**2.10**

|  |  |  |
| --- | --- | --- |
| 步骤 | \*x | \*y |
| 初始 | a | b |
| 第一步 | a | a^b |
| 第二步 | a^a^b = 0^b = b | a^b |
| 第三步 | b | a^b^b = a^0 = a |

**2.11**

**A.**

first = last = k.

**B.**

因为first和last指向同一个元素, 在执行第一步操作后, 会将\*x和\*y的值都置为0, 而后续操作无法弥补这个错误, 所以导致最后结果为0.

**C.**

将第4行改为first < last;

**2.14**

|  |  |  |  |
| --- | --- | --- | --- |
| 表达式 | 值 | 表达式 | 值 |
| x & y | 0x20 | x && y | 0x01 |
| x | y | 0x7F | x || y | 0x01 |
| ~x | ~y | 0xDF | !x || !y | 0x00 |
| x & !y | 0x00 | x && ~y | 0x01 |

**2.23**

**A.**

|  |  |  |
| --- | --- | --- |
| w | fun1(w) | fun2(w) |
| 0x00000076 | 0x00000076 | 0x00000076 |
| 0x87654321 | 0x00000021 | 0x00000021 |
| 0x000000C9 | 0x000000C9 | 0xFFFFFFC9 |
| 0xEDCBA987 | 0x00000087 | 0xFFFFFF87 |

**B.**

fun1直接提取参数二进制形式的低8位, 得到[0, 255]之间的一个整数.

fun2不仅提取参数二进制形式的低8位, 而且进行了符号拓展, 得到[-128, 127]之间的一个整数.

**2.35**

当x = 0时, 一定有x \* y = 0, 不会有乘法溢出.

当x ≠ 0 时, 证明如下:

**1.**

因为x, y都是w位的二进制向量所表示的数字, 所以x和y的整数乘积x· y一定可以用2w位补码数字来表示. 令m为高w位的补码数字, n为低w位的无符号数. 则有:

x·y = n + m2w (1)

且已知p为x和y补码乘法的结果, 由于补码乘法直接截断整数乘法结果的低w位作为结果, 所以有:

T2Uw(p) = n (2)

其中, T2Uw函数将一个w位二进制补码表示的数字映射到同样的二进制串表示的无符号数.

(2)表明, p和n有同样的二进制串, 只不过分别按照有符号数和无符号数理解.

下面不加证明地引入一个Lemma:

Lemma 1:

T2Uw(x) = xw-12w + x

其中x为一个有符号数, 其二进制形式为[xw-1xw-2…x1x0], T2Uw函数定义同上.

将Lemma 1应用于(2), 有:

T2Uw(p) = n = pw-12w + p (3)

将(3)代入(1), 有:

x·y = n + m2w = pw-12w + p + m2w = p + (pw-1 + m)2w (4)

在(4)中, 令t = pw-1 + m, 有:

x·y = p + t2w (5)

即为所求.

易知x·y ≠ p iff 乘法溢出, 结合(5)易得:

t ≠ 0 iff乘法溢出

**2.**

由整数除法的性质, 易得:

p = xq + r (6)

其中, q为p除以x的商, r为余数, 且|r| < |x|.

**3.**

将(6)代入(5), 有:

x·y = xq + r + t2w (7)

当q = y 时, 有:

x·y = x·y + r + t2w (8)

由(8), 有:

r + t2w = 0 (9)

于是有 r = - t2w成立(r ≠ 0且t ≠ 0)或r = t = 0成立.

根据2. , 有|r| < |x|成立, 且|x| ≤ 2w-1成立.

若r = - t2w成立, 则|r| < 2w-1, 则|- t2w| < 2w-1, 则|t| < 1/2.

由t = pw-1 + m, 于是解得t = 0与t ≠ 0矛盾.

所以当q = y 时, 一定有r = t = 0成立.

当r = t = 0时, 由(7)易得q = y.

综上, q = y iff r = t = 0.

结合以上三条, 可以证明该方法是正确的.

**2.47**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 位 | e | E | 2E | f | M | 2E×M | V | 十进制 |
| 0 00 00 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 00 01 | 0 | 0 | 1 | 1 / 4 | 1 / 4 | 1 / 4 | 1 / 4 | 0.25 |
| 0 00 10 | 0 | 0 | 1 | 1 / 2 | 1 / 2 | 1 / 2 | 1 / 2 | 0.5 |
| 0 00 11 | 0 | 0 | 1 | 3 / 4 | 3 / 4 | 3 / 4 | 3 / 4 | 0.75 |
| 0 01 00 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 |
| 0 01 01 | 1 | 0 | 1 | 1 / 4 | 5 / 4 | 5 / 4 | 5 / 4 | 1.25 |
| 0 01 10 | 1 | 0 | 1 | 1 / 2 | 3 / 2 | 3 / 2 | 3 / 2 | 1.5 |
| 0 01 11 | 1 | 0 | 1 | 3 / 4 | 7 / 4 | 7 / 4 | 7 / 4 | 1.75 |
| 0 10 00 | 2 | 1 | 2 | 0 | 1 | 2 | 2 | 2 |
| 0 10 01 | 2 | 1 | 2 | 1 / 4 | 5 / 4 | 5 / 2 | 5 / 2 | 2.5 |
| 0 10 10 | 2 | 1 | 2 | 1 / 2 | 3 / 2 | 3 | 3 | 3 |
| 0 10 11 | 2 | 1 | 2 | 3 / 4 | 7 / 4 | 7 / 2 | 7 / 2 | 3.5 |
| 0 11 00 | -------- | -------- | -------- | -------- | -------- | -------- |  | -------- |
| 0 11 01 | -------- | -------- | -------- | -------- | -------- | -------- | NaN | -------- |
| 0 11 10 | -------- | -------- | -------- | -------- | -------- | -------- | NaN | -------- |
| 0 11 11 | -------- | -------- | -------- | -------- | -------- | -------- | NaN | -------- |

**2.52**

|  |  |  |  |
| --- | --- | --- | --- |
| 格式A | | 格式B | |
| 位 | 值 | 位 | 值 |
| 011 0000 | 1 | 0111 000 | 1 |
| 101 1110 | 15 / 2 | 1001 111 | 15 / 2 |
| 010 1001 | 25 / 32 | 0110 100 | 3 / 4 |
| 110 1111 | 31 / 2 | 1011 000 | 16 |
| 000 0001 | 1 / 64 | 0001 000 | 1 / 64 |

**2.63**

int sra(int x, int k) // do sra with the result of srl

{

int xsrl = (unsigned) x >> k;

int w = sizeof(int) << 3;

if (x > 0) // the symbol bit is 0

return xsrl;

else

return ((2 << (w - k - 1)) - 1) ^ (~xsrl);

}

unsigned srl(unsigned x, int k) // do srl with the result of sra

{

unsigned xsra = (int) x >> k;

int w = sizeof(int) << 3;

return ((2 << (w - k - 1)) - 1) & xsra;

}

**2.75**

unsigned unsigned\_high\_prod(unsigned x, unsigned y)

{

int w = sizeof(unsigned) \* 8;

return signed\_high\_prod(X, y) + (x >> (w - 1)) \* y + (y >> (w - 1)) \* x;

}

**2.81**

**A.**

非永真. 当x = 0, y = INT\_MIN时, 有 –x = 0, -y = INT\_MIN, 此时有 x > y且-x > -y.

**B.**

永真. 补码的加减乘(包括左移)和顺序无关.

**C.**

非永真. 当x = -1, y = 1时, ~x + ~y = 0xFFFFFFFE, 而~(x + y) = 0xFFFFFFFF.

**D.**

永真. 有符号数和无符号数有相同的二进制串.

**E.**

永真. 先右移一位再左移一位, 相当于令x的二进制串的最低位为0. 若x为偶数, 操作后x不变; 若x为奇数, 则操作后x减少1. 又因为INT\_MIN是偶数, 因此不存在溢出情况, 所以左边总是 <= x.

**2.86**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 描述 | Hex | M | E | V |
| -0 | 0x8000 | 0 | -62 | ----------------- |
| 最小的值>1 | 0x3F01 | 257/256 | 0 | 257 x 2-8 |
| 256 | 0x4700 | 1 | 8 | ----------------- |
| 最大的非规格化数 | 0x00FF | 255/256 | -62 | 255 x 2-70 |
|  |  | ----------------- | ----------------- | ----------------- |
| 十六进制表示为3AA0的数 | ----------------- | 13/8 | -5 | 13 x 2-8 |

**2.90**

**A.**

11.00 1001 0000 1111 1101 1011

**B.**

11.001001[001]...

**C.**

第九位.