

GitHub - We're hiring engineers to make GitHub fast. Interested? Email fast@github.com with details of your exceptional performance work you've done in the past.

Execute the initialization program. What is the sum of all values left in memory after it completes?

Your puzzle answer was `8471403462063`.

--- Part Two ---

For some reason, the sea port's computer system still can't communicate with your ferry's docking program. It must be using **version 2** of the decoder chip!

A **version 2** decoder chip doesn't modify the values being written at all. Instead, it acts as a **memory address decoder**. Immediately before a value is written to memory, each bit in the bitmask modifies the corresponding bit of the destination **memory address** in the following way:

- If the bitmask bit is `0`, the corresponding memory address bit is **unchanged**.
- If the bitmask bit is `1`, the corresponding memory address bit is **overwritten with 1**.
- If the bitmask bit is `X`, the corresponding memory address bit is **floating**.

A **floating** bit is not connected to anything and instead fluctuates unpredictably. In practice, this means the floating bits will take on **all possible values**, potentially causing many memory addresses to be written all at once!

For example, consider the following program:

```
mask = 00000000000000000000000000000000X1001X
mem[42] = 100
mask = 00000000000000000000000000000000X0XX
mem[26] = 1
```

When this program goes to write to memory address `42`, it first applies the bitmask:

```
address: 00000000000000000000000000000000101010 (decimal 42)
mask:    00000000000000000000000000000000X1001X
result:  00000000000000000000000000000000X1101X
```

After applying the mask, four bits are overwritten, three of which are different, and two of which are **floating**. Floating bits take on every possible combination of values; with two floating bits, four actual memory addresses are written:

```
0000000000000000000000000000000011010 (decimal 26)
0000000000000000000000000000000011011 (decimal 27)
00000000000000000000000000000000111010 (decimal 58)
00000000000000000000000000000000111011 (decimal 59)
```

Next, the program is about to write to memory address `26` with a different bitmask:

```
address: 0000000000000000000000000000000011010 (decimal 26)
mask:    00000000000000000000000000000000X0XX
result:  000000000000000000000000000000001X0XX
```

This results in an address with three floating bits, causing writes to **eight** memory addresses:

You can also [\[Share\]](#) this puzzle.