Your NAME

**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 |  |  |
|  |  |  |  |
|  |  |  |  |

**a. AVR assembly code that has been assembled and working**

.def countH = R25 ; upper byte of 16-bit counter

.def countL = R24 ; lower byte of 16-bit counter

.def Zcount = R23 ; counter that keeps track of non 5 entries

.def Ycount = R22 ; counter to keep track of divisible by 5 entries

.EQU STARTADDS = 0x0222

LDI r16, high(RAMEND)

out SPH, R16

LDI R16, low(RAMEND)

out SPH, R16

LDI XH, high(STARTADDS)

LDI XL, low(STARTADDS) ;load x pointer with memory location 0x0222

LDI YH, 0x04

LDI YL, 0x00 ;load y pointer with memory location 0x0400

LDI ZH, 0X06

LDI ZL, 0X00 ;load z pointer with memory location 0x0600

LDI R16, 0X00

LDI R17, 0X00

LDI R18, 0X00

LDI R19, 0X00 ;clear the registers used to store final values

LDI countL, 0

LDI countH, 0

LDI Ycount, 0

LDI Zcount, 0 ; clear all counters

; loop to load the 300 number starting at data space 0x0222. values taken by adding the first byte

; of the address with the upper byte

addloop:

MOV R20, XL ; load the lower byte of address into R20

ADD R20, XH ; add R20 + upper byte

st X+, R20 ; store the value into memory space

ADIW countH:countL, 1 ; increment 16-bit counter

cpi countL, 44 ; check if the lower byte of the counter is 0x44

brne addloop

cpi countH, 1 ; if 16-bit counter = 0x144 (300) exit the loop

brne addloop

LDI XH, high(STARTADDS)

LDI XL, low(STARTADDS) ; reset X register to address of 0x0222

; Numbers divisble by 5 are loaded into memory at address space 0x0400

; all other numbers loaded into 0x0600.

loadloop:

LD R20, X+ ; load memory data into R20

MOV R21, R2 ; copy to R21 to check if divisible by 5

; loop to subtract 5 from the data until the value is less than 5. If the remainder is zero

; the value is divisble by 5 and is to be loaded into 0x400

divide5:

CPI R21, 5

BRLO compare ; check if the value is already less than 5. if so go immediately to compare with zero

SUBI R21, 5

CPI R21, 5

BRSH divide5

compare:

CPI r21, 0

BREQ storefives ; if the value of R21 is 0 the original value is divisible by 5

ST Z+, R20 ; if not divisible by 5 load into 0x600 and increment Z

INC Zcount ; add 1 to the counter for Z address space

SBIW countH:countL, 1 ; decrease 16-bit counter

cpi countL, 0x00 ; check if lower byte of 16-bit counter is 0

BRNE loadloop

cpi countH, 0x00 ; if counter is 0 all values have been parsed

BREQ addnumbers

RJMP loadloop

storefives:

ST Y+, R20 ; value is divisible by 5. load into memory starting at 0x0400

INC Ycount ; increment counter for how many values stored at memory starting at 0x0400

SBIW countH:countL, 1 ; 16-bit counter decrement segment

cpi countL, 0x00

BRNE loadloop

cpi countH, 0x00

BRNE loadloop

;now that the values are all stored in the proper locations sum the non divisible by 5 values together

addnumbers:

LD R20, -Z ; get a value from the 0x0600 range, decrement the Z pointer

ADD R18, R20 ; add the value to R18

BRCS numcarry ; check if overflow occured

INC countL ; increment the counter

CP countL, Zcount ; if the counter = total Zcount the values in the 0x0600 range are done

BRNE addnumbers

RJMP next

numcarry:

INC R19 ; if overflow occured, increment R19

INC countL

CP countL, Zcount

BRNE addnumbers

next:

LDI countL, 0 ;reset the counter

addfives:

LD R20, -Y ;same thing as the previous loop, but for the values in the 0x0400 range

ADD R16, R20

BRCS fivecarry

INC countL

CP countL, Ycount

BRNE addfives

RJMP done

fivecarry:

INC R17

INC countL

CP countL, Ycount

BRNE addfives

; read and add the values pointed to by the Y register, decrement until

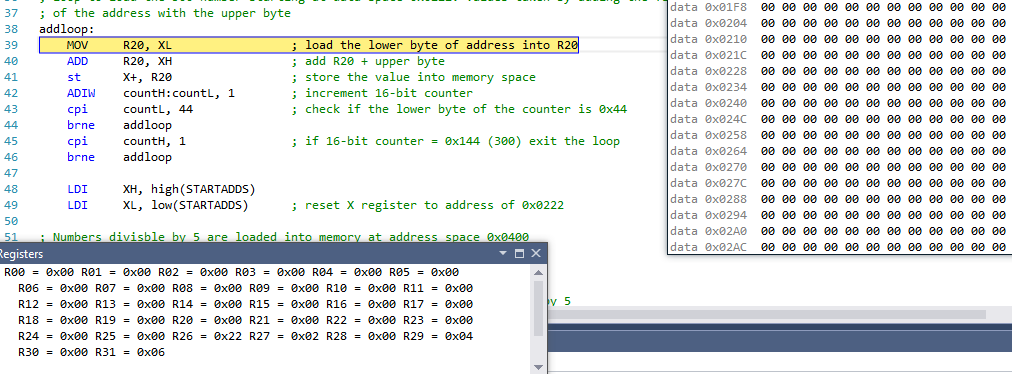
; Y points to 0x0600 to ensure the proper value is reached.

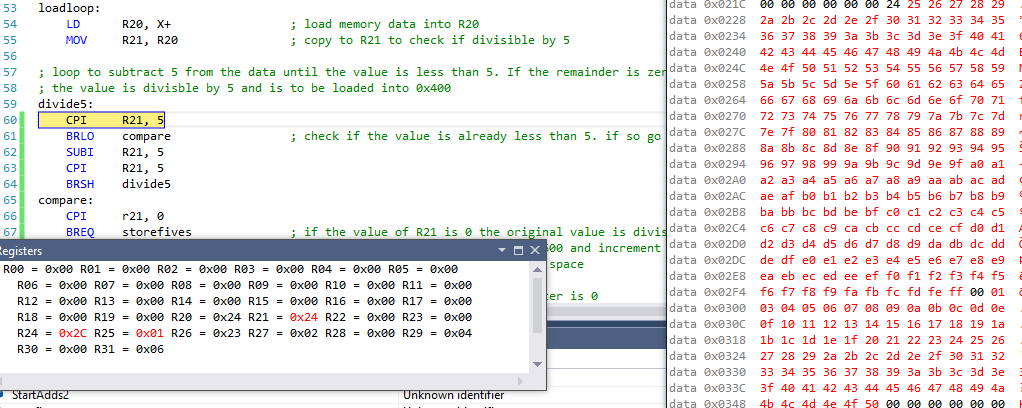
done: RJMP done ; everything is done. infinite loop

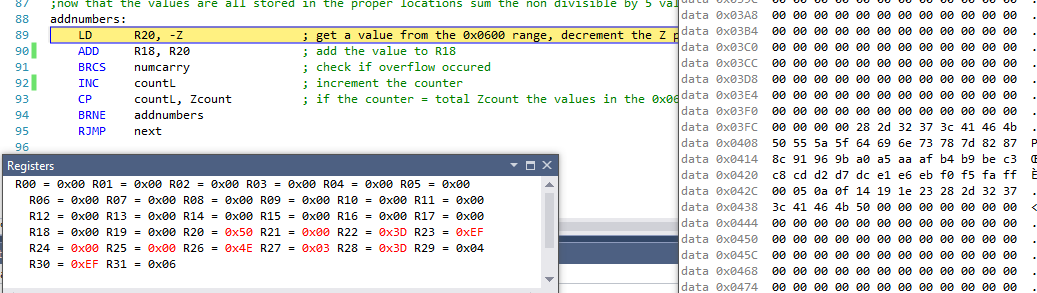
**b A word document that contains the flow chart of the assembly code along with the**

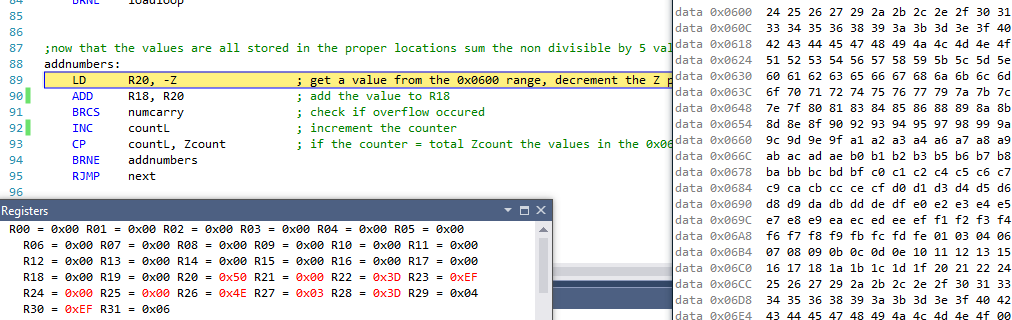
**screenshots of the AVRStudio6 during debugging at the beginning and end of Task 1, 2, 3**

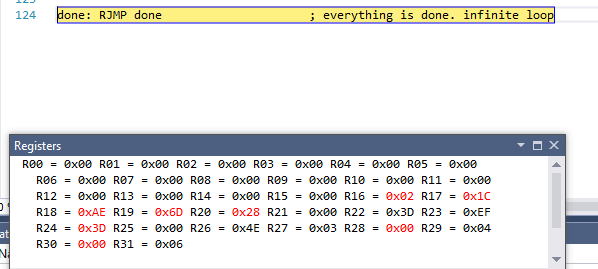
**and 4. Screen/File capture of Task 5.**











**Verify your algorithm and answers using C programming.**

#include <stdio.h>

int main(void)

{

int StartAdds1 = 34;

int StartAdds2 = 2;

int sumfive=0;

int sumOther=0;

int storeValues;

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

storeValues = StartAdds1 + StartAdds2;

if(StartAdds1 == 255){

StartAdds1 = 0;

StartAdds2 = StartAdds2 + 1;

}

else

StartAdds1 = StartAdds1 + 1;

if(storeValues == 256)

storeValues = 0;

else if (storeValues == 257)

storeValues = 1;

if(storeValues%5 == 0)

sumfive = sumfive + storeValues;

else

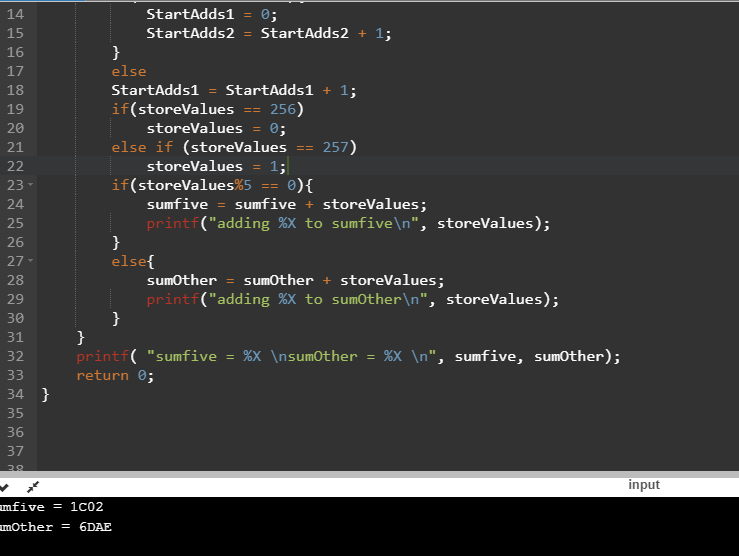
sumOther = sumOther + storeValues;

}

printf( "sumfive = %X \nsumOther = %X \n", sumfive, sumOther);

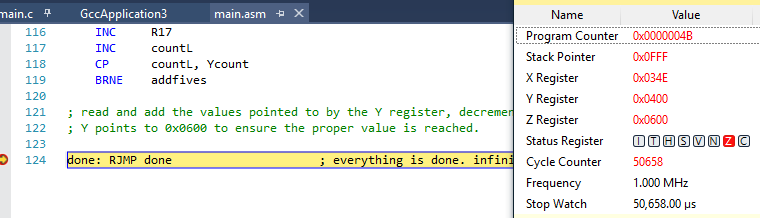
return 0;

}

****

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

using the process status information the algorithm would take 3,166ns with a 16MHz frequency.



**Student Academic Misconduct Policy**

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

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

Phillip Sortomme