CS130 Lab Exercise #5

Overview

For this exercise, I recommend using the second of the three ARMv8 Emulators which you will find in Participation Activity 2.24.2. Using this emulator is not a requirement but I will use it to evaluate your most marvelous creations. The problems that follow are to be solved by creating, executing, and testing assembly language code using any or all of the following constructs from *zyBooks* Section 2.24:

This ARM simulator (alpha) supports the following language features:

Instructions:

```
ADD, ADDI, ADDIS, ADDS, AND, ANDI, B, B.EQ, B.GE, B.GT, B.HI, B.HS, BL, B.LE,
B.LO, B.LS, B.LT, B.MI, B.NE, B.PL, B.VC, B.VS, BR, CBZ, CBNZ, EOR, EORI, LDUR,
LDURB, LDURH, LDURSW, LSL, LSR, ORR, ORRI, SUB, STUR, STURB, STURH, STURW,
SUBI, SUBIS, SUBS
```

Note that **MOVK** and **MOVZ** which are capable of 'moving' 16-bit values are not supported.

Label Example:

```
Label1:
   ADD X1, X2, X3
```

Comment Example:

```
   ADD X1, X2, X3 // Adds j + k
```

Note: For the LEGv8 emulator in Section 2.24, it is possible to pre-load a 64-bit constant value into memory and then use the LDUR instruction to load that value into a register. While this method works quite nicely, pre-loading memory may not be possible in the real world. Therefore, Lab Exercise #5 requires students to enter 'immediates' using an alternate method, one such as is presented in the companion document "How to Enter Numeric Values Greater than 1023 Into the *zyBooks* Emulator".

Step #1

Ensure that 'Preset Register' X3 is set to 4000. Create a LEGv8 assembly language program to multiply together the integers **12345** and **222**. Place the product in memory location **4000** which is pointed to by the contents of Register **X3**.

You will learn all too quickly that you cannot directly load into a register a numeric value greater than **+1023** or smaller than **-1024** using a so-called 'immediate' instruction such as **LDUR**. See note above. This limitation is not a deficiency of the emulator, but rather a consequence of the design of the fields in the LEGv8 instructions.

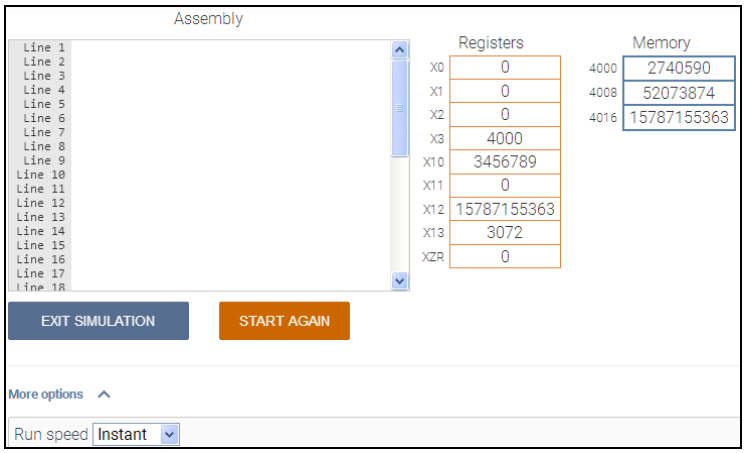
Therefore, you will see that a big part of this lab exercise is creating a method for loading registers. Nasty! After loading and running your program, take a screenshot that shows as a minimum the contents of 'Registers' and 'Memory' after your program has completed. Notice also that "Run Speed" was probably set initially to "Instant" when you first opened Section 2.24.

After loading and running your program for Step 1 above, take a screenshot that shows as a minimum the contents of 'Registers' and 'Memory'.

(Continue to Step #2 on the next page.)

Step #2

After loading and running your program for Step 1 on the previous page as well as for Steps 3 through 8 which follow, take screenshots that show as a minimum the contents of 'Registers' and 'Memory'. Example:



Step #3

Create an assembly language program to multiply together the integers **234567** and **222**. Place the product in memory location **4008**. Use Register **X3** as the base address **4000**.

Step #4

Create an assembly language program to multiply together the integers **3456789** and **4567**. Place the product in memory location **4016**. Use Register **X3** as the base address **4000**.

Step #5

Logically AND the results of Steps 3 and 4 and place the result in memory location **4024**.

Step #6

Logically OR the results of Steps 3 and 4 and place the result in memory location **4032**.

Step #7

Logically XOR the results of Steps 3 and 4 and place the result in memory location **4040**.

Step #8

Complement the bits from Step 4 and place the result in location **4048**.

Step #9

Create a document that, for each step, contains (a) your assembly code and (b) accompanying screenshot. E-mail to RSturlaCS130@GMail.com.