21世纪高等院校计算机教材系列

微型计算机原理及应用技术学习指导与习题解答

朱金钧　张雪梅　朱　薇　编著

机械工业出版社
CHINA MACHINE PRESS

21世纪高等院校计算机教材系列

微型计算机原理及应用技术学习指导与习题解答

朱金钧　张雪梅　朱　薇　编著

机械工业出版社

本书是机械工业出版社出版的《微型计算机原理及应用技术（第2版）》的配套教材。全书包含了计算机基础知识、8086微处理器及其系统、Pentium系列微处理器的体系结构、指令系统、汇编语言程序设计、微机存储器系统、输入/输出和中断、接口技术和微机总线技术等内容。本书每章的内容均分为两部分，第一部分析教材内容，并详细演示典型例题的解法；第二部分为教材习题解答。

本书可以作为高等院校电子、计算机及相关专业本专科生、研究生的教材和复习指导书，也可以作为教师的教学参考书。本书还可以作为从事微型计算机应用开发的工程技术人员的参考用书。

图书在版编目（CIP）数据

微型计算机原理及应用技术学习指导与习题解答/朱金钧，张雪梅，朱薇编著. —北京：机械工业出版社，2007.5
（21世纪高等院校计算机教材系列）
ISBN 978-7-111-21429-8

Ⅰ.微… Ⅱ.①朱… ②张… ③朱… Ⅲ.微型计算机－高等学校－教学参考资料 Ⅳ.TP36

中国版本图书馆CIP数据核字（2007）第063826号

机械工业出版社（北京市百万庄大街22号 邮政编码 100037）
策　　划：胡毓坚
责任编辑：赵　慧
责任印制：洪汉军

北京汇林印务有限公司印刷

2007年6月第1版·第1次印刷
184mm×260mm·14印张·345千字
0001—5000册
标准书号：ISBN 978-7-111-21429-8
定价：21.00元

凡购本图书，如有缺页、倒页、脱页，由本社发行部调换
销售服务热线电话：（010）68326294
购书热线电话：（010）88379639　88379641　88379643
编辑热线电话：（010）88379739
封面无防伪标均为盗版

出 版 说 明

信息技术是当今世界发展最快、渗透性最强、应用最广的关键技术，是推动经济增长和知识传播的重要引擎。在我国，随着国家信息化发展战略的贯彻实施，信息化建设已进入了全方位、多层次推进应用的新阶段。现在，掌握计算机技术已成为21世纪人才应具备的基本素质之一。

为了进一步推动计算机技术的发展，满足计算机学科教育的需求，机械工业出版社聘请了全国多所高等院校的一线教师，进行了充分的调研和讨论，针对计算机相关课程的特点，总结教学中的实践经验，组织出版了这套“21世纪高等院校计算机教材系列”。

本套教材具有以下特点：

(1) 反映计算机技术领域的新发展和新应用。

(2) 注重立体化教材的建设，多数教材配有电子教案、习题与上机指导或多媒体光盘等。

(3) 针对多数学生的学习特点，采用通俗易懂的方法讲解知识，逻辑性强、层次分明、叙述准确而精炼、图文并茂，使学生可以快速掌握，学以致用。

(4) 符合高等院校各专业人才的培养目标及课程体系的设置，注重培养学生的应用能力，强调知识、能力与素质的综合训练。

(5) 适合各类高等院校、高等职业学校及相关院校的教学，也可作为各类培训班和自学用书。

机械工业出版社

前　言

本书是机械工业出版社出版的《微型计算机原理及应用技术(第2版)》(朱金钧,麻新旗等编著)的配套教材,也可以作为一本介绍微机原理,帮助读者实践和学习微机应用技术的参考书。

本书对照原教材也分为9章,采用了理论和例题相结合的新颖方式讲解,力图由浅入深地帮助读者掌握教材中的理论知识和技术。本书作者多年从事"微型计算机原理及应用技术"课程的教学和科研实践,本书中的很多内容就是作者的经验和工作成果的结晶,例如在本书第8章的内容中,有许多实例就是科研成果的总结,读者可以直接选取、模仿和应用。

本书可以作为高等院校电子、计算机及相关专业的本专科生、研究生的教材和复习指导书及教师的教学参考书,也可以作为从事微型计算机应用开发的工程技术人员的参考用书。

全书由朱金钧统稿,每章第一部分由朱金钧和朱薇编写,第二部分由张雪梅编写。

本书在编写过程中得到麻新旗、周万珍、张会莉、薛增涛、杨奎河、秦敏等老师的帮助,在此深表谢意。

由于作者水平有限,书中难免有不妥和错误之处,敬请读者批评指正。

编　者

目　录

第 1 章　计算机基础知识

电子数字计算机是 20 世纪最重大的科技成就之一。《微型计算机原理及应用技术》一书是为研究、应用计算机技术打基础的著作，作为计算机工作者必须扎实地学好书中的内容。

1.1　分析教材内容

本章主要介绍了计算机的发展、数制和码制的运算基础、计算机系统组成和计算机基本工作原理，其中需要重点掌握的是数制和码制的运算基础、计算机系统组成和计算机基本工作原理。

1.1.1　分析重点、难点问题

本章要求重点掌握数制和码制的运算基础、计算机系统组成和计算机基本工作原理三方面的内容。

1. 数制和码制

首先强调数和码的区别，数是我们工作、学习和生活中经常用到的数据，如 88、－100 等，正数的符号‘＋’可以省略，负数的符号‘－’要写在数前。码是用数字编排的、表示某种物理意义的编码。对于数码，不仅是数的符号数字化，还要事先定义码长(字长)。这是初学者经常出错的地方。

(1) 进位计数制

按进位的方法进行计数称为进位计数制。在计算机中，常用二进制和十六进制。一个 R 进制数具有以下主要特点：

- 具有 R 个不同的数字符号：0、1、2、……、R－1。
- 逢 R 进一。

R 进制数 S 可用多项式(称为按权展开式)表示为：

$$S=a_{n-1}\times R^{n-1}+a_{n-2}\times R^{n-2}+\cdots+a_1\times R^1+a_0\times R^0+a_{-1}\times R^{-1}+\cdots+a_{-m}\times R^{-m}$$

其中，n 为小数点前的位数，m 为小数点后的位数，a_i 是 R 进制的一个数字符号。R 称为基数。

1) 二进制数。二进制数的特点是：

- 具有二个不同的数字符号，即 0 和 1。
- 逢二进一。

一个二进制数可以用它的按权展开式表示，计算结果为十进制数。例如：

$$(10110.101)_2=1\times2^4+0\times2^3+1\times2^2+1\times2^1+0\times2^0+1\times2^{-1}+0\times2^{-2}+1\times2^{-3}$$
$$=(22.625)_{10}$$

2) 十六进制数。十六进制数的特点是：

- 具有十六个不同的数字符号，即 0～9 和 A～F。

● 逢十六进一。

一个十六进制数可以用它的按权展开式表示，计算结果为十进制数。例如：

$$(1AF.4)_{16}=1\times16^2+10\times16^1+15\times16^0+4\times16^{-1}=(431.25)_{10}$$

3）各种数制之间的转换：

● 二进制、十六进制转换成十进制。采用按权展开式计算求和的方法，如前例。

● 十进制转换成二进制、十六进制。整数部分采用除基取余法，小数部分采用乘基取整法。例如将十进制数 22.625 转换为二进制数。

整数部分：

2 | 22

2 | 11 …余0（低位）

2 | 5 …余1

2 | 2 …余1

2 | 1 …余0

0 …余1（高位）

小数部分：

0.625

× 2

1.25 …取整数1（高位）

× 2

0.5 …取整数0

× 2

1.0 …取整数1（低位）

所以：$(22)_{10}=(10110)_2$　　$(0.625)_{10}=(0.101)_2$

故：$(22.625)_{10}=(10110.101)_2$

此处容易出错的地方是：整数部分转换结果高、低位写错，如：10110 写成 01101。

又例如：将十进制数 430.25 转换为十六进制数。

整数部分：

16 | 430

16 | 26 …余14→E（低位）

16 | 1 …余10→A

0 …余1（高位）

小数部分：

0.25

× 16

4.0 …取整数4

故：$(430.25)_{10}=(1AE.4)_{16}$

注意：①整数部分转换，每次只求整数商，将余数作为转换结果的一位，重复对整数商除基数，一直除到商为 0 为止。②小数部分转换，每次把乘积的整数取走作为转换结果的一位，对剩下的小数继续进行乘法运算。对某些数可以乘到积的小数为 0（如上述两例），这种转换结果是精确的；对某些数（如 0.3）永远不能乘到积的小数为 0，这时要根据精度要求，取适当的位数，这种转换结果是不精确的。

● 二进制与十六进制间的相互转换。因为 $2^4=16$，所以每一位十六进制数对应四位二进制数，因此，只要用四位二进制数代替对应的一位十六进制即可完成十六进制到二进制的转换。

例如：十六进制数　1　A　E　.　4

↓　↓　↓　↓

0001　1010　1110　.　0100

即　1AE.4H=110101110.01B

将二进制数转换为十六进制数，只要以小数点为分界，分别向左和向右每四位二进制位分为一组（若最高位或最低位不够四位则补 0），对应转换为十六进制数即可。

例如:二进制数　　110101110.01

↓

0001　1010　1110　.0100

↓　↓　↓　↓

十六进制数　1　A　E　.4

即　$(110101110.01)_2=(1AE.4)_{16}$

(2) 二进制编码

① BCD 码(Binary Coded Decimal)是表示十进制数的二进制编码。每位十进制数需要用 4 位二进制编码,编码的表示方法很多,较常用的是 8421BCD 码,表 1-1 列出了 8421BCD 码和十进制数的对应关系。

表 1-1　BCD 编码表

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

② 字母与字符的编码。为了在计算机中表示字母和字符,字母和字符也必须按照特定的规则,用二进制编码才能在机器中表示。编码可以有各种方式,目前微机中最普遍采用的是 ASCII 码。

ASCII 码采用 7 位二进制编码,故可表示 $2^7=128$ 个字符,其中包括数码(0～9),以及英文字母等可打印的字符。数码 0～9 用 0110000～0111001 来表示。另外,在计算机中,汉字编码采用国标码(GB18030-2000),它采用单、双、四字节混合编码,每个字节的最高位为 1,并以此来区分汉字和 ASCII 码。

(3) 数码的表示

为了在计算机中应用带符号数,计算机中的符号也要数字化,即要数码来表示,通常规定数据的最高位为符号位,即若是字长为 8 位,则 D_7 位是符号位,D_6～D_0 为数字位,并规定符号位用 0 表示正,用 1 表示负。

如:$X=(00001100)_2$　则 X 的值为 +12　而:$Y=(10001100)_2$　则 Y 的值为 −12。

在计算机中,数码有三种表示法——原码、反码和补码。用原码、反码、补码表示的数称为机器数,机器数对应的数学值称为真值。

① 原码。如上所述,正数的符号位用 0 表示,负数的符号位用 1 表示,数值位保持不变。这种表示法称为原码。原码的定义为:

若 $X\geqslant+0$ 则 $[X]_{原}=X$

若 $X\leqslant-0$ 则 $[X]_{原}=2^{n-1}-X$　其中 n 为原码的位数。

例如,设用 8 位表示原码,则 $[+124]_{原}=01111100$,$[-124]_{原}=11111100$

原码表示法简单易懂,而且与真值转换方便。但若是两个异号数相加(或两个同号数相

减)就需要做减法。为了简化运算器设计(把减法运算转换为加法运算),引入了反码和补码。

② 反码。反码的定义为:若 $X \geqslant +0$ 则 $[X]_{反}=X$

若 $X \leqslant -0$ 则 $[X]_{反}=2^n-1+X$ 其中 n 为反码的位数。

显然,正数的反码表示与原码相同,最高位为符号位,其余位为数值位。

例如: $X=+4$,则 $[X]_{反}=[X]_{原}=00000100$

而负数的反码应当表示为该数的原码除符号位外其余位按位取反。

例如: $X=-31$,则 $[X]_{原}=10011111$,$[X]_{反}=11100000$

负数的反码表示与原码有很大的区别:最高位相同,仍是"1",但数据位的值完全相反,这一点要特别注意。8 位二进制反码的表示有以下特点:

- "0"有两种表示方法:$[+0]_{反}=00000000$,$[-0]_{反}=10000000$
- 8 位二进制反码真值范围为 $-127 \sim +127$;16 位反码真值范围为 $-32767 \sim +32767$。
- 当一个带符号数用反码表示时,最高位为符号位。若符号位为 0,说明该数是正数,后面各位即是其真值;若符号位为 1,说明该数为负数,其真值为后面各位按位取反。例如:已知 $[X]_{反}=00000101$,则 $X=+5$;$[Y]_{反}=11111110$,则 $Y=-1$。

③ 补码。补码的定义为:若 $X \geqslant +0$ 则 $[X]_{补}=X$

若 $X \leqslant -0$ 则 $[X]_{补}=2^n+X$ 其中 n 为补码的位数。

当 $X \geqslant 0$ 时,$[X]_{补}=X$,即正数的补码为原正数不变,显然正数的原码、反码、补码相同。当 $X \leqslant 0$,$[X]_{补}=2^n-1+X+1=[X]_{反}+1$,即负数的补码等于负数的反码加 1,也就是等于负数原码除符号位外按位取反、最低位加 1。下面举例说明补码的求法与应用。

$[+3]_{补}=[+3]_{原}=[+3]_{反}=00000011$ $[-3]_{补}=[-3]_{反}+1=11111100+1=11111101$

$[+0]_{补}=[+0]_{原}=[+0]_{反}=00000000$ $[-0]_{补}=[-0]_{反}+1=11111111+1=00000000$

特别注意,$[+0]_{补}=[-0]_{补}==00000000$,即 0 的补码为 0,且只有一种表示方法。

8 位带符号数的补码表示有以下特点:

- $[+0]_{补}=[-0]_{补}=00000000$。
- 8 位二进制补码真值范围为 $-128 \sim +127$,16 位补码真值范围为 $-32768 \sim +32767$。
- 一个用补码表示的二进制数,最高位为符号位,当符号位为"0"即正数时,其余位即为此数的二进制值;但当符号位为"1"即负数时,其余位不是此数的二进制值,其值为后面各位按位取反、最低位加 1。

例如: $[X]_{补}=00001111$,则 $X=+15$; $[X]_{补}=11110001$,则 $X=-15$

$[X]_{补}=10000000$,则 $X=-128$

- 当采用补码表示时,可以把减法运算转换为加法运算,即 $[X \pm Y]_{补}=[X]_{补}+[\pm Y]_{补}$。

例如: $X=64-9=64+(-9)$,则 $[X]_{补}=[64]_{补}+[-9]_{补}$,

$[+64]_{补}=01000000$,$[+9]_{原}=00001001$,$[-9]_{补}=11110111$,

于是,

```
  01000000          01000000
+ 11110111        - 00001001
----------        ----------
 [1]00110111        00110111
  └→自然丢失
```

显然，加法和减法运算的结果完全相同。原因是对 8 位二进制数，当运算结果超过 $2^8=256$ 时，在机器上又体现为从零开始，即 8 位字长的最大数不能超过 256，也就是模是 256。正是由于此，减法运算才能转换为加法运算。

④ 补码运算的溢出及其判断方法。所谓溢出是指运算结果超出了规定长度数据的表数范围，在此特指带符号数的补码运算溢出。例如，对于 8 位字长的二进制补码数，其表数范围为 $-128\sim+127$。如果运算结果超出了此范围，就会产生溢出。

例如，用 8 位二进制补码分别计算 $60+100$、$(-60)+(-100)$、$60+(-100)$。

已知 $[60]_{补}=00111100$，$[-60]_{补}=11000100$，$[100]_{补}=01100100$，$[-100]_{补}=10011100$。

$$\begin{array}{lll}
[60]_{补}= \ 00111100 & [-60]_{补}= \ 11000100 & [60]_{补}= \ 00111100 \\
+[100]_{补}= \ 01100100 & +[-100]_{补}= \ 10011100 & +[-100]_{补}= \ 10011100 \\
\hline
\quad 10100000 & \quad 101100000 & \quad 11011000 \\
\downarrow & \text{自然丢失}\leftarrow\downarrow & \downarrow \\
\text{符号} & \text{符号} & \text{符号}
\end{array}$$

即 $[60+100]_{补}=10100000$，两个正数相加，结果却为负数，显然是错误的；

$[(-60)+(-100)]_{补}=01100000$，两个负数相加，结果却为正数，显然也是错误的；

$[60+(-100)]_{补}=11011000=-40$。

前两个运算结果之所以不正确，是因为其相加结果分别为 +160 和 −160，均超出了表数范围，使结果的数值部分占据了符号位，产生了溢出错误。但一个正数与一个负数相加，一定不会产生溢出错误。

在计算机中，经常根据加法运算中在最高位与次高位的两个进位来判断：$OV=C_7 XOR C_6$（XOR 是逻辑异或运算）$OV=0$ 不溢出；$OV=1$ 溢出。（其中：C_7 为最高位进位，C_6 为次高位的进位。）

此处应注意进位与溢出的区别。进位是指运算结果的最高位向更高位的进位，而溢出是用最高位进位（即 C_y）与次高位进位的逻辑异或结果来判断的。通过上例可以看出，有进位不一定就有溢出，无进位也不一定就无溢出。同理，有溢出不一定就有进位，无溢出也不一定就无进位，进位和溢出是两个不同性质的概念，不能混淆。

2. 计算机系统组成

计算机系统由计算机硬件和软件两部分组成。

(1) 计算机硬件

计算机硬件由运算器、控制器、存储器、输入设备、输出设备五大部分组成。

控制器主要由指令指针寄存器（IP）、指令寄存器（IR）、指令译码器（ID）和控制信号发生器组成。指令指针寄存器中存放将要执行的指令的地址，按此地址从内存取出指令，并送到指令寄存器中暂存，然后由指令译码器进行分析译码，根据译码结果由控制信号发生器发出一系列控制信号到运算器、存储器等各功能部件来完成该指令的执行。一个简单计算机的硬件电路结构框图如图 1-1 所示。

(2) 计算机软件

计算机软件包括系统软件和应用软件两大部分。

系统软件是由计算机设计者或软件生产厂家提供的，是供用户使用和管理计算机的软件。

包括：

1）各种语言的汇编或解释、编译程序；

2）操作系统、调试程序、故障诊断程序；

3）程序库。

应用软件是用户用各种语言编制的解决各种问题的软件。

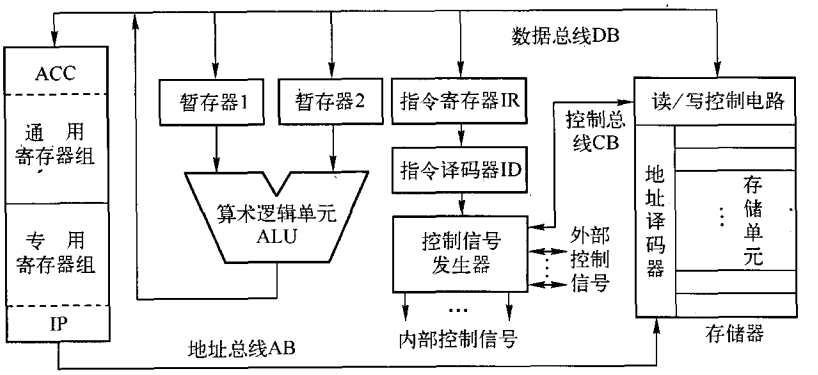

图 1-1　简单计算机的硬件电路结构示意图

3. 计算机基本工作原理

计算机内部有两类信息在流动，即数据流和控制流。程序和数据通过输入设备送入存储器中。控制器从存储器中逐条取出命令并加以分析，按照命令的功能规定向各个部件发出一系列的控制信号来执行这条命令，如存储器把参加运算的数据送给运算器，运算器按规定的运算操作进行计算，并把结果送回到存储器中保存。最后把处理的结果通过输出设备送给外界。这就是计算机工作的基本过程。至于具体控制流中的控制信号，将在下面章节中介绍。下面以图 1-1的计算机硬件电路为模型用一个简单的例子来说明计算机执行程序的主要过程。

假设我们已通过输入设备把程序和数据存入到存储器中，如图 1-2 所示。本例中，假设程序段共有 3 条指令，每条指令占用 1 个存储单元。第一条指令 MOV ACC,[265]的功能是将内存地址为 265 的单元内容读出并送入 CPU 内部寄存器 ACC 中；第二条指令 ADD ACC,[266]的功能是将 ACC 的内容与 266 单元的内容相加，结果存到 ACC 中；第三条指令 MOV [267],ACC 的功能是将 ACC 的内容存到 267 单元中。

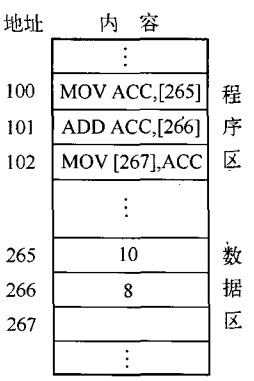

图 1-2　示例程序和数据

当执行该程序时，用户首先通过键盘，把程序第一条指令的地址(100)送到 CPU 的指令指针寄存器(IP)中并启动执行。执行过程如下：

1）取指阶段：把 IP 中存放的指令地址送到存储器(M)：IP→M；之后，IP 自动加 1，指向下一条指令：IP + 1→IP。存储器对地址译码，找到 100 单元，并读出其内容，送入指令寄存器(IR)：M→IR。

2）分析执行阶段：IR 把指令送入指令译码器(ID)，ID 对指令译码后，由控制信号发生器产生如下一系列控制信号来执行这条指令：

① 操作数地址(265)送到存储器：addr→M。

② 读存储器，把指定单元的内容送到规定的累加器 ACC 中：M→ACC。

至此，第一条指令执行完毕。由上可以看出，一条指令的执行分成取指阶段和分析执行阶段。需要强调的是，不管是取指阶段还是分析执行阶段，每一项操作如地址传送、存储器读写、数据传送等，都是在控制器发出的控制信号指挥下按照严格的时间顺序一步一步地完成的。由于在取指阶段取出指令后 IP 已自动加 1，指向了下一条指令，因此，以后的执行过程就是周而复始地重复上述的取指、分析和执行指令，把每条指令顺序执行完。简要描述如下：

- 取指阶段：IP→M，IP + 1→IP，M→IR。
- 分析执行阶段：IR→ID，译码、执行 addr→M；M→暂存器 1；ACC→暂存器 2；ALU 执行“ADD”运算；ALU 结果→ACC。
- 取指阶段：IP→M，IP + 1→IP，M→IR。
- 分析执行阶段：IR→ID，译码、执行 addr→M（送地址）；ACC→M（送数据）；存储器写操作。

至此，上述程序段执行完毕。

读完上段内容后，我们头脑里已有了计算机执行程序的大概过程，实际上这就是计算机自动加工信息的原理。其中会提出许多如何实现的问题，那要到后面的章节解决。

4. 微型计算机系统的组成及特点

微型计算机系统与其他计算机系统一样，也是由硬件和软件两部分组成的。

（1）微型机硬件结构及特点

在计算机中，把运算器和控制器合在一起称为中央处理机，简称 CPU（Center Processing Unit）。在微型机中，CPU 通常是一个大规模集成电路芯片，也称为微处理器（Microprocessor，简称 μP）。CPU 与内存合起来称为主机。主机、总线、接口电路与外围设备组成了微型机的硬件。

总线是微型机中连接各功能部件并传送信息的一组信号线，分为 3 类，即地址总线 AB（Address Bus）、数据总线 DB（Data Bus）和控制总线 CB（Control Bus）。总线结构是微型机的独特结构，如图 1-3 所示。

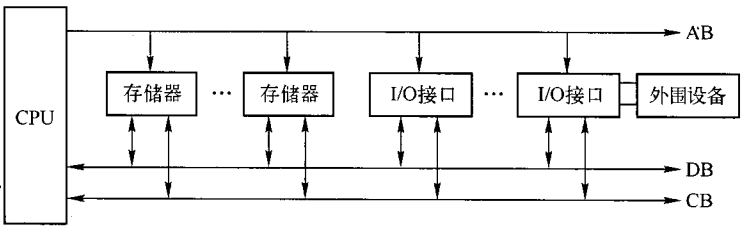

图 1-3　微型机的总线结构特点

有了总线结构以后，系统中各功能部件之间的相互关系变为各个部件面向总线的单一关系。一个部件或设备只要符合总线标准，就可以连接到采用这种总线标准的系统中，使系统功能能很简便地得到扩展。

数据总线用来传输数据。从结构上看，数据总线是双向的，即数据既可以从 CPU 送到其他部件，也可以从其他部件传送到 CPU。数据总线的位数（也称宽度）是微型机的一个重要指标，它和微处理器的位数相对应，反映计算机数据运算和传送的能力。和其他类型计算机一

样，微型机中数据的含义也是广义的。数据总线上传送的数据不一定就是数学意义上的数值，还可以是其他编码、指令代码、状态量，或者是一个控制量。

地址总线专门用来传送地址信息。因地址总是从 CPU 送出去的，所以和数据总线不同，地址总线是单向的。地址总线的位数决定了 CPU 可以直接寻址的内存范围。比如，8 位微型机的地址总线一般是 16 位，因此，最大内存容量为 $2^{16}=64$ KB；16 位微型机的地址总线为 20 位，因此，最大内存容量为 $2^{20}=1$ MB。

控制总线用来传输控制信号，其中包括 CPU 送往存储器和输入/输出接口电路的控制信号，如读信号、写信号、中断响应信号、DMA 响应信号等；还包括其他部件送到 CPU 的信号，如准备好信号、中断请求信号、总线请求信号等。

(2) 微型计算机系统的主要技术指标

1) 常用的名词术语包括：

- 位(bit)。位是计算机所能表示的最小的数据单位。它只能有两种状态“0”和“1”，即二进制位。
- 字(Word)。计算机中作为一个整体参与运算、处理和传送的一串二进制数，是计算机中信息的基本单位。
- 字长(Word Length)。计算机中每个字所包含的二进制位数称为字长。字长通常等于数据总线的位数和通用寄存器的位数。8 位微处理器的字长为 8 位，32 位微处理器的字长为 32 位。
- 字节(Byte)。8 位二进制数称为一个字节。字节的长度是固定的，但不同计算机的字长是不同的。8 位机的字长就等于 1 个字节，而 16 位机的字长等于 2 个字节。
- 指令。指挥计算机进行基本操作的命令。它是计算机能够识别的一组二进制编码。通常一条指令由操作码和操作数两部分组成，前者指出应该进行什么操作，后者指出参与操作的数据的来源。
- 指令系统。计算机所能执行的全部指令的集合称为计算机的指令系统。
- 程序。完成某一任务的指令(或语句)的有序集合称为程序。也就是根据执行过程，将对应的操作指令按一定顺序排列在一起。

2) 主要技术指标。一台计算机性能优劣，是由它的系统结构、指令系统、硬件组成、外围设备以及软件配备齐全与否等因素决定的。其主要性能指标为：

- 字长。字长越长表示的数据精度就越高。计算机的字长是计算机内部一次可以处理的二进制代码的位数。
- 内存储器容量。容量越大处理信息的能量越强。运算速度。速度越高，执行作业的时间越短。
- 外围设备配备。

1.1.2 典型例题解析

第一章的习题主要反映在两个方面，其一是计算机概念的解答，其二数制、码制及其运算。关于计算机概念的问题，教材的习题已很多，请参考本章的教材习题解答。下面以第二方面内容做典型例题解析。

1. 不同计数制数的转换问题

【例 1-1】将 88 和 −100 转换为二进制数。

答：88＝1011000B　−100＝−1100100B　此处有的读者容易和码的概念混淆，用符号位“1”代表“−”号。得到 −100＝11100100B 的错误解。

【例 1-2】将 1ABH 转换为十进制数。

解：$1ABH=1\times16^2+10\times16+11\times1=427$

【例 1-3】将 −100 转换为六进制数。

解：

6 ⌊100

6 ⌊16　… 余4（低位）

6 ⌊2　… 余4

0　… 余2（高位）

所以：$(-100)_{10}=(-244)_6$

2. 数制和码制的转换问题

【例 1-4】将二进制数 11101110 转换为 BCD 码。

解：首先将二进制数 11101110 转换为十进制数 11101110B＝238。然后每位十进制数用四位二进制数连续表示出。$(001000111000)_{BCD}$

【例 1-5】已知字长 8 bit，$[X]_{补}=10000000$，求 X 的真值。

解：因为补码零惟一性，10000000 不是负零，而是一种特殊表示形式，即符号位和最高有效位重合。所以看符号位时最高位为 1，真值为负，连最高位一起求反加 1，得到真值的值 128。X 的真值为 −128。

【例 1-6】运算下列无符号数，并判断是否发生溢出。

（1）
$$\begin{array}{r} 10101010 \\ +\ 01100010 \\ \hline \end{array}$$

（2）
$$\begin{array}{r} 10100010 \\ +\ 01010100 \\ \hline \end{array}$$

解：无符号数运算用最高位的进位判断溢出与否。（1）运算结果最高位有进位，故有溢出。实际上 170＋98＝268 超出了 8 bit 字长能表示的最大的无符号数 255。（2）运算结果最高位无进位，没有溢出，结果为 247。

【例 1-7】若字长为 8 bit，用补码加法计算下列带符号数式，并判断是否发生溢出。

（1）50 − 100　　（2）−50 − 100

解：（1）50 − 100 可写成 50＋(−100)，以补码加法计算，写为$[50]_{补}+[-100]_{补}$

$$\begin{array}{rl} 00110010 & [50]_{补} \\ +\ 10011100 & [-100]_{补} \\ \hline 11001110 & [50]_{补}+[-100]_{补} \end{array}$$

最高位与次高位均无进位，故无溢出。读结果时，应把补码结果变为原码，即为10110010B，十进制数为−50

（2）−50 − 100 可写成(−50)＋(−100)，以补码加法计算，写为$[-50]_{补}+[-100]_{补}$，于是，

$$\begin{array}{rl} 11001110 & [-50]_{补} \\ +\ 10011100 & [-100]_{补} \\ \hline 01101010 & [-50]_{补}+[-100]_{补} \end{array}$$

最高位有进位而次高位无进位，故溢出。很清楚，两个负数相加不可能得正数。

1.2 教材习题解答

1．计算机中为什么都采用二进制数而不采用十进制数？

【解】计算机的基本功能是对数的运算和处理。计算机中，通过数字化编码技术，对所表示的数值、文字、符号及控制信息等进行数字编码，这种数字化表示方法不仅要适合于人的自然习惯，同时要满足机器中所用器件、线路的工作状态以及数据可靠传输与易于校验纠错等方面的要求。一个具有两种不同的稳定状态且能相互转换的器件，就可以用来表示一位二进制数，由于二进制的器件易于制造且工作可靠，并且二进制数的运算规则也最简单，因此目前计算机中均采用二进制数来表示各种信息及进行信息处理。

2．写出下列用原码或补码表示的机器数的真值：

(1) 01101101　(2) 10001101　(3) 01011001　(4) 11001110

【解】

(1) $[X]_{原}=01101101=+109$　　$[X]_{补}=01101101=+109$

(2) $[X]_{原}=10001101=-13$　　$[X]_{补}=10001101=-115$

(3) $[X]_{原}=01011001=+89$　　$[X]_{补}=01011001=+89$

(4) $[X]_{原}=11001110=-78$　　$[X]_{补}=11001110=-50$

3．填空：

(1) $(1234)_{10}=(\quad)_2=(\quad)_{16}$

(2) $(34.6875)_{10}=(\quad)_2=(\quad)_{16}$

(3) $(271.33)_{10}=(\quad)_2=(\quad)_{16}$

(4) $(101011001001)_2=(\quad)_{10}=(\quad)_{16}$

(5) $(1AB.E)_{16}=(\quad)_{10}=(\quad)_2$

(6) $(10101010.0111)_2=(\quad)_{10}=(\quad)_{16}$

【解】

(1) $(1234)_{10}=(10011010010)_2=(4D2)_{16}$

(2) $(34.6875)_{10}=(100010.1011)_2=(22.B)_{16}$

(3) $(271.33)_{10}=(100001111.010101)_2=(10F.54)_{16}$

(4) $(101011001001)_2=(2761)_{10}=(AC9)_{16}$

(5) $(1AB.E)_{16}=(427.875)_{10}=(110101011.111)_2$

(6) $(10101010.0111)_2=(170.4375)_{10}=(AA.7)_{16}$

4．已知 X=36，Y=－136，Z=－1250，请写出 X、Y、Z 的 16 位原码、反码和补码。

【解】

$[X]_{原}=0000\ 0000\ 0010\ 0100$

$[Y]_{原}=1000\ 0000\ 1000\ 1000$

$[Z]_{原}=1000\ 0100\ 1110\ 0010$

$[X]_{反}=0000\ 0000\ 0010\ 0100$

$[Y]_{反}=1111\ 1111\ 0111\ 0111$