Brandon Pollack

Lab 6

Report

Here is all of the code from this lab

Part 1:

/\* A collection of inits for the EBI and stack init at the bottom \*/

.equ IOPORT = 0x5000

.equ SRAMPORT = 0x370000

.equ LCDPORT\_COM = 0x4000

.equ LCDPORT\_DAT = 0x4001

.macro TRIPORT\_ALE\_ONE\_INIT

ldi R16, 0b01110111

sts PORTH\_DIR, R16 //set port pins as outputs for RE and ALE and WE CS1 and CS0

ldi R16, 0b01110011

sts PORTH\_OUT, R16 //WE and RE is active low so it must be set

ldi R16, 0xFF

sts PORTJ\_DIR, R16 //set datalines as outputs (manual says so)

sts PORTK\_DIR, R16 //set address lines as outputs

ldi R16, 0x01

sts EBI\_CTRL, R16 //turn on 3 port SRAM ALE1 EBI

.endmacro

.macro CS0\_INIT

ldi ZH, HIGH(EBI\_CS0\_BASEADDR) //all the set up for CS0, since EBI won't work without it

ldi ZL, LOW(EBI\_CS0\_BASEADDR)

ldi R16, ((IOPORT>>8) & 0xF0)

st Z+, R16

ldi R16, ((IOPORT>>16) & 0xFF)

st Z, R16

ldi R16, 0x11

sts EBI\_CS0\_CTRLA, R16

.endmacro

.macro CS1\_INIT

ldi ZH, HIGH(EBI\_CS1\_BASEADDR) //set up CS1 for the SRAM

ldi ZL, LOW(EBI\_CS1\_BASEADDR)

ldi R16, ((SRAMPORT>>8) & 0xF0)

st Z+, R16

ldi R16, ((SRAMPORT>>16) & 0xFF)

st Z, R16

ldi R16, 0b00011101

sts EBI\_CS1\_CTRLA, R16

.endmacro

.macro CS2\_INIT

ldi ZH, HIGH(EBI\_CS2\_BASEADDR) //set up CS1 for the SRAM

ldi ZL, LOW(EBI\_CS2\_BASEADDR)

ldi R16, ((LCDPORT\_COM>>8) & 0xF0)

st Z+, R16

ldi R16, ((LCDPORT\_COM>>16) & 0xFF)

st Z, R16

ldi R16, 0x01

sts EBI\_CS2\_CTRLA, R16

.endmacro

.macro STACK\_INIT

ldi R16, 0xFF

out CPU\_SPL, R16

ldi R16, 0x3F

out CPU\_SPH, R16 //init stack pointer

.endmacro

.macro LCD\_INIT

ldi XH, high(LCDPORT\_COM)

ldi XL, low(LCDPORT\_COM)

call LCD\_BF\_WAIT

ldi R16, 0b00111000 // two lines, bigger font, 8 bits

st X, R16

call LCD\_BF\_WAIT

ldi R16, 0b00001111 // display on cursor on curor blink

st X, R16

call LCD\_BF\_WAIT

ldi R16, 0b00000001 // clear disp

st X, R16

call LCD\_BF\_WAIT

ldi R16, 0b00000011 // cursor home

st X, R16

.endmacro

/\*

\* Lab6.asm

\*

\* Created: 3/21/2013 6:52:29 PM

\* Author: Brandon

\*/

.include "Atxmega128A1udef.inc"

.include "EBI\_INITS.asm"

.equ NameLocation = 0x1000

.org 0x0

rjmp main

.org NameLocation

.db "Brandon Pollack", 0

.org 0x100

main:

STACK\_INIT

TRIPORT\_ALE\_ONE\_INIT

CS0\_INIT

CS1\_INIT

CS2\_INIT

jumphere:

ldi XH, high(LCDPORT\_COM)

ldi XL, low(LCDPORT\_COM)

call LCD\_BF\_WAIT

ldi R16, 0b00111000 // two lines, bigger font, 8 bits

st X, R16

call LCD\_BF\_WAIT

ldi R16, 0b00001111 // display on cursor on curor blink

st X, R16

call LCD\_BF\_WAIT

ldi R16, 0b00000001 // clear disp

st X, R16

call LCD\_BF\_WAIT

ldi R16, 0b00000011 // cursor home

st X, R16

ldi ZH, high(NameLocation << 1)

ldi ZL, low(NameLocation << 1)

call OUT\_STRING\_LCD

done:

rjmp done;

OUT\_CHAR\_LCD: //outs R16 to LCD

call LCD\_BF\_WAIT

ldi XH, high(LCDPORT\_DAT)

ldi XL, low(LCDPORT\_DAT)

st X, R16

ret

OUT\_STRING\_LCD: //put address of string in Z register

push R16

stringloop:

lpm R16, Z+

cpi R16, 0

breq string\_done

call OUT\_CHAR\_LCD

rjmp stringloop

string\_done:

pop R16

ret

LCD\_BF\_WAIT:

/\*push R0

clr R0

ldi XH, high(LCDPORT\_COM)

ldi XL, low(LCDPORT\_COM)

notready:

ld R0, X

sbrc R0, 7

rjmp notready

pop R0

ret\*/

push R16

push r17

ldi R16, 0

ldi R17, 0

AGAIN:

NOP

NOP

NOP

NOP

NOP

NOP

NOP

NOP

NOP

NOP

NOP

NOP

NOP

NOP

NOP

INC R16

CPI R16, 0

BREQ CARRY

BACK:

CPI R17, 0x01

BRNE AGAIN

BREQ RETURN

CARRY:

INC R17

rjmp BACK

RETURN:

pop r17

pop r16

RET

Part 2:

/\*

\* Lab6\_Part2.asm

\*

\* Created: 3/24/2013 4:40:51 PM

\* Author: Brandon

\*/

.include "Atxmega128A1udef.inc"

.include "EBI\_INITS.asm"

.equ LUT = 0x1000

.org 0x0

rjmp main

.org LUT

.db "0.00 0.04 0.08 0.12 0.16 0.20 0.24 0.28 0.31 0.35 0.39 0.43 0.47 0.51 0.55 0.59 0.63 0.67 0.71 0.75 0.79 0.83 0.87 0.91 0.94 0.98 1.02 1.06 1.10 1.14 1.18 1.22 1.26 1.30 1.34 1.38 1.42 1.46 1.50 1.54 1.57 1.61 1.65 1.69 1.73 1.77 1.81 1.85 1.89 1.93 1.97 2.01 2.05 2.09 2.13 2.17 2.20 2.24 2.28 2.32 2.36 2.40 2.44 2.48 2.52 2.56 2.60 2.64 2.68 2.72 2.76 2.80 2.83 2.87 2.91 2.95 2.99 3.03 3.07 3.11 3.15 3.19 3.23 3.27 3.31 3.35 3.39 3.43 3.46 3.50 3.54 3.58 3.62 3.66 3.70 3.74 3.78 3.82 3.86 3.90 3.94 3.98 4.02 4.06 4.09 4.13 4.17 4.21 4.25 4.29 4.33 4.37 4.41 4.45 4.49 4.53 4.57 4.61 4.65 4.69 4.72 4.76 4.80 4.84 4.88 4.92 4.96 5.00"

.org 0x100

main:

STACK\_INIT

TRIPORT\_ALE\_ONE\_INIT

CS0\_INIT

CS1\_INIT

CS2\_INIT

ADC\_8bit\_INIT

ADC\_CH0\_INIT

LCD\_INIT

ldi R17, 5 //this is the size of each text from the table

loopforever:

ldi ZL, low(LUT<<1)

ldi ZH, high(LUT<<1)

lds R16, ADCA\_CH0\_RES

mov R20, R16 //R20 will be used to change hex to ASCII

MUL R16, R17

mov ZL, R0

ADD ZH, R1

lpm R16, Z+

call OUT\_CHAR\_LCD

lpm R16, Z+

call OUT\_CHAR\_LCD

lpm R16, Z+

call OUT\_CHAR\_LCD

lpm R16, Z+

call OUT\_CHAR\_LCD //calls the X.XX value

ldi R16, 'V'

call OUT\_CHAR\_LCD

ldi R16, ' '

call OUT\_CHAR\_LCD // puts the space between this and hex

ldi R16, '('

call OUT\_CHAR\_LCD //parenthases around hex

ldi R16, '0'

call OUT\_CHAR\_LCD

ldi R16, 'x'

call OUT\_CHAR\_LCD

call OUT\_HEX

ldi R16, ')'

call OUT\_CHAR\_LCD

ldi XL, low(LCDPORT\_COM)

ldi XH, high(LCDPORT\_COM)

call LCD\_BF\_WAIT

ldi R16, 0x02

st X, R16

rjmp loopforever

OUT\_CHAR\_LCD: //outs R16 to LCD

call LCD\_BF\_WAIT

ldi XH, high(LCDPORT\_DAT)

ldi XL, low(LCDPORT\_DAT)

st X, R16

ret

OUT\_STRING\_LCD: //put address of string in Z register

push R16

stringloop:

lpm R16, Z+

cpi R16, 0

breq string\_done

call OUT\_CHAR\_LCD

rjmp stringloop

string\_done:

pop R16

ret

LCD\_BF\_WAIT:

/\*push R0

clr R0

ldi XH, high(LCDPORT\_COM)

ldi XL, low(LCDPORT\_COM)

notready:

ld R0, X

sbrc R0, 7

rjmp notready

pop R0

ret\*/

push R16

push r17

ldi R16, 0

ldi R17, 0

AGAIN:

NOP

NOP

NOP

NOP

NOP

NOP

NOP

NOP

NOP

NOP

NOP

NOP

NOP

NOP

NOP

INC R16

CPI R16, 0

BREQ CARRY

BACK:

CPI R17, 0x01

BRNE AGAIN

BREQ RETURN

CARRY:

INC R17

rjmp BACK

RETURN:

pop r17

pop r16

RET

OUT\_HEX:

mov R21, R20

cbr R21, 0b00001111

lsr R21

lsr R21

lsr R21

lsr R21

cpi R21, 10

brsh Letter

ldi R22, 0x30

ADD R21, R22

mov R16, R21

call OUT\_CHAR\_LCD

continue:

cbr R20, 0b11110000

cpi R20, 10

brsh Letter20

ldi R22, 0x30

ADD R20, R22

mov R16, R20

call OUT\_CHAR\_LCD

ret

Letter:

ldi R22, 0x37

ADD R21, R22

mov R16, R21

call OUT\_CHAR\_LCD

rjmp continue

Letter20:

ldi R22, 0x37

ADD R20, R22

mov R16, R20

call OUT\_CHAR\_LCD

ret

Part 3:

.macro LCD\_INIT

ldi XH, high(LCDPORT\_COM)

ldi XL, low(LCDPORT\_COM)

call LCD\_BF\_WAIT

ldi R16, 0b00111000 // two lines, bigger font, 8 bits

st X, R16

call LCD\_BF\_WAIT

ldi R16, 0b00001111 // display on cursor on curor blink

st X, R16

call LCD\_BF\_WAIT

ldi R16, 0b00000001 // clear disp

st X, R16

call LCD\_BF\_WAIT

ldi R16, 0b00000011 // cursor home

st X, R16

.endmacro

.macro ADC\_8bit\_INIT

ldi R16, 1

sts PORTA\_DIRCLR, R16

sts ADCA\_CTRLA, R16 //enable the ADC

ldi R16, 0b00011100 //turn on free run and set the conversion mode to 8 bit signed

sts ADCA\_CTRLB, R16

ldi R16, 0b00010000 //set teh reference to VCC/1.6 ~= 2.0625

sts ADCA\_REFCTRL, R16 //which the 5 volts on the POT is divided by the board to fit the constraint of

ldi R16, 0b00000011 // set the prescaler to div32 (2MHZ/32 = 62.5 KHZ)

sts ADCA\_PRESCALER, R16

.endmacro

.macro ADC\_CH0\_INIT

ldi R16, 0b10000001

sts ADCA\_CH0\_CTRL, R16 //start taking readings on CH0

.endmacro

/\*

\* Lab6\_Part3.asm

\*

\* Created: 3/24/2013 9:19:54 PM

\* Author: Brandon

\*/

.include "Atxmega128A1udef.inc"

.include "EBI\_INITS.asm"

.equ NameLocation = 0x5000

.org 0x0

rjmp main

main:

.org 0x100

STACK\_INIT

TRIPORT\_ALE\_ONE\_INIT

CS0\_INIT

CS1\_INIT

CS2\_INIT

ADC\_8bit\_INIT

ADC\_CH0\_INIT

LCD\_INIT

loop:

call DELAY500

cp R18, R27

breq nochangesinbuttons

ldi XL, low(LCDPORT\_COM)

ldi XH, high(LCDPORT\_COM)

call LCD\_BF\_WAIT

ldi R16, 0x01

st X, R16

nochangesinbuttons:

call GETKEYPAD

mov R27, R18

cpi R18, 0x00

breq ZERO0

cpi R18, 0x01

breq ONE1

cpi R18, 0x02

breq TWO2

cpi R18, 0x03

breq THREE3

cpi R18, 0x04

breq FOUR4

rjmp loop

ZERO0:

jmp ZERO

ONE1:

jmp ONE

TWO2:

jmp TWO

THREE3:

jmp THREE

FOUR4:

jmp FOUR

.org NameLocation

.db "Brandon Pollack", 0

.org NameLocation + 0x100

LUT:

.db "0.00 0.04 0.08 0.12 0.16 0.20 0.24 0.28 0.31 0.35 0.39 0.43 0.47 0.51 0.55 0.59 0.63 0.67 0.71 0.75 0.79 0.83 0.87 0.91 0.94 0.98 1.02 1.06 1.10 1.14 1.18 1.22 1.26 1.30 1.34 1.38 1.42 1.46 1.50 1.54 1.57 1.61 1.65 1.69 1.73 1.77 1.81 1.85 1.89 1.93 1.97 2.01 2.05 2.09 2.13 2.17 2.20 2.24 2.28 2.32 2.36 2.40 2.44 2.48 2.52 2.56 2.60 2.64 2.68 2.72 2.76 2.80 2.83 2.87 2.91 2.95 2.99 3.03 3.07 3.11 3.15 3.19 3.23 3.27 3.31 3.35 3.39 3.43 3.46 3.50 3.54 3.58 3.62 3.66 3.70 3.74 3.78 3.82 3.86 3.90 3.94 3.98 4.02 4.06 4.09 4.13 4.17 4.21 4.25 4.29 4.33 4.37 4.41 4.45 4.49 4.53 4.57 4.61 4.65 4.69 4.72 4.76 4.80 4.84 4.88 4.92 4.96 5.00"

SchwartzLocation:

.db "May the Schwartz", 0, "Be With You", 0

CreativityLocation:

.db "I had plans"

GETKEYPAD:

push R16

push R17

// ldi R18, 0xFF //fill 18 wiht a value that can't get returned. Notice it does not get pushed onto the stack!!!

ldi R16, 0x0F

sts PORTD\_DIR, R16 //make sure the lower four bits are outputs and the upper are inputs

ldi R16, 0x07

sts PORTD\_OUT, R16 //make the first column low and others high

NOP

NOP

NOP

NOP

NOP //have to wait a little while to compensate for bouncing

lds R17, PORTD\_IN // read if there is something not high on portD

SBRS R17, 7

ldi R18, 0x01

SBRS R17, 6

ldi R18, 0x01

SBRS R17, 5

ldi R18, 0x01

SBRS R17, 4

ldi R18, 0x04

ldi R16, 0x0B

sts PORTD\_OUT, R16

NOP

NOP

NOP

NOP

NOP

lds R17, PORTD\_IN

SBRS R17, 7

ldi R18, 0x02

SBRS R17, 6

ldi R18, 0x02

SBRS R17, 5

ldi R18, 0x02

SBRS R17, 4

ldi R18, 0x04

ldi R16, 0x0D

sts PORTD\_OUT, R16

NOP

NOP

NOP

NOP

NOP

lds R17, PORTD\_IN

SBRS R17, 7

ldi R18, 0x03

SBRS R17, 6

ldi R18, 0x03

SBRS R17, 5

ldi R18, 0x03

SBRS R17, 4

ldi R18, 0x04

ldi R16, 0x0E

sts PORTD\_OUT, R16

NOP

NOP

NOP

NOP

NOP

lds R17, PORTD\_IN

cpi R17, 0xFF

SBRS R17, 7

ldi R18, 0x04

SBRS R17, 6

ldi R18, 0x00

SBRS R17, 5

ldi R18, 0x04

SBRS R17, 4

ldi R18, 0x04

pop r17

pop r16

RET

ZERO:

ldi ZH, high (SchwartzLocation<<1)

ldi ZL, low(SchwartzLocation<<1)

call OUT\_STRING\_LCD

ldi XH, high(LCDPORT\_COM)

ldi XL, low(LCDPORT\_COM)

call LCD\_BF\_WAIT

ldi R16, 0xC0

st X, R16

call OUT\_STRING\_LCD

jmp loop

ONE:

ldi ZH, high(NameLocation<<1)

ldi ZL, low(NameLocation<<1)

call OUT\_STRING\_LCD

jmp loop

TWO:

ldi R17, 5 //this is the size of each text from the table

ldi ZL, low(LUT<<1)

ldi ZH, high(LUT<<1)

lds R16, ADCA\_CH0\_RES

mov R20, R16 //R20 will be used to change hex to ASCII

MUL R16, R17

mov ZL, R0

ADD ZH, R1

lpm R16, Z+

call OUT\_CHAR\_LCD

lpm R16, Z+

call OUT\_CHAR\_LCD

lpm R16, Z+

call OUT\_CHAR\_LCD

lpm R16, Z

call OUT\_CHAR\_LCD //calls the X.XX value

ldi R16, 'V'

call OUT\_CHAR\_LCD

ldi R16, ' '

call OUT\_CHAR\_LCD // puts the space between this and hex

ldi R16, '('

call OUT\_CHAR\_LCD //parenthases around hex

ldi R16, '0'

call OUT\_CHAR\_LCD

ldi R16, 'x'

call OUT\_CHAR\_LCD

call OUT\_HEX

ldi R16, ')'

call OUT\_CHAR\_LCD

ldi XL, low(LCDPORT\_COM)

ldi XH, high(LCDPORT\_COM)

call LCD\_BF\_WAIT

ldi R16, 0x02

st X, R16

jmp loop

THREE:

ldi XL, low(LCDPORT\_COM)

ldi XH, high(LCDPORT\_COM)

ldi R25, 0x01

st X, R25

jmp loop

FOUR:

ldi ZL, low(CreativityLocation<<1)

ldi ZH, high(CreativityLocation<<1)

call OUT\_STRING\_LCD

jmp loop

OUT\_CHAR\_LCD: //outs R16 to LCD

call LCD\_BF\_WAIT

ldi XH, high(LCDPORT\_DAT)

ldi XL, low(LCDPORT\_DAT)

st X, R16

ret

OUT\_STRING\_LCD: //put address of string in Z register

push R16

stringloop:

lpm R16, Z+

cpi R16, 0

breq string\_done

call OUT\_CHAR\_LCD

rjmp stringloop

string\_done:

pop R16

ret

LCD\_BF\_WAIT:

/\*push R0

clr R0

ldi XH, high(LCDPORT\_COM)

ldi XL, low(LCDPORT\_COM)

notready:

ld R0, X

sbrc R0, 7

rjmp notready

pop R0

ret\*/

push R16

push r17

ldi R16, 0

ldi R17, 0

AGAINLCD:

NOP

NOP

NOP

NOP

NOP

NOP

NOP

NOP

NOP

NOP

NOP

NOP

NOP

NOP

NOP

INC R16

CPI R16, 0

BREQ CARRYLCD

BACKLCD:

CPI R17, 0x01

BRNE AGAINLCD

BREQ RETURNLCD

CARRYLCD:

INC R17

rjmp BACKLCD

RETURNLCD:

pop r17

pop r16

RET

OUT\_HEX:

mov R21, R20

cbr R21, 0b00001111

lsr R21

lsr R21

lsr R21

lsr R21

cpi R21, 10

brsh Letter

ldi R22, 0x30

ADD R21, R22

mov R16, R21

call OUT\_CHAR\_LCD

continue:

cbr R20, 0b11110000

cpi R20, 10

brsh Letter20

ldi R22, 0x30

ADD R20, R22

mov R16, R20

call OUT\_CHAR\_LCD

ret

Letter:

ldi R22, 0x37

ADD R21, R22

mov R16, R21

call OUT\_CHAR\_LCD

rjmp continue

Letter20:

ldi R22, 0x37

ADD R20, R22

mov R16, R20

call OUT\_CHAR\_LCD

ret

DELAY500:

push R16

push r17

ldi R16, 0

ldi R17, 0

AGAIN:

NOP

NOP

NOP

NOP

NOP

NOP

NOP

NOP

NOP

NOP

NOP

NOP

NOP

NOP

NOP

INC R16

CPI R16, 0

BREQ CARRY

BACK:

CPI R17, 0x90

BRNE AGAIN

BREQ RETURN

CARRY:

INC R17

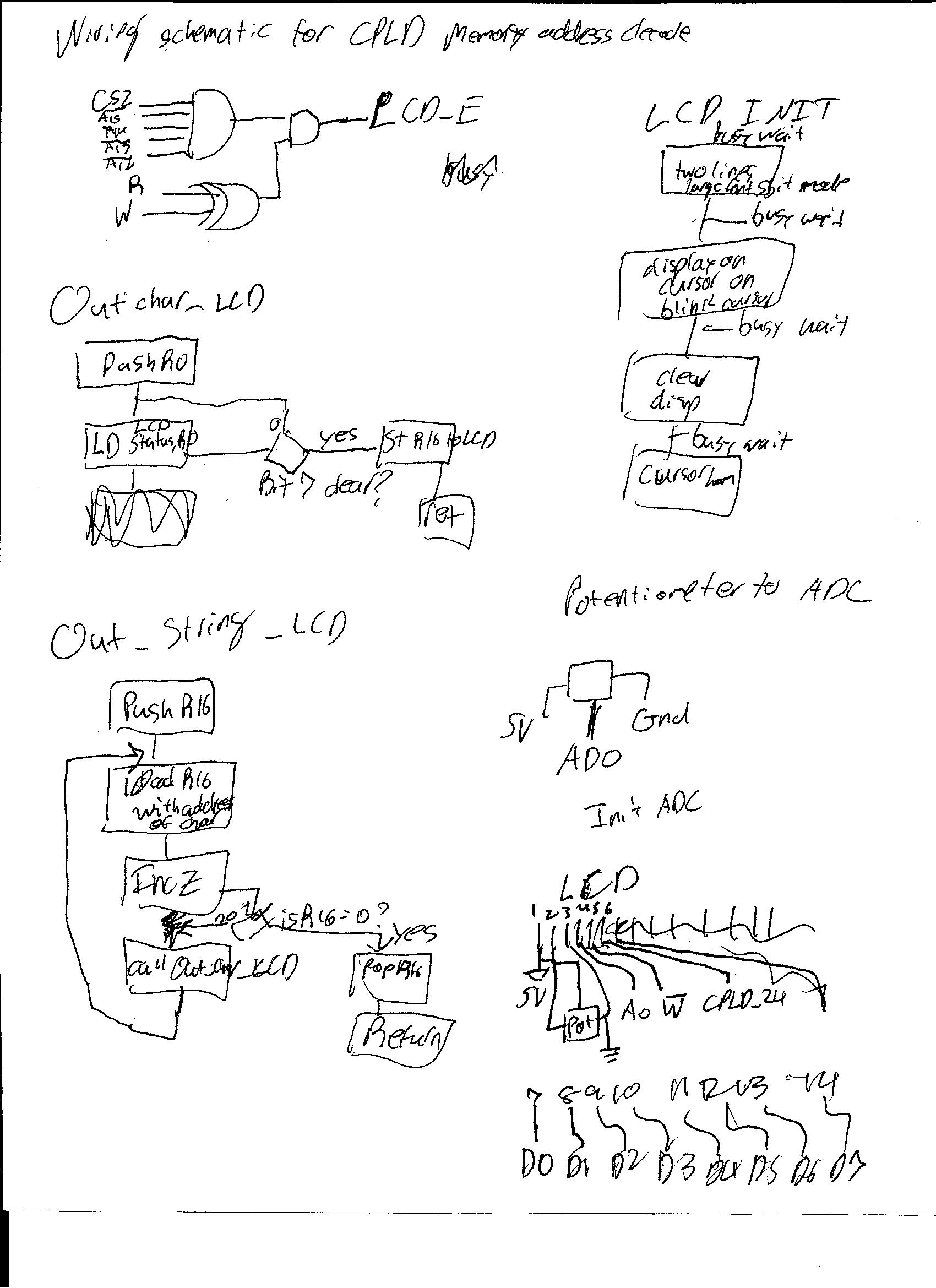
rjmp BACK

RETURN:

pop r17

pop r16

RET

Schematics for Lab 6:

Quartus stuff:

