Experiment 4: Stack and Subrotouine

(2-6).12.2024
Res. Asst. Batuhan Cengiz
cengiz16@itu.edu.tr

Clowns to the left of me
Jokers to the right
Here I am stuck in the middle with you

Stealers Wheel

1 Introduction

This experiment aims to enhance the practical experience about function calls and usage of the stack. Students are recommended to check Stack operations and also the differences between **CALL** and **JMP** instructions from the documents on Ninova.

2 Part 1

A simple program that consists of several function calls is given below. The given program takes an array of 8-bit integers, does arbitrary operations on them, and stores them in a memory location. Run the following program in order to observe and understand the basics of function call mechanism in microcomputer systems. You should run the given program step by step and fill Table 1 for the first iteration of the **Mainloop** by using Debug Mode of CSS. Also, please add this table to your report with sufficient explanations.
Note: The content of the stack should include all the stack, not only the top element.

```
Setup       mov #array, r5
            mov #resultArray, r10

Mainloop    mov.b @r5, r6
            inc r5
            call #func1
            mov.b r6, 0(r10)
            inc r10
            cmp #lastElement, r5
            jlo Mainloop
            jmp finish

func1       dec.b r6
            mov.b r6, r7
            call #func2
            mov.b r7, r6
            ret

func2       xor.b #0FFh, r7
            ret

;Integer array,
array   .byte 1, 0, 127, 55
lastElement

finish      nop
```

Additionally, you should include the following line above .text section in your main.asm to define the uninitialized memory location where the program stores the output.

```
result      .bss resultArray ,5
```

As seen in the given program, there are two functions defined such as **func1** and **func2**. The first function uses **R6** to take its input and return the related output. In the same manner, the second function uses **R7**. On the other hand, the main program uses **R5** and **R10** in order to store addresses of input and output arrays.
For part 1, you are asked to fill in the given table below by the values for the first iteration.

Code	PC	R5	R10	R6	R7	SP	Content of the Stack
mov #array, r5							
mov #resultArray, r10							
mov.b @r5, r6							
inc r5							
call #func1							
dec.b r6							
mov.b r6, r7							
call #func2							
xor.b #0FFh, r7							
ret							
mov.b r7, r6							
ret							
mov.b r6, 0(r10)							
inc r10							

3 Part 2

In this part, you are required to implement subroutines, named *Add, Subtract, Multiply, Divide.*

- Add(a,b)=a+b
- Subtract(a,b)=a-b
- Multiply(a,b)=a*b
- Divide(a,b)=a/b (Integer division)

When implementing these functions, you are required to pass the parameters of the method **through the stack** and retrieve the result in the same manner. Also, you should benefit from Add and Subtract subroutines to implement Multiply and Divide. You may check the *Assembly_language_tutorial* from Ninova for information on how to pass parameters to a subroutine by means of the stack and how to retrieve the result.

4 Part 3

In this part, you are required to implement a subroutine that calculates dot product of two vectors of same length. The dot product could be written as a recursive way.

$$dot(A, B, i, N) = \begin{cases} \&A * \&B + dot(A+1, B+1, i+1, N), & i < N \\ 0 & i = N \end{cases}$$

Here A represents the starting address of the first array, B represents the starting address of the second array, i represents the current indice and N represents the length of the arrays.

For the given arrays below, calculate their dot products recursively.

```
array_A    .word 15, 3, 7, 5
array_B    .word 2, -1, 7, 3
```