Imamshah, Nicholas
EEL 3744
Section: 6957
HW4

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;
}
```