Lab 0

Michael Arboleda
Lab Section: 7F34

May 17, 2017

b. Answers to all pre-lab questions

1) **What minimum lab average is required in order to be eligible to pass the course?**

   ANS: 65%

2) **Can you drop this lab if ... a) you overslept? b) project for other class due?**

   ANS: Yes, Yes

3) **How late can you arrive for lab and still be admitted? How late can you arrive for lab and still be allowed to take the lab quiz?**

   ANS: You can be up to 30 minutes late and still be admitted. You can be up to 10 minutes late and still take the quiz.

4) **What is the lab makeup policy if you miss a single lab?**

   Ans: If you miss a single-lab, you cannot make it up

5) **When soldering a wire to a pin, what should the soldering iron touch and what should the un-melted solder touch?**

   ANS: The soldering iron should touch the pin. The un-melted solder should touch the wire.

6) **What instruction can be used to read from program memory (flash)? Can you use any registers with this instruction?**

   ANS: LPM (and ELPM for extended address) is used to read from program memory. You cannot use any register, you must use the Z register

7) **What are the key differences between program and data memory? See section 7 in the ATxmega128A1U manual**

   ANS: Program memory can hold executable code. Data Memory cannot hold executable code. Data memory also includes the internal SRAM and EEPROM for nonvolatile data storage.

8) **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?**

   ANS: Memory address 0x2000 and above are for the ram. This is because 0x1000 to 0x1FFF is the EEPROM. The .DESG used in the lab is 0x3744. This was because it was in the directions of the lab document.

c. Problems Encountered

At first I did not understand the data reading. This was clarified in lecture when Dr. Schwartz went over the CPU_RAMPZ register

d. Future Work/Applications

After doing this lab there is clearly so much to learn. There are many other instructions I could have used instead of the ones I used. I could try using those instructions instead so I can understand them and not be stuck knowing only a small subset of all the commands

e. Schematics

N/A (No hardware to change)

g. Pseudocode/Flowcharts

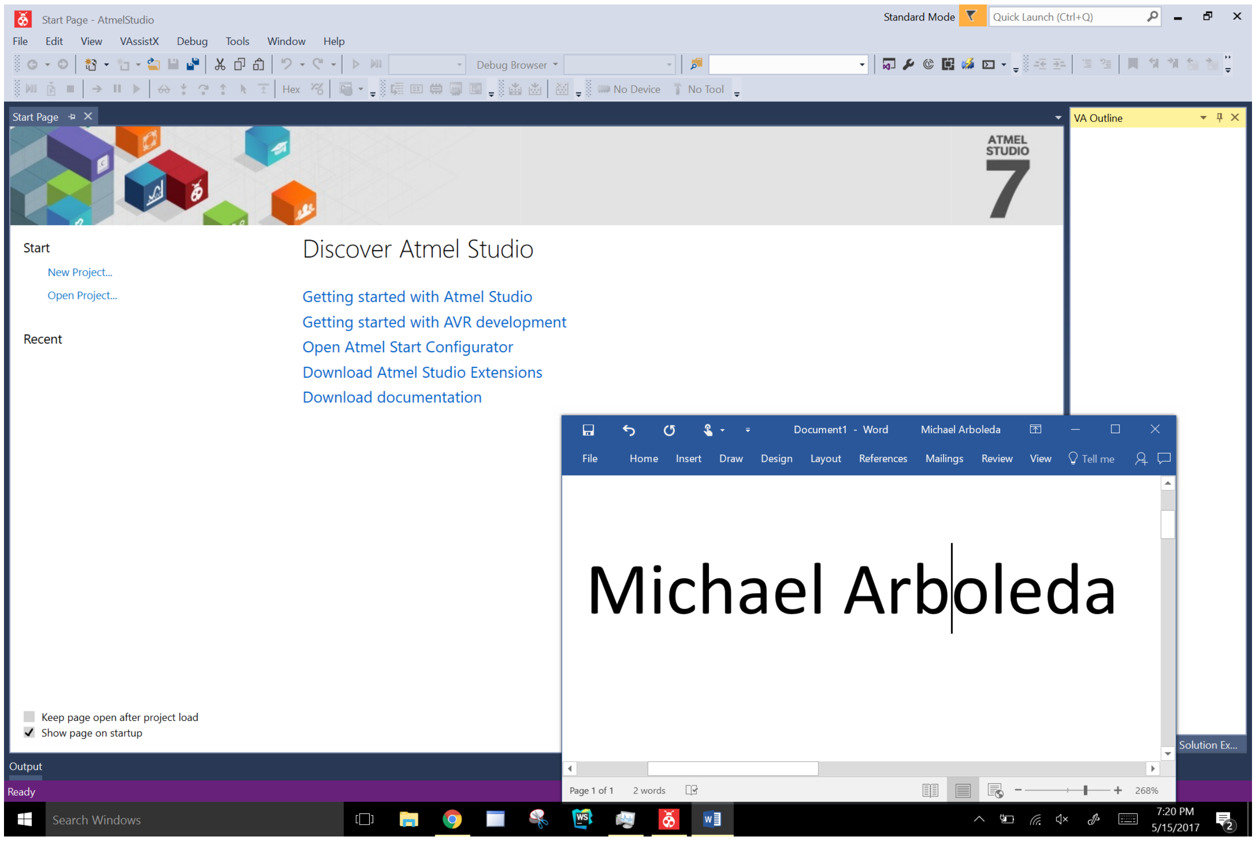
Pseudocode for lab0.asm:

```
* Create array pointer
* Create pointer for storing

DO-LOOP
    * Get Data From Array
    * Increment array pointer

    IF(DATA LESS THAN 166){
        IF(DATA GRATER THAN 36){
            * Add data and 0x11
        }
        * Store value
    }
WHILE(DATA is not 0)
```

h. Program Code

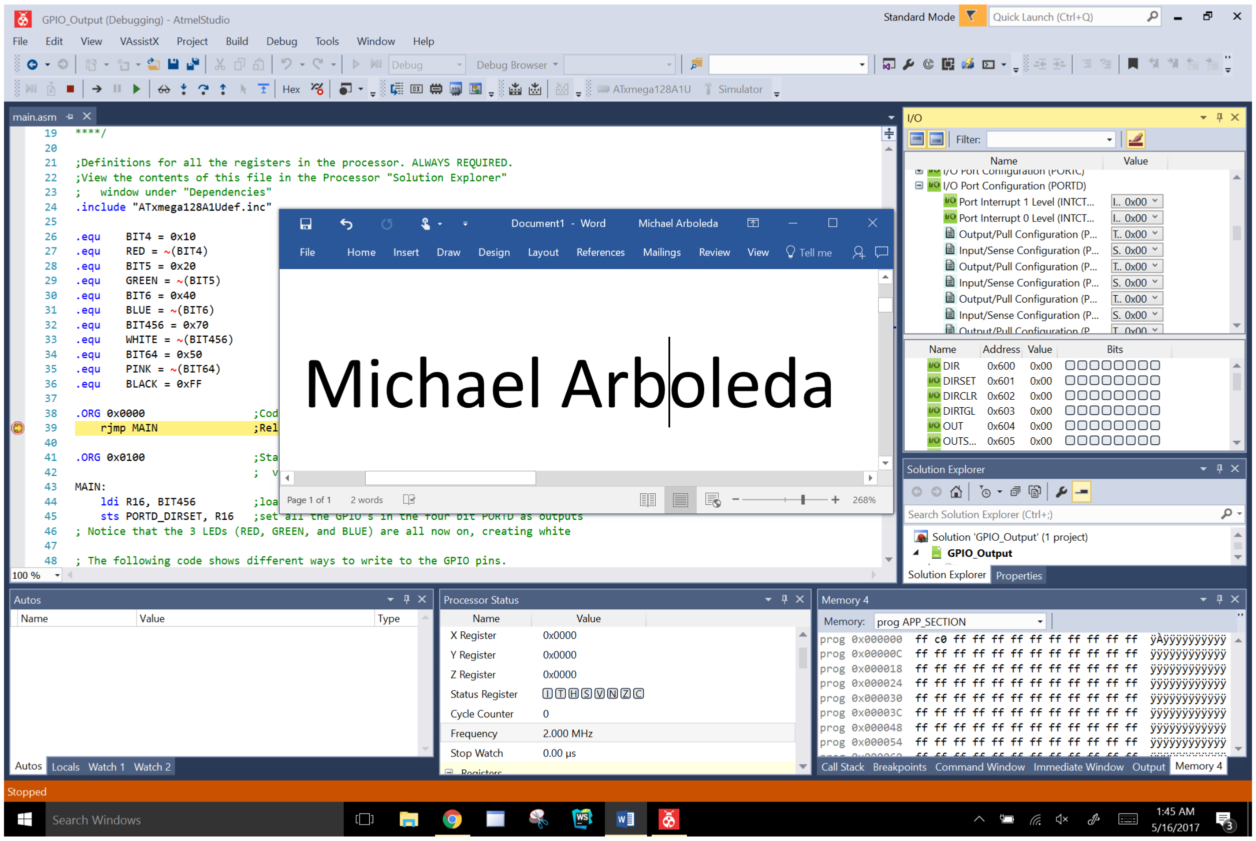
```
; Lab 0 Part C
; Name:             Michael Arboleda
; Section:          7F34
; TA Name:          Wesley Piard
; Description: This code filters data based on ASCII Value
;
; lab0.asm
; Created: 5/16/2017 4:14:49 PM
;
.include "ATxmega128A1Udef.inc"

.equ outputTable = 0x3744
.equ inputTable  = 0xF000
.equ upperBound  = 0166
.equ lowerBound  = 38
.equ addition    = 0x11

.ORG 0x0000                     ;Code starts running from address 0x0000.
        rjmp MAIN               ;Relative jump to start of program.

.CSEG                           ;Code segment start
        .ORG 0xF000             ;Start table address

Input_Table:
        .db 0x3d, 0x7F, 84, 102, 0x7B, 0172, 0x20, 0x64,\
        0x7E, 0x3F, 060, 0x33, 0x7B, 121, 118, 0x21, 0x78,\
        0x77, 0 ; Table Values

.DSEG                           ;Code segment stop

.ORG 0x3744                     ;Start output table address
Output_Table:
        .byte 18;

.CSEG
.ORG 0x0100             ;Start program at 0x0100 so we don't overwrite
                        ;Vectors that are at 0x0000-0x00FD

MAIN:
        ;Set up Y-register
        ldi YL, low(outputTable);LOAD address to write to - low 2 bits
        ldi YH, high(outputTable);LOAD address to write to - high 2 bits

        ;Set up z-register
        ldi R16, 1                      ;LOAD 1 to R16
        sts CPU_RAMPZ, R16              ;STORING Extended Address
        ldi ZL, low(inputTable << 1);LOAD adrs to read from-low 2 bits
        ldi ZH, high(inputTable << 1);LOAD adrs to read from-high 2 bits

        ;set up regs to hold commonly used values
        ldi R16, lowerBound             ;LOAD 38 (for comparision)
        ldi R17, upperBound             ;LOAD 166 (for comparision)
        ldi R18, addition               ;LOAD 0x11 (for addition)

;START DO W?HILE LOOP
DO:                             ;Top of do-while loop
        elpm R19, z+            ;LOAD data from array
                                ;Post increment z-reg
;IF
        ;if greater than 165, it fails
        cp R19, R17             ;COMPARE R17 with R19
        BRGE ENDIF              ;BRANCH if R19 >= 0166

;INNER IF
;THIS part corrisponds with an nested if statement
        ;If less than 38, if fails
        cp R19, R16             ;COMPARE R16 with R19
        BRLT END_INNER_IF       ;BRANCH if R19 < R17 (37)

        ;add 0x11 to data
        add R19, R18            ;Add R19:= R19 + R18 (add 0x11)

END_INNER_IF:
        ;Store Value to table
        st  y+, R19     ;Store new data to proper location
                        ;increment location(store) pointer
;WHILE part of DO-WHILE loop
ENDIF:
        cpi R19, 0      ;Compare R19 with #0
        BRNE DO         ;BRANCH to DO if R19 does not equal 0

        ;Stores 0 at end
        st  y, R19      ;Store new data to proper location
                        ;HI This will always store 0
```

i. Appendix


Start Page - AtmelStudio
File Edit View VAssistX Debug Tools Window Help
Start Page
Start
New Project...
Open Project...
Recent
Discover Atmel Studio
Getting started with Atmel Studio
Getting started with AVR development
Open Atmel Start Configurator
Download Atmel Studio Extensions
Download documentation
Keep page open after project load
Show page on startup
Document1 - Word
Michael Arboleda
Page 1 of 1 2 words
7:20 PM
5/15/2017

Michael Arboleda

Figure 1: PART A. ATMEL Installation


GPIO_Output (Debugging) - AtmelStudio
File Edit View VAssistX Project Build Debug Tools Window Help
Standard Mode
Quick Launch (Ctrl+Q)
Debug
Debug Browser
Hex
ATxmega128A1U
Simulator
main.asm
Document1 - Word
File Home Insert Draw Design Layout References Mailings Review View Tell me
Page 1 of 1
2 words
268%
100 %
Solution Explorer
Properties
Search Solution Explorer (Ctrl+;)
Autos Locals Watch 1 Watch 2
Call Stack Breakpoints Command Window Immediate Window Output Memory 4
Stopped
Search Windows
1:45 AM
5/16/2017

```
****/

;Definitions for all the registers in the processor. ALWAYS REQUIRED.
;View the contents of this file in the Processor "Solution Explorer"
;   window under "Dependencies"
.include "ATxmega128A1Udef.inc"

.equ    BIT4 = 0x10
.equ    RED = ~(BIT4)
.equ    BIT5 = 0x20
.equ    GREEN = ~(BIT5)
.equ    BIT6 = 0x40
.equ    BLUE = ~(BIT6)
.equ    BIT456 = 0x70
.equ    WHITE = ~(BIT456)
.equ    BIT64 = 0x50
.equ    PINK = ~(BIT64)
.equ    BLACK = 0xFF

.ORG 0x0000                     ;Cod
    rjmp MAIN                   ;Rel

.ORG 0x0100                     ;Sta
                                ;  v
MAIN:
    ldi R16, BIT456             ;loa
    sts PORTD_DIRSET, R16       ;set all the GPIO's in the four bit PORTD as outputs
; Notice that the 3 LEDs (RED, GREEN, and BLUE) are all now on, creating white

; The following code shows different ways to write to the GPIO pins.
```

Michael Arboleda

I/O

Name	Value
I/O Port Configuration (PORTC)	
I/O Port Configuration (PORTD)	
Port Interrupt 1 Level (INTCT...	I.. 0x00
Port Interrupt 0 Level (INTCT...	I.. 0x00
Output/Pull Configuration (P...	T.. 0x00
Input/Sense Configuration (P...	S. 0x00
Output/Pull Configuration (P...	T.. 0x00
Input/Sense Configuration (P...	S. 0x00
Output/Pull Configuration (P...	T.. 0x00
Input/Sense Configuration (P...	S. 0x00
Output/Pull Configuration (P...	T.. 0x00

Name	Address	Value	Bits
DIR	0x600	0x00	☐☐☐☐☐☐☐☐
DIRSET	0x601	0x00	☐☐☐☐☐☐☐☐
DIRCLR	0x602	0x00	☐☐☐☐☐☐☐☐
DIRTGL	0x603	0x00	☐☐☐☐☐☐☐☐
OUT	0x604	0x00	☐☐☐☐☐☐☐☐
OUTS...	0x605	0x00	☐☐☐☐☐☐☐☐

Solution 'GPIO_Output' (1 project)
GPIO_Output

Autos

Name	Value	Type

Processor Status

Name	Value
X Register	0x0000
Y Register	0x0000
Z Register	0x0000
Status Register	I T H S V N Z C
Cycle Counter	0
Frequency	2.000 MHz
Stop Watch	0.00 µs
Registers	

Memory 4

Memory: prog APP_SECTION

```
prog 0x000000  ff c0 ff ff ff ff ff ff ff ff ff ff  ÿÀÿÿÿÿÿÿÿÿÿÿ
prog 0x00000C  ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x000018  ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x000024  ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x000030  ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x00003C  ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x000048  ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x000054  ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿ
```

Figure 2: PART B. ATMEL Tutorial