CSC 230 Assignment 3

Due Sunday, **November 21**, 2019 at 11:55pm

Late submissions will not be graded.

Overview

The goal of this assignment is to implement the Collatz sequence explorer. We have already seen the Collatz sequence in Assignment 1, where we counted the number of elements in the sequence. To do this, we implemented an algorithm which computes each value in the sequence and counts how many there are. For this assignment, in addition to counting them, you are also expected to output each value and its position (or count) in the sequence on the liquid crystal display (LCD).

The initial starting value for the sequence will be obtained from the user via the buttons on the LCD Shield (as described in Specifications below). Upon user confirmation, the device should display the initial value, whatever it is, and its count of 0 at the designated locations on LCD. After a short time-delay, the device will advance to the next value and display its corresponding count of 1. The value and its count will continue advancing with a predetermined time-delay between them until the value reaches 1, at which point the last value and its count will remain on the screen until the user selects another value. The time-delay period will be chosen by the user, as described in Specifications below.

While the Collatz sequence is advancing on the screen, the user should be able to continue to interact with the input prompt using the LCD shield buttons. At any time, when the user changes the speed setting, the time-delay should be adjusted accordingly and immediately. Similarly, when the user changes the starting value and confirms the selection, immediately upon confirmation, the sequence display will update with the new starting value and start advancing from there.

This assignment is composed of several sub-problems that can be completed and tested separately before combining them into the final solution. This assignment builds on the previous assignments and labs. You are welcome to use your own previously written code or the solutions provided to you on conneX during this course. If you use any of the solutions from conneX, you should reference them appropriately in the comments. For example, if you copied and modified the init_to_string function from Lab 7, then you could say: "This function is a modified version of the function int_to_string from Lab 7, CSC 230, Fall 2019" in the comments.

Specifications

- **Program start:**

  When the device starts, or when the reset button is pressed, the LCD screen should display your first and last name and the phrase "CSC 230 - Fall 2019" for a period of 1 second. Then, the screen should change to the same as shown in the picture below. The cursor should be set to the least significant digit of the initial Collatz value ("n="). The speed should be initialized at 0. The initial Collatz sequence value should be initialized at 0, and its count should be initialized at 0.

- **The LCD screen layout:**

	n	=	**0**	**0**	**0**	*****				S	P	D	:	**0**	
c	n	t	:			0		v	:						0

The input fields are highlighted using boldface.

  - **Top row** is designated for prompting the user for the input.
    - The first three digits (initialized as zeroes), which are next to the "n=", are designated for inputting the new starting value of the Collatz sequence to be displayed. These digits are updated one at a time when the cursor is on each of them and the user presses UP or DOWN buttons on the LCD shield. For example, if the cursor is on the first of the three digits and that digit is currently showing as "5", pressing UP will change it to "6". This initial starting value could range between 0 and 999.
    - The asterisk (*****) is for confirming the starting value for the new Collatz sequence. When the cursor is on the asterisk and the user presses UP or DOWN, the current value and its count will be updated immediately to the new value and its initial new count of zero. Then it will continue advancing (or not) according to the current speed setting.
    - The last digit (also initialized as zero), which is next to the "SPD:" is for controlling the amount of time to wait before advancing to the next value in the given Collatz sequence. This speed digit is updated when the cursor is on it and the user presses UP or DOWN in a similar fashion as the first three digits described above. This speed value could range between 0 and 9.

  - **Bottom row** is designated for displaying the current value and its count in the given Collatz sequence.
    - The numbers are periodically updated to the next value and its count based on a user-determined time-delay.

- The time-delay between advances to the next value is determined by speed value, which is displayed in the first row of the LCD screen and explained further below.
- The maximum possible count could be 3 decimal digits long and the maximum possible value could be 6 decimal digits long, hence the corresponding space provided between "0" and ":" on the second row in the picture above.

Suggestion: Store two strings of length 17[1] in memory and periodically update the LCD with those strings (lcd_puts function). To change the message being displayed, update these strings (e.g. via ISR), and not the LCD, as it will get updated as per previous sentence.

- **The cursor:**
  - Since we have several values that can be updated by the user via the same (shared) set of input controlls (UP and DOWN buttons), we need some way to indicate specifically which value the user is about to change. To do that, we keep track which one is currently being edited and make the corresponding location on the LCD blink. The blinking effect can be achieved by repeatedly displaying the blank (space) character for a period of time and then the actual character that is supposed to be in that location for a period of time.
  - The user should be able to press the LEFT and RIGHT buttons to move the cursor between the three digits for the initial Collatz value, the asterisk, and the sped value. So, 5 possible positions in total. For example, if the user navigated to the asterisk (*), the asterisk would blink; then, if a user was to press the RIGHT button, the cursor would advance to the speed selection and the sped value would start to blink instead of the asterisk.

- **The starting value:**

  The initial value for the next Collatz sequence to be computed is obtained from the user.

  - Input is in range between 0 and 999 (three decimal digits).
  - Provided by the user via the UP and DOWN buttons. Each digit is updated when the cursor is on that digit (on the LCD).
  - Specifies the starting value for the next Collatz sequence.

- **The delay/speed:**

  The frequency at which the new Collatz value is computed and displayed is obtained from the user.

---

1. 17 = 16 for each LCD character in one row + 1 for the terminating zero.

  - Input is in range between 0 and 9 (one decimal digit).
  - Provided by the user via the UP and DOWN buttons, when the cursor is on the speed digit (on the LCD).
  - Specifies the delay between Collatz sequence value updates:
    1 = 1/16 sec. (max. speed of advancing to the next value)
    2 = 1/8 sec.
    3 = 1/4 sec.
    4 = 1/2 sec. (= 2 Hz)
    5 = 1.0 sec. (= 1 Hz)
    6 = 1.5 sec.
    7 = 2.0 sec. (= 0.5 Hz)
    8 = 2.5 sec.
    9 = 3.0 sec.
    0 = full stop (no advancing when speed is 0)

- **Limits:**
  - Max size of Collatz value ("v:"): 3 bytes.
  - Max size of Collatz value's count/position ("cnt:"): 1 byte.
  - Decimal range of the displayed value ("v:"): 0-999999.
  - Decimal range of the displayed count ("cnt:"): 0-255.
  - Decimal range of the speed prompt ("SPD:") : 0-9.
  - Decimal range of the starting value prompt ("n="): 0-999.

Additional notes and resources

In this assignment you will need to:

- Add 24-bit numbers.
- Display characters on the LCD screen.
- Check which button is pressed on the LCD shield. Both the solution to Lab 4 and the solution to Assignment 2 have a suitable starting function that can be modified to work for this assignment.
- Convert between ASCII characters and binary integers that are as large as 3 bytes (or 2^{24}). Lab 7 has a good starting algorithm that needs to be extended to handle larger values.
- Use at least one timer and interrupts for controlling the frequency at which the Collatz sequence advances from one value to the next. Lab 8 has a timer-driven interrupt example.
- Online AVR Timer/Counter calculator: https://eleccelerator.com/avr-timer-calculator/.
- Online Collatz sequence calculator: https://www.dcode.fr/collatz-conjecture.

Grading guidelines.

This assignment is worth 9% of your total grade, this value is distributed among the following categories:

⇒ All functions must protect (back-up or preserve) registers.

3% ⇒ Correct interrupt service routine (ISR). The timing speeds up and slows down as per specifications. Zero speed results in a complete stop. This component requires knowledge of interrupts, timers, functions, and register protection.

1% ⇒ At least one function written by you must receive and return a parameter via the stack. This component requires knowledge of functions and stack.

5% ⇒ User interface. Marks are distributed in various proportions among the following:

- Credits screen (approximately one second).
- Cursor blinking and moving based on user input.
- Buttons operate as per specifications. Buttons are responsive. Holding down a button doesn't result in multiple presses being registered.
- Converting binary to ASCII or vice versa.
- Correct Collatz values and counts (on LCD).

BONUS (1% to the total course grade) if your program uses a second ISR and a separate Timer/Counter for checking the button input, this has a prerequisite that the buttons operate as per specifications and are responsive, and that holding a button doesn't register as multiple presses.

Submission

Submit your solution via conneX. When doing so, verify that your file is actually uploaded correctly and is not corrupted. You can do so by navigating back to the Assignments section, then downloading your `a3.asm` submission (which you just uploaded), opening it in an editor and visually verifying its contents.

It must be possible to build and run your program on the equipment provided in the labs (using the same procedure discussed in the lab sessions), otherwise your solution will not be graded.

The solution is worth 9% of your final grade and **must be your individual work**. You may discuss the assignment with your fellow students, but you must write your own code from scratch. Sharing code in any way (or receiving shared code), either electronically or over the shoulder of another student, will be considered plagiarism, even if the code is modified after being shared.

Late submissions will not be graded.