Εθνικό **Μ**ετσόβιο **Π**ολυτεχνείο

Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Εργαστήριο Μικροϋπολογιστών, ακ. έτος 2024 - 2025

Λύσεις στην 4η Εργαστηριακή Άσκηση

Όνομα: Ειρήνη	Όνομα: Γεώργιος
Επώνυμο: Σιμιτζή	Επώνυμο: Οικονόμου
ΑΜ: 03121063	ΑΜ: 03121103
Εργαστηριακή Ομάδα: 30	

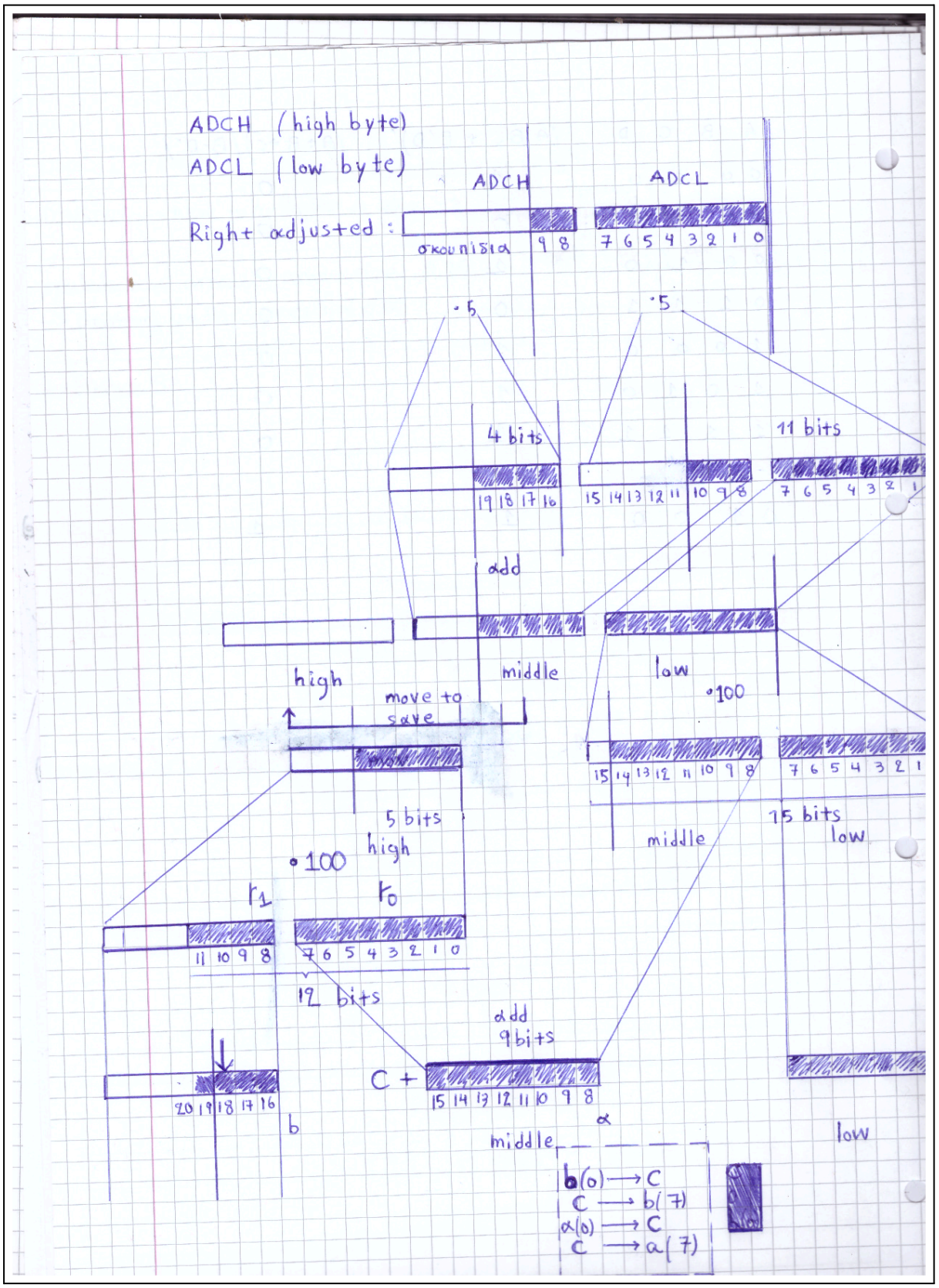
Ζήτημα 4.1

Στο ζήτημα αυτό υλοποιούμε κώδικα Assembly ο οποίος μετατρέπει κάθε 1Sec την τάση που υπάρχει κάθε φορά στην είσοδο A1 του ADC. Η ρουτίνα στην διεύθυνση 0x02A παρουσιάζεται παρακάτω.

```
routine:
    ; ADC value in high_ADC:low_ADC (10 bits)
    ; Vin (ADC*500) value in high_reg:mid_reg:low_reg (19 bits)
    ; Max(low_ADC) * 5 = 1275 = 10011111011 (11 bits)
    ldi Vref, 5     ; Vref = 5V * 100
    lds low_ADC, ADCL
    mul low_ADC, Vref
    mov low_reg, r0
    mov mid_reg, r1

    ; Max(high_ADC) * 5 = 15 = 1111 (4 bits)
    lds high_ADC, ADCH
    andi high_ADC, 0x03 ; Bit mask to keep two bits of ADCH (right-justified)
    mul high_ADC, Vref
    add mid_reg, r0     ; 5 bits

    ; ADC * 5 value in mid_ADC:low_ADC (13 bits)
    ; Max(low_reg) * 100 = 25500 = 1100011110011100 (15 bits)
    ldi temp, 100       ; 100 to scale up
                        ; Two decimal places
    mul low_reg, temp
    mov low_reg, r0
    mov high_reg, mid_reg ; Save previous mi_reg to high_reg
    mov mid_reg, r1

    ; Max(high_reg) * 100 = 12 bits
    mul high_reg, temp
    add mid_reg, r0
    mov high_reg, r1
    ; If there is a carry, the carry flag (C) will be set
    clr temp            ; Rc = 0x00
    adc high_reg, temp  ; Rd = Rd + Rr + C

    clc                 ; Make sure flag C is not affected for the following
                        ; shift routine
```

Στην ρουτίνα φροντίζουμε να υπολογίσουμε την τάση από τον δοσμένο τύπο, δηλαδή, να πολλαπλασιάζουμε τον ADC με 5, έπειτα, με 100 για να κρατήσουμε ακρίβεια δύο δεκαδικών ψηφίων.

Οι 8-bit καταχωρητές επηρεάζονται με τον τρόπο που φαίνεται παρακάτω.



Όπως φαίνεται, το να διαιρούμε με 1024 = 2^10 ισοδυναμεί με ένα logical shift right κατά 10 της τριπλέτας high:middle:low. Ωστόσο, αυτό οδηγεί πρακτικά στο clear του

καταχωρητή low, οπότε, κρατάμε ως ισοδύναμο το logical shift right κατά 2 στο ζεύγος high:middle. Για να βρούμε το πλήθος των εκατοντάδων, δεκάδων και μονάδων σε έναν αριθμό, αρκεί να διαιρέσουμε τον αριθμό διαδοχικά με 100, 10 και 1, στον δεκαδικό σύστημα. Επειδή η εντολή διαίρεσης δεν υποστηρίζεται στην Assembly του AVR, θα χρησιμοποιήσουμε διαδοχικές αφαιρέσεις για να υπολογίσουμε τα αποτελέσματα της διαίρεσης.

```
140     ; This subroutine divides the two 16-bit numbers
141     ; "dd8uH:dd8uL" (dividend) and "dv16uH:dv16uL" (divisor).
142     ; The result is placed in "dres16uH:dres16uL" and the remainder in
143     ; "drem16uH:drem16uL".
144
145     div16u:
146         clr drem16uL                ; clear remainder Low byte
147         sub drem16uH,drem16uH       ; clear remainder High byte and carry
148         ldi dcnt16u,17              ; init loop counter
149
150     d16u_1:
151         rol dd16uL                  ; shift left dividend
152         rol dd16uH
153         dec dcnt16u                 ; decrement counter
154         brne d16u_2                 ; if done
155         ret                         ;    return
156
157     d16u_2:
158         rol drem16uL                ; shift dividend into remainder
159         rol drem16uH
160         sub drem16uL,dv16uL         ; remainder = remainder - divisor
161         sbc drem16uH,dv16uH
162         brcc d16u_3                 ; if result negative
163         add drem16uL,dv16uL         ;    restore remainder
164         adc drem16uH,dv16uH
165         clc                         ; clear carry to be shifted into result
166         rjmp d16u_1                 ; else
167
168     d16u_3:
169         sec                         ; set carry to be shifted into result
170         rjmp d16u_1
```

Ζήτημα 4.2

Στο ζήτημα αυτό υλοποιούμε κώδικα C παρόμοιο με τον προηγούμενο, που θα εκτυπώνει την τάση στην LCD οθόνη με ακρίβεια δύο δεκαδικών ψηφίων. Τροποποιούμε κατάλληλα τις συναρτήσεις της LCD οθόνης με τα καίρια σημεία παρακάτω.

```
6   void write_2_nibbles(uint8_t input){
7       uint8_t temp= input;
8       PORTD = (PIND & 0x0f) + (temp & 0xf0); //LCD Data High Bytes
9
10      PORTD|=0x08; //enable pulse
11      _delay_us(2);
12      PORTD&=~(0x08);
13
14      input=(input<<4)|(input>>4);
15      PORTD = (PIND & 0x0f) + (input & 0xf0); //LCD Data Low Bytes
16
17     PORTD|=0x08;  //enable pulse
18      _delay_us(2);
19      PORTD&=~(0x08);
```

```
72      lcd_init();
73      uint32_t adc_value;
74      int output;
75      while (1) {
76          ADCSRA|= (1<<ADSC);     //Start ADC
77          while((ADCSRA & 0x40)==0x40){}   //Wait until ADC is finished
78          adc_value=ADC;
79          output=(adc_value*500)>>10; //VOLTAGE to V - maximum value is 500mV and shift 10 and divide with 1024
80          lcd_clear_display();
81       //break down output to digits
82      int digits[3];
83      digits[2]=output%10;
84      output/=10;
85      digits[1]=output%10;
86      output/=10;
87      digits[0]=output%10;
88      lcd_data(digits[0] + 0x30); //convert digits to ascii characters
89      lcd_data('.');
90      lcd_data(digits[1] + 0x30);
91      lcd_data(digits[2] + 0x30);
92      lcd_data('V');
93
94          _delay_ms(1000);}
```

Ζήτημα 4.3

Στο ζήτημα αυτό υλοποιούμε κώδικα C για την επιτήρηση ενός χώρου όπου υπάρχει αυξημένος κίνδυνος ύπαρξης μονοξειδίου του άνθρακα (CO). Συγκεκριμένα, αν οποιαδήποτε στιγμή η συγκέντρωση του CO ξεπεράσει τα 70ppm ανιχνεύεται το αέριο (GAS DETECTED).

Για να υπολογίσουμε την τάση στην έξοδο του αισθητήρα ULPSM-CO 968-001, χρησιμοποιούμε τον δοσμένο τύπο. Σύμφωνα με τον τύπο αυτό, για να ανιχνεύσουμε τιμές μεγαλύτερες των 70 ppm, αρκεί να ελέγχουμε αν το αποτέλεσμα του ADC είναι μεγαλύτερο από την τιμή 205. Παρακάτω παρουσιάζονται τα καίρια κομμάτια του κώδικα.

```
132         while (1)
133         {
134             _delay_ms(100);
135             if (gas_detected)
136             {
137                 PORTB = 63;
138                 lcd_clear_display();
139                 char signal[] = "GAS DETECTED";
140                 for (int i=0; signal[i]!=0; i++)
141                 {
142                     lcd_data(signal[i]);
143                 }
144                 while (gas_detected)
145                 {
146                     led_data(gas_value);
147                     _delay_ms(100);
148                     ADCSRA |= (1<<ADSC);
149                 }
150             }
151             else
152             {
153                 PORTB = 0;
154                 lcd_clear_display();
155                 char signal[] = "CLEAR";
156                 for (int i=0; signal[i]!=0; i++)
157                 {
158                     lcd_data(signal[i]);
159                 }
160                 while (!gas_detected)
161                 {
162                     led_data(gas_value);
163                     _delay_ms(100);
164                     ADCSRA |= (1<<ADSC);
165                 }
166             }
```