Laboratory 4
Using the Stack for RPN Calculations

Introduction

The objective of this laboratory is to provide you with more experience in using the stack as a general data structure. To achieve this, you will be implementing a simple Reverse Polish Notation (RPN) calculator.

An RPN calculation uses postfix notation, in which the operands precede the operators (+, –, ×, ÷), rather than infix notation, where the operators appear between the operands. For example, an RPN expression of 5 7 + is equivalent to the infix expression (5 + 7). Similarly, 7 3 9 + + 11 – would be equivalent to (7 + (3 + 9)) – 11.

((3 + 9) + 7) - (11)

Procedure

To begin this laboratory, you will start with your solution to Part 1 of Laboratory 3, which implements the binary to 7-segment pattern lookup for values in the range of 0–9. That code read the switch values into register `R5` before it was converted into a single digit pattern (stored in register `R6`) which was then stored to the 7-segment display address.

Modifying the Program

1. The solution in Laboratory 3 was only able to display a single binary encoded digit on the 7-segment displays. (i.e. only the values 0 through 9.) Extend your subroutine so that it include the full range of patterns `0`–`9` and `A`–`F`. Next, modify your program so it can take the full (unsigned) 10-bit number selected via the switches, and display its value on the 7-segment display as four hexidecimal digits. For example, the switch settings of `1010111001` would display as `02B9`, and `1111101101` would display as `03ED`.

2. As you did in Laboratory 3 add the following assembler directive…

   ```
   .equ    STACK_BASE,    0x30000000
   ```

   …at the top of your program to define a label for the memory location which will serve as the base of the stack.

   To implement the RPN calculator, we will use the switches to represent the operand values, while the pushbuttons (key switches) will represent the operators. For our purposes we will only implement two operators, namely addition (`KEY3`) and subtraction (`KEY2`).

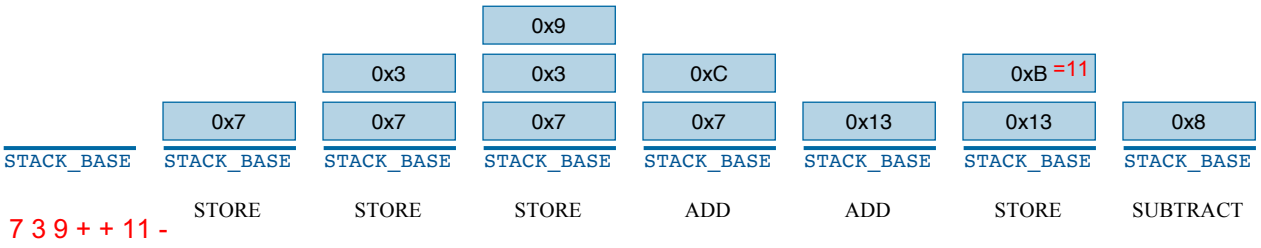
   `KEY0` will be used as the STORE button to capture each operand. Whenever the user presses the STORE button, the current (10-bit) value of the switches should be pushed onto the top of the stack. This same value should be shown on the 7-segment displays.

   When the user presses either of the two operator buttons (`KEY3` or `KEY2`), the top two values are popped off the stack, and the indicated operation is performed, with the result then being pushed back onto the stack. The 7-segment displays should show the result of the operation (the value just pushed).

   For example, for the calculation…

   7 3 9 + + 11 –

   …the stack would progress as shown in the figure on the following page.



Verify that the program functions correctly for the example above, and for larger operands (ex. `0x3FF 0x25C 0x3ED` + –), and for longer sequences.

25C + 3ED = 0649 => 3FF - 0649 = FDB6

3. As a final small modification, `KEY1` may be used to CLEAR the calculation (i.e. by reinitializing the stack pointer back to its `STACK_BASE` value). When this occurs, the 7-segment display should either show the value zero or be blank.