CPE301 – SPRING 2019

Design Assignment 1B

Student Name: Clayton Higbee

Student #: 5001189864

Student Email: higbee@unlv.nevada.edu

Primary Github address: <https://github.com/claytonjhigbee/CODSWORTH_MAIN>

Directory: <https://github.com/claytonjhigbee/CODSWORTH_MAIN/tree/master/DA/LAB1B>

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

No components were needed for this assignment

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

|  |
| --- |
| ;  ; LAB\_1B.asm  ;  ; Created: 2/22/2019 9:35:24 AM  ; Author : clayt  ;  .EQU STARTADDS = 0x0200  .ORG 0  ;X SRAM Location: 0x0200 - Store all 99 numbers  LDI R26, LOW(STARTADDS) ; Initialize XL Register  LDI R27, HIGH(STARTADDS) ; Initialize XH Register  ; Y SRAM Location: 0x0400 - Store only numbers divisible by 3  LDI R28, 0x00 ; Initialize YL Register  LDI R29, 0x04 ; Initialize YH Register  ; Z SRAM Location: 0x0600 - Store only numbers not divisible by 3  LDI R30, 0x00 ; Initialize ZL Register  LDI R31, 0x06 ; Initialize ZH Register  ; Add the Total sum of numbers stored from 0x0400  LDI R16, 0 ; Initialize Sum Low  LDI R17, 0 ; Initialize Sum High  ; Add the Total sum of numbers stored from 0x0600  LDI R18, 0 ; Initialize Sum Low  LDI R19, 0 ; Initialize Sum High  ; Counter of how many total numbers have been stored  LDI R21, 99 ; 99 total numbers to be stored  ; 99 Numbers to be between 10 and 255  LDI R20 , 11 ; First Number to be stored  START:  ST X+, R20 ; Store the current number in the STARTADDS Location  ; Check if the number is divisible by 3 or not  RJMP DIVIS3CHECK  MAIN:  INC R20 ; Increment the Number to be counted  DEC R21 ; Decrement Counter  CPI R21, 0 ; Compare to zero to see if counter is finished  BREQ END ; If count is done, END program  RJMP START  END: JMP END ; Infinite Loop, do nothing  DIVIS3CHECK:  MOV R22, R20  LA1:  SUBI R22, 3  IN R23, 0x3F ; Get the entire SREG from IO Registers  ; Check if Z flag(bit 1) or C Flag(bit 0) are currently set  MOV R24, R23 ; Store SREG in R24 for inspection  SUBI R24, 0b00110101 ; Check to see if not divisible by 3, If Carry bit is set after subtraction, then not divisible  BREQ NOTDIVISIBLE  MOV R24, R23 ; Store SREG in R24 for inspection  SUBI R24, 0b00000010 ; Check to see if divisible by 3, If Zero bit is set after subtraction, then its divisible  BREQ DIVISIBLE  ; If neither are met, then continue to subtract by 3  RJMP LA1  DIVISIBLE:  ST Y+, R20 ; Store current value in Divisible List  LDI R25, 0 ; For Convenience  ADD R16, R20 ; Add to total sum LOW Byte  ADC R17, R25 ; Add to total sum HIGH Byte  RJMP MAIN  NOTDIVISIBLE:  ST Z+, R20 ; Store current value in Not Divisible List  LDI R25, 0 ; For Convenience  ADD R18, R20 ; Add to total sum LOW Byte  ADC R19, R25 ; Add to total sum HIGH Byte  RJMP MAIN |

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

No modified code

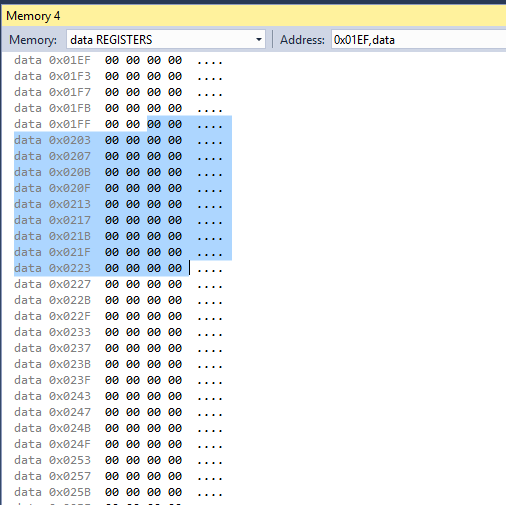
1. **SCHEMATICS**

No schematics needed

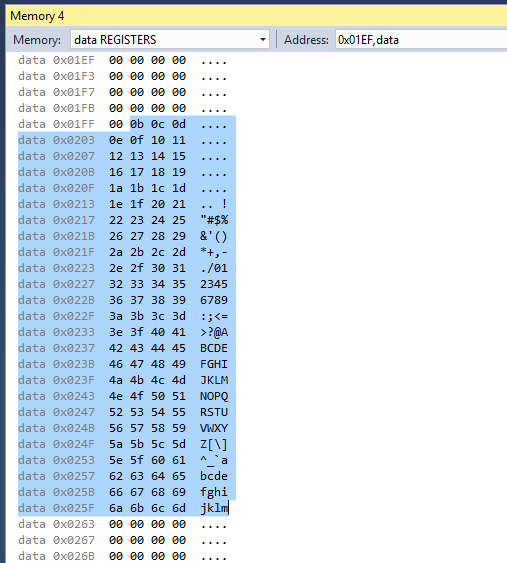
1. **SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)**

Task 1 – Store 99 numbers, between 10 and 255, starting at SRAM location 0x0200

Initial SRAM memory location:



Result after program run:



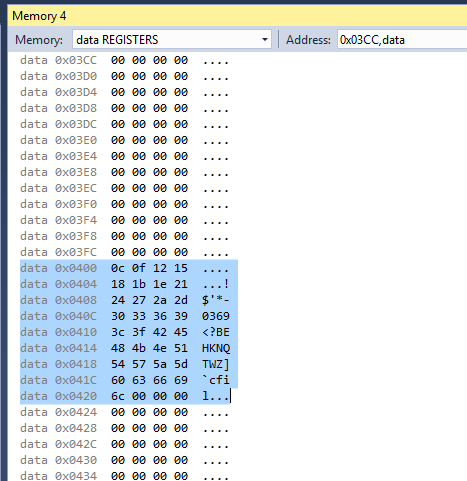
Implementing this code, I stored 11 through 109 in the SRAM locations from 0x0200 to 0x0262. These numbers are represented here in hexadecimal rather than decimal.

Task 2 – Check through the numbers to see if they are divisible by 3 or not. If divisible, store in locations starting from 0x0400. If not, store in locations starting from 0x0600.

The following code is used to check if the current number in the storing series is divisible by 3 or not. This is done by subtracting 3 continuously from the number until the SREG either receives H, S, N, and C flags or just the Z flag. If the first occurs, its not divisible. If the second occurs, it is divisible. Divisible numbers will be stored in locations from 0x0400 and non-divisible numbers will be stored starting in 0x0600.

|  |
| --- |
| DIVIS3CHECK:  MOV R22, R20  LA1:  SUBI R22, 3  IN R23, 0x3F ; Get the entire SREG from IO Registers  ; Check if Z flag(bit 1) or C Flag(bit 0) are currently set  MOV R24, R23 ; Store SREG in R24 for inspection  SUBI R24, 0b00110101 ; Check to see if not divisible by 3, If Carry bit is set after subtraction, then not divisible  BREQ NOTDIVISIBLE  MOV R24, R23 ; Store SREG in R24 for inspection  SUBI R24, 0b00000010 ; Check to see if divisible by 3, If Zero bit is set after subtraction, then its divisible  BREQ DIVISIBLE  ; If neither are met, then continue to subtract by 3  RJMP LA1  DIVISIBLE:  ST Y+, R20 ; Store current value in Divisible List  LDI R25, 0 ; For Convenience  ADD R16, R20 ; Add to total sum LOW Byte  ADC R17, R25 ; Add to total sum HIGH Byte  RJMP MAIN  NOTDIVISIBLE:  ST Z+, R20 ; Store current value in Not Divisible List  LDI R25, 0 ; For Convenience  ADD R18, R20 ; Add to total sum LOW Byte  ADC R19, R25 ; Add to total sum HIGH Byte  RJMP MAIN |

Results of store into 0x0400 Section

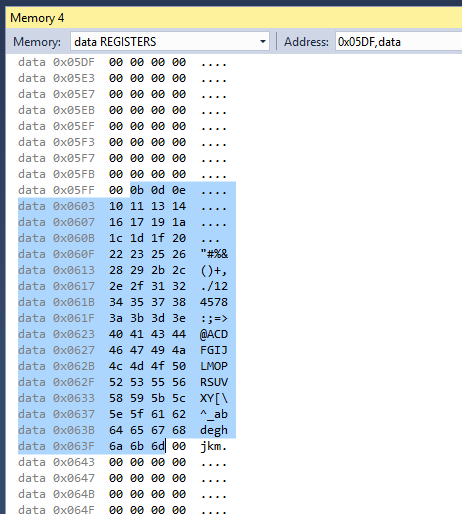


Storage occurred from 0x0400 to 0x0420

Resulting Decimal Numbers Stored:

|  |  |
| --- | --- |
|  | 0x0400 |
| Location | Divisble by 3 |
| 400 | 12 |
| 401 | 15 |
| 402 | 18 |
| 403 | 21 |
| 404 | 24 |
| 405 | 27 |
| 406 | 30 |
| 407 | 33 |
| 408 | 36 |
| 409 | 39 |
| 40A | 42 |
| 40B | 45 |
| 40C | 48 |
| 40D | 51 |
| 40E | 54 |
| 40F | 57 |
| 410 | 60 |
| 411 | 63 |
| 412 | 66 |
| 413 | 69 |
| 414 | 72 |
| 415 | 75 |
| 416 | 78 |
| 417 | 81 |
| 418 | 84 |
| 419 | 87 |
| 41A | 90 |
| 41B | 93 |
| 41C | 96 |
| 41D | 99 |
| 41E | 102 |
| 41F | 105 |
| 420 | 108 |

Results of store into 0x0600 Section



Storage occurred from 0x0600 to 0x0641

Resulting Decimal Numbers Stored:

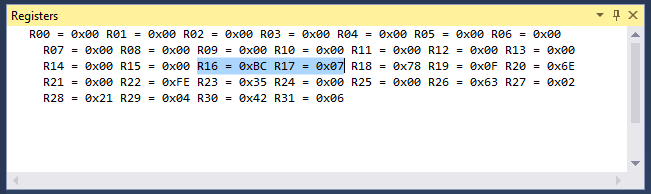
|  |  |
| --- | --- |
|  | 0x0600 |
| Location | Not Divisible by 3 |
| 600 | 11 |
| 601 | 13 |
| 602 | 14 |
| 603 | 16 |
| 604 | 17 |
| 605 | 19 |
| 606 | 20 |
| 607 | 22 |
| 608 | 23 |
| 609 | 25 |
| 60A | 26 |
| 60B | 28 |
| 60C | 29 |
| 60D | 31 |
| 60E | 32 |
| 60F | 34 |
| 610 | 35 |
| 611 | 37 |
| 612 | 38 |
| 613 | 40 |
| 614 | 41 |
| 615 | 43 |
| 616 | 44 |
| 617 | 46 |
| 618 | 47 |
| 619 | 49 |
| 61A | 50 |
| 61B | 52 |
| 61C | 53 |
| 61D | 55 |
| 61E | 56 |
| 61F | 58 |
| 620 | 59 |
| 621 | 61 |
| 622 | 62 |
| 623 | 64 |
| 624 | 65 |
| 625 | 67 |
| 626 | 68 |
| 627 | 70 |
| 628 | 71 |
| 629 | 73 |
| 62A | 74 |
| 62B | 76 |
| 62C | 77 |
| 62D | 79 |
| 62E | 80 |
| 62F | 82 |
| 630 | 83 |
| 631 | 85 |
| 632 | 86 |
| 633 | 88 |
| 634 | 89 |
| 635 | 91 |
| 636 | 92 |
| 637 | 94 |
| 638 | 95 |
| 639 | 97 |
| 63A | 98 |
| 63B | 100 |
| 63C | 101 |
| 63D | 103 |
| 63E | 104 |
| 63F | 106 |
| 640 | 107 |
| 641 | 109 |

Task 3 – Sum up numbers for each divisible and non-divisible number lists respectively into R16:R17 and R18:R19.

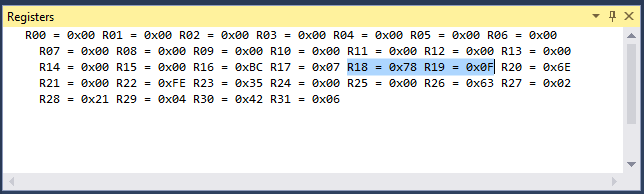
Within the code sections checking for divisibility, the numbers were also already prepped to be summed and stored.

|  |
| --- |
| DIVISIBLE:  ST Y+, R20 ; Store current value in Divisible List  LDI R25, 0 ; For Convenience  ADD R16, R20 ; Add to total sum LOW Byte  ADC R17, R25 ; Add to total sum HIGH Byte  RJMP MAIN  NOTDIVISIBLE:  ST Z+, R20 ; Store current value in Not Divisible List  LDI R25, 0 ; For Convenience  ADD R18, R20 ; Add to total sum LOW Byte  ADC R19, R25 ; Add to total sum HIGH Byte  RJMP MAIN |

Results for Divisible List Sum:



Results for Non-Divisible List Sum:



Task 4 – Verify your algorithm and answers using C or any high-level program

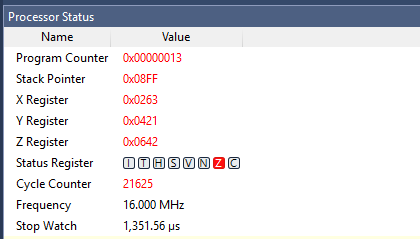
I chose to use Excel as the high-level implementation. It allowed me to easily show all the results. The Excel file that produced these results is named: LAB1B\_TYY

Results are posted below:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | 0x0200 |  |
| # Count | Location | Number to be Stored | Divisible by 3? |
| 1 | 200 | 11 | FALSE |
| 2 | 201 | 12 | TRUE |
| 3 | 202 | 13 | FALSE |
| 4 | 203 | 14 | FALSE |
| 5 | 204 | 15 | TRUE |
| 6 | 205 | 16 | FALSE |
| 7 | 206 | 17 | FALSE |
| 8 | 207 | 18 | TRUE |
| 9 | 208 | 19 | FALSE |
| 10 | 209 | 20 | FALSE |
| 11 | 20A | 21 | TRUE |
| 12 | 20B | 22 | FALSE |
| 13 | 20C | 23 | FALSE |
| 14 | 20D | 24 | TRUE |
| 15 | 20E | 25 | FALSE |
| 16 | 20F | 26 | FALSE |
| 17 | 210 | 27 | TRUE |
| 18 | 211 | 28 | FALSE |
| 19 | 212 | 29 | FALSE |
| 20 | 213 | 30 | TRUE |
| 21 | 214 | 31 | FALSE |
| 22 | 215 | 32 | FALSE |
| 23 | 216 | 33 | TRUE |
| 24 | 217 | 34 | FALSE |
| 25 | 218 | 35 | FALSE |
| 26 | 219 | 36 | TRUE |
| 27 | 21A | 37 | FALSE |
| 28 | 21B | 38 | FALSE |
| 29 | 21C | 39 | TRUE |
| 30 | 21D | 40 | FALSE |
| 31 | 21E | 41 | FALSE |
| 32 | 21F | 42 | TRUE |
| 33 | 220 | 43 | FALSE |
| 34 | 221 | 44 | FALSE |
| 35 | 222 | 45 | TRUE |
| 36 | 223 | 46 | FALSE |
| 37 | 224 | 47 | FALSE |
| 38 | 225 | 48 | TRUE |
| 39 | 226 | 49 | FALSE |
| 40 | 227 | 50 | FALSE |
| 41 | 228 | 51 | TRUE |
| 42 | 229 | 52 | FALSE |
| 43 | 22A | 53 | FALSE |
| 44 | 22B | 54 | TRUE |
| 45 | 22C | 55 | FALSE |
| 46 | 22D | 56 | FALSE |
| 47 | 22E | 57 | TRUE |
| 48 | 22F | 58 | FALSE |
| 49 | 230 | 59 | FALSE |
| 50 | 231 | 60 | TRUE |
| 51 | 232 | 61 | FALSE |
| 52 | 233 | 62 | FALSE |
| 53 | 234 | 63 | TRUE |
| 54 | 235 | 64 | FALSE |
| 55 | 236 | 65 | FALSE |
| 56 | 237 | 66 | TRUE |
| 57 | 238 | 67 | FALSE |
| 58 | 239 | 68 | FALSE |
| 59 | 23A | 69 | TRUE |
| 60 | 23B | 70 | FALSE |
| 61 | 23C | 71 | FALSE |
| 62 | 23D | 72 | TRUE |
| 63 | 23E | 73 | FALSE |
| 64 | 23F | 74 | FALSE |
| 65 | 240 | 75 | TRUE |
| 66 | 241 | 76 | FALSE |
| 67 | 242 | 77 | FALSE |
| 68 | 243 | 78 | TRUE |
| 69 | 244 | 79 | FALSE |
| 70 | 245 | 80 | FALSE |
| 71 | 246 | 81 | TRUE |
| 72 | 247 | 82 | FALSE |
| 73 | 248 | 83 | FALSE |
| 74 | 249 | 84 | TRUE |
| 75 | 24A | 85 | FALSE |
| 76 | 24B | 86 | FALSE |
| 77 | 24C | 87 | TRUE |
| 78 | 24D | 88 | FALSE |
| 79 | 24E | 89 | FALSE |
| 80 | 24F | 90 | TRUE |
| 81 | 250 | 91 | FALSE |
| 82 | 251 | 92 | FALSE |
| 83 | 252 | 93 | TRUE |
| 84 | 253 | 94 | FALSE |
| 85 | 254 | 95 | FALSE |
| 86 | 255 | 96 | TRUE |
| 87 | 256 | 97 | FALSE |
| 88 | 257 | 98 | FALSE |
| 89 | 258 | 99 | TRUE |
| 90 | 259 | 100 | FALSE |
| 91 | 25A | 101 | FALSE |
| 92 | 25B | 102 | TRUE |
| 93 | 25C | 103 | FALSE |
| 94 | 25D | 104 | FALSE |
| 95 | 25E | 105 | TRUE |
| 96 | 25F | 106 | FALSE |
| 97 | 260 | 107 | FALSE |
| 98 | 261 | 108 | TRUE |
| 99 | 262 | 109 | FALSE |

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0x0400 |  | 0x0600 |
| Location | Divisble by 3 | Location | Not Divisible by 3 |
| 400 | 12 | 600 | 11 |
| 401 | 15 | 601 | 13 |
| 402 | 18 | 602 | 14 |
| 403 | 21 | 603 | 16 |
| 404 | 24 | 604 | 17 |
| 405 | 27 | 605 | 19 |
| 406 | 30 | 606 | 20 |
| 407 | 33 | 607 | 22 |
| 408 | 36 | 608 | 23 |
| 409 | 39 | 609 | 25 |
| 40A | 42 | 60A | 26 |
| 40B | 45 | 60B | 28 |
| 40C | 48 | 60C | 29 |
| 40D | 51 | 60D | 31 |
| 40E | 54 | 60E | 32 |
| 40F | 57 | 60F | 34 |
| 410 | 60 | 610 | 35 |
| 411 | 63 | 611 | 37 |
| 412 | 66 | 612 | 38 |
| 413 | 69 | 613 | 40 |
| 414 | 72 | 614 | 41 |
| 415 | 75 | 615 | 43 |
| 416 | 78 | 616 | 44 |
| 417 | 81 | 617 | 46 |
| 418 | 84 | 618 | 47 |
| 419 | 87 | 619 | 49 |
| 41A | 90 | 61A | 50 |
| 41B | 93 | 61B | 52 |
| 41C | 96 | 61C | 53 |
| 41D | 99 | 61D | 55 |
| 41E | 102 | 61E | 56 |
| 41F | 105 | 61F | 58 |
| 420 | 108 | 620 | 59 |
|  |  | 621 | 61 |
|  |  | 622 | 62 |
|  |  | 623 | 64 |
|  |  | 624 | 65 |
|  |  | 625 | 67 |
|  |  | 626 | 68 |
|  |  | 627 | 70 |
|  |  | 628 | 71 |
|  |  | 629 | 73 |
|  |  | 62A | 74 |
|  |  | 62B | 76 |
|  |  | 62C | 77 |
|  |  | 62D | 79 |
|  |  | 62E | 80 |
|  |  | 62F | 82 |
|  |  | 630 | 83 |
|  |  | 631 | 85 |
|  |  | 632 | 86 |
|  |  | 633 | 88 |
|  |  | 634 | 89 |
|  |  | 635 | 91 |
|  |  | 636 | 92 |
|  |  | 637 | 94 |
|  |  | 638 | 95 |
|  |  | 639 | 97 |
|  |  | 63A | 98 |
|  |  | 63B | 100 |
|  |  | 63C | 101 |
|  |  | 63D | 103 |
|  |  | 63E | 104 |
|  |  | 63F | 106 |
|  |  | 640 | 107 |
|  |  | 641 | 109 |
| SUMS: | 1980 |  | 3960 |
|  | in R17:R16 |  | in R19:18 |
| Binary: | 11110111100 | Binary: | 111101111000 |
| Hex: | 0x07BC | Hex: | 0x0F78 |

Task 5 – Program Execution time at 16MHz



**21625 Cycles accumulates to 1,351.56μs at 16Mhz.**

FINAL NOTE: I think it was crucial in the design of this code to combine the summing and storing procedures in the operation of the code. It was easy to break up the storage and sum into the related types of divisible and non-divisible, basically allowing it to do it virtually at the same time.

Please let me know if I can make any improvements on this design assignment as well as other pending ones to come.

Thank you!

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

No board screenshots needed

1. **VIDEO LINKS OF EACH DEMO**

No video links needed

1. **GITHUB LINK OF THIS DA**

<https://github.com/claytonjhigbee/CODSWORTH_MAIN/tree/master/DA/LAB1B>

**Student Academic Misconduct Policy**

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

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

Clayton Higbee