胡伟轩 编

weichuli
jikong
zhixitong

微处理机控制系统

——在机械工业中的应用

微处理机控制系统

——在机械工业中的应用

胡伟轩 编

华中工学院出版社

内 容 简 介

MC6800 微处理机是目前世界上广泛应用的主要微机之一，国产 060MCS系列微机与其兼容，本书较为详细地介绍了 MC6800 微处理机控制系统的基本结构、系统各部分的基本原理和功能、指令系统、基本编程技术、输入/输出与接口技术、数据的采集与处理等等；还特别介绍了MC6800 微处理机控制系统的若干基本控制环节、调速系统及一些应用于机械工业的实例；另外，还以一定篇幅介绍了面向工业控制的 MC14500一位微机控制系统及其应用实例；最后扼要介绍了多微处理机控制系统。

本书内容丰富、实用性强，文字通俗、流畅，书中适当处还配有练习和思考题，以便于读者自学。

本书既可作为工科院校大学生、研究生的教材，也可供广大工程技术人员、科研人员和工科院校师生自学使用。

微处理机控制系统

——在机械工业中的应用

胡伟轩　编

责任编辑　黎秋萍

华中工学院出版社出版

（武昌喻家山）

新华书店湖北发行所发行

华中工学院出版社沔阳印刷厂印刷

开本：787×1092 1/16　印张：18　字数：426,000

1986年7月第一版　1986年7月第一次印刷

印数：1—3,000

统一书号：15255—058　定价：3.00 元

前　言

目前世界上正在出现一个以电子计算机为中心的新的技术革命浪潮。电子计算机的出现、发展和应用，引起了传统的生产方式、生产结构、以及社会生活等方面的变化。随着计算机技术的不断发展，这些变化将更大、更深刻。计算机的应用将不断地、更深刻地渗透到各个领域。

以微处理机为中心，配合存贮器、输入/输出接口、传感器、模/数及数/模转换器、输入/输出设备等组成的微处理机工业控制系统，由于其价格便宜、适应性强、灵活性大、能很好地满足一般工业控制系统的要求、便于推广，因此它的应用越来越广泛，是当前工业控制自动化的发展方向。

MC6800系列微处理机（与国产060MCS系列微处理机兼容）是一种具有代表性的机种。它应用很广泛，非常适合于工业控制，易于学习和掌握。它的结构、指令系统、输入/输出接口等与6502微处理机（也是一种应用很广的机种，“苹果”机上的CPU芯片就是采用6502）有很多相类似之处，它们有相同的用途。并且，6800系列的设备能较容易地与MC6800相连接，以构成一个以MC6800为主的微机控制系统。因此，本书着重介绍MC6800微处理机工业控制系统，包括它的基本结构、系统各部分的功能、指令系统、基本编程技术、输入/输出与接口技术、中断、数据的采集与处理、基本工业控制环节、调速系统以及若干工业应用实例等等。另一方面，考虑到一位微型计算机（一种面向工业控制的微型计算机）正在越来越广泛地应用于工业控制系统中，多位、多级微型计算机工业控制系统和一位微型计算机工业控制系统是当前微型计算机在工业中应用的两个重要方面和发展方向，因此，本书还较详细地介绍了MC14500（国内同类产品为5G14500、WGK-110、MIC-100等系列）一位微处理机工业控制系统的基本结构、指令系统、基本编程方法及若干应用例子。这样就可同时满足读者学习多位微型计算机控制系统和一位微型计算机控制系统的要求，从而扩大了本书的适用范围。本书最后还扼要介绍了多微处理机工业控制系统及其设计方法。

本书虽以介绍MC6800微处理机为主，但所叙述的基本原理和方法同样适用于其它类型的微处理机工业控制系统。只要读者完全掌握了书中的内容，就能举一反三，不难掌握和应用其它类型的微处理机系统了。书中所列举的例子都是工业控制系统中常见的基本控制环节和控制系统。

本书力求通俗易懂，讲清基本概念，并注重实用，以使读者学习之后对多位或一位微处理机控制系统有一个清楚的、完整的概念，并能掌握它的基本应用技术。本书适宜于作为从事微型计算机工业应用，尤其是在机械工业控制系统中的应用的广大工程技术人员、科研人员、教师、学生和工人学习微型计算机工业控制系统的自学教材，也可作为机械类大学生、研究生学习微型计算机工业控制系统的教材。

本书承蒙华中工学院朱涵梁副教授、马德华讲师进行了认真的详细的审查，提出许多极其宝贵的意见，在此表示衷心感谢。

在编写过程中，作者虽然翻阅了大量的国内外有关书籍、手册和文章，但由于水平有限，书中难免有许多缺点和错误，恳切希望读者批评指正。

编　者

1985年5月于华中工学院

目　录

第一章 数 字 制

1-1 概 述

本章将扼要地叙述在计算机技术中常用的几种数字制：二进制、二-十进制和十六进制。掌握计算机技术中几种常用的数字制，是使用计算机的基本功。若读者对本章内容已经熟悉，则可直接阅读第二章。

1-2 二 进 制

1. 二进制(Binary)

二进制数的特点是：

(a) 二进制数据或代码的每一数位称为位(Bit)。二进制数的每位只有两种可能值：0或1，例如二进制数10110101。

(b) 加法运算时逢二进一。减法运算时借一为二。

在数字电路中，一定范围的电压值被定义为逻辑0，而另一定范围的电压值则被定义为逻辑1。对TTL (Transistor-Transistor Logic晶体管-晶体管逻辑电路) 来说，0—0.8V之间的电压就用逻辑0表示，而2—5V之间电压就用逻辑1来表示。

在计算机中，一组数码是作为一个整体来处理或运算的，称为一个计算机字，简称字(Word)。每一个字所包含的位数称为字长。不同的计算机有不同的字长，但每一种类型的计算机都有固定的字长。通常，大型计算机的字长较长，例如为32位；小型计算机用中等字长，例如为16位；微型计算机通常用8位字长，但也有用16位、32位和4位的；在一位微型计算机中，运算数据的字长只有一位。

计算机的字通常分成若干个字节。字节也是存贮器容量的常用单位。通常是8位作为一个字节(Byte)，例如二进制数11010011就是一个字节。若某存贮器的容量为1021单元/每单元8-bit，则称为1024字节或1K字节。

在任一8位二进制数$b_7b_6b_5b_4b_3b_2b_1b_0$中，b_0称为最低有效位(Least Significant Bit，简写为LSB)，而b_7则称为最高有效位(Most Significant Bit，简写为MSB)。

和十进制数一样，二进制数中每一位都有一个权(Power)。例如二进制数$b_7b_6b_5b_4b_3b_2b_1b_0$，最低位b_0的权是2^0，b_1的权是2^1，b_2的权是2^2，…，b_7的权是2^7。即各位的权分别为$2^0=1$，$2^1=2$，$2^2=4$，$2^3=8$，$2^4=16$，$2^5=32$，$2^6=64$，$2^7=128$。

对于小数，例如$.b_{-1}b_{-2}b_{-3}$，它们对应的权为2^{-1}，2^{-2}，2^{-3}。

2. 二进制数变换为十进制数

只要把二进制数中的每位数乘上它对应的权，然后把它们相加起来，就能将二进制数变换为十进制数。

例

$01101010_2 = 0\times2^7+1\times2^6+1\times2^5+0\times2^4+1\times2^3+0\times2^2+1\times2^1+0\times2^0 = 106_{10}$

$110.101_2 = 1\times2^2+1\times2^1+0\times2^0+1\times2^{-1}+0\times2^{-2}+1\times2^{-3} = 6.625_{10}$

注：下标2表示是二进制数，下标10表示是十进制数。

3. 十进制数变换为二进制数

对于整数，可用重复相除法。

例　变十进制整数95为二进制数。

	余数	
2|95		
2|47	……1	（最低位）
2|23	……1	
2|11	……1	
2|5	……1	↑
2|2	……1	|
2|1	……1	|
0	……1	（最高位）

即十进制数95的二进制表示式为1011111。

对于分数，可用重复相乘法。

例　变十进制数0.8125为二进制数。

	0.8125
	× 2
1 ……	1.6250
	× 2
1 ……	1.2500
	× 2
0 ……	0.5000
	× 2
1 ……	1.0000

↓

即十进制数0.8125的二进制表示式为.1101。

4. 二进制数的运算

(1) 加法运算：逢二进一

例

```
    111 111 11 ……进位
     101001.101……被加数
+    011010.011……加数
-------------------
    1000100.000……和
```

(2) 减法运算：借一为二

例

```
    11  1      ……借位
    1100101    ……被减数
-    110011    ……减数
-------------
    0110010    ……结果
```

5. 正负数的表示

对于8位二进制数，通常用其最高位(b_7)作为符号位来表示数的正负。 如图1-1所示。

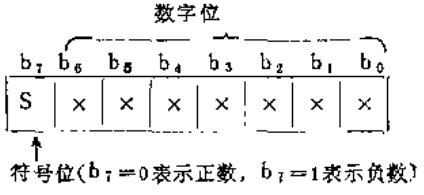

图1-1 正负数的表示

例如：00100101_2是一个正数，表示十进制数+37。

10100101_2是一个负数，表示十进制数−37。

但用符号位来表示一个二进制数的正负，在运算时有时会产生错误的结果。

例 $+7_{10}$加-5_{10}

```
    00000111₂
+ 10000101₂
-----------
10001100₂ (−12₁₀)
```

正确的结果应为$+2_{10}$，但运算结果却为-12_{10}，显然这是错误的。因此，在进行运算时常用2的补码来表示二进制数。下面介绍二进制数的几种表示形式。

6. 1的补码(One's Complement)

在1的补码（即反码）表示式中，所有正数都用一般的二进制数来表示，例如 $+2_{10}$就用00000010_2表示；负数则用它的反码来表示，例如-2_{10}，则用11111101_2(即00000010_2的反码)来表示。反码就是将0变为1， 1变为0。

注意：在这种表示方式中，正数的最高位$b_7=0$，负数的最高位$b_7=1$。

7. 2的补码(Two's Complement)

用2的补码来表示一个二进制数时，同样，用最高位(b_7)为0来表示正数，用最高位(b_7)为1来表示负数。

在2的补码表示方式中，一个正二进制数用一般的二进制数形式来表示，其最高位$b_7=0$。例如$+5_{10}$用00000101_2来表示，一个负二进制数用2的补码表示时，可用如下方法写出。

(a) 先写出与这个负二进制数对应的正二进制数的表示式；

(b) 求这个正二进制数的反码；

(c) 反码加 1；

(d) 若最高位(b_7)有进位，则丢掉此进位。

例　求 $+25_{10}$ 和 -25_{10} 的 2 的补码表示式。

解　(a) $+25_{10}=00011001_2$　　($b_7=0$)

(b) -25_{10}

因　(i) $+25_{10}=00011001_2$

(ii) $+25_{10}$ 的反码：11100110_2

(iii) 反码加 1：$11100110_2+1=11100111_2$

故 -25 的 2 的补码为 11100111_2　　($b_7=1$)

求某一个负二进制数的 2 的补码表示式的另一种方法就是用 0 减原码（即数的二进制表示式）。

例　求 -25_{10} 的 2 的补码

借位（丢掉）：1

$$\begin{array}{r} 00000000_2 \quad (0) \\ -\ 00011001_2 \quad (25_{10}) \\ \hline 11100111_2 \quad (-25_{10}) \end{array}$$

其结果与上述方法所求结果一样。

因为负数常以 2 的补码形式表示，因此，在得出结果时往往还需将其 2 的补码变回它的原码。例如，设运算结果为 11110100_2，因为 $b_7=1$，因此它是一个负数，是以 2 的补码形式来表示的数。要知道这个负数的实际值，还需求出它的原码。求原码的方法是：先求反码，然后加 1。

11110100_2 的反码为 00001011_2

反码加 1：$00001011_2+1=00001100_2$

这是十进制数的 $+12$，因此 $11110100_2=-12_{10}$。

8. 用2的补码表示的数的范围

对 8 位二进制数来说，以 2 的补码形式表示的最大正数为：$01111111_2=127_{10}$。

以 2 的补码形式表示的最大负数为：$10000000_2=-128_{10}$。

因此，在 $+127_{10}$ 与 -128_{10} 之间的数均可用 2 的补码形式来表示。

9. 用2的补码形式来进行二进制数的运算

用 2 的补码形式来表示一个二进制数时，二进制数的加减运算就变得比较简单，且不会出现第 5 点所述的错误。

例1 求 $19_{10}-11_{10}=?$

解　因 $19_{10}-11_{10}=19_{10}+(-11_{10})$

$+19_{10}$ 的 2 的补码表示式为：00010011_2

(-11_{10}) 的 2 的补码表示式为：

$$+11_{10}=00001011_2$$

$+11_{10}$的反码 = 11110100

反码加 1：$11110100_2 + 1 = 11110101_2$

即(-11_{10})的 2 的补码表示式为11110101_2

所以　　00010011_2……$(+19_{10})$

$+11110101_2$……(-11_{10})

$1\ 00001000_2$……$(+8_{10})$

进位(丢掉)

即　$(+19_{10})+(-11_{10})=8_{10}$

例2　求$(-13_{10})+(-19_{10})=?$

解　先分别找出-13_{10}和-19_{10}的 2 的补码表示式。

-13_{10}的 2 的补码表示式：

$13_{10}=00001101_2$

$\overline{13}_{10}=11110010_2$　　($\overline{13}_{10}$表示13_{10}的反码)

$-13_{10}=11110011_2$

-19_{10}的 2 的补码表示式：

$19_{10}=00010011_2$

$\overline{19}_{10}=11101100_2$　($\overline{19}_{10}$表示19_{10}的反码)

$-19_{10}=11101101_2$

(-13_{10})与(-19_{10})相加：

11110011_2……(-13_{10})

$+11101101_2$……(-19_{10})

11100000_2

因为$b_7=1$，所以相加结果是以 2 的补码形式表示的负数。它的原码为：

$00011111_2+1=00100000_2$

结果为-32_{10}。

10，溢出(Overflow)

在讨论什么叫溢出之前先看几个例子。

例1　求$64_{10}+66_{10}=?$

解

1　　……进位

01000000_2……(64_{10})

$+01000010_2$……(66_{10})

10000010_2……(-126_{10})

64_{10}加66_{10}本应等于130_{10}(注意：$+130_{10}>+127_{10}$)，但运算结果却为-126_{10}，显然是不正确的。这是因为这两个正数相加的结果大于 8 位寄存器所能容纳的正数的最大值$+127_{10}$，从而使b_6位向b_7位产生一个进位，这个进位影响了符号位（位b_7），产生了错误的结果。这种情况通常称为上溢。

例2　求$(-64_{10})+(-66_{10})=?$

解

$$\begin{array}{l} 11000000_2\cdots\cdots(-64_{10}) \\ +10111110_2\cdots\cdots(-66_{10}) \\ \hline 1\ 01111110_2\cdots\cdots(+126_{10}) \end{array}$$

$(-64_{10})+(-66_{10})$本应等于-130_{10}(注意：$-130_{10}<-128_{10}$)，但运算结果却为$+126_{10}$，显然是不正确的。这是因为这两个负数相加的结果小于-128_{10}，从而使b_7位向高位产生一个进位，这个进位影响了符号位，因而产生了错误的结果。这种情况通常称为下溢。

例3 求$(-1_{10})+(-1_{10})=?$

解

$$\begin{array}{l} 11111111_2\cdots\cdots(-1_{10}) \\ +11111111_2\cdots\cdots(-1_{10}) \\ \hline 1\ 11111110_2\cdots\cdots(-2_{10}) \end{array}$$

结果为-2_{10}，这是正确的。

从例1和例2可见，当运算结果大于$+127_{10}$或小于-128_{10}时就会产生溢出，致使符号位受到影响，从而产生错误的结果。

从例3可见，位b_6虽然向位b_7产生一个进位，但同时位b_7也向高位产生了一个进位，因而符号位并未受到影响，其运算结果是正确的。

从上面几个例子可以看出，在下列四种情况下会产生溢出：

(a) 大的正数相加，其和大于$+127_{10}$；

(b) 大的负数相加，其和小于-128_{10}；

(c) 从一个大的负数中减去一个大的正数；

(d) 从一个大的正数中减去一个大的负数。

在计算机内部通常有一标志位用来标志这种溢出状态，详见第三章条件码寄存器一节。

表1-1 BCD表

二进制码	十进制数
0 0 0 0	0
0 0 0 1	1
0 0 1 0	2
0 0 1 1	3
0 1 0 0	4
0 1 0 1	5
0 1 1 0	6
0 1 1 1	7
1 0 0 0	8
1 0 0 1	9
1 0 1 0	不用
1 0 1 1	
1 1 0 0	
1 1 0 1	
1 1 1 0	
1 1 1 1	

1-3 二-十进制（BCD）表示法

所谓BCD(Binary Coded Decimal)表示法就是用二进制码来表示十进制数。因为在十进制数中每位都有十个状态(即十个数码0,1,2,3,4,5,6,7,8,9)，因此必须用四位二进制码来表示十进制数中的一位。但四位二进制码有十六个状态(即从0000_2—1111_2，亦即从0_{10}—15_{10})，因此只用前面十个状态(即0000_2—1001_2)来表示十进制数中的一位，而后面六个状态(即1010_2—1111_2)就不用。表1-1就是BCD表。

例如，一个十进制数15用BCD码表示时，可写为00010101。

$$\underbrace{0001}_{1}\quad\underbrace{0101}_{5}$$

关于BCD码的加法运算请参阅3-3节。

1-4 十 六 进 制

1. 概述

多数微处理机系统都是采用16位地址总线和8位数据总线，因此它们可以用64K存贮器。若用二进制数来表示一个16位地址，则需要16位长，显然这是很冗长的。因此微处理机系统通常都采用十六进制(Hexdecimal，常简写为Hex)数。

在十六进制中，每位都包含16个状态(即0—15)，每个状态分别用0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F等16个数码来表示。表1-2是十六进制、二进制和十进制数的对照表。在十六进制加法运算中是逢16进1，而在减法运算中是借1为16。

2. 十六进制与二进制之间的变换

十六进制与二进制之间的变换是很简单的。因为十六进制中每位可有16个数码，所以若用二进制表示，则每位需要4位二进制。

例如，将一个二进制数1100000101111111101变换为十六进制数，有

$$\underbrace{0011}_{3}\ \underbrace{0000}_{0}\ \underbrace{0101}_{5}\ \underbrace{1111}_{F}\ \underbrace{1101}_{D}$$

即$1100000101111111101_2 = 305FD_{16}$

例如，将一个16进制数1CB09变换为二进制数，有

$$\begin{array}{ccccc} 1 & C & B & 0 & 9 \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ 0001 & 1100 & 1011 & 0000 & 1001 \end{array}$$

即 $1CB09_{16} = 1110010110000 1001_2$

表1-2 十进制、二进制和十六进制对照表

十进制 (Decimal)	二进制 (Binary)	十六进制 Hexdecimal)
0	0 0 0 0	0
1	0 0 0 1	1
2	0 0 1 0	2
3	0 0 1 1	3
4	0 1 0 0	4
5	0 1 0 1	5
6	0 1 1 0	6
7	0 1 1 1	7
8	1 0 0 0	8
9	1 0 0 1	9
10	1 0 1 0	A
11	1 0 1 1	B
12	1 1 0 0	C
13	1 1 0 1	D
14	1 1 1 0	E
15	1 1 1 1	F

3. 十六进制与十进制之间的变换

(1) 将十六进制数变换为十进制数

在十六进制中，每一个十六进制数均可用下式表示：

$$H_0 \times 16^0 + H_1 \times 16^1 + H_2 \times 10^2 + H_3 \times 10^3 + \cdots$$

这里，H_0是最低位(即第一位)的数字，H_1是第二位数字，…。用这个式子可以很方便地将十六进制数变换为十进制数。例如，将十六进制数2FC变换为十进制数。

由于在这个数中，$H_0 = C$, $H_1 = F$, $H_2 = 2$

所以 $2FC_{16} = 2 \times 16^2 + F \times 16^1 + C \times 16^0 = 764_{10}$

(2) 将十进制数变换为十六进制数

可用重复相除法将十进制数变换为十六进制数，即，将十进制数用16来重复相除，每次相除所得之余数就是16进制数的一位数字。

例如将十进制数8715变换为十六进制数，有

$$
\begin{array}{l|l}
16 \mid 8715 & \text{余数} \\
16 \mid 544 & \cdots\cdots B \\
16 \mid 34 & \cdots\cdots 0 \\
16 \mid 2 & \cdots\cdots 2 \\
0 & \cdots\cdots 2
\end{array}
$$

即 $8715_{10}=220B_{16}$

(3) 小数的变换

例1 将十六进制数0.B7变换为十进制数。

解 $0.B7_{16}=B\times16^{-1}+7\times16^{-2}$

$$=\frac{11}{16}+\frac{7}{256}\approx0.715_{10}$$

例2 将十进制数0.8125变换为十六进制数。

解 第一步：将0.8125_{10}变换为二进制数

$$0.8125_{10}=0.1101_2$$

第二步：将二进制数变换为十六进制数

$$0.1101_2=0.D_{16}$$

即 $0.8125_{10}=0.D_{16}$

4. 十六进制数的加减运算

(1) 加法运算：逢十六进一

例 求$(B5)_{16}+(96)_{16}=?$

解

$$
\begin{array}{rl}
 & 1 \quad\cdots\cdots\text{进位} \\
 & B\ 5\cdots\cdots\text{被加数} \\
+ & 9\ 6\cdots\cdots\text{加数} \\
\hline
 & 1\ \ 4\ B\cdots\cdots\text{和}
\end{array}
$$

5+6等于十进制的11，即十六进制的B。B+9等于十进制的11+9=20。20大于16，所以向高位进一位。20−16=4，所以余下4。

(2) 减法运算：借一为十六

例 求$(E2)_{16}-(47)_{16}=?$

$$
\begin{array}{rl}
 & 1 \quad\cdots\cdots\text{借位} \\
 & E\ \ 2\ \cdots\cdots\text{被减数} \\
- & 4\ \ 7\ \cdots\cdots\text{减数} \\
\hline
 & 9\ \ B\ \cdots\cdots\text{差}
\end{array}
$$

因2小于7，所以2−7要向高位借位。借一位过来等于16。16+2=18，18−7=11(即16进制的B)。因为第二位已被借走了1，所以E−4实际上变成了D−4=9。

5. 16的补码(16's Complement)

(1) 16的补码

和二进制数的运算一样，若十六进制用它的补码形式来表示，则减法运算可变为加法运

算。

求16的补码时，可将16进制数先变换为二进制数，然后求2的补码，再将2的补码变换为16进制数形式，则得16的补码。

另一种求16的补码的简便方法是：

(a) 将一个能使相加的结果为16的数加到16进制数的最低位。这个数就是16的补码的最低位。

(b) 分别将某些能使每位相加的结果为15的数加到16进制数的其它各位中去，被加入的各个数就是对应位的16的补码。

(c) 若16进制数的最低位为0，则16的补码的最低位亦为0，但加到第二位的数应使其相加结果为16。

例1 求十六进制数－A3的16的补码表示式。

因为 $D+3=10_{16}$（即十进制数的16）

$5+A=F$ （即十进制数的15）

所以－A3的16的补码表示式为5D。

例2 求十六进制数－10的16的补码表示式。

因为最低位为0，所以补码的最低位亦为0。但第二位的补码应为F，因为$F+1=10_{16}$（即十进制数的16）。

所以-10_{16}的16的补码表示式为$F0_{16}$。

(2) 用补码形式进行运算

例1 求$1E_{16}-10_{16}=?$

因为-10_{16}的16的补码表示式为$F0_{16}$，所以

$$\begin{array}{r} 1E_{16} \\ +F0_{16} \\ \hline 10E_{16} \end{array}$$

进位（丢掉）

因为结果的最高位为0，表示结果为正数，所以$1E_{16}-10_{16}=0E_{16}$。

例2 求$2A_{16}-47_{16}=?$

因为-47_{16}的16的补码表示式为$B9_{16}$，所以

$$\begin{array}{r} 2A_{16} \\ +B9_{16} \\ \hline E3_{16} \end{array}$$

结果的最高位为1(因为$E3_{16}=11100011_2$，位$b_7=1$)，表示结果是个负数，所以还要求出它的实际结果，即求E3的原码。求E3的原码与求其16的补码一样，E3的原码为1D(因为$D+3=10_{16}$，$1+E=F$)。

所以$2A_{16}-47_{16}=-1D_{16}$。

第二章　微处理机系统结构

2-1 微处理机和微型计算机

1. 概述

在60年代和70年代，集成电路（Integrated Circuit，简写为IC）和大规模集成电路（Large Scale Integrated，简写为LSI）的出现，大大地推动了计算机技术的发展，使得计算机能够向大容量、高速度和微型化方向发展。1971年美国Intel公司生产了世界上第一个集成电路微处理机Intel 4004。这是4位微处理机。1974年，Intel公司又生产了8位集成电路微处理机Intel 8080，Motorola公司生产了8位集成电路微处理机MC6800。Intel 8080系列和MC6800系列是两种最常用的微处理机系列。随后，许多型号的微处理机不断出现，例如MOS Technology公司的6502，Zilog公司的Z80，Fairchild公司的F8，Rockwell公司的PPS-8等等。近几年来已经出现了16位和32位的微处理机。如Motorola MC68000，Intel 8086，Zilog Z8000，National Semicoductor NS 16000等等。由于微处理机和微型计算机的体积小，成本低，又具有多用性和灵活性，因此它在工业控制、测量仪器、企业管理、财政管理、交通运输控制，以至日常生活中都已逐渐广泛地被采用。微型计算机的应用大大地推动了工业控制、测试技术和管理水平的向前发展。家庭用的计算机也已大量生产。可以预料，在不久的将来，许多家庭都将会采用微处理机或微型计算机来执行诸如家庭财政收支记录和管理、厨房工作自动化、防火、防盗、温度调节系统控制、游戏、帮助复习功课等功能。

2. 什么叫微处理机

在讨论微处理机之前，首先介绍一下什么叫处理机（Processor，有些书叫处理器）。处理机有时也叫中央处理机（Central Processing Unit，简称为CPU）。CPU是计算机的核心部分，由算术/逻辑单元、寄存器、计数器、控制器等组成。它具有取数、译码、执行程序指令、控制、计算和作出决定等功能。计算机中的控制、计算和作出决定等都是由CPU来完成的。

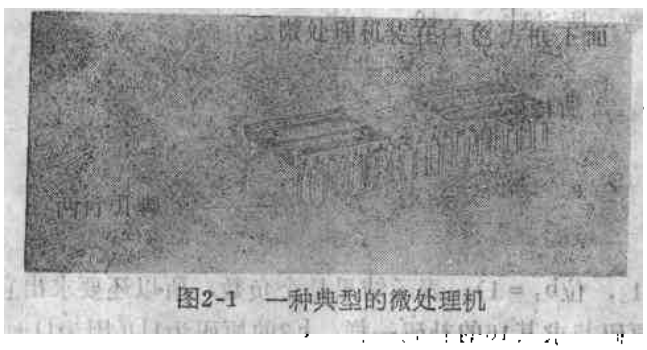

图2-1　一种典型的微处理机

而微处理机（Microprocessor或Microprocessor Unit，简写为μP或MPU。有些书称它为微处理器）却是单个的小型器件，它执行CPU的功能，但是它是单独集成在一个很小的硅片（芯片，Chip）上的。

图2-1所示是一种典型的微处理机。这是一种双列直插式组件。这个器件大约2吋长，0.5吋宽。这个尺寸是组件外壳的尺寸，实际的微处理机芯片只占其中非常小的区间（装在图中的白色方框下面）。

从图2-1可见，微处理机有两排引脚，用于插入底座。微处理机通过这些引脚、底座和其它器件相联。这种类型的组件通常称为双列直插式组件（Dual In-Line Package，简写