Lab: 1
Nicholas Imamshah
Section: 6957

b) Prelab Questions

1. What instruction can be used to read from program memory (flash)? Can you use any registers with this instruction?
    a. lpm
    b. This instruction can only use the Z register (not X and Y), because it is the only register that can access the Flash Program Memory.
2. What is the difference between program and data memory?
    a. The primary difference is that the program memory is the reprogrammable flash memory for program storage, whereas the data memory is where the internal SRAM resides which allows for writing out of data.
3. When using RAM (not EEPROM), what memory locations can be utilized for the .DSEG? Why? What .DSEG did you use in this lab and why?
    a. 0x2000 – 0x3FFF. These addresses are the size of the internal SRAM. In this lab we used 0x2016 for the .DSEG, which works because it is (a) within the SRAM range and (b) the current year.

c) Problems Encountered

It took me some time to fully understand how to calculate the address for data in program memory, i.e., placing the table in memory.

d) Future Work/Applications

This lab was a very general introduction to programming microprocessors, which lends itself to any myriad of applications; interfacing with sound and light systems, controlling mechanical systems in household and commercial environments.

Lab: 1
Nicholas Imamshah
Section: 6957

e) Schematics

N/A

f) Decoding Logic

N/A

g) Pseudocode/Flowcharts

```
initialize pointers
load value from Table
compare with NUL, if equal -> program complete
compare with lower bound, if greater than or equal -> continue
compare with upper bound, NOT less than -> restart loop

if value within bounds:
    XOR w/ 0x37
    Store result in Output Table
    Restart loop
```

h) Program Code

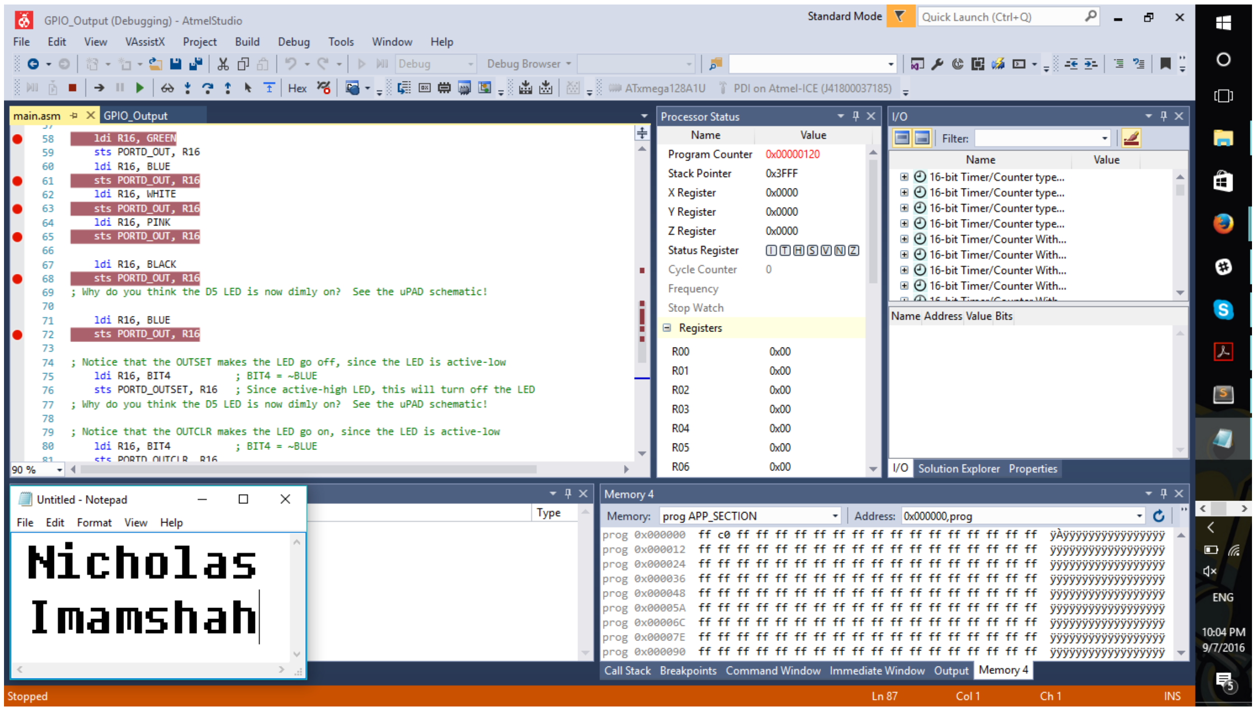
I just realized as I'm typing this that I could have written the section that branches to CHECK_H more efficiently, oh well

```
/*
 * lab1.asm
 *
 * Lab 1 Part B
 * Name: Nicholas Imamshah
 * Section: 6957
 * TA Name: Daniel Gonzalez
 * Description: The purpose of this program is to filter data from an array in memory.
 */

.nolist     ; This works, but the below file can't be removed for lss file.
.include "ATxmega128A1Udef.inc"
.list

.equ u_bnd = 0x80
.equ l_bnd = 0x16
.equ NUL = 0x00

.org 0x0000
        rjmp MAIN

.org 0x1BA2
```

```
Table: .db 0xA4, 0x70, 0x81, 0x58, 0x58, 0x53, 0x96, 0x17, 0x5D, 0xEE, 0x58, 0xF1, 0x83, 0xDB, 0x55, 0x99, 0x16, 0xC2, 0xD7, 0xF5,
0x00

.dseg
.org 0x2016
OutTable: .byte 256

.cseg
.org 0x200
MAIN:
        ldi ZL, low(Table << 1)         ;load the low byte of the Table address into ZL register
        ldi ZH, high(Table << 1)        ;load the high byte of the Table address into ZH register

        ldi YL, low(OutTable)           ;load the low byte of the OutTable address into YL register
        ldi YH, high(OutTable)          ;load the high byte of the OutTable address into YH register

        ldi r17, 0x37                           ;load 0x37 for XORing later

LOOP:
        lpm r16, Z+             ;load value from table and Post-Increment Z

        cpi r16, NUL    ;check if we've hit the NUL value (i.e., end of table marker)
        breq DONE               ;if we've hit NUL, we're DONE

        cpi r16, l_bnd  ;compare value with the lower bound
        brsh CHECK_H    ;if >=l_bnd (unsigned), check upper bound

CHECK_H:
        cpi r16, u_bnd  ;compare value with the upper bound
        brsh LOOP               ;if >=u_bnd (unsigned), conditions not met, return to LOOP

        eor r16, r17    ;XOR value meeting conditions with 0x37
        st Y+, r16      ;store value meeting conditions to OutTable
        rjmp LOOP               ;continue LOOP

DONE:
        rjmp DONE       ;infinite loop because program has completed.
```

i) Appendix

Lab: 1
Nicholas Imamshah
Section: 6957


GPIO_Output (Debugging) - AtmelStudio

main.asm

```
    ldi R16, GREEN
    sts PORTD_OUT, R16
    ldi R16, BLUE
    sts PORTD_OUT, R16
    ldi R16, WHITE
    sts PORTD_OUT, R16
    ldi R16, PINK
    sts PORTD_OUT, R16

    ldi R16, BLACK
    sts PORTD_OUT, R16
; Why do you think the D5 LED is now dimly on?  See the uPAD schematic!

    ldi R16, BLUE
    sts PORTD_OUT, R16

; Notice that the OUTSET makes the LED go off, since the LED is active-low
    ldi R16, BIT4           ; BIT4 = ~BLUE
    sts PORTD_OUTSET, R16   ; Since active-high LED, this will turn off the LED
; Why do you think the D5 LED is now dimly on?  See the uPAD schematic!

; Notice that the OUTCLR makes the LED go on, since the LED is active-low
    ldi R16, BIT4           ; BIT4 = ~BLUE
    sts PORTD_OUTCLR, R16
```

Processor Status

Name	Value
Program Counter	0x00000120
Stack Pointer	0x3FFF
X Register	0x0000
Y Register	0x0000
Z Register	0x0000
Status Register	ITHSVNZ
Cycle Counter	0
Frequency	
Stop Watch	
Registers	
R00	0x00
R01	0x00
R02	0x00
R03	0x00
R04	0x00
R05	0x00
R06	0x00

Untitled - Notepad

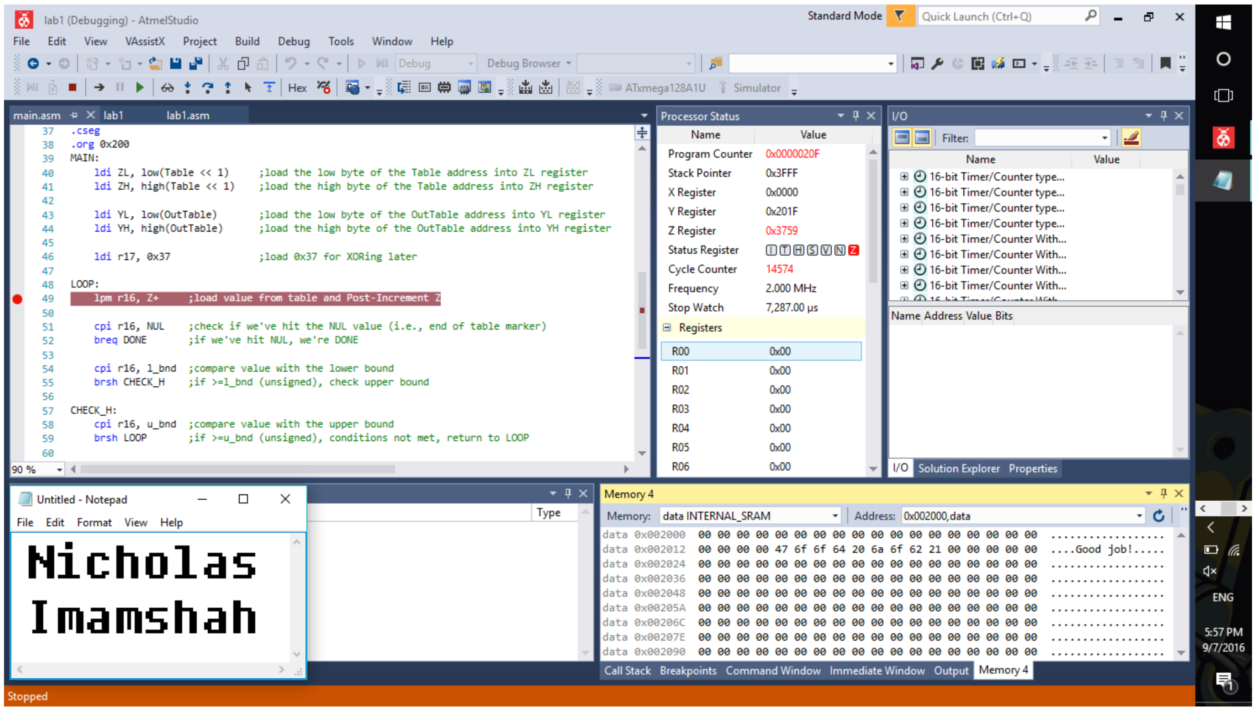
Nicholas
Imamshah

Memory 4 — Memory: prog APP_SECTION   Address: 0x000000,prog

```
prog 0x000000  ff c0 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÀÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x000012  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x000024  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x000036  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x000048  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x00005A  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x00006C  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x00007E  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x000090  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
```

Stopped   Ln 87   Col 1   Ch 1   INS

Atmel Tutorial


lab1 (Debugging) - AtmelStudio

lab1.asm

```
.cseg
.org 0x200
MAIN:
    ldi ZL, low(Table << 1)     ;load the low byte of the Table address into ZL register
    ldi ZH, high(Table << 1)    ;load the high byte of the Table address into ZH register

    ldi YL, low(OutTable)       ;load the low byte of the OutTable address into YL register
    ldi YH, high(OutTable)      ;load the high byte of the OutTable address into YH register

    ldi r17, 0x37               ;load 0x37 for XORing later

LOOP:
    lpm r16, Z+     ;load value from table and Post-Increment Z

    cpi r16, NUL    ;check if we've hit the NUL value (i.e., end of table marker)
    breq DONE       ;if we've hit NUL, we're DONE

    cpi r16, l_bnd  ;compare value with the lower bound
    brsh CHECK_H    ;if >=l_bnd (unsigned), check upper bound

CHECK_H:
    cpi r16, u_bnd  ;compare value with the upper bound
    brsh LOOP       ;if >=u_bnd (unsigned), conditions not met, return to LOOP

```

Processor Status

Name	Value
Program Counter	0x0000020F
Stack Pointer	0x3FFF
X Register	0x0000
Y Register	0x201F
Z Register	0x3759
Status Register	ITHSVNZ
Cycle Counter	14574
Frequency	2.000 MHz
Stop Watch	7,287.00 µs
Registers	
R00	0x00
R01	0x00
R02	0x00
R03	0x00
R04	0x00
R05	0x00
R06	0x00

Untitled - Notepad

Nicholas
Imamshah

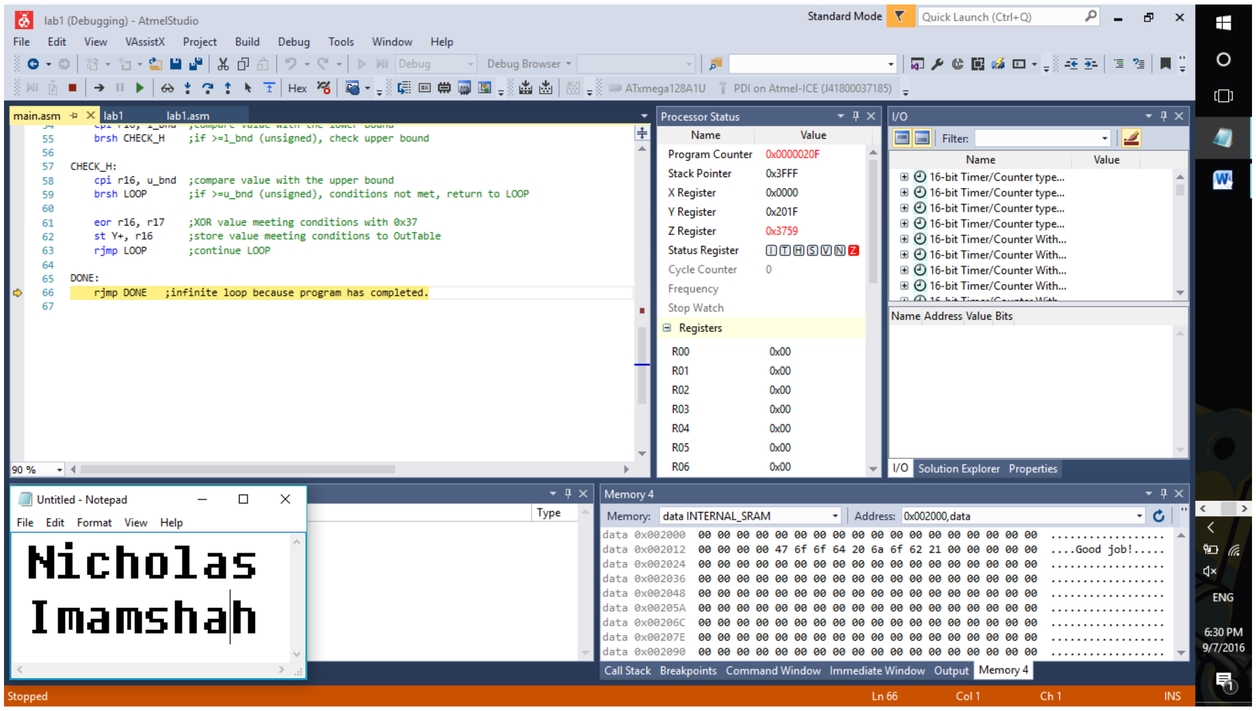
Memory 4 — Memory: data INTERNAL_SRAM   Address: 0x002000,data

```
data 0x002000  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ..................
data 0x002012  00 00 00 00 47 6f 6f 64 20 6a 6f 62 21 00 00 00 00 00  ....Good job!.....
data 0x002024  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ..................
data 0x002036  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ..................
data 0x002048  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ..................
data 0x00205A  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ..................
data 0x00206C  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ..................
data 0x00207E  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ..................
data 0x002090  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ..................
```

Stopped

Simulator

Lab: 1
Nicholas Imamshah
Section: 6957



```
	brsh CHECK_H	;if >=l_bnd (unsigned), check upper bound

CHECK_H:
	cpi r16, u_bnd	;compare value with the upper bound
	brsh LOOP	;if >=u_bnd (unsigned), conditions not met, return to LOOP

	eor r16, r17	;XOR value meeting conditions with 0x37
	st Y+, r16	;store value meeting conditions to OutTable
	rjmp LOOP	;continue LOOP

DONE:
	rjmp DONE	;infinite loop because program has completed.
```

Name	Value
Program Counter	0x0000020F
Stack Pointer	0x3FFF
X Register	0x0000
Y Register	0x201F
Z Register	0x3759
Status Register	ITHSVNZ
Cycle Counter	0
Frequency	
Stop Watch	
Registers	
R00	0x00
R01	0x00
R02	0x00
R03	0x00
R04	0x00
R05	0x00
R06	0x00

Memory: data INTERNAL_SRAM Address: 0x002000,data

```
data 0x002000  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ..................
data 0x002012  00 00 00 00 47 6f 6f 64 20 6a 6f 62 21 00 00 00 00 00  ....Good job!.....
data 0x002024  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ..................
data 0x002036  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ..................
data 0x002048  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ..................
data 0x00205A  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ..................
data 0x00206C  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ..................
data 0x00207E  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ..................
data 0x002090  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ..................
```

Nicholas
Imamshah

Stopped Ln 66 Col 1 Ch 1 INS

Atmel ICE