Almost everything in this problem is a bitwise operation, except the subtraction in $F(x, y, z)$.
However, notice that $(x \mid y)$ will contain all the set bits of x, while $(x \ \& \ z)$ cannot contain any bits that aren't set in x.
So, $(x \ \& \ z)$ is a *submask* of $(x \mid y)$, and this means the subtraction operation is really just bitwise XOR.
That is, we can rewrite the function as $F(x, y, z) = (x \mid y) \oplus (x \ \& \ z)$.

Now that everything we're dealing with is bitwise operation, it helps to look at what's going on bit-by-bit.

Let's fix a bit k and see what happens.
Let x_k denote the value of the k-th bit of x (which is either 0 or 1). Similarly define $A_k, B_k, F_k(x, A, B)$.
We already know A_k and B_k, our aim is to find out what values x_k can possibly take.

We have $F_k(x, A, B) = (x_k \mid A_k) \oplus (x_k \ \& \ B_k)$, and we'd like to maximize $F_k(x, A, B)$.

There are 4 cases depending on their values:

- $A_k = B_k = 1$
  Here, $(x_k \mid A_k) = 1$, and $(x_k \ \& \ B_k) = x_k$.
  So, $F_k(x, A, B) = 1 \oplus x_k$, which is maximized when $x_k = 0$.
- $A_k = B_k = 0$
  Here, $(x_k \mid A_k) = x_k$, and $(x_k \ \& \ B_k) = 0$.
  So, $F_k(x, A, B) = x_k \oplus 0$, which is maximized when $x_k = 1$.
- $A_k = 1$ and $B_k = 0$
  Here, $F_k(x, A, B) = 1 \oplus 0 = 1$, and is independent of x_k. So, we can choose x_k to be either 0 or 1
- $A_k = 0$ and $B_k = 1$
  Here, $F_k(x, A, B) = x_k \oplus x_k = 0$, once again we can choose x_k freely.

Putting the above together:

- If $A_k = B_k$, x_k is fixed and we have no choice.
- If $A_k \neq B_k$, x_k can be chosen freely.

So, the answer is simply 2^d, where d is the number of bits where A and B differ in their binary representations.
d can be computed by iterating over each bit independently; or you can notice that it's just the number of set bits in $(A \oplus B)$ for an $O(1)$ solution.