For this first lab, we were only to do part 1, so I’ll explain my solution to each “task” the subroutine had to solve:

1. Split the decimal into two equal halves.
   1. We can do this for any even-digited number by dividing it by 1 followed by 0s, where n is the number of digits. For example, here we have 1233, so we divide it by 1 followed by two 0s (aka 100). Since the division function we’ve covered in class calculates both the quotient and remainder, the quotient will be the top half of the number and the remainder will be the bottom half.
2. Calculate the squares of these two halves.
   1. Since 12 and 33 are both 1-byte integers, we can use the built in MUL instruction to get 12\*12 and 33\*33 (which will be returned into r0:r1 both times).
3. Add the squares together.
   1. Same as we’ve done before, just ADD the low bytes and then ADC the high bytes.
4. Check if the result is the same as the original number. If it is, return 1, else return 0.
   1. Do a comparison between the low byte of our result and the low byte of the original number. Same with the high byte. If either of these are not equal, return 0. Else return 1.

For proof of the program working, we can see in this image that the result of splitting, squaring and adding (stored in r6:r7) is the same as the original number (stored in r16:r17), and that our returned value is 1 (stored in r25).

