1. Modified Problem of Exercise 2.61 on page 165.

A. Any two bits in the least significant byte of x equal 0. B. Any two bits of x equal 1. C. Any bit in the most significant byte of x equals 1. D. At least two bits of x equal 1.

A.모든게 0

!(x & 0xff)

B. 모든게 1

!(~x);

C.

!((x & 0xff000000) ^ (0xff000000))

D.

int leastTB(int x)

{

int a = 1 | (1 << 8);

a = a | (a << 16);

int result = (x & a) + ((x >> 1) & a) + ((x >> 2) & a) + ((x >> 3) & a)

+ ((x >> 4) & a) + ((x >> 5) & a) + ((x >> 6) & a) + ((x >> 7) & a);

result = result + (result >> 16);

result = result + (result >> 8);

result = result & 0xff;

result = result - 2;//1의 개수가 2를 넘느냐>=0 넘으면 msb 0; <0 안넘으면 msb 1;

return !((result >> 31) & 1);

}

2. Exercise 2.72 on page 169

A. sizeof연산자가 size\_t값을 return하는데 이는 unsigned 형이다. int형과 unsigned형간의 연산 결과는 unsigned라서 항상 0이상의 값을 리턴하므로 maxbyte값에 상관없이 이루어짐. 조건이 충족되는 것이다.

B. if구문 안의 조건을 maxbytes>=sizeof(val)로 수정한다.

3. Modified Problem of Exercise 2.77 on page 171

A. K = 19

B. K = -9

C. K = 54

D. K = - 122

A. 19=2^4+2^2-2^0

(x<<4)+(x<<2)-x;

B. -9=-2^3+-2^0;

-(x<<3)-x

C.54=64-8-2

(x<<6)-(x<<3)-(x<<1)

D. -122=-128+8-2

-(x<<7)+(x<<3)-(x<<1)

4. Exercise 2.81 on page 171

A.

0xffffffff<< k

B.

~(0xffffffff << k) << j

5. Exercise 2.82 on page 171

A. x가 –1이고 y=0인 경우 좌항은 1이지만 우항은 0이되어 같지 않다.(FALSE)

B. always correct x<<k=x\*(2^k)와 multiple distributtion으로 인해 성립(TRUE)

C. ~x=-x-1 ~y=-y-1이므로 ~x+~y+1=-x-y-1이다.

~(x+y)=-(x+y)-1이므로 양변은 일치함을 알 수 있다.(TRUE)

D. unsigned와 singed의 bit표현은 같다.(TRUE)

E. right shift한 후 다시 left shift하는 과정에서 least significant bit로부터 2개의 bit값이 0이되기에 x보다 작거나 같아질 수 밖에 없다.(TRUE)

6. Exercise 2.88 on page 174

1 01111 001 -> -9/8 -> 1 0110 0010 -> -9/8

0 10110 011 -> 176 -> 0 1110 0110 -> 176

1 00111 010 -> -5/1024 ->1 0000 0101 -> -5/1024

0 00000 111 -> 7\*2^(-17) ->0 0000 0001-> 1/1024

1 11100 000 -> -(2)^13 -> 1 1110 1111 -> -248

0 10111 100 -> 384 -> 0 1111 0000 ->+∞

7. Exercise 2.89 on page 174.

A. 똑같은 type으로 캐스팅되기 때문에 만약 올림이나 내림 되면 똑같은 방식으로 된다.(TRUE)

B. x=0, 이고 y=Tmin일 경우 x-y는 overflow를 발생시키에 dx-dy는 double(x-y)와 같지 않다.(FALSE)

C. double형 데이터타입에서 associative law 가 성립한다. (TRUE)

D. overflow가 있을 수 있다. 예를 들어 dx=Tmax, dy=Tmax-1, dz=Tmax-2 값일 때 좌항은 dx\*dy의 곱셈에서 일부 비트가 삭제되고 우항은 dy\*dz에서 일부비트가 삭제되는데 이는 다른 결과값을 발생시킬 수 있다.(FALSE)

E. dx=0인 경우 dx/dx=NaN이고 dz=1인 경우 dz/dz=1이므로 같지 않다.(FALSE)

8. Exercise 2.90 on page 175

float fpwr2(int x)

{

unsigned exp, frac;

unsigned u;

if (x < -149)

{

/\*too small. return0.0\*/

exp = 0;

frac = 0;

}

else if (x < -126)

{

/\* denormalized result\*/

exp = 0;

frac = 1 << (unsigned)(x + 149);

}

else if (x <128)

{

/\*Normalized result\*/

exp = x + 127;

frac = 0;

}

else {

/\*Too big. return +infinity\*/

exp = 0xff;

frac = 0;

}

u = exp << 23 | frac;

return u2f(u);

}

9. Exercise 2.93 on page 178

float\_bits float\_absval(unsigned f)

{

/\*compute |f| if f is NaN, then return f.\*/

unsigned exp = f >> 23 & 0xff;

unsigned frac = f & 0x7fffff;

int check\_nan = (exp == 0xff) && (frac != 0);

if (check\_nan)

{

return f;

}

else

{

return 0x7fffffff & f;

}

}

10. Modified Problem of Exercise 2.96 on page 178

Compute (unsigned int) f

/\* If f is NaN or overflow, return 0x7FFFFFFF

If f is negative, return 0x80000000 \*/

unsigned float\_f2i(unsigned f)

{

unsigned sign = (f >> 31) & 1;

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

unsigned frac = f & 0x7fffff;

int E = exp - 127;

unsigned M;

int x;

if ((exp == 0xff) & (frac != 0))//음수 양수에 상관없이 NaN인 경우

return 0x7fffffff;

if (E < 0)//less than 1

{

return 0;

}

else if (E > 30)//overflow

{

if (sign == 1)//음수의 경우

return 0x80000000;

return 0x7fffffff;

}

else

{

M = frac | 0x800000;

if (E > 23)

{

x = M << (E - 23);

}

else {

x = M >> (23 - E);

}

}

if (sign)

return -x;

return x;

}