CPE 325: Intro to Embedded Computer System

Lab01

Laboratory Assignment #1

Submitted by: Hannah Kelley

Date of Experiment: August 25, 2025

Report Deadline: September 3, 2025

Demonstration Deadline: September 3, 2025

Theory

Topic 1: Code Composer Studio Tools and Features

a) The Memory Window is used to show memory contents and how they change during the runtime of a program. It shows every memory address and its contents. You can also change the contents of memory from this window.
b) The Console Window is used to simulated inputs and outputs like a terminal window. Compiler errors and warnings are also displayed here.
c) The Variable Window is used to view the names, values, types, and locations of all variables used in your program. It is useful because it lets you watch how data changes in real time, making it easier to verify logic and spot errors.
d) Breakpoints are markers you set in a program's code that tell a debugger to pause execution at specific lines. They're useful because they let you inspect variables, program flow, and memory step by step, helping you identify and fix bugs more efficiently.

Topic 2: Code Composer Studio Commands

a) In Code Composer Studio, you can control program execution using several debugging commands. Resume (F8) runs the program continuously until it either finishes or hits a breakpoint, while Suspend (Alt+F8) pauses execution at the current instruction. Terminate (Shift+F8) completely stops the program and ends the debug session. For stepping through code Step Into (F5) executes the next line and enters any function that is called, while Step Over (F6) executes the next line but runs called functions in a single step. Step Out (F7) continues running until the current function finishes and then pauses at the line following the function call in the caller. Finally, the Restart command halts execution and reruns the program from the beginning without leaving the debug session. These commands together allow you to run the program freely, pause it, or move through code step by step to analyze its behavior.

Results & Observation

Program 1:

Program Description:

This program implements a Caesar cipher in C to encrypt a given message by shifting the ASCII values of alphabetical characters three positions to the right. After the shift all letters retain their case (upper and lower). Non-alphabetical characters such as numbers, spaces, and punctuation remain unchanged.

Program Output:

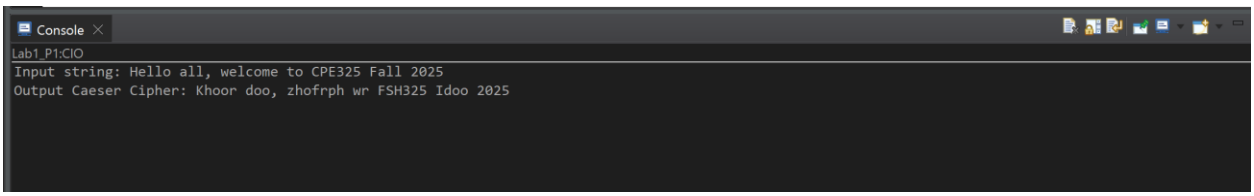

Figure 01: Caesar Cipher Output

Program Flowchart:

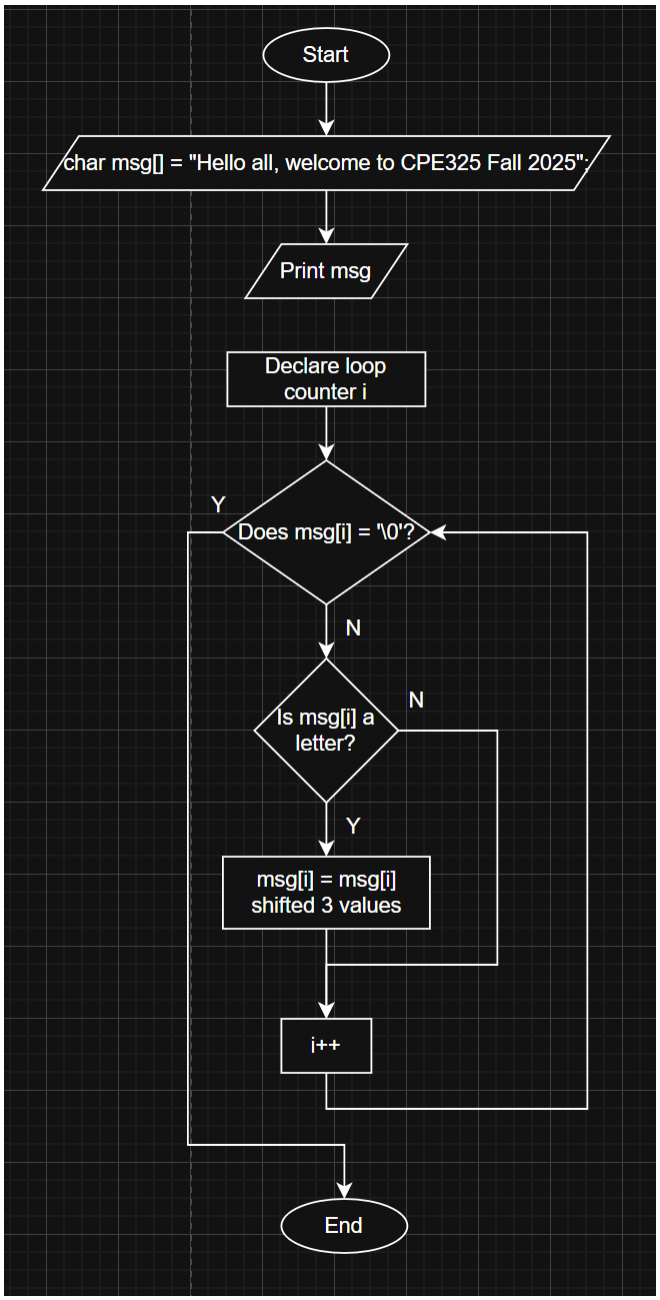

Figure 02: Caesar Cipher Flowchart

Program 2:

Program Description:

This program computes the greatest common divisor (GCD) of two integers using the Euclidean algorithm. The original version contained both syntax and logical errors, which were corrected to ensure successful compilation and accurate results. The corrected program works for any integer inputs and displays the input values along with their GCD as output.Program Output:

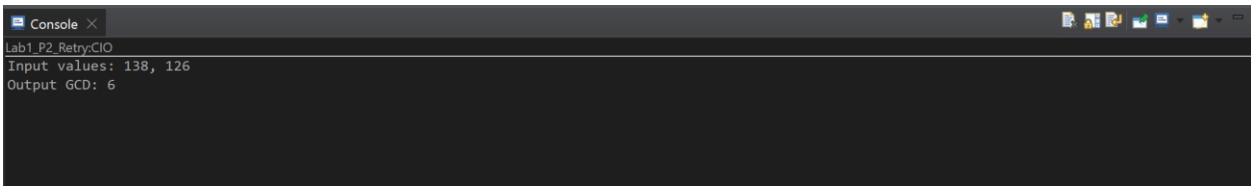

Figure 03: Euclidean Algorithm Output

Program Output:

For the input values 138 and 126 the get_gcd function takes 567 clock cycles to run. For the input values 150 and 144 the get_gcd function takes 383 clock cycles to run. For the input values 1444 and 600 the get_gcd function takes 566 clcok cycles to run. The clock cycle counts suggest that the runtime of get_gcd does not simply increase with larger input values. Instead, it depends on the relationship between the two numbers and how quickly the Euclidean algorithm reduces them to their greatest common divisor.

Program Flowchart:

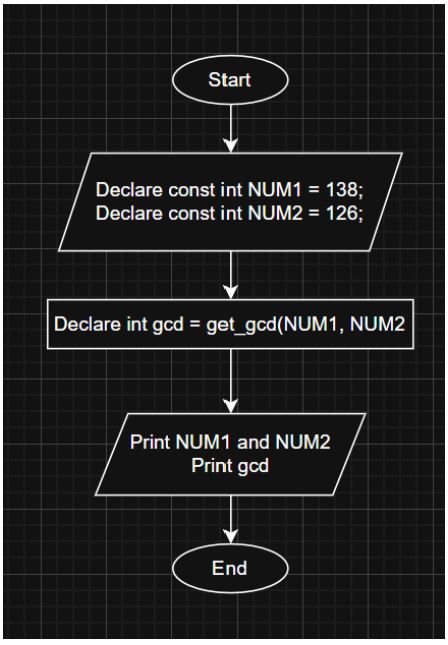

Figure 04: Euclidean Algorithm Flowchart

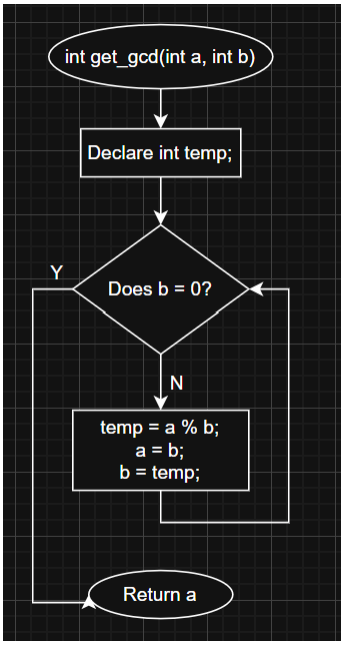

Figure 05: get_gcd Function Flowchart

Appendix

Table 01: Program 1 source code

```
/*-------------------------------------------------------------------------
 * File:        Lab01_P2.c
 * Description: Use a Caesar Cipher to shift a string by 3 characters
 *
 * Board:       5529
 * Input:       the string to shift (array of char)
 * Output:      the shifted string
 * Author:      Hannah Kelley
 * Date:        August 26, 2025
 *-------------------------------------------------------------------------*/
#include <msp430.h>
#include <stdio.h>
#include <string.h>  // for strlen

int main(void) {
    // stop watchdog timer
    WDTCTL = WDTPW | WDTHOLD;

    // Hardcoded string input value
    char msg[] = "Hello all, welcome to CPE325 Fall 2025";
    printf("Input string: %s\n", msg);

    // Caesar cipher +3
    int i; // didn't work unless it was declared her
    for (i = 0; msg[i] != '\0'; i++) { // loop through char array
        char c = msg[i]; // get the char to be considered

        // if the char is an uppercase letter
        if ( c >= 'A' && c <= 'Z') {
            // shift up three letters with wraparound
            msg[i] = ((c - 'A' + 3) % 26) + 'A';
        }
        // if the char is a lowercase letter
        else if (c >= 'a' && c <= 'z') {
            // shift up three letters with wraparound
            msg[i] = ((c - 'a' + 3) % 26) + 'a';
        }
    }

    // Print input values and output GCD
    printf("Output Caeser Cipher: %s\n", msg);
    return 0;   // end of program
}
```

Table 02: Program 2 source code

```
/*-------------------------------------------------------------------------
 * File:        Lab01_P2.c
 * Description: This function calculates the greatest common divisor of two
integers using the Euclidean algorithm
 *
 * Board:       5529
 * Input:       two integer values 'a' and 'b'
 * Output:      the GCD of `a` and `b`
 * Author:      Md Firoz Mahmud
 * Date:        May 25, 2025
 *-------------------------------------------------------------------------*/
#include <msp430.h>
#include <stdio.h>

// This function calculates the greatest common divisor of two integers
using the Euclidean algorithm
// Function returns the GCD of `a` and `b`
int get_gcd(int a, int b) {
    int temp;
    while (b != 0) {
        temp = a % b;
        a = b;
        b = temp; // Correctly calculates remainder to continue the
algorithm
    }
    return a;
}

int main(void) {
    // stop watchdog timer
    WDTCTL = WDTPW | WDTHOLD;

    // Hardcoded input values
    const int NUM1 = 138;
    const int NUM2 = 126;

    // Calculate GCD
    int gcd = get_gcd(NUM1, NUM2);

    // Print input values and output GCD
    printf("Input values: %d, %d\n", NUM1, NUM2);
    printf("Output GCD: %d\n", gcd); // Corrected for clear and correct
formatting

    return 0;

}
```