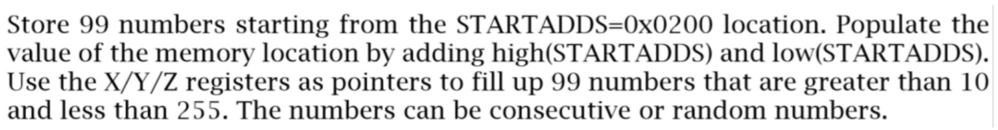
CPE301 – SPRING 2019

Design Assignment 1B

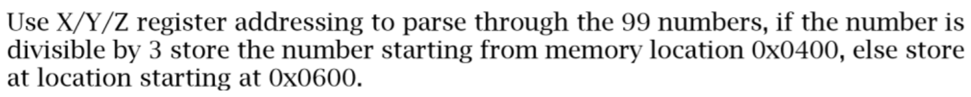
Student Name: Robert Sander
Student #: 5002102412
Student Email: sander1@unlv.nevada.edu
Primary Github address: https://github.com/sanderUNLV/submission_DA.git
Youtube link: https://youtu.be/cYE0_ewTX2Q

1. DEVELOPED CODE OF TASK 1/B


Store 99 numbers starting from the STARTADDS=0x0200 location. Populate the value of the memory location by adding high(STARTADDS) and low(STARTADDS). Use the X/Y/Z registers as pointers to fill up 99 numbers that are greater than 10 and less than 255. The numbers can be consecutive or random numbers.

```
; Author : Robert Sander

.include<m328pdef.inc>

.CSEG
.ORG 0x00

       LDI R19, 99                 ;R19 = 99 (R19 for counter)
       LDI R16,0x0B         ;load R16 with value 0xB0 (value to be copied)
       LDI XL,LOW($200)     ;load the low byte of X with value 0x00
       LDI XH,HIGH($200)    ;load the high byte of X with value 0x2
L1:
       ST X+,R16     ;copy R16 to memory location X and increments the location by 1
       INC R16              ;Increments R16 by 1 every loop, 99 times
       DEC R19              ;decrement the counter by 1
       BRNE L1              ;loop until R19 (counter) = zero

END: RJMP END
```

2. DEVELOPED CODE OF TASK 2/B from TASK 1/B


Use X/Y/Z register addressing to parse through the 99 numbers, if the number is divisible by 3 store the number starting from memory location 0x0400, else store at location starting at 0x0600.

```
; Author : Robert Sander

.include<m328pdef.inc>

.ORG 0x00

       LDI R16, 99                 ;R16 = 99 (R16 FOR COUNTER)
       LDI R17,0x0B         ;LOAD R17 WITH VALUE 0x0B
       LDI XL,LOW($200)     ;LOAD THE LOW BYTE OF X WITH VALUE 0x00 - STARTING POSITION OF
THE POPULATED NUMBERS
       LDI XH,HIGH($200)    ;LOAD THE HIGH BYTE OF X WITH VALUE 0x2 - STARTING POSITION OF
THE POPULATED NUMBERS
       LDI YL,LOW($400)     ;LOAD THE LOW BYTE OF Y WITH VALUE 0x00 - FOR THE NUMBERS THAT
ARE DIVISBLE BY THREE
       LDI YH,HIGH($400)    ;LOAD THE HIGH BYTE OF Y WITH VALUE 0x4 - FOR THE NUMBERS THAT
ARE DIVISBLE BY THREE
       LDI    ZL, LOW($600) ;LOAD THE LOW BYTE OF Z WITH VALUE 0x00 - FOR THE NUMBERS THAT
ARE NOT DIVISBLE BY THREE
       LDI ZH,HIGH($600)    ;LOAD THE HIGH BYTE OF Z WITH VALUE 0x6 - FOR THE NUMBERS THAT
ARE NOT DIVISBLE BY THREE
L1:
       ST X+, R17    ;COPY R17 TO MEMORY LOCATION X AND INCREMENTS THE LOCATION BY 1
       LDI R18, 0    ;LOAD THE VALUE 0 INTO R18
       ADD R18, R17  ;ADD R17 TO R18 AND STORE IN R18 - LINE 19
GREATERTHANTHREE:
       SUBI R18, 3   ;SUBTRACT 3 FROM R18 AND STORE THE RESULT IN R18
       CPI R18, 0    ;COMPARE R18 WITH THE VALUE 0
       BRNE DONTSAVE ;IF R18 IS NOT EQUAL TO 0 GO TO 'DONTSAVE:'
       ST Y+, R17 ;COPY R17 TO MEMORY LOCATION Y AND INCREMENTS THE LOCATION BY 1 - FOR
THE NUMBERS THAT ARE DIVISBLE BY THREE
DONTSAVE:
       CPI R18, 3 ;COMPARE R18 WITH THE VALUE 3
       BRGE GREATERTHANTHREE       ;IF R18 IS GREATER THAN OR EQUAL TO 3 GO TO
'GREATERTHANTHREE:'
       CPI R18, 0 ;COMPARE R18 WITH THE VALUE 0
       BREQ SKIP ;IF R18 IS NOT EQUAL TO 0 GO TO 'SKIP:'
       ST Z+, R17 ;COPY R17 TO MEMORY LOCATION Z AND INCREMENTS THE LOCATION BY 1 - FOR
THE NUMBERS THAT ARE NOT DIVISBLE BY THREE
SKIP:
       INC R17              ;INCREMENTS R17 BY 1 EVERY LOOP, 99 TIMES
       DEC R16              ;DECREMENT THE COUNTER BY 1
       BRNE L1              ;LOOP UNTIL R16 (COUNTER) = ZERO

END: RJMP END

;TO CHECK IF DIVISIBLE BY THREE
;SUBTRACT 3 UNTIL NUMBER EQUALS ZERO, IF THERE IS A REMAINDER AFTER IT GOES TO ZERO THEN
NUMBER IS NOT DIVISIBLE BY THREE
;STORE IN 0x0600
;IF NO REMAINDER, NUMBER IS DIVISBLE BY THREE
;STORE IN 0x0400
```

3. DEVELOPED CODE OF TASK 3/B from TASK 2/B

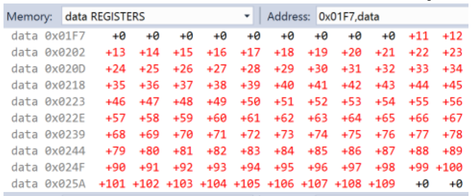
Use X/Y/Z register addressing to simultaneously add numbers from memory location 0x0400 and 0x0600 and store the sums at R16:R17 and R18:R19 respectively. Pay attention to the carry overflow.

```
; Author : Robert Sander

.include<m328pdef.inc>

.ORG 0x00

       LDI R16, 99                  ;R16 = 99 (R16 FOR COUNTER)
       LDI R17,0x0B          ;LOAD R17 WITH VALUE 0x0B
       LDI XL,LOW($200)      ;LOAD THE LOW BYTE OF X WITH VALUE 0x00 - STARTING POSITION OF
THE POPULATED NUMBERS
       LDI XH,HIGH($200)     ;LOAD THE HIGH BYTE OF X WITH VALUE 0x2 - STARTING POSITION OF
THE POPULATED NUMBERS
       LDI YL,LOW($400)      ;LOAD THE LOW BYTE OF Y WITH VALUE 0x00 - FOR THE NUMBERS THAT
ARE DIVISBLE BY THREE
       LDI YH,HIGH($400)     ;LOAD THE HIGH BYTE OF Y WITH VALUE 0x4 - FOR THE NUMBERS THAT
ARE DIVISBLE BY THREE
       LDI    ZL, LOW($600) ;LOAD THE LOW BYTE OF Z WITH VALUE 0x00 - FOR THE NUMBERS THAT
ARE NOT DIVISBLE BY THREE
       LDI ZH,HIGH($600)     ;LOAD THE HIGH BYTE OF Z WITH VALUE 0x6 - FOR THE NUMBERS THAT
ARE NOT DIVISBLE BY THREE
L1:
       ST X+, R17    ;COPY R17 TO MEMORY LOCATION X AND INCREMENTS THE LOCATION BY 1
       LDI R18, 0    ;LOAD THE VALUE 0 INTO R18
       ADD R18, R17  ;ADD R17 TO R18 AND STORE IN R18 - LINE 19
GREATERTHANTHREE:
       SUBI R18, 3   ;SUBTRACT 3 FROM R18 AND STORE THE RESULT IN R18
       CPI R18, 0    ;COMPARE R18 WITH THE VALUE 0
       BRNE DONTSAVE ;IF R18 IS NOT EQUAL TO 0 GO TO 'DONTSAVE:'
       ST Y+, R17 ;COPY R17 TO MEMORY LOCATION Y AND INCREMENTS THE LOCATION BY 1 - FOR
THE NUMBERS THAT ARE DIVISBLE BY THREE
DONTSAVE:
       CPI R18, 3 ;COMPARE R18 WITH THE VALUE 3
       BRGE GREATERTHANTHREE       ;IF R18 IS GREATER THAN OR EQUAL TO 3 GO TO
'GREATERTHANTHREE:'
       CPI R18, 0 ;COMPARE R18 WITH THE VALUE 0
       BREQ SKIP ;IF R18 IS NOT EQUAL TO 0 GO TO 'SKIP:'
       ST Z+, R17 ;COPY R17 TO MEMORY LOCATION Z AND INCREMENTS THE LOCATION BY 1 - FOR
THE NUMBERS THAT ARE NOT DIVISBLE BY THREE
SKIP:
       INC R17               ;INCREMENTS R17 BY 1 EVERY LOOP, 99 TIMES
       DEC R16               ;DECREMENT THE COUNTER BY 1
       BRNE L1               ;LOOP UNTIL R16 (COUNTER) = ZERO

       LDI R20, 99   ;R20 = 99 (R16 for counter)
       LDI R16, 0    ;LOAD THE VALUE 0 INTO R16
       LDI R17, 0    ;LOAD THE VALUE 0 INTO R17
       LDI R18, 0    ;LOAD THE VALUE 0 INTO R18
       LDI R19, 0    ;LOAD THE VALUE 0 INTO R19
       CLC    ;CLEAR THE CARRY FLAG
L2:
       LD R21, -Y    ;LOAD VALUE AT ADDRESS Y INTO R21 AND DECREMENT Y BY 1
```

```
        ADC R16, R21  ;ADD WITH CARRY R21 AND R16 AND STORE THE RESULT IN R16
        BRCC NOCARRY_Y       ;IF CARRY FLAG IS CLEAR GO TO 'NOCARRY_Y:'
        INC R17       ;INCREMENT R17 BY 1 IF CARRY IS NOT CLEAR
NOCARRY_Y:
        CLC    ;CLEAR THE CARRY FLAG
               LD R22, -Z    ;LOAD VALUE AT ADDRESS Z INTO R22 AND DECREMENT Z BY 1
        ADC R18, R22  ;ADD WITH CARRY R22 AND R18 AND STORE THE RESULT IN R18
        BRCC NOCARRY_Z       ;IF CARRY FLAG IS CLEAR GO TO 'NOCARRY_Z:'
        INC R19       ;INCREMENT R19 BY 1 IF CARRY IS NOT CLEAR
NOCARRY_Z:
        CLC    ;CLEAR THE CARRY FLAG
        DEC R20       ;DECREMENT THE COUNTER BY 1
        BRNE L2

END: RJMP END

;TO CHECK IF DIVISIBLE BY THREE
;SUBTRACT 3 UNTIL NUMBER EQUALS ZERO, IF THERE IS A REMAINDER AFTER IT GOES TO ZERO THEN
NUMBER IS NOT DIVISIBLE BY THREE
;STORE IN 0x0600
;IF NO REMAINDER, NUMBER IS DIVISBLE BY THREE
;STORE IN 0x0400
```

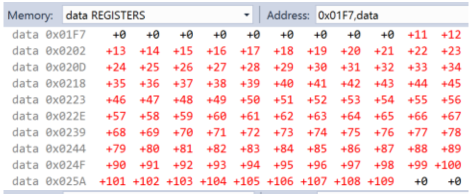
4. SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)

4.1

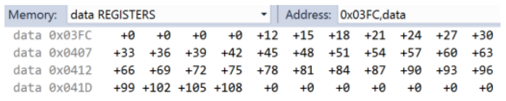


```
Memory: data REGISTERS          Address: 0x01F7,data
data 0x01F7    +0   +0   +0   +0   +0   +0   +0   +0   +0  +11  +12
data 0x0202   +13  +14  +15  +16  +17  +18  +19  +20  +21  +22  +23
data 0x020D   +24  +25  +26  +27  +28  +29  +30  +31  +32  +33  +34
data 0x0218   +35  +36  +37  +38  +39  +40  +41  +42  +43  +44  +45
data 0x0223   +46  +47  +48  +49  +50  +51  +52  +53  +54  +55  +56
data 0x022E   +57  +58  +59  +60  +61  +62  +63  +64  +65  +66  +67
data 0x0239   +68  +69  +70  +71  +72  +73  +74  +75  +76  +77  +78
data 0x0244   +79  +80  +81  +82  +83  +84  +85  +86  +87  +88  +89
data 0x024F   +90  +91  +92  +93  +94  +95  +96  +97  +98  +99 +100
data 0x025A  +101 +102 +103 +104 +105 +106 +107 +108 +109   +0   +0
```

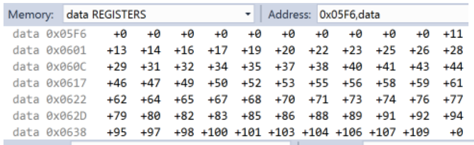
4.2



```
Memory: data REGISTERS          Address: 0x01F7,data
data 0x01F7    +0   +0   +0   +0   +0   +0   +0   +0   +0  +11  +12
data 0x0202   +13  +14  +15  +16  +17  +18  +19  +20  +21  +22  +23
data 0x020D   +24  +25  +26  +27  +28  +29  +30  +31  +32  +33  +34
data 0x0218   +35  +36  +37  +38  +39  +40  +41  +42  +43  +44  +45
data 0x0223   +46  +47  +48  +49  +50  +51  +52  +53  +54  +55  +56
data 0x022E   +57  +58  +59  +60  +61  +62  +63  +64  +65  +66  +67
data 0x0239   +68  +69  +70  +71  +72  +73  +74  +75  +76  +77  +78
data 0x0244   +79  +80  +81  +82  +83  +84  +85  +86  +87  +88  +89
data 0x024F   +90  +91  +92  +93  +94  +95  +96  +97  +98  +99 +100
data 0x025A  +101 +102 +103 +104 +105 +106 +107 +108 +109   +0   +0
```

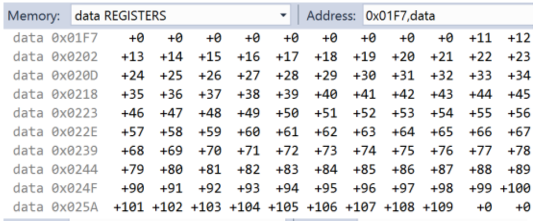


```
Memory: data REGISTERS          Address: 0x03FC,data
data 0x03FC    +0   +0   +0   +0  +12  +15  +18  +21  +24  +27  +30
data 0x0407   +33  +36  +39  +42  +45  +48  +51  +54  +57  +60  +63
data 0x0412   +66  +69  +72  +75  +78  +81  +84  +87  +90  +93  +96
data 0x041D   +99 +102 +105 +108   +0   +0   +0   +0   +0   +0   +0
```

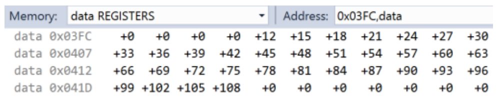


```
Memory: data REGISTERS          Address: 0x05F6,data
data 0x05F6    +0   +0   +0   +0   +0   +0   +0   +0   +0   +0  +11
data 0x0601   +13  +14  +16  +17  +19  +20  +22  +23  +25  +26  +28
data 0x060C   +29  +31  +32  +34  +35  +37  +38  +40  +41  +43  +44
data 0x0617   +46  +47  +49  +50  +52  +53  +55  +56  +58  +59  +61
data 0x0622   +62  +64  +65  +67  +68  +70  +71  +73  +74  +76  +77
data 0x062D   +79  +80  +82  +83  +85  +86  +88  +89  +91  +92  +94
data 0x0638   +95  +97  +98 +100 +101 +103 +104 +106 +107 +109   +0
```

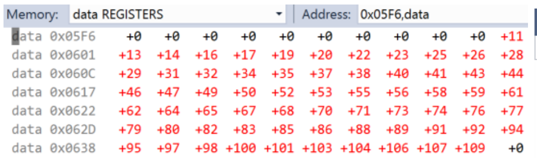
4.3



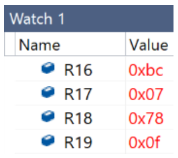
```
Memory: data REGISTERS          Address: 0x01F7,data
data 0x01F7    +0   +0   +0   +0   +0   +0   +0   +0   +0  +11  +12
data 0x0202   +13  +14  +15  +16  +17  +18  +19  +20  +21  +22  +23
data 0x020D   +24  +25  +26  +27  +28  +29  +30  +31  +32  +33  +34
data 0x0218   +35  +36  +37  +38  +39  +40  +41  +42  +43  +44  +45
data 0x0223   +46  +47  +48  +49  +50  +51  +52  +53  +54  +55  +56
data 0x022E   +57  +58  +59  +60  +61  +62  +63  +64  +65  +66  +67
data 0x0239   +68  +69  +70  +71  +72  +73  +74  +75  +76  +77  +78
data 0x0244   +79  +80  +81  +82  +83  +84  +85  +86  +87  +88  +89
data 0x024F   +90  +91  +92  +93  +94  +95  +96  +97  +98  +99 +100
data 0x025A  +101 +102 +103 +104 +105 +106 +107 +108 +109   +0   +0
```



```
Memory: data REGISTERS          Address: 0x03FC,data
data 0x03FC    +0   +0   +0   +0  +12  +15  +18  +21  +24  +27  +30
data 0x0407   +33  +36  +39  +42  +45  +48  +51  +54  +57  +60  +63
data 0x0412   +66  +69  +72  +75  +78  +81  +84  +87  +90  +93  +96
data 0x041D   +99 +102 +105 +108   +0   +0   +0   +0   +0   +0   +0
```



```
Memory: data REGISTERS          Address: 0x05F6,data
data 0x05F6    +0   +0   +0   +0   +0   +0   +0   +0   +0   +0  +11
data 0x0601   +13  +14  +16  +17  +19  +20  +22  +23  +25  +26  +28
data 0x060C   +29  +31  +32  +34  +35  +37  +38  +40  +41  +43  +44
data 0x0617   +46  +47  +49  +50  +52  +53  +55  +56  +58  +59  +61
data 0x0622   +62  +64  +65  +67  +68  +70  +71  +73  +74  +76  +77
data 0x062D   +79  +80  +82  +83  +85  +86  +88  +89  +91  +92  +94
data 0x0638   +95  +97  +98 +100 +101 +103 +104 +106 +107 +109   +0
```



Watch 1

Name	Value
R16	0xbc
R17	0x07
R18	0x78
R19	0x0f

Verification:

R16 0xBC - 1011 1100 (LOWER)
128+32+16+8+4=188

R17 0x07 - 0000 0111 (HIGHER)
1024+512+256=1792

4.4 R16+R17: 1792+188=1980

R18 0x78 - 0111 1000 (LOWER)
64+32+16+8=120

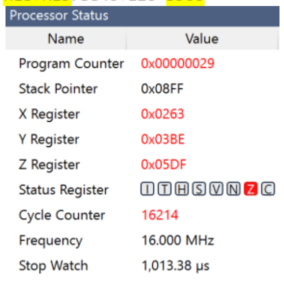
R19 0x0F - 0000 1111 (HIGHER)
3840

R18+R19: 3840+120=3960

Address: 0x400
12 +15 +18 +21 +24 +27 +30 +33 +36 +39 +42 +45 +48 +51 +54 +57 +60 +63+66 +69 +72 +75 +78 +81 +84 +87 +90 +93 +96 +99 +102 +105 +108=1,980

Address: 0x600
+11 +13 +14 +16 +17 +19 +20 +22 +23 +25 +26 +28 +29 +31 +32 +34 +35 +37 +38 + 40 +41 +43 +44 +46 +47 +49 +50 +52 +53 +55 +56 +58 +59 +61 +62 +64 +65 +67 +68 +70 +71 +73 +74 +76 +77 +79 +80 +82 +83 +85 +86 +88 +89 +91 +92 +94 +95 +97 +98 +100 +101 +103 +104 +106 +107 +109=3,960

4.5


Processor Status

Name	Value
Program Counter	0x00000029
Stack Pointer	0x08FF
X Register	0x0263
Y Register	0x03BE
Z Register	0x05DF
Status Register	I T H S V N Z C
Cycle Counter	16214
Frequency	16.000 MHz
Stop Watch	1,013.38 µs

This subroutine takes 1013.38µs to execute

5. VIDEO LINKS OF EACH DEMO

https://youtu.be/cYE0_ewTX2Q

6. GITHUB LINK OF THIS DA

https://github.com/sanderUNLV/submission_DA.git

Student Academic Misconduct Policy
http://studentconduct.unlv.edu/misconduct/policy.html

"This assignment submission is my own, original work".
-Robert Sander