\* bitAnd - x&y using only ~ and |

\* Example: bitAnd(6, 5) = 4

\* Legal ops: ~ |ujjjjjjjjjjjhujujjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjj

\* Max ops: 8

\* Rating: 1

\*/

int bitAnd(int x, int y) {

return ~((~x)|(~y));

}

/\*

\* getByte - Extract byte n from word x

\* Bytes numbered from 0 (LSB) to 3 (MSB)

\* Examples: getByte(0x12345678,1) = 0x56

\* Legal ops: ! ~ & ^ | + << >>

\* Max ops: 6

\* Rating: 2

\*/

int getByte(int x, int n) {

int result = x>>(n<<3);

result = result & 255;

return result;

}

/\*

\* logicalShift - shift x to the right by n, using a logical shift

\* Can assume that 0 <= n <= 31

\* Examples: logicalShift(0x87654321,4) = 0x08765432

\* Legal ops: ! ~ & ^ | + << >>

\* Max ops: 20

\* Rating: 3

\*/

int logicalShift(int x, int n) {

int a = ((1<<31)>>n)<<1;

a = ~a;

return (x>>n)&a;

}

/\*

\* bitCount - returns count of number of 1's in word

\* Examples: bitCount(5) = 2, bitCount(7) = 3

\* Legal ops: ! ~ & ^ | + << >>

\* Max ops: 40

\* Rating: 4

\*/

int bitCount(int x) {

int mask1,mask2,mask4,mask8,mask16;

mask1 = (0x55)|(0x55<<8);

mask1 = (mask1)|(mask1<<16);

mask2 = (0x33)|(0x33<<8);

mask2 = (mask2)|(mask2<<16);

mask4 = (0x0f)|(0x0f<<8);

mask4 = mask4|(mask4<<16);

mask8 = (0xff)|(0xff<<16);

mask16 = (0xff)|(0xff<<8);

x=(x&mask1)+((x>>1)&mask1);

x=(x&mask2)+((x>>2)&mask2);

x=(x+(x>>4))&mask4;

x=(x+(x>>8))&mask8;

x=(x+(x>>16))&mask16;

return x;

}

/\*

\* bang - Compute !x without using !

\* Examples: bang(3) = 0, bang(0) = 1

\* Legal ops: ~ & ^ | + << >>

\* Max ops: 12

\* Rating: 4

\*/

int bang(int x) {

return ~((x|(~x+1))>>31)&1;

}

/\*

\* tmin - return minimum two's complement integer

\* Legal ops: ! ~ & ^ | + << >>

\* Max ops: 4

\* Rating: 1

\*/

int tmin(void) {

return 1<<31;

}

/\*

\* fitsBits - return 1 if x can be represented as an

\* n-bit, two's complement integer.

\* 1 <= n <= 32

\* Examples: fitsBits(5,3) = 0, fitsBits(-4,3) = 1

\* Legal ops: ! ~ & ^ | + << >>

\* Max ops: 15

\* Rating: 2

\*/

int fitsBits(int x, int n) {

int number = 32+(~n+1);

return !(x^((x<<number)>>number));

}

/\*

\* divpwr2 - Compute x/(2^n), for 0 <= n <= 30

\* Round toward zero

\* Examples: divpwr2(15,1) = 7, divpwr2(-33,4) = -2

\* Legal ops: ! ~ & ^ | + << >>

\* Max ops: 15

\* Rating: 2

\*/

int divpwr2(int x, int n) {

int mask1 = x>>31;

int mask2 = mask1&((1<<n)+(~0));

return (x+mask2)>>n;

}

/\*

\* negate - return -x

\* Example: negate(1) = -1.

\* Legal ops: ! ~ & ^ | + << >>

\* Max ops: 5

\* Rating: 2

\*/

int negate(int x) {

return ~x+1;

}

/\*

\* isPositive - return 1 if x > 0, return 0 otherwise

\* Example: isPositive(-1) = 0.

\* Legal ops: ! ~ & ^ | + << >>

\* Max ops: 8

\* Rating: 3

\*/

int isPositive(int x) {

return !(((x>>31)|(!x)));

}

/\*

\* isLessOrEqual - if x <= y then return 1, else return 0

\* Example: isLessOrEqual(4,5) = 1.

\* Legal ops: ! ~ & ^ | + << >>

\* Max ops: 24

\* Rating: 3

\*/

int isLessOrEqual(int x, int y) {

int signx = (x>>31)&1;

int signy = (y>>31)&1;

int yx = ((y+(~x+1))>>31)&1;

int signSame = (!(signx^signy))&(!yx);

int signDiffer = (signx^signy)&signx;

return (signSame|signDiffer);

}

/\*

\* ilog2 - return floor(log base 2 of x), where x > 0

\* Example: ilog2(16) = 4

\* Legal ops: ! ~ & ^ | + << >>

\* Max ops: 90

\* Rating: 4

\*/

int ilog2(int x) {

int sign,shift1,shift2,shift3,shift4,shift5;

sign=!!(x>>16);

shift1=sign<<4;

x=x>>shift1;

sign=!!(x>>8);

shift2=sign<<3;

x=x>>shift2;

sign=!!(x>>4);

shift3=sign<<2;

x=x>>shift3;

sign=!!(x>>2);

shift4=sign<<1;

x=x>>shift4;

sign=!!(x>>1);

shift5=sign;

return shift1+shift2+shift3+shift4+shift5;

}