

微型电子计算机原理

上 册

江 涛 编

北京工业学院九系901室

1 9 8 1 . 8

前 言

微型电子计算机自从七十年代初问世以来，由于它不但具有一般电子数字计算机所共有的各种优点，而且还具有体积小，可靠性高，适应性强，价格低廉，结构模块化，能满足多方面的需要等特点。因而受到人们的注意。许多非计算机专业、甚至非电子专业的科技人员，为了加快四个现代化的建设也正在研究。然而摆在他们面前的资料多半是外商产品的技术说明，微型机原理方面的内容很少。

为了更好地开展微型机及其系统的研究工作，我们编写这份材料的重点是微型电子计算机原理，芯片也是结合“原理”作典型介绍。

编者认为掌握了微型计算机原理后，各种芯片的功能和系统的研制是不难掌握的。但是由于篇幅和业务水平所限，加之编写时间仓促，必定会有许多缺点和错误，希望批评指正。

微型电子计算机原理 (一)

目 录

第一章 微型计算机的基本知识

§ 1 数制与码制.....	1
一、二进制	
二、二——十进位	
三、ASCII码	
§ 2 数字逻辑与部件.....	14
一、基本逻辑门电路	
二、逻辑代数	
三、触发器	
四、基本功能部件	
§ 3 微处理机组成.....	36
一、概述	
二、微型机组成	

第二章 微型计算机程序设计

§ 1 引言.....	2-1
§ 2 微型计算机的基本指令.....	2-2
一、微型机的逻辑结构	
二、指令格式	
三、寻址方式	
§ 3、程序设计方法及实例.....	2-11
一、程序流程图	

二、机器语言及汇编语言

三、程序设计方法及实例

四、乘、除及十进制运算程序

§ 4 8080的指令系统..... 2-14

第三章微处理器及其内部操作

§ 1 Intel 8080微处理器模块组成..... 3- 1

一、概述

二、8224时钟发生器和驱动器

三、8080—8位CPU微处理器

四、8224系统控制器和总线驱动器

§ 2 微处理器的时序..... 3-12

一、一条指令的执行过程

二、8080 CPU的时序

三、FETCH机器周期的时序

§ 3 微处理器内部操作举例..... 3-23

例一、暂停指令HLT

例二、存寄存器间传送指令 $MUV R_1, R_2$

例三、累加器和寄存器相加指令 $ADD R$

§ 4 8085和Z-80..... 3-29

一、Intel 8085

二、Z-80

第四章 半导体存储器

§ 1 随机存取存储器 RAM 4- 2

一、RAM的组成

二、存贮单元	
三、RAM 的结构	
四、RAM 的易失性问题	
§ 2 只读存贮器 ROM	4-23
一、MOS 固定存贮器	
二、可编程只读存贮器 PROM	
三、可改写只读存贮器 EPROM	
四、只读存贮器的应用举例	
§ 3 半导体存贮与 CPU 之间的联接	4-41
一、半导体存贮器实例	
二、半导体存贮器与 CPU 之间的联接	
三、8205 芯片及其应用	
四、系统联接实例	
§ 4 CPU 与存贮间信息交换	4-51
一、概述	
二、存贮器 $\longleftrightarrow$ 存贮器型指令, $M \longleftrightarrow M$ 型	
三、寄存器 $\longleftrightarrow$ 存贮器型指令, $R \longleftrightarrow M$ 型	
四、存贮器 $\longleftrightarrow$ 累加器型指令, $M \longleftrightarrow A$ 型	
五、8080 指令系统的时序流程表	

第五章 输入输出技术及外围芯片

§ 1 概述	5-18
§ 2 输入输出指令及 I/O 设备编址	5-5
一、输入输出指令	
二、I/O 设备编址	

§ 3	同步传送和异步传送.....	5-11
一、	同步传送	
二、	异步传送	
§ 4	程序中断传送.....	5-21
一、	概述	
二、	程序中断传送过程	
三、	8080 微型计算机系统的程序中断传送	
§ 5	8212 和 8214 芯片及其应用.....	5-39
一、	八位输入输出接口芯片 8212	
二、	优先权中断控制器 8214	
三、	小结	
§ 6	可编程的中断控制器 8259.....	5-60
一、	管脚配置与内部逻辑电路	
二、	组成及其功能	
三、	级联	
四、	工作流程	
五、	初始化及其控制字格式	
六、	工作方式及其控制字格式	
七、	8259 的基本操作	
八、	读操作	
§ 7	DMA 传送.....	5-77
一、	DMA 传送原理	
二、	8080 CPU 的保持 (HOLD) 时序及状态转换	
§ 8	可编程的 DMA 控制器 8257	5-83
一、	管脚配置与内部逻辑电路	
二、	基本功能概述	
三、	8257 的框图	
四、	8257 操作摘要	

第一章 微型计算机的基本知识

§ 1 数制与代码

一、二进制

1. 二进制计数制

计算机的基本技能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。最常用的数制是十进制, 它用 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 十个数码(数字符号), 以一定的规律, 排成数列来表示任意数值。这一规律是“逢十进一”。即十进制, 例如 1980 年, 即是一千九百八十年的缩写 ($1 \times 10^3 + 9 \times 10^2 + 8 \times 10^1 + 0 \times 10^0$)。

但日常生活中, 并不都是采用十进制, 例如一年等于 12 个月, 是十二进制, 一小时等于 60 分, 一分等于 60 秒是六十进制, 旧秤一斤等于 16 两是十六进制。

在计算机中, 数是采用二进制, 它只有两个数字符号“0”和“1”, 而且由低位向高位是“逢二进一”。例如二进制数 101, 它意味着是 $1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$, 用十进制表示, 它就是 5。与十进制数字 0~12 对应的二进制数如表 1-1 所示。(表见下页)

在进位计数制里(如逢二进一), 任意一个数 S_j , 可表示为:

$$S_j = K_n J^n + K_{n-1} J^{n-1} + K_{n-2} J^{n-2} \dots$$

$$\dots K_1 J^1 + K_0 J^0 + K_{-1} J^{-1} \dots K_{-m} J^{-m}$$

可缩写成:

$$S_j = \sum_{i=-m}^n K_i J^i$$

表 1—1

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
11	1011
12	1100

式中进位基数 $J=10$ 时, 即为十进制

$J=2$ 时, 即为二进制

$J=8$ 时, 即为八进制

K_i 可以是 $0, 1, 2, \dots (J-1)$ 中任一数码

m, n 正整数

进位制有如下规律:

(1) 每个进位数都有一个固定的基数 J , 它的每个数位都可取 J 个不同的数字符号中的一个, 而且是“逢 J 进一”的, 即每位计满 J 就向高位进一。

(2) 进位计数制都能写成 $S_j = \sum_{i=n}^{-m} K_i J^i$, 这样的展开式, 它

的每一位数码 K_i 对应有一个固定的值 J^i ， J^i 称为 K_i 的“权”，所以 S_j 的表达式也称为进位计数制的按权展开式。

例如：二进制的数 101.01 自左至右每一位数所对应的权分别为：
 2^2 ， 2^1 ， 2^0 ， 2^{-1} ， 2^{-2} 。

$$\text{即：} S_2 = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2}$$

对应的十进制数为：

$$S_{10} = 4 + 1 + 0.25 = 5.25$$

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

在微处理机中经常使用“位”、“字节”、“字”，等名词。

“位”，二进制数中的一个数位通常称为“位”。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。

数左端的位称为最高有效位，权最大；数右端的位称为最低有效位，权最小。图 1—1 表示一个由 16 位组成的二进制数。

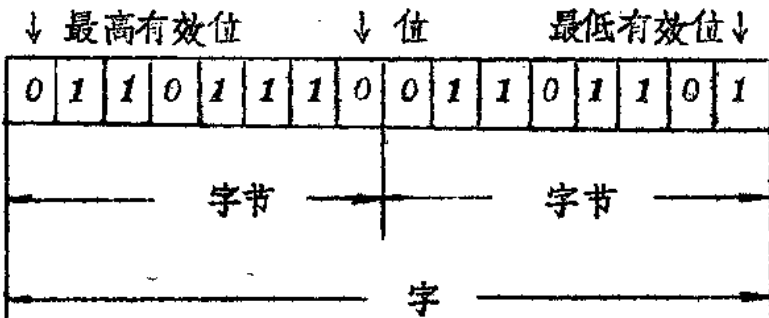


图 1—1 位、字节、字

“字节”微处理器和外部设备之间交换信息多以 8 位为单位，这个 8 位单位称为“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，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 位长度的字。

2. 十进制到二进制转换

一般地说，十进制数具有整数部分与小数部分，每一部分都应换算成相当的二进制数。用小数点把两部分结合起来就是一个完整的二进制数。把十进制数转成二进制数的方法有：

整数 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，否则记作 0，再用 2 除第一次所得的商。如此重复下去，直到商数为 0。例如：

2	53		余数
2	26	→	1
2	13	→	0
2	6	→	1
2	3	→	0
2	1	→	1
	0	→	1

$(53)_{10} = (110101)_2$

小数 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
$0.2500 \times 2 = 0.5$	→ 0
$0.5 \times 2 = 1.0$	→ 1
$0.0 \times 2 = 0.0$	→ 0

于是 $(0.5625)_{10} = (0.10010)_2$ 。

3. 二进制数的算术运算

加法

二进制加法规则

$$0 + 0 = 0 \quad 0 + 1 = 1$$

$$1 + 0 = 1 \quad 1 + 1 = 10$$

↑ 进位

利用此规则可进行二进制数的加法运算，例如

$$\begin{array}{r} 101 = 5_{10} \\ + 010 = 2_{10} \\ \hline 111 = 7_{10} \end{array} \quad \begin{array}{r} 11 \leftarrow \text{进位} \\ 111 = 7_{10} \\ + 101 = 5_{10} \\ \hline 1100 = 12_{10} \end{array}$$

在微型计算机中二进制数的加法是由运算器中的加法器实现的。

减法

二进制减法规则

$$0 - 0 = 0 \quad 1 - 0 = 1$$

$$1 - 1 = 0 \quad 10 - 1 = 1$$

↑ 借位

利用此规则可进行二进制数的减法运算，例如：

借 位

$$\begin{array}{r} 101 = 5_{10} \\ - 010 = 2_{10} \\ \hline 011 = 3_{10} \end{array}$$

由上例可见，减法没有象加法那样容易，每当遇到“0—1”的情况时，就需要向高位“借”。因为高位的“1”是2（即 $1+1=10$ ）。向高位借“1”由计算机实现是很困难的。进行减法运算时，只需要把减数符号变更一下即可作加法运算，如 $x-y=x+(-y)$ 。这样一来，唯一的问题是要找到表示负数（ $-y$ ）的适当方法。

为了了解负数在计算机中是怎样表示的，可回顾一下十进制运算。在十进制运算中，如减9或97之类的数字时，往往先减掉10或100，然后再加上1或3，这样的简化方法自古以来就在心算中经常使用着。

例如：
$$12-9=12+1-10=3$$

$$125-97=125+3-100=28$$

我们称9和1是以10为模互为补数，97和3是以100为模互为补数，利用补数我们可写减法的通式为：

被减数+减数的补数—模数

同样，二进制减法也可用加“减数的补码”来实现。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010110的补码是101010。

即：
$$010110$$

$$+ 101010$$

$$1000000$$

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。

$$010110 \quad \text{数}$$

$$101001 \quad \text{二进制反码}$$

(2) 二进制反码加1即为二进制补码

$$\begin{array}{r}
 101001 \quad \text{二进制反码} \\
 + \quad \quad 1 \quad \text{加1} \\
 \hline
 101010 \quad \text{二进制补码}
 \end{array}$$

例:

$$12 - 9 = 3 \quad 125 - 97 = 28$$

$$1001 \quad 9_{10} \quad 1100001 \quad 97_{10}$$

$$0110 \quad 9 \text{ 的二进制反码} \quad 0011110 \quad 97 \text{ 的二进制反码}$$

$$0111 \quad 9 \text{ 的二进制补码} \quad 0011111 \quad 97 \text{ 的二进制补码}$$

$$1100 \quad 12_{10} \quad 1111101 \quad 125_{10}$$

$$+ 0111 \quad 9_{10} \text{ 的二进制补码} \quad + 0011111 \quad 97_{10} \text{ 的二进制补码}$$

$$\begin{array}{r}
 10011 \quad 3_{10} \quad \quad \quad 10011100 \\
 \hline
 \end{array}$$

↑
模丢去

↑
模丢去

乘法

二进制乘法规则

$$0 \times 0 = 0 \quad 0 \times 1 = 0$$

$$1 \times 0 = 0 \quad 1 \times 1 = 1$$

可见,乘法规则特别简单,除 $1 \times 1 = 1$ 外,其他情况都等于0,无需“九九口诀”。

利用此规则可实现二进制数乘法运算。

例如 $111 \quad 7_{10}$ 被乘数

$$\begin{array}{r}
 \times 101 \quad 5_{10} \quad \text{乘数} \\
 \hline
 111 \\
 000 \\
 + 111 \\
 \hline
 100011 \quad 35_{10}
 \end{array}$$

由上式可见，乘法的运算过程是“相加”和“移位”的交替过程，即根据乘数的每个数位（自低位至高位）是“1”或是“0”，来决定加上被乘数或“0”，每使用下一个乘数位，部分积就向左移一位。

除法

二进制除法运算和十进制除法一样，是“相减”和“移位”过程。

$$18 \div 2 = 9$$

$$10 \div 5 = 2$$

$$14 \div 4 = 3.5$$

$$\begin{array}{r} 1001 \cdot 9_{10} \\ 10 \overline{) 10010} \\ \underline{10} \\ 00 \\ \underline{00} \\ 01 \\ \underline{00} \\ 10 \\ \underline{10} \\ 0 \end{array}$$

$$\begin{array}{r} 10 \\ 101 \overline{) 1010} \\ \underline{101} \\ 000 \\ \underline{000} \\ 0 \end{array}$$

$$\begin{array}{r} 11.1 \\ 100 \overline{) 1110.0} \\ \underline{100} \\ 110 \\ \underline{100} \\ 100 \\ \underline{100} \\ 0 \end{array}$$

从二进制四则运算可以看出，二进制运算规则简单，便于机器执行。特别是引入了补码以后，二进制的四则运算可归纳为“相加”和“移位”两种操作，这样既节省了设备又简化了运算器结构。

可是，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，所用数字符号为0~7。与十进制数字0到10对应的八进制数如表1—2所示。（表见下页）

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，只需把二进制按每三位一组写成等值的八进制数。

例： 110101111001 二进制数

110 101 111 001 3位一组

6 5 7 1 与各三位组相当的八进制数

因此， $(110101111001)_2 = (6571)_8$ 。

必须注意，微型计算机不是以八进制进行运算的，而是以二进制进行运算的，使用八进制只是为了避免读写较大的二进制数。

表 1—2

十进制	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

二、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，数码转换过程不是用电子线路，就是要求额外的计算机操作时间，用二——十进制表示法，即可大大简化这种数转换。

所谓二——十进制(Binary——Coded Decimal),就是用四位二进制编码来表示一位十进制数。在二——十进制中,仍是“逢十进”。但十进制中每个数字符号0, 1, ……9,都是用四位二进制的符号“0”和“1”表示。如表1—3所示。

十进制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:

表 1—3

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

$$0110001101000111 = (6347)_{10}$$

将BCD数换算成相当的二进制数是很简单的。

例. 试将BCD数01010111(十进制数57)换算成相当的二进制数。

$$\begin{array}{cc} \text{权} & (8421) \times 10 & (8421) \times 1 \\ & 0101 & 0111 \end{array}$$

为简化乘法运算，可把加权因子 10 表示成 $8 + 2$ ，于是，BCD 数 01010111 就相当于

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

按二进制的加法和乘法法则有

$$\begin{array}{r}
 0111 \quad \times 1 \\
 0101 \quad \times 2 \\
 + 0101 \quad \times 8 \\
 \hline
 0111001
 \end{array}$$

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。

二——十进制的加、减运算

在微处理机内，二——十进制运算算是用二进制加法电路实现的。由于两个 BCD 代码在此线路中是按 16 进制进位，而 BCD 数则要求 10 进制。所以，当两个 BCD 代码进行相加时，若此结果大于 9 或产生了进位，则需要加 6 修正。所以在微处理机的指令系统中专门设置了一条十进制调整指令 (DAA)。

在微处理机内，二——十进制运算分两步进行 (即二条指令)。

(1) 将两个 BCD 数按二进制进行相加 (用一条 ADD 指令)

例： $x = 257$ $y = 169$

$$\begin{array}{r}
 00100101111 \quad \text{BCD} \quad 257 \\
 + 000101101001 \quad \text{BCD} \quad 169 \\
 \hline
 001111000000
 \end{array}$$

(2) 将运算结果进行加力修正 (用一条 DAA 指令)