# Laboratory 3

# Substring Counting Program

**Concepts:**

* Writing programs in assembly using CodeWarrior

**Objectives:**

* Develop a program to meet a specific list of requirements in assembly using the CodeWarrior IDE, and debug it for correctness.

**Files Needed:**

* Lab03.zip from Blackboard

**Assignment**

Download Lab03.zip and unzip it. Open the project file in CodeWarrior. The program must count the number of times that a substring of bytes appears in a longer string of bytes.

The main assembly file already has several sets of test inputs entered. Note that the program’s inputs are stored in the same locations, so only one of them should be uncommented at a time. You can and should use additional sets of inputs to test your program.

Write a main program in assembly that meets the following requirements:

1. The address of the long string to be searched is supplied at memory location $3010. Its length in bytes is supplied as a two-byte unsigned integer in $3012.
2. The address of the substring string to be searched for is supplied at memory location $3014. Its length in bytes is supplied as a two-byte integer in $3016.
3. The program must store the number of times that the substring appears in the main string as a one-byte unsigned integer stored to $3018. Instances of the substring may overlap.
4. The program must preserve the inputs.

You may make the following assumptions:

1. The lengths of the long string and substring are both positive, non-zero values, and the length of the long string is higher the length of the substring.
2. The addresses of the strings point to valid memory, and the long string and substring do not overlap.

Demonstrate the working program to the instructor by submitting a video demo.

**Deliverables/Scoring:**

Successful demonstration of the program is required for acceptance of the lab report, then

* 10 points - Compliance with posted lab report guidelines.
* 90 points – Assembly code for the program.

Submit the deliverables according to the lab report guidelines posted on Blackboard. Note that a PDF report with the code and an assembly file must be submitted in a ZIP file.

Hints:

One of the main purposes of this program is to overload the amount registers available and require several variables to be stored in memory. Locations $3000 to $300F were skipped to give you room for this (not that all 16 bytes are needed).

The program is most easily written as a loop nested inside a loop. The inner loop should be close to the program written for lab 2 with a few changes. One index register is a pointer to the substring, and the other index register points to the long string. The inner loop should have a counter set to length of the substring. It checks byte-by byte by incrementing the index registers to determine if the substring matches an equal-sized part of the long string.

The outer loop should reset the substring pointer substring back to the beginning of the substring, and it should set the long string’s pointer to the next byte of the long string. I.e. start searching for the substring at the first byte of the long string in the first pass, then start searching for the substring at the second byte of the long string in the second pass, and so on.

The outer loop does not execute once for each byte in the long string, since this may match bytes past the end of the long string. For example, a 3-byte short string can only be found in a 5-byte long string in three places; bytes 1, 2, 3, in bytes 2, 3, 4, or in bytes 3, 4, 5.