Εργαστήριο Μικροϋπολογιστών
8η Άσκηση

Ομάδα: Δ12
Βακαλόπουλος Θεόδωρος, ΑΜ: 03114013
Μαυρομμάτης Ιάσων, ΑΜ: 03114771
Νικητοπούλου Δήμητρα, ΑΜ: 03114954

Άσκηση 1

Ο κώδικας της άσκησης φαίνεται παρακάτω:

```
.include "m16def.inc"
.def reg = r21
.def myr24 = r19
.def myr25 =r18
.def temp = r17

main:
   ldi reg, low(RAMEND)
   out SPL, reg
   ldi reg, high(RAMEND)
   out SPH, reg
   ser reg
   out DDRA, reg          ;PortA as output
   rcall my_routine
   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

one_wire_reset:
   sbi DDRA ,PA4 ; PA4 configured for output
   cbi PORTA ,PA4 ; 480 ?sec reset pulse
   ldi r24 ,low(480)
   ldi r25 ,high(480)
   rcall wait_usec
   cbi DDRA ,PA4 ; PA4 configured for input
   cbi PORTA ,PA4
   ldi r24 ,100 ; wait 100 ?sec for devices
   ldi r25 ,0 ; to transmit the presence pulse
   rcall wait_usec
   in r24 ,PINA ; sample the line
   push r24
   ldi r24 ,low(380) ; wait for 380 ?sec
   ldi r25 ,high(380)
   rcall wait_usec
   pop r25 ; return 0 if no device was
   clr r24 ; detected or 1 else
   sbrs r25 ,PA4
   ldi r24 ,0x01
   ret

one_wire_receive_bit:
   sbi DDRA ,PA4
   cbi PORTA ,PA4 ; generate time slot
   ldi r24 ,0x02
   ldi r25 ,0x00
   rcall wait_usec
   cbi DDRA ,PA4 ; release the line
   cbi PORTA ,PA4
   ldi r24 ,10 ; wait 10 ms
   ldi r25 ,0
   rcall wait_usec
   clr r24 ; sample the line
   sbic PINA ,PA4
   ldi r24 ,1
   push r24
   ldi r24 ,49 ; delay 49 ?s to meet the standards
   ldi r25 ,0 ; for a minimum of 60 ?sec time slot
   rcall wait_usec ; and a minimum of 1 ?sec recovery time
   pop r24
   ret

one_wire_transmit_bit:
   push r24 ; save r24
   sbi DDRA ,PA4
   cbi PORTA ,PA4 ; generate time slot
   ldi r24 ,0x02
   ldi r25 ,0x00
```

```
    rcall wait_usec
    pop r24 ; output bit
    sbrc r24 ,0
    sbi PORTA ,PA4
    sbrs r24 ,0
    cbi PORTA ,PA4
    ldi r24 ,58 ; wait 58 ?sec for the
    ldi r25 ,0 ; device to sample the line
    rcall wait_usec
    cbi DDRA ,PA4 ; recovery time
    cbi PORTA ,PA4
    ldi r24 ,0x01
    ldi r25 ,0x00
    rcall wait_usec
    ret

one_wire_receive_byte:
    ldi r27 ,8
    clr r26
    loop_:
    rcall one_wire_receive_bit
    lsr r26
    sbrc r24 ,0
    ldi r24 ,0x80
    or r26 ,r24
    dec r27
    brne loop_
    mov r24 ,r26
    ret

one_wire_transmit_byte:
    mov r26 ,r24
    ldi r27 ,8
    _one_more_:
    clr r24
    sbrc r26 ,0
    ldi r24 ,0x01
    rcall one_wire_transmit_bit
    lsr r26
    dec r27
    brne _one_more_
    ret


my_routine:
    rcall one_wire_reset
    cpi r24,0x01
    brne not_connected
    ldi r24,0xcc
    rcall one_wire_transmit_byte
    ldi r24,0x44
```

```
   rcall one_wire_transmit_byte
not_ready:
   rcall one_wire_receive_bit
   sbrs r24,0
   rjmp not_ready
wake_up:
   rcall one_wire_reset
   cpi r24,0x01
   brne not_connected
   ldi r24,0xcc
   rcall one_wire_transmit_byte
   ldi r24,0xbe
   rcall one_wire_transmit_byte
   rcall one_wire_receive_byte
   mov myr24,r24
   rcall one_wire_receive_byte
   mov myr25,r24
   cpi myr25,0xff
   brne cont2
   dec myr24 ;2's->1's complement

cont2:
   rjmp show_temperature
not_connected:
   ldi myr25,0x80
   clr myr24
show_temperature:
    mov temp,myr24
    lsr temp
    out PORTA,temp
    ldi r24, LOW(1000)
    ldi r25, HIGH(1000)
    rcall wait_msec      ;Wait for 1s
    ret
```

Σχόλια:

- Το πρόγραμμά μας περιλαμβάνει ένα μικρό κυρίως μέρος που κάνει τις απαραίτητες αρχικοποιήσεις κι έπειτα παραχωρεί τον έλεγχο στη ρουτίνα my_routine που θα εκτελέσει τα ζητούμενα της εκφώνησης.
- Επειδή η απεικόνιση γίνεται σε συμπλήρωμα ως προς 1, ελέγχω αν το most significant byte είναι 0xff που υποδηλώνει αρνητικό αριθμό και αν είναι αφαιρώ 1 για να μετατρέψω την αναπαράσταση σε μορφή συμπληρώματος ως προς 2 σε συμπλήρωμα ως προς 1.

Άσκηση 2

Ο κώδικας της άσκησης φαίνεται παρακάτω:

```
.include "m16def.inc"
.def hundreds = r20
.def tens = r21
.def units = r22
.def reg = r16
.def myr25 =r17
.def myr24 =r18
.def temp =r19

.org 0x00
rjmp reset

 .DSEG
 _tmp_:.byte 2

 .CSEG

reset:
    ldi reg, low(RAMEND)
    out SPL, reg
    ldi reg, high(RAMEND)
    out SPH, reg
            ldi r24 ,0xfc ;11111100, PD7-2 output
    out DDRD ,r24
    ldi r24 ,(1 << PC7) | (1 << PC6) | (1 << PC5) | (1 << PC4)
    out DDRC ,r24
    ldi r26,low(_tmp_) ;r26-r27 -> X
    ldi r27,high(_tmp_)
    clr reg
    st X+,reg
    st X,reg
    ser reg
    out DDRA, reg          ;PortA as output
    rcall lcd_init

main:
    rcall my_routine
            cpi myr25,0x80
    breq no_device
    jmp calc_and_show

no_device:
     ldi r24, 10     ;Initialise for 0.01s delay
     rcall scan_keypad_rising_edge
     rcall keypad_to_hex
```

```
    cpi r24,0xff ; no key read
    breq no_device
    mov myr25,r24
    CLC
    rol myr25
    rol myr25
    rol myr25
    rol myr25

read2nd:
    ldi r24, 10     ;Initialise for 0.01s delay
    rcall scan_keypad_rising_edge
    rcall keypad_to_hex
    cpi r24, 0xff  ; no key read
    breq read2nd
    or myr25,r24

read3rd:
    ldi r24, 10     ;Initialise for 0.01s delay
    rcall scan_keypad_rising_edge
    rcall keypad_to_hex
    cpi r24, 0xff  ; no key read
    breq read3rd
    mov myr24,r24
    CLC
    rol myr24
    rol myr24
    rol myr24
    rol myr24

read4th:
    ldi r24, 10     ;Initialise for 0.01s delay
    rcall scan_keypad_rising_edge
    rcall keypad_to_hex
    cpi r24, 0xff  ; no key read
    breq read4th
    or myr24,r24
    cpi myr25,0xff
    brne calc_and_show
    dec myr24 ;2's->1's complement

calc_and_show:
        rcall lcd_init
   cpi myr25,0x80
   brne continue

   rcall display_no_device                  ;If no device detected
        ldi r24, LOW(1000)
```

```
    ldi r25, HIGH(1000)
    rcall wait_msec      ;Wait for 1s
    rjmp main

continue:
    clr reg ;flag=0
    ldi temp, 0xff
    cp myr25, temp
    breq negative

    ldi r24, '+'      ;If temperature is over 0
    rcall lcd_data
    rjmp next

negative:             ;If temperature is under 0
    ldi r24, '-'
    rcall lcd_data
    com myr24

next:
    mov temp, myr24
    andi temp, 0x01
    cpi temp, 0x00
    breq hex_to_bcd
    ldi reg, 0xff     ;If r24 is odd-> flag = 1

hex_to_bcd:
    clr units
    clr tens
    clr hundreds
    lsr myr24           ;Divide by 2 to convert to Celsius degrees
loop1:
    subi myr24, 0x64
    brcs case1
    inc hundreds
    rjmp loop1
case1:
    ldi temp, 0x64
    add myr24, temp
loop2:
    subi myr24, 0x0A
    brcs case2
    inc tens
    rjmp loop2
case2:
    ldi temp, 0x0A
    add myr24, temp
    mov units, myr24
```

```
    rcall print_temperature
            ldi r24,0xb2
    rcall lcd_data
            ldi r24, 'C'
    rcall lcd_data ;print oC
            ldi r24, LOW(1000)
    ldi r25, HIGH(1000)
    rcall wait_msec      ;Wait for 1s
    rjmp main


print_temperature:
    cpi hundreds, 0x00
    breq check_tens
    ldi temp, 0x30
    add hundreds, temp
    mov r24, hundreds
    rcall lcd_data

check_tens:
    cpi tens, 0x00
    brne print_tens
    cpi hundreds, 0x00
    breq print_units

print_tens:
            ldi temp, 0x30
    add tens, temp
    mov r24, tens
    rcall lcd_data

print_units:
            ldi temp, 0x30
    add units, temp
    mov r24, units
    rcall lcd_data

    cpi reg, 0x00
    breq leave
    ldi r24, '.'
    rcall lcd_data
    ldi r24, '5'
    rcall lcd_data

leave:
    ret
```

```
display_no_device:      ;Display "NO Device"
    ldi r24, 'N'
    rcall lcd_data
    ldi r24, 'O'
    rcall lcd_data
    ldi r24, ' '
    rcall lcd_data
    ldi r24, 'D'
    rcall lcd_data
    ldi r24, 'e'
    rcall lcd_data
    ldi r24, 'v'
    rcall lcd_data
    ldi r24, 'i'
    rcall lcd_data
    ldi r24, 'c'
    rcall lcd_data
    ldi r24, 'e'
    rcall lcd_data
    ret



my_routine:
    rcall one_wire_reset
    cpi r24,0x01
    brne not_connected
    ldi r24,0xcc
    rcall one_wire_transmit_byte
    ldi r24,0x44
    rcall one_wire_transmit_byte
not_ready:
    rcall one_wire_receive_bit
    sbrs r24,0
    rjmp not_ready
wake_up:
    rcall one_wire_reset
    cpi r24,0x01
    brne not_connected
    ldi r24,0xcc
    rcall one_wire_transmit_byte
    ldi r24,0xbe
    rcall one_wire_transmit_byte
    rcall one_wire_receive_byte
    mov myr24,r24
    rcall one_wire_receive_byte
            mov myr25,r24
```

```
    cpi myr25,0xff
    brne cont2
    dec myr24 ;2's->1's complement

cont2:
    rjmp show_temperature
not_connected:
    ldi myr25,0x80
    clr myr24
show_temperature:
            mov temp,myr24
     lsr temp
     out PORTA,temp
     ldi r24, LOW(1000)
     ldi r25, HIGH(1000)
     rcall wait_msec      ;Wait for 1s
     ret

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

one_wire_reset:
    sbi DDRA ,PA4 ; PA4 configured for output
    cbi PORTA ,PA4 ; 480 ?sec reset pulse
    ldi r24 ,low(480)
    ldi r25 ,high(480)
    rcall wait_usec
    cbi DDRA ,PA4 ; PA4 configured for input
    cbi PORTA ,PA4
    ldi r24 ,100 ; wait 100 ?sec for devices
```

```
    ldi r25 ,0 ; to transmit the presence pulse
    rcall wait_usec
    in r24 ,PINA ; sample the line
    push r24
    ldi r24 ,low(380) ; wait for 380 ?sec
    ldi r25 ,high(380)
    rcall wait_usec
    pop r25 ; return 0 if no device was
    clr r24 ; detected or 1 else
    sbrs r25 ,PA4
    ldi r24 ,0x01
    ret

one_wire_receive_bit:
    sbi DDRA ,PA4
    cbi PORTA ,PA4 ; generate time slot
    ldi r24 ,0x02
    ldi r25 ,0x00
    rcall wait_usec
    cbi DDRA ,PA4 ; release the line
    cbi PORTA ,PA4
    ldi r24 ,10 ; wait 10 ?s
    ldi r25 ,0
    rcall wait_usec
    clr r24 ; sample the line
    sbic PINA ,PA4
    ldi r24 ,1
    push r24
    ldi r24 ,49 ; delay 49 ?s to meet the standards
    ldi r25 ,0 ; for a minimum of 60 ?sec time slot
    rcall wait_usec ; and a minimum of 1 ?sec recovery time
    pop r24
    ret

one_wire_transmit_bit:
    push r24 ; save r24
    sbi DDRA ,PA4
    cbi PORTA ,PA4 ; generate time slot
    ldi r24 ,0x02
    ldi r25 ,0x00
    rcall wait_usec
    pop r24 ; output bit
    sbrc r24 ,0
    sbi PORTA ,PA4
    sbrs r24 ,0
    cbi PORTA ,PA4
    ldi r24 ,58 ; wait 58 ?sec for the
    ldi r25 ,0 ; device to sample the line
```

```
    rcall wait_usec
    cbi DDRA ,PA4 ; recovery time
    cbi PORTA ,PA4
    ldi r24 ,0x01
    ldi r25 ,0x00
    rcall wait_usec
    ret

one_wire_receive_byte:
    ldi r27 ,8
    clr r26
    loop_:
    rcall one_wire_receive_bit
    lsr r26
    sbrc r24 ,0
    ldi r24 ,0x80
    or r26 ,r24
    dec r27
    brne loop_
    mov r24 ,r26
    ret

one_wire_transmit_byte:
    mov r26 ,r24
    ldi r27 ,8
    _one_more_:
    clr r24
    sbrc r26 ,0
    ldi r24 ,0x01
    rcall one_wire_transmit_bit
    lsr r26
    dec r27
    brne _one_more_
    ret

write_2_nibbles:
    push r24
    in r25 ,PIND
    andi r25 ,0x0f
    andi r24 ,0xf0
    add r24 ,r25
    out PORTD ,r24
    sbi PORTD ,PD3
    cbi PORTD ,PD3
    pop r24
    swap r24
    andi r24 ,0xf0
    add r24 ,r25
```

```
    out PORTD ,r24
    sbi PORTD ,PD3
    cbi PORTD ,PD3
    ret

lcd_data:
    sbi PORTD ,PD2
    rcall write_2_nibbles
    ldi r24 ,43
    ldi r25 ,0
    rcall wait_usec
    ret

lcd_command:
    cbi PORTD ,PD2
    rcall write_2_nibbles
    ldi r24 ,39
    ldi r25 ,0
    rcall wait_usec
    ret

lcd_init:
    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
    ldi r24 ,0x30
    out PORTD ,r24
    sbi PORTD ,PD3
    cbi PORTD ,PD3
    ldi r24 ,39
    ldi r25 ,0
    rcall wait_usec
    ldi r24 ,0x20
    out PORTD ,r24
    sbi PORTD ,PD3
    cbi PORTD ,PD3
    ldi r24 ,39
    ldi r25 ,0
    rcall wait_usec
    ldi r24 ,0x28
    rcall lcd_command
```

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

scan_row:
     ldi r25,0x08
     back:
     lsl r25
     dec r24
     brne back
     out PORTC, r25
     nop
     nop
     in r24, PINC
     andi r24, 0x0f
     ret

 scan_keypad:
     ldi r24,0x01
     rcall scan_row
     swap r24
     mov r27,r24
     ldi r24,0x02
     rcall scan_row
     add r27,r24
     ldi r24,0x03
     rcall scan_row
     swap r24
     mov r26,r24
     ldi r24,0x04
     rcall scan_row
     add r26,r24
     movw r24,r26
     ret

 scan_keypad_rising_edge:
     mov r22, r24
     rcall scan_keypad
     push r24
     push r25
     mov r24, r22
```

```
    clr r25
    rcall wait_msec
    rcall scan_keypad
    pop r23
    pop r22
    and r24,r22
    and r25,r23
    ldi r26,low(_tmp_) ;r26-r27 -> X
    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
    ret


keypad_to_hex:
   movw r26 ,r24
   ldi r24 ,0x0e
   sbrc r26 ,0
   ret
   ldi r24 ,0x00
   sbrc r26 ,1
   ret
   ldi r24 ,0x0f
   sbrc r26 ,2
   ret
   ldi r24 ,0x0d
   sbrc r26 ,3
   ret
   ldi r24 ,0x07
   sbrc r26 ,4
   ret
   ldi r24 ,0x08
   sbrc r26 ,5
   ret
   ldi r24 ,0x09
   sbrc r26 ,6
   ret
   ldi r24 ,0x0c
   sbrc r26 ,7
   ret
   ldi r24 ,0x04
   sbrc r27 ,0
```

```
        ret
        ldi r24 ,0x05
        sbrc r27 ,1
        ret
        ldi r24 ,0x06
        sbrc r27 ,2
        ret
        ldi r24 ,0x0b
        sbrc r27 ,3
        ret
        ldi r24 ,0x01
        sbrc r27 ,4
        ret
        ldi r24 ,0x02
        sbrc r27 ,5
        ret
        ldi r24 ,0x03
        sbrc r27 ,6
        ret
        ldi r24 ,0x0a
        sbrc r27 ,7
        ret
        ldi r24,0xff
ret
```

Σχόλια:

- Το πρόγραμμά μας αν είναι συνδεδεμένο με θερμόμετρο εμφανίζει τη θερμοκρασία που αυτό δείχνει. Σε αντίθετη περίπτωση δέχεται τη θερμοκρασία από το πληκτρολόγιο.
- Έχουμε μετατρέψει την keypad_to_ascii σε keypad_to_hex που μας επιστρέφει κατευθείαν τις δεκαεξαδικές τιμές που αντιστοιχούν στα πλήκτρα που πατήθηκαν ή το 0xff αν δεν έχει πατηθεί κάτι ακόμα. Αυτές οι τιμές τοποθετούνται στη συνέχεια, με ολίσθηση ή μη, στο αντίστοιχο byte των myr25-myr24.
- Αν πρόκειται για τον αριθμό ‘8000’ τυπώνουμε το μήνυμα ‘NO Device’ κι επανερχόμαστε στην αρχή.
- Στο κομμάτι του υπολογισμού, ελέγχουμε πρώτα αν πρόκειται για αρνητικό αριθμό, εμφανίζουμε το ‘-‘ και στη συνέχεια παίρνουμε το συμπλήρωμα του αριθμού για να το εμφανίσουμε.
- Ελέγχουμε, ακόμη, αν το τελευταίο bit είναι 1 και στην περίπτωση που είναι ενεργοποιούμε ένα flag που μας υποδεικνύει να τυπώσουμε αργότερα και το .5.
- Τέλος, εκτελούμε με λογική δεξιά ολίσθηση το /2 και αμέσως εμφανίζουμε το αποτέλεσμα με τον κλασικό τρόπο μετατροπής του αριθμού από hex σε bcd κι επανερχόμαστε στην αρχή.