Brian Lopez

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**

N/A, everything was done in the simulator in Atmel Studio

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

**Calculate and store 300 numbers starting at 0x0222:**

LDI ZL, 0x22

LDI ZH, 0x02

LDI R25, 0x02

LDI R24, 0x22

LDI R20, 0 ; counter for number of locations stored

storeLp1:

MOV R18, R25

ADD R18, R24 ; getting HIGH(STARTADD) + LOW(STARTADD)

ST Z+, R18 ; storing the results and going to next location

INC R20

CPI R20, 0 ; once count has reached its limit, end current loop

BREQ Lp1Done

ADIW R25:R24, 1 ; increment the address by 1 to get new operands to add

RJMP storeLp1

Lp1Done:

LDI R20, 0

storeLp2: ; exactly the same as first loop, done only 45 times to make

MOV R18, R25 ; up for the 300 needed.

ADD R18, R24

ST Z+, R18

INC R20

CPI R20, 46

BREQ Lp2Done

ADIW R25:R24, 1

RJMP storeLp2

Lp2Done: ; done storing values into STARTADD

1. **MODIFIED CODE OF TASK 2 AND TASK 3**

**Parse through the 300 numbers, if the number is divisible by 5, store into 0x0400 and forward, otherwise store at 0X0600. Also get the sum of each one and store to R16:R17 and R18:R19 respectively.**

Lp2Done: ; done storing values into STARTADD to STARTADD+300

LDI R20, 0 ; used for a counter

LDI R21, 0 ; will contain the current value of Z's pointer

LDI ZL, 0x22 ; parse through values that were just stored

LDI ZH, 0x02

LDI XL, 0x00 ; will store values that are divisible by 5

LDI XH, 0x04

LDI YL, 0x00 ; will store everything else

LDI YH, 0x06

CLR R16

CLR R17

CLR R18

CLR R19

GoThruMem1:

LD R21, Z+

MOV R22, R21

div1:

CPI R21, 5

BREQ isdiv1

BRLT notdiv1

SUBI R21, 5 ; will continuously subtract 5 until it either reaches 0 or

RJMP div1

isdiv1:

ST X+, R22

ADD R16, R22 ; add num divisible by 5 to r17:r16. First add number to lower

ADC R17, R0 ; half, then add the upper half with the carry bit if used

RJMP divDone1

notdiv1:

ST Y+, R22

ADD R18, R22 ; add every other number to r18:r19 word

ADC R19, R0

divDone1:

INC R20

CPI R20, 0 ; checks if r20 is 0 since that is when R20 reaches its max value and resets

BREQ mem1LpDone

RJMP GoThruMem1

mem1LpDone:

CLR R20

GoThruMem2:

LD R21, Z+

MOV R22, R21

div2:

CPI R21, 5

BREQ isdiv2

BRLT notdiv2

SUBI R21, 5 ; will continuously subtract 5 until it either reaches 0 or

RJMP div2

isdiv2:

ST X+, R22

ADD R16, R22 ; add num divisible by 5 to r17:r16. First add number to lower

ADC R17, R0 ; half, then add the upper half with the carry bit if used

RJMP divDone2

notdiv2:

ST Y+, R22

ADD R18, R22 ; add every other number to r18:r19 word

ADC R19, R0

divDone2:

INC R20

CPI R20, 46 ; checks if r20 is 46 since that will be when it has gone through

BREQ mem2LpDone ; 300 numbers

RJMP GoThruMem2

mem2LpDone:

1. **TASK 4**

**C code equivalent:**

int main(void)

{

int nums[300];

int startAddLw = 34; // Lower half of STARTADD, (0x22 = 34)

int startAddHi = 2; // upper half of STARTADD

int div5[300]; // will keep numbers that are divisible by 5

int evr[300]; // will keep every other number

int fvCnt = 0; // index count for numbers divisible by 5

int evrCnt = 0; // index count for the rest of the numbers

int fvSum = 0;

int evrSum = 0;

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

{

if ((startAddHi + startAddLw) >= 256)

{

nums[i] = (startAddHi + startAddLw) - 256; // If the sum is >255, subtract 256 since

} // asm equiv will not have any overflow

else

nums[i] = startAddHi + startAddLw;

startAddLw++;

if (startAddLw == 256)

{

startAddHi++;

startAddLw = 0;

}

}

for (int i = 0; i < 300; i++) // parse through the numbers

{

if ((nums[i] % 5) == 0) // number is divisible by 5

{

fvSum += nums[i];

div5[fvCnt] = nums[i];

fvCnt++;

}

else // everything else

{

evrSum += nums[i];

evr[evrCnt] = nums[i];

evrCnt++;

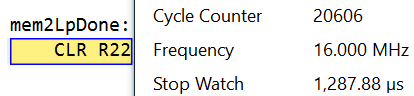
}

}

1. **TASK 5**

**Execution time @ 16MHz:**

**At the end of the simulation, there are 20606 cycles which gets a time of 1287.88 us**

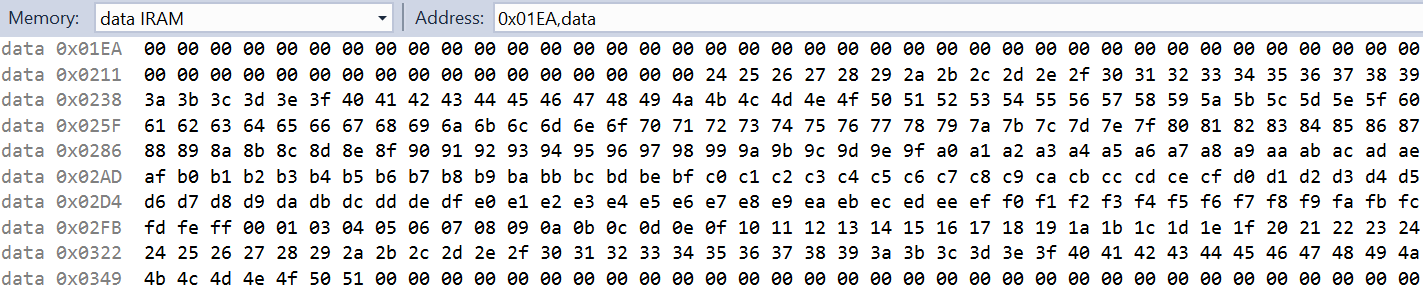
****

1. **SCHEMATICS**

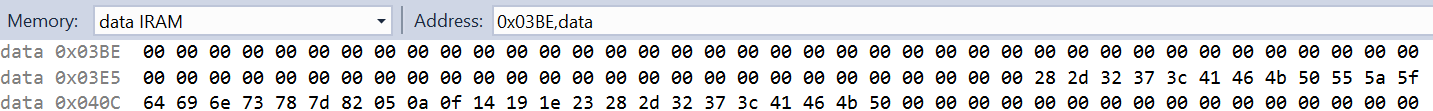
N/A

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

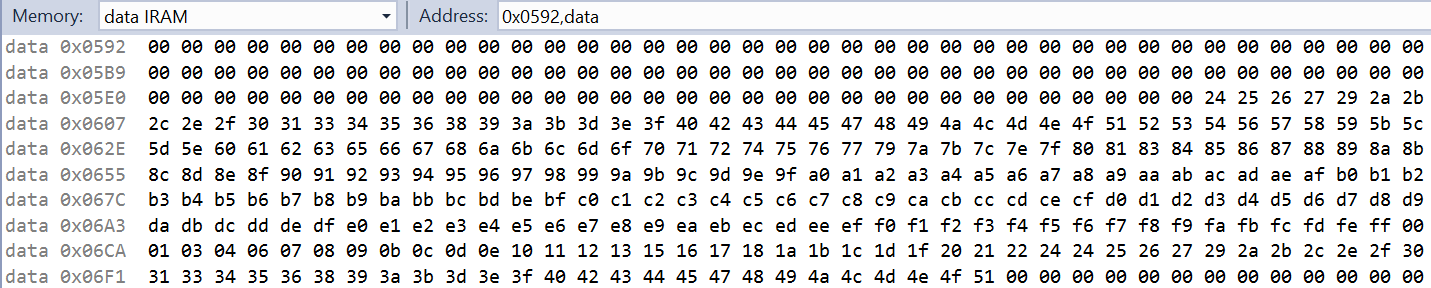
**Task1:** Stored 300 numbers starting at 0x0222 memory location:



**Task2**: At 0x0400 (numbers divisible by 5):



At 0x0600 (everything else):



**Task3**: Values of the sums of numbers divisible by 5 and the rest:

Screenshot of the final values of each register:



R17:R16 , which represents the #s divisible by 5 = 08:117 or 0875 which equals **2165**

R19:R18, the sum of everything else, = 129:176 = 81B0, which equals **33200**

**Task4**: I could not figure out how to get the values of variables in Atmel Studio for the C code, so I just ran the same code in my own C compiler to get the results:

Screen Clipping This has the R17:R16 and R19:R18 sums

They are different to what I got in my asm code, this may be because of ignoring the sum overflow, or from me not adding to the words correctly.

However, adding the contents of the memory locations individually does get me the same values that are stored into the registers that contain the sum.

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

N/A, everything done in simulator

1. **VIDEO LINKS OF EACH DEMO**

N/A, everything done in simulator

1. **GITHUB LINK OF THIS DA**

**Student Academic Misconduct Policy**

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

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

Brian Lopez