3

(1) 11001.1101 = 31.64 = 19.d

(2) 45.375 = 55.3 = 2d.6 = 0100 0101. 0011 0111 0101

(3)

(4) 78.75 = 100 1110. 11

9

(1) 先得出32768的机器数为0x00008000然后逐位取反, 末位加一, 得到-32768的机器数为0xFFFF8000.

(2) 522 = 512 + 10, 所以机器数为0x020A.

(3) 65530 = 15 x (163 + 162 + 16) + 10, 所以机器数为0x0000FFFA

(4) @符号对应的ascii码是0100 0000, 故机器数为0x40.

(5) a是负数, 符号位为1, 将-a变为二进制小数为, 指数部分为0, 加上偏置常数为127, 最后一位舍入后有进位, 于是得到a的机器数为0xBF8CCCCD.

(6) b是正数, 符号位为0, 将其变为二进制小数为1.0101 x 23, 指数部分为3, 加上偏置常数为1026, 于是b的机器数是0x4025 0000 0000 0000.

10

(1) x的符号位为1, 说明表示负数, 逐位取反, 末位加一后得到-x的机器数0x0000FFFA, 所以x的真值为-65530.

(2) y的符号位为1, 说明表示负数, 逐位取反, 末位加一后得到-y的机器数0x2004, 所以y的真值为-8196.

(3) 无符号数, 直接得到真值为232 – 6 = 4 294 967 290.

(4) 2AH = 0010 1010B, 对应的字符为\*, 所以真值为\*.

(5) 将机器数写成二进制形式, 符号位为1, 说明是负数, 指数部分减去偏置常数为9, 尾数部分加上隐藏为1为1.5625, 所以真值为-1.5625 x 29 = -800.

(6) 将机器数写成二进制形式, 符号位为1, 说明是负数, 指数部分减去偏置常数为3, 尾数部分加上隐藏为1为1.01001, 所以真值为-1.01001 x 23 = -10.25.

15

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| x | y | x^y | x&y | x|y | ~x|~y | x&!y | x&&y | x||y | !x||!y | x&&~y |
| 0x5F | 0xA0 | 0xFF | 0x00 | 0xFF | 0xFF | 0x00 | 0x01 | 0x01 | 0x00 | 0x01 |
| 0xC7 | 0xF0 | 0x37 | 0xC0 | 0xF7 | 0x3F | 0x00 | 0x01 | 0x01 | 0x00 | 0x01 |
| 0x80 | 0x7F | 0xFF | 0x00 | 0xFF | 0xFF | 0x00 | 0x01 | 0x01 | 0x00 | 0x01 |
| 0x07 | 0x55 | 0x52 | 0x05 | 0x57 | 0xFA | 0x00 | 0x01 | 0x01 | 0x00 | 0x01 |

21

根据optarith里的代码, 可以看出程序先用一个临时变量t存了x的值, 然后将x左移四位, 然后再用x减去t, 相当于16x – x = 15x, 故M为15. 对于y, 在y为负数的时候加了一个修正值, 从修正值为3 = 22 – 1便可得出N应该等于4. 再看后一步处理过的y被右移了2位, 验证了N等于4的事实. 综上, M为15, N为4.

29

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 表示 | X | x | Y | y | X + Y | x + y | OF | SF | CF | X - Y | x - y | OF | SF | CF |
| 无符号 | 0xB0 | 176 | 0x8C | 140 | 0x3C | 60 | 1 | 0 | 1 | 0x24 | 36 | 0 | 0 | 0 |
| 带符号 | 0xB0 | -80 | 0x8C | -116 | 0x3C | 60 | 1 | 0 | 1 | 0x24 | 36 | 0 | 0 | 0 |
| 无符号 | 0x7E | 126 | 0x5D | 93 | 0xDB | 219 | 1 | 1 | 0 | 0x21 | 33 | 0 | 0 | 0 |
| 带符号 | 0x7E | 126 | 0x5D | 93 | 0xDB | -37 | 1 | 1 | 0 | 0x21 | 33 | 0 | 0 | 0 |

计算机底层并不区分有符号和无符号数, 所以运算后的机器数都是一样的, 标志位设置也是一样的. 对于第一组数据, 176+140 > 256, 所以CF = 1, 同时由于字长为8位的限制, 高于8位的结果被舍去, 所以结果机器数为0x3C. 这时结果的符号位和操作数的符号位不同且两个操作数的符号位相同, 所以OF = 1, 易得SF = 0. 对于减法而言, 计算机底层使用补码加法来代替, 转换后原理同加法. 第二组数据和第一组类似.

33

int div32(int x)

{

int b = (x >> 31) & 0x1F;

return (x + b) >> 5;

}

40

float fpower2(int x)

{

unsigned exp, frac, u;

if (x < -149) // 值太小, 返回0.0

{

exp = 0;

frac = 0;

}

else if (x < -126) // 返回非规格化数

{

exp = 0;

frac = 1 << (149 + x);

}

else if(x < 128) // 返回规格化数

{

exp = x + 127;

frac = 0;

}

else // 值太大. 返回正无穷

{

exp =0xFF ;

frac = 0;

}

u = exp << 23 | frac;

return u2f(u);

}