2.58

typedef unsigned char\* byte\_pointer;

int is\_little\_endian()

{

int x = 1;

return \*((byte\_pointer)& x);

}

2.62

int int\_shifts\_are\_arithmetic() {

return -2 >> 1 == -1;

}

2.66

int leftmost\_one(unsigned x)

{

x |= (x >> 1);

x |= (x >> 2);

x |= (x >> 4);

x |= (x >> 8);

x |= (x >> 16);

return x^(x>>1);

}

2.70

int fits\_bits(int x, int n)

{

x = x >> (n - 1);

return !x || !(~x);

}

2.74

int tsub\_ok(int x, int y)

{

int a = (x > 0) && (y > 0) && (x - y < 0);

int b = (x < 0) && (y < 0) && (x - y > 0);

return a | b;

}

2.78

int divide\_power2(int x, int k)

{

return (x + (1 << k) - 1) >> k;

}

2.82

A.不是总为1。当x或y为TMin时，-x和-y也为TMin，等式左右不相等。

B.总为1。∵17 \* y + 15 \* x = 16 \* (x + y) + y - x = (x + y) << 4 + y – x. ∴等式总为1.

C.总为1。∵~x + ~y + 1 = ~x + 1 + ~y + 1 – 1 = -x + (-y) – 1 = - (x + y) – 1 = ~(x + y) + 1 – 1 = ~(x+y). ∴等式总为1.

D.总为1。无符号数与有符号数的位极表示相同。

E.总为1。算数右移补充最高位，即高位数值不会发生变化。左移的时候会在最右侧补充0，可能使得数值变小。

2.86

|  |  |  |
| --- | --- | --- |
| 描述 | 扩展精度 | |
| 值 | 十进制 |
| 最小的非规格化数 |  |  |
| 最小的正规格化数 |  |  |
| 最大的规格化数 |  |  |

2.90

float fpwr2(int x)

{

/\* Result exponent and fraction \*/

unsigned exp, frac;

unsigned u;

if (x < -149) {

/\* Too small. Return 0.0 \*/

exp = 0;

frac = 0;

} else if (x < -126) {

/\* Denormalized result \*/

exp = 0;

frac = 1<<(x+149);

} else if (x < 128) {

/\* Normalized result. \*/

exp = x + 127;

frac = 0;

} else {

/\* Too big. Return +oo \*/

exp = 0xFF;

frac = 0;

}

/\* Pack exp and frac into 32 bits \*/

u = exp << 23 | frac;

/\* Return as float \*/

return u2f(u);

}

2.94

typedef unsigned float\_bits;

float\_bits float\_twice(float\_bits f)

{

unsigned sign = f >> 31;

unsigned exp = (f >> 23) & 0xff;

unsigned frac = f & 0x7fffff;

if (exp == 0xff)

{

return f;

}

else if (exp == 0)

{

frac <<= 1;

}

else if (exp == 0xfe)

{

exp = 0xff;

frac = 0;

}

else

{

exp++;

}

return (sign << 31) | ((exp) << 23) | (frac);

}