Audie Escala

CPE301 – SPRING 2018

Design Assignment 01

**DO NOT REMOVE THIS PAGE DURING SUBMISSION:**

The student understands that all required components should be submitted in complete for grading of this assignment.

|  |  |  |  |
| --- | --- | --- | --- |
| **NO** | **SUBMISSION ITEM** | **COMPLETED (Y/N)** | **MARKS**  **(/MAX)** |
| 1 | COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS |  |  |
| 2. | INITIAL CODE OF TASK 1/A |  |  |
| 3. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 2/B |  |  |
| 3. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 3/C |  |  |
| 3. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 4/D |  |  |
| 3. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 5/E |  |  |
| 5. | SCREENSHOTS OF EACH TASK OUTPUT |  |  |
| 7. | GOOGLECODE LINK OF THE DA |  |  |
|  |  |  |  |
|  |  |  |  |

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

List of Components used

Block diagram with pins used in the Atmega328P

1. **INITIAL/DEVELOPED CODE IN ASSEMBLY OF TASK 1**

; ==============================================================================

; TASK 1

;===============================================================================

; Code Segment that stores 300 numbers fromt STARTADDS = 0x0222

LDI r16,1 ; counter starting at number 1

LDI XL, LOW($0222) ; Load the low byte of X with 0x20

LDI XH, HIGH($0222) ; Load the high byte of X with 0x02

L1:ST X+, r16 ; store number into X address (numbers 1 to 255)

inc r16 ; increment the number

cpi r16,255 ; compare if we cycled through 255 numbers

BRNE L1 ; branch to L1 if not equal

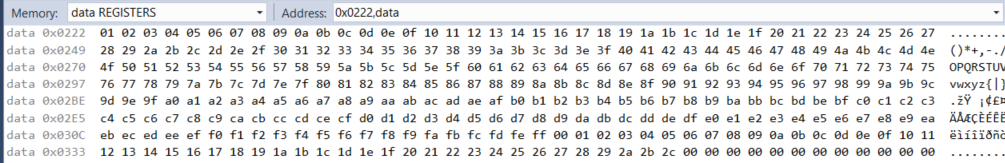
L2: ST X+, r16 ; store number into X address (numbers 256 to 300)

inc r16 ; increment the number

cpi r16, 45 ; compare if we cycled through the last 45 numbers

BRNE L2 ; branch to L2 if not equal

Task 1 Screen Shot of 0x0222



1. **DEVELOPED CODE IN ASSEMBLY OF TASK 2**

; ==============================================================================

; TASK 2

;===============================================================================

; Parse through the number is divisble by 5 store the number to location 0x0400

; else store number to location 0x0600

ldi XL, LOW($0222) ;load low byte of 0x22 to XL

ldi XH, HIGH($0222) ;load high byte of 0x02 to XH

ldi YL, LOW($0400) ;load low byte of 0x00 to YL

ldi YH, HIGH($0400) ;load high byte to 0x04 to YH

ldi ZL, LOW($0600) ;load low byte of 0x00 to ZL

ldi ZH, HIGH($0600) ;load high byte of 0x06 to ZH

ldi r16, 0

loop:

ldi r21,0

ld r20, X+ ; dividend

add r21, r20 ; store just in case else

Divide: ; loops to divide number by 5

cpi r20,0 ; compare if current number is 0

breq NotDivisible

cpi r20,1 ; comapre if current number is 1

breq NotDivisible

cpi r20,2 ; compare if current number is 2

breq NotDivisible

cpi r20,3 ; compare if current number is 3

breq NotDivisible

cpi r20,4 ; compare if current number is 4

breq NotDivisible

cpi r20,5 ; compare if current number is 5

breq Divisible

subi r20,5 ; subtract current number by 5

jmp Divide ; loop back to Divide

Divisible:

st Y+, r21 ; if divisble by 5 store into Y address

jmp done1

NotDivisible:

st Z+, r21 ; if not divisbly by 5 store into Z address

done1:

cpi r20,0 ; if number hits 256, we take note of it

breq incrementC

jmp notZero

incrementC:

inc r16 ; increment checker if number hits 256

notZero:

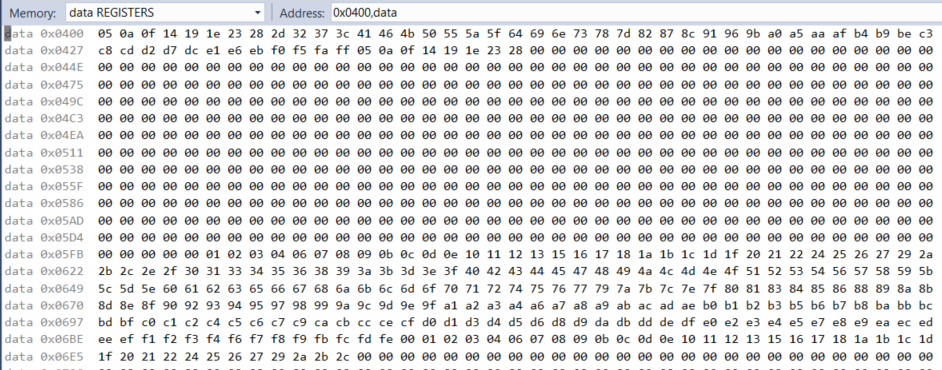
cpi r16,2 ; checking if number hits 256

breq done2

jmp loop

done2:

Task 2 Screenshot of 0x400 and 0x600



1. **DEVELOPED CODE IN ASSEMBLY OF TASK 3**

; ==============================================================================

; TASK 3

;===============================================================================

; Simultaneously add numbers from memory location Y and Z

; Store the sums R16:R17 to Y

; Store the sums R18:R19 to Z

ldi YL, LOW($0400) ;load low byte of 0x00 to YL

ldi YH, HIGH($0400) ;load high byte to 0x04 to YH

ldi ZL, LOW($0600) ;load low byte of 0x00 to ZL

ldi ZH, HIGH($0600) ;load high byte of 0x06 to ZH

ldi r21,0

ldi r16,0 ;clearing r16

ldi r17,0 ;clearing r17

ldi r18,0 ;clearing r18

ldi r19,0 ;clearing r19

ldi r23,0

Sum:

ld r20, Y+ ;load number in Y address to r20

ld r22, Z+ ;load number in Z address to r22

cpi r20, 0 ; compare if Y is 0

brne notDone

cpi r22,0 ;compare if Z is 0

brne notDone

inc r23 ; if both is 0 increment time met once

cpi r23,2 ; if both is 0 twice means we jump to Done

breq Done

notDone:

add r16, r20 ; add low bit Y

adc r17, r21 ; add high bit Y

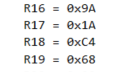
add r18, r22 ; add low bit Z

adc r19, r21 ; add high bit Z

jmp Sum ; loop back to sum again

Done: jmp Done

Task 3 Screenshot of Y and Z sum in R16:R17 and R18:R19



1. **DEVELEPED CODE IN C TASK 4**

#include <avr/io.h>

#include <stdint.h>

int main(void)

{

*uint16\_t* Ysum = 0, Zsum = 0;

*uint8\_t* array0[300]; //X

*uint8\_t* array1[300]; //Y

*uint8\_t* array2[300]; //Z

//Checking Task 1

for(*uint8\_t* i =1; i<300; i++){

array0[i] =i;

}

//Checking Task 2

for(*uint16\_t* i = 0; i < 300; i++){

if(array0[i] %5 == 0){

array1[i] = i;

}

else{

array2[i] = i;

}

}

//Checking Task 3

int array1size = sizeof(array1);

for(*uint8\_t* i = 0; i < array1size; i++){

Ysum = Ysum + array1[i];

}

int array2size = sizeof(array2);

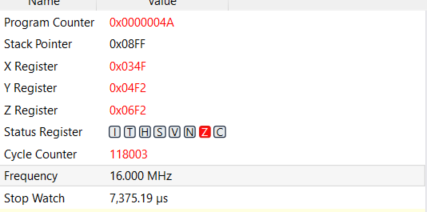
for(*uint8\_t* i = 0; i < array2size; i++){

Zsum = Zsum + array2[i];

}

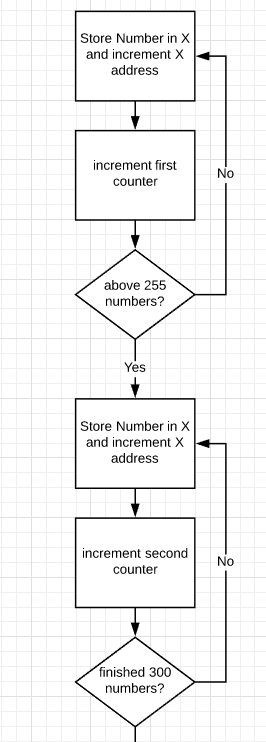
}

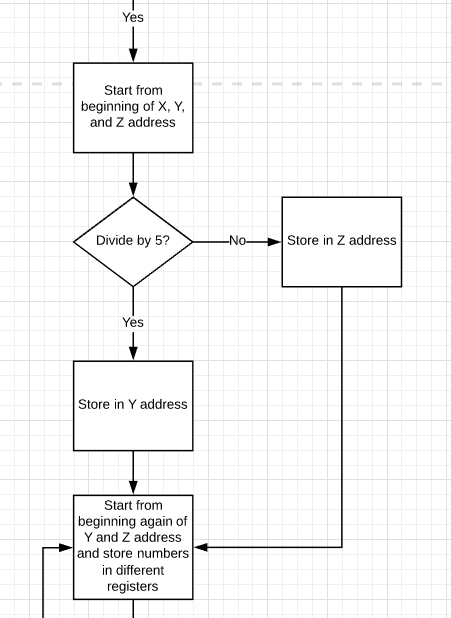
1. **TASK 5 SCREEN CAPTURE**

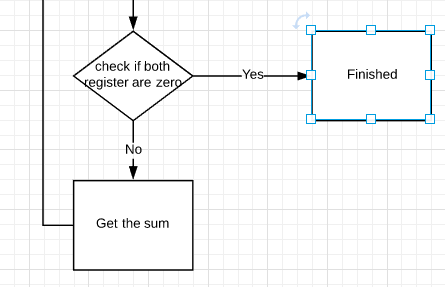


Execution time = 0x4A \* 7,375us = 545,750us => 0.54575 seconds

1. **Flowchart**







1. **GITHUB LINK OF DA1**

**Student Academic Misconduct Policy**

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

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

Audie Escala