Lince Rumainum
Assembly Programming Project Documentation
CS 2614 Computer Organization

For this project, to be able to solve the given problem, the first thing to do is taking the input ASCII characters of the two digits numbers using the input register. We will be able to do that by using the input-output instruction INP (Input character to accumulator) to get the user's input. The input register will take the characters and then store it as low-order bits corresponding to it. From there we will have to go into several different loops to get the result we want, which is the mean.

Since there are no while or for loop, we will use the memory-reference instruction BUN (Branch unconditionally) to do the loop. The first loop will be used to calculate and then store the sum of the numbers from 01 to *N* (the input number). For this calculation the loop condition will have a counter (i.e. SUMCTR), which is the 2's complement of *N*. We will use the register-reference instruction CMA (Complement accumulator) to get the complement of *N* and increment that value. That counter will be inside of the loop and incremented using the memory-reference instruction ISZ (Increment and skip if zero) that will exit the loop once it added all the numbers from 01 to *N*. Once the sum is calculated and stored, the program will then enter the next loop. This loop is the start of calculating the mean value.

The first loop is getting the quotient value. We will use the memory-reference instruction BUN to do the loop. The sum will be the dividend, the input *N* is the divisor. Since we cannot do division, we will go into the loop where we take the sum and add it to the 2's complement of *N* (essentially subtract the sum with *N*). It will keep subtracting the difference to *N* until the result is negative while keeping track of the quotient by incrementing the quotient variable when it does not. Once the result become negative, the register-reference instruction SNA (Skip next instruction if accumulator is negative) will exit the BUN loop for the quotient and store the reminder into its variable. Since the reminder variable is negative when we exit the quotient loop, we will need to add the reminder with N to get the actual reminder value. Once we do that, we will then use another memory-reference instruction BUN to do the loop to calculate the decimal point of the mean. Depending on how many digits behind the decimal points we want, we will need to shift the reminder to the left (at least once) until the subtraction between the reminder and *N* is positive. To be able to do that we will use the register-reference instruction SPA (Skip next instruction if accumulator is positive). Once the reminder is shifted enough times, we will then calculate the reminder the same way as we did the quotient. We combine the quotient and the reminder and use the input-output instruction OUT (Output the character from the accumulator) to convert the low-order-bits into the ASCII character corresponding to it.