ECE 109 Program 1: `even.asm`

Spring 2021

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 add every even number between two values entered by the user.

Details

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

1. The program sends a newline to the console and then prints "Enter Start Number (0-16): ", which serves as a prompt to tell the user that the program is waiting for input. The prompt string is a string, followed by a single space.
2. The user types a one- or two-digit number (0-16) on the keyboard. As the numbers are typed on the keyboard they should be echoed onto the console window. Any illegal input values are ignored after they are echoed. The program must check and accept only numerical digits. The program should recognize the entry as complete if it sees: [a] 1 valid digit plus a return, or [b] 2 valid digits, return not checked here.
3. The program sends a newline to the console and then prints "Enter End Number (0-16): ", which serves as a prompt to tell the user that the program is waiting for input. The prompt string is a string, followed by a single space.
4. The user types a one- or two-digit number (0-16) on the keyboard. As the numbers are typed on the keyboard they should be echoed onto the console window. Any illegal input values are ignored after they are echoed. The program must check and accept only numerical digits. The program should recognize the entry is complete if it sees: [a] 1 valid digit plus a return, or [b] 2 valid digits, return not checked here.
5. If the user types the character "q" at any time the program should display a newline and the string "Thank you for playing!" and then Halt.

6. The program adds every even number (inclusive of the start and end) between the values that have been entered on the keyboard. The program then converts this binary sum to a two-digit decimal number and converts the digits to ASCII code.
7. The program sends a newline to the console and then prints the following on the console:

   The sum of every even number between the two numbers is: zz

   where zz is the two-digit sum. A linefeed (newline) is printed at the end of the string. (Your code will not print in boldface, of course; that's just used for emphasis here.) Leading zeroes shall NOT be printed in the value.
8. The program returns to step 1.

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

Example run (user input is shown in bold):

```
Enter Start Number (0-16): 3
Enter End Number (0-16): 16
The sum of every even number between the two numbers is: 70
```

Another example run:

```
Enter Start Number (0-16): 2
Enter End Number (0-16): 8
The sum of every even number between the two numbers is: 20
```

Another example run:

```
Enter Start Number (0-16): 3
Enter End Number (0-16): q
Thank you for playing!
```

Another example run:

```
Enter First Number (0-16): ag3
Enter Second Number (0-16): az1bx0
The sum of every even number between the two numbers is: 28
```

Another example run:

```
Enter First Number (0-16): 4
Enter Second Number (0-16): 4
The sum of every even number between the two numbers is: 4
```

Hints and Suggestions

- As always, **design before you code**! Draw a flowchart to organize and document your thinking before you start writing the program. Remember the 80/20 Ratio!!
- Write one section of the program at a time. Then **TEST** it. Then add the second part. And so on.
- 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 the digits entered on the keyboard are received in ASCII code. These must be converted to binary before adding them.
- 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.
- 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. You must load the file **lc3os.obj** into Pennsim when loading your user code. 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.* The TAs will chose a set of start/next values to check the function of your code.

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 **even.asm**. Submit via **Moodle**.

  The program will be graded by one of the TAs, and your grade will be posted in the Moodle 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:

10 points	Detailed, one full page, **flowchart should be submitted to Moodle by the date shown!** This flowchart should be one full page maximum. Enough detail should be included to understand your methods. This chart must be computer created (.ppt, etc.); no hand-drawn charts will be accepted. Please ask your Problem Session TA for assistance if you need it.
5 points:	File submitted with the **proper name**.
10 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, submission date 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.
50 points:	The program **performs all functions correctly**: (5 pts) Prints the correct prompt strings for Start and End Numbers (10 pts) Reads and echoes characters from the user. Ignores characters other than 0-9, and q. (10 pts) Properly converts the Start and End characters to binary values (12 pts) Prints the correct sum of even numbers of the entered numbers. (3 pts) Blanks any leading zeroes in the output sum. (5 pts) Prints the output string using EXACTLY the right format. (No extra spaces, characters, linefeeds, etc.) (5 pts) Program detects the "q" and halts after printing string and linefeed. (DO NOT prompt for another character.)