CPE301 – SPRING 2019

Design Assignment X

Student Name: Xianjie Cao

Student #: 5004222179

Student Email: [caox2@unlv.nevada.edu](mailto:caox2@unlv.nevada.edu)

Primary Github address: <https://github.com/c1029324620/Mocha.git>

Directory: Mocha/DesignAssignments/LAB1/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**

Atmel Stuido 7, Assembly language and Github.

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

;

; DA1B.asm

;

; Created: 2/11/2020 7:37:01 PM

; Author : c1029

;

; Replace with your application code

.include<m328pdef.inc>

.equ STARTADD = 0x0300

.equ DIVI7 = 0x0500

.equ DIVI3 = 0x0600

.equ DIVI\_BOTH = 0x700

.equ DIVI\_NOT = 0x800

.def num = r20

.def denominator = r21

.cseg

.org 0x00

ldi r17, 200 ;r17 =200, used as counter to store 200 numbers

ldi r16, 26 ;starts at 26

ldi ZL, LOW(STARTADD) ;load the low byte of Z with 0x00

ldi ZH, HIGH(STARTADD) ;load the high byte of Z with 0x03

Store\_LP:

st Z+, r16 ;copy r16 to memory location Z, then increment the Z register

inc r16 ;increment the value to be stored

dec r17 ;decrement the counter

brne Store\_LP ;loop until counter = 0

;finish storing

;pointers point to memory locations that data should be stored.

ldi r18, LOW(DIVI7)

ldi r19, HIGH(DIVI7)

ldi r22, LOW(DIVI3)

ldi r23, HIGH(DIVI3)

ldi r24, LOW(DIVI\_BOTH)

ldi r25, HIGH(DIVI\_BOTH)

ldi r26, LOW(DIVI\_NOT)

ldi r27, HIGH(DIVI\_NOT)

ldi r17, 200 ; used as counter to check the 200 numbers.

ldi ZL, LOW(STARTADD) ;start checking numbers

ldi ZH, HIGH(STARTADD)

check\_LP:

ld r16, Z+ ;load value in memory address Z into r16

mov num, r16 ;move r16 into num

ldi denominator, 7 ;set denominator to 7

check7\_LP:

sub num, denominator ;keet subtrac 7

brcc check7\_LP

add num, denominator ;num is the remainder now

brne set3\_LP ;check if the remainder is zero, if not, check if the number is divisible by 3 or not

breq set\_both ; if it is zero, check if the number is divisible by both or not

set3\_LP:

mov num, r16 ;reset num

ldi denominator, 3 ;set denominator to 3

check3\_LP:

sub num, denominator ;keep subtract 3

brcc check3\_LP

add num, denominator

brne store\_not ;if the remainder is not zero, store the number as not divisible by both.

mov YL, r22 ;store to DIVI3

mov YH, r23 ; r23:r22 have the memory address that is for not divisible for both.

st Y, r16 ;store the number into DIVI\_NOT memory address.

inc r22 ;increment the memory address

dec r17 ; decrement the counter.

breq done\_checking ;if counter is equal to zero means we have done our checking, move to calculating sum.

jmp check\_LP

set\_both:

mov num, r16

ldi denominator, 3

check\_both: ;check if the number is divisible by both

sub num, denominator

brcc check\_both

add num, denominator

brne store\_7

;store into DIVI\_BOTH

mov YL, r24

mov YH, r25

st Y, r16

inc r24

dec r17

breq done\_checking

jmp check\_LP

store\_not:

mov YL, r26

mov YH, r27

st Y, r16

inc r26

dec r17

breq done\_checking

jmp check\_LP

store\_7:

mov YL, r18

mov YH, r19

st Y, r16

inc r18

dec r17

breq done\_checking

jmp check\_LP

done\_checking:

;calculating the sum

;sum for 0x500 is going to be stored at r18:r17:r16

ldi ZL, LOW(DIVI7)

ldi ZH, HIGH(DIVI7)

;clear register 18:17:16

clr r16

clr r17

clr r18

L1:

ld r29, Z+

add r16, r29

adc r17, r0 ; add carry with zero register

adc r18, r0 ; add carry with zero register

cpi r29, 0x00 ;compare with 0x00, 0x00 implies the end of data.

brne L1

;sum for 0x600 is going to be stored at r21:r20:r19

ldi ZL, LOW(DIVI3)

ldi ZH, HIGH(DIVI3)

clr r19

clr r20

clr r21

L2:

ld r29, Z+

add r19, r29

adc r20, r0

adc r21, r0

cpi r29, 0x00

brne L2

;sum for 0x700 is going to be stored at r24:r23:r22

ldi ZL, LOW(DIVI\_BOTH)

ldi ZH, HIGH(DIVI\_BOTH)

clr r22

clr r23

clr r24

L3:

ld r29, Z+

add r22, r29

adc r23, r0

adc r24, r0

cpi r29, 0x00

brne L3

;sum for 0x800 is going to be stored at r27:r26:r25

ldi ZL, LOW(DIVI\_NOT)

ldi ZH, HIGH(DIVI\_NOT)

clr r25

clr r26

clr r27

L4:

ld r29, Z+

add r25, r29

adc r26, r0

adc r27, r0

cpi r29, 0x00

brne L4

DONE:

jmp DONE

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

**C++ program to verify the result:**

#include<iostream>

using namespace *std*;

int main()

{

int x = 26;

int num[200] = {0}, divi7[200] = {0}, divi3[200] ={0}, divi\_both[200] = {0}, divi\_not[200] = {0}; //array to store data

//initialized to a zero array.

int cnt\_7 = 0, cnt\_3 = 0, cnt\_both = 0, cnt\_not = 0; //counters to count how many elements in each array

int sum\_7 = 0, sum\_3 = 0, sum\_both = 0, sum\_not = 0;

for(int i = 0; i < 200; i++) //generate consecutive numbers from 26 to 225, total of 200 numbers

{

num[i] = x + i;

}

//check if they are divisible by 7 or 3 or both or both not.

for(int i = 0; i < 200; i++)

{

if(num[i] % 7 == 0 && num[i] % 3 == 0)

{

divi\_both[cnt\_both++] = num[i];

}

else if(num[i] % 7 == 0)

{

divi7[cnt\_7++] = num[i];

}

else if(num[i] % 3 == 0)

{

divi3[cnt\_3++] = num[i];

}

else

{

divi\_not[cnt\_not++] = num[i];

}

}

//calculating the sum of each array.

for(int i = 0; i < 200; i++)

{

sum\_7 += divi7[i];

sum\_3 += divi3[i];

sum\_both += divi\_both[i];

sum\_not += divi\_not[i];

}

cout << hex << "sum\_7: " << sum\_7 << "\nsum\_3: " << sum\_3 << "\nsum\_both: " << sum\_both << "\nsum\_not: " << sum\_not;

cout << endl;

return 0;

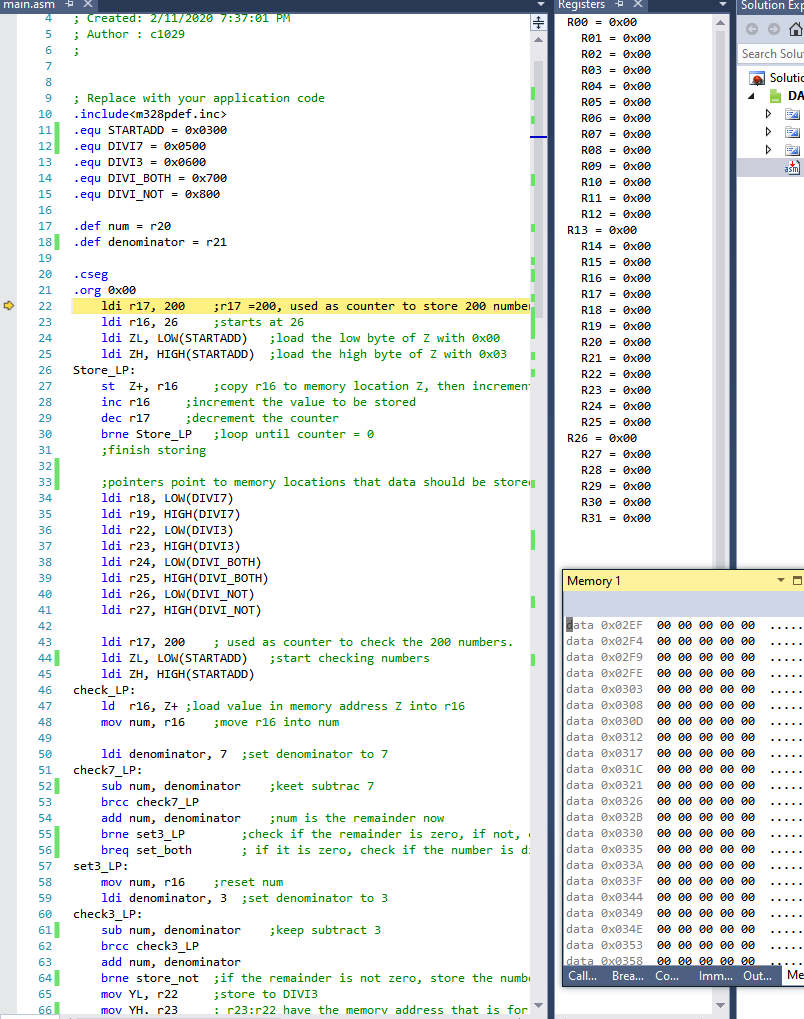
}

1. **SCHEMATICS**

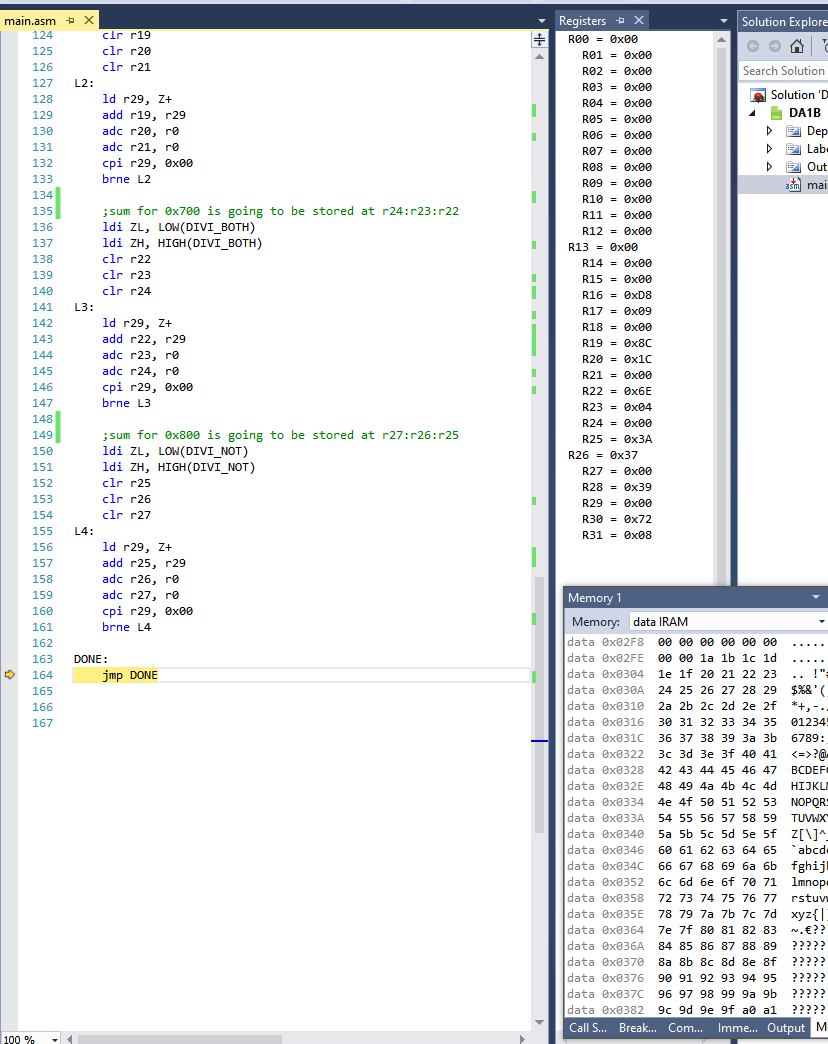
Schematic does not apply for this design assignment

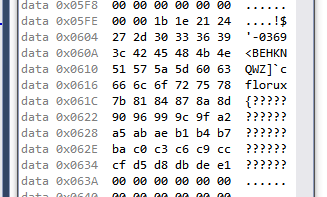
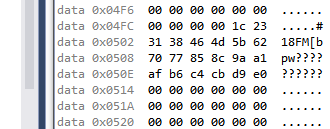
1. **SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)**

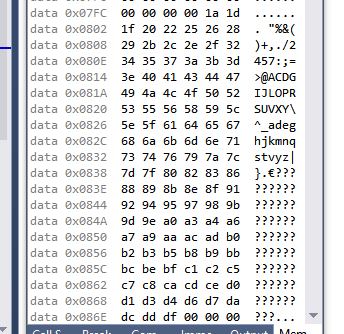
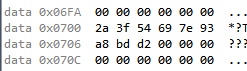
Program Starts:



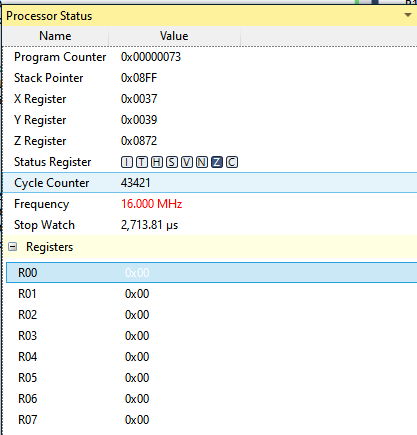
End of the program:







**The execution time and number of clock cycles at 16MHz is as following:**



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

Does not apply for this design assignment.

1. **VIDEO LINKS OF EACH DEMO**

<https://youtu.be/APCJmvFKM5s>

1. **GITHUB LINK OF THIS DA**

<https://github.com/c1029324620/Mocha.git>

**Student Academic Misconduct Policy**

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

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

Xianjie Cao