ECE 109 Program 1: `ascii.asm`

Spring 2017

Due March 27, 2017 @ 11:45pm

This programming assignment must be completed individually. Do not share your code with or receive code from any other student. The only people who may see your code are the instructor and the ECE 109 TAs.

Evidence of copying or other unauthorized collaboration will be investigated as a potential academic integrity violation. **The minimum penalty for cheating on a programming assignment is a grade of -100 on the assignment.** If you are tempted to copy because you're running late, or don't know what you're doing, you will be better off missing the assignment and taking a zero. Providing your code to someone is cheating, just as much as copying someone else's work.

DO NOT copy code from the Internet, or use programs found online or in textbooks as a "starting point" for your code. Your job is to design and write this program from scratch, on your own. Evidence of using external code from any source will be investigated as a potential academic integrity violation.

This LC-3 assembly language program will print the binary ASCII code for a single character typed by the user.

Details

The program must start at address x3000. Here's how the program must behave:

1. The program prints "`> `", which serves as a prompt to tell the user that the program is waiting for input. The prompt string is a greater-than symbol, followed by a single space.
2. The user types a character on the keyboard. The user does not hit <ENTER> after the character is typed. The user character is printed on the console[1] followed by a linefeed character.
3. The program a string that shows the 8-bit binary representation of the ASCII code for the character. The string must follow exactly the format shown below, where *X* is replaced by the user's character and *bbbbbbbb* is replaced by the ASCII code:

   `The ASCII code for 'X' is bbbbbbbb`

   A linefeed is printed at the end of the string. (Your code will not print in boldface, of course; that's just used for emphasis here.)
4. The program halts.

Do not use subroutines (the JSR instruction) for this assignment. Points will be deducted if you do.

[1] This is called "echoing". It's a way to give feedback to the user that the character was typed. It is not done automatically by the GETC service routine. Your program must do this explicitly. The linefeed in Step 2 is not part of echoing, since the user didn't type Enter, but it's part of the required user interface for this program. (Try it without printing the linefeed, and see how silly it looks.)

Example run (user input is shown in bold):

```
> 6
The ASCII code for '6' is 00110110
```

Another example run:

```
> m
The ASCII code for 'm' is 01101101
```

Hints and Suggestions

- As always, **design before you code**! Draw a flowchart to organize and document your thinking before you start writing the program.
- Use OUT to print a single character to the console. Use PUTS to print a string of characters to the console.
- Use GETC to read a character from the keyboard. If you want the character to be "echoed" to the console, you must also use OUT to print the character after it's read.
- Remember that OUT prints a character, not a value. To print a one (e.g., as part of the binary code), R0 must contain the value x0031 (the ASCII code for '1'), not the value x0001.
- Since we're using the TRAP instruction (for GETC, etc.), don't put any useful values in R7.
- NOTE: You don't need to "convert" the input character to ASCII – it's already ASCII!! You also don't need to "convert" it from ASCII to binary – it's already binary!! You need to figure out which bits of the character are ones and which are zeroes. (HINT: What value can you AND with the character to find out if a particular bit is a one or a zero?)
- There are several different ways to solve this problem. Some ways are easier than others, but the important thing is to solve the problem. The TAs may give you some hints, but you should figure out on your own how to extract and print the bits.
- Use the PennSim simulator and assembler. There are other simulators and assemblers out there, but there are some differences. Your program will be graded using PennSim, and no other tools will be used or considered.
- *Test your program with a wide variety of inputs.*

Administrative Info

Updates or clarifications on Moodle:

Any corrections or clarifications to this program spec will be posted on Moodle, using the discussion forum. It is important that you read these postings, so that your program will match the updated specification. (I strongly recommend that you "subscribe" to that forum, so that you get an email for every posting.)

What to turn in:

- Assembly Language Source file – it must be named **ascii.asm**. Submit via **Moodle**.

  The program will be graded by one of the TAs, and your grade will be posted in the WebAssign gradebook. You will also get back some information about what you got wrong, and how points were deducted (if any).

  DO NOT submit .obj or .sym files. Do not submit a .txt file, a .doc file, or anything like that. It must be a simple text file with the proper .asm extension. If we are unable to open or interpret your file, you will get a zero for the assignment (even if it has the right name!).

Grading criteria:

5 points: File submitted with the **proper name**.

10 points: Flowchart verified by TA in Problem Session 5

20 points: **Program is complete**. There is code to perform every necessary part of the program's function (print the prompt, read the input, print the result).

15 points: Assembles with no warnings and no errors using PennSim. (If the program is not reasonably complete, then only partial credit is available here. In other words, you won't get 15 points for assembling a few comments or trivial lines of code.)

10 points: **Proper coding style, comments, and header**. Use indentation to easily distinguish labels from opcodes. Leave whitespace between sections of code. Use comments and meaningful labels to make your code more readable. Your file **must** include a header (comments) that includes your name and a description of the program. Don't cut-and-paste the description from the program spec – that's plagiarism. Describe the program in your own words.

40 points: The program **performs all functions correctly**:

(5 pts) Prints the prompt string.
(10 pts) Reads and echoes character from the user.
(10 pts) Prints the correct 8-bit binary number.
(10 pts) Prints the output string using EXACTLY the right format. (No extra spaces, characters, linefeeds, etc.)
(5 pts) Program halts after printing string and linefeed. (DO NOT prompt for another character.)