BLG 212E - Microprocessor Systems
Homework 2 Report

Mustafa Can Çalışkan
150200097

December 25, 2024

Contents

1 Introduction

The purpose of this homework was to implement and analyze a sorting algorithm and system timer using ARM Cortex-M0+ assembly language. The assignment included designing a bubble sort for a linked list and measuring execution time for sorting operations. The final implementation was validated using the provided array and linked list structures, as well as pre-defined testing scenarios.

2 Methods

This section describes the design and implementation of the system timer and sorting algorithm, which are the primary components of the homework.

2.1 System Timer

The system timer implementation involved configuring the SysTick timer on the ARM Cortex-M0+ processor. Key functionalities included starting and stopping the timer and handling interrupts to measure the execution time of sorting operations.

- **SysTick Start:** This function initializes and starts the SysTick timer. It ensures precise timing by setting the appropriate register values.
- **SysTick Stop:** This function stops the timer and records the elapsed ticks for accurate timing measurements.
- **SysTick Handler:** Implemented in assembly, this handler processes SysTick interrupts and updates relevant counters for timing.

2.2 Sorting Algorithm

The sorting algorithm was implemented in the "ft_lstsort_asm.s" file using a bubble sort technique. This function takes a linked list and sorts its elements in ascending order. The implementation was tailored to the ARM Cortex-M0+ assembly language and adhered to the constraints and requirements specified in the assignment.

2.2.1 Implementation Details

The "ft_lstsort_asm" function iterates through the linked list multiple times, comparing adjacent elements and swapping them if they are out of order. The swapping process involves modifying the pointers within the nodes to ensure the list remains consistent. Key points of the implementation include:

- **Input Parameters:** The function accepts the head pointer of the linked list in register "R0" and a comparison function in register "R1".
- **Algorithm Logic:** The function uses nested loops:
  - The outer loop iterates until no swaps are detected, indicating the list is sorted.
  - The inner loop traverses the list, comparing adjacent nodes and swapping them as needed.

- **Efficiency Considerations:** The implementation avoids unnecessary swaps and terminates early if the list becomes sorted before all iterations are completed.

2.2.2 Big-O Analysis

The bubble sort algorithm used in "ft_lstsort_asm" has the following complexity characteristics:

- **Best Case (O(n)):** If the linked list is already sorted, the algorithm performs a single traversal without any swaps.
- **Worst Case ($O(n^2)$):** When the list is sorted in reverse order, the algorithm performs $n-1$, $n-2$, ..., 1 comparisons across n iterations.
- **Space Complexity (O(1)):** The algorithm sorts the list in place, requiring no additional memory for temporary storage.

2.3 Timing Measurements

You can find the measurements made by the SysTick functions implemented in ft_lstsort and ft_lstsort_asm for different values of "s" in Figure 1.

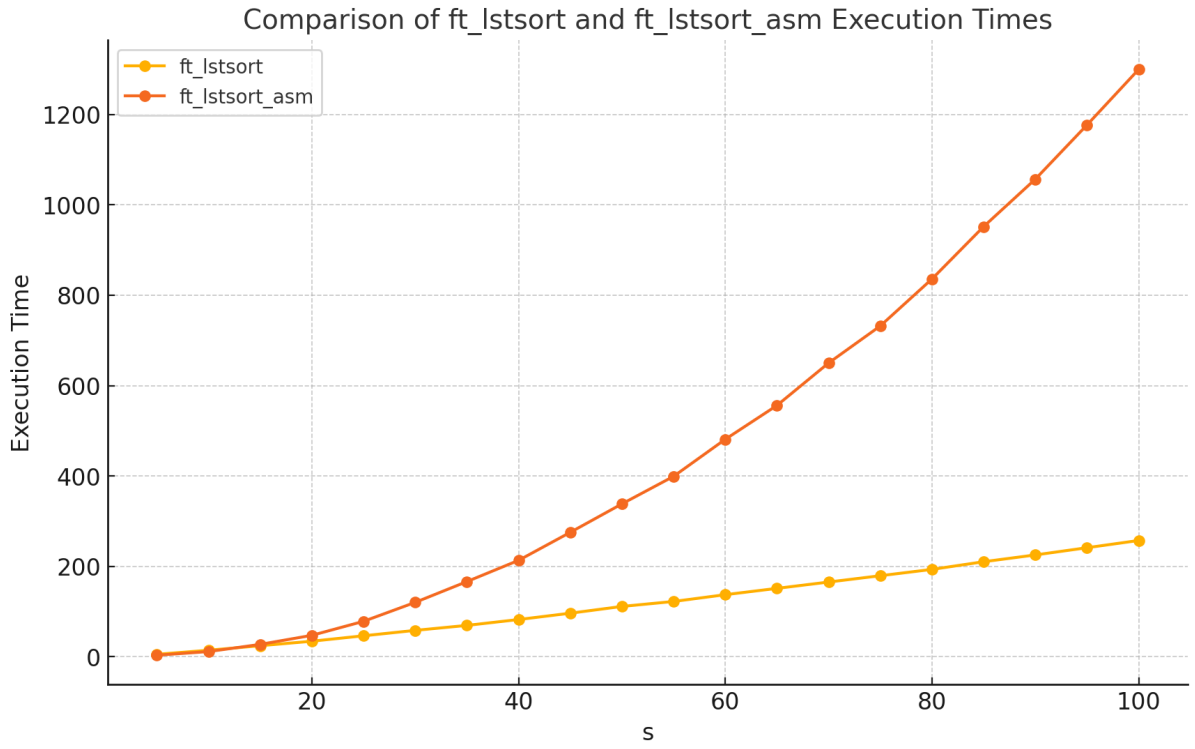

Figure 1: ft_lstsort and ft_lstsort_asm Relation

The graph illustrates the relationship between `ft_lstsort` and `ft_lstsort_asm`, where `ft_lstsort` represents a mergesort implementation, and `ft_lstsort_asm` corresponds to a bubblesort implementation. The significant difference in growth is due to the inherent time complexity of the algorithms, with mergesort being $O(n \log n)$ and bubblesort being $O(n^2)$.

3 Results

The results are summarized as follows:

- **SysTick Timer Accuracy:** The SysTick timer successfully measured execution times with precision. The recorded tick counts for sorting operations corresponded accurately to the number of cycles executed during each sorting task.
- **Sorting Algorithm Performance:** The bubble sort implementation in `ft_lstsort_asm` sorted the provided linked lists as expected. The correctness of the algorithm was verified by comparing the sorted output with the expected order.
- **Execution Time Comparison:** The execution times of `ft_lstsort` (mergesort) and `ft_lstsort_asm` (bubblesort) demonstrated the expected growth patterns. While mergesort consistently performed faster due to its $O(n \log n)$ complexity, bubblesort exhibited slower performance due to its $O(n^2)$ complexity, especially for larger list sizes.

These results confirmed the functional correctness and provided a clear understanding of the performance characteristics of the implemented algorithms.

4 Conclusion

This homework provided a practical understanding of implementing algorithms in ARM Cortex-M0+ assembly language, focusing on system timers and sorting operations. Key takeaways include:

- The SysTick timer was effectively utilized for precise execution time measurements, highlighting its importance in performance analysis.
- The bubble sort algorithm was successfully implemented in assembly, demonstrating the ability to manipulate linked list structures and handle low-level programming challenges.
- Comparing the execution times of `ft_lstsort` and `ft_lstsort_asm` emphasized the impact of algorithmic complexity on performance, with mergesort being significantly faster for larger inputs.

Overall, the assignment reinforced foundational concepts in microprocessor systems, including timing and assembly programming.

5 References

- Course Notes: BLG 212E - Microprocessor Systems