Electrical and Computer Engineering
Aristotle University Of Thessaloniki

Microprocessors and Peripherals
Report 1

Meliopoulos Marios #9680
Papadakis Konstantinos Fotios #10371
18/4/24

1.1 Assignment Info

This assignment is comprised of 4 files
- The main body in C language using UART serial communication(main.c)
- A hashing function in ARM Assembly language(hash.h)
- A multi-digit to single-digit converter function in ARM Assembly language(sindig.s)
- A sum of consecutive integers function in ARM Assembly language(sum.s)

1.2 Problems

The main problem we encountered working on this project was the inability to easily find resources for questions and problems we encountered. ARM assembly documentation online is limited compared to other languages and information is generally not concentrated in a single space for easy access and reference. Additionally the lack of examples provided by the instructors written in assembly and C that we could use as a reference for getting started along with information that isn't as easily approachable for beginners added an extra level of difficulty. Also UART and the terminal Tera Term we use to gain access to the messages being conveyed through serial communication was very daunting at first. But with the help of the instructors we were able to set them up successfully. Nonetheless we believe it would be for the better to have a more streamlined tutorial on Keil and UART including all the necessary settings and procedures leading to and a basic overview of what each setup setting does.

1.3 Testing

For the purposes of testing the function of our code we used both localized tests to make sure that each function works and then tests that tested the flow from top to bottom. During each function's runtime we used the debug mode and breaking points to monitor the values of variables and registers and the contents of the memory space we allocated. Giving a variety of different input strings ranging from capital to lowercase letters, numbers, operators symbols and other characters and by comparing values with the expected values we calculated and made sure each step of the code had the desired effect. Finally we repeated these tests using Tera Term and UART after acquiring the NUCLEO board.

2.1 hash.s

This function takes a string input and finds its hash value through the use of a lookup table that links certain characters with values. Then each character depending on its type will affect the hash value in a specific way.

The general idea of the function follows a “compare and branch” philosophy. The system takes one character at a time and tries to figure out what category of character it is, based on its ASCII value, to find out how it will affect the hash. After that a branch is executed to a block of code that manipulates the hash accordingly. Then the code loops again at the start to check the next character in the string input and so on until the end of the given string where the hash value is returned.

2.2 sindig.s

The multi-digit to single-digit converter function, as the name implies, takes the multi-digit result from the hashing function we saw earlier and it adds its digits together. This process repeats until we are left with a single digit. Initially we figure out the quotient and the remainder of the division (hashing_result / 10). The quotient in our case is the hashing_result with one less digit in the end and the remainder is the digit we removed to handle separately. It should be noted that to find the remainder we have to subtract the result of the multiplication, between the quotient and the divisor, from the dividend. Now, having calculated the remainder we add it to a register that symbolizes the digit sum initially set to zero. While looping, we add up all the remainders of the divisions that are about to follow and store them inside r0 as output. We then take the quotient of the division and if it’s not zero then there are more digits to add to the sum and we repeat this process until it is. We store the sum to use as the new number that is going to undergo the same process if it is bigger than 9, since that would mean we still have a multi-digit number in our hands. Finally if the sum is a single digit <9 we return r0, where we stored the sum.

2.3 sum.s

The “sum of consecutive integers” function is really simple. We just create a loop which adds our counter register to our sum register each iteration. The counter and the sum are initialized as zero. Each iteration we add the counter to the sum and then increment the counter by one. When the sum reaches the number of which we want to find the “sum of consecutive integers” we break out of the loop and return the sum.