CPE301 – SPRING 2019

Design Assignment 1b

Student Name: Do Le

Student #: 2001183621

Student Email: led2@unlv.nevada.edu

Primary Github address: <https://github.com/DoVietLe/>

Directory: <https://github.com/DoVietLe/assignments>

1. **COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS**

No components were used for this assignment.

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

; No initial codes were given for this assignment.

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

.ORG 0

; Defines the starting address for the arrays.

.EQU STARTADDS = 0x300

.EQU DIV7 = 0x500

.EQU DIV3 = 0x600

.EQU DIVBOTH = 0x700

.EQU DIVNONE = 0x800

; Calculates the modulus by repeated subtraction. Overwrites @0 with the modulus.

.MACRO MOD

subtract:

CP @0, @1

BRLO done

SUB @0, @1

RJMP subtract

done:

.ENDMACRO

start:

; (PART 1). Store 200 numbers between 26 and 226 starting at 0x300.

LDI XH, HIGH(STARTADDS)

LDI XL, LOW(STARTADDS)

LDI R16, 200

LDI R17, 26

loop:

ST X+, R17

INC R17

DEC R16

BRNE loop

; (PART 2/3).

; Initializes the pointers for the five arrays.

LDI XH, HIGH(STARTADDS)

LDI XL, LOW(STARTADDS)

; Loads register to keep track of the lower byte of memory address.

LDI R25, LOW(DIV7)

LDI R24, LOW(DIV3)

LDI R23, LOW(DIVBOTH)

LDI R22, LOW(DIVNONE)

; Initializes result of sum by clearing all result registers.

CLR R0

CLR R1

CLR R2

CLR R3

CLR R4

CLR R5

CLR R6

CLR R7

CLR R8

CLR R9

CLR R10

CLR R11

CLR R12

CLR R13

; Initializes the counter for the loop.

LDI R20, 200

iter:

; Reads in the next value from the array starting at 0x300.

LD R16, X+

; Calculates the modulus by 7.

MOV R17, R16

LDI R18, 7

MOD R17, R18

; Calculates the modulus by 3.

MOV R18, R16

LDI R19, 3

MOD R18, R19

; If the number is divisible by 3 or 7, go to divisible to determine where the number is placed.

; Otherwise, store in array starting at DIVNONE and jump to the endif.

CPI R17, 0

BREQ divisible

CPI R18, 0

BREQ divisible

LDI YH, HIGH(DIVNONE)

MOV YL, R22

ST Y, R16

; Adds the value to register pairs R12:R11:R10.

ADD R10, R16

ADC R11, R0

ADC R12, R0

INC R22

RJMP endif

divisible:

; If divisible by 7, store the number, otherwise skip to elseif0.

CPI R17, 0

BRNE elseif0

LDI YH, HIGH(DIV7)

MOV YL, R25

ST Y, R16

; Adds the value to register pairs R3:R2:R1.

ADD R1, R16

ADC R2, R0

ADC R3, R0

INC R25

elseif0:

; If divisible by 3, store the number, otherwise skip to elseif1.

CPI R18, 0

BRNE elseif1

LDI YH, HIGH(DIV3)

MOV YL, R24

ST Y, R16

; Adds the value to register pairs R6:R5:R4.

ADD R4, R16

ADC R5, R0

ADC R6, R0

INC R24

elseif1:

; If divisible by both, store the number, otherwise skip to endif.

CPI R17, 0

BRNE endif

CPI R18, 0

BRNE endif

LDI YH, HIGH(DIVBOTH)

MOV YL, R23

ST Y, R16

; Adds the value to register pairs R9:R8:R7

ADD R7, R16

ADC R8, R0

ADC R9, R0

INC R23

endif:

; Decrement the loop counter, then repeat the loop.

DEC R20

BRNE iter

end:

RJMP end

1. **SCHEMATICS**

No schematics were needed for this assignment.

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

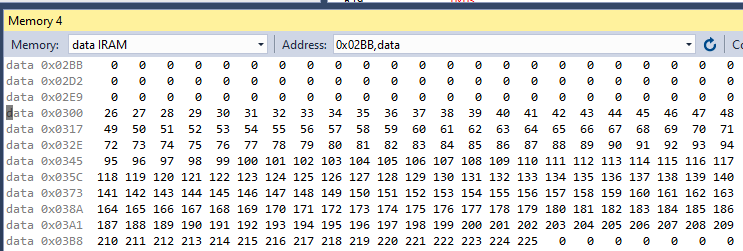
Store 200 numbers starting from the STARTADDS=0x0300 location. Populate the value of the memory location by adding high(STARTADDS) and low(STARTADDS). Use the X/Y/Z registers as pointers to fill up 200 numbers that are greater than 25 and less than 226. The numbers can be consecutive or random numbers.

Figure : Consecutive numbers stored in memory.

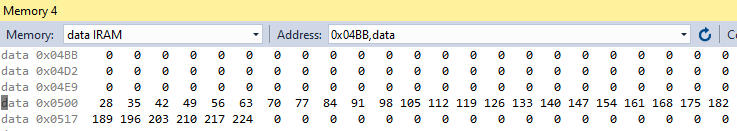
Use X/Y/Z register addressing to parse through the 200 numbers, if the number is divisible by 7 store the number starting from memory location 0x0500, if the number is divisible by 3 store the number starting from memory location 0x0600, if they are divisible by both store them at 0x0700, else store at location starting at 0x0800.

Figure : Numbers divisible by 7.

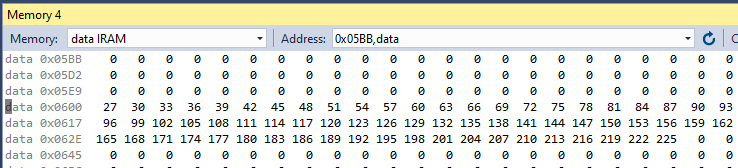


Figure : Numbers divisible by 3.

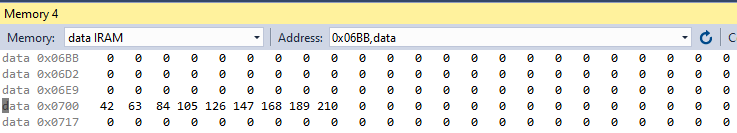


Figure : Numbers divisible by 3 and divisible by 7.

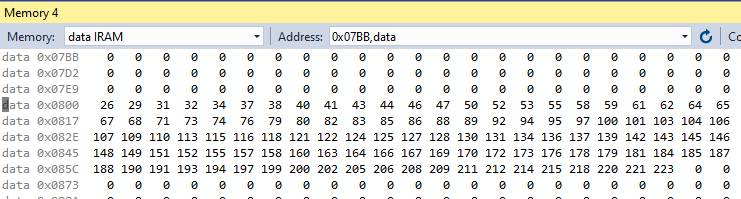


Figure : Numbers not divisible by 3 and not divisible by 7.

Use X/Y/Z register addressing to simultaneously add numbers from memory location 0x0500, 0x0600, 0x0700 and 0x0800 and store the sums at R18:R17:R16 and R21:R20:R19 respectively. Pay attention to the carry overflow.

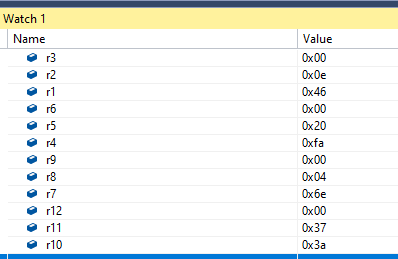


Figure : Sum of all four arrays independently. (Reads 0xE46, 0x20FA, 0x46E, 0x373A in order).

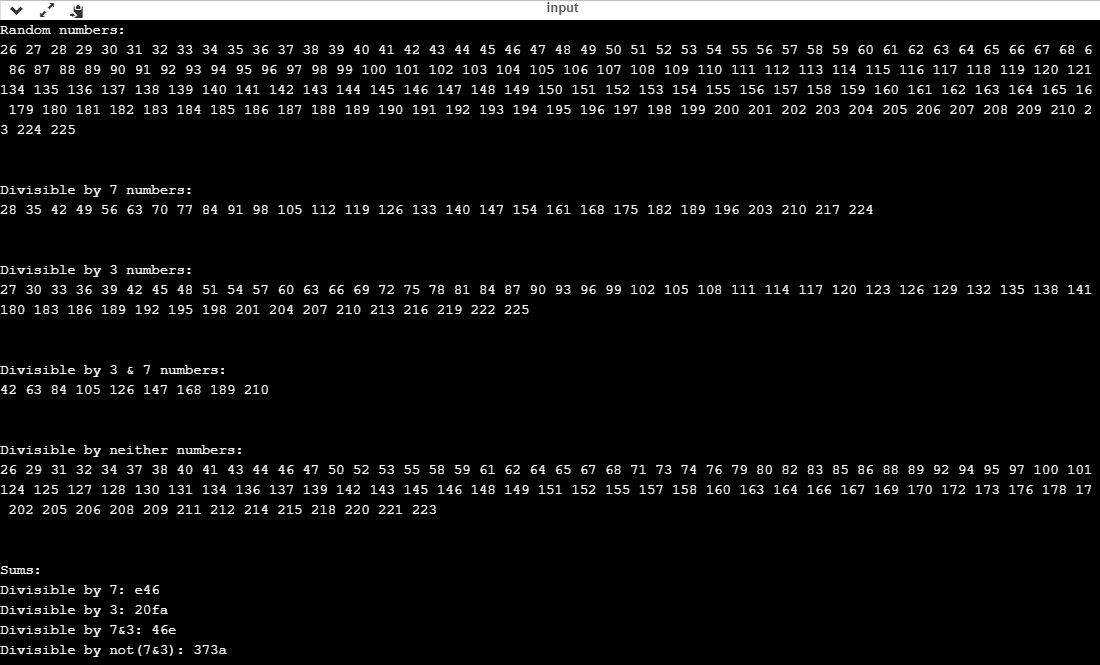
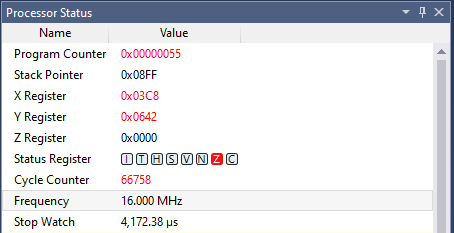
Verify your algorithm and answers using C or any high-level program.

Figure : Output of verification code (verified with C++ using https://www.onlinegdb.com/online\_c++\_compiler)

Determine the execution time @ 16MHz/#cycles of your algorithm using the simulation.

Figure : Processory status from beginning to end of program (takes 4,172.38µs).

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

The board was not used in this assignment.

1. **VIDEO LINKS OF EACH DEMO**

The board was not used in this assignment.

1. **GITHUB LINK OF THIS DA**

<https://github.com/DoVietLe/assignments/tree/master/ESD301/LAB01b>

**Student Academic Misconduct Policy**

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

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

Do Viet Le