Lab 3: More Assembly Language Programming

CompEng 3151: Digital Engineering Lab II

Last revised: July 16, 2019 (BJZ, RJS)

Description/Overview

In the last lab, you learned how to build assembly programs for the AVR microcontroller. For this lab, you will extend your knowledge of assembly in completing some basic math calculations both with static calculations (hard-coded math) and by reading input and outputting the result of a calculation.

Objectives

Students will:

1) Apply the assembly language programming techniques learned to create a program that performs basic math operations and outputs the result
2) Read inputs switches, perform calculations on the input data and then output the result

Materials Required

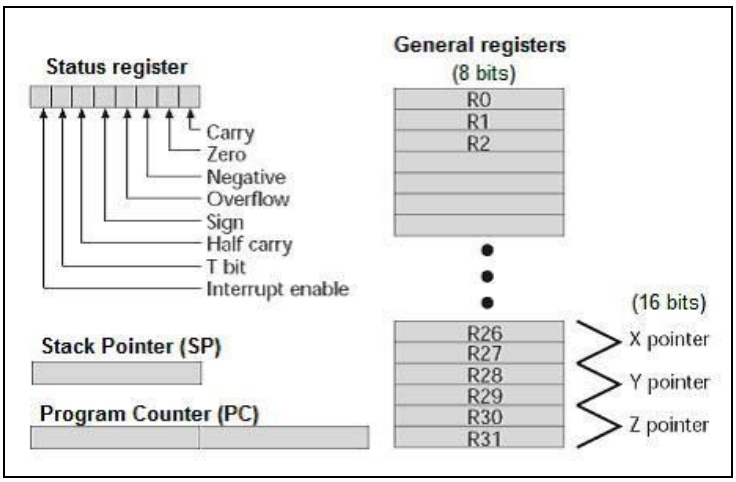
- Atmel Studio 7 Installed on PC (ECE CLC Computers will have this)
- AVR Simon Board with a USB cable

Preparation

In order to be successful with this lab, students should review Chapters 3, 4 & 5 in the Mazidi textbook. In particular, students should look at the sections on the status bits and how they are affected by various arithmetic operations, the branch operations that work with the status bits, and how data can be serially read and written.

Background

Register File



Except for the program counter (a 16-bit register), the AVR architecture implements 8-bit registers: 32 general purpose (from R0 to R31), status, and stack pointer.

On the general purpose registers, the last 6 ones can be paired and used as three 16-bit registers (X, Y, and Z). Also important, several instructions can only be used with the upper half of the register file (from R16 to R31)

The status register contains eight different flags (indicators) that are updated according to the results of every operation the CPU performs. For example, if the

preceding operation resulted in a zero, the flag 'zero' is set to 1, otherwise is 0. In many cases, the register is verified before executing the instruction in turn. Your textbook summarizes what instructions affect the various flag bits (Mazidi page 71-74).

Procedure

1) *Multiplication Problem*
    a. Multiplication can be generalized as a series of additions. However, if the numbers you are multiplying get quite large, then it is obviously very inefficient to perform the operation in this manner. For this part of the lab, you will write this generalized multiplication routine for 3x5 that utilizes loops and a series of additions. *There are a few multiplication instructions that do exist in the instruction set for the AVR. However, for this lab, you are to find another way to logically complete the operation.*

2) *3-Bit Multiplier Circuit*
    a. For the second part of this lab, you are going to expand on the program you wrote so that it can take an input of two 3-bit numbers. These numbers will come from SW1-SW3 and SW7-SW9 (as labeled on the Simon Board) (Note that these switches are actually push buttons.). As the buttons are pressed, the product should be displayed in binary using the LEDs 1-4B as the MSB and LEDs 6B-9 as the LSB.

    This may seem fairly easy to do (at least coming up with the basic process), but you're going to need to draw on some knowledge of how the AVR Simon Board is wired up to the input switches. In the case of the six switches, you'll be able to read all of them at same time, but the individual bits will need to be read out and written to different registers before the actual arithmetic can take place. Once the arithmetic is done, you'll have to put the data in a certain order to display it on the LEDs.

    One method you could use to put the data in the correct order is to serialize it after it is read from the input port and then before it is output to the LEDs (see pages 183-185 in the Mazidi book).

Deliverables

For this lab, you are going to submit your first lab report. You should follow the format of the lab report provided by your TA (likely on Canvas). In your report, be sure to include the code you created for both parts of this lab and make sure it is very well commented. Also make sure you answer the questions below either in your results or conclusion section.

Questions/Observations

1. The first program you wrote conceivably can be used to multiply two 8-bit numbers together but it will result in an overflow due to the size of the product. For example, what will the result of your program be if the two numbers were say 0x1A and 0x28? How could you detect such an overflow and compensate for it?

References

- Muhammad Ali Mazidi, Sarmad Naimi, and Sepehr Naimi. 2010. *AVR Microcontroller and Embedded Systems: Using Assembly and C (1st ed.).* Prentice Hall Press, Upper Saddle River, NJ, USA.
- Vasconcelos, Jorge, 2009. *Brief Guide to the AVR Architectures and the AT90USBkey Demonstration Board.* John Hopkins University, Department of Computer Science.