CPE301 – SPRING 2019

Design Assignment 1B

Student Name: Cody McDonald

Student #: 5000382538

Student Email: mcdonc4@unlv.nevada.edu

Primary Github address: https://github.com/elev8rProcrastinator/submission\_da.git

Directory:

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**

Atmel Studio 7

1. **INITIAL/MODIFIED/DEVELOPED CODE OF TASK 1/A**

My entire program:

;

; CPE310\_DA1B.asm

;

; Created: 2/22/2019 3:05:56 PM

; Author : Cody

;

.CSEG

.ORG 0X00

;Initialize all constants

.EQU COUNT = 99

.EQU STARTADDS = 0x200 ;address for all numbers being filled in

.EQU DIVISIBLE = 0x400 ;address for all divisble numbers

.EQU NOTDIV = 0x600 ;address for all non divisible numbers

;Set all high and low bits for the address registers

LDI XL, LOW(STARTADDS)

LDI XH, HIGH(STARTADDS)

LDI YL, LOW(DIVISIBLE)

LDI YH, HIGH(DIVISIBLE)

LDI ZL, LOW(NOTDIV)

LDI ZH, HIGH(NOTDIV)

;set dummy and count registers

LDI R23, COUNT

LDI R20, 0

;ADD HIGH AND LOW

MOV R20, XL

ADD R20, XH

;cycle through the numbers that need to be sorted

INCREMENT1:

INC R20

CPI R20, 11

BREQ STORENUM

RJMP INCREMENT1

INCREMENT2:

INC R20

;store each number into x register and increment the value

STORENUM:

ST X+,R20

DEC R23

BRNE INCREMENT2

;PROBLEM 2: PARSE THROUGH CODE

LDI XL, LOW(STARTADDS) ;start at initial address

LDI XH, HIGH(STARTADDS)

;start loop for parsing

DIV3:

LD R22, X+

MOV R24, R22

CPI R22, 0

BREQ PROB3

;check if it's divisble

DIVIDE:

SUBI R22,3

BREQ divisible\_store

BRMI negative\_store

RJMP DIVIDE

;store value into y register if divisbile

divisible\_store:

ST Y+,R24

RJMP DIV3

;store into z if the number is not divisible by 3

negative\_store:

ST Z+,R24

RJMP DIV3

;Problem 3

;Initialize the pointers to initial addresses

PROB3:

LDI YL, LOW(DIVISIBLE)

LDI YH, HIGH(DIVISIBLE)

LDI ZL, LOW(NOTDIV)

LDI ZH, HIGH(NOTDIV)

;Sum all Y pointer values

SUMY:

LDI R20,0 ;dummy register

LD R21,Y+ ;check when Y is empty register

ADD R16, R21 ;add each value into R18

ADC R17, R20 ;add each carry into R19

CPI R21,0 ;Compare R21 with 0

BREQ SUMZ ;Stop loop when 0 value is found and go to sumz

RJMP SUMY ;jump back to sumz if 0 condition isn't satisfied

SUMZ:

LDI R20,0 ;dummy register

LD R21,Z+ ;check when Z is empty register

ADD R18, R21 ;add each value into R18

ADC R19, R20 ;add each carry into R19

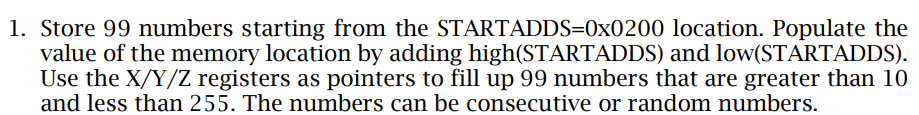
CPI R21,0 ;Compare R21 with 0

BREQ end ;Stop loop when 0 value is found

RJMP SUMZ ;jump back to sumz if 0 condition isn't satisfied

end: rjmp end

1. **DEVELOPED MODIFIED CODE OF TASK 2/A from TASK 1/A**



;Initialize all constants

.EQU COUNT = 99

.EQU STARTADDS = 0x200 ;address for all numbers being filled in

.EQU DIVISIBLE = 0x400 ;address for all divisble numbers

.EQU NOTDIV = 0x600 ;address for all non divisible numbers

;Set all high and low bits for the address registers

LDI XL, LOW(STARTADDS)

LDI XH, HIGH(STARTADDS)

LDI YL, LOW(DIVISIBLE)

LDI YH, HIGH(DIVISIBLE)

LDI ZL, LOW(NOTDIV)

LDI ZH, HIGH(NOTDIV)

;set dummy and count registers

LDI R23, COUNT

LDI R20, 0

;ADD HIGH AND LOW

MOV R20, XL

ADD R20, XH

;cycle through the numbers that need to be sorted

INCREMENT1:

INC R20

CPI R20, 11

BREQ STORENUM

RJMP INCREMENT1

INCREMENT2:

INC R20

;store each number into x register and increment the value

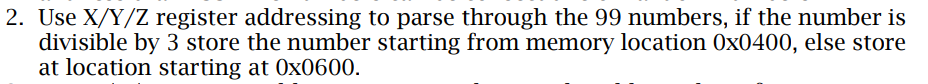
STORENUM:

ST X+,R20

DEC R23

BRNE INCREMENT2

1. **DEVELOPED MODIFIED CODE OF TASK 2/A from TASK 1/B**



;PROBLEM 2: PARSE THROUGH CODE

LDI XL, LOW(STARTADDS) ;start at initial address

LDI XH, HIGH(STARTADDS)

;start loop for parsing

DIV3:

LD R22, X+

MOV R24, R22

CPI R22, 0

BREQ PROB3

;check if it's divisble

DIVIDE:

SUBI R22,3

BREQ divisible\_store

BRMI negative\_store

RJMP DIVIDE

;store value into y register if divisbile

divisible\_store:

ST Y+,R24

RJMP DIV3

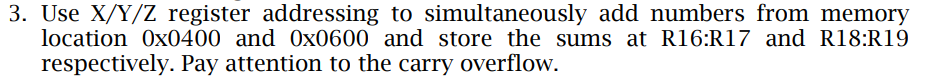
;store into z if the number is not divisible by 3

negative\_store:

ST Z+,R24

RJMP DIV3

1. **DEVELOPED MODIFIED CODE OF TASK 2/A from TASK 1/C**



;Problem 3

;Initialize the pointers to initial addresses

PROB3:

LDI YL, LOW(DIVISIBLE)

LDI YH, HIGH(DIVISIBLE)

LDI ZL, LOW(NOTDIV)

LDI ZH, HIGH(NOTDIV)

;Sum all Y pointer values

SUMY:

LDI R20,0 ;dummy register

LD R21,Y+ ;check when Y is empty register

ADD R16, R21 ;add each value into R18

ADC R17, R20 ;add each carry into R19

CPI R21,0 ;Compare R21 with 0

BREQ SUMZ ;Stop loop when 0 value is found and go to sumz

RJMP SUMY ;jump back to sumz if 0 condition isn't satisfied

SUMZ:

LDI R20,0 ;dummy register

LD R21,Z+ ;check when Z is empty register

ADD R18, R21 ;add each value into R18

ADC R19, R20 ;add each carry into R19

CPI R21,0 ;Compare R21 with 0

BREQ end ;Stop loop when 0 value is found

RJMP SUMZ ;jump back to sumz if 0 condition isn't satisfied

end: rjmp end

1. **DEVELOPED MODIFIED CODE OF TASK 2/A from TASK 1/D**



**Here is the c-program I created to verify my algorithm**

/\*

\* CPE301\_DA1B\_verification.c

\*

\* Created: 2/22/2019 10:14:20 PM

\* Author : Cody

\*/

#include <avr/io.h>

int main(void){

int \*x ;

int \*y ;

int \*z ;

// set addresses for each of the address registers

x = 0x0200;

y = 0x0400;

z = 0x0600;

int ySum = 0;

int zSum = 0;

//initialize some indices for loops

int i = 0;

int i2 = 0;

//Begin loop for inserting numbers into the sorting

for (i=0;i<=255;i++){

if (i <= 10 || i >= 255 ){ //decide whether the number is larger than 255 or less than 10

//do nothing

}

else if(i2 < 99){ // store only 99 numbers

\*x = i; // let x value equal i

x++; // set x up for next value

i2++; // increase index

//Check if divisible by 3

if (i%3 == 0){ //if divisible, store into y

\*y=i;

y++;

ySum = ySum +i; // add all values of y

}

else{ // otherwise store into z

\*z=i;

z++;

zSum = zSum +i; // add all values of z

}

}

//output the values for ySum and zSum and store them in the next spot of memory

\*y = ySum;

\*z = zSum;

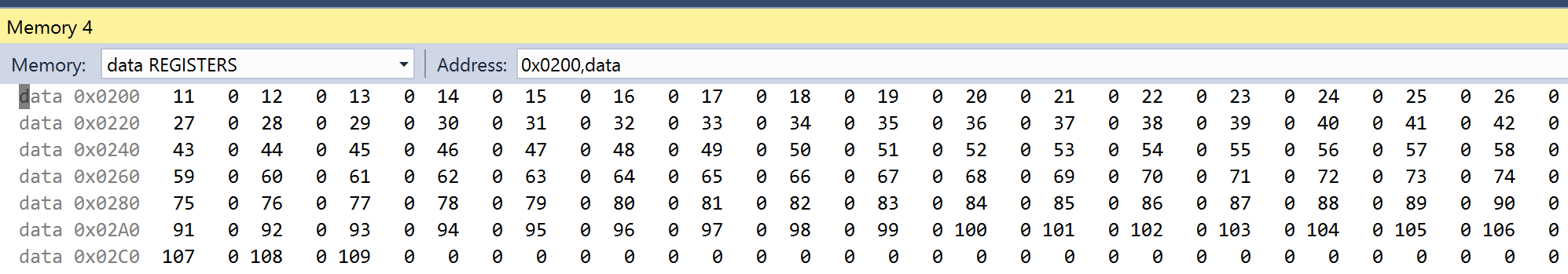
}

return 0;

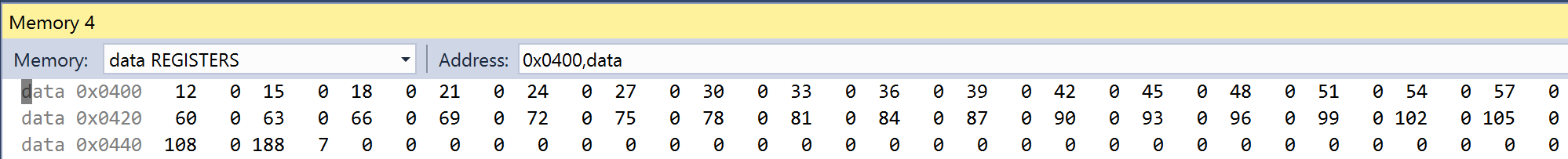
}

**Here are the memory locations which indicate the same values as the ones located in the assembly program:**

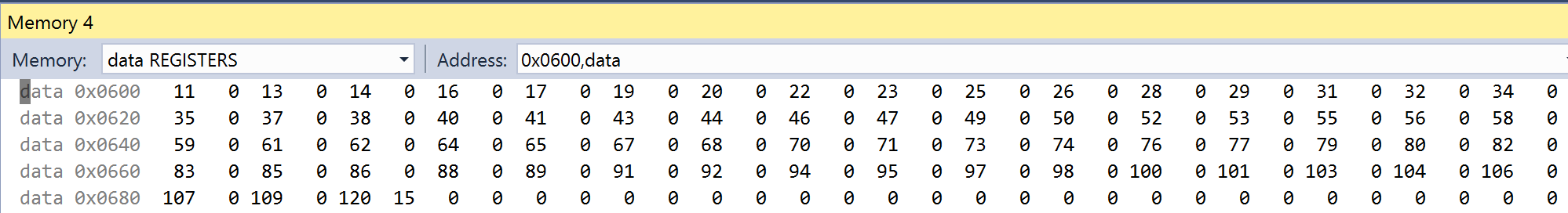
**X Register:**



**Y Register:**

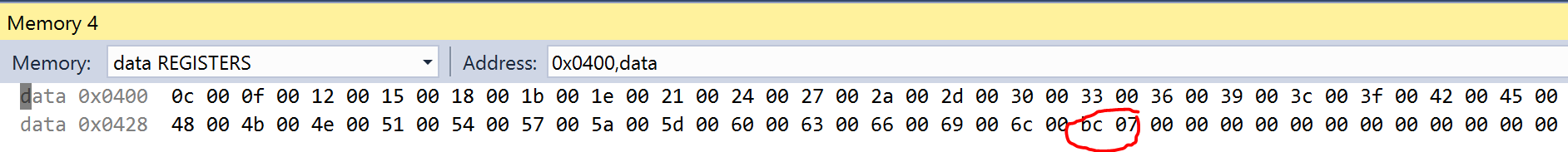


**Z Register:**

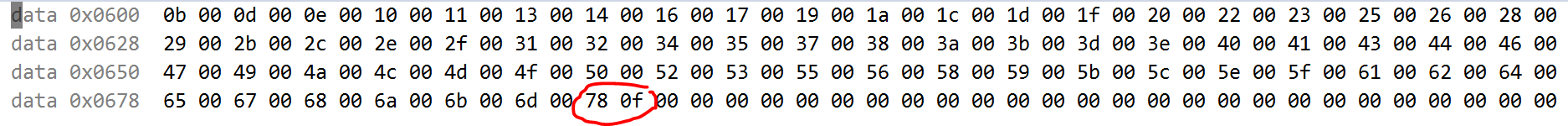


**Values were converted to hexadecimal to view the sums**

**ySum:**



**zSum:**

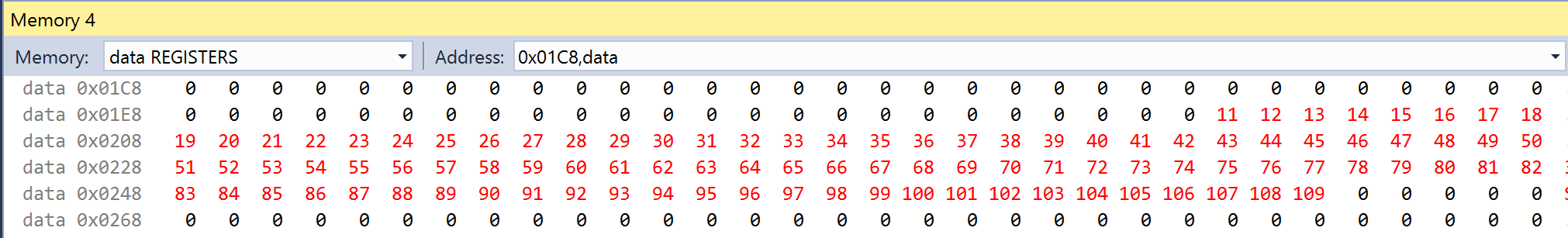


1. **SCHEMATICS**

N/A

1. **SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)**

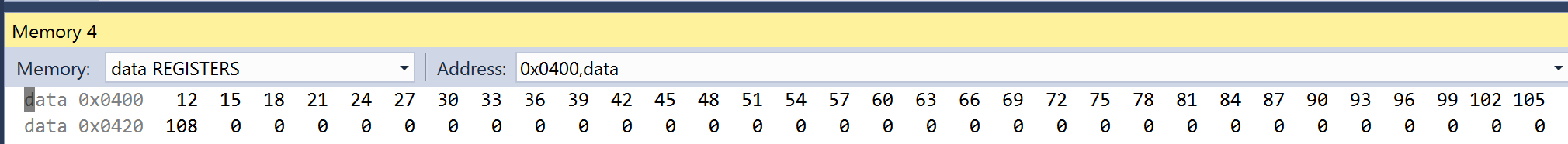
The first portion of this assignment required the above code to store 99 numbers greater than 10 and less than 255. Here is the memory location with the stored values:



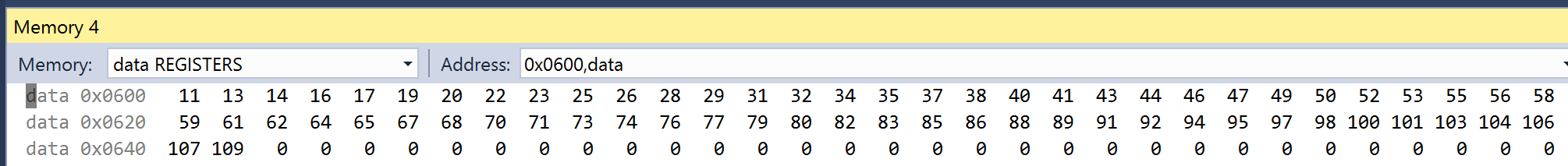
Values are shown in decimal to illustrate 99 numbers greater than 10 and less than 255

The second task required us to parse through all the values in the x register and sort the values divisible by 3 and not divisible into registers y and z respectively.

All divisible values stored into Y register from location 0x400:

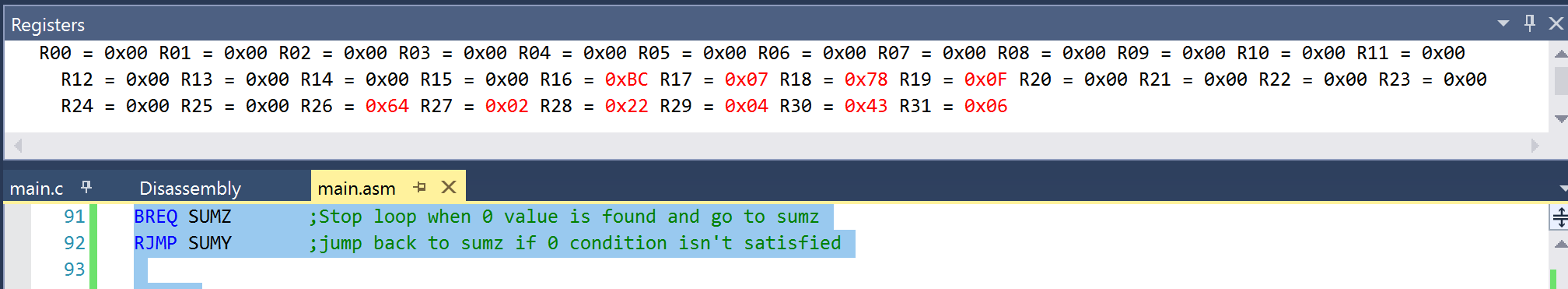


All non divisible values stored in Z register from location 0x600:

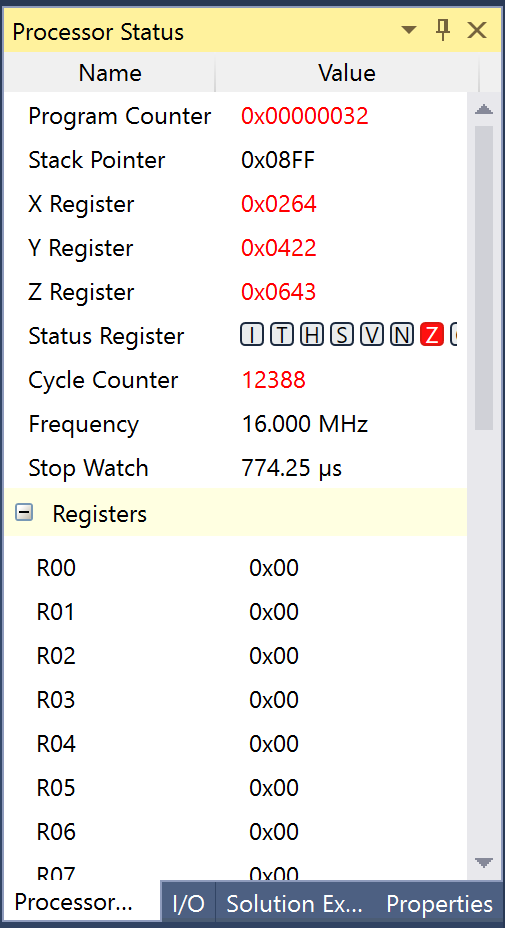


The 3rd portion required us to sum all the respective values in each of the y and z registers

Here is my sum for the y and z registers located between registers R16:R17 ($07BC) and R18:R19 ($0F78).



Here is my program’s runtime at 16MHz:



1. **SCREENSHOT OF EACH DEMO (BOARD SETUP)**

**N/A**

1. **VIDEO LINKS OF EACH DEMO**

N/A

1. **GITHUB LINK OF THIS DA**

**Student Academic Misconduct Policy**

<http://studentconduct.unlv.edu/misconduct/policy.html>

“This assignment submission is my own, original work”.

Cody McDonald