ECE-251 Project#1 Documentation

Instructor: Professor Mohammed Billoo
Di Mei, Zhihao Wang

Introduction

In this project, an ARM assembly program is presented and it is required to run in the Raspberry Pi assembler. This assembly program asks for inputs of two strings and then concatenates these two strings together if each string contains no more than 10 characters (10 bytes). This program will also return the size of the concatenated string. If the first input string exceeds 10 characters, the program will stop running and exit with an error code of "21". If the second input string exceeds 10 characters, the program will stop running and exit with an error code of "22". In addition to the source code, "Makefile" and "README" files are also provided. The "README" file gives instructions on how to use the "Makefile" file to execute the program.

Architecture

The program is composed of three parts: ".data", ".text", and ".global". In the ".data" part, all variables are 4-byte aligned and for the variable "string_read", where we store the concatenated string, we use ".skip 100" to allocate enough space for this array of characters. In the ".text" part, we have the function "str_length", which returns the length of a string. We make this function because for the whole program, the length of the string will be calculated for two times if each string's length is less than 11. To increase the conciseness of the code, we create this "str_length" function.

In the ".global" part, it includes a main function. This main function scans (by using "scanf") the first input string and then outputs the length of that string. If the length doesn't exceed 10, it will scan the second string and directly concatenate these two strings together. If any string's length exceeds 10, the branch "b" will bring the program to return the error code. Else, this program will print (by using printf) the concatenated string and return the its length.

Challenges

The first major challenge we faced was to output a string. We first failed with printf by loading "string_load_addr" into r1 and loading [r1] into r1. Since we were outputting a string, we didn't need to dereference r1. After fixing that bug, we chose to use scanf and printf for the input and output functions. The next challenge we faced was to write the "str_length" function in ".text". We weren't able to store the output at first. We then figured out how to

call argument and store the output with “str_length” function which made the code more concise. To set the limits of ten characters per input string, we chose to use three branches “error1”, “error 2”, and “end” to return different error codes, so that there would not be bugs when the user input over 10 characters.

README

The README file was attached with the source code and Makefile.