--- Day 24: Crossed Wires ---

You and The Historians arrive at the edge of a large grove somewhere in the jungle. After the last incident, the Elves installed a small device that monitors the fruit. While The Historians search the grove, one of them asks if you can take a look at the monitoring device; apparently, it's been malfunctioning recently.

The device seems to be trying to produce a number through some boolean logic gates. Each gate has two inputs and one output. The gates all operate on values that are either true (1) or false (0).

- AND gates output 1 if both inputs are 1; if either input is 0, these gates output 0.
- OR gates output 1 if one or both inputs is 1; if both inputs are 0, these gates output 0.
- XOR gates output 1 if the inputs are different; if the inputs are the same, these gates output 0.

Gates wait until both inputs are received before producing output; wires can carry 0, 1 or no value at all. There are no loops; once a gate has determined its output, the output will not change until the whole system is reset. Each wire is connected to at most one gate output, but can be connected to many gate inputs.

Rather than risk getting shocked while tinkering with the live system, you write down all of the gate connections and initial wire values (your puzzle input) so you can consider them in relative safety. For example:

```
x00: 1
x01: 1
x02: 1
y00: 0
y01: 1
y02: 0

x00 AND y00 -> z00
x01 XOR y01 -> z01
x02 OR y02 -> z02
```

Because gates wait for input, some wires need to start with a value (as inputs to the entire system). The first section specifies these values. For example, `x00: 1` means that the wire named `x00` starts with the value `1` (as if a gate is already outputting that value onto that wire).

The second section lists all of the gates and the wires connected to them. For example, `x00 AND y00 -> z00` describes an instance of an AND gate which has wires `x00` and `y00` connected to its inputs and which will write its output to wire `z00`.

In this example, simulating these gates eventually causes `0` to appear on wire `z00`, `0` to appear on wire `z01`, and `1` to appear on wire `z02`.

Ultimately, the system is trying to produce a number by combining the bits on all wires starting with `z`. `z00` is the least significant bit, then `z01`, then `z02`, and so on.

In this example, the three output bits form the binary number `100` which is equal to the decimal number `4`.

Here's a larger example:

```
x00: 1
x01: 0
x02: 1
x03: 1
x04: 0
y00: 1
y01: 1
y02: 1
y03: 1
y04: 1

ntg XOR fgs -> mjb
y02 OR x01 -> tnw
kwq OR kpj -> z05
x00 OR x03 -> fst
tgd XOR rvg -> z01
vdt OR tnw -> bfw
bfw AND frj -> z10
ffh OR nrd -> bqk
y00 AND y03 -> djm
y03 OR y00 -> psh
bqk OR frj -> z08
tnw OR fst -> frj
gnj AND tgd -> z11
bfw XOR mjb -> z00
x03 OR x00 -> vdt
gnj AND wpb -> z02
x04 AND y00 -> kjc
djm OR pbm -> qhw
nrd AND vdt -> hwm
kjc AND fst -> rvg
y04 OR y02 -> fgs
y01 AND x02 -> pbm
ntg OR kjc -> kwq
psh XOR fgs -> tgd
qhw XOR tgd -> z09
pbm OR djm -> kpj
x03 XOR y03 -> ffh
x00 XOR y04 -> ntg
bfw OR bqk -> z06
nrd XOR fgs -> wpb
frj XOR qhw -> z04
bqk OR frj -> z07
```

```
bqk OR rvj -> z07
y03 OR x01 -> nrd
hwm AND bqk -> z03
tgd XOR rvg -> z12
tnw OR pbm -> gnj
```

After waiting for values on all wires starting with z, the wires in this system have the following values:

```
bfw: 1
bqk: 1
djm: 1
ffh: 0
fgs: 1
frj: 1
fst: 1
gnj: 1
hwm: 1
kjc: 0
kpj: 1
kwq: 0
mjb: 1
nrd: 1
ntg: 0
pbm: 1
psh: 1
qhw: 1
rvg: 0
tgd: 0
tnw: 1
vdt: 1
wpb: 0
z00: 0
z01: 0
z02: 0
z03: 1
z04: 0
z05: 1
z06: 1
z07: 1
z08: 1
z09: 1
z10: 1
z11: 0
z12: 0
```

Combining the bits from all wires starting with z produces the binary number 0011111101000. Converting this number to decimal produces 2024.

Simulate the system of gates and wires. What decimal number does it output on the wires starting with z?