计算机组成原理

第 2 讲

左德承

哈尔滨工业大学计算学部

容错与移动计算研究中心

1.3 计算机系统的主要技术指标

性能评价

Concluding Remarks

1.3

- Cost/performance is improving
  - Due to underlying technology development
- Hierarchical layers of abstraction
  - In both hardware and software
- Instruction set architecture
  - The hardware/software interface
- Execution time: the best performance measure
- Power is a limiting factor
  - Use parallelism to improve performance

1.3 计算机硬件的主要技术指标

◆ 计算机系统结构设计的几点思考

- 整机概念
  - 木桶原理
  - **Amdahl**定律
  - 挖掘系统并行性
- 平衡与折中
  - 最优？
  - 性能、可靠性、功耗、成本
  - 专用还是通用
  - 超算、高端容错计算机、穿戴计算机

Amdahl定律

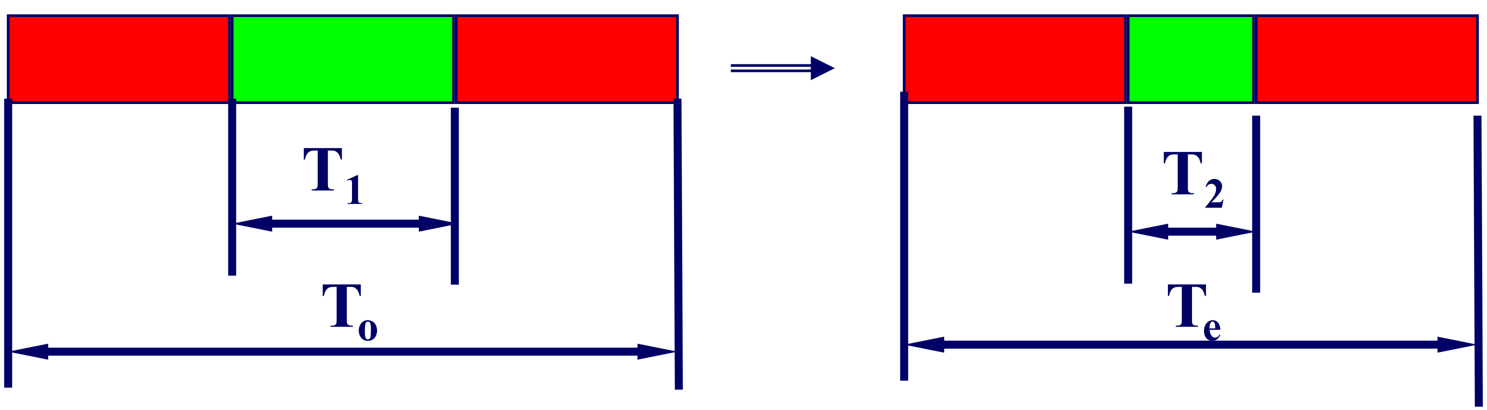
- 系统性能加速比，受限于该部件在系统中所占的重要性
- 假设我们对机器（部件）进行某种改进，那么机器系统（部件）的加速比就是

$$\text{系统加速比} = \frac{\text{系统性能}_{\text{改进后}}}{\text{系统性能}_{\text{改进前}}} = \frac{\text{总执行时间}_{\text{改进前}}}{\text{总执行时间}_{\text{改进后}}}$$

- 核心概念：时间
- 系统加速比告诉我们改进后的机器比改进前快多少

Amdahl定律

- 系统加速比依赖于两个因素
  - “可改进比例”：可改进部分在原系统计算时间中所占的比例 ，它总是小于等于1的
    - T_1/T_0
  - “部件加速比”可改进部分改进以后的性能提高，一般情况下它是大于1的
    - T_1/T_2



Amdahl的系统执行时间

- 部件改进后，系统的总执行时间等于不可改进部分的执行时间加上可改进部分改进后的执行时间，即：

- $\text{总执行时间}_{\text{改进后}}$

$$= (1-\text{可改进比例}) \times \text{总执行时间}_{\text{改进前}} + \frac{\text{可改进比例} \times \text{总执行时间}_{\text{改进前}}}{\text{部件加速比}}$$

$$= \text{总执行时间}_{\text{改进前}} \times \left[(1-\text{可改进比例}) + \frac{\text{可改进比例}}{\text{部件加速比}} \right]$$

Amdahl定律的观点

- **性能增加的递减规则**
  - 仅仅对计算机中的一部分做性能改进，则改进越多，系统获得的效果越小
- **Amdahl定律的一个重要推论**
  - 针对整个任务的一部分进行优化，则最大加速比不大于 $\dfrac{1}{1-\text{可改进比例}}$
- **Amdahl定律衡量一个“好”的计算机系统**
  - 具有高性能价格比的计算机系统是一个带宽平衡的系统，而不是看它使用的某些部件的性能

1.3 计算机硬件的主要技术指标

1. 机器字长　CPU 一次能处理数据的位数

与 CPU 中的 寄存器位数 有关

2. 运算速度

- 主频
- 吉普森法 $T_M = \sum_{i=1}^{n} f_i t_i$
- MIPS　每秒执行百万条指令
- FLOPS　每秒浮点运算次数
- CPI　执行一条指令所需时钟周期数

CPU执行时间的计算

重要指标：CPI：Cycles Per Instruction

CPU时间＝IC×CPI× 时钟周期时间

假定CPI_i和 C_i分别为第i类指令的CPI和指令条数，则程序的总时钟数为：

总时钟数 $= \sum_{i=1}^{n} CPI_i \times C_i$ 所以， **CPU时间= 时钟周期** $\times \sum_{i=1}^{n} CPI_i \times C_i$

CPI **用来衡量以下各方面的综合结果**

Instruction Set Architecture（ISA）

Implementation of that architecture（Organization & Technology）

Program（Compiler、Algorithm）

问题：计算机性能与ISA、计算机组织（Organization）、计算机实现技术（Technology）三者的关系是什么？

CPU执行时间的计算

重要指标：CPI：Cycles Per Instruction

CPU时间＝IC×CPI× 时钟周期时间

	Instr.count (指令条数)	**CPI （每条指令平均时钟数）**	**Clock rate （时钟频率）**
Programming (程序）	√	√	
Compiler （编译器）	√	√	
Instr. Set Arch （指令系统）	√	√	
Organization （组成）		√	√
Technology （实现）			√

CPU性能公式举例 1.3

某程序的目标代码由4类指令组成，它们在程序中所占比例和各自CPI如下表：

表 1.4 各类指令所占的比例及 CPI

指令类型	CPI	所占比例
算术逻辑运算指令	1	60%
访存指令	2	18%
转移指令	4	12%
其他指令	8	10%

- 求该程序的CPI
- 若该CPU的主频为400MHz，求该机的MIPS

$$CPI = \sum_{i=1}^{n}(CPI_i \times P_i) = 1 \times 0.6 + 2 \times 0.18 + 4 \times 0.12 + 8 \times 0.1 = 2.24$$

$$MIPS = \frac{f}{CPI} = \frac{400}{2.24} = 178.6$$

CPU性能公式举例 1.3

- 若计算机A和B是基于相同指令集设计的两种不同的计算机，A的时钟周期为2ns，某程序在A上运行时的CPI为3。B的时钟周期为4ns，同一程序在B上运行时的CPI为2。就这个程序而言，计算机A和B哪个更快？

$$T_{CPU_A} = CPI_A \times IC \times T_A \quad T_{CPU_B} = CPI_B \times IC \times T_B$$

$$\frac{T_{CPU_A}}{T_{CPU_B}} = \frac{CPI_A \times IC \times T_A}{CPI_B \times IC \times T_B} = \frac{3 \times IC \times 2}{2 \times IC \times 4} = 0.75$$

加强：性能评价的认识和理解

- 时钟频率
- MIPS
- MFLOPS
- 单一程序的执行时间(SuperPI)
- 基准测试分数
  - SPEC CPU
  - Linpack
  - TPC
  - Winstone
  - 3DMark2001,3DMark2003
  - …

常用的度量方式

什么是好的度量方式

- 线性性(Linearity)
  - 符合人的直觉
  - 例子速度m/s
  - 相反的例子dB
- 可靠性
  - 当A的性能度量值好于B时,如果系统A总是比系统B快,则称该度量方式是可靠的
  - 相反的例子MIPS
- 可重复性
- 容易测量
- 独立性
  - 生产商试图影响度量方式

单一程序的运行时间

- 我们最终关心的是程序的执行时间
- 线性，容易测量，可重复，可靠（? ,但仅对此程序！),独立，因此时间是一个好的度量
- 程序的代表性问题
  - Superpi, 天气预报
  - 从这个度量我们可以得出什么结论?
  - 不能得出什么结论
  - 严格的可靠性度量是不存在的!

基准测试程序

- 优点
  - 一组有代表性的程序，比单个程序更能代表用户的负载状况
  - 测试程序分数是程序运行时间的函数(线性或非线性)
  - 以运行时间直接为度量，或是以单位时间内完成的操作为度量
- 缺点
  - 仍然不具有可靠性
  - 可能不独立
- SPEC
  - System Performance Evaluation Cooperative
  - SPEC CPU(fp, int), SPEC WEB, SPEC AppServer, SPEC HPC, SPEC OMP …

SPEC

- SPEC CPU的测量方法
  - 测量每个程序的运行时间
  - 正规化：除一个标准系统对该程序的运行时间，得到一个正规化后的值
  - 算出上述正规化值的几何平均作为SPEC分数
  - 非线性

几何平均？

一个例子

程序	S1	S2	S3
1	417	244	134
2	83	70	70
3	66	153	135
4	39, 449	33, 527	66, 000
5	772	368	369
几何平均	587	503	499
排名	3	2	1

程序执行时间及其几何平均值的排名

几何平均

程序	S1	S2	S3
1	417	244	134
2	83	70	70
3	66	153	135
4	39, 449	33, 527	66, 000
5	772	368	369
总时间	40, 787	34, 362	66, 798
算术平均	8, 157	6, 872	13, 342
排名	2	1	3

程序	S1	S2	S3
1	417	244	134
2	83	70	70
3	66	153	135
4	39, 449	33, 527	66, 000
5	772	368	369
几何平均	587	503	499
排名	3	2	1

几何平均：减少个别值对结果的影响

相对速度/排名

程序	S1	S2	S3
1	1.0	0.59	0.32
2	1.0	0.84	0.85
3	1.0	2.32	2.05
4	1.0	0.85	1.67
5	1.0	0.48	0.45
几何平均	1.0	0.86	0.84
排名	3	2	1

以S1为基准正规化及其几何平均值的排名

- 和前面的排名次序一致

相对速度/排名

	System 1	System 2	System 3
	1.71	1.0	0.55
	1.19	1.0	1.0
	0.43	1.0	0.88
	1.18	1.0	1.97
	2.10	1.0	1.0
几何平均	1.17	1.0	0.99
排名	3	2	1

以S2为基准正规化及其几何平均值的排名

- 选取不同的基准不影响系统排名

第 2 部分　计算机的运算方法

2.1 无符号数和有符号数

2.2 数的定点表示和浮点表示

2.3 定点运算

2.4 浮点四则运算

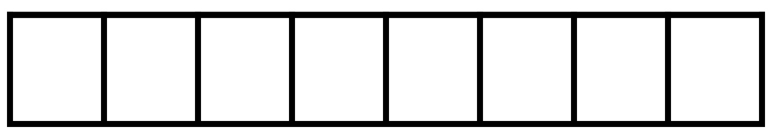
2.5 算术逻辑单元

2.1 无符号数和有符号数

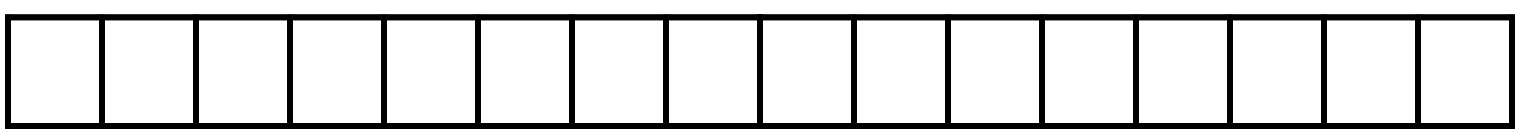
一、无符号数

寄存器的位数

反映无符号数的表示范围



8 位　　　　0 ~ 255



16 位　　　　0 ~ 65535

二、有符号数

1. 机器数与真值

真值	机器数	
带符号的数	符号数字化的数	
+ 0.1011	0 || 1011	小数点的位置
– 0.1011	1 || 1011	小数点的位置
+ 1100	0 || 1100	小数点的位置
– 1100	1 || 1100	小数点的位置

2. 原码表示法

(1) 定义

整数

$$[x]_{原} = \begin{cases} 0, \ x & 2^n > x \geq 0 \\ 2^n - x & 0 \geq x > -2^n \end{cases}$$

x 为真值　　n 为整数的位数

如　$x = +1110$　　$[x]_{原} = 0\,,1110$

用 逗号 将符号位和数值部分隔开

$x = -1110$　　$[x]_{原} = 2^4 + 1110 = 1\,,1110$

带符号的绝对值表示

小数

$$[x]_{原} = \begin{cases} x & 1 > x \geq 0 \\ 1 - x & 0 \geq x > -1 \end{cases}$$

x 为真值

如

$x = +0.1101 \qquad [x]_{原} = 0\,.\,1101$

用 小数点 将符号位和数值部分隔开

$x = -0.1101 \qquad [x]_{原} = 1 - (-0.1101) = 1\,.\,1101$

$x = +0.1000000 \qquad [x]_{原} = 0\,.\,1000000$

用 小数点 将符号位和数值部分隔开

$x = -0.1000000 \qquad [x]_{原} = 1 - (-0.1000000) = 1\,.\,1000000$

(2) 举例

例 6.1　已知 $[x]_{原} = 1.0011$　求 x　-0.0011

解：由定义得

$$x = 1 - [x]_{原} = 1 - 1.0011 = -0.0011$$

例 6.2　已知 $[x]_{原} = 1,1100$　求 x　-1100

解：由定义得

$$x = 2^4 - [x]_{原} = 10000 - 1,1100 = -1100$$

例 6.3　已知 $[x]_{原} = 0.1101$　求 x

解：根据 定义 $\because [x]_{原} = 0.1101$

$\therefore \quad x = +0.1101$

例 6.4　求 $x = 0$ 的原码

解：设 $x = +0.0000$　　$[+0.0000]_{原} = 0.0000$

$x = -0.0000$　　$[-0.0000]_{原} = 1.0000$

同理，对于整数　　$[+0]_{原} = 0,0000$

$[-0]_{原} = 1,0000$

$\therefore \quad [+0]_{原} \neq [-0]_{原}$

原码的特点：简单、直观

但是用原码作加法时，会出现如下问题：

要求	数1	数2	实际操作	结果符号
加法	正	正	加	正
加法	正	负	减	可正可负
加法	负	正	减	可正可负
加法	负	负	加	负

能否 只作加法？

找到一个与负数等价的正数 来代替这个负数

就可使 减 ⟶ 加

3. 补码表示法

(1) 补的概念

- 时钟

逆时针

$$\begin{array}{r} 6 \\ -\ 3 \\ \hline 3 \end{array}$$

顺时针

$$\begin{array}{r} 6 \\ +\ 9 \\ \hline 15 \\ -\ 12 \\ \hline 3 \end{array}$$

可见 -3 可用 $+9$ 代替 减法 ⟶ 加法

称 $+9$ 是 -3 以 12 为模的 补数

记作 $-3 \equiv +9 \pmod{12}$

同理 $-4 \equiv +8 \pmod{12}$

$-5 \equiv +7 \pmod{12}$

时钟以12为模

结论

- 一个负数加上 **“模”** 即得该负数的补数
- 一个正数和一个负数互为补数时
  它们绝对值之和即为 **模** 数
  - 计数器（模 16） $1011 \longrightarrow 0000$ ？

$$\begin{array}{r} 1011 \\ -\ 1011 \\ \hline 0000 \end{array} \qquad \begin{array}{r} 1011 \\ +\ 0101 \\ \hline \boxed{1}0000 \end{array}$$

自然去掉

可见 -1011 可用 $+0101$ 代替

记作 $-1011 \equiv +0101 \quad (\mathrm{mod}\ 2^4)$

同理 $-011 \equiv +101 \quad (\mathrm{mod}\ 2^3)$

$-0.1001 \equiv +1.0111 \quad (\mathrm{mod}\ 2)$

(2) 正数的补数即为其本身

两个互为补数的数 $\boxed{-1011 \quad \equiv \quad +0101}$ $(\bmod 2^4)$

分别加上模 $+10000$ $\qquad +10000$

结果仍互为补数 $+0101 \equiv +\boxed{1}0101$

丢掉

$\therefore +0101 \equiv +0101 \quad (\bmod 2^4)$

可见 $+0101 \longrightarrow +0101$

$\quad ? \longrightarrow -1011$

$? \; \boxed{0},0101 \longrightarrow +0101$

$? \; \boxed{1},0101 \longrightarrow -1011$

$2^{4+1} - 1011 = 100000$ $\qquad (\bmod 2^{4+1})$

$$\begin{array}{r} 100000 \\ -1011 \\ \hline 1,0101 \end{array}$$

用 逗号 将符号位和数值部分隔开

(3) 补码定义

整数

$$[x]_{补} = \begin{cases} 0, \ x & 2^n > x \geq 0 \\ 2^{n+1} + x & 0 > x \geq -2^n \ (\text{mod } 2^{n+1}) \end{cases}$$

x 为真值　　　　n 为整数的位数

如　　$x = +1010$　　　　$x = -1011000$

$[x]_{补} = 0,1010$　　　　$[x]_{补} = 2^{7+1} + (-1011000)$

$$\begin{array}{r} = 100000000 \\ - \quad 1011000 \\ \hline 1,0101000 \end{array}$$

用 逗号 将符号位和数值部分隔开

小数

$$[x]_{补} = \begin{cases} x & 1 > x \geq 0 \\ 2 + x & 0 > x \geq -1 \end{cases} \quad (\bmod 2)$$

x 为真值

如 $x = +0.1110$ $\qquad$ $x = -0.1100000$

$[x]_{补} = 0.1110$

$$\begin{aligned}[x]_{补} &= 2 + (-0.1100000) \\ &= 10.0000000 \\ &- \quad 0.1100000 \\ &\overline{\quad\ 1.0100000} \end{aligned}$$

用 小数点 将符号位

和数值部分隔开

(4) 求补码的快捷方式

设 $x = -1010$ 时

则 $[x]_{补} = 2^{4+1} - 1010 = 11111 + 1 - 1010$

$$\begin{array}{r} 100000 \\ -\ \ 1010 \\ \hline 1,0110 \end{array} \qquad \begin{array}{r} 11111 + 1 \\ -\ \ 1010 \quad\ \ \\ \hline \boxed{10101} + 1 \\ = 1,0110 \end{array}$$

又 $[x]_{原} = \boxed{1,1010}$

当真值为 **负** 时，**补码** 可用 **原码除符号位外每位取反，末位加 1 求得**

(5) 举例

例 6.5　已知 $[x]_{补} = 0.0001$

求 x

解：由定义得　$x = +0.0001$

例 6.6　已知 $[x]_{补} = 1.0001$

求 x

解：由定义得

$$x = [x]_{补} - 2$$
$$= 1.0001 - 10.0000$$
$$= -0.1111$$

$[x]_{补} \xrightarrow{?} [x]_{原}$

$[x]_{原} = 1.1111$

$\therefore x = -0.1111$

例 6.7 已知 $[x]_{补} = 1,1110$

求 x

解：由定义得

$$x = [x]_{补} - 2^{4+1}$$

$$= 1,1110 - 100000$$

$$= -0010$$

$$[x]_{补} \xrightarrow{?} [x]_{原}$$

$$[x]_{原} = 1,0010$$

$$\therefore x = -0010$$

当真值为 负 时，原码 可用 补码除符号位外

每位取反，末位加 1 求得

练习 求下列真值的补码 2.1

真值		$[x]_{补}$	$[x]_{原}$
$x = +70$	$= \ 1000110$	0, 1000110	0,1000110
$x = -70$	$= -1000110$	1, 0111010	1,1000110
$x = \ 0.1110$		0.1110	0.1110
$x = -0.1110$		1.0010	1.1110
$x = \boxed{0.0000}$	$[+0]_{补} = [-0]_{补}$	$\boxed{0.0000}$	0.0000
$x = \boxed{-0.0000}$		$\boxed{0.0000}$	1.0000
$x = -1.0000$		1.0000	不能表示

由小数补码定义

$$[x]_{补} = \begin{cases} x & 1 > x \geq 0 \\ 2 + x & 0 > x \geq -1 \ (\mathrm{mod}\ 2) \end{cases}$$

$$[-1]_{补} = 2 + x = 10.0000 - 1.0000 = 1.0000$$

4. 反码表示法

(1) 定义

整数

$$[x]_{反} = \begin{cases} 0，x & 2^n > x \geq 0 \\ (2^{n+1}-1)+x & 0 \geq x > -2^n \ (\bmod\ 2^{n+1}-1) \end{cases}$$

x 为真值　　　　n 为整数的位数

如　$x = +1101$　　　　$x = -1101$

$[x]_{反} = 0,1101$　　　　$[x]_{反} = (2^{4+1}-1)-1101$

$= 11111 - 1101$

$= 1,0010$

用 逗号 将符号位

和数值部分隔开

小数

$$[x]_{反} = \begin{cases} x & 1 > x \geq 0 \\ (2 - 2^{-n}) + x & 0 \geq x > -1 \ (\mathrm{mod}\ 2 - 2^{-n}) \end{cases}$$

x 为真值　　n 为小数的位数

如

$x = +0.1101$　　　　$x = -0.1010$

$[x]_{反} = 0.1101$　　　　$[x]_{反} = (2 - 2^{-4}) - 0.1010$

$= 1.1111 - 0.1010$

$= 1.0101$

用 小数点 将符号位

和数值部分隔开

(2) 举例

例6.8　已知 $[x]_{反} = 0,1110$　求 x

解：　由定义得　$x = +1110$

例6.9　已知 $[x]_{反} = 1,1110$　求 x

解：　由定义得　$x = [x]_{反} - (2^{4+1} - 1)$

$= 1,1110 - 11111$

$= -0001$

例 6.10　求 0 的反码

解：　设 $x = +0.0000$　　$[+0.0000]_{反} = 0.0000$

$x = -0.0000$　　$[-0.0000]_{反} = 1.1111$

同理，对于整数　$[+0]_{反} = 0,0000$　$[-0]_{反} = 1,1111$

$\therefore\ [+0]_{反} \neq [-0]_{反}$

三种机器数的小结

- 最高位为符号位，书写上用“,”（整数）或“.”（小数）将数值部分和符号位隔开
- 对于正数，原码 = 补码 = 反码
- 对于负数，符号位为 1，其数值部分
  原码除符号位外每位取反末位加 1 $\longrightarrow$ 补码
  原码除符号位外每位取反 $\longrightarrow$ 反码

例6.11 设机器数字长为 8 位（其中 1 位为符号位）对于整数，当其分别代表无符号数、原码、补码和反码时，对应的真值范围各为多少？

二进制代码	无符号数对应的真值	原码对应的真值	补码对应的真值	反码对应的真值
00000000	0	+0	±0	+0
00000001	1	+1	+1	+1
00000010	2	+2	+2	+2
⋮	⋮	⋮	⋮	⋮
01111111	127	+127	+127	+127
10000000	128	-0	-128	-127
10000001	129	-1	-127	-126
⋮	⋮	⋮	⋮	⋮
11111101	253	-125	-3	-2
11111110	254	-126	-2	-1
11111111	255	-127	-1	-0

例6.12　已知 $[y]_{补}$　　求 $[-y]_{补}$

解：设 $[y]_{补} = y_0 . y_1 y_2 \cdots y_n$

<I>　$\boxed{[y]_{补} = 0. y_1 y_2 \cdots y_n}$

$[y]_{补}$连同符号位在内，　每位取反，　末位加 1

即得$[-y]_{补}$

$$\boxed{[-y]_{补} = 1.\overline{y_1}\,\overline{y_2} \cdots \overline{y_n} + 2^{-n}}$$

<II>　$\boxed{[y]_{补} = 1. y_1 y_2 \cdots y_n}$

$[y]_{补}$连同符号位在内，　每位取反，　末位加 1

即得$[-y]_{补}$

$$\boxed{[-y]_{补} = 0.\overline{y_1}\,\overline{y_2} \cdots \overline{y_n} + 2^{-n}}$$

5. 移码表示法

补码表示很难直接判断其真值大小

如 十进制	二进制	补码	
$x = +21$	$+10101$	$0,10101$	错
$x = -21$	-10101	$1,01011$	大
$x = +31$	$+11111$	$0,11111$	错
$x = -31$	-11111	$1,00001$	大

$x + 2^5$

$+10101 + 100000 = 110101$ 大 正确

$-10101 + 100000 = 001011$

$+11111 + 100000 = 111111$ 大 正确

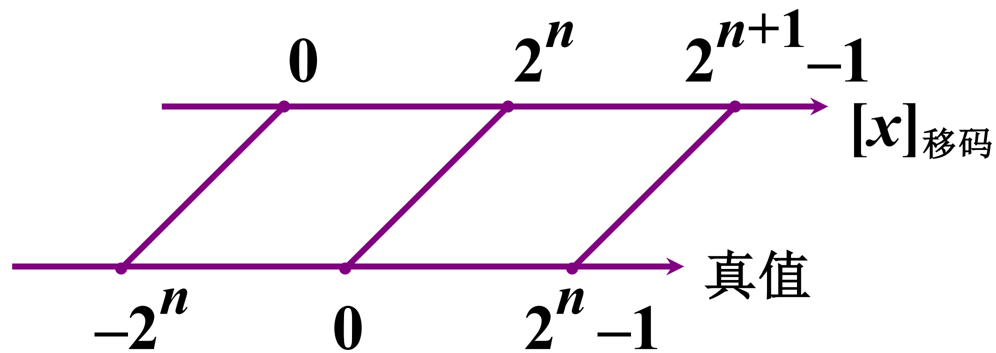
$-11111 + 100000 = 000001$

(1) 移码定义

$$[x]_{移} = 2^n + x \quad (2^n > x \geq -2^n)$$

x 为真值，n 为 整数的位数

移码在数轴上的表示



如 $\quad x = 10100$

$$[x]_{移} = 2^5 + 10100 = 1,10100$$

$$x = -10100$$

$$[x]_{移} = 2^5 - 10100 = 0,01100$$

用 逗号 将符号位和数值部分隔开

(2) 移码和补码的比较

设 $x = +1100100$

$[x]_{移} = 2^7 + 1100100 = 1,1100100$

$[x]_{补} = 0,1100100$

设 $x = -1100100$

$[x]_{移} = 2^7 - 1100100 = 0,0011100$

$[x]_{补} = 1,0011100$

补码与移码只差一个符号位

(3) 真值、补码和移码的对照表

真值 x (n=5)	$[x]_{补}$	$[x]_{移}$	$[x]_{移}$对应的十进制整数
- 1 0 0 0 0 0	1 0 0 0 0 0	0 0 0 0 0 0	0
- 1 1 1 1 1	1 0 0 0 0 1	0 0 0 0 0 1	1
- 1 1 1 1 0	1 0 0 0 1 0	0 0 0 0 1 0	2
⋮	⋮	⋮	⋮
- 0 0 0 0 1	1 1 1 1 1 1	0 1 1 1 1 1	31
± 0 0 0 0 0	0 0 0 0 0 0	1 0 0 0 0 0	32
+ 0 0 0 0 1	0 0 0 0 0 1	1 0 0 0 0 1	33
+ 0 0 0 1 0	0 0 0 0 1 0	1 0 0 0 1 0	34
⋮	⋮	⋮	⋮
+ 1 1 1 1 0	0 1 1 1 1 0	1 1 1 1 1 0	62
+ 1 1 1 1 1	0 1 1 1 1 1	1 1 1 1 1 1	63

(4) 移码的特点

- 当 $x = 0$ 时 $[+0]_{移} = 2^5 + 0 = 1,00000$

  $[-0]_{移} = 2^5 - 0 = 1,00000$

  $\therefore [+0]_{移} = [-0]_{移}$

- 当 $n = 5$ 时 最小的真值为 $-2^5 = -100000$

  $[-100000]_{移} = 2^5 - 100000 = 000000$

  可见，最小真值的移码为全 0

  用移码表示浮点数的阶码

  能方便地判断浮点数的阶码大小