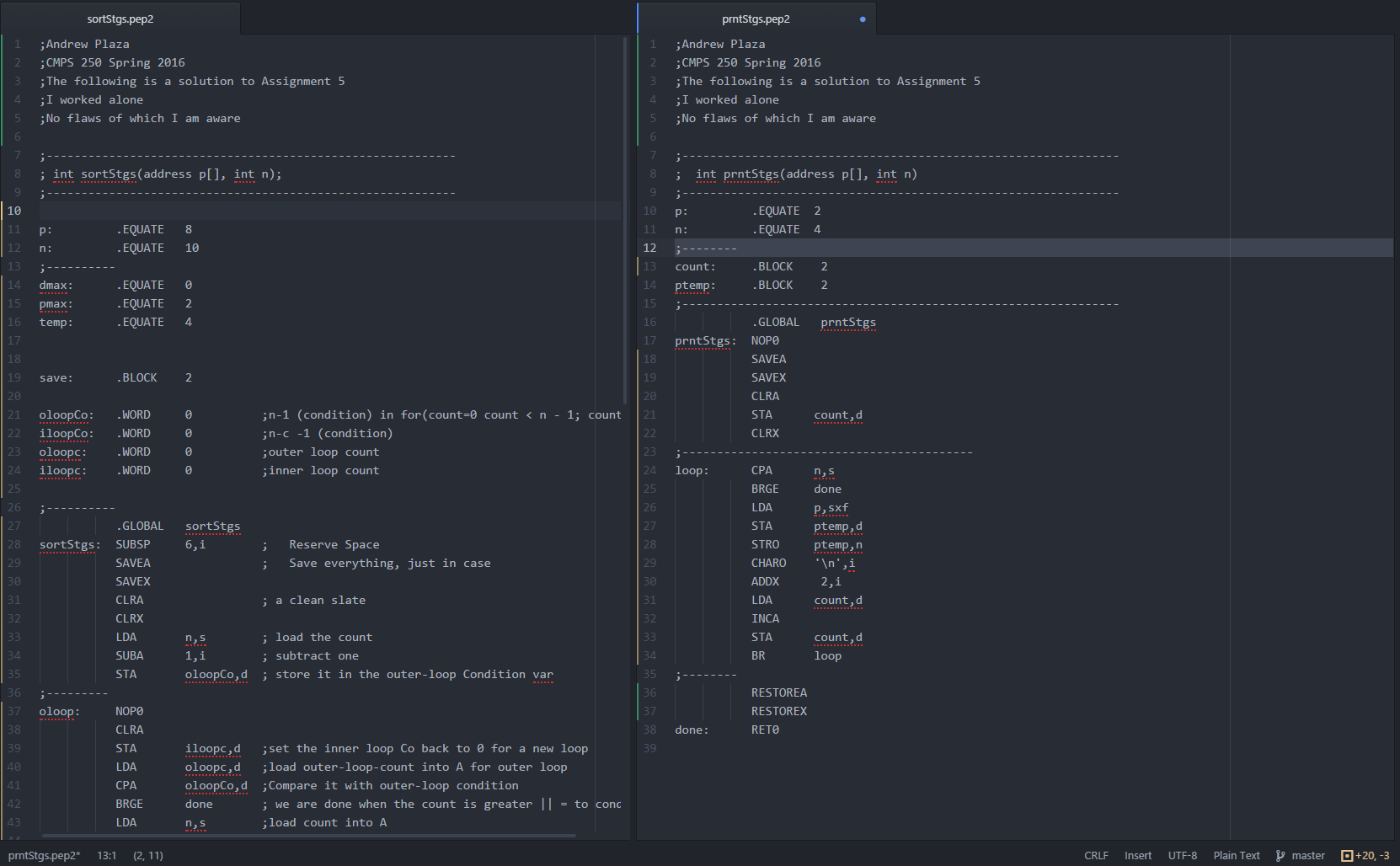
Andrew Plaza

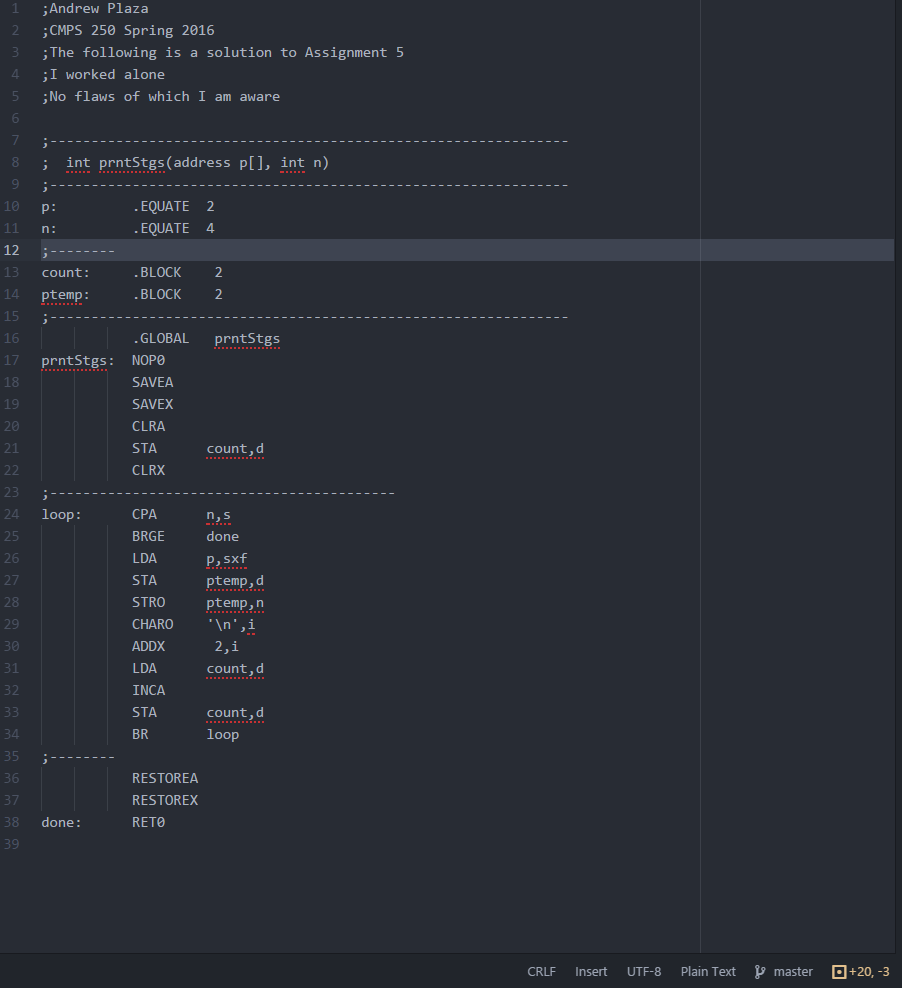
Professor Jackowitz

CMPS 250

27 April 2016

Assignment 5

Code In It’s Entirety:

prntStgs.pep2

<p is offset 2 bytes from stack

<n offset by 4 bytes

<count is a block of 2 bytes

<ptemp is as well

<Save A and X and clear them for a “clean slate”

<count is set to 0 at first

<compare the count with n(the amount of strings) stack-relative. If A is >= we are done

<load the first address into A

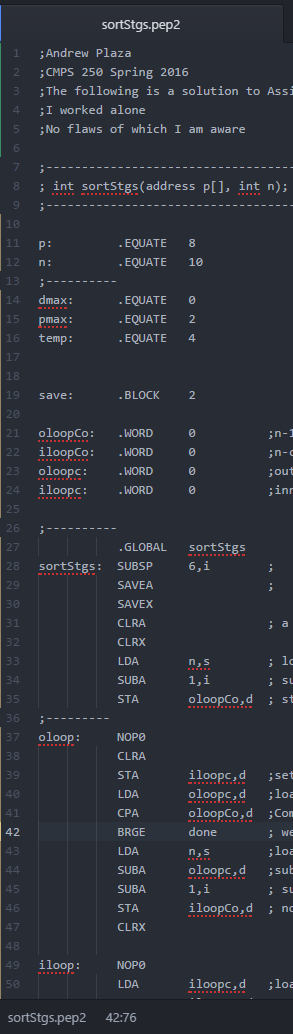
<print the string using indirect addr mode (n)

<Increment X to get to next addr in array

<load count, increment it

<store count back into A

This repeats until count is >= to our amount of Strings

sortStgs.pep2

<p is 8 from the stack, since there is 6offset reserved (line 28)

<by the same logic, n is 10 from the SP

<dmax is 0 from the stack, since once we call sCompTo a push occurs, and ScompTo needs this 2 from the SP

<by the same logic pmax is 2 from the SP

<temp is 4 from the SP, but it is not used in this implementation

<the condition for the outer-loop

<condition for the inner loop

<the count for the outer-loop

<the count for the inner-loop

**Creating a ‘clean slate’ and getting ready**

<the space (6) being reserved for putting things onto the stack

<Save A and X, clear them. This creates a “clear slate”

<load n (the count) into A, stack-relative

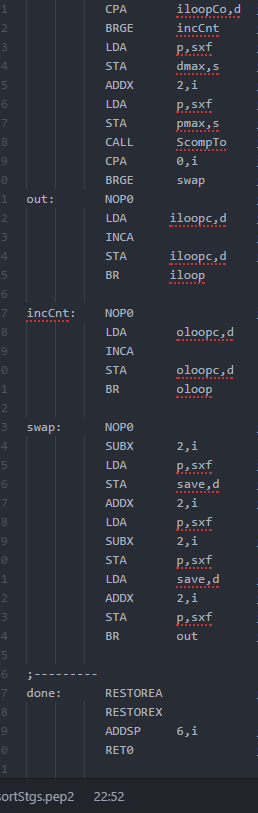
<Subtract one, this is now the outer-loop condition(EX if the count is 10 oloopCo = 9)

**Creating the outer-loop**

<Clear A at the start of the outer-loop and store it in the inner-loop-count, effectively restarting the inner loop

<load the outer-loop count into A and compare it against it’s condition, the outer-loop will continue as long as it’s count is < it’s condition

<create the inner loop condition by subtracting the outer-loop-count from the overall number of strings there are, and then subtracting 1 on top of that

**The Inner Loop**

Inner-loop count is loaded into A, as lon as the iloopc is < it’s condition, we continue. If it is not, we branch to increment the outer-loop count

<We load an address into the A register and store it in dmax on the stack, then increment X by two and load the next address into the a register. We store that address in pmax on the stack. <Now we can call ScompTo in order to compare the two strings these addresses are pointing to

<If the first String is greater than the second, we need to swap them.

<”out” increments the inner-loop count

<”incCnt” increments the outer-loop count

**The Swap**

We subtract 2 from the X register to get to the first addr we need to swap with, and save it in save, directly

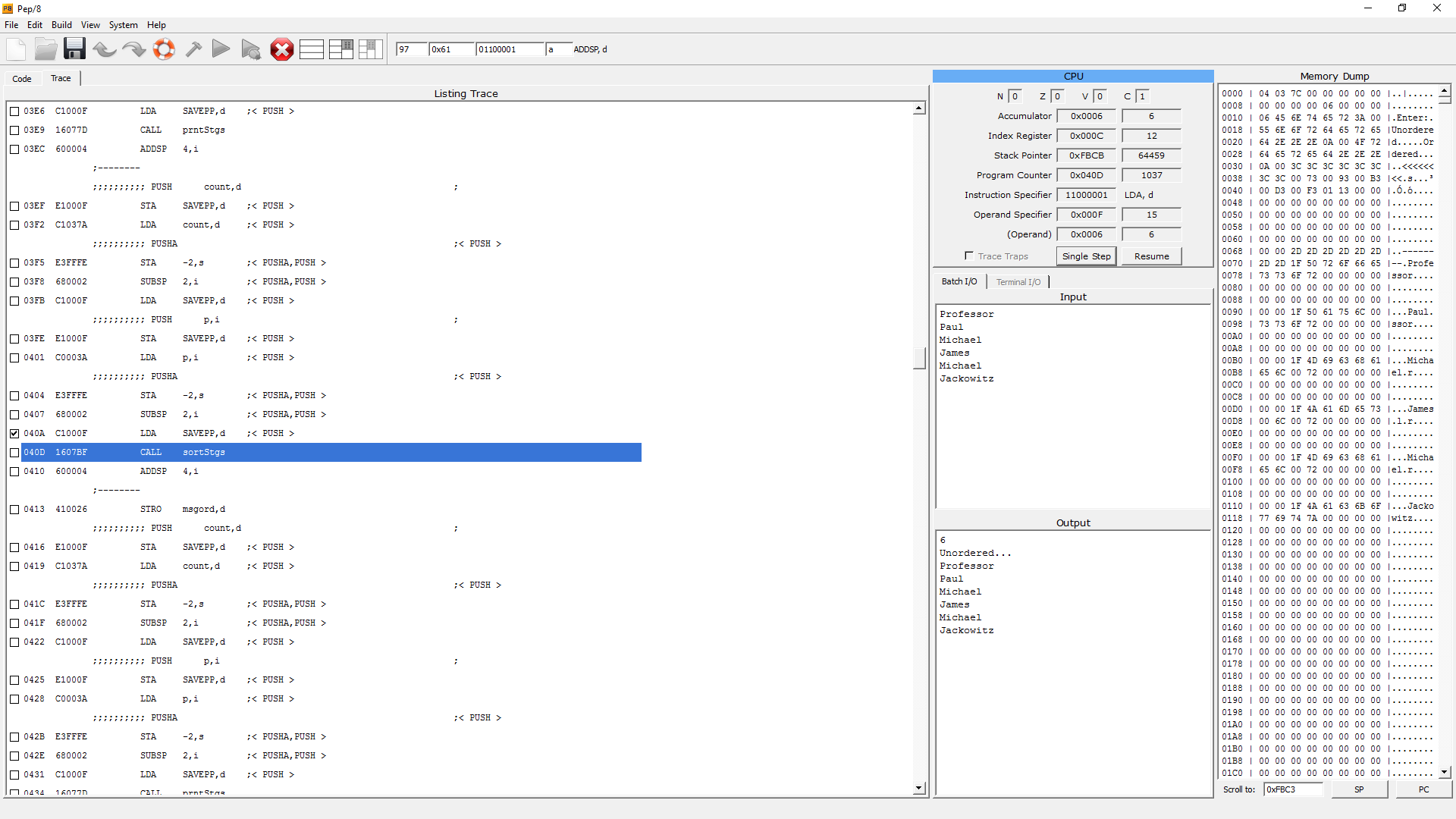
<Then we increment X by two to get to the next addr, subtract X by two, and store the next addr in the first position.

<then we load save, add 2 to X to get back to the second position, and store save there.

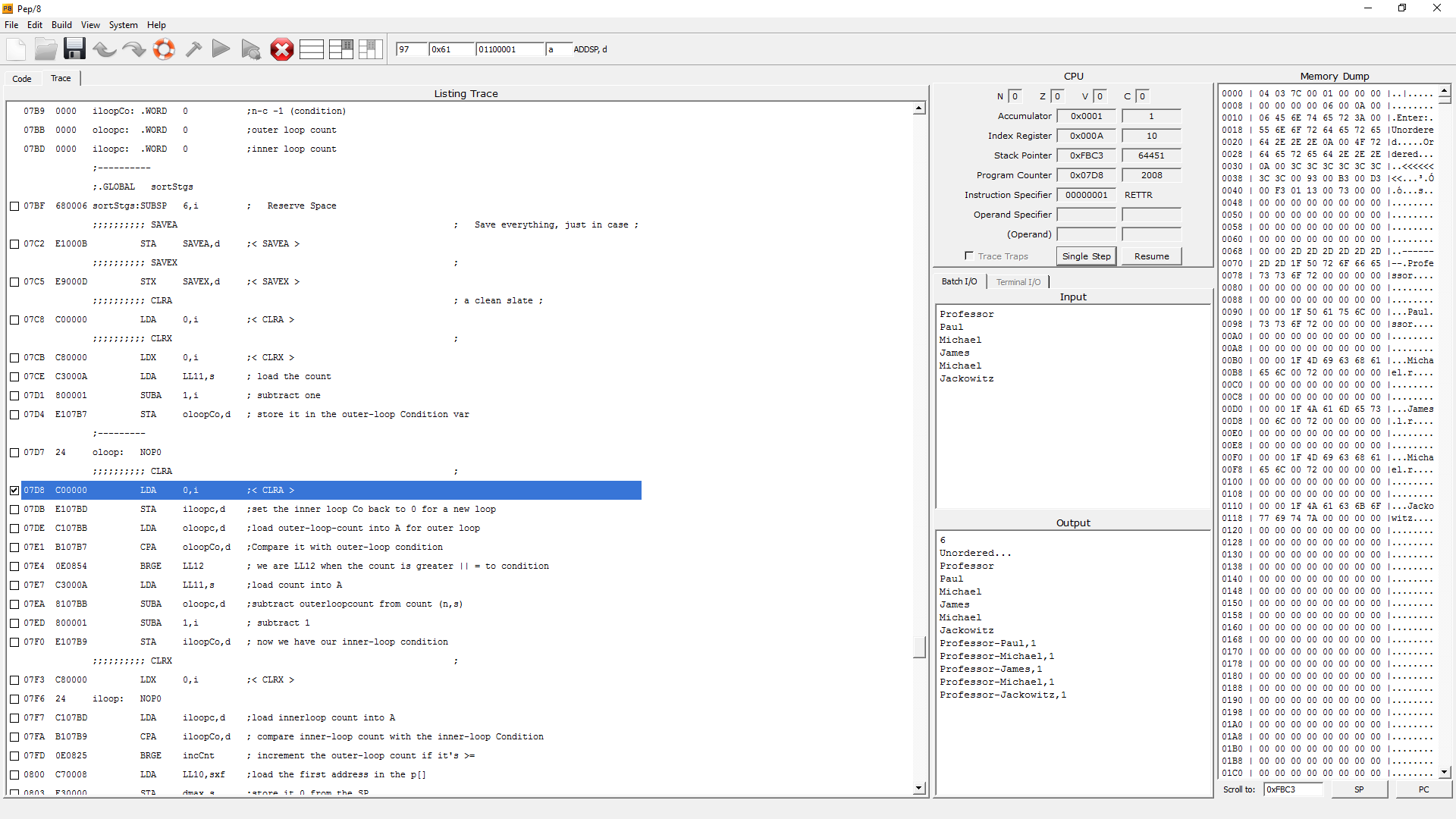
<then we branch to “out” in order to increment the inner-loop-count

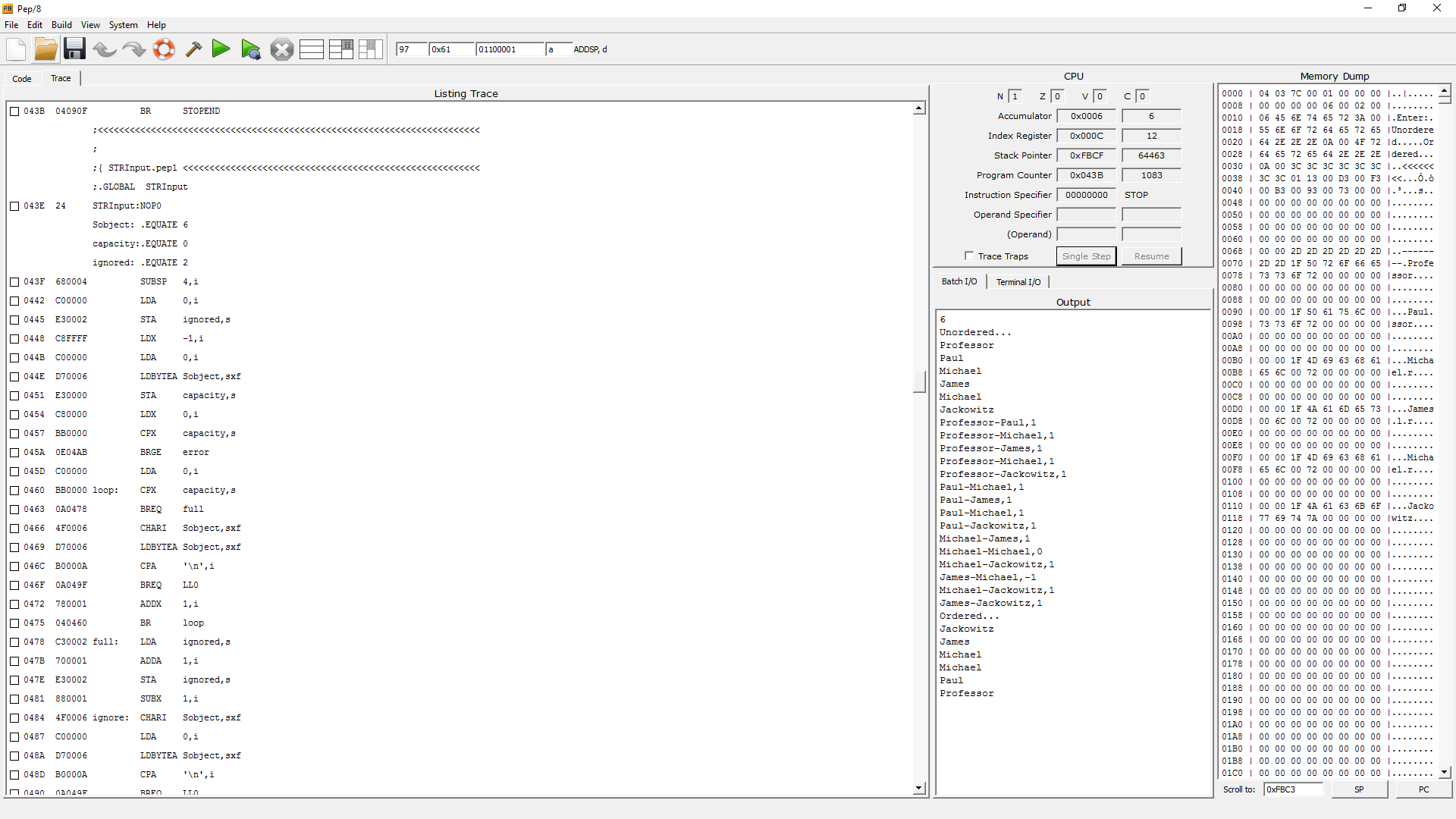
**Restore to Previous State**

Now we restore A and X to their state before sortStgs was called, and add 6 onto the stack because the program has finished



The state of the program after “readStgs” is called, and “prntStgs” is called for the first time. The Output displays “Unordered…” with all the strings printed, meaning “prntStgs” has done it’s job. In the memory dump, we see that all the Strings to sort are after the eight hyphens, which means readStgs has done it’s job and all is set for sortStgs to work

This screenshot is after the first iteration of the outer-loop. We see that it compared “Professor” with every other string, and found that it is greater than every other string. This must mean that *only* it’s address is moved to the end of the sort. If we look at the last memory dump, “73” (the address in hex of “Professor” is at the beginning of the “p” array, denoted by the first “<” chevrons. In this screenshot, “73” is moved to the end of the “p” array, but the position of “Professor” itself in the “a” array has not changed. This must mean that the sort is coming along successfully.

After the rest of the passes of bubble sort, we see that the method successfully sorted the Strings in ascending order.