Εργαστήριο Μικροεπεξεργαστών

3^η Εργαστηριακή άσκηση

Γέροντας Αλέξανδρος 321 - 2015029

Συμπληρώνουμε το τμήμα δηλώσεων ως εξής:

```
.dseg
 2
     .ORG 0x0100
      arrayA: .byte 10
       arrayB: .byte 10
 5
       arrayC: .byte 10
 6
 7
8
       .ORG 0x0000
9
10
   start:
11 Idi r16, 0b11101010; set PCn for STB and acknowledge
12
        out DDRC, r16
13
14
       ldi r16, 0b00000000 ; clear portC
        out PORTC, r16
15
16
17
        out DDRB, r16 ; PBn will be used for input
18
        out DDRD, r16; PDn will be used for input
19
20
        ldi r18, 30 ; initialize r18 for delay
21
22
        ldi r27, high(arrayA); initialize X
        ldi r26, low(arrayA)
23
24
25
        ldi r29, high(arrayB) ; initialize Y
        ldi r28, low(arrayB)
27
28
        ldi r31, high(arrayC) ; initialize Z
        ldi r30, low(arrayC)
29
```

Όπου τα array A,B,C χρησιμοποιούνται για να αποθηκεύσουν τα δεδομένα από τις συσκευές Σ1, Σ2, Σ3 αντίστοιχα και οι καταχωρητές X,Y,Z (οι οποίοι αποτελούνται από ζευγάρια καταχωρητών) χρησιμοποιούνται για τον προσδιορισμό της θέσης των πινάκων. Το PCn θα χρησιμοποιηθεί για το STB και το acknowledge. Για αυτό θέτουμε στις θέσεις PC0, PC2, PC4 (με την χρήση του καταχωρητή DDRC) την τιμή 0 για το STB και στις θέσεις PC1, PC2, PC3 την τιμή 1 για το acknowledge. Στη συνέχεια μηδενίζουμε το portC και θέτουμε τα PCn και PDn την τιμή 0 καθώς θα χρησιμοποιηθούν για είσοδο δεδομένων.

Στη συνέχεια το πρόγραμμα προχωράει στον βρόγχο polling.

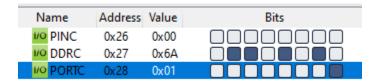
```
31
    polling:
32
        sbic PORTC, 0 ; if portC[0] = 1
        rcall devA ; read input from devA
33
34
        sbic PORTC, 2 ; if portC[2] = 1
35
        rcall devB ; read input from devB
36
37
        sbic PORTC, 4 ; if portC[4] = 1
38
39
        rcall devC ; read input from devC
40
41
        rjmp polling ; jmp to polling and check the devices again
```

Ο βρόγχος ελέγχει αν τα bit στις θέσεις PortC[0], PortC[2], PortC[4] έχουν πάρει την τιμή 1 και αν ναι καλεί την αντίστοιχη ρουτίνα. Στο τέλος δεν τερματίζει απλά καλεί ξανά τον εαυτό του.

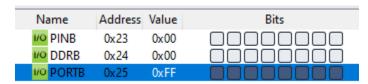
Η ρουτίνα για την 1^η συσκευή έχει ως εξής:

- Αρχικά διαβάζει τα bit από την συσκευή.
- Τα αποθηκεύει στον πίνακα arrayA και αυξάνει τον μετρητή X κατά 1.
- Στέλνει το acknowledge στην συσκευή.
- Γίνεται ένα delay για περίπου 10 μsec.
- Γίνεται έξοδος από την ρουτίνα.

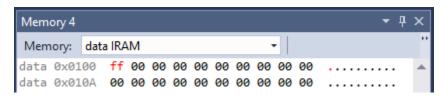
Για δοκιμή αρχικά δίνουμε στο portC[0] την τιμή 1 (στο simulator)



Βάζουμε στο portB μερικά δεδομένα π.χ την τιμή ff στο δυαδικό



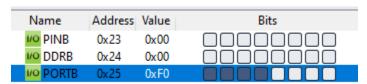
Βλέπουμε ότι στην 1^η θέση του πίνακα arrayA μπήκε η τιμή ff,



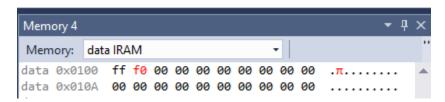
και ότι στο portC[1] μπήκε η τιμή 1.

Name	Address	Value	Bits
No binc	0x26	0x00	
₩ DDRC	0x27	0x6A	
WO PORTC	0x28	0x02	

Βάζουμε αυτή τη φορά στο portB την τιμή f0.



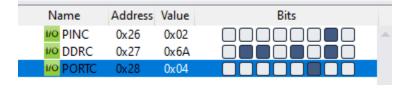
Βλέπουμε πως μπήκε στην 2^n θέση του πίνακα δίπλα στο 1^o στοιχείο.



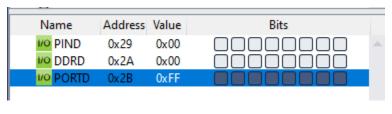
Αντιστοίχως η ρουτίνα για την 2^n συσκευή:

Διαβάζονται τα δεδομένα από τον καταχωρητή PortD στον καταχωρητή r17, και γίνεται and στον καταχωρητή r17 έτσι ώστε να κοπούν τα bit τα οποία δεν θέλουμε. Ακολουθούμε την ίδια διαδικασία όπως στην 1^η συσκευή:

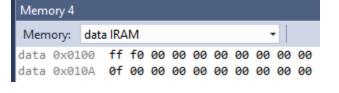
Θέτουμε στο portC[3] την τιμή 1:



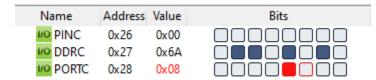
Δίνουμε στο PortD την τιμή FF:



Βλέπουμε πως στον arrayB μπήκε η τιμή 0f καθώς τα πρώτα bit έχουν κοπεί



Και το acknowledge μπήκε στην θέση 4

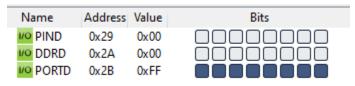


Η ρουτίνα για την 3^η συσκευή:

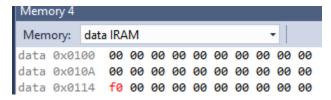
Δίνουμε στο portC[4] την τιμή 1:

Name	Address	Value	Bits
₩O PINC	0x26	0x00	0000000
₩ DDRC	0x27	0х6А	
I/O PORTC	0x28	0x10	

Και στο portD την τιμή FF:



Καθώς κόβονται τα τελευταία 4 bit στο arrayC γίνεται εισαγωγή της τιμής f0 στο arrayC:



Και το acknowledge μπαίνει στην 5^{η} θέση του portC



Τέλος στην ρουτίνα delay γίνεται ένα nop και μειώνεται η τιμή r18 μέχρι να μηδενιστεί. Πριν την επιστροφή από την ρουτίνα δίνουμε στον καταχωρητή r18 την τιμή r18 έτσι ώστε να είναι έτοιμος για την επόμενη κλήση της ρουτίνας.

```
delay:
    nop
    dec r18    ; r18 = r18 - 1
    breq return ; if r18 == 0 return
    rjmp delay ; else goto delay

return:
    ldi r18, 30 ; before return set r18 = 30
    ret
```