CPE301 – SPRING 2019

Design Assignment 1B

Student Name: Rocky Yasuaki Gonzalez
Student #: 5003229733
Student Email: gonzar14@unlv.nevada.edu
Primary Github address: https://github.com/rockyg1995/ihswppdar.git
Directory: C:\Users\rocky\Documents\CpE 301+L - Embedded Systems Design\CpE 301\Repository\DesignAssignments\DA1B

Submit the following for all Labs:

1. In the document, for each task submit the modified or included code (only) with highlights and justifications of the modifications. Also, include the comments.
2. Use the previously create a Github repository with a random name (no CPE/301, Lastname, Firstname). Place all labs under the root folder ESD301/DA, sub-folder named LABXX, with one document and one video link file for each lab, place modified asm/c files named as LabXX-TYY.asm/c.
3. If multiple asm/c files or other libraries are used, create a folder LabXX-TYY and place these files inside the folder.
4. The folder should have a) Word document (see template), b) source code file(s) and other include files, c) text file with youtube video links (see template).

1. COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS

N/A, (Atmel Studio 7 Project Only)

2. INITIAL/DEVELOPED CODE OF TASK 1/A

```
; DA1B - Rocky Gonzalez.asm
;
; Created: 2/17/2019 8:17:42 PM
; Author: rocky
;

.include <m328pdef.inc>     ; Include library for .SET and .ORG directives

.SET STARTADDS = 0x0200     ; Create 'STARTADDS' to begin at address '0x0200'
.SET DIV = 3                ; Create 'DIV' to be a value of '3'

       LDI R23, 99                ; Store the value '99' into Counter (register R19)
       LDI XL, LOW(STARTADDS)     ; Store address in Lower X-pointer
       LDI XH, HIGH(STARTADDS)    ; Store address in Upper X-pointer
       LDI YL, LOW(0x0400)        ; Store address in Lower Y-pointer
       LDI YH, HIGH(0x0400)       ; Store address in Upper Y-pointer
       LDI ZL, LOW(0x0600)        ; Store address in Lower Z-pointer
       LDI ZH, HIGH(0x0600)       ; Store address in Upper Z-pointer
       LDI R20, 0x07              ; Arbitrary value, consecutive addition executed
       MOV R21, R20               ; Retain Arbitrary Value (Since 'R16' will Change)

; ---------------------------------------------------------------------------------
; Loop to Store '7, 14, ..., n' Values Starting at Address 0x0200 (Using X-Pointer)
Cont1: CPI R20, 11   ; Check if 'R20 > 10'
       BRLO L1       ; If 'R20 <= 10', Add '7' More to 'R20'
       RJMP LoadY    ; Load Value into Y or Z Address depending on Divisibility
Cont2: ST  X+, R20   ; Store 'R20' into 'X', then increment address location
       ADD R20, R21  ; Increment 'R20' with itself (consecutive addition)
       DEC R23       ; Else, Continue to Decrement Counter Value (register R23)
       BRNE Cont1    ; Repeat 'Cont1' until all values are in address starting at 0x0200
       RJMP end      ; Check if values are divisible by 3, Store in Y and Z Pointers

L1:    ADD R20, R21  ; Add '7' more so that 'R20 > 10'
       RJMP Cont1    ; Recheck
; ---------------------------------------------------------------------------------
; Store Divisible Values by 3 into 'Y', and Non-Divisible Values into 'Z'
LoadY: MOV R22, R20         ; Store 'R20' into temporary register 'R22'
Repeat:
       SUBI R22, DIV        ; Subtract the value '3' from 'R22'
       CPI R22, DIV+1       ; Check if 'R22 > 3', Set 'Carry' for SREG when 'R22 <= 3'
       BRSH Repeat          ; If 'R22 > 3', Repeat subtracting '3'
       SUBI R22, DIV        ; Otherwise, Subtract '3' Once More to Check Divisible by '3'
       BRNE LoadZ           ; If Not Divisible, Load Value into Z Address

       ST Y+, R20           ; Else, Load Divisible Value into Y Address
       Add R16, R20         ; Add value into R17:R16
       BRCS ext1            ; If Overflow in R16, increment value into R17
       RJMP Cont2           ; Continue back to Original Operation
```

```
LoadZ: ST Z+, R20           ; Load Non-Divisible Value into Z-Address
       Add R18, R20         ; Add value into R19:R18
       BRCS ext2            ; If Overflow in R18, increment value into R19
       RJMP Cont2           ; Continue back to Original Operation
; ---------------------------------------------------------------------------------
; Incrementing R17:R16 (High Register)
ext1:  CLC                  ; Clear Carry in Status Register
       INC R17              ; Increment R!7:R16 (High Register)
       RJMP Cont2           ; Continue Original Loop
; ---------------------------------------------------------------------------------
; Incrementing R19:R18 (High Register)
ext2:  CLC                  ; Clear Carry in Status Register
       INC R19              ; Increment R!9:R18 (High Register)
       RJMP Cont2           ; Continue Original Loop
; ---------------------------------------------------------------------------------
; End of Program
end:   RJMP end             ; End Program/Loop Forever
```

3. DEVELOPED (VERIFICATION) CODE OF TASK 2/A from TASK 1/A

```
/// @author Rocky Gonzalez
/// @note da1b_arrays
/// @file da1b_arrays.cpp
/// @version 2019-02-23
/// @brief The program is used to verify values for DA1B.

#include <iostream>

using namespace std;

int main() {

    unsigned int array_x[100];     // Create an array to store X-Pointer Values
    unsigned int array_y[100];     // Create an array to store Y-pointer Values
    unsigned int array_z[100];     // Create an array to store Z-Pointer Values

    unsigned char store = 0;       // Values that range from "0-255" (Byte)
    unsigned char i;               // Iteration for Array X
    int total_y = 0;               // Total Value of Y Array
    int total_z = 0;               // Total Value of Z Array

    cout << "Array X" << endl;     // Display "Array X" into Monitor
    for (i = 0; i < 99; i++) {     // Loop and Store Values of "7" greater than 10
        if (store <= 10) {         //
            store = store + 7;     // If not greater than 10, add 7 more
        }
        if (store <= 10) {         //
            store = store + 7;     // If still not greater than 10, add 7 more
        }
        array_x[i] = store;        // Store Values into Array X
        cout << array_x[i] << " "; // Display into Terminal
        store = store + 7;         // Increment "7"
    }
    cout << endl << endl;          // Line Break

    unsigned char j = 0;           // Iteration for Array Y
```

```
    unsigned char k = 0;          // Iteration for Array Z

    cout << "Array Y" << endl;     // Display "Array Y" into Monitor
    for (i = 0; i < 99; i++) {     // Loop X Data
        if(array_x[i]%3 == 0) {    // If divisible by 3, store it into Array Y and Display
            array_y[j] = array_x[i];            // Store into Y Data
            total_y = total_y + array_x[i];     // Add Value (total)
            j = j++;                            // Increment Y Array
            cout << array_y[j] << " ";          // Create List of Y Values
        }
    }
    cout << endl << endl;

    cout << "Array Z" << endl;     // Display "Array Z" into Monitor
    for (i = 0; i < 99; i++) {     // Loop X Data
        if(array_x[i]%3 != 0) {    // If not divisible by 3, Store in Array Z and Display
            array_z[k] = array_x[i];            // Store into Z Data
            total_z = total_z + array_x[i];     // Add Value (total)
            k = k++;                            // Increment Z Array
            cout << array_z[k] << " ";          // Create List of Z Values
        }
    }
    cout << endl << endl;                                  // Line Break
    cout << "Total Y: " << total_y << endl << endl;    // Totality of Y Values
    cout << "Total Z: " << total_z << endl << endl;    // Totality of Z Values
}
```

4. SCHEMATICS

N/A (Assembly Coding Only)

5. SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)

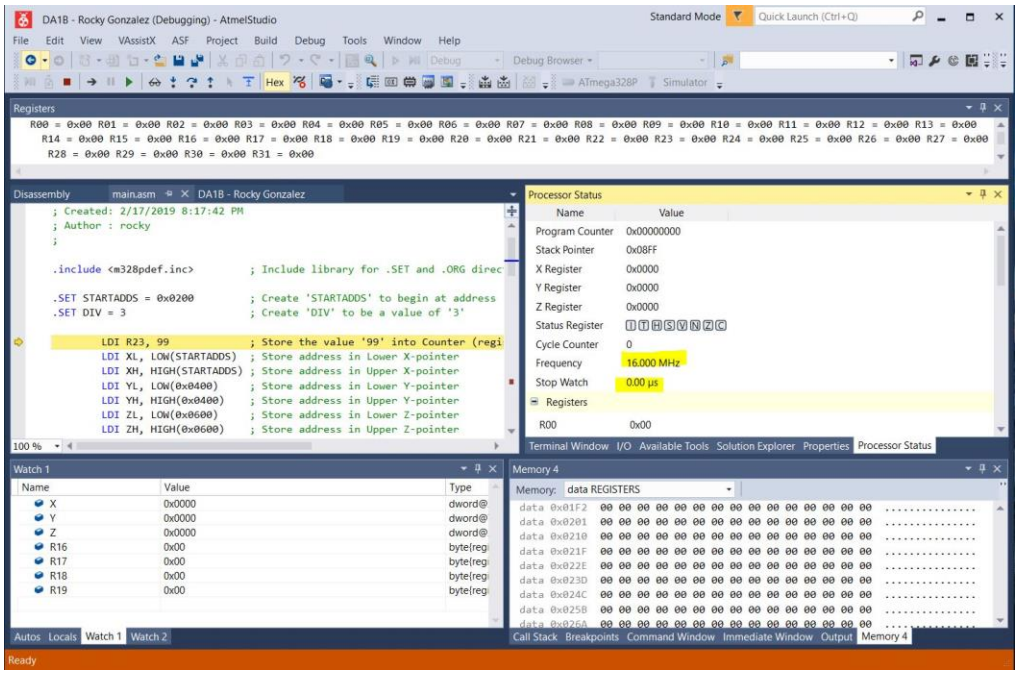

Figure 1a – Before Start of Program Execution

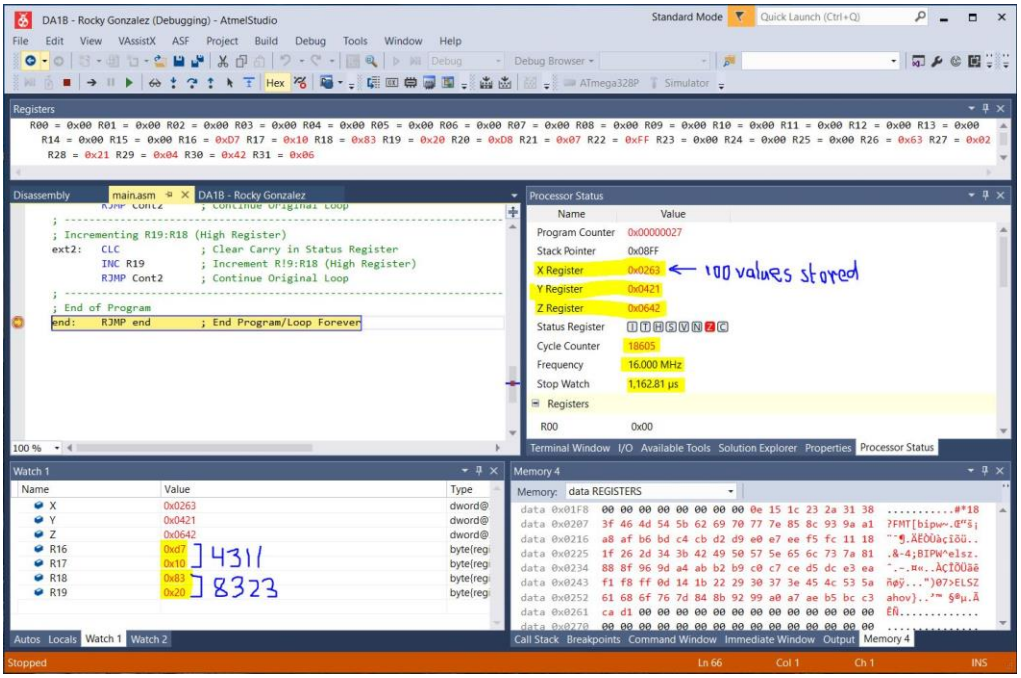

Figure 1b – Total Values from R17:R16 (4,311) and R19:R18 (8,323)/Total Time of Execution (1,162.61µs)

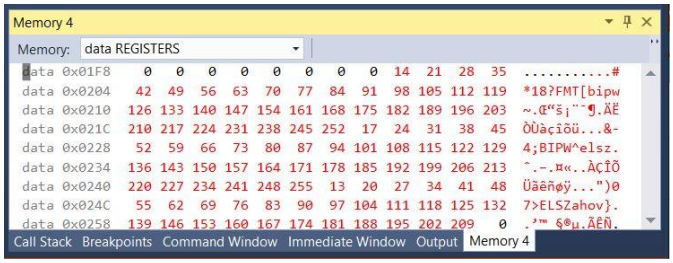

Figure 1c – All Values Stored from X-Pointer (Starting at 0x0200)

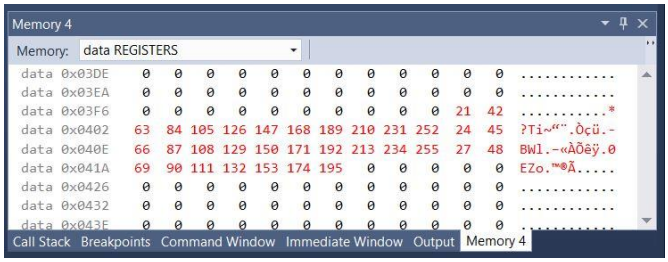

Figure 1d – Divisible Values Stored from Y-Pointer (Starting at 0x0400)

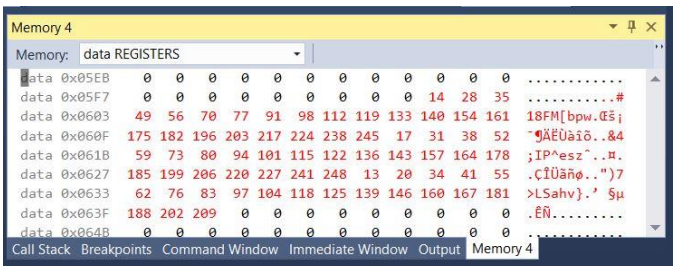

Figure 1e – Non-divisible Values Stored from Z-Pointer (Starting at 0x0600)

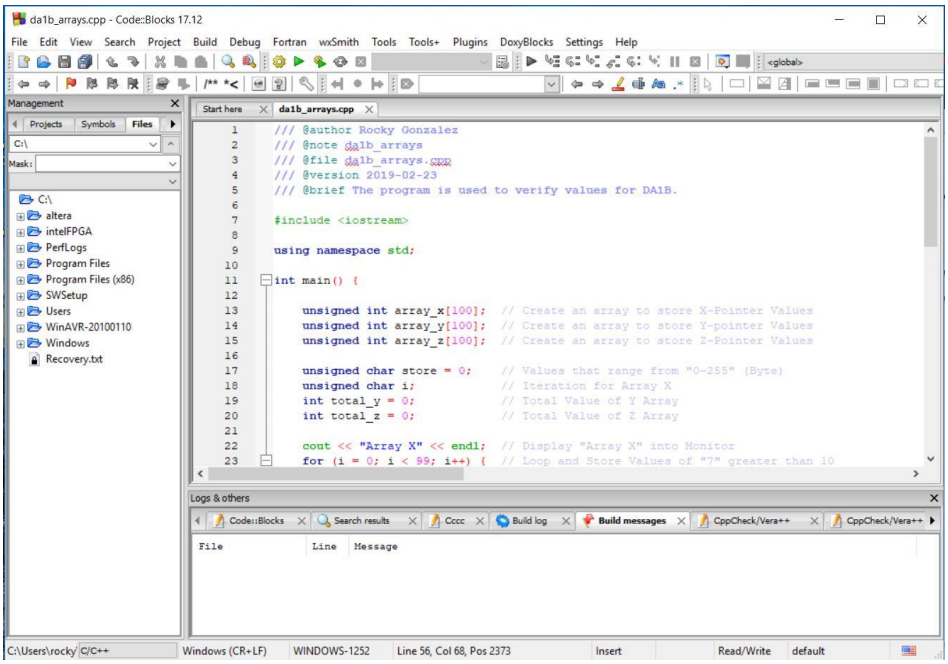

Figure 1f – Verification using C++ through CodeBlocks

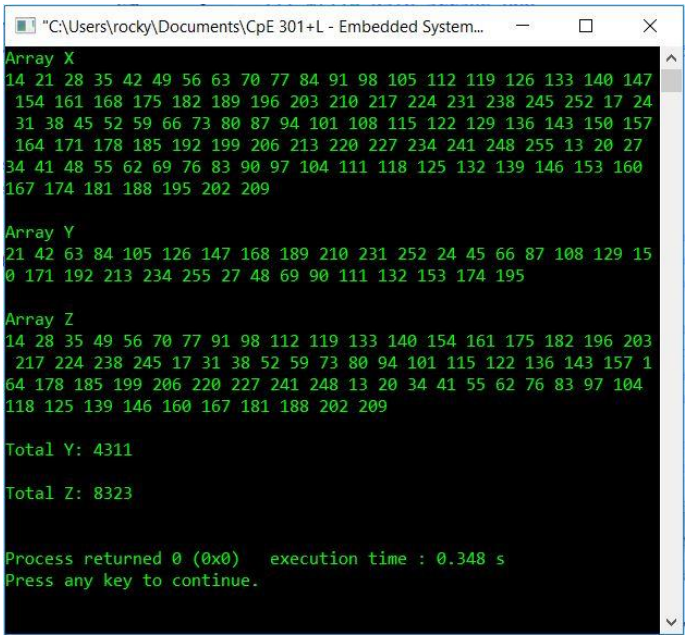

Figure 1g – Output Monitor from C++ Coding

6. SCREENSHOT OF EACH DEMO (BOARD SETUP)

N/A (Assembly Coding Only)

7. VIDEO LINKS OF EACH DEMO

https://youtu.be/lM5o84oR52E

8. GITHUB LINK OF THIS DA

C:\Users\rocky\Documents\CpE 301+L - Embedded Systems Design\CpE 301\Repository\DesignAssignments\DA1B

Student Academic Misconduct Policy
http://studentconduct.unlv.edu/misconduct/policy.html

"*This assignment submission is my own, original work*".
Rocky Gonzalez