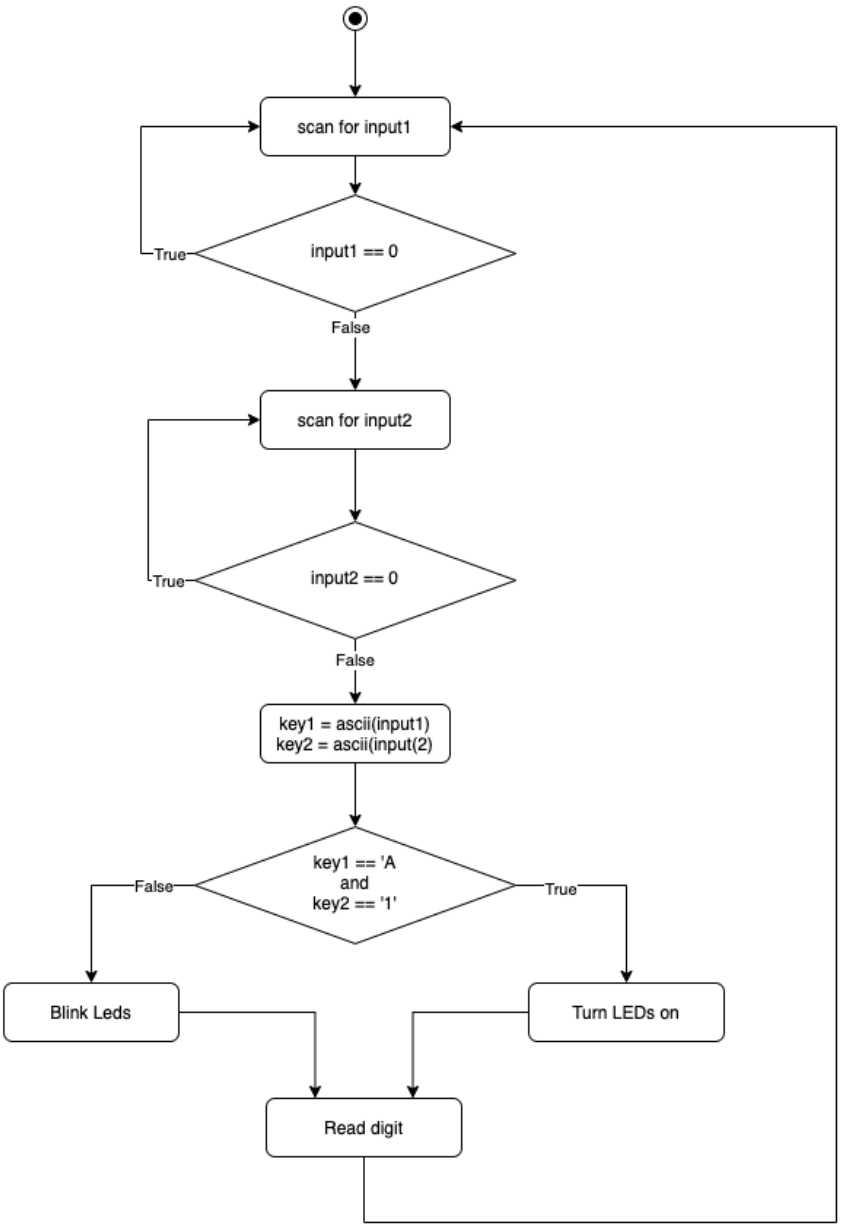
Εργαστήριο Μικροϋπολογιστών
3η Σειρά Ασκήσεων

Ιάσονας Νικολάου (03116129) - Χαρίτων Χαριτωνίδης (03116694)

Άσκηση 1

Flowchart:



Κώδικας:

```
#define F_CPU 8000000UL
#include <avr/io.h>
#include <util/delay.h>

int keypad_to_ascii(int x) {
	int ascii[] = {'*','0','#','D','7','8','9','C','4','5','6','B','1','2','3','A'};
	for (int i=0; i<16; i++) {
		if (x & (1 << i)) {
			return ascii[i];
		}
	}
	return 'Z';
}

int scan_row_sim(char row) {
	PORTC = row << 4;
	_delay_us(500);
	asm volatile ("nop");
	asm volatile ("nop");
	return (PINC & 0x0f);
}

int scan_keypad_sim() {
	int output = 0;
	for (int row=1; row<=8; row <<= 1) {
		output <<= 4;
		output = output | scan_row_sim(row);
	}
	PORTC = 0x00;
	return output;
}
```

```
int prev_input;

int scan_keypad_rising_edge() {
	int input = scan_keypad_sim();
	_delay_ms(15);
	input &= scan_keypad_sim(); // bouncing
	int tmp = input;
	input = input & ~prev_input;
	prev_input = tmp;
	return input;
}

int main(void)
{
	while (42) {
		DDRC = 0xf0;
		DDRB = 0xff; // B: output
		int input1, input2;
		char key1, key2;
		scan_keypad_rising_edge();
		while (1) { // read 1st digit
			input1 = scan_keypad_rising_edge();
			if (input1) {
				break;
			}
		}
		scan_keypad_rising_edge();
		while (1) { // read 2nd digit
			input2 = scan_keypad_rising_edge();
			if (input2) {
				break;
			}
```

```
            }

            key1 = keypad_to_ascii(input1);
            key2 = keypad_to_ascii(input2);

            if (key1 == 'A' && key2 == '1') {
                PORTB = 0xff;
                _delay_ms(4000);
                PORTB = 0x00;
            }
            else {
                for (int i=0; i<4; ++i) {
                    PORTB = 0xff;
                    _delay_ms(500);
                    PORTB = 0x00;
                    _delay_ms(500);
                }
            }
        }

}
```

Άσκηση 2

Κώδικας:

```
.DSEG
_tmp_: .byte 2

.CSEG

main:
        ser r26
```

```
        out DDRD, r26 ; set PORTD as output
        out DDRB, r26 ; set PORTB as output
        ldi r26, 0xf0 ; set 4 MSB of PORTC as output
        clr r24
        rcall lcd_init_sim
        out DDRC, r26
        ldi r24, low(RAMEND) ; initialize stack pointer
        out SPL, r24
        ldi r24, high(RAMEND)
        out SPH, r24
        rcall scan_keypad_rising_edge_sim

input1: ; read while input1 != 0
        rcall scan_keypad_rising_edge_sim
        mov r16, r24
        or r16, r25
        breq input1
        mov r16, r24 ; r17:r16 <- input1
        mov r17, r25
        rcall scan_keypad_rising_edge_sim

input2: ; read while input2 != 0
        rcall scan_keypad_rising_edge_sim
        mov r18, r24
        or r18, r25
        breq input2
        mov r18, r24 ; r19:r18 <- input2
        mov r19, r25

        ; convert input1 to ascii1
        mov r24, r16
        mov r25, r17
        rcall keypad_to_ascii_sim
```

```
        mov r20, r24 ; r20 <- ascii1

        ; convert input2 to ascii2
        mov r24, r18
        mov r25, r19
        rcall keypad_to_ascii_sim
        mov r21, r24 ; r21 <- ascii2

        cpi r20, 'A'
        brne fail
        cpi r21, '1'
        brne fail
        jmp pass

pass:
        clr r24
        rcall lcd_init_sim
        ldi r24, 'W'
        rcall lcd_data_sim
        ldi r24, 'E'
        rcall lcd_data_sim
        ldi r24, 'L'
        rcall lcd_data_sim
        ldi r24, 'C'
        rcall lcd_data_sim
        ldi r24, 'O'
        rcall lcd_data_sim
        ldi r24, 'M'
        rcall lcd_data_sim
        ldi r24, 'E'
        rcall lcd_data_sim
        ldi r24, ' '
        rcall lcd_data_sim
```

```
        ldi r24, 'A'
        rcall lcd_data_sim
        ldi r24, '1'
        rcall lcd_data_sim
leds_on:
        ser r26
        out PORTB, r26
        ldi r24, low(4000)     ; wait for 4 secs.
        ldi r25, high(4000)
        rcall wait_msec
        clr r26
        out PORTB, r26
        jmp main

fail:
        clr r24
        rcall lcd_init_sim
        ldi r24, 'A'
        rcall lcd_data_sim
        ldi r24, 'L'
        rcall lcd_data_sim
        ldi r24, 'A'
        rcall lcd_data_sim
        ldi r24, 'R'
        rcall lcd_data_sim
        ldi r24, 'M'
        rcall lcd_data_sim
        ldi r24, ' '
        rcall lcd_data_sim
        ldi r24, 'O'
        rcall lcd_data_sim
        ldi r24, 'N'
        rcall lcd_data_sim
```

```
blink: ; blink 4 times
        ser r26
        out PORTB, r26
        ldi r24, low(500)
        ldi r25, high(500)
        rcall wait_msec
        clr r26
        out PORTB, r26
        ldi r24, low(500)
        ldi r25, high(500)
        rcall wait_msec
        ser r26
        out PORTB, r26
        ldi r24, low(500)
        ldi r25, high(500)
        rcall wait_msec
        clr r26
        out PORTB, r26
        ldi r24, low(500)
        ldi r25, high(500)
        rcall wait_msec
        ser r26
        out PORTB, r26
        ldi r24, low(500)
        ldi r25, high(500)
        rcall wait_msec
        clr r26
        out PORTB, r26
        ldi r24, low(500)
        ldi r25, high(500)
        rcall wait_msec
        ser r26
        out PORTB, r26
```

```
        ldi r24, low(500)
        ldi r25, high(500)
        rcall wait_msec
        clr r26
        out PORTB, r26
        ldi r24, low(500)
        ldi r25, high(500)
        rcall wait_msec
        jmp main

wait_msec:
        push r24
        push r25
        ldi r24, low(998)
        ldi r25, high(998)
        rcall wait_usec
        pop r25
        pop r24
        sbiw r24, 1
        brne wait_msec
        ret

wait_usec:
        sbiw r24, 1
        nop
        nop
        nop
        nop
        brne wait_usec
        ret

scan_row_sim:
        out PORTC, r25
```

```
        push r24
        push r25
        ldi r24, low(500)
        ldi r25, high(500)
        rcall wait_usec
        pop r25
        pop r24
        nop
        nop
        in r24,PINC
        andi r24, 0x0f
        ret

scan_keypad_sim:
        push r26
        push r27
        ldi r25, 0x10
        rcall scan_row_sim
        swap r24
        mov r27, r24
        ldi r25, 0x20
        rcall scan_row_sim
        add r27, r24
        ldi r25, 0x40
        rcall scan_row_sim
        swap r24
        mov r26, r24
        ldi r25, 0x80
        rcall scan_row_sim
        add r26, r24
        movw r24, r26
        clr r26
        out PORTC, r26
```

```
        pop r27
        pop r26
        ret

scan_keypad_rising_edge_sim:
        push r22
        push r23
        push r26
        push r27
        rcall scan_keypad_sim
        push r24
        push r25
        ldi r24, 15
        ldi r25 ,0
        rcall wait_msec
        rcall scan_keypad_sim
        pop r23
        pop r22
        and r24, r22
        and r25, r23
        ldi r26, low(_tmp_)
        ldi r27, high(_tmp_)
        ld r23, X+
        ld r22, X
        st X, r24
        st -X, r25
        com r23
        com r22
        and r24, r22
        and r25, r23
        pop r27
        pop r26
        pop r23
```

```
        pop r22
        ret

keypad_to_ascii_sim:
        push r26
        push r27
        movw r26, r24
        ldi r24,'*'
        sbrc r26, 0
        rjmp return_ascii
        ldi r24,'0'
        sbrc r26, 1
        rjmp return_ascii
        ldi r24,'#'
        sbrc r26, 2
        rjmp return_ascii
        ldi r24, 'D'
        sbrc r26, 3
        rjmp return_ascii
        ldi r24, '7'
        sbrc r26, 4
        rjmp return_ascii
        ldi r24, '8'
        sbrc r26, 5
        rjmp return_ascii
        ldi r24, '9'
        sbrc r26, 6
        rjmp return_ascii
        ldi r24, 'C'
        sbrc r26, 7
        rjmp return_ascii
        ldi r24, '4'
        sbrc r27, 0
```

```
        rjmp return_ascii
        ldi r24, '5'
        sbrc r27, 1
        rjmp return_ascii
        ldi r24, '6'
        sbrc r27, 2
        rjmp return_ascii
        ldi r24, 'B'
        sbrc r27, 3
        rjmp return_ascii
        ldi r24, '1'
        sbrc r27, 4
        rjmp return_ascii
        ldi r24, '2'
        sbrc r27, 5
        rjmp return_ascii
        ldi r24, '3'
        sbrc r27, 6
        rjmp return_ascii
        ldi r24, 'A'
        sbrc r27, 7
        rjmp return_ascii
        clr r24
        rjmp return_ascii
return_ascii:
        pop r27
        pop r26
        ret

write_2_nibbles_sim:
    push r24
    push r25
    ldi r24, low(6000)
```

```
    ldi r25, high(6000)
    rcall wait_usec
    pop r25
    pop r24
    push r24
    in r25, PIND
    andi r25, 0x0f
    andi r24, 0xf0
    add r24, r25
    out PORTD, r24
    sbi PORTD, PD3
    cbi PORTD, PD3
    push r24
    push r25
    ldi r24, low(6000)
    ldi r25 ,high(6000)
    rcall wait_usec
    pop r25
    pop r24
    pop r24
    swap r24
    andi r24 ,0xf0
    add r24, r25
    out PORTD, r24
    sbi PORTD, PD3
    cbi PORTD, PD3
    ret

lcd_data_sim:
    push r24
    push r25
    sbi PORTD, PD2
    rcall write_2_nibbles_sim
```

```
    ldi r24 ,43
    ldi r25 ,0
    rcall wait_usec
    pop r25
    pop r24
    ret

lcd_command_sim:
    push r24
    push r25
    cbi PORTD, PD2
    rcall write_2_nibbles_sim
    ldi r24, 39
    ldi r25, 0
    rcall wait_usec
    pop r25
    pop r24
    ret

lcd_init_sim:
    push r24
    push r25
    ldi r24, 40
    ldi r25, 0
    rcall wait_msec
    ldi r24, 0x30
    out PORTD, r24
    sbi PORTD, PD3
    cbi PORTD, PD3
    ldi r24, 39
    ldi r25, 0
    rcall wait_usec
    push r24
```

```
push r25
ldi r24,low(1000)
ldi r25,high(1000)
rcall wait_usec
pop r25
pop r24
ldi r24, 0x30
out PORTD, r24
sbi PORTD, PD3
cbi PORTD, PD3
ldi r24,39
ldi r25,0
rcall wait_usec
push r24
push r25
ldi r24 ,low(1000)
ldi r25 ,high(1000)
rcall wait_usec
pop r25
pop r24
ldi r24,0x20
out PORTD, r24
sbi PORTD, PD3
cbi PORTD, PD3
ldi r24,39
ldi r25,0
rcall wait_usec
push r24
push r25
ldi r24 ,low(1000)
ldi r25 ,high(1000)
rcall wait_usec
pop r25
```

```
pop r24
ldi r24,0x28
rcall lcd_command_sim
ldi r24,0x0c
rcall lcd_command_sim
ldi r24,0x01
rcall lcd_command_sim
ldi r24, low(1530)
ldi r25, high(1530)
rcall wait_usec
ldi r24 ,0x06
rcall lcd_command_sim
pop r25
pop r24
ret
```