微处理器与微型计算机

第　一　册

〔美　国〕 BRANKO SOUČEK 著

南通电子仪器厂技术情报室　译

南京工学院计算机教研室　校

南通电子仪器厂技术情报室

说　　明

《微处理器与微型计算机》是一本教科书。内容比较全面，基础部分的论述比较系统，简明易懂，可供广大科技人员尤其是电子工程技术人员用作微处理器与微型计算机方面的入门读物。这类书籍在中文资料中目前尚为数不多。因此，尽管我们本身关于这一领域的知识还十分贫乏，但为了适应当前的急需，还是译印了此书。当然，我们的翻译是十分勉强的，错误与不妥之处在所难免，有些名词术语也不够统一，这首先是由于译者的外语水平与专业知识有限，其次也由于本书篇幅较大，译校的时间又比较仓促。

本书蒙南京工学院电子计算机教研室部分老师分工校对，他们在校对工作中表现了极大的热忱与认真负责的精神，为此，本室愿借此深表敬意与谢意。

本书付印前，又由译者对照原版校阅了全部译稿，在文字上作了一些调整，因此，所有错漏不妥之处概由本室负责，欢迎同志们批评指正。

序　言

成本低而效能高的微处理器的大批生产，是电子工程与计算机科学这两个领域中的大事。在电子工程中，微处理器可取代数字系统和机电逻辑设计，在计算机科学中，微型计算机正向小型计算机和一般计算机提出挑战。

本书首先阐述了适用于各种微处理器的一般程序设计和接口技术，然后详细介绍了一些有代表性的微处理器簇。希望本书对不少领域的工程技术人员、科学工作者及管理人员都能有所帮助。

本书可用作大学教科书，也可供从事实际工作的科技人员用作参考书。本书着重阐述功能与系统，力求简明易懂。虽然本书对不少基础知识进行了复习，但读者在有关基础知识方面起码应有大学水平。

本书共分三部分。第一部分介绍微处理器的程序设计与接口技术。首先阐述数字代码、逻辑系统、微型计算机结构、简单的十六进制程序设计、汇编语言程序设计，以及宏指令与高级语言的应用，然后集中介绍接口技术，并阐述输入－输出传送方式、直接内存存取、中断，以及接口器件和芯片。这些都是设计和使用面向微处理器的系统时所必需的基础知识。

第二部分详细介绍有代表性的微处理器簇。首先介绍简单的4－位微处理器。它能很好地代替数字系统及机电逻辑，然后介绍各种8－位微处理器。使一般计算机面临有力冲击的新型高效的16－位微处理器也在这一部分作了介绍。对每一微处理器簇，都详细介绍了它所适用的芯片、输入8－输出总线、指令系统、以及寻址方式。这一部分还为程序设计与接口技术列举了大量例子。

第三部分介绍新型微处理器及专用微型系统。首先介绍了微型芯片装置对颇有功效的小型计算机的替代，以及新近发展起来的微处理器、存贮器及输入－输出芯片。还介绍了控制应用中的高速微型系统。对可使微程控中央处理器及设备控制器结构简化的反极型LSI电路也作了介绍。

本书使用了大量图表、例题及参考资料，第一至第五章附有习题。这些例题与习题可供大学生或初学者作为使用微处理器技术的练习。总的说来，本书从初学者的水平出发，然后循序渐进地引入较深的接口、程控及应用技术领域。

目　　录

第一章　计数制与数字代码…… 1

1.1　十进制与二进制…… 1

1.2　二进制数的算术运算…… 5

1.3　八进制、二－十进制(BCD)、葛莱(Gray) …… 8

1.4　数据格式……11

第二章　逻辑运算、数字电路及微型芯片……18

2.1　基本逻辑电路……18

2.2　触发器……23

2.3　基本功能电路……27

2.4　数字系统的设计……33

2.5　集成逻辑电路……43

2.6　数字计算机……45

2.7　微处理器及存贮芯片……51

第三章　微型计算机的基本指令……57

3.1　概述……57

3.2　程序编码……59

3.3　指令分类……64

3.4　计算机基本指令系统……65

3.5　计算机基本寻址方法……74

第四章　微处理器程序设计……80

4.1　语言……80

4.2　控制操作……86

4.3　FORTRAN控制操作……90

4.4　循环……93

4.5 FORTRAN循环……100
4.6 子程序……103
4.7 FORTRAN子程序……110
4.8 算术与逻辑操作……111
4.9 FRTRAN算术……121
4.10 输入输出程序设计……122
4.11 FORTRAN输入输出……125
第五章 微处理器接口技术……129
5.1 计算机基本结构……129
5.2 指令的执行……131
5.3 程控输入输出传送……136
5.4 程序输入输出传送的接口器件……141
5.5 无条件传送、条件传送及中断程序传送……148
5.6 直接存贮存取……157
5.7 控制字与状态字……163

第 一 部 分

微处理器的程序设计与接口技术

第 一 章

计 数 制 与 数 字 代 码

本章介绍计数制与数字代码，重点介绍计算机技术中使用最广的二进制代码，阐述了二进制数的算术运算，并对信息编码中的各种代码进行了比较，对在微处理器与微型计算机程序准备阶段广泛使用的八进制与十六进制数，以及特别适用于位置转换器的葛莱(Gray)码也作了介绍，此外还规定了ASCII字符系统。

1.1 十进制与二进制

定义

计算机的基本技术要求是其表示和存贮数的能力，以及对所表示的数进行运算的能力。数可以用不同的计数制表示。绝大部分人早在童年时代就学会了用十进制计数。这种计数制以十的倍数来计数，可能是由于人有十个指头的缘故。因为使用了十个基本数字(十个指头)，所以这种计数制的基数为10，其基本数字为0，1、2、3、4、5、6、7、8、9，给这些数字以不同的数位或权，就可表示比10大的数。10是十进制的基数，我们还可用其它基数b构成其它计数制。

一般地讲，以一固定的正整数b为基数的计数制的数$(N)_b$可用数位记数法表示成

$$(N)_b=(P_n P_{n-1}\cdots P_1 P_2 \cdot P_{-1} P_{-2}\cdots)$$

b为该计数制的基数，在该计数制中，它始终是个恒定的正整数。各数位的P_i为下列整数

$$0\leqslant P_i\leqslant b-1 \qquad\qquad i=\cdots-2,\ -1,\ 0,\ 1,\ 2\cdots n$$

每一数位的值称为数位系数或权。例如：

$$
\begin{aligned}
346 = 3 \times 100 &= 300 \\
4 \times 10 &= 40 \\
6 \times 1 &= 6 \\
\hline
&346
\end{aligned}
$$

简单的十进制加权表为

$$\cdots 10^3 10^2 10^1 10^0 \cdot 10^{-1} 10^{-2} \cdots$$

总之，在基数为b的计数制中各相邻数位的权自左向右为

$$\cdots b^3 b^2 b^1 b^0 \cdot b^{-1} b^{-2} \cdots$$

符号“·”称为小数点，小数点右边称为数的小数部分，左边称为整数部分。在十进制中，小数点称为十进制小数点。

机器可以以任何计数制为基础来设计，但所有现代数字计算机都是以二进制（基数为2）为基础的。为什么要使用这种新的计数制呢？这是因为区别两个实体要比区别十个实体容易。大多数物理量只具有两个状态：电灯发光或熄灭，开关的接通或断开，物质是磁化的或是去磁的，电流是正的或负的，纸带或卡片有没有穿孔等等。设计一个只能区别两种状态（二进制的0、二进制的1）的电路要比设计一个能区别十种状态（十进制的0到9）的电路来得简便並可靠得多。

二进制的基数是2，其小数点称为二进制小数点，可取的数字为0和1，各相邻数位（自左至右）的权为

$$\cdots 2^3 2^2 2^1 2^0 \cdot 2^{-1} 2^{-2} 2^{-3} \cdots$$

按此加权表可把二进制数换算成较熟悉的十进制数，例如，求与二进制数 10101 等值的十进制数（表1.1）。

表1.1

2^4	2^3	2^2	2^1	2^0	（加权表）
1	0	1	0	1	（二进制）

数位系数	
1 × 1	= 1
0 × 2	= 0
1 × 4	= 4
0 × 8	= 0
1 × 16	= 16
十进制	= 21

十进制与二进制的异同点可由其计数过程说明，十进制的计数方法是各数位上的数都按0，1，2，…8，9 的次序递增，当这一数位的数达到 0(10)时，就向左邻的数位进1，因

为二进制使用两个数字，各数位只经历两次变化就要向左邻数位进一1 。因此，与十进制数字 0～10对应的二进制数如表1.2所示。

表1.2

十进制	二进制
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001
10	1010

在计算机技术中，广泛使用着一些特殊的名词，如“位”、“字节”、“字”。

位　二进制的一个数位通常称为“位”(bit)。因此，1010这个数就称为 4 －位 二 进 制数，101称为 3 －位数，依此类推。数左端的称为最高有较位，其权最大；数右端的位称为最低有较位，其权最小。图1.1表示一个由16位组成的二进制数。

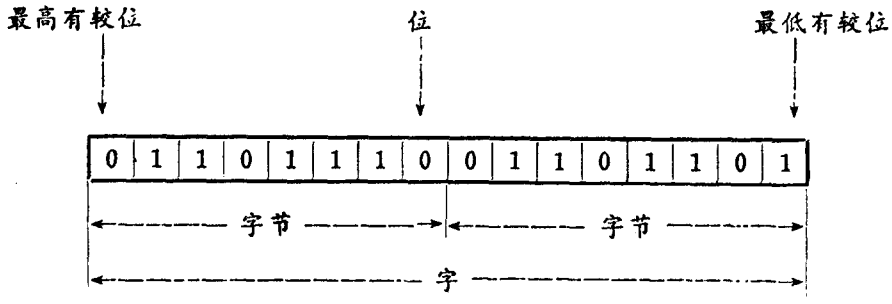


图1.1　　位、字节、字

字节　计算机和数据设备的发展，使器件间的信息交换能以 8 －位为单位，这个 8 －位单位称为“字节”。许多新型数字计算机及控制器都以此表 8 、16、24、32 位的数(即 1 、2 、3 、4 个字节)。图1.1表示一个由两个字节组成的二进制数。

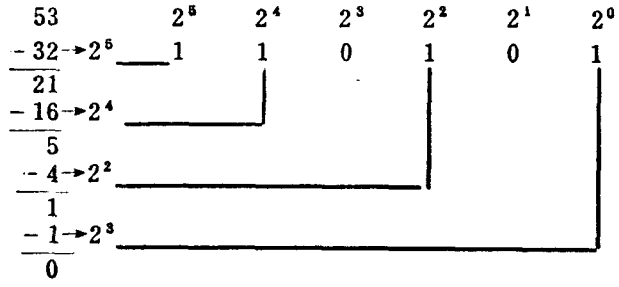
字　计算机有许多存贮二进制信息的存贮单元或寄存器，同一机器中，大部分寄存器的

长度n都相同，每一寄存器都可存贮n位二进制信息。存贮在一个寄存器中的信息称为一个“字”。图1.1表示的是一个16－位计算机字。

十进制到二进制的换算

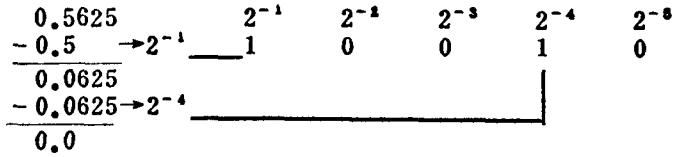
一般地说，十进制数具有一个整数部分与一个小数部分，每一部分都应分别换算成相当的二进制数，用一个二进制小数点把两个部分结合起来就是一个完整的二进制数。把十进制数换算成相当的二进制数的方法通常有两种，即减法及乘除法等。

减法　（本书按参考文献4的方法）从十进制数中减去2的最高可能，幂而给二进制数的适当加权位置记作1。这样连续运算下去，直到十进制数变成0为止。若在第一次相减后，余数不足2的低一次幂，则给适当的加权位置记作0。现举一整数换算的例子如下：



于是$(53)_{10}=(110101)_2$

下面是小数换算的例子



除法　十进制整数要换成相当的二进制数时，可采用连除2的方法。用2除十进制数时，若有余数，则给二进制数的最低位记作1，若无余数，则记作0，再用2除第一次所得的商。如此重复下去，直到商数为0（本书按参考文献4的方法）。例如：

		余数	
2	53	→ 1	
2	26	→ 0	$(53)_{10}=(110101)$
2	13	→ 1	
2	6	→ 0	
2	3	→ 1	
2	1	→ 1	
	0		

乘法　十进制小数要换算成相当的二进制数时，可用2连乘。若乘积小于1，则最高有效位为0，若乘积大于1，则最高有效位为1，第二位数字可按相同的法则，对第一次所得乘积的小数部分进行运算，如此重复下去，直到所需精度为止。例如：

	进位
$0.5625\times 2=1.1250$	1
$0.1250\times 2=0.2500$	0
$0.250\times 2=0.5$	0
$0.5\times 2=1.0$	1
$0.0\times 0=0.0$	0

于是$(0.5625)_{10}=(.10010)_2$。

1.2　二进制数的算术运算

加法

二进制加法与十进制加法相同，但它不是在和数达到10、而是在和数达到2(1+1)时就进位的。例如：

$$\begin{array}{r} 101=5_{10} \\ +\ 010=2_{10} \\ \hline 111=7_{10} \end{array}$$

$$\begin{array}{r} 11\leftarrow\text{进位} \\ 111=7_{10} \\ +\ 101=5_{10} \\ \hline 1100=12_{10} \end{array}$$

让我们看第二个例子(111加101)。第一列1+1=0而进位1，第二列，1+进位的1=0而再进位1，第三列，1+1=0，进位1，並加上先前的进位1，即1+1+1=11，故得数为1100(十进制的12)，这正是7+5的正确结果。

数字式机器中二进制数的加法是由一个称为功能加法器的特殊单元进行的。

减法

二进制数可按十进制减法的方法直接相减。设计一台既有功能减法器而又有功能加法器的机器是完全可能的，但计算机的设计人员並不采取这种方法。进行减法运算时，只需把减数符号变更一下即可作加法运算。唯一的问题是要找到表示负数的适当方法。

为了了解负数在计算机中是怎样表示的，可用一只如车用里程计之类的机械寄存器为例。此寄存器正转时，作加法运算，反转时，就作减法运算。下面以一个五位寄存器的反转为例：

```
0 0 0 0 4
0 0 0 0 3
0 0 0 0 2
0 0 0 0 1
0 0 0 0 0
9 9 9 9 9
9 9 9 9 8
9 9 9 9 7
```

数99997与－3是对应的。为了证明这一点，可作如下的加法运算：

```
  0 0 0 0 4
+ 9 9 9 9 7
-----------
1 0 0 0 0 1
```

若向左进位的1略去不计，则实际上就是做的减法运算：

$$4 - 3 = 1$$

本例的数99997称为3的十进制补码(或补数)。所以，在十进制中，若负数用十进制补码表示，负号即可去掉。

同样，在数字式机器中，可把二进制的负数表示成二进制补码后再进行减法运算。

一个数的二进制补码是这样的一个数，它与该数之和是个1。例如：二进制数010 110 110 110的二进制补码等于101 001 001 010，其加法运算如下：

```
  010 110 110 110
+ 101 001 001 010
-----------------
 1000 000 000 000
```

求一个数的二进制补码时，可按下列两步进行：

1.先求二进制反码，即把所有的位都改成相反的值，

```
101 110 110 110    数
101 001 001 001    二进制反码
```

2.二进制反码加1即为二进制补码

```
  101  001  001  001     二进制反码
+                 1     加1
-------------------------
  101  001  001  010     二进制补码
```

下面举例说明减法运算

```
        7 - 3 = 4                    12 - 5 = 7
   0011      3₁₀                0101      5₁₀
   1100      3的二进制反码      1010
   1101      3的二进制补码      1011
+                            +
   0111      7₁₀                1100      12₁₀
 1 0100      4₁₀              1 0111      7₁₀
```

乘法

在二进制乘法中、每使用下一个乘数，部分积就向左移一位，若乘数为 0，则部分积为 0，若乘数部分为 1，则部分积等于被乘数。例如：

```
5×3=15          5×5=25          5×10=50
 101   5₁₀       101   5₁₀        101   5₁₀
  11   3₁₀      101   5₁₀       1010   10₁₀
 101             101             101
101             000             000
1111  15₁₀      101             101
                11001  25₁₀     101
                               110010  50₁₀
```

除法

根据二进制减法与乘法的规则，二进制除法可按与十进制除法相同的方式进行，例如：

```
      18 : 2 = 9              10 : 5 = 2              14 : 4 = 3.5
       1001  9₁₀                  10  2₁₀                  11.1  3.5₁₀
2₁₀  10)10010  18₁₀       5₁₀  101)1010  10₁₀      4₁₀  100)1110.0  14₁₀
      10                        101                      100
      00                        000                      110
      00                        000                      100
      01                         0                       100
      00                                                 100
      10                                                  0
      10
       0
```

1.3　八进制、二－十进)制BCD)、葛莱(Gray)码。

八进制

把一个数表示成二进制数时，其位数要比表示成十进制数时多得多，例如，$(35)_{10}=(100011)_2$，因此，在读写较大的二进制数时很容易搞错。为简化二进制数的表示方法，可采用八进制。八进制的基数为 8，所用数字为0到7。与十进制数字0到9对应的八进制数如表1.3所示。

表1.3

十进进	二进制	八进制
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6
7	111	7
8	1000	10
9	1001	11
10	1010	12

由于八进制的基数为$8=2^3$，所以，把二进制数换算成八进制数时需把二进制数按数位每三位分为一组，每个3－位组均可用表1.3所给的等值八进制数字表示，例如：

110101111001	二进制数
110 101 111 001	3－位组
6　5　7　1	与各3－位组相当的八进制数

因此，$(110\ 101\ 111\ 001)_2=(6571)_8$

十进制数到八进制数的换算，以及八进制数的算术运算可按二进制数的同样法则进行。

必须注意，计算机不是以八进制进行操作的，而是按二进制操作的，使用八进制只是为了避免读写较大的二进制数。

二－十进制(BCD)

显然，人最熟悉的是十进制。因此，不少计算机输入输出设备都是以这样的方式操作

的；在计算机一方，所传送或接收的是二进制数，而在与操作人员衔接的一方，则传送或接收十进制数。这样数码换算过程不是要求额外的电子学考虑，就是要求额外的计算机操作时间。用一种叫做二－十进制(BCD)的混合表示法，即可大大简化这种数码转换。

BCD代码是一种面向基数10的代码，因此，它与十进制数直接有关，而同时它所使用的又只是二进制的数字0与1。

BCD的构成，是把十进制数的每一位数字换成它所对应的4－位二进制数。例如，6_{10}可表示成0 1 1 0，所以整数0到9可表示成表1.4所示的情况。

表1.4

十进制	二进制	BCD
0	0	0000
1	01	0001
2	10	0010
3	11	0011
4	100	0100
5	101	0101
6	110	0110
7	111	0111
8	1000	1000
9	1001	1001

十进制数的每一个数字直接由它所相当的BCD代码代替后，就是这个十进制数的BCD形式。例如：

0110	0011	0100	0111
6	3	4	7

例。试将BCD数0101 0111(十进制数57)换算成相当的二进制数。

权	(8421)×10	(8421)×1
	0101	0111

为简化乘法运算，可把加权因子10表示成8＋2，于是，BCD数0101 0111就相当于

$$[(0101)\times(8+2)]+(0111)\times 1$$

按二进制的加法和乘法法则有

0111	×1
0101	×2
+0101	×8
0111001	

这正是与十进制数57相当的二进制数。由此可见，用于二进制加法运算的电路可以用来把BCD代码换算成二进制代码。

通常同时用两个BCD数字来给字母、符号以及数进行编码。例如，若用两位的十进制数00，01，02，…，09表示数字0，1，2，3，…9，则其余的两位数10到99即可用来表行A，B，…Y，Z，以及其它一些符号(如：十，一，？)。首先把字母和符号按十进制编码，然后再把十进制代码按BCD编码，则任何数、字母、以及符号对数字系统都可简单地表示为二进制形式。

余3BCD制　在计算机中，尤其在电子制表机中，为了简化十进制反码的形成过程，有时数用余3BCD的形式。给每一数位加3后，再换算成它对应的4－位二进制形式，就成为余3制数。表1.5给出了余3制数字及其十进制反码。可见，只要把所有的1换成0，0换成I，就是十进制反码了。

表1.5

十进制	余3制	余3十进制反码
0	0011	1100
1	0100	1011
2	0101	1010
3	0110	1001
4	0111	1000
5	1000	0111
6	1001	0110
7	1010	0101
8	1011	0101
9	1100	0011

十六进制　十六进制是BCD表示法的扩展，它用额外的符号表示在BCD中没有利用过的六种4－位组合，因此，我们必须熟悉采用16个符号的计数方法，並创造出6个新的数字符号。常用的十六进制为：

0，1，2，3，4，5，6，7，8，9，A，B，C，D，E，F

其中A－F表示4－位组中的1010到1111(即十进制的10到15)。十六进制数见表1.6。

葛莱(Gray)码

二进制代码虽然在进行计算时非常适用，但在对二进制编码的两个相邻位置间的转换进

行取样时，它就可能会暴露出严重问题。用计算机控制轴角位置或线性位置就是一个例子。编码器的两个以二进制编码的相邻位置，其代码的各对应数位之间，可能有好几位都不相同，例如，位置7与8的代码为0111与1000。在编码器的这两部分之间，到底那个位置是准确的转换点，常常会一时无法区分。因此，系统就可能产生诸如1010，0101，1100之类的输出，这些输出与原位置0111及新位置1000之间的区别都很大，数字控制系统就会把这类信息看作位置误差，从而产生校正位置的驱动力。

表1.6

十进制	二进制	Gray	十六进制
0	0000	0	0
1	0001	01	1
2	0010	11	2
3	0011	10	3
4	0100	110	4
5	0101	111	5
6	0110	101	6
7	0111	100	7
8	1000	1100	8
9	1001	1101	9
10	1010	1111	A
11	1011	1110	B
12	1100	1010	C
13	1101	1011	D
14	1110	1001	E
15	1111	1000	F

若两个相邻位置的代码只有一位不同，就可减少这种转换误差。由于在转换期间所有其它位都不变，则转换瞬间的读出就只能是老位置或新位置，而不会象使用二进制代码时所遇到的那样，会是编码器的整个位置中的其他任何位置。

Gray码是一系列称为反射二进制代码中的一种，对位置转换器非常适用。Gray码与二进制代码很相似，往往具有相同的位长与相同的最高有效位。它与二进制代码的主要区别是Gray码中两相邻数只有一位不同。为便于比较，表1.6列出了一些以这两种代码表示的数字。

1.4 数据格式

计算机操作的是二进制信息，它能很方便地由电子器件体现二进制信息，并把它们存放

在存贮器中。二进制信息可代表定点数、浮点数、二进制编码的字符及计算机指令。

定点数

大多数计算机是按二进制补码运算的。在这种情况下，可按下列方法用计算机字来存贮正数和负数。以字的最高有效位作为符号位，0表示正数，1表示负数；字的其余部分表示数的大小，正值所表示的就是该数，负值所表示的是该数的二进制补码。所以，若计算机字有n位，则可用n－1位来表示0到2^{n-1}之间的任一整数值。

下面是一个12－位计算机的例子

正数：	000	000	000	000	0_{10}
	000	000	000	001	1_{10}
	000	000	000	010	2_{10}
	011	111	111	111	$2^{11}-1=2047_{10}$
负数：	111	111	111	111	-1_{10}
	111	111	111	110	-2_{10}
	100	000	000	001	$-(2^{11}-1)=-2047_{10}$
	100	000	000	000	$-2^{11}=-2048_{10}$

可见，12－位字的计算机可直接表示-2048_{10}到$+2047_{10}$之间所有的数。为表示更大的数，可用二个计算机字，或者用浮点式表示。

浮点数

浮点式计数法是：把数分成尾数(数值部分)及阶码(以某数为底数)两部分。例如，在十进制中，15可写成：

尾数		阶码
0.15	×	10^2
1.5	×	10^1
15.0	×	10^0
150.0	×	10^{-1}
1500.0	×	10^{-2}

计算机的浮点式表示法与此例相仿，但因为计算机是以二进制信息工作的，所以尾数与阶码都用二进制表示。由于尾数与阶数都可以为正，也可以为负，所以需以两个位来表示符号位。图1.2为36－位数的定点式及浮点式。小计算机可用几个字来表示浮点数。