**ECE-3226-50:** Lab #2

Introduction to Assembly Programming & Using Basic Operations

Robert Campbell

**Objective:**

The purpose of this lab is to become familiarized with writing programs in assembly, starting with basic operations. These operations include arithmetic, shift, rotate, and jump instructions.

**Equipment:**

AVR Studio 7.0

**Procedure:**

**Part 1:**

* 1. An incrementor that counts up by even numbers
  2. A decrementor that counts down from 255 by 4’s

1. 

Figure 1

1. 

Figure 2

**Part 2:**

1. 1. Run the provided program and monitor the register values for the logically shift right operation
   2. A program that divides the decimal number 103 by decimal number 8
   3. Run the provided program and monitor the register values for the arithmetic shift right operation
   4. A program that multiplies the decimal value 9 by the decimal value 16
   5. A program that multiplies the previous decimal values using the MUL operation
2. 

Figure 3

1. 

Figure 4

**Part 3:**

1. 1. Run the provided program and observe the registers to gain familiarity with multi-byte addition
   2. A program that adds the decimal value 150 million to the decimal value 1.25 billion
2. 

Figure 5

1. 

Figure 6

**Questions:**

**Question 1\_1:** How would you modify the program you created to count up by the odd numbers (i.e. 1, 3, 5, 7, 9, ...)?



Figure 7

**Question 2\_1:** What is the decimal value in register r20 after executing each instruction? List the hex and decimal values in a table in your report.

See figure 3

**Question 2\_2:** Considering the result after executing each LSR instruction, what mathematical operation is being performed by the LSR instruction?

Divide by 2

**Question 2\_3:** What mathematical operation is the above program (in its entirety) performing?

Divide by 32

**Question 2\_4:** What are the equivalent decimal values for both signed and unsigned representations of content of register r20 after executing each instruction? List the hex and both signed and unsigned decimal values in a table in your report.

See Figure 4 2c for the instruction by instruction register values.

|  |  |  |
| --- | --- | --- |
| Hex | Unsigned | Signed |
| 0xBA | 186 | -70 |
| 0xDD | 221 | -35 |
| 0xEE | 238 | -18 |
| 0xF7 | 247 | -9 |
| 0xFB | 251 | -5 |
| 0xFD | 253 | -3 |

**Question 2\_5:** Does there seem to be any corresponding mathematical operation being performed by the ASR for unsigned numbers? How about for signed numbers?

ASR functions as divide by 2 for signed numbers, but does not have a corresponding operation for unsigned numbers

**Question 2\_6:** What complementary instruction exists in the instruction set that performs a multiply by 2?

LSL, logically shift left

**Question 2\_7:** Does this instruction work for multiplying both signed and unsigned numbers by 2? If not, which representation does it work for?

It works for both signed and unsigned numbers.

**Question 3\_1:** What do the initial two lines you placed in the 'Declarations' section of your assembly program do? What is the .equ directive for?

It creates a symbol that the assembler will track and recognize so that:

Any time NumA appears, it inserts the decimal value 8600

Any time NumB appears, it inserts the decimal value 6600

**Question 3\_2:** What results are stored in registers r16 and r17 after the program executes? What does this program do?

The program adds the two-byte numbers 8600 and 6600 together, storing the low-byte of the result in r16 and the high-byte of the result in r17.

**Question 3\_3:** Would the results be different in the instruction "adc r17, r21" was replaced with the instruction "add r17, r21"? How so? Why is the "adc" instruction needed?

In this instance, no it wouldn't be different. ADC adds the carry bit in with the LSBs of the two bytes being added. Since the previous operation did not result in a carry out the two instructions are functionally identical, however this is not guaranteed in multiple precision addition.

**Question 3\_4:** How would you modify the program to subtract NumB from NumA? Try out your modification. Does it perform as expected? Provide the result.

Replace add r16,r20 with sub r20,r16, adc r17,r21 with subc r21,r17. It does perform as expected with the result being 0xF830 with the sign flag active.



Figure 8



Figure 9

**Question 3\_5:** Find (and report) a pair of numbers for NumA and NumB that result in a carry of 1 between exactly two of the four pairs of byte additions between NumA and NumB. Show the numbers in hex.

0x4080C040  
+ 0x90A09020

0xD1215060

**Discussion/Conclusion:**

The goal with this experiment was to gain familiarity with writing programs in assembly and some of the different operators it contains. This was done successfully and provided some insight into looking at some programming problems in a different way. One challenge that was faced was trying to reduce the number of instructions for Part 2e (in fig 4). I had not learned the MUL and MULS instructions yet and spent a little while trying to find other ways to reduce the number of lines used. It was resolved by finding out about the necessary instructions and implementing them.