

**The University of Texas At Dallas**  
**Electrical Engineering Department**  
**EE 2110: Beginning Digital Logic and Computer System Fundamentals**  
**Experiment #8 – Letter Insertion Routine**

1. **Introduction:** Laboratory Experiment 8 furthers your experience by building on the skills developed in prior experiments. It also serves somewhat as a “lab practicum” or final exam to help you determine if you have gained a moderate competence in assembly language programming. There will be no report for this exercise, but you are expected to do a more sophisticated job in designing and coding the program than in your other lab programming tasks. You are still encouraged to work outside class to do the programming, either together or separately. Once again student teams are urged to meet and compare programs (and make sure that they have a working solution) prior to the lab session.
2. **Goal of this exercise:** Move a step up in sophistication by designing a program with a recursive loop.
3. **Basis of experiment:** The primary focus of this experimental exercise involves the use of a recursive algorithm which will place random letters in an already-existing alphabetical list. A single letter will be input from the console repetitively. The program will then insert the character into a list of alphabetized letters defined in the data declaration. The program provides further opportunity for using a variety of MIPS/SPIM instructions and for using syscall 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 computer unit.
5. **Pre-Work:**
  - Make sure that you have completed all software reading assignments (Pervin and P&H).
  - Make sure that you and your partner know how the program is to work (if necessary, flow-chart the program to make sure that you understand the sequence of events as the program executes).
  - Both lab partners should work on and attempt to run the modified program before coming to lab. In particular, this program should be working before you come to class.
6. **Program Description:** You are to write a program that does the following:
  - The program accepts a single-letter keyboard input at a time. The letter inputs must be lower case a through z (no upper-case or non-letter characters allowed; only lower-case letters). Only one letter must be input at a time; make sure that you assure that (a) only one letter is input, and that (b) the character is ONLY a lower-case letter.
  - In the program data declaration, you must declare a string of letters (named “str”) which is a sequence of all 26 lower-case letters in alphabetical order.
  - The program must input the character from the keyboard, determine its place in the string, and make a space in the string for this character, inserting it in the proper alphabetical order in the sequence of letters.
  - After the string of 26 letters, declare a sequence of 30 null bytes (using the “.space” directive). This will be expansion space for the string as you insert characters into the alphabetized list, and the list grows longer. When this expansion space is used up, your program must end. It will also end any time that you input the number “0” (“zero”) instead of a small letter.

- At any point, you can print the current string (i.e., with as many letter inserts as have been made so far) by inputting capital P from the keyboard.
- Finally, this program **MUST** use a recursive routine to insert the letter in the correct point in the program. Actually, using recursion is not the only way to accomplish this programming task, but it does make it relatively straightforward.

**7. Comments About The Program:** This program represents a nice “design” opportunity. It is not as easy as the programs in previous programming labs, but it should not be too difficult if you have been consistently doing class homework and your own programming in the previous exercises. Remember to store the contents of \$ra on the stack each time you jal, since your program may call itself quite a number of times as it tries to find the correct location and make space for each letter.

**8. Experimental Procedure:** Note: 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.
- 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 regular laboratory report form. In your write-up, discuss difficulties 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. Also include the following items:

- What was the biggest difficulty that you had in completing the program?
- What was the biggest problem that you had with SPIM for this fourth exercise?
- How did you determine when you had found the correct position in the string?
- How did the recursive loop help you “keep your place” in the string so that you could easily insert the character properly?
- Did you try single-stepping through part of the program? You might try it for at least one letter placement. As you can see, single-stepping through this program can be somewhat laborious!