ECSE324 Lab2 Report - G66

Gengyi Sun , Liang Zhao

October 2019

1 Subroutines

1.1 Stack

Stack is a First In Last Out(FILO) data structure. These are build-in ARM Stack functions(PUSH, POP) and Stack Pointer (SP) is used to refer to the address of elements in the stack. We used STR R0, [SP] to store numbers in the stack, and LDR R1, [SP] load the elements in the stack to the registers.

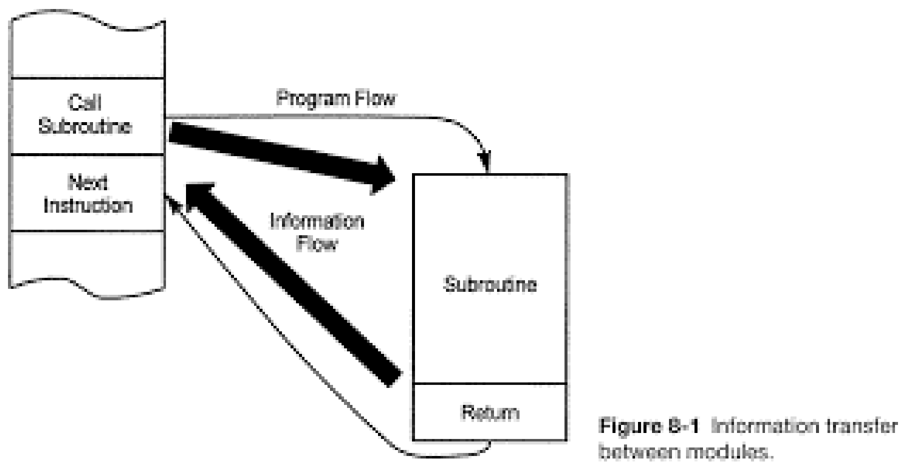
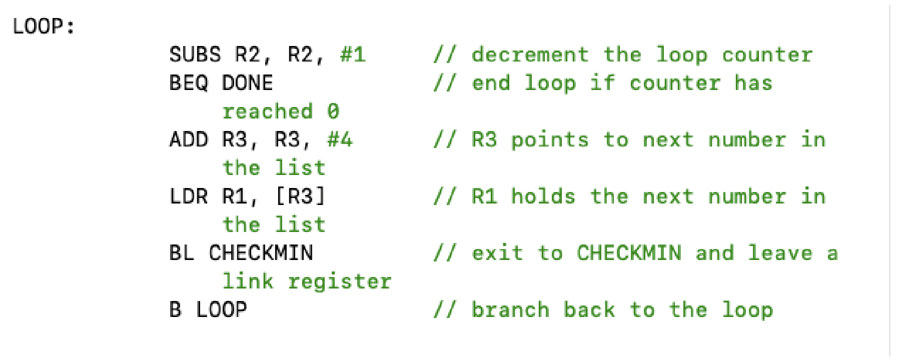


Figure 1: Demostration of Subroutine Calls

1.2 The Subroutine Calling Convention

Subroutine means that the execution procedure is temporarily left the main procedure and run in another routine and return to where it left after the sub routine is finished.
There are built-in key words for calling subroutines in ARM. In this lab, We used BL to call a subroutine and stored the linked register(LR) in a stack. After

the subroutine is about to finished, we pop out the LR and use key word BX to regain the address where we entered the subroutine.
For finding the minimum number, we used a loop to iterate the list, but instead of using branch in lab1, we defined CHECKMIN as a function in our subroutine.

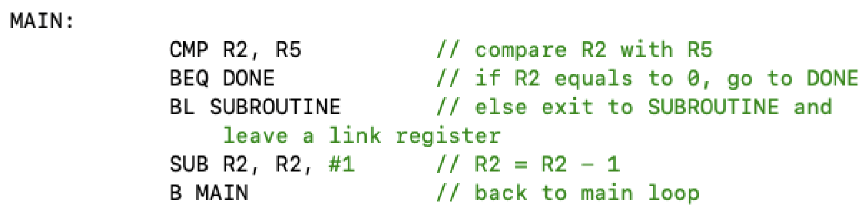


```
LOOP:
            SUBS R2, R2, #1     // decrement the loop counter
            BEQ DONE            // end loop if counter has
                reached 0
            ADD R3, R3, #4      // R3 points to next number in
                the list
            LDR R1, [R3]        // R1 holds the next number in
                the list
            BL CHECKMIN         // exit to CHECKMIN and leave a
                link register
            B LOOP              // branch back to the loop
```

Figure 2: Code Segment: BL Calling Subroutine

1.3 Recursive subroutine to calculate factorial of an integer number

We implement the factorial by separating it to two parts, the MAIN and FACT_SUBROUTINE. The MAIN is used for decrementing the factorial number.



```
MAIN:
            CMP R2, R5          // compare R2 with R5
            BEQ DONE            // if R2 equals to 0, go to DONE
            BL SUBROUTINE       // else exit to SUBROUTINE and
                leave a link register
            SUB R2, R2, #1      // R2 = R2 - 1
            B MAIN              // back to main loop
```

Figure 3: Code Segment: Factorial

The FACT_SUBROUTINE part is for calculating the product of ‘factorial of n’ and ‘n+1’.

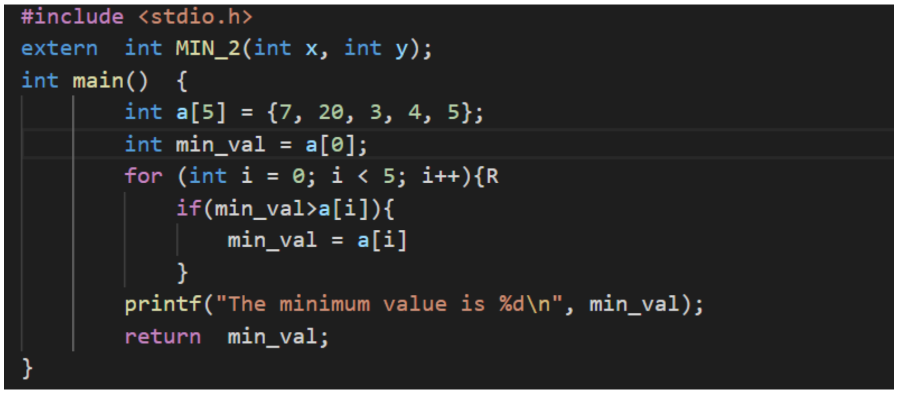
```
SUBROUTINE:
            PUSH {LR}           // push link register to a stack
            MUL R0, R0, R2      // R0 = R0 * R2
            POP {LR}            // pop link register from the
                stack
            BX LR               // return to next address where
                we exit the MAIN loop
```

Figure 4: Code Segment: Factorial

2 C Programming

2.1 C Program

Finding the least number in C is much easier to code than in assembly. First all the numbers are stored in an array. Then initialize min_val as the first element in the array.
The program loops through the array and comparing min_val with the current element in the loop. If min_val is greater than the element, assign the value of the element to min_val, otherwise do nothing.

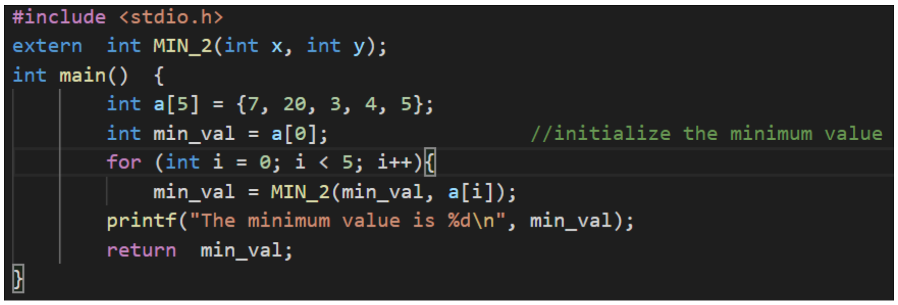


```c
#include <stdio.h>
extern  int MIN_2(int x, int y);
int main()  {
        int a[5] = {7, 20, 3, 4, 5};
        int min_val = a[0];
        for (int i = 0; i < 5; i++){R
            if(min_val>a[i]){
                min_val = a[i]
            }
        printf("The minimum value is %d\n", min_val);
        return  min_val;
}
```

Figure 5: Code Segment: Looping to Find Minimum in C

2.2 Calling an Assembly Subroutine in C

The logic behind finding the minimum value in an array by calling subroutine in assembly is the same. By contrasting, the only difference is the if-statement used previously is replaced by a subroutine in assembly. This subroutine performs similar function as the if-statement: it also compares the two values in the registers, moving the smaller one into R0.



```c
#include <stdio.h>
extern  int MIN_2(int x, int y);
int main()  {
        int a[5] = {7, 20, 3, 4, 5};
        int min_val = a[0];                 //initialize the minimum value
        for (int i = 0; i < 5; i++){
            min_val = MIN_2(min_val, a[i]);
        printf("The minimum value is %d\n", min_val);
        return  min_val;
}
```

Figure 6: Code Segment: From C Calling Assembly Subroutine