Vincent Tuason

CPE301 – SPRING 2018

Design Assignment 1

**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 |  |  |
| 4. | SCHEMATICS |  |  |
| 5. | SCREENSHOTS OF EACH TASK OUTPUT |  |  |
| 5. | SCREENSHOT OF EACH DEMO |  |  |
| 6. | VIDEO LINKS OF EACH DEMO |  |  |
| 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 OF TASK 1**

;Code segment that puts 300 numbers memory starting from 0x0222

.org 0

.DEF counterL = R20 ;define R21:R20 as my counter holding the value 300

.DEF counterH = R21

LDI counterL, 0x2c ;300 in hex = Ox012c load lower bits to counterL

LDI counterH, 0x01 ;load higher bits to counterH

.DEF divFiveSumL = R17 ;store sum of divisible by 5 numbers in R16:R17

.DEF divFiveSumH = R16

.DEF notDivFiveSumL = R19 ;store sum of not divisible by 5 numbers in R18:R19

.DEF notDivFiveSumH = R18

LDI XL, 0x22 ;Load lower half of 0x0222 memory location to XL

LDI XH, 0x02 ;Load higher half of 0x0222 memory location to XH

loop1:

mov R23, XH ;place XH to r23

add R23, XL ;add the lower bits of the value in X to R23

ST X+, R23 ;store the number in R23 into the 0x0222 location

mov R24, R23 ;move the value in R23 to R24

subi counterL, 0x01 ;sub 1 from 300

sbci counterH, 0x00 ;sub 0 from higher half and include carry

cpi counterL, 0 ;check if the lower bits = 0

brne loop1 ;if lower bits != 0, then keep looping

cpi counterH, 0 ;if lowers bits = 0 and and higher bits = 0, then stop if not, keep looping

brne loop1 ;if counterH does not equal 0, then keep looping

1. **MODIFIED CODE OF TASK 2**

;The code below will parse through 300 numbers and check if the number is divisible by 5. If it is ;store it in 0x0400, if it is not, store it in 0x0600. The code below will be nested in one loop

LDI YL, 0x00 ;Load lower half of 0x0400 memory location to YL

LDI YH, 0x04 ;Load higher half of 0x0400 memory location to YH

LDI ZL, 0x00 ;Load lower half of 0x0600 memory location to ZL

LDI ZH, 0x06 ;Load higher half of 0x0600 memory location to ZH

mov R24, R23 ;move the value in R23 to R24

subFive:

subi R24, 0x05 ;sub 5 from R24

cpi R24, 0x05 ;if the difference is greater than 5 keep subtracting by 5

brsh subFive ;branch to subFive if R24 is same as 5 or higher than 5

cpi R24, 0x00 ;if the difference is less than 5, compare it to 0

brne notDivisibleByFive ;if the difference is not equal to 0, the number is not div. by 5

st Y+, R23

jmp nextNumber

notDivisibleByFive:

st Z+, R23

nextNumber:

subi counterL, 0x01 ;sub 1 from 300

sbci counterH, 0x00 ;sub 0 from higher half and include carry

cpi counterL, 0 ;check if the lower bits = 0

brne loop1 ;if lower bits != 0, then keep looping

cpi counterH, 0 ;if lowers bits = 0 and and higher bits = 0, then stop if not, keep looping

brne loop1 ;if counterH does not equal 0, then keep looping

1. **MODIFIED CODE OF TASK 3**

**;**The follow code will sum the values that are divisible by 5 and values not divisible by 5 right after checking if the value is divisible by 5 or not. The following code will also be nested under 1 loop

clr R24 ;clears R24 so it has the value 0

add divFiveSumL, R23 ;add R23 to lower bits of divisible by 5 sum

adc divfiveSumH, R24 ;since the value in R23 is 8 bits long, Im adding R24(has a value of 0) to the higher bits of the sum

clr R24

add notDivFiveSumL, R23 ;add R23 to lower bits of not divisible by 5 sum

adc notDivFiveSumH, R24 ;since the value in R23 is 8 bits long, Im adding R24(has a value of 0) to the higher bits of the sum

1. **Complete Code (Performs Task 1, 2 and 3)**

.org 0

.DEF counterL = R20 ;define R21:R20 as my counter holding the value 300

.DEF counterH = R21

LDI counterL, 0x2c ;300 in hex = Ox012c load lower bits to counterL

LDI counterH, 0x01 ;load higher bits to counterH

.DEF divFiveSumL = R17 ;store sum of divisible by 5 numbers in R16:R17

.DEF divFiveSumH = R16

.DEF notDivFiveSumL = R19 ;store sum of not divisible by 5 numbers in R18:R19

.DEF notDivFiveSumH = R18

LDI XL, 0x22 ;Load lower half of 0x0222 memory location to XL

LDI XH, 0x02 ;Load higher half of 0x0222 memory location to XH

LDI YL, 0x00 ;Load lower half of 0x0400 memory location to YL

LDI YH, 0x04 ;Load higher half of 0x0400 memory location to YH

LDI ZL, 0x00 ;Load lower half of 0x0600 memory location to ZL

LDI ZH, 0x06 ;Load higher half of 0x0600 memory location to ZH

loop1:

mov R23, XH ;place XH to r23

add R23, XL ;add the lower bits of the value in X to R23

ST X+, R23 ;store the number in R23 into the 0x0222 location

mov R24, R23 ;move the value in R23 to R24

subFive:

subi R24, 0x05 ;sub 5 from R24

cpi R24, 0x05 ;if the difference is greater than 5 keep subtracting by 5

brsh subFive ;branch to subFive if R24 is same as 5 or higher than 5

cpi R24, 0x00 ;if the difference is less than 5, compare it to 0

brne notDivisibleByFive ;if the difference is not equal to 0, the number is not div. by 5

st Y+, R23 ;if it did not branch store that number in 0x0400

clr R24 ;clears R24 so it has the value 0

add divFiveSumL, R23 ;add R23 to lower bits of divisible by 5 sum

adc divfiveSumH, R24 ;since the value in R23 is 8 bits long, Im adding R24(has a value of 0) to the higher bits of the sum

jmp nextNumber ;after storing the number, repeat the loop for the next number

notDivisibleByFive:

st Z+, R23 ;store number that is not div. by 5 into 0x0600

clr R24

add notDivFiveSumL, R23 ;add R23 to lower bits of not divisible by 5 sum

adc notDivFiveSumH, R24 ;since the value in R23 is 8 bits long, I am adding R24(has a value of 0) to the higher bits of the sum

nextNumber:

subi counterL, 0x01 ;sub 1 from 300

sbci counterH, 0x00 ;sub 0 from higher half and include carry

cpi counterL, 0 ;check if the lower bits = 0

brne loop1 ;if lower bits != 0, then keep looping

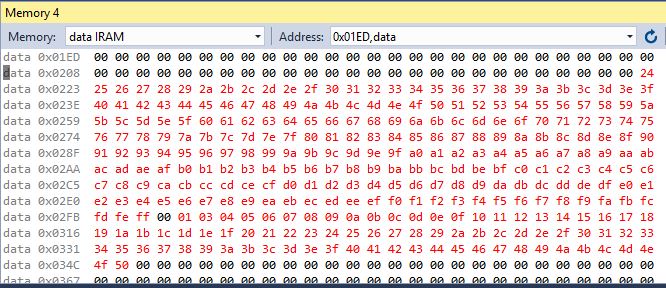
cpi counterH, 0 ;if lowers bits = 0 and and higher bits = 0, then stop if not, keep looping

brne loop1 ;if counterH does not equal 0, then keep looping

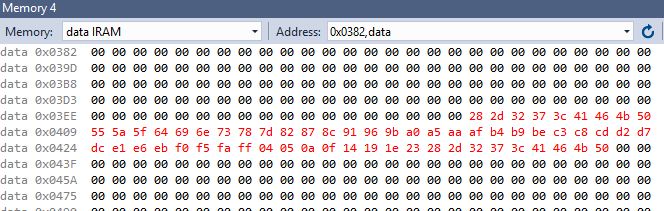
done:

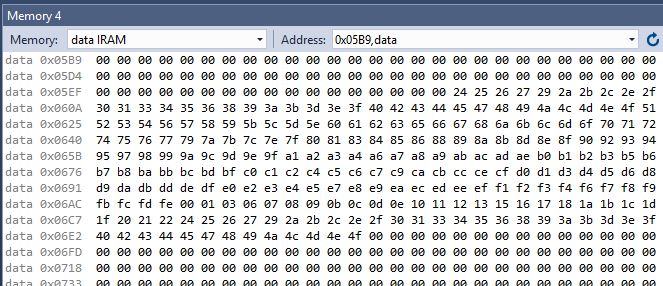
1. **SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)**

The image below is the output of task 1:

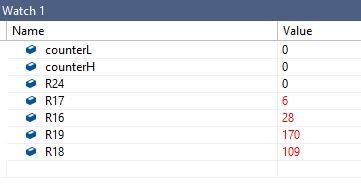


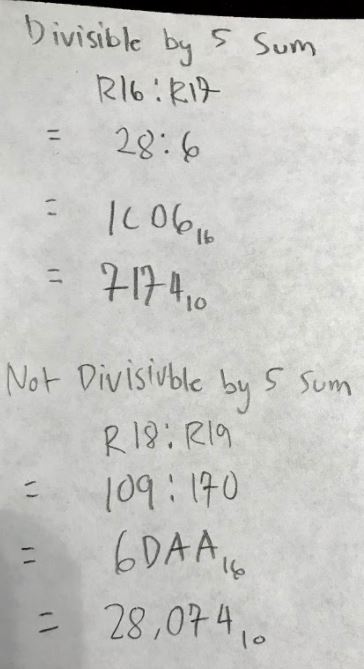
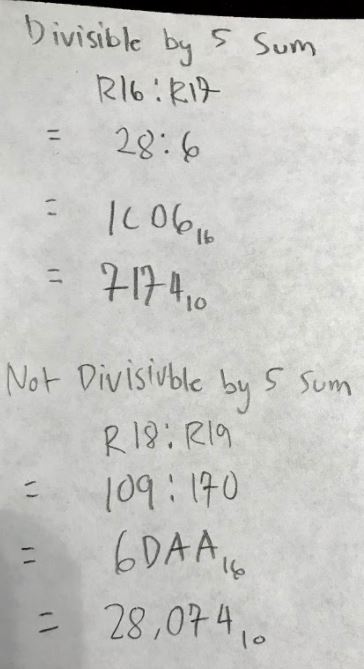
The image below is the output of task 2:





The image below is the output of task 3:



The C-Code and the image output below is for task 4:

#include <math.h>

#include<stdio.h>

#include<stdlib.h>

int main()

{

int firstNum = 36;

int X[300];

int Y[300];

int Z[300];

int j = 0; //iterator for Y-List

int k = 0; //iterator for Z-List

int ySum = 0;

int zSum = 0;

for(int i=0; i<300;i++)

{

X[i] = i + 36;

if(X[i]%5 == 0)

{

Y[j] = X[i];

ySum = ySum + Y[j];

j++;

}

else

{

Z[k] = X[i];

zSum = zSum + Z[k];

k++;

}

}

printf("%x\n",ySum);

printf("%x\n",zSum);

return 0;

}



1. **Flowchart**

Decrement counter

If(counter != 0)

Loop again

Flow chart is on the following page

Assign value 300 in R21:R20

Assign R16:R17 as divisible by 5 sum

Assign R18:R19 as not divisible by 5 sum

Make X, Y and Z into a pointer that points to the memory locations 0x0222, 0x0400, 0x0600 respectively

START

Decrement counter

If(counter != 0)

Loop again

Value < 5

Subtract 5 until value is less than 5

Value >= 5

If(value == 0)

Store into Y which points to memory 0x0400

Increment Y

Add value into divisible by 5 register in R16:R17

Check Value

If(value != 0 && value < 5)

Store into Z which points to memory 0x0600

Increment Z

Add value into not divisible by 5 register in R18:R19

Store value into X (which points to 0x0222)

Then Increment X

1. **Video link of task 1, 2 and 3**

<https://youtu.be/av6qGqU4sC4>

1. **GITHUB LINK OF THIS DA**

https://github.com/vason13/vasonGIT.git

**Student Academic Misconduct Policy**

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

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

Vincent Tuason