Computer Organization Lab 1: Basic Assembly Language Programming

Hongtao Xu
260773785

Part 1 a) Factorial and Function Calls

Brief description:
In this part, I need to implement the recursive procedures for factorial function first. A recursive subroutine is a subroutine that calls itself. The factorial Factorial, in mathematics, the product of all positive integers less than or equal to a given positive integer and denoted by that integer and an exclamation point. For instance, if we want to calculate fact (4), which equals to 4*fact (3), equals to 4*3*fact (2), equals to 4*3*2*fact (1), finally gets the result 24.

Logic:
In the Editor, I define the values for R0, R1, and R2 respectively at first. Then I used the 'BL' method to jump to 'if' brunch to run the main logic. In the 'if' branch, firstly I 'PUSH' the LR to write logic without any disturbance from outside. Then, I compare the value R1 and number 2, if R1 is smaller than 2, it goes to 'back1' brunch, which means the logic is over, while if R1 is larger than or equal to 2, the logic is continuing. Next, we multiply R2 and R1 together and store them in R2. Then we move the value of R2 into R0, R0 is the place where we have the final value. Then we do the subtraction to R1 by 1. Finally, we 'POP' the value. The 'BL' method at the end would call the 'if' function again and again, and the logic would stop until R1 is smaller than 2. This is the recursive process and function overall.

As a result, if we assign number 4 to R1, the final answer stored in R0 is 24, which is the same value as the fact (4), which means the function is corrected.

Part 1 b) Fibonacci and Function Calls

Brief description:
In the 2nd question of part 1, I need to implement the recursive procedures for Fibonacci function. In mathematics, the Fibonacci sequence is a series of numbers where a number is the addition of the last two numbers, starting with 0, and 1. The Fibonacci Sequence are 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55… and the expression is Xn =X(n-1) + X(n-2). For example, Fib (5) is computed as follows: Fib (5) = Fib (4) + Fib (3) = (Fib (3) + Fib (2)) + (Fib (2) + Fib (1)) = ((Fib (2) + Fib (1)) + 1) + (1 + 1) = (1 + 1 + 1) + (1 + 1) = 5.

Logic:
In the Editor, I call the value for R0, R1, R2, R3, and R4 respectively at first. Then I go to the 'if' branch by 'BL if' to write my main logic. In the 'if' branch, firstly I 'PUSH' the LR to write logic without any disturbance from outside. Then, I compare the value R1 and number 3, if R1 is smaller than 3, it goes to 'less' brunch, which means the logic is over, while if R1 is larger than or equal to 3, the logic is continuing. Firstly, I add R2 and R3 together and store them in the R5. Next, I move the value from R5 to R0, R0 is the place where we store the final value. Then I move the value of R3 to R2 and R5 to R3. We then subtract R4 by 1 and store the value. Finally, we compare the values R4 and number 3. If R4 is less than 3, it goes to 'total' brunch and finishes the logic then, while if R4 is larger than 3. It goes to 'if' brunch again until either R1 is less than 3 or R4 is less than 3, the logic is over. That is the whole process of the recursive function applied to calculating the Fibonacci sequence.

As a result, if we assign number 6 to R1, the final answer stored in R0 is 8, which is the same value as the Fib (6), which means the function is corrected.

Part 2: 2D Convolution

Brief description:
In this question, I will implement the 2D convolution algorithm which is usually used in image processing applications. The 2D convolution can be used to implement image filters in our daily social media. In mathematical, n signal processing, multidimensional discrete convolution refers to the mathematical operation between two

functions f and g on an n-dimensional lattice that produces a third function, also of n-dimensions. Multidimensional discrete convolution is the discrete analog of the multidimensional convolution of functions on Euclidean space.

Logic:
In this question, we are asking to filter through a 10*10 size 2D image by a 5*5 size Kernel with the 2 steps' stride. In detail, at first, we take a 5*5 size image on the 2D Image's top left corner, we do the dot product with it and the 5*5 size Kernel. For example, in this question, the dot product is 1*183+207*1+128*0-1*30-1*109 (1^{st} column) + … (and 4 more columns) = results, which forms the first element on the output result. Then we move 2 steps to the right and form the next element on the output result. Finally, we would get a 10*10 size output result.

For the 2D convolution algorithm written in the C language, at first, it compares y and the image height, if y smaller, it goes to the next loop, which compares x and the image width, if x smaller, it goes to the inner loop, which compares i and Kernel width, if i smaller, it goes to next loop which compares j and Kernel height, if j is smaller, in the loop, there are 2 temp values created. After compares the values of temp1 and temp2 respectively, the sum is calculated. This sum is the first element on the output result.

In the Editor, I firstly define numbers in the 2D image and numbers in the Kernel into R0 and R1 separately. Then, I assign values into each parameter separately from R2-R12. Then I use compare method to define for loop for y and x in order. Same as defining i and j. Then I calculate the values for temp1 and temp2 using values of x, i, ksw and y, j, khw. I compare the values of temp1 and temp2 with the other numbers inside one, two, three, and four branches. In the end, I calculate the sum and get the output result.

Part 3: Bubble Sort

Brief description:
In this part, we need to implement the bubble sort algorithm, which is a simple sorting algorithm that repeatedly steps through the list, compares adjacent elements, and swaps them if they are in the wrong order. The pass through the list is repeated until the list is sorted. The algorithm, which is a comparison sort, is named for the way smaller or larger elements "bubble" to the top of the list. Here are 5 numbers and we need to use bubble sort to arrange them in ascending order.

Logic:
First of all, I read through the example of the bubble sort implemented in C and using pointers. In the C code, it defines the size and the number of the array. There are two for loops, the outer and inner for loop. For every time the inner loop completes once, the outer loop increases by 1. The inner loop is to compare the values and swap if those two numbers are in the wrong order, while the outer loop is to continue the array for the next two numbers.

After understanding the algorithm in C, I would write them in CPUlator. In the Editor, first of all, I define the 5 words in the array: -1, 2, 0,12, -7, 23. We start the logic, we first load array into R0, and then we move 4 numbers into R1 and 0 number into R2, which is the size and step respectively. Noticed that there is always "size-1" written in the C algorithm, in this way, I directly assign 4 numbers instead of 5 here. Next is 'steploop', I compare value from R2(size-1) and R1(step), if R2 is smaller than R1, which means if the step is smaller than 'size-1', it goes to the iloop, which means i++. In the 'iloop', we subtract R2-R1 and store it into R4. We compare the value of R3 (i) and R4(size-1-step), if larger, we add 1 to the size and go back to the 'steploop' for running the outer loop, while if smaller, we moved to the 'assign' loop. In the assigned loop, we compare one number(ptr+i) and its following number (ptr+i+1), if the number is smaller than its following number, means they are in the right order and we increase R3 by 1 and back to 'iloop' again to compare next 2 numbers, while if the number is larger than its following number, we need to go to 'swap' loop. In the swap loop, we swap the two numbers in the right order, and then R3 increases by 1 and goes back to the 'iloop' again. Until R3 is larger than the R4, that's one inner cycle completion, we increase R2 by 1 and back to 'steploop'. Until R2 is larger than R1, the whole process is finished. That's the whole process of implementing bubble sort in CPUlator.

Challenge faced and possible improvement：
The challenge I meet in bubble sort is to understand the relationships of two for loop in C algorithm and how to implement them in the hardware programming. I did lots of research and even review COMP250 to help me understand it better.