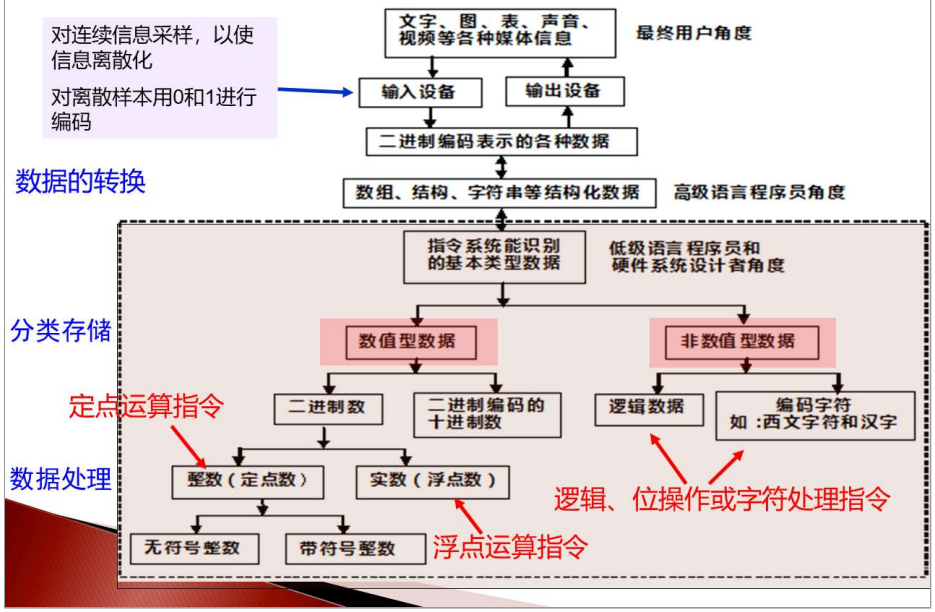
华中农业大学 HUAZHONG AGRICULTURAL UNIVERSITY

第二章:数据的机器级表示

问题的引入：信息从用户到计算机



结论：

- 不管以什么形态出现，在计算机内部，数据最终都由机器指令来处理。
- 从计算机指令集体系结构（Instruction Set Architecture，简称ISA）角度来看，计算机中底层的机器级表示数据只有几类简单的基本数据类型。

信息的二进制编码

Q：C语言中哪些类型是数值数据？哪些是非数值数据...？

- 机器级数据分两大类：
  - 数值数据：无符号/带符号整数、浮点数（实数）、十进制数
  - 非数值数据：逻辑数（包括位串）、西文字符和汉字
- 计算机内所有信息都用二进制编码
- 用二进制编码的原因：
  - 制造二个稳定态的物理器件容易
  - 二进制编码、计数、运算规则简单
  - 与逻辑命题真假对应，便于逻辑运算和实现。
- 真值和机器数的概念
  - 真值：现实中世界中数的值
  - 机器数：计算机内部的0/1序列

Ch2：Data Representation
第二章 数据的机器级表示

- 2.1 数值数据的表示
- 2.2 非数值数据表示
- 2.3 数据的宽度、存储排列、纠/检错

2.1 数值数据的表示

- 定点数的表示
  - 进位计数制
  - 定点数的二进制编码
    - 原码、补码、移码表示
  - 定点整数的表示
    - 无符号整数、带符号整数
- 浮点数格式和表示范围
  - 浮点数的规格化
  - IEEE754浮点数标准
- C语言程序中的整数类型、浮点数类型

一、进位计数制

1. 基本概念

€ **进位计数制：**
用少量的数字符号（也称码），按先后次序把它们排成数位，由低到高进行计数，计满进位，这样的方法称为进位计数制.

€ **基数：**进位制的基本特征数，即所用到的数字符号个数。

€ **权：**进位制中各位“1”所表示的值

2. 进制数的表示

1)用下标加以标注
$(10000111.1011)_2$
$(1011)_{10}$

2)用后缀字母表示
B 二进制
O 八进制
D 十进制
H 十六进制

3.常用几种进制的对应关系

十进制	二进制	八进制	十六进制
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	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

4. 进制转换

(1)、X ⟶ 十　　:按权展开的多项式

(2)、十 ⟶ 二

整数转换规则：除2取余，直至商为0，先得低位

(除基取余法)

小数转换规则：乘2取整，至小数部分为0或取近似值

(乘基取整法)

填空题　30分

将下列十进制数转换成二进制数

1、$(327)_{10}$ = ([填空1])$_2$

2、$(0.8125)_{10}$ =([填空2])$_2$

3、$(0.2)_{10}$=([填空3])$_2$

提问：根据第3题的练习，你有什么结论？

4. 进制转换

(3) 二 ⟶ 八 ⟶十六

二 ⟶ 八：（整数）从低位开始，每三位一 组，　最高位不足三位，左边补0；

（小数）从高位开始，每三位一组，最低位不足，右边补0。

二进制小数：　11101101.0101101

八进制分组：　011，101，101，010，110，100

八进制数为：　3　5　5. 2　6　4

十六进制分组为：　1110，1101，0101，1010

十六进制数为：　E　D.　5　A

二、 符号数的表示格式与编码

复习概念

机器数：数在计算机中的二进制表示形式

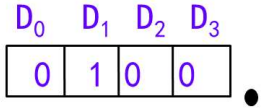
真值：机器数的形式值不等于所代表的数的真正的数值.

12

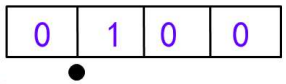
1、数的表示格式

（1）定点数及其表示

定点数：数据格式中小数点的位置固定不变.
计算机中的定点数只采用纯整数或者纯小数表示



0100　整数4

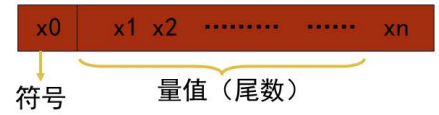


0100　小数0.5

（1）定点数的表示

对定点数：　$X = X_0 X_1 \cdots\cdots X_n$

在定点机中：



故：$|x| = \begin{cases} [0, 1-2^{-n}]，x为纯小数 \\ [0, 2^n-1]，x为纯整数 \end{cases}$

□ Signed integer,定点整数

—— 计算机必须能处理正数(positive) 和负数(negative)，MSB表示数符。

三种定点编码方式

- Signed magnitude （原码）

  现用来表示浮点（实）数的尾数
- One's complement （反码）

  现已不用于表示数值数据
- Two's complement （补码）

1950s后，整数都用补码来表示；
但浮点数的尾数用原码定点小数表示。

✓Sign and Magnitude （原码的表示）

1. 数学定义

假设用 $X_0X_1X_2\cdots\cdots X_n$ 来表示一个定点数

定点小数：$[x]_{原} = \begin{cases} x，1>x>=0 \\ 1-x = 1-|x|，0>=x>-1 \end{cases}$

定点整数：$[x]_{原} = \begin{cases} x，2^n > x >= 0 \\ 2^n-x = 2^n+|x|，0 >= x > -2^n \end{cases}$

✓原码表示法

2. 特殊数的表示　[+0] 原 = 0.00…0

[−0] 原 = 1.00…0

3. 表示规律　*0正1负，数码位不变。*

4. 表示范围

原码表示数的范围（以n=5为例）

二进制真值		原码
零	+0.0000 -0.0000	0.0000 1.0000
正数	+0.0001 …… +0.1111	0.0001 …… 0.1111
负数	-0.0001 …… -0.1111	1.0001 …… 1.1111

∴ 正数有 $2^{n-1}-1$个

负数有 $2^{n-1}-1$个

✓原码表示法

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

- 容易理解，　但是：
  - ✓ 0 的表示不唯一，故不利于程序员编程
  - ✓ 加、减运算方式不统一
  - ✓ 需额外对符号位进行处理，故不利于硬件设计
  - ✓ 特别当 a<b时，实现 a−b比较困难

✓补码表示法

1. 数学定义

设小数：X0.X1X2…Xn

定点小数：

$$[x]补 = \begin{cases} x & ,1 > x \geq 0 \\ 2+x = 2-|x| & ,0 \geq x \geq -1 \end{cases} \quad \text{Mod } 2$$

定点整数：

$$[x]补 = \begin{cases} x\ , & 2^n > x \geq 0 \\ 2^{n+1} + x = 2^{n+1} - |x|\ , & 0 \geq x \geq -2^n \end{cases}$$

✓补码表示法

2. 特殊数的表示

0的补码形式唯一：0.00…0　(mod 2)

3. 表示规律　*0正1负，从右至左，见1后反*

4. 表示范围

补码表示数的范围（以n=5为例）：

二进制真值		补码
零	±0.0000	0.0000
正数	+0.0001 …… +0.1111	0.0001 …… 0.1111
负数	-0.0001 …… -0.1111 -1.0000	1.1111 …… 1.0001 1.0000

∴ 正数有　$2^{n-1}-1$个

负数有　2^{n-1} 个

课堂练习

1. 设机器数有8位，求123和-123的补码表示。

解: 123 = 127- 4 = 01111111B - 100B = 01111011B

-123= - 01111011B

$[01111011]_{补}= 2^8 + 01111011 = 100000000 + 01111011$

$= 01111011 \pmod{2^8}$，即 7BH。

$[-01111011]_{补}= 2^8 – 01111011 = 10000\ 0000 - 01111011$

= 1111 1111 – 0111 1011 +1

= 1000 0100 +1 ⟵ 各位取反，末位加1

= 1000 0101，即 85H。

特殊数的补码

假定机器数有n位

① $[-2^{n-1}]_{补}= 2^n - 2^{n-1} = 10…0$（n-1个0） $(\mod 2^n)$

② $[-1]_{补}= 2^n - 0…01 = 11…1$（n个1） $(\mod 2^n)$

③ $[-1.0]_{补}= 2 - 1.0 = 1.00…0$（n-1个0） $(\mod 2)$

④ $[+0]_{补}= [-0]_{补}= 00…0$（n个0）

·如何理解补码? –模运算（modular运算）

- 计算机中硬件（如运算器、寄存器）能表示的数据位数是有限的，所以其运算都是有模运算，当运算结果超过最大表示范围（也就是模）时，就会溢出，并自动舍弃溢出量。

·如何理解补码? – 补数

对于两个整数a, b. 如果用某个正整数k去除a, b, 所得的余数相同, 则称a, b对于模k来说是同余数, 也叫互补, 即a, b对模k互补时, a, b在模k的意义下是相等的. 记作:

$a = b \pmod{k}$

如: $13=25 \pmod{12}$

推广：

$a + k = a \pmod{k}$

$a +2k = a \pmod{k}$

……

$a +nk = a \pmod{k}$

当a为负数时?

当a为负数时

如: a = −5, k = 12时, 则有

−5 + 12 = −5 (mod 12)

即: 7 = −5 (mod 12)

记作: [−5]补=7

说明:

在模12的意义下, −5相当于+7, 这样就将正负数之间的相加转化为正数的相加.

小结:

补码正是利用补数概念，把负数映射到正数域中（平移模值，小数的模为2，n位整数的模为2^n），从而将数的正负符号数码化，将减法运算转换为加法运算。

运算器是一个模运算系统，适合用补码表示和运算。

✓ 变形补码/模4补码

- 双符号位表示
- 左符号是真正的符号位，右符号可用来判断“溢出”

Decimal	补码	变形补码	Decimal	Bitwise Inverse	补码	变形补码
0	0000	00000	-0	1111	0000	00000
1	0001	00001	-1	1110	1111	11111
2	0010	00010	-2	1101	1110	11110
3	0011	00011	-3	1100	1101	11101
4	0100	00100	-4	1011	1100	11100
5	0101	00101	-5	1010	1011	11011
6	0110	00110	-6	1001	1010	11010
7	0111	00111	-7	1000	1001	11001
8	1000	01000	-8	0111	1000	11000

+0和−0表示唯一

值太大，用4位补码无法表示，故“溢出”！但用变形补码可保留符号位和最高数值位。

✓反码表示法

定义略。物理实现：触发器

▲：0的反码：

[+0] 反 = 0, 00…0

[−0] 反 =1, 11…0

[x]补=[x]反+2^{-n} 通过反码求补码

0正1负，数码位取反

✓Excess (biased) notion-移码表示

- 什么是“excess (biased) notation-移码表示”？

  将每一个数值加上一个偏置常数（ Excess / bias）

- 一般来说，当编码位数为n时，bias取 $2^{n-1}-1$

Ex. n=4: $E_{biased} = E + 2^3$ （bias= 2^3 =1000B）

⇒ −8 (+8) ~ 0000B
−7 (+8) ~ 0001B
…
0 (+8) ~ 1000B
…
+7 (+8) ~ 1111B

移码主要用来表示浮点数阶码！

0 负1正，数码位同补码

✓移码表示法

应用：表示浮点数的阶码：

如： x = + 10101 [x]移 = 1,10101
x = - 10101 [x]移 = 0,01011

- 为什么要用移码来表示指数（阶码）？

  便于浮点数加减运算时的对阶操作（比较大小）

例：$1.01 \times 2^{-1} + 1.11 \times 2^{3}$
补码：->111 < 011 ?
(-1) (3)

简化比较

$1.01 \times 2^{-1+4} + 1.11 \times 2^{3+4}$
移码：011 < 111
(3) (7)

（2） Unsigned integer 无符号整数

- 在不出现负值的场合下，可使用无符号数表示。
  如：地址，编号。
- 编码中没有符号位，能表示的最大值大于位数相同的带符号整数

例如， 8位带符号整数最大为127（0111 1111）
8位无符号整数最大是255（1111 1111）
总是整数，所以经常简称为“无符号数”

0000 0000 0000 0000 0000 0000 0000 1011

MSB
(Most Significant Bit)
最高有效位

LSB
(Least Significant Bit)
最低有效位

符号数 vs. 无符号数

- 扩充操作有差别

  如，MIPS提供了两种加载指令
  无符号数：lbu $t0, 0($s0) ; $t0高24位补0 （0扩展）
  带符号整数： lb $t0, 0($s0) ; $t0高24位补符 （符号扩展）

- 数的比较有差异

  无符号数：MSB为1的数比MSB为0的数大
  带符号整数： MSB为1的数比MSB为0的数小

- 溢出判断有差异

扩展操作举例

例1（扩展操作）：在32位机器上输出si, usi, i, ui的十进制（真值）和十六进制值（机器数）是什么？

```
short  si = -32768;
unsigned short  usi = si;
int  i = si;
unsingned  ui = usi ;
```

提示：
$32768=2^{15}$
=1000 0000 0000 0000B

	真值	机器数
si =	-32768	80 00
usi =	32768	80 00
i =	-32768	FF FF 80 00
ui =	32768	00 00 80 00

现象：
带符号整数：符号扩展
无符号数：0扩展

考虑C代码

```
1. int x=-1;
2. unsigned u=2147483648
3. printf("x=%u=%d"x,x);
4. printf("u=%u=%d"u,u);
```

（32位机）输出：

x=4294967295=-1

u=2147483648=-2147483648

小组讨论2：程序分析

```
int sum ( int a[], unsigned len )
{
        int i, sum=0;
        for ( i=0; i<= len-1; i++)
            sum +=a[i];
        return sum;
}
```

```
int sum ( int a[], int len )
{
        int i, sum=0;
        for ( i=0; i<= len-1; i++)
            sum +=a[i];
        return sum;
}
```

当参数len=0时，返回值应为0，
但在机器上执行时，却发生访存异常。为什么？

C语言程序中的整数

无符号数：unsigned int ； 带符号整数： int

同时有无符号和带符号整数：带符号数强制转换为无符号数

以下表达式在32位用补码表示的机器上执行，结果是什么？

关系表达式	类型	结果	说明
0 == 0U	无	1	00…0B = 00…0B
-1 < 0	带	1	11…1B (-1) < 00…0B (0)
-1 < 0U	无	0*	11…1B (2^{32}-1) > 00…0B(0)
2147483647 > -2147483647 - 1	带	1	011…1B (2^{31}-1) > 100…0B (-2^{31})
2147483647U > -2147483647 - 1	无	0*	011…1B (2^{31}-1) < 100…0B(2^{31})
2147483647 > (int) 2147483648U	带	1*	011…1B (2^{31}-1) > 100…0B (-2^{31})
-1 > -2	带	1	11…1B (-1) > 11…10B (-2)
(unsigned) -1 > -2	无	1	11…1B (2^{32}-1) > 11…10B (2^{32}-2)

带*的结果与常规预想的相反！

✓移位操作

- 对于带符号的定点数应采用算术移位方式，即只对数值部分移位而符号位不动。
- 在计算机内部移位操作在移位器中进行，移位器位数固定，所以移位前后数的位数不变。
  - 右移时低位要移出，高位补足相应的位数，低位移出时可能会使有效数字丢失所以要考虑相应的舍入方式
  - 左移时高位要移出，低位补足相应的位数，左移后数值扩大到原数的若干倍，因此有可能会发生溢出

✓移位操作（略）

各种编码的数值部分的移位规则如下：

① 原码

左移：高位移出，末位补0。移出非零时，发生溢出。

右移：高位补0，低位移出。移出时进行舍入操作。

② 补码

左移：高位移出，末位补0。移出非符时，发生溢出。

右移：高位补符，低位移出。移出时进行舍入操作。

③ 反码

左移：高位移出，末位补符。移出非符时，发生溢出。

右移：高位补符，低位移出。移出时进行舍入操作。

✓移位操作（略）

6）填充处理

在计算机内部，有时需要将一个取来的短数扩展为一个长数，此时要进行填充处理。

对于定点小数，在低位进行。

对于定点整数，在符号位后的数值高位进行。

① 原码

定点小数：在原数的末位后面补足0。

定点整数：符号位不变，在原数的符号位后补足0。

② 补码

定点小数：在原数的末位后面补足0。

定点整数：符号位不变，在原数的符号位后用数符补足所需的位数。

小结

- 定点数的表示
  - 进位计数制
  - 定点数的二进制编码
    - 原码、反码、补码、移码表示
  - 定点整数的表示
    - 无符号整数、带符号整数