**b) Prelab Questions**

1. What instruction can be used to read from program memory (flash)? Can you use any registers with this instruction?
   1. lpm
   2. 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?
   1. 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?
   1. 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.

**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

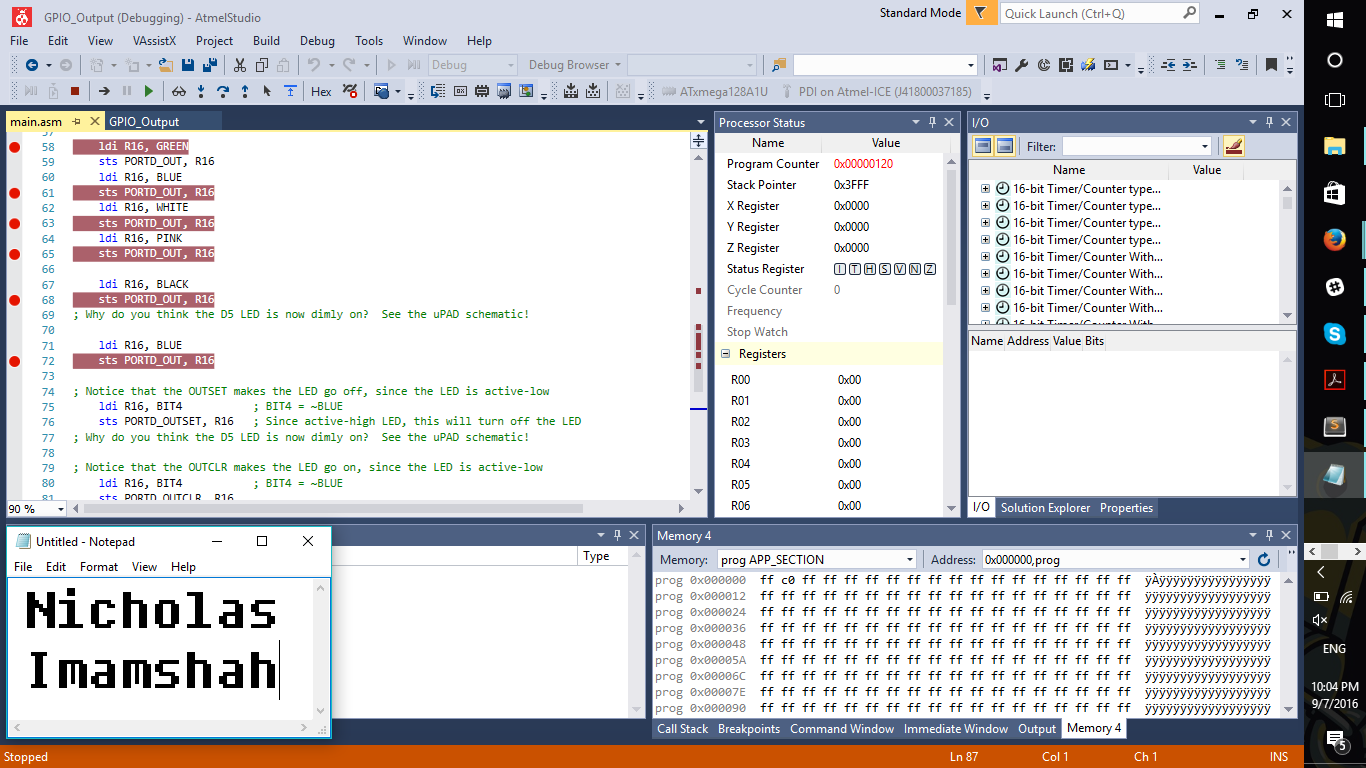
st Y+, r16 ;store value meeting conditions to OutTable

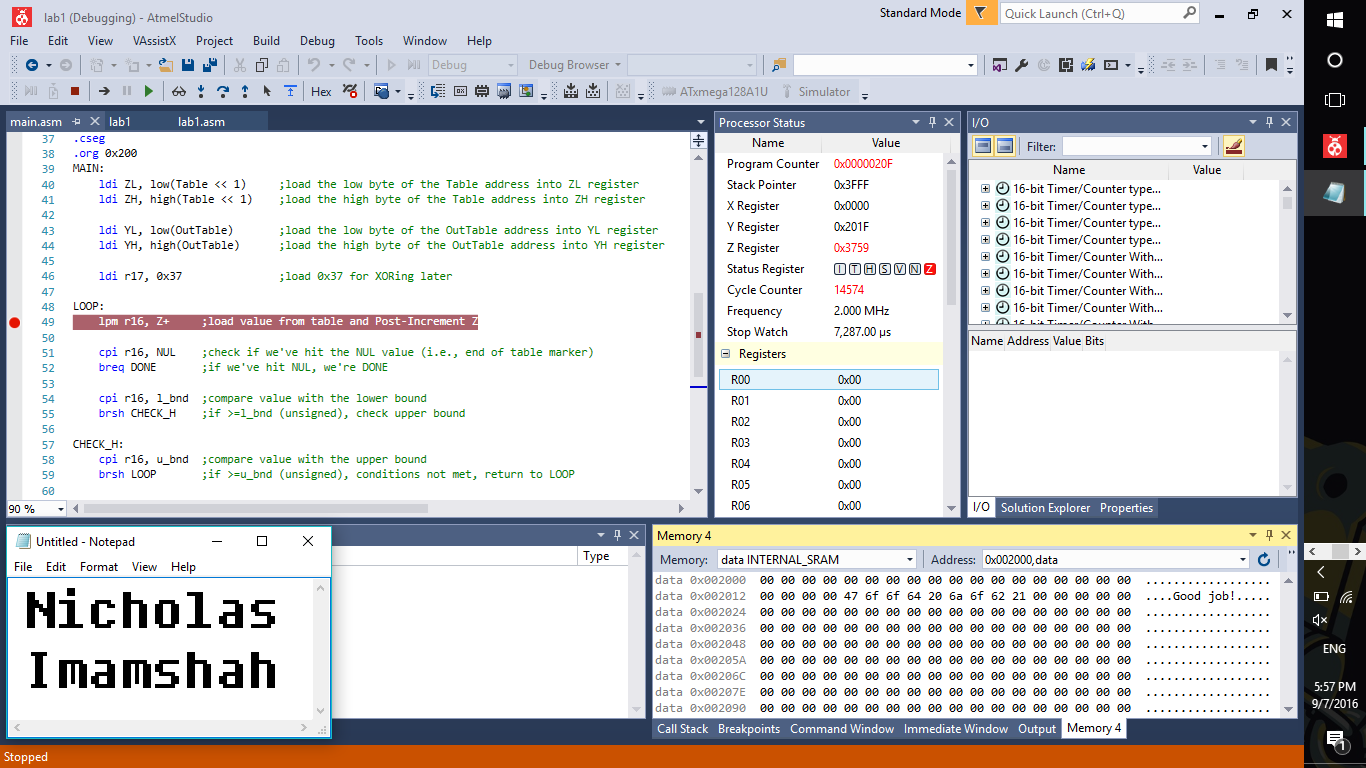
rjmp LOOP ;continue LOOP

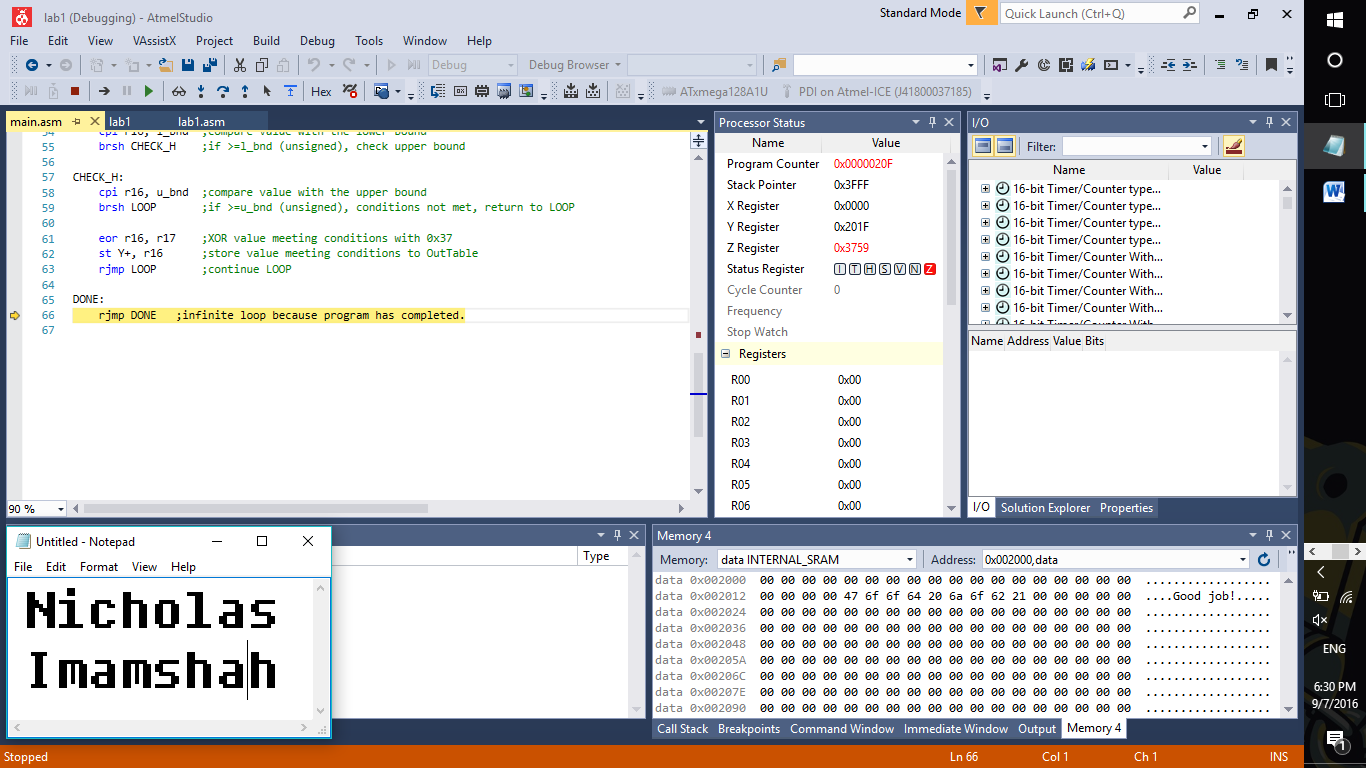
DONE:

rjmp DONE ;infinite loop because program has completed.

**i) Appendix**

  
Atmel Tutorial

  
Simulator

  
Atmel ICE