Project Report: Stopwatch Program in Assembly Language using Emu8086

Group Members :

* Adil Butt-076
* Muhammad Fayyaz-077
* Ali Haider Tahir-058



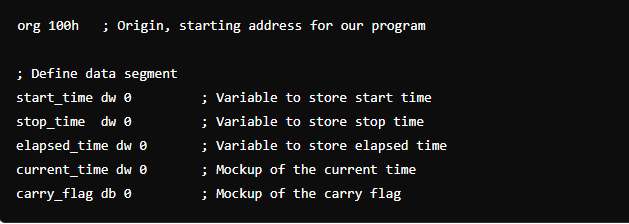
**Introduction**

This project demonstrates a simple stopwatch program implemented in Assembly language using Emu8086. The program involves initializing, starting, stopping, resetting, updating, and displaying the elapsed time using various registers, flags, memory addresses, labels, and variables.

**Code Explanation**

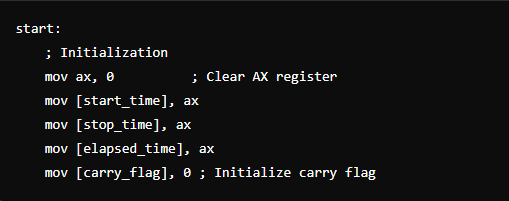
**Data Segment**

The data segment is used to define variables that store the start time, stop time, elapsed time, current time (for simulation purposes), and a carry flag to indicate whether the stopwatch is running.



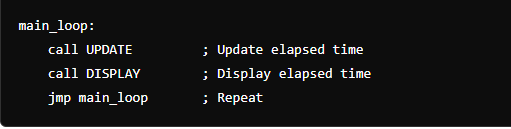
**Initialization**

The **start** label initializes the variables and the carry flag.



**Main Loop**

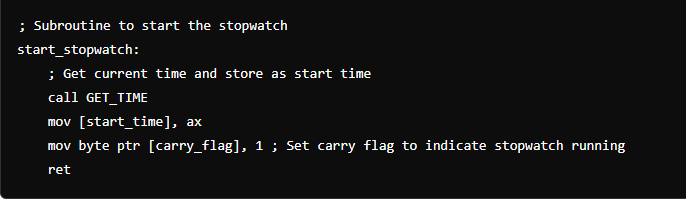
The **main\_loop** continuously updates and displays the elapsed time.



### Subroutines

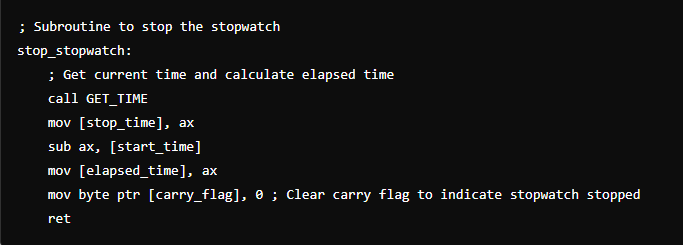
#### Start Stopwatch

The **start\_stopwatch** subroutine records the current time as the start time and sets the carry flag to indicate that the stopwatch is running.



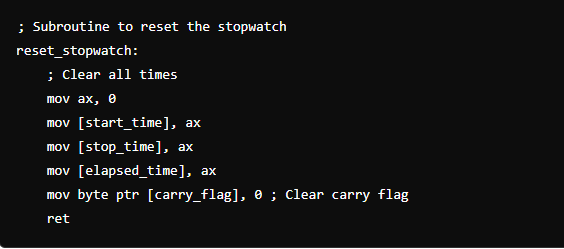
#### Stop Stopwatch

The **stop\_stopwatch** subroutine records the current time as the stop time, calculates the elapsed time, and clears the carry flag to indicate that the stopwatch is stopped.



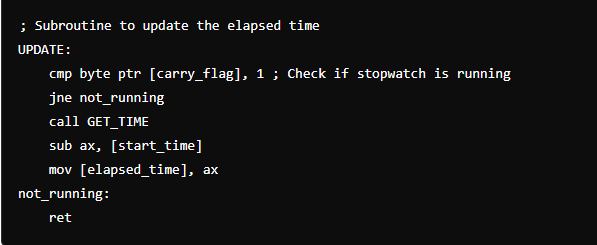
#### Reset Stopwatch

The **reset\_stopwatch** subroutine clears all the time variables and the carry flag.



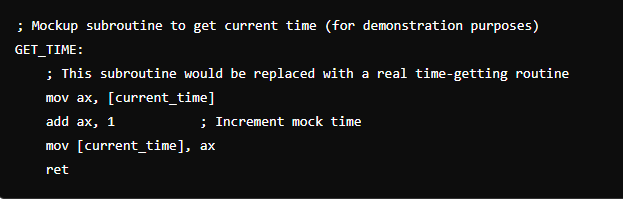
#### Update Elapsed Time

The **UPDATE** subroutine checks if the stopwatch is running (using the carry flag) and updates the elapsed time.



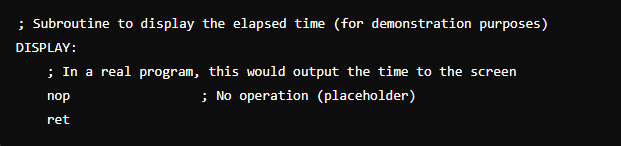
#### Get Current Time

The **GET\_TIME** subroutine simulates getting the current time by incrementing a mock current time variable.



#### Display Elapsed Time

The **DISPLAY** subroutine is a placeholder for displaying the elapsed time. In a real program, this would output the time to the screen.

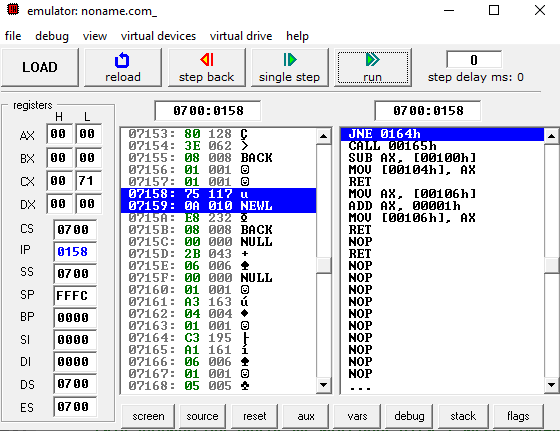


**Flow of the Program**

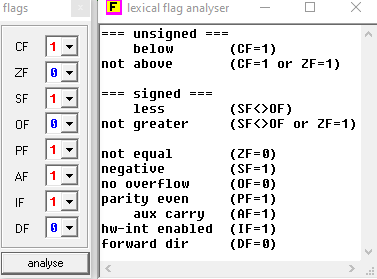
1. **Initialization**:
   * The program starts by initializing all time variables and the carry flag to 0.
2. **Main Loop**:
   * The main loop continuously calls the **UPDATE** and **DISPLAY** subroutines to keep the elapsed time updated and displayed.
3. **Starting the Stopwatch**:
   * The **start\_stopwatch** subroutine is called to record the current time as the start time and to set the carry flag.
4. **Stopping the Stopwatch**:
   * The **stop\_stopwatch** subroutine is called to record the current time as the stop time, calculate the elapsed time, and clear the carry flag.
5. **Resetting the Stopwatch**:
   * The **reset\_stopwatch** subroutine is called to clear all time variables and the carry flag.
6. **Updating Elapsed Time**:
   * The **UPDATE** subroutine checks if the stopwatch is running by examining the carry flag. If it is running, it updates the elapsed time by subtracting the start time from the current time.
7. **Getting Current Time**:
   * The **GET\_TIME** subroutine simulates getting the current time by incrementing a mock time variable.
8. **Displaying Elapsed Time**:
   * The **DISPLAY** subroutine is a placeholder for displaying the elapsed time on the screen.

**Screenshots:**

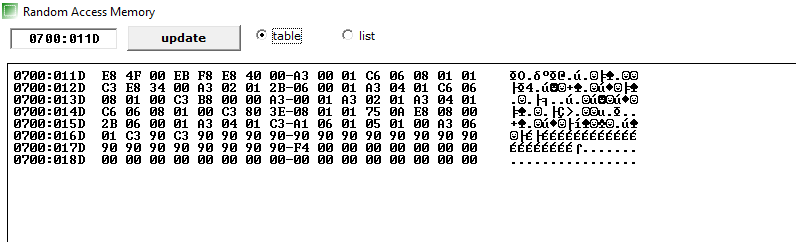
**Registers:**

****

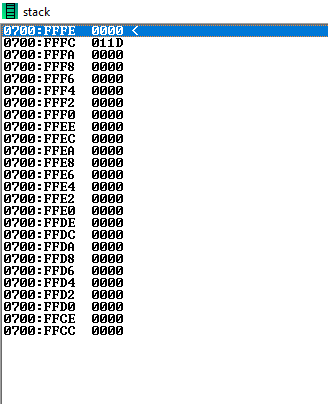
**Flags:**

****

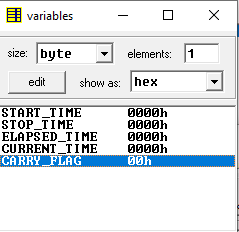
**Memory:**

****

**Stack:**

****

**Variables:**

****

**Conclusion**

This project demonstrates the implementation of a simple stopwatch using Assembly language in Emu8086. It covers initialization, starting, stopping, resetting, updating, and displaying the elapsed time using various assembly language constructs. This example can be expanded to include real-time hardware interactions and more sophisticated display routines for a complete stopwatch application.

This project report provides a comprehensive overview of the stopwatch program's flow and functionality, along with detailed comments in the code to explain each part.