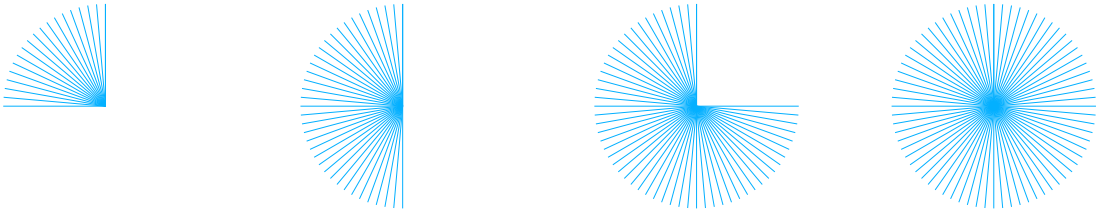
开课吧

计算机硕士

「高频考点」

计算机组成原理



考研

目　录

一、计算机系统概述

二、数据的表示和运算

三、存储器层次结构

四、指令系统

五、中央处理器(CPU)

六、总线

七、输入输出(I/O)系统

编写人：朱 俊 审核人：于方泽 封面设计：乔宇、刘锦浩、郑悦

勘误信箱:

亲爱的开课吧学员你好，感谢您选择开课吧开启学习之旅。在使用这份学习资料时，希望你能感受到开课吧全体教职人员的用心。经过软件人工三审三校的资料难免有万分之三的概率出现小差错，我们设立勘误激励邮箱，感谢大家和小开共同成长。在使用本资料的同时，如您火眼金睛发现了小错误。可将资料的名称-页码带图反馈到本邮箱:

kaikeba0104@126.com。一经采纳小开感激不尽。

一、计算机系统概述

(一)计算机层次结构

1.计算机硬件的基本组成

计算机硬件主要指计算机的实体部分,通常有运算器,控制器,存储器,输入和输出五部分.

CPU 是指将运算器和控制器集成到一个电路芯片中.

2.计算机软件的分类

计算机软件按照面向对象的不同可分两类:

系统软件:用于管理整个计算机系统,合理分配系统资源,确保计算机正常高效地运行,这类软件面向系统.(包括:标准程序库,语言处理程序,OS,服务程序,数据库管理系统,网络软件)

应用软件:是面向用户根据用户的特殊要求编制的应用程序,这类软件通常实现用户的某类要求.

3.计算机的工作过程

(1)计算机的工作过程就是执行指令的过程

指令由操作码和操作数组成:

操作码	地址码

操作码指明本指令完成的操作

地址码指明本指令的操作对象

(2)指令的存储

指令按照存储器的地址顺序连续的存放在存储器中【指令和数据一般放在一起；哈弗结构（指令和数据分开存储）】.

(3)指令的读取

为了纪录程序的执行过程,需要一个记录读取指令地址的寄存器,称为指令地址寄存器,或者程序计数器.指令的读取就可以根据程序计数器所指出的指令地址来决定读取的指令,由于指令通常按照地址增加的顺序存放,故此,每次读取一条指令之后,程序计数器加一就为读取下一条指令做好准备.

(4)执行指令的过程

在控制器的控制下,完成以下三个阶段任务:

1)取指令阶段　　按照程序计数器取出指令,程序计数器加一

2)指令译码阶段　　分析操作码,决定操作内容,并准备操作数

3)指令执行阶段　　执行操作码所指定内容

(二) 计算机的性能指标

1.吞吐量,响应时间

(1) 吞吐量:单位时间内的数据输出数量.

(2) 响应时间:从事件开始到事件结束的时间,也称执行时间.

2. CPU 时钟周期,主频,CPI,CPU 执行时间

(1) CPU 时钟周期:机器主频的倒数,TC

【总线的时钟周期】

(2)主频:CPU 工作主时钟的频率,机器主频 Rc【对指令的处理】

(3)CPI:执行一条指令所需要的平均时钟周期

(4)CPU 执行时间:

TCPU=In × CPI × TC

In 执行程序中指令的总数

CPI 执行每条指令所需的平均时钟周期数

TC 时钟周期时间的长度

【其中，TCPU/TC 可以获取到需要执行的时钟周期个数，即，频率】

3. MIPS,MFLOPS

(1)MIPS:(Million Instructions Per Second)

$$MIPS=\frac{\text{指令条数}}{\text{执行时间}\times 10^6}=\frac{In}{Te\times 10^6}=\frac{\text{时钟频率}}{CPI\times 10^6}=\frac{In}{In\times CPI\times Tc\times 10^6}=\frac{Rc}{CPI\times 10^6}$$

Te:执行该程序的总时间=指令条数/(MIPS × 10^6)

In:执行该程序的总指令数

Rc:时钟周期 Tc 的到数

MIPS 只适合评价标量机,不适合评价向量机.标量机执行一条指令,得到一个运行结果.而向量机执行一条指令,可以得到多个运算结果.

(2) MFLOPS: (Million Floating Point Operations Per Second)

MFLOPS=Ifn/(Te × 10^6)

Ifn:程序中浮点数的运算次数

MFLOPS 测量单位比较适合于衡量向量机的性能.一般而言,同一程序运行在不同的计算机上时往往会执行不同数量的指令数,但所执行的浮点数个数常常是相同的.

特点:

1.MFLOPS 取决于机器和程序两方面,不能反映整体情况,只能反映浮点运算情况

2.同一机器的浮点运算具有一定的同类可比性,而非同类浮点操作仍无可比性

二、数据的表示和运算

(一) 数制与编码

1.进位计数制及其相互转换

1)进位计数制

进位计数制是指按照进位制的方法表示数,不同的数制均涉及两个基本概念:基数和权.

基数:进位计数制中所拥有数字的个数.

权:每位数字的值等于数字乘以所在位数的相关常数,这个常数就是权.

任意一个 R 进制数 X,设整数部分为 n 位,小数部分为 m 位,则 X 可表示为:

$$X = a_{n-1}r^{n-1} + a_{n-2}r^{n-2} + \cdots + a_0 r^0 + a_{-1}r^{-1} + a_{-2}r^{-2} + \cdots + a_{-m}r^{-m}$$

$$(X)r = \sum_{i=n-1}^{-m} K_i r^i$$

2)不同数制间的数据转换

⑴二,八,十六进制数转换成十进制数

利用上面讲到的公式: (N)2=∑Di • 2^i ,(N)8=∑Di • 8^i, (N)16=∑Di • 16^i,进行计算.

⑵十进制数转换成二进制数

通常要对一个数的整数部分和小数部分分别进行处理,各自得出结果后再合并.

①对整数部分,一般采用除 2 取余数法,其规则如下:

将十进制数除以 2,所得余数(0 或 1)即为对应二进制数最低位的值.然后对上次所得商除以 2,所得余数即为二进制数次低位的值,如此进行下去,直到商等于 0 为止,最后得的余数是所求二进制数最高位的值.

②对小数部分,一般用乘 2 取整数法,其规则如下:

将十进制数乘以 2,所得乘积的整数部分即为对应二进制小数最高位的值,然后对所余数的小数部分部分乘以 2,所得乘积的整数部分为次高位的值,如此进行下去,直到乘积的小数部分为 0,或结果已满足所需精度要求为止.

⑶二进制数,八进制数和十六进制数之间的转换

八进制数和十六进制数是从二进制数演变而来的:

由 3 位二进制数组成 1 位八进制数;

由 4 位二进制数组成 1 位十六进制数.

对一个兼有整数和小数部分的数以小数点为界,小数点前后的数分别分组进行处理,不足的位数用 0 补足.

对整数部分将 0 补在数的左侧,对小数部分将 0 补在数的右侧.这样数值不会发生差错.

2.真值和机器数

真值:数据的数值通常以正(+)负(-)号后跟绝对值来表示,称之为“真值”.

机器数:在计算机中正负号也需要数字化,一般用 0 表示正号,1 表示负号.把符号数字化的数成为机器数.

【真值 --> 机器数 ，正数不变，负数（符号位除外，其余取反，+1）】

（二）定点数的表示和运算

1.定点数的表示

1)无符号数的表示

无符号数就是指正整数,机器字长的全部位数均用来表示数值的大小,相当于数的绝对值.

对于字长为 n+1 位的无符号数的表示范围为: $0\sim 2^{n+1}-1$

2)带符号数的表示（真值范围-n-1～n)

带符号数是指在计算机中将数的符号数码化.在计算机中,一般规定二进制的最高位为符号位,最高位为“0”表示该数为正,为“1”表示该数为负.这种在机器中使用符号位也被数码化的数称为机器数.

根据符号位和数值位的编码方法不同,机器数分为原码,补码和反码.

⑴原码表示法

机器数的最高位为符号位,0 表示正数,1 表示负数,数值跟随其后,并以绝对值形式给出.这是与真值最接近的一种表示形式.

原码的定义:

$$[X]_{原}=\begin{cases}X; & 0\le X<1\\ 1-X=1+|X|; & -1<X\le 0\end{cases}$$

⑵补码表示法

机器数的最高位为符号位,0 表示正数,1 表示负数,其定义如下:

$$[X]_{补}=\begin{cases}X; & 0\le X<1\\ 2+X=2-|X|; & -1<X\le 0\end{cases}$$

⑶反码表示法

机器数的最高位为符号,0 表示正数,1 表示负数.反码的定义:

$$[X]_{反}=\begin{cases}X; & 0\le X<1\\ 2-2^{-n}+X=2-|X|; & -1<X\le 0\end{cases}$$

	原码	补码	反码
整数	$[x]_{原}=\begin{cases}0,x & 2^n>x\ge 0\\ 2^n-x & 0\ge x>-2^n\end{cases}$	$[x]_{补}=\begin{cases}0,x & 2^n>x\ge 0\\ 2^{n+1}+x & 0\ge x>-2^n\end{cases}$ (mod 2^{n+1})	$[x]_{反}\begin{cases}0,x & 2^n>x\ge 0\\ (2^{n+1}-1)+x & 0\ge x>-2^n\end{cases}$ (mod$(2^{n+1}-1)$)
小数	$[x]_{原}\begin{cases}x & 1>x\ge 0\\ 1-x & 0>x\ge -1\end{cases}$	$[x]_{补}=\begin{cases}x & 1>x\ge 0\\ 2+x & 0>x\ge -1\end{cases}$ (mod 2)	$[x]_{反}=\begin{cases}x & 1>x\ge 0\\ (2-2^{-n})+x & 0>x\ge -1\end{cases}$ (mod$(2-2^{-n})$)

0	$[+0]_{原}$ =0.0000 ≠ $[-0]_{原}$ =1.0000	$[+0]_{补}=[-0]_{补}$=0.0000	$[+0]_{反}$=0.0000≠ $[-0]_{反}$=1.1111
		负数原码求反+1	负数每位求反
移码 $[X]_{移}=2^n+X\ \ (2^n>x\geq -2^n)$ 移码表示中零也是唯一的 真值的移码和补码仅差一个符号位.若将补码的符号位由 0 改为 1 或从 1 改为 0 即可得到真值的移码 乘法运算可用移码和加法来实现,两个 n 位数相乘,总共要进行 n 次加法运算和 n 次移位运算			

三种机器数的特点可以归纳为:

· 三种机器数的最高位均为符号位.符号位和数值位之间可用“.”(对于小数)或“,”(对于整数)隔开

· 当真值为正时,原码,补码和反码的表示形式均相同,即符号位用“0”表示,数值部分与真值部分相同

· 当真值为负时,原码,补码和反码的表示形式不同,其它符号位都用“1”表示,而数值部分有这样的关系,即补码是原码的“求反加 1”,反码是原码的“每位求反”.

2.定点数的运算

1)定点数的位移运算

左移,绝对值扩大;右移,绝对值缩小.

算术移位规则

符号位不变

	码制	添补代码
正数		0
负数	原	0
	补	右移添 0
		左移添 1
	反	1

算术移位和逻辑移位的区别:

算术移位:带符号数移位;

逻辑移位:无符号数移位;

2)原码定点数的加/减运算;

对原码表示的两个操作数进行加减运算时,计算机的实际操作是加还是减,不仅取决指令中的操作码,还取决于两个操作数的符号.而且运算结果的符号判断也较复杂.

例如,加法指令指示做(+A)+(- B)由于一操作数为负,实际操作是做减法(+A)-(+B),结果符号与绝对值大的符号相同.同理,在减法指令中指示做(+A)- (- B)实际操作做加法(+A)+(+B),结果与被减数符号相同.由于原码加减法比较繁琐,相应地需要由复杂的硬件逻辑才能实现,因此在计算机中很少被采用.

3)补码定点数的加/减运算;

⑴加法

整数 $[A]_{补}+[B]_{补}=[A+B]_{补}(\mathrm{mod}\ 2^{n+1})$

小数 $[A]_{补}+[B]_{补}=[A+B]_{补}(\mathrm{mod}\ 2)$

⑵减法

整数 $[A]_{补}-[B]_{补}=[A+(-B)]_{补}=[A]_{补}+[-B]_{补}(\mathrm{mod}\ 2^{n+1})$

小数 $[A]_{补}-[B]_{补}=[A+(-B)]_{补}=[A]_{补}+[-B]_{补}(\mathrm{mod}\ 2)$

无需符号判定,连同符号位一起相加,符号位产生的进位自然丢掉

4)定点数的乘/除运算

⑴一位乘法

<1>原码定点一位乘法

两个原码数相乘,其乘积的符号为相乘两数的异或值,数值两数绝对值之积.

设 $[X]_{原}=X0\ X1\ X2\cdots Xn$

$[Y]_{原}=Y0\ Y1\ Y2\cdots Yn$

$[X\cdot Y]_{原}=[X]_{原}\cdot[Y]_{原}=(X0\oplus Y0)\mid(X1\ X2\cdots Xn)\cdot(Y1\ Y2\cdots Yn)$

符号 | 表示把符号位和数值邻接起来.

原码两位乘和原码一位乘比较		
	原码一位乘	原码两位乘
符号位	$x_0\oplus y_0$	$x_0\oplus y_0$
操作数	绝对值	绝对值的补码
移位	逻辑右移	算术右移
移位次数	n	$\frac{n}{2}$（n为偶数）
最多加法次数	n	$\frac{n}{2}+1$（n为偶数）

<2>定点补码一位乘法

有的机器为方便加减法运算,数据以补码形式存放.乘法直接用补码进行,减少转换次数.具体规则如下:

$[X\cdot Y]_{补}=[X]_{补}(-Y0+0.Y1\ Y2\cdots Yn)$

⑶补码除法

笔算除法和机器除法的比较	
笔算除法	机器除法
商符单独处理	符号位异或形成
心算上商	$\lvert x\rvert-\lvert y\rvert>0$ 上商1 $\lvert x\rvert-\lvert y\rvert<0$ 上商0
余数 不动 低位补“0” 减右移一位 的除数	余数 左移一位 低位补“0” 减 除数

2 倍字长加法器	1 倍字长加法器
上商位置不固定	在寄存器 最末位上商

<1>定点原码一位除法

1>恢复余数法

被除数(余数)减去除数,如果为 0 或者为正值时,上商为 1,不恢复余数;如果结果为负,上商为 0,再将除数加到余数中,恢复余数.余数左移 1 位.

2>加减交替法

当余数为正时,商上 1,求下一位商的办法,余数左移一位,再减去除数;当余数为负时,商上 0,求下一位商的办法,余数左移一位,再加上除数.

<2>定点补码一位除法(加减交替法)

1〉如果被除数与除数同号,用被除数减去除数;若两数异号,被除数加上除数.如果所得余数与除数同号商上 1,否则,商上 0,该商为结果的符号位.

2〉求商的数值部分,如果上次商上 1,将除数左移一位后减去除数;如果上次商上 0,将余数左移一位后加除数,然后判断本次操作后的余数,如果余数与除数同号商上 1,如果余数与除数异号商上 0.如此重复执行 n-1 次(设数值部分 n 位).

3〉商的最后一位一般采用恒置 1 的办法,并省略了最低+1 的操作.此时最大的误差为 2^{-n}.

溢出概念和判别方法

当运算结果超出机器数所能表示的范围时,称为溢出.显然,两个异号数相加或两个同号数相减,其结果是不会溢出的.仅当两个同号数相加或者两个异号数相减时,才有可能发溢出的情况,一旦溢出,运算结果就不正确了,因此必须将溢出的情况检查出来.判别方法有三种:

1〉当符号相同的两数相加时,如果结果的符号与加数(或被加数)不相同,则为溢出.

2〉当任意符号两数相加时,如果 C=Cf,运算结果正确,其中 C 为数值最高位的进位,Cf 为符号位的进位.如果 C≠Cf ,则为溢出,所以溢出条件=C ⊕ Cf .

(三) 浮点数的表示和运算

1.浮点数的表示

【32 位 1 + 8 + 23

1--浮点数十进制 --> 除符号位之外转换成二进制

110.0110 --> 1.10 0110 * 2^2

2--按照格式填充

符号位：实际

阶码：指数+127 --> 二进制 2+127 = 129 -->二进制

尾数：10 0110 --> 机器码 ，011001+1 = 011010 000000....

1)浮点数的表示范围;

浮点数是指小数点位置可浮动的数据,通常以下式表示:

$$N=M\times R^{E}$$

其中,N 为浮点数,M(Mantissa)为尾数(可正可负),E(Exponent)为阶码(可正可负),R(Radix)称为"阶的基数(底)",而且 R 为一常数,一般为 2,8 或 16.在一台计算机中,所有数据的 R 都是相同的,于是不需要在每个数据中表示出来.因此,浮点数的机内表示一般采用以下形式:

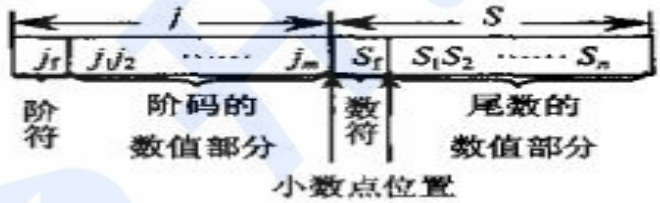
浮点数的机内表示一般采用以下形式:

Ms	E	M
1 位	n+1 位	m 位

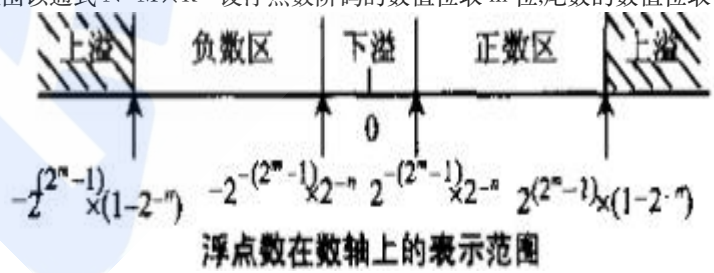
Ms 是尾数的符号位,设置在最高位上.

E 为阶码(移码),有 n+1 位,一般为整数,其中有一位符号位,设置在 E 的最高位上,用来表正阶或负阶.

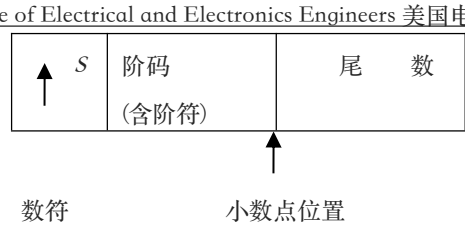
M 为尾数(原码),有 m 位,由 Ms 和 M 组成一个定点小数.Ms=0,表示正号,Ms=1,表示负.为了保证数据精度属数通常用规格化形式表示:当 R = 2,且尾数值不为 0 时,其绝对值大于或等于(0.5)10.对非规格化浮点数,通过将尾数左移或右移,并修改阶码值使之满足规格化要求.



浮点数的表示范围以通式 $N=M\times R^{E}$ 设浮点数阶码的数值位取 m 位,尾数的数值位取 n 位



2)IEEE754 标准(Institute of Electrical and Electronics Engineers 美国电气和电子工程协会)



根据 IEEE 754 国际标准,常用的浮点数有三种格式:

	符号位 *S*	阶码	尾数	总位数
短实数	1	8	23	32
长实数	1	11	52	64
临时实数	1	15	64	80

单精度格式 32 位,阶码为 8 位,尾数为 23 位.另有一位符号位 S,处在最高位.

由于 IEEE754 标准约定在小数点左部有一位隐含位,从而实际有效位数为 24 位.这样使得尾数的有效值变为 1.M .

例如,最小为 x1.0…0,,最大为 x1.1…1.规格化表示.故小数点左边的位横为 1,可省去.

阶码部分采用移码表示,移码值 127,1 到 254 经移码为-126 到+127.

S(1 位)	E(8 位)	M(23 位)	N(共 32 位)
符号位	0	0	0
符号位	0	不等于 0	(-1)S·2-126·(0.M) 为非规格化数
符号位	1 到 254 之间	-	(-1)S·2E-127·(1.M) 为规格化数
符号位	255	不等于 0	NaN(非数值)
符号位	255	0	无穷大

0 有了精确的表示,无穷大也明确表示.对于绝对值较小的数,可以采用非规格化数表示,减少下溢精度损失.非规格化数的隐含位是 0,不是 1.

2.浮点数的加/减运算

加减法执行下述五步完成运算:

1)“对阶”操作　比较两浮点数阶码的大小,求出其差ΔE,保留其大值 E,E=max(Ex, Ey).当ΔE≠0 时,将阶码小的尾数右移ΔE 位,并将其阶码加上ΔE,使两数的阶码值相等.

2)尾数加减运算　执行对阶之后,两尾数进行加减操作.

3)规格化操作　规格化的目的是使得尾数部分的绝对值尽可能以最大值的形式出现.

4)舍入　在执行右规或者对阶时,尾数的低位会被移掉,使数值的精度受到影响,常用“0”舍“1”入法.当移掉的部分最高位为 1 时,在尾数的末尾加 1,如果加 1 后又使得尾数溢　出,则要再进行一次右规.

5)检查阶码是否溢出　阶码溢出表示浮点数溢出.在规格化和舍入时都可能发生溢出,若阶码正常,加/减运算正常结束.若阶码下溢,则设置机器运算结果为机器零,若上溢,则设置溢出标志.

定点数和浮点数可从如下几个方面进行比较

①当浮点机和定点机中的位数相同时,浮点数的表示范围比定点数大得多

②当浮点数位规格化数时,其相对绝对远比定点数高

③浮点数运算要分阶码部分和尾数部分,而且运算结果都要求规格化,故浮点运算步骤比定点运算的步骤多,运算速度比定点运算的低,运算线路比定点运算的复杂

④在溢出的判断方法上,浮点数是对规格化的阶码进行判断,而定点数是对数值本身进行判断

总之,浮点数在数的表示范围,数的精度,溢出处理和程序编程方面(不取比例因子)均优于定点数.但在运算规则即硬件成本方面又不如定点数

(四) 算术逻辑单元 ALU

1.串行加法器和并行加法器

1)串行进位加法器

并行加法器可以同时对数据的各位进行相加,一般用 n 个全加器来实现 2 个操作数的各位同时向加.其操作数的各位是同时提供的,由于进位是逐位形成,低位运算所产生的进位会影响高位的运算结果.

串行进位(也称波形进位)加法器,逻辑电路比较简单,但是最高位的加法运算,一定要等到所有低位的加法完成之后才能进行,低位的进位要逐步的传递到高位,逐级产生进位,因此运算速度比较慢.

2)并行进位加法器

为了提高运算速度,减少延迟时间,可以采用并行进位法,也叫提前进位或先行进位.

全加器中,输入 Ai ,Bi,Ci-1,输出:

Si = Ai Bi Ci-1+Ai Bi Ci-1+Ai Bi Ci-1+Ai Bi Ci-1

Ci = Ai Bi Ci-1+Ai Bi Ci-1+Ai Bi Ci-1+Ai Bi Ci-1 = Ai Bi + (Ai+Bi)Ci-1

进位产生函数:Gi = Ai Bi

进位传递函数:Pi = Ai+Bi

Ci = Gi + Pi Ci-1

C4 = G4 + P4G3 + P4P3G2 + P4P3P2G1 + P4P3P2P1C0

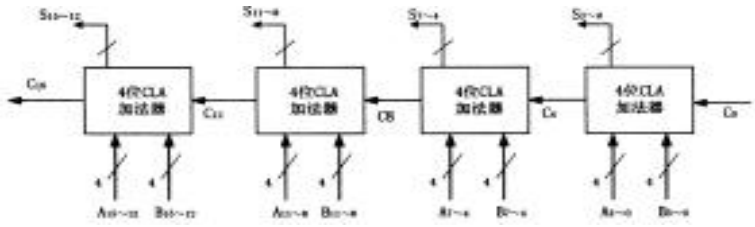
并行进位加法器的运算速度很快,形成最高进位输出的延迟时间很短,但是以增加硬件逻辑线路为代价.对于长字长的加法器,往往将加法器分成若干组,在组内采用并行进位,组间则采用串行进位或并行进位,由此形成多种进位结构.

⑴单级先行进位

单级先行进位方式将 n 位字长分为若干组,每组内采用并行进位方式,组与组之间册采用串行进位方式.

⑵多级先行进位

多级先行进位在组内和组间都采用先行进位方式.



16 位单级先行进位加法器

2.算术逻辑单元 ALU 的功能和机构

ALU 部件是运算器中的主要组成部分,又称多功能函数发生器,主要用于完成各种算术运算和逻辑运算.

ALU 的算术运算部件包含加法器,减法器,乘法器,除法器,增量器(+1),减量器(-1),BCD 码运算器等组件.

ALU 的主要工作是根据 CPU 指令要求执行各种指定运算,如加法,减法,乘法,除法,比较,逻辑移位等操作.

通用寄存器组是一组存取速度最快的存储器,用于保存参加运算的操作数和中间结果.访问寄存器无需高速缓存,也不需要运行总线周期,因此指令的执行速度很快.几乎所有的指令都要将寄存器指定为一个操作数,有些指令还要求将操作数存放在专用的寄存器中.

专用寄存器通常用于表示 CPU 所处于某种系统状态,ALU 中有两个重要的状态寄存器:指令指针寄存器 IP(即程序计数器 PC)和标志寄存器 FLAGS.

三、存储器层次结构

(一) 存储器基本概念

提高存储器带宽 ①缩短储存周期 ②增加存储字长,使每个周期可读/写更多的二进制数 ③增加存储体	内存 地址线 n,数据线数 k 芯片的容量为 $2^k \times k$ 位
20 位的地址可以访问 1MB 的存储空间,32 位的地址可以访问 4GB 的内存空间,64 位可以访问 1800 万 TB	

静态 RAM 和动态 RAM 之间的比较。目前,动态 RAM 的应用比静态 RAM 要广泛的多:

①同样大小的芯片中,动态的 RAM 的集成度远高于静态 RAM,DRAM 的基本单元电路为一个 MOS 管,SRAM 的基本单元电路可为 4~6 个 MOS 管

②DRAM 行、列按先后顺序输送,减少了芯片引脚,封装尺寸也减少

③DRAM 的功耗比 SRAM 小

④DRAM 的价格比 SRAM 的价格便宜

DRAM 也有缺点

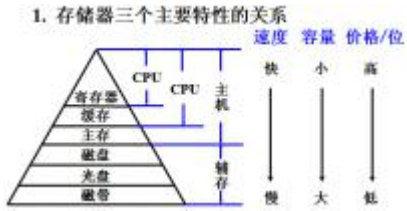
①由于使用动态元件(电容),因此它的速度比 SRAM 低

②DRAM 需再生,需配置再生电路,也消耗一部分功率.通常容量不大的 Cache 大多用 SRAM 实现存储器与 CPU 连接

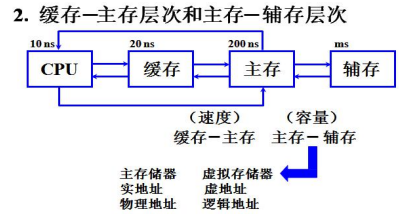
对比项目	SRAM	DRAM
储存信息	触发器	电容
破坏性读出	非	是
需要刷新	非	是
行列地址	同时送	分两次
运行速度	快	慢
集成度	低	高
发热量	大	小
存储成本	高	低

(二) 存储器的层次化结构

存储器有 3 个重要的指标:速度,容量和每位价格,一般来说,速度越快,位价越高;容量越大,位价越低,容量大,速度就越低.上述三者的关系用下图表示:



存储系统层次结构主要体现在缓存-主存-辅存这两个存储层次上,如下图所示:



缓存-主存层次主要解决 CPU 和主存速度不匹配的问题

主存-辅存层次主要解决存储系统的容量问题

从 CPU 角度来看缓存-主存层次的速度接近于缓存,高于主存; 其容量和价位却接近于主存,这就从速度和成本的矛盾中获得了理想的解决办法.

主存-辅存层次从整体分析,其速度接近于主存,容量接近于辅存,平均价位也接近于低速的、廉价的存储价位,这又解决了速度、容量、成本这三者之间的矛盾.

现代计算机系统几乎都具有这两个存储层次,构成了缓存、主存、辅存三级存储系统.

(三) 半导体随机存取存储器

1.SRAM 存储器的工作原理

SRAM 静态存储单元的每个存储位需要四到六个晶体管组成.比较典型的是六管存储单元,即一个存储单元存储一位信息"0"或"1".静态存储单元保存的信息比较稳定,信息为非破坏性读出,故不需要重写或者刷新操作;另一方面,其结构简单,可靠性高,速度较快,但其占用元件较多,占硅片面积大,且功耗大,所以集成度不高.

2.DRAM 存储器的工作原理

常见的 DRAM 存储单元有三管式和单管式两种,它们的共特点是靠电容存储电荷的原理来寄存信息.若电容上存有足够的电荷表示“”,电容上无电荷表示"0".电容上的电荷一般只能维持 1-2ms,因此即使电源不掉电,电容上的电荷会自动消失.因此,为保证信息的不丢失,必须在 2ms 之内就要对存储单元进行一次恢复操作,这个过程称为再生或者刷新,与 SRAM 相比,DRAM 具有集成度更高,功耗低等特点,目前被各类计算机广泛使用.

(四) 只读存储器

前面介绍的 DRAM 和 SRAM 均为可任意读 / 写的随机存储器,当掉电时,所存储的内容消失,所以是易失性存储器.只读存储器,即使停电,存储内容也不丢失.根据半导体制造工艺不同,分为 ROM,PROM,EPROM,E2ROM 和 Flash Memory

1.只读存储器(ROM)

掩模式 ROM 由芯片制造商在制造时写入内容,以后只能读而不能再写入.其基本存储原理是以元件的“有 / 无”来表示该存储单元的信息(“1”或“0”),可以用二极管或晶体管作为元件,显而易见,其存储内容是不会改变的.

2.可编程序的只读存储器(PROM)

PROM 可由用户根据自己的需要来确定 ROM 中的内容,常见的熔丝式 PROM 是以熔丝的通和断开来表示所存的信息为“1”或“0”.刚出厂的产品,其熔丝是全部接通的.根据需要断开某些单元的熔丝(写入).显而易见,断开后的熔丝是不能再接通了,因而一次性写入的存储器.掉电后不会影响其所存储的内容.

3.可擦可编程序的只读存储器(EPROM)

为了能修改 ROM 中的内容,出现了 EPROM.利用浮动栅 MOS 电路保存信息,信息改写用紫外线照射即可擦除.

4.可电擦可编程序只读存储器(E2PROM)

E2PROM 的编程序原理与 EPROM 相同,擦除原理完全不同,重复改写次数有限制(因氧化层被磨损),一般 10 万次.

其读写操作可按每个位或每个字节进行,类似 SRAM,但每字节的写入周期要几毫秒,比 SRAM 长得多.E2PROM 每个存储单元采则 2 个晶体管.其栅极氧化层比 EPROM 薄,因此具有电擦除功能.

5.快除读写存储器(Flash Memory) F1ash Memory 是在 EPROM 与 E2PROM 基础上发展起来的,其读写过程和 E2PROM 不同,F1ash Memory 的读写操作一般是以块为单位.

(五) 主存储器与 CPU 的连接

1 个存储器的芯片的容量是有限的,它在字数或字长方面与实际存储器的要求都有很大差距,所以需要在字向和位向进行扩充才能满足需要.根据存储器所需的存储容量和所提供的芯片的实际容量,可以计算出总的芯片数.一个存储器的容量为 M×N 位,若使用 L×K 位存储器芯片,那么,这个存储器共需要 M/L×N/K 存储器芯片.

1.位扩展

位扩展指的是用多个存储器器件对字长进行扩充.位扩展的连接方式是将多片存储器的地址,片选己,读写控制端 R/W 可相应并联,数据端分别引出.

2)字扩展

字扩展指的是增加存储器中字的数量.

静态存储器进行字扩展时,将各芯片的地址线,数据线,读写控制线相应并联,而由片选信号来区分各芯片的地址范围.

3)字位扩展

实际存储器往往需要字向和位向同时扩充.

(六) 双口 RAM 和多模块存储器

1.双端口存储器

双端口存储器是一种具有两个单独的读/写端口及控制电路的存储器,通过增加一个读/写端口,双端口存储器扩展了存储器的的信息交换能力.

2.多模块存储器

为了解决 CPU 与主存储器之间的速度匹配问题,在高速存储器中,普遍采用并行主存系统.即利用类似存储器扩展(位扩展,字扩展,字位扩展)的方法,将 n 个字长为 W 位的存储器并行连接,构建一个更大的存储器.并行主存有单体多字方式,多体并行方式和多体交叉方式.

(七) 高速缓冲存储器

1.问题的提出

避免 CPU“空等”现象

CPU 和主存(DRAM)的速度差异

程序访问的局部性原理

1.程序访问的局部性

从大量的统计中得到的一个规律是,程序中对于存储空间 90%的访问局限于存储空间的 10%的区域中,而另外 10%的访问则分布在存储空间的其余 90%的区域中.这就是通常说的局部性原理.访存的局部性规律包括两个方面:

时间局部性:如果一个存储项被访问,则可能该项会很快被再次访问.

空间局部性:如果一个存储项被访问,则该项及其邻近的项也可能很快被访问.

2.Cache 的基本工作原理

Cache 通常由两部分组成,块表和快速存储器.其工作原理是:处理机按主存地址访问存储器,存储器地址的高段通过主存-Cache 地址映象机构借助查表判定该地址的存储单元是否在 Cache 中,如果在,则 Cache 命中,按 Cache 地址访问 Cache.否则,Cache 不命中,则需要访问主存,并从主存中调入相应数据块到 Cache 中,若 Cache 中已写满,则要按某种算法将 Cache 中的某一块替换出去,并修改有关的地址映象关系.

从这个工作原理我们可以看出,它已经涉及到了两个问题.首先是定位,然后是替换的问题.

Cache 的存在对程序员是透明的.其地址变换和数据块的替换算法均由硬件实现.通常 Cache 被集成到 CPU 内以提高访问速度.

3.Cache 和主存之间的映射方式

因为处理机访问都是按主存地址访问的,而 Cache 的空间远小于主存,如何知道这一次的访问内容是不是在 Cache 中,在 Cache 中的哪一个位置呢? 这就需要地址映象,即把主存中的地址映射成 Cache 中的地址.让 Cache 中一个存储块(空间)与主存中若干块相对应,如此,访问一个主存地址时,就可以对应地知道在 cache 中哪一个地址了.地址映象的方法有三种:直接映象,全相联映象和组相联映象.

直接映象就是将主存地址映象到 Cache 中的一个指定地址.任何时候,主存中存储单元的数据只能调入到 Cache 中的一个位置,这是固定的,若这个位置已有数据,则产生冲突,原来的块将无条件地被替换出去.

全相联映象就是任何主存地址可映象到任何 Cache 地址的方式.在这种方式下,主存中存储单元的数据可调入到 Cache 中的任意位置.只有在 Cache 中的块全部装满后才会出现块冲突.

组相联映象指的是将存储空间的页面分成若干组,各组之间的直接映象,而组内各块之间则是全相联映象.

4.Cache 中主存块的替换算法

在直接映象方式下,不存在块替换的算法,因为每一块的位置映象是固定的,需要哪一块数据就可直接确定地将该块数据调入上层确定位置.而其他两种映象就存在替换策略的问题,就是要选择替换到哪一个 Cache 块.即替换算法.

	思想	优点	缺点
随机算法 RAND	用软的或硬的随机数产生器产生上层中要被替换的页号	简单,易于实现	没有利用上层存储器使用的"历史信息",没有反映等程序局部性,命中率低.
先进先出 FIFO	选择最早装入上层的页作为被替换的页	实现方便,利用了主存历史的信息	不能正确反映程序局部性原理,命中率不高,可能出现一种异常现象.
近期最少使用 LRU	选择近期最少访问的页作为被替换的页	比较正确反映程序局部性,利用访存的历史信息,命中率较高	实现较复杂
优化替换算法 OPT	将未来近期不用的页换出去	命中率最高,可作为衡量其他替换算法的标准	不现实,只是一种理想算法

5.Cache 写策略

对 Cache 的写操作,情况比读操作要复杂一些.由于写入 Cache 时,并没有写入主存,因此就出现 Cache 和主存数据不一致的情况.如何处理 Cache 和主存不一致的方法就称为更新策略.

更新策略	思想	优点	缺点
写回法	是指在 CPU 执行写操作时,信息只写入 Cache 中,仅当需要替换时,才将改写过的 Cache 块先送回主存(写回),然后再调块(设置 dirty 位)	有利于省去许多将中间结果写入主存的无谓开销.	需设修改位增加 Cache 的复杂性

全写法(写直达法)	在写操作时,将数据同时写入 Cache 和主存	实现开销小,简单	为了写中间结果浪费了不少时间

另外,当写不命中时(也就是写 Cache 块时,这块早被人替换出去而在 Cache 中找不到时)是不是要把这块再取回 Cache 中,有两个解决方法:

◆不按写分配法,就是直接写到主存里,不再把该地址对应的块调回 Cache 中.

◆按写分配法,就是写到主存,而且把这一块从主存中调入到 Cache.

一般写回法用按写分配法,全写法则采用不按写分配.

(八) 虚拟存储器

1.虚拟存储器的基本概念

虚拟存储器是主存的扩展,虚拟存储器的空间大小取决于计算机的访存能力而不是实际外存的大小,实际存储空间可以小于虚拟地址空间.从程序员的角度看,外存被看作逻辑存储空间,访问的地址是一个逻辑地址(虚地址),虚拟存储器使存储系统既具有相当于外存的容量又有接近于主存的访问速度.

虚拟存储器的访问也涉及到虚地址与实地址的映象,替换算法等,这与 Cache 中的类似,前面我们讲的地址映象以块为单位,而在虚拟存储器中,地址映象以页为单位.设计虚拟存储系统需考虑的指标是主存空间利用率和主存的命中率.

虚拟存储器与 Cache 存储器的管理方法有许多相同之处,它们都需要地址映象表和地址变换机构.但是二者也是不同的.

虚拟存储器的三种不同管理方式:按存储映象算法,分为段式,页式和段页式等,这些管理方式的基本原理是类似的.

2.页式虚拟存储器

页式管理:是把虚拟存储空间和实际空间等分成固定大小的页,各虚拟页可装入主存中的不同实际页面位置.页式存储中,处理机逻辑地址由虚页号和页内地址两部分组成,实际地址也分为页号和页内地址两部分,由地址映象机构将虚页号转换成主存的实际页号.

页式管理用一个页表,包括页号,每页在主存中起始位置,装入位等.页表是虚拟页号与物理页号的映射表.页式管理由操作系统进行,对应用程序员的透明的.

3.段式虚拟存储器

段式管理: 把主存按段分配的存储管理方式.它是一种模块化的存储管理方式,每个用户程序模块可分到一个段,该程序模块只能访问分配给该模块的段所对应的主存空间.段长可以任意设定,并可放大和缩小.

系统中通过一个段表指明各段在主存中的位置.段表中包括段名(段号),段起点,装入位和段长等.段表本身也是一个段.段一般是按程序模块分的.

4.段页式虚拟存储器

段页式管理:是上述两种方法的结合,它将存储空间按逻辑模块分成段,每段又分成若干个页,访存通过一个段表和若干个页表进行.段的长度必须是页长的整数倍,段的起点必须是某一页的起点.

5.TLB(快表)

在虚拟存储器中进行地址变换时,需要虚页号变换成主存中实页号的内部地址变换,这一般通过查内页表实现.当表中该页对应的装入位为真时,表示该页在主存中,可按主存地址问主存;如果装入位为假时,表示该页不在存储器中,就产生页失效中断,需从外存调入页.

中断处理时先通过外部地址变换,一般通过查外页表,将虚地址变换为外存中的实际地址,到外存中去选页,然后通过 I/0 通道调入内存.当外存页面调入主存中时还存在一个页面替换略的问题.

提高页表的访问速度是提高地址变换速度的关键.因为,每次访存都要读页表,如果页存放在主存中,就意味着访存时间至少是两次访问主存的时间,这样查表的代价大大.只有内部地址变换速度提高到使访问主存的速度接近于不采用虚拟存储器时的访主存速度时,虚拟存储器才能实用.

根据访存的局部性,表内各项的使用的概率不是均匀分布的.在一段时间内,可能只用表中的很少几项,因此应重点提高使用概率高的这部分页表的访问速度,可用快速硬件构成全表小得多的部分表格,而将整个表格放在主存中,这就引出了快表和慢表的概念和技术.

查表时,根据虚页表同时查找快表和慢表,当在快表中查到该虚页号时,就能很快找到对应的实页号,将其送入主存实地址寄存器,同时使慢表的查找作废,这时主存的访问速度没降低多少.

如果在快表中查不到,则经过一个访主存的时间延迟后,将从慢表中查到的实页送入实地址寄存器,同时将此虚页号和对应的实页号送入快表,这里也涉及到用一个替换算法从快表中替换出一行.

快表的存在对所有的程序员都是透明的.

软磁盘存储器		
	硬盘	软盘
速度	高	低
磁头	固定、活动、浮动	活动、接触盘片
盘片	固定盘、盘组大部分不可换	可换盘片
价格	高	低
环境	苛刻	

四、指令系统

(一) 指令格式

1. 指令的基本格式

操作码字段	地址码字段

操作码　反映机器做什么操作

(1) 长度固定　用于指令字长较长的情况，RISC　如 IBM 370　操作码 8 位

(2) 长度可变　操作码分散在指令字的不同字段中

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,一条指令必须包含下列信息:

⑴操作码,具体说明了操作的性质及功能.

⑵操作数的地址.

⑶操作结果的存储地址.

⑷下一条指令的地址.

从上述分析可知,一条指令实际上包括两种信息即操作码和地址码.

操作码(operation code)用来表示该指令所要完成的操作(如加,减,乘,除,数据传送等),其长度取决于指令系统中的指令条数.如操作码占 7 位,则该机器最多包含 2^7=128 条指令.

地址码用来描述该指令的操作对象,或直接给出操作数或指出操作数的存储器地址或寄存器地址(即寄存器名).

操作码的长度不固定会增加指令译码和分析难度,使控制器的设计复杂.

操作码	寻址地址	形式地址 A

形式地址　指令字中的地址

有效地址　操作数的真实地址

约定　指令字长=存储字长=机器字长

2.定长操作码指令格式

1)零地址指令

格式:

OP

OP——　操　作　码　　【JUM】

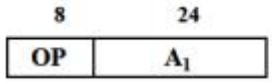
指令中只有操作码,而没有操作数或没有操作数地址.这种指令有两种可能:

⑴无需任何操作数,如空操作指令,停机指令等.

⑵所需的操作数是默认的.如堆栈结构计算机的运算指令,所需的操作数默认在堆栈中,由堆栈指针 SP 隐含指出,操作结果仍然放回堆栈中.又如 Intel 8086 的字符串处理指令,源,目的操作数分别默认在源变址寄存器 SI 和目的变址寄存器 DI 所指定的存储器单元中.

2)一地址指令

格式:



2 次访存

(ACC) OP (A_1) ——→ACC　　寻址范围 $2^{24}=16$ M

OP——操作码

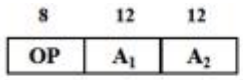
A——操作数的存储器地址或寄存器名

指令中只给出一个地址,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,对这一地址所指定的操作数执行相应的操作后,产生的结果又存回该地址中.

在某些字长较短的微型机中(如早期的 Z80,Intel8080,MC6800 等),大多数算术逻辑指令也采用这种格式,第一个源操作数由地址码 A 给出,第二个源操作数在一个默认的寄存器中,运算结果仍送回到这个寄存器中,替换了原寄存器内容,通常把这个寄存器称累加器.

3)二地址指令

格式:



(A_1) OP (A_2) ——→A_1　　4 次访存

或

(A_1) OP (A_2) ——→A_2　　寻址范围 $2^{12}=4$ K

若结果存于 ACC　3次访存　若ACC 代替 A_1（或A_2）

OP——操作码

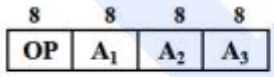
A_1——第一个源操作数的存储器地址或寄存器地址.

A_2——第二个源操作数和存放操作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,在本例中即为 A2 指定的地址

4)三地址指令

格式:



4 次访存

寻址范围 $2^8=256$

(A_1) OP (A_2) ——→A_3　　若 A_3 用 A_1 或 A_2 代替

OP——操作码

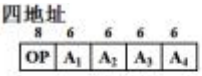
A_1——第一个源操作数的存储器地址或寄存器地址

A_2——第二个源操作数的存储器地址或寄存器地址

A_3——操作结果的存储器地址或寄存器地址

其操作是对 A1,A2 指出的两个源操作数进行操作码(OP)所指定的操作,结果存入 A3 中.

5) 四地址



设指令字长为32位

操作码固定为8位

4次访存

寻址范围 $2^6=64$

若PC代替A_4

A_1第一操作数地址

A_2第二操作数地址

A_3结果的地址

A_4下一条指令地址

(A_1) OP (A_2) ——A_3

6)多地址指令

在某些性能较好的大,中型机甚至高档小型机中,往往设置一些功能很强的,用于处理成批数据的指令,如字符串处理指令,向量,矩阵运算指令等.

为了描述一批数据,指令中需要多个地址来指出数据存放的首地址,长度和下标等信息

3.扩展操作码指令格式

设某机器的指令长度为16位,包括4位基本操作码字段和三个4位地址字段,其格式下:

OP(4)	A_1(4)	A_2(4)	A_3(4)

4位基本操作码有16个码点(即有16种组合),若全部用于表示三地址指令,则只有16条.但,若三地址指令仅需15条,两地址指令需15条,一地址指令需15条,零地址指令需16条,共61条指令,应如何安排操作码?

显然,只有4位基本操作码是不够的,必须将操作码的长度向地址码字段扩展才行.

一种可供扩展的方法和步骤如下:

⑴15条三地址指令的操作码由4位基本操作码从0000~1110给出,剩下一个码点1111用于把操作码扩展到A1,即4位扩展到8位;

⑵15条二地址指令的操作码由8位操作码从11110000~11111110给出,剩下一个码点11111111用于把操作码扩展到A2,即从8位扩展到12位;

⑶15条一地址指令的操作码由12位操作码从111111110000~111111111110给出,剩下的一个码点111111111111用于把操作码扩展到A3,即从12位扩展到16位;

⑷16条零地址指令的操作码由16位操作码从1111111111110000~1111111111111111给出.

指令字长取决于操作码的长度、操作数地址的长度和操作数地址的个数.为了提高指令的运行速度和节省存储空间,通常尽可能的吧常用的指令(如数据传输指令、算逻运算指令等)设计成单字长或短字长格式的指令.

操作数类型	
地址	地址实际也可以看做是一种数据,在许多情况下要计算操作数的地址,这时地址可看作无符号的整数
数字	计算机中常见的数字有定点数、浮点数和十进制数字
字符	在应用计算机时,文本或者字符串也是一种常见的数据类型
逻辑数据	计算机除了做算术运算外,有时还做逻辑运算,此时n个0和1的组合不是被看做算术数字而被看做逻辑数
奔腾Pentium处理器的数据类型有逻辑数、有符号数(补码)、无符号数、压缩和未压缩的BCD码、地址指针、位串以及浮点数(符合IEEE754标准)等	

指令操作类型		
1.数据传送	数据传送包括寄存器与寄存器,寄存器与存储单元,存储单元与存储单元之间的传送	
2.算术逻辑操作	这操作可实现算术运算(加,减,乘,除,增 1,减 1,取负即求补)逻辑运算(与,或,非,异或)	
3.移位	移位可分为算术移位,逻辑移位和循环移位三种	
4.转移	无条件转移	不受任何约束条件直接把程序转移到下一条需执行指令的地址
	条件转移	根据当前指令的执行结果决定是否需要转移
	调用与返回	●子程序可在多处被调用 ●子程序调用可出现在子程序中,即允许子程序嵌套 ●每个 CALL 指令都对应一条 RETURN 指令
		CPU 必须记住返回地址,使子程序能准确返回,返回地址存放在以下 3 处 ●寄存器内.机器内设有专用寄存器,专用于存放返回地址 ●子程序的入口地址内 ●栈顶内.现代计算机都设有堆栈,执行 RETURN 指令后,便可自动从堆栈内取出应返回的地址
	陷 阱 (Trap) 与陷阱指令	其实是一种意外事故的中断,一般不提供给用户使用,作为隐指令,再出现故障时,由 CPU 自动产生并执行
5.输入输出	对于 I/O 单独编址的计算机而言,通常设有输入输出指令,他完成从外设中的寄存器读入一个数据到 CPU 寄存器内,或将数据从 CPU 的寄存器输出至某外设的寄存器中	
6.其它	包括等待指令、停机指令、空操作指令、开中断指令、关中断指令、置条件码指令等	
备注	有些大型或巨型机还设有向量指令,可对整个向量或矩阵进行求和求积运算	

(二) 指令的寻址方式

1.有效地址的概念

操作数的真实地址称为有效地址,记做 EA,它是寻址方式和形式地址共同来决定的.

2.数据寻址和指令寻址

寻址方式是指确定本条指令的数据地址以及下一条将要执行的指令的地址,与硬件结构密切相关,寻址方式分为指令寻址和数据寻址两大类

指令寻址分为顺序寻址和跳跃寻址两种.

顺序寻址可以通过程序计数器 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,是通过对 PC 的运算得到新的下一条指令的地址.

3.常见寻址方式

设计指令格式应考虑的各种因素

指令系统集中反映了机器的性能,又是程序员编程的依据,高档机必须能兼容低档机的程序运行,称之为“向上兼容”.

指令格式集中体现了指令系统的功能.为此,在确定指令系统时,必须从以下几个方面综合考虑.

①操作类型：包括指令数及操作的难易程度

②数据类型：确定哪些数据类型可以参加操作

③指令格式：包括指令字长、操作码位数、地址码位数、地址个数、寻址方式类型、以及指令字长和操作码位数是否可变等.

④寻址方式：包括指令和操作数具体有哪些寻址方式.

⑤寄存器个数：寄存器的多少直接影响指令的执行时间.

<table>
<tr><td>寻址方式</td><td colspan="3">详情</td></tr>
<tr><td rowspan="2">指令寻址</td><td>顺序寻址</td><td colspan="2">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</td></tr>
<tr><td>跳跃寻址</td><td colspan="2">跳跃寻址则通过转移类指令实现</td></tr>
<tr><td rowspan="9">数据寻址</td><td rowspan="2">1.立即寻址</td><td colspan="2">操作数本身设在指令字内,即形式地址 A 不是操作数地址而是操作数本身</td></tr>
<tr><td colspan="2">●指令执行阶段不访存
●A 的位数限制了这类指令所能表述的立即数的范围</td></tr>
<tr><td rowspan="2">2.直接寻址</td><td colspan="2">指令中的形式地址 A 就是操作数的真实地址 EA,即 EA=A</td></tr>
<tr><td colspan="2">●执行阶段访问一次存储器
●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</td></tr>
<tr><td rowspan="2">3.隐含寻址</td><td colspan="2">指令字中不明显给出操作数的地址,其操作数的地址隐含在操作码或某个寄存器中</td></tr>
<tr><td colspan="2">●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</td></tr>
<tr><td rowspan="2">4.间接寻址</td><td colspan="2">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的地址,也就是说,有效地址是由形式地址间接提供的,即为间接地址,即 EA=(A)</td></tr>
<tr><td>优点</td><td>1.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
2.它便于编制程序</td></tr>
</table>

		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
	5.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的编号,即 EA=R	
		●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,因此寄存器寻址在计算机中得到广泛应用 ●执行阶段不访存,只访问寄存器,执行速度快 ●寄存器个数有限,可缩短指令字长	
	6.寄存器间接寻址	有效地址 EA+=(R_i),因有效地址	
		●有效地址在寄存器中, 操作数在存储器中,执行阶段访存 ●便于编制循环程序	
	7.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)	
		采用专用寄存器作基址寄存器	●可扩大寻址范围 ●有利于多道程序 ●BR 内容由操作系统或管理程序确定 ●在程序的执行过程中 BR 内容不变,形式地址 A 可变
		采用通用寄存器作基址寄存器	●由用户指定哪个通用寄存器作为基址寄存器 ●基址寄存器的内容由操作系统确定 ●在程序的执行过程中 R_0 内容不变,形式地址 A 可变
	8.变址寻址	变址寻址与基址寻址极为相似.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 EA=A+(IX)	
		●可扩大寻址范围 ●IX 的内容由用户给定 ●在程序的执行过程中 IX 内容可变,形式地址 A 不变 ●便于处理数组问题	
	9.相对寻址	~ 的有效地址是将 PC 的内容(即当前指令地址)与指令字中的形式地址 A(A 是相对于当前指令的位移量(可正可负,补码)相加而成,即 EA=(PC)+A	
		●A 的位数决定操作数的寻址范围 ●程序浮动 ●广泛应用于转移指令	

	10.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) ●硬堆栈 多个寄存器 ●软堆栈 制定的存储空间

(三) RISC (精简指令集) 的特点

1)优先选取使用频率最高的一些简单指令;	选用使用频度较高的一些 简单指令,复杂指令的功能由简单指令来组合
2)指令长度固定;	指令长度固定、指令格式种类少、寻址方式少
3)只有取数/存数指令(load/store)访问内存;	只有 LOAD / STORE 指令访存
4)CPU 中的寄存器数量很多;	CPU 中有多个通用寄存器
5)大部分指令在一个或小于一个机器周期完成;	采用流水技术一个时钟周期内完成一条指令
6)硬布线控制逻辑为主,不用或少用微码控制;	采用 组合逻辑实现控制器
7)一般用高级语言编程,特别重视编译优化,以减少程序执行时间.【eg: 编译器把相同的指令放一起】	采用优化的编译程序

五、中央处理器(CPU)

(一) CPU 的功能和基本结构

CPU 主要是由运算器和控制器组成,由于运算器(实现算术运算和逻辑运算)部分在第二部分介绍过,所以本节主要介绍控制器的组成和工作原理.

1. 控制器的功能

计算机对信息进行处理(或计算)是通过程序的执行而实现的,程序是完成某个确定算法的指令序列,要预先存放在存储器中.控制器的作用是控制程序的执行,它必须具有以下基本功能:

1)取指令

2)分析指令

3)执行指令

计算机不断重复顺序执行上述三种基本操作:取指,分析,执行;再取指,再分析,再执行,如此循环,直到遇到停机指令或外来的干预为止.

4)控制程序和数据的输入与结果输出

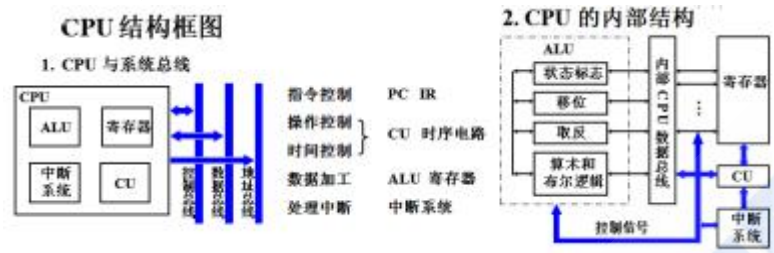
根据程序的安排或人的干预,在适当的时候向输入输出设备发出一些相应的命令来完成 I/O 功能,这实际上也是通过执行程序来完成的.

5)对异常情况和某些请求的处理

当机器出现某些异常情况,诸如算术运算的溢出和数据传送的奇偶错等;或者某些外来请求,诸如磁盘上的成批数据需送存储器或程序员从键盘送入命令等,此时由这些部件或设备发出:

⑴“中断请求”信号.

⑵DMA 请求信号.



2．控制器的组成

根据对控制器功能分析,得出控制器的基本组成如下:

1)程序计数器(PC)

即指令地址寄存器.在某些计算机中用来存放当前正在执行的指令地址;而在另一些计算机中则用来存放即将要执行的下一条指令地址;而在有指令预取功能的计算机中,一般还需要增加一个程序计数器用来存放下一条要取出的指令地址.

有两种途径来形成指令地址,其一是顺序执行的情况,通过程序计数器加“1”形成下一条指令地址(如存储器按字节编址,而指令长度为 4 个字节,则加“4”).其二是遇到需要改变顺序执行程序的情况,一般由转移类指令形成转移地址送往程序计数器,作为下一条指令的地址.

2)指令寄存器(IR)

用以存放当前正在执行的指令,以便在指令执行过程中,控制完成一条指令的全部功能.

3)指令译码器或操作码译码器

对指令寄存器中的操作码进行分析解释,产生相应的控制信号.

在执行指令过程中,需要形成有一定时序关系的操作控制信号序列,为此还需要下述组成部分.

4)脉冲源及启停线路

脉冲源产生一定频率的脉冲信号作为整个机器的时钟脉冲,是机器周期和工作脉冲的基准信号,在机器刚加电时,还应产生一个总清信号(reset).启停线路保证可靠地送出或封锁时钟脉冲,控制时序信号的发生或停止,从而启动机器工作或使之停机.

5)时序控制信号形成部件

当机器启动后,在 CLK 时钟作用下,根据当前正在执行的指令的需要,产生相应的时序控制信号,并根据被控功能部件的反馈信号调整时序控制信号.例如,当执行加法指令时,若产生运算溢出的异常情况,一般不再执行将结果送入目的寄存器(或存储单元)的操作,而发出中断请求信号,转入中断处理;又如执行条件转移指令时,根据不同的条件产生不同的控制信号,从而进入适当的程序分支.

(二) 指令执行过程

1．指令执行的时序

计算机工作的过程是取指令,分析指令,执行指令三个基本动作的重复.考虑到所有的器件中(寄存器,存储器)存储器的速度最慢,因此,取最慢的器件工作时间(周期)作为整个工作的最长同步标准.

计算机的工作时序是按照存储器的工作周期划分的.每个存储器工作周期又称为机器周期.因此,每个机器周期至少完成一个基本操作.一般最长的操作是访问存储器(读/写),这个时间也用于访问外设接口(寄存器).如果,某个操作,比如利用运算器执行一次运算,如果不访问存储器,即使占用的时间很短,但是,也必须为其划分一个机器周期.因此,机器周期是计算时序划分的最大单位.

现在我们为计算机的执行时间进行最基本的划分:由于计算机不断地重复执行每个指令,所以,我们将执行的时间划分为一条一条指令执行所占用的时间,如下:

执行指令 1	执行指令 2	执行指令 3	执行指令 4	执行指令 5

我们将每指令占用的时间称为指令周期.由于每条指令的功能不一样,因此执行的时间也不同,指令周期长短不一样.

而每条指令的执行,又可以是取指令,分析指令,执行指令.由于取指令必须访问存储器,所以占用一个机器周期.分析指令是由指令译码电路完成的,所占用的时间极短,无需分配一个完整的机器周期.一般是在取指周期后期(结束之前的很短时间内)就可以完成.指令的执行较为复杂:可能不访问存储器;访问一次存储器;访问两次存储器等.因此,可能是一个机器周期到几个机器周期.

因此,每条指令的执行过程如下:

取指周期	执行周期 1	执行周期 2	执行周期 3	执行周期 4

第一个机器周期总是取指周期,而指令的地址总是从 PC 中获得,当发出读取存储器命令后,指令总是从数据总线 DB 送回,CPU 接受到指令之后,将指令放在指令寄存器 IR 之中.指令在 IR 中一直保留到取下一条指令为止.

第二个机器周期开始,根据指令有所不同:

执行一次 ALU 运算:分配一个机器周期.

执行访问一次存储器:分配一个机器周期.

所以,根据指令执行的不同情况,将会得到不同指令执行所占用的机器周期.

根据每个机器周期完成的任务不同,我们将每个机器周期按照任务命名.如同用取指周期命名第一个机器周期一样.

2．指令执行过程举例

假设指令格式如下:

操作码	rs,rd	rs1	imm(Disp)

rs,rd,rsl 为通用寄存器地址;imm(或 disp)为立即数(或位移量).

加法指令功能:将寄存器(rs)中的一个数与存储器中的一个数(其地址为(rsl) + disp)相加,结果放在寄存器 rd 中,rs 与 rd 为同一寄存器.

加法指令完成以下操作:

①取指周期

从存储器取指令,送入指令寄存器,并进行操作码译码(分析指令).

程序计数器加 1,为下一条指令作好准备.

控制器发出的控制信号:PC→AB,W/R=0,M/IO=1;DB→IR;PC + 1.

②计算地址周期

计算数据地址,将计算得到的有效地址送地址寄存器 AR.

控制器发出的控制信号:rsl→GR,(rsl)→ALU,disp→ALU(将 rsl 的内容与 disp 送 ALU);“+”(加法命令送 ALU);ALU→AR(有效地址送地址寄存器).

③取数周期

到存储器取数.

控制器发出的控制信号:AR→AB,W/R=0,M/IO=1;DB→DR(将地址寄存器内容送地址总线,同时发访存读命令,存储器读出数据送数据总线后,打入数据寄存器).

④执行周期

进行加法运算,结果送寄存器,并根据运算结果置状态位 N,Z,V,C.

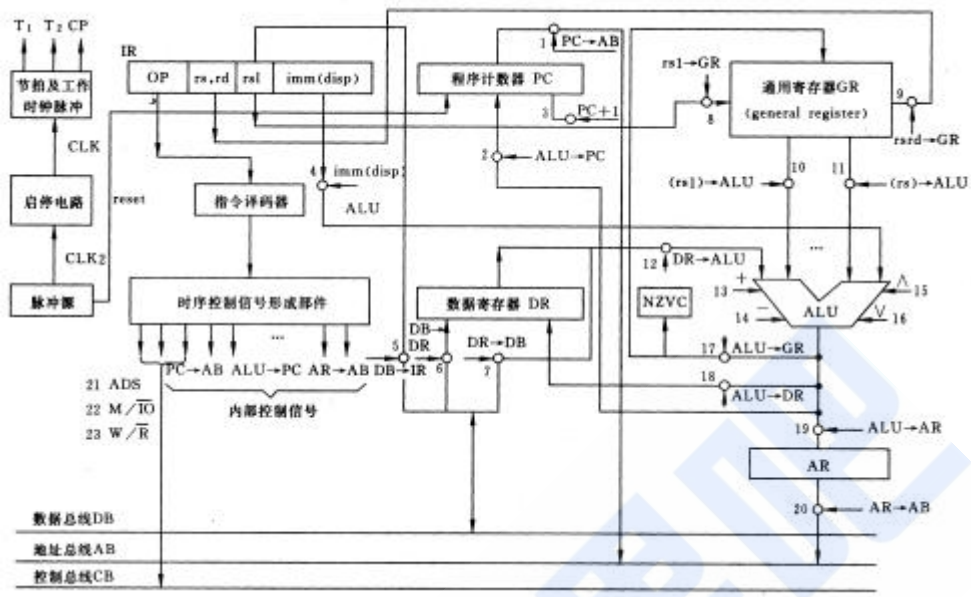
控制器送出的控制信号:rs,rd→GR, (rs)→ALU,DR→ALU(两个源操作数送 ALU);

ALU→rd(运算结果送寄存器 rd)

(三)数据通路的功能和基本结构

CPU 的数据通路是连接 CPU 内部各个部件以及和 CPU 外部个部件之间的数据和控制信号的连接关系图.

数据通路的基本结构:



(四) 控制器的功能和工作原理

1.硬布线控制器

控制器控制信号的产生是采用逻辑电路,也称组合逻辑电路控制方式.“时序控制信号形成部件”是由硬逻辑布线完成的,实际设计中,需要几十~几百条指令,确定每条指令所需的机器周期,将情况相同的指令归并在一起.

⑴时序与节拍

每一步由一个机器周期来完成,假设采用 4 个机器周期,总之,需要 4 个不同的信号输出,代表 4 个不同的周期.

⑵操作码译码器

指令的操作码部分指出本指令将执行什么指令,如加法,减法等.对于不同的指令,采用不同的代码表示.

⑶操作控制信号的产生

以加法指令为例,加法指令的完成是由 4 个机器周期 cy1,cy2,cy3,cy4 组成,分别是取指,计算地址,取数,计算 4 个机器周期.

将所有的机器周期的操作控制信号的逻辑表达式全部写出来,就会得到各个操作控制信号的所有表达式,再将这些表达式安每个操作控制信号组合起来,就得到某个操作控制信号的表达式.

取指周期需要产生的操作控制信号如下:

PC→AB=cy1　　;将 PC 送地址总线

ADS=cy1 · T1　　;存储器地质有效

M/IO=cy1　　;存储器操作

W/R=cy1　　;读操作

DB→IR=cy1　　;将读出的结果送 IR

PC+1=cy1　　;将程序计数器加 1

计算地址周期 cy2 需要完成有效地址((rs1)+Disp)的计算.产生的操作控制信号如下:

rs1→GR=加法指令 · cy2　　;送通用寄存器地址

(rs1)→ALU=加法指令 · cy2　　;通用寄存器送 ALU

Disp→ALU=加法指令 · cy2　　;偏移量送 ALU

“+”=加法指令 · cy2　　;ALU 执行加法操作

ALU→AR=加法指令 · cy2　　;运算结果送地址总线

例如,“+”操作控制信号在加法指令的 cy2(计算有效地址)和 cy4(操作数相加)时需要;减法指令的 cy2(计算有效地址)时需要;转移指令的 cy2(计算有效地址)时需要;….

所以,“+”操作控制信号的逻辑表达式如下:

“+”=加法指令 · (cy2+cy4)+减法指令 · cy2+转移指令 · cy2+…

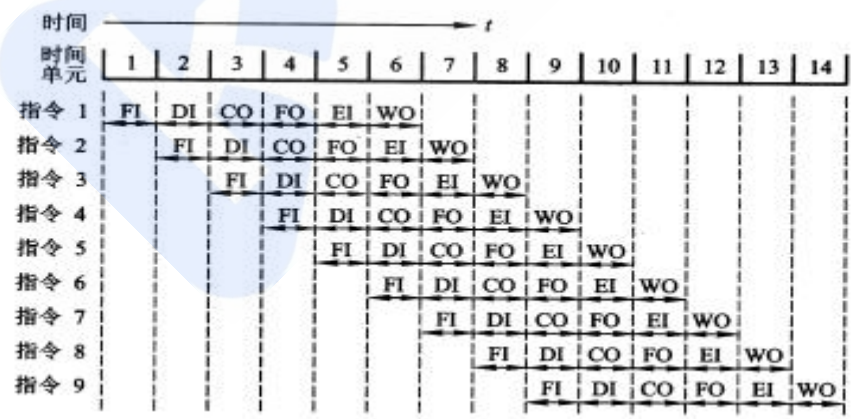
设机器有 7 位操作码(OP_0 ~ OP_6),假设加法指令的操作码为 0001100,形成的加法指令信号的逻辑表达式为:

加法指令= $OP_0OP_1OP_2OP_3OP_4OP_5OP_6$

如,某机器 128 条指令,用 7 位操作码(OP0 ~ OP6),如果其中有 16 条算术逻辑运算指令,可以将这些指令的 3 位操作码都设计相同的编码,如 OP0OP1OP2= 001,而其他位 OP3 ~ OP6 编码表示 16 个不同的指令.

设命令 A 是所有算术逻辑运算在 cy2 周期需要产生的,逻辑表达式:

A=加法指令 · cy2+减法指令 · cy2+逻辑加指令 · cy2+…



=(加法指令 · +减法指令+逻辑加指令+…) · cy2

= OP0 · OP1 · OP2 · cy2

只需要一个与门,就可实现命令 A.

(五) 指令流水线

1.指令流水线的基本概念

⑴流水线基本原理

流水线技术是一种显著提高指令执行速度与效率的技术.方法是:指令取指完成后,不等该指令执行完毕即可取下一条指令.

如果把一条指令的解释过程进一步细分,例如,把分析,执行两个过程分成取指,译码,执行,访存和写回寄存器五个子过程,并用五个子部件分别处理这五个子过程.

这样只需在上一指令的第一子过程处理完毕进入第二子过程处理时,在第一子部件中就开始对第二条指令的第一子过程进行处理.随着时间推移,这种重叠操作最后可达到五个子部件同时对五条指令的子过程进行操作.

指令六级流水时序

⑵影响流水线性能的因素

在流水线中会出现三种相关,影响流水线的畅通流动,这三种相关是结构相关,数据相关和控制相关.

结构相关是当多条指令进人流水线后,硬件资源满足不了指令重叠执行的要求时产生的.

数据相关是指令在流水线中重叠执行时,当后继指令需要用到前面指令的执行结果时发生的.

控制相关是当流水线遇到分支指令和其他改变 PC 值的指令时引起的.

⑶流水线性能

流水线的性能通常用吞吐率,加速比和效率 3 项指标来衡量.

1)吞吐率

在指令流水线中,吞吐率是指单位时间内流水线所完成的指令或输出结果的数量.

2)加速比

流水线的加速比是指 m 段流水线的速度与等功能的非流水线的速度之比.

3)效率

效率是指流水线中个功能段的利用率.

2.超标量和动态流水线的基本概念

⑴超标量

在超标量的处理器结构中,整数和浮点数运算,装入,存储以及条件转移等普通操作指令可以同时启动并独立执行.

超标量流水 CPU 是指集成了多条流水线结构的 CPU,当流水线满载时,每个时钟周期可以完成一条以上的指令.

⑵动态流水线

流水线按功能可分成单功能流水线和多功能流水线两种.

单功能流水线只完成一种功能.如浮点加法或乘法流水线.

多功能流水线则可完成多种功能,它允许在不同时间,甚至同一时间内在流水线内连接不同功能段的子集来实现不同功能.

流水线按工作方式可分为静态流水线和动态流水线两种.

静态流水线中,同一时间内它只能以一种功能方式工作.它可以是单功能的,也可以是多功能的.当是多功能流水线时,则从一种功能方式变为另一种功能方式时,必须先排空流水线,然后为另一种功能设置初始条件后方可使用.显然,不希望这种功能的转换频繁的发生,否则将严重影响流水线的处理效率.

动态流水线则允许在同一时间内将不同的功能段连接成不同的功能子集(前提条件是功能部件的使用不发生冲突),以完成不同的运算功能.显然,动态流水线必是多功能流水线,而单功能流水线则必是静态的.

六、总线

(一)总线基本概念

1.总线的基本概念

总线是连接各个部件的信息传输线,是各个部件共享的传输介质,总线上信息的传送分为串行和并行传输

2.总线通信方式

同步通信	通信双方由统一时标控制数据传送称为同步通信	一般用于总线长度较短,各部件存取时间比较一致的场合
异步通信	异步通信克服了同步通信的缺点,允许各模块速度的不一致性,给设计者充分的灵活性和选择余地.	⑴不互锁方式.主模块发出请求信号后,不等待接到从模块的回答信号,而是经过一段时间.确认从模块已收到请求信号后,便撤消其请求信号; 从设备接到请求信号后,在条件允许时发出回答信号,并经过一段时间,确认主设备已收到回答信号后,自动撤消回答信号.可见通信双方并无互锁关系. ⑵半互锁方式.主模块发出请求信号,待接到从模块的回答信号后再撤消其请求信号,存在着简单的互锁关系: 而从模块发出回答信号后,不等待主模块回答,在一段时间后便撤消其回答信号,无互锁关系.故称半互锁方式. ⑶全互锁方式.主模块发出请求信号,待从模块回答后再撤其请求信号; 从模块发出回答信号,待主模块获知后,再撤消其回答信号.故称全互锁方式.

半同步通信	半同步通信集同步与异步通信之优点,既保留了同步通信的基本特点,如所有的地址,命令,数据信号的发出时间,都严格参照系统时钟的某个前沿开始,而接收方都采用系统时钟后沿时刻来进行判断识别.同时又像异步通信那样,允许不同速度的模块和谐地工作.为此增设了一条“等待”()响应信号线.	半同步通信适用于系统工作速度不高,但又包含了许多工作速度差异较大的各类设备的简单系统.半同步通信控制方式比异步通信简单,在全系统内各模块又在统一的系统时钟控制下同步工作,可靠性较高,同步结构较方便.其缺点是对系统时钟频率不能要求太高,故从整体上来看,系统工作的速度还是不很高.
分离式通信	分离式通信的基本思想是将一个传输周期(或总线周期)分解为两个子周期	这种方式控制比较复杂,一般在普通微机系统很少采用

按照系统总线传输信息的不同

数据总线	双向传输	高电平为 1 低电平为 0	
地址总线	单向传输	地址线位数 n 与存储单元 N 数之间的关系 $N=2^n$	
控制总线	读写控制线 片选线	每根都是单向的 总体输入输出	用来发出各种控制信号的传输线

通信总线	串行总线	近距离 小于 30 米
	并行总线	远距离 几米到数公里

2.总线的分类

1)片内总线:芯片内部的总线

2)系统总线:计算机各部件之间 的信息传输线

数据总线:双向 与机器字长,存储字长有关

地址总线:单向 与存储地址, I/O 地址有关

控制总线:部分出部分入 控制器控制所有部件

3)通信总线:用于 计算机系统之间 或 计算机系统,与其他系统(如控制仪表,移动通信等)之间的通信

传输方式:串行通信总线和并行通信总线

3.总线的组成及性能指标

总线的结构通常分为单总线结构和多总线结构.

单总线结构是将 CPU,主存,I/O 设备(通过 I/O 接口)都挂在一组总线上.

多总线结构的特点是将速度较低的 I/O 设备从单总线上分离出来,形成主总线与 I/O 设备总线分开的结构.

总线的性能指标

1)总线宽度:数据总线的根数

2)总线带宽:数据传输率

3)时钟同步/异步:总线上的数据与时钟同步的称为同步总线,与时钟不同步的称为异步总线

4)总线复用:一条信号线上分时传送两种信号.

5)信号线数:地址总线,数据总线和控制总线三种总线数的总和.

6)总线控制方式:包括突发工作,自动配置,总裁方式,逻辑方式,技术方式等.

7)其他指标:负载能力,电源电压,总线宽度能否扩展等.

(二)总线仲裁

由于总线上连接着多个部件,何时由哪个部件发送信息,如何定时,如何防止信息丢失,如何避免多个设备同时发送,如何规定接收部件等一系列问题都需要总线控制器统一管理,主要包括总线的判优控制(仲裁逻辑)和通信控制.

总线仲裁逻辑可分为集中式和分布式两种,前者将控制逻辑集中在一处(如在 CPU 中),后者将控制逻辑分散在总线的各个部件之上.

1.集中仲裁方式

集中仲裁方式有三种:

⑴链式查询

当一个或多个设备同时发出总线使用请求信号 BR 时,中央仲裁器发出的总线授权信号 BG 沿着菊花链串行的从一个设备依次传送到下一个设备,到达离出发点最近的发出总线请求的设备之后就不再往下传.

⑵计数器定时查询

总线上个设备通过总线请求信号 BR,发出请求,中央仲裁器接收到请求信号后,在总线忙信号 BS 为“0”的情况下,让计数器开始计数,计数值通过一组地址线发往各设备.每个设备有一个地址判别电路,如果地址线上的计数值与总线请求设备地址一致,则该设备对 BS 线置“1”,表示该设备获得了总线使用权,同时中止计数查询.

⑶独立请求方式

每个连接到总线的设备都有一组单独的总线请求信号 BRi 与总线授权信号 BGi.每个设备请求使用总线时,它们各自发出自己的总线请求信号.中央仲裁器中设置了一个专门的排队电路,由它根据一定的优先次序决定优先响应哪个设备的请求,然后给该设备总线授权信号 BGi

2.分布仲裁方式

同集中式仲裁相比,分布式仲裁不需要中央仲裁器,而是让各个主设备功能模块都有自己的仲裁号和仲裁电路.需要使用总线时,各个设备的功能模块将自己唯一的仲裁号发送到共享的总线上,各自的仲裁电路再将从仲裁总线上获得的仲裁号和自己的仲裁号相对比,获胜的仲裁号将保留在仲裁总线上,相应设备的总线

请求获得响应.

(三) 总线操作和定时

目前在总线上的操作主要有以下几种:

1)读和写

读是将从设备(如存储器)中的数据读出并经总线传输到主设备(如 CPU);写是主设备到从设备的数据传输过程.

2)块传送

主设备给出要传输的数据块的起始地址后,就可以利用总线对固定长度的数据一个接一个的读出或写入.

3)写后读或读后写

主设备给出地址一次,就可以进行先写后读或者先读后写操作,先读后写往往用于校验数据的正确性,先写后读往往用于多道程序的对共享存储资源的保护.

4)广播和广集

主设备同时向多个从设备传输数据的操作模式称为广播.广集操作和广播操作正好相反,它将从多个从设备的数据在总线上完成 AND 或 OR 操作,常用于检测多个中断源.

所谓定时,是指事件出现在总线上的时间关系.总线常用的定时协议有同步定时方式和异步定时方式

1.同步定时方式

同步定时方式要求所有的模块由统一的始终脉冲进行操作的控制,各模块的所有动作均在时钟周期的开始产生,并且多数动作在一个时钟周期内完成.

2.异步定时方式

异步定时方式是一种应答方式或者互锁机制的定时方式.对于异步操作,操作的发生由主设备或从设备的的特定信号来确定.总线上一个事件的发生取决于前一个事件的发生,双方互相提供联络信号.

(四) 总线标准

总线标准就是系统与各模块,模块与模块之间的一个互连的标准界面.

目前流行的总线标准有以下几种:

1.系统总线

1)ISA----工业标准体系(Industry Standard Architecture),它是最早出现的微型计算机总线标准,应用在 IBM 的 AT 机上.直到现在,微型计算机主板或工作站主板上还保留有少量的 ISA 扩展槽.

2)EISA----扩展工业标准体系(Extended Industry Standard Architecture),主要用于 286 微机.EISA 对 ISA 完全兼容.

3)VESA----视频电子标准协会(Video Electronic Standard Association),是按照局部总线标准设计的一种开放总线,只适合于 486 的一种过渡标准,已淘汰.

4)PCI----外围设备互联(Peripheral Component Interconnection),PCI 局部总线是高性能的 32 位或 64 位总线,它是专门为高集成度的外围部件,扩充插板和处理器/存储器系统而设计的互连机制.

5)AGP----是一种新型的视频接口的技术标准,专用于连接主存和图形存储器.AGP 总线宽 32 位,时钟频率 66MHz,能以 133MHz 工作,最高的传输速率可达 533Mbps.

2.设备总线

1)IDE----集成驱动电子设备(Integrated Drive Electronics),它是一种在主机处理器和磁盘驱动器之间广泛使用的集成总线.绝大部分 PC 的硬盘和相当数量的 CD-ROM 驱动器都是通过这种接口和主机连接的.

2)SCSI----小型计算机系统接口(Small Computer System Interface),现在这种接口不再局限于将各种设备与小型计算机直接连接起来,它已经成为各种计算机(包括工作站,小型机,甚至大型机)的系统接口.

3)RS-232----(Recommended Standard-232C),是由美国电子工业协会 EIA(Electronic Industries Association)推荐的一种串行通信总线标准.

4) USB----USB(Universal Serial Bus)接口基于通用的连接技术,可实现外设的简单快速连接,已达到方便用户,降低成本,扩展微机连接外设范围的目的.

七、输入输出(I/O)系统

(一) I/O 系统基本概念

除了 CPU 存储器两大模块之外,计算机硬件系统的第三个关键部分就是输入输出模块,也称输入输出系统.输入输出系统的发展概况

1)早期　　分散连接,CPU 和 I/O 设备　串行 工作,程序查询方式

2)接口模块和 DMA 阶段　　总线连接,CPU 和 I/O 设备　并行 工作,中断方式和 DMA 方式

3)具有通道结构的阶段

4)具有 I/O 处理机的阶段

输入输出系统应该由 I/O 软件和 I/O 硬件两部分组成

I/O 软件

⑴ I/O 指令　CPU 指令的一部分

操作码	命令码	设备码

⑵ 通道指令 通道自身的指令

指出数组的首地址、传送字数、操作命令

如 IBM/370 通道指令为 64 位

2. I/O 硬件

设备 I/O 接口

设备 设备控制器　通道

输入输出系统软件的主要任务是:

①何将用户编制的程序(或数据)输入至主机内;

②如何将运算结果输送给用户;

③如何实现 I/O 系统与主机工作的协调等.

(二)I/O 设备与主机的联系方式

1.I/O 编址方式

通常将 I/O 设备码视为地址码,对 I/O 地址码的编址可采用两种方式:

统一编址　　用取数、存数指令

就是将 I/O 地址看作是存储器地址的一部分.如在 64K 地址的存储空间中,划出 8K 地址作为 I/O 的地址,.凡是在这 8K 地址范围内的访问,就是对 I/O 的访问,所用的指令与访存指令相似.

不统一编址　有专门的 I/O 指令

就是指 I/O 地址和存储器地址是分开的,所有对 I/O 的访问必须有专用的 I/O 指令.

显然统一编址占用了存储空间,减少了主存容量,但无需专用的 I/O 指令.不统一编址由于不占用主存空

间,故不影响主存容量,但需设 I/O 专用指令.因此,设计机器时,需根据实际情况权衡考虑选取何种编址方式.

2.设备寻址 用设备选择电路识别是否被选中

由于每台设备都赋予一个设备号,因此,当要启动某一设备时,可由 I/O 指令的设备码字段直接指出该设备的设备号. 通过接口电路中的设备选择电路,便可选中要交换信息的设备.

3.传送方式

- 并行传送
- 串行传送

4.联络方式 三种联络方式

- ⑴立即响应方式
- ⑵异步工作采用应答信号联络
- ⑶同步工作采用同步时标联络

5.I/O 与主机的连接方式

I/O 设备与主机的连接方式通常有两种: 辐射式和总线式.

采用辐射式连接方式时,要求每台 I/O 设备都有一套控制线路和一组信号线,因此所用的器件和连线较多,对 I/O 设备的增删都比较困难.这种连接方式大多出现在计算机发展的初期阶段.

(三)I/O 端口及其编址

CPU 采用 2 种方法访问 I / O 设备,也称 2 种不同的 I / O 端口编址方法:

⑴专门的 I / O 指令

例如,指令:IN 完成输入,指令 OUT 完成输出操作.指令的地址码字段指出输入输出设备的设备代码.

由相应的控制信号(如 M / IO#)来区分 CPU 执行的是什么指令.

⑵利用访问存储器指令完成 I / O 功能

从主存的地址空间中分出一部分地址码作为 I / O 的设备代码,当访问到这些地址时,表示被访的不是主存储器,而是 I / O 设备端口.没有控制信号区分存储器和端口,采用的是将该段地址译码输出连接到外设的端口.

(四) I/O 方式

(I/O 设备与主机信息传送的控制方式)

1.程序查询方式

程序查询方式的核心问题是每时每刻需要不断查询 I/O 设备是否准备好.CPU 不断地询问外设是否准备好:如果准备好,CPU 执行 IO 操作;否则,CPU 一直等待.CPU 大部分时间处于等待状态,利用率不高.

2.程序中断方式

⑴中断的基本概念

计算机在执行程序的过程中,当出现异常情况或者特殊情况时,CPU 停止当前程序的运行,转向对这些异常情况或者特殊情况的处理,处理结束之后再返回到现行程序的间断处继续运行,该过程就是中断.

⑵中断响应过程

当多个中断源向 CPU 提出中断请求时,CPU 在任何一个时刻只能接受一个中断源的请求,所以,当多个中断源同时请求时,CPU 必须对各个中断源的请求进行排队,且只能接受级别最高的中断源的请求,不允许低级别的中断源中断正在运行的中断服务程序.

每个设备都配备一个中断请求触发器和中断屏蔽触发器,当中断请求触发器为“1”时,表示该设备向 CPU 提出中断请求,如果中断屏蔽触发器为“1”时,表示该设备被屏蔽,即封锁其中断源的请求.

当多个中断源同时向 CPU 提出请求,CPU 需要对这些中断源的请求进行排队,也称中断判优,有两种判优的方法:

1)查询法

由测试程序按一定优先排队次序检查各个设备的“中断触发器”(或称为中断标志),当遇到第一个“1”标志时,即找到了优先进行处理的中断源,通常取出其设备码,根据设备码转入相应的中断服务程序.

2)串行排队链法

由硬件组成一个串行的优先链,称作排队链.

一旦 CPU 确定接受某个中断源的请求,就需要执行该设备的中断服务程序,因此需要找到中断服务程序的入口地址.入口地址的寻找可以用软件或硬件的方法实现.硬件向量法就是通过向量地址来寻找设备的中断服务程序的入口地址.中断向量地址形成部件可以通过向接受请求的中断源发送中断响应信号,然后由被响应的设备回送设备码,根据设备码来产生中断向量地址.

⑶中断处理过程

中断处理过程可分以下几个步骤:

1)关中断

进入不可再次响应中断的状态,由硬件自动实现.因为接下去要保存断点,保存现场.在保存现场过程中,即使有更高级的中断源申请中断,CPU 也不应该响应;否则,如果现场保存不完整,在中断服务程序结束之后,也就不能正确地恢复现场并 继续执行现行程序.

2)保存断点和现场.

为了在中断处理结束后能正确地返回到中断点,在响应中断时,必须把当前的程序计数器 PC 中的内容(即断点)保存起来.

现场信息一般指的是程序状态字,中断屏蔽寄存器和 CPU 中各寄存器的内容.

3)判别中断源,转向中断服务程序.

在多个中断源同时请求中断的情况下,本次实际响应的只能是优先权最高的那个中断源.所以,需进一步判别中断源,并转入相应的中断服务程序入口.

4)开中断.

因为接下去就要执行中断服务程序,开中断将允许更高级中断请求得到响应,实现中断嵌套.

5)执行中'断服务程序.

不同中断源的中断服务程序是不同的,实际有效的中断处理工作是在此程序段中实现的.

6)退出中断.

在退出时,又应进入不可中断状态,即关中断,恢复现场,恢复断点,然后开中断,返回原程序执行.

⑷多重中断和中断屏蔽的概念.

多重中断是指在处理某一个中断过程又发生了新的中断请求,从而中断该服务程序的执行,又转去进行新的中断处理.这种重叠处理中断的现象又称为中断嵌套.

中断屏蔽

当产生中断请求后,用程序方式有选择地封锁部分中断,而允许其余部分中断仍得到响立,称为中断屏蔽.

实现方法是为每个中断源设置一个中断屏蔽触发器来屏蔽该设备的中断请求.具体说,用程序方法将该触发器置"1",则对应的设备中断被封锁,若将其置"0",才允许该设备的中断请求得到响应.由各设备的中断屏蔽触发器组成中断屏蔽寄存器.

3.DMA 方式

DMA 是 I / O 设备与主存储器之间由硬件组成的直接数据通路,用于高速 I / O 设备与主存之间的成组数据传送.

数据传送是在 DMA 控制器控制下进行的,由 DMA 控制器给出当前正在传送的数据字的主存地址,并统计传送数据的个数以确定一组数据的传送是否已结束.在主存中要开辟连续地址的专用缓冲器,用来提供或接收传送的数据.在数据传送前和结束后要通过程序或中断方式对缓冲器和 DMA 控制器进行预处理和后处理.

3.DMA 方式与程序中断方式的比较

	中断方式	DMA 方式
⑴ 数据传送	程序	硬件
⑵ 响应时间	指令执行结束	存取周期结束
⑶ 处理异常情况	能	不能
⑷ 中断请求	传送数据	后处理
⑸ 优先级	低	高

⑴DMA 控制器的组成

1)主存地址寄存器(MAR)　　寄存器初始值为主存缓冲区的首地址.

2)外围设备地址寄存器(ADR)　该寄存器存放 I / O 设备的设备码,具体内容取决于 I / O 设备的数据格式和地址字编址方式.

3)字数计数器(WC)　　该计数器对传送数据的总字数进行统计 .

4)控制与状态寄存器(CSR)　　该寄存器用来存放控制字和状态字.

5)数据缓冲寄存器(DBR)　　该寄存器用来暂存 I / O 设备与主存传送的数据.

6)中断机构　　当字计数器溢出(全“0”)时,表示一批数据交换完成,由“溢出信号”通过中断机构向 CPU 提出中断请求,请求 CPU 作 DMA 操作后处理.

⑵DMA 传送过程

1)DMA 预处理

在进行 DMA 数据传送之前要用程序做一些必要的准备工作.先由 CPU 执行几条输入输出指令,测试设备状态,向 DMA 控制器的设备地址寄存器中送入设备地址并启动设备,在主存地址寄存器中送入交换数据的主存起始地址,在数据字数寄存器中送入交换的数据个数.在这些工作完成之后,CPU 继续执行原来的程序.

2)DMA 控制 I/O 设备与主存之间的数据交换

I/O 设备启动后,若为输入数据,则要进行以下操作:

<1>从输入介质读入一个字到数据缓冲寄存器 DBR 中.

<2>向 CPU 发 DMA 请求,在取得总线控制权后,将 DBR 中的数据送人主存的数据寄存器.

<3>将 DMA 中的 MAR 内容送主存的地址寄存器,启动写操作,将数据写入主存.

<4>将 WC 内容减 1,将 MAR 的内容加 1,给出下一个字的地址.

<5>判断 WC 是否为“0”,若不是,说明还有数据需要传送,检查无错后准备下一字的输入.若 WC 为 0,表明一组数据已传送完毕,此时应置结束标志,向 CPU 发中断请求.

3)CPU 中断原程序进行后处理

若需继续交换数据,则又要对 DMA 控制器进行初始化;若不需交换数据,则停止外设;若为出错,则转错误诊断及处理程序.

三种方式的 CPU 工作效率比较

⑶DMA 三种工作方式:

1)CPU 暂停方式	主机响应 DMA 请求后,让出存储总线,直到一组数据传送完毕后,DMA 控制器才把总线控制权交还给 CPU
2)CPU 周期窃取方式	DMA 控制器与主存储器之间传送一个数据,占用(窃取)一个或几个 CPU 周期,即 CPU 暂停工作一个或几个周期,然后继续执行程序
3)CPU 和 DMA 交替工作方式	当 CPU 的工作周期比主存访问时间长时,使用这种方式.这种方式不需要总线使用权的申请,建立和归还过程.将 CPU 周期分成 C1 和 C2 两个周期,其中 C1 专门用于给 DMA 访问主存,C2 提供给 CPU 访问主存.总线使用权通过 C1 和 C2 分别控制.CPU 和 DMA 各自有自己的访问地址寄存器,数据寄存器和读写信号

4.通道方式

I/O 通道是计算机系统中代替 CPU 管理控制外设的独立部件,是一种能执行有限 I/O 指令集合——通道命令的 I/O 处理机.

在通道控制方式下,一个主机可以连接几个通道.每个通道又可连接多台 I/O 设备,这些设备可具有不同速度,可以是不同种类.这种输入输出系统增强了主机与通道操作的并行能力以及各通道之间,同一通道的各设备之间的并行操作能力.同时也为用户提供了增减外围设备的灵活性.

采用通道方式组织输入输出系统,多使用主机—通道—设备控制器—I/0 设备四级连接方式.通道通过执行通道程序实施对I/O 系统的统一管理和控制,因此,它是完成输入输出操作的主要部件.在CPU 启动通道后,通道自动地去内存取出通道指令并执行指令.直到数据交换过程结束向 CPU 发出中断请求,进行通道结束处理工作.

I/O 通道的种类:

根据多台设备共享通道的不同情况,可将通道分为三类:字节多路通道,选择通道和数组多路通道.

⑴字节多路通道(低速,分时)

字节多路通道(multiplexor channel)是一种简单的共享通道,在时间分割的基础上,服务于多台低速和中速面向字符的外围设备.

字节多路通道包括多个子通道,每个子通道服务于一个设备控制器,可以独立地执行通直指令.每个子通道都需要有字符缓冲寄存器,I/O 请求标志/控制寄存器,主存地址寄存器和字节计数寄存器.而所有子通道的控制部分是公共的,由所有子通道所共享.通常,每个通道的有关指令和参量存放在主存固定单元中.当通道在逻辑上与某一设备连通时,将这些指令和参量取出来,送入公共控制部分的寄存器中使用.

字节多路通道要求每种设备分时占用一个很短的时间片,不同的设备在各自分得的时间片内与通道建立传输连接,实现数据的传送.

⑵选择通道(高速,独占)

选择通道每次只能从所连接的设备中选择一台 I/O 设备的通道程序,此刻该通道程序独占了整个通道.当它与主存交换完数据后,才能转去执行另一个设备的通道程序,为另一台设备服务.因此,连接在选择通道上的若干设备,只能依次使用通道与主存传送数据.数据传送是以成组(数据块)方式进行,每次传送一个数据块,因此,传送速率很高.选择通道多适合于快速设备(磁盘),这些设备相邻字之间的传送空闲时间极短.

⑶数组多路通道(综合)

数组多路通道把字节多路通道和选择通道的特点结合起来.它有多个子通道,既可以执行多路通道程序,像字节多路通道那样,所有子通道分时共享总通道;又可以用选择通道那样的方式传送数据.

数组多路通道具有多路并行操作能力,又具有很高的数据传送速率,赢得了吞吐率的较大提高.它的缺点是增加了控制的复杂性.

开课吧

使 命

让每个人都能公平和便利地获取优质教育服务，并实现可持续职业成长

愿 景

打造全球顶尖的人才科技公司，创办一所国际知名大学，成为令人尊敬的企业

价值观

关心并热爱、只为赋能人才、极致敢为、坦诚开放、拥抱变化、始终创业