第1章 微型计算机基本结构及信息表示（二）1.3~1.5

1.3 数制及数制之间的转换方法

1.4 计算机中信息的基本表示方式

1.5 文字在计算机中的存储方式—字符编码

1.3 数制及数制之间的转换方法

1.3.1 数制

1. 数制的概念

通俗地说，数制（Number system）就是计数的法则，它用一组固定的数码和一套统一的规则来表示数字的大小。例如，人们日常生活中使用的数制是十进制（Decimal system），它使用0、1、2、3、4、5、6、7、8、9这十个数码，并定义以下规则：自然界中所有的数字都用这十个数码表达，满十进一，且规定同一个数码在从左到右不同的位置上所表示的数值大小不同。人类普遍使用十进制，可能与远古时代用十指记数这个习惯有关。

2. 基数计数法

基数计数法（Radix notation），也称按位计数法或进位计数法，该计数方法是以基数和位权来表示的计数方法，任何一个数制都包含基数和位权这两个基本要素。

数制中的基数（Radix number）表示基本符号的个数。例如，十进制的基数就是10，二进制的基数就是2，十六进制的基数为16。

数制中的位权（Position weight）表示某一位上的1所表示数值的大小（所处位置重要性的度量），一般简称权（weight）。例如，十进制数693.85，该数中最左边的6代表600，而600=6*10^2，这里的10^2就是6所处位置的“权”

有了基数与权概念，任意一个数x可表示成按权展开，例如：$26.38 = \sum_{i=1}^{-2} \alpha_i R^i = 2 * 10^1 + 6 * 10^0 + 3 * 10^{-1} + 8 * 10^{-2}$

3. 计算机中常用的数制

表 1-1 计算机中常用的二进制、十进制、十六进制

数制	数码	数码个数	基数	进位规则	借位规则	书写前缀	书写后缀
二进制	0、1	2	2	逢二进一	借一当二	0b	B
十进制	0、1、2、3、4、5、6、7、8、9	10	10	逢十进一	借一当十	（无）	D
十六进制	0、1、2、3、4、5、6、7、8、9、A、B、C、D、E、F	16	16	逢十六进一	借一当十六	0x	H

说明：十六进制数码中的 A、B、C、D、E、F 分别对应十进制的 10、11、12、13、14、15

【练习1-3】将二进制数101.1101及十六进制数8BD.A6F按权形式展开。
【练习1-4】写出八进制数码个数、基数、进位规则、借位规则。

1.3.2 数制之间的转换方法

1．其他进制数与十进制数之间的转换

（1）其他进制数转为十进制数：“按权展开求和”

【例1-1】 将二进制数0b1011.101转为十进制数。

解：$0b1011.101=1*2^3+0*2^2+1*2^1+1*2^0+1*2^{-1}+0*2^{-1}+1*2^{-2}=11.625$。

【练习1-5】把十六进制数0x6A8转为十进制数。

1.3.2 数制之间的转换方法

1. 其他进制数与十进制数之间的转换

（2）十进制数转为其他进制数：一般采用“乘除法”

【例】 将十进制数89.86转为二进制制数。

解：89.86=0b1011001.1101111，计算方法如下：

整数部分

被除数	除数	商	余数	
89	÷2=	44	1	整数部分最低位
44	÷2=	22	0	
22	÷2=	11	0	
11	÷2=	5	1	
5	÷2=	2	1	
2	÷2=	1	0	
1	÷2=	0	1	整数部分最高位

小数部分

被乘数	乘数	乘积	整数	
0.86	*2=	1.72	1	小数部分最高位
0.72	*2=	1.44	1	
0.44	*2=	0.88	0	
0.88	*2=	1.76	1	
0.76	*2=	1.52	1	
0.52	*2=	1.04	1	
0.04	*2=	0.08	0	小数部分最低位
0.08	*2=	0.16	0	
0.16	*2=	0.32	0	.....（取决于期望的精度）

【练习1-6】把十进制数56. 23转为二进制数和十六进制数。

2. 二进制数与十六进制数之间的转换

表 1-2 十六进制数与二进制数的对应关系

十六进制数	二进制数	十六进制数	二进制数
0	0000	8	1000
1	0001	9	1001
2	0010	A	1010
3	0011	B	1011
4	0100	C	1100
5	0101	D	1101
6	0110	E	1110
7	0111	F	1111

二进制数转换为十六进制数的基本方法：以小数点为界，整数部分向左，每4 位二进制数为一组，不足4位的，高位补0，然后用1位十六进制数码表示对应的二进制数即可；小数部分向右，每4 位二进制数为一组，不足4 位的，低位补0，然后用1位十六进制数码表示对应的二进制数即可。

十六进制数转换为二进制数的基本方法：把每位十六进制数码用4位二进制数表示，书写时根据具体情况去除不影响结果的整数部分的前置0与小数部分的后置0，使之符合平时书写习惯即可。

2. 二进制数与十六进制数之间的转换

【例1】 将二进制数0b1011001.1101111转为十六进制数，将十六进制数0x6A8.DC转为二进制数。

解：0b1011001.1101111B=0b 0101 1001 .1101 1110 = 0x59.DE。

0x6A8.DC=0b 0110 1010 1000.1101 1100=0b11010101000.110111。

【例2】 将二进制数0b111001转为十六进制数，将十六进制数0x6AF转为二进制数。

解：0b111001=0b11 1001= 0x39。

0x6AF=0b 0110 1010 1111=0b110 1010 1111。

【练习1-7】 将二进制数0b101001.110101转为十六进制数，将十六进制数0x27B5.3D转为二进制数。

3．利用工具查看进制转换结果

1.4 计算机中信息的基本表示方式

1.4.1 计算机中信息表示的相关基本概念（重点）

1．位、字节、机器字长

硬件上，计算机中的所有数据均表现为二进制 。“位”（bit）是单个二进制数码的简称，是可以拥有两种状态的最小二进制值，分别用“0”和“1”表示。计算机中信息单位是8位二进制数，即“字节”（byte），是计算机中信息基本度量单位。

机器字长是指计算机在运算过程中一次能吞吐的二进制数据位数，表示了CPU内部数据通路的宽度，它等于数据总线条数，与CPU内数据寄存器的宽度是一致的。

计算机中使用二进制，可做如下理解：第一，二进制只取两个数码0和1，物理上可以用两个不同的稳定状态的元器件来表示；第二，它的运算规则简单，基数为2，进位规则是“逢二进一”，借位规则是“借一当二”；第三，计算机的理论基础是逻辑和代数，当二进制与只使用“真”和“假”两个值与逻辑代数建立联系后，就为计算机的逻辑设计提供了便利的工具，如集成电路中门电路的设计。

2．机器数与真值

数的符号书写用“±”号表达，称为真值。在规定了用0表示正数、1表示负数之后，以二进制形式形式存储于计算机内部，称为机器数。机器数有不同的编码表示。

例如，整数通常采用补码表示方式，下面将阐述其编码方法及缘由。

1.4.2 整数在计算机中的补码表示方法（难点）

1．原码、反码与补码的基本含义与求法

表 1-3　原码、反码与补码基本含义与求法举例

内容	真值（十进制）	机器数		
		原码	反码	补码
符号位	用 ± 书写	最高位为符号位，0 表示正数，1 表示负数		
数值部分		二进制绝对值	正数的数值部分就是该数的二进制表示，负数的数值部分是该数的二进制逐位取反	正数的数值部分是该数的二进制表示，负数的数值部分是该数的二进制逐位取反后得到的值+1
0	0	**0** 000 0000		
正数	+1~+127	**0** 0000001 ~ **0** 1111111		
负数	−1	**1** 000 0001	**1** 111 1110	**1** 111 1111
	−2	**1** 000 0010	**1** 111 1101	**1** 111 1110
	…	…	…	…
	−127	**1** 111 1111	**1** 000 0000	**1** 000 0001
特殊	−128	无法表示	无法表示	**1** 000 0000（多出一个最小值）
	−0	**1** 000 000	**1** 111 1111	**0** 000 0000（没有出现−0 问题）

下面分析一下为什么设计补码这种表示方式？

第一，原码与反码对特殊数据的表示有二义性。如出现了-0问题，见表1-3。0就是0，哪还有+0、-0如何理解？如何参与运算？

第二，原码与反码表示解决不了符号位变成了数字之后参与运算问题。以8位为例，在原码表示中，计算：1+（-1）=$(0000\ 0001)_{原}$+ $(1000\ 0001)_{原}$=$(1000\ 0010)_{原}$ = -2，这是不对的。在反码表示中，计算：（-1）+（-2）=$(1111\ 1110)_{反}$+$(1111\ 1101)_{反}$=$(1111\ 1011)_{反}$ =$(1000\ 0100)_{原}$ = -4，这也是不对的。

第三，补码表示可以解决以上问题。首先，没有+0、-0问题了，见表1-3，而且可以用原码中-0(1000 0000)，在补码中表示为-128，形成了-128，-127，…，-1，0，1，…，127，共256个8位有符号数的完整表达。其次，以8位为例，在补码表示中，计算：1+（-1）=0000 0001+ 1111 1111=0000 0000=0，这是对的。又用补码表示计算：（-1）+（-2）=$(1111\ 1111)_{补}$+$(1111\ 1110)_{补}$=$(1111\ 1101)_{补}$=$(1000\ 0011)_{原}$= -3，这也是对的。

第四，使用补码表示，可以将真值的减法运算变为机器中加法运算，使得CPU内部不需要设计减法器。例如，1-2=1+(-2)= $(0000\ 0001)_{补}$+$(1111\ 1110)_{补}$=$(1111\ 1111)_{补}$=$(1000\ 0001)_{原}$= -1，正确。

2. 对补码设计原理的简明理解

补码设计的基本数学原理：

首先，理解“模（modulo）”的概念。先看生活中具有1～12小时指针的机械闹钟，到12小时后，又从0开始（12就是0），即超过12就溢出了。若说是18点，即6点，18/12的余数是6，数学上称之为模运算，符号“mod”，即18 mod 12 = 6，读做“18模12的结果为6”。

现在我们看，若机械闹钟指针指向8点，要把它拨到指向5点，有两种方法：

方法一：回拨，即逆时针拨3小时，即用减法：8-3=5；

方法二：正拨，即顺时针拨9小时，即用加法：8+9=5（不对啊，8+9怎么等于5？可对于这个闹钟，这样的操作是对的）。看看实际数学过程：(8+9) mod 12 =5，即8-3与8+9具有等同效果。减法运算变成了加法运算。同时，注意这个9=12-3，给出了顺时针拨多少小时的一个求法，可以表示成：8-3与8+(12-3)是等效的。

从一般意义上理解“同余数”。

3．求补码的简单方法及由补码求真值的简单方法

（1）由真值求补码的简单方法

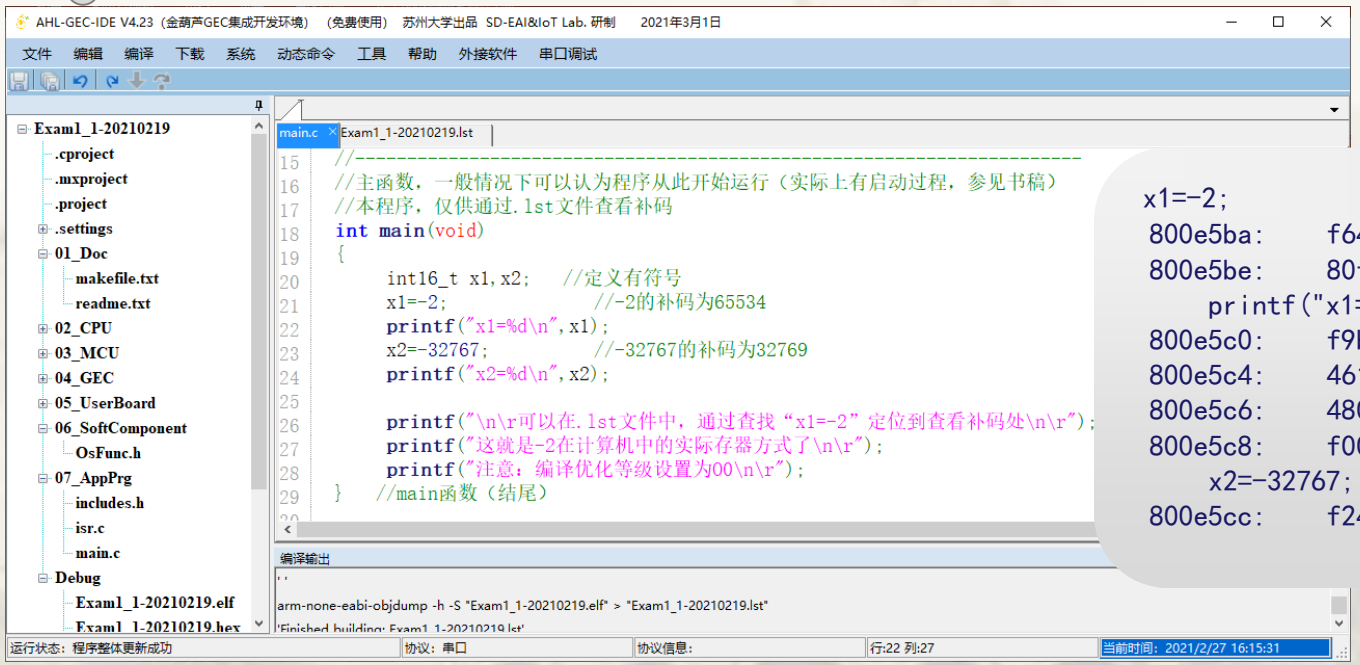
对应n位字长，模m=2^n，整数表达范围是：-2^{n-1}～$(2^{n-1}-1)$，设真值记为x_z，其补码为：

$$x_b=\begin{cases} x_z, & x_z \geqslant 0 \\ 2^n-|x_z|, & x_z \leqslant 0 \end{cases} \qquad (1\text{-}1)$$

【练习1-9】给出32字长的整数表达范围与补码计算方法。

【练习1-10】利用AHL-GEC-IDE开发工具 查找补码。

集成开发环境下载：http://sumcu.suda.edu.cn/AHLwGECwIDE/main.htm
电子资源下载：http://sumcu.suda.edu.cn/wjyl/list.htm
运行环境：Windows10,还建议安装VS2019




```
x1=-2;                  //-2的补码为65534
 800e5ba:    f64f 73fe    movw        r3, #65534    ; 0xfffe
 800e5be:    80fb         strh        r3, [r7, #6]
    printf("x1=%d\n",x1);
 800e5c0:    f9b7 3006    ldrsh.w     r3, [r7, #6]
 800e5c4:    4619         mov         r1, r3
 800e5c6:    480d         ldr         r0, [pc, #52]; (800e5fc <main+0x48>)
 800e5c8:    f001 fa60    bl          800fa8c <myprintf>
    x2=-32767;          //-32767的补码为32769
 800e5cc:    f248 0301    movw        r3, #32769    ; 0x8001
```

【练习1-11】参照“Exam1_1”工程，找出-1和113的补码。

3．求补码的简单方法及由补码求真值的简单方法

（1）由补码求真值的简单方法

对应n位字长，模m=2^n，已知其补码x_b求其真值x_z

$$x_z=\begin{cases} x_b, & 0 \le x_b \le 2^{n-1}-1 \\ x_b-2^n, & 2^{n-1} \le x_b \le 2^n-1 \end{cases} \qquad (1-2)$$

【练习1-9】给出32字长的整数表达范围与补码计算方法。

4. 有符号整数与无符号整数的取值范围

计算机中的数用补码表示，若用一个字节表达有符号整数，其范围是-128～+127，用两个字节表达有符号整数，其范围是-32768～+32767。用一个字节表达无符号整数，其范围是0～255，用两个字节表达无符号整数，其范围是0～65535。

【练习1-13】类比一下，用四个字节、八个字节表达有符号整数与无符号整数，其范围分别是多少？

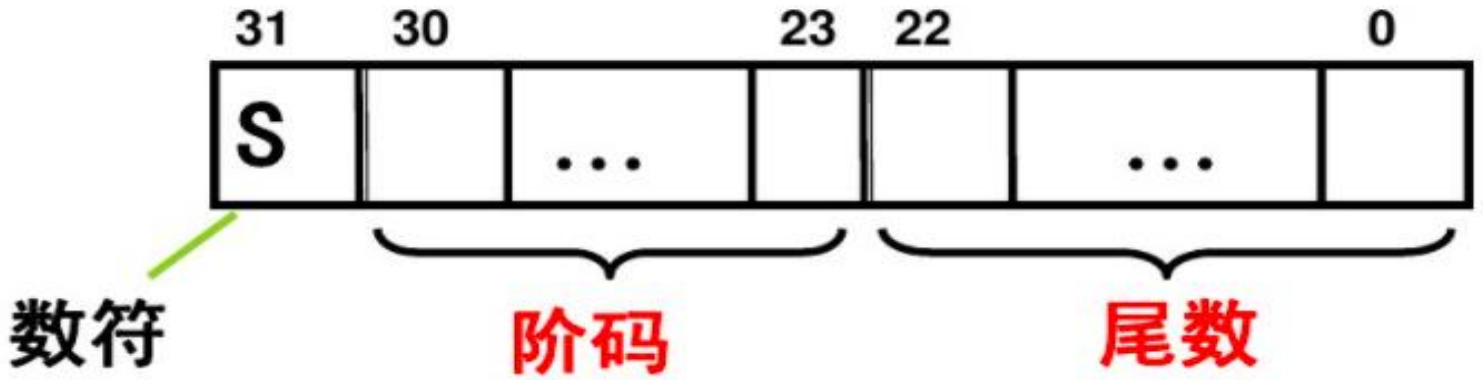
1.4.3 实数在计算机中的浮点数表示方法（难点）

计算机中如何存储数学中带小数点的实数，在高级语言，用浮点数表示，如C语言中单精度浮点数用4个字节表示，双精度浮点数用8个字节表示。
（本节仅要求了解其存储形式）

IEEE于1985年制订了二进制浮点运算标准 IEEE 754（IEEE Standard for Binary Floating-Point Arithmetic，ANSI/IEEE Std 754-1985），后来经修订后，标准号改为 IEC 60559。

1.4.3 实数在计算机中的浮点数表示方法（难点）

IEEE754标准的32位浮点数格式为：



S：数符，0正1负。

阶码：8位以2为底，阶码 = 阶码真值 + 127 。

尾数：23位，采用隐含尾数最高位1的表示方法，

实际尾数24位，尾数真值 = 1 + 尾数

这种格式的非0浮点数真值为：$(-1)^{S} \times 2^{阶码-127} \times (1 + 尾数)$

1.4.3 实数在计算机中的浮点数表示方法（难点）

[例1] $-(0.11)_2$用IEEE短实数浮点格式表示：

$-(0.11)_2=-(1+0.1)*2^{-1}$；　隐含尾数最高位为1

符号为：1

阶码：阶码=真值+127=-1+127=126= $(01111110)_2$

尾数：0.100...00

该浮点数编码：

【练习1-14】利用AHL-GEC-IDE开发工具查找浮点数存储值的步骤
Exam1_2给出了一个浮点数存储例子，电子资源的..\02-Document文件夹下的补充阅读材料中给出了**浮点数的具体计算方法**

1.5 文字在计算机中的存储方式—字符编码（重点）

1.5.1 英文编码—ASCII码

计算机处理的一切信息用“0、1”两个符号存储，但却能处理诸如英文、汉字及其他文字信息。人们把像“a、b、c、你、我、他、……”这类信息称为字符（character）。计算机要能处理它们，必须用二进制表示，给出一些规则，规定“a”用什么二进制表示，“b”用什么二进制表示，等等，这种方式称为字符编码（Character encoding）。因历史发展与应用场合不同，字符编码有许多不同方式，常用的英文编码方式主要有ASCII码，常用的中文编码方式主要有GB2312

1. ASCII码的发布者及发布时间

ASCII码（American Standard Code for Information Interchange），中文翻译为：美国信息交换标准代码。被国际标准化组织（International Organization for Standardization, ISO）定为国际标准，称为ISO 646标准，适用于所有拉丁文字字母。ASCII码由美国国家标准学会（American National Standard Institute , ANSI）于1967年第一次规范发布，1986年为最近一次更新。

2. ASCII码的内容概要

ASCII 码使用一个字节进行编码，分为标准ASCII 码与扩展ASCII 码。标准ASCII 码也叫基础ASCII码，规定最高位为0，其他7位表示数值，其范围为0～127，包括编码32个控制符、10个数字、52个大小写字母及其他符号。

表 1-5　标准 ASCII 概括总结

分类	十六进制值	二进制值	十进制值	符号
32 个控制符	0x00	0000 0000	0	NUL（null）空字符
	……	……	……	
	0x1F	0001 1111	31	US（unit separator）单元分隔符
空格及 15 个标点符号	0x20	0010 0000	32	（space）空格
	0x21～0x2F	0010 0001～0010 1111	33～47	！ " # $ % & （右单引号） ( ) * + , — . /
10 个数字	0x30～0x39	0011 0000～0011 1001	48～57	0～9
6 个符号	0x3A～0x40	0011 1010～0100 0000	58～64	: ; < = ? @
26 个大写字母	0x41～0x5A	0100 0001～0101 1010	65～90	大写字母：A～Z
5 个符号	0x5B～0x60	0101 1010～0110 0000	91～96	\（反斜杠）]（右中括号） ^（脱字符）_（下划线）‘（左单引号）
26 个小写字母	0x61～0x7A	0110 0001～0111 1010	97～122	小写字母：a～z
4 个符号	0x7B～0x7E	0111 1011～0111 1110	123～126	{ | } ～
删除符号	0x7F	0111 1111	127	DEL (delete)

1.5.2 中文编码—GB2312及GBK

1．GB2312及GBK的发布者与发布时间

中文编码《信息交换用汉字编码字符集》是由中国国家标准总局1980年发布，标准号是GB 2312-1980。GB2312标准共收录6763个汉字，为了表示更多的汉字，1995年又颁布了《汉字编码扩展规范》（GBK）GBK与GB2312标准兼容，同时支持ISO/IEC10646-1和GB 13000-1的全部中、日、韩（CJK）汉字，共计20902字。GB 18030-2005《信息技术-中文编码字符集》收录了70244个汉字。

2．GB2312及GBK的内容概要

GB2312基本集共收入汉字6763个和非汉字图形字符682个，每个汉字用两个字节编码，分区进行，区号01-94，每区含有94个位号，这种编码方式也称为区位码。举例来说，“啊”字是GB2312之中的第一个汉字，它的区码为16，位码为01，分别用十六进制表示，分放在高低字节，成为两字节的区位码0x1001，区位码加上0x2020就是国标码0x3021，再加上0x8080就是存储在计算机中的机内码0xB0A1，这就是汉字的计算机编码。

为什么不直接使用国标码将汉字存储在计算机内部呢？

汉字机内码的每个字节都大于128，解决了与西文字符的ASCII码冲突的问题，也给编程判断提供了依据。

3．编码的查看

（1）可以利用word编辑器获得汉字的编码

（2）可以利用微信小程序“金葫芦微机原理学习”查看

本讲作业：第1章习题 6~12