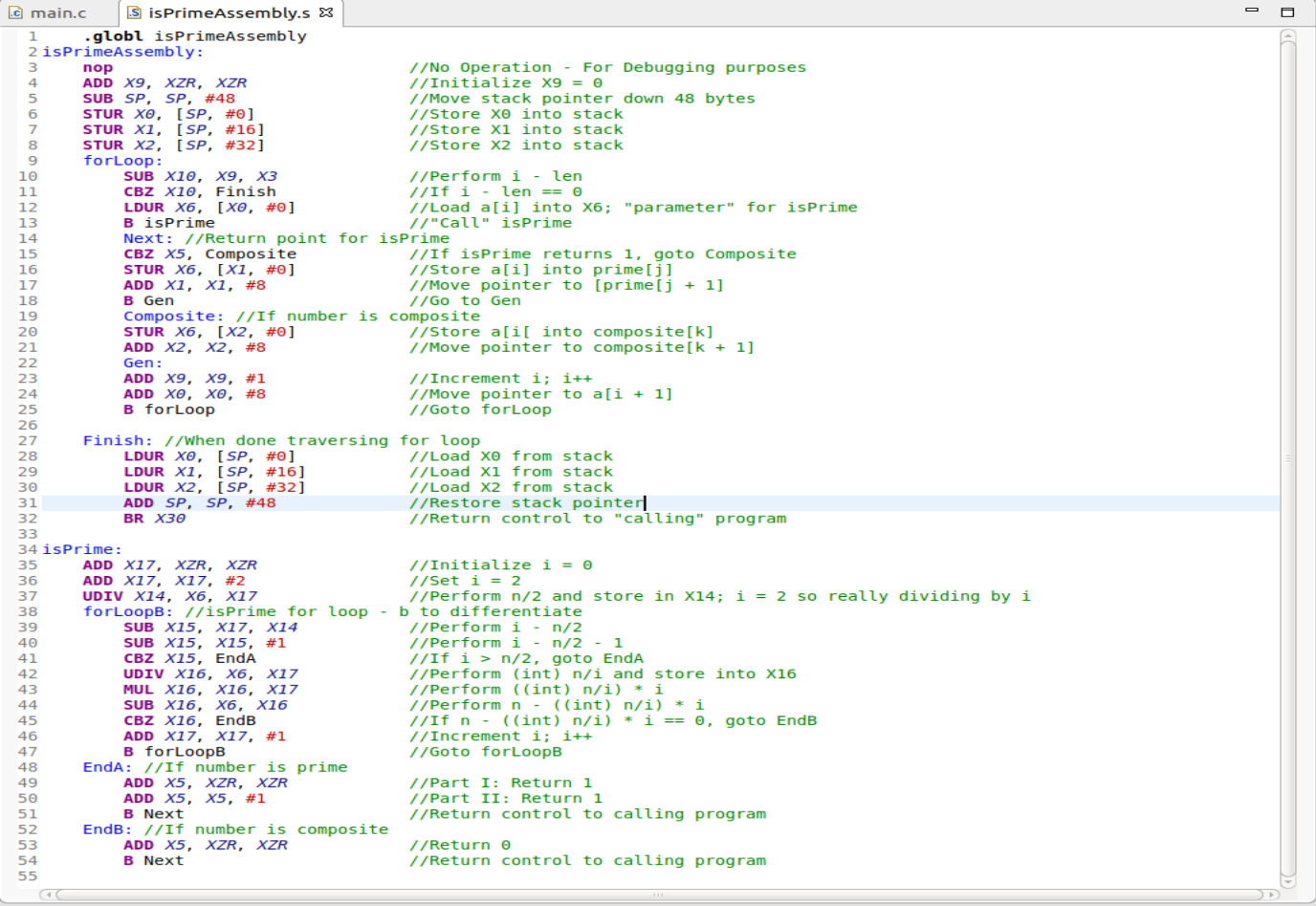
Victor Lam

Prof. Gummesson

ECE 331

5 October 2020

Code:



I designed my overall logic structure based off of the given C code but with a few adjustments. I didn’t need and couldn’t really use all the variables in the original C code like ‘j’, ‘k’, and ‘temp’. Instead of accessing a specific array index from the original array pointer, I decided to iterate the pointer to the next element of the array. I loaded the value of a[i] into a register so it would be easily accessible for use by both isPrime and for storing it into one of the arrays. For my isPrime, I used a for loop again. Since we were only checking if the modulus was returning a 0 or not, I took advantage of the properties of integer division where decimals are cut off. This means that if the number isn’t perfectly divisible by ‘i’, it would return an integer regardless. When I multiplied this integer back by ‘i’, it would not be equal to the original number. This made it easy as I only had to check whether they were equal by subtraction. If the values were equal, I would immediately branch and return ‘0’ (putting 0 into X5) indicating the number was composite. Otherwise, the for loop would continue to run. If the loop finishes running, ie i > n/2, then it would return 1 indicating the number was prime. After going back to isPrimeAssembly, the value of register X5 was checked to see if the number was prime or composite. This would then determine which array it was added to. After storing the value in the memory location, the pointer for the prime/composite array was incremented by 8 to the next element in the array for future use. The ‘Gen’ label is where the updates happen; incrementing i and incrementing the pointer for the array a[] to the next index, and going back to the top of the for loop to be run again. Since I changed the values of the pointers, to get them back to their original values to check the memory, I stored the pointers onto the stack in the beginning and I restored them to their original values by popping them from the stack.

Screenshots:

Figure 1. Array “a[]”

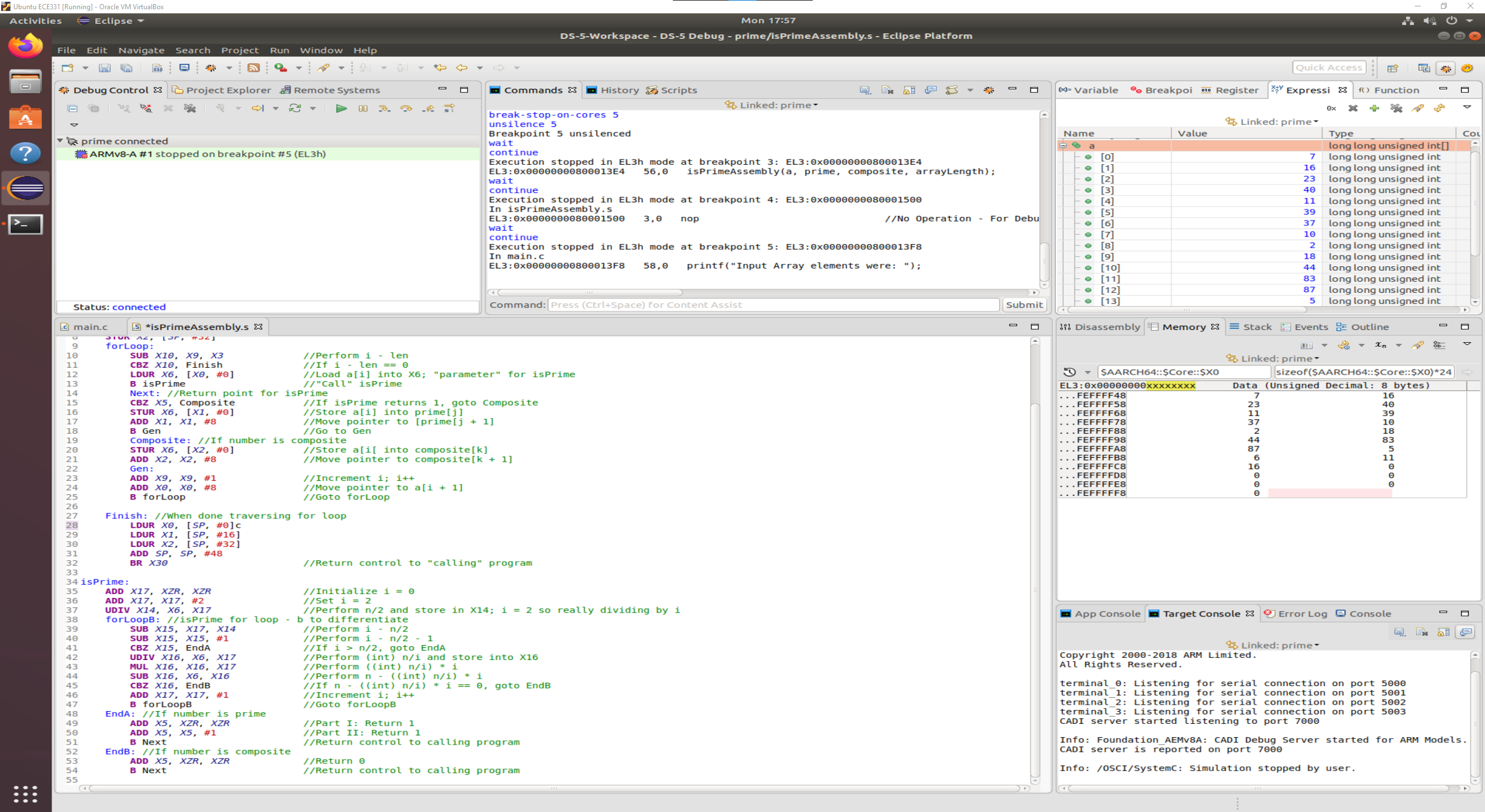


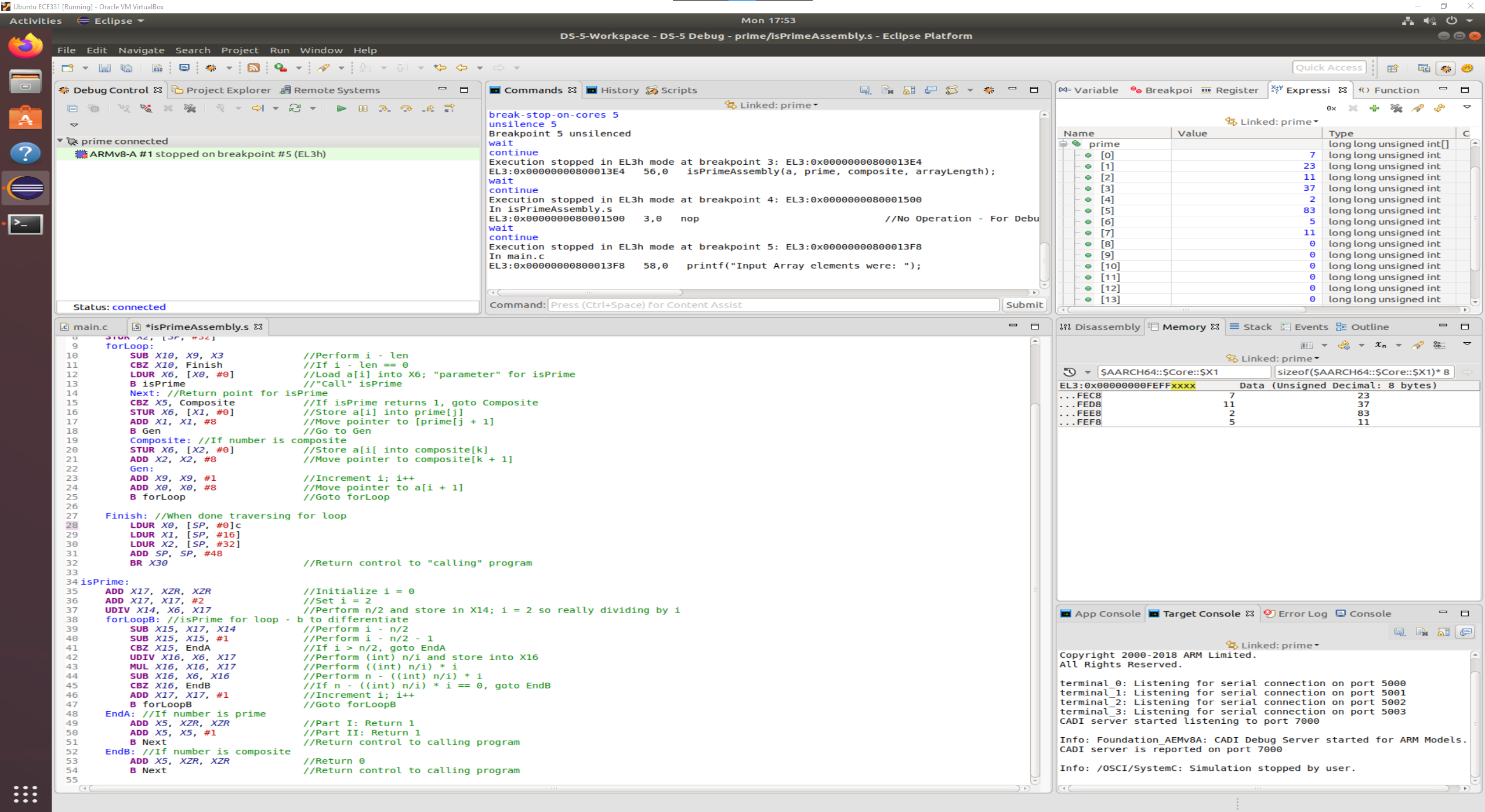
Figure 2. Array “prime[]”

Figure 3. Array “composite[]”

