multiply.pep & divide.pep:

Given to me as a challenge by my teacher. I completed this algorithm before the class could cover it. I then set to work reversing the algorithm to create divide.pep

The problem:

In pep8 you only have instructions for: add, subtract and bit shift (multiply/divide by 2) but no ability to multiply or divide.

My method:

5 \* 5 = 25

101 \* 101 = ???

(4+1)\*5 = 25 --> 4\*5 =20, 1\*5 = 5

100\*101 = 10100 1\*101=101

10100

+ 101

-------

11001

Now to make it generic. What if I destroyed the first number, while keeping track of the exponent. I would simply bit shift the second number that many times every time I lost data from the first number and then add it to a sum total.

22 21 20

1 0 1 --> 101 \* 20 + 101 \* 22 = 101 + 10100 = 11001

Now division is simply a reversal of the algorithm... nothing is that simple though.

25 / 5 = 5

(20 + 5)/5 --> 4+1 = 5

11001...

hmm... maybe a different algorithm was needed. I chose to mimic how we learn division. We first find a multiple of our divisor that is just smaller than our target and subtract, tallying the multiple up top before repeating. To make this work for a computer I would need to over shoot my target number and then to go back one step. So I would first bit shift the smaller number, keeping track of how many times it was shifted, until it was bigger than the first number. I then bit shifted the second number left and subtracted it.

11001 / 101

20 = 101, 21 = 1010, 22 = 10100, 23 = 101000 > 11001

3 as an exponent is to big so we remember 2

11001 – 10100 = 101 --> 22 or 4 is recorded

next we backtrack until our number is less than or equal to our remainder

21 = 1010 > 101

20 = 101 <= 101 --> 101-101= 0 --> record 20 or 1

Either our exponent is at 0 or our remainder is at 0 so we stop and tally.

4+1 = 5 remainder 0