RALPH MAGO

CPE301 – SPRING 2016

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)** |
| 0. | FLOWCHART OF THE CODE | Y |  |
| 1. | INITIAL CODE OF TASK 1/A | Y |  |
| 2. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 2/B | Y |  |
| 3. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 3/C | Y |  |
| 4. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 4/D | Y |  |
| 5. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 5/E | Y |  |
| 6. | SCREENSHOTS OF EACH TASK OUTPUT | Y |  |
| 7. | GITHUB LINK OF THE DA | Y |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 0. | INITIAL CODE OF TASK 1/A |  |  |

C:\Users\r\AppData\Local\Temp\Untitled Diagram-1.png

|  |  |  |  |
| --- | --- | --- | --- |
| 1. | INITIAL CODE OF TASK 1/A |  |  |

/\*

\* AssemblerApplication1.asm

\*

\* Created: 2/24/2016 4:19:47 PM

\* Author: magor

\*/

/\*

AVR Assembly - DA1

Program tasks:

a) Store 25 numbers starting from the RAMEND/2 location. [RAMEND/2 is RAM\_MIDDLE]

- Capture the lower 8-bits (lower byte) of the variable/memory location RAM\_MIDDLE address and

use them as your values.

- RAM\_MIDDLE is incremented or decremented to get the subsequent 24 numbers.

- X/Y/Z registers are used as pointers to fill up 25 numbers from RAM\_MIDDLE.

b) Use X/Y/Z register to parse through the 25 numbers and add all numbers divisible by

7 and place the result in R20:21.

c) Same, only add all numbers divisible by 3 and place result in R23:24.

d) Check and set register R07.3 if the sum is greater than 8-bits.

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

- The execution time is Stop Watch divided by Cycle Counter.

\*/

LDI R24, 0 ;zero register

LDI R25, 25 ;load 25 as counter, 25 numbers

LDI ZH, 0x00 ; high byte of RAMBEGIN

LDI ZL, 0x18 ; low byte of RAMBEGIN (arbitrary value)

LDI YL, LOW(RAMEND) ; low byte of RAMEND

LDI YH, HIGH(RAMEND) ; high byte of RAMEND

ADC YL, ZL ; add low byte to set RAM\_MIDDLE

ADC YH, ZH ; add with carry high byte

LSR YH ; logical shift right, divides unsigned value by 2 (RAMEND/2)

ROR YL ; rotate right with carry (right Y = RAM\_MIDDLE = RAMEND/2)

; Sum of #s divisible by 7 are in X, #s divisible by 3 are in Z

LDI XH, 0 ; high byte clear

LDI XL, 0 ; low byte clear

LDI ZH, 0 ; high byte clear

LDI ZL, 0 ; low byte clear

/\*

------------------------------------------------------------------------

START OF PROCEDURE

------------------------------------------------------------------------

\*/

sumLoop:

MOV R16, YL ; copy YL into R16

ST Y, R16 ; store contents of R16 the data space of Y

MOV R17, R16 ; copy R16 to check if div by 7

/\*

------------------------------------------------------------------------

DIVISIBLE BY 7

------------------------------------------------------------------------

\*/

div7\_Check:

CPI R17, 0 ; compare immediate, # div by 7 is remainder = 0

BREQ div7\_Done ; skip next subtraction if remainder is 0

SUBI R17, 7 ; subtract 7 to simulate division

BRCC div7\_Check ;branch if carry clear, branch if > 0

div7\_Done:

CPI R17, 0 ; check if remainder is 0 (div by 7)

BRNE div7\_False ; skip addition if number not divisible by 7

ADC XL, R16 ; add number to sum of numbers divisible by 7

ADC XH, R24 ; add carry to R24

div7\_False:

MOV R17, R16 ;copy R16 to check if div by 3

/\*

------------------------------------------------------------------------

DIVISIBLE BY 3

------------------------------------------------------------------------

\*/

div3\_Check:

CPI R17, 0 ;check if remainder is 0 (means number is divisible by 3)

BREQ div3\_Done ;skip next subtraction if remainder is 0

SUBI R17, 3 ;subtract 3 to simulate division

BRCC div3\_Check ;branch if > 0

div3\_Done:

CPI R17, 0 ;check if remainder is 0 (div by 3)

BRNE div3\_False ;skip addition if number not divisible by 3

ADC ZL, R16 ;add number to sum of numbers divisible by 3

ADC ZH, R24 ;add carry if any

div3\_False:

ADIW YH:YL, 1 ;add immediate to word, point to next location in ram

DEC R25 ;decrement counter

CPI R25, 0 ;stops looping after storing 25 numbers

BRNE sumLoop ;loop back for next number

/\*

------------------------------------------------------------------------

OVERFLOW CHECK

------------------------------------------------------------------------

\*/

ADIW XH:XL, 0 ; add immediate 0 to word, use X register to check for overflow

BRCS sumOverflow\_Check ; branch if carry set, check carry for overflow

MOV R20, XH ; if no overflow, save sum of numbers div by 7 into R20:R21

MOV R21, XL

sumOverflow\_Check:

ADIW ZH:ZL, 0 ;pause here to check for overflow

BRCS done ;check carry for overflow on numbers div by 3

MOV R23, ZH ;if no overflow, save sum of numbers div by 3 into R23:R24

MOV R24, ZL ;

MOV R18, R23 ; sum registers R23:24

ADC R18, R24

ADC R18, R20 ; sum registers R20:21 and store in R18

ADC R18, R21

LDI R19, 8 ; set bit 3 (0001000)

CPI R18, 0 ; compare if R18 has carry

BRCS setReg7bit3 ; branch with carry set if there is carry

setReg7bit3:

MOV R7, R19 ; set bit 3 to R7 if the sum of R23:R24 and R20:R21 is over 8 bits

done:

|  |  |  |  |
| --- | --- | --- | --- |
| 2. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 2/B |  |  |

\*/

b) Use X/Y/Z register to parse through the 25 numbers and add all numbers divisible by

7 and place the result in R20:21.

/\*

/\*

------------------------------------------------------------------------

DIVISIBLE BY 7

------------------------------------------------------------------------

\*/

div7\_Check:

CPI R17, 0 ; compare immediate, # div by 7 is remainder = 0

BREQ div7\_Done ; skip next subtraction if remainder is 0

SUBI R17, 7 ; subtract 7 to simulate division

BRCC div7\_Check ;branch if carry clear, branch if > 0

div7\_Done:

CPI R17, 0 ; check if remainder is 0 (div by 7)

BRNE div7\_False ; skip addition if number not divisible by 7

ADC XL, R16 ; add number to sum of numbers divisible by 7

ADC XH, R24 ; add carry to R24

div7\_False:

MOV R17, R16 ;copy R16 to check if div by 3

/\*

------------------------------------------------------------------------

OVERFLOW CHECK

------------------------------------------------------------------------

\*/

ADIW XH:XL, 0 ; add immediate 0 to word, use X register to check for overflow

BRCS sumOverflow\_Check ; branch if carry set, check carry for overflow

MOV R20, XH ; if no overflow, save sum of numbers div by 7 into R20:R21

MOV R21, XL

|  |  |  |  |
| --- | --- | --- | --- |
| 3. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 3/C |  |  |

/\*

c) Same, only add all numbers divisible by 3 and place result in R23:24.

\*/

/\*

------------------------------------------------------------------------

DIVISIBLE BY 3

------------------------------------------------------------------------

\*/

div3\_Check:

CPI R17, 0 ;check if remainder is 0 (means number is divisible by 3)

BREQ div3\_Done ;skip next subtraction if remainder is 0

SUBI R17, 3 ;subtract 3 to simulate division

BRCC div3\_Check ;branch if > 0

div3\_Done:

CPI R17, 0 ;check if remainder is 0 (div by 3)

BRNE div3\_False ;skip addition if number not divisible by 3

ADC ZL, R16 ;add number to sum of numbers divisible by 3

ADC ZH, R24 ;add carry if any

div3\_False:

ADIW YH:YL, 1 ;add immediate to word, point to next location in ram

DEC R25 ;decrement counter

CPI R25, 0 ;stops looping after storing 25 numbers

BRNE sumLoop ;loop back for next number

/\*

------------------------------------------------------------------------

OVERFLOW CHECK

------------------------------------------------------------------------

\*/

ADIW XH:XL, 0 ; add immediate 0 to word, use X register to check for overflow

BRCS sumOverflow\_Check ; branch if carry set, check carry for overflow

MOV R20, XH ; if no overflow, save sum of numbers div by 7 into R20:R21

MOV R21, XL

sumOverflow\_Check:

ADIW ZH:ZL, 0 ;pause here to check for overflow

BRCS done ;check carry for overflow on numbers div by 3

MOV R23, ZH ;if no overflow, save sum of numbers div by 3 into R23:R24

MOV R24, ZL ;

|  |  |  |  |
| --- | --- | --- | --- |
| 4. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 4/D |  |  |

/\*

d) Check and set register R07.3 if the sum is greater than 8-bits.

\*/

MOV R18, R23 ; sum registers R23:24

ADC R18, R24

ADC R18, R20 ; sum registers R20:21 and store in R18

ADC R18, R21

LDI R19, 8 ; set bit 3 (0001000)

CPI R18, 0 ; compare if R18 has carry

BRCS setReg7bit3 ; branch with carry set if there is carry

setReg7bit3:

MOV R7, R19 ; set bit 3 to R7 if the sum of R23:R24 and R20:R21 is over 8 bits

|  |  |  |  |
| --- | --- | --- | --- |
| 5. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 5/E |  |  |

/\*

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

- The execution time is Stop Watch divided by Cycle Counter.

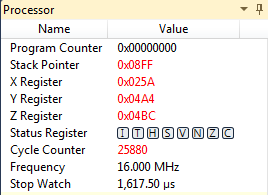
\*/

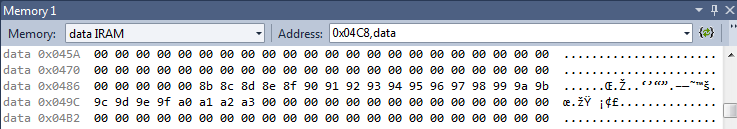
Execution Time = 1,617.50µs / 25880 = 62.5 ns

|  |  |  |  |
| --- | --- | --- | --- |
| 6. | SCREENSHOTS OF EACH TASK OUTPUT |  |  |

TASK 1/A:

25 values are stored in data IRAM starting from data 0x048b to data 0x04a3.





TASK 2/B:

R20:R21 contains the sum of numbers divisible by 7.

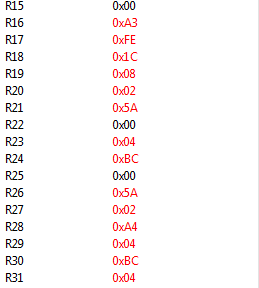
R20 -> low byte -> 0x02 R21 -> high byte -> 0x5A

TASK 3/C:

R20:R21 contains the sum of numbers divisible by 3.

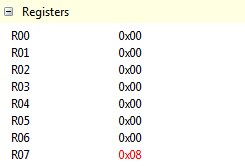
R23 -> low byte -> 0x04 R24 -> high byte -> 0xBC





TASK 4/D:

R07 bit is set to 3 (0x08) when overflow occurs.



TASK 5/E:

Execution Time = 1,617.50µs / 25880 = 62.5 ns



|  |  |  |  |
| --- | --- | --- | --- |
| 7. | GITHUB LINK OF THE DA |  |  |
| https://github.com/magor1/embedded-design-VM.git | | | |

**Student Academic Misconduct Policy**

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

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

RALPH MAGO