

The University of Texas At Dallas
Electrical Engineering Department
EE 2110: Beginning Digital Logic and Computer System Fundamentals
Experiment #7 – Developing a More Complex Loop

1. **Introduction:** Laboratory Experiment 7 furthers programming experience by building on the skills developed in Experiments 5 and 6. The programming exercise will still be in the mode where lab teams are encouraged to work outside class to do the programming, either together or separately. It is strongly suggested that student teams meet and compare programs (and make sure that they have a working solution) prior to the lab session.
2. **Goal of this exercise:** Gain further experience in designing a program and in building a program loop. Also, gain additional experience with the SPIM assembler and use of system calls for input/output.
3. **Basis of experiment:** The primary focus of this experimental exercise involves the use of a sorting algorithm programmed into a loop. A series of randomly-arranged letters will be input from the console. The program will then alphabetize the list and output it to the console in alphabetized order. The program provides further opportunity for using a variety of MIPS/SPIM instructions and for using syscalls 4 and 8 and the “.space” directive.
4. **Experimental Equipment List:** The following items are required for this experimental procedure:

- Pervin text for reference.
- SPIM installed on your PC.

5. **Pre-Work:**

- Make sure that you are up-to-date in reading assignments in Pervin and P& per the 2310 syllabus.
- Read the program description below. Make sure that you clearly understand it.
- Make sure that you and your partner know how the program is to work. Flow-chart the program, if necessary, to make sure that you understand the sequence of events as the program executes.
- Both lab partners should work on and attempt to complete the program before coming to lab.

6. **Program Description:** You are to write a program that does the following:

- Analyzes a string of small (non-capital) letters in memory and puts them in alphabetical order. (The sequence could be of most any length up to a line, but for consistency, make the string 12-15 random small letters. The sequence should be labeled “string 1” in the data statements, followed by a “.space” directive to reserve at least 20-30 characters (this is extra room in case you input a few extra characters, so that you will still have a null-terminated string, since “.space” clears all byte spaces to 0).
- Inputs the string from the keyboard using a syscall 8. They should be placed in the reserved space labeled “string 1.”
- Alphabetizes the string using a loop and outputs the alphabetized string to the simulated console.
- Assures all characters in the set are really small letters – not capitals, punctuation, numbers, etc. If a character is incorrect, it should be deleted from the alphabetized list.
- The output of ordered letters is preceded by a statement: “The alphabetized string is: ”

- 7. Comments About Program and Hint:** This is an example of a program which can be done with a recursive loop, such as those demonstrated in class. However, in this case, there is a somewhat easier approach that you can use. Simply start by comparing the characters all to “a” and if there are any “a’s,” store them in a new string (say, “string 2”). Then compare to “b,” “c,” etc. Such a loop will also let you start out each comparison by making sure that the character is a-z. If not, throw it away and do the next compare. When you get to a 0, stop, since your “.space” command has assured that the string is null-terminated. Although you have to go through the entire loop 26 times (a-z), the computer loop is so fast that this is really not a problem. If you really wish to do a sort/compare such as in the homework problem noted above (in which you only go through the loop once, but you go back-and-forth in the loop a great deal!), you are welcome to do so, but the approach described above is easier to implement.
- 8. Experimental Procedure:** Note: As usual, bring a thumb drive to class with your program on it.
- Load your program onto the PC or simply open it from the flash drive.
 - Open SPIM and begin debug. If you have any serious issues or problems, please consult the TA.
 - If the program runs, make sure that the output string is correctly alphabetized.
 - When your program runs correctly, please call the TA to verify correct operation with a demonstration.
 - When your program is verified, you have completed the exercise.
- 9. Software Removal:** The experimental procedure is complete. Please close the PCSPIM program and remove your media. Also check to make sure that your work area is clean before leaving.
- 10. Laboratory Report:** Use the customary laboratory report form. In your write-up, discuss any problem you had getting the program to run, and the actual verification of the correct assembly. Be sure to include a complete print-out of the final program (team members may use copies of the same printout in their report). Also answer the following questions in your report:
- What complexities in the program would be introduced by letting the data input be capital letters also, and including them in the sequencing?
 - What was the biggest difficulty that you had in completing the program?
 - What was the biggest problem that you had with SPIM for this exercise?
 - How do you compare loop generation in assembly language with loops in high-level-language programming?