Practice Problem 2.13 (solution page 183)

The Digital Equipment VAX computer was a very popular machine from the late 1970s until the late 1980s. Rather than instructions for Boolean operations AND and OR, it had instructions **bis** (bit set) and **bic** (bit clear). Both instructions take a data word **x** and a mask word **m**. They generate a result **z** consisting of the bits of **x** modified according to the bits of **m**. With **bis**, the modification involves setting **z** to 1 at each bit position where **m** is 1. With **bic**, the modification involves setting **z** to 0 at each bit position where **m** is 1.

To see how these operations relate to the C bit-level operations, assume we have functions **bis** and **bic** implementing the bit set and bit clear operations, and that we want to use these to implement functions computing bitwise operations **|** and **^**, without using any other C operations. Fill in the missing code below. *Hint:* Write C expressions for the operations **bis** and **bic**.

```
/* Declarations of functions implementing operations bis and bic */
int bis(int x, int m);
int bic(int x, int m);

/* Compute x|y using only calls to functions bis and bic */
int bool_or(int x, int y) {
    int result = bis(x,y);
    return result;
}

/* Compute x^y using only calls to functions bis and bic */
int bool_xor(int x, int y) {
    int result = _________;
    return result;
}
```

$bis(x,y) = x \mid y$

$bic(x,y) = x \,\&\, (\sim y)$

$x \wedge y = (x \,\&\, (\sim y)) \mid (y \,\&\, (\sim x))$

$z = x \mid y = bis(x,y)$

Set z to 1 at each bit where y is 1.

$z = x \wedge y = bis(bic(x,y), bic(y,x))$

if $x_i = 1$ and $y_i = 1$, set $x_i = 0$
if $x_i = 0$, x_i will remain 0

if $y_i = 1$ and $x_i = 1$, set $y_i = 0$
if $y_i = 0$, y_i will remain 0 no matter what x_i is

if $x_i \neq y_i$, set z_i to 1

XOR

x\y	0	1
0	0	1
1	1	0