**CPE301 – SPRING 2020**

# Design Assignment 1B

Student Name: Dan Ray Dimapilis
Student #: 5002763761
Student Email: dimapd1@unlv.nevada.edu
Primary Github address: ---
Directory: ---

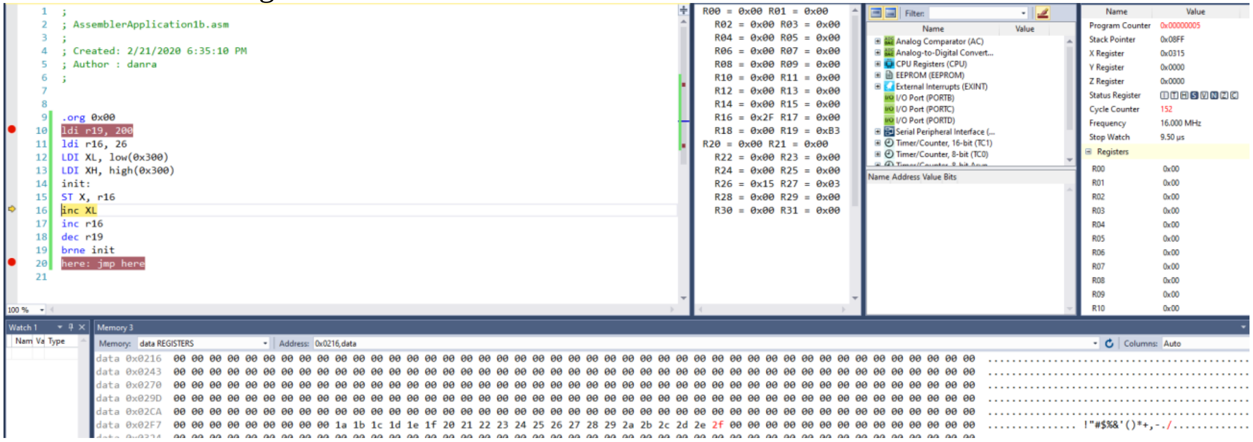
CODE:

```
;
; AssemblerApplication1b.asm
;
; Created: 2/21/2020 6:35:10 PM
; Author : danra
;


.org 0x00
sub r3, r3           ;checker if div3 == TRUE
sub r7, r7           ;                    div7 ==  TRUE
ldi r20, 0           ; div3==F & div7 ==F
sub r15, r15         ;array position
ldi r19, 200                ;200 int inputs             - counter
ldi r16, 26                        ;starting value
LDI XL, low(0x300)          ;address to store lower half
LDI XH, high(0x300)         ;                                  upper half
init:
       ;initialize 200 numbers from 26-225
ST X, r16                          ;store int to mem space
inc XL                             ;go to next mem alloc/addr
inc r16                                   ;get next num, input
dec r19                                   ;decrement counter
brne init



LDI XL, low(0x300)
LDI XH, high(0x300)
LDI YL, low(0x500)
LDI YH, high(0x500)
ldi r19, 200
ldi r16, 26

div7:
ldi ZL, low(0x600)
ldi ZH, high(0x600)
ld r1, X                    ;r1 = arr[0]
ldi r17, 7                         ;r17 = 7
lp7:
sub r1, r17                        ;r1 = arr[0] - 7
breq zero7                         ;if no remainder, divisible by 7
```

```
brpl lp7                           ;keep loop unless have remainder
jmp notdiv7                                                   ;go here if negative, not
div by 7

zero7:        ;test code, arr[i] is divisible by 7
inc r7
ld r1, X                           ;r1 = arr[0]
st Y, r1                           ;store to 0x500 for div7
jmp div3

notdiv7       :                                                      ;check if div by 3
dec r7
div3:
ld r1, X                           ;r1 = arr[0]
ldi r17, 3                         ;r17 = 3
lp3:
sub r1, r17                        ;r1 = arr[0] - 3
breq zero3                         ;if no remainder, divisible by 3
brpl lp3                           ;keep loop unless have remainder
jmp notdiv3                                                   ;go here if negative, not
div by 7

notdiv3:
dec r3
jmp final_checking

zero3:
inc r3
ld r1, X                           ;r1 = arr[0]
st Z, r1                           ;store to 0x600 for div7

final_checking:

dec r7
brmi skip
dec r3
brpl both
jmp nextnum

both:
ldi ZL, low(0x700)          ;store both to 0x700
ldi ZH, high(0x700)
add ZL, r15
ld r1, X
st Z, r1
jmp nextnum

skip:
dec r3
brpl nextnum
ldi ZL, low(0x800)                 ;store else to 0x800
ldi ZH, high(0x800)
add ZL, r15
ld r1, X
st Z, r1


nextnum:
```

```
sub r3, r3              ;checker if div3 == TRUE
sub r7, r7              ;                    div7 ==  TRUE
inc r15                         ;i++
inc XL
dec r19
breq exit
jmp div7

exit:
jmp exit                                          ;****part 1 and part 2 done*****
                                                          ;part 3, do sums
;yeah...lost on how im gonna do this part.
```

SIMULATION images:



```
;
; AssemblerApplication1b.asm
;
; Created: 2/21/2020 6:35:10 PM
; Author : danra
;


.org 0x00
ldi r19, 200
ldi r16, 26
LDI XL, low(0x300)
LDI XH, high(0x300)
init:
ST X, r16
inc XL
inc r16
dec r19
brne init
here: jmp here
```

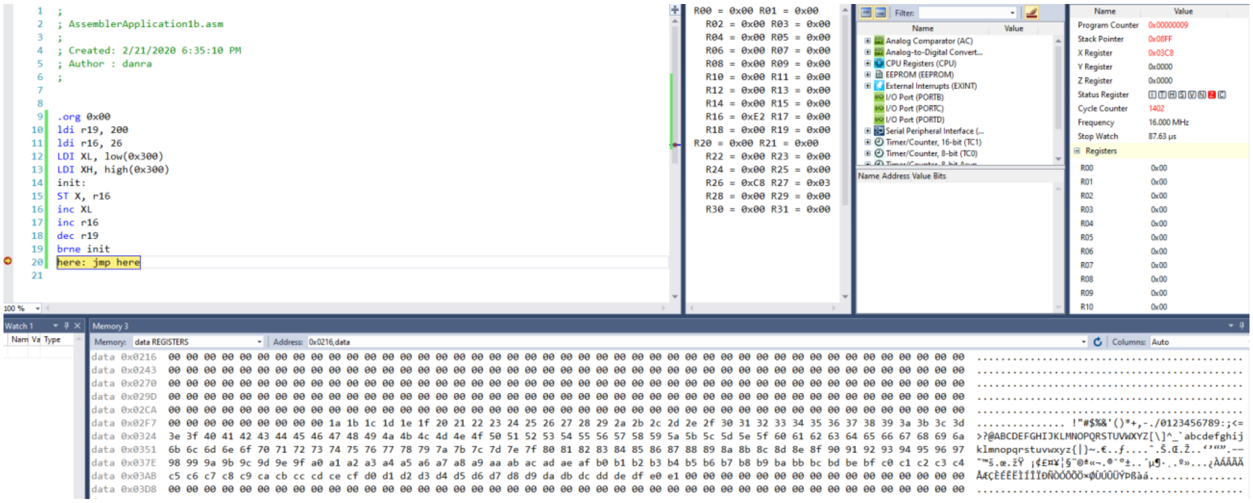
```
R00 = 0x00 R01 = 0x00
  R02 = 0x00 R03 = 0x00
  R04 = 0x00 R05 = 0x00
  R06 = 0x00 R07 = 0x00
  R08 = 0x00 R09 = 0x00
  R10 = 0x00 R11 = 0x00
  R12 = 0x00 R13 = 0x00
  R14 = 0x00 R15 = 0x00
  R16 = 0x2F R17 = 0x00
  R18 = 0x00 R19 = 0xB3
R20 = 0x00 R21 = 0x00
  R22 = 0x00 R23 = 0x00
  R24 = 0x00 R25 = 0x00
  R26 = 0x15 R27 = 0x03
  R28 = 0x00 R29 = 0x00
  R30 = 0x00 R31 = 0x00
```

| Name | Value |
|---|---|
| Program Counter | 0x00000005 |
| Stack Pointer | 0x08FF |
| X Register | 0x0315 |
| Y Register | 0x0000 |
| Z Register | 0x0000 |
| Status Register | ITHSVNZC |
| Cycle Counter | 152 |
| Frequency | 16.000 MHz |
| Stop Watch | 9.50 µs |
| Registers | |
| R00 | 0x00 |
| R01 | 0x00 |
| R02 | 0x00 |
| R03 | 0x00 |
| R04 | 0x00 |
| R05 | 0x00 |
| R06 | 0x00 |
| R07 | 0x00 |
| R08 | 0x00 |
| R09 | 0x00 |
| R10 | 0x00 |

Memory: data REGISTERS   Address: 0x0216,data   Columns: Auto

```
data 0x0216  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .............................................
data 0x0243  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .............................................
data 0x0270  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .............................................
data 0x029D  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .............................................
data 0x02CA  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .............................................
data 0x02F7  00 00 00 00 00 00 00 00 00 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ............... !"#$%&'()*+,-./..............
```

initialization part 1



```
;
; AssemblerApplication1b.asm
;
; Created: 2/21/2020 6:35:10 PM
; Author : danra
;


.org 0x00
ldi r19, 200
ldi r16, 26
LDI XL, low(0x300)
LDI XH, high(0x300)
init:
ST X, r16
inc XL
inc r16
dec r19
brne init
here: jmp here
```

```
R00 = 0x00 R01 = 0x00
  R02 = 0x00 R03 = 0x00
  R04 = 0x00 R05 = 0x00
  R06 = 0x00 R07 = 0x00
  R08 = 0x00 R09 = 0x00
  R10 = 0x00 R11 = 0x00
  R12 = 0x00 R13 = 0x00
  R14 = 0x00 R15 = 0x00
  R16 = 0xE2 R17 = 0x00
  R18 = 0x00 R19 = 0x00
R20 = 0x00 R21 = 0x00
  R22 = 0x00 R23 = 0x00
  R24 = 0x00 R25 = 0x00
  R26 = 0xC8 R27 = 0x03
  R28 = 0x00 R29 = 0x00
  R30 = 0x00 R31 = 0x00
```

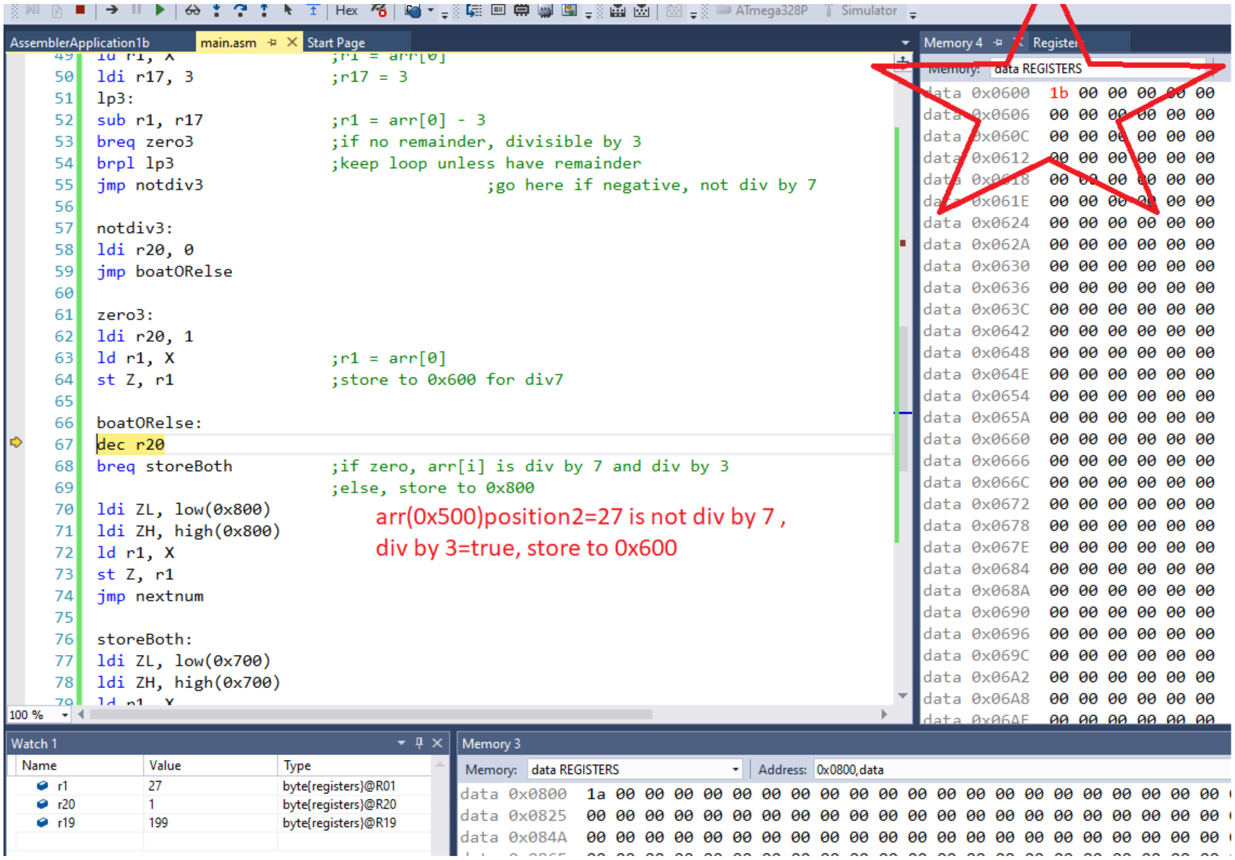
| Name | Value |
|---|---|
| Program Counter | 0x00000009 |
| Stack Pointer | 0x08FF |
| X Register | 0x03C8 |
| Y Register | 0x0000 |
| Z Register | 0x0000 |
| Status Register | ITHSVNZC |
| Cycle Counter | 1402 |
| Frequency | 16.000 MHz |
| Stop Watch | 87.63 µs |
| Registers | |
| R00 | 0x00 |
| R01 | 0x00 |
| R02 | 0x00 |
| R03 | 0x00 |
| R04 | 0x00 |
| R05 | 0x00 |
| R06 | 0x00 |
| R07 | 0x00 |
| R08 | 0x00 |
| R09 | 0x00 |
| R10 | 0x00 |

Memory: data REGISTERS   Address: 0x0216,data   Columns: Auto

```
data 0x0216  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .............................................
data 0x0243  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .............................................
data 0x0270  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .............................................
data 0x029D  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .............................................
data 0x02CA  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .............................................
data 0x02F7  00 00 00 00 00 00 00 00 00 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33 34 35 36 37 38 39 3a 3b 3c 3d  ............... !"#$%&'()*+,-./0123456789:;<=
data 0x0324  3e 3f 40 41 42 43 44 45 46 47 48 49 4a 4b 4c 4d 4e 4f 50 51 52 53 54 55 56 57 58 59 5a 5b 5c 5d 5e 5f 60 61 62 63 64 65 66 67 68 69 6a  >?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghij
data 0x0351  6b 6c 6d 6e 6f 70 71 72 73 74 75 76 77 78 79 7a 7b 7c 7d 7e 7f 80 81 82 83 84 85 86 87 88 89 8a 8b 8c 8d 8e 8f 90 91 92 93 94 95 96 97  klmnopqrstuvwxyz{|}~.€..ƒ....ˆ.Š.Œ.Ž..‘’“”.–—
data 0x037E  98 99 9a 9b 9c 9d 9e 9f a0 a1 a2 a3 a4 a5 a6 a7 a8 a9 aa ab ac ad ae af b0 b1 b2 b3 b4 b5 b6 b7 b8 b9 ba bb bc bd be bf c0 c1 c2 c3 c4  ˜™š.œ.žŸ ¡¢£¤¥¦§¨©ª«¬.®¯°±..´µ¶·¸.º»...¿ÀÁÂÃÄ
data 0x03AB  c5 c6 c7 c8 c9 ca cb cc cd ce cf d0 d1 d2 d3 d4 d5 d6 d7 d8 d9 da db dc dd de df e0 e1 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ÅÆÇÈÉÊËÌÍÎÏÐÑÒÓÔÕÖ×ØÙÚÛÜÝÞßàá................
data 0x03D8  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .............................................
```

initialization part 2, all 200 numbers are stored

DIVISION photos:



```
ld r1, X                    ;r1 = arr[0]
ldi r17, 3                  ;r17 = 3
lp3:
sub r1, r17                 ;r1 = arr[0] - 3
breq zero3                  ;if no remainder, divisible by 3
brpl lp3                    ;keep loop unless have remainder
jmp notdiv3                                 ;go here if negative, not div by 7

notdiv3:
ldi r20, 0
jmp boatORelse

zero3:
ldi r20, 1
ld r1, X                    ;r1 = arr[0]
st Z, r1                    ;store to 0x600 for div7

boatORelse:
dec r20
breq storeBoth              ;if zero, arr[i] is div by 7 and div by 3
                            ;else, store to 0x800
ldi ZL, low(0x800)
ldi ZH, high(0x800)
ld r1, X
st Z, r1
jmp nextnum

storeBoth:
ldi ZL, low(0x700)
ldi ZH, high(0x700)
ld r1, X
```

arr(0x500)position2=27 is not div by 7 ,
div by 3=true, store to 0x600

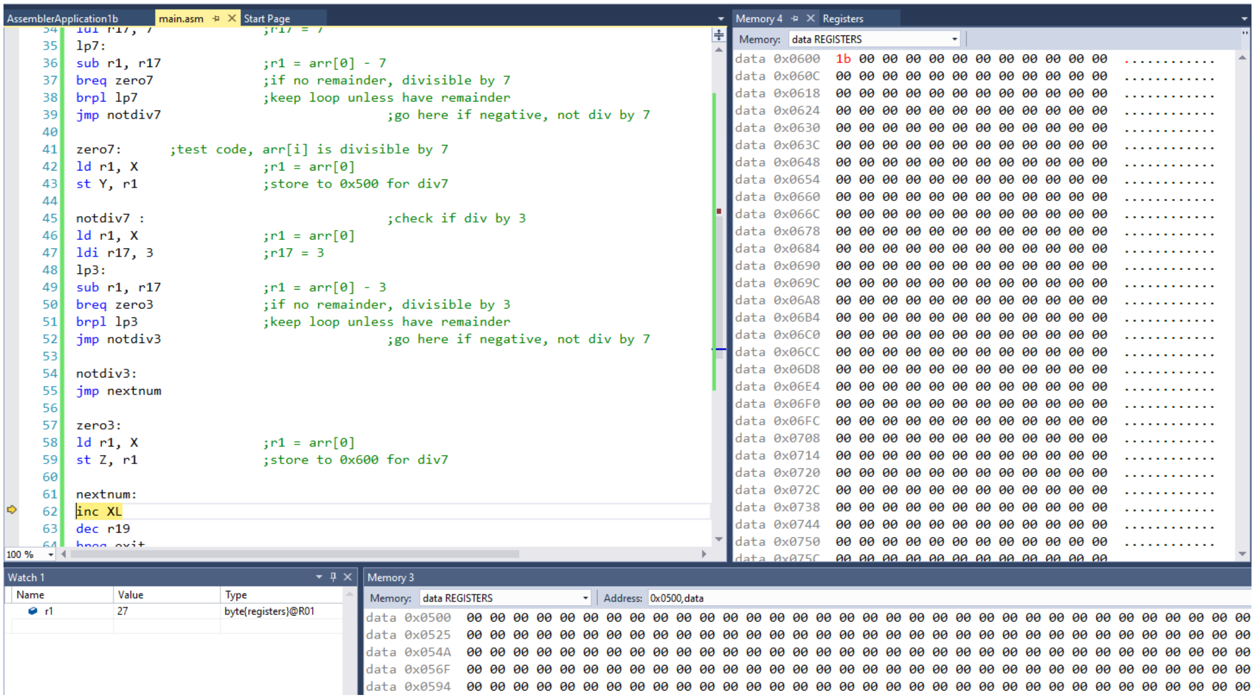
Memory: data REGISTERS

```
data 0x0600  1b 00 00 00 00 00
data 0x0606  00 00 00 00 00 00
data 0x060C  00 00 00 00 00 00
data 0x0612  00 00 00 00 00 00
data 0x0618  00 00 00 00 00 00
data 0x061E  00 00 00 00 00 00
data 0x0624  00 00 00 00 00 00
data 0x062A  00 00 00 00 00 00
data 0x0630  00 00 00 00 00 00
data 0x0636  00 00 00 00 00 00
data 0x063C  00 00 00 00 00 00
data 0x0642  00 00 00 00 00 00
data 0x0648  00 00 00 00 00 00
data 0x064E  00 00 00 00 00 00
data 0x0654  00 00 00 00 00 00
data 0x065A  00 00 00 00 00 00
data 0x0660  00 00 00 00 00 00
data 0x0666  00 00 00 00 00 00
data 0x066C  00 00 00 00 00 00
data 0x0672  00 00 00 00 00 00
data 0x0678  00 00 00 00 00 00
data 0x067E  00 00 00 00 00 00
data 0x0684  00 00 00 00 00 00
data 0x068A  00 00 00 00 00 00
data 0x0690  00 00 00 00 00 00
data 0x0696  00 00 00 00 00 00
data 0x069C  00 00 00 00 00 00
data 0x06A2  00 00 00 00 00 00
data 0x06A8  00 00 00 00 00 00
data 0x06AE  00 00 00 00 00 00
```

| Name | Value | Type |
|---|---|---|
| r1 | 27 | byte{registers}@R01 |
| r20 | 1 | byte{registers}@R20 |
| r19 | 199 | byte{registers}@R19 |

Memory: data REGISTERS Address: 0x0800,data

```
data 0x0800  1a 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
data 0x0825  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
data 0x084A  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```



```
ldi r17, 7                  ;r17 = 7
lp7:
sub r1, r17                 ;r1 = arr[0] - 7
breq zero7                  ;if no remainder, divisible by 7
brpl lp7                    ;keep loop unless have remainder
jmp notdiv7                                 ;go here if negative, not div by 7

zero7:      ;test code, arr[i] is divisible by 7
ld r1, X                    ;r1 = arr[0]
st Y, r1                    ;store to 0x500 for div7

notdiv7 :                                   ;check if div by 3
ld r1, X                    ;r1 = arr[0]
ldi r17, 3                  ;r17 = 3
lp3:
sub r1, r17                 ;r1 = arr[0] - 3
breq zero3                  ;if no remainder, divisible by 3
brpl lp3                    ;keep loop unless have remainder
jmp notdiv3                                 ;go here if negative, not div by 7

notdiv3:
jmp nextnum

zero3:
ld r1, X                    ;r1 = arr[0]
st Z, r1                    ;store to 0x600 for div7

nextnum:
inc XL
dec r19
breq exit
```

Memory: data REGISTERS

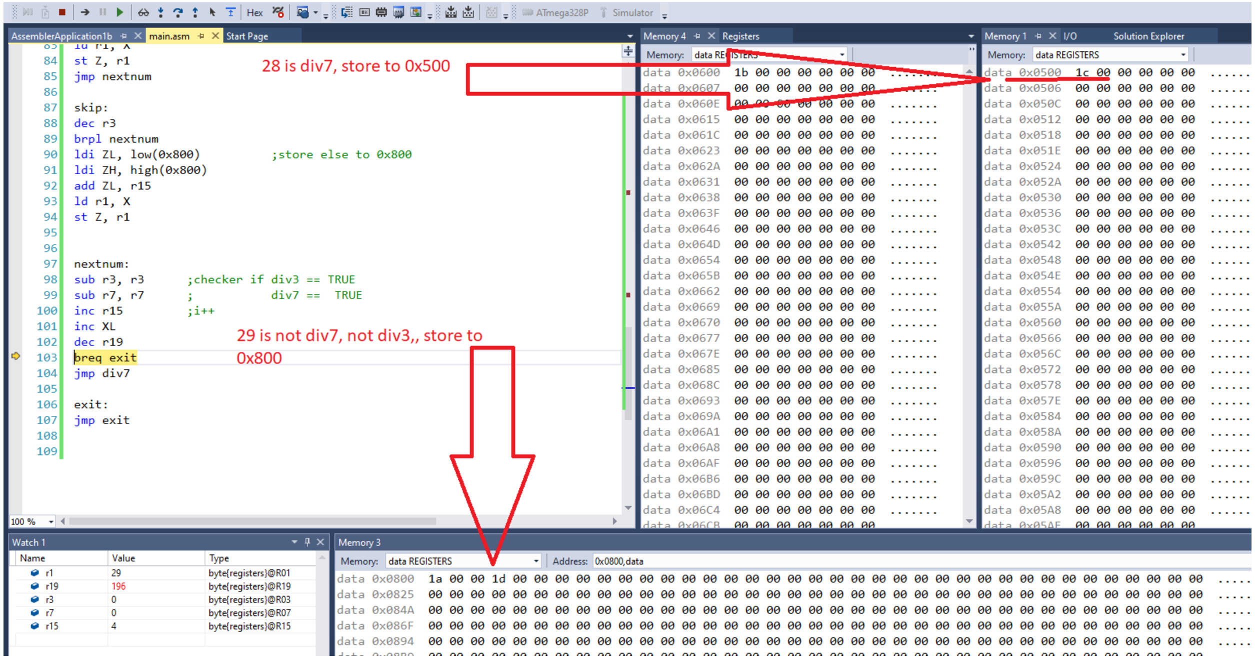
```
data 0x0600  1b 00 00 00 00 00 00 00 00 00 00 00  ............
data 0x060C  00 00 00 00 00 00 00 00 00 00 00 00  ............
data 0x0618  00 00 00 00 00 00 00 00 00 00 00 00  ............
data 0x0624  00 00 00 00 00 00 00 00 00 00 00 00  ............
data 0x0630  00 00 00 00 00 00 00 00 00 00 00 00  ............
data 0x063C  00 00 00 00 00 00 00 00 00 00 00 00  ............
data 0x0648  00 00 00 00 00 00 00 00 00 00 00 00  ............
data 0x0654  00 00 00 00 00 00 00 00 00 00 00 00  ............
data 0x0660  00 00 00 00 00 00 00 00 00 00 00 00  ............
data 0x066C  00 00 00 00 00 00 00 00 00 00 00 00  ............
data 0x0678  00 00 00 00 00 00 00 00 00 00 00 00  ............
data 0x0684  00 00 00 00 00 00 00 00 00 00 00 00  ............
data 0x0690  00 00 00 00 00 00 00 00 00 00 00 00  ............
data 0x069C  00 00 00 00 00 00 00 00 00 00 00 00  ............
data 0x06A8  00 00 00 00 00 00 00 00 00 00 00 00  ............
data 0x06B4  00 00 00 00 00 00 00 00 00 00 00 00  ............
data 0x06C0  00 00 00 00 00 00 00 00 00 00 00 00  ............
data 0x06CC  00 00 00 00 00 00 00 00 00 00 00 00  ............
data 0x06D8  00 00 00 00 00 00 00 00 00 00 00 00  ............
data 0x06E4  00 00 00 00 00 00 00 00 00 00 00 00  ............
data 0x06F0  00 00 00 00 00 00 00 00 00 00 00 00  ............
data 0x06FC  00 00 00 00 00 00 00 00 00 00 00 00  ............
data 0x0708  00 00 00 00 00 00 00 00 00 00 00 00  ............
data 0x0714  00 00 00 00 00 00 00 00 00 00 00 00  ............
data 0x0720  00 00 00 00 00 00 00 00 00 00 00 00  ............
data 0x072C  00 00 00 00 00 00 00 00 00 00 00 00  ............
data 0x0738  00 00 00 00 00 00 00 00 00 00 00 00  ............
data 0x0744  00 00 00 00 00 00 00 00 00 00 00 00  ............
data 0x0750  00 00 00 00 00 00 00 00 00 00 00 00  ............
data 0x075C  00 00 00 00 00 00 00 00 00 00 00 00
```

| Name | Value | Type |
|---|---|---|
| r1 | 27 | byte{registers}@R01 |

Memory: data REGISTERS Address: 0x0500,data

```
data 0x0500  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
data 0x0525  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
data 0x054A  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
data 0x056F  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
data 0x0594  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

number is 27, divisible by 3, store to 0x600



```
ld r1, X
st Z, r1
jmp nextnum

skip:
dec r3
brpl nextnum
ldi ZL, low(0x800)          ;store else to 0x800
ldi ZH, high(0x800)
add ZL, r15
ld r1, X
st Z, r1


nextnum:
sub r3, r3       ;checker if div3 == TRUE
sub r7, r7       ;           div7 ==  TRUE
inc r15          ;i++
inc XL
dec r19
breq exit
jmp div7

exit:
jmp exit
```

28 is div7, store to 0x500

29 is not div7, not div3,, store to 0x800

Memory: data REGISTERS

```
data 0x0600  1b 00 00 00 00 00 00  .......
data 0x0607  00 00 00 00 00 00 00  .......
data 0x060E  00 00 00 00 00 00 00  .......
data 0x0615  00 00 00 00 00 00 00  .......
data 0x061C  00 00 00 00 00 00 00  .......
data 0x0623  00 00 00 00 00 00 00  .......
data 0x062A  00 00 00 00 00 00 00  .......
data 0x0631  00 00 00 00 00 00 00  .......
data 0x0638  00 00 00 00 00 00 00  .......
data 0x063F  00 00 00 00 00 00 00  .......
data 0x0646  00 00 00 00 00 00 00  .......
data 0x064D  00 00 00 00 00 00 00  .......
data 0x0654  00 00 00 00 00 00 00  .......
data 0x065B  00 00 00 00 00 00 00  .......
data 0x0662  00 00 00 00 00 00 00  .......
data 0x0669  00 00 00 00 00 00 00  .......
data 0x0670  00 00 00 00 00 00 00  .......
data 0x0677  00 00 00 00 00 00 00  .......
data 0x067E  00 00 00 00 00 00 00  .......
data 0x0685  00 00 00 00 00 00 00  .......
data 0x068C  00 00 00 00 00 00 00  .......
data 0x0693  00 00 00 00 00 00 00  .......
data 0x069A  00 00 00 00 00 00 00  .......
data 0x06A1  00 00 00 00 00 00 00  .......
data 0x06A8  00 00 00 00 00 00 00  .......
data 0x06AF  00 00 00 00 00 00 00  .......
data 0x06B6  00 00 00 00 00 00 00  .......
data 0x06BD  00 00 00 00 00 00 00  .......
data 0x06C4  00 00 00 00 00 00 00  .......
data 0x06CB  00 00 00 00 00 00 00
```

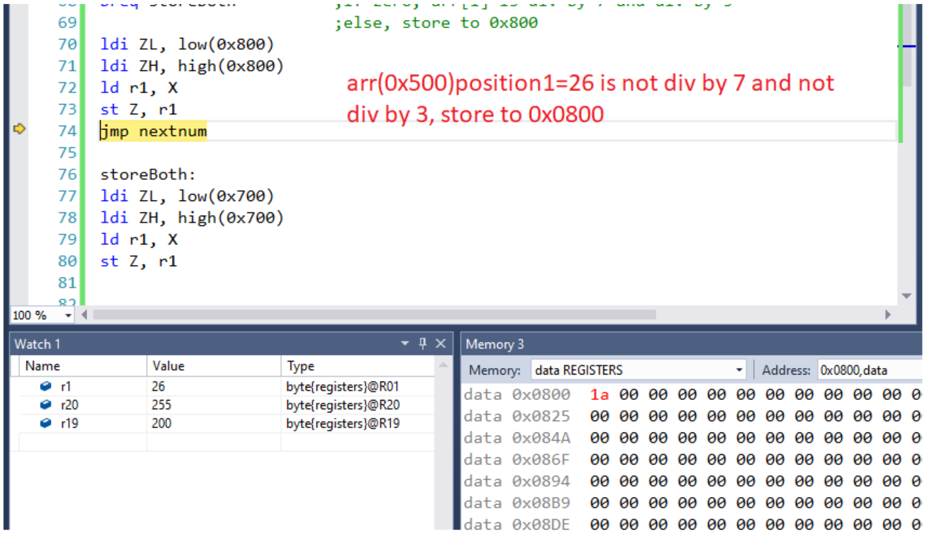
Memory: data REGISTERS

```
data 0x0500  1c 00 00 00 00 00  ......
data 0x0506  00 00 00 00 00 00  ......
data 0x050C  00 00 00 00 00 00  ......
data 0x0512  00 00 00 00 00 00  ......
data 0x0518  00 00 00 00 00 00  ......
data 0x051E  00 00 00 00 00 00  ......
data 0x0524  00 00 00 00 00 00  ......
data 0x052A  00 00 00 00 00 00  ......
data 0x0530  00 00 00 00 00 00  ......
data 0x0536  00 00 00 00 00 00  ......
data 0x053C  00 00 00 00 00 00  ......
data 0x0542  00 00 00 00 00 00  ......
data 0x0548  00 00 00 00 00 00  ......
data 0x054E  00 00 00 00 00 00  ......
data 0x0554  00 00 00 00 00 00  ......
data 0x055A  00 00 00 00 00 00  ......
data 0x0560  00 00 00 00 00 00  ......
data 0x0566  00 00 00 00 00 00  ......
data 0x056C  00 00 00 00 00 00  ......
data 0x0572  00 00 00 00 00 00  ......
data 0x0578  00 00 00 00 00 00  ......
data 0x057E  00 00 00 00 00 00  ......
data 0x0584  00 00 00 00 00 00  ......
data 0x058A  00 00 00 00 00 00  ......
data 0x0590  00 00 00 00 00 00  ......
data 0x0596  00 00 00 00 00 00  ......
data 0x059C  00 00 00 00 00 00  ......
data 0x05A2  00 00 00 00 00 00  ......
data 0x05A8  00 00 00 00 00 00  ......
data 0x05AE  00 00 00 00 00 00
```

| Name | Value | Type |
|---|---|---|
| r1 | 29 | byte{registers}@R01 |
| r19 | 196 | byte{registers}@R19 |
| r3 | 0 | byte{registers}@R03 |
| r7 | 0 | byte{registers}@R07 |
| r15 | 4 | byte{registers}@R15 |

Memory: data REGISTERS    Address: 0x0800,data

```
data 0x0800  1a 00 00 1d 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
data 0x0825  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
data 0x084A  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
data 0x086F  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
data 0x0894  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
data 0x08B9  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```



```
breq storeBoth              ;if zero, arr[i] is div by 7 and div by 3
                            ;else, store to 0x800
ldi ZL, low(0x800)
ldi ZH, high(0x800)
ld r1, X
st Z, r1
jmp nextnum

storeBoth:
ldi ZL, low(0x700)
ldi ZH, high(0x700)
ld r1, X
st Z, r1
```

arr(0x500)position1=26 is not div by 7 and not div by 3, store to 0x0800

| Name | Value | Type |
|---|---|---|
| r1 | 26 | byte{registers}@R01 |
| r20 | 255 | byte{registers}@R20 |
| r19 | 200 | byte{registers}@R19 |

Memory: data REGISTERS    Address: 0x0800,data

```
data 0x0800  1a 00 00 00 00 00 00 00 00 00 00 00 0
data 0x0825  00 00 00 00 00 00 00 00 00 00 00 00 0
data 0x084A  00 00 00 00 00 00 00 00 00 00 00 00 0
data 0x086F  00 00 00 00 00 00 00 00 00 00 00 00 0
data 0x0894  00 00 00 00 00 00 00 00 00 00 00 00 0
data 0x08B9  00 00 00 00 00 00 00 00 00 00 00 00 0
data 0x08DE  00 00 00 00 00 00 00 00 00 00 00 00 0
```

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

"*This assignment submission is my own, original work*".
dan ray Dimapilis

**all works in this document and in this repository are for unlv homeworks only, spring 2020**