第 3 章　计算机的运算基础

计算机一般使用电子器件的两种状态来表示信息，如高电平和低电平、通路和断路，因此，计算机内部是一个二进制数字世界。本章主要讨论以下问题。

（1）计算机使用二进制的理论基础是数理逻辑。什么是数理逻辑？

（2）计算机内部使用二进制，在日常生活中人们习惯使用十进制，二进制数如何与十进制数进行转换？

（3）任何数据都必须以二进制形式存储在计算机中，如何将各种类型的数据表示成二进制的形式？如何将指令表示成二进制形式？

（4）计算机之所以具有逻辑处理能力，是由于其内部具有能够实现各种逻辑功能的逻辑电路。逻辑电路的基本原理是什么？逻辑电路是如何工作的？

（5）计算机的基本部件是什么？如何理解内存？处理器是如何工作的？

【情景问题】模拟数据与数字数据

真实世界的信息大多是连续的、无限的，如天气的变化、移动的距离、色彩的渐变、声音的波……用连续形式表示的信息称为模拟信息。模拟是术语 analog 的翻译，它不属于中国人的思维模式，以后我们还可能会遇到很多这样的概念和术语，我们不能用中国文化里的内涵去理解，就像外国人很难理解什么是道，什么是太极一样。

计算机内部是一个二进制数字世界，而且计算机内存是有限的，计算机的硬件设备能处理的信息也是有限的，数据处理首先要解决的问题是如何用有限的计算机表示无限的真实世界。解决方法是数字化，将连续的信息分割成独立的片段，然后单独表示每一个片段。换言之，就是把一个连续的实体分割成若干离散的元素，然后用二进制数字单独表示每个离散元素。用离散形式表示的数字化信息称为数字信息。

用有限的计算机精确地表示无限的真实世界几乎是不可能的，只能将目标定为满足实际的计算需要，满足人类的视觉及听觉等感官。例如，水银温度计是一种模拟设备，水银柱按温度的正比例以连续方式在管子中变化，我们校准这个管子，给它标上刻度，就能够读出当前的温度。例如，实际温度是 26.8℃，水银柱的确指在相应的位置，如图 3.1 所示。但即使我们标记得再详细，也会有细微的误差，能够达到使用的精度需求就足够了。

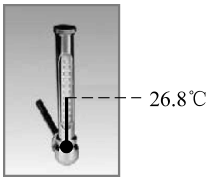


图 3.1　温度的表示

3.1　数理逻辑基础

3.1.1　数理逻辑的起源和发展

逻辑(logic)一词源于希腊文 logoc，有“思维”和“表达思考的言辞”之意。一直以来，人

们希望使用数学方法来研究思维,具体地说,使用一种符号语言来代替自然语言对思维过程进行描述,把人类的思维过程转换为数学的计算。

用数学方法来描述和处理思维最早由德国数学家莱布尼茨提出,但直到1847年英国数学家乔治·布尔发表著作《逻辑的数学分析》后才有所发展。1879年德国数学家弗雷格(G. Frege)在《概念语言———一种按算术的公式语言构成的纯思维公式语言》一书中建立了第一个比较严格的逻辑演算系统,英国逻辑学家怀特海(A. N. Witehead)和罗素(B. Russell)合著的《数学原理》一书对当时数理逻辑的成果进行了总结,使得数理逻辑形成了专门的学科。

英国数学家乔治·布尔(George Boole)16岁就开始任教以维持生活,20岁时对数学产生了浓厚的兴趣,开始广泛涉猎著名数学家牛顿、拉普拉斯、拉格朗日等人的数学名著,并写下了大量笔记。1847年,发表了著作 *The Mathematical Analysis of Logic*,在这本书中阐述了逻辑学公理,建立了逻辑代数,因此,逻辑代数也称为布尔代数。逻辑代数建立在两个逻辑值0、1和三个逻辑运算与、或、非的基础上,这种简化的二值逻辑为计算机的二进制数、开关逻辑元件和逻辑电路的设计铺平了道路,并最终为计算机的发明奠定了数学基础。

数理逻辑是用数学的方法来研究推理规律的科学,它采用符号的方法来描述和处理思维形式、思维过程和思维规律,即把逻辑思维所涉及的概念、判断、推理用符号来表示,用公理化体系来刻画,并基于符号串形式的演算来描述推理过程的一般规律,从而实现人类思维过程的演算化、机械化,最终计算机化(即在计算机上实现)。1930年以前,数理逻辑的发展主要针对纯数学的需要,其后,数理逻辑开始被应用于所有开关线路的理论中,并在计算机科学等方面得以应用,成为计算机科学的基础理论之一。

数理逻辑的主要分支有公理化集合论、证明论、递归函数论、模型论等。逻辑代数是数理逻辑的基础部分,而逻辑代数源自对命题逻辑的研究。

3.1.2 命题代数与逻辑代数

所谓**命题**是一个有具体意义且能够判断真假的陈述句。判断是对事物表示肯定或否定的一种思维形式,所以表达判断的命题总是具有"真"(true,T)或"假"(false,F)两种取值,命题所具有的值称为命题的真值。

命题分为**原子命题**和**复合命题**两种类型。原子命题是不能被分解为更简单的陈述句的命题,而复合命题则是将原子命题用连接词复合而成的命题。

例3.1 以下是几个命题实例。

(1) 长春市是吉林省的省会城市。

(2) 3乘以8等于16。

(3) 姚大龙既擅长书法又擅长绘画。

其中,第一个命题是原子命题,该命题为真,即它的真值为T;第二个命题也是原子命题,该命题为假,即它的真值为F;第三个命题是复合命题,其真值需要根据实际情况确定。

命题代数和普通代数一样,用字母 A,B,C…表示变量,称为**命题变量**(或命题变元),但是命题变元的取值只有两种:T或F。连接词相当于普通代数中的运算符,在命题代数

中最基本的连接词是与、或、非等。

两个命题 A 和 B 的与(又称 A 和 B 的合取)记为 $A \wedge B$,表示当且仅当 A 和 B 同时为真时 $A \wedge B$ 为真,其他情况下 $A \wedge B$ 为假,其真值表如图 3.2(a)所示,与运算的实例如图 3.2(b)所示。

A	B	$A \wedge B$
T	T	T
T	F	F
F	T	F
F	F	F

(a) 与运算的真值表

A：姚大龙擅长书法。
B：姚大龙擅长绘画。
$A \wedge B$：姚大龙既擅长书法又擅长绘画。
只有当命题 A 和 B 均为真时 $A \wedge B$ 才为真。

(b) 与运算的一个实例

图 3.2　与运算

两个命题 A 和 B 的或(又称 A 和 B 的析取)记为 $A \vee B$,表示当且仅当 A 和 B 同时为假时 $A \vee B$ 为假,其他情况下 $A \vee B$ 为真,其真值表如图 3.3(a)所示,或运算的实例如图 3.3(b)所示。

A	B	$A \vee B$
T	T	T
T	F	T
F	T	T
F	F	F

(a) 或运算的真值表

A：姚大龙擅长书法。
B：姚大龙擅长绘画。
$A \vee B$：姚大龙擅长书法或绘画。
只有当命题 A 和 B 均为假时 $A \vee B$ 才为假。

(b) 或运算的一个实例

图 3.3　或运算

命题 A 的非(又称 A 的否)记为 $\neg A$,表示当 A 为真时 $\neg A$ 为假,当 A 为假时 $\neg A$ 为真,其真值表如图 3.4(a)所示,非运算的实例如图 3.4(b)所示。

A	$\neg A$
T	F
F	T

(a) 非运算的真值表

A：姚大龙擅长书法。
$\neg A$：姚大龙不擅长书法。
当命题 A 为真时 $\neg A$ 为假。

(b) 非运算的一个实例

图 3.4　非运算

例 3.2　将下列语句符号化,即表示为命题代数。

(1) 我和他既是同学又是兄弟。

(2) 我和他之间至少有一个去西部。

解　(1) 可表示为 $A \wedge B$,其中 A：我和他是同学,B：我和他是兄弟。

(2) 可表示为 $A \vee B$,其中 A：我去西部,B：他去西部。

可以将命题代数直接推广到逻辑代数。在逻辑代数中,逻辑变量的取值只有“真”和“假”,通常以 1 和 0 来表示;通常用符号“·”表示与运算(在不至于混淆的情况下,符号“·”也可以省略),用符号“+”表示或运算,用上画线“¯”表示非运算。逻辑运算的基本规则如

表 3.1 所示。需要强调的是,参与逻辑运算的数值没有符号位。

表 3.1 逻辑运算的基本规则

与	或	非
$0 \cdot 0=0$	$0+0=0$	
$0 \cdot 1=0$	$0+1=1$	$\overline{0}=1$
$1 \cdot 0=0$	$1+0=1$	$\overline{1}=0$
$1 \cdot 1=1$	$1+1=1$	

香农(C. E. Shannon)1916 年出生于美国密歇根州,1936 年毕业于密歇根大学,获得数学和电子工程学士学位,1940 年获得麻省理工学院数学博士学位和电子工程硕士学位。1938 年,香农将逻辑代数应用于开关电路,因此,逻辑代数也称为开关代数。香农在 1948 年发表的论文《通信的数学原理》和 1949 年发表的论文《噪声下的通信》中,解决了过去许多悬而未决的问题,被尊称为"信息论之父"。

思考题

1. 为什么规定命题必须是一个陈述句?疑问句不可以做命题吗?

2. 生门和死门:一位哲学家被关在一个监狱里,监狱有两扇门,一扇通向自由(生门),一扇通向死亡(死门),监狱里有两个看守,两个看守都知道哪个是生门哪个是死门,但是一个说真话一个说假话,哲学家只能说一句话,哲学家该如何活着出去?

3.2 二进制

逻辑代数只使用 1(真)和 0(假)两个数,这样,当二进制的加法、乘法等算术运算与逻辑代数的逻辑运算建立了对应关系后,就可以用逻辑部件来实现二进制数据的各种运算。

3.2.1 进位计数制

按进位(当某一位的值达到某个固定量时,就要向高位产生进位)的原则进行计数的方法称为**进位计数制**,简称**进制**。在日常生活中,人们使用最多的是十进制,此外,人们也使用许多非十进制的计数方法,例如,时间采用的是六十进制,即 60 秒为 1 分钟,60 分钟为 1 小时;月份采用的是十二进制,即 1 年有 12 个月。

不同的进制以基数来区分,若以 r 代表基数,则

$r=10$ 为十进制,可使用 0,1,2,…,9 共 10 个数码;

$r=2$ 为二进制,可使用 0,1 共 2 个数码;

$r=8$ 为八进制,可使用 0,1,2,…,7 共 8 个数码;

$r=16$ 为十六进制,可使用 0,1,2,…,9,A,B,C,D,E,F 共 16 个数码。

r 进制数通常写作$(a_n \cdots a_1 a_0 . a_{-1} \cdots a_{-m})_r$,其中 $a_i \in \{0,1,\cdots,r-1\}(-m \leqslant i \leqslant n)$。例如,二进制数 1101 写作$(1101)_2$,十进制数 689.12 写作$(689.12)_{10}$。

进位计数制采用位置记数法(数码按顺序排列)表示数,处于不同位置上的数码代表不同的值,例如,在十进制中,数码 8 在个位上表示 8,在十位上表示 80,在百位上表示 800,而在小数点后 1 位表示 0.8,所以,每个位置都对应一个位权值。对于 r 进制数$(a_n\cdots a_1a_0.a_{-1}\cdots a_{-m})_r$,小数点左面的位权值依次为 $r^0,r^1,\cdots,r^n$,小数点右面的位权值依次为 $r^{-1},\cdots,r^{-m}$。每个位置上的数码所表示的数值等于该数码乘以该位置的位权值。例如,十进制数 198.63 可以表示成:$198.63=1\times10^2+9\times10^1+8\times10^0+6\times10^{-1}+3\times10^{-2}$。

进位计数制在执行算术运算时,遵守“逢 r 进 1,借 1 当 r”的规则,其中 r 为进制。如十进制的规则为“逢 10 进 1,借 1 当 10”,二进制的规则为“逢 2 进 1,借 1 当 2”。二进制的算术运算规则非常简单,如表 3.2 所示。

表 3.2　二进制的算术运算规则

加	减	乘	除
0+0=0	0−0=0	0×0=0	0÷0(没有意义)
0+1=1	0−1=1(向高位借 1)	0×1=0	0÷1=0
1+0=1	1−0=1	1×0=0	1÷0(没有意义)
1+1=0(向高位进 1)	1−1=0	1×1=1	1÷1=1

例 3.3　计算 1010+10 和 1010−100 的值。

解

$$\begin{array}{r} 1010 \\ +\quad 10 \\ \hline 1100 \end{array} \qquad \begin{array}{r} 1010 \\ -\quad 100 \\ \hline 110 \end{array}$$

则:1010+10=1100,1010−100=110。

例 3.4　计算 1010×101 和 10101÷100 的值。

解

$$\begin{array}{r} 1010 \\ \times\quad 101 \\ \hline 1010 \\ 0000 \\ 1010 \\ \hline 110010 \end{array} \qquad \begin{array}{r|l} 101 & \\ \hline 100 & 10101 \\ \hline 101 & \\ 100 & \\ \hline 1 & \end{array}$$

则:1010×101=110010,10101÷100=101 余 1。

二进制是德国数学家莱布尼茨在 18 世纪发明的,他的发明受了中国八卦的启迪。莱布尼茨曾写信给当时在康熙皇帝身边工作的法国传教士白晋,询问有关八卦的问题并仔细研究过八卦,莱布尼茨还把自己制造的一台手摇计算器送给了康熙皇帝。

3.2.2　二进制数和十进制数之间的转换

由于人们习惯使用十进制,而计算机内部使用的是二进制,所以,计算机系统需要进行

十进制数和二进制数之间的转换。

将二进制数转换为十进制数只需要将二进制数按位权值展开然后求和,所得结果即为对应的十进制数。

例 3.5 将二进制数 1101.11 转换为十进制数。

解 $1101.11=1\times2^3+1\times2^2+0\times2^1+1\times2^0+1\times2^{-1}+1\times2^{-2}=13.75$

则:$(1101.11)_2=(13.75)_{10}$。

将十进制数转换为二进制数需要将十进制数分解为整数部分和小数部分,对两部分分别进行转换,然后相加得到转换的最终结果。

将十进制整数转换为二进制整数的规则是"**除基取余,逆序排列**",即将十进制整数逐次除以二进制的基数 2,直到商为 0,然后将得到的余数逆序排列,先得到的余数为低位,后得到的余数为高位。

例 3.6 将十进制整数 46 转换为二进制整数。

解

除数	商	余数	
46	23	0	↑
23	11	1	逆序排列
11	5	1	
5	2	1	
2	1	0	
1	0	1	

则:$(46)_{10}=(101110)_2$。

十进制整数转换为 r 进制整数的数学原理

已知十进制整数 x,设 x 对应的 r 进制整数 $y=(a_n\cdots a_1a_0)_r$,则

$y=(a_n\cdots a_1a_0)_r=a_nr^n+\cdots a_1r^1+a_0r^0=((\cdots(a_nr+a_{n-1})r+\cdots+a_2)r+a_1)r+a_0$

将 y 除以 r,商为 $(\cdots(a_nr+a_{n-1})r+\cdots+a_2)r+a_1$,余数为 a_0;

所得商再除以 r,商为 $(\cdots(a_nr+a_{n-1})r+\cdots)r+a_2$,余数为 a_1;

以此类推,直至商为 0,余数为 a_n。

将十进制小数转换为二进制小数的规则是"**乘基取整,正序排列**",即将十进制小数逐次乘以二进制的基数 2,直到积的小数部分为 0,然后将得到的整数正序排列,先得到的整数为高位,后得到的整数为低位。

例 3.7 将十进制小数 0.375 转换为二进制小数。

解

乘数	积	整数	
0.375	0.75	0	↓
0.75	1.5	1	正序排列
0.5	1.0	1	

则:$(0.325)_{10}=(0.011)_2$。

并不是所有的十进制小数都可以精确地转换为二进制小数,如果乘基取整后的小数部

分始终不为 0，则可以根据精度要求转换到一定的位数为止。

例 3.8　将十进制小数 0.325 转换为二进制小数。

解

乘数	积	整数	
0.325	0.65	0	正序排列
0.65	1.3	1	
0.3	0.6	0	
0.6	1.2	1	
0.2	0.4	0	
0.4	0.8	0	
0.8	1.6	1	
0.6	1.2	1	

此后处于无限循环状态，假设精度为小数点后 8 位，则：$(0.325)_{10}=(0.01010011)_2$。

思考题

1. 将 3 个苹果均分成 7 份，应该如何切分才能使切的刀数最少？

2. 如果有 1000 个苹果，有 10 个箱子，现要把 1000 个苹果放在 10 个箱子里面，如何放才能使得以后不管要多少个苹果，都可以整箱付货？

3. 一个工人工作 7 天，老板有一根黄金，每天要给工人 1/7 的黄金作为工资，老板只能切这段黄金 2 刀，请问怎样切才能每天都给工人 1/7 的黄金？

3.3　数字化原理——信息的编码

计算机内部是一个二进制数字世界，所有信息必须进行二进制编码才能存储到计算机中，进而被计算机程序加工和处理。

3.3.1　整数的编码

在数学中，整数的长度指该数所占的实际位数，例如 135 的长度是 3，5 的长度是 1，实际应用时有几位就写几位。在计算机中，整数的长度指该数所占的二进制位数，而且同类型的数据长度一般是固定的，由机器的字长确定，不足部分用 0 补足。换言之，计算机中同一类型的数据具有相同长度，与数据的实际长度无关。假设某计算机系统中整数占 2 字节（即 16 位二进制），则所有整数的长度都是 16 位，则$(68)_{10}=(1000100)_2=(0000000001000100)_2$。在以下讨论中，为简单起见，不失一般性，假设用 8 位二进制表示一个整数。

整数有正数和负数之分，由于计算机使用二进制 0 和 1，因此，可以采用 1 位二进制数表示数值数据的符号，通常用“0”表示正号，用“1”表示负号，也就是对数值数据的符号进行编码。

补码是一种使用最广泛的整数表示方法，其编码规则为：正数的补码其符号位为 0，其余各位与数的绝对值相同；负数的补码其符号位为 1，其余各位是数的绝对值取反，然后在最末位加 1。

例 3.9　$X=+1000101$　$[X]_{补}=\mathbf{0}1000101$

$X=-1000101$　$[X]_{补}=\mathbf{1}0111011$

由于$[+0]_{补}=00000000$,$[-0]_{补}=[-0]_{反}+1=11111111+1=00000000$,因此,补码表示法的优点之一就是零的表示唯一。补码表示法的另一个优点是方便进行算术运算。首先,对于加法运算,符号位可以作为数值参与运算,无须单独处理;其次,减法运算可以转换为加法运算,从而使正负数的加减运算转换为单纯的加法运算,简化了运算规则和逻辑电路。

例 3.10 计算 68+12 和 68−12 的值。

解 $68=+1000100$ $[68]_{补}=01000100$

$12=+0001100$ $[12]_{补}=00001100$

$-12=-0001100$ $[-12]_{补}=[-12]_{反}+1=11110011+1=11110100$

	01000100	$[68]_{补}$			01000100	$[68]_{补}$
+	00001100	$[12]_{补}$		+	11110100	$[-12]_{补}$
	01010000	$[80]_{补}$		1	00111000	$[56]_{补}$

由于 8 位二进制表示一个整数,所以最高位的进位自然丢失,同时得到正确的结果。

补码的理论基础是模数的概念。从物理意义上讲,模数是某种计量器的容量。例如钟表的模数是 12,如果现在的准确时间是 6 点,而你的手表显示的是 8 点,怎样把表拨准呢?可以有两种方法:把时针往后拨 2 小时,或往前拨 10 小时。之所以这两种方法的效果是一样的,是因为 2 和 10 对模数 12 互为补数。模数系统有这样一个结论:一个数 A 减去另一个数 B,等价于数 A 加上负 B,也等价于数 A 加上数 B 的补数。用 mod 表示取模运算,则 $8-2=8+(-2)=8+(-2 \bmod 12)=(8+10) \bmod 12$(注意:多于模数 12 的部分相当于进位丢掉了)。

例 3.11 计算 68+61 的值。

解 $68=+1000100$ $[68]_{补}=01000100$

$61=+0111101$ $[61]_{补}=00111101$

	01000100	$[68]_{补}$
+	00111101	$[61]_{补}$
	10000001	$[-127]_{补}$

但是,68+61=129,这种错误称为**溢出**。产生溢出的原因是要表示的值超过了系统能够表示的值的范围,例如,4 位二进制数表示的整数范围是$-2^3 \sim 2^3-1$(注意有 1 位是符号位),如表 3.3 所示,8 位二进制数表示的整数范围是$-2^7 \sim 2^7-1$,以此类推。

表 3.3 4 位二进制数表示的整数范围

位串	0111	0110	0101	0100	0011	0010	0001	0000	1111	1110	1101	1100	1011	1010	1001	1000
数值	7	6	5	4	3	2	1	0	−1	−2	−3	−4	−5	−6	−7	−8

注:二进制位串是对应数值的补码表示,左边第一位是符号位。

3.3.2 浮点数的编码

数值数据既有正数和负数之分,又有整数和小数之分,整数和小数都可以表示为浮点

数。一个浮点数 X 可以用如下方式(即科学记数法)表示：

$$X = M \times r^E \tag{3.1}$$

其中，r 表示基数，由于计算机采用二进制，因此，基数为 2。E 为 r 的幂，称为数 X 的**阶码**，其值确定了数 X 的小数点的位置。M 为数 X 的有效数字，称为数 X 的**尾数**，其位数反映了数据的精度。

从式(3.1)可以看出，尾数 M 的小数点可以随 E 值的变化而左右浮动，所以，这种表示法称为**浮点表示法**。目前，大多数计算机都把尾数 M 规定为纯小数，把阶码 E 规定为整数。

基数一旦被计算机定义好了，就不能再改变了，因此，浮点表示法无须表示基数，是隐含的。这样，计算机中浮点数的表示由阶码和尾数两部分组成，如图 3.5 所示，其中阶码和尾数可以采用补码表示。

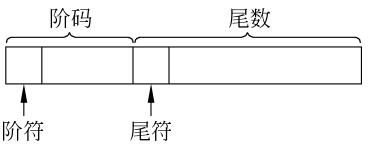


图 3.5　浮点表示法的一般格式

威廉·卡亨(William Kahan)1933 年出生于加拿大多伦多，1954 年在多伦多大学获数学学士学位，1958 年获博士学位。他曾在大学任教，先后在 IBM、惠普、英特尔等公司工作，积累了丰富的工程实践经验。卡亨在英特尔公司工作期间主持 8087 芯片的设计和开发工作，成功地实现了浮点运算部件。卡亨在 IEEE 浮点运算标准的制定、惠普计算机体系结构设计、数值计算、误差分析、自动诊断等方面也做出了突出的贡献，并获得了 1989 年图灵奖。

例 3.12　设 $X=3.625$，假设用 12 位二进制数表示一个浮点数，其中阶码占 4 位，尾数占 8 位，则其浮点表示如下：

$$(3.625)_{10} = (11.101)_2 = 0.11101 \times 2^{10}$$

阶码为+10，其补码为 **0**10，由于阶码占 4 位，则阶码表示为 **0**010(注意是在阶码的前面补 0，因为阶码是整数)；尾数为+0.11101，其补码为 **0**11101，由于尾数占 8 位，则尾数表示为 **0**1110100(注意是在尾数的后面补 0，因为尾数是纯小数)。最后，X 的浮点表示为 **0**010**0**1110100。

例 3.13　设 $X=3.625$，假设用 8 位二进制数表示一个浮点数，其中阶码占 3 位，尾数占 5 位，则其浮点表示如下：

$$(3.625)_{10} = (11.101)_2 = 0.11101 \times 2^{10}$$

阶码为+10，其补码为 **0**10；尾数为+0.11101，其补码为 **0**11101，由于尾数占 5 位，空间不够，则尾数表示为 **0**1110。最后，X 的浮点表示为：**0**10**0**1110。但是 **0**10**0**1110 是 3.5 的浮点表示，也就是说，由于尾数的空间不够大，从而产生了**截断误差**。使用较长的二进制位表示尾数可以减少截断误差的产生，事实上，今天所用的大多数计算机都使用 32 位二进制数来表示一个浮点数。

3.3.3　字符的编码

表示字符的简单方法是列出所有字符，对每一个字符赋予一个二进制位串构成**字符集**。微机上常用的字符集是**标准 ASCII 码**(American Standard Code for Information Interchange，美

国信息交换标准代码),由7位二进制数表示一个字符,总共可以表示128个字符。表3.4给出了标准ASCII码编码表,每个字符都有一个由二进制位串决定的编码值。例如,a的编码值为97,b的编码值为98。

表3.4 标准ASCII码编码表

$b_4b_3b_2b_1$	$b_7b_6b_5$							
	000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	"	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	'	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	|
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	↑	n	~
1111	SI	US	/	?	O	←	o	DEL

扩展ASCII码由8位二进制数表示一个字符,总共可以表示256个字符,通常各个国家都把扩展ASCII码作为自己国家语言文字的代码,但无法满足国际需要,于是出现了Unicode。**Unicode**由16位二进制数表示一个字符,总共可以表示2^{16}个字符,即65 000多个字符,能够表示世界上所有语言的所有字符,包括亚洲国家的表意字符,此外,还能表示许多专用字符(如科学符号)。

贝莫(Bob Bemer)1941年获得航空工程学位,之后在很多有影响的计算机公司工作。贝莫首先认识到如果一台计算机要与另一台计算机通信,需要传送文本信息的标准代码。1960年,贝莫发布了关于60多种计算机代码的调查报告,从而说明了对标准代码的需要。贝莫拟定了标准委员会的工作计划,编写了大量关于编码的文献,提出了转义符的概念。2003年5月,贝莫得到了IEEE-CS颁发的计算机先驱奖,以表彰他通过ASCII码和转义符为满足世界对各种字符集和符号的需要所做出的贡献。

3.3.4　汉字的编码

计算机在我国的应用中，汉字的输入、处理和输出功能是必不可少的，实现汉字处理的前提是对汉字进行编码。我国于 1981 年颁布了《中华人民共和国国家标准信息交换汉字编码》(GB 2312—1980)，该标准根据汉字的常用程度确定了汉字字符集，共收录汉字、数字序号、标点符号、汉语拼音符号等各种符号 7445 个，其中一级汉字 3755 个，二级汉字 3008 个，此外还包括 682 个西文字符和图符。

为了在计算机系统的各个环节方便和确切地表示汉字，需要使用多种汉字编码。例如，由输入设备产生的汉字输入码、用于计算机内部存储和处理的汉字机内码、用于汉字显示和打印输出的汉字字形码等。在汉字的处理过程中，各种汉字编码的转换过程如图 3.6 所示。

图 3.6　汉字编码转换过程

1. 汉字输入码

对于用户而言，在计算机中使用汉字首先遇到的问题就是如何使用西文键盘有效地将汉字输入计算机中。为了便于汉字的输入，中文操作系统都提供了多种汉字输入法，常用的有五笔字型、微软拼音等，不同的输入法对应不同的汉字输入码。例如，汉字“西”用微软拼音输入法时，需依次按下“x”“i”，则“xi”即为“西”字的输入码。

2. 汉字机内码

汉字的机内码是统一的，输入汉字后，需要将汉字输入码转换为汉字机内码。机内码是在计算机内部存储和处理使用的汉字编码，每个汉字用两个 7 位的二进制数表示，在计算机中用 2 字节表示，为了与 ASCII 码相区别，将每个字节的最高位置为 1。例如，“西”字的机内码是 11001110 11110111。

3. 汉字字形码

汉字是一种象形文字，可以将汉字看成一个特殊的图形，这种图形很容易用点阵来描述。所谓**点阵**就是把汉字图形放在一个网格(例如坐标纸)内，凡是有笔画通过的格点为黑点，用 1 来表示；否则为白点，用 0 来表示。那么，黑白点信息就可以用二进制数来表示。汉字字形码就是一个汉字字形的点阵编码，全部汉字字形码的总和称为**汉字库**。显然，表示汉字的点阵越大，汉字就越美观清晰，所需的存储量也就越多。图 3.7 所示是“王”字的 16×16 点阵。

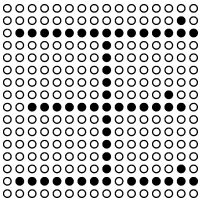


图 3.7　汉字字形码点阵示意图

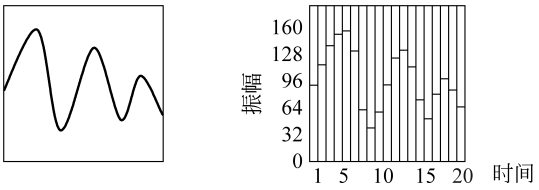
3.3.5　声音的编码

声音是随时间连续变化的波，称为**声波**。声波传到人的耳朵，引起耳膜振动，这就是人们听到的声音。声音信号(又称音频信号)是一种模拟信号，主要由振幅和频率来描述。振幅反映声音的音量大小，频率反映声音振动一次的时间。

将声音数字化，就是每隔一段时间对声波进行采样，将采样点的振幅值用一组二进制数

来表示,如图 3.8(a)所示。

在图 3.8(b)中,横轴是时间,纵轴是声波的振幅,因此,采样是在横轴和纵轴两个维度上进行了离散化。显然,采样的间隔时间越短,数字化音频的质量也就越高,声音质量越接近原始声音,而所需的存储量也越多。例如,音乐 CD 的采样频率是 44kHz,假定它是双声道,每声道占用 2 字节存储采样值,则 1 秒钟的音乐就需要 44 000×2×2≈160KB 的存储量,存储一首 4 分钟长的歌曲,总计需要 4×60×160≈36MB 存储量,可见,数字化的声音文件需要相当大的存储量。



(a) 连续的声波　　(b) 连续声波的离散采样

图 3.8　声音的数字化

人耳能听到声音的频率范围是 20Hz～20kHz,实际上,人类只需要 3.4kHz 的可用信息,这就是电话线路的语音信号带宽。当然,采样频率的提高意味着声音质量的提高,如普通声道的带宽是 11kHz,立体声的带宽是 22kHz,高保真立体声的带宽是 44kHz。

3.3.6　图形和图像的编码

在计算机中,图形和图像是两个不同的概念。**图形**一般指通过绘图软件绘制的,由直线、圆、弧等曲线组成的画面,即图形是由计算机产生的;**图像**是由扫描仪、数码相机等输入设备捕捉的画面,即图像是真实的场景或图片,被输入了计算机。

数字化一幅图形通常采用的是矢量技术,就是把图形分解为一些基本元素,通过图形的基本元素及其属性来表示图形。图形的基本元素有点、线、矩形、圆和椭圆等,属性主要指诸如线的风格、宽度和色彩等影响图形输出效果的内容。矢量技术是用数学方法描述图形的几何形状,因此,存储的数据是绘制图形的数学描述。矢量图形可以进行组合、编辑、放大等处理,如图 3.9 所示。

图 3.9　矢量图形的处理

数字化一幅图像通常采用的是位图技术,就是把图像分解为一些点,这些点称为**像素**,每个像素由一种颜色构成,如图 3.10 所示。颜色是对到达视网膜的各种频率的光的感觉。人类的视网膜有 3 种颜色感光视锥细胞,分别对应红、绿、蓝三原色,人眼可以感觉的所有颜色都由这 3 种颜色混合而成。因此,在计算机中,颜色通常用 RGB(Red-Green-Blue)的组合

来表示。用于表示颜色的二进制位数称为**色深度**，显然，色深度的位数越多，能够表示的颜色就越多，图像也就越逼真。增强彩色指色深度为 16 位的颜色，RGB 中的每个数值用 5 位二进制数表示，剩下的 1 位用于表示颜色的透明度。真彩色指色深度为 24 位的颜色，RGB 中的每个数值用 8 位二进制数表示，每个数值的范围是 0～255，能够表示 1670 万种以上的颜色。

图 3.10　图像与像素

表示一幅图像使用的像素个数称为**分辨率**。如果使用了足够的像素并将这些像素按正确的顺序排列，人们就会认为看到的是一幅图像。由于视觉的停留效果，当每秒钟变化的画面超过 15 帧时，连续出现的各个画面在人眼中产生的视觉停留就会相互连接，因此，视频可以看作连续的图像，随时间变化的图像序列就构成了数字视频。

> 打印机和显示器上使用的字体通常采用矢量技术，这样方便得到可缩放字体。例如，微软公司研发的 TrueType 字体是一种描述如何绘制文本符号的系统。但是，矢量技术还不能提供照片级质量的图像，这就是现在的数码相机采用位图技术的原因。位图技术的缺点是不能方便地放大图像，实际上，放大这种图像只有一个方法，就是把像素变大，这会使图像呈现颗粒状。数码相机中提供了数字变焦技术来解决图像放大后的颗粒状问题。

3.3.7　指令的编码

一个二进制位只能表示两种状态。例如，如果把食物分成甜的和酸的两类，只用一个二进制位即可，可以规定 0 表示食物是甜的，1 表示食物是酸的。同理，2 个二进制位可以表示 4 种状态，因为 2 位可以构成 4 种组合，即 00、01、10、11。例如，如果把食物分成酸、甜、苦、辣 4 种，则需要 2 个二进制位。一般来说，***n* 个二进制位能表示 2^n 种不同的状态**。

> 虽然在技术上只需要最少的二进制位来表示一组状态，在实际表示时常常会多分配一些位数，通常是 2 的幂的倍数。这是因为计算机体系结构一次能够寻址和移动的位数通常是 2 的幂，例如 8、16、32 等。

由于指令系统中包含指令的数量有限，所以，处理器的设计者只需要列出所有的指令，再给每个指令分配一个二进制编码即可。例如，8086/8088 的指令系统共有 133 条基本指令，由于 $2^7<133<2^8$，因此，可以用 8 位二进制数表示一条指令，如 11110100 表示加法指令。因此，处理器的电子器件能够识别指令系统中的每一个二进制编码，计算机硬件只能够识别并执行机器指令。

思考题

1. 在计算机中，一般用一定长度的二进制位来表示整数和浮点数，如果实际应用中要处理非常大的整数或精度要求非常高的浮点数，如何表示这样的数据？

2. 英文字母在计算机处理过程中只有一种编码——ASCII 或 Unicode，为什么汉字在计算机处理过程中有多种编码？

3. 声音等多媒体信息数字化后,其特点就是数据量非常庞大,处理多媒体数据所需的高速传输速度也是计算机内部所不能承受的。如何理解这句话?

3.4 逻辑电路

计算机是电子设备,计算机硬件需要使用许多功能电路,例如触发器、寄存器、计数器、译码器、比较器、半加器、全加器等。这些功能电路都是由基本的逻辑电路经过逻辑组合而成的,再把这些功能电路有机地集成起来,就可以组成一个完整的计算机硬件系统。

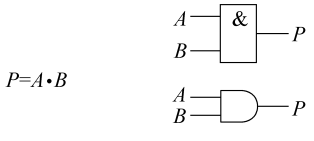
3.4.1 门

门(也称逻辑门)是对电信号执行基础运算的设备,是处理二进制数的基本电路,是构成数字电路的基本单元。一个门接收一个或多个输入信号,生成一个输出信号。由于门处理的是二进制数据,所以,每个门的输入和输出只能是0(对应低电平)或1(对应高电平)。

门的表示方法有3种:①逻辑表达式,即数学表示法;②逻辑框图,即图形符号表示法,本书中门的逻辑框图包含两种表示方式,上面的逻辑框图是国家标准局规定的符号,下面的逻辑框图是国际上通常采用的符号;③真值表,列出了所有可能的输入组合及其相应输出的表。

基本的门是与门、或门和非门,其他复杂的门都可以由这三种门组合而成。其他常用的门还有异或门、与非门和或非门。

与门具有逻辑乘法功能,只有当输入 A 和 B 同时为1时,输出 P 才为1,否则输出 P 为0。图3.11是与门的示意图。

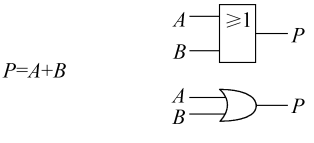


A	B	P
0	0	0
0	1	0
1	0	0
1	1	1

(a) 逻辑表达式　(b) 逻辑框图　(c) 真值表

图 3.11　与门示意图

或门具有逻辑加法功能,当输入 A 和 B 中有一个为1时,输出 P 就为1,否则输出 P 为0。图3.12是或门的示意图。

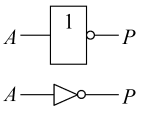


A	B	P
0	0	0
0	1	1
1	0	1
1	1	1

(a) 逻辑表达式　(b) 逻辑框图　(c) 真值表

图 3.12　或门示意图

非门具有逻辑取反功能,它只有一个输入和一个输出,当输入 A 为0时,输出 P 为1,当输入 A 为1时,输出 P 为0。图3.13是非门的示意图。



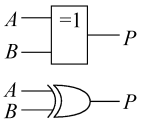
A	P
0	1
1	0

$P=\overline{A}$

(a) 逻辑表达式　　(b) 逻辑框图　　(c) 真值表

图 3.13　非门示意图

异或门表示仅当输入 A 和 B 相同时输出 P 为 0，否则输出 P 为 1。注意异或门和或门之间的区别，异或门是不可兼或，而或门是可兼或。图 3.14 是异或门的示意图。



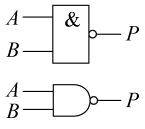
A	B	P
0	0	0
0	1	1
1	0	1
1	1	0

$P=A\oplus B$

(a) 逻辑表达式　　(b) 逻辑框图　　(c) 真值表

图 3.14　异或门示意图

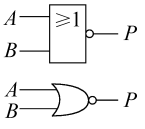
与非门和或非门分别是与门和或门的对立门，换言之，与非门是让与门的输出再经过一个非门，如图 3.15 所示，或非门是让或门的结果再经过一个非门，如图 3.16 所示。



$P=\overline{A\cdot B}$

(a) 逻辑表达式　　(b) 逻辑框图

图 3.15　与非门示意图



$P=\overline{A+B}$

(a) 逻辑表达式　　(b) 逻辑框图

图 3.16　或非门示意图

需要说明的是，门可以接收 3 个或更多的输入，其定义与具有两个输入的门是一致的。

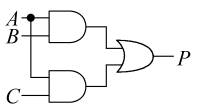
门为计算机的各种功能电路提供了构件。**电路**是由多个门组合而成的，可以执行算术运算、逻辑运算、存储数据等各种复杂操作。

电子计算机由具有各种逻辑功能的逻辑部件组成，这些逻辑部件按其结构可分为两大类：一类是组合电路，输入值明确决定了输出；另一类是时序电路，输出是输入值和电路现有状态的函数。有了组合电路和时序电路，再进行合理的设计，就可以表示和实现逻辑代数的基本运算。

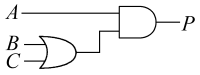
3.4.2　组合电路

把一个门的输出作为另一个门的输入，就可以把门组合成组合电路。在图 3.17(a)中，两个与门的输出被用作一个或门的输入(图中的连接点表示两条线是相连的)。在图 3.17(b)中，或门的输出被用作一个与门的输入。这两个不同的电路对应的真值表是相同的，如图 3.17(c)所示，即对于每个输入的组合，两个电路生成完全相同的输出。

各种算术运算可归结为相加和移位这两个最基本的操作，因而运算器以加法器为核心。对二进制数执行加法的电路称为**加法器**，功能较强的计算机具有专门的乘除部件和浮点运



(a) $P=A\cdot B+A\cdot C$



(b) $P=A\cdot(B+C)$

A	B	C	$A\cdot B$	$A\cdot C$	$B+C$	P
0	0	0	0	0	0	0
0	0	1	0	0	1	0
0	1	0	0	0	1	0
0	1	1	0	0	1	0
1	0	0	0	0	0	0
1	0	1	0	1	1	1
1	1	0	1	0	1	1
1	1	1	1	1	1	1

(c) 图(a)和图(b)中组合电路的真值表

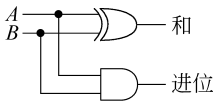
图 3.17　组合电路示例

算部件,这些部件都以加法器为核心,但又增加了一些移位逻辑和控制逻辑。

两个二进制数相加的结果可能产生进位值,计算两个 1 位二进制数的和并生成正确进位的电路称为**半加器**。两个 1 位二进制数相加的真值表如图 3.18(a)所示。注意得到的是两个输出——和与进位,所以,半加器电路应该有两个输出,并且和对应的是异或门,进位对应的是与门,如图 3.18(b)所示。

输入		输出	
A	B	和	进位
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

(a) 半加器的真值表



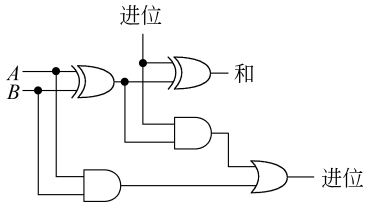
(b) 半加器的逻辑电路

图 3.18　半加器示意图

半加器没有把进位(即进位输入)考虑在计算之内,所以,半加器只能计算两个 1 位二进制数的和,而不能计算两个多位二进制数的和。考虑进位输入的电路称为**全加器**。可以用两个半加器构造一个全加器,把半加器的和再与进位输入相加,如图 3.19 所示。

输入			输出	
A	B	进位	和	进位
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

(a) 全加器的真值表



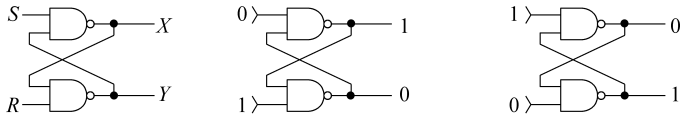
(b) 全加器的逻辑电路

图 3.19　全加器示意图

要实现两个 8 位的二进制数相加,只需要复制 8 次全加器电路,一位的进位输出将作为下一位的进位输入,最左边的进位输入是 0,最右边的进位输出作为溢出被舍弃。

3.4.3　时序电路

数字电路的一个重要作用是存储数据，其存储功能是由时序电路（也称存储器电路）实现的。存储器电路有很多种，图 3.20(a)为一个用与非门设计的 ***S-R* 锁存器**。



(a) *S-R*锁存器　(b) 初始化为1　(c) 初始化为0

图 3.20　*S-R* 锁存器工作原理

在图 3.20(a)中，每个与非门都有一个外部输入（S 或 R）和一个来自输出的输入（X 或 Y），如果输出 X 为 1，输出 Y 为 0，S 和 R 也都为 1，则输出 X 保持为 1。同理，如果输出 X 为 0，输出 Y 为 1，S 和 R 也都为 1，则输出 X 保持为 0。因此，无论输出 X 的值是什么，如果 S 和 R 都为 1，则电路就保持当前状态。输出 X 在任意时刻的值是这个电路存储的值。

那么，如何把一个值存入 *S-R* 锁存器呢？暂时把 S 置为 0，保持 R 为 1，如图 3.20(b)所示，可以把 *S-R* 锁存器设置为 1，然后将 S 恢复为 1，*S-R* 锁存器将保持 1 的状态。同理，暂时把 R 置为 0，保持 S 为 1，如图 3.20(c)所示，可以把 *S-R* 锁存器设置为 0，然后将 R 恢复为 1，*S-R* 锁存器将保持 0 的状态。因此，通过控制 S 和 R 的值，*S-R* 锁存器就可以实现存储功能。

一个 *S-R* 锁存器存储一位二进制数，把这个思想扩展，就可以设计出容量较大的存储器电路。

3.4.4　集成电路

集成电路（也称芯片）是嵌入了多个门的硅片，这些硅片被封装在塑料或陶瓷中，边缘有引脚，可以焊接在电路板上或插入合适的插槽中，每个引脚连接着一个门的输入或输出、电源或接地。

一个小规模集成电路芯片 SSI 只有几个独立的门，图 3.21 展示了一个具有 14 个引脚的 SSI 芯片，其中 8 个用作门的输入、4 个用作门的输出、1 个接地、1 个接电源。用不同的门可以制成类似的芯片。

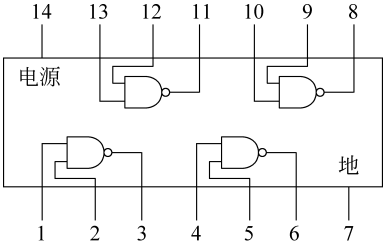


图 3.21　包含独立与非门的集成电路

超大规模集成电路 VLSI 的门数量超过 100 000 个，这是否意味着 VLSI 芯片需要有多于 300 000 个引脚？答案是 VLSI 芯片上的门不是完全独立的，VLSI 芯片上嵌入的电路具有很高的门引脚比。也就是说，许多门被组合在一起，创建的复杂电路只需要很少的输入和输出值。

计算机中最重要的集成电路莫过于中央处理器（CPU），每个 CPU 都有大量的引脚，这些引脚把 CPU 和存储器与输入/输出设备连接在一起，计算机系统的所有通信都是通过这些引脚完成的。

思考题

1. 计算机硬件在进行表达式计算时,通常要求操作数占用相同的二进制位数,并且要求存储方式也相同。例如,计算机硬件可以直接将两个16位整数相加,但是不能直接将一个16位整数和一个32位整数相加。解释其中的道理。

2. 微型计算机中CPU的引脚有多少个?还有哪些芯片带有引脚?各芯片引脚的个数是否应该有严格的规定?为什么?

3.5 计算机部件

冯·诺依曼计算机有5个基本部件:运算器、控制器、存储器、输入设备和输出设备,时至今日,所有的计算机都没有突破冯·诺依曼计算机的基本结构。

3.5.1 存储器

随着计算机系统的不断发展,计算机应用领域的日益扩大,对存储器的要求也越来越高,现代计算机已演变成以存储器为核心的计算机系统。

1. 存储器的层次结构

存储器的性能指标主要有3个:存储容量、存取速度和每位价格。**存储容量**指存储器可以容纳的二进制信息总量。显然,存储器的存储容量越大,存储的信息就越多。存储器的最小存储单位是**位**(bit,简称b),每一位可以存储一位二进制数,8位为1 **字节**(byte,简称B)。存储容量通常以字节为基本单位,由于计算机的存储容量一般都很大,所以,常见的单位也有千字节(KB)、兆字节(MB)、吉字节(GB)、太字节(TB)等,其换算关系如下:

$$1KB=2^{10}B=1024B$$
$$1MB=2^{20}B=1\ 048\ 576B$$
$$1GB=2^{30}B=1\ 073\ 741\ 824B$$
$$1TB=2^{40}B=1\ 099\ 511\ 627\ 776B$$

在存储器容量的度量单位中,前缀K、M、G、T的使用是有误差的,因为这些前缀在其他应用领域中已经用来作为10的幂的单位,例如km指的是1000m,MHz指的是1 000 000Hz,而这些前缀在计算机领域中是2的幂的单位。

存取速度可以用存取时间和存取周期两个参数来衡量,存取时间指从CPU发出有效存储地址从而启动一次存储器读/写操作,到读/写操作完成所经历的时间;存储周期指连续启动两次独立的存储器读/写操作所需的最小时间间隔。**每位价格**指存储器的价格与存储容量的比,一般地,设C是具有S位存储容量的存储器的价格,则每位价格$V=C/S$。

存储器的容量、速度和价格之间存在如下关系:存取时间越短则每位价格就越高;存储容量越大则每位价格就越低;存储容量越大则存取时间就越长。这就要求设计存储系统时需要在容量、速度和价格之间进行权衡。解决这个问题的方法是采用层次结构,即在一台计算机中,采用具有各种存取速度、存储容量和访问方式的存储器,这些存储器构成了一个层

次结构，如图 3.22 所示。

在“高速缓存-内存储器-外存储器”三级存储系统中，各级存储器承担的职能不同。高速缓存主要强调快速存取，以便使存取速度和处理器的运算速度相匹配；外存储器主要强调大的存储容量，以满足计算机大容量的存储要求；内存储器介于高速缓存和外存储器之间，要求选取适当的存储容量和存取速度，使它能容纳系统的核心软件和较多的用户程序。追求整个存储器系统具有更高的性能价格比是三级存储系统的核心思想。

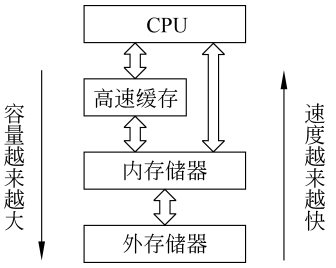


图 3.22　存储器的层次结构

2. 内存储器

CPU 的主要工作是执行程序，某一时刻 CPU 只能处理一条指令和几组数据，因此，计算机需要一个空间存储其余的指令和数据，等待 CPU 处理，这个存储空间就是内存储器。**内存储器也称为内存或主存**，它直接与 CPU 相连，存储容量较小，但存取速度较快，用于保存正在使用（或经常使用）的程序和数据。

内存储器由单独的、可编址的存储单元组成，**存储单元**是可管理的最小单位，典型的存储单元是单个字节，每个存储单元的编号称为**地址**。地址具有唯一标识存储单元的作用，一般从 0 开始连续编号，如图 3.23 所示。

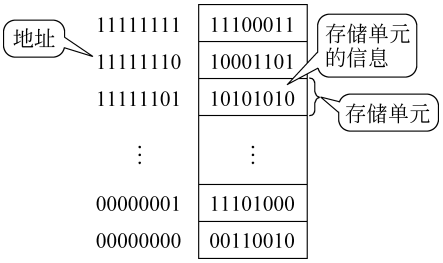


图 3.23　存储器示意图

可以将内存与宾馆的房间进行类比：位——床位；一个二进制位可以存储一个二进制数——一张床可以容纳一个人（假设男为 0 女为 1）；存储单元——房间；内存地址——房间号；内存容量——床位总数。

如果要访问存储器中某个存储单元的信息，就必须知道这个单元的地址，然后按地址存入或取出信息，CPU 对内存的一次存取通常在微秒级时间内完成。向存储器里存入信息也称为**写入**，写入的新内容覆盖了原来的旧内容；从存储器里取出信息也称为**读出**，信息读出后并不破坏原来存储的内容，因此，信息可以重复读出。需要强调的是：存储位不能是空的，必须存放 0 或 1，换言之，任意时刻存储单元的内容都不能是空的，一定是 0 和 1 的编码。

内存储器有两种：随机存储器 RAM 和只读存储器 ROM。RAM 芯片包含可以存储指令和数据的电路，存储在 RAM 里的数据只不过是硅片上流过微电路的电流，这意味着只要断电，存储在 RAM 里的信息就会丢失，因此，RAM 又称为易失性存储器。RAM 占了存储

器的绝大部分,是内存性能的决定性因素。ROM中存储的信息是不会丢失的,因此又称为非易失性存储器。一般情况下,ROM中的信息是固化的,是生产厂家写入的固定指令和数据,计算机只能从ROM中读取信息,而不能写进任何信息。例如BIOS程序是写在ROM中对计算机进行初始化的指令集合。

3. 外存储器

由于计算机的内存储器具有易失性,所以需要将数据和程序存储在永久性存储设备上。**外存储器也称为辅助存储器,或简称外存、辅存**,存储容量大,但只能和内存储器交换信息,在脱机状态下不能被计算机系统的其他部件直接访问。常用的辅助存储器有硬盘、光盘、U盘、移动硬盘、磁带等。

> 术语联机(on-line)和脱机(off-line)通常分别用来描述连接于计算机和没有连接于计算机的设备。联机意味着设备已经与计算机相连,不需要人的干预就可以使用;脱机意味着设备在被计算机使用之前需要人的干预——将这个设备接通电源,或者将这个设备与计算机相连接。

(1) 硬盘。硬盘由坚硬的合金盘片组成,存储容量大且存取速度较快,并且具有直接访问文件或记录的功能。

(2) 光盘。光盘是用激光在磁光介质上进行读写的存储器,能够存储大量的数据,主要有CD ROM、WORM CD和MO三种类型。CD ROM是只读存储器,它只能读取预先录制好的数据,不能改变其内容;WORM CD是一次写入多次读出存储器;MO是多次写入多次读出存储器,就像硬盘一样可以反复使用。

(3) U盘。U盘也称闪存盘,是一种可擦除的存储芯片,不仅具有可读可写的优点,而且所写入的数据在断电后也不会丢失。随着USB接口出现并逐步流行,U盘逐渐代替了软盘成为最热门的移动存储设备。

(4) 移动硬盘。移动硬盘通过一根电缆与计算机的USB接口连接,不需要重新启动计算机就可以连接或断开,实现完全的即插即用。

(5) 磁带。磁带采用顺序存取方式,如果要访问某一信息,必须从磁带的头部开始顺序检索,因而访问速度较慢,但是,磁带能以较低的成本高密度地存储大量数据。如今,磁带主要用在大型机上备份数据,或用于执行一些对时间要求不是很高的操作。

4. 高速缓冲存储器

计算机存取数据的时间主要取决于内存,据统计,CPU大约有70%的工作是对内存进行读写操作,因此,内存的存储容量和存取速度直接影响到系统的速度及整机性能。目前,随着硬件制造水平不断提高,计算机的内存容量越来越大,速度越来越快,但内存的存取速度与CPU的处理速度相比仍有很大差距。为了使较慢速的内存与高速的CPU相匹配,现代计算机系统大多采用了高速缓冲存储技术。

高速缓冲存储器(cache,简称缓存)介于内存和CPU之间,位置可以在CPU芯片的内部,也可以在CPU芯片的外部。它的存取速度比内存快,但价格昂贵,所以存储容量较小,主要用来存放当前内存中即将使用(或使用最多)的程序块和数据块,并以接近CPU的速度向CPU提供程序指令和数据。