

Ενσωματωμένα Συστήματα

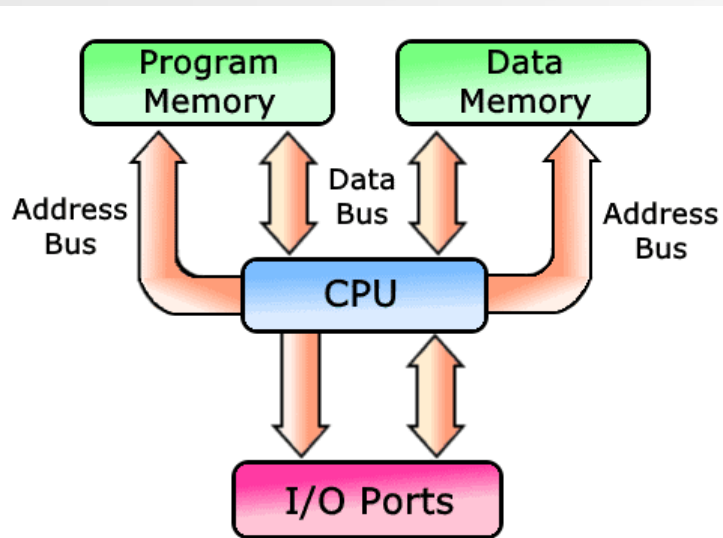
(6^ο εξάμηνο)

02-Digital Ports

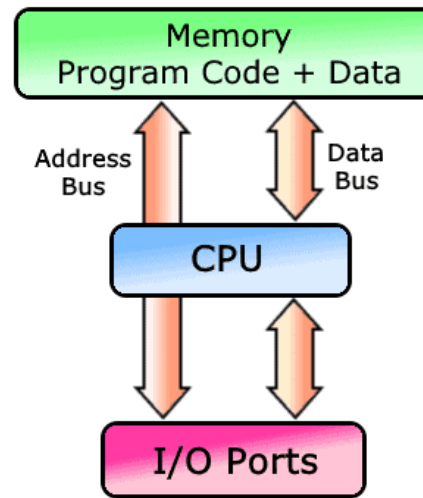
Διδάσκουσα: Παπαδοπούλου Μαρία
Επίκουρη Καθηγήτρια

Θεσσαλονίκη 2025

Κατάταξη ανάλογα με την αρχιτεκτονική του μΕ

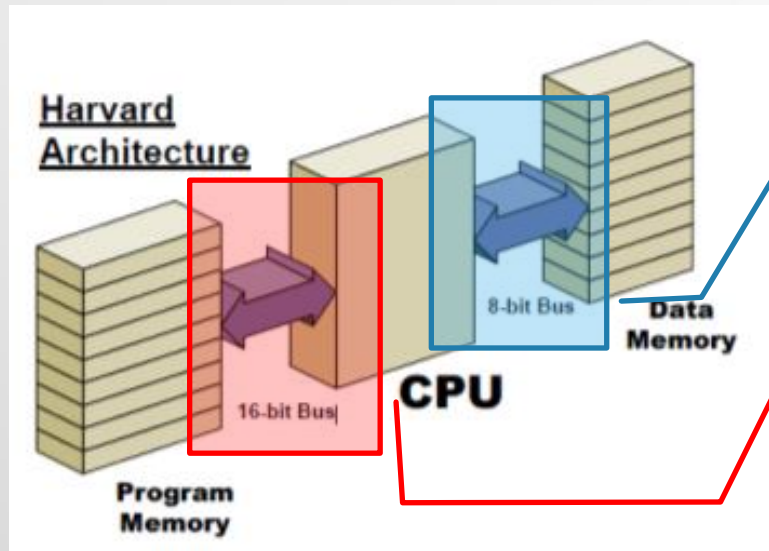


Harvard Architecture



Von Neumann Architecture

Αρχιτεκτονική του $\mu\text{E PIC18F4550}$



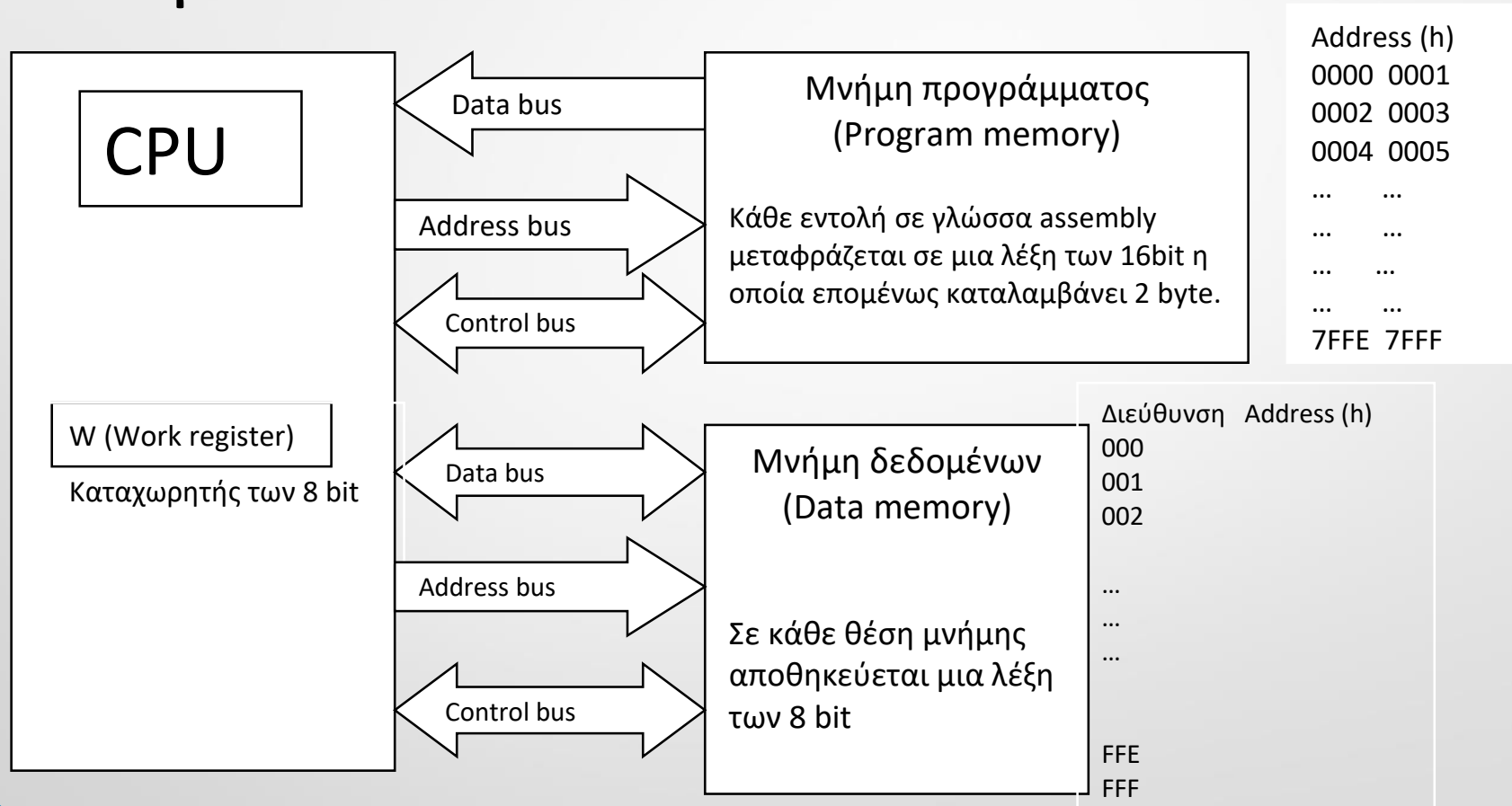
Ο δίαυλος δεδομένων προς τη μνήμη δεδομένων είναι των 8 bit

Ο δίαυλος δεδομένων προς τη μνήμη προγράμματος είναι 16 bit.

Επομένως μεταφέρονται συγχρόνως από τη μνήμη προγράμματος 16 bit.

Οι εντολές σε γλώσσα μηχανής είναι κατά κανόνα λέξεις των 16 bit

Σύνδεση ΚΜΕ (CPU) με μνήμες προγράμματος και δεδομένων



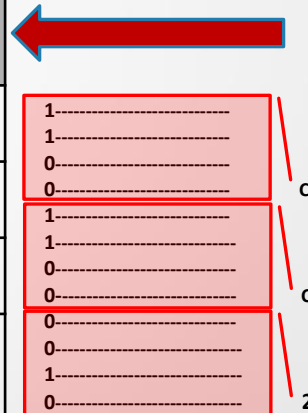
Στην αρίθμηση των θέσεων μνήμης χρησιμοποιούμε το δεκαεξαδικό αριθμητικό σύστημα!

Μπορούμε να τοποθετήσουμε περιεχόμενο σε μια θέση μνήμης ή να διαβάσουμε το περιεχόμενο από μια θέση μνήμης (Write/ Read)

Data Bus
8 bit
(Δίαυλος δεδομένων)

Περιεχόμενα θέσεων μνήμης (b)	Διευθύνσεις στο δεκαεξαδικό αριθμητικό σύστημα (h)	Διευθύνσεις στο δυαδικό αριθμητικό σύστημα (b)	Διευθύνσεις στο δεκαδικό αριθμητικό σύστημα (d)
0101 0101	000	0000 0000 0000	0
1111 1111	001	0000 0000 0001	1
1000 1110	002	0000 0000 0010	2
.	.	.	.
1010 1111	0FE	0000 1111 1110	254
1001 1010	0FF	0000 1111 1111	255
1111 0000	100	0001 0000 0000	256
1111 1000	101	0001 0000 0001	257

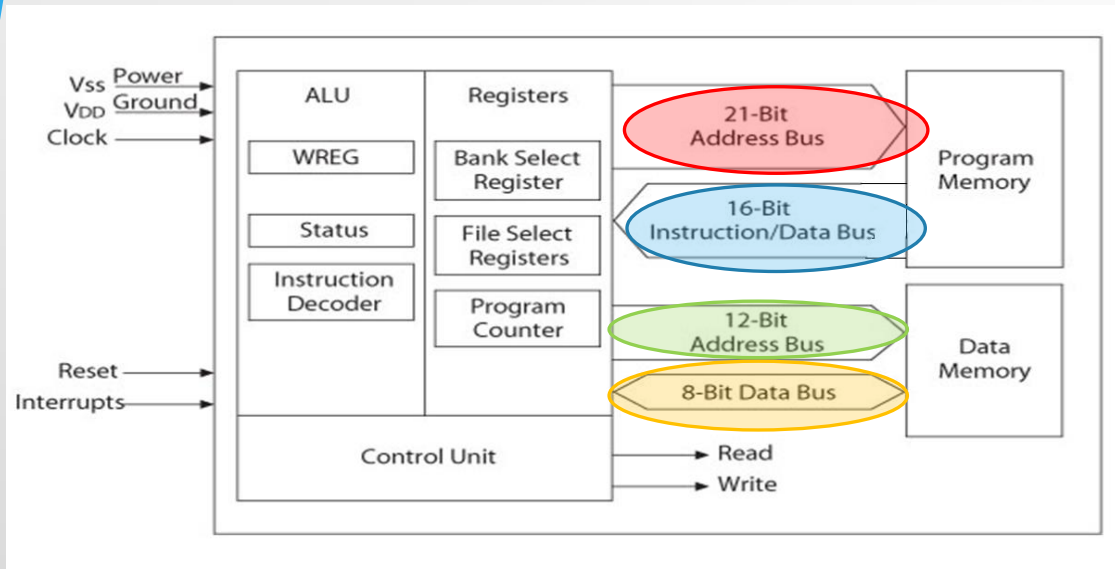
Address Bus
12 bit (Δίαυλος διευθύνσεων)



Διεύθυνση CC_{2h}
(Αποδίδεται από την ΚΜΕ)

12 αγωγοί για επιλογή της διεύθυνσης από την οποία θα γίνει η ανάγνωση ή εγγραφή δεδομένων

PIC18F – MCU and Memory



Data Memory: 2 K (2^{11})

Περιοχή διευθύνσεων: 000 to 7FF_h

Καταχωρητές των 8-bit

Program Memory: 32 K (2^{15})

Περιοχή διευθύνσεων: 000000 to 007FFF_h

Καταχωρητές των 16-bit

Ο δίαυλος διευθύνσεων των 21 bit θα μπορούσε να έχει πρόσβαση σε

2 MB
 2^{21}

Η μνήμη προγράμματος είναι οργανωμένη σε λέξεις των 16 bit

Ο δίαυλος διευθύνσεων των 12 bit θα μπορούσε να έχει πρόσβαση σε

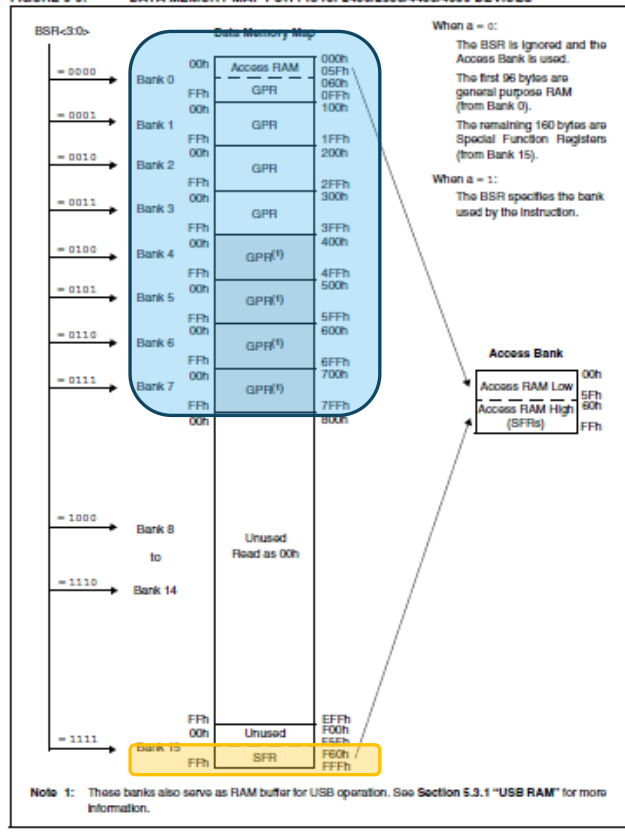
4 KB
 2^{12}

Η μνήμη δεδομένων είναι οργανωμένη σε λέξεις των 8 bit

Μνήμη δεδομένων του μΕ PIC18F4550

PIC18F2455/2550/4455/4550

FIGURE 5-5: DATA MEMORY MAP FOR PIC18F2455/2550/4455/4550 DEVICES



- Η μνήμη δεδομένων του μΕ είναι από τη διεύθυνση $(000)_h$ έως $(7FF)_h$. Δηλαδή από 0000 0000 0000 έως 111 1111 1111

Το πλήθος των διευθύνσεων είναι $2^{11} = 2^1 \times 2^{10} = 2 \times 1024 = 2 \text{ K}$
Σε κάθε θέση μνήμης αποθηκεύεται ένα byte (8 bits).

Η μνήμη δεδομένων του μΕ PIC 18F4550 είναι 2 K

Προσοχή!

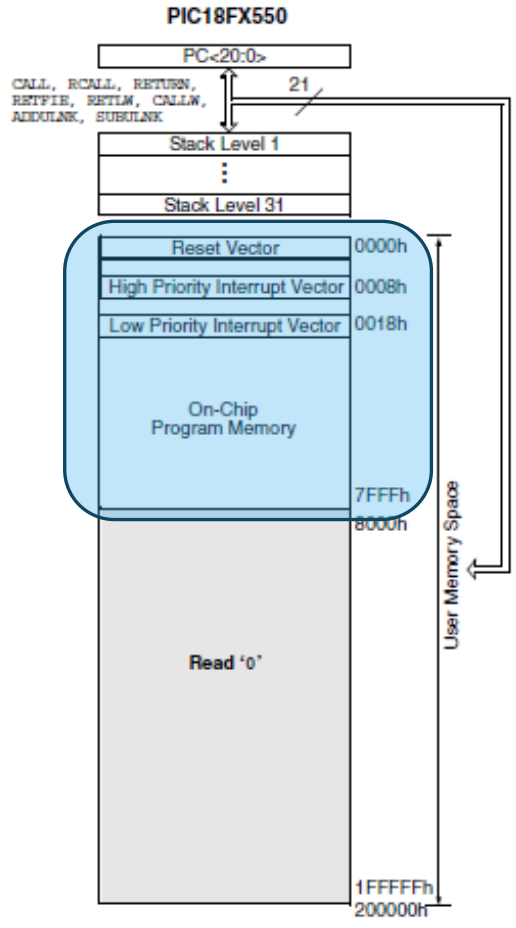
$2^{10} = 1024 = 1 \text{ K}$ για τις μνήμες

$2^{20} = 1 \text{ K} \times 1 \text{ K} = 1 \text{ M}$ (1 048 576 θέσεις μνήμης)

$2^{30} = 1 \text{ K} \times 1 \text{ K} \times 1 \text{ K} = 1 \text{ G}$ (1 073 741 824 θέσεις μνήμης)

- **Προσοχή**, χρησιμοποιούνται και οι θέσεις μνήμης $(F60)_h$ έως $(FFF)_h$.
- Αυτές οι θέσεις μνήμης χρησιμοποιούνται για τους Special Function Registers (SFR)
- Πρόκειται για θέσεις μνήμης που έχουν ειδική λειτουργία

Μνήμη προγράμματος με PIC18F4550



Προσοχή:

- Οι τιμές στο δίαυλο διευθύνσεων της μνήμης προγράμματος τοποθετούνται από τον **Program Counter**
- Ο **Program Counter** είναι ένας καταχωρητής τοποθετημένος μέσα στην ΚΜΕ (CPU)

Ο δίαυλος διευθύνσεων περιλαμβάνει 21 αγωγούς.
Θα μπορούσε επομένως να έχει πρόσβαση σε 2^{21} θέσεις μνήμης, δηλαδή $2^1 \times 2^{10} \times 2^{10} = 2 \times 1 \text{ K} \times 1 \text{ K} = 2 \text{ M}$ θέσεις μνήμης

Όπως φαίνεται στο σχήμα, το υλοποιημένο κομμάτι της μνήμης είναι από τη διεύθυνση 0000_h έως τη διεύθυνση 7FFF_h

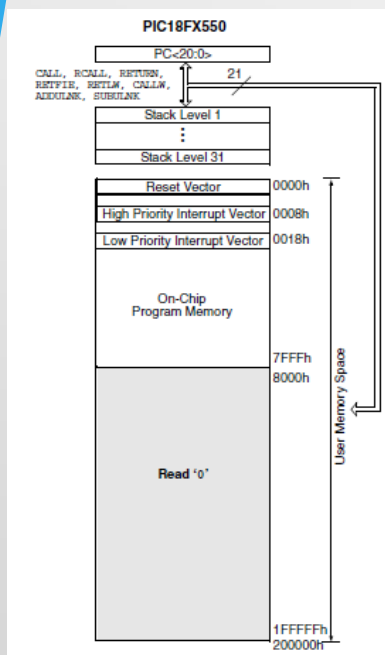
Δηλαδή από τη διεύθυνση 0000 0000 0000 0000
Έως τη διεύθυνση 111 1111 1111 1111

Η συνολική μνήμη προγράμματος είναι $2^{15} = 2^5 \times 2^{10} = 32 \text{ K}$

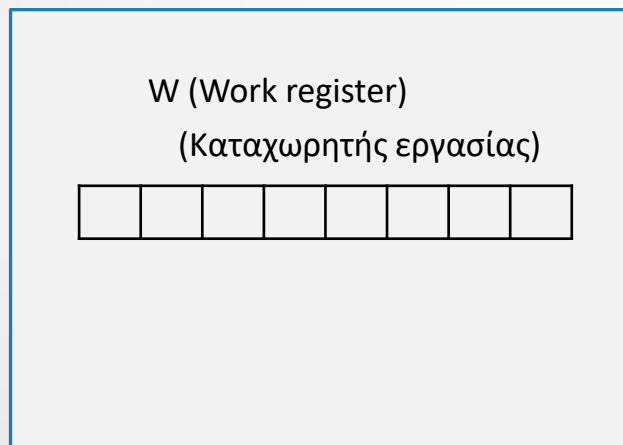
Δημιουργία προγράμματος σε γλώσσα Assembly (1/6)

Να γραφεί πρόγραμμα με το οποίο στη θέση μνήμης 060h τοποθετείται η τιμή 2Ah και στη θέση μνήμης 061h η τιμή FFh. Στη συνέχεια, το περιεχόμενο της θέσης μνήμης 060h μεταφέρεται στη θέση μνήμης 061h. Μετά την εκτέλεση των παραπάνω το πρόγραμμα να εκτελεί έναν ατέρμονα (χωρίς τέλος) βρόχο χωρίς να εκτελεί κάποια άλλη εντολή. Το πρόγραμμα να τοποθετηθεί στη μνήμη προγράμματος από τη διεύθυνση 0800h και μετά.

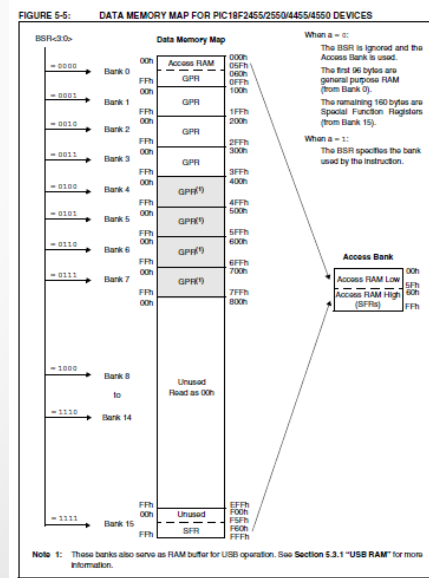
Δημιουργία προγράμματος σε γλώσσα Assembly (2/6)



CPU (Central Processing Unit) ΚΜΕ (κεντρική Μονάδα Επεξεργασίας)



PIC18F2455/2550/4455/4550



Μνήμη προγράμματος

Το πρόγραμμα θα γραφεί από τη διεύθυνση 0800h και μετά. (Πώς θα το τοποθετήσουμε σε αυτήν τη θέση;

Μνήμη δεδομένων

2A_h → Να γίνει περιεχόμενο της θέσης μνήμης 060_h
 FF_h → Να γίνει περιεχόμενο της θέσης μνήμης 061_h
 Το περιεχόμενο της θέσης μνήμης 060_h να μεταφερθεί στο περιεχόμενο της θέσης μνήμης 061_h

Δημιουργία προγράμματος σε γλώσσα Assembly (3/6)

Μνήμη προγράμματος

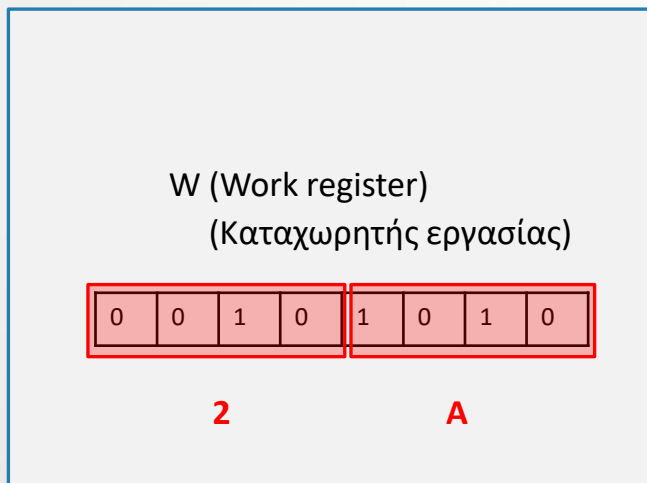
16 bit Instructions

Εντολές των 16 bit

Reset Vector	0000h
	0002h
	0004h
·	
·	
·	
	0800h
	0802h
	0804h
	0806h
	0808h
·	
·	
·	
	7FFFh

Το πρόγραμμα
θα γραφεί από
τη διεύθυνση
0800h και μετά

CPU (Central Processing Unit) ΚΜΕ (κεντρική Μονάδα Επεξεργασίας)



Μνήμη δεδομένων

Οργανωμένη σε
λέξεις των 8 bit

	000h
	001h
·	
·	
·	
	060h
	061h

Μνήμη δεδομένων

Η τοποθέτηση τιμών στις θέσεις της μνήμης δεδομένων και η μεταφορά τιμών από και προς τις θέσεις της μνήμης δεδομένων γίνεται δια μέσου του καταχωρητή εργασίας W

Δημιουργία προγράμματος σε γλώσσα Assembly (4/6)

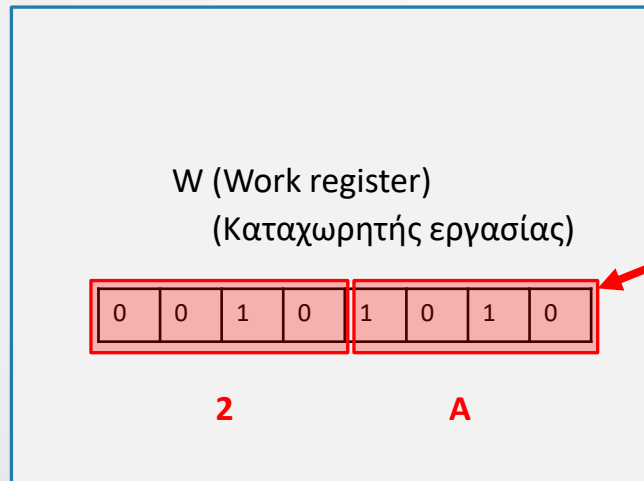
Μνήμη προγράμματος

16 bit Instructions
Εντολές των 16 bit

Reset Vector	0000h
	0002h
	0004h
·	
·	
·	
	0800h
	0802h
	0804h
	0806h
	0808h
·	
·	
·	
	7FFFh

Το πρόγραμμα
θα γραφεί από
τη διεύθυνση
0800h και μετά

CPU (Central Processing Unit) ΚΜΕ (κεντρική Μονάδα Επεξεργασίας)



Μνήμη δεδομένων Οργανωμένη σε λέξεις των 8 bit

	000h
	001h
·	
·	
·	
0010 1010	060h
	061h

Μνήμη δεδομένων

Η τοποθέτηση τιμών στις θέσεις της μνήμης δεδομένων και η μεταφορά τιμών από και προς τις θέσεις της μνήμης δεδομένων γίνεται δια μέσου του καταχωρητή εργασίας W

Δημιουργία προγράμματος σε γλώσσα Assembly (5/6)

Μνήμη προγράμματος

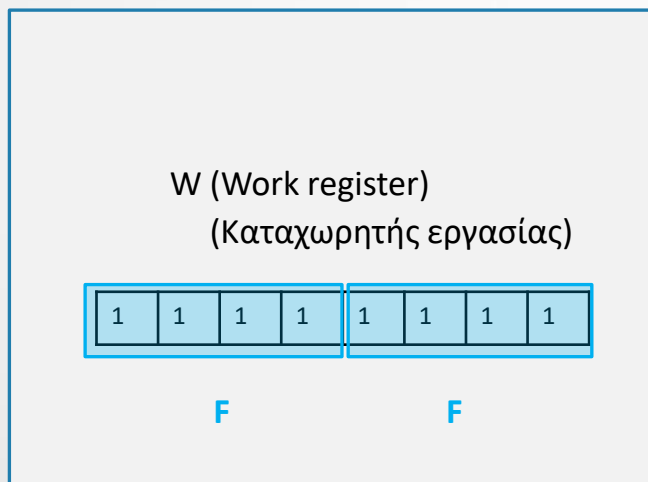
16 bit Instructions

Εντολές των 16 bit

Reset Vector	0000h
	0002h
	0004h
·	
·	
·	
	0800h
	0802h
	0804h
	0806h
	0808h
·	
·	
·	
	7FFFh

CPU (Central Processing Unit)

ΚΜΕ (κεντρική Μονάδα Επεξεργασίας)



Το πρόγραμμα
θα γραφεί από
τη διεύθυνση
0800h και μετά

Μνήμη δεδομένων

Οργανωμένη σε
λέξεις των 8 bit

	000h
	001h
·	
·	
·	
0010 1010	060h
	061h

Μνήμη δεδομένων

Η τοποθέτηση τιμών στις θέσεις της μνήμης δεδομένων και η μεταφορά τιμών από και προς τις θέσεις της μνήμης δεδομένων γίνεται δια μέσου του καταχωρητή εργασίας W

Δημιουργία προγράμματος σε γλώσσα Assembly (6/6)

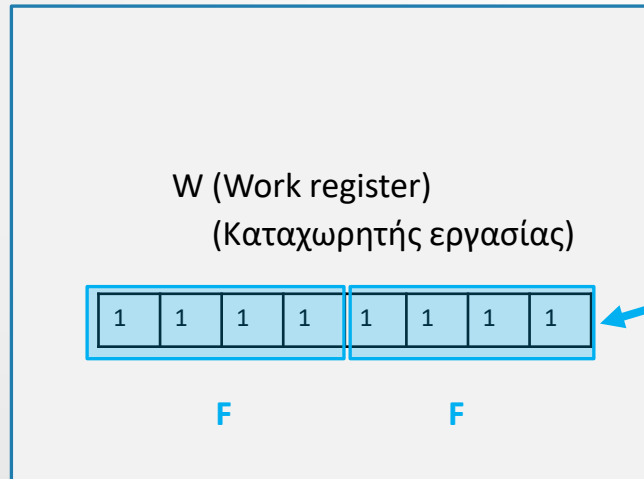
Μνήμη προγράμματος

16 bit Instructions
Εντολές των 16 bit

Reset Vector	0000h
	0002h
	0004h
·	
·	
·	
	0800h
	0802h
	0804h
	0806h
	0808h
·	
·	
·	
	7FFFh

Το πρόγραμμα
θα γραφεί από
τη διεύθυνση
0800h και μετά

CPU (Central Processing Unit) ΚΜΕ (κεντρική Μονάδα Επεξεργασίας)



Μνήμη δεδομένων

Οργανωμένη σε
λέξεις των 8 bit

	000h
	001h
·	
·	
·	
0010 1010	060h
1111 1111	061h

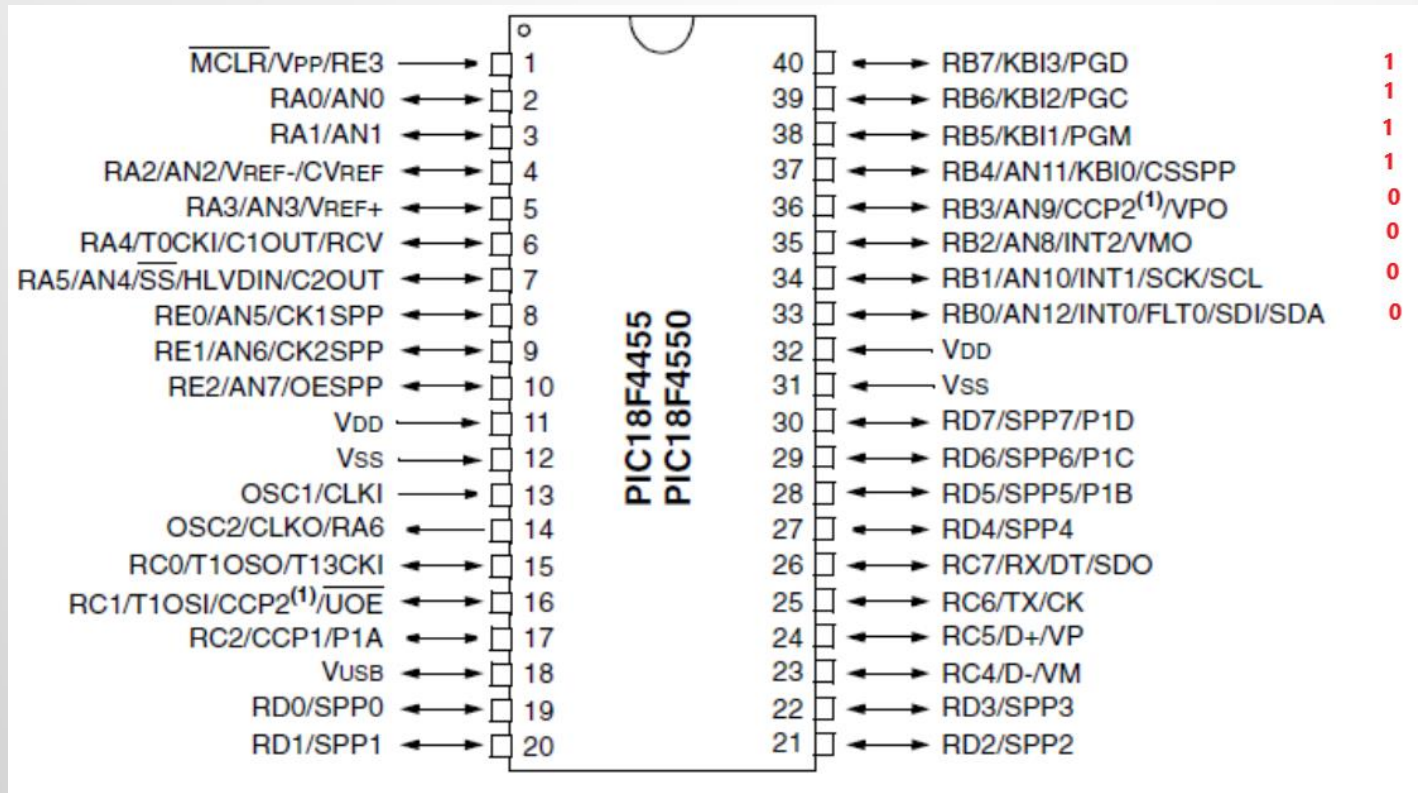
Μνήμη δεδομένων

Η τοποθέτηση τιμών στις θέσεις της μνήμης δεδομένων και η μεταφορά τιμών από και προς τις θέσεις της μνήμης δεδομένων γίνεται δια μέσου του καταχωρητή εργασίας W

Άσκηση 1

Χρήση της πόρτας B ως έξοδος

Να γραφεί πρόγραμμα που να εμφανίζει στην πόρτα B του μικροελεγκτή PIC18F4550 την τιμή 1111 0000b (0xF0)



Παράλληλη πόρτα B (1/2)

Καταχωρητής κατεύθυνσης της πόρτας B

PORTB Direction Register, TRISB: Είναι ένας Special Function Register.
Βρίσκεται στη διεύθυνση F93h. Υπάρχει αντίγραφο του στη διεύθυνση 093h

1	1	0	0	0	0	0	1
---	---	---	---	---	---	---	---

Καταχωρητής δεδομένων της πόρτας B

PORTB Data Register, PORTB: Είναι ένας Special Function Register
Βρίσκεται στη διεύθυνση F81h. Υπάρχει αντίγραφο του στη διεύθυνση 081h

--	--	--	--	--	--	--	--

Ακροδέκτες
πόρτας B

RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0

Είσοδος Είσοδος Έξοδος Έξοδος Έξοδος Έξοδος Έξοδος Είσοδος

Special Function Registers

F97h	—
F96h	TRISE ⁽³⁾
F95h	TRISD ⁽³⁾
F94h	TRISC
F93h	TRISB
F92h	TRISA
F91h	— ⁽²⁾

F87h	—
F84h	PORTE
F83h	PORTD ⁽³⁾
F82h	PORTC
F81h	PORTB
F80h	PORTA

Οι τιμές που τοποθετούνται στον καταχωρητή κατεύθυνσης της πόρτας B καθορίζουν ποιοι ακροδέκτες θα είναι είσοδοι και ποιοι θα είναι έξοδοι:

1 → ο αντίστοιχος ακροδέκτης της πόρτας B θα είναι **είσοδος**

0 → ο αντίστοιχος ακροδέκτης της πόρτας B θα είναι **έξοδος**

Παράλληλη πόρτα B (2/2)

Να γραφούν οι εντολές με τις οποίες οι ακροδέκτες RB7, RB6, RB5, RB4 της πόρτας B γίνονται είσοδοι και οι ακροδέκτες RB3, RB2, RB1 και RB0 της πόρτας B γίνονται έξοδοι.

set_tris_b(0xF0);

- Τοποθετώντας την κατάλληλη τιμή στον καταχωρητή κατεύθυνσης της πόρτας B καθορίζουμε ποιοι ακροδέκτες της πόρτας B θα είναι είσοδοι και ποιοι θα είναι έξοδοι.
- Χρησιμοποιούμε το όνομα *TRISB* για τον καταχωρητή κατεύθυνσης της πόρτας B και *PORT B* για τον καταχωρητή δεδομένων της πόρτας B

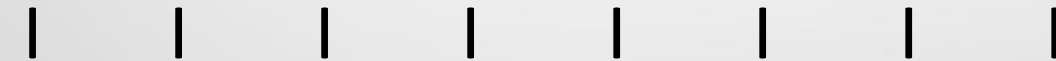
PORTB Direction Register, TRISB , καταχωρητής κατεύθυνσης της πόρτας B

1	1	1	1	0	0	0	0
---	---	---	---	---	---	---	---

PORTB Data Register, PORTB, καταχωρητής δεδομένων της πόρτας B

--	--	--	--	--	--	--	--

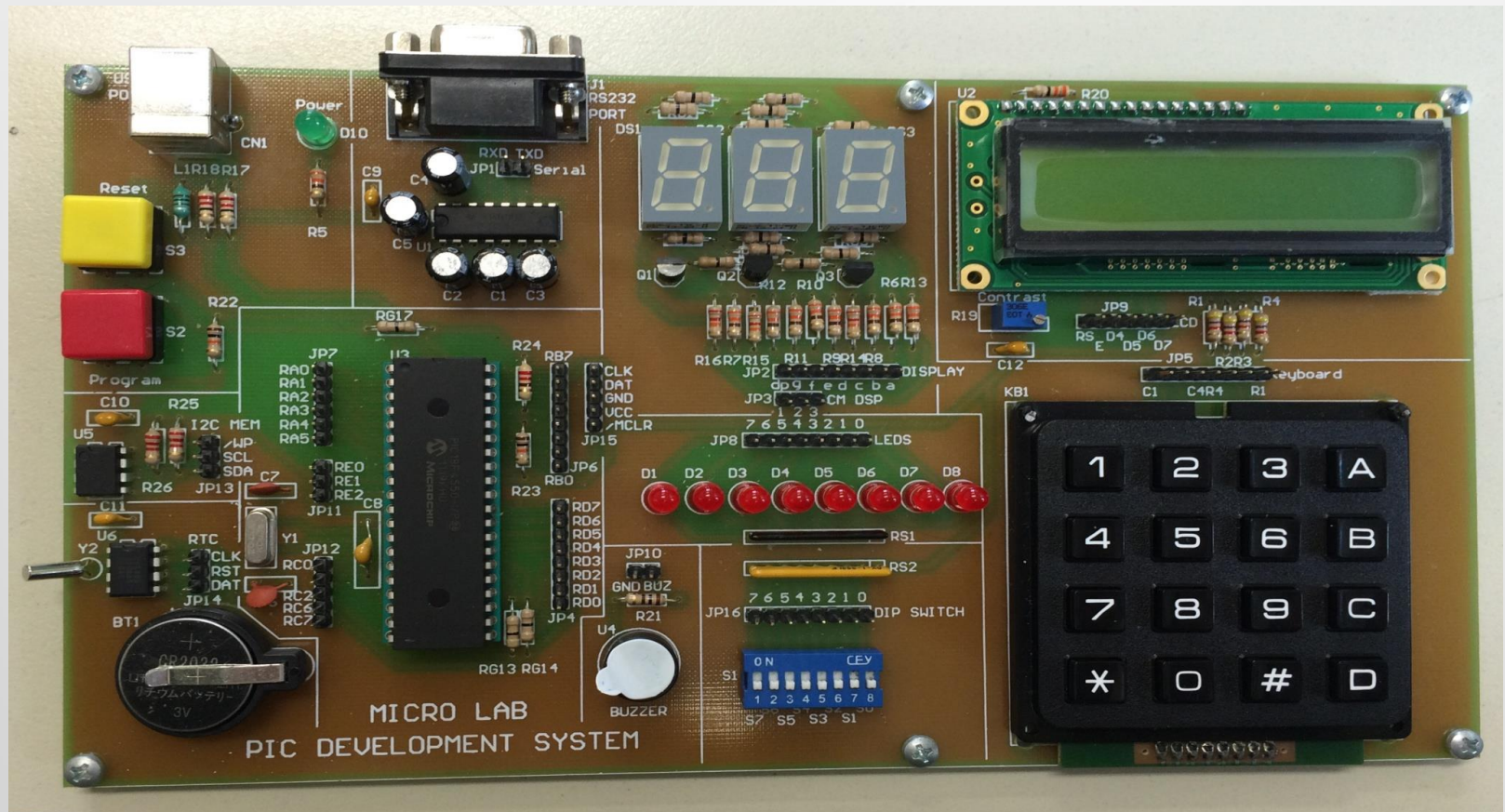
Ακροδέκτες
πόρτας B



RB7 RB6 RB5 RB4 RB3 RB2 RB1 RB0

Είσοδος Είσοδος Είσοδος Είσοδος Έξοδος Έξοδος Έξοδος Έξοδος

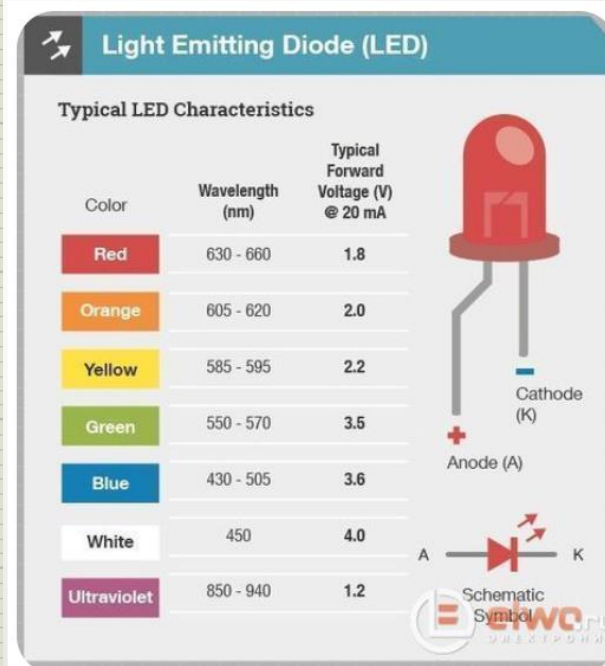
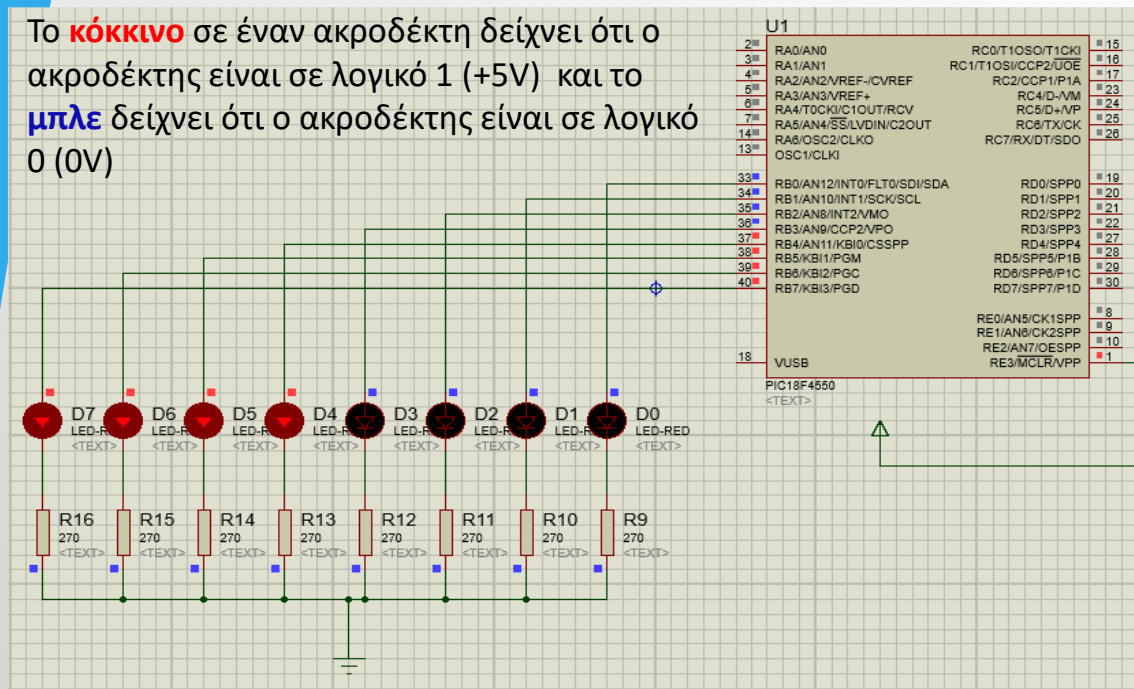
Πλακέτα ανάπτυξης εφαρμογών του εργαστηρίου «Ενσωματωμένα Συστήματα»



Άσκηση 00α

Χρήση της πόρτας B ως πόρτας εξόδου

Το **κόκκινο** σε έναν ακροδέκτη δείχνει ότι ο ακροδέκτης είναι σε λογικό 1 (+5V) και το **μπλε** δείχνει ότι ο ακροδέκτης είναι σε λογικό 0 (0V)



Σε γλώσσα C το πρόγραμμα αποτελείται από δύο εντολές

`set_tris_b(0x00);` Να γίνουν όλοι οι ακροδέκτες της πόρτας B έξοδοι

`PORTB = 0xF0;` Ο καταχωρητής δεδομένων της πόρτας B να πάρει την τιμή 1111 0000 (πρέπει προηγουμένως να δηλωθεί η θέση του στη μνήμη με μια οδηγία)

Αρχεία του project της άσκησης

Περιεχόμενα φακέλου
όπου θα αναπτυχθεί το
Project της άσκησης

askisi-1a.c

- Το αρχείο σε γλώσσα C (το αρχείο προγράμματος που δημιουργήσαμε)

askisi-1a.hex

- Το αποτέλεσμα της μετάφρασης του προγράμματος από γλώσσα C σε γλώσσα μηχανής
- Είναι το αρχείο που θα φορτωθεί στο μικροελεγκτή
- Ανοίγει με το notepad

18F4550.h

- Αρχείο με πληροφορίες για το μικροελεγκτή που χρησιμοποιούμε
- Ανοίγει με notepad

main.h

- Αρχείο με τις αρχικές ρυθμίσεις του μικροελεγκτή
- Ανοίγει με notepad

Pre-processor directives (CCS C compiler)

include

byte

```
#include <main.h> // Το αρχείο <main.h> περιέχει αρχικές ρυθμίσεις  
// Πρέπει να τοποθετηθεί οπωσδήποτε στον ίδιο  
// φάκελο στον οποίο θα αναπτύξετε το project σας.  
// Μπορείτε να το ανοίξετε με το notepad ή το word pad  
// και να δείτε το περιεχόμενο του
```

```
#byte PORTB=0xF81 //F81 είναι η θέση του καταχωρητή δεδομένων της  
// πόρτας B στη μνήμη του μικροελεγκτή  
// Δίνουμε στη διεύθυνση 0xF81 το όνομα PORTB
```

#BYTE

Syntax: #BYTE *id* = *x*

Elements: *id* is a valid C identifier,
x is a C variable or a constant

Purpose: If the *id* is already known as a C variable then this will locate the variable at address *x*. In this case the variable type does not change from the original definition. If the *id* is not known a new C variable is created and placed at address *x* with the type int (8 bit)

Warning: In both cases memory at *x* is not exclusive to this variable. Other variables may be located at the same location. In fact when *x* is a variable, then *id* and *x* share the same memory location.

SPECIAL FUNCTION REGISTER MAP

FOR PIC18F2455/2550/4455/4550 DEVICES

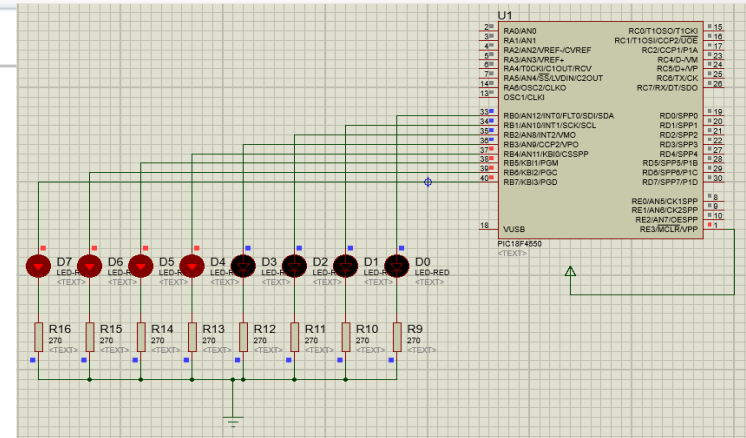


F84h	PORTE
F83h	PORTD ⁽³⁾
F82h	PORTC
F81h	PORTB
F80h	PORTA

Άσκηση 00a εμφάνιση στην πόρτα B της τιμής 1111000

exercise_0.c*

```
1  #include <main.h> // Συμπεριλαμβάνουμε το αρχείο
2  // αρχικών ρυθμίσεων main.h
3  #byte PORTB =0xF81 // αποδίδουμε στη θέση μνήμης 0xF81
4  // το όνομα PORTB
5  // Δηλαδή δημιουργούμε μια μεταβλητή
6  // των 8 bit της οποίας η τιμή θα
7  // αποθηκεύεται στη θέση μνήμης F81h
8  // Η θέση μνήμης F81h είναι ο καταχωρητής
9  // δεδομένων της πόρτας B
10
11 void main(void){
12
13  set_tris_b(0x00); // κάνουμε την πόρτα B έξοδο
14  // Δηλαδή δίνουμε στον καταχωρητή κατεύθυνσης
15  // της πόρτας B την τιμή 0000 0000
16
17  PORTB =0xF0;      //Αποδίδουμε στον καταχωρητή δεδομένων
18  //της πόρτας B την τιμή 1111 0000
19
20  while(TRUE) {    // ατέρμων βρόχος
21      }
22
23  } // κλείνει η αγκύλη της main()
24
```



SPECIAL FUNCTION REGISTER MAP
FOR PIC18F2455/2550/4455/4550 DEVICES



F84h	PORTE
F83h	PORTD ⁽³⁾
F82h	PORTC
F81h	PORTB
F80h	PORTA

Άσκηση 00b. Χρήση της παράλληλης θύρας του μικροελεγκτή για εξαγωγή δεδομένων (άναμμα και σβήσιμο των LED)

Πρόγραμμα:

Προσοχή: Στο φάκελο στον οποίο θα αναπτύξετε το project θα πρέπει να τοποθετήσετε τα αρχεία main.h και 18f4550.h και add.txt.

```
#include<main.h>           //Το αρχείο <main.h> περιέχει αρχικές ρυθμίσεις
                             //Πρέπει να τοποθετηθεί οπωσδήποτε στον ίδιο φάκελο στον οποίο θα
                             //αναπτύξετε το project σας
#byte PORTB=0xF81          //F81 είναι η θέση του καταχωρητή δεδομένων της πόρτας B
                             // στη μνήμη του μικροελεγκτή
                             // Δίνουμε στην διεύθυνση 0xF81 το όνομα PORTB

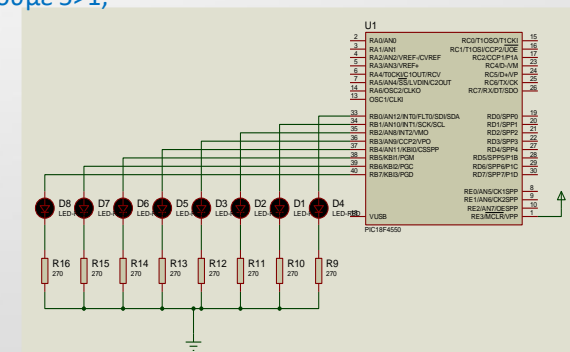
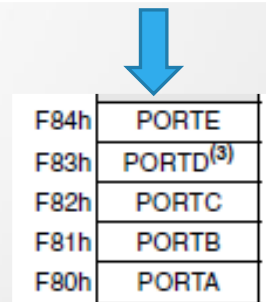
// *****Από εδώ ξεκινάει το κύριο πρόγραμμα*****
void main()
{
    //άνοιγμα αγκύλης της συνάρτησης main
    set_tris_b(0x00); //Η θύρα B γίνεται έξοδος (καταχωρητής κατεύθυνσης=0000 0000)

    // Με την παρακάτω δομή while(TRUE){ } εκτελείται συνεχώς (για πάντα) το σύνολο των
    // εντολών που είναι μέσα στις αγκύλες. Η δεσμευμένη λέξη TRUE στη γλώσσα C αντιστοιχεί
    // στην αληθή συνθήκη. Αντί για TRUE θα μπορούσαμε για παράδειγμα να βάλουμε 5>1,
    // δηλαδή μια συνθήκη που ισχύει πάντα.

    while(TRUE) {           //Βρόχος που δεν τελειώνει ποτέ (συνθήκη πάντα αληθής)
        PORTB=0b11111111; //Όλοι οι ακροδέκτες της πόρτας B λαμβάνουν την τιμή 1
        delay_ms(100);     // Καθυστέρηση 100 ms
        PORTB=0b00000000; //Όλοι οι ακροδέκτες της πόρτας B λαμβάνουν την τιμή 0
        delay_ms(100);     // Καθυστέρηση 100ms
        //κλείσιμο της αγκύλης του while
    }
    // κλείσιμο της αγκύλης της main
}
```

FOR PIC18F2455/2550/4455/4550 DEVICES

SPECIAL FUNCTION REGISTER MAP



Άσκηση 00c

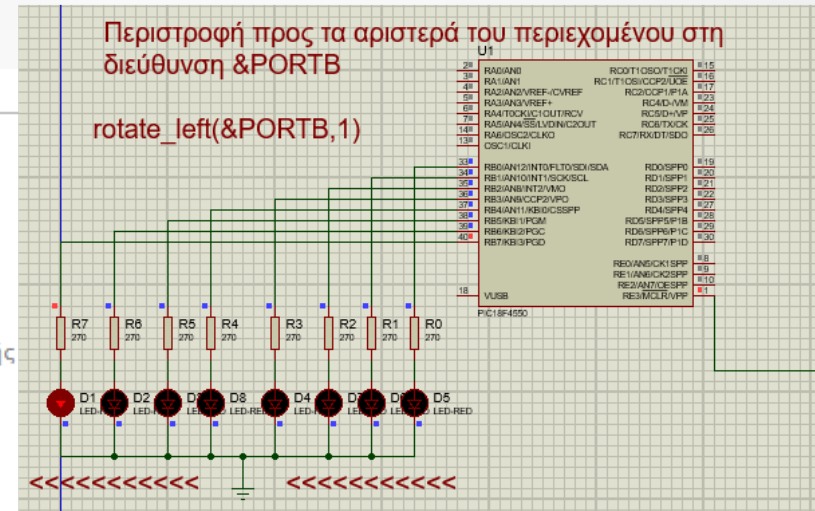
Περιστρεφόμενη τελεία

exercise_0.c*

```

1  #include <main.h> // Συμπεριλαμβάνουμε το αρχείο
2      // αρχικών ρυθμίσεων main.h
3  #byte PORTB =0xF81 // αποδίδουμε στη θέση μνήμης 0xF81
4      // το όνομα PORTB
5      // Δηλαδή δημιουργούμε μια μεταβλητή
6      // των 8 bit της οποίας η τιμή θα
7      // αποθηκεύεται στη θέση μνήμης F81h
8      // Η θέση μνήμης F81h είναι ο καταχωρητής
9      // δεδομένων της πόρτας B
10
11 void main(void){
12
13     set_tris_b(0x00); // κάνουμε την πόρτα B έξοδο
14     // Δηλαδή δίνουμε στον καταχωρητή κατεύθυνσης
15     // της πόρτας B την τιμή 0000 0000
16
17     PORTB =0b10000000; //Αποδίδουμε στον καταχωρητή δεδομένων
18     //της πόρτας B την τιμή 1111 0000
19
20     while(TRUE) { // ατέρμων βρόχος
21
22         delay_ms(100); // καθυστέρηση 100 χιλιοστά του δευτερολέπτου
23         rotate_left(&PORTB,1); // περιστροφή αριστερά της τιμής που
24             // περιέχεται στη διεύθυνση &PORTB
25     }
26
27     } // κλείνει η αγκύλη της main()
28

```



SPECIAL FUNCTION REGISTER MAP

FOR PIC18F2455/2550/4455/4550 DEVICES

F84h	PORTE
F83h	PORTD ⁽³⁾
F82h	PORTC
F81h	PORTB
F80h	PORTA

Εντολή περιστροφής byte

`rotate_left()`

Syntax: `rotate_left (address, bytes)`

Parameters: `address` is a pointer to memory, `bytes` is a count of the number of bytes to work with.

Returns: undefined

Function: Rotates a bit through an array or structure. The address may be an array identifier or an address to a byte or structure (such as `&data`). Bit 0 of the lowest BYTE in RAM is considered the LSB.

Availability: All devices

Requires: Nothing

Examples:

```
x = 0x86; // 0b10000110
rotate_left( &x, 1); // x is now 0x0d // 0b00001101
```

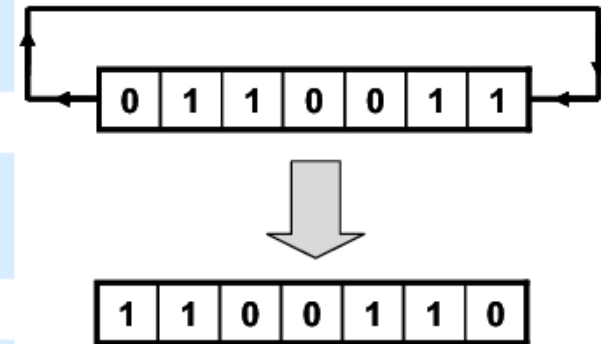
Example Files: None

Also See: [rotate_right\(\)](#), [shift_left\(\)](#), [shift_right\(\)](#)

218

`rotate_left(&PORTB,1)`

Περιστροφή byte αριστερά



Άσκηση 00d. Κίνηση τελείας δεξιά-αριστερά

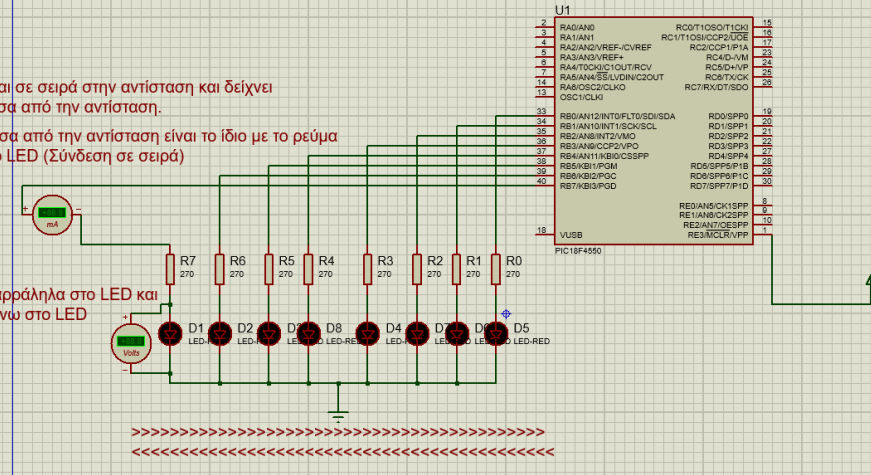
exercise_00d.c

```
1 #include <main.h> // Συμπεριλαμβάνουμε το αρχείο
2 // αρχικών ρυθμίσεων main.h
3 #byte PORTB =0xF81 // αποδίδουμε στη θέση μνήμης 0xF81
4 // το όνομα PORTB
5 // Δηλαδή δημιουργούμε μια μεταβλητή
6 // των 8 bit της οποίας η τιμή θα
7 // αποθηκεύεται στη θέση μνήμης F81h
8 // Η θέση μνήμης F81h είναι ο καταχωρητής
9 // δεδομένων της πόρτας B
10 int8 i;
11 void main(void){
12     set_tris_b(0x00); // κάνουμε την πόρτα B έξοδο
13     // Δηλαδή δίνουμε στον καταχωρητή κατεύθυνσης
14     // της πόρτας B την τιμή 0000 0000
15     PORTB =0b10000000; //Αποδίδουμε στον καταχωρητή δεδομένων
16     //της πόρτας B την τιμή 1000 0000
17
18     while(TRUE) { // ατέρμων βρόχος
19         for(i=1;i<=7;i++){
20             delay_ms(50);
21             PORTB=PORTB/2; //Διαίρεση με το 2
22             // σημαίνει μετακίνηση όλων
23             //των bit της πόρτας B κατά μια θέση δεξιά
24         }
25
26         for(i=7;i>=1;i--){
27             delay_ms(50);
28             PORTB=PORTB*2; //Πολλαπλασιασμός με το 2
29             // σημαίνει μετακίνηση όλων
30             //των bit της πόρτας B κατά μια θέση αριστερά
31         }
32     } // κλείνει η αγκύλη της while(TRUE)
33
34     } // κλείνει η αγκύλη της main()
35
36 }
```

Το αμπερόμετρο συνδέεται σε σειρά στην αντίσταση και δείχνει το ρεύμα που περνάει μέσα από την αντίσταση.

Το ρεύμα που περνάει μέσα από την αντίσταση είναι το ίδιο με το ρεύμα που περνάει μέσα από το LED (Σύνδεση σε σειρά)

Το βολτόμετρο συνδέεται παράλληλα στο LED και δείχνει την πτώση τάσης πάνω στο LED



Άσκηση 00d. Κίνηση τελείας δεξιά-αριστερά

exercise_00d.c

```

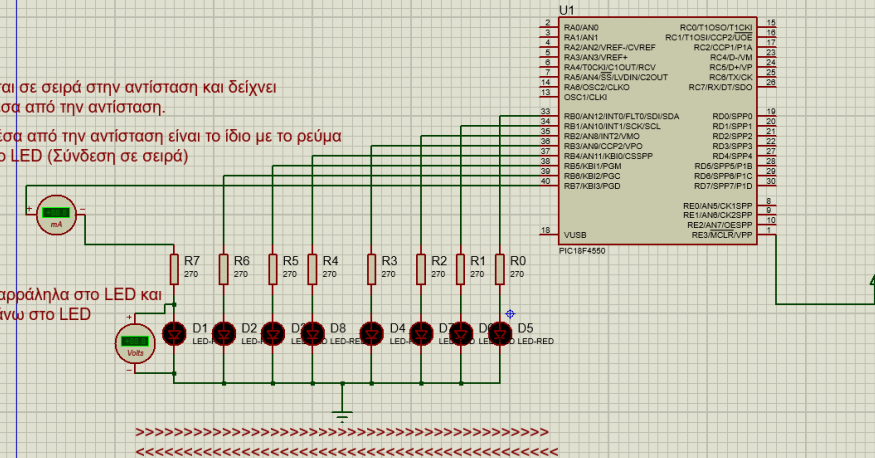
1  #include <main.h> // Συμπεριλαμβάνουμε το αρχείο
2  // αρχικών ρυθμίσεων main.h
3  #byte PORTB =0xF81 // αποδίδουμε στη θέση μνήμης 0xF81
4  // το όνομα PORTB
5  // Δηλαδή δημιουργούμε μια μεταβλητή
6  // των 8 bit της οποίας η τιμή θα
7  // αποθηκεύεται στη θέση μνήμης F81h
8  // Η θέση μνήμης F81h είναι ο καταχωρητής
9  // δεδομένων της πόρτας B
10 int8 i;
11 void main(void){
12     set_tris_b(0x00); // κάνουμε την πόρτα B έξοδο
13     // Δηλαδή δίνουμε στον καταχωρητή κατεύθυνσης
14     // της πόρτας B την τιμή 0000 0000
15     PORTB =0b10000000; //Αποδίδουμε στον καταχωρητή δεδομένων
16     //της πόρτας B την τιμή 1000 0000
17
18     while(TRUE) { // ατέρμων βρόχος
19         for(i=1;i<=7;i++){
20             delay_ms(50);
21             PORTB=PORTB/2; //Διαίρεση με το 2
22             // σημαίνει μετακίνηση όλων
23             // των bit της πόρτας B κατά μια θέση δεξιά
24         }
25
26         for(i=7;i>=1;i--){
27             delay_ms(50);
28             PORTB=PORTB*2; //Πολλαπλασιασμός με το 2
29             // σημαίνει μετακίνηση όλων
30             // των bit της πόρτας B κατά μια θέση αριστερά
31         }
32     } // κλείνει η αγκύλη της while(TRUE)
33
34     } // κλείνει η αγκύλη της main()
35
36
37

```

Το αμπερόμετρο συνδέεται σε σειρά στην αντίσταση και δείχνει το ρεύμα που περνάει μέσα από την αντίσταση.

Το ρεύμα που περνάει μέσα από την αντίσταση είναι το ίδιο με το ρεύμα που περνάει μέσα από το LED (Σύνδεση σε σειρά)

Το βολτόμετρο συνδέεται παράλληλα στο LED και δείχνει την πτώση τάσης πάνω στο LED



Διαίρεση με 2 => μετακίνηση του 1 δεξιά

1000 0000	→	128 (αρχική τιμή της PORTB)
0100 0000	→	64 (τιμή για i=1)
0010 0000	→	32 (τιμή για i=2)
0001 0000	→	16 (τιμή για i=3)
0000 1000	→	8 (τιμή για i=4)
0000 0100	→	4 (τιμή για i=5)
0000 0010	→	2 (τιμή για i=6)
0000 0001	→	1 (τιμή για i=7)

Άσκηση 00d. Κίνηση τελείας δεξιά-αριστερά

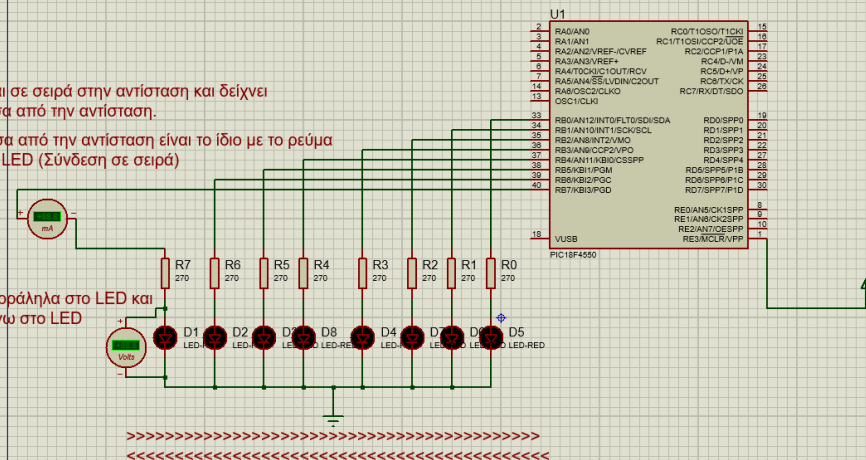
exercise_00d.c

```
1 #include <main.h> // Συμπεριλαμβάνουμε το αρχείο
2 // αρχικών ρυθμίσεων main.h
3 #byte PORTB =0xF81 // αποδίδουμε στη θέση μνήμης 0xF81
4 // το όνομα PORTB
5 // Δηλαδή δημιουργούμε μια μεταβλητή
6 // των 8 bit της οποίας η τιμή θα
7 // αποθηκεύεται στη θέση μνήμης F81h
8 // Η θέση μνήμης F81h είναι ο καταχωρητής
9 // δεδομένων της πόρτας B
10 int8 i;
11 void main(void){
12     set_tris_b(0x00); // κάνουμε την πόρτα B έξοδο
13     // Δηλαδή δίνουμε στον καταχωρητή κατεύθυνσης
14     // της πόρτας B την τιμή 0000 0000
15     PORTB =0b10000000; //Αποδίδουμε στον καταχωρητή δεδομένων
16     //της πόρτας B την τιμή 1000 0000
17     while(TRUE) { // ατέρμων βρόχος
18         for(i=1;i<=7;i++){
19             delay_ms(50);
20             PORTB=PORTB/2; //Διαίρεση με το 2
21             // σημαίνει μετακίνηση όλων
22             //των bit της πόρτας B κατά μια θέση δεξιά
23         }
24         for(i=7;i>=1;i--){
25             delay_ms(50);
26             PORTB=PORTB*2; //Πολλαπλασιασμός με το 2
27             // σημαίνει μετακίνηση όλων
28             //των bit της πόρτας B κατά μια θέση αριστερά
29         }
30     } // κλείνει η αγκύλη της while(TRUE)
31 } // κλείνει η αγκύλη της main()
```

Το αμπερόμετρο συνδέεται σε σειρά στην αντίσταση και δείχνει το ρεύμα που περνάει μέσα από την αντίσταση.

Το ρεύμα που περνάει μέσα από την αντίσταση είναι το ίδιο με το ρεύμα που περνάει μέσα από το LED (Σύνδεση σε σειρά)

Το βολτόμετρο συνδέεται παράλληλα στο LED και δείχνει την πτώση τάσης πάνω στο LED

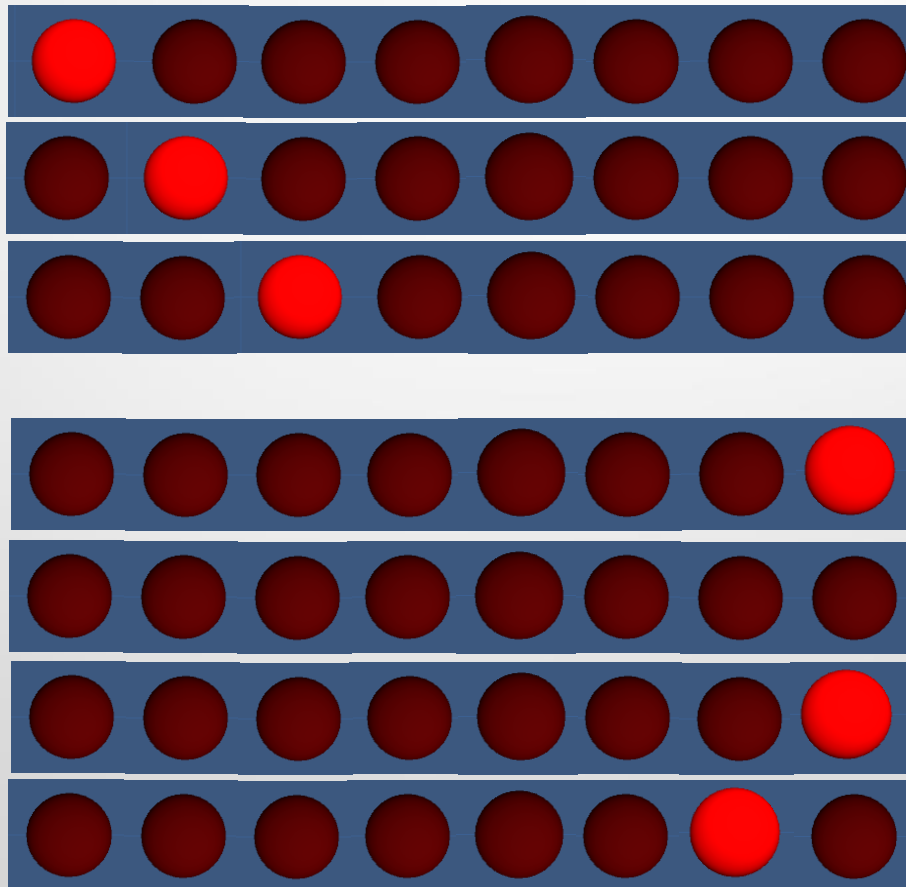


Πολλαπλασιασμός με 2 => μετακίνηση του 1 αριστερά

0000 0001 → 1 (αρχική τιμή της PORTB)
0000 0010 → 2 (τιμή για i=7)
0000 0100 → 4 (τιμή για i=6)
0000 1000 → 8 (τιμή για i=5)
0001 0000 → 16 (τιμή για i=4)
0010 0000 → 32 (τιμή για i=3)
0100 0000 → 64 (τιμή για i=2)
1000 0000 → 128 (τιμή για i=1)

Άσκηση 00ε. Κίνηση τελείας δεξιά-αριστερά_ΠΑΡΑΛΛΑΓΗ

Ποιες αλλαγές πρέπει να γίνουν στον κώδικα ώστε η τελεία όταν μετακινείται από δεξιά προς τα αριστερά να εξαφανίζεται μετά το LSB και να ξαναεμφανίζεται στο LSB. Στη συνέχεια μετακινείται προς τα αριστερά και μόλις φτάνει στο MSB να εξαφανίζεται και να εμφανίζεται πάλι στο MSB για να μετακινηθεί πάλι προς τα δεξιά;



Ενσωμάτωση κώδικα σε γλώσσα Assembly μέσα σε πρόγραμμα σε γλώσσα C

#ASM #ENDASM

Syntax: `#ASM`
 or
 `#ASM ASIS`
 `code`
 `#ENDASM`

Elements: `code` is a list of assembly language instructions

Purpose: The lines between the `#ASM` and `#ENDASM` are treated as assembly code to be inserted. These may be used anywhere an expression is allowed. The syntax is described on the following page. Function return values are sent in `W0` for 16-bit, and `W0, w1` for 32 bit. Be aware that any C code after the `#ENDASM` and before the end of the function may corrupt the value.

If the second form is used with `ASIS` then the compiler will not do any optimization on the assembly. The assembly code is used as-is.

Examples:

```
int find_parity(int data){  
    int count;  
    #asm  
    MOV #0x08, W0  
    MOV W0, count  
    CLR W0  
loop:  
    XOR.B data,W0  
    RRC data,W0  
    DEC count,F  
    BRA NZ, loop  
    MOV #0x01,W0  
    ADD count,F  
    MOV count, W0  
    MOV W0, _RETURN_  
    #endasm  
}
```

Example [ex_qlint.c](#)

Files:

Also See: None

`#asm`

...

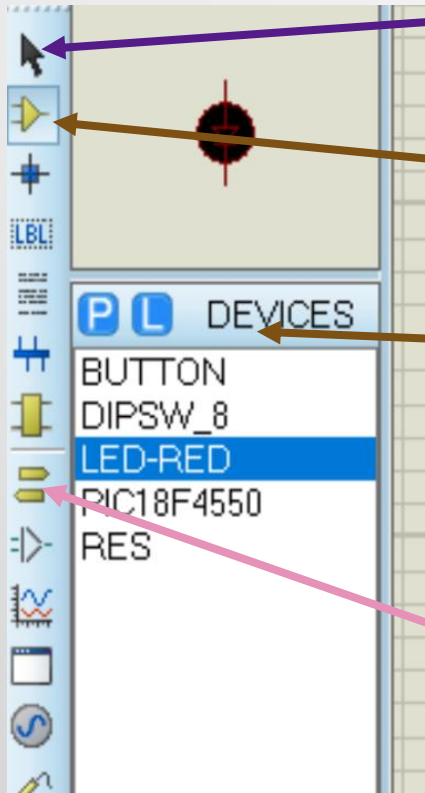
...

...

`#endasm`


Κώδικας σε γλώσσα assembly ο οποίος παρεμβάλλεται μέσα σε ένα πρόγραμμα σε γλώσσα C

Πρόγραμμα σχεδίασης και προσομοίωσης ηλεκτρονικών κυκλωμάτων Proteus



- Επιλογή περιοχής για αντιγραφή ή μετακίνηση (Selection mode)

- Επιλογή εξαρτήματος (Component mode)

Πατάμε το κουμπί  για εισαγωγή εξαρτημάτων και στη συνέχεια το P από το **P L DEVICES** για να επιλέξουμε το εξάρτημα που θέλουμε να εισάγουμε. Αφού επιλέξουμε και εισάγουμε ένα εξάρτημα, αυτό εμφανίζεται στη λίστα **P L DEVICES**. Στη συνέχεια πιάνουμε το εξάρτημα από τη λίστα **P L DEVICES** και το μεταφέρουμε στην επιφάνεια εργασίας

- Εισαγωγή τροφοδοσίας (POWER) και γείωσης (GROUND)
- Πιάνουμε το POWER ή το GROUND και το μεταφέρουμε στην επιφάνεια εργασίας