CS2002 Practical 2 - x86 Assembler

Matriculation Number: 160001362

27/02/2018

Contents

1 Overview

This practical specified the commenting of AT&T x86 64-bit assembly code, the implementation of a print function to print three stack frames during the execution of a sort algorithm (and to write an analysis of the output), and an analysis of the functions of sort0.s (and how sort1.s, sort2.s, and sort3.s differ from this file). The practical also specified that as an extension the sort.c source file should be compiled for other architectures such as MIPS and ARM, and that the results should be analysed and compared with x86-64 assembly.

The first stage of the practical has been achieved with the assembly in the sort-commented.s file having been commented for the sortsub function.

The functionality requested by the second stage of the practical has been implemented in the stack.c file in the print_stack and print_frame functions which can be run in terminal by using the command ./main-plus which will print the address, offset and value of each value in each stack frame. Analysis of the results of this function can be found in the analysis section further into this report.

The third section of this practical, analysis of the functions of sort0.s and a comparison between this file and the other sort files, can also be found in the analysis section of this report.

The extension comparing MIPS and ARM assembly with x86-64 assembly is discussed under the 'extensions' heading of this report.

2 Design and Implementation

In the second section of the practical, the stack_frame function runs an in-line assembly command to move the value stored in the base pointer register (rbp) into a local variable. This variable is then dereferenced to retrieve the value of the base pointer from the previous stack frame. My own function print_frame is then called with the two base pointers and the number of stack frames to print as arguments.

The print_frame function then iterates over the addresses between the previous base pointer and the current base pointer in increments of 8 bytes and prints out the address of each value, as well as its offset from the base pointer in bytes, and the value stored at the current address. The print_frame function is then recursively called until there are no more frames to print.

3 Analysis

3.1 Section 2 - Findings

3.1.1 Comparison of sort0.s and sort-plus.s

3.2 Section 3

3.2.1

3.2.2

3.2.3

3.2.4

4 Evaluation

5 Conclusion

6 Extension