**CPE301 – SPRING 2020**

Design Assignment 1B

Student Name: Minsung Cho

Student #: 2001446442

Student Email: chom3@unlv.nevada.edu

Primary Github address: <https://github.com/cho-minsung?tab=repositories>

Directory: assignment1b

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**

No components used

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

Insert initial code here

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

the actual code

;

; assignment1b.asm

;

; Created: 2/21/2020 12:02:41 AM

; Author : Owner

;

.ORG 0x0

.EQU STARTADDS = 0X0300 ;location to store all the data

.EQU DIVBY7 = 0x0500 ;location to store data divisible by 7

.EQU DIVBY3 = 0x0600 ;location to store data divisible by 3

LDI R16, 0xFF ;255. I will decrement this by one and store it.

LDI R17, 0x19 ;25, lower cap of the data stored.

LDI R20, 7 ;register to add 7 to another register later

LDI R21, 21 ;register to add 21 to another register later

LDI R22, 3 ;register to add 3 to another register later

LDI XH, HIGH(STARTADDS) ;sets the location of X pointer

LDI XL, LOW(STARTADDS)

LDI YH, HIGH(DIVBY7) ;sets the location of Y pointer

LDI YL, LOW(DIVBY7)

LDI ZH, HIGH(DIVBY3) ;sets the location of Z pointer

LDI ZL, LOW(DIVBY3)

STORE: ;storing loop

ST X+, R16 ;store starting from 255 to pointer location X and increment

ADD R1, R16 ;r3:r1 represents the total addition of all data.

ADC R2, R0

ADC R3, R0

DEC R16 ;decrement the datum by 1

CP R16, R17 ;if the upper cap equals to the lower cap, it will kick out of STORE loop.

BRNE STORE

SORT: ;sorting preparation loop

LDI XH, HIGH(STARTADDS) ;resets the X pointer to the initial locaiton.

LDI XL, LOW(STARTADDS)

LDI R19, 0xe5 ;basically how many times the data are going to be tested. the 'i' of 'for' loop.

DIVTEST: ;one datum will go thru this test once and will be sorted out.

LD R24, X ;temporary registers to store and decrement for individual scenarios. the last register is not touched because it will be the one that will be stored.

LD R25, X

LD R17, X

LD R16, X+

CLC ;just in case, because we need to use carry to determine the negativity.

SUB21: ;subtracts 21 to find if the datum is divisible by both 3 and 7 because 21 is a multiple of those.

SUBI R24, 21

BRCC SUB21 ;if the number goes to negative, it will kick out of the subtraction loop.

ADD R24, R21 ;add back for the remainder determination.

CPI R24, 0

BREQ BOTH ;if the remainder is 0, then it will send to loop to store the datum to add all the 21-divisible numbers.

SUB7: ;same goes with 7. Bigger number first.

SUBI R25, 7

BRCC SUB7

ADD R25, R20

CPI R25, 0

BREQ DIV7

SUB3:

SUBI R17, 3

BRCC SUB3

ADD R17, R22

CPI R17, 0

BREQ DIV3

NEITHER: ;if the number went through all 21, 7, and 3 and still cannot be situated, it will be added with the rest of the non-divisible numbers.

ADD R13, R16 ;simultaneously adding the values.

ADC R14, R0

ADC R15, R0

DEC R19 ;counter is decremented and it will be kicked out if it reaches 0.

BRNE DIVTEST

CPI R19, 0x0

BREQ END

DIV3: ;the same storing process with NEITHER but for 3 and 7, I use the Y and Z pointers to store individual numbers as arrays.

ST Z+, R16

ADD R4, R16

ADC R5, R0

ADC R6, R0

DEC R19

BRNE DIVTEST

CPI R19, 0x0

BREQ END

DIV7:

ST Y+, R16

ADD R7, R16

ADC R8, R0

ADC R9, R0

DEC R19

BRNE DIVTEST

CPI R19, 0x0

BREQ END

BOTH: ;for both(21), I do not store individual numbers and just add all the numbers.

ADD R10, R16

ADC R11, R0

ADC R12, R0

DEC R19

BRNE DIVTEST

CPI R19, 0x0

BREQ END

END:

JMP END

Code X, C verification

#include <stdio.h>

int main()

{

int STARTADDS[200];

int divby7[200];

int divby3[200];

int divbyboth[200];

int divbyneither[200];

int i=0, a=0, b=0, c=0, d=0;

printf("STARADDS array\n");

while(i<200)

{

STARTADDS[i]=i+26;

divby3[i] = 0;

divby7[i] = 0;

divbyboth[i] = 0;

divbyneither[i] = 0;

printf("%d ", STARTADDS[i]);

i=i+1;

}

printf("\n");

i = 0;

while(i<200)

{

if(STARTADDS[i]%3)

{

if(STARTADDS[i]%7)

{

divbyneither[i] = STARTADDS[i];

d=d+divbyneither[i];

i=i+1;

}

else

{

divby7[i]= STARTADDS[i];

b = b + divby7[i];

i=i+1;

}

}

else if(STARTADDS[i]%7)

{

divby3[i] = STARTADDS[i];

a = a + divby3[i];

i=i+1;

}

else

{

divbyboth[i] = STARTADDS[i];

c = c + divbyboth[i];

i=i+1;

}

}

i=0;

printf("divby3 array\n");

while(i<200)

{

printf("%d ", divby3[i]);

i=i+1;

}

printf("\n");

i=0;

printf("divby7 array\n");

while(i<200)

{

printf("%d ", divby7[i]);

i=i+1;

}

printf("\n");

i=0;

printf("divbyboth array\n");

while(i<200)

{

printf("%d ", divbyboth[i]);

i=i+1;

}

printf("\n");

i=0;

printf("divbyneither array\n");

while(i<200)

{

printf("%d ", divbyneither[i]);

i=i+1;

}

printf("\n");

printf("sum of divby3 is %d \n", a);

printf("sum of divby7 is %d \n", b);

printf("sum of divbyboth is %d \n", c);

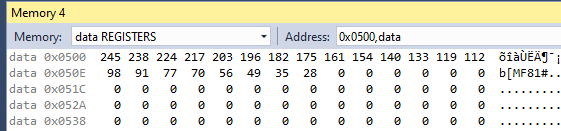
printf("sum of divbyneither is %d \n", d);

return 0;

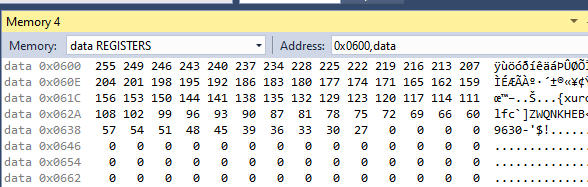
1. }**SCHEMATICS**

No schematics needed

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

****

screenshot of what’s stored in 0x0500, which represents the 7-divisibles.



screenshot of what’s stored in 0x0600, which represents the 3-divisibles.



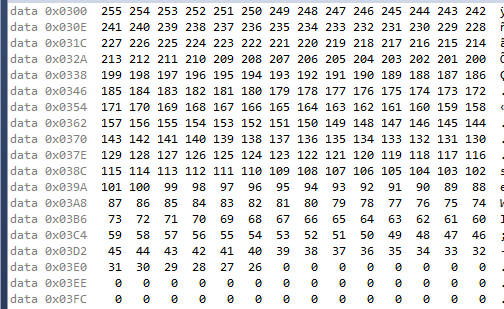
all = 32315

div3 = 9240

div7 = 3003

both = 1617

neither = 18429

****

Screenshot of what’s stored in 0x0300 after part 1.

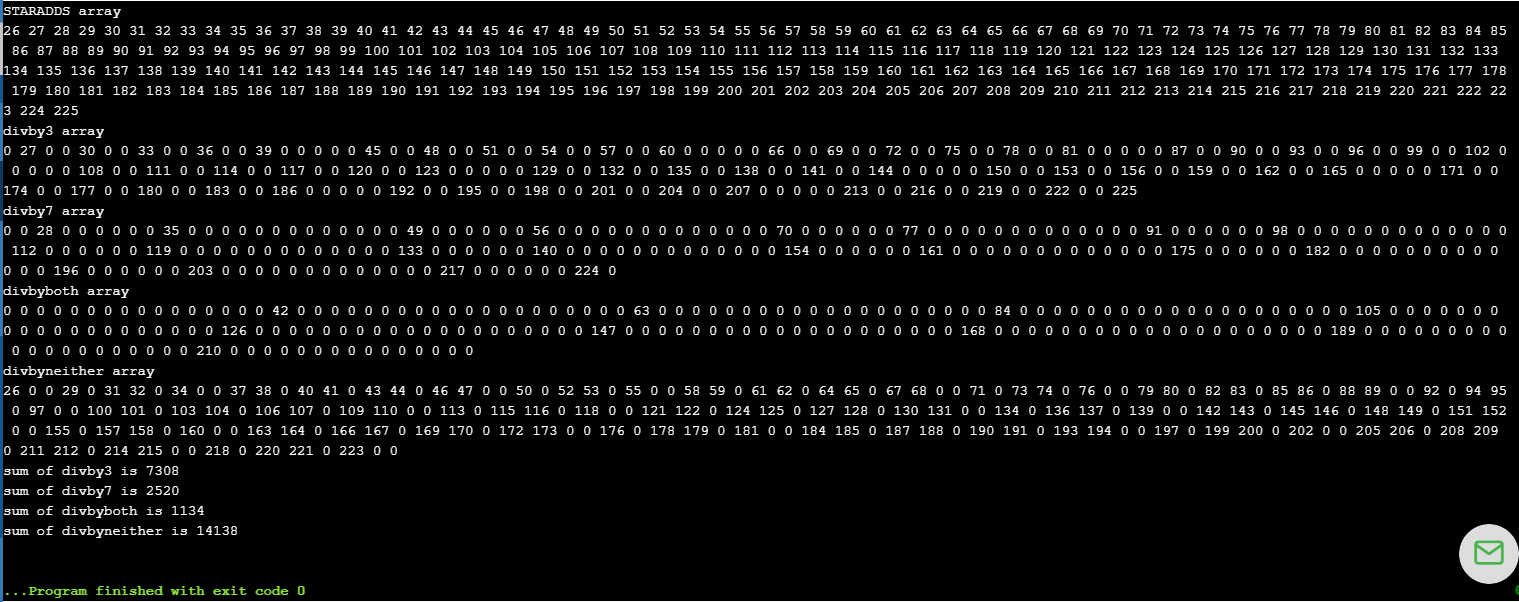


Figure 1. verification of expected result in C code.

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

No board setup required

1. **VIDEO LINKS OF EACH DEMO**

https://youtu.be/CXs654bJyO8

1. **GITHUB LINK OF THIS DA**

<https://github.com/cho-minsung/assignment1B/tree/master>

**Student Academic Misconduct Policy**

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

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

Minsung Cho