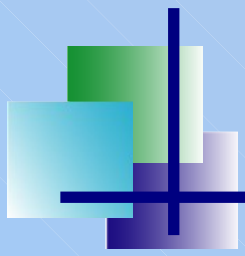


Η Γλώσσα Προγραμματισμού C

Β. Πόθος

Δ.Π.Μ.Σ.
Ηλεκτρονική και Επεξεργασία της Πληροφορίας (Η.Ε.Π.)
http://www.hep.upatras.gr/

e-mail: bpothos@upatras.gr

OpenOffice.org Impress

Δομή προγράμματος σε C

Ένα πρόγραμμα σε C γράφεται με την ακόλουθη σειρά:

```
#include <stdio.h>          //ενσωματώνει στο πρόγραμμα μας το αρχείο επικεφαλίδας stdio.h

int main(void)              //δήλωση προγράμματος (επικεφαλίδα προγράμματος)
{                           //έναρξη προγράμματος
        δήλωση μεταβλητών
        εντολή 1;
        εντολή 2;
        ……
        εντολή n;
        return 1;
}                           //τέλος προγράμματος
```

```c
#include "stdio.h"

int main(void) {
    int x;
    int y;
    int sum;

    x=5;
    y=3;
    sum = x+y;
    return 1;
}
```

Βασικοί κανόνες της C

Στη γλώσσα C ισχύουν οι ακόλουθοι κανόνες:

- Κάθε πρόγραμμα ξεκινά με τη συνάρτηση main(). Η συνάρτηση αυτή είναι μοναδική και μπορεί να βρίσκεται οπουδήποτε μέσα σε ένα πρόγραμμα.
- Κάθε συνάρτηση ξεκινάει και τελειώνει με τα σύμβολα { και }.
- Κάθε εντολή τελειώνει με το ελληνικό ερωτηματικό (;).

```
#include "stdio.h"

int main(void) {
    int x;
    int y;
    int sum;

    x=5;
    y=3;
    sum = x+y;
    return 1;
}
```

Δήλωση μεταβλητών

Η δήλωση μιας μεταβλητής γίνεται με την ακόλουθη εντολή:

<τύπος> <όνομα μεταβλητής>;

Παραδείγματα δηλώσεων

- int x; η μεταβλητή x δηλώνεται ακέραιου τύπου
- float y; η μεταβλητή y δηλώνεται πραγματικού τύπου
- char z; η μεταβλητή z δηλώνεται τύπου χαρακτήρα
- double k; η μεταβλητή k δηλώνεται τύπου double

Τύποι μεταβλητών

Στη γλώσσα C υπάρχουν οι ακόλουθοι τύποι:

Λέξη	Bytes	φάσμα
char	1	-128 εως 127
int	2	-32768 εως 32767
short	2	-32768 εως 32767
long	4	-21477483648 εως 21477483647
unsigned char	1	0 εως 255
unsigned int	2	0 εως 65535
unsigned short	2	0 εως 65535
unsigned long	4	Ο εως 4.294.967.295
float	4	1.2E – 38 εως 3.4E38[1]
double	8	2.2E – 308 εως 1.8E308[2]

Τρόποι δηλώσεων μεταβλητών

- Συνεχόμενη δήλωση μεταβλητών ίδιου τύπου
  float mikos, platos, embadon;

- Δήλωση με ταυτόχρονη αρχικοποίηση μεταβλητής
  int i = 1;

- Μεταβλητές που δεν αλλάζουν τιμή (Σταθερές)
  const float pi = 3.14;

- Η #define ορίζει ένα συμβολικό όνομα ή συμβολική σταθερά ίση με μια συγκεκριμένη ακολουθία χαρακτήρων:

#define MAX 20

Από εκεί και κάτω οποιαδήποτε εμφάνιση του ονόματος θα αντικαθίσταται από το αντίστοιχο κείμενο αντικατάστασης.

Προσοχή: Η #define δεν θέλει ; στο τέλος

Συσχέτιση διεύθυνσης αποθήκευσης με αναπαριστώμενη ποσότητα

Πως σχετίζεται η διεύθυνση αποθήκευσης με την αναπαριστούμενη ποσότητα;

Στο ερώτημα αυτό έχουν διεθνώς ακολουθηθεί δύο προσεγγίσεις:

- Η προσέγγιση του big-endian θεωρεί ότι στη μικρότερη διεύθυνση από τις θέσεις που καταλαμβάνει μια ποσότητα αποθηκεύεται το πιο σημαντικό κομμάτι της. Τα υπόλοιπα λιγότερο σημαντικά κομμάτια αποθηκεύονται σε διαδοχικές υψηλότερες διευθύνσεις.

- Η προσέγγιση του little-endian θεωρεί την αποθήκευση των ποσοτήτων έτσι ώστε η μικρότερη διεύθυνση να αποθηκεύει το λιγότερο σημαντικό κομμάτι της ποσότητας και τα διαδοχικώς πιο σημαντικά κομμάτια σε αύξουσες διευθύνσεις.

Στην πράξη δεν υπάρχει σημαντικό πλεονέκτημα της μιας από την άλλη μέθοδο αποθήκευσης.

Παράδειγμα big-endian

Για παράδειγμα έστω ότι πρέπει να αποθηκευτεί η 32bit ποσότητα $F2341088^{H}$ σε μία μνήμη που κάθε της θέση αποθηκεύει ένα byte.

Θεωρώντας ότι η αποθήκευση ξεκινάει από τη θέση μνήμης $33FE^{H}$ τότε σε μια big-endian μηχανή θα είχαμε τον εξής πίνακα αποθήκευσης :

Διεύθυνση	Δεδομένο
$33FE_H$	11110010 ($F2_H$)
$33FF_H$	00110100 (34_H)
3400_H	00010000 (10_H)
3401_H	10001000 (88_H)

Παράδειγμα little-endian

Για παράδειγμα έστω ότι πρέπει να αποθηκευτεί η 32bit ποσότητα $F2341088^{H}$ σε μία μνήμη που κάθε της θέση αποθηκεύει ένα byte.

Θεωρώντας ότι η αποθήκευση ξεκινάει από τη θέση μνήμης $33FE^{H}$ τότε σε μια little-endian μηχανή θα είχαμε τον εξής πίνακα αποθήκευσης :

Διεύθυνση	Δεδομένο
$33FE_H$	10001000 (88_H)
$33FF_H$	00010000 (10_H)
3400_H	00110100 (34_H)
3401_H	11110010 ($F2_H$)

Συναρτήσεις εισόδου

Συνάρτηση scanf:

scanf("<προσδιοριστής>", &<μεταβλητή>);

- Είσοδος ακεραίων: scanf("%d", &num);
- Είσοδος πραγματικών: scanf("%f", &num);
  Προσδιοριστές: %f, %e, %g
- Είσοδος χαρακτήρων: scanf("%c", &ch);
  Προσδιοριστές: %c, %s

Προσοχή: Αν η μεταβλητή είναι πίνακας δεν χρειάζεται διεύθυνση

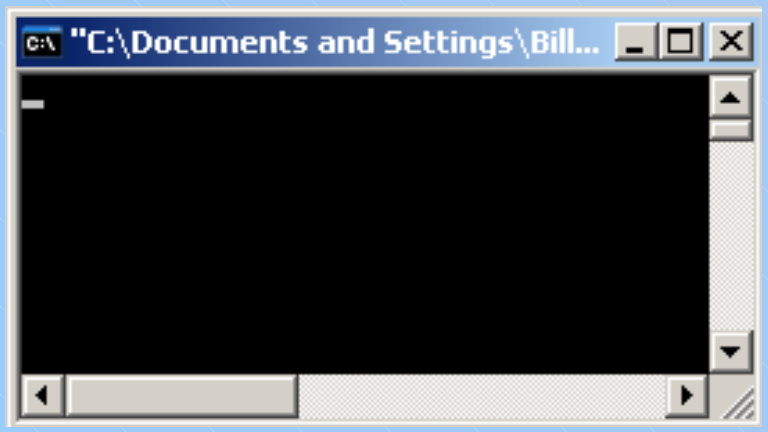
Συναρτήσεις εισόδου

```
#include "stdio.h"

int main(void) {
      int age;
      char Name[6];

      scanf("%d",&age);
      scanf("%s",Name);
      return 1;
}
```

&age →

Name →

	0xA0	
age	0xA1	
	0xA2	
Name[0]	0xA3	
Name[1]	0xA4	
Name[2]	0xA5	
Name[3]	0xA6	
Name[4]	0xA7	
Name[5]	0xA8	
	0xA9	
	0xAA	



{1} - Εκτελείται η εντολή scanf("%d",&age);

Το σύστημα περιμένει να πάρει από μια συσκευή εισόδου (εξ' ορισμού το πληκτρολόγιο) έναν ακέραιο.

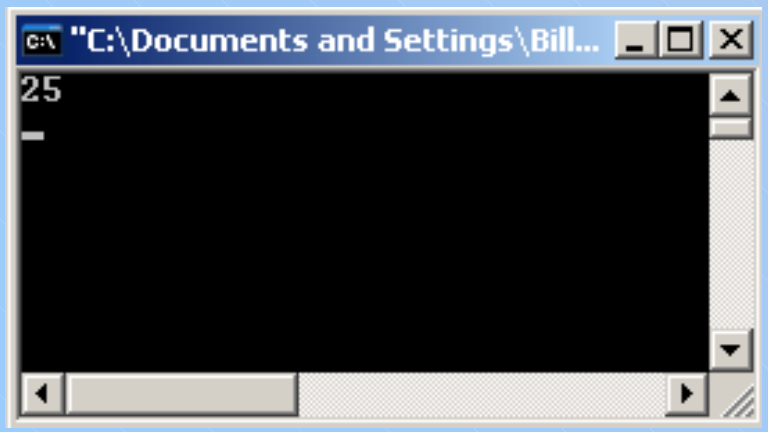
Συναρτήσεις εισόδου

```
#include "stdio.h"

int main(void) {
      int age;
      char Name[6];

      scanf("%d",&age);
      scanf("%s",Name);
      return 1;
}
```

&age →

Name →

	0xA0	
age	0xA1	19^{H}
	0xA2	00^{H}
Name[0]	0xA3	
Name[1]	0xA4	
Name[2]	0xA5	
Name[3]	0xA6	
Name[4]	0xA7	
Name[5]	0xA8	
	0xA9	
	0xAA	



{2} - Εκτελείται η εντολή scanf("%s",Name);

Η προηγούμενη scanf() εντολή εκτελέστηκε. Η scanf() έβαλε την τιμή 25, στην θέση μνήμης που έχει οριστεί η μεταβλητή age - έστω 0xA1. Ισχύει: $0x0019^{H} = 25^{Dec}$.

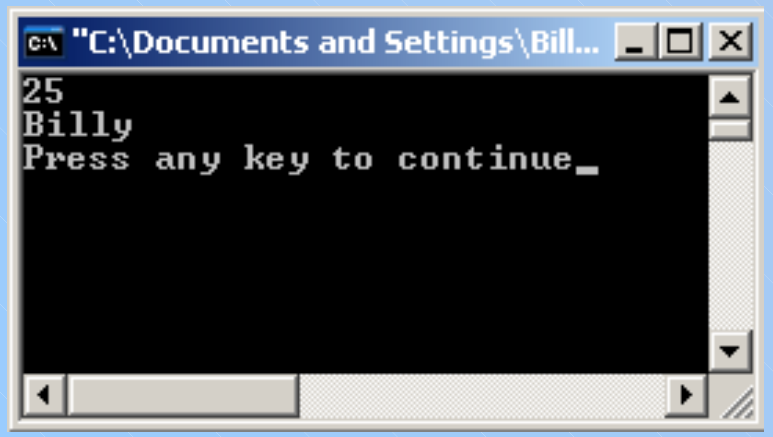
Συναρτήσεις εισόδου

```
#include "stdio.h"

int main(void) {
      int age;
      char Name[6];

      scanf("%d",&age);
      scanf("%s",Name);
      return 1;
}
```

&age →

Name →

	0xA0	
age	0xA1	19^H
	0xA2	00^H
Name[0]	0xA3	42^H - "B"
Name[1]	0xA4	69^H - "i"
Name[2]	0xA5	$6C^H$ - "l"
Name[3]	0xA6	$6C^H$ - "l"
Name[4]	0xA7	79^H - "y"
Name[5]	0xA8	00^H - "\0"
	0xA9	
	0xAA	



{3} - Εκτελείται η εντολή return; (τέλος)

Η προηγούμενη scanf() εντολή εκτελέστηκε. Η scanf() έβαλε τις τιμές 42^H,69^H,$6C^H$,$6C^H$, 79^H, 00^H, στον πίνακα Name.

Συναρτήσεις εξόδου

Συνάρτηση printf:

printf("<περιγραφή>", <ακολουθία μεταβλητών>);

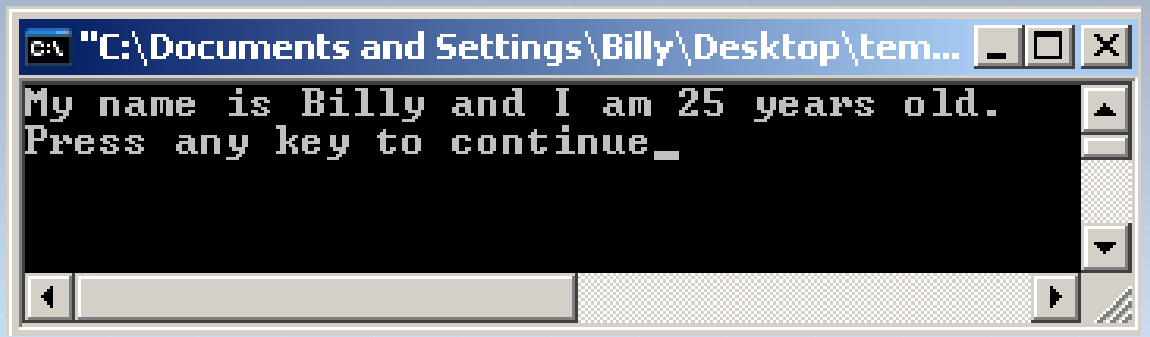
- Έξοδος ακεραίων: printf("%d", num);
  Προσδιοριστές: %d, %x, %o

- Έξοδος πραγματικών: printf("%f", num);
  Προσδιοριστές: %f, %e, %g

- Έξοδος χαρακτήρων: printf("%c", ch); (χαρακτήρας)
  Προσδιοριστές: %c, %d printf ("%d", ch); (Κωδικός ASCII)

Συναρτήσεις εξόδου

```c
#include "stdio.h"

int main(void) {
	int age=25;
	char Name[6]="Billy";

	printf("My name is %s and I am %d years old.\n",Name,age);
	return 1;
}
```



Προσδιοριστές μορφής εκτύπωσης

Οθόνης:

- "\t" -> Αφήνει ένα tab
- "\n" -> Αλλάζει γραμμή

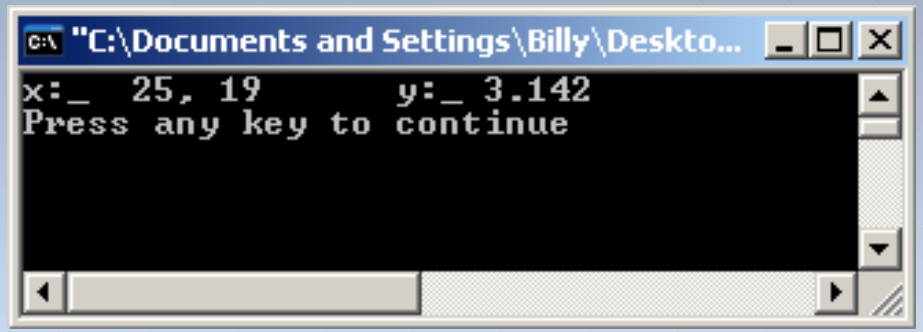
Αριθμών:

- %<ακέρ><προσδιορ> (καθορισμός πλάτους πεδίου).
  π.χ. %3d

- %[<ακέρ>][.<ακέρ>]<προσδιορ>
  (καθορισμός πλάτους πεδίου και δεκαδικών ψηφίων).
  π.χ. %6.1f, %.2f, %6f

Προσδιοριστές μορφής εκτύπωσης

```c
#include "stdio.h"

int main(void) {
      int x=25;
      double pi = 3.14159;

      printf("x:_%4.d, %x \t y:_%6.3f \n",x,x,pi);
      return 1;
}
```



Σχόλια στο κείμενο

- Κάθε γραμμή στον κώδικα που ξεκινά με // ή /*...*/ είναι ένα σχόλιο (comment).

- Οι χαρακτήρες που βρίσκονται ανάμεσα στα /* και */ και μετά από το // αγνοούνται από το μεταγλωττιστή και χρησιμοποιούνται για να κάνουν το πρόγραμμα πιο ευκολονόητο.

Δομές

- Οι δομές μπορούν να ειδωθούν σαν ομάδες μεταβλητών, οι οποίες αναφέρονται σε κάποιο συγκεκριμένο αντικείμενο.
- Χρησιμοποιείται η λέξη-κλειδί struct για να οριστούν.

```
#include "stdio.h"

struct tag_Sensor {
     int Ready; char Instruction_or_Data; int IData[256]; unsigned char DataType;
};
struct tag_Sensor Sensor;

int main(void) {
     Sensor.DataType = 0x00;
     Sensor.Instruction_or_Data = 0x01;
     Sensor.IData[0]=0xFF;
     // Suppose we have a function that initialize a sensor.
     Sensor.Ready = Initialize_Sensor(Sensor);
     return 1;
}
```

Τελεστές

Οι σχεσιακοί τελεστές χρησιμοποιούνται για να δημιουργήσουν συνθήκες.
Στη γλώσσα C είναι οι ακόλουθοι:

Σχεσιακοί Τελεστές
< (μικρότερο)
<=(μικρότερο ή ίσο)
> (μεγαλύτερο)
>=(μεγαλύτερο ή ίσο)
== (ισότητα)
!= (διάφορο)

Παραδείγματα σχεσιακών τελεστών

x==3
y>=10
z!=0
a>0
b<20

Σημείωση

Πρέπει να τονίσουμε τη διαφορά ανάμεσα στην εντολή καταχώρισης η οποία συμβολίζεται με το = και τον τελεστή της ισότητας ο οποίος συμβολίζεται με ==.

Λογικοί Τελεστές

• Οι λογικοί τελεστές χρησιμοποιούνται προκειμένου να συνδυάσουν πολλές απλές συνθήκες μέσα σε μια σύνθετη συνθήκη. Η τιμή μιας συνθήκης απλής ή σύνθετης είναι είτε true (αληθής) είτε false (ψευδής). Στη γλώσσα C οι λογικοί τελεστές είναι οι ακόλουθοι είναι οι ακόλουθοι:

Λογικοί Τελεστές
&& (AND)
|| (OR)
! (NOT)

Παραδείγματα απλών και σύνθετων συνθηκών

```
if (x!=0)
if (x>=0 && x<5)
if (x==9 || x==10)
if !(x<0)
while (x>=0 && x<=10) || (x>=20 && x<=30)
for (i=0; x!=-1; i++)
```

Τελεστές Bits

- Οι τελεστές bits υλοποιούν λογικές πράξεις, συνδυάζοντας τα ορίσματά τους bit προς bit, χωρίς να ερμηνεύουν το περιεχόμενο.
- Χρησιμεύουν στον έλεγχο/προγραμματισμό ports γενικότερα.

x&y	x and y	1001 & 0101 = 0001
x|y	x or y	1001 | 0101 = 1101
x^y	x xor y	1001 ^ 0101 = 1100
~x	not x	~0101 = 1010

- Υπάρχουν και οι τελεστές ολίσθησης, οι οποίοι μετατοπίζουν τα bit των ορισμάτων τους προς τα δεξιά και προς τ' αριστερά.

x >> y ολισθαίνει την τιμή x, y φορές δεξιά.

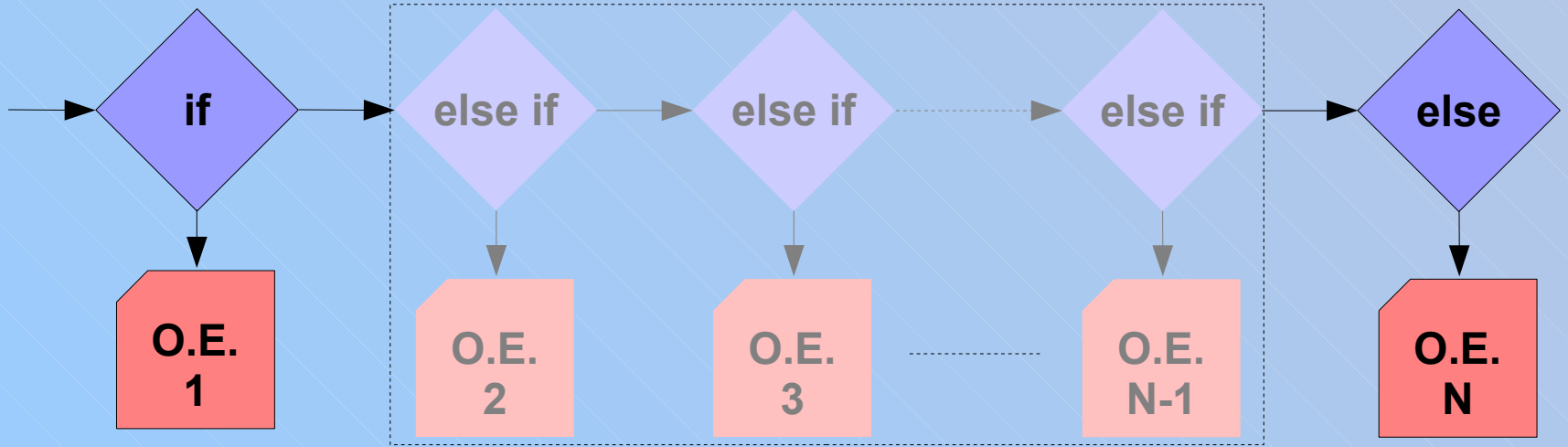
x << y ολισθαίνει την τιμή x, y φορές αριστερά.

```
x      10011001
x<<2   01100100  /* logical shift */
x>>2   00100110  /*unsigned, logical shift*/
x>>2   11100110  /*signed, arithmetic shift*/
```

Η εντολή if

- Ανάλογα με τον αν ικανοποιούνται η όχι κάποιες συνθήκες, η εντολή if επιτρέπει την εκτέλεση διαφορετικών εντολών. Έχει την ακόλουθη σύνταξη:

```
if (συνθήκη 1) { ομάδα εντολών 1; }
else if (συνθήκη 2) { ομάδα εντολών 2; }
else if (συνθήκη 3) { ομάδα εντολών 3; }
else if ... ... ...
else if (συνθήκη N-1) { ομάδα εντολών N-1; }
else { ομάδα εντολών N; }
```



Εντολή πολλαπλής επιλογής switch

- Η εντολή πολλαπλής επιλογής χρησιμοποιείται όταν ελέγχουμε την τιμή μιας μεταβλητής (όχι συνθήκης) και ανάλογα με την τιμή αυτή εκτελούμε είτε μια ομάδα εντολών 1 είτε μια ομάδα εντολών 2 είτε μια εντολών n. Αν η μεταβλητή δεν πάρει καμία από τις τιμές 1, 2, … n τότε εκτελείται η ομάδα εντολών που βρίσκεται στο default. Η εντολή switch γενικά έχει την ακόλουθη μορφή:

```
switch (μεταβλητή)
{
      case τιμή 1:  ομάδα εντολών 1;
                    break;
      case τιμή 2:  ομάδα εντολών 2;
                    break;
            ……
      case τιμή n:  ομάδα εντολών n;
                    break;
      default:      ομάδα εντολών n+1;
}
```

```
#include "stdio.h"

int main(void) {
      int x=2;
      switch(x) {
            case 1: x++; break;
            case 2: x--; break;
            default:  x=0;
      }
      printf("%d\n",x);
      return 1;
}
```

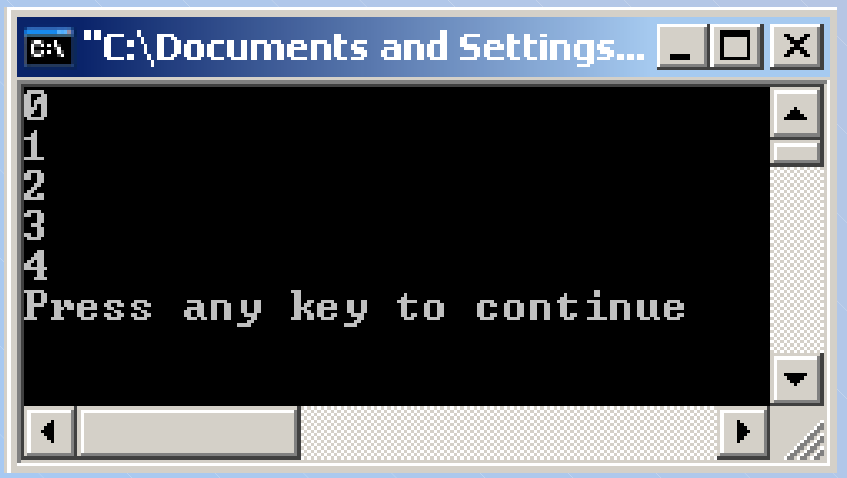
x:1

Εντολή επανάληψης for

- Η εντολή for επαναλαμβάνει μια ομάδα εντολών συνήθως ένα συγκεκρι-μένο αριθμό φορών αν και μπορεί να χρησιμοποιηθεί ακόμα και για την επανάληψη μιας ομάδας εντολών για όσο χρόνο μια συνθήκη είναι αληθής. Έχει την ακόλουθη σύνταξη:

```
for (αρχικές τιμές; συνθήκη; μεταβολές)    {
      ομάδα εντολών;
}
```

```
#include "stdio.h"

int main(void) {
      int i=0;
      for (i=0; i<5; i++) {
            printf("%d\n",i);
      }
      return 1;
}
```



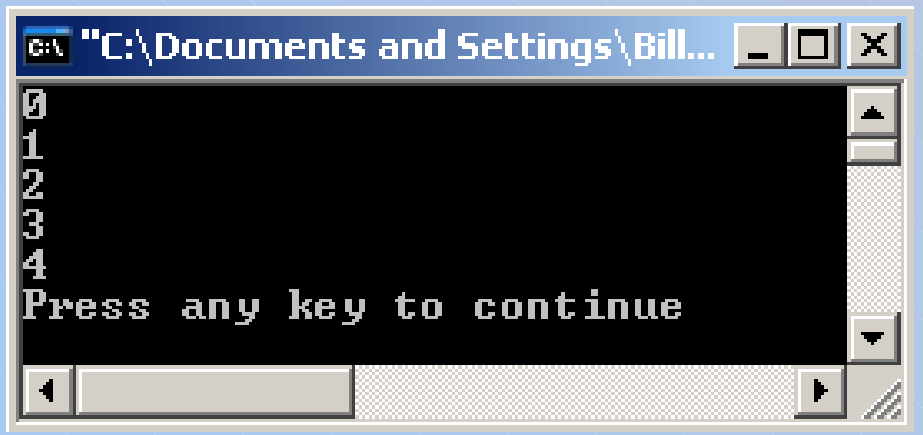
Εντολή Επανάληψης while

Ο βρόχος **while** λειτουργεί ως εξής:

- Ελέγχεται η συνθήκη μέσα στις παρενθέσεις
- Εάν είναι αληθής τότε εκτελείται ο κώδικας που βρίσκεται στις αγκύλες
- Η συνθήκη ελέγχεται ξανά και εάν είναι αληθής επανεκτελείται ο κώδικας
- Η εκτέλεση του κώδικα που είναι μέσα στις αγκύλες παύει να γίνεται όταν η συνθήκη της while γίνει ψευδής

while (συνθήκη) { ομάδα εντολών; }

```
#include "stdio.h"

int main(void) {
      int x=0;
      while (x < 5){
            printf("%d\n",x);
            x++;
      }
      return 1;
}
```

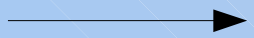


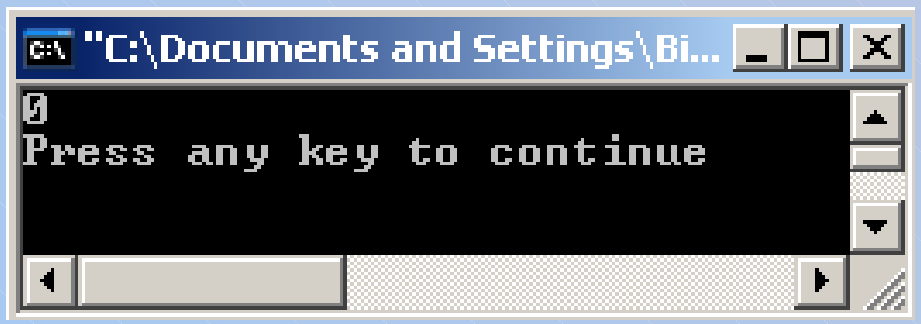
Εντολή επανάληψης do..while

- Η εντολή do while είναι αντίστοιχη με την εντολή while με τη διαφορά ότι πρώτα εκτελείται η ομάδα εντολών και μετά ελέγχεται η συνθήκη, δηλαδή η ομάδα εντολών θα εκτελεστεί τουλάχιστον μια φορά ακόμα και αν η συνθήκη είναι ψευδής. Έχει την ακόλουθη σύνταξη:

do { ομάδα εντολών; } while (συνθήκη);

```
#include "stdio.h"

int main(void) {
      int x=0;
      do {
            printf("%d\n",x);
            x++;
      } while (x < 0);
      return 1;
}
```





Μονοδιάστατοι Πίνακες

Δήλωση Πίνακα

<τύπος> <όνομα-πίνακα> [<μέγεθος>];

float bathmos[5];

Αρχικοποίηση

```
float bathmos[5]= {10, 8, 12.5,15.5, 14};
float bathmos[]= {10, 8, 12.5,15.5, 14};
```

Προσοχή στο πλήθος των αρχικών τιμών

Προσπέλαση Μονοδιάστατου Πίνακα

- Τα στοιχεία ενός πίνακα αποθηκεύονται σε γειτονικές (διαδοχικές) θέσεις μνήμης.
- Ο πίνακας είναι μια δομή τυχαίας προσπέλασης.

```
short bathmos[5];
```

Π.χ.

```
bathmos[3] = 0x1234;
bathmos[0] = 0xFFEE;
```

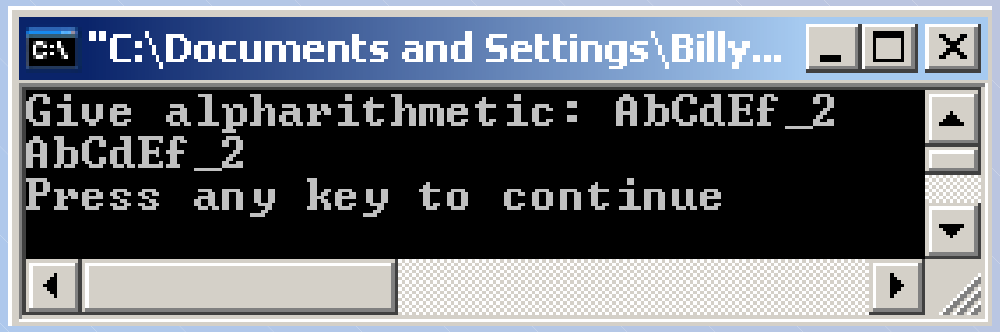
Επίσης ισχύει η συνθήκη:

bathmos == 0xA2

Index	Reference	Address	Data
		0xA1	
1	bathmos[0]	0xA2	EE^H
		0xA3	FF^H
2	bathmos[1]	0xA4	
		0xA5	
3	bathmos[2]	0xA6	
		0xA7	
4	bathmos[3]	0xA8	34^H
		0xA9	12^H
5	bathmos[4]	0xAA	
		0xAB	
		0xAC	
		0xAD	

Διάβασμα-Εκτύπωση Μονοδιάστατου Πίνακα

```c
#include <stdio.h>
#define MAX_CHARS 80

int main(void) {
	char str[MAX_CHARS];
	int i=0;

	printf("Give alpharithmetic: ");
	scanf("%s", str);
	while (str[i] != 0) {
		printf("%c", str[i]);
		i++;
	}
	printf("\n");
	return 1;
}
```



Reference	Address	Data
	0xB0	0xXX
str[0]	0xB1	"A"
str[1]	0xB2	"b"
str[2]	0xB3	"C"
str[3]	0xB4	"d"
str[4]	0xB5	"E"
str[5]	0xB6	"f"
str[6]	0xB7	"_"
str[7]	0xB8	"2"
str[8]	0xB9	0x00
str[9]	0xBA	0xXX
str[10]	0xBB	0xXX
...	...	...

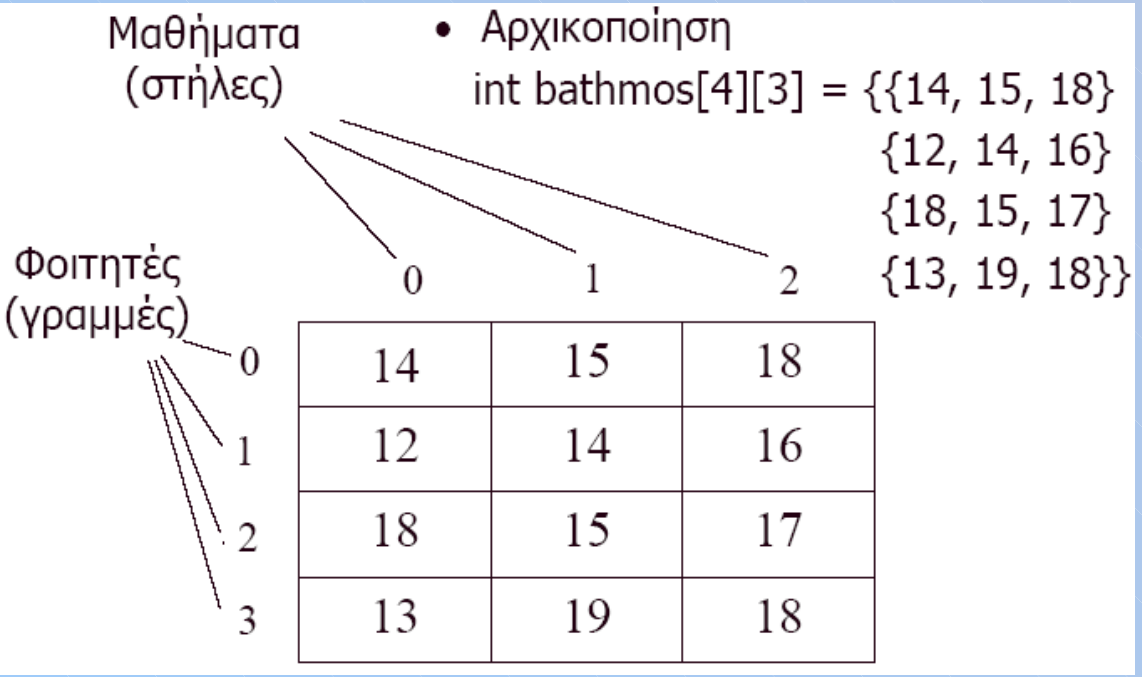
Δυσδιάστατος Πίνακας

- Δήλωση δυσδιάστατου πίνακα

  π.χ int bathmos[10][4];

  (πίνακας 10 στοιχείων, κάθε στοιχείο πίνακας 4 ακεραίων ή πίνακας 10 γραμμών και 4 στηλών)

- Αποθήκευση στη μνήμη κατά γραμμές (σαν μονοδιάστατος)

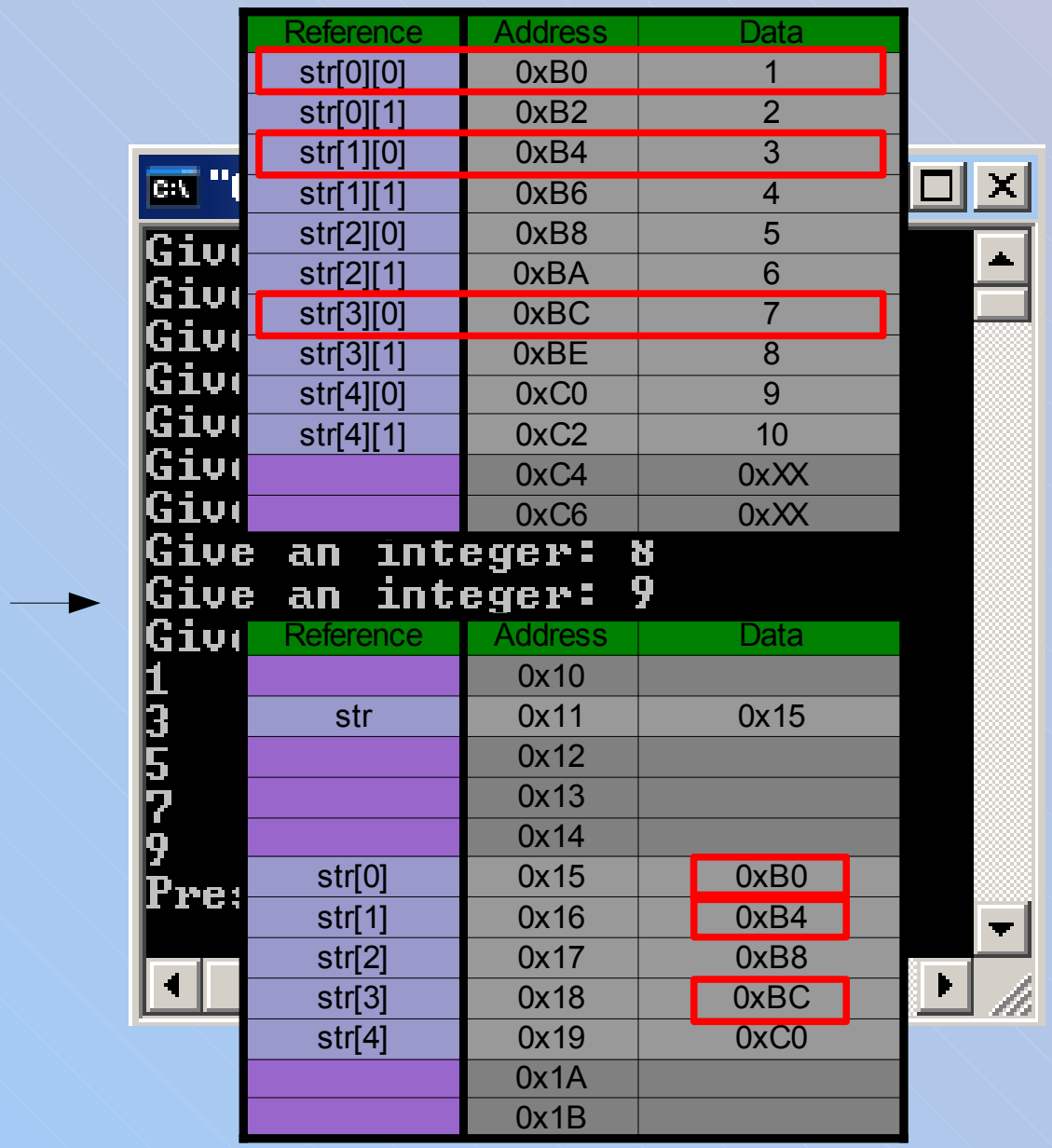


- Αρχικοποίηση

```
int bathmos[4][3] = {{14, 15, 18}
                     {12, 14, 16}
                     {18, 15, 17}
                     {13, 19, 18}};
```

Διάβασμα-Εκτύπωση Δυσδιάστατου Πίνακα

```c
#include <stdio.h>
#define MAX_X 5
#define MAX_Y 2

int main(void) {
    int str[MAX_X][MAX_Y]; int i, j;

    for (i = 0; i < MAX_X; i++) {
        for (j = 0; j < MAX_Y; j++){
            printf("Give an integer: ");
            scanf("%d", &str[i][j]);
        }
    }
    for (i = 0; i < MAX_X; i++){
        for (j = 0; j < MAX_Y; j++) {
            printf("%d\t", str[i][j]);
        }
        printf("\n");
    }
    return 1;
}
```



Συναρτήσεις

Οι συναρτήσεις, αποτελούν κομμάτια κώδικα, τα οποία μπορούν να καλεστούν από το κυρίως πρόγραμμα, ή από άλλες συναρτήσεις.

Η main() είναι κι αυτή μια συνάρτηση - η πρώτη που καλείται όταν τρέχει κάποιο C πρόγραμμα.

Χρησιμοποιούνται για να τμηματοποιήσουν τον κώδικα σε υπομέρους κομμάτια τα οποία μπορούν να διαχειριστούν ευκολότερα.

Χρησιμεύουν:

- Βελτίωση αναγνωσιμότητας
- Ευκολότερη συντήρηση
- Αποφυγή επαναλήψεων

Δήλωση Συνάρτησης

- Ο ορισμός συνάρτησης αποτελείται από δυο τμήματα
  - την επικεφαλίδα
  - την ομάδα εντολών, μέσα στα άγκιστρα του σώματος εντολών.

- Ο ορισμός μιας συνάρτησης έχει την ακόλουθη μορφή:

```
<τύπος επιστρεφόμενης τιμής> ή <void> <όνομα_συνάρτησης> (<λίστα παραμέτρων> ή <void>)
{
    ομάδα εντολών;
    return επιστρεφόμενη τιμή; ή τίποτα αν έχουμε δηλώσει void
}
```

Κλήση συνάρτησης

- Καλείται η συνάρτηση για εκτέλεση με συγκεκριμένα ορίσματα:

<όνομα-συν> (<όρισ1>, <όρισ2>, ... ,<όρισΝ>);

πραγματικά ορίσματα (σταθερές, μεταβλητές, εκφράσεις)

- Τα πραγματικά ορίσματα πρέπει να είναι του ίδιου αριθμού και τύπου με τα τυπικά ορίσματα

π.χ

```
swap (x, y);
draw_circle (a/2.0, 2.0 *b, c);
max_num = max (num1, num2);
x = y + max (num1/2.0, 3.0*num2);
printf("ο μέγιστος είναι: %d\n", max (x1, x2));
```

Μηχανισμός κλήσης συνάρτησης

Έχουμε την συνάρτηση:

```
int max(int a, int b) {
    return ( (a>b) ? a : b );
}
```

```
if (a > b) {
    return a;
} else {
    return b;
}
```

Έστω καλείται η max:

```
int max_num = max(12,17);
```

max: a = 12;

max: b = 17;

```
if (12 > 17) {
    return 12;
} else {
    return 17;
}
```

Θέσεις δηλώσεων συναρτήσεων

Οι συναρτήσεις προτού χρησιμοποιηθούν θα πρέπει να έχουν δηλωθεί ή και οριστεί.

Δήλωση πρώτα

```c
#include <stdio.h>

int square(int y);

int main(void) {
      int x;
      for (x=1; x <= 10; x++) {
            printf("%d\n",square(x));
      }
      return 1;
}

int square(int y) {
      return y*y;
}
```

Ορισμός πρώτα

```c
#include <stdio.h>

int square(int y) {
      return y*y;
}

int main(void) {
      int x;
      for (x=1; x <= 10; x++) {
            printf("%d\n",square(x));
      }
      return 1;
}
```

Αρχεία κεφαλίδας

Τα αρχεία κεφαλίδας (Header files) αποτελούν αρχεία που περιέχουν ορισμούς συναρτήσεων ή τύπων μεταβλητών, τα οποία ενσωματώνονται στα .c αρχεία με την εντολή #include.

Π.χ.

```
#include "Graphics.h"
#include <stdio.h>
#include "c:/dev_c/extras/windows/winbgi32.h"
```

Τα σύμβολα < και > χρησιμοποιούνται αν τα αρχεία .h βρίσκονται σε εξ' ορισμού καταλόγους, για τους οποίους είναι ενημερωμένο το περιβάλλον ανάπτυξης, ενώ τα σύμβολα " χρησιμοποιούνται για τον κατάλογο του workspace ή για άλλου καταλόγους που μπορεί να υπάρχουν τα .h αρχεία.

Αρχεία κεφαλίδας

- Τα αρχεία κεφαλίδας χρησιμοποιούνται για να μπορούν πολλά αρχεία .c να μπορούν να αναγνωρίσουν την ύπαρξη συναρτήσεων και μεταβλητών και κατά συνέπεια να μπορούν να επικοινωνήσουν / συνδυαστούν μεταξύ τους.

- Καλό είναι να έχουν την δομή:

```
#ifndef <Var>
#define <Var>
...
#endif
```

Έτσι ένα αρχείο .h το οποίο γίνεται include ταυτόχρονα σε πολλά .c αρχεία, δεν θα αντιγραφτεί πολλές φορές - μια σε κάθε αρχείο προκαλώντας σφάλμα στον linker - αλλά μόνο μια φορά.

- Συνιστάται κάθε .h αρχείο να αντιστοιχεί σε ένα και μόνο .c αρχείο, δηλαδή να περιέχει μόνο τους ορισμούς των συναρτήσεων που ορίζονται σε αυτό το .c αρχείο, καθώς και τους ορισμούς των τιμών και δομών που αυτό χρησιμοποιεί (ορισμοί #define, και structs).

Η φιλοσοφία της C

Συστήματα και C

- Όπως όλες οι γλώσσες προγραμματισμού και η C αποτελεί έναν υψηλού-επιπέδου τρόπο ανάθεσης διαδοχικών εντολών σε ψηφιακά συστήματα επεξεργαστών.

- Στην C ο χρήστης υλοποιεί με γραπτές εντολές τον αλγόριθμό του, σε αρχεία κειμένου με επέκταση .c και .h, και κάνοντας χρήση συγκεκριμένων εργαλείων που του παρέχει το περιβάλλον ανάπτυξης κάθε φορά, μετατρέπει τον αλγόριθμο σε κώδικα μηχανής.

- Σε μικρο-υπολογιστικά συστήματα, ο τρόπος υλοποίησης αλγορίθμων σε C απαιτεί την πλήρη γνώση της διασύνδεσης κατά hardware των αυτών συστημάτων. Γι' αυτό η C βρίσκει μεγάλη εφαρμογή σε τέτοιου είδους συστήματα.

Δημιουργία εκτελέσιμων αρχείων

- Η C πλέον παρέχεται με ολοκληρωμένα περιβάλλοντα ανάπτυξης (IDE), τα οποία εκτός από εξειδικευμένους text-editors, παρέχουν περισσότερες λειτουργίες που βοηθούν στην ευκολότερη και γρηγορότερη δημιουργία κώδικα (μέσω οπτικών διατάξεων, π.χ. MFC), καθώς και στην καλύτερη αποσφαλμάτωση (debuggers).

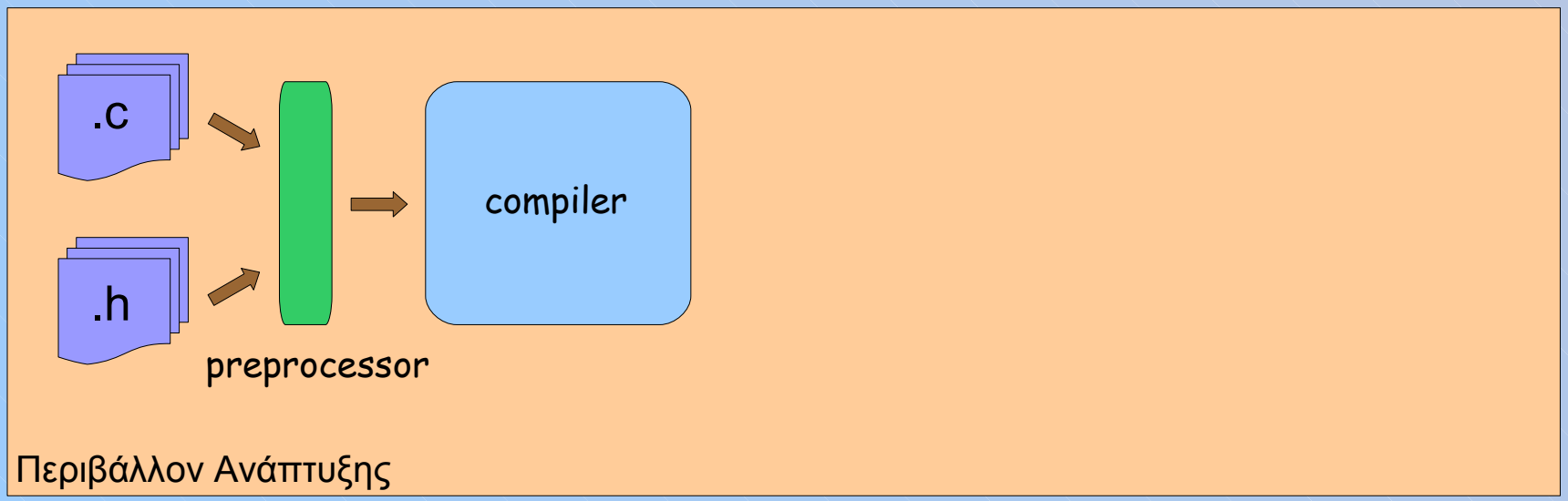
Preprocessor

- Ο Preprocessor είναι ένα πρόγραμμα το οποίο χρησιμοποιεί εντολές που αρχίζουν από το σύμβολο # προκειμένου να εκτελέσει διάφορες διεργασίες κειμένου πάνω στον πηγαίο κώδικα των αρχείων .c και .h
- Τέτοιες διεργασίες είναι π.χ., η αντιγραφή των δεδομένων ενός .h αρχείου μέσα σε ένα .c αρχείο (# include<>), ο ορισμός συμβολοσειρών (#define), η προσθήκη κώδικα υπό συνθήκες (#ifdef, #ifndef, #end), κ.τ.λ.

Compiler

- Ο compiler παίρνει τα αρχεία που παράγει ο preprocessor και δημιουργεί για κάθε αρχείο τον αντίστοιχο κώδικα μηχανής.

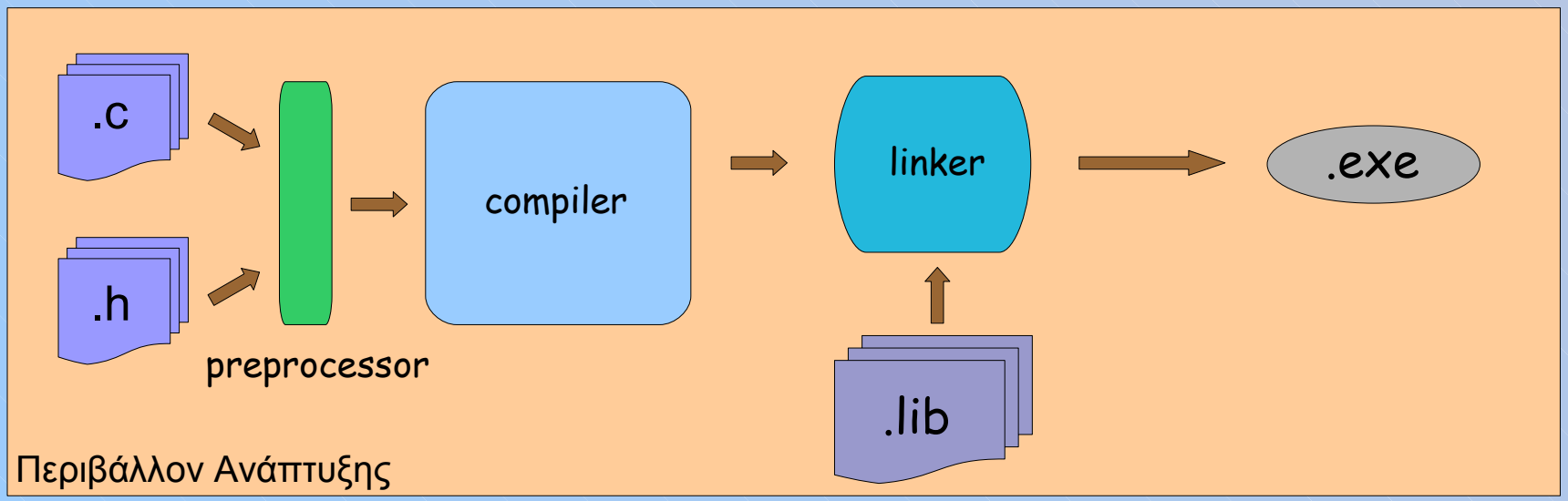
- Τα αρχεία που δημιουργεί έχουν συνήθως επέκταση .obj, τα οποία μπορούν να αποτελούν απλή συνάρτηση, μέχρι ένα ολοκληρωμένο πρόγραμμα.



Linker

- Ο Linker αποτελεί ένα τρίτο πρόγραμμα το οποίο ενώνει τα αρχεία του κώδικα μηχανής (.obj) που δημιουργήθηκαν από τον compiler, σε ένα άλλο αρχείο.

- Ταυτόχρονα, ενσωματώνει στο τελικό αρχείο απαραίτητους και ήδη υπάρχοντες κώδικες μηχανής που υπάρχουν σε αρχεία .lib.



C σε μικροϋπολογιστές

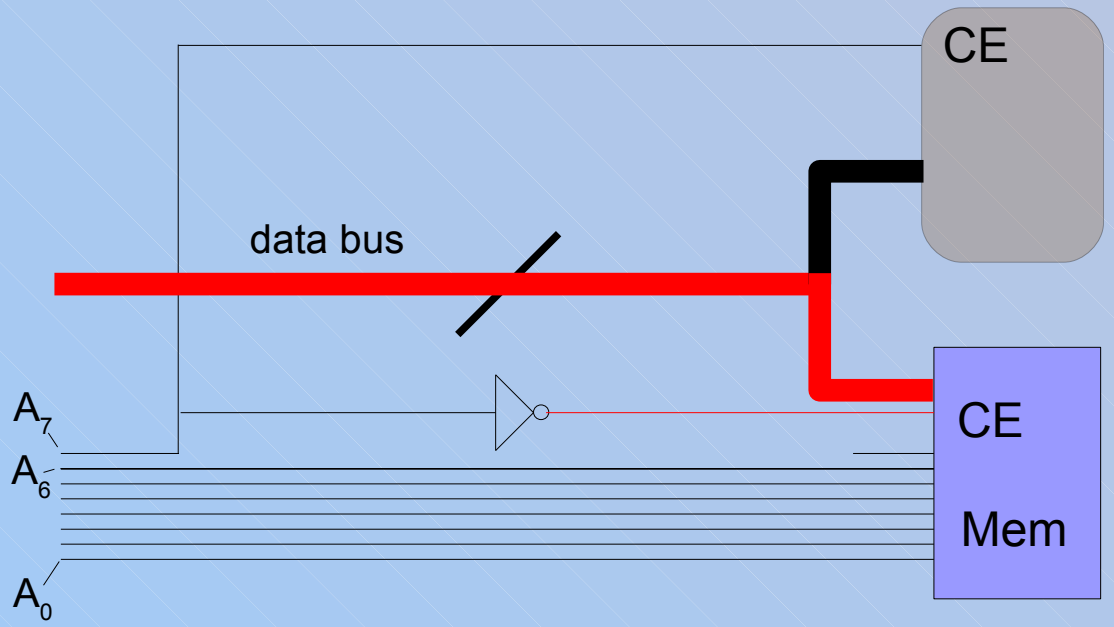
Όλα είναι ... μνήμη!

- Οι μικροϋπολογιστές στηρίζονται στην δυνατότητα αποθήκευσης, ανάκτησης και χρήσης δυαδικών τιμών σε στοιχεία μνήμης.

- Επίσης ο τρόπος επικοινωνίας των μικροϋπολογιστών με εξωτερικές ψηφιακές συσκευές, είναι παρόμοιος με τον τρόπο διαχείρισης της μνήμης. Διαχειρίζονται δηλαδή τα δεδομένα επικοινωνίας, σαν να διαχειρίζονταν δεδομένα σε κάποια υποθετική μνήμη.

Αναφορά σε θέσεις μνήμης 0x00 έως 0x7F ενεργοποιούν το chip μνήμης, και επομένως είναι δυνατή η μεταβίβαση δεδομένων από και προς την μνήμη.

Memory selected with: 0xxx xxxx



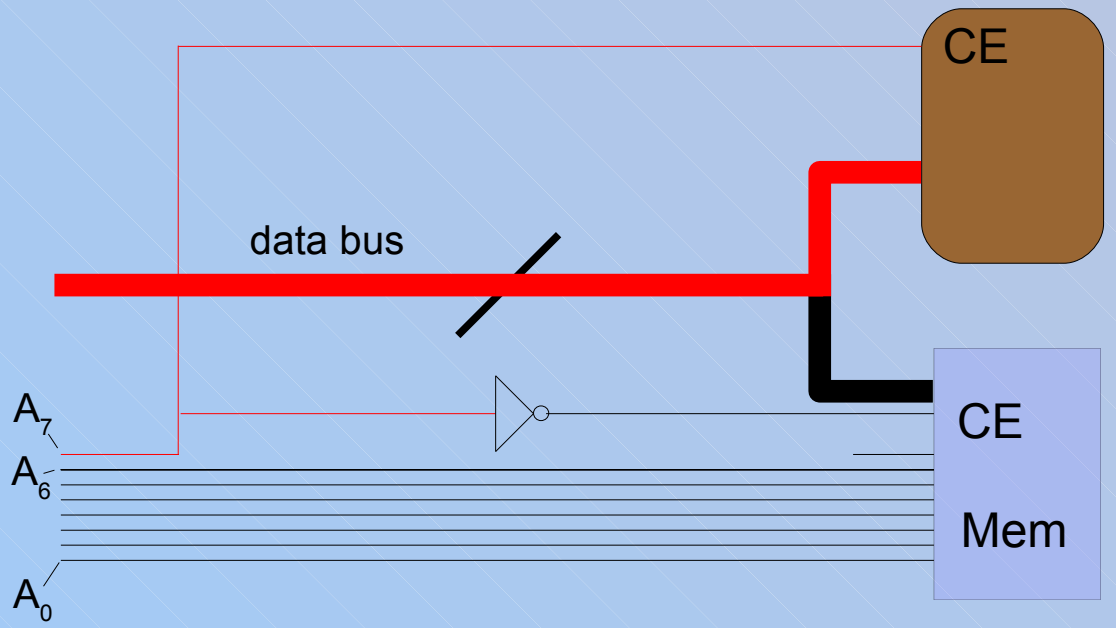
Όλα είναι ... μνήμη!

- Οι μικροϋπολογιστές στηρίζονται στην δυνατότητα αποθήκευσης, ανάκτησης και χρήσης δυαδικών τιμών σε στοιχεία μνήμης.

- Επίσης ο τρόπος επικοινωνίας των μικροϋπολογιστών με εξωτερικές ψηφιακές συσκευές, είναι παρόμοιος με τον τρόπο διαχείρισης της μνήμης. Διαχειρίζονται δηλαδή τα δεδομένα επικοινωνίας, σαν να διαχειρίζονταν δεδομένα σε κάποια υποθετική μνήμη.

Αναφορά σε θέσεις μνήμης 0x80 έως 0xFF ενεργοποιούν την περιφερειακή συσκευή, και έτσι είναι δυνατή η μεταβίβαση δεδομένων από και προς το συγκεκριμένο chip.

Memory selected with: 1xxx xxxx



Μεταβλητές στην C

Οι μεταβλητές στην C χαρακτηρίζονται κυρίως από 4 πράγματα.

- Την θέση μνήμης που βρίσκονται.
- Τον αριθμό των bytes που χρησιμοποιούν για να αναπαραστήσουν αριθμούς.
- Τον τρόπο διαχείρισής τους (π.χ. long, float, pointers κ.τ.λ.).
- Το πεδίο ορισμού τους.

Λέξη	Bytes	φάσμα
char	1	-128 εως 127
int	2	-32768 εως 32767
short	2	-32768 εως 32767
long	4	-21477483648 εως 21477483647
unsigned char	1	0 εως 255
unsigned int	2	0 εως 65535
unsigned short	2	0 εως 65535
unsigned long	4	Ο εως 4.294.967.295
float	4	1.2E – 38 εως 3.4E38[1]
double	8	2.2E – 308 εως 1.8E308[2]

Δήλωση μεταβλητών στη C

```
int x; x=18321; // x=0x4791 - Little Endian
```

0xB0	1001 0001 (91^H)
0xB1	0100 0111 (47^H)
0xB2	
0xB3	
0xB4	
0xB5	
0xB6	
0xB7	
0xB8	
0xB9	
0xBA	
0xBB	
0xBC	
0xBD	
0xBE	
0xBF	
0xC0	
0xC1	
0xC2	

Δήλωση μεταβλητών στη C

```
int x; x=18321; // x=0x4791 - Little Endian

char  b = 123; // b = 0x7B - Little Endian
```

0xB0	1001 0001 (91^H)
0xB1	0100 0111 (47^H)
0xB2	0111 1011 ($7B^H$)
0xB3	
0xB4	
0xB5	
0xB6	
0xB7	
0xB8	
0xB9	
0xBA	
0xBB	
0xBC	
0xBD	
0xBE	
0xBF	
0xC0	
0xC1	
0xC2	

Δήλωση μεταβλητών στη C

```
int x; x=18321; // x=0x4791 - Little Endian

char  b = 123; // b = 0x7B - Little Endian


long (xdata) Power = 0xAAFFAAEE _at_ 0xBB;
    // C51: 4 bytes, at address mem: 0xBB
```

0xB0	1001 0001 (91^H)
0xB1	0100 0111 (47^H)
0xB2	0111 1011 ($7B^H$)
0xB3	
0xB4	
0xB5	
0xB6	
0xB7	
0xB8	
0xB9	
0xBA	
0xBB	1110 1110 (EE^H)
0xBC	1010 1010 (AA^H)
0xBD	1111 1111 (FF^H)
0xBE	1010 1010 (AA^H)
0xBF	
0xC0	
0xC1	
0xC2	

Δήλωση μεταβλητών στη C

```
int x; x=18321; // x=0x4791 - Little Endian

char  b = 123; // b = 0x7B - Little Endian


long (xdata) Power = 0xAAFFAAEE _at_ 0xBB;
     // C51: 4 bytes, at address mem: 0xBB


unsigned int y = 1; // y=0x0001 - Little Endian
```

0xB0	1001 0001 (91^H)
0xB1	0100 0111 (47^H)
0xB2	0111 1011 ($7B^H$)
0xB3	0000 0001 (01^H)
0xB4	0000 0000 (00^H)
0xB5	
0xB6	
0xB7	
0xB8	
0xB9	
0xBA	
0xBB	1110 1110 (EE^H)
0xBC	1010 1010 (AA^H)
0xBD	1111 1111 (FF^H)
0xBE	1010 1010 (AA^H)
0xBF	
0xC0	
0xC1	
0xC2	

Δήλωση μεταβλητών στη C

```
int x; x=18321; // x=0x4791 - Little Endian

char  b = 123; // b = 0x7B - Little Endian


long (xdata) Power = 0xAAFFAAEE _at_ 0xBB;
     // C51: 4 bytes, at address mem: 0xBB


unsigned int y = 1; // y=0x0001 - Little Endian

char Enable = 0 _at_ 0xC2;
     // C51: 1 byte at address mem: 0xC2
```

0xB0	1001 0001 (91^H)
0xB1	0100 0111 (47^H)
0xB2	0111 1011 ($7B^H$)
0xB3	0000 0001 (01^H)
0xB4	0000 0000 (00^H)
0xB5	
0xB6	
0xB7	
0xB8	
0xB9	
0xBA	
0xBB	1110 1110 (EE^H)
0xBC	1010 1010 (AA^H)
0xBD	1111 1111 (FF^H)
0xBE	1010 1010 (AA^H)
0xBF	
0xC0	
0xC1	
0xC2	0000 0000 (00^H)

Δείκτες στην C

- Οι δείκτες είναι και αυτοί μεταβλητές με την διαφορά πως η τιμή ενός δείκτη, αναφέρεται πάντα σε διεύθυνση μνήμης.

- Δηλώνονται με τον ίδιο τρόπο με τις μεταβλητές, χρησιμοποιώντας μπροστά το σύμβολο *.

```
int *px; /* Ο px είναι δείκτης σε έναν ακέραιο.
            Θεωρώντας ότι το σύστημα έχει δυνατότητα
            διευθυνσιοδότησης 65K, ο δείκτης
            χρειάζεται 2 Byte μνήμη. */
```

0x00A0	XXXX XXXX (XX^H)
0x00A1	XXXX XXXX (XX^H)
0x00A2	
0x00A3	
0x00A4	
0x00A5	
0x00A6	
0x00A7	
0x00A8	
0x00A9	
0x00AA	
0x00AB	
0x00AC	
0x00AD	
0x00AE	
0x00AF	

Δείκτες στην C

- Οι δείκτες είναι και αυτοί μεταβλητές με την διαφορά πως η τιμή ενός δείκτη, αναφέρεται πάντα σε διεύθυνση μνήμης.

- Δηλώνονται με τον ίδιο τρόπο με τις μεταβλητές, χρησιμοποιώντας μπροστά το σύμβολο *.

```
int *px; /* Ο px είναι δείκτης σε έναν ακέραιο.
            Θεωρώντας ότι το σύστημα έχει δυνατότητα
            διευθυνσιοδότησης 65K, ο δείκτης
            χρειάζεται 2 Byte μνήμη. */

int x=0x1234;
```

0x00A0	XXXX XXXX (XX^H)
0x00A1	XXXX XXXX (XX^H)
0x00A2	0011 0100 (34^H)
0x00A3	0001 0010 (12^H)
0x00A4	
0x00A5	
0x00A6	
0x00A7	
0x00A8	
0x00A9	
0x00AA	
0x00AB	
0x00AC	
0x00AD	
0x00AE	
0x00AF	

Δείκτες στην C

- Οι δείκτες είναι και αυτοί μεταβλητές με την διαφορά πως η τιμή ενός δείκτη, αναφέρεται πάντα σε διεύθυνση μνήμης.

- Δηλώνονται με τον ίδιο τρόπο με τις μεταβλητές, χρησιμοποιώντας μπροστά το σύμβολο *.

```
int *px; /* Ο px είναι δείκτης σε έναν ακέραιο.
            Θεωρώντας ότι το σύστημα έχει δυνατότητα
            διευθυνσιοδότησης 65K, ο δείκτης
            χρειάζεται 2 Byte μνήμη. */

int x=0x1234;

px = &x; // Ο px παίρνει την διεύθυνση του x.
```

0x00A0	1010 0010 ($A2^H$)
0x00A1	0000 0000 (00^H)
0x00A2	0011 0100 (34^H)
0x00A3	0001 0010 (12^H)
0x00A4	
0x00A5	
0x00A6	
0x00A7	
0x00A8	
0x00A9	
0x00AA	
0x00AB	
0x00AC	
0x00AD	
0x00AE	
0x00AF	

Δείκτες στην C

- Οι δείκτες είναι και αυτοί μεταβλητές με την διαφορά πως η τιμή ενός δείκτη, αναφέρεται πάντα σε διεύθυνση μνήμης.

- Δηλώνονται με τον ίδιο τρόπο με τις μεταβλητές, χρησιμοποιώντας μπροστά το σύμβολο *.

```
int *px; /* Ο px είναι δείκτης σε έναν ακέραιο.
            Θεωρώντας ότι το σύστημα έχει δυνατότητα
            διευθυνσιοδότησης 65K, ο δείκτης
            χρειάζεται 2 Byte μνήμη. */

int x=0x1234;

px = &x; // Ο px παίρνει την διεύθυνση του x.

*px = 0x5678; // Αλλαγή δεδομ. εκεί που δείχνει ο px.
```

0x00A0	1010 0010 ($A2^H$)
0x00A1	0000 0000 (00^H)
0x00A2	0111 1000 (78^H)
0x00A3	0101 0110 (56^H)
0x00A4	
0x00A5	
0x00A6	
0x00A7	
0x00A8	
0x00A9	
0x00AA	
0x00AB	
0x00AC	
0x00AD	
0x00AE	
0x00AF	

Δείκτες στην C

- Οι δείκτες είναι και αυτοί μεταβλητές με την διαφορά πως η τιμή ενός δείκτη, αναφέρεται πάντα σε διεύθυνση μνήμης.

- Δηλώνονται με τον ίδιο τρόπο με τις μεταβλητές, χρησιμοποιώντας μπροστά το σύμβολο *.

```
*px = 0x5678; // Αλλαγή δεδομ. εκεί που δείχνει ο px.

px = 0x00AC; // Αλλαγή δεδομ. του px.
```

0x00A0	1010 1100 (AC^H)
0x00A1	0000 0000 (00^H)
0x00A2	0111 1000 (78^H)
0x00A3	0101 0110 (56^H)
0x00A4	
0x00A5	
0x00A6	
0x00A7	
0x00A8	
0x00A9	
0x00AA	
0x00AB	
0x00AC	
0x00AD	
0x00AE	
0x00AF	

Δείκτες στην C

- Οι δείκτες είναι και αυτοί μεταβλητές με την διαφορά πως η τιμή ενός δείκτη, αναφέρεται πάντα σε διεύθυνση μνήμης.

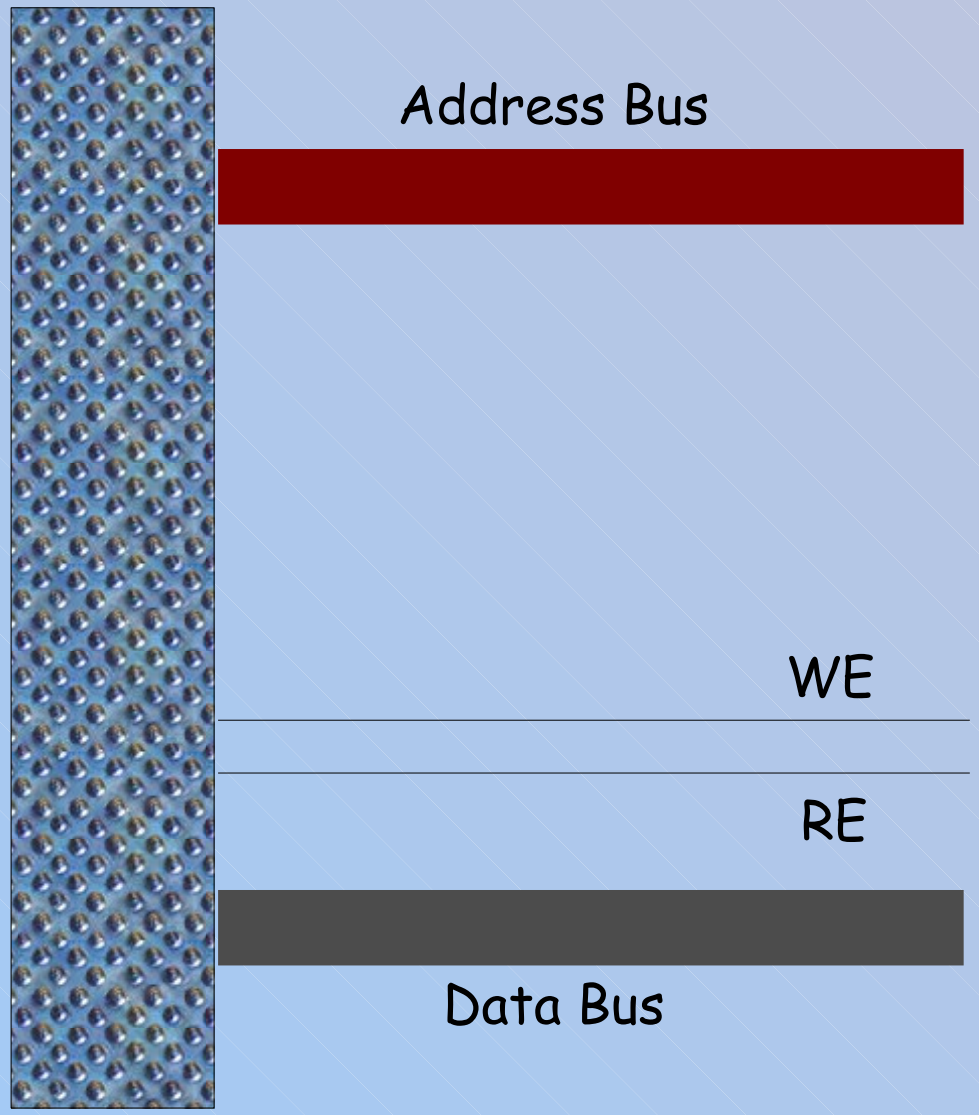
- Δηλώνονται με τον ίδιο τρόπο με τις μεταβλητές, χρησιμοποιώντας μπροστά το σύμβολο *.

```
*px = 0x5678; // Αλλαγή δεδομ. εκεί που δείχνει ο px.

px = 0x00AC; // Αλλαγή δεδομ. του px.

*px = 0xFEFA; // Αλλαγή δεδομ. εκεί που δείχνει ο px.
```

0x00A0	1010 1100 (AC^H)
0x00A1	0000 0000 (00^H)
0x00A2	0111 1000 (78^H)
0x00A3	0101 0110 (56^H)
0x00A4	
0x00A5	
0x00A6	
0x00A7	
0x00A8	
0x00A9	
0x00AA	
0x00AB	
0x00AC	1111 1010 (FA^H)
0x00AD	1111 1110 (FE^H)
0x00AE	
0x00AF	

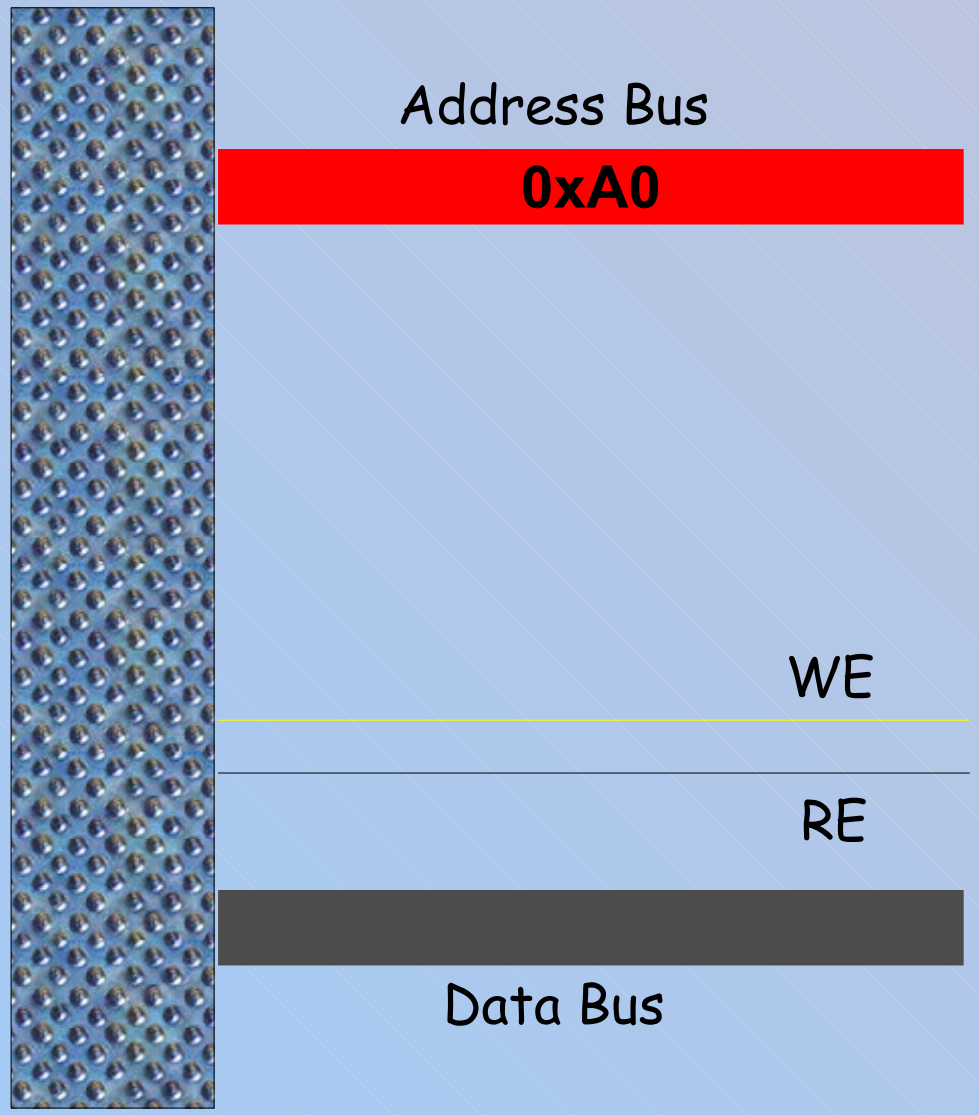
Διαδικασία εκτέλεσης απλών εντολών

```
char x, y; //Ορισμός μεταβλητών
           //στην μνήμη. Έστω το
           // x στην θέση: 0xA0 &
           // το y στην: 0xAA.
```



Διαδικασία εκτέλεσης απλών εντολών

```
char x, y; //Ορισμός μεταβλητών
           //στην μνήμη. Έστω το
           // x στην θέση: 0xA0 &
           // το y στην: 0xAA.

x = 5;
```

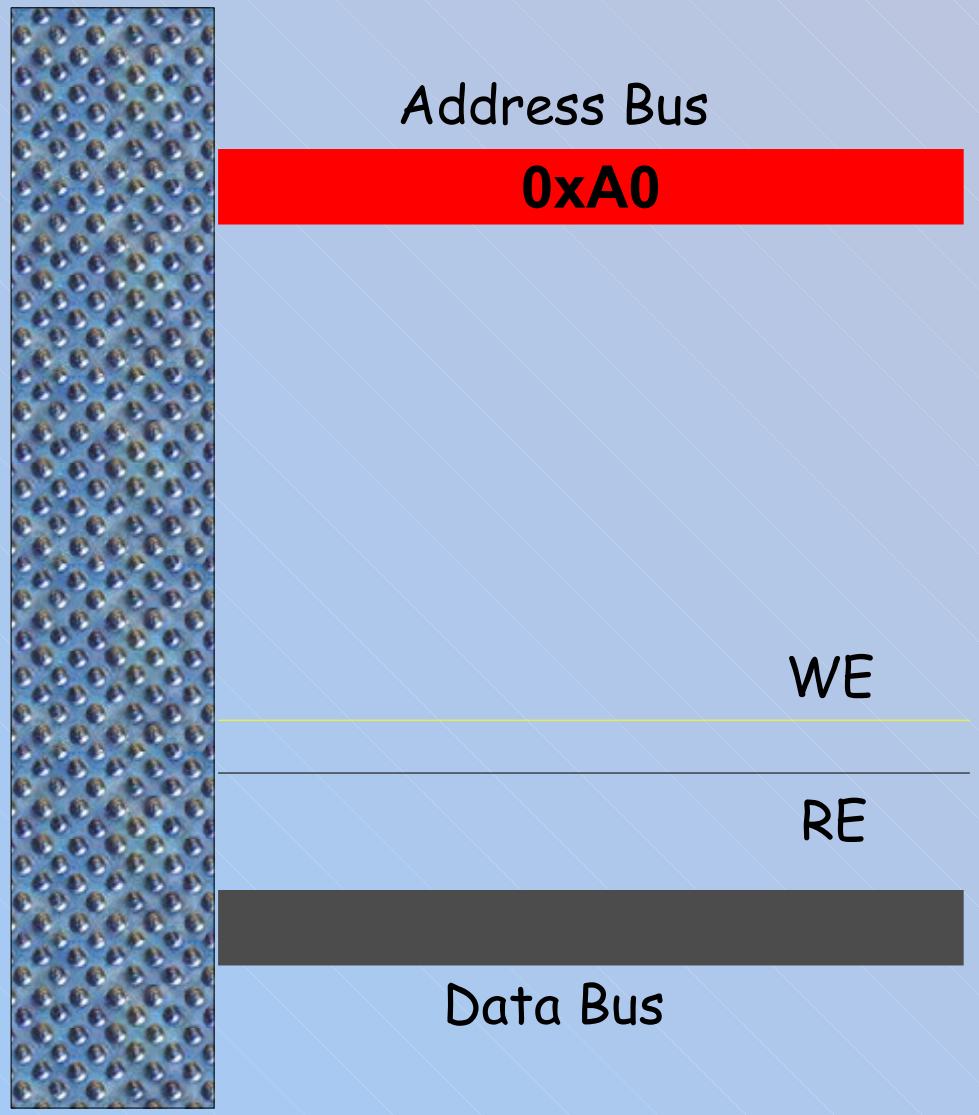


Διαδικασία εκτέλεσης απλών εντολών

```
char x, y; //Ορισμός μεταβλητών
           //στην μνήμη. Έστω το
           // x στην θέση: 0xA0 &
           // το y στην: 0xAA.

x = 5;
```

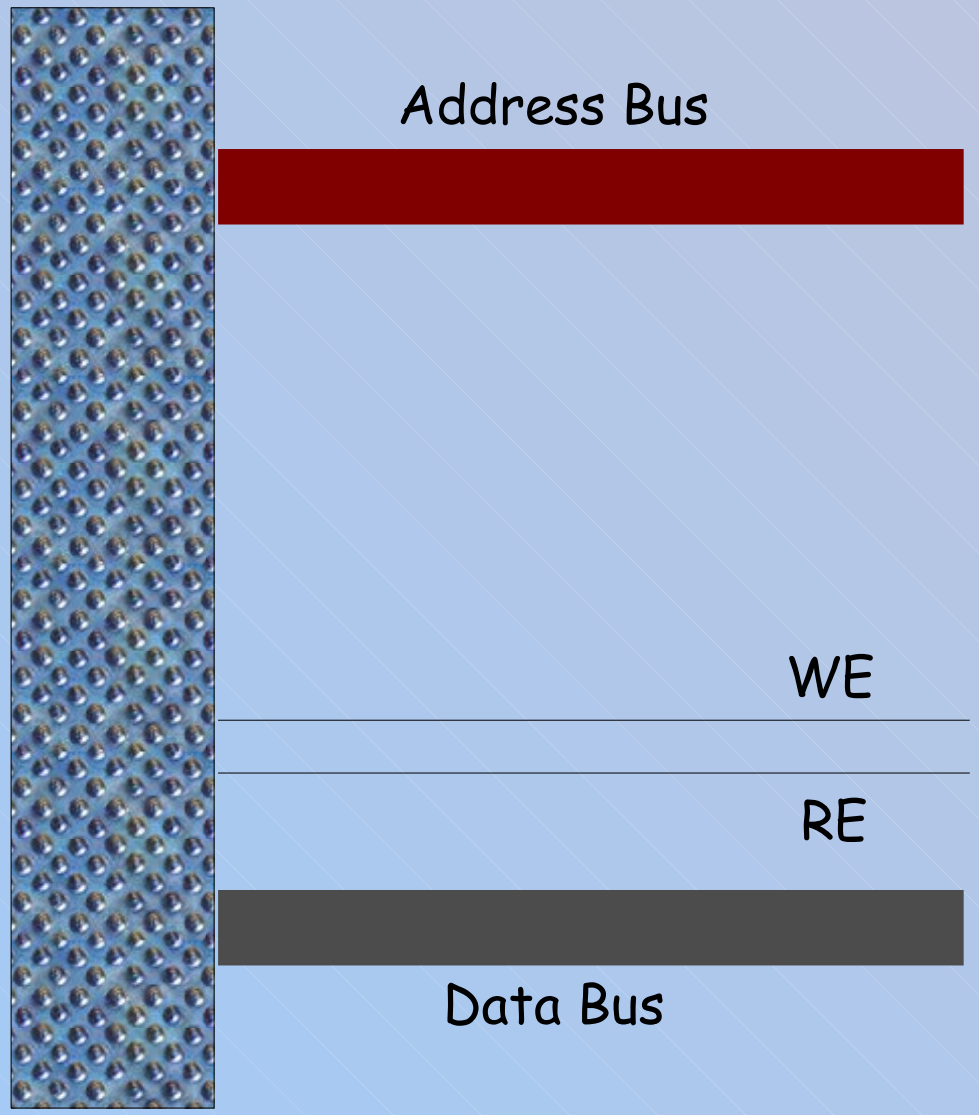
Διαδικασία εκτέλεσης απλών εντολών

```
char x, y; //Ορισμός μεταβλητών
           //στην μνήμη. Έστω το
           // x στην θέση: 0xA0 &
           // το y στην: 0xAA.

x = 5;
```



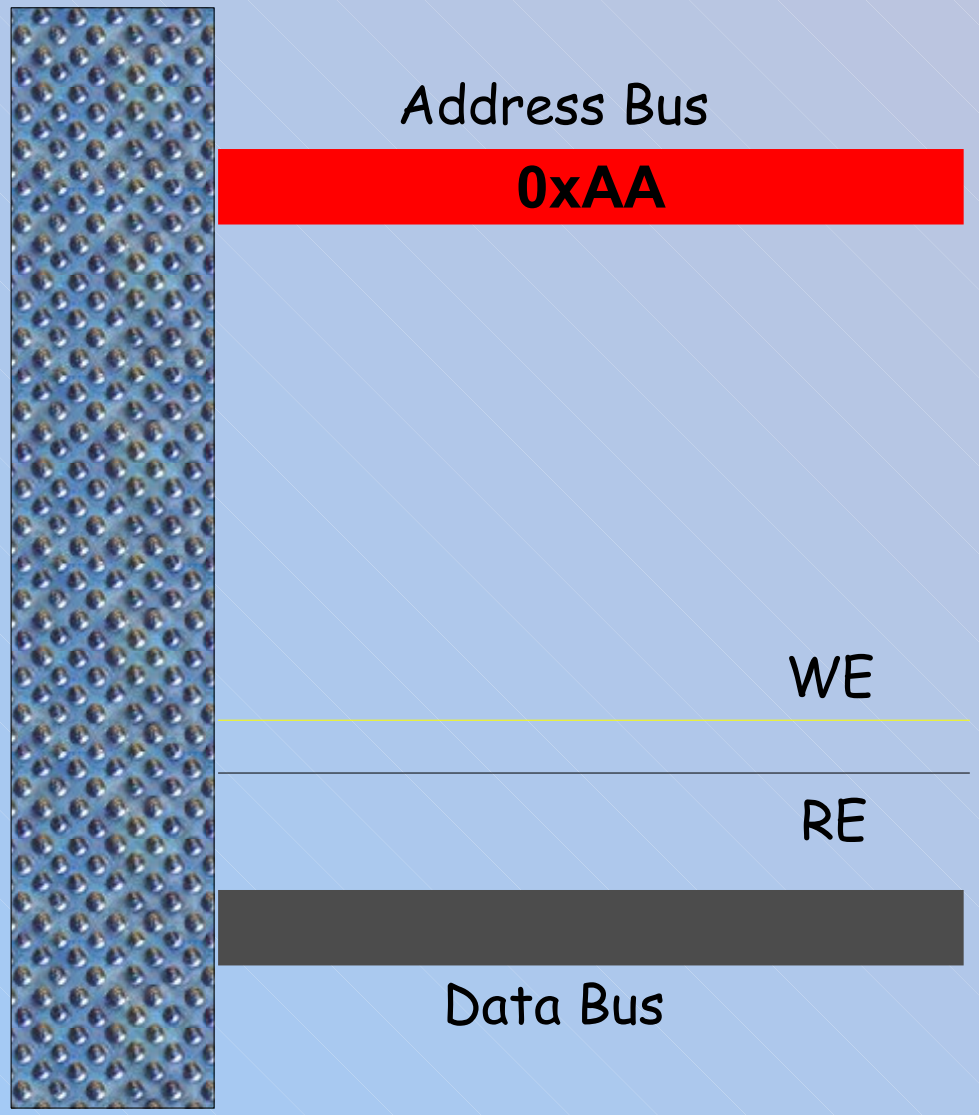
Διαδικασία εκτέλεσης απλών εντολών

```
char x, y; //Ορισμός μεταβλητών
           //στην μνήμη. Έστω το
           // x στην θέση: 0xA0 &
           // το y στην: 0xAA.

x = 5;
```



Διαδικασία εκτέλεσης απλών εντολών

```
char x, y; //Ορισμός μεταβλητών
           //στην μνήμη. Έστω το
           // x στην θέση: 0xA0 &
           // το y στην: 0xAA.

x = 5;
y = 3;
```

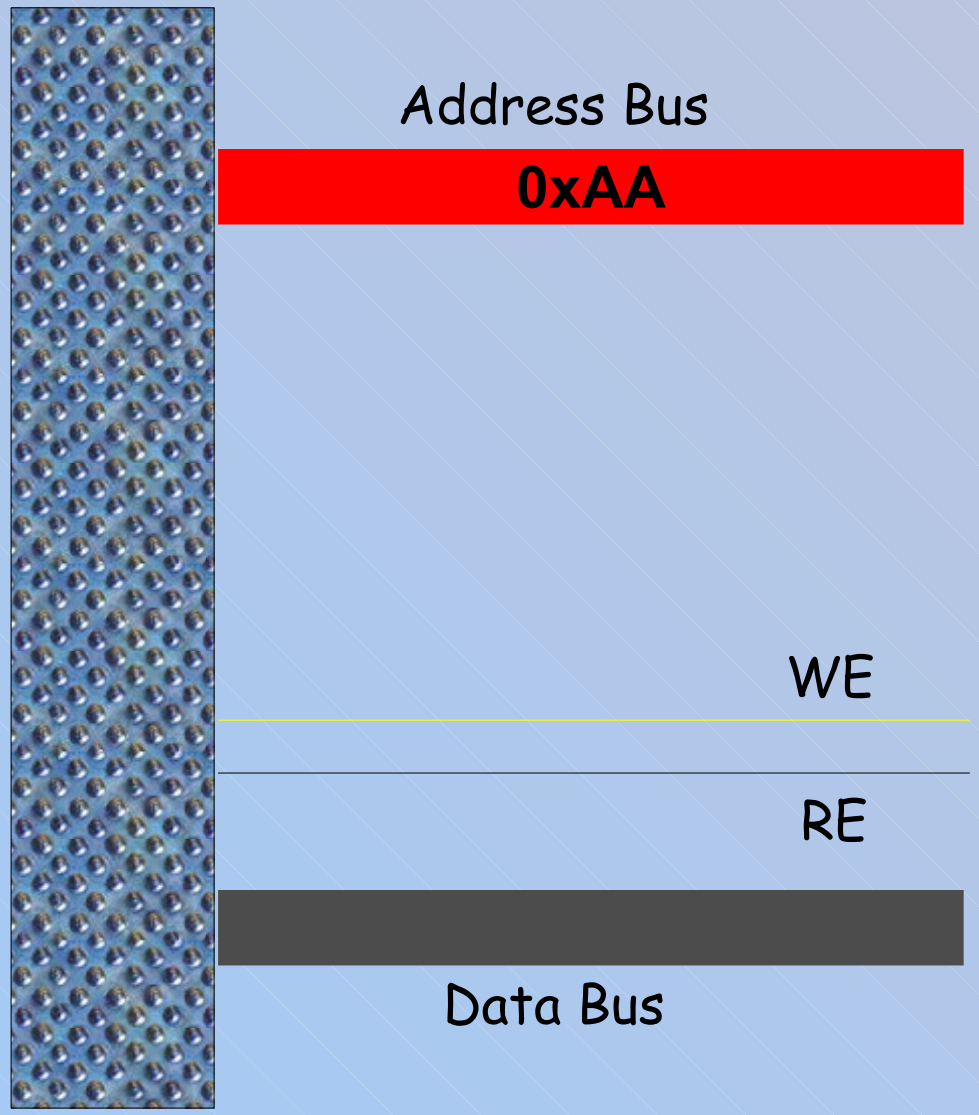


Διαδικασία εκτέλεσης απλών εντολών

```
char x, y; //Ορισμός μεταβλητών
           //στην μνήμη. Έστω το
           // x στην θέση: 0xA0 &
           // το y στην: 0xAA.

x = 5;
y = 3;
```

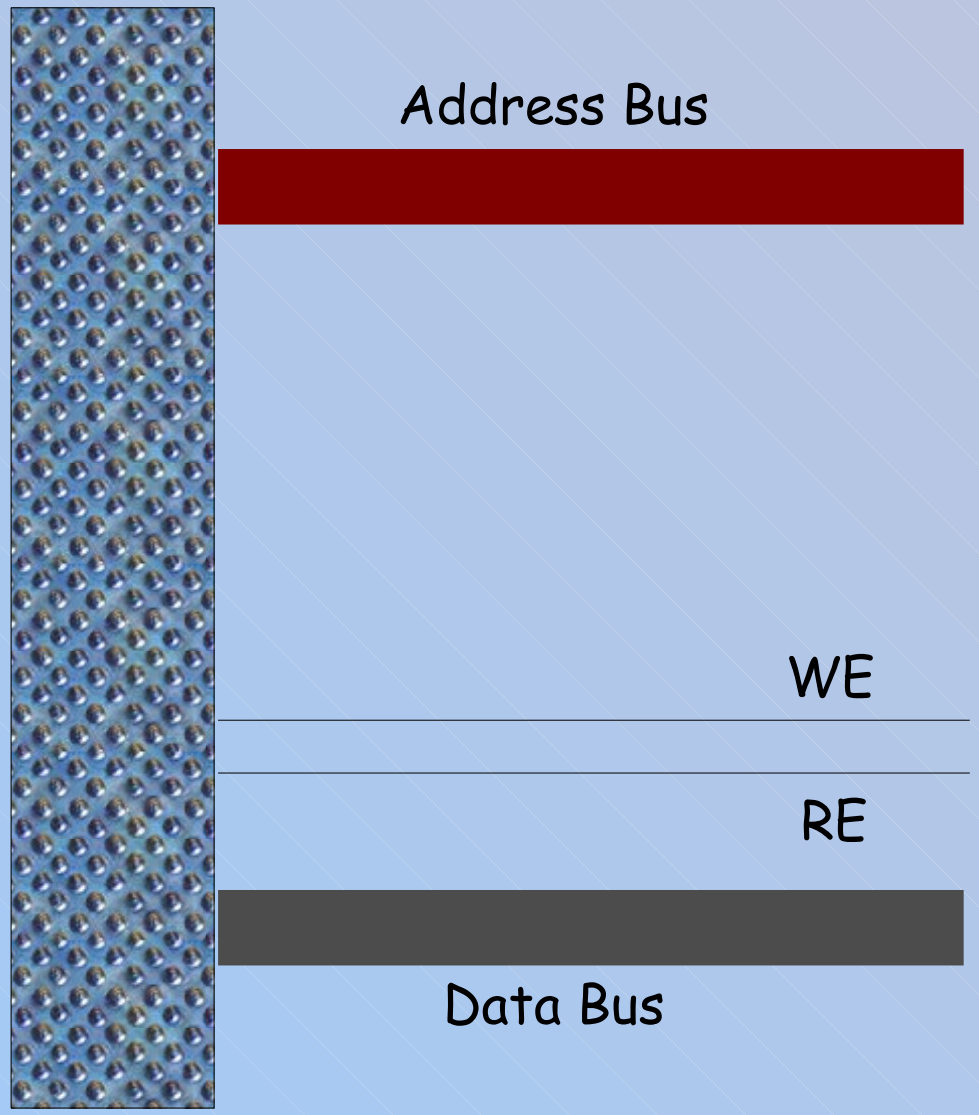
Διαδικασία εκτέλεσης απλών εντολών

```
char x, y; //Ορισμός μεταβλητών
           //στην μνήμη. Έστω το
           // x στην θέση: 0xA0 &
           // το y στην: 0xAA.

x = 5;
y = 3;
```



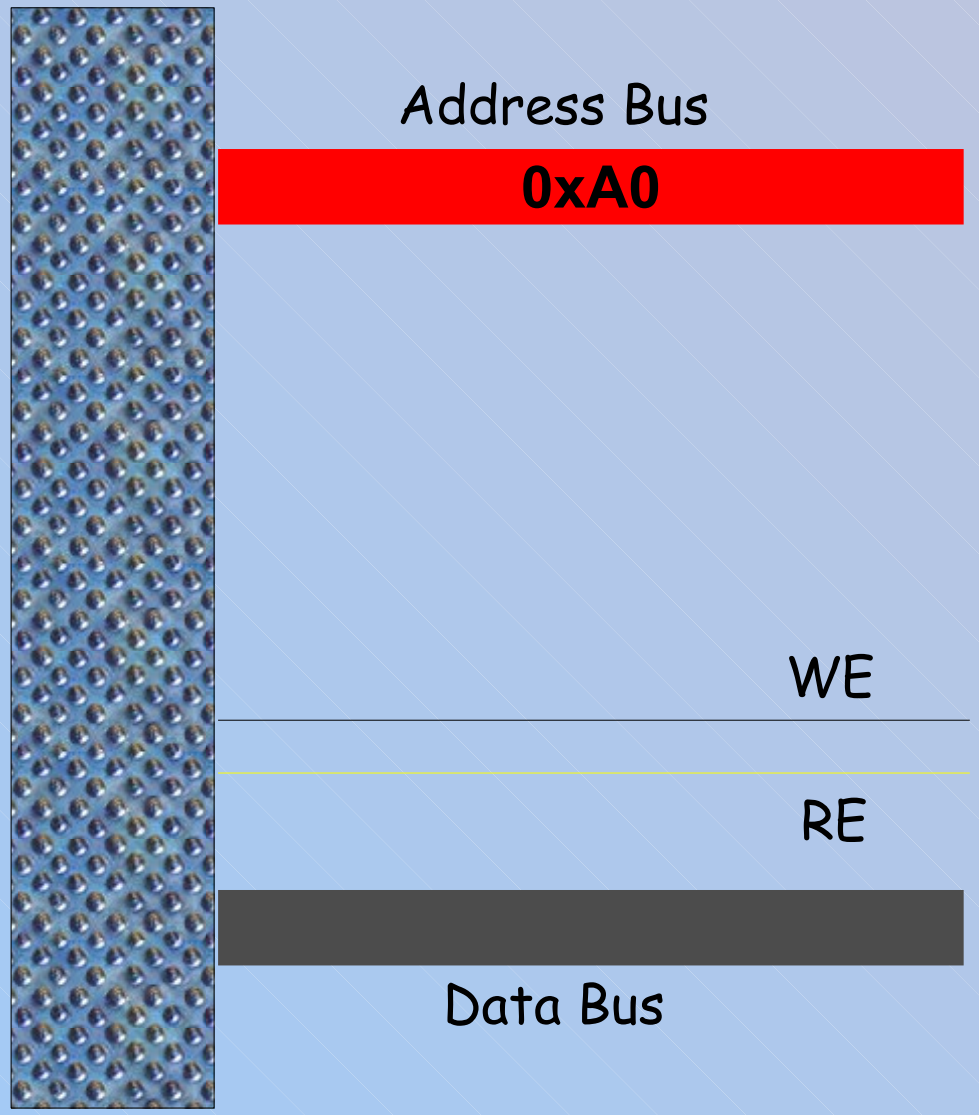
Διαδικασία εκτέλεσης απλών εντολών

```
char x, y; //Ορισμός μεταβλητών
           //στην μνήμη. Έστω το
           // x στην θέση: 0xA0 &
           // το y στην: 0xAA.

x = 5;
y = 3;
```



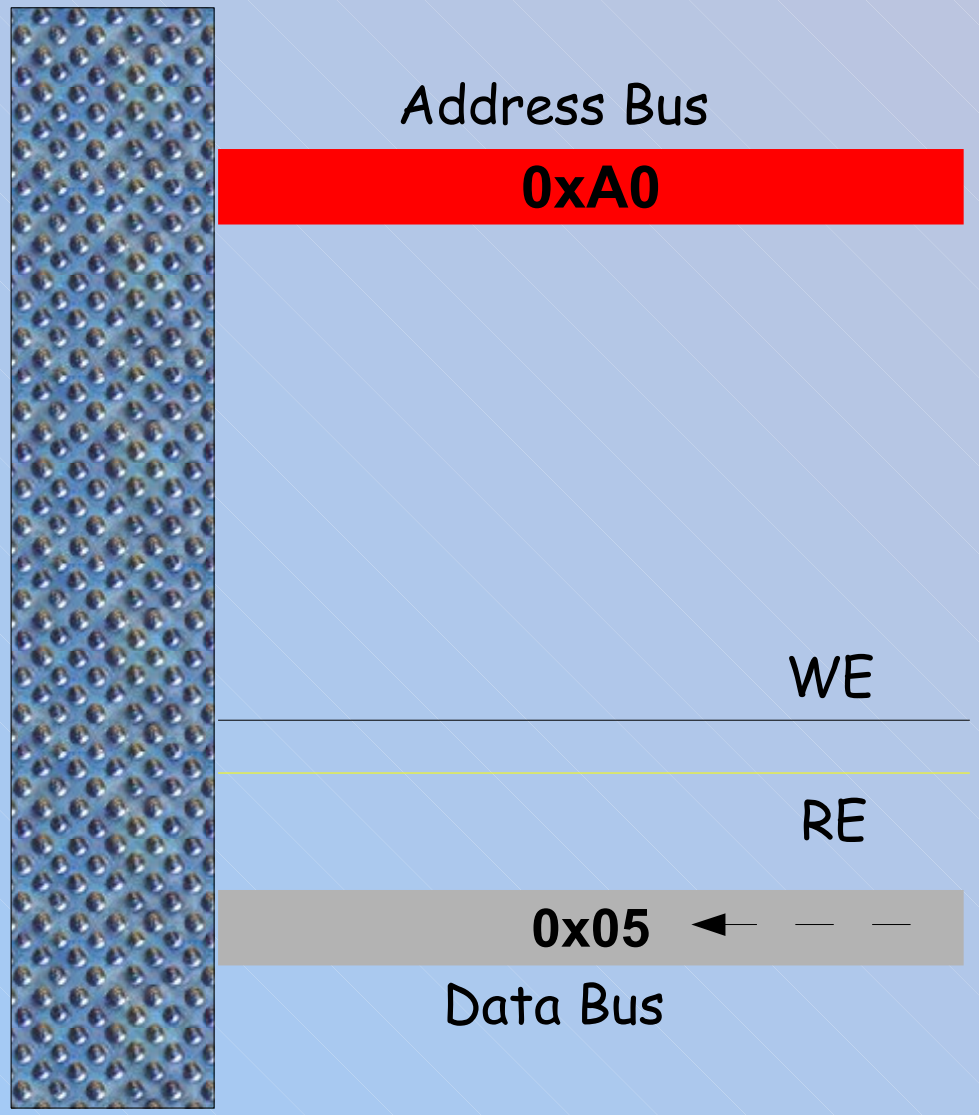
Διαδικασία εκτέλεσης απλών εντολών

```
char x, y; //Ορισμός μεταβλητών
           //στην μνήμη. Έστω το
           // x στην θέση: 0xA0 &
           // το y στην: 0xAA.

x = 5;
y = 3;

x = x + y;
```



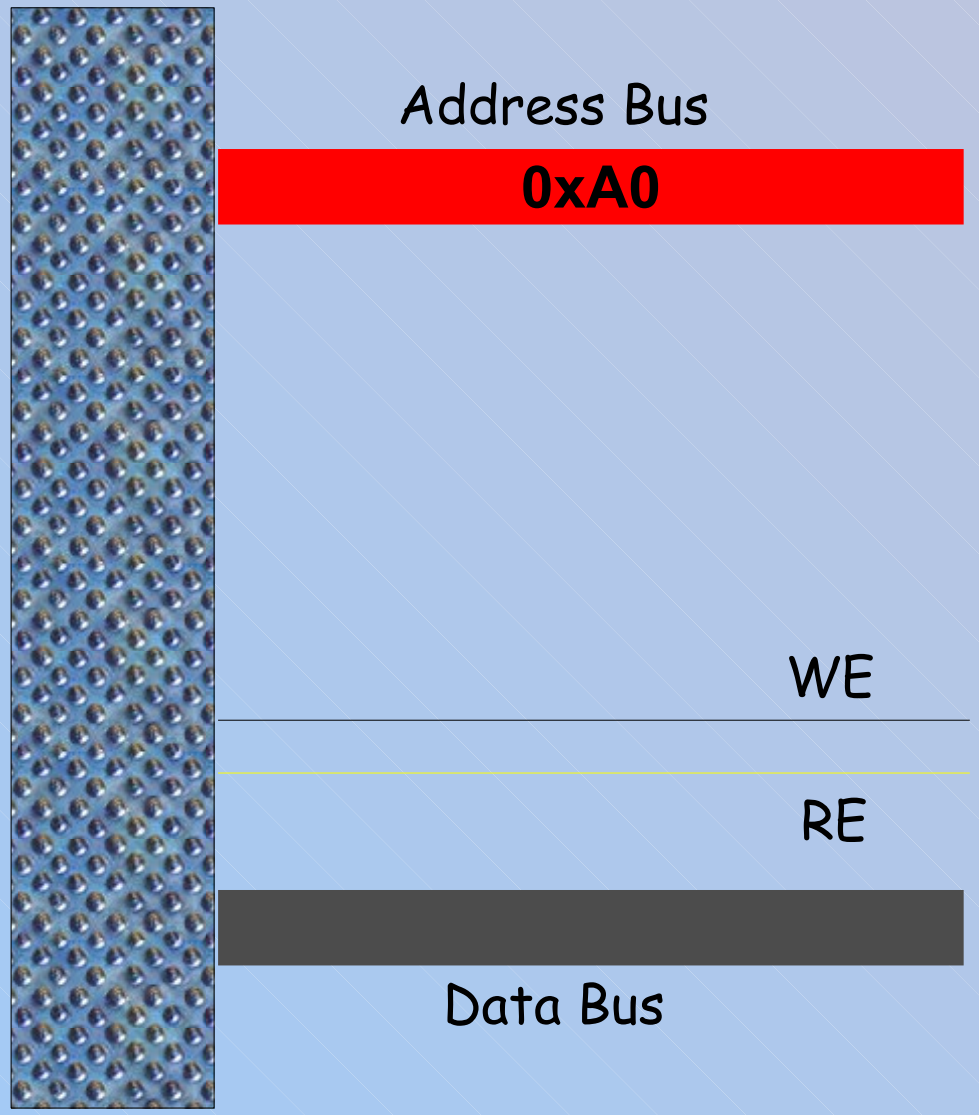
Διαδικασία εκτέλεσης απλών εντολών

```
char x, y; //Ορισμός μεταβλητών
           //στην μνήμη. Έστω το
           // x στην θέση: 0xA0 &
           // το y στην: 0xAA.

x = 5;
y = 3;

x = x + y;
```



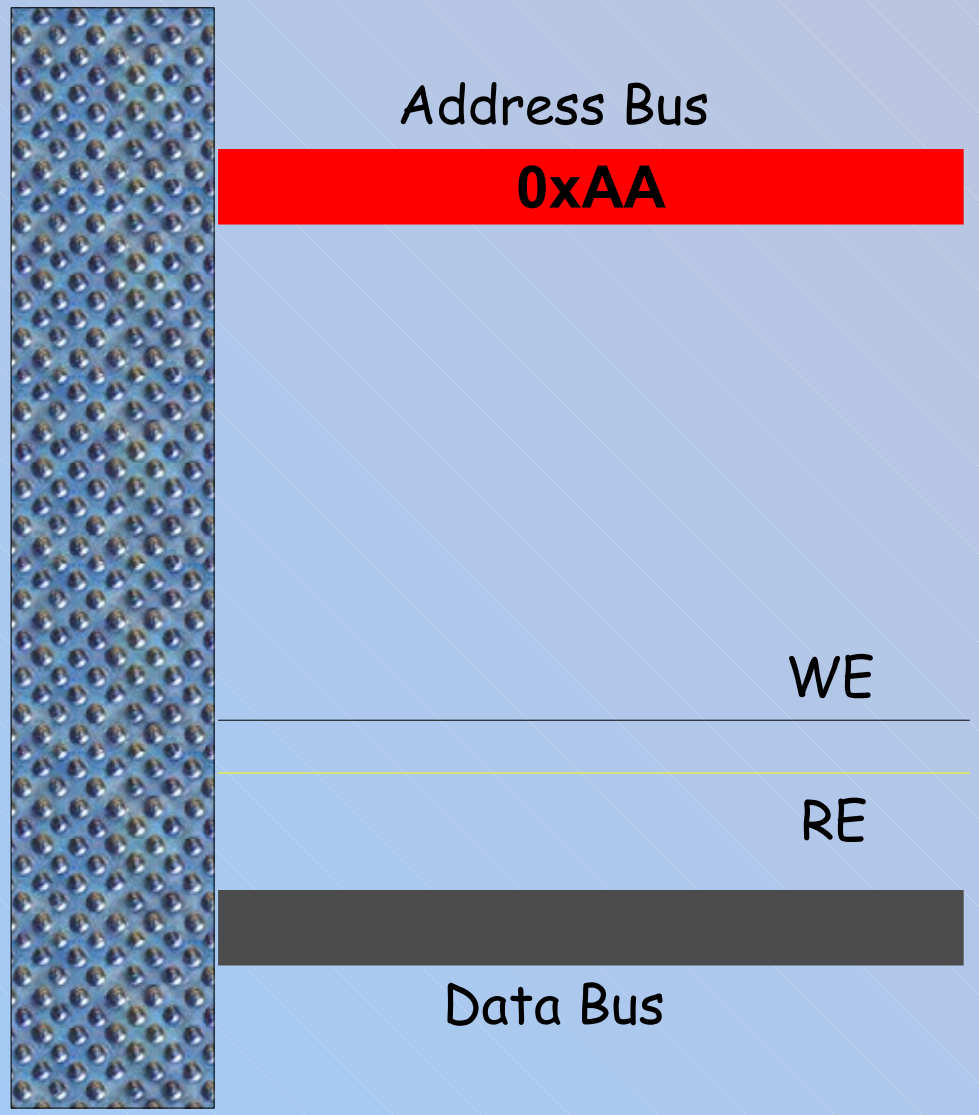
Διαδικασία εκτέλεσης απλών εντολών

```
char x, y; //Ορισμός μεταβλητών
           //στην μνήμη. Έστω το
           // x στην θέση: 0xA0 &
           // το y στην: 0xAA.

x = 5;
y = 3;

x = x + y;
```



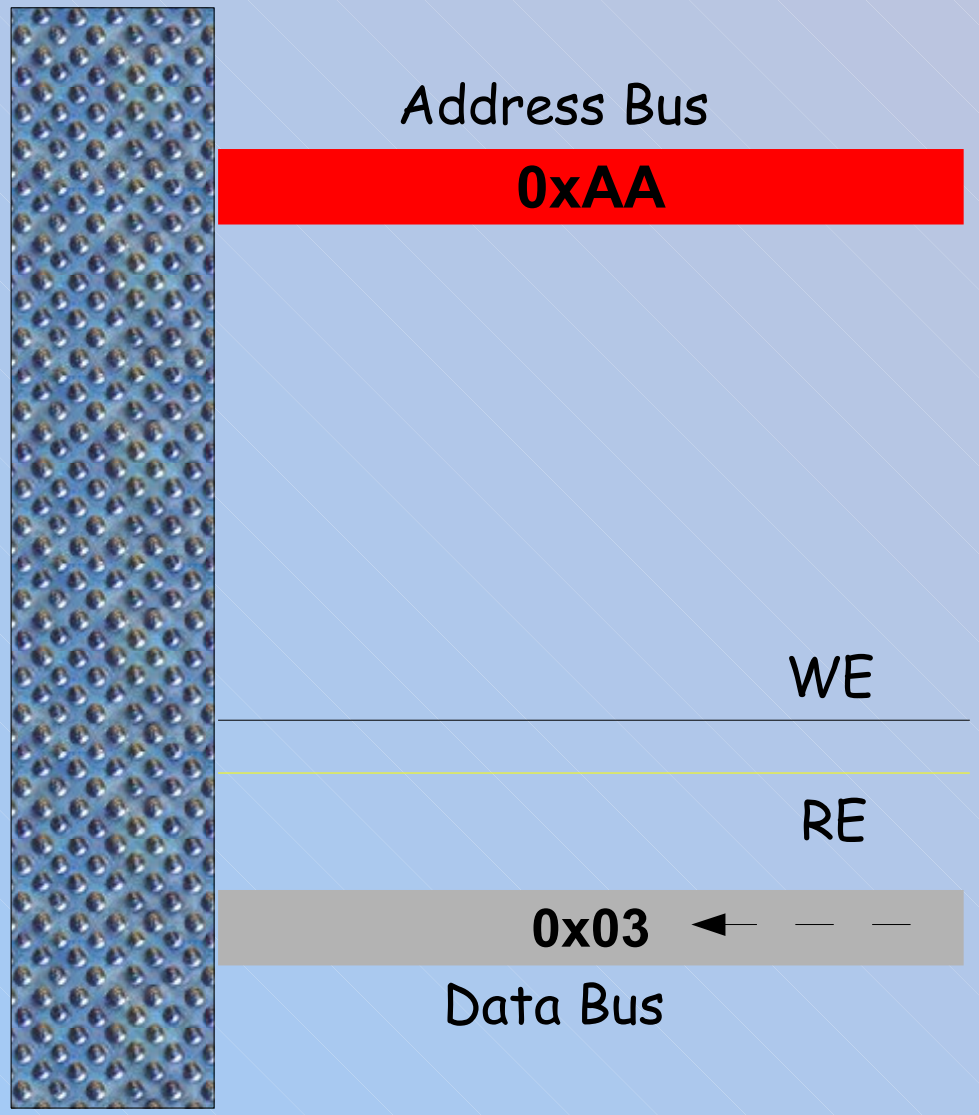
Διαδικασία εκτέλεσης απλών εντολών

```
char x, y; //Ορισμός μεταβλητών
           //στην μνήμη. Έστω το
           // x στην θέση: 0xA0 &
           // το y στην: 0xAA.

x = 5;
y = 3;

x = x + y;
```



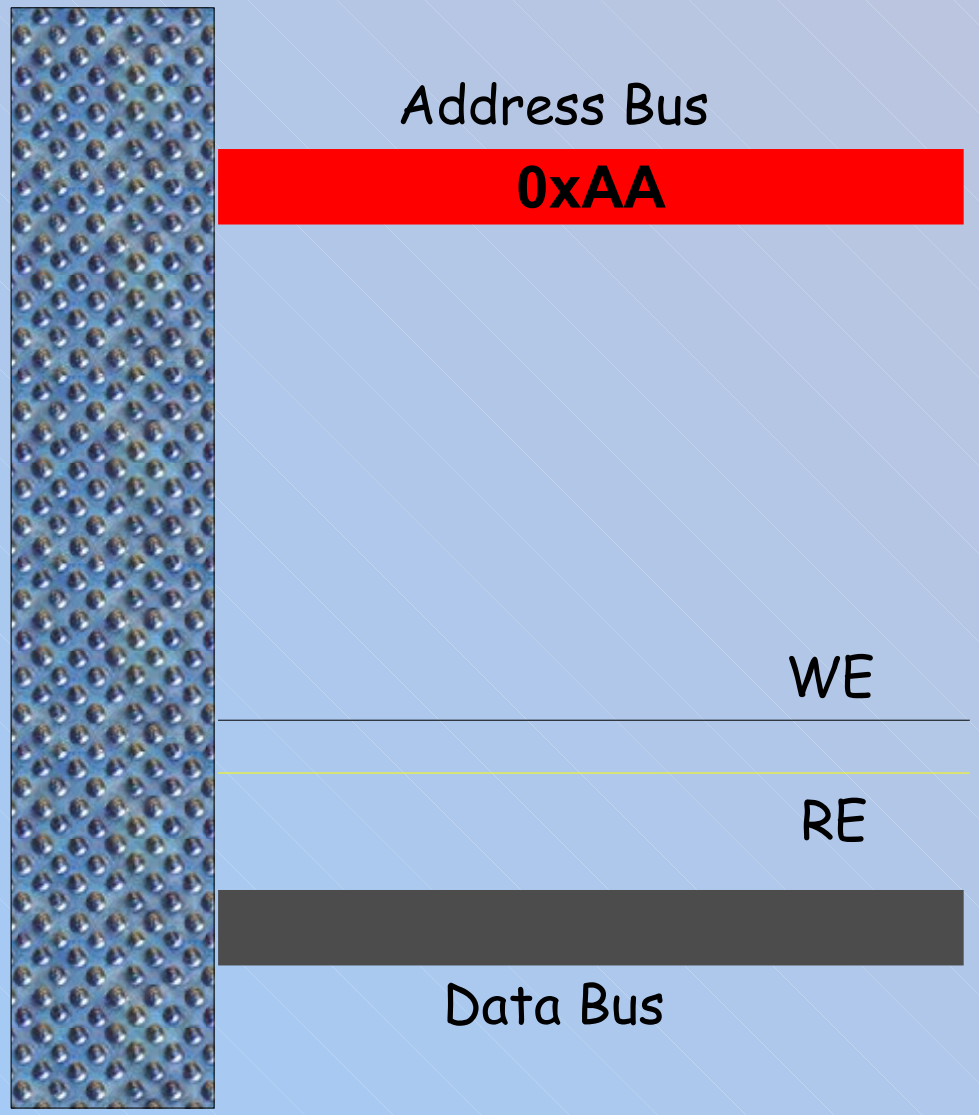
Διαδικασία εκτέλεσης απλών εντολών

```
char x, y; //Ορισμός μεταβλητών
           //στην μνήμη. Έστω το
           // x στην θέση: 0xA0 &
           // το y στην: 0xAA.

x = 5;
y = 3;

x = x + y;
```



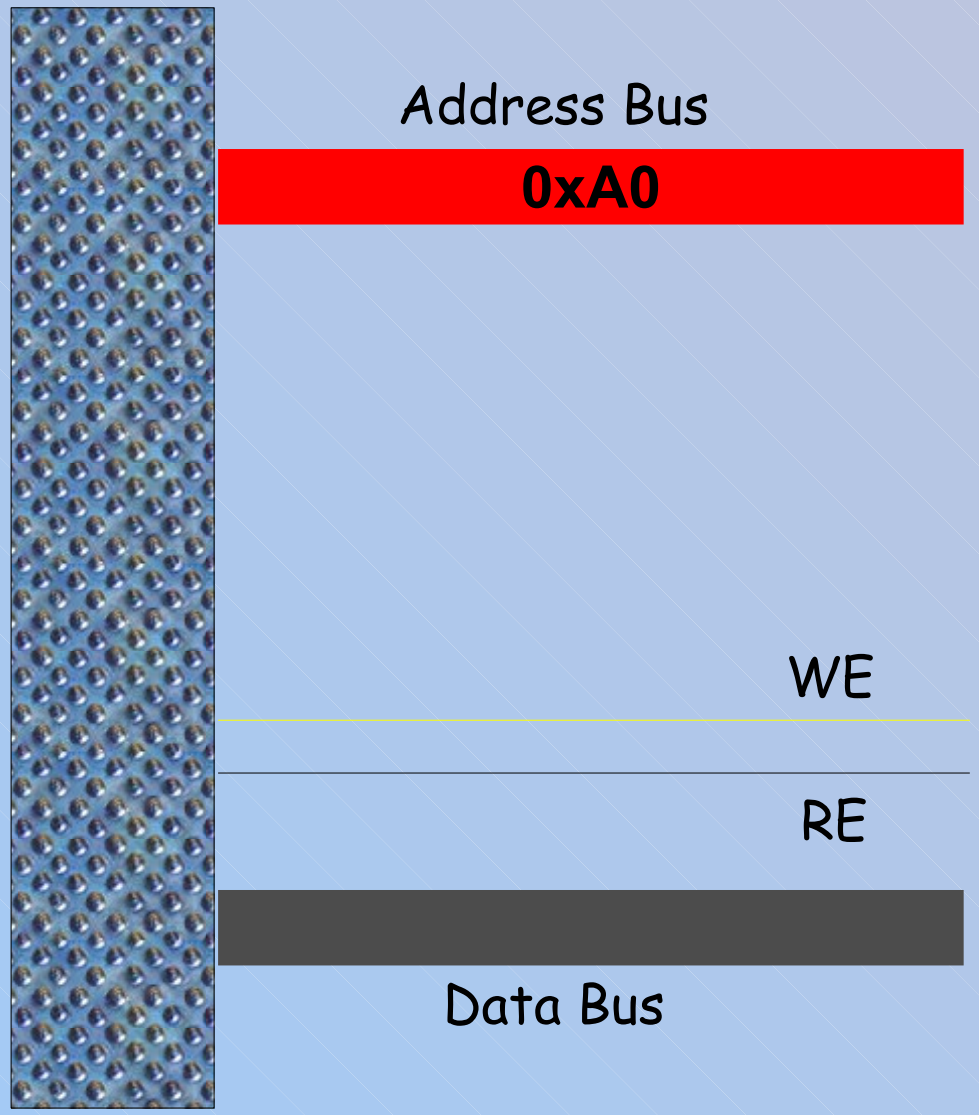
Διαδικασία εκτέλεσης απλών εντολών

```
char x, y; //Ορισμός μεταβλητών
           //στην μνήμη. Έστω το
           // x στην θέση: 0xA0 &
           // το y στην: 0xAA.

x = 5;
y = 3;

x = x + y;
```



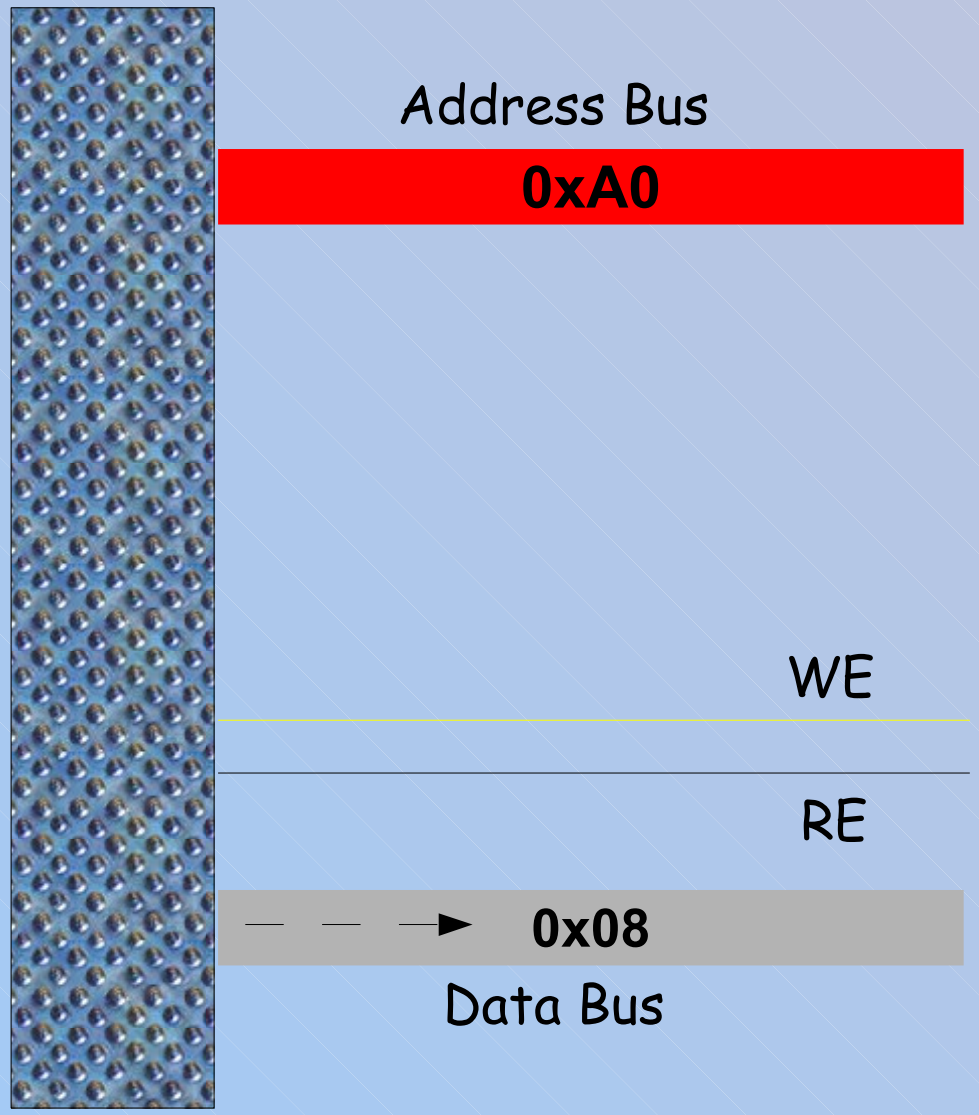
Διαδικασία εκτέλεσης απλών εντολών

```
char x, y; //Ορισμός μεταβλητών
           //στην μνήμη. Έστω το
           // x στην θέση: 0xA0 &
           // το y στην: 0xAA.

x = 5;
y = 3;

x = x + y;
```



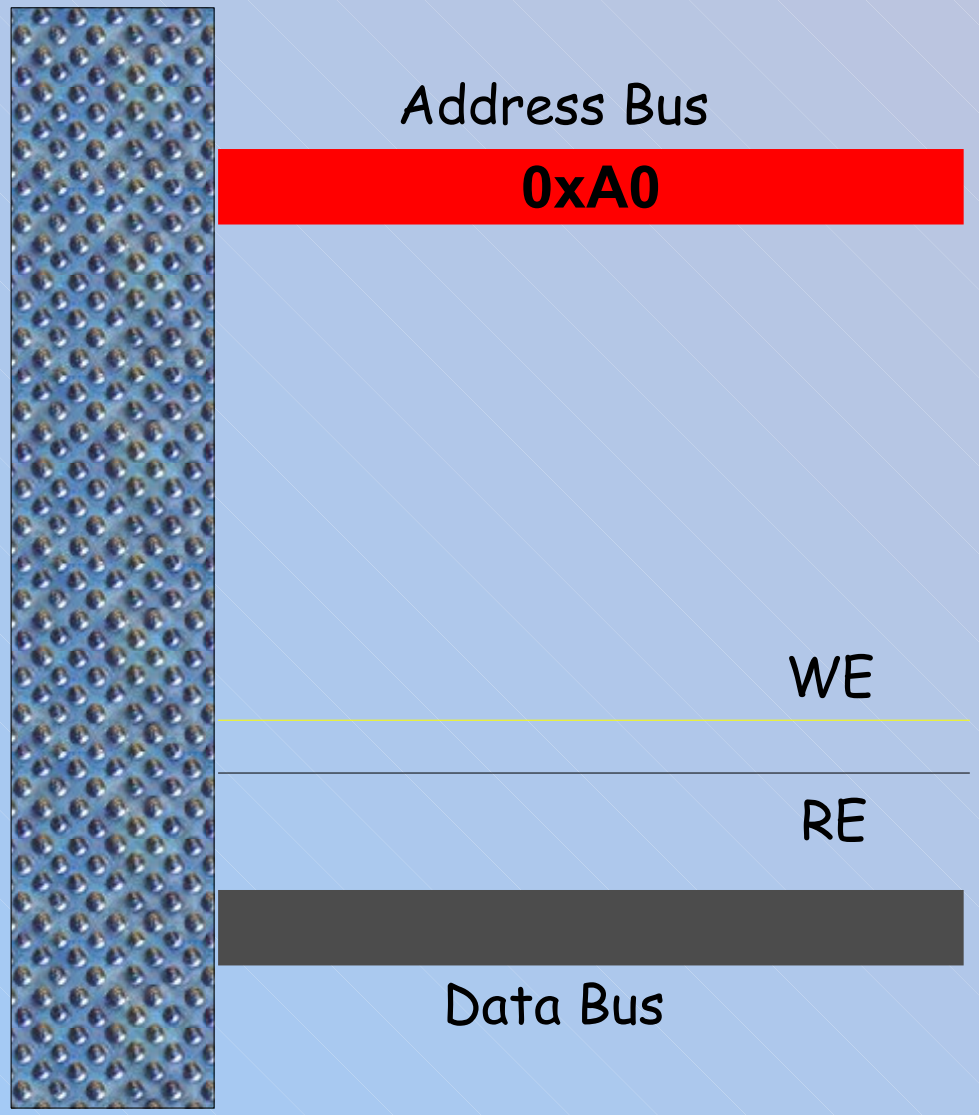
Διαδικασία εκτέλεσης απλών εντολών

```
char x, y; //Ορισμός μεταβλητών
           //στην μνήμη. Έστω το
           // x στην θέση: 0xA0 &
           // το y στην: 0xAA.

x = 5;
y = 3;

x = x + y;
```



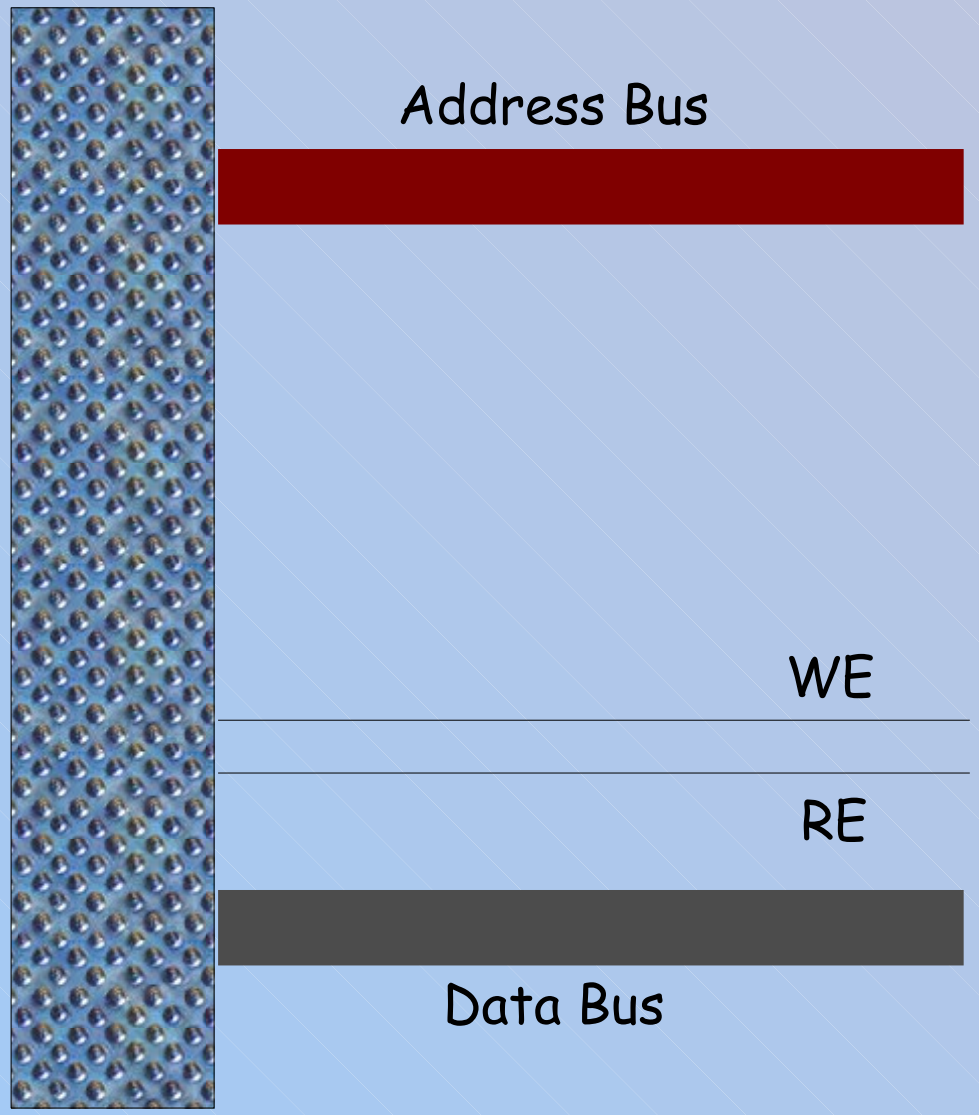
Διαδικασία εκτέλεσης απλών εντολών

```
char x, y; //Ορισμός μεταβλητών
           //στην μνήμη. Έστω το
           // x στην θέση: 0xA0 &
           // το y στην: 0xAA.

x = 5;
y = 3;

x = x + y;
```



Διαδικασία εκτέλεσης απλών εντολών

```
char x, y; //Ορισμός μεταβλητών
           //στην μνήμη. Έστω το
           // x στην θέση: 0xA0 &
           // το y στην: 0xAA.

x = 5;
y = 3;

x = x + y; // x = 8;
```

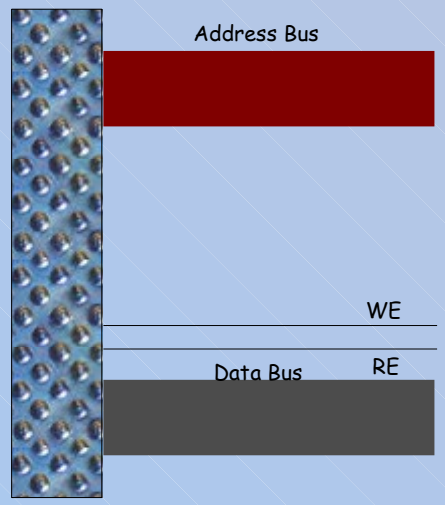


Διαδικασία εκτέλεσης απλών εντολών

- Στην C σε μικροϋπολογιστές, ο χρήστης πρέπει να δίνει ιδιαίτερη βαρύτητα, πέρα από τις τιμές των μεταβλητών του, και στις θέσεις μνήμης που αυτές χρησιμοποιούν.

- Έτσι μπορεί να εκμεταλλευτεί τα σήματα του address bus (π.χ. A0 - A7) ώστε με συγκεκριμένους συνδυασμούς τους να θέτει σε κατάλληλη λειτουργία τα διάφορα περιφερειακά συστήματα.

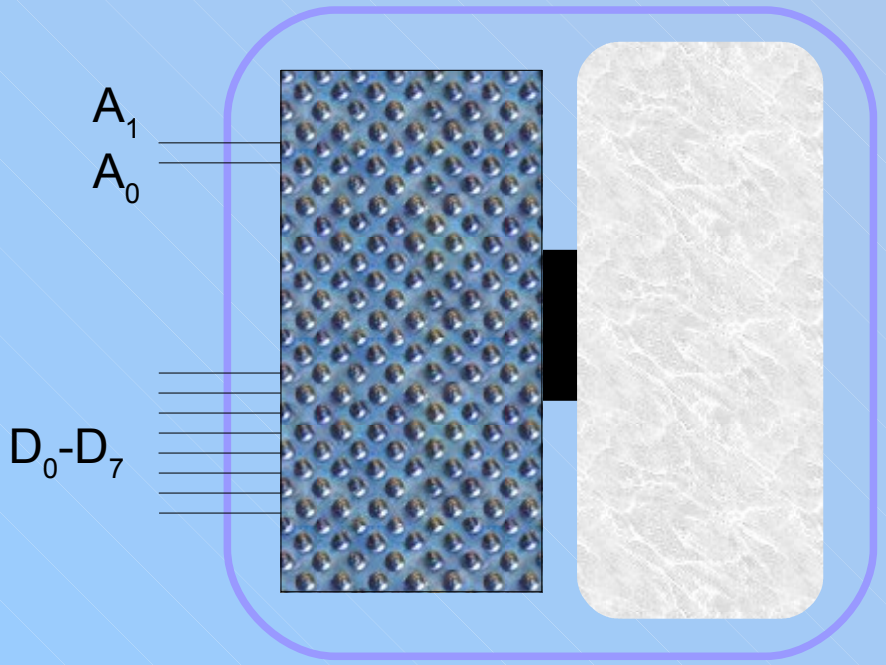
- Τους συνδυασμούς αυτούς μπορεί να τους δημιουργήσει με απλή αναφορά σε μια μεταβλητή που την έχει ορίσει στην αντίστοιχη συγκεκριμένη θέση μνήμης.

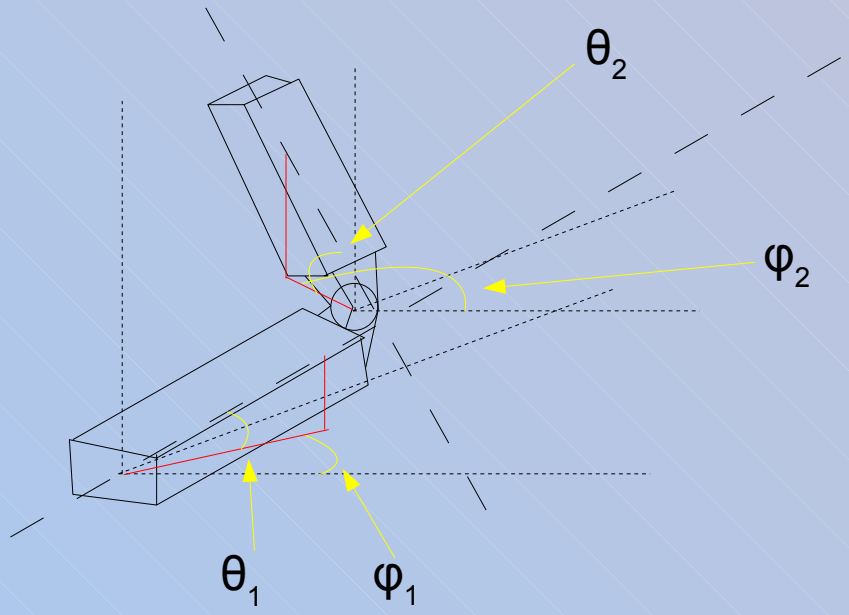


Π.χ. Ρομποτική

Οι 4εις δυνατοί συνδυασμοί A_0 και A_1 για τον μικροελεγκτή αντιστοιχούν στις τέσσερις διαφορετικές γωνίες περιστροφής του βραχίονα.

Στο Data bus δίνεται η γωνία περιστροφής.






```
char xdata F1 _at_ 0x0000;
char xdata Th1 _at_ 0x0001;
char xdata F2 _at_ 0x0002;
char xdata Th2 _at_ 0x0003;

F1 = 128; // 180 μοίρες.
F2 = 64; // 90 μοίρες.
Th1 = 0; // 0 μοίρες.
Th2 = 32; // 45 μοίρες.
```

Παράρτημα: ASCII code

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com