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 and many assemblers you only have instructions for: add, subtract and bit shift (multiply/divide by 2) but no ability to generically 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, remembering the exponent. I would simply bit shift the second number that many times every time I lost a 1 from the first number.

22 21 20

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

Now division is simply a reversal of the algorithm... nothings that simple though.

25 / 5 = 5

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

11001...

hmm... maybe a different algorithm was needed. I chose to next 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

1 = 101, 2 = 1010, 4 = 10100, 8 = 101000

8 is to big so we remember 4

11001 – 10100 = 101 --> record 4

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

101-101 = 0 --> record 1

4+1 = 5