

第二单元学习笔记

yinxuhao [xuhao_yin@163.com]

December 19, 2022

Contents

1	引言	2
2	信息存储	2
2.1	十六进制表示法	2
2.2	字数据大小	3
2.3	寻址和字节顺序	3
2.4	布尔代数	4
2.5	移位运算	5
3	整数表示	5

信息的表示和处理

1 引言

孤立地讲，单个的位不是非常有用，将位组合在一起，再加上某种解释 (interpretation)，即赋予不同的可能位模式以含意。我们就能表示任何有限集合的元素。

三种重要的数字表示：

- 1. 无符号unsigned编码给予传统的二进制表示法
- 2. 补码two's-complement编码是表示有符号整数的最常见的方式。
- 3. 浮点数floating-point编码是表示实数的科学计数法的以 2 为基数的版本。

数据溢出overflow是产生 bug 的一大原因。负数下溢产生极大的正数；正数上溢产生极小的负数。

浮点运算有完全不同的数学属性。

- 1. 由于表示的精度有限，浮点运算是不可结合的。例如

(3.14 + 1e20) - 1e20 = 0.0

but

(3.14 + 1e20 - 1e20) = 3.14

- 2. 该属性不同的原因，是处理数字表示有限性的方式不同——
整数虽只能编码一个相对较小的数值范围，然该表示法是精确的；
浮点数虽可以编码相对较大的数值范围，但这种表示只是近似的。

书中建议的本章学习方式：

深入学习数学语言
学习编写公式和方程式
以及重要属性的推导

2 信息存储

大多数计算机使用 8 位的块或者字节作为最小的可寻址内存单位，而不是内存中单独的比特。

机器级程序将内存视为一个非常大的字节数组，称为虚拟内存，所有可能的地址的集合称为虚拟地址空间virtual address space.

每个程序对象可以简单地视为一个字节块，而程序本身就是一个字节序列。

2.1 十六进制表示法

Hex digit	0	1	2	3	4	5	6	7
Decimal value	0	1	2	3	4	5	6	7
Binary value	0000	0001	0010	0011	0100	0101	0110	0111
Hex digit	8	9	A	B	C	D	E	F
Decimal value	8	9	10	11	12	13	14	15
Binary value	1000	1001	1010	1011	1100	1101	1110	1111

Figure 1: 十六进制表示法。每个十六进制数字都对 16 个值中的一个进行了编码

十六进制转二进制：将十六进制的每一位转换为二进制格式，然后拼接。例如：

十六进制	1	7	3	A	4	C
二进制	0001	0111	0011	1010	0100	1100

所以 $binary_{0x173a4c_{16}} = 000101110011101001001100_2$ 。

二进制转十六进制：将二进制从右到左做 4 个一组的划分，如最左侧不足 4 位则以 0 补之。然后将每个 4 位转换为对应的十六进制数字拼接即可。例如：

二进制	11	1100	1010	1101	1011	0011
十六进制	3	C	A	D	B	3

所以， $hex_{1111001010110110110011_2} = 3cadb3_{16}$

2.2 字数据大小

每台计算机都有一个字长，指明指针数据的标称大小。
C 数据类型的典型大小见下图：

C declaration		Bytes	
Signed	Unsigned	32-bit	64-bit
[signed] char	unsigned char	1	1
short	unsigned short	2	2
int	unsigned	4	4
long	unsigned long	4	8
int32_t	uint32_t	4	4
int64_t	uint64_t	8	8
char *		4	8
float		4	4
double		8	8

Figure 2: 基本 C 数据类型的典型大小 (以字节为单位)

2.3 寻址和字节顺序

小端法little endian: 最低有效字节在最前面放着。

大端法big endian: 最高有效字节在最前面放着。

具体示例见下图：

```
#include <stdio.h>

typedef unsigned char *byte_pointer;

void show_bytes(byte_pointer start, size_t len) {
    size_t i;
    for(i = 0; i < len; i++) {
```

Big endian					
	0x100	0x101	0x102	0x103	
...	01	23	45	67	...

Little endian					
	0x100	0x101	0x102	0x103	
...	67	45	23	01	...

Figure 3: 大端法与小端法

```

        printf(" %.2x", start[i]);
    }
    printf("\n");
}

void show_int(int x) {
    show_bytes((byte_pointer) &x, sizeof(int));
}

void show_float(float x);

void show_pointer(void *x);

void test_show_bytes(int val) {
    int ival = val;
    float fval = (float) val;
    int *pval = &ival;
    show_int(ival);
    show_float(fval);
    show_pointer(pval);
}

```

通过以上代码，可以打印出数据的两位十六进制格式输出。对比结果可以发现，int和float的结果一样，只是排列的大小端不同，而指针值不同，与机器相关。

二进制代码是不兼容的。

2.4 布尔代数

\sim		$\&$	0	1	\mid	0	1	\sim	0	1
0	1	0	0	0	0	0	1	0	0	1
1	0	1	0	1	1	1	1	1	1	0

Figure 4: 布尔代数的运算。二进制 0 和 1 代表逻辑值 TRUE 和 FALSE. 以上四张图依次是逻辑运算符 NOT AND OR EXCLUSIVE-OR

位向量一个很有用的应用就是表示有限集合。利用位向量 $[a_{w-1}, \dots, a_1, a_0]$ 可以编码任何子集 $A \in 0, 1, \dots, w-1$ 。

例如，定义规则 $a_i = 1 \iff i \in A$ 。

位向量 $a \doteq [01101001]$ 表示集合 $A = 0, 3, 5, 6$ ，而位向量 $b \doteq [01010101]$ 表示集合 $B = 0, 2, 4, 6$ 。

编码集合的使用方法是使用布尔运算。

例如： $a \& b \rightarrow [01000001]$ ，对应于 $A \cap B = 0, 6$ 。

它的实际应用，还有使用位向量作为掩码有选择地使用或屏蔽一些信号，该掩码就是设置为有效信号的集合。

C 语言中的位级运算，其实是按照各个位对应的位运算来的。

而 C 语言中的逻辑运算 (`||`、`&&`、`!`) 则是把所有的非零参数都表示 TRUE，参数 0 表示为 FALSE。它们只返回 1 或 0。而位级运算只在参数特殊时才与之有相同的结果。

2.5 移位运算

$x \ll k$: 左移 k 位，即丢弃最高 k 位，右端补充 k 个 0。

$x \gg k$: 右移 k 位，支持逻辑右移和算术右移。逻辑右移在左端补充 k 个 0，算术右移则在左端补充 k 个最高有效位 (符号位)。

对无符号数，右移必须是逻辑的。

移位运算符是从左至右可结合的。

3 整数表示

Symbol	Type	Meaning
$B2T_w$	Function	Binary to two's complement
$B2U_w$	Function	Binary to unsigned
$U2B_w$	Function	Unsigned to binary
$U2T_w$	Function	Unsigned to two's complement
$T2B_w$	Function	Two's complement to binary
$T2U_w$	Function	Two's complement to unsigned
$TMin_w$	Constant	Minimum two's-complement value
$TMax_w$	Constant	Maximum two's-complement value
$UMax_w$	Constant	Maximum unsigned value
$+^t_w$	Operation	Two's-complement addition
$+^u_w$	Operation	Unsigned addition
$*^t_w$	Operation	Two's-complement multiplication
$*^u_w$	Operation	Unsigned multiplication
$-^t_w$	Operation	Two's-complement negation
$-^u_w$	Operation	Unsigned negation

Figure 5: 整数的数据与算术操作术语。下标 w 表示数据中表示中的位数