ELEC 2220 Computer Systems
Homework #7
Due: Wednesday, June 7

PART 1

Revise the program from Part 2 of the last assignment in the following two ways, repeat the debug procedure for each, and submit the source program and debug results for each.

a. Modify it to use a single pointer register for all memory references, instead of a separate pointer for each variable. This pointer register should be initialized to the address of the first variable in the data area (*Sue*), with a constant added to the pointer register after each operation to change the pointer to the address of the next variable.

b. Modify it to use a single pointer register for all memory references. However, after initializing the pointer register to the address of the first variable, ***do not alter the contents of this register***. Instead, each variable address should be that pointer plus an offset.

Comment on the differences between the three versions of the program. Which was the most efficient (fewest instructions)? Which was the most straightforward to write?

PART 2

Write and execute a program that adds each element of ARRAY1 to the corresponding element of ARRAY2, with the result written to the corresponding element of ARRAY3, where ARRAY1, ARRAY2 and ARRAY3 are 10-integer (32-bits each) arrays in a data area. Initialize each array with 10 unique data values. An equivalent C instruction sequence might look like:

```
for (k = 0; k < 10; k = k+1) {
  ARRAY3[k] = ARRAY1[k] + ARRAY2[k];
}
```

Use a separate "base address" register to point to the start of each array. Access data using that base address plus a common "index" or "offset" register. (The index is the same for all three arrays.) Change the index register after storing each result, so the next operation will use the next elements of each array. Do not change the base register.

You must use a "loop" ("for loop" or "while loop") in your program that repeats N times, corresponding to arrays of N words. **Do not** simply write N sets of instructions! The program should be able to be modified to add arrays of arbitrary size by simply changing the value of N. ***See examples in Chapter 6, sections 6.6-6.8, of the text.***

One way to create such a loop is by initializing a register to the value N, to be used as a loop counter. At the end of the set of instructions to be repeated, subtract 1 from that count (use instruction **SUBS** to subtract 1 and set flags) and then branch back to the start of the loop if the result is non-zero (use instruction **BNE**). Exit the loop when the count reaches 0.

Print and submit your assembly language source program with the corresponding uVision Debug Window, showing the final results of the program. In the Debug Window, highlight the three data arrays in the Memory Window so we can verify that the third array is the sum of the first two arrays.

Discuss what you would change if the arrays comprised N bytes, rather than N words.

Suggestions:

- Use the debugger initialization file to write initial values to the three arrays (unless you prefer to do so interactively in the debug window.)
- Format the Memory Window of the debugger to display 32-bit signed numbers in decimal format (to simplify verifying the results.)