HW4 Write Up

1)

(a)

ASM: 7 Instructions

No optimization: 42 Instructions

Size optimization:

(b)

r24 gets 0x03 when average(2,4)

(c) 0x3FF2

(d) Stored indirectly using Y pointer which is pointing to the stack in SRAM

2)

ASM: ~104 Instructions

C Compiled: ~221 Instructions

Part 1 Code

ASM

/\* hw4-1asm.asm

\*

\* HW 4-1 ASM

\* Name: Nicholas Imamshah

\* Section: 6957

\* TA Name: Daniel Gonzalez

\* Description: The purpose of this program is to compute the average of two 8-bit integers

\*/

.nolist

.include "ATxmega128A1Udef.inc"

.list

.org 0x0000

rjmp MAIN

.org 0x200

MAIN:

ldi r16, 0x02 ;load num1

ldi r17, 0x04 ;load num2

rcall AVERAGE ;compute average

nop ;dummy instr for breakpoint

AVERAGE:

add r16, r17 ;add the numbers

asr r16 ;arithmetic shift right to divide by 2

ret

C

/\* hw4-1c.c

\*

\* HW 4-1 C

\* Name: Nicholas Imamshah

\* Section: 6957

\* TA Name: Daniel Gonzalez

\* Description: The purpose of this program is to compute the average of two 8-bit integers using C programming.

\*/

//////////////////////////////////////INCLUDES///////////////////////////////////////

#include <avr/io.h>

//////////////////////////////////INITIALIZATIONS////////////////////////////////////

#define F\_CPU 2000000

/////////////////////////////////////PROTOTYPES//////////////////////////////////////

uint8\_t average(uint8\_t a, uint8\_t b);

///////////////////////////////////MAIN FUNCTION/////////////////////////////////////

int main(void)

{

uint8\_t result = average(2, 4);

}

/////////////////////////////////////FUNCTIONS///////////////////////////////////////

uint8\_t average(uint8\_t a, uint8\_t b)

{

uint8\_t avg = (a+b)/2;

return avg;

}

Part 2 Code

ASM

/\*

\* lab2c.asm

\*

\* Lab 2 Part C

\* Name: Nicholas Imamshah

\* Section: 6957

\* TA Name: Daniel Gonzalez

\* Description: The purpose of this program is to interface with a keypad.

\*/

KEYPAD:

ldi r19, 0x18 ;Need OPC set to PULLUP for all Keypad pins

sts PORTF\_PIN7CTRL, r19

sts PORTF\_PIN6CTRL, r19

sts PORTF\_PIN5CTRL, r19

sts PORTF\_PIN4CTRL, r19

sts PORTF\_PIN3CTRL, r19

sts PORTF\_PIN2CTRL, r19

sts PORTF\_PIN1CTRL, r19

sts PORTF\_PIN0CTRL, r19

ldi r16, 0x0F ;set the LSNibble of PORTF to output

sts PORTF\_DIRSET, r16

rcall KEYSCAN

ret ;return after initializing keypad

KEYSCAN:

ldi r16, 0xFF ;load default output for keypad

rcall COL1 ;scan column 1

rcall COL2 ;scan column 2

rcall COL3 ;scan column 3

rcall COL4 ;scan column 4

ret ;return after loading result

KEYPRESSED:

lds r16, PORTF\_IN ;check PortF's input again

cpi r16, 0xF0 ;if it is < 0xF0, then one of the keys are pressed

brlo KEYPRESSED ;loop until this is not the case

rjmp KEYPAD

INIT:

sts PORTF\_OUT, r17 ;initiates the bits for each columns scan

nop

lds r17, PORTF\_IN ;get the input bits from PortF

ori r17, 0x0F ;bit mask the input to simplify code

ret

COL1:

ldi r17, 0x0E ;column 1 is 0b1110

rcall INIT ;check for pressed key

cpi r17, 0xEF ;check if row 1

breq PRESS\_1

cpi r17, 0xDF ;check if row 2

breq PRESS\_4

cpi r17, 0xBF ;check if row 3

breq PRESS\_7

cpi r17, 0x7F ;check if row 4

breq PRESS\_ST

ret

PRESS\_1: ;load value corresponding to key pressed

ldi r16, 0x01

ret

PRESS\_4:

ldi r16, 0x04

ret

PRESS\_7:

ldi r16, 0x07

ret

PRESS\_ST:

ldi r16, 0x0E

ret

COL2:

ldi r17, 0x0D ;column 2 is 0b1011

rcall INIT ;check for pressed key

cpi r17, 0xEF ;check if row 1

breq PRESS\_2

cpi r17, 0xDF ;check if row 2

breq PRESS\_5

cpi r17, 0xBF ;check if row 3

breq PRESS\_8

cpi r17, 0x7F ;check if row 4

breq PRESS\_0

ret

PRESS\_2: ;load value corresponding to key pressed

ldi r16, 0x02

ret

PRESS\_5:

ldi r16, 0x05

ret

PRESS\_8:

ldi r16, 0x08

ret

PRESS\_0:

ldi r16, 0x00

ret

COL3:

ldi r17, 0x0B ;column 3 is 0b0111

rcall INIT ;check for pressed key

cpi r17, 0xEF ;check if row 1

breq PRESS\_3

cpi r17, 0xDF ;check if row 2

breq PRESS\_6

cpi r17, 0xBF ;check if row 3

breq PRESS\_9

cpi r17, 0x7F ;check if row 4

breq PRESS\_NUM

ret

PRESS\_3: ;load value corresponding to key pressed

ldi r16, 0x03

ret

PRESS\_6:

ldi r16, 0x06

ret

PRESS\_9:

ldi r16, 0x09

ret

PRESS\_NUM:

ldi r16, 0x0F

ret

COL4:

ldi r17, 0x07 ;column 4 is 0b1110

rcall INIT ;check for pressed key

cpi r17, 0xEF ;check if row 1

breq PRESS\_A

cpi r17, 0xDF ;check if row 2

breq PRESS\_B

cpi r17, 0xBF ;check if row 3

breq PRESS\_C

cpi r17, 0x7F ;check if row 4

breq PRESS\_D

ret

PRESS\_A: ;load value corresponding to key pressed

ldi r16, 0x0A

ret

PRESS\_B:

ldi r16, 0x0B

ret

PRESS\_C:

ldi r16, 0x0C

ret

PRESS\_D:

ldi r16, 0x0D

ret

C

/\* keypad.c

\*

\* Keypad in C

\* Name: Nicholas Imamshah

\* Section: 6957

\* TA Name: Daniel Gonzalez

\* Description: The purpose of this program is to interface the XMEGA processor

\* with an external Keypad.

\*/

//////////////////////////////////////INCLUDES///////////////////////////////////////

#include <avr/io.h>

//////////////////////////////////INITIALIZATIONS////////////////////////////////////

#define F\_CPU 2000000

#define PF\_OPC 0x18

uint8\_t keys[] = {0x1, 0x4, 0x7, 0xE, 0x2, 0x5, 0x8, 0x0, 0x3, 0x6, 0x9, 0xF, 0xA, 0xB, 0xC, 0xD};

/////////////////////////////////////PROTOTYPES//////////////////////////////////////

void keypad\_init(void);

uint8\_t keyscan(void);

uint8\_t scan(uint8\_t);

///////////////////////////////////MAIN FUNCTION/////////////////////////////////////

int main(void)

{

keypad\_init();

uint8\_t key;

while(1)

{

key = keyscan();

}

}

/////////////////////////////////////FUNCTIONS///////////////////////////////////////

void keypad\_init(void)

{

PORTF.PIN7CTRL = PF\_OPC; // Set OPC to Pull-Up for all Keypad pins

PORTF.PIN6CTRL = PF\_OPC;

PORTF.PIN5CTRL = PF\_OPC;

PORTF.PIN4CTRL = PF\_OPC;

PORTF.PIN3CTRL = PF\_OPC;

PORTF.PIN2CTRL = PF\_OPC;

PORTF.PIN1CTRL = PF\_OPC;

PORTF.PIN0CTRL = PF\_OPC;

PORTF.DIRSET = 0x0F; // Set LSNibble of PortF as Output

}

uint8\_t keyscan(void)

{

uint8\_t line = 0x0F;

uint8\_t input, index, key;

for (uint8\_t i = 0; i < 4; i++) // Iterate columns

{

line &= ~(0x01 << i); // Iterate shift 0x08 by i and not to hit each col

PORTF.OUT = line; // Output value for col

asm("nop");

input = PORTF.IN & 0xF0; // Read Input and bitmask off Output bits

if (input < 0xF0)

{

switch (input)

{

case 0xE0:

index = 0x00;

break;

case 0xD0:

index = 0x01;

break;

case 0xB0:

index = 0x02;

break;

case 0x70:

index = 0x03;

break;

}

key = keys[index+4\*i];

while ((PORTF.IN & 0xF0) < 0xF0);

break;

} else

{

key = 0xFF;

}

}

return key;

}