Monty Sourjah

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 | Y |  |
| 2. | INITIAL CODE OF TASK 1/A | Y |  |
| 3. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 2/B | Y |  |
| 3. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 3/C | Y |  |
| 3. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 4/D | Y |  |
| 3. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 5/E | Y |  |
| 4. | SCHEMATICS |  |  |
| 5. | SCREENSHOTS OF EACH TASK OUTPUT | Y |  |
| 5. | SCREENSHOT OF EACH DEMO | Y |  |
| 6. | VIDEO LINKS OF EACH DEMO | Y |  |
| 7. | GOOGLECODE LINK OF THE DA | Y |  |
|  |  |  |  |
|  |  |  |  |

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

List of Components used

Block diagram with pins used in the Atmega328P

START

Prepare transfer of

data from Flash to

STARTADDS

STARTADDS

RAM



Flash

memory

Copy 300 bytes from Flash

memory to STARTADDS



Initialize the

Parsing Stage

Read byte from STARTADDS

Is divisible By

5

?



DIVBY5

RAM

Store in

DIVBY5

Store in

NOTDIVBY5

YES

NO

ReadBytes=ReadBytes‐1

Have we parsed

all 300 bytes?

NO

Initializethe SUMS

stage. Set that both

types of Sums need to be

done.

Readbyte from

DIVBY5 Ram.

Sum its value to sum

of multiples of 5.

Sumsof multiples

of 5 to do?

YES

Sumsof non‐

multiples of 5 to do?

NO

Readbyte from

NOTDIVBY5 Ram.

Sum its value to sum of

non‐multiples of 5.

YES

Have we summed

all 300 bytes?

NO

NO

END

YES

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

.INCLUDE "M328pDEF.INC" ; Atmel Xplained mini 328p

.cseg ;Indicates that the next segment refers to program memory

.org 0 ;burn into ROM starting at 0

; Iteration Counter

.def ITCOUNTERH=r20

.def ITCOUNTERL=r21

.def DIVCOUNTERH = r22

.def DIVCOUNTERL = r23

.def UNDIVCOUNTERH = r24

.def UNDIVCOUNTERL = r25

.equ LASTITERATION=299 ; Max value of the decreasing counter - from 0 to 299 (300 iterations)

rjmp start

DATATABLE:

.db 0x03 ,0x12 ,0x5d ,0xd4 ,0x27 ,0xc6 ,0xe1 ,0x68 ,0x0b ,0x3a ,0x25 ,0xbc ,0xaf ,0x6e ,0x29 , \

0xd0 ,0x13 ,0x62 ,0xed ,0xa4 ,0x37 ,0x16 ,0x71 ,0x38 ,0x1b ,0x8a ,0xb5 ,0x8c ,0xbf ,0xbe , \

0xb9 ,0xa0 ,0x23 ,0xb2 ,0x7d ,0x74 ,0x47 ,0x66 ,0x01 ,0x08 ,0x2b ,0xda ,0x45 ,0x5c ,0xcf , \

0x0e ,0x49 ,0x70 ,0x33 ,0x02 ,0x0d ,0x44 ,0x57 ,0xb6 ,0x91 ,0xd8 ,0x3b ,0x2a ,0xd5 ,0x2c , \

0xdf ,0x5e ,0xd9 ,0x40 ,0x43 ,0x52 ,0x9d ,0x14 ,0x67 ,0x06 ,0x21 ,0xa8 ,0x4b ,0x7a ,0x65 , \

0xfc ,0xef ,0xae ,0x69 ,0x10 ,0x53 ,0xa2 ,0x2d ,0xe4 ,0x77 ,0x56 ,0xb1 ,0x78 ,0x5b ,0xca , \

0xf5 ,0xcc ,0xff ,0xfe ,0xf9 ,0xe0 ,0x63 ,0xf2 ,0xbd ,0xb4 ,0x87 ,0xa6 ,0x41 ,0x48 ,0x6b , \

0x1a ,0x85 ,0x9c ,0x0f ,0x4e ,0x89 ,0xb0 ,0x73 ,0x42 ,0x4d ,0x84 ,0x97 ,0xf6 ,0xd1 ,0x18 , \

0x7b ,0x6a ,0x15 ,0x6c ,0x1f ,0x9e ,0x19 ,0x80 ,0x83 ,0x92 ,0xdd ,0x54 ,0xa7 ,0x46 ,0x61 , \

0xe8 ,0x8b ,0xba ,0xa5 ,0x3c ,0x2f ,0xee ,0xa9 ,0x50 ,0x93 ,0xe2 ,0x6d ,0x24 ,0xb7 ,0x96 , \

0xf1 ,0xb8 ,0x9b ,0x0a ,0x35 ,0x0c ,0x3f ,0x3e ,0x39 ,0x20 ,0xa3 ,0x32 ,0xfd ,0xf4 ,0xc7 , \

0xe6 ,0x81 ,0x88 ,0xab ,0x5a ,0xc5 ,0xdc ,0x4f ,0x8e ,0xc9 ,0xf0 ,0xb3 ,0x82 ,0x8d ,0xc4 , \

0xd7 ,0x36 ,0x11 ,0x58 ,0xbb ,0xaa ,0x55 ,0xac ,0x5f ,0xde ,0x59 ,0xc0 ,0xc3 ,0xd2 ,0x1d , \

0x94 ,0xe7 ,0x86 ,0xa1 ,0x28 ,0xcb ,0xfa ,0xe5 ,0x7c ,0x6f ,0x2e ,0xe9 ,0x90 ,0xd3 ,0x22 , \

0xad ,0x64 ,0xf7 ,0xd6 ,0x31 ,0xf8 ,0xdb ,0x4a ,0x75 ,0x4c ,0x7f ,0x7e ,0x79 ,0x60 ,0xe3 , \

0x72 ,0x3d ,0x34 ,0x07 ,0x26 ,0xc1 ,0xc8 ,0xeb ,0x9a ,0x05 ,0x1c ,0x8f ,0xce ,0x09 ,0x30 , \

0xf3 ,0xc2 ,0xcd ,0x04 ,0x17 ,0x76 ,0x51 ,0x98 ,0xfb ,0xea ,0x95 ,0xec ,0x9f ,0x1e ,0x99 , \

0x00 ,0x3f ,0x70 ,0x33 ,0x7f ,0x60 ,0x07 ,0x76 ,0xab ,0x69 ,0xfb ,0x45 ,0xe8 ,0x1d ,0xda , \

0xae ,0xa5 ,0xf0 ,0x13 ,0x67 ,0x7e ,0xa1 ,0xf5 ,0xcc ,0x00 ,0x3f ,0x70 ,0x33 ,0x7f ,0x60 , \

0x07 ,0x76 ,0xab ,0x69 ,0xfb ,0x45 ,0xe8 ,0x1d ,0xda ,0xae ,0xa5 ,0xf0 ,0x13 ,0x67 ,0x7e

start:

; \*\*\* STORAGE STAGE \*\*\*

; Initialize Iteration Counter with 299

ldi ITCOUNTERL, (LASTITERATION & 0xFF)

ldi ITCOUNTERH, ((LASTITERATION >> 8) & 0xFF)

; X will point to the STARTADDS

ldi XH, high(STARTADDS)

ldi XL, low(STARTADDS)

; The Lookup table is stored in ROM, where bytes are stored as words.

; We need to multiply by 2

ldi ZH, high(2\*DATATABLE)

ldi ZL, low(2\*DATATABLE)

; Fill STARTADDS from our Lookup Table

FillLoop:

;Load Program memory(lpm).Read the table, then increment Z

lpm r16, Z+

;Register indirect addressing mode(Auto-increment)

;copy R16 to memory location X.Store R16 in RAM and inc X

st X+, r16

;subtract immediate

subi ITCOUNTERL, 1

; SuBtract with Carry immediate

sbci ITCOUNTERH, 0

;Branch if Carry Cleared

brcc FillLoop

Task 2.

ldi ITCOUNTERL, (LASTITERATION & 0xFF)

ldi ITCOUNTERH, ((LASTITERATION >> 8) & 0xFF)

; X will point to the start of STARTADDS

ldi XH, high(STARTADDS)

ldi XL, low(STARTADDS)

; Y will point to the start of DIVBY5

ldi YH, high(DIVBY5)

ldi YL, low(DIVBY5)

; Z will point to the start of NOTDIVBY5

ldi ZH, high(NOTDIVBY5)

ldi ZL, low(NOTDIVBY5)

; Initilize counters of multiples and not multiples of 5

clr DIVCOUNTERH

clr DIVCOUNTERL

clr UNDIVCOUNTERH

clr UNDIVCOUNTERL

clr r19

ldi r18,1

DivideLoop:

ld r16, X+

mov r17, r16 ; Save it for later

; We will check divisibility using successive subtrations of

; 5 from r16 until the carry flag is set

DivideLoopAgain:

;Subtract Immediate

subi r16, 5

brcc DivideLoopAgain ;Branch if Carry Clear

subi r16, -5 ; This the same as adding 5

cpi r16,0 ;Compare with immediate

breq RemainingIsZero ;Branch if equal

st Z+, r17 ;Register indirect addressing mode(Auto-increment)

;copy R17 to memory location Z.Store R16 in RAM and inc Z

add UNDIVCOUNTERL, r18

adc UNDIVCOUNTERH, r19

rjmp DivideCheckEnd

RemainingIsZero:

st Y+, r17 ;Register indirect addressing mode(Auto-increment)

;copy R17 to memory location Y.Store R17 in RAM and inc Y

add DIVCOUNTERL, r18

adc DIVCOUNTERH, r19

; Now, we will check if the 300 iterations limit was reached.

DivideCheckEnd:

subi ITCOUNTERL, 1 ;Subtract Immediate

sbci ITCOUNTERH, 0

;Branch if Carry Clear

brcc DivideLoop

Task3.

;Task 3

; \*\*\*\* ADDITIONS STAGE \*\*\*

; Initialize

; X will point to the start of DIVBY5

ldi XH, high(DIVBY5)

ldi XL, low(DIVBY5)

; Y will point to the start of NOTDIVBY5

ldi YH, high(NOTDIVBY5)

ldi YL, low(NOTDIVBY5)

; Zero our sum registers r16:r17 and r18\_r19

clr r16

clr r17

clr r18

clr r19

clr r0 ; This register will be used as a dummy 0 to perform addition with carry

ldi r20, 3 ; This register works as a flag to control the flow

; Our sums loop will start here

SumLoop:

;Compare r20 with 1

cpi r20, 1 ; Have with done only additions of multiples of 5 in the last iteration?

breq SumDivBy5

cpi r20, 3 ; Also all kinds of additions in the last iteration?

brne SumNotDiv ; Skip if we are done with the multiples of 5

SumDivBy5:

clc

subi DIVCOUNTERL,1 ;Subtract Immediate

sbci DIVCOUNTERH, 0 ;Subtract Immediate with Carry

;Logical AND with Immediate

andi r20, 0xfe ; Clear up our flag - assume we are not adding multiples of 5

brcs SumNotDiv ;Branch if Carry Set

;Logical OR with Immediate

ori r20,1 ; We are indeed adding multiples of 5

ld r1, X+ ; Load from memory to a temp register and increment memory pointer

add r17, r1

adc r16, r0 ;Add with Carry

SumNotDiv:

; Have with done additions of non-multiples of 5 in the last iteration but not addition of multiples of 5 in this itertation?

cpi r20, 2

breq SumNotDivBy5

; Have with done additions of non-multiples of 5 in the last iteration and addition of multiples of 5 in this itertation?

cpi r20, 3 ;Compare r20 with 3

brne SumCheckEnd ;Branch if Not Equal

SumNotDivBy5:

clc

subi UNDIVCOUNTERL,1 ;Subtract Immediate

sbci UNDIVCOUNTERH, 0 ;Subtract Immediate with Carry

;Logical AND with Immediate

andi r20, 0xfd ; Clear up our flag for no-multiples of 5.

brcs SumCheckEnd

ori r20, 2 ;Logical OR with Immediate

ld r1, Y+ ; Load from memory to a temp register and increment memory pointer

add r19, r1

adc r18, r0 ;;Add with Carry

SumCheckEnd:

;Compare r20 with 0

cpi r20, 0 ; If no operation has been done in this iteration we are done.

brne SumLoop ;Branch if Not Equal

; Expected result

; Sum multiles of 5 = 0x1ed7

; Sum not multiples of 5 = 0x7663

stop:

rjmp stop

Task 4.

start:

; \*\*\* STORAGE STAGE \*\*\*

; Initialize Iteration Counter with 299

ldi ITCOUNTERL, (LASTITERATION & 0xFF)

ldi ITCOUNTERH, ((LASTITERATION >> 8) & 0xFF)

; X will point to the STARTADDS

ldi XH, high(STARTADDS)

ldi XL, low(STARTADDS)

; The Lookup table is stored in ROM, where bytes are stored as words.

; We need to multiply by 2

ldi ZH, high(2\*DATATABLE)

ldi ZL, low(2\*DATATABLE)

; Fill STARTADDS from our Lookup Table

FillLoop:

;Load Program memory(lpm).Read the table, then increment Z

lpm r16, Z+

;Register indirect addressing mode(Auto-increment)

;copy R16 to memory location X.Store R16 in RAM and inc X

st X+, r16

;subtract immediate

subi ITCOUNTERL, 1

; SuBtract with Carry immediate

sbci ITCOUNTERH, 0

;Branch if Carry Cleared

brcc FillLoop

; \*\*\* PARSING STAGE \*\*\*

; Now, we will separate the numbers into divisible by 5 and

; not divisible by 5

; Initialize again the Iteration Counter with 299

ldi ITCOUNTERL, (LASTITERATION & 0xFF)

ldi ITCOUNTERH, ((LASTITERATION >> 8) & 0xFF)

; X will point to the start of STARTADDS

ldi XH, high(STARTADDS)

ldi XL, low(STARTADDS)

; Y will point to the start of DIVBY5

ldi YH, high(DIVBY5)

ldi YL, low(DIVBY5)

; Z will point to the start of NOTDIVBY5

ldi ZH, high(NOTDIVBY5)

ldi ZL, low(NOTDIVBY5)

; Initilize counters of multiples and not multiples of 5

clr DIVCOUNTERH

clr DIVCOUNTERL

clr UNDIVCOUNTERH

clr UNDIVCOUNTERL

clr r19

ldi r18,1

DivideLoop:

ld r16, X+

mov r17, r16 ; Save it for later

; We will check divisibility using successive subtrations of

; 5 from r16 until the carry flag is set

DivideLoopAgain:

;Subtract Immediate

subi r16, 5

brcc DivideLoopAgain ;Branch if Carry Clear

subi r16, -5 ; This the same as adding 5

cpi r16,0 ;Compare with immediate

breq RemainingIsZero ;Branch if equal

st Z+, r17 ;Register indirect addressing mode(Auto-increment)

;copy R17 to memory location Z.Store R16 in RAM and inc Z

add UNDIVCOUNTERL, r18

adc UNDIVCOUNTERH, r19

rjmp DivideCheckEnd

RemainingIsZero:

st Y+, r17 ;Register indirect addressing mode(Auto-increment)

;copy R17 to memory location Y.Store R17 in RAM and inc Y

add DIVCOUNTERL, r18

adc DIVCOUNTERH, r19

; Now, we will check if the 300 iterations limit was reached.

DivideCheckEnd:

subi ITCOUNTERL, 1 ;Subtract Immediate

sbci ITCOUNTERH, 0

;Branch if Carry Clear

brcc DivideLoop

;Task 3

; \*\*\*\* ADDITIONS STAGE \*\*\*

; Initialize

; X will point to the start of DIVBY5

ldi XH, high(DIVBY5)

ldi XL, low(DIVBY5)

; Y will point to the start of NOTDIVBY5

ldi YH, high(NOTDIVBY5)

ldi YL, low(NOTDIVBY5)

; Zero our sum registers r16:r17 and r18\_r19

clr r16

clr r17

clr r18

clr r19

clr r0 ; This register will be used as a dummy 0 to perform addition with carry

ldi r20, 3 ; This register works as a flag to control the flow

; Our sums loop will start here

SumLoop:

;Compare r20 with 1

cpi r20, 1 ; Have with done only additions of multiples of 5 in the last iteration?

breq SumDivBy5

cpi r20, 3 ; Also all kinds of additions in the last iteration?

brne SumNotDiv ; Skip if we are done with the multiples of 5

SumDivBy5:

clc

subi DIVCOUNTERL,1 ;Subtract Immediate

sbci DIVCOUNTERH, 0 ;Subtract Immediate with Carry

;Logical AND with Immediate

andi r20, 0xfe ; Clear up our flag - assume we are not adding multiples of 5

brcs SumNotDiv ;Branch if Carry Set

;Logical OR with Immediate

ori r20,1 ; We are indeed adding multiples of 5

ld r1, X+ ; Load from memory to a temp register and increment memory pointer

add r17, r1

adc r16, r0 ;Add with Carry

SumNotDiv:

; Have with done additions of non-multiples of 5 in the last iteration but not addition of multiples of 5 in this itertation?

cpi r20, 2

breq SumNotDivBy5

; Have with done additions of non-multiples of 5 in the last iteration and addition of multiples of 5 in this itertation?

cpi r20, 3 ;Compare r20 with 3

brne SumCheckEnd ;Branch if Not Equal

SumNotDivBy5:

clc

subi UNDIVCOUNTERL,1 ;Subtract Immediate

sbci UNDIVCOUNTERH, 0 ;Subtract Immediate with Carry

;Logical AND with Immediate

andi r20, 0xfd ; Clear up our flag for no-multiples of 5.

brcs SumCheckEnd

ori r20, 2 ;Logical OR with Immediate

ld r1, Y+ ; Load from memory to a temp register and increment memory pointer

add r19, r1

adc r18, r0 ;;Add with Carry

SumCheckEnd:

;Compare r20 with 0

cpi r20, 0 ; If no operation has been done in this iteration we are done.

brne SumLoop ;Branch if Not Equal

; Expected result

; Sum multiles of 5 = 0x1ed7

; Sum not multiples of 5 = 0x7663

stop:

rjmp stop

.dseg

.org 0x222

STARTADDS: .byte 300

.org 0x400

DIVBY5: .byte 300

.org 0x600

NOTDIVBY5: .byte 300

C code:

/\*Monty Sourjah Spring 2018 CPE301 Assignment01 C code\*/

#include <stdio.h>

#include <stdlib.h>

unsigned char DATATABLE[] = {

0x03 ,0x12 ,0x5d ,0xd4 ,0x27 ,0xc6 ,0xe1 ,0x68 ,0x0b ,0x3a ,0x25 ,0xbc ,0xaf ,0x6e ,0x29 ,

0xd0 ,0x13 ,0x62 ,0xed ,0xa4 ,0x37 ,0x16 ,0x71 ,0x38 ,0x1b ,0x8a ,0xb5 ,0x8c ,0xbf ,0xbe ,

0xb9 ,0xa0 ,0x23 ,0xb2 ,0x7d ,0x74 ,0x47 ,0x66 ,0x01 ,0x08 ,0x2b ,0xda ,0x45 ,0x5c ,0xcf ,

0x0e ,0x49 ,0x70 ,0x33 ,0x02 ,0x0d ,0x44 ,0x57 ,0xb6 ,0x91 ,0xd8 ,0x3b ,0x2a ,0xd5 ,0x2c ,

0xdf ,0x5e ,0xd9 ,0x40 ,0x43 ,0x52 ,0x9d ,0x14 ,0x67 ,0x06 ,0x21 ,0xa8 ,0x4b ,0x7a ,0x65 ,

0xfc ,0xef ,0xae ,0x69 ,0x10 ,0x53 ,0xa2 ,0x2d ,0xe4 ,0x77 ,0x56 ,0xb1 ,0x78 ,0x5b ,0xca ,

0xf5 ,0xcc ,0xff ,0xfe ,0xf9 ,0xe0 ,0x63 ,0xf2 ,0xbd ,0xb4 ,0x87 ,0xa6 ,0x41 ,0x48 ,0x6b ,

0x1a ,0x85 ,0x9c ,0x0f ,0x4e ,0x89 ,0xb0 ,0x73 ,0x42 ,0x4d ,0x84 ,0x97 ,0xf6 ,0xd1 ,0x18 ,

0x7b ,0x6a ,0x15 ,0x6c ,0x1f ,0x9e ,0x19 ,0x80 ,0x83 ,0x92 ,0xdd ,0x54 ,0xa7 ,0x46 ,0x61 ,

0xe8 ,0x8b ,0xba ,0xa5 ,0x3c ,0x2f ,0xee ,0xa9 ,0x50 ,0x93 ,0xe2 ,0x6d ,0x24 ,0xb7 ,0x96 ,

0xf1 ,0xb8 ,0x9b ,0x0a ,0x35 ,0x0c ,0x3f ,0x3e ,0x39 ,0x20 ,0xa3 ,0x32 ,0xfd ,0xf4 ,0xc7 ,

0xe6 ,0x81 ,0x88 ,0xab ,0x5a ,0xc5 ,0xdc ,0x4f ,0x8e ,0xc9 ,0xf0 ,0xb3 ,0x82 ,0x8d ,0xc4 ,

0xd7 ,0x36 ,0x11 ,0x58 ,0xbb ,0xaa ,0x55 ,0xac ,0x5f ,0xde ,0x59 ,0xc0 ,0xc3 ,0xd2 ,0x1d ,

0x94 ,0xe7 ,0x86 ,0xa1 ,0x28 ,0xcb ,0xfa ,0xe5 ,0x7c ,0x6f ,0x2e ,0xe9 ,0x90 ,0xd3 ,0x22 ,

0xad ,0x64 ,0xf7 ,0xd6 ,0x31 ,0xf8 ,0xdb ,0x4a ,0x75 ,0x4c ,0x7f ,0x7e ,0x79 ,0x60 ,0xe3 ,

0x72 ,0x3d ,0x34 ,0x07 ,0x26 ,0xc1 ,0xc8 ,0xeb ,0x9a ,0x05 ,0x1c ,0x8f ,0xce ,0x09 ,0x30 ,

0xf3 ,0xc2 ,0xcd ,0x04 ,0x17 ,0x76 ,0x51 ,0x98 ,0xfb ,0xea ,0x95 ,0xec ,0x9f ,0x1e ,0x99 ,

0x00 ,0x3f ,0x70 ,0x33 ,0x7f ,0x60 ,0x07 ,0x76 ,0xab ,0x69 ,0xfb ,0x45 ,0xe8 ,0x1d ,0xda ,

0xae ,0xa5 ,0xf0 ,0x13 ,0x67 ,0x7e ,0xa1 ,0xf5 ,0xcc ,0x00 ,0x3f ,0x70 ,0x33 ,0x7f ,0x60 ,

0x07 ,0x76 ,0xab ,0x69 ,0xfb ,0x45 ,0xe8 ,0x1d ,0xda ,0xae ,0xa5 ,0xf0 ,0x13 ,0x67 ,0x7e };

unsigned char STARTADDS[300];

unsigned char DIVBY5[300];

unsigned char NOTDIVBY5[300];

int main (void)

{

/\* STORAGE STAGE \*/

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

STARTADDS[i]=DATATABLE[i];

/\* PARSING STAGE \*/

unsigned short div5index=0;

unsigned short notdiv5index=0;

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

{

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

DIVBY5[div5index++]=STARTADDS[i];

else

NOTDIVBY5[notdiv5index++]=STARTADDS[i];

}

/\* ADDITIONS STAGE \*/

unsigned short sumMultiple5=0;

unsigned short sumNonMultiple5=0;

while (div5index || notdiv5index)

{

if (div5index)

sumMultiple5+=DIVBY5[--div5index]; /\* pre-subtract the index \*/

if (notdiv5index)

sumNonMultiple5+=NOTDIVBY5[--notdiv5index]; /\* pre-subtract the index \*/

}

*printf*("sumMultiple5=0x%x Sum of non-Multiples=0x%x", sumMultiple5,sumNonMultiple5 );

return 0;

}

Task5.

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

Insert only the modified sections here. Use more sections if needed

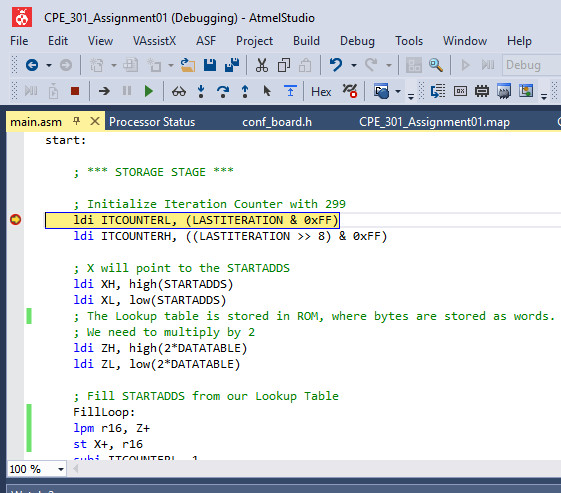
1. **SCHEMATICS**

Use fritzing.org

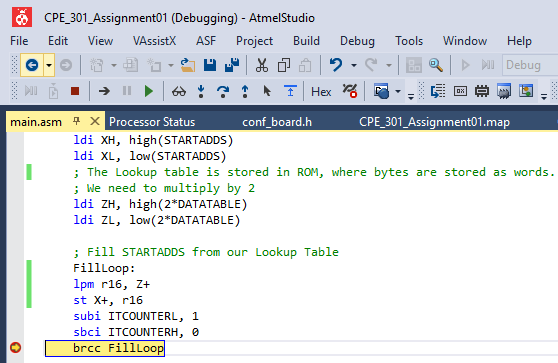
1. **SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)**

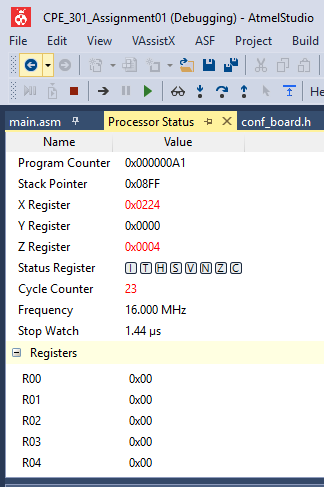
Task 1

Start



Stop





prog 0x0000 96 c0 03 12 5d d4 27 c6 e1 68 0b 3a 25 bc af 6e 29 –À..]Ô'Æáh.:%.¯n)

prog 0x0011 d0 13 62 ed a4 37 16 71 38 1b 8a b5 8c bf be b9 a0 Ð.bí¤7.q8.ŠµŒ¿..

prog 0x0022 23 b2 7d 74 47 66 01 08 2b da 45 5c cf 0e 49 70 33 #.}tGf..+ÚE\Ï.Ip3

prog 0x0033 02 0d 44 57 b6 91 d8 3b 2a d5 2c df 5e d9 40 43 52 ..DW¶‘Ø;\*Õ,ß^Ù@CR

prog 0x0044 9d 14 67 06 21 a8 4b 7a 65 fc ef ae 69 10 53 a2 2d ..g.!¨Kzeüï®i.S¢-

prog 0x0055 e4 77 56 b1 78 5b ca f5 cc ff fe f9 e0 63 f2 bd b4 äwV±x[ÊõÌÿþùàcò.´

prog 0x0066 87 a6 41 48 6b 1a 85 9c 0f 4e 89 b0 73 42 4d 84 97 .¦AHk..œ.N.°sBM.—

prog 0x0077 f6 d1 18 7b 6a 15 6c 1f 9e 19 80 83 92 dd 54 a7 46 öÑ.{j.l.ž.€ƒ’ÝT§F

prog 0x0088 61 e8 8b ba a5 3c 2f ee a9 50 93 e2 6d 24 b7 96 f1 aè.º¥</î©P“âm$·–ñ

prog 0x0099 b8 9b 0a 35 0c 3f 3e 39 20 a3 32 fd f4 c7 e6 81 88 ¸..5.?>9 £2ýôÇæ.ˆ

prog 0x00AA ab 5a c5 dc 4f 8e c9 f0 b3 82 8d c4 d7 36 11 58 bb «ZÅÜOŽÉð...Ä×6.X»

prog 0x00BB aa 55 ac 5f de 59 c0 c3 d2 1d 94 e7 86 a1 28 cb fa ªU¬\_ÞYÀÃÒ.”ç.¡(Ëú

prog 0x00CC e5 7c 6f 2e e9 90 d3 22 ad 64 f7 d6 31 f8 db 4a 75 å|o.é.Ó".d÷Ö1øÛJu

prog 0x00DD 4c 7f 7e 79 60 e3 72 3d 34 07 26 c1 c8 eb 9a 05 1c L.~y`ãr=4.&ÁÈëš..

prog 0x00EE 8f ce 09 30 f3 c2 cd 04 17 76 51 98 fb ea 95 ec 9f .Î.0óÂÍ..vQ˜ûê.ìŸ

prog 0x00FF 1e 99 00 3f 70 33 7f 60 07 76 ab 69 fb 45 e8 1d da .™.?p3.`.v«iûEè.Ú

prog 0x0110 ae a5 f0 13 67 7e a1 f5 cc 00 3f 70 33 7f 60 07 76 ®¥ð.g~¡õÌ.?p3.`.v

prog 0x0121 ab 69 fb 45 e8 1d da ae a5 f0 13 67 7e 5b e2 41 e0 «iûEè.Ú®¥ð.g~[âAà

prog 0x0132 b2 e0 a2 e2 f0 e0 e2 e0 05 91 0d 93 51 50 40 40 d8 .à¢âðàâà.‘.“QP@@Ø

prog 0x0143 f7 5b e2 41 e0 b2 e0 a2 e2 d4 e0 c0 e0 f6 e0 e0 e0 ÷[âAà.à¢âÔàÀàöààà

prog 0x0154 66 27 77 27 88 27 99 27 33 27 21 e0 0d 91 10 2f 05 f'w'ˆ'™'3'!à.‘./.

prog 0x0165 50 f0 f7 0b 5f 00 30 21 f0 11 93 92 0f 83 1f 03 c0 Pð÷.\_.0!ð.“’.ƒ..À

prog 0x0176 19 93 72 0f 63 1f 51 50 40 40 78 f7 b4 e0 a0 e0 d6 .“r.c.QP@@x÷´à àÖ

prog 0x0187 e0 c0 e0 00 27 11 27 22 27 33 27 00 24 43 e0 41 30 àÀà.'.'"'3'.$CàA0

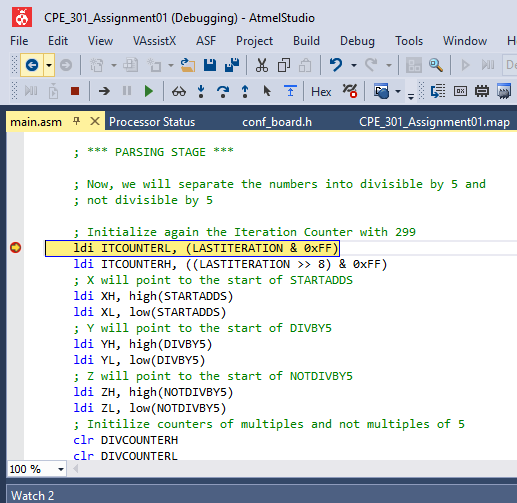
prog 0x0198 11 f0 43 30 49 f4 88 94 71 50 60 40 4e 7f 20 f0 41 .ðC0Iôˆ”qP`@N. ðA

prog 0x01A9 60 1d 90 11 0d 00 1d 42 30 11 f0 43 30 49 f4 88 94 `......B0.ðC0Iôˆ”

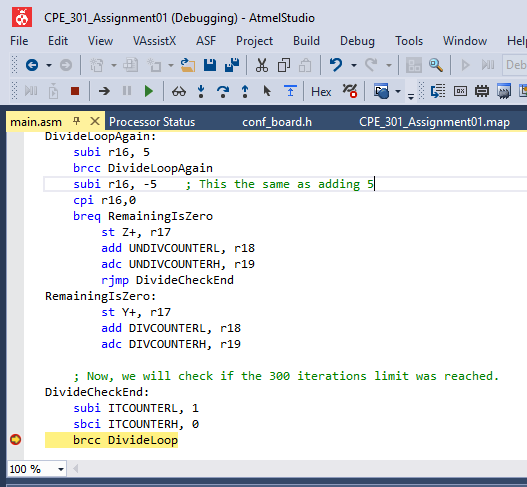
prog 0x01BA 91 50 80 40 4d 7f 20 f0 42 60 19 90 31 0d 20 1d 40 ‘P€@M. ðB`..1. .@

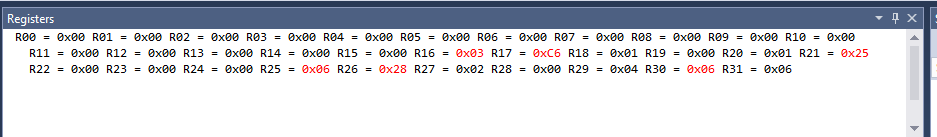
Task 2.

Start:



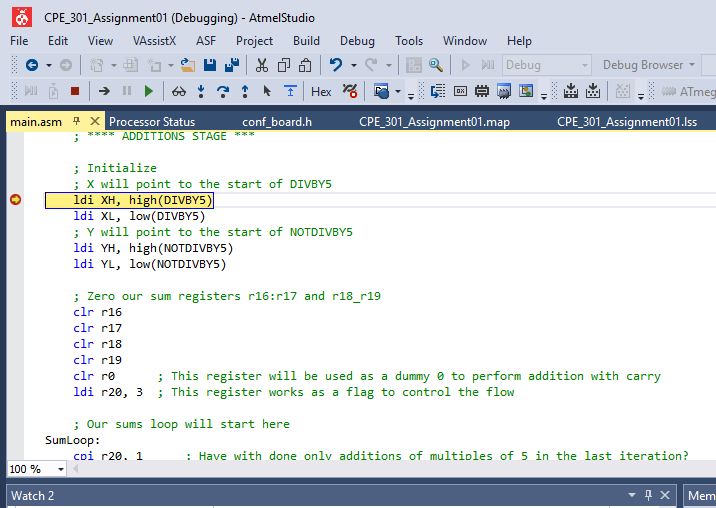
End:



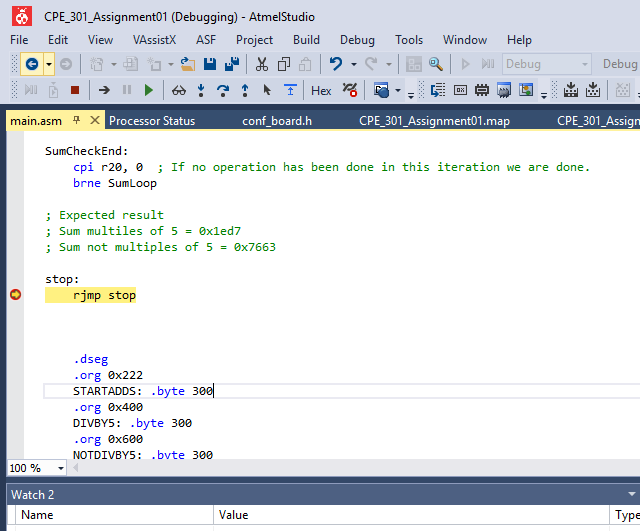


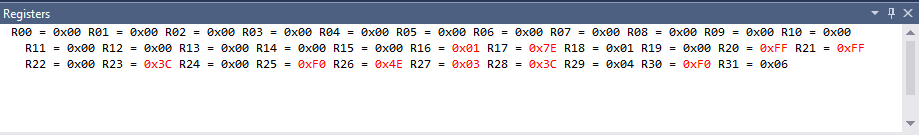
Task 3.

Start:



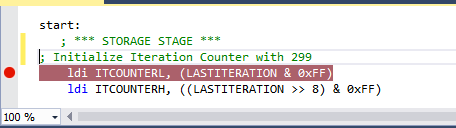
End:



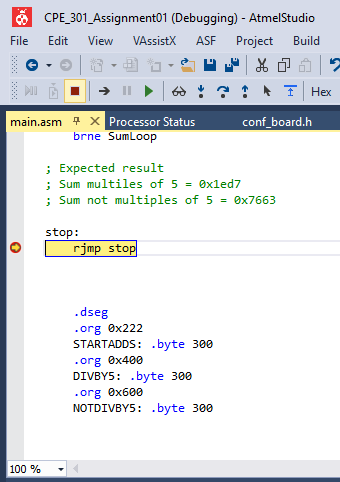


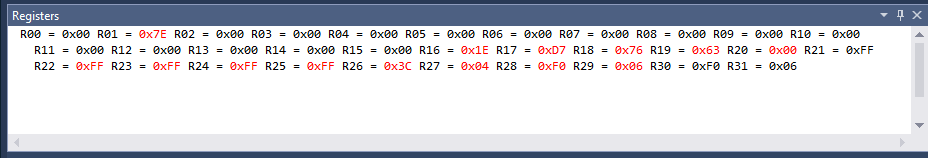
Task 4.

Start:

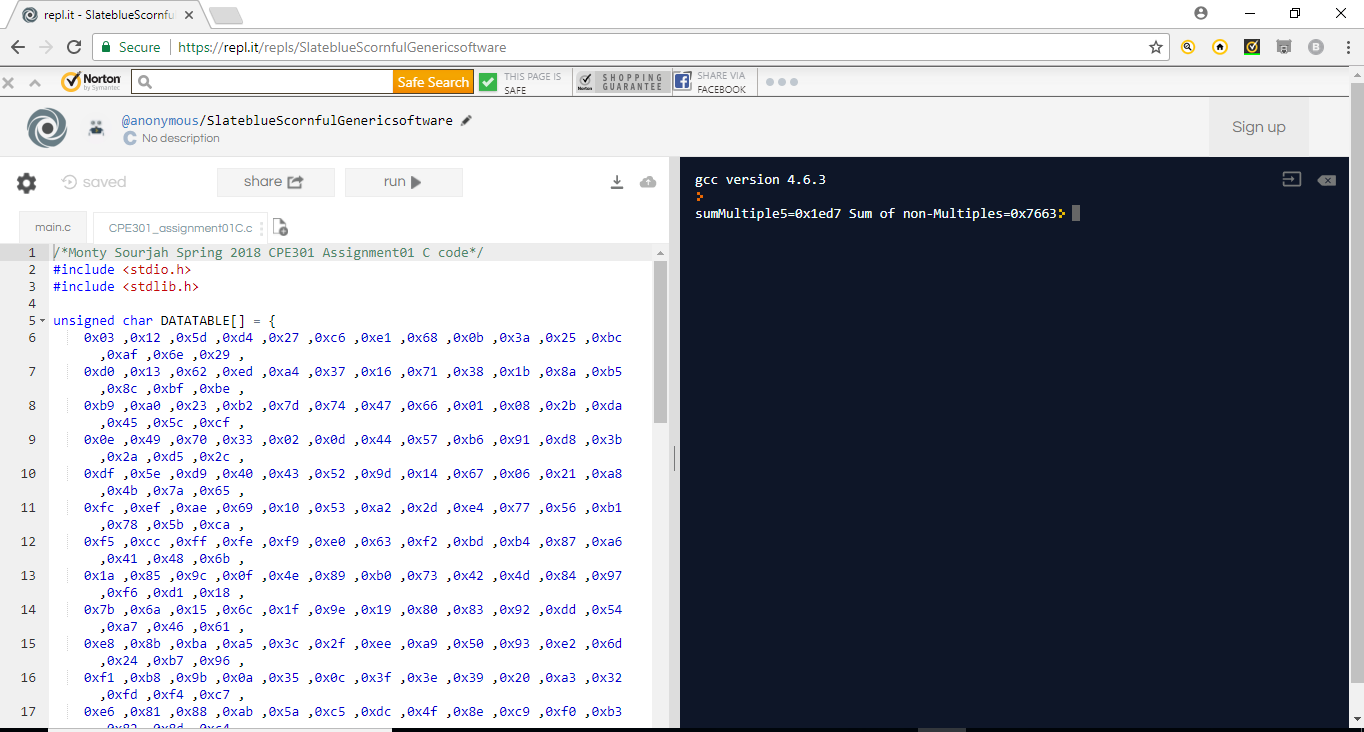


End:

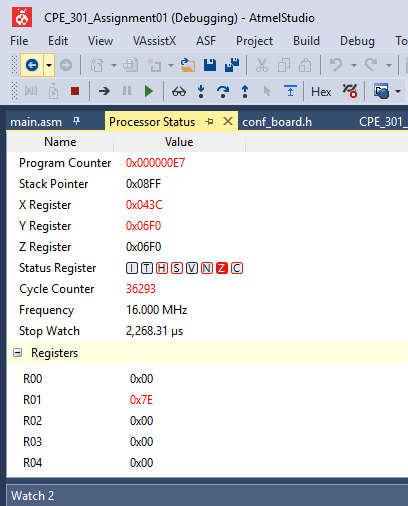




C code :(On line compiler)



Task 5:



Execution time=2268.31 micro seconds.

1. **SCREENSHOT OF EACH DEMO (BOARD SETUP)**
2. **VIDEO LINKS OF EACH DEMO**

Task1:

<http://www.youtube.com/watch?v=Ct_UrOCpdOA>

Task2:

<http://www.youtube.com/watch?v=h0SaDDK3-ZE>

Task3:

<http://www.youtube.com/watch?v=6zldWbikCCs>

Task4:

<http://www.youtube.com/watch?v=bOPfRgrXpfA>

Task5:

<http://www.youtube.com/watch?v=lFr_WaelxB8>

1. **GITHUB LINK OF THIS DA**

<https://github.com/ballasl/CPE_301_Assignment01>

**Student Academic Misconduct Policy**

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

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

NAME OF THE STUDENT