Lab 5

CS 130, Spring 2021

Intro: In this lab we are going to use add and subtract instructions to multiply and divide unsigned whole numbers. We'll use two processors: an eight-bit processor (6502), and a 64-bit processor (LegV8). For each, we will have to determine the range and the maximum value of an eight-bit register and a 64-bit register.

The following is a general algorithm for the repetitive-add multiply:

Simple Multiply Algorithm for Unsigned numbers
(Repeating add)

```
multiplier      multiplicand   product
    V               V             V
   20       x      5       =    100
```

Start
Enter multiplier
Enter multiplicand
Zero out product
Loop:
add multiplicand to product
Check for Carry
  If Carry, go to Error
decrement multiplier
Is multiplier 0?
  if yes, go to End
  if no, go back to Loop
Error:
replace product with FFh
End:

NOTE: multiplier, multiplicand, and product could be three memory locations, three registers, or some combination of them.

Part A: 6502, Multiplying and dividing (Note: Give the final answers in hex and decimal).

1. Multiply. We will be using the ADC instruction, the only add instruction in the 6502 instruction set. We will have to clear the Carry flag (CLC) before each addition to ensure it isn't set. We'll enter the multiplicand in the accumulator and the multiplier in the X-register. Use load immediates. The product could be in the accumulator or a memory location (use addresses 700_{16} and above).

   a. Test with 5 & 20 and print screen at the end. Label it PtA1a. What was the answer? _____

   b. Test with 3 & 120 and print screen at the end. Label it PtA1b. What was the answer? _____

   c. Test with 8 & 31 and print screen at the end. Label it PtA1c. What was the answer? _____

2. Divide. We will be using the SBC instruction, the subtract with carry instruction. We will have to set the Carry flag (SEC) before each addition to ensure it isn't set. We'll enter the dividend in the accumulator and the divisor in the X-register. Use load immediates. The quotient could be in the accumulator or a memory location (use addresses 700_{16} and above).

   a. Test with 20 & 5 and print screen at the end. Label it PtA2a. What was the answer? _____

   b. Test with 120 & 3 and print screen at the end. Label it PtA2b. What was the answer? _____

   c. Test with 8 & 32 and print screen at the end. Label it PtA2c. What was the answer? _____

NOTE: The first number is the dividend and the second one the divisor.

3. The product of two 1-byte numbers will fit in 2 bytes. This is true for all size registers: the product of two *n*-bit numbers will fit in *2n*-bits.

   With this in mind, let's extend the routine in Pta1. Instead of going to the error routine when the number is too big for the register (this is what the Carry flag is telling us in PtA1), let's jump to an *extend* routine. This routine will save the accumulator in a byte in memory (the *lo-byte* of the product); zero out the accumulator; set the carry bit; and resume the loop. At the end of the loop, the accumulator will be saved in the byte after *lo-byte*. This will be the *hi-byte* of the product. Now we will be able to multiply any two 8-bit numbers and get an answer.

   a. Test with 255 & 255 and print screen at the end. Label it PtA3a. What was the answer? _____

   b. Test with 120 & 57 and print screen at the end. Label it PtA3b. What was the answer? _____

   c. Test with 20 & 5 and print screen at the end. Label it PtA3c. What was the answer? _____

Part B: ARM, Adding and Subtracting (using the LegV8 emulator in Section 2.24 of the text).

We will repeat PtA1 and PtA2 above using the LegV8 instructions. Of course, we can use much bigger numbers than the 6502.

1. Multiply.
    a. Use 123657 and 8. Label the screenshot PtB1a.
    b. Use 657,839,333,222 and 333,839,657,222. Label the screenshot PtB1b.
    c. Use 123,700,00 and 800,111,777. Label the screenshot PtB1c.

2. Divide.
    a. Use 123657 and 8. Label the screenshot PtB2a.
    b. Use 657,839,333,222 and 333,839,657,222. Label the screenshot PtB2b.
    c. Use 123,700,00 and 800,111,777. Label the screenshot PtB1c.