LAB #3: Reverse Polish Notation Calculator

The Lab write-up is due to your TA at the beginning of your next scheduled lab. Don't put this off to the last minute! There is pre-lab work to complete before the start of the next lab. **NO LATE LAB REPORTS WILL BE ACCEPTED.**

1 Objectives

- Use the parallel I/O functions of the MC9S12C32.
- Learn about stack manipulation and embedded arithmetic.
- Design a Reverse Polish Notation Calculator.

2 Background

In this lab, you will use the parallel I/O ports on the MC9S12C32 to design a Reverse Polish Notation calculator. This calculator will accept inputs given in Reverse Polish Notation and calculate answers to expressions. This device will accept 8-bit inputs taken from dipswitches (the dipswitches on the project board), execute a 16-bit operation (add, subtract, multiply, and divide) and display the results on the LCD screen. Answers to expressions that do not cause overflow or underflow will be displayed normally to the LCD screen. In case of value overflow or underflow, the device should display an overflow or underflow error to the LCD screen. You will also display a stack underflow error if an insufficient number of operands are present on the stack.

You will need to use 6 push button switches for input. One for "push", one for "pop", one for "add", one for "subtract", one for "multiply" and one for "divide". You may use any of the available push buttons, but it is recommended that you use the 4 push buttons on the project board for add, subtract, multiply, divide and use the 2 push buttons on the module for push and pop. You must use the 8 dipswitches as your input, and these values will be interpreted as 8-bit **unsigned** integers, which are viewed as a positive numbers in their 16-bit representation. Hence, while your inputs are only 8-bits, your calculator will perform 16-bit operations and display a 16-bit signed result. In the case of a divide your stack will end up with the remainder on the top of the stack and the quotient will be the next value on the stack. Namely if you want to use the quotient in the next operation you will "pop" the stack to evict the remainder to get the quotient at the top of the stack.

Note that the 6812 does not provide multiply or divide op-codes. Hence you will need to write code to perform these operations – namely multiples can be done with shifts and adds, and divides can be done with shifts and subtracts.

You will need to explicitly check for value overflow and underflow. This will require that you pay attention to the bits in the condition code register. For example if adding 2 negative numbers results in a positive number then an underflow has occurred. Similarly if multiplying 2 values of opposite signs result in a positive value then underflow has also occurred. Overflow occurs when 2 positive values are added and the 16-bit result is negative. Similarly if a multiply of 2 values with similar signs results in a 16-bit number that appears negative then overflow has also occurred. In the event of an overflow you need to display "Err OVF" to the LCD screen. In the event of an underflow you need to display "Err UDF". The result of a "push" will cause the value on the input dip switches to be

displayed to the LCD screen and the value placed on the stack. The result of a "pop" will cause the element at the top of the stack to be removed and the new "top" value to be displayed or else do not display anything if the stack is empty. The result of a successful addition will be pushed onto the stack displayed as an integer to the LCD screen. The result of a successful subtraction will be pushed onto the stack and displayed as either a positive integer or negative integer to the LCD screen. In the event of a negative integer, you need to display a leading "-" symbol for negative numbers. The product of a successful multiplication will be pushed onto the stack and displayed to the LCD screen. Finally, the quotient of a divide operation should be pushed onto the stack displayed on the top line of the LCD screen and the remainder on the bottom line. The remainder should ALSO be pushed onto the stack AFTER the quotient has already been pushed. (Hint: divide operations cannot overflow or underflow) (Another Hint: a very simple LCD implementation will simply just display the last element on the stack)

An example input to calculate (2+3) * (4+5) on your calculator will consist of the following sequence of actions. Note that in this example the stack is growing down and the base of the stack is at the top of the diagram's frame. Input a 2 on the dipswitches and "push" that onto the stack. "Push" a 3 onto the stack, press the "add" button and put that answer onto the stack and display the result "5" to the LCD. "Push" a 4 and 5 onto the stack and again place the result 9 onto the stack. Then push the "multiply" button and push the result onto the stack and display a "45" to the LCD. Finally do a pop to clear the stack.

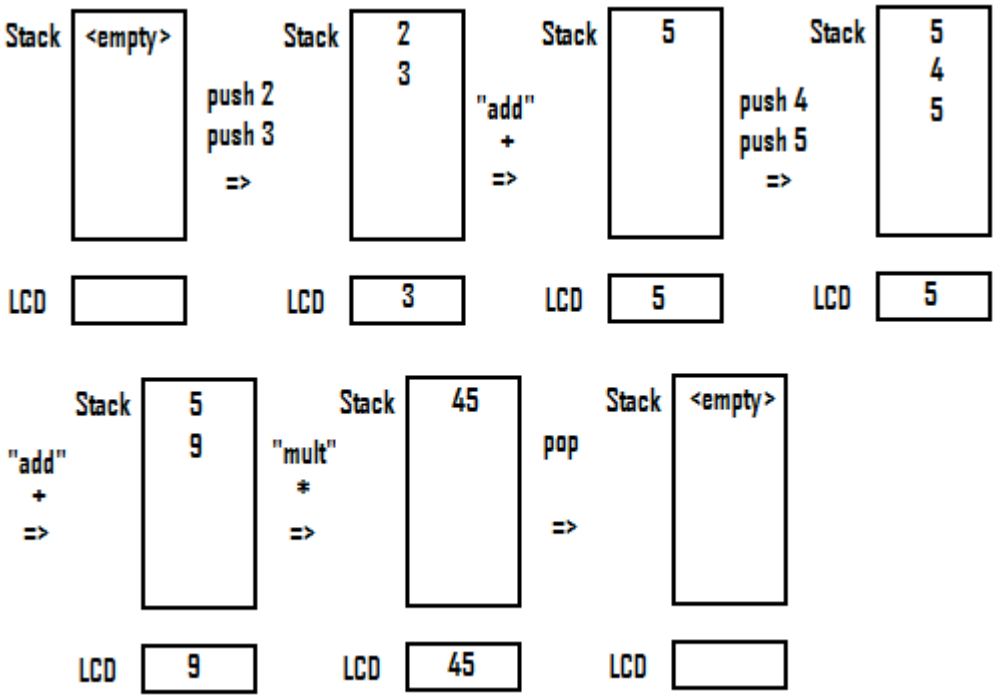

Fig 1: Steps to execute (2+3) * (4+5) on your calculator.

Finally there is one additional error that you will have to check. Since each operation requires 2 operands on the stack you will need to make sure that there are at least 2 values on the stack before you perform the operation. If there are less than 2 operands on the stack you should display "Err StkUF" to indicate that this operation could not be performed since the stack would underflow and potentially destroy whatever values might have been allocated to these locations.

3 Pre-lab

Complete tasks 1 and 2 before coming to lab.

4 Tasks

1. Prepare a schematic for your design. The schematic should contain information about each function connected (PB4, SW 2, etc…), the User I/O signal number on the project board, the MCU board port number, and the MCU Port (PT0, PT1, etc.)

2. Design the program (either in C or assembly or both) for your calculator. Your code does not need to be fully functional, but you should at least have the following completed:

    a) Your code should be capable of reading input from the dipswitches.
    b) Your code should be capable of distinguishing between a push, pop, add, subtract, multiply, or divide. You should have wrappers for each of these events, even if they do not do anything.
    c) Your code should be capable of displaying integers to the LCD screen.
3. Finishing coding your solution. Connect your circuit and debug your software. Then check-off your working Reverse Polish Notation Calculator with your TA.

5 Writeup

Include the following items:

1. Your hardware schematic
2. A description of your design and why you chose to implement it that way.
3. A printout of your well-commented code.
4. Describe any problems that you encountered.