CHAPTER 1

第1章 微型计算机基础

1.1 数制与编码

数制与编码是微型计算机的基本数字逻辑基础，是学习微型计算机的必备知识。数制与编码的知识一般会在数字逻辑或计算机文化基础中学习，但数制与编码的知识与当前课程的关系并非“不可或缺”，又比较枯燥。在微型计算机原理或单片机的教学中，教师普遍感觉到，学生这方面的知识不扎实。在此，以提纲挈领的形式再理一理。

1.1.1 数制及转换方法

数制是计数的方法，通常采用**进位计数制**。在微型计算机的学习与应用中，主要有十进制、二进制和十六进制3种计数方法。日常生活采用的是十进制，微型计算机硬件电路采用的是二进制，但为了更好地记忆与描述微型计算机的地址、程序代码以及运算数字，一般采用十六进制。

1. 各种进位计数制及其表示方法

各种进位计数制及其表示方法如表1.1所示。

表1.1 二进制、十进制与十六进制的计数规则与表示方法

进位制	计数规则	基数	各位的权	数 码	权值展开式	表示法	
						后缀字符	下标
二进制	逢二进一	2	2^i	0,1	$(b_{n-1}\cdots b_1 b_0 b_{-1}\cdots b_{-m})_2=\sum_{i=-m}^{n-1} b_i\cdot 2^i$	B	$()_2$
十进制	逢十进一	10	10^i	0,1,2,3,4,5,6,7,8,9	$(d_{n-1}\cdots d_1 d_0 d_{-1}\cdots d_{-m})_{10}=\sum_{i=-m}^{n-1} d_i\cdot 10^i$	D	$()_{10}$
						通常缺省表示	
十六进制	逢十六进一	16	16^i	0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F	$(h_{n-1}\cdots h_1 h_0 h_{-1}\cdots h_{-m})_{16}=\sum_{i=-m}^{n-1} h_i\cdot 16^i$	H	$()_{16}$

注：i是各进制数码在数字中的位置，i值是以小数点为界，往左依次为0、1、2、3、…，往右依次为−1、−1、−3、…。

2. 数制之间的转换

任意进制之间相互转换，整数部分和小数部分必须分别进行。各进制的相互转换关系如图 1.1 所示。

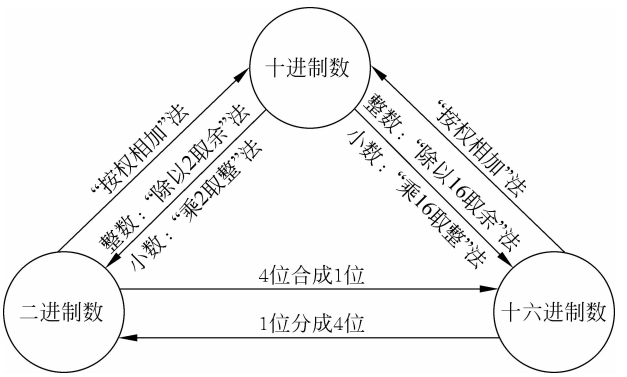


图 1.1 各进制的相互转换关系图

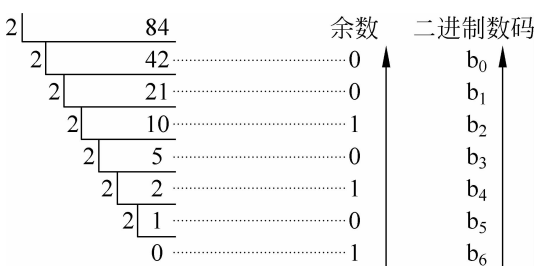
(1) 二进制、十六进制转十进制

将二进制、十六进制数按权值展开式展开相加所得数，即为十进制数。

(2) 十进制转二进制

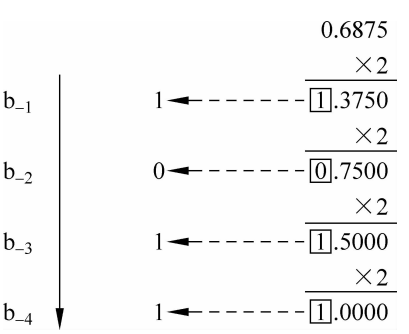
十进制转二进制要分成整数部分与小数部分，而且其转换方法是完全不同的。

① 十进制整数部分转换成二进制——除 2 取余法，并倒序排列，如下所示。



$(84)_{10}=(1010100)_2$

② 十进制小数转换成二进制小数——乘 2 取整法，如下所示。



$(0.6875)_{10}=(0.1011)_2$

将上述两部分合起来，则有

$$(84.6875)_{10}=(1010100.1011)_2$$

(3) 二进制与十六进制互转

① 二进制转十六进制。

以小数点为界，往左、往右每 4 位二进制数为一组，每 4 位二进制数用 1 位十六进制数表示，往左高位不够用 0 补齐，往右低位不够用 0 补齐。例如：

$$(111101.011101)_2=(\underline{0011}\ \underline{1101}.\underline{0111}\ \underline{0100})_2=(3D.74)_{16}$$

② 十六进制转二进制。

每位十六进制数用 4 位二进制数表示，再将整数部分最高位的 0 去掉，小数部分最低位的 0 去掉。例如：

$$(3C20.84)_{16}=(\underline{0011}\ \underline{1100}\ \underline{0010}\ \underline{0000}.\underline{1000}\ \underline{0100})_2=(11110000100000.100001)_2$$

数制转换工具

利用 PC 附件中的计算器(科学型)可实现各数制间的相互转换。单击任务栏中的“开始”按钮，选择“所有程序”→“附件”→“计算器”，即可打开计算器工具，在计算器工具界面“查看”菜单栏中选择“科学型”，计算器界面即为科学型计算器工具界面，如图 1.2 所示。

图 1.2　科学型计算器与各进制转换

转换方法：先选择被转换数制的类型，输入转换数字，再选择目标转换数制类型，此时，看到的就是转换后的数字。如 96 转换为十六进制、二进制，先选择数制类型为十进制，如图 1.2 上部所示，在输入框中输入数字 96，然后选择数制类型为十六进制，此时，显示框中看到的数字即为转换后的十六进制数字 60，如图 1.2 中部所示；再选择数制类型为二进制，此时，显示框中看到的数字即为转换后的二进制数字 1100000，如图 1.2 底部所示。

3. 二进制数的运算规则

(1) 加法运算规则

$$0+0=0,\quad 0+1=1,\quad 1+1=0(\text{有进位})$$

(2) 减法运算规则

$$0-0=0,\quad 1-0=1,\quad 1-1=0,\quad 0-1=1(\text{有借位})$$

(3) 乘法运算规则

$$0\times0=0,\quad 1\times0=1,\quad 1\times1=1$$

1.1.2 微型计算机中数的表示方法

1. 机器数与真值

数学中的正、负用符号“+”和“-”表示，计算机中是如何表示数的正、负呢？在计算机中数据是存放在存储单元内，而每个存储单元则由若干二进制位组成，其中每一数位或是0，或是1。刚好数的符号或为“+”号，或为“-”号，这样就可用一个数位表示数的符号。在计算机中规定用“0”表示“+”，用“1”表示“-”。用来表示数的符号的数位称为“符号位”(通常为最高数位)，于是数的符号在计算机中已数码化了，但从表示形式上看符号位与数值位毫无区别。

设有两个数 x_1，x_2：

$$x_1=+1011011\text{B};\quad x_2=-1011011\text{B}$$

它们在计算机中分别表示为(带下划线部分为符号位，字长为8位)：

$$x_1=\underline{0}1011011\text{B};\quad x_2=\underline{1}1011011\text{B}$$

为了区分这两种形式的数，把机器中以编码形式表示的数称为机器数(上例中 $x_1=\underline{0}1011011$ B及 $x_2=\underline{1}1011011$B)，而把原来一般书写形式表示的数称为真值($x_1=+1011011$ B及 $x_2=-1011011$B)。

若一个数的所有数位均为数值位，则该数为无符号数；若一个数的最高数位为符号位而其他数位为数值位，则该数为有符号数。由此可见，对于同一存储单元，它存放的无符号数和有符号数所能表示的数值范围是不同的[如存储单元为8位，当它存放无符号数时，因有效的数值位为8位，故该数的范围为0～255；当它存放有符号数时，因有效的数值位为7位，故该数的范围(补码)为-128～+127]。

2. 原码

对于一个二进制数，如用最高数位表示该数的符号(“0”表示“+”号，“1”表示“-”号)，其余各数位表示其数值本身，则称为原码表示法：

若 $x=\pm x_1x_2\cdots x_{n-1}$，则$[x]_{原码}=x_0x_1x_2\cdots x_{n-1}$。

其中，x_0 为原机器数的符号位，它满足：

$$x_0=\begin{cases}0, & x\geqslant 0\\ 1, & x<0\end{cases}$$

3. 反码

$[x]_{原}=0x_1x_2\cdots x_{n-1}$，则$[x]_{反}=[x]_{原}$。

$[x]_{原}=1x_1x_2\cdots x_{n-1}$，则$[x]_{反}=1\,\overline{x_1}\,\overline{x_2}\cdots\overline{x_{n-1}}$。

也就是说，正数的反码与其原码相同(反码=原码)，而负数的反码为保持原码的符号位不变，数值位按位取反。

4. 补码

(1) 补码的引进

首先以日常生活中经常遇到的钟表“对时”为例说明补码的概念。假定现在是北京标

准时间 8 时整，而一只表却指向 10 时整。为了校正此表，可以采用倒拨和顺拨 2 种方法。倒拨是逆时针拨 2 小时，时针指向 8，把倒拨视为减法，相当于 10－2＝8；顺拨是将时针顺时针拨 10 小时，时针同样也指向 8，把顺拨视为加法，相当于 10＋10＝12（自动丢失）＋8＝8。这自动丢失的数（12）就叫做模（mod），上述的加法称为“按模 12 的加法”，用数学式可表示为：

$$10+10=12+8=8 \pmod{12}$$

因时针转一圈会自动丢失一个数 12，故 10－2 与 10＋10 是等价的，称 10 和－2 对模 12 互补，10 是 －2 对模 12 的补码。引进补码概念后，就可将原来的减法 10－2＝8 转化为加法 10＋10＝12（自动丢失）＋8＝8（mod 12）。

（2）补码的定义

通过上面的例子不难理解计算机中负数的补码表示法。设寄存器（或存储单元）的位数为 n 位，它能表示的无符号数最大值为 2^n-1，逢 2^n 进 1（即 2^n 自动丢失）。换句话说，在字长为 n 的计算机中，数 2^n 和 0 的表示形式一样。若机器中的数以补码表示，则数的补码以 2^n 为模，即：

$$[x]_{补}=2^n+x \pmod{2^n}$$

若 x 为正数，$[x]_{补}=x$；若 x 为负数，$[x]_{补}=2^n+x=2^n-|x|$，即负数 x 的补码等于模 2^n 加上其真值或减去其真值的绝对值。

在补码表示法中，零只有唯一的表示形式：0000…0。

（3）求补码的方法

根据上述介绍可知，正数的补码等于原码。下面介绍负数求补码的 3 种方法。

① 根据真值求补码。

根据真值求补码就是根据定义求补码，即有：

$$[x]_{补}=2^n+x=2^n-|x|$$

即负数的补码等于 2^n（模）加上其真值，或者等于 2^n（模）减去其真值的绝对值。

② 根据反码求补码（推荐使用方法）。

$$[x]_{补}=[x]_{反}+1$$

③ 根据原码求补码。

负数的补码等于其反码加 1，这也可理解为负数的补码等于其原码各位（除符号位外）取反并在最低位加 1。如果反码的最低位是 1，它加 1 后就变成 0，并产生向次最低位的进位。如次最低位也为 1，它同样变成 0，并产生向其高位的进位（这相当于在传递进位），这进位一直传递到第一个为 0 的位为止，于是可得到这样的转换规律：从反码的最低位起直到第一个为 0 的位以前（包括第一个为 0 的位），一定是 1 变 0，第一个为 0 的位以后的位都保持不变，由于反码是由原码求得，因此可得从原码求补码的规律为：从原码的最低位开始到第一个为 1 的位之间（包括此位）的各位均不变，此后各位取反，但符号位保持不变。

特别指出，在计算机中凡是带符号的数一律用补码表示且符号位参加运算，其运算结果也是用补码表示，若结果的符号位为“0”，表示结果为正数，此时可以认为它是以原码形式表示的（正数的补码即为原码）；若结果的符号位为“1”，表示结果为负数，它是以补码形

式表示的，若是求解该运算结果，必须还原为原码，即对该结果求补，即：

$$[[x]_{补}]_{补}=[x]_{原}$$

1.1.3 微型计算机中常用编码

由于微型计算机不但要处理数值计算问题，还要处理大量非数值计算问题。因此，除了直接给出二进制数外，不论是十进制数还是英文字母、汉字以及某些专用符号都必须编成二进制代码，这样它们才能被计算机识别、接收、存储、传送及处理。

1. 十进制数的编码

在微型计算机中，十进制数除了转换成二进制数外，还可用二进制数对其进行编码：用 4 位二进制数表示 1 位十进制数，使它既具有二进制数的形式又具有十进制数的特点。二-十进制码又称为 BCD 码(Binary-Coded Decimal)，有 8421 码、5421 码、2421 码以及余 3 码等几种，其中最常用的是 8421 码。8421 码与十进制数的对应关系如表 1.2 所示，每位二进制数位都有固定的“权”，各数位的权从左到右分别为 2^3、2^2、2^1、2^0，即 8、4、2、1，这与自然二进制数的权完全相同，故 8421BCD 码又称为自然权 BCD 码。其中 1010～1111 这 6 个代码是不允许出现的，属非法 8421BCD 码。

表 1.2 8421BCD 码编码

十进制数	8421BCD 码	十进制数	8421BCD 码
0	0000	5	0101
1	0001	6	0110
2	0010	7	0111
3	0011	8	1000
4	0100	9	1001

由于 BCD 码低位与高位之间是“逢十进一”，而 4 位二进制数(即十六进制数)是“逢十六进一”，因此用二进制加法器进行 BCD 码运算时，如果 BCD 码运算的低、高位的和都在 0～9，其加法运算规则与二进制加法完全一样；如果相加后某位(BCD 码位，低 4 位或高 4 位)的和大于 9 或产生了进位，此位应进行“加 6 调整”。通常在微型计算机中都设置 BCD 码的调整电路，机器执行一条十进制调整指令，机器就会自动根据刚才的二进制加法结果进行修正。由于 BCD 码向高位借位是“借一当十”，而 4 位二进制数(1 位十六进制数)是“借 1 当 16”，因此在进行 BCD 码减法运算时，如果某位(BCD 码位)有借位时，必须在该位进行“减 6 调整”。

2. 字符编码

由于微型计算机需要进行非数值处理(如指令、数据的输入、文字的输入及处理)，必须对字母、文字以及某些专用符号进行编码。微型计算机系统的字符编码多采用美国信息交换标准代码——ASCII 码(American Standand Code for Information Interchange)，ASCII 码是 7 位代码，共有 128 个字符，详见附录 A 所示。其中 94 个是图形字符，可在字符印刷或显示设备上打印出来，包括数字符号 10 个、英文大小写共 52 个以及其他字符 32 个，另外 34 个是控制字符，包括传输字符、格式控制字符、设备控制字符、信息分隔符

和其他控制字符，这类字符不打印、不显示，但其编码可进行存储，在信息交换中起控制作用。其中，数字 0～9 对应的 ASCII 码为 30H～39H，英文大写字母 A～Z 对应的 ASCII 码为 41H～5AH，小写字母 a～z 对应的 ASCII 码为 61H～7AH，这些规律性对今后的码制转换的编程非常有用。

我国于 1980 年制定了国家标准 GB 1988—80，即《信息处理交换用的 7 位编码字符集》，其中除用人民币符号“￥”代替美元符号“$”外，其余与 ASCII 码相同。

1.2　微型计算机原理

1946 年 2 月 15 日，第一台电子数字计算机（Electronic Numerical Integrator and Computer，ENIAC）问世，标志着计算机时代的到来。

ENIAC 是电子管计算机，体积庞大，时钟频率仅有 100kHz。与现代计算机相比，ENIAC 的各方面性能都显得微不足道，但它的问世开创了计算机科学的新纪元，对人类的生产和生活方式产生了巨大影响。

1946 年 6 月，匈牙利籍数学家冯·诺依曼提出“程序存储”和“二进制运算”的思想，进一步构建了由运算器、控制器、存储器、输入设备和输出设备组成的这一经典的计算机结构，如图 1.3 所示。电子计算机技术的发展，相继经历了电子管计算机、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计算机 5 个时代，但是，计算机的结构始终没有突破冯·诺依曼提出的计算机的经典结构框架。

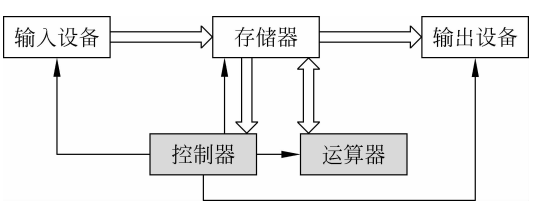


图 1.3　电子计算机的冯·诺依曼经典结构

1.2.1　微型计算机的基本组成

随着集成电路技术的飞速发展，1971 年 1 月，Intel 公司的德·霍夫将运算器、控制器以及一些寄存器集成在一块芯片上，此即为微处理器或中央处理单元，简称 CPU，形成了以微处理器为核心的总线结构框架。

如图 1.4 所示为微型计算机的组成框图，由微处理器、存储器（ROM、RAM）和输入/输出接口（I/O 接口）和连接它们的总线组成。微型计算机配上相应的输入/输出设备（如键盘、显示器）就构成了微型计算机系统。

1）微处理器（中央处理单元，CPU）

微处理器由运算器和控制器两部分组成，是计算机的控制核心。

（1）运算器

运算器由算术逻辑单元（ALU）、累加器和寄存器等几部分组成，主要负责数据的算

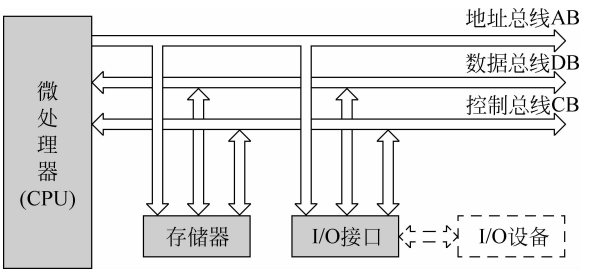


图 1.4 微型计算机组成框图

术运算和逻辑运算。

(2) 控制器

控制器是发布命令的“决策机构”,即协调和指挥整个计算机系统操作。控制器由指令部件、时序部件和微操作控制部件 3 部分组成。

指令部件是一种能对指令进行分析、处理和产生控制信号的逻辑部件,是控制器的核心。通常,指令部件由程序计数器(Program Counter,PC)、指令寄存器(Instruction Register,IR)和指令译码器(Instruction Decode,ID)3 部分组成。

时序部件由时钟系统和脉冲发生器组成,用于产生微操作控制部件所需的定时脉冲信号。

微操作控制部件是根据指令译码器判断出的指令功能后,形成相应的伪操作控制信号,用以完成该指令所规定的功能。

2) 存储器(RAM、ROM)

通俗来讲,存储器是微型计算机的仓库,包括程序存储器和数据存储器两部分。程序存储器用于存储程序和一些固定不变的常数和表格数据,一般由只读存储器(ROM)组成;数据存储器用于存储运算中输入、输出数据或中间变量数据,一般由随机存取存储器(RAM)组成。

3) 输入/输出接口(I/O 接口)

微型计算机的输入/输出设备(简称外设,如键盘、显示器等),有高速的,也有低速的,有机电结构的,也有全电子式的,由于种类繁多且速度各异,因而它们不能直接同高速工作的 CPU 相连。输入/输出接口(I/O 接口)是 CPU 与输入/输出设备(简称外设,如键盘、显示器等)的连接桥梁,I/O 接口的作用相当于一个转换器,保证 CPU 与外设间协调工作。不同的外设需要不同的 I/O 接口。

4) 总线

CPU 与存储器、I/O 接口是通过总线相连的,包括地址总线、数据总线与控制总线。

(1) 地址总线(AB)

地址总线用作 CPU 寻址,地址总线的多少标志着 CPU 的最大寻址能力。若地址总线的根数为 16,即 CPU 的最大寻址能力为 $2^{16}=64K$。

(2) 数据总线(DB)

数据总线用于 CPU 与外围器件(存储器、I/O 接口)交换数据,数据总线的多少标志 CPU 一次交换数据的能力,决定 CPU 的运算速度。CPU 的位数就是指数据总线的宽

度，如 16 位机，就是指计算机的数据总线为 16 位。

（3）控制总线（CB）

控制总线用于确定 CPU 与外围器件交换数据的类型，主要为读和写两种类型。

1.2.2　指令、程序与编程语言

一个完整的计算机是由硬件和软件两部分组成的，缺一不可。上面所述为计算机的硬件部分，是看得到、摸得着的实体部分，但计算机硬件只有在软件的指挥下，才能发挥其效能。计算机采取“存储程序”的工作方式，即事先把程序加载到计算机的存储器中，当启动运行后，计算机便自动地按照程序进行工作。

指令是规定计算机完成特定任务的命令，微处理器就是根据指令，指挥与控制计算机各部分协调地工作。

程序是指令的集合，是解决某个具体任务的一组指令。在用计算机完成某个工作任务之前，人们必须事先将计算方法和步骤编制成由逐条指令组成的程序，并预先将它以二进制代码（机器代码）的形式存放在程序存储器中。

编程语言分为机器语言、汇编语言和高级语言。

（1）机器语言是用二进制代码表示的，是机器能直接识别和执行的语言。因此，用机器语言编写的程序称为目标程序。机器语言具有灵活、直接执行和速度快的优点，但可读性、移植性以及重用性较差，编程难度较大。

（2）汇编语言用英文助记符描述指令，是面向机器的程序设计语言。采用汇编语言编写程序，既保持了机器语言的一致性，又增强了程序的可读性并且降低了编写难度，但使用汇编语言编写的程序，机器不能直接识别，还要由汇编程序或者叫汇编语言编译器转换成机器指令。

（3）高级语言是采用自然语言描述指令功能的，与计算机的硬件结构及指令系统无关，它有更强的表达能力，可方便地表示数据的运算和程序的控制结构，能更好地描述各种算法，而且容易学习掌握。但高级语言编译生成的程序代码一般比用汇编程序语言设计的程序代码要长，执行的速度也慢。高级语言并不是特指的某一种具体的语言，而是包括很多编程语言，如目前流行的 Java、C、C++、C＃、Pascal、Python、Lisp、Prolog、FoxPro、VC，这些语言的语法、命令格式都不相同。目前，在单片机、嵌入式系统应用编程中，主要采用 C 语言编程，具体应用中还增加了面向单片机、嵌入式系统硬件操作的语句，如 Keil C（或称为 C51）。

1.2.3　微型计算机的工作过程

微型计算机的工作过程就是执行程序的过程，计算机执行程序是一条指令一条指令执行的。执行一条指令的过程分为 3 个阶段：取指、指令译码与执行指令。每执行完一条指令，自动转向下一条指令的执行。

1. 取指

根据程序计数器 PC 中的地址，到程序存储器中取出指令代码，并送到指令寄存器 IR 中。然后，PC 自动加 1，指向下一指令（或指令字节）地址。

2. 指令译码

指令译码器对指令寄存器中的指令代码进行译码，判断当前指令代码的工作任务。

3. 执行指令

判断出当前指令代码任务后，控制器自动发出一系列微指令，指挥计算机协调动作，完成当前指令指定的工作任务。

图 1.5 为微型计算机工作过程的示意图，程序存储器从 0000H 起存放了如下所示的指令代码。

汇编源程序	对应的机器代码
ORG 0000H	;伪指令，指定下列程序代码从 0000H 地址开始存放
MOV A, #0FH	740FH
ADD A, 20H	2520H
MOV P1, A	F590H
SJMP $	80FEH

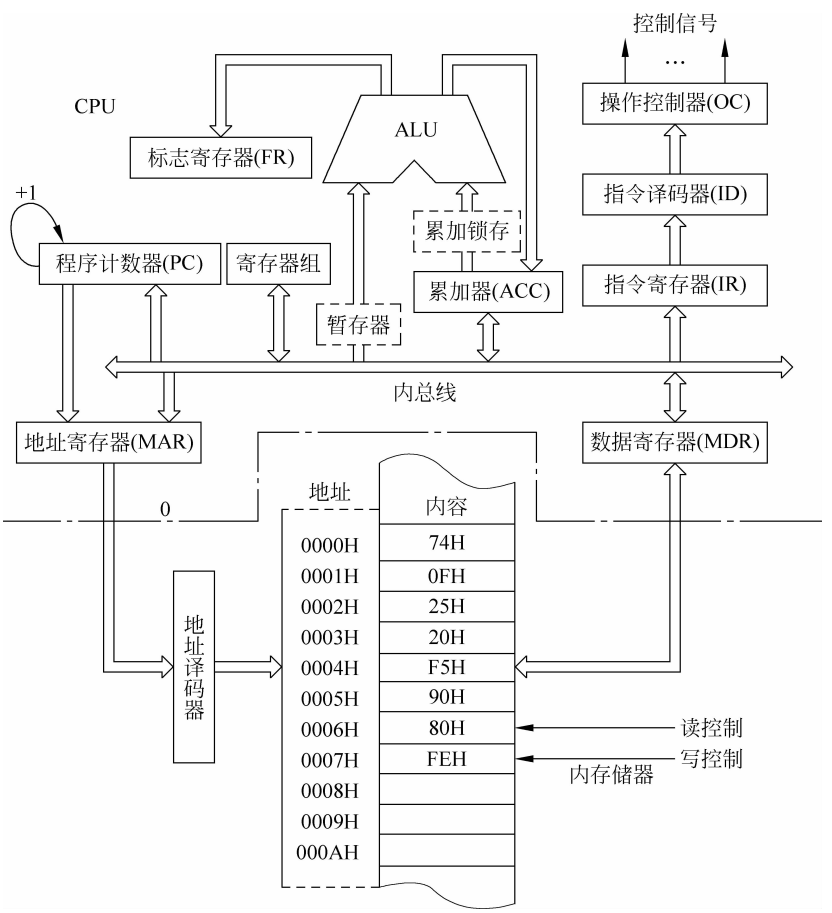


图 1.5 微型计算机工作过程示意图