南京大学120周年校庆
120th ANNIVERSARY
NANJING UNIVERSITY
1902 ~ 2022

1902 2022

数据的机器级表示-1

Data Representation at Machine Level

李杉杉

南京大学120周年校庆
120th ANNIVERSARY
NANJING UNIVERSITY
1902 - 2022

CONTENT

目录

01 位和数据类型

南京大学120周年校庆
120th ANNIVERSARY
NANJING UNIVERSITY
1902 – 2022

位和数据类型

```
#include<stdio.h>

int main () {
        Int X;
        scanf ("%d", &x);
        printf ("2 + %d = %d.", x , 2 + x);
        printf ("2 - %d = %d.", x , 2 - x);
}
```

数据的表示

数据的运算

- 现代计算机里所有信息都是采用数字化的形式表示的
  - 整数、小数、文字、图像、声音等等
- 当一个C程序执行时，在计算机内部究竟发生了什么
  - 需要从最底层—计算机内表示数值的方式开始
- 如果程序处理的是图像、视频、声音、文字等数据，那么计算机将：
  - 如何获取这些数据？
  - 如何表示这些数据？
  - 如何处理这些数据？

位和数据类型

抽象层次

- 问题
- 算法
- 程序
  - 语言处理
  - 操作系统
- 指令集结构
- 微处理器
- 逻辑电路
- 元件

感知环境信息

↓

树、链表等结构化数据描述

↓

int, float, array, struct等类型

↓

指令指定寄存器或内存中数据

↓

ALU中运算或总线上传输的数据

↓

逻辑门

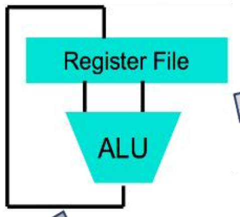
↓

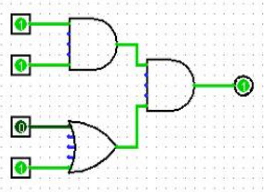
位信息

```
temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;
```

```
lw   $t0, 0($2)
lw   $t1, 4($2)
sw   $t1, 0($2)
sw   $t0, 4($2)
```

```
0000 1001 1100 0110 1010 1111 0101 1000
1010 1111 0101 1000 0000 1001 1100 0110
1100 0110 1010 1111 0101 1000 0000 1001
0101 1000 0000 1001 1100 0110 1010 1111
```





位和数据类型

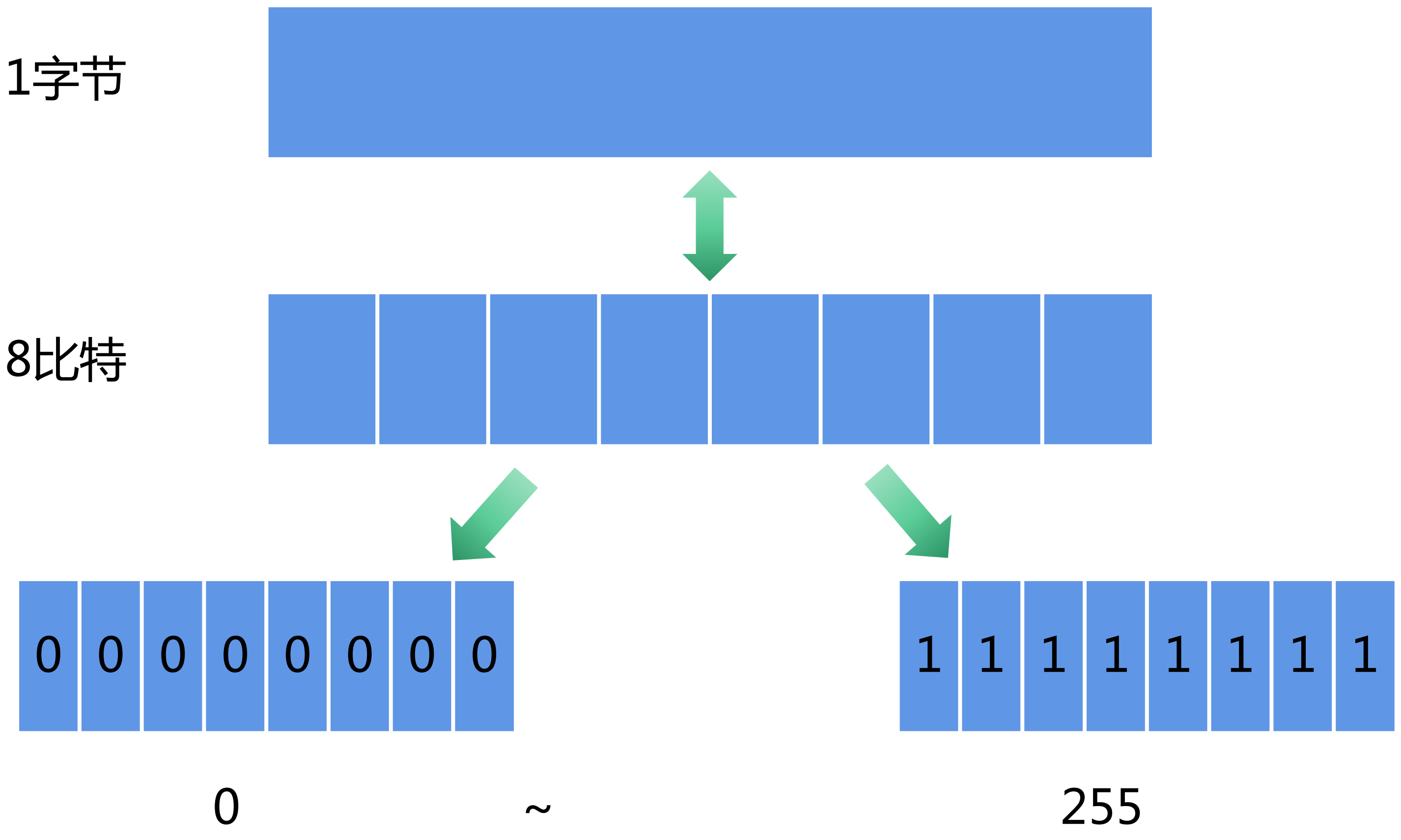
信息的最小单位—位/比特

- 在冯·诺依曼结构中，所有信息（代码和数据）都采用二进制编码
  - 编码：用少量简单的基本符号对复杂多样的信息进行一定规律的组合
- 在计算机内部，数以亿计、微小、快速的电子元件控制电子的流动
- 这些元件对电路中电压的有无做出反应
- 如果存在电压用“1”表示，不存在电压用“0”表示
- “0”和“1”被称为比特（bit），或位
- “二进制位”（binary digit）的缩写

位和数据类型

信息存储/读取的基本单元—字节

- 比特=位=bit=b
- 字节=Byte=B
- 1字节 = 8比特



位和数据类型

为什么是二进制，不是十进制？

- 既然计算机可以识别电压数值，是否可以使用电压值做出10进制的信息表示
- 第一台通用计算机ENIAC采用的是十进制，但效率太低，电路太复杂
    - 测量电压的具体值，10种状态（0-9）表示复杂
    - 要求电路的电压值必须稳定，不允许从3.3伏波动到2.9伏
    - 运算组合状态过多（55种）

位和数据类型

为什么是二进制，不是十进制？

- 采用二进制
  - 只需使用两个基本符号0和1，个数少
  - 物理上容易实现，只需两个稳定态的物理器件
  - 数字电路的两个状态表示（电压高低/有无），无需测量电压的具体值
  - 二进制编码、计数运算规则简单，运算组合状态最少（3种）
  - 对应逻辑命题中的“真”和“假”
  - 便于使用逻辑电路实现算术运算（一个异或门）

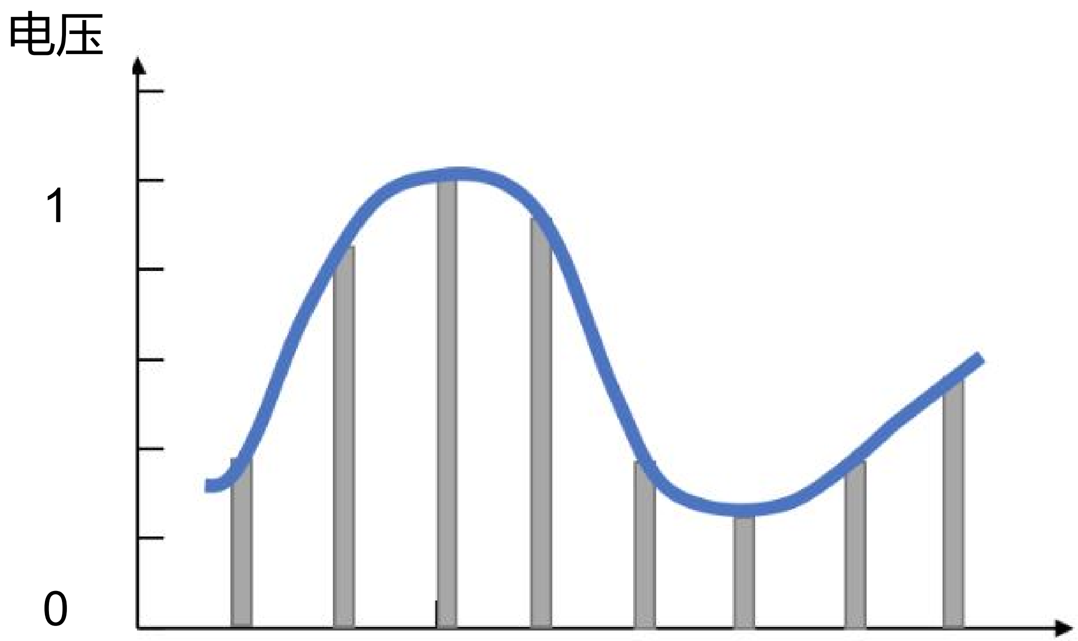
位和数据类型

两种稳定状态

- 自然界的事物通常具有两种稳定的状态
  - 磁盘
    - 使用1和0表示磁化和未磁化
  - 光盘
    - 使用1和0表示凹（聚光）和凸（散光）
  - 手电筒/电灯（光源）
    - 亮和灭

两种稳定状态——接近0和远离0

- 计算机并不是区分电压的绝对不存在（即0）和绝对存在（即1）
- 计算机电路区分的是接近0的电压和远离0的电压
  - 如果计算机把1.1伏的电压表示为1，把0伏的电压表示为0
  - 那么0.9伏的电压也会被视作1，而0.2伏的电压会被当作0



位和数据类型

位组合：计算机中的比特/位表示任意数值的方式

- 计算机要解决一个真正的问题，必须能唯一的识别出许多不同的数值，而不仅仅是0和1
- 为了唯一的识别出多个数值，必须对多个位进行组合
- 如果用8位（对应8根线路上的电压）
  - 使用01001110表示某一个特定值，用11100111表示另一个值
  - 最多能区分出256（即2^8）个不同的值
  - $(00000000)_2 \sim (11111111)_2$
  - 对应十进制：：$(0)_{10} \sim (255)_{10}$
  - 对应十六进制：(00)16 ~ $(FF)_{16}$
- 如果有k位，最多能区分出2^k个不同的值
  - 这些k位的每一种组合都是一个编码
  - 对应着某个特定的值

位和数据类型

位运算

- 除了需要表示出不同的数值之外，还需要对这种表示出来的信息进行运算
- 1679年，德国数学家莱布尼茨（戈特弗里德·威廉·莱布尼茨，Gottfried Wilhelm (von) Leibniz，1646年7月1日 - 1716年11月14日）发表了一篇关于二进制表示及算术运算的论文
- 1854年，英国数学家乔治·布尔（ George Boole，1815年11月2日 - 1864年12月8日）给出了二进制的逻辑运算，布尔代数即由此得名
- 这些工作奠定了现代计算机工作的数学基础

计算机中的数据类型

- 同一个数值，多种表示方法
- 整数：6
  - 十进制计数法：“6”
  - 一元计数法：“正一”
  - 罗马字符：“VI”
  - 二进制计数法：“0110”
- 某种表示法：编码，运算，数据类型

位和数据类型

计算机中的数据类型

- 如果在计算机上能对以某种表示法编码的信息进行运算，这种特殊的表示法就可以被称为数据类型
- 定义数据的性质和操作规则，编程语言中的重要概念
- 在大多数计算机上存在着多种数值表示方法
    - 用来表示进行算术运算的正负整数的二进制补码整数
    - 用来表示从键盘输入计算机或显示在计算机显示器上的字符的ASCII码
    - 类似于十进制“科学计数法”的浮点数数据类型

南京大学120周年校庆
120th ANNIVERSARY
NANJING UNIVERSITY
1902 – 2022

02 进位计数制

进位/位置计数制（定位数制）

- 一种记数（对数量计数的）方式
- 使用有限种数字符号来表示所有的数值
- R进制，逢R进一
- 基数/底数（R）：使用的数字符号的数目
- 位权（R^i）：进位计数制所表示的数中，用来表示i位上一个实际数值大小的固定常数
- K_i：位号为i位上的数字符号

$$N = \sum_{i=-m}^{n-1} K_i \times R^i$$

二 进 制：0 1（计算机中使用）

八 进 制：0 1 2 3 4 5 6 7

十 进 制：0 1 2 3 4 5 6 7 8 9（日常生活广泛使用）

十六进制：0 1 2 3 4 5 6 7 8 9 A B C D E F

二进制

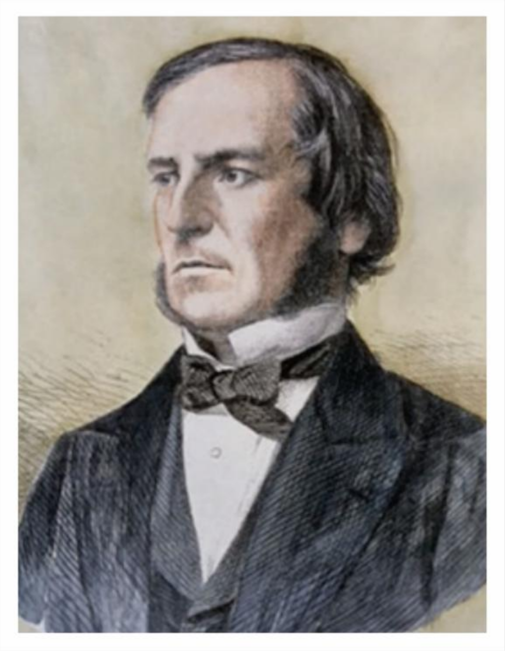
- 计算机采用二进制数表示数值
  - 基数为R=2
  - 数字符号为0和1
  - 8位（1字节）有效数字，n=8
  - 如数字$N=(30)_{10}$可以表示为：
    - $(00011110)_2 = 0\times2^7+0\times2^6+0\times2^5+1\times2^4+1\times2^3+1\times2^2+1\times2^1+0\times2^0$
    - i=0时，$K_0 = 0$，$R^0 = 2^0$

$$N = \sum_{i=-m}^{n-1} K_i \times R^i$$

进位计数制

二进制

- 二进制是计算机编码存储和操作信息的核心
- 围绕0和1的丰富数学知识体系
- 数学家George Boole
  - 布尔代数
  - 逻辑值True（真）和False(假）
  - 编码为二进制的1和0
  - 逻辑推理的基本原则



进位计数制

十六进制表示法

- 由二进制的位置计数法发展而来
- 二进制占位多、容易出错
- 方便人们手工处理二进制数位

二进制	1111 1010 0001 1101 0011 0111 1011
十六进制	0XFA1D37B 0xfa1d37b 0xFa1D37b

八进制表示法

- 八进制数与二进制数之间的转换
  - 一位八进制数相当于3位二进制数
  - 八进制数转换成二进制数，或二进制数转换成八进制数很方便
- 例如：$(563)_8 = (101,110,011)_2$

  $(0.764)_8 = (0.111,110,100)_2$

进位计数制

数制转换

- 十进制数转二进制数
- 二进制数转十进制数
- 二进制数转八进制
- 二进制数转十六进制数

十进制转二进制

11

整数部分

除尽为止：1011

0.625

小数部分

求得位数满足要求为止

进位计数制

二进制转十进制

- 逐位码权累加求和

$$N = \sum_{i=-m}^{n-1} K_i \times R^i$$

例：

$(10110110)_2$

$= 1\times2^7+0\times2^6+1\times2^5+1\times2^4+0\times2^3+1\times2^2+1\times2^1+0\times2^0$

$= 128 + 0 + 32 + 16 + 0 + 4 + 2 + 0$

$= 182$

进位计数制

二进制转十六进制

- 小数点为界
- 向两边四位一组变为一位十六进制数
- 例：

$(1001\ 1100\ .\ 01)_2$

$= (1001\ 1100\ .\ 01\textcolor{red}{00})_2$

$= (9C.4)_{16}$

二进制转八进制

- 小数点为界
- 向两边三位一组变为一位八进制数
- 例：

$(10\ 011\ 100\ .\ 01)_2$

$= (10\ 011\ 100\ .\ 01\textcolor{red}{0})_2$

$= (234.2)_8$

常用的2的幂

- $2^5 = 32$, $2^6 = 64$, $2^7 = 128$, $2^8 = 256$, $2^9 = 512$
- $2^{10} = 1024$ (1 KB), $2^{11} = 2048$, $2^{12} = 4096$
- $2^{13} = 8192$, $2^{14} = 16384$, $2^{15} = 32768$
- $2^{16} = 65536$, $2^{20} = 1048576$ (1 MB)
- $2^{30} = 1073741824$ (1 GB), $2^{40} = 1$ TB
- $2^{50} = 1$ PB, $2^{60} = 1$ EB
- $2^{70} = 1$ ZB, $2^{80} = 1$ YB

南京大学120周年校庆
120th ANNIVERSARY
NANJING UNIVERSITY
1902 – 2022

03 整数数据类型

整数数据类型

- 编程语言中的基本数据类型之一
- 表示整数：包括正整数、负整数和零
- 整数数据类型有很多用途，
  - 存储和处理整数值，如计数（任务执行次数、选课人数）、索引、循环计数器等
  - 声明整型变量并对其进行赋值，也可以进行整数运算，如加减乘除等

整数的二进制表示

- 无符号整数（Unsigned Integer）
  - 仅能表示非负整数（包括零）
  - 不使用符号位，将所有位都用于表示数值
  - unsigned int y = 20; // 无符号，仅表示正数
- 有符号整数（Signed Integer）（默认情况）
  - 可以表示正数、负数和零
  - 使用符号位和数值位一起编码而成
  - 最高位作为符号位，0表示正数，1表示负数
  - signed int x = -10; // 有符号，表示负数

整数数据类型

无符号整数

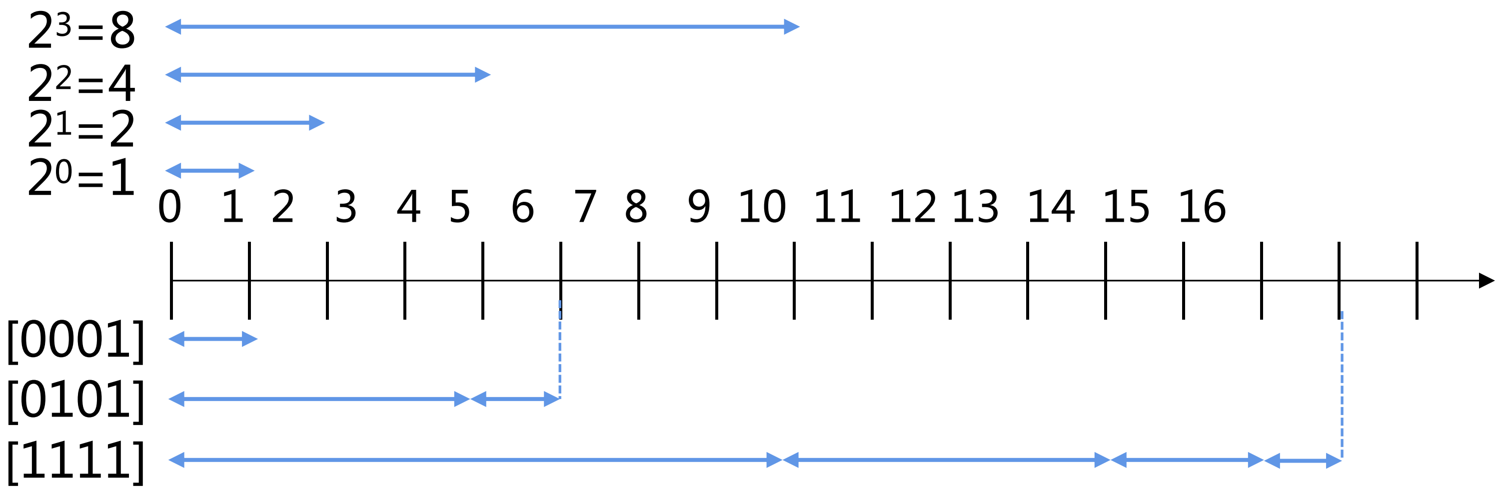
- 二进制向量X有k位数$X = [x_{k-1}, x_{k-2}, \ldots\ x_1\ ,\ x_0]$
- $B2U_k(X) = x_{k-1} \cdot 2^{k-1} + x_{k-2} \cdot 2^{k-2} + \ldots + x_1 \cdot 2^1 + x_0 \cdot 2^0$

x_{k-1}	x_{k-2}	…	x_1	x_0

整数数据类型

无符号整数

- $B2U_4([0001])=0\cdot 2^3+0\cdot 2^2+0\cdot 2^1+1\cdot 2^0=1$
- $B2U_4([0101])=0\cdot 2^3+1\cdot 2^2+0\cdot 2^1+1\cdot 2^0=5$
- $B2U_4([1111])=1\cdot 2^3+1\cdot 2^2+1\cdot 2^1+1\cdot 2^0=15$



整数数据类型

无符号整数

字长	二进制	十六进制	最大值
8	1111 1111	0xFF	2^8-1
16	1111...1111（16位）	0xFFFF	2^{16}-1
32	1111...1111（32位）	0xFFFFFFFF	2^{32}-1
64	1111...1111（64位）	0xFFFF...FFFF(16位）	2^{64}-1

字长：计算机CPU（ALU）一次能并行处理的二进制数据的位数，这里被处理的数据称为字（word）

字长为k，可表示从0到2^k-1共2^k个整数

整数数据类型

无符号整数

字长	二进制	十六进制	最大值
8	1111 1111	0xFF	2^8-1
16	1111...1111（16位）	0xFFFF	$2^{16}-1$
32	1111...1111（32位）	0xFFFFFFFF	$2^{32}-1$
64	1111...1111（64位）	0xFFFF...FFFF(16位）	$2^{64}-1$

字长为k，可表示从0到2^k-1共2^k个整数

整数数据类型

有符号整数

- 二进将k位的2^k个不同的数字分为两半，一半表示正数，另一半表示负数
- k=4的时候可以表示2^k=16个数字
  - 从+1到+7的7个正数
  - 从-1到-7的7个负数
  - 剩下2个，一个用来表示0，还剩一个码字可以用来表示那个数字？
  - 从+1到+7，从-1到-7，是哪些码字与其匹配？

整数数据类型

有符号整数

- 二进制表示：原码、反码、补码表示法
  - 编码是为了解决正负号问题
  - 计算机中几乎不用反码，运算普遍使用补码
  - 二进制（有符号）补码的运算
  - 二进制-十进制转换

整数数据类型

正数

- 使用位置计数法直接表示
- k位，用2^k个码字的一半来表示从$[0,2^{k-1}-1]$的正数
- 第一位代表符号位（正数符号位为0）
- 原码、反码、补码
- 三码合一（相同）
- k=4，真值范围[0,7]，表示的最大整数为7

二进制（原码/反码/补码）	十进制
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7

整数数据类型

负数：原码

- 符号数值表示法（直观）
- 符号位0/1表示数据正/负
- 符号位+数值绝对值
- k=4
    - 负数真值范围[-(2^{k-1}-1), -1]
    - 表示的最小整数为-7

二进制（原码）	十进制
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	-0
1001	-1
1010	-2
1011	-3
1100	-4
1101	-5
1110	-6
1111	-7

整数数据类型

负数：反码

- 另一种思路
- 正数原码，按位取反
- k=4
  - 负数真值范围[-(2^{k-1}-1), -1]
  - 表示的最小整数为-7

原码	反码	十进制
0000	0000	0
0001	0001	1
0010	0010	2
0011	0011	3
0100	0100	4
0101	0101	5
0110	0110	6
0111	0111	7
1111	1000	-7
1110	1001	-6
1101	1010	-5
1100	1011	-4
1011	1100	-3
1010	1101	-2
1001	1110	-1
1000	1111	-0

负数：原码和反码

- 0的表示不唯一，不利于程序员编程
  - $[+0]_{原} = 0000 \quad [-0]_{原} = 1000$
  - $[+0]_{反} = 0000 \quad [-0]_{反} = 1111$
- 加、减运算方式不统一，尤其当a<b时，实现a-b比较困难
  - $(1)_{10}-(2)_{10}=(1)_{10}+(-2)_{10}=(0001)_{原码}+(1010)_{原码}=(1011)_{原码}=(-3)_{10}(\quad)$
  - $(4)_{10}-(3)_{10}=(4)_{10}+(-3)_{10}=(0100)_{反码}+(1100)_{反码}=(10000)_{反码}=(0)_{10}(\quad)$
- 需要额外对符号位进行处理，不利于硬件设计

? 更合理的表示？

补码

整数数据类型

负数：补码

- 负数表示法：尽可能使逻辑电路最简单
- 几乎所有的计算机都使用相同的基本结构——算术逻辑单元（Arithmetic and Logic Unit，ALU）来进行加法运算
- ALU并不知道（也不关心）所加的两个位组合表示什么，它只是简单的将二进制数相加
- 0100 (4)
- + ???? (-3)
- = 0001 (1)

整数数据类型

负数：补码

- A+（-A）=0

```
      0101（+5）
+        ?（-5）
=     0000（0）

      0101（+5）
+     1011（-5）
=     0000（0）
```

补码	十进制
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	
1001	
1010	
1011	-5
1100	
1101	
1110	
1111	

整数数据类型

负数：补码

- A+（-A）=0（A表示正数）
- -A的表示：
  - -A的反码加1
  - 或A取反加1

	A
+	A的反码
=	11…11
+	00…01
=	00…00（0）

补码	十进制
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	
1001	
1010	
1011	-5
1100	
1101	
1110	
1111	

整数数据类型

负数：补码

- 在对每个数值加0001后，应得到正确的结果
  - 验证-7~-1的二进制表示
- 1000：-8

    1000（？）
  + 0001（1）
  = 1001（-7）

- 1111和0000：-1和0
-     1111（-1）
- +   0001（1）
- =   (1)0000（0）
- 在做补码算术运算时这个进位总是被忽略

补码	十进制
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	-8
1001	-7
1010	-6
1011	-5
1100	-4
1101	-3
1110	-2
1111	-1

整数数据类型

负数：补码

- 模运算编码（受到计算机字长限制）
- 正数原码按位取反加1
- 或反码加1
- k=4
    - 负数真值范围[2^{k-1}, -1]
    - 表示的最小整数为-8

补码	十进制
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	-8
1001	-7
1010	-6
1011	-5
1100	-4
1101	-3
1110	-2
1111	-1

负数：补码

- **-6的二进制补码表示是什么（采用4位表示）？**

1. A：6，0110
2. -A的反码 1001（A按位取反）
3.

$$
\begin{array}{rl}
 & 1001 \\
+ & \underline{0001} \\
= & 1010\ (-6)
\end{array}
$$

验证

$$
\begin{array}{rl}
 & 0110(6) \\
+ & \underline{1010(-6)} \\
= & (1)0000
\end{array}
$$

补码拓展：

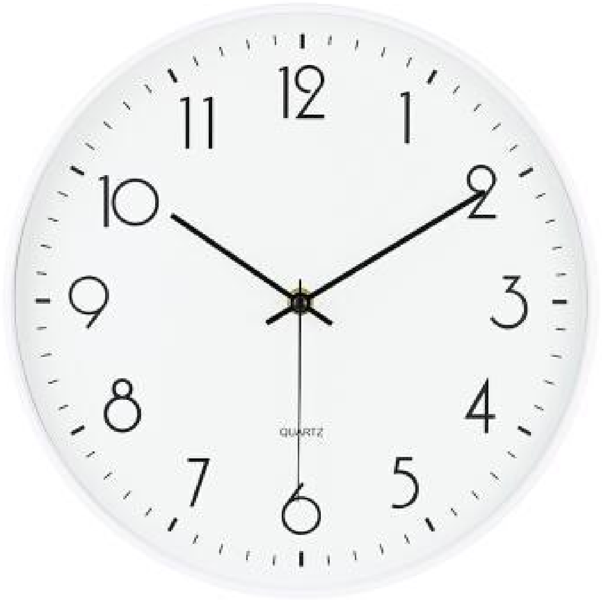
- 补码是一种模运算编码
- “模”是指一个计量系统的计数范围
  - 如时钟的计量范围是0～11,模=12
- 计算机也可以看成一个计量系统，它也有一个计量范围，即都存在一个“模”
  - 表示n位的计算机计量范围是0～2^n-1,模=2^n
- “模”实质上是计量系统产生“溢出”的量，它的值在计量系统上表示不出来，计量系统上只能表示出模的余数
- 任何有模的计量系统，均可化减法为加法运算，简化设计，降低成本

整数数据类型

补码拓展：

在一个模运算系统中，一个数与它除以“模”后的余数等价。

时钟是一种模12系统

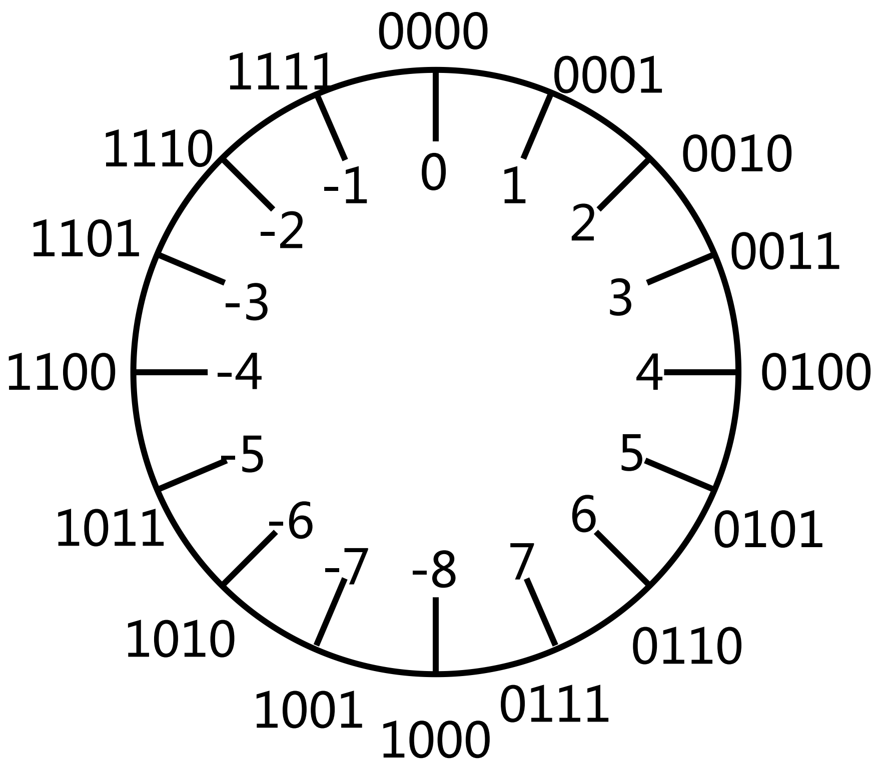


- **例**：图中时针指向的是10点，要将它拨向6点，有两种拨法
  - 1、逆时针拨4格，10-4=6
  - 2、顺时针拨8格，10+8=18[mod 12]=6
  - -4 = 8 [mod 12]
- **加和减的统一**：
  - 一个负数的补码等于模减该负数的绝对值
  - 对于某一确定的模，某数减去小于模的另一数,总可以加上另一数负数的补码来代替
  - **一个负数的补码**等于将**对应正数补码各位取反、末位加一**

整数数据类型

补码拓展：一个n位运算器，只能保留低n位的运算结果，即模为2^n的补码运算

模为2^4的时钟系统



补码的定义：

$$[X]_c = 2^n + X \ (-2^n \le X \le 2^n,\ mod\ 2^n)$$

补码表示：

000…000 ~ 011…111：表示的值不变

100…000 ~ 111…111：表示的值由

$$2^{k-1} \sim 2^k - 1$$

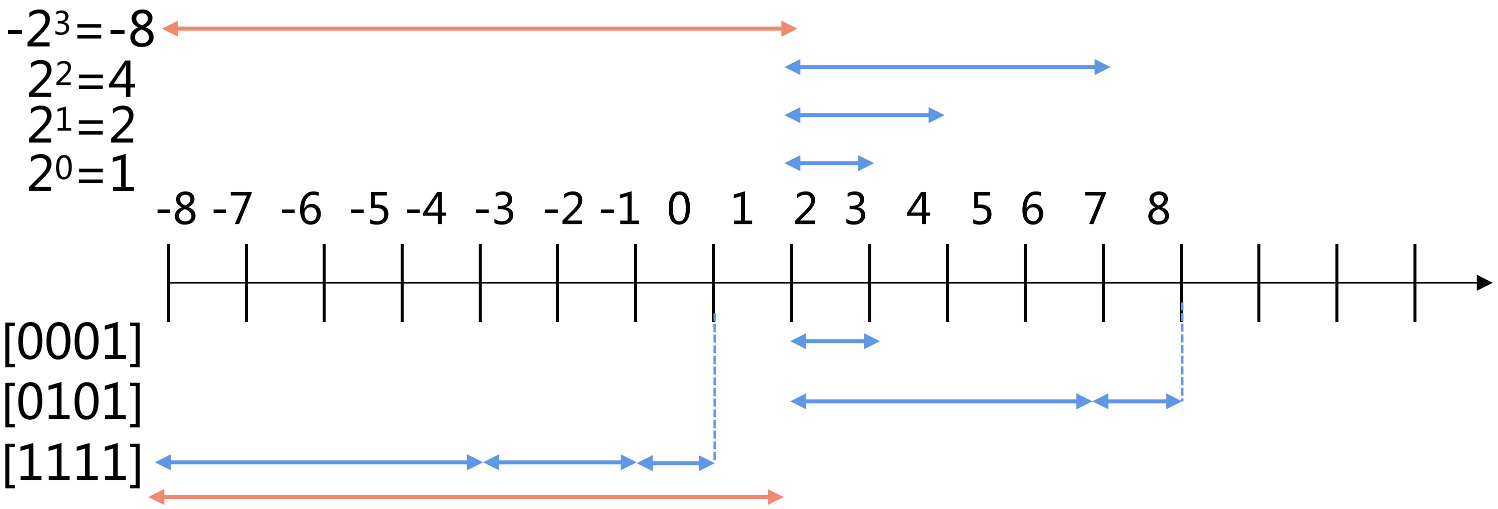
变为

$$-2^{k-1} \sim -1$$

整数数据类型

有符号补码整数：

- $B2U_k(y)=0\cdot 2^{k-1}+y_{k-2}\cdot 2^{k-2}+\ldots+y_1\cdot 2^1+y_0\cdot 2^0$
- $B2U_k(y)=-1\cdot 2^{k-1}+y_{k-2}\cdot 2^{k-2}+\ldots+y_1\cdot 2^1+y_0\cdot 2^0$
- 例：
    - $B2U_4([0101])=0\cdot 2^3+1\cdot 2^2+0\cdot 2^1+1\cdot 2^0=5$
    - $B2U_4([1011])=-1\cdot 2^3+0\cdot 2^2+1\cdot 2^1+1\cdot 2^0=-5$



整数数据类型

有符号补码整数：

字长	二进制	十六进制	最小值
8	1000 0000	0x80	-2^7
16	1000...0000（16位）	0x8000	-2^{15}
32	1000...0000（32位）	0x80000000	-2^{31}
64	1000...0000（64位）	0x8000...0000(16位)	-2^{63}

字长	二进制	十六进制	最大值
8	0111 1111	0x7F	2^7-1
16	0111...1111（16位）	0x7FFF	$2^{15}-1$
32	0111...1111（32位）	0x7FFFFFFF	$2^{31}-1$
64	0111...1111（64位）	0x7FFF...FFFF(16位)	$2^{63}-1$

字长为k，可表示从-2^{k-1}到$2^{k-1}-1$共2^k个整数

有符号补码整数：特殊数值

字长	有符号整数（-1）	无符号整数（最大值）
8	1111 1111=-1	2^8-1
16	1111...1111（16位）=-1	$2^{16}-1$
32	1111...1111（32位）=-1	$2^{32}-1$
64	1111...1111（64位）=-1	$2^{64}-1$

$-1 \neq 1000\ 0001$

有符号补码整数的二进制-十进制转换：

- 一个8位的二进制补码数采取如下格式：

  $a_7\ a_6\ a_5\ a_4\ a_3\ a_2\ a_1\ a_0$

- 首先检查最高位a_7,确定符号
  - 最高位为0，正数，此时补码与原码相同，直接转换
  - 最高位为1，负数，此时对补码取反加1，再进行转换
- 转换：$a^6x2^6 + a^5x2^5 + a^4x2^4 + a^3x2^3 + a^2x2^2 + a^1x2^1 + a^0x2^0$
- 最后，如果最高位为1，是负数，转换后的十进制数需加一个负号前缀

整数数据类型

示例：$(01110110)_2$和$(11110010)_2$

- $(01110110)_2$
  - 检查最高位a_7,确定符号位为0，正数，此时补码与原码相同，直接转换
  - 转换：$2^6+2^5+2^4+2^2+2^1 = 64+32+16+4+2=(118)_{10}$
- $(11110010)_2$
  - 检查最高位a_7,确定符号位为1，负数，此时对补码取反加1，得到00001110
  - 转换：$1x2^3 + 1x2^2 + 1x2^1=(14)_{10}$
  - 最高位为1，是负数，转换后的十进制数需加一个负号前缀，结果为$(-14)_{10}$

有符号补码整数的十进制-二进制转换：

- 为十进制数A找到一个对应的8位二进制表示：
  - $a_7\ a_6\ a_5\ a_4\ a_3\ a_2\ a_1\ a_0$
- 当最高位a_7 为0，正数，$A = a_6\ x2^6 + a_5\ x2^5 + a_4\ x2^4 + a_3\ x2^3 + a_2\ x2^2 + a_1\ x2^1 + a_0\ x2^0$
- 如何找到a_i（$i = 0$，1，… 6）的值？

正二进制数若最右端的数字为1，奇数；否则为偶数

整数数据类型

示例：+123（字长为8）

- 正数，最高位a7=0　$123 = a_6\times2^6+ a_5\times2^5+ a_4\times2^4+ a_3\times2^3+ a_2\times2^2+ a_1\times2^1+ a_0\times2^0$
- 奇数，a_0=1
- 在等式两端同时减去1　$122 = a_6\times2^6+ a_5\times2^5+ a_4\times2^4+ a_3\times2^3+ a_2\times2^2+ a_1\times2^1$
- 在等式两端同时除以2　$61 = a_6\times2^5+ a_5\times2^4+ a_4\times2^3+ a_3\times2^2+ a_2\times2^1+ a_1\times2^0$
- 奇数，a_1=1
- 在等式两端同时减去1　$60 = a_6\times2^5+ a_5\times2^4+ a_4\times2^3+ a_3\times2^2+ a_2\times2^1$
- 在等式两端同时除以2　$30 = a_6\times2^4+ a_5\times2^3+ a_4\times2^2+ a_3\times2^1+ a_2\times2^0$
- *偶数*，a_2=0　$15 = a_6\times2^3+ a_5\times2^2+ a_4\times2^1+ a_3\times2^0$
- 奇数，a_3=1　$7 = a_6\times2^2+ a_5\times2^1+ a_4\times2^0$
- 奇数，a_4=1　$3 = a_6\times2^1+ a_5\times2^0$
- 奇数，a_5=1　$1 = a_6\times2^0$
- 奇数，a_6=1

+123的二进制表示为0111 1011

有符号补码整数的十进制-二进制转换：

- 将*N* 的绝对值“除2取余”得到二进制表示；
- 若为正十进制数，则在二进制数前加0，得到k位结果；
- 若为负十进制数，计算负数的补码
  - 正数的原码按位“取反加1”
  - 负数的反码加1

整数数据类型

示例：+123

除2取余，倒着取

		余数	
2	123		
2	61	1	低位
2	30	1	
2	15	0	
2	7	1	
2	3	1	
2	1	1	
	0	1	高位

- 将余数从高位向低位依次排列
- 1111011
- 正数，最高位补0
- 二进制表示为 0111 1011

整数数据类型

示例：-215（字长16）

		余数
2	215	
2	107	1
2	53	1
2	26	1
2	13	0
2	6	1
2	3	0
2	1	1
	0	1

结果 $(215)_{10} = (0000000011010111)_2$

或写为：215**D** = 0000000011010111**B**

正数二进制->负数补码

原码$(-215)_{10} = (1000000011010111)_2$

反码$(-215)_{10} = (1111111100101000)_2$

补码$(-215)_{10} = (1111111100101001)_2$

习题

- 书面作业
  - 6.1
  - 6.2
  - 6.3
  - 6.4
  - 6.5
  - 6.7
  - 6.8

谢　谢

诚耀百廿 雄创一流