Lab 2 Report

Part 2

In this part we implemented a program to find the maximum from a list of numbers. We have done this program in the first lab, but in this lab we are going to implement a subroutine to find the maximum.

We start of by setting register R4 to hold the address of RESULT which we initiated to have the value 0. We then load register R2 to hold the number of elements, N, in the list we want to work with. Then we assign register R3 to point to the first number in the list. The first number will be treated as the current maximum.

After that we enter out subroutine using BL. What BL does is branches to the subroutine but before that stores the address of the best instruction in the link register.

We enter the subroutine loop where we push the LR containing the address of the next instruction into our stack. We then decrement the loop counter by one. We use BEQ to see if the the value of R2 (N) is equal to 0. If not we make R3 point to the next number in the list, our effective address is now of the second number in the list. We load R1 with the second number in the list and then compare the first and the second number. If the second number is larger than the first number, we replace the value stored in R0 with the value in R1, in not we branch back to the loop and keep on going through the list of numbers and replace R0 with the greater number each time. After the list has been scanned and R0 updated, LR is popped, we are taken to STR operation where we update the RESULT to store the maximum.

Fibonacci

For this part we implemented a number of subroutines to calculate the Fibonacci of a given number.
First we make register R0 point towards N, N is the number of we want to find the Fibonacci of. We load the value of N to register R1. Then we enter the subroutine FIBLOOP. In FIBLOOP we PUSH the registors R3, R4 and LR onto the stack. We then compare the value stored in R1(N) with 2. If is is greater than 2, We branch to LOOP2, inside LOOP2 we decrement the value of N by one to get N-1 and store that in R1 and then we decrement again to now have N-2. We store N-2 in R3. After that we branch back to FIBLOOP. We again push R3, R4 and the LR to the stack, this time the LR points to the first MOV operation.

We again compare the value stored in R1 to N. If the value is less than 2, we branch to ELSE. Inside ELSE, we set the value of R1 to 1 and then branch to RETURN. In RETURN we pop R3, R4 and LR from the stack. Then we branch to the address stored in the LR. In this instance the

LR is pointing to MOV. In move we make register R4 hold the value of R1 which we recently set to be 1. We then move the value stored in R3 (N-2) in R1 and then branch to FIBLOOP.

We keep on doing this each time updating the value of R4 by adding R1 and R4. After all of the registers have been popped, we are taken to a STR instruction. We then store the value of R1 into RESULT.

An improvement we make is that we did not need to push the register R4 everytime. Another improvement would be that we could have implemented the functionality of ELSE directly after _start.

Push pop

We implemented a push instruction and a pop instruction without using the PUSH and POP instructions. For the push, the idea is to decrement the stack pointer by 4 (grow the stack) and then store the register to the top of the stack.
For the pop, first, load the contents of the stack pointer to a register and then add the stack pointer by 4 (shrink the stack).

Pure C program

This part is a C program. The objective is to implement a program so that it can find the maximum number in an array.
First, create an array that only contains integer numbers. Then initialize the max_val (the maximum number in the array). We set the value of max_val to be the first element of array. After that, we created a for loop that goes through every element in the array. If the next element is greater than the max_val (the current maximum number), then update the max_val to be this element.

Calling an assembly program subroutine from C

In this part, the objective is still to find the maximum number in an array, but calls an subroutine which is implemented by assembly program. The assembly program compares two number and returns the larger number of the two.

We first created an array. And we set the value of max_val to be the first element of array. After that, we created a for loop that goes through every element in the array. Then we call the subroutine.
The idea is we compare the current maximum number with the next element by calling the subroutine. The result of subroutine will update the value of max_val. The result returned is the maximum number of the array.