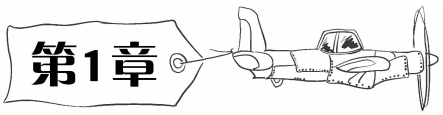


计算机基础知识

本章知识点与要求：

(1) 理解微型计算机的组成、各部分的作用、工作原理与过程。

(2) 了解单片机的产生、应用与发展趋势，理解单片机的特点。

(3) 掌握进位计数制的表示及其相互转换方法。

(4) 掌握机器数及其表示方法和运算。

1.1 概述

1.1.1 计算机

1. 计算机的发展历史

1946 年 2 月 15 日，第一台电子数字计算机(ENIAC)问世，这标志着计算机时代的到来。ENIAC 是电子管计算机，虽然它与现代的计算机相比有许多不足之处，但它的问世开创了计算机科学技术的新纪元，对人类的生产和生活方式产生了巨大的影响。

匈牙利籍数学家冯·诺依曼在计算机方案的设计上做出了重要的贡献。1946 年 6 月，他又提出了“程序存储”和“二进制运算”的思想，进一步构建了计算机系统由运算器、控制器、存储器、输入设备和输出设备组成这一计算机的经典结构，如图 1-1 所示。

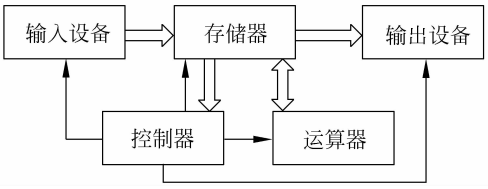


图 1-1　计算机的经典结构

按元器件的发展来划分，计算机技术的发展相继经历了 5 个时代：电子管计算机、晶体管计算机、集成电路计算机、大规模集成电路计算机和超大规模集成电路计算机。

第一代：电子管计算机(1946—1958)　采用磁鼓存储器，使用机器语言汇编语言编程。如世界上第一台数字计算机 ENIAC。

第二代：晶体管计算机(1958—1964)　磁芯作主存储器，磁盘作外存储器，开始使用高级语言编程。

第三代：集成电路计算机(1964—1971)　使用半导体存储器，出现了多终端计算机和计算机网络。

第四代：大规模集成电路计算机(1971—1981)　出现了微型计算机、单片微型计算机，外部设备多样化。

第五代：超大规模集成电路计算机(1981 年至今)　集成度更高，功能日新月异。

虽然计算机在半个多世纪的时间里得到飞速发展，但是计算机的结构仍然没有突破冯·诺依曼提出的计算机的经典结构框架。

未来计算机的发展趋势是：微型化、网络化和智能化。

2. 计算机的种类

计算机可以按照不同的分类方法分为以下几种：

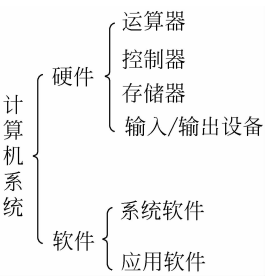
(1) 模拟和数字型(按信号)；

(2) 通用和专用型(按用途)；

(3) 巨型机、大型机、小型机、微型机(按功能和规模)；

(4) PC 和单片机。

计算机系统的组成如下：



小结：

(1) 冯·诺依曼体系结构：有五大部件(硬件)，包括运算器、控制器、存储器、输入和输出设备。

(2) 计算机在系统组成上分为硬件和软件(Body and Soul，肉体和灵魂)。

(3) 计算机信息：二进制形式，指令和数据。

1.1.2　微型计算机

微型计算机简称微机，由微处理器、存储器及 I/O 端口电路组成。各部分通过地址总线(AB)、数据总线(DB)和控制总线(CB)相连。微型计算机的结构如图 1-2 所示。

1.1.3　单片机

单片机即单片式微型计算机(micro computer unit，MCU)，是将计算机主机(CPU、内存和 I/O 端口)集成在一小块硅片上的微型计算机。

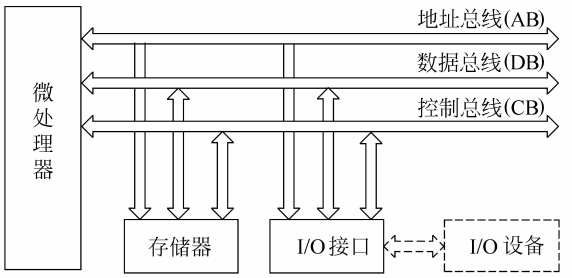


图 1-2　微型计算机的结构

单片机为工业测控而设计，又称微控制器。它具有“三高”优势（集成度高、可靠性高、性价比高）。

单片机主要应用于工业检测与控制、计算机外设、智能仪器仪表、通信设备、家用电器等。特别适合于嵌入式微型机应用系统。

虽然单片机出现的历史并不长，但发展十分迅猛。它的产生与发展和微处理器的产生与发展大体同步，自 1971 年美国 Intel 公司首先推出 4 位微处理器以来，单片机的发展大致经历了以下三个主要阶段。

1. 初级阶段

20 世纪 70 年代，是单片机发展的初级阶段和低性能单片机阶段。

1971 年 11 月 Intel 公司首先设计出集成度为 2000 只晶体管/片的 4 位微处理器 Intel 4004，并配有 RAM、ROM 和移位寄存器，构成了第一台 MCS-4 微处理器，而后又推出了 8 位微处理器Intel 8008。其他各公司相继推出了 8 位微处理器。

1976 年，Intel 公司推出了 MCS-48 系列单片机。采用将 8 位 CPU、1KB ROM、64B RAM、27 根 8 位并行 I/O 线和 1 个 8 位定时/计数器等集成于一块半导体芯片上的单片结构。虽然其寻址范围有限（不大于 4KB），也没有串行 I/O 端口，RAM、ROM 容量小，中断系统也较简单，但功能可满足一般工业控制和智能化仪器、仪表等的需要。

该阶段单片机的特点是：存储器容量较小，寻址范围小（不大于 4KB），无串行接口，指令系统功能不强。

典型代表：MCS48 系列，4 位机和 8 位机。

2. 成熟阶段

20 世纪 80 年代，是单片机性能的完善提高阶段。

1980 年，Intel 公司推出了 MCS-51 系列单片机。芯片内集成 8 位 CPU、4KB ROM、128B RAM、4 个 8 位并行口、1 个全双工串行口、两个 16 位定时/计数器。寻址范围 64KB，并有控制功能较强的布尔处理器。这一阶段推出的高性能 8 位单片机普遍带有串行口，有多级中断处理系统，多个 16 位定时器/计数器。片内 RAM、ROM 的容量加大，且寻址范围可达 64KB，个别片内还带有 A/D 转换接口。

1982 年，Intel 公司推出了高性能的 16 位单片机 MCS-96 系列单片机。芯片内集成 16

位 CPU、8KB ROM、232B RAM、5 个 8 位并行口、1 个全双工串行口、两个 16 位定时/计数器。寻址范围 64KB。片上还有 8 路 10 位 ADC、1 路 PWM 输出及高速 I/O 部件等。由于其采用了最新的制造工艺，使芯片集成度高达 12 万只晶体管/片。

该阶段单片机的特点是：结构体系完善，性能已大大提高，面向控制的特点进一步突出。现在，MCS-51 已成为公认的单片机经典机种。

典型代表：MCS51、MCS96 系列，8、16 位机。

3. 高速发展阶段

20 世纪 90 年代以后，是 8、16、32 位机全面高速发展和微控制器化阶段。单片机在集成度、功能、速度、可靠性、应用领域等全方位向更高水平发展。

该阶段单片机的特点是：片内面向测控系统外围电路增强，使单片机可以方便灵活地用于复杂的自动测控系统及设备。

“微控制器”的称谓更能反映单片机的本质。

1.1.4 嵌入式系统

嵌入式系统(embedded system)是单片机近年来的新型应用形态，它是以产品为对象的特殊结构的计算机系统，是将单片机嵌入到应用产品之中的系统。

嵌入式系统属于专用计算机，主要应用于智能仪表、智能传感器、智能家电、智能办公设备、汽车及军事电子设备等应用系统。

单片机体积小、价格低、可靠性高，其非凡的嵌入式应用形态对于满足嵌入式应用需求具有独特的优势。

1.2 单片机中数的表示方法

1.2.1 数制及其转换

1. 进位计数制

数制(即计数制，又称记数制)是人们利用符号来计数的科学方法。数制有很多种，但在计算机使用中常用的数制为十进制、二进制和十六进制。十进制是人们日常生活中最熟悉的进位计数制；二进制是在计算机系统中采用的进位计数制；十六进制是人们在计算机指令代码和数据的书写中经常使用的数制。

下面介绍数制的基与权。

当进位计数制采用位置表示法时，同一数字在不同的数位所代表的数值是不同的。每一种进位计数应包含两个基本的因素。

(1) 基数 R(radix)：它代表计数制中所用到的数码个数。如二进制计数中用到 0 和 1 两个数码，而八进制计数中用到 0～7 共 8 个数码。一般来说，基数为 R 的计数制(简称 R 进制)中，包含 0，1，…，$R-1$ 个数码，进位规律为“逢 R 进 1”。

(2) 位权 W(weight)：进位计数制中，某个数位的值是由这一位的数码值乘以处在这一位的固定常数决定的，通常把这一固定常数称为位权值，简称位权。各位的位权是以 R

为底的幂。如十进制数基数 $R=10$，则个位、十位、百位上的位权分别为 10^0、10^1、10^2。

一个 R 进制数 N，可以用以下两种形式表示。

(1) 并列表示法，或称位置计数法：

$$(N)_R = (K_{n-1}K_{n-2}\cdots K_1K_0K_{-1}K_{-2}\cdots K_{-m})_R$$

(2) 多项式表示法，或称按权展开式：

$$\begin{aligned}(N)_R = {} & K_{n-1}\times R^{n-1}+K_{n-2}\times R^{n-2}+\cdots+K_0\times R^0\\ & +K_{-1}\times R^{-1}+\cdots+K_{-m}\times R^{-m}\\ = {} & \sum_{i=-m}^{n-1}K_i\times R^i\end{aligned}$$

其中，m、n 为正整数；n 代表整数部分的位数；m 代表小数部分的位数；K_i 代表 R 进制中的任一个数码，$0\leqslant K_i\leqslant R-1$。

1) 十进制 ND

十进制数有两个主要特点：

(1) 有 10 个不同的数字符号：0，1，2，…，9；

(2) 低位向高位进位的规律是“逢 10 进 1”。

因此，同一个数字符号在不同的数位所代表的数值是不同的。如 555.5 中 4 个 5 分别代表 500、50、5 和 0.5。

任意一个十进制数 N 都可以表示成按权展开的多项式：

$$\begin{aligned}(N)_{10} = {} & K_{n-1}\times 10^{n-1}+K_{n-2}\times 10^{n-2}+\cdots+K_0\times 10^0\\ & +K_{-1}\times 10^{-1}+\cdots+K_{-m}\times 10^{-m}\\ = {} & \sum_{i=-m}^{n-1}K_i\times 10^i\end{aligned}$$

例如，543.21 可表示为

$$543.21 = 5\times 10^2+4\times 10^1+3\times 10^0+2\times 10^{-1}+1\times 10^{-2}$$

十进制用于计算机输入输出，人机交互。

2) 二进制 NB

对二进制数，$R=2$，K_i 取 0 或 1，进位规律为“逢 2 进 1”。

任一个二进制数 N 可表示为

$$\begin{aligned}(N)_2 = {} & K_{n-1}\times 2^{n-1}+K_{n-2}\times 2^{n-2}+\cdots+K_0\times 2^0+K_{-1}\times 2^{-1}+\cdots+K_{-m}\times 2^{-m}\\ = {} & \sum_{i=-m}^{n-1}K_i\times 2^i\end{aligned}$$

例如，$(1001.101)_2=1\times 2^3+0\times 2^2+0\times 2^1+1\times 2^0+1\times 2^{-1}+0\times 2^{-2}+1\times 2^{-3}$。

二进制为计算机中使用的数据形式。

3) 八进制 NQ

对八进制数，$R=8$，K_i 可取 0～7 共 8 个数码中的任意一个，进位规律为“逢 8 进 1”。

任意一个八进制数 N 可以表示为

$$\begin{aligned}(N)_8 = {} & K_{n-1}\times 8^{n-1}+K_{n-2}\times 8^{n-2}+\cdots+K_0\times 8^0\\ & +K_{-1}\times 8^{-1}+\cdots+K_{-m}\times 8^{-m}\end{aligned}$$

$$= \sum_{i=-m}^{n-1} K_i \times 8^i$$

例如，$(246.12)_8 = 2\times8^2+4\times8^1+6\times8^0+1\times8^{-1}+2\times8^{-2}$。

4）十六进制 NH

对十六进制数，$R=16$，K_i 可取 0～15 共 16 个数码中的任一个，但 10～15 分别用 A、B、C、D、E、F（或 a、b、c、d、e、f）表示，进位规律为“逢 16 进 1”。

任意一个十六进制数 N 可表示为

$$(N)_{16} = K_{n-1}\times16^{n-1}+K_{n-2}\times16^{n-2}+\cdots+K_0\times16^0 + K_{-1}\times16^{-1}+\cdots+K_{-m}\times16^{-m}$$

$$= \sum_{i=-m}^{n-1} K_i \times 16^i$$

例如，$(2D07.A)_{16} = 2\times16^3+13\times16^2+0\times16^1+7\times16^0+10\times16^{-1}$。

十六进制用于表示二进制数。

表 1-1 给出了以上 3 种进制数与十进制数的对应关系。为避免混淆，除用 $(N)_R$ 的方法区分不同进制数外，还常用数字后加字母作为标注。其中字母 B（binary）表示二进制数；字母 Q（octal 的缩写为字母 O，但为区别数字 0，故写成 Q）表示八进制数；字母 D（decimal）或不加字母表示十进制数；字母 H（hexadecimal）表示十六进制数。如，101、101D、101B、101H。

表 1-1　3 种进制数与十进制数的对应关系

十进制数	二进制数	八进制数	十六进制数	十进制数	二进制数	八进制数	十六进制数
0	0000B	0Q	0H	8	1000B	10Q	8H
1	0001B	1Q	1H	9	1001B	11Q	9H
2	0010B	2Q	2H	10	1010B	12Q	AH
3	0011B	3Q	3H	11	1011B	13Q	BH
4	0100B	4Q	4H	12	1100B	14Q	CH
5	0101B	5Q	5H	13	1101B	15Q	DH
6	0110B	6Q	6H	14	1110B	16Q	EH
7	0111B	7Q	7H	15	1111B	17Q	FH

小结：

（1）十进制 ND

符号集：0～9。规则：逢十进一。

例如，$1234.5 = 1\times10^3+2\times10^2+3\times10^1+4\times10^0+5\times10^{-1}$。

按权展开式以 10 为基数，各位系数为 0～9。

一般表达式：

$$\mathrm{ND} = d_{n-1}\times10^{n-1}+d_{n-2}\times10^{n-2}+\cdots+d_0\times10^0+d_{-1}\times10^{-1}+\cdots$$

（2）二进制 NB

符号集：0、1。规则：逢二进一。

例如，1101.101B=$1\times2^3+1\times2^2+0\times2^1+1\times2^0+1\times2^{-1}+0\times2^{-2}+1\times2^{-3}$。

按权展开式以2为基数，各位系数为0、1。由于数码只有0、1，所以加权展开式求和时简化为对权的“取”或“舍”。

一般表达式：

$$NB=b_{n-1}\times2^{n-1}+b_{n-2}\times2^{n-2}+\cdots+b_0\times2^0+b_{-1}\times2^{-1}+\cdots$$

(3) 十六进制NH

符号集：0～9，A～F。规则：逢十六进一。

例如，DFC.8H=$13\times16^2+15\times16^1+12\times16^0+8\times16^{-1}$。

按权展开式以16为基数，各位系数为0～9，A～F。

一般表达式：

$$NH=h_{n-1}\times16^{n-1}+h_{n-2}\times16^{n-2}+\cdots+h_0\times16^0+h_{-1}\times16^{-1}+\cdots$$

2. 进位计数制之间的转换

1) 二、八、十六进制数转换成十进制数

规则：按相应的权表达式展开，再按十进制求和。

例如，1011.1010B=$1\times2^3+1\times2^1+1\times2^0+1\times2^{-1}+1\times2^{-3}$=11.625

DFC.8H=$13\times16^2+15\times16^1+12\times16^0+8\times16^{-1}$=3580.5

2) 二进制数与十六进制数之间的转换

由于$2^4=16$，1位十六进制数可用4位二进制数来表示，这样二进制数与十六进制数之间的转换就很方便。

(1) 二进制数→十六进制数

方法是：采用“合4为1”的原则。从小数点开始，向左和向右把整数和小数部分每4位分为一组。整数部分最高位的一组不足4位，在其左边补0到4位；小数部分最低位的一组不足4位，在其右边补0到4位。然后将每组二进制数用对应的十六进制数代替，即得到转换结果。

例如，1101000101011.001111B=0001 1010 0010 1011 . 0011 1100=1A2B.3C

1 A 2 B . 3 C

1111101.11B=0111 1101 . 1100=7D.CH

7 D . C

(2) 十六进制数→二进制数

方法与二进制数到十六进制数转换过程相反，采用“1分为4”的原则，将每一位十六进制数用对应的4位二进制数取代即可。

例如，3AF.2H=0011 1010 1111 . 0010=1110101111.001B

3 A F . 2

4D5E.6FH=0100 1101 0101 1110 . 0110 1111=100110101011110.01101111B

4 D 5 E . 6 F

3) 二进制数与八进制数之间的转换

由于$2^3=8$，故可采用“合3为1”的原则，即从小数点开始分别向左、右两边各以3位为一组进行二-八换算；若不足3位的以0补足，便可将二进制数转换为八进制数。或采用“1

分为3”的原则，便可将八进制数转换为二进制数。

例如，1111011.0101B＝001 111 011.010 100＝173.24Q
　　1　7　3．2　4

1357.246Q＝001 011 101 111.010 100 110＝1011101111.01010011B
　　1　3　5　7．2　4　6

4）十进制数转换成二、十六进制数

（1）十进制→二进制

规则：整数　除以2、取余数，余数逆序排列；

　　小数　乘以2、取整数，整数顺序排列。

以小数点为起点求得整数和小数的每一位。

例 1-1　将十进制数123.375转换为二进制数。

解　依照上述基本方法有

整数部分		小数部分	
2⌊123		0.375×2＝0.75	整数部分＝0
2⌊61	余数＝1	0.75×2＝1.5	整数部分＝1
2⌊30	余数＝1	0.5×2＝1.0	整数部分＝1
2⌊15	余数＝0		
2⌊7	余数＝1		
2⌊3	余数＝1		
2⌊1	余数＝1		
0	余数＝1		

结果：123.375＝1111011.011B

（2）十进制→十六进制

规则：整数　除以16、取余数，余数逆序排列；

　　小数　乘以16、取整数，整数顺序排列。

以小数点为起点求得整数和小数的每一位。

例 1-2　① 208转换成十六进制数

16⌊208　　余＝0

16⌊13　　余＝13＝DH

0

结果：208＝D0H

② 0.625转换成十六进制数

0.625×16＝10.0　　整数部分＝10＝AH

结果：0.625＝0.AH

5）十进制数转换成任意 R 进制数

与上面方法类似，规则：

整数　“除基取余”：十进制整数不断除以转换进制基数 R，直至商为0。每除一次取一个余数，从低位排向高位。

小数　“乘基取整”：用转换进制的基数 R 乘以小数部分，直至小数为0或达到转换精

度要求的位数。每乘一次取一次整数，从最高位排到最低位。

例 1-3　① 将十进制数 168 转换为二、八、十六进制数

2|168							
2|84	余数 0，	$K_0=0$					
2|42	余数 0，	$K_1=0$					
2|21	余数 0，	$K_2=0$					
2|10	余数 1，	$K_3=1$					
2|5	余数 0，	$K_4=0$	8|168				
2|2	余数 1，	$K_5=1$	8|21	余数 0，$K_0=0$		16|168	
2|1	余数 0，	$K_6=0$	8|2	余数 5，$K_1=5$		16|10	余数 8，$K_0=8$
0	余数 1，	$K_7=1$	0	余数 2，$K_2=2$		0	余数 10，$K_1=A$
结果：168＝10101000B			168＝250Q			168＝A8H	

② 将 0.686 转换成二、八、十六进制数(用小数点后 5 位表示)

$0.686\times2=1.372, K_{-1}=1$	$0.686\times8=5.488, K_{-1}=5$	$0.686\times16=10.976, K_{-1}=A$
$0.372\times2=0.744, K_{-2}=0$	$0.488\times8=3.904, K_{-2}=3$	$0.976\times16=15.616, K_{-2}=F$
$0.744\times2=1.488, K_{-3}=1$	$0.904\times8=7.232, K_{-3}=7$	$0.616\times16=9.856, K_{-3}=9$
$0.488\times2=0.976, K_{-4}=0$	$0.232\times8=1.856, K_{-4}=1$	$0.856\times16=13.696, K_{-4}=D$
$0.976\times2=1.952, K_{-5}=1$	$0.856\times8=6.848, K_{-5}=6$	$0.696\times16=11.136, K_{-5}=B$

结果：$0.686\approx0.10101$B，　$0.686\approx0.53716$Q，　$0.686\approx0.$AF9DBH

从以上例子可以看出，二进制表示的数越精确，所需的数位就越多，这样不利于书写和记忆，而且容易出错。另外，若用同样数位表示数，则八、十六进制数所表示数的精度较高。所以在汇编语言编程中常用八进制或十六进制数作为二进制数的缩码，来书写和记忆二进制数，便于人-机信息交换。在 MCS-51 系列单片机编程中，通常采用十六进制数。

1.2.2　BCD 码

1. 二-十进制编码 BCD 码

BCD(binary coded decimal)码是用二进制代码表示的十进制数，即对十进制数采用二进制数进行编码。这种编码既具有二进制数的形式(由 0 和 1 组成)，又有十进制数的特点(逢十进一)，称为二-十进制码，又称 BCD 码。BCD 码有 8421 码、5421 码、2421 码、余 3 码等不同类型。最常用的一种是 8421BCD 码(见表 1-2)，其中 8、4、2、1 分别为 4 位二进制数的位权值。

表 1-2　8421BCD 码表

十进制数	BCD 码	十进制数	BCD 码	十进制数	BCD 码
0	0000B	4	0100B	8	1000B
1	0001B	5	0101B	9	1001B
2	0010B	6	0110B		
3	0011B	7	0111B		

二进制码在 1010B～1111B 范围时，对 8421BCD 码而言属于非法码。

例 1-4 求十进制数 876 的 BCD 码。

[876]BCD = 1000 0111 0110

876 = 36CH = 11 0110 1100B

2. BCD 码存储方式

计算机中的存储单元通常以字节(8 位二进制数)为单位，在一个字节中存放 BCD 码有两种方式，即压缩的 BCD 码和非压缩的 BCD 码。

一个 BCD 码有 4 个二进制位，所以在一个字节中可存放两个 BCD 码，这种存储方式称为压缩 BCD 码表示法。以压缩 BCD 码表示十进制数时，一个字节表示 2 位十进制数。

在一个字节中若低 4 位为 BCD 码，高 4 位全为 0，这种存放形式称为非压缩的 BCD 码表示法。

3. BCD 码运算

十进制调整：计算机实际按二进制法则计算，加入十进制调整操作后可计算 BCD 码。

十进制调整方法：当计算结果有非 BCD 码或产生进位、借位时，进行加 6 或减 6 调整。

BCD 码的加法运算：BCD 码的低位与高位之间是“逢十进一”，而 4 位二进制数(即十六进制)是“逢十六进一”。因此，用二进制加法器进行 BCD 码加法运算时，若 BCD 码的各位之和在 0～9 之间，则其加法运算和二进制运算规则一致，即结果是正确的；若 BCD 码的各位之和大于 9 或者产生进位，则此位需要“加 6 修正”。

BCD 码的减法运算：BCD 码的低位向高位借位是“借一当十”，而 4 位二进制数(即十六进制)是“借一当十六”。因此，进行 BCD 码减法运算时，若某位有借位，则此位需要“减 6 修正”。

当有多位 BCD 码运算时，每一位均需要按上述方法修正。

例 1-5 计算 BCD 码 78＋59。

```
  0111 1000        [78]BCD
+ 0101 1001      + [59]BCD
-----------      ----------
  1101 0001        [137]BCD   产生非 BCD 码和进位
+ 0110 0110      + 66 调整
-----------      ----------
1 0011 0111        结果：137
```

例 1-6 计算 BCD 码 38－29。

```
  0011 1000        [38]BCD
+ 1101 0111      - [29]BCD
-----------      ----------
1 0000 1111        产生非 BCD 码(具体运算方法见后面关于“补码”的介绍)
+ 1111 1010        - 06 调整
-----------      ----------
1 0000 1001        结果无借位：9
```

1.2.3 ASCII 码

目前在计算机系统中普遍采用的字符编码系统是制定于 1963 年的美国标准信息交换

码，简称 ASCII 码（American Standard Code for Information Interchange），如表 1-3 所示，主要用于计算机与计算机及外设之间传递信息。

表 1-3 美国标准信息交换码 ASCII 码表

行 列	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

ASCII 码是用 7 位二进制数编码来表示 128 个字符和符号，一个 ASCII 码存放在一个字节的低 7 位，字节的最高位为 0。ASCII 码表包括 10 个十进制数 0～9，大写和小写英文字母各 26 个，32 个通用控制符号，34 个专用符号，共 128 个字符。其中数字 0～9 的 ASCII 编码分别为 30H～39H，英文大写字母 A～Z 的 ASCII 编码从 41H 开始依次编至 5AH。ASCII 编码从 20H～7EH 均为可打印字符，而 00H～1FH 为通用控制符，它们不能被打印出来，只起控制或标志的作用，如 0DH 表示回车（CR），0AH 表示换行控制（LF），04H（EOT）为传送结束标志。

在通信中，常在 7 位 ASCII 码的最高位之前加上 1 位作奇偶校验位，以确定数据传输是否正确。

奇偶校验有奇校验和偶校验。偶校验的含义是，包括校验位在内的所有为 1 的位数之和为偶数。如字母 A 的 ASCII 码 1000001B 的偶校验码是 01000001B；同理，奇校验的含义是包括校验位在内的所有为 1 的位数之和为奇数。

1.2.4 单片机中数的表示方法

计算机中的数按数的性质分，分为整数（无符号整数、有符号整数）和小数（定点数、浮点

数)；按符号来分，分为有符号数(正数、负数)和无符号数。在计算机中也存在着如何表示正、负数的问题。由于计算机只能识别 0 和 1，因此，在计算机中通常把一个二进制数的最高位作为符号位，以表示数值的正与负(若用 8 位表示一个数，则 D7 位为符号位；若用 16 位表示一个数，则 D15 位为符号位)。

机器中，数的符号用“0”、“1” 表示。最高位作符号位，“0”表示“＋”，“1”表示“－”。

1. 真值和机器数

数在计算机内的表示形式称为机器数。机器数就是机器中数的表示形式，其位数通常为 8 的倍数，而这个数本身称为该机器数的真值。真值就是机器数所代表的实际数值。

例如，一个 8 位机器数与它的真值对应关系如下：

真值：　$X_1=+84=+1010100B$，　$X_2=-84=-1010100B$

机器数：　$[X_1]_{机}=01010100B$，　$[X_2]_{机}=11010100B$

真值：　$X_3=+45H=+1000101B$，　$X_4=-55H=-1010101B$

机器数：　$[X_3]_{机}=01000101B$，　$[X_4]_{机}=11010101B$

2. 有符号数的表示方法

有符号数由符号位和数值位两部分组成。如前所述，数学中的正、负用符号“＋”、“－”来表示，而在计算机中规定用“0”表示“＋”、用“1”表示“－”。例如，N1＝＋1011B，N2＝－1011B，在计算机中用 8 位二进制数可分别表示为

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	1	0	1	1
符号	数值部分						

D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	0	1	0	1	1
符号	数值部分						

这样，数的符号位在计算机中已经数码化了。符号位被数码化了的数就称为机器数，把原来的数值称为机器数的真值。例如，一个字节的数 00000111B、10000011B 就是机器数，而＋0000111B、－0000011B 就是机器数的真值。

3. 无符号数的表示方法

用来表示数的符号的数位称为符号位。无符号数没有符号位，全部有效位都用于表示数值。

n 位的无符号二进制数 X，它可以表示的数的范围为 $0\leqslant X\leqslant 2^n-1$。若结果超出了数的可表示范围，则会产生溢出而出错。

4. 数的定点和浮点表示

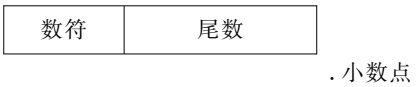
1) 定点表示

定点法中，约定所有数据的小数点隐含在某个固定位置。这个固定的位置是事先约定

好的，不必用符号表示。用定点法表示的实数称为定点数。通常，定点表示采用以下两种方法。

(1) 定点整数表示法

小数点固定在最低数值位之后，机器中能表示的所有数都是整数。其格式如下：

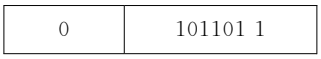


当用 n 位表示数 N 时，1 位为符号位，$n-1$ 位为数值位，则 N 的可表示范围是

$$-2^{n-1} \leqslant N \leqslant 2^{n-1}-1$$

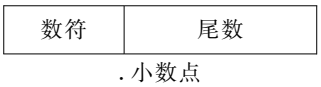
若 $n=8$，则 $-128 \leqslant N \leqslant 127$；若 $n=16$，则 $-32\,768 \leqslant N \leqslant 32\,767$。

例如，若 $N=+1011011$，$n=8$，则用定点整数法可将 N 表示为



(2) 定点小数表示法

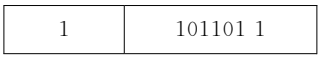
小数点固定在最高数值位之前，即在数符与数值之间，机器中能表示的所有数即为纯小数。其格式如下：



当用 n 位表示数 N 时，1 位为符号位，$n-1$ 位为数值位，则 N 的可表示范围是

$$-(1-2^{1-n}) \leqslant N \leqslant 1-2^{1-n}$$

例如，若 $N=-0.1011011$，$n=8$，则用定点小数法可将 N 表示为



2) 浮点表示

浮点法中，数据的小数点位置并不是固定的，而是可浮动的，类似十进制数的科学表示法。可将任意一个二进制数 N 表示成

$$N=\pm M \cdot 2^{\pm E}$$

其中，M 为尾数，为纯二进制小数；E 称为阶码。可见，一个浮点数有阶码和尾数两部分，且都带有表示正负的阶码符与数符，其格式为

数符表示数的正、负，阶符指明尾数小数点向右或向左浮动的方向，阶码 E 指明尾数小数点移动的位数，所以阶符和阶码表明了数值 N 的小数点的位置。

浮点数能表示的数值范围很大。为了提高精度，发挥尾数有效位的最大作用，还规定尾数数字部分原码的最高位为 1，称为规格化表示法。如 0.000101 表示为

$$2^{-3} \times 0.101$$

5. 原码、反码和补码

计算机中的有符号数或机器数，有三种表示形式，即原码、反码和补码。

对于符号数、机器数，数 X 的原码记作$[X]_{原}$，反码记作$[X]_{反}$，补码记作$[X]_{补}$。

1) 原码(true form)

对于带符号二进制数(字节、字或双字)，直接用最高位表示数的符号，数值用其绝对值表示，该形式称为原码。正数的符号位用 0 表示，负数的符号位用 1 表示，数值部分用真值的绝对值来表示。

(1) 正数的原码

若真值为正数 $X=+K_{n-2}K_{n-3}\cdots K_1K_0$(即 $n-1$ 位二进制正数)，则

$$[X]_{原}=0K_{n-2}K_{n-3}\cdots K_1K_0$$

(2) 负数的原码

若真值为负数 $X=-K_{n-2}K_{n-3}\cdots K_1K_0$(即 $n-1$ 位二进制负数)，则

$$\begin{aligned}[X]_{原}&=1K_{n-2}K_{n-3}\cdots K_1K_0\\&=2^{n-1}+K_{n-2}K_{n-3}\cdots K_1K_0\\&=2^{n-1}-(-K_{n-2}K_{n-3}\cdots K_1K_0)\\&=2^{n-1}-X\end{aligned}$$

例如，+115 和−115 在计算机中(设机器字长为 8 位)，其原码可分别表示为

$$[+115]_{原}=01110011\text{B};\quad [-115]_{原}=11110011\text{B}$$

(3) 零的原码

若真值为零，则原码有两种表示法：

$$[+0]_{原}=000\cdots00$$

$$[-0]_{原}=100\cdots00$$

可见，真值 X 与原码$[X]_{原}$的关系为

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

原码小结：

(1) 表示方法：最高位为符号位，0 表示“+”，1 表示“−”。数值位与真值数值位相同。例如，8 位原码机器数如下：

真值：　　$x_1=+1010100\text{B}$，　　$x_2=-1010100\text{B}$

机器数：　　$[x_1]_{原}=01010100$，　　$[x_2]_{原}=11010100$

(2) 特点：原码表示简单直观，但 0 的表示不唯一，加减运算复杂。

(3) 8 位二进制原码能表示的范围是：−127～+127。

2) 反码

一个正数的反码等于该数的原码。一个负数的反码，等于该负数的原码符号位不变(即为 1)，数值位按位求反(即 0 变 1，1 变 0)；或者在该负数对应的正数原码上连同符号位逐位求反。例如：

带符号正数 +1000101B，原码为 01000101B，反码为 01000101B(45H)；

带符号负数 −1010101B，原码为 11010101B，反码为 10101010B(AAH)。

又如：$X=+103$，则$[X]_{反}=[X]_{原}=01100111B$；$X=-103$，$[X]_{原}=11100111B$，则$[X]_{反}=10011000B$。

(1) 正数的反码：$[X]_{反}=[X]_{原}$；

(2) 负数的反码：$[X]_{反}=1\overline{K}_{n-2}\overline{K}_{n-3}\cdots\overline{K}_1\overline{K}_0$；

(3) 零的反码：$[+0]_{反}=000\cdots00$

$[-0]_{反}=111\cdots11$

真值 X 与反码$[X]_{反}$的关系为

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

反码小结：

(1) 表示方法：对一个数 X，若 $X>0$，则

$$[X]_{反}=[X]_{原}$$

若 $X<0$，则$[X]_{反}$＝对应原码的符号位不变，数值部分按位求反。

例如：$X=-52D=-0110100B$，$[X]_{原}=10110100$，$[X]_{反}=11001011$。

(2) 特点：二进制数采用原码和反码表示时，符号位不能同数值一道参加运算。

3) 补码(two's complement)

在计算机中，对带符号数的运算均采用补码。

(1) 补码的概念

在日常生活中有许多“补”数的事例。如钟表，假设标准时间为 6 点整，而某钟表却指在 9 点，若要把表拨准，可以有两种拨法：一种是倒拨 3 小时，即 9－3＝6；另一种是顺拨 9 小时，即 9＋9＝6。尽管将表针倒拨或顺拨不同的时数，但却得到相同的结果，即 9－3 与 9＋9 是等价的。这是因为钟表采用 12 小时进位，超过 12 就从头算起，即：9＋9＝12＋6，该 12 称为模(mod)。

模(mod) 或者模数(module)为一个系统的量程或此系统所能表示的最大数，记为 M 或者 mod M。它会自然丢掉，如：

$$9-3=9+9=12+6\rightarrow 6\quad(\text{mod}12\text{ 自然丢掉})$$

通常称＋9 是－3 在模为 12 时的补数。于是，引入补数后使减法运算变为加法运算。

例如，11－7＝11＋5→4 (mod12)

＋5 是－7 在模为 12 时的补数，减 7 与加 5 的效果是一样的。

一个 n 位的二进制计数器(或者存储单元、寄存器)，它的容量为 2^n，即它的模为 2^n(可以表示 2^n 个不同的数)。字长为 n 的计算机中，数 2^n 和 0 的表示形式一样。

任何有模的计量器，均可化减法为加法运算。

对于 n 位计算机来说，数 X 的补码定义为

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

即正数的补码就是它本身，负数的补码是真值与模数相加而得。

① 正数的补码与其原码相同，即$[X]_{补}=[X]_{原}$；

② 零的补码为零，$[+0]_{补}=[-0]_{补}=000\cdots00$；

③ 负数才真正有求补码的问题。

(2) 负数补码的求法

补码的求法一般有两种。

① 用补码定义式

在用补码定义式求补码的过程中,由于做一次减法很不方便,故该法一般不用。

例如:$X=-0101111B$, $n=8$,则

$[X]_{补}=2^8+(-0101111\text{B})=100000000\text{B}-0101111\text{B}=11010001\text{B}(\text{mod}2^8)$

② 用原码求反码,再在数值末位加 1 可得到补码,即:$[X]_{补}=[X]_{反}+1$。

例 1-7 $X=-52\text{D}=-0110100\text{B}$

$[X]_{原}=10110100$

$[X]_{反}=11001011$

$[X]_{补}=[X]_{反}+1=11001100$

$[+0]_{补}=[+0]_{原}=00000000$

$[-0]_{补}=[-0]_{反}+1=11111111+1=100000000$

补码的优点是可以将减法运算转换为加法运算,同时数值连同符号位可以一起参加运算。例如:45H−55H=−10H,用补码运算时可以表示为

$$[45\text{H}]_{补}+[-55\text{H}]_{补}=[-10\text{H}]_{补}$$

见下表。

$[45\text{H}]_{补}$:	01000101
$+[-55\text{H}]_{补}$:	10101011
结果:	11110000

结果 11110000B 为补码;求补得到原码为 10010000B;真值为 −0010000B(即−10H)。

几个典型的带符号数据的 8 位编码如表 1-4 所示。

表 1-4 几个典型的带符号数据的 8 位编码表

真值	原码	反码	补码
+127	01111111B	01111111B	01111111B(7FH)
+1	00000001B	00000001B	00000001B(01H)
+0	00000000B	00000000B	00000000B(00H)
−0	10000000B	11111111B	00000000B(00H)
−1	10000001B	11111110B	11111111B(FFH)
−127	11111111B	10000000B	10000001B(81H)
−128	—	—	10000000B(80H)

补码小结:

(1) 表示方法:正数的补码表示与原码相同。负数补码的符号位为 1,数值位逐位求反,再在末位加 1。

例 1-8　求 8 位补码机器数：

$X=+4$　　00000100　　$[X]_{补}=00000100$

$X=-4$　　10000100

X1111011　　$[X]_{补}=11111100$

(2) 特点："0"的表示唯一，加减运算方便。在计算机中数均是用补码表示。

(3) 8 位二进制补码能表示的范围为：－128～＋127，若超过此范围，则产生溢出。

(4) 数的补码与"模"有关，"模"即计数系统的量程。

当 $X<0$ 时，$[X]_{补}=$模$-|X|$。8 位二进制数的模为

$$2^8=256$$

当 $X<0$ 时，

$$\begin{aligned}[X]_{补}&=2^8-|X|\\&=256-|X|=255-|X|+1\\&=[X]_{反}+1\end{aligned}$$

规则：求反加 1，符号位不变。

例如，10001101B，其补码为：11110010。

特殊数 10000000，该数在原码中定义为－0，在反码中定义为－127，在补码中定义为－128；对无符号数，(10000000)B＝128。

综上所述可归纳为：

正数的原码、反码、补码就是该数本身；

负数的原码其符号位为 1，数值位不变；

负数的反码其符号位为 1，数值位逐位求反；

负数的补码其符号位为 1，数值位逐位求反并在末位加 1。

例如：

带符号正数＋1000101B，反码为 01000101B，补码为 01000101B(45H)；

带符号负数－1010101B，反码为 10101010B，补码为 10101011B(ABH)。

4) 补码的运算规则与溢出判别

(1) 补码的运算规则

补码的运算规则如下：

① $[X+Y]_{补}=[X]_{补}+[Y]_{补}$

该运算规则说明：任何两个数相加，无论其正负号如何，只要对它们各自的补码进行加法运算，就可得到正确的结果，该结果是补码形式。

② $[X-Y]_{补}=[X]_{补}+[-Y]_{补}$

该运算规则说明：任意两个数相减，只要对减数连同"－"号求补，就变成$[$被减数$]_{补}$与$[-$减数$]_{补}$相加，该结果是补码形式。

③ $[[X]_{补}]_{补}=[X]_{原}$

对于运算产生的补码结果，若要转换为原码表示，则正数的结果$[X]_{补}=[X]_{原}$；负数则只要对该补结果再进行一次求补运算，$[[X]_{补}]_{补}=[X]_{原}$，就可得到负数的原码结果。

例 1-9　设 $X=$00100101B，$Y=$00110011B，用补码求 $X+Y$。

解　$[X]_{补}=$00100101，$[Y]_{补}=$00110011，可得

$$[X+Y]_{补}=[X]_{补}+[Y]_{补}=00100101+00110011=01011000$$

由于符号位为0是正数,所以

$$[X+Y]_{原}=[X+Y]_{补}=01011000$$

则

$$X+Y=(01011000)_2=+88$$

例 1-10 设 X、Y 值同例 1-9,用补码求 $X-Y$。

解 $[-Y]_{补}=11001101$,可得

$$[X-Y]_{补}=[X]_{补}+[-Y]_{补}=00100101+11001101=11110010$$

由于符号位为1是负数,所以

$$[X-Y]_{原}=[[X-Y]_{补}]_{补}=10001110$$

则

$$X-Y=-(00001110)_2=-14$$

例 1-11 设 X、Y 值同例 1-9,用补码求 $Y-X$。

解 $[-X]_{补}=11011011$,可得

$$[Y-X]_{补}=[Y]_{补}+[-X]_{补}=00110011+11011011=100001110(\text{模}\ 2^8\ \text{自然丢失})$$

则

$$Y-X=(00001110)_2=+14$$

例 1-12 设 X、Y 值同例 1-9,用补码求$(-X)+(-Y)$。

解 $[(-X)+(-Y)]_{补}=[-X]_{补}+[-Y]_{补}=11011011+11001101$

$=110101000$ (模 2^8 自然丢失)

$[(-X)+(-Y)]_{原}=[[(-X)+(-Y)]_{补}]_{补}=11011000$

则

$$(-X)+(-Y)=-(01011000)_2=-88$$

上述运算结果是正确的,但有时在补码运算中可能会出现错误的结果,看下面的例子。

例 1-13 设 $X=+100$,$Y=+50$,用补码运算求 $X+Y$,$(-X)+(-Y)$。

解 $[X]_{补}=01100100$,$[Y]_{补}=00110010$

$[-X]_{补}=10011100$,$[-Y]_{补}=11001110$

$[X+Y]_{补}=[X]_{补}+[Y]_{补}=01100100+00110010=10010110$

$[X+Y]_{原}=[[X+Y]_{补}]_{补}=11101010$

$X+Y=-(01101010)_2=-106$

正确结果应为

$$X+Y=+150$$

$$[-X]_{补}+[-Y]_{补}=10011100+11001110=01101010$$

$$[[-X]_{补}+[-Y]_{补}]_{原}=01101010$$

$$(-X)+(-Y)=+(01101010)=+106$$

正确结果应为

$$(-X)+(-Y)=-150$$

(2) 溢出的判别

计算机中判别溢出的方法通常采用双高位判别法。双高位判别法利用符号位(K_{n-1}位)及最高数值位(K_{n-2}位)的进位情况来判断是否发生了溢出。为此,需引进两个符号:C7和C6。

C7:若符号位发生进(借)位,则C7=1;否则C7=0。即C7为最高位的进(借)位;

C6:若最高数值位发生进(借)位,则C6=1;否则C6=0。即C6为次高位的进(借)位。

当两个正数补码相加时,若数值部分之和大于2^{n-1},则数值部分必有进位C6=1;而符号位却无进位C7=0。这时C7C6的状态为"01",发生正溢出。

当两个负数补码相加时,若数值部分绝对值之和大于2^{n-1},则数值部分补码之和必小于2^{n-1},C6=0;而符号位肯定有进位C7=1,这时C7C6的状态为"10",发生负溢出。

当不发生溢出时,C7和C6的状态是相同的,即C7C6的状态为"00"或"11"。

例 1-14

01011001	(+89)	10010010	(−110)
01101100	(+108)	10100100	(−92)
+) 011110000	(进位)	+) 100000000	(进位)
011000101	(−59)	100110110	(+54)
C7 = 0,C6 = 1,正溢出		C7 = 1,C6 = 0,负溢出	

例 1-15

00110010	(+50)	11101100	(−20)
01000110	(+70)	11100010	(−30)
+) 000001100	(进位)	+) 111000000	(进位)
001111000	(+120)	111001110	(−50)
C7 = 0,C6 = 0,无溢出		C7 = 1,C6 = 1,无溢出	

例 1-16

01010101	(+85)	10111100	(−68)
11011101	(−35)	00011101	(+29)
+) 110111010	(进位)	+) 001111000	(进位)
100110010	(+50)	011011001	(−39)
C7 = 1,C6 = 1,无溢出		C7 = 0,C6 = 0,无溢出	

综上所述,对计算机而言,补码的引入使带符号数的运算都按加法处理。如果C7和C6的值相等,则表示运算结果正确,没有溢出,运算结果的正与负由符号位决定(如例1-15、例1-16)如果C7和C6的值不等,则表示运算结果不正确,发生了溢出现象(如例1-14)。

在计算机中,常用"异或"电路来判别有无溢出发生。两个8位带符号二进制数相加或相减时,若C7⊕C6=1,则结果产生溢出。对16位或32位的运算,也有类似结论。

5) 机器数与真值之间的转换

已知一个负数的补码,求其真值的方法是:对该补码求补(符号位不变,数值位取反加1)即得到该负数的原码(符号位+数值位),依该原码可知其真值。

举例:

(1) $X_1=+127$,$X_2=-127$,分别求其原码、补码。

$$[X_1]_{原} = [X_1]_{补} = 01111111 = 7FH$$
$$[X_2]_{原} = 11111111 = FFH$$
$$[X_2]_{补} = 10000001 = 81H$$

(2) $X_1 = +255, X_2 = -255$,分别求其原码、补码。

$$[X_1]_{原} = [X_1]_{补} = 0000000011111111 = 00FFH$$
$$[X_2]_{原} = 1000000011111111 = 80FFH$$
$$[X_2]_{补} = 1111111100000001 = FF01H$$

1.3 单片机的内部结构

典型单片机的内部结构如图 1-3 所示。

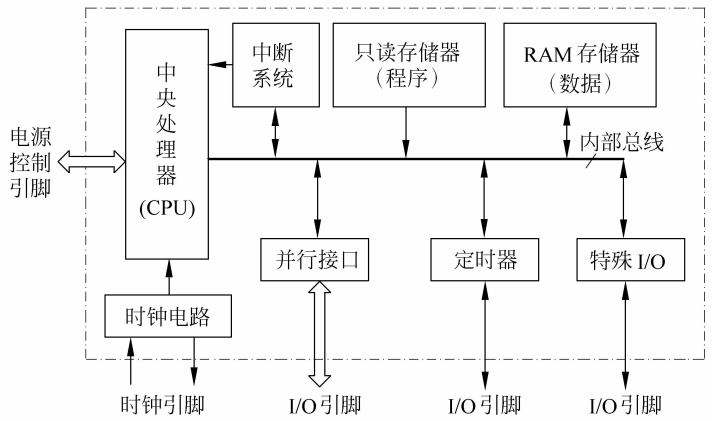


图 1-3 单片机的内部结构

基本功能部件包括中央处理器(CPU)、存储器、输入/输出接口(I/O)和总线等。

1.3.1 中央处理器

中央处理单元(central processing unit,CPU)是单片机的核心部件,由运算器和控制器组成,完成单片机的运算和控制功能。

运算器又称算术逻辑部件(arithmetic logical unit,ALU),主要完成对数据的算术运算和逻辑运算。

控制器(controller)是整个计算机的指挥中心,它负责从内部存储器中取出指令并对指令进行分析、判断,并根据指令发出控制信号,使计算机的有关部件及设备有条不紊地协调工作,保证计算机能自动、连续地运行。

CPU 中还包括若干寄存器(register),它们的作用是存放运算过程中的各种数据、地址或其他信息。寄存器种类很多,主要有以下几种。

通用寄存器:向 ALU 提供运算数据,或保留运算中间或最终的结果。

累加器 A:这是一个使用相对频繁的特殊的通用寄存器,有重复累加数据的功能。

程序计数器 PC:存放将要执行的指令地址。