CS130 Lab Exercise #7 – 2-D Array

Background

This exercise encourages students to visualize an array of characters organized as eight rows with each containing eight characters. However, in reality, this array consists of eight doublewords stored in consecutive memory locations. Each character, therefore, is a one of eight bytes in each doubleword.

Description

In this assignment, you are to create a 2-dimensional array of one-byte values in Memory. Each of the consecutive eight lines in this memory array contains eight bytes. The array, therefore, is to be considered as eight bytes wide by eight lines high. Each of the 64 data values is one byte in size, not eight bytes.

Requirements – Part I

a. Create a LEGv8 program that meets the requirements below using instructions available in *zyBooks* Section 2.24
b. Initialize Register **X3 = 4000** as the array's Base Address. Do not use the 'Preset' dialog boxes.
c. Populate each of the 64 bytes of this array. The first byte is to contain a value of **0** (zero), the next byte (moving to the right) is to contain a value of **1** (one), and so forth. For each group of eight consecutive bytes, add the byte values together to form a single doubleword.
d. On the first line, the first byte is 0×2^0, the second byte has a weighted value of 1×2^8, and the eighth byte has a weighted value of 7×2^{56}.
e. Add these weighted values together to form a single doubleword value for the first line of the **8 x 8** array.
f. On the second line, the first byte has a weighted value of 8×2^0, the second byte has a weighted value of 9×2^8, the third has a weighted value of 10×2^{16}, and the last byte on the first line has a weighted value of 15×2^{56}.
g. The last byte in the array, the byte on the lower right of the eighth line, has a weighted value of 63×2^{56}.
h. If your code is correct, fetching any byte will yield a value identical to the offset from the Base Address of that byte.
i. Your spiffy finished array should occupy Memory locations **4000** through **4063**.
j. Fetch the byte located at Memory location **4019** and store it Register **X20**.
k. Fetch the byte located at Memory location **4028** and store it Register **X21**.
l. Fetch the byte located at Memory location **4039** and store it Register **X22**.
m. Fetch the byte located at Memory location **4051** and store it Register **X23**.
n. Fetch the byte located at Memory location **4063** and store it Register **X24**.
o. Label and save your source code as YourLastName Lab Exercise #7 Part 1.Txt.
p. Proudly e-mail your Babbage-Quality finished code to RSturlaCS130@GMail.com.

Requirements – Part II

a. Create a LEGv8 program that stores the following ten numbers into Memory locations **4000** through **4072**: **42**, **68**, **35**, **1**, **70**, **25**, **79**, **59**, **63**, **65**
b. Sort these numbers and store the sorted values H → L in Memory addresses **4080** through **4152**
c. Calculate the sum of these ten numbers and place this value in Memory address **4160**.
d. Using successive subtraction, calculate the [integer] average value. Store the average in **X4**.
e. Store the 'remainder' in **X5**.
f. Label and save your source code as YourLastName Lab Exercise #7 Part 2.Txt.
g. Proudly e-mail your Babbage-Quality finished code to RSturlaCS130@GMail.com.