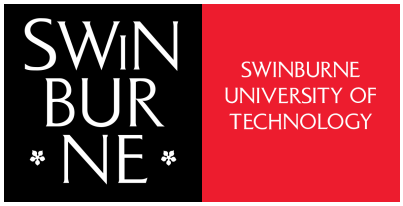

Swinburne University of Technology

Stopwatch Program – Solution Overview and User Guide

COS10004
Computer Systems

Name : Marco Giacoppo
Student ID : 104071453
Video Link : https://youtu.be/YNqemCEYCpM

Table of Contents

Overview

The Stopwatch Program is designed to function as a digital stopwatch with minute and second counters. It runs on an ARM assembly language platform which is called *ARMlite Simulator V1.2.4 © Peter Higginson 2020-23.* It utilizes memory addresses to store and manipulate the values. The program allows users to start and stop the stopwatch, as well as reset and split the time. The stopwatch time is displayed using a simple text-based interface. It is displayed in the text output screen and it uses the 'NewLine' function that makes it look like it refreshes every second. In my solution, I put all the time in 1 register which is R1, this way I only need to str/ldr myself. At first, I used a bunch of registers to store the time value, this made me 'push' the registers multiple times making my code unclear and it's technically memory usage as I read in the discussion board.

Solution Design

The program is structured using subroutines for different functionalities. Here is a breakdown of the main subroutines and their purposes:

- 'secondOnes'
  - Increments the secondOnes digit by 1.
  - Checks if it reaches 10, moves to the next digit (secondTens).
  - Calls the 'Print' subroutine to display the updated time.
  - Handles the pause and reset actions based on user input.
  - Calls the 'delay' subroutine to introduce a delay.

- 'secondTens'
  - Resets the secondOnes digit to 0.
  - Increments the secondTens digit by 1.
  - Checks if it reaches 6, moves to the next digit (minuteOnes)
  - Calls the 'delay' subroutine to introduce a delay.

- 'minuteOnes'
  - Resets the secondTens digit to 0.
  - Increments the minuteOnes digit by 1.
  - Checks if it reaches 10, moves to the next digit (minuteTens)
  - Calls the 'delay' subroutine to introduce a delay.

- ‘minuteTens’
  - Resets the minuteOnes digit to 0.
  - Increments the minuteTens digit by 1.
  - Checks for the condition to stop the stopwatch.
  - Calls the ‘delay’ subroutines to introduce a delay.

- delay
  - Implements a delay by calculating the elapsed time between iterations.
  - Uses a loop to wait until the required delay time is reached.

- Print
  - Displays the current stopwatch time by accessing and printing the values stored in memory addresses.
  - The split time values are retrieved from the memory locations ‘minTens’, ‘minOnes’, ‘secTens’, ‘secOnes’, and the are printed after the elapsed time.
  - Utilizes string constants for the colon separator and newline characters.

- Pause and Reset
  - Handle user input to pause and reset the stopwatch.
  - Check for specific key codes (R for reset, P for pause) and perform the corresponding actions.
- Split
  - Handle user input to split the time.
  - Check for specific key code (S) and perform the action by taking the split time values that are stored in the memory locations ‘minTens’, ‘minOnes’, ‘secTens’, ‘secOnes’.

Instructions how to Run and Use the Program

To run and use the Program, follow these steps

- Set up the ARM assembly language platform (ARMlite simulator).
- Load the program into the ARMlite simulator environment.
- Start the program execution.
- The stopwatch will start automatically and display the time on the text output screen.
- To pause the stopwatch, press the 'P' key. Pressing the 'P' key again will resume the stopwatch.
- To reset the stopwatch, press the 'R' key.
- The stopwatch will stop automatically when it reaches 99:99.
- The current stopwatch time will be displayed on the screen, with a colon separator between minutes and seconds.
- After stopping or resetting, you can start the stopwatch again by pressing ‘P’ key.

Issues/ Bugs

- After stopping the program and running it again, the stopwatch resumes counting from the previous time instead of starting from zero. The expected behaviour is for the stopwatch to reset and begin counting again when restarted.
- When the split button ‘S’ is pressed, the split time is not displayed instantly. Instead, it requires waiting for the next time cycle (1 second) for the split time to appear. The desired behaviour is to display the split time immediately upon pressing the split button.
- When the pause button ‘P’ is pressed, the pause functionality is not run instantly. It also takes 1 second cycle to implement. I’m guessing there’s a more efficient way of handling this.

Conclusion

In conclusion, the current implementation of the stopwatch program has shown some issues and bugs that need to be addressed. The timer continuation issue prevents the timer from resetting when the program is stopped and restarted, resulting in the timer continuing from the previous time. This behaviour deviates from the expected functionality where the timer should start from zero upon restarting.

Additionally, the delayed split time display poses a usability issue. When the split button 'S' is pressed, the split time does not appear instantly but rather waits for the next time cycle (1 second) before displaying. This delay hinders the user's ability to accurately track split times during timing sessions.

To enhance the program's usability and address these issues, modifications should be made to ensure that the timer resets to zero when stopped and restarted. Furthermore, the split time display should be updated immediately upon pressing the split button, providing users with real-time split time information.

By addressing these issues, the stopwatch program can offer a more reliable and user-friendly experience, meeting the expectations of users who rely on accurate timing and split time measurements.