CS130 Lab Exercise #6

Objective

In this lab exercise, students will use the LEGv8 emulator from *zyBooks* Section 2.24 to create programs to perform a variety of operations possible with the LEGv8 Instruction Set that is posted on "Canvas". Students will submit their assembly level code and – where required – screenshots of the emulator as parts of a "Word" document. For each required program, include your assembly source code.

Requirements

Below is a list of the programs to be created along with a set of requirements for each:

1. a. Using the second LEGv8 "ARM Simulator" found in Section 2.24.2, ensure that Register **X3** contains 4000.
   b. Using the manual preset feature of the simulator, enter **305,419,896** into Register **X0**.
   c. Include in you program an instruction to store this value into Memory location **4000**
   d. Load each of the low order four bytes of **X0** – one at a time – into Registers **X4** through **X7**. (Refer to the Instruction Set for LEGv8 to determine the appropriate instruction to use for this step. Ask yourself: "Self, which instruction(s) will allow me 'byte access'?")
   e. Using Register **X12** as a repository, reverse the byte positions of the value that was copied into Memory location **4000**. NOTE: For swapping the bytes, maintain a data size of 4-bytes even though each LEGv8 register is 8-bytes wide. That is, **Byte 0** moves to the position originally containing **Byte 3** and **Byte 1** moves to the position originally containing **Byte 2**. Example: Original hex values: **0D, 08, 0F, 06** and new hex values: **06, 0F, 08, 0D**.
   f. Copy this new 4-byte value into Memory location **4008**.
   g. Assemble, test, and debug your code if necessary and then capture a screenshot of the Simulator after your program has run successfully.
   h. Retest your program using an initial value of **4,275,878,552**.
   i. Capture a screenshot of the Simulator after your program has run successfully.
   j. Create an evolving "Word" document and enter into it your code and two screenshots.
2. a. Again using the second LEGv8 "ARM Simulator" found in Section 2.24.2, ensure that Register **X3** contains **4000**.
   b. Using the manual preset feature of the simulator, enter **305,419,896** into Register **X0**.
   c. Store this value into Memory location **4000**. (Sounding similar so far?)
   d. Load each of the eight 'nibbles' (i. e. four bits) of **X0** – one at a time – into Registers **X4** through **X11**. Be careful here! You will have a 'high nibble' and a 'low nibble' when you extract each nibble. For the high nibble – before storing it – don't forget to shift it two places to the right to ensure that it is scaled properly for storage.
   e. Using Register **X12** as a repository, reverse both the nibble positions within a byte and then each byte position as well of the 4-byte value that was copied into Memory location **4000**.
   f. Copy this new 4-byte value into Memory location **4008**.
   g. Assemble, test, and debug your code if necessary and then capture a screenshot of the Simulator after your program has run successfully.
   h. Update you evolving "Word" document by entering into it both your code and screenshot.
   i. Retest your program using an initial value of **4,275,878,552**.
   j. Capture a screenshot of the Simulator after your program has run successfully.
   k. Update your evolving "Word" document by adding your code and new screenshot.
3. Submit your properly-named document containing the assembly language code for both Step #1 and Step #2 along with you four screenshots to RSturlaCS130@GMail.com.