第 1 章　计算机基础知识

本章是为非计算机专业未学过计算机原理的学生编写的，内容为计算机方面最基本的基础知识。

1.1　计算机的发展概况

一、计算机的发展历史

电子计算机是一种能够高速而精确地进行各种数据处理的机器，这是人类生产和科学技术发展的产物，它的出现又有力地推动了生产力的发展，目前计算机已应用到国民经济的各个领域，当代社会已离不开计算机。自从计算机诞生以来，它的发展经历了四代：

第一代计算机(20 世纪 40 年代末期至 20 世纪 50 年代末期）是电子管计算机，所使用的元件主要是电子管。世界上第一台电子计算机是由美国宾夕法尼亚大学的 J. W. Mauchly 和 J. P. Eckert 研制成的 ENIAC 计算机,这台计算机用了 18800 只电子管，加法速度是每秒 5000 次,乘法速度是每秒 56 次。

第二代计算机(20 世纪 50 年代末期至 20 世纪 60 年代末期)是晶体管计算机,所用的主要元件是晶体管。1957 年，美国研制成了第一台晶体管计算机(TRANSACS-1000 机)。

第三代计算机(20 世纪 60 年代中期至 20 世纪 70 年代初）是集成电路计算机，所用的元器件是小规模或中规模集成电路，如 IBM 公司于 1964 年推出的 IBM360 系列机。

第四代计算机(20 世纪 70 年代初期开始）是大规模集成电路计算机。所使用的元件是大规模或超大规模集成电路。 1971 年 IBM370 系列机首先使用了大规模集成电路构成主存贮器。1975 年研制成功了以大规模集成电路做主存贮器和逻辑元件的大型计算机，例如 470V/6 型 M-190 机等。

当前计算机的发展趋势是微型化、网络化和智能模拟。

二、计算机的种类

计算机有模拟计算机和数字计算机两种。

模拟计算机 (analog computer ）是对模拟量进行操作的计算机，这种计算机解题速度快,但精度差。

数字计算机 (digital computer ）是对数据进行算术和逻辑操作的计算机，这种计算机在运算过程中全部自动化，具有运算速度快、精度高、通用性强等特点。

现在我们通常所说的电子计算机，实际上是指电子数字计算机。

电子计算机的种类很多。根据设计的目标来分，有通用计算机和专用计算机；根据用途

来分，有科学计算、数据处理和工业控制计算机；若根据规模和功能来分，有巨型机、大型机、中型机、小型机、超小型机和微型机。

微型计算机又可以划分为多片大规模集成电路构成的通用微型机（如 PC 机）和单片大规模集成电路为主组成的微型计算机（简称单片机）。

1.2 计算机的系统组成

电子计算机是模仿人脑部分功能的一种工具，俗称为“电脑”。它的结构特点与工作过程和人脑有许多相似之处，电子计算机的工作原理模拟人手工计算的过程。试看一下人用算盘来计算(2436 + 3748 − 4569)的过程，如果我们把算盘记为 R，则计算过程如表 1-1 所示。

表 1-1 使用算盘解题过程

序　号	操 作 命 令	注　释
0	0→R	清除算盘盘面
1	2436→R	在算盘上拨上 2436
2	(R)+3748→R	在算盘中加上 3748
3	(R)−4569→R	在算盘中减去 4569
4	(R)抄送纸上	抄送运算结果
5	停止	计算结束

在执行六步操作以后，在算盘上存放着运算结果 1615。表 1-1 是用算盘求解过程的形式描述，算盘 R 具有累加的作用。改变表中的操作命令，就可以实现其它问题的计算。

计算过程中的每一步都是一条指示人完成相应操作的命令，我们将这种执行某种操作的命令称为指令，完成某种功能的一组指令系列称之为程序。表 1-1 列出的六条指令就构成了一个程序，编制解题程序的过程称为程序设计。

在手工计算中，是由使用算盘的人按写在纸上的程序和数据来进行计算的，因此使用算盘的人、算盘、记录数据和程序的纸是计算过程中必不可少的组成部分。

若用电子计算机模拟上述解题过程，计算机必须具备下列条件：

(1) 为了能进行各种数据运算，机器内必须有一个相当于算盘的运算器。

(2) 为了保存和记录原始数据、解题程序和运算的中间结果，机器内必须有容量足够大的存贮器，这相当于手工计算时用的纸张。

(3) 必须有按照解题程序指挥、控制各个部件协同工作的控制器，这相当于手工计算中的人脑。

(4)必须具备将原始数据和程序送入机器内部的输入设备和给出计算结果的输出设备。

(5) 机器内应有必要的程序，以便开机后执行该程序，启动系统工作，自动地投入运行状态。

一个计算机系统由硬件和软件两部分组成。硬件指运算器、控制器、存贮器和输入/输出

设备。其中运算器和控制器是计算机硬件的核心，称为中央处理器 CPU（central processing unit）

计算机的硬件结构如图 1-1 所示。

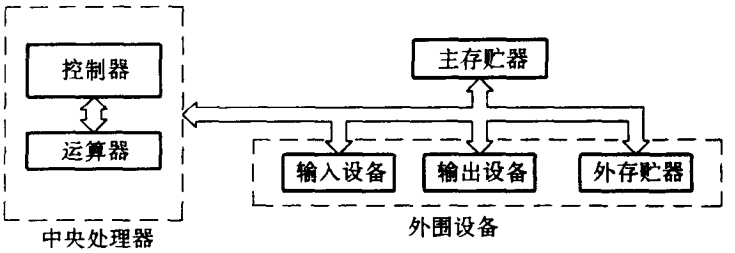
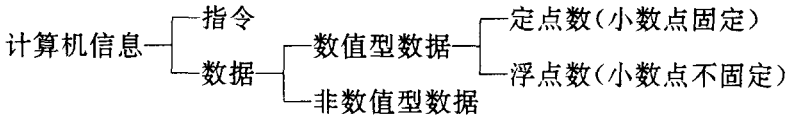


图 1-1 电子计算机硬件结构

计算机系统中的各类程序及文件统称为软件。它包括使系统自动工作或提高计算机工作效率的系统软件和实现某一应用目标的应用软件。软件是计算机系统工作的“灵魂”。

计算机的工作也可以认为是信息加工过程。计算机中的信息是指数据或指令，它们是以一定的编码形式表示的，其意义各不相同，大致可分为：



1.3 数制及其转换

一、进位计数制

进位计数制可概括如下：

● 有一个固定的基数 r，数的每一位只能取 r 个不同的数字，即符号集是 {0，1，2，…，r－1}；

● 逢 r 进位，它的第 i 个数位对应于一个固定的值 r^i，r^i 称为该位的“权”。小数点左面各位的权是基数 r 的正次幂，依次为 0，1，2，…，m 次幂，小数点右面各位的权是基数 r 的负次幂，依次为 －1，－2，…，－n 次幂。

以下我们用 $(\quad)_r$ 表示括号内的数是 r 进制数。将 r 进制数（$a_m a_{m-1}\cdots a_1 a_0 \cdot a_{-1} a_{-2}\cdots a_{-n}$）按权展开，表达式为：

$$a_m \times r^m + a_{m-1} \times r^{m-1} + \cdots + a_1 \times r^1 + a_0 \times r^0 + a_{-1} \times r^{-1} + a_{-2} \times r^{-2} + \cdots + a_{-n} r^{-n}$$

1. 十 进 制 数

十进制数的基数：r＝10，符号集为{0，1，2，3，4，5，6，7，8，9}，其权为：…，10^2，10^1，10^0，10^{-1}，10^{-2}，…。

例 1 $(987.32)_{10} = 9 \times 10^2 + 8 \times 10^1 + 7 \times 10^0 + 3 \times 10^{-1} + 2 \times 10^{-2}$

2. 八 进 制 数

八进制数的基数 $r=8$**,符号集为** $\{0,1,2,3,4,5,6,7\}$**,其权为**：…，8^2，8^1，8^0，8^{-1}，8^{-2}，…。

例 2 $(7061.304)_8=7\times8^3+0\times8^2+6\times8^1+1\times8^0+3\times8^{-1}+0\times8^{-2}+4\times8^{-3}$

3. 十六进制数

十六进制数的基数 $r=16$**,符号集为** $\{0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F\}$**,其权为**：…，$16^2$，$16^1$，$16^0$，$16^{-1}$，$16^{-2}$ …。

例 3 $(-A0.8F)_{16}=-(10\times16^1+0\times16^0+8\times16^{-1}+15\times16^{-2})$

4. 二 进 制 数

二进制数的基数 $r=2$**,符号集为** $\{0,1\}$**,权为**…，2^2，2^1，2^0，2^{-1}，2^{-2}，…。

例 4 $(1011.101)_2=1\times2^3+0\times2^2+1\times2^1+1\times2^0+1\times2^{-1}+0\times2^{-2}+1\times2^{-3}$

十进制、二进制、八进制和十六进制数码对照见表 1-2，**二进制与十进制小数对照见表** 1-3。

表 1-2 十进制、二进制、八进制、十六进制数码对照表

十进制	二进制	八进制	十六进制	十进制	二进制	八进制	十六进制
0	0000	00	0	8	1000	10	8
1	0001	01	1	9	1001	11	9
2	0010	02	2	10	1010	12	A
3	0011	03	3	11	1011	13	B
4	0100	04	4	12	1100	14	C
5	0101	05	5	13	1101	15	D
6	0110	06	6	14	1110	16	E
7	0111	07	7	15	1111	17	F

表 1-3 二进制与十进制小数对照表

二进制小数	十进制小数	二进制小数	十进制小数
0.1	0.5	0.00001	0.03125
0.01	0.25	0.000001	0.015625
0.001	0.125	⋮	⋮
0.0001	0.0625		

二、二进制编码的十进制数

常用二进制编码的十进制数有 8421BCD **码(简称** BCD **码)、**2421 **码、**5211 **码和余** 3 **码等。它们都是用** 4 **位二进制数来表示** 1 **位十进制数。前三种码都是有权码,余** 3 **码为无权码,**

而 2421 码和 5211 码表示的十进制数不唯一，8421BCD 码和余 3 码唯一地表示一个十进制数。这四种编码的关系如表 1-4 所示。

表 1-4　四种编码的关系

8421BCD 码	2421 码	5211 码	余 3 码
0000	0000(或 0000)	0000(或 0000)	0011
0001	0001(或 0001)	0001(或 0010)	0100
0010	0010(或 1000)	0011(或 0100)	0101
0011	0011(或 1001)	0101(或 0110)	0110
0100	0100(或 1010)	0111(或 0111)	0111
0101	1011(或 0101)	1000(或 1000)	1000
0110	1100(或 0110)	1010(或 1001)	1001
0111	1101(或 0111)	1100(或 1011)	1010
1000	1110(或 1110)	1110(或 1101)	1011
1001	1111(或 1111)	1111(或 1111)	1100

三、进位计数制之间的转换

不同基的进位计数制之间数的转换，一般有下面几种方法。

1. 直接相乘法

将表示成 r 进制数的 M 转换为 t 进制数。即基数 r 用基数 t 来表示，M 的各位数字用 t 进制的数系来表示，然后作乘法和加法，结果便是 t 进制数。

例 5　把十进制数 725 转换为二进制数。

$$\begin{aligned}(725)_{10} &= 7\times10^2+2\times10^1+5\times10^0\\ &= 111\times1010^2+10\times1010^1+101\times1010^0\\ &= (1011010101)_2\end{aligned}$$

2. 余数法（适合于整数部分转换）

将表示成 r 进制的整数 M 转换为 t 进制数的整数，除以 t 取余法。

例 6　把十进制数 62 转换为 2 进制数。

$$\begin{array}{r|l} 2 & 62 \cdots\cdots \text{余数}=0 \\ 2 & 31 \cdots\cdots \text{余数}=1 \\ 2 & 15 \cdots\cdots \text{余数}=1 \\ 2 & 7 \cdots\cdots \text{余数}=1 \\ 2 & 3 \cdots\cdots \text{余数}=1 \\ & 1 \cdots\cdots \text{余数}=1 \end{array}$$

低位 ↑ 高位

结果：$(62)_{10}=(111110)_2$

3. 取整法（适用于小数部分转换）

将r进制数的小数转换为t进制的小数，乘t取整法。

例7　把十进制小数0.375转换为二进制数。

$$
\begin{aligned}
&0.375 \times 2 = 0.750\cdots\cdots \text{整数} = 0 \qquad \text{高位}\\
&0.75 \times 2 = 1.50 \quad \cdots\cdots \text{整数} = 1 \qquad \downarrow\\
&0.50 \times 2 = 1.00 \qquad \text{整数} = 1\\
&(0.375)_{10} = (0.011)_2 \qquad \text{低位}
\end{aligned}
$$

注意 将r进制小数转换为t进制小数时，有时会是无限循环小数，这时可根据要求进行取舍。

4. 递归法（适合于计算机转换）

把r进制数M转换为t进制数，其方法是拆成整数和小数两个部分，然后把用递归算法产生的已转换成t进制数的整数和小数部分拼起来：

例8　将十进制数4827.625转换为二进制数。

$$
\begin{aligned}
(4827)_{10} &= (((4 \times 10 + 8) \times 10 + 2) \times 10 + 7) \times 10^0\\
&= ((100 \times 1010 + 1000) \times 1010 + 10 \times 1010) + 111\\
&= (1001011011011)_2\\
(0.625)_{10} &= (6 + (2 + 5 \times 10^{-1}) \times 10^{-1}) \times 10^{-1}\\
&= (110 + (10 + 101 \times 1010^{-1}) \times 1010^{-1}) \times 1010^{-1}\\
&\approx (0.101)_2
\end{aligned}
$$

结果：$(4827.625)_{10} = (1001011011011)_2 + (0.101)_2$

$= (1001011011011.101)_2$

1.4　计算机中数的表示方法

一、真值和机器数

一个数是由符号和数值两部分组成的。例如：

$$N_1 = +\ 1001010\ (+\ 74)$$
$$N_2 = -\ 1001010\ (-\ 74)$$

在计算机中数的符号也是用二进制码表示的，一般正数的符号用‘0’表示，负数的符号用‘1’表示。例如：

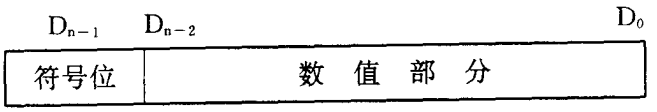
$$N_1 = 01001010\ (+\ 74)$$
$$N_2 = 11001010\ (-\ 74)$$

一个数在机器中的表示形式称为机器数，而把这个数本身称为真值。

二、带符号数的表示方法

上面提到的机器数表示方法，用‘0’表示正，用‘1’表示负。这种表示数的方法，称为带符

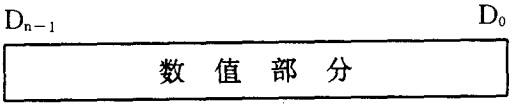
号数的表示方法。在机器中的一般表示形式为：



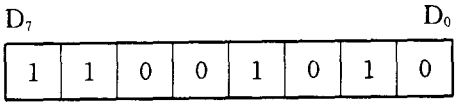
机器数最高位为符号位，其余的（n－1）位为数值部分。

三、无符号数的表示方法

无符号数没有符号位，机器的全部有效位都用来表示数的大小。无符号数在机器中的一般形式为：



例如：



（即 202）

四、数的定点和浮点表示

十进制数 485.23 也可以表示为 0.48523×10^3 ，而在计算机内也有类似的两种数的表示方法，那就是定点数和浮点数。

1. 定点表示法

计算机内的定点数格式为：

小数点固定在数值部分的最高位之前或最低位之后。

2. 浮点表示方法

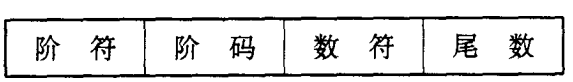
浮点数格式

浮点表示法即指小数点的位置是不固定的，而是浮动的。例如：$N_1 = 2^1 \times 0.1011$ 和 $N_2 = 2^3 \times 0.1011$，这两个数的有效数字相同，但小数点的位置不一样。对于任何一个二进制数 N 都可以表示为：

$$N = \pm m \times 2^{\pm e}$$

其中 $m \geqslant 0$ 称为 N 的尾数；m 前面的符号称为数符；e 称为 N 的阶码，为非负整数，其前面的符号称为阶符（阶码和阶符决定 N 的小数点位置）。

计算机内浮点数格式为：



规格化数

由于一个数的浮点表示不是唯一的，为了使数据的有效位数最大，并使运算的精度尽可

能高，计算机的浮点数采用规格化浮点数表示。规格化浮点数定义如下：

若 $N = \pm m \times 2^{\pm e}$，则

$$\frac{1}{2} \leqslant m < 1$$

五、原码、补码和反码

原码、补码和反码都是带符号数在机器中的表示方法。在介绍这三种编码方法之前，先介绍模的概念和性质。

我们把一个计量器的容量，称为模或模数，记为 M 或 mod M。例如：一个 n 位二进制计数器，它的容量为 2^n，所以它的模为 2^n（即可表示 2^n 个不同的数）又如 时钟可表示 12 个钟点，它的模为 12。

模具有这样的性质，当模为 2^n 时，2^n 和 0 表示形式是相同的。例如：一个 n 位二进制计数器，可以从 0 计数到 2^{n-1}，如果再加 1，计数器就变成了零。所以，2^n 和 0 在 n 位计数器中的表示形式是一样的。同样，时钟的 0 点和 12 点在钟表上的表示形式是相同的。

1. 原　码

前面介绍的带符号数在机器中的表示方法，实际上就是原码表示法。原码表示方法是最简单的一种表示方法，只要把真值的符号部分用 0 或 1 表示即可。例如：

$$N_1 = +1001010$$
$$N_2 = -1001010$$

其原码记为：

$$[N_1]_{原} = 01001010$$
$$[N_2]_{原} = 11001010$$

由上述原码的表示形式，可将原码定义为：

$$[X]_{原} = \begin{cases} 2^n + X & 0 \leqslant X < 2^{n-1} \\ 2^{n-1} + X & -2^{n-1} < X \leqslant 0 \end{cases}$$

其中 X 为真值的 $(n-1)$ 位绝对值，n 为机器可表示的二进制码位数。

在原码表示中，“0”有两种表示形式（机器 0）：

$$[+0]_{原} = \underbrace{00\cdots0}_{n个“0”} \quad (\mathrm{mod}\, 2^n)$$
$$[-0]_{原} = \underbrace{100\cdots0}_{n-1个“0”} \quad (\mathrm{mod}\, 2^n)$$

2. 补　码

我们首先介绍同余的概念，然后从同余概念导出补码的概念，进而给出补码的定义和性质。

如果有两个整数 a 和 b，当用某一个正整数 M 去除所得余数相等时，则称 a 和 b 对模 M 是同余的。

当 a 和 b 对 M 同余时，就称 a、b 在以 M 为模时是相等的，记为：

$$a = b \pmod{M}$$

例如：$a = 16$，$b = 4$，若模为 12，则 16 和 4 在以 12 为模时是同余的：

$$16 = 4 \pmod{12}$$

事实上 16 点和 4 点在以 12 为模的钟表上指示是一样的。

由同余的概念可以得出：

$$M + a = a \pmod{M}$$
$$2M + a = a \pmod{M}$$

因此，当 a 为负数时，如 $a = -3$，在以 10 为模时，有

$$10 + (-3) = -3 \pmod{10}$$
$$12 + (-3) = -3 \pmod{12}$$

钟表上的 9 点可以看成为到 12 点缺 3 个小时。

这样，以 10 为模时，负数(-3)可以转化为正数（+ 7）了。这时我们说，当以 10 为模时，“ -3 ”的补码为“ 7”，同理“ -2 ”的补码为“ 8”。

在计算机中，可以表示的二进制码位数是一定的，如果是 n 位，那么它的模是 2^n，2^n 和 0 在机器中的表示形式是完全一样的。以 2^n 为模也称为以 2 为模。

如果 n 位二进制码的最高位表示符号位，则补码的表示形式为：

● $X = +X_{n-2}X_{n-1}\cdots X_1X_0$ 时：

$$[X]_{补} = 2^n + X = 0X_{n-2}X_{n-3}\cdots X_1X_0 \pmod{2^n}$$

● $X = -X_{n-2}X_{n-3}\cdots X_1X_0$ 时：

$$[X]_{补} = 2^n + X = 2^{n-1} + 2^{n-1} - X_{n-2}X_{n-3}\cdots X_1X_0$$
$$= 1\overline{X}_{n-2}\overline{X}_{n-3}\cdots\overline{X}_1\overline{X}_0 + 1 \pmod{2^n}$$

综上所述，X 的补码可定义为：

$$[X]_{补} = 2^n + X$$

当 X 为正数时 [X]补码为 X 的区别只是符号位用零代替，当 X 为负数时，从 2^n 中减去 X 的绝对值。特殊地，X 为纯小数时，即 $X = \pm 0.X_{-1}X_{-2}\cdots X_{-n-1}$，补码可表示为：

$$[X]_{补} = \begin{cases} X & 1 > X \geqslant 0 \\ 2 + X & 0 > X \geqslant -1 \end{cases}$$

补码具有下列性质：

$$[X + Y]_{补} = [X]_{补} + [Y]_{补}$$
$$[X - Y]_{补} = [X]_{补} + [-Y]_{补}$$

请读者根据补码的定义加以证明。

3. 反　码

在补码表示法中已提到负数的补码可以通过对原码（除符号位外），各位求反后加“ 1”得到，如果只求反不加 1，就得到另一种机器数的表示方法——反码表示法。因此，反码定义

如下：

$$[X]_{反}=\begin{cases}2^n+X & 0\leqslant X<2^{n-1}\\ (2^n-1)+X & -2^{n-1}<X\leqslant 0\end{cases}$$

从定义可看出，X 为正数时$[X]_{反}$与 X 的差别只是用零代替符号位。 X为负数时，用“1”代替负号位，其它各位求反。

1.5 指令和指令系统

指令是指示计算机执行某种操作的命令，指令是以一组二进制码表示的，称为机器指令。计算机只能识别和执行机器指令。在计算机中，指令是依次地存贮于存贮器中的，这部分存贮器常称之为程序存贮器。

指令的编码规则称为指令格式，一条指令的二进制码位数称为指令的长度，不同类型的计算机，指令的长度和格式是不一样的，所能执行的指令类型和数目也不同，通常把一台计算机所能执行的全部指令的集合称为指令系统。

一、指令格式

指令的具体格式依赖于计算机的结构特征，但指令的组成是一样的，都包含操作码和操作数两个部分。指令的一般格式为：

操作码　　操作数

操作码用来表示执行什么样的操作，如加法、减法等。操作码的位数取决于一台计算机的指令系统中指令的条数。例如：对于 32条指令的指令系统，操作码为 5 位 若指令系统中有 N 条指令，操作码的位数为 n，则有关系式：

$$N\leqslant 2^n$$

操作数用以指出参加操作的数据或数据的存贮地址。

不同类型的指令，操作数的个数是不一样的。在具有多个操作数的指令中，把它们分别称为第一操作数、第二操作数等。例如：加法指令，把两个数 a 和 b 相加，a 和 b就是参加操作的两个操作数。对于加法等操作，有些计算机指令还指出存放操作结果的地址，另外一些计算机把运算结果总是存放在某一个寄存器中（称为累加器，见运算器一节）。

若指令的操作数字段内容就是一个参加操作的数据，这种操作数称为立即数。大多数的指令，操作数存放于寄存器或存贮器中（这部分存贮器常称为数据存贮器），指令的操作数字段仅指出操作数所在的寄存器或存贮器地址。

存贮器由若干单元组成，每个单元都有一个相应的地址（这类似于一幢大楼，有若干房间，每个房间都有房间号），计算机是根据地址对存贮器存取数据的。存贮器的地址也是由一组二进制码表示的，地址的位数依赖于存贮器空间的大小。若存贮器共有 M 个单元，地址有 m 位，则有：

$$2^m\geqslant M$$

产生操作数地址的方式称为寻址方式，一般有下列几种：

(1) 立即寻址：操作数字段的内容就是参加操作的数据。

(2) 直接寻址：操作数本身就是一个有效地址，即操作数字段的内容就是参加操作的数据所在的存贮器单元地址。

(3) 间接寻址：操作数字段的内容给出某个单元的地址，该地址单元的内容才是操作数的地址。

(4) 变址寻址：操作数给出一个位移量 D 和一个变址器号 X ,将变址器的内容和位移量相加，即(X) + D 作为有效地址；

(5) 相对寻址：操作数给出一个相对数 X ，把它和一个基地址或现行指令地址相加，得到的和作为有效地址。

二 、 指 令 类 型

一台计算机的功能在很大程度上依赖于它的指令系统，指令系统中指令越丰富，指令的功能越强，则程序设计越方便，速度越快，效率也越高。按照指令的功能，大致上可以分成以下几种类型：

(1)数据传送指令：其功能是在计算机的各个部件之间传送数据，把一处的数据复制到另一处，如存贮器和寄存器之间、寄存器和寄存器之间的数据传送。

(2)算术指令 其功能是进行算术运算,如加、减、乘、除等。

(3)逻辑指令 其功能是进行逻辑运算,如逻辑与、逻辑或、移位等。

(4) 程序控制指令：其功能是改变和控制程序中指令执行的顺序。

(5)输入/输出指令：其功能是完成 CPU、存贮器和外部设备之间的数据传输。

(6) 其它指令：这是控制机器运行状态的指令，如停机指令、等待和空操作指令等。

三、字和　字长

如前所述，计算机中的数据和指令都是一组二进制编码，它们是作为一个整体来进行处理和运算的,统称为‘ 机器字 ”,简称字。一个机器字所包含的二进制码位数称为字长,更确切地说,字长是指 CPU 一次可处理（如数据传送、数据运算等）的二进制数的位数。计算机的字长和存贮器单元、运算器中各部件的位数相一致。

机器字的位数越多，它所表示的数据有效位数也越多，精度也越高，运算的误差也越小。在运算速度一定的情况下“ 字长 ’长的计算机,处理数据的速度也高。

字长是衡量计算机性能的一个重要指标，为了便于处理，计算机的字长为字节的整数倍,一个字节为 8 位二进制码。根据字长分类，计算机又可以分为 8 位机、16 位机、32 位机和 64 位机等。

1.6 存　贮　器

计算机之所以能够快速自动地进行各种复杂的运算，就是因为事先把解题的程序和数据存放在存贮器中，在运算过程中再由存贮器快速地提供给 CPU 进行运算，这就是所谓

“程序存贮工作方式”。所以，存贮器的职能是存贮程序和数据。

一、存贮器的主要技术指标

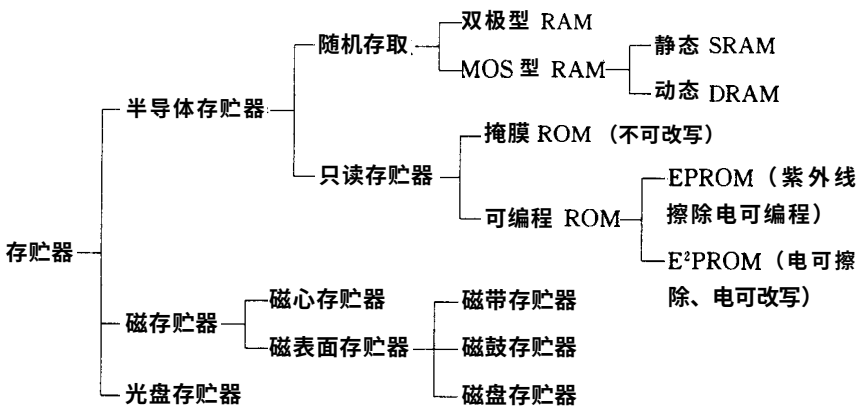
存贮器的主要技术指标有 存贮容量、体积、存取时间、功耗、可靠性和成本等。

存取时间是指将信息存入存贮器或从存贮器中读出信息所需要的时间，存取时间的长短将影响计算机系统的工作速度。

存贮器容量是指它能够存贮的信息量。计算机的存贮器一般以存贮单元组织的，每个存贮单元存放一个机器字。但为了统一起见，存贮器的容量一般以它所能存贮的字节（可存 8 位二进制数)数表示,如 2K 字节，4K 字节，…，256K 字节，512K 字节等。

二、存贮器的种类

根据存贮器同 CPU 的关系，可分为主存贮器（也称内存）和辅助存贮器（也称外存）两种。根据所使用的材料不同，常用的存贮器分类如下：



由于半导体存贮器具有速度快、体积小、集成度高、成本低等一系列的优点，所以现代的计算机的主存贮器普遍地使用半导体存贮器，而外存贮器一般都选用存贮密度高、容量大的磁盘存贮器和光盘存贮器。

三、半导体存贮器结构

图 1-2 给出了半导体存贮器结构的一般形式。

半导体存贮器由地址寄存器、地址译码驱动器、存贮矩阵、数据寄存器、读/写时序控制逻辑等部分组成。

1. 地址寄存器和地址译码驱动器

地址寄存器接收 CPU 从地址总线送来的地址 $A_0 \sim A_i$，地址的位数和存贮器的容量有关,如 2KB 的存贮器地址线为 11 位 $(i = 10)$，地址经过译码驱动器选中存贮器中相应的一个存贮单元。

2. 存 贮 矩 阵

存贮矩阵也称为存贮体，它是由许多能存贮二进制信息的存贮单元组成的，每个存贮单元由若干位组成，每位实际上是一个双稳态触发器（也称寄存器）。

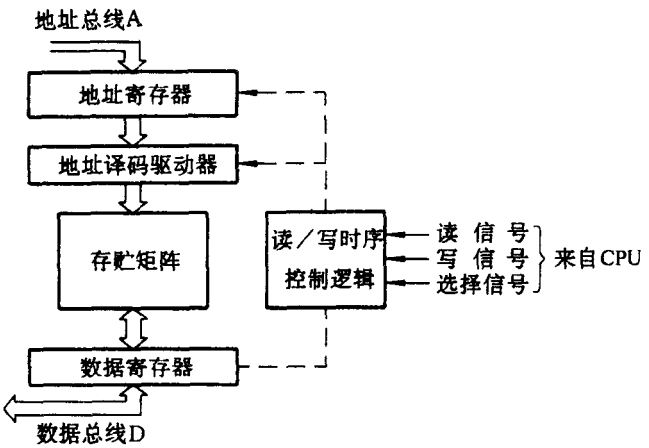


图 1-2　半导体存贮器的结构

3. 数据寄存器

数据寄存器用于存放从存贮单元中读出的信息并把它送到数据总线，也用于接收 CPU 在数据总线上送来的数据信息并把它存入相应的存贮单元中。

4. 读/写时序控制逻辑

该模块接收 CPU 送来的读信号、写信号和选择信号，由这三个信号来控制将信息写入相应的存贮单元，或从选中的存贮单元中读出信息。

5. 地址总线和数据总线

这是存贮器和 CPU 之间的信息传输线,分别用于传送地址和数据(详见 1.10 节总线)。

1.7 运　算　器

计算机的数据运算可以归纳为两类：一类是算术运算，另一类是逻辑运算。运算器是计算机进行算术和逻辑运算的功能部件。

一、算术运算

算术运算包括加、减、乘、除四则运算。

1. 加法和减法运算

运算方法和数的表示形式有关，在计算机中，最常用的是补码，补码的加减法运算最简单，符号位可以和数值位一样参加运算。因为

$$\begin{aligned}[X]_{补}+[Y]_{补} &= 2^n + X + 2^n + Y \\ &= 2^n + (X+Y) \\ &= [X+Y]_{补}\end{aligned}$$

所以有

$$[X+Y]_{补} = [X]_{补} + [Y]_{补}$$

例 9 若 $[X]_{补} = 10111\ (-9)$

$[Y]_{补} = 11110\ (-2)$

则

$$\begin{array}{rl} & [X]_{补} = 10111 \quad (-9) \\ +) & [Y]_{补} = 11110 \quad (-2) \\ \hline & [X]_{补} + [Y]_{补} = 10101 \quad (-11) \end{array}$$

所以 $[X+Y]_{补} = 10101 \quad (\mathrm{mod}\ 2^5)$

若用真值表示： $X + Y = (-9) + (-2) = -11$

又

$$\begin{aligned} [X]_{补} - [Y]_{补} &= [X]_{补} + [-Y]_{补} = 2^n + X + 2^n + (-Y) \\ &= 2^n + X + (-Y) \\ &= [X-Y]_{补} \end{aligned}$$

例 10 $[X]_{补} = 10111,\ [Y]_{补} = 11110,\ [-Y]_{补} = 00010$

$$\begin{array}{rl} & [X]_{补} = 10111\ (-9) \\ +) & [-Y]_{补} = 00010\ (+2) \\ \hline & [X]_{补} + [-Y]_{补} = 11011\ (-7) \end{array}$$

$$[X-Y]_{补} = [X]_{补} + [-Y]_{补} = 11011\ (-7,\ \mathrm{mod}\ 2^5)$$

2. 乘　法

乘法运算包括符号运算和数值运算。相同符号两数相乘之积为正，符号相异的两数相乘之积为负数。

数值运算是对两个数的绝对值相乘，它们可以被看作无符号的两个数相乘。

例 11 1011×1101

$$\begin{array}{rll} & 1011 & \text{被乘数} \\ \times) & 1101 & \text{乘数} \\ \hline & 1011 & \left.\begin{array}{l}\\\\\\\end{array}\right\} \text{部分积} \\ & 0000 & \\ & 1011 & \\ +) & 1011 & \\ \hline & 10001111 & \text{乘积} \end{array}$$

可见，两个 n 位无符号数相乘，乘积的位数为 2n，乘积等于各部分积之和。由乘数从低位到高位逐位去乘被乘数，当乘数的相应位为 1 时，则该次部分积等于被乘数 乘数相应位为 0 时，部分积为 0。从低位至高位被乘数逐次左移一位，加在左下方，在乘数的相应位为 0 时加 0.

我们也可以首先用乘数的高位去乘被乘数、求部分积、右移相加，其结果也一样。

3. 除　法

除法运算也包括符号运算和数值运算。两个同符号数相除，商为正数；异号的两数相除，商为负数。

数值运算是对两个数的绝对值相除。

例 12 011010 ÷ 101

```
                    101      商
除数 101  √011010            被除数
        −) 101
        ------
          00110              部分余数
         −)101
        ------
            001              余数
```

从上例可见，商数是一位位求得的，首先将除数和被除数的高 n 位比较，如果除数小于被除数的高 n 位，商为 1，然后从被除数数中减去除数，从而得到部分余数否则商为 0。重复上述过程，将除数和新的部分余数（即改变了的被除数）进行比较，直至被除数所有的位都处理完为止，最后便得到商和余数。

由于减法可通过补码加法实现，所以加减乘除四则运算都可以用加法运算来代替。

二、逻辑运算

基本的逻辑运算有下面三种。

1. 逻辑或运算

逻辑或运算也称为逻辑加，用符号“∨”或“+”表示。函数关系为：

$$C = A \vee B$$

其中 $A = a_{n-1}a_{n-2}\cdots a_1a_0$，$B = b_{n-1}b_{n-2}\cdots b_1b_0$，$C = c_{n-1}c_{n-2}\cdots c_1c_0$。

a_i	b_i	c_i	$i = 0 \sim n-1$
0	0	0	
0	1	1	
1	0	1	
1	1	1	

用以实现或运算的逻辑电路称为或门，其符号如图 1-3 所示。

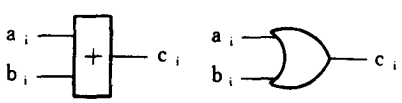


图 1-3 或门的逻辑符号

2. 逻辑与运算

逻辑与运算也称为逻辑乘，运算符号为“∧”或“·”。函数关系为：

$$C = A \wedge B$$

其中 $A = a_{n-1}a_{n-2}\cdots a_1a_0$，$B = b_{n-1}b_{n-2}\cdots b_1b_0$，$C = c_{n-1}c_{n-2}\cdots c_1c_0$。

a_i	b_i	c_i
0	0	0
0	1	0
1	0	0
1	1	1

$i = 0 \sim n-1$

用以实现与运算的逻辑电路称为与门，其符号如图 1-4 所示。

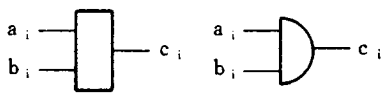


图 1-4　与门的逻辑符号

3. 逻辑非运算

逻辑非运算又称为逻辑否定。如有变量 A，A 的上面加一横 $\overline{A}$ 表示 A 的逻辑非。函数关系为：

$$C = \overline{A}$$

其中 $A = a_{n-1}a_{n-2}\cdots a_1a_0$，$C = c_{n-1}c_{n-2}\cdots c_1c_0$。

a_i	c_i
0	1
1	0

$i = 0 \sim n-1$

实现非运算的逻辑电路称为非门，逻辑符号如图 1-5 所示。

逻辑运算除上述三种基本运算以外，还有逻辑异或运算和逻辑同或运算。

逻辑异或运算也称按位加或称半加，通常用符号⊕表示。函数关系为：

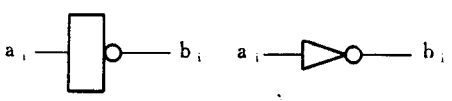


图 1-5　非门的逻辑符号

$$C = A \oplus B = \overline{A} \cdot B + A \cdot \overline{B}$$

逻辑同或运算通常用符号⊙表示。函数关系为：

$$C = A \odot B = A \cdot B + \overline{A} \cdot \overline{B}$$

读者可自行写出异或运算和同或运算的函数关系（用真值表表示出来）。

计算机中常使用复合的门电路，它们有：

- 与非门，其逻辑符号如图 1-6(a)所示。函数关系为：

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

A	B	C
0	0	1
0	1	1
1	0	1
1	1	0

● 或非门，其逻辑符号如图 1-6(b)所示。函数关系为：

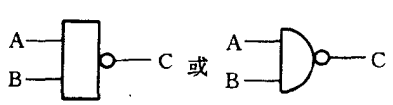
$$C = \overline{A + B}$$

A	B	C
0	0	1
0	1	0
1	0	0
1	1	0

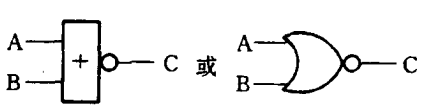
● 与或非门，其逻辑符号如图 1-6(c)所示。函数关系为

$$F = \overline{A \cdot B + C \cdot D}$$

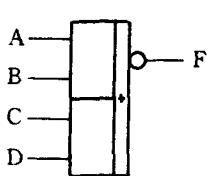
A	B	C	D	F
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0



(a)与非门



(b)或非门



(c)与或非门

图 1-6　复合门逻辑符号