The description of the program was that it needed to be able to:

“[R]ead an 8-bit array (40 elements) from program memory. Swap the upper 4 bits and lower 4 bits of every element and store it to data memory only if the number is between 39 and 117 inclusive, else store a 0. Also, determine the sum of the numbers in data memory and store it in a register.”

From here, it was evident the program would need 2 loops:

1. A loop to load the array elements, swap the nibbles and store it in data memory according to the conditions stated (), that is, a “swap” loop, and
2. A loop to go through the information in data memory and add it, that is, an “add” loop.

For the swap, after browsing through the AVR Instruction Manual the “SWAP” and “BRLT” commands seemed useful, and thus these were used to swap the nibbles of each element, along with testing if the swapped number was less than 39 or 118 (and then branching from that).

For evidence it’s working, there is an attached a picture of the registers & data memory after execution, along with a notepad file of the program done by hand. When the program is done manually (write the numbers 1-40 in binary, swap their nibbles, and set what their value should be according to the conditions), it matches up with the values written that the program writes into data memory. Then, when adding these values together by hand, the result is 00x04BF, the same as what the program has stored in r1:r0 after the “add” loop.

