DIANZIKEJIDAXUECHUBANSHE
XILIEJIAOCAI

高等学校
电子信息类 规划教材

大专计算机

汇编语言程序设计

罗万钧 沈 新 吴方中

UEST P.H.
电子科技大学出版社
UESTC PUBLISHING HOUSE

内 容 提 要

本书以 MCS－51 单片机指令系统及 Intel 8088/8086 CPU 指令系统为主，讲解汇编语言程序设计的基本理论和方法。该书的特点是内容简洁、取材新颖、实用性强、系统性好。该书有较多的应用程序实例，并配有相应的练习题和上机实习辅导。

全书共分十三章，第一章内容为计算机基础，第二章至第五章内容为 MCS—51 系列单片机的指令系统和程序设计方法，第六章介绍 16 位单片机 MCS—96 的指令系统和应用实例，第七章至第十二章系统介绍 8088/8086 汇编语言程序设计方法，第十三章介绍 80386/80486 程序设计基础。

本书是全国大专计算机应用专业的“九五”规划教材，但也可以用于大学本科（专科）电子类（如控制专业、智能化仪器仪表专业）及中高级程序设计培训班的教材或参考书。

声 明

高等学校电子信息类规划教材

汇编语言程序设计

罗万钧 沈 新 吴方中

出 版：电子科技大学出版社 （成都建设北路二段四号，邮编：610054）

责任编辑：唐雅邻

发 行：电子科技大学出版社

印 刷：四川建筑印刷厂

开 本：787×1092 1/16 印张 29 字数 776 千字

版 次：1998 年 12 月第一版

印 次：1998 年 12 月第一次印刷

书 号：ISBN 7－81065－055－6/TP・33

印 数：1—4000 册

定 价：32.00 元

前　言

本教材系按电子工业部的《1996—2000年全国电子信息类专业教材编审出版规划》，由大专计算机专业教学指导委员会编审、推荐出版。本教材由苏州市广播电视大学罗万钧副教授担任主编，上海第二工业大学江庚和教授担任主审，责任编委为李逊林。

本课程的参考学时数为80学时，主要内容为两大部分，第一部分为以MCS－51为主的单片机汇编语言程序设计，第二部分为以8088/8086为主的80X86汇编语言程序设计。它们既是相对独立的两部分，又是互为补充、紧密联系的两部分。在使用本教材时，应结合实际需要，选择教材中的有关内容进行教学。可以选择一种指令系统为主，另一种指令系统为辅进行教学，也可以只选择其中的一种指令系统进行教学。由于本教材是两种指令系统的汇编语言的有机结合，因此可以节省教学时间，深入浅出地学习，达到事半功倍的效果。在需要进行汇编语言程序设计的场合，MCS－51与8088/8086是应用最多的两种汇编语言，本教材提供了大量的应用程序实例，结合实例对汇编语言源程序的开发工具和开发过程进行了详尽的说明，使本教材具有更强的系统性和实用性。

苏州市轻工职工大学的吴方中讲师编写了教材的第一章及第三章，苏州中银电子公司的沈新高级程序员编写了教材的第二章及第四章，教材的其他章节由苏州市广播电视大学的罗万钧副教授编写，并由罗万钧统编了全稿。由于编者水平有限，书中难免还存在一些缺点和错误，殷切期望广大读者批评指正。

编　者

1998年8月

第一章　计算机基础

为什么有些人觉得学习汇编语言比较困难?因为汇编语言是面向机器的语言,它与机器的指令系统有关,所以在学习本书时,应当对计算机的基本结构与组成有个大致的了解,知道计算机是如何工作的,计算机执行程序特别是执行指令的过程,然后再通过指令系统的学习去掌握汇编语言程序设计的方法。

§1.1　计算机的基本结构与组成

1.1.1　电子计算机的基本组成

计算机由硬件子系统和软件子系统两大部分组成。

1.硬件子系统

所谓硬件子系统(简称硬件系统),系指构成计算机系统的物理实体或称物理装置。它由运算器、控制器、存储器、输入输出接口等部件构成主机(简称计算机)。主机再配以输入输出设备,便构成了计算机的硬件系统,如图 1-1 所示。

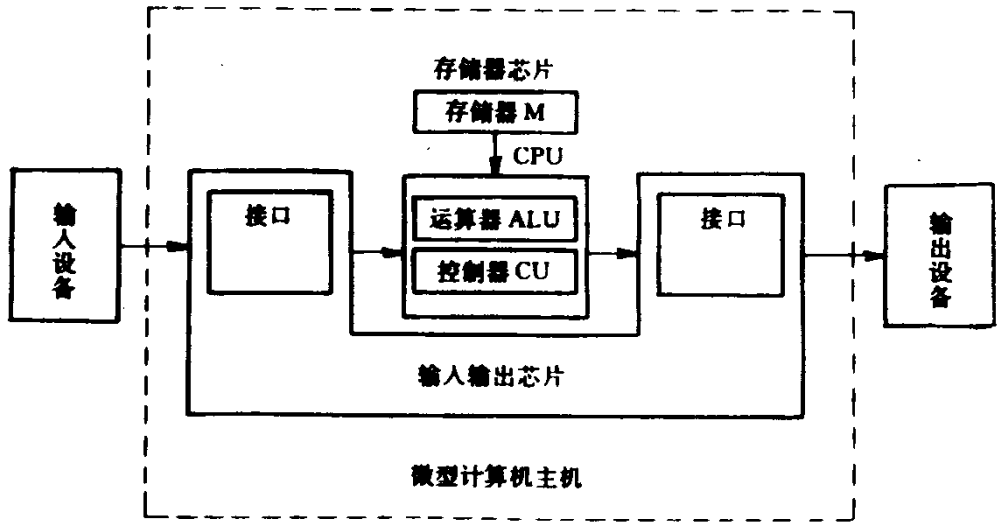


图 1-1　计算机硬件组成

在微型计算机中,将运算器和控制器集成在一片大规模集成电路中,称之为微处理器(或中央处理机 CPU)。将主机集成在一片大规模集成电路的芯片上,又称为单片机。

输入设备:输入设备是人与计算机交往的入口。常用的输入设备有键盘、光电输入机等。

输出设备:输出设备是计算机与人交往的输出窗口,计算机通过它把运算结果或各种信息

以数字、字符、图形等形式表示出来。常用的输出设备有打印机、阴极射线管(CRT)显示器和绘图仪等。

存储器:是存储数据和程序的部件。计算机的存储器分为内存储器和外存储器两大类别。内存储器主要采用半导体存储器,它包括随机存取存储器(RAM)和只读存储器(包括 ROM、可编程 ROM 即 PROM、可用紫外线光擦除的 PROM 即 EPROM、以及可用电改写的 PROM 即 EEPROM 或 E^2PROM)。内存储器是主机的一部分,而外存储器如磁盘、磁带机和磁鼓等,则属于输入输出设备,有时也称为外部设备或外设。

输入输出芯片(I/O 芯片):是计算机与输入输出设备之间的接口。

运算器:是计算机对各种信息进行算术运算和逻辑运算的主要部件。

控制器:是计算机的控制指挥中心,它的功能是识别翻译指令代码,安排操作次序并向计算机各部分发出适当的控制信号,以便执行机器指令,使计算机能自动地、协调一致地工作。

运算器、控制器集成在一块芯片时,称为中央处理器(CPU)或称为微处理器(MPU)。

2. 软件子系统

计算机的软件子系统(简称软件系统),系指计算机系统所使用的各种程序的集合。包括系统软件、程序设计语言及应用软件等。

没有软件系统的计算机系统(称为裸机系统),是没有什么用途的。当然,没有硬件系统,软件系统也就无立足之地了。现代的计算机硬件系统和软件系统之间的分界线并不明显,总的趋势是两者统一融合,在发展上互相促进。

1.1.2 微机硬件系统结构

由于本书介绍的单片机 MCS－51 以及 8086/8088 指令系统的汇编语言都是微型计算机的汇编语言,所以我们主要介绍微机硬件系统的结构。

所谓微机硬件系统的结构系指由各部件构成系统的连接方式。一种典型的微机硬件系统结构如图 1-2 所示。

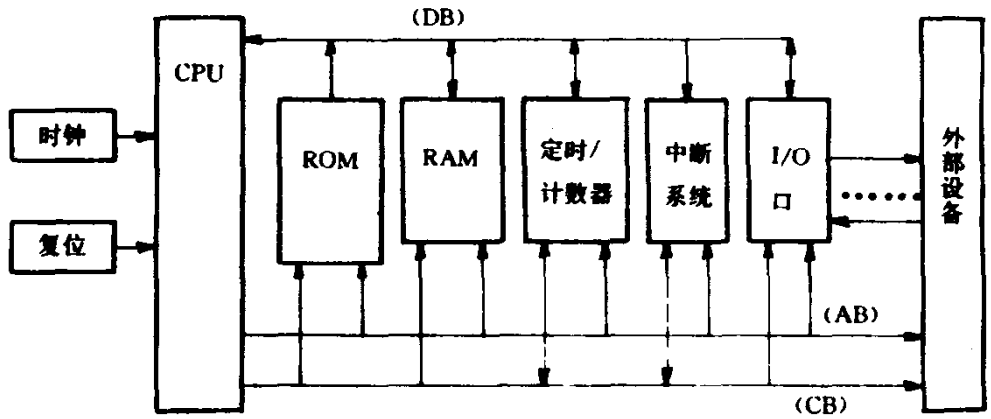


图 1-2　微型计算机系统结构

这种硬件系统结构是单总线(或称系统总线)的系统结构。单总线是一簇用来进行信息传递的公用信号线,它由地址总线(AB)、数据总线(DB)、控制总线(CB)组成。系统中各部件均挂在单总线上。这种系统结构简单、易于扩充,所以目前绝大多数微机硬件系统均采用这种结构。

由于单片机是一个不带外部设备的微型计算机,可看成是一个不带监控程序、没有显示器及键盘的单板机,如图 1-3 所示。

由图 1-3 可见,MCS－51 单片机内部结构也属于单总线结构。

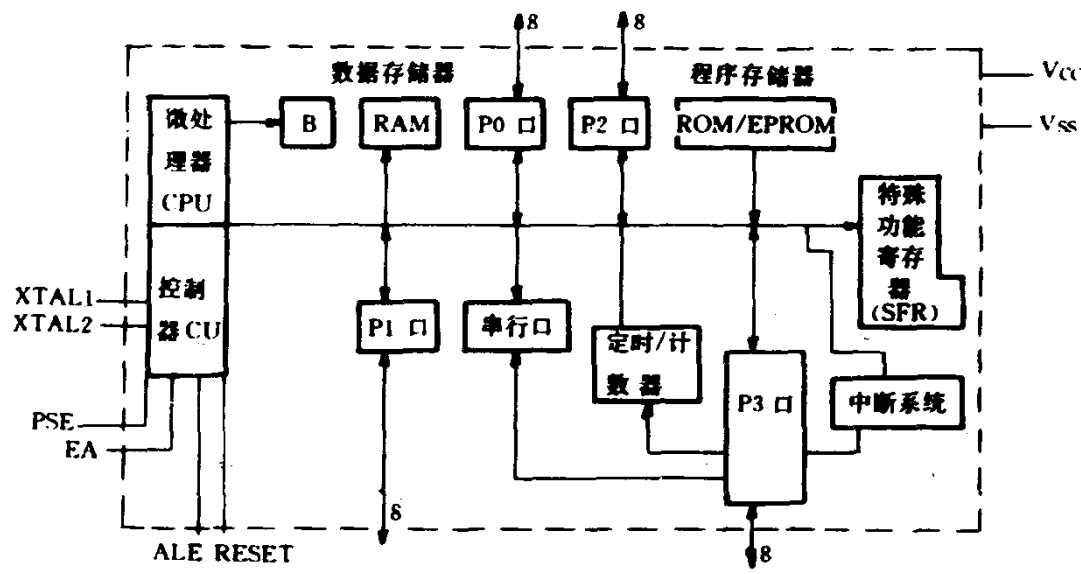


图 1-3　MCS－51 单片机内部结构

1.1.3　存储器组织

1. 基本概念:字节・字・双字・字长

存储器用来存放数据和程序。在计算机内部,数据和程序都用二进制代码的形式来表示。在微机中,一般用 8 位二进制代码作为一个字节(即 1Byte＝8bit)。

用 16 位二进制位或两个字节组成一个字,即 1 Word＝2 Byte＝16 bit。

将两个字组成一个双字,即 1 Doubleword＝2 Word＝32 bit。

如果用字表示一个数,称为数据字;用字表示一条指令,称为指令字。数据字和指令字也可用双倍字长或多倍字长来表示。

微机的字长多为 8 位和 16 位,高档微机的字长可达 32 位。例如 MCS－51 单片机的字长为 8 位,而 MCS－96 单片机,8086/8088 CPU 的字长为 16 位,80286 的字长亦为 16 位,而 80386 及 80486 的字长则为 32 位。

一个存储器可包括很多存储单元,存储单元的内容为数据或指令,存储单元的编号称为地址。

2. 存储器组织

现假设存储器由 256 个字组成,字长为 8 位,其结构如图 1-4 所示。

存储器由 256 个单元组成,每个单元存储 1B,这种规格的存储器,通常称为 256×8 位的读写存储器(即随机存取存储器 RAM)。

随机存取存储器由存储体、地址译码器和控制部件组成。地址译码器接收从地址总线来的地址码,经译码器译码选中相应的存储单元,便从其中读出信息或写入信息。为了能选中地址为 00H～FFH(十六进制表示)的 256 个存储单元中的 1 个,应向地址译码器送入 8 位地址代码。控制部件则用来控制存储器的读和写的操作。

3. 读写工作过程

从存储器读出信息的过程如图 1-5 所示。

这时,存储器的地址译码器接收从地址总线送来的地址代码,当微处理器发来的读控制信号到达控制部件后,便能从被选中的存储单元中读出八位二进制信息,送到数据总线并经数据总线送到微处理器。

向存储器写入的过程如图 1-6 所示。

写入操作是将数据总线上的数据存入被选中的单元中去的。这时，存储器的地址译码器从地址总线接收地址代码，微处理器将数据送到数据总线上，由微处理器来的写控制信号，将数据写入指定的单元中。

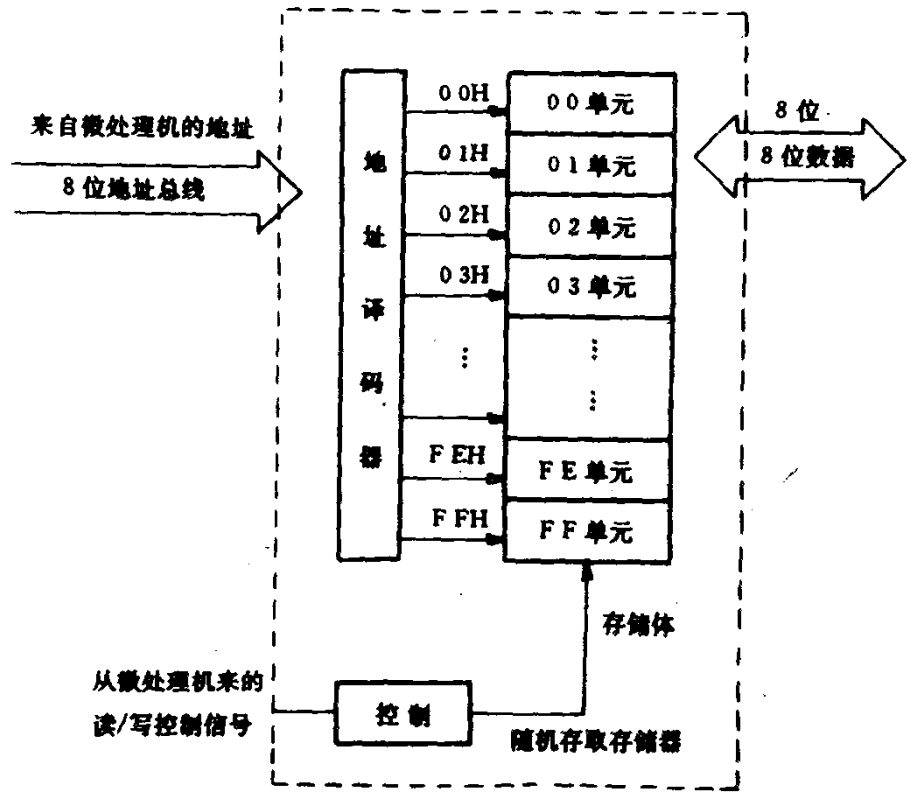


图 1-4 随机存取存储器结构简图

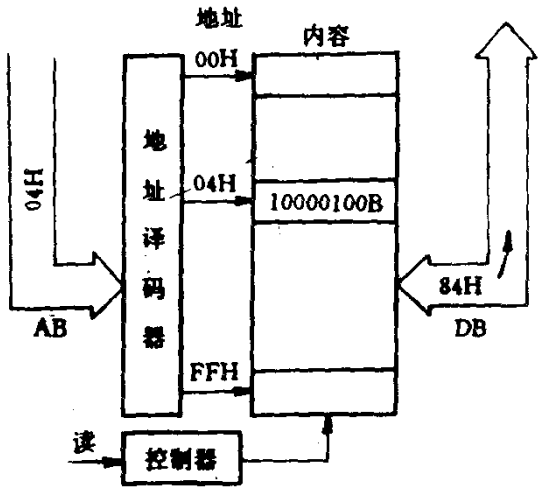


图 1-5 从存储器读出信息的过程

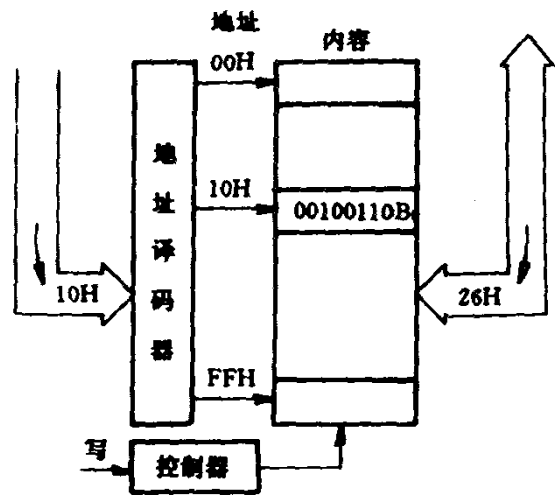


图 1-6 向存储器写入信息的过程

§1.2 计算机中的数制与码制

1.2.1 计算机中采用的数制

数制也称为进位计数制。日常生活中，人们习惯采用十进制数进行计算，而计算机内部的信息则是以二进制代码来表示的。

1. 十进制数的特点

(1) 它有 10 个不同的符号(或称为元素、系数),即 0,1,2,3,4,5,6,7,8,9。

(2) 它是逢 10 进位的,因为 1 位十进制数只能用 0～9 中的 1 个符号表示,对于大于等于 10 的数,就要用多位十进制数来表示。例如,1 个十进制数为 1234.56,其中的“4”代表个位数,而“3”,由于它处于十位的位置上,因而它所代表的数已不是“3”本身,它代表的是 30 或 3×10。同理,其中的“5”,代表的是 5×10^{-1}。我们可以把该数表示成如下形式:

$$1234.56=1\times10^{3}+2\times10^{2}+3\times10^{1}+4\times10^{0}+5\times10^{-1}+6\times10^{-2}$$

一般地说,任意一个十进制数 N,都可以表示为:

$$\begin{aligned}N=\pm[&K_n\times10^{n}+K_{n-1}\times10^{n-1}+\cdots+K_1\times10^{1}+K_0\times10^{0}\\&+K_{-1}\times10^{-1}+K_{-2}\times10^{-2}+\cdots+K_{-m}\times10^{-m}]\\=\pm&\sum_{i=-m}^{n}[K_i\times10^{i}]\end{aligned}$$

其中,K_i 为 0,1,2,3,4,5,6,7,8,9。

2. 数的一般表示法

对不同的数制,N 可表示为:

$$N=\pm\sum_{i=-m}^{n}[K_i\times R^{i}]$$

R 称为进位制的基数。在十进制数中,R=10;在二进制中,R=2;在八进制中,R=8;在十六进制中,R=16 等等。

R^i 称为第 i 位的位权,不同进位制中的位权是不一样的(即与 R 有关)。同一数制中不同的位的权也不相同(即与 i 值有关)。

K_i 称为第 i 位的系数。在 R 为基数时,$K_i=0,1,\cdots,R-1$。

例如:R=2, $K_i=0,1$;

R=8, $K_i=0,1,2,\cdots,7$;

R=10, $K_i=0,1,2,\cdots,9$;

R=16, $K_i=0,1,2,\cdots,15$。

常用的几种进位计数制的表示方法见表 1-1。

表 1-1 常用进位计数制数的表示方法

十进制数	二进制数	八进制数	十六进制数
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10
17	10001	21	11
	…	…	…

3. 各种数制间的转换

(1) 十进制与其他计数制之间的转换

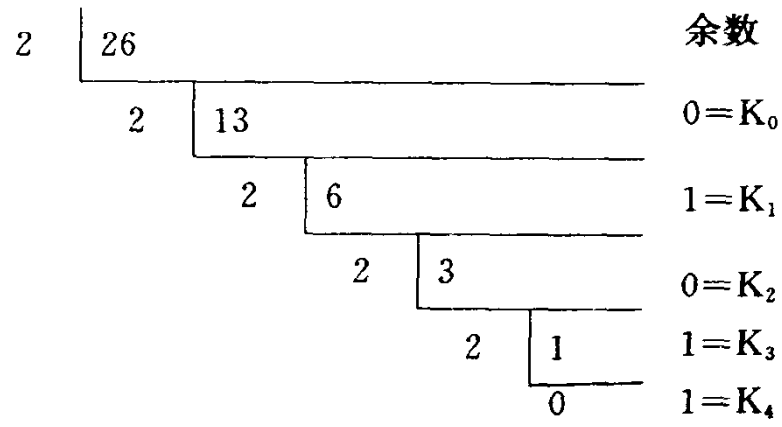
在十进制数与二进制数之间转换中，整数转换与小数转换是不一样的，下面分别叙述之。

① 十进制整数转换成二进制整数

将十进制整数转换成二进制整数采用除 2 取余法。

例 1　将十进制数 26 转换成二进制数。

解



于是 $(26)_{10}=(K_4K_3K_2K_1K_0)=(11010)_2$，其中，$(X)_n$ 表示 X 是 n 进制数。

除 2 取余法是将 $(26)_{10}$这个数不断地除以 2，除尽，即余数为 0，$K_i=0$；除不尽，即余数为 1，$K_i=1$。

现在我们再验算一下：

$$(11010)_2=1\times2^4+1\times2^3+0\times2^2+1\times2^1+0\times2^0$$
$$=16+8+0+2+0=(26)_{10}$$

② 十进制纯小数转换成二进制的纯小数

把十进制纯小数转换成二进制的纯小数采用小数部分乘 2 取整法。

例 2　将十进制纯小数 0.625 转换成二进制数。

解

0.625
×)　　2
——————
1.250　　整数位为 1，即 $K_{-1}=1$

0.25
×)　　2
——————
0.50　　整数位为 0，即 $K_{-2}=0$

0.5
×)　　2
——————
1.0　　整数位为 1，即 $K_{-3}=1$

故$(0.625)_{10}=(0.K_{-1}K_{-2}K_{-3})_2=(0.101)_2$。

将小数反复乘 2，若所得新数的整数部分为 1，则相应的二进制位为 1；若整数部分为 0，则相应二进制位为 0，从小数的高位逐次向低位进行，直到满足精度要求为止。

验算：

$$(0.101)_2=1\times2^{-1}+0\times2^{-2}+1\times2^{-3}=(0.625)_{10}$$

③ 十进制混合小数转换成二进制数

将十进制混合小数转换成二进制数时，可采用整数、小数部分分别转换的办法。它们的逆运算，即二进制数转换成十进制数，正如例1、例2的验算那样，按通用表达式展开求和即可得到。

不难验证，将十进制整数转换成八进制整数，可用“除8取余”法，十进制纯小数可用“乘8取整”法；而八进制数转换成十进制数，按通用表达式展开求和即得。例如：

$$(1503.54)_8=1\times8^3+5\times8^2+0\times8^1+3\times8^0+5\times8^{-1}+4\times8^{-2}$$
$$=512+320+0+3+0.625+0.0625=(835.6875)_{10}$$

同样，十进制整数转换成十六进制整数可用“除16取余”法，十进制小数转换成十六进制小数可用“乘16取整”法。而十六进制数转换成十进制数可用通用表达式展开求和。

(2) 二进制与八进制之间的转换

因为八进制数可用三位二进制数表示，故二进制数转换为八进制数，可以以小数点为界，整数部分自右向左每三位为一组，不够位数则前加零；小数部分自左向右每三位为一组，不够位数则后加零，然后按二进制与八进制的规律直接转换。例如：

$$(1101001.0100111)_2=(001101001.010011100)_2=(151.234)_8$$

八进制数到二进制数的转换是上述过程的逆运算，即将八进制数每一位展开为三位二进制数进行。例如：

$$(653.503)_8=(110101011.101000011)_2$$

(3) 二进制与十六进制之间的转换

因为十六进制数可用四位二进制数表示，所以在十六进制与二进制之间也存在类似八进制与二进制之间的简单而又直接的转换规律，即以小数点为界，整数部分自右向左，每四位为一组，不够的位用前面加零补足；小数部分自左向右，每四位为一组，不够的位用后面加零补足，然后按二进制与十六进制的规律直接转换即可。例如：

$$(16D.44)_{16}=(000101101101.01000100)_2$$
$$(100101111.101000011)_2=(12F.A18)_{16}$$

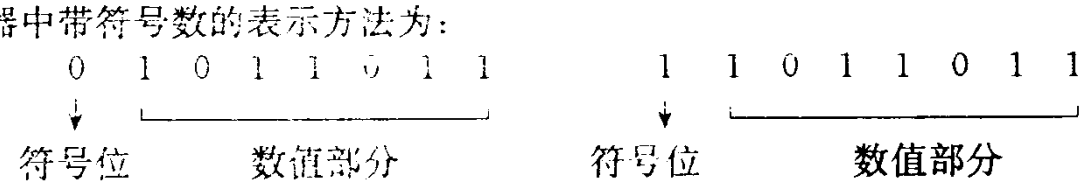
1.2.2 计算机中的码制

1. 机器数和真值

一个数若不考虑它的符号，即为无符号数。具体的数显然有正负，用数学符号“+”或“-”表示的数，如N=+91或N=-91。并把这种用“+”，“-”号表示的数，称为真值。

但是，计算机中所有的数都用0或1进行编码，机器并不认识正负号，只有将正负号也用0或1进行编码，机器才能识别。通常在符号位用“0”表示正，用“1”表示负。这种表示数的方法，称为带符号数的表示方法。

在机器中带符号数的表示方法为：



以上两数，符号位为0的表示$(+91)_{10}$，符号位为1的表示$(-91)_{10}$。一个数在机器中的表

示形式，称为机器数。机器数和它的真值之间存在着一一对应的关系。

对于无符号数在机器中的表示方法，其不同之处是：它没有符号位，全部二进制位均用于表示数的大小。对八位二进制数来讲，表示范围为$(0\sim255)_{10}$。

2. *原码·补码·反码*

原码、补码、反码是带符号数的机器数的表示方法。

(1) 原码表示法

前面介绍的带符号数在机器中的表示方法，实际上就是原码表示法。只要将一个数的真值的符号部分用 0 或 1 表示，即得到该数的原码。例如 X＝＋1001010，因 X＞0，所以其原码$[X]_{原}=01001010$。正数原码的符号位用 0 表示。又如 X＝－1001010，因为 X＜0，所以其原码$[X]_{原}=11001010$。负数原码的符号位用 1 表示。由上面原码的表示形式，可将原码定义为：

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

其中，n 为二进制数的位数，其模为2^n，n＝4 时，即模 mod 2^n 为 16。若 4 位全部用来表示数，其范围为 0～15，即 0～2^n-1。由于符号位占用 1 位，所以其范围将减少，X 是正数时，其范围为＋0～＋7，即$0\leqslant X<2^{4-1}$或$0\leqslant X<2^3$。当 X 为负数时，其范围为 0～－7，即$-2^{4-1}<X\leqslant 0$或$-2^3<X\leqslant 0$。当 X＝8 时，原码表示的数其范围为－127～＋127，即$(11111111)_2\sim(01111111)_2$。

其中的零有正零和负零两种情况，机器中均作零处理：

$$[+0]_{原}=\underset{\text{符号位}}{0}\ \underbrace{0\ 0\ 0\ 0\ 0\ 0\ 0}_{n-1\text{个}0}$$

$$[-0]_{原}=\underset{\text{符号位}}{1}\ \underbrace{0\ 0\ 0\ 0\ 0\ 0\ 0}_{n-1\text{个}0}$$

一般采用正零表示法。

(2) 补码表示法

我们知道两个正数相加，其和为正数；两个负数相加，其和为负数。而当一个正数与一个负数相加时，必须做减法，和的符号位与绝对值较大的那个数的符号相同。为了确定和的符号，必须判断两个数的绝对值的大小，这样将使求和的过程及控制线路变得更加复杂。当采用补码表示法时，正负数的加减运算被转换为单纯的补码相加运算，从而使问题的解决变得简单。

在日常生活中，补码的例子是很多的，时钟的校对就是其中一例。设标准时间是 6 点正，而时针指向 10 点正。要将时针拨到 6 点正，可有两种拨法：

① 10－4＝6，采用倒拨的办法实现，将时针倒拨 4 小时。

② 10＋8＝6，采用顺拨的办法实现，将时针顺拨 8 小时。

这儿倒拨与顺拨的效果是相同的，是相互补充的，即在时钟以 12 为模的条件下，－4 与＋8是互补的。

用数学表达式写成：

$$-4=+8(\text{mod } 12)$$

也称－4 与 8 对模 12 是同余的。

显然，时间为 6 点正与时间为 18 点正，时针的位置是相同的，当然也可以表示为：

6 与 18 对模 12 是同余的。

但在模 12 情况下，18 也只能用 6 点表示。因此，对正数讲，其补码就是它自身；但对负数讲，例如 12+(−4)=8，即模加负数的结果比模小，因而负数可以找到一个比模小的同余数，它就是其补码。至此，我们可以对模 2^n 系统的二进制补码定义如下：

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

其中，n 为二进制数的位数，当 n=8 时，补码表示的数的范围是−128～+127。

由于正数的补码就是它自身，不用另求；而对负数的补码，则需通过减法来求得，这是很不方便的。例如：

$X=(+1000000)_2$，因 $X>0$，所以 $[X]_{补}=(01000000)_2$。

$X=(-0001010)_2$，因 $X<0$，所以 $[X]_{补}=2^8-(0001010)_2=(11110110)_2$。

对负数的补码，其最高位是符号位，取值为 1，而具体的数值位(1110110)恰好为其原码值(0001010)按位求反再加 1，即 $[X]_{补}=11110101+1=11110110$。

所以，负数的补码，其符号位为 1，其值等于其原码除符号位外按位求反，然后再加 1 得到。

此外，不难验证下列三个公式的正确性，它们是：

① $[X-Y]_{补}=[X]_{补}-[Y]_{补}$

② $[[X]_{补}]_{补}=X$

③ $[-Y]_{补}=[[Y]_{补}]_{求补}$

此处的求补操作是指将原码连同符号位一起按位求反，然后再加 1 得到的结果。求补实质上是求一个数的相反数的补码的操作。

例如：求 $X=(-1010)_2$ 的补码可用公式：

$$\begin{aligned}[X]_{补}&=[-1010]_{补}=[[1010]_{补}]_{求补}=[00001010]_{求补}\\&=11110101+1=(11110110)_2\end{aligned}$$

(3) 反码表示法

在补码表示法中已提到负数的补码可以通过对原码除符号位不变外，按位求反+1 得到。如果求反不加 1，就得到另一种机器数的表示法，叫做反码表示法。可从补码的定义推出反码的定义：

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

可见，正数的反码就是它自身；负数的反码，其符号位为 1，数值部分按位取反。此时，零有两种情况：

$[+0]_{反}=0\ 0\ 0\ 0\ 0\ 0\ 0\ 0$

符号位　　数值

$[-0]_{反}=1\ 1\ 1\ 1\ 1\ 1\ 1\ 1$

符号位　　数值

对于一个由反码表示的带符号数，当符号位为 0(即正数)时，后面的几位为数值部分；但当符号位为 1(即负数)时，后面几位表示的不是此负数的数值，只有把它们按位取反后，才表

示它的二进制值。

3. 数的编码方法

(1) 二-十进制编码(BCD 码)

BCD 码又称为十进制数的二进制编码。1 位 BCD 码用 4 位二进制编码来表示。编码方法很多，计算机中最常用的编码是 8421BCD 码，即最高位的权为 8，依次逐步降低，其权值分别为 4，2，1，BCD 码的编码如表 1-2 所示。

表 1-2　8421 BCD 码编码表

十进制数	8421 BCD 码
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	0001　0000
11	0001　0001
12	0001　0010
…	…
19	0001　1001
…	…

MCS－51 及 8086/8088 等多数 CPU 的指令系统中，均有一条十进制调整指令，使计算机能对用 BCD 码表示的十进制数进行加减法的运算，甚至实现十进制数的乘法和除法运算。

(2) 字符编码

将用汇编语言或高级语言编写的程序(称为源程序或文本程序)送入计算机时，数据和信息是用字符的编码表示的。显然，包括字母在内的各种字符，如数字符号 0～9，字母 A～Z，a～z，符号＋，－，＊，/，·等必须用特定的规则，以二进制编码的形式表示，才能被计算机所识别。编码的方式很多，普遍采用的 ASCII 码(美国标准信息交换码)用 7 位二进制编码，可表示 128 个字符编码，其数码 0～9 是用$(30)_{16}$～$(39)_{16}$表示的，字母 A～Z 是用$(41)_{16}$～$(5A)_{16}$表示的。详见表 1-3。

(3) 代码书写形式

当计算机的信息用 8 位、16 位或 32 位二进制编码组成时，不仅书写困难，也不便于阅读，而且容易出错。通常用八进制、十六进制数来书写，为了区别各种数制表示的数码，在二进制数后面加一个字母 B(Binary)，如 01011101B，与我们在前面用$(01011101)_2$所表示的意义相同；在八进制数的后面加上一个字母 O(Octal)，例如 036O，与我们前用$(036)_8$所表示的意义相同；在十六进制数的后面加上一个字母 H(Hexadecimal)，例如 7EH，与以前用$(7E)_{16}$所表示的意义相同；在十进制数的后面加一个字母 D(Decimal)，字母 D 可省略不写，但在任何情况下，八进制数和十六进制数后一定要加上字母 O 及 H。

表 1-3　ASCII 码表

低位 / 高位	0 0000	1 0001	2 0010	3 0011	4 0100	5 0101	6 0110	7 0111	8 1000	9 1001	A 1010	B 1011	C 1100	D 1101	E 1110	F 1111
0 0000	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1 0001	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2 0010	SP	!	"	#	$	%	&	'	(	)	*	+	,	−	.	/
3 0011	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4 0100	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5 0101	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	↑	←
6 0110	'	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7 0111	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

§1.3　机器语言·汇编语言·高级语言

1.3.1　指令和程序

目前的计算机基本上都是按冯·诺依曼的方式工作的，基本方式是程序的存储和自动执行。在计算机执行某一任务之前，首先要为这一任务编写一个程序。通俗地讲，程序就是计算机执行任务的步骤或操作过程，它又是由一系列能完成某一操作的机器指令组成的，因此，程序是指令的有序集合。在编好程序以后，还要把它事先存储到内存储器的程序区（或称为指令代码区），程序一旦启动，计算机便能自动地从头至尾执行完毕，并输出计算结果。由于在执行中不需要人进行干预，所以充分发挥了计算机高速运算的优点。

1.3.2　机器语言(Machine Language)

机器语言就是二进制编码的机器指令。计算机在执行程序的过程中，首先将指令从内存储器的指令代码区取至 CPU 的指令寄存器(IR)，然后，对指令代码进行译码，从而针对不同指令产生不同的操作控制信号，以完成指令所指定的功能（或操作）。显然，只有这种指令代码符合其计算机事先约定的代码时，计算机才能理解和执行它。用这种指令代码书写的程序叫做机器语言程序。例如 7405H，在 MCS－51 单片机的指令中，其功能是将一个立即数 05H 送至累加器 A。

用机器语言编写的程序，通常又称为目标程序(Object Program)，它能被机器理解和执行，但不便于人们阅读、理解和交流；编写程序时，容易出错，给程序的编写、阅读和修改带来了

很多的困难。

1.3.3 汇编语言(Assembly Language)

为了便于理解和记忆指令，人们采用了能帮助记忆的英文缩写作符号(称为指令助记符)来代替机器指令的操作码。

用指令助记符及符号地址所书写的指令叫做汇编格式指令。

由汇编指令、汇编伪指令、汇编宏指令及相应的语法规则组成的总体称为汇编语言。

把用汇编语言格式编写的程序叫做汇编语言程序。

例如 MCS－51 的机器指令代码和汇编助记符对应关系如下：

机器指令代码	汇编语言指令助记符
7405H	MOV A, #05H
2406H	ADD A, #06H
E530H	MOV 30H, A
80FEH	HERE: SJMP HERE

其中，第 1 条指令是传送指令，操作码助记符为 MOV，是 MOVE 的缩写，其功能是将常数 05H 送入累加器 A。

第 2 条为加法指令，操作码助记符为 ADD(就是英语的做加法的意思)。这条指令的功能是将常数 06H 与累加器 A 中的内容相加，且将相加的结果(0BH)保留在 A 中。

第 3 条为传送指令，其功能是将 A 中的内容送到 8051 单片机内部数据存储器的 30H 单元地址中去。

第 4 条指令为无条件跳转指令，它相对于当前的指令地址某一偏移量，转入 HERE 所指的指令中去执行。本例中为一死循环操作，相当于机器处于原地踏步方式运行。

从该例可以看出，用汇编语言编写程序要比用机器语言书写程序方便得多，它对程序的执行有一定程度的描述，便于人们的阅读和记忆。由于汇编语言程序与机器语言程序是一一对应的，所以汇编语言程序也是针对特定机器的程序，这在一定程序上有碍于程序在不同类型的机器之间进行交流。

虽然汇编语言程序为人们对程序的编写、阅读和交流带来了方便，然而机器只认识(理解)用机器语言编写的目标程序。这种翻译工作可由人们通过查表的方法来进行，这种工作称为人工汇编(或称代真)。人工汇编是一种单调、重复、繁重而效率极低、极易出错的工作，它可以由计算机自己来完成，这种汇编过程叫做机器汇编。

汇编过程示意图如图 1-7 所示。

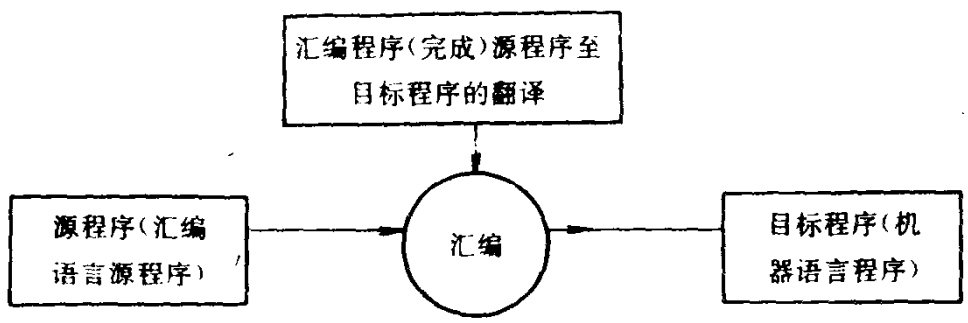


图 1-7　汇编过程示意图

其中的汇编程序对不同的汇编语言是不同的，对 MCS－51 单片机有 MCS－51 Assembler 汇编程序；对 8086/8088 汇编语言，有 MASM－86 Assembler。对同一种机器也有不同的版本或区别很大的汇编程序。

1.3.4 高级语言(High Level Language)

汇编语言仍然是面向机器的语言，它要求编程人员对特定的机器的指令系统有详细深入的了解。高级语言则是面向使用者(用户)的语言，它更接近于人们日常习惯的数学语言，它不要求编程人员对机器指令系统有深入的了解，一个用高级语言编写的程序可以在各种不同型号的机器上运行(只要配有该种语言的翻译或编译程序)。高级语言的程序由语句组成，每条语句的功能相当于若干条机器指令，因此语句的功能很强，用语句编写程序要容易得多。

目前通用的高级语言有 BASIC，FORTRAN，COBOL，PASCAL，C 语言等。

虽然高级语言使用方便，但对于实时控制的应用场合，却存在着明显的不足之处：与汇编语言程序相比，要多占用 0.5～1 倍的存储单元，执行程序花费的时间要长 0.5～2 倍。汇编语言可直接用输入输出指令通过低级接口与外部设备打交道，而高级语言却不那么方便。

使用汇编语言编写程序，能充分发挥机器硬件的作用，能高效地使用机器。汇编语言是编写计算机通过软件的基础，在计算机控制、计算机检测系统以及为个人计算机编写外设驱动程序和通讯程序中有着广泛的应用，因而掌握汇编语言是十分重要的。

§1.4 计算机执行程序的过程

1.4.1 微处理器(MPU 或 CPU)的结构

在 §1.1 中，我们介绍了计算机的组成、计算机的结构及存储器的组织，为了弄清计算机执行程序的过程，还应当了解微处理器的结构。

如图 1-8 所示，微处理器由运算部件、累加器、数据寄存器、程序计数器、地址寄存器、指令寄存器、指令译码器、微操作控制线路和状态标志寄存器等组成。现将各部件的功能简述如下。

1. 运算部件(ALU)

运算部件又称为算术逻辑部件，用来进行算术及逻辑运算。

2. 累加器(A 或 ACC)

在进行算术逻辑运算时，它在运行前，存放一个操作数，运算后，用来存放运算结果。

3. 数据寄存器(DR)

数据寄存器 DR 用来暂存数据或指令。从存储器读出的若为指令，则暂存于 DR 中的指令通过内部数据总线送到指令寄存器(IR)中；若读出的是数据，则通过内部数据总线送到有关寄存器或运算部件中。在向存储器写入数据时，数据是经数据寄存器 DR，再经数据总线 DB 写入存储器的。

4. 程序计数器(PC)

程序计数器中存放着指令地址。根据 PC 中的指令地址，从存储器中取出将要执行的指令。

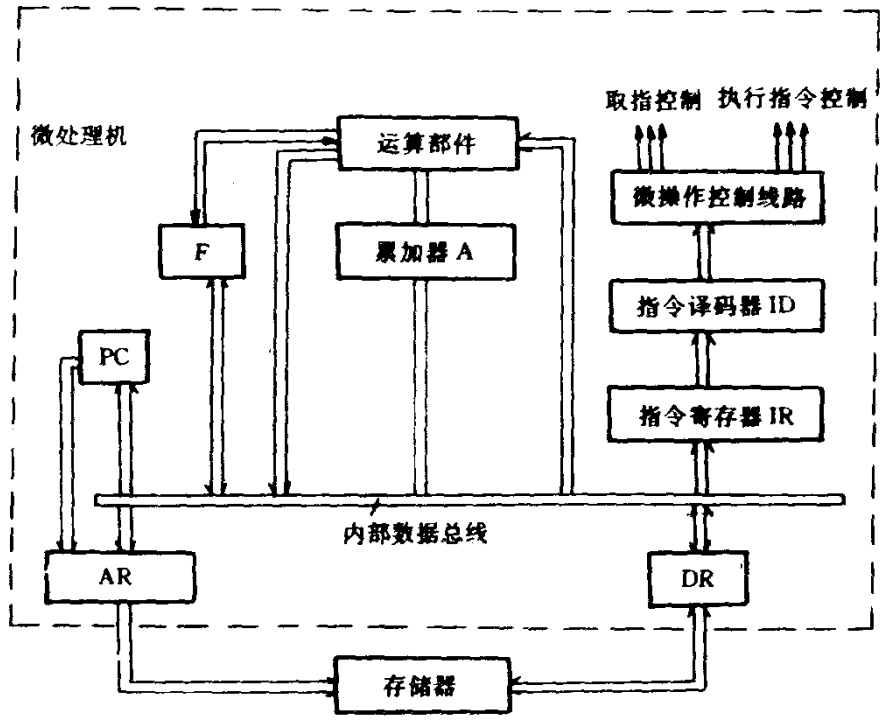


图 1-8　微处理器内部结构

5. 地址寄存器(AR)

地址寄存器 AR 用来存放指令地址或操作数地址。

在取指令时，将 PC 中存放的指令送 AR，根据此地址从存储器中取出指令代码；在向存储器存入数据时，也要先将写入地址送到 AR，根据给定地址向存储器写入数据。

6. 指令寄存器(IR)

指令寄存器 IR 用来存放从存储器取出的将要执行的指令。

7. 指令译码器(ID)

指令译码器 ID 用来对指令寄存器 IR 中的指令进行译码，以便确定该指令应执行什么操作。

8. 微操作控制线路

微操作控制线路用来产生取指令及执行指令的各种微操作控制信号，因为每条指令所执行的操作是不同的，所以每条指令都对应着控制信号的一种组合，以确定相应的操作序列。

9. 状态标志寄存器(F)

状态标志寄存器 F 用来寄存执行指令时产生的状态标志信号，如是否产生算术溢出，是否有进位发生、符号位是 0 还是 1 等。根据执行指令时所产生的状态标志信号，可以决定程序的流向。

1.4.2　计算机执行指令的过程

计算机执行指令的过程或执行指令序列的过程，就是执行程序的过程，也就是不断地取指令、执行指令的过程。

微机的工作过程可分为两个阶段。开始时，微处理器进入取指令阶段。在取指阶段，从存储器取出指令，送指令寄存器，经译码器译码后，进入第二阶段。在执行指令阶段，微处理器执行指令所规定的操作。

不同指令的取指阶段是由一系列相同操作组成的，也就是说，对于不同的指令，取指令的