

**ELEC 2220 Computer Systems**  
**Homework #9**  
**Due: Monday, June 16**

In this exercise, you are to “compile” the following C program fragment into CPU12 assembly language and test it. Note that the calculation is the similar to that of homework #7, except that it is repeated multiple times, each time for corresponding elements of several arrays.

```
char   bob[5];           // array of 5 8-bit values
char   joe[5];           // array of 5 8-bit values
char   ann[5];           // array of 5 8-bit values
char   pat[5];           // array of 5 8-bit values
char   i;                // 8-bit value
for (i=0; i<5; i=i+1) {           //execute loop 5 times, for i=0, 1, 2, 3, 4
    bob[i] = (bob[i] + joe[i]) - (ann[i] + 8) + pat[i]; //perform calculation on ith array elements
}
```

Suggestions:

1. In RAM, define four 5-byte arrays: bob, joe, ann, pat, and one byte for variable i.
2. Define the following initial (decimal) values for the arrays (use “dc.b”). The initial value of i should be undefined (use “ds.b”).  
bob = {1,2,-3,-4,5}  
joe = {30,-25,-30,40,50}  
ann = {15,20,-30,-5,25}  
pat = {12, -8, 22, -38,52}
3. Write the program to perform the “for loop”.
4. As the last instruction of the program use: **bra \***  
This instruction branches to itself in an “endless loop”, so that your windows will not go blank.
5. Assemble and test the program in CodeWarrior. You may either single-step through the program, or else set a breakpoint at the last instruction and run to that point.
6. In the debugger “variables” window, display all variables as “decimal”. This will facilitate checking the results.
7. Hand-calculate the expected results, and compare them to the final values in array bob.
8. Print and turn in your assembly language source program with the corresponding “debug” screen, captured from CodeWarrior, showing the final results of the program. In the debug screen, highlight the four data arrays in the Variables and Memory windows so we can easily view the results.

Suggestions:

1. The “for loop” begins by setting the value of variable i to 0.
2. Before performing the “body” of the loop, the program must check the terminating condition ( $i < 5$ ). If that condition is not true, then the program should jump to some instruction after the last instruction of the loop body. Otherwise, the loop body should be executed.
3. The terminating condition can be checked with either a “subtract” instruction (suba or subb) or a “compare” instruction (cmpa or cmpb). Both perform a subtract operation and set the condition flags. Note that the conditional branch instruction “bge” jumps if there is a “greater than or equal to” condition, and “blt” branches if there is a “less than” condition.
4. After the calculation, 1 should be added to variable i in memory, and the program should return to step 2. Note that memory variable i should always contain its current value.
5. In the body of the loop, indexed addressing must be used.