A指令执行后执行，这样能保持流水线闲置的时候尽可能少。但是这必须是在D和B、C指令不存在前后依存关系的情况下采用。统计的结果还是不错的，50%的条件转移指令能够进行这样的优化。

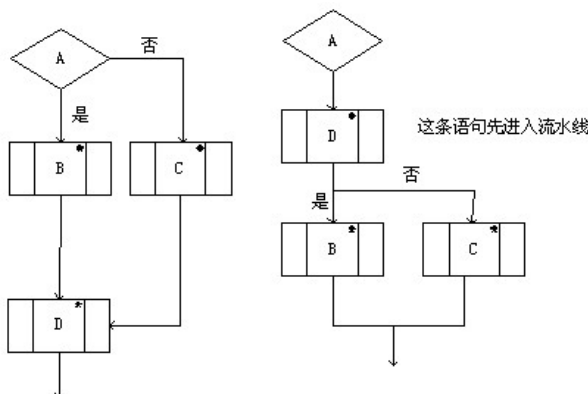


图7-17 进入流水线的条件转移指令

## 2.资源共享

由于使用流水线，若相邻的两条指令都对同一个资源进行操作时，或者前一条指令的输出是后一条指令的输入，在没有流水线情况下是正常的，在流水线时就可能出现错误。例如：前一条指令是写，后一条指令是读，当前一条指令保存结果没有完成时，后一条指令的读操作数就已经开始，这样后一条指令读到的就是未改写的数据。

为了解决这个问题，当遇到资源冲突时，就只好暂停后读指令进入流水线，就降低了流水线的效率，显然，流水线步骤越多越容易引起资源冲突的发生。

也可以在编译系统上作文章，当发现相邻的语句存在资源共享冲突的时候，在两者之间插入其他语句，将两条指令进入流水线的的时间拉开，以避免错误。

## 3.寄存器相关

如果相邻的指令使用了相同的寄存器，这也会使得流水线失常。

通常的解决方法是如果此时还有其他寄存器可用，则给两个指令分配不同的寄存器，以避免冲突的发生。这对拥有大量的通用寄存器的精简指令系统计算机是个不错的方法。

## 4.中断系统

当有中断发生时，和条件转移指令类似，流水线也不得不停止，以载入中断处理程序，由于中断的其他方面的优点，这种影响对流水线而言，是不可避免的。

流水线响应中断有两种方式，一种是立即停止现有的流水线，称为精确断点法，这种方法能够立即响应中断，缩短了中断响应时间，但是增加了中央处理器的硬件复杂度。

还有一种是在中断时，在流水线内的指令继续执行，停止流水线的入口，当所有流水线内的指令介绍执行后，再执行中断处理指令。这种方式中断响应时间较长，这种方式称为不精确断点法，优点是实现控制简单。

版权方授权希赛网发布，侵权必究

[上一节](#)    [本书简介](#)    [下一节](#)

第 7 章：计算机的体系结构和主要部件

作者：希赛教育软考学院    来源：希赛网    2014年01月27日

## 精简指令计算机

### 7.6 精简指令计算机

本节将介绍精简指令计算机。

版权方授权希赛网发布，侵权必究

[上一节](#)    [本书简介](#)    [下一节](#)

第 7 章：计算机的体系结构和主要部件

作者：希赛教育软考学院    来源：希赛网    2014年01月27日

## 指令系统

### 7.6.1 指令系统

指令系统是中央处理器所有指令的汇集。也是高级软件编制的基础。指令系统的选择和确定涉及很多方面，是个复杂的问题。

通常一个指令可分解为：

前半部分的操作码确定指令的类型，后面的地址码确定指令所要处理的数据，根据地址码的个数可以有四址指令、三址指令，甚至是0址指令。

根据指令的长度特点，一个指令系统可能是定长指令字结构，即指令系统中所有的指令的长度都相同，特点是控制简单。如果指令的长度不固定，复杂的指令长度较长，就是变长指令字结构，这个结构的指令很容易扩展，但是增加了硬件的复杂度。

#### 1.寻址方式

根据地址码代表的地址类型，指令系统可以有如下几种。

##### 1) 立即寻址

地址码就是操作数，这种寻址方式不必再次访问内存去取操作数，当然，也无须修改操作数。

##### 2) 直接寻址

地址码就是主存内数据的绝对地址，不必做任何换算。不足之处在于寻址范围有限，地址码的位数限制了寻址空间，而计算机的发展趋势是计算机拥有越来越大的内存。如果使用变长指令结构，则该指令就会变得臃肿。

##### 3) 寄存器寻址

地址码的地址是寄存器的地址。和内存寻址比较而言，访问寄存器的速度是非常快的，所以使用寄存器寻址有非常快的速度，不足之处在于寄存器的数量有限。

##### 4) 间接寻址

地址码指向主存中的数据，这个数据仍然是一个地址，这种方式提高了寻址的灵活性，扩大了寻址的范围。但由于要多次读主存，速度大为降低。

##### 5) 寄存器间接寻址

地址码保存的是寄存器地址，相对应的寄存器中保存的是数据的地址，这样既快又有灵活性，是一种广泛使用的寻址方式。

##### 6) 变址寻址

变址寻址，是将地址码和变址寄存器内容相加后形成数据的地址，由于变址寄存器能自动修改，这种方式对数组运算、字符串操作等批量数据的处理非常有效。

##### 7) 基址寻址

和变址寻址类似，是将变址寄存器改成基址寄存器，不同之处在于基址寄存器的内容是基本不变的，改变的是指令中地址码部分的内容，主要作用是扩大寻址空间和再定位。

##### 8) 页面寻址

页面寻址要将主存分组成若干个区，寻址时使用区码加区内偏移的方式确定位置。

##### 9) 相对寻址

相对寻址也和变址寻址类型相似，这里变址寄存器换成了程序计数器。

#### 2.指令类型

就功能而言，常用的指令有如下几种。

##### 1) 数据传送指令



这些指令负责数据在寄存器，主存之间的传递，其中数据交换指令是数据的双向移动，而堆栈操作指令是专门对堆栈的PUSH和POP操作。

## 2) 运算类指令

包括算术运算、逻辑运算、移位运算方面的指令，这类指令一般会改变标志寄存器的状态。算术运算和逻辑运算的区别见7.1.3节。

移位运算是一种特殊的运算，分算术移位、逻辑移位和循环移位3类。算术移位中左移在空位补0,右移则补符号位。逻辑移位无论左移右移都补0.循环移位是把移动空位由移出的位来填补的移位，又分带符号位移动的大循环和不带符号位移动的小循环。

其中算术左移等于是做乘以2的操作，而算术右移等于做除以2的操作。

## 3) 程序控制指令

控制程序流程改变的指令非常多，包括条件转移指令、无条件转移指令、循环控制指令、程序返回指令、中断指令等。

## 4) 输入/输出类指令

这类指令只有输入/输出独立编码才需要，对于统一编码的部分而言是不需要的。

## 5) 数据处理类指令

数据处理类指令包括一些比较复杂的指令，如数据转换指令、字符串操作指令、压缩和扩展指令等。

版权方授权希赛网发布，侵权必究

[上一节](#)

[本书简介](#)

[下一节](#)

# CISC和RISC

## 7.6.2 CISC和RISC

随着硬件成本的下降，人们倾向于向中央处理器加入越来越多、越来越复杂的指令，同时，为了兼容老产品，原来的指令也要保留。这样，整个指令系统就向着越来越大、越来越复杂的趋势发展。在计算机处理能力越来越强的同时，中央处理器的设计也越来越复杂，这无疑在大大增加了设计的同时，更增加了设计失误的可能性。

事物的另外一个方面在于，增加指令的复杂性和中央处理器功能的增加似乎不一定是成正比的，人们发现在许多方面存在一个称为20%~80%的定律，即系统中20%的部分发挥了80%的作用，通过对CISC指令系统的研究，发现系统在80%的时间里执行的是20%的指令。

于是出现了精简指令设计思想。这种计算机的指令结构不追求全面和复杂，而是只实现那些经常被执行的指令，由于指令比复杂指令结构计算机少得多，所以称为精简指令计算机。

先看著名的公式： $P = I \times CPI \times T$

式中 P-计算机执行程序所需要的时间；

I-机器指令数；

CPI-平均每条指令所需要的机器周期数；

T-每个机器周期的时间。

CISC在指令数上占优，而RISC在CPI上有则快得多，这是两种结构的两个方向。从这个公式可以发现，在理论上两者都有优势，不能认为精简指令计算机就好，复杂指令计算机就不好，事实上这两种设计方法很难找到完全的界线，而且在实际的芯片中，这两种设计方法也有相互渗透的地方，如表7-9所示。

表7-9 两者的简单对比表

	CISC	RISC
指令条数	多	只选取最常见的指令
指令复杂度	高	低
指令长度	变化	短、固定
指令执行同期	随指令变化大	大多在一个机器同期完成
指令格式	复杂	简单
寻址方式	多	极少
涉及访问主存指令	多	极少，大部分只有存两条指令
通用寄存器数量	一般	大量
译码方式	微程序控制	硬件电路
对编译系统要求	低	高

[版权方授权希赛网发布，侵权必究](#)

[上一节](#)

[本书简介](#)

[下一节](#)

## RISC结构特点

### 7.6.3 RISC结构特点

#### 1.精简指令系统采用硬件布线逻辑控制

不是说硬件方式不够灵活吗？确实，对于复杂指令系统这是它的致命缺点，但对于指令少、格式简单的精简指令系统而言，采用微程序控制就显得多余了，而硬件方式占用元件少的优点体现出来了，而腾出的地方可以在中央处理器中放置更多的通用寄存器，同时硬件方式的译码速度也要快得多。

#### 2.大量通用寄存器

RISC大量减少了对内存的访问指令，许多精简指令计算机只有两条指令和内存交换数据：LOAD和STORE.计算的中间结果放在寄存器中，由于访问寄存器的速度远远高于访问主存的速度，这种减少存储器访问次数的做法提高了精简指令计算机的效率。

同时，大量的通用寄存器使得可以使用寄存器完成程序调用的参数传递，这样在程序调用时，减少了对主存的访问，而在结构化的程序中，大量使用程序调用，这无疑也提高了精简指令计算机的效率。

精简指令计算机采用称为“重叠寄存器窗口技术”来组织寄存器，设有3个过程，如图7-18所示。

在图7-18中X的输出是Y的输入，通过这种方式传递参数，不需要保护现场，减少了堆栈的操作。当然这样的调用，层数是有限制的，当层数过多，甚至是递归函数时，就一定要借助于主存的堆栈了。

#### 3.RISC普遍采用流水线技术

由于RISC指令结构简单，除了访问主存的指令速度慢一些外，其他指令指行时间大多相同，这有利于发挥流水线的优势。同时，大量的寄存器使得编译优化成为可能，可以尽量减少数据相关的

问题，减少流水线闲置时间。

#### 4.RISC的存储系统

为了加快主存存取，大多数精简指令计算机采用了缓存技术，有的是把指令和数据缓存分开设置，这样取指和读数可以同时进行。当指令或数据在缓存中时，就不需要去主存中提取，如果频繁地修改某个地址的数据，数据缓存也能减少写主存的次数。显然缓存越大，越能减少主存存取次数。

如果缓存命中失败，就必须访问主存，这就增加了控制电路的复杂性。

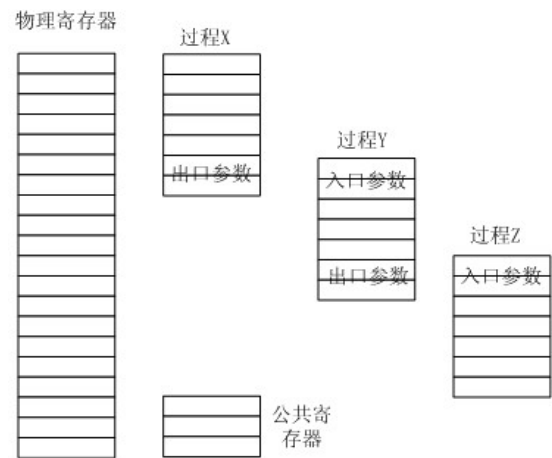


图7-18 寄存器窗口

#### 5.对编译器的影响

精简指令系统对编译系统的影响有两个方面，一方面由于指令简单，选择就有限、寻址方式少，这样的编译系统不需要在众多的指令中选择最佳指令和指令组合，这降低了对编译系统的要求。但另外一个方面，由于存在大量的通用寄存器，如何充分利用这些寄存器，如何优化编译，使得编译后的指令代码更能适应流水线的要求，这都对精简指令计算机的编译器提出了更高的要求。而如果没有编译器的支持，精简指令计算机的多寄存器、流水线技术、移动寄存器窗口等都难以发挥最佳的作用。

#### 6.ARM处理器简介

在ARM 32位体系结构是32位嵌入式RISC结构，ARM是一类处理器，有多个公司生产基于ARM体系结构的处理器产品。ARM使用了加载/存储体系结构，固定指令长度，但没有实现寄存器窗口，流水线改良的延迟转移和所有指令同期。

版权方授权希赛网发布，侵权必究

[上一节](#)      [本书简介](#)      [下一节](#)

### 并行处理技术和并行处理机

#### 7.7 并行处理技术和并行处理机

本节将介绍并行处理技术和并行处理机。

版权方授权希赛网发布，侵权必究

## 概念

---

### 7.7.1 概念

并行即是两个工作进行过程中，存在时间上的重叠。

有关并行的严格定义介绍如下。

#### 1.同时性

多个事件同时发生。

#### 2.并发性

多个事件在同一个时间隔内发生。

虽然绝对同时是不存在的，只是事件的间隙太小，以至于不能被机器识别。

#### 3.控制并行

控制并行使多个操作同时进行，如流水线，它的并行性受到流水线长度、功能单元等的限制，对程序员而言，这种并行是透明的。

#### 4.数据并行

数据并行是多个相同的处理单元对多个不同的数据同时进行相同的处理，这种方式开发潜力最大，与控制并行相比，代码容易编号和调试。

#### 5.粒度

粒度是指进程中所含计算量的尺寸大小，通常使用其中的指令数作为计量方式。

#### 6.时延

时延是各子系统之间通信开销的时间量度。

#### 7.并行度

在并行计算机中，单位时间内执行程序的处理单元的数量。

并行性是一个广泛的概念，从硬件到软件各个层次都在利用并行性以提高计算机性能。

版权方授权希赛网发布，侵权必究

## 并行性的等级

---

### 7.7.2 并行性的等级

从低到高级别的并行性等级介绍如下。

#### 1.指令或语句级

典型粒度（指令数）小于20,一条指令的各微指令同时运行。优点是可借助于优化的编译器，自动将源码编译成并行指令。

## 2.循环级

如果循环中的指令不相关，可使之向量化，让这些指令同时执行。粒度大约不到500条，是在并行或向量计算机上运行的最优程序结构。

以上两种均认为是细粒度计算。

## 3.过程级和子程序级

典型粒度小于2 000条。这种并行性比以上两种细粒度级要困难，需要分析过程之间的相关性。目前，还没有好的编译器能编译出好的中粒度或粗粒度程序。

## 4.作业程序级

对应的是独立的程序，粒度数为几万条指令。一般由加载程序和操作系统来处理这种级别的并行。时间轮片法、非占先式、占先式等方法为操作系统常用的程序并行调动法。详细的讨论请参考操作系统部分。

一般粒度越细，并行潜力越大，同时通信和调度的开销也增加。另外，最早硬件只处理指令内部的并行性，而由操作系统等软件负责程序的并行，而现在发展趋势是硬件越来越冲当中粒度和粗粒度的并行实现者，多处理机就是这样的例子。

上面讨论的是指令执行方面的并行性，在数据并行性方面也有不同的级别，从低到高分别是：一次只对一字的一位进行处理的位串字串方式，一次对一字的全部位进行处理的位并字串方式，一次对许多字的同一位进行处理的位串字并方式，对许多字的全部或部分位进行处理的位并字并方式。

[版权方授权希赛网发布，侵权必究](#)

[上一节](#)

[本书简介](#)

[下一节](#)

## 提高计算机并行性的措施

### 7.7.3 提高计算机并行性的措施

分时也可以看做是一种并行，从宏观上看各任务是并行的，实际上是把时间切片，各工作之间占一定时间片，虽然实际上没有时间上的重叠，但是在各任务没有真正完全占用资源的情况下特别有效，并具备低成本的优势。这样的例子有很多，DMA控制器和中央处理器实现对总线控制转移的方式之一就是分时，操作系统在单个中央处理器的硬件系统上实现各任务的并行时，其实也是采用分时的方法。

分时可能是各部分平均占有不同的时间片，也可以占有的时间片各不相同，占有时间片多的优先级高。分时处理也称为资源共享。

更多的并行是以空间换时间，使用多个功能单元同时工作。多个功能单元可能是同类型的，如阵列机，这种方式也称为资源重复。资源重复也经常用于提高系统的可靠性方面，随着硬件价格的不断降低，资源重复越来越多地用在了提高计算机并行处理能力上。

多个功能单元也可能是不同类型的，如流水线技术将读取、译码、执行等工作分开，形成流水线。这种方式也称为时间重叠，在同一个时间内完成任务的不同功能。

并行性仍然在不停的发展之中，在输入处理中，中断技术、DMA技术、通道技术都是为不断提

高并行性而发展处理的，在指令处理方面的流水线技术、并行计算机、多处理机、大规模并行处理机（MPP）、对称多处理机（SMP）的不断发展也提高了计算机的并行处理能力。其中，MPP和SMP集中几千个中央处理器进行协作计算。在软件方面，编译系统对指令的优化、操作系统对任务的调度都在提高系统的并行性。

在提高计算机并行性的各方面，硬件的发展最迅速，而主要的问题是集中在软件和应用方面。编译器对粗粒度的并行性还没有好的解决方法，操作系统对多处理器的支持还有限，特别是对并行计算的编程仍然十分困难和费力。

版权方授权希赛网发布，侵权必究

[上一节](#)    [本书简介](#)    [下一节](#)

## 并行处理机

### 7.7.4 并行处理机

并行处理机是基于SIMD结构的，即以增加硬件资源，同时对多个数据进行类似的处理，提高速度的方法可以是简单地增加处理单元的个数，所以潜力大。

并行处理机只有一个控制单元，但有多个处理单元。在控制单元的控制下，所有处理单元进行同样的运算，不同的是进行运算的数据不同。如果是非并行计算的程序部分，则由控制器完成。从这里我们可以看出，并行处理机特别适用于矩阵计算。所以，并行处理机也称为矩阵计算机。

#### 1.并行处理机性能参数

处理单元个数。

控制器的指令系统。控制器处理串行处理指令和量类指令。

运算器的指令系统。包括了算术、逻辑运算、数据寻位、屏蔽等数据操作指令。这些指令是有控制器传播到处于“活跃”状态的运算器上，并有运算器执行。这些指令是向量类指令。

屏蔽方案集。控制器如何允许禁止某个运算器。

数据寻位功能集。运算器之间通信的设置模式的集合。

按照存储器的分布，并行处理器分为共享式并行处理机和分布式存储并行处理机，集中和共享看来是计算机系统永恒的两个方面。

#### 2.分布式存储并行处理

分布式存储并行如图7-19所示。

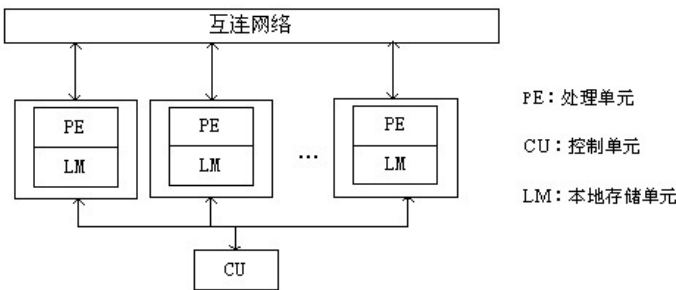


图7-19 分布式存储并行

在分布式存储并行处理机中，主要是使得每个处理单元需要使用的数据尽量在本地存储单元

中，如果需要其他处理单元的数据，就不得不需要借助于互联网或者是控制器的转发了。

### 3.共享式分布处理机

共享式分布处理如图7-20所示。

和分布式存储并行处理机不同的是增加了共享的存储器，共享的存储器难免在不同的处理单元访问时发生冲突，这就需要设计一些算法来分配数据，所有共享存储器并行计算机的互联网络结构也更加复杂。

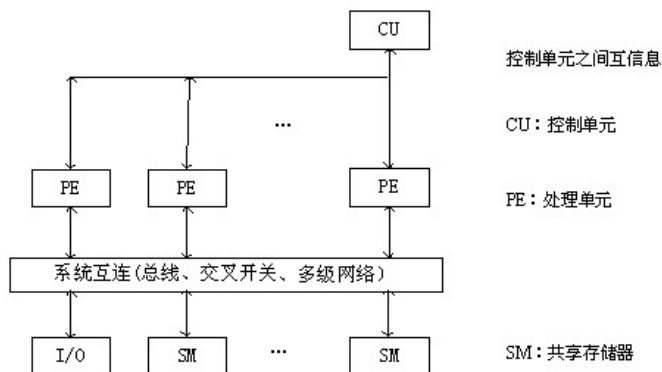


图7-20 共享式分布处理

### 4.并行处理机的特点

#### 1) 强大的向量运算能力

并行计算机通常是专用计算机，它在有限差分、矩阵、信号处理等向量计算和矩阵计算中有特别的优势，人们能够通过数学理论和算法把这些计算转换成对数组或者向量的处理，然后利用并行处理计算机的多个处理单元对数组或向量的不同数据部分进行相同的操作，这样能够获得很高的运算速度。所以，并行计算机特别适合于需要大量的这样运算的应用中。

#### 2) 使用资源重复的并行方式

并行计算机的各个运算器的运算是在控制器的操作下“同时”进行的，提高对大型计算的运算速度在于增加运算器的数量，而不是并行计算机的复杂程度，所以对于并行计算机适合的特定领域，并行计算机的能够处理非常庞大的运算。

#### 3) 标量运算的作用

在并行处理计算机中，不可避免地仍然存在普通的标量运算和程序控制，这些是由控制器来完成的，所以控制器的性能对并行处理机的性能也有很大的作用。

#### 4) 编译系统

具备识别高级语言中的向量计算的编译系统，对提高程序的通用性很有帮助。

版权方授权希赛网发布，侵权必究

上一节

本书简介

下一节

## 例题分析

### 例题1 (2011年5月试题1)

在CPU中用于跟踪指令地址的寄存器是\_\_ (1) \_\_。

(1) A. 地址寄存器(MAR)      B. 数据寄存器(MDR)

C. 程序计数器(PC)      D. 指令寄存器(IR)

例题分析：

本题主要考查寄存器的相关内容。

程序计数器是用于存放下一条指令所在单元的地址的地方。在程序执行前，必须将程序的起始地址，即程序的一条指令所在的内存单元地址送入程序计数器，当执行指令时，CPU将自动修改程序计数器的内容，即每执行一条指令程序计数器增加一个量，使其指向下一个待指向的指令。程序的转移等操作也是通过该寄存器来实现的。

指令寄存器一般用来保存当前正在执行的一条指令。

数据寄存器主要是用来保存操作数和运算结果等信息的，其目的是为了节省读取操作数所需占用总线和访问存储器的时间。

地址寄存器一般用来保存当前CPU所访问的内存单元的地址，以方便对内存的读写操作。

作为程序员，应该要能控制其所编写程序的执行过程，这就需要利用程序计数器来实现，因此程序员能访问的是程序计数器。

例题答案：(1) C

### 例题2 (2011年5月试题2)

指令系统中采用不同寻址方式的目的是\_\_(2)\_\_\_。

(2) A. 提高从内存获取数据的速度      B. 提高从外存获取数据的速度

C. 降低操作码的译码难度      D. 扩大寻址空间并提高编程灵活性

例题分析：

寻址是指寻找操作数的地址或下一条将要执行的指令地址。数据和指令一般都需要存放在一些存储器的存储单元中，存储器对这些存储单元进行编号，这些编号就是数据和指令的地址，如果在应用中需要用到这些数据或指令时，就通过它们的地址到存储器中去寻找，这就是寻址。

假如某主机的主存容量可达1MB，而指令中的地址码字段最长仅16位，只能直接访问主存的一小部分，而无法访问到整个主存空间，而采用不同的寻址方式可以实现对整个主存空间的访问。就是在字长很长的大型机中（地址码字段足够长），即使指令中能够拿出足够的位数来作为访问整个主存空间的地址，但为了灵活、方便地编制程序，也需要对地址进行必要的变换。

综上所述，可知本题答案选D。

例题答案：(2) D

### 例题3 (2011年5月试题3)

在计算机系统中采用总线结构，便于实现系统的积木化构造，同时可以\_\_(3)\_\_\_。

(3) A. 提高数据传输速度      B. 提高数据传输量

C. 减少信息传输线的数量      D. 减少指令系统的复杂性

例题分析：

总线是在计算机中连接两个或多个功能部件的一组共享的信息传输线，它的主要特征就是多个部件共享传输介质。它是构成计算机系统的骨架，是各个功能部件之间进行信息传输的公共通道，借助总线的连接，计算机各个部件之间可以传送地址、数据和各种控制信息。在计算机系统中采用总线结构，便于实现系统的积木化构造，同时可以有效减少信息传输线的数量。

例题答案：(3) C



#### 例题4 ( 2011年5月试题4 )

原码表示法和补码表示法是计算机中用于表示数据的两种编码方法，在计算机系统中常采用补码来表示和运算数据，原因是采用补码可以\_\_(4)\_\_\_。

- ( 4 ) A . 保证运算过程与手工运算方法保持一致  
B . 简化计算机运算部件的设计  
C . 提高数据的运算速度  
D . 提高数据的运算精度

例题分析：

在计算机中，各类运算都等可以采用补码进行，特别是对于有符号数的运算。在计算机中设计补码的目的一是为了使符号位能与有效值部分一起参加运算，从而简化运算规则，使运算部件的设计更简单；二是为了使减法运算转换为加法运算，进一步简化计算机中运算器的线路设计。因此在计算机系统中常采用补码来表示和运算数据，原因是采用补码可以简化计算机运算部件的设计。

例题答案：( 4 ) B

#### 例题5 ( 2011年5月试题5 )

计算机中的浮点数由三部分组成：符号位S，指数部分E（称为阶码）和尾数部分M。在总长度固定的情况下，增加E的位数、减少M的位数可以\_\_(5)\_\_\_。

- ( 5 ) A . 扩大可表示的数的范围同时降低精度  
B . 扩大可表示的数的范围同时提高精度  
C . 减小可表示的数的范围同时降低精度  
D . 减小可表示的数的范围同时提高精度

例题分析：

浮点数的表示形式如下：

$$N=M \times r^E$$

其中r是浮点数阶码的底，与尾数的基数相同，通常r=2。E和M都是带符号的定点数，E叫做阶码，M叫做尾数。浮点数的一般格式如图所示，浮点数的底是隐含的，在整个机器数中不出现。阶码的符号位为Es，阶码的大小反映了在数N中小数点的实际位置；尾数的符号位为Ms，它也是整个浮点数的符号位，表示了该浮点数的正、负。

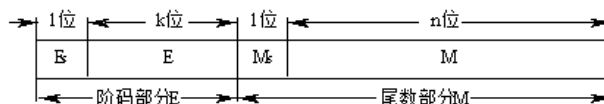


图7-29 浮点数的一般格式

浮点数的大小由阶码部分决定，而其精度由尾数部分决定，因此增加E的位数、减少M的位数可以扩大可表示的数的范围同时降低精度。

例题答案：( 5 ) A

版权方授权希赛网发布，侵权必究

[上一节](#)      [本书简介](#)      [下一节](#)