

Experiment 4: Stack and Subrotouine

06.11.2023

*Res. Asst. Yusuf Huseyin Sahin
sahinyu@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 and Preliminary

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.

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.

Experiment 4: Stack and Subroutine

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							

```

1 Setup      mov #array , r5
              mov #resultArray , r10
3
5 Mainloop   mov.b @r5, r6
              inc r5
              call #func1
7              mov.b r6, 0(r10)
              inc r10
9              cmp #lastElement , r5
              jlo Mainloop
11             jmp finish
13 func1      dec.b r6
              mov.b r6, r7
15              call #func2
              mov.b r7, r6
17              ret
19 func2      xor.b #0FFh, r7
              ret
21
23 ;Integer array ,
array        .byte 1, 0, 127, 55
lastElement
25
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.

2 (20 pts) Part 1

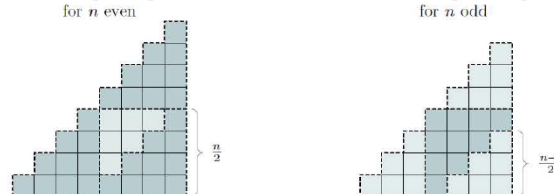
For part 1, you are asked to fill in the given table by the values for the first iteration.

3 (30 pts) Part 2

In this part, you are required to implement multiply, div and power subroutines, both taking two **words** as parameters. Power subroutine should use multiply subroutine. 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. **You should backup (push) the registers to use in the subroutine at the start of the subroutine. Then, before returning, you should pop them.**

4 (50 pts) Part 3

The summation of the all integers between $[1 - n]$ could be calculated as follows:

$$\begin{array}{cc}
 S(n) = 3S\left(\frac{n}{2}\right) + S\left(\frac{n}{2} - 1\right) & S(n) = 3S\left(\frac{n-1}{2}\right) + S\left(\frac{n+1}{2}\right) \\
 \text{for } n \text{ even} & \text{for } n \text{ odd}
 \end{array}$$


$$S(n) = \begin{cases} 1, & n = 1 \\ 3, & n = 2 \\ 3S\left(\frac{n}{2}\right) + S\left(\frac{n}{2} - 1\right), & n > 2 \text{ and even} \\ 3S\left(\frac{n-1}{2}\right) + S\left(\frac{n+1}{2}\right), & n > 2 \text{ and odd} \end{cases}$$

Write the recursive subroutine S to calculate the summation.