ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

ΑΝΟΙΚΤΑ ακαδημαϊκά μαθήματα ΠΠ

# ΑΡΧΕΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

## Κεφάλαιο 1


Επιμέλεια:
Βασίλης Παλιουράς , Αναπληρωτής Καθηγητής
Ευάγγελος Δερματάς , Αναπληρωτής Καθηγητής
Σταύρος Νούσιας , Βοηθός Ερευνητή
Πολυτεχνική Σχολή
Τμήμα Ηλεκτρολόγων Μηχανικών & Τεχνολογίας Υπολογιστών

# Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τυπου άδειας χρησης, η άδεια χρησης αναφερεται ρητώς.





# Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου των διδασκόντων καθηγητών.
- Το έργο «Ανοικτά Ακαδήμαϊκά Μαθήματα στο Πανεπιστήμιο Πατρών» έχει χρηματοδοτηθεί μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκη Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικους πόρους.

# Ανάπτυξη

Το παρόν εκπαιδευτικό υλικό αναπτύχθηκε στο τμήμα Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών του Πανεπιστημίου Πατρών

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ & ΘΡΗΣΚΕΥΜΑΤΩΝ, ΠΟΛΙΤΙΣΜΟΥ & ΑΘΛΗΤΙΣΜΟΥ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

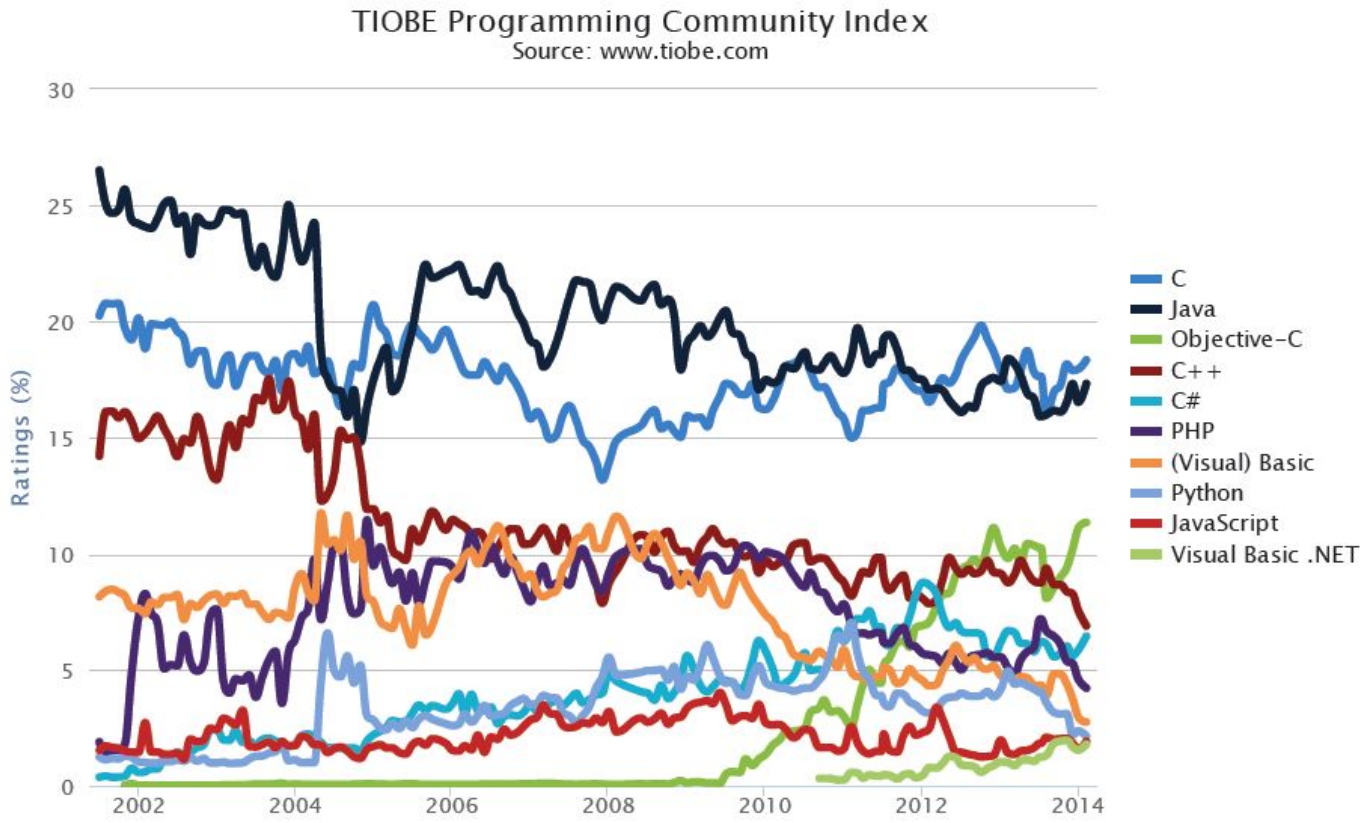
# Αντικείμενο του μαθήματος

- Τεχνικές για την ανάπτυξη προγραμμάτων
- «Σωστά προγράμματα», «καλύτερα προγράμματα», «ποιότητα κώδικα»,…
- Γλώσσα C
  - C11 ISO/IEC 9899:2011
  - C99 ISO/IEC 9899:1999
  - C90 ISO/IEC 9899:1990

# Η αρχική φιλοσοφία της C (όχι τυπικά)

- Ο προγραμματιστής ξέρει τι κάνει => εμπιστοσύνη
- Υποστηρίζει επίπεδα αφαίρεσης αλλά όχι τόσο ώστε να κρύψει το υλικό που εκτελεί τα προγράμματα.
  - Ο προγραμματιστής έχει πρόσβαση στο υλικό όταν είναι αναγκαίο.
  - Υποστηρίζεται η ανάπτυξη κώδικα ευκολονόητου και απλού στη συντήρηση.
  - Παρέχει ταχύτητα χωρίς πάρα πολύ προσπάθεια.
  - Για προγραμματισμό σε χαμηλό επίπεδο μπορείς να γράψεις assembly συμβατή με C, αλλά δεν το χρειάζεσαι πολύ συχνά.
  - Για κατασκευή compilers (μεταγλωττιστών), assembly υψηλής ποιότητας μπορεί να προκύψει από καλογραμμένο κώδικα C.
  - Για τον προγραμματιστή, η κατανόηση της οργάνωσης του υπολογιστή παραμένει σημαντική.

# Χρήση γλωσσών προγραμματισμού




# Χρήση γλωσσών προγραμματισμού

| Programming Language | 2014 | 2009 | 2004 | 1999 | 1994 | 1989 |
|---|---|---|---|---|---|---|
| C | 1 | 2 | 2 | 1 | 1 | 1 |
| Java | 2 | 1 | 1 | 13 | - | - |
| Objective-C | 3 | 38 | 48 | - | - | - |
| C++ | 4 | 3 | 3 | 2 | 2 | 3 |
| C# | 5 | 8 | 9 | 30 | - | - |
| PHP | 6 | 5 | 6 | - | - | - |
| (Visual) Basic | 7 | 4 | 5 | 3 | 3 | 7 |
| Python | 8 | 6 | 11 | 27 | 22 | - |
| JavaScript | 9 | 9 | 8 | 20 | - | - |
| Perl | 10 | 7 | 4 | 5 | 17 | 23 |
| Lisp | 14 | 17 | 15 | 12 | 7 | 2 |

# Γιατί C;

- Υποστηρίζεται από πάρα πολλές πλατφόρμες .
- Είναι φορητή
- Στενή σχέση με το unix
- Κυκλοφορεί αρκετό καιρό
- Η μόνη γλώσσα που υποθ έτεις με ασφάλεια ότι υποστηρίζ εται σε κάθε Unix και παραλλαγ ές (linux)
- Είναι απλή.
- Είναι ευέλικτη.
- Είναι γρήγορη.
- Είναι δωρεάν
- Μεγάλη εγκαταστημένη βάση.
- Δημοφιλής για δεκαετίες
- Ώριμη: Δεν αναμένονται μεγάλες αλλαγές

# Ένα πρόγραμμα σε C

```
//Πάντα υπάρχει μία και μόνο συνάρτηση main

#include <stdio.h>

main ( ) {        //  Ο κώδικας οργανώνεται με αγκιστρα

    printf ("Hello world!\n");

}           //  Γράφουμε κώδικα μόνο μέσα σε συναρτήσεις
```

# Ένα πρόγραμμα σε C

```
#include <stdio.h>

int main() {

    int i, j;
    int sum ;
    i = 5 ;
    j = 6 ;
    sum = i + j ;
    printf ("sum: %d\n", sum);
    return 0;
}

// Δηλώνουμε τα ονόματα πριν τα χρησιμοποιήσουμε !!!
```

# Διαφορετικές γλώσσες – διαφορετική συμπεριφορά
# C versus Python

```
#include <stdio.h>
int main(int argc, char *argv[]) {
 int i = 1<<30;
 printf("%d\n", i);
 system("pause");
}
```

Σημαίνει $2^{30}$

```
>>> a = 1 <<30
>>> a
        1073741824
```

```
#include <stdio.h>
int main(int argc, char *argv[]) {
 int i = 1<<30;
 printf("%d\n", i);
 system("pause");
}
```

```
>>> a = 1<<31
>>> a
        2147483648
```

# Ένα πείραμα σε Python

```
global a
a = 1<<30
def test():
        for i in range(0, 50):
                b = a + 1
if __name__ == '__main__':
        from timeit import Timer
        t = Timer('test()','from __main__ import test')
        print t.timeit(1000000)
```

**Χρόνος:** 10.7764198383 sec

```
global a
a = 1<<31
def test():
        for i in range(0, 50):
                b = a + 1
if __name__ == '__main__':
        from timeit import Timer
        t = Timer('test()','from __main__ import test')
        print t.timeit(1000000)
```

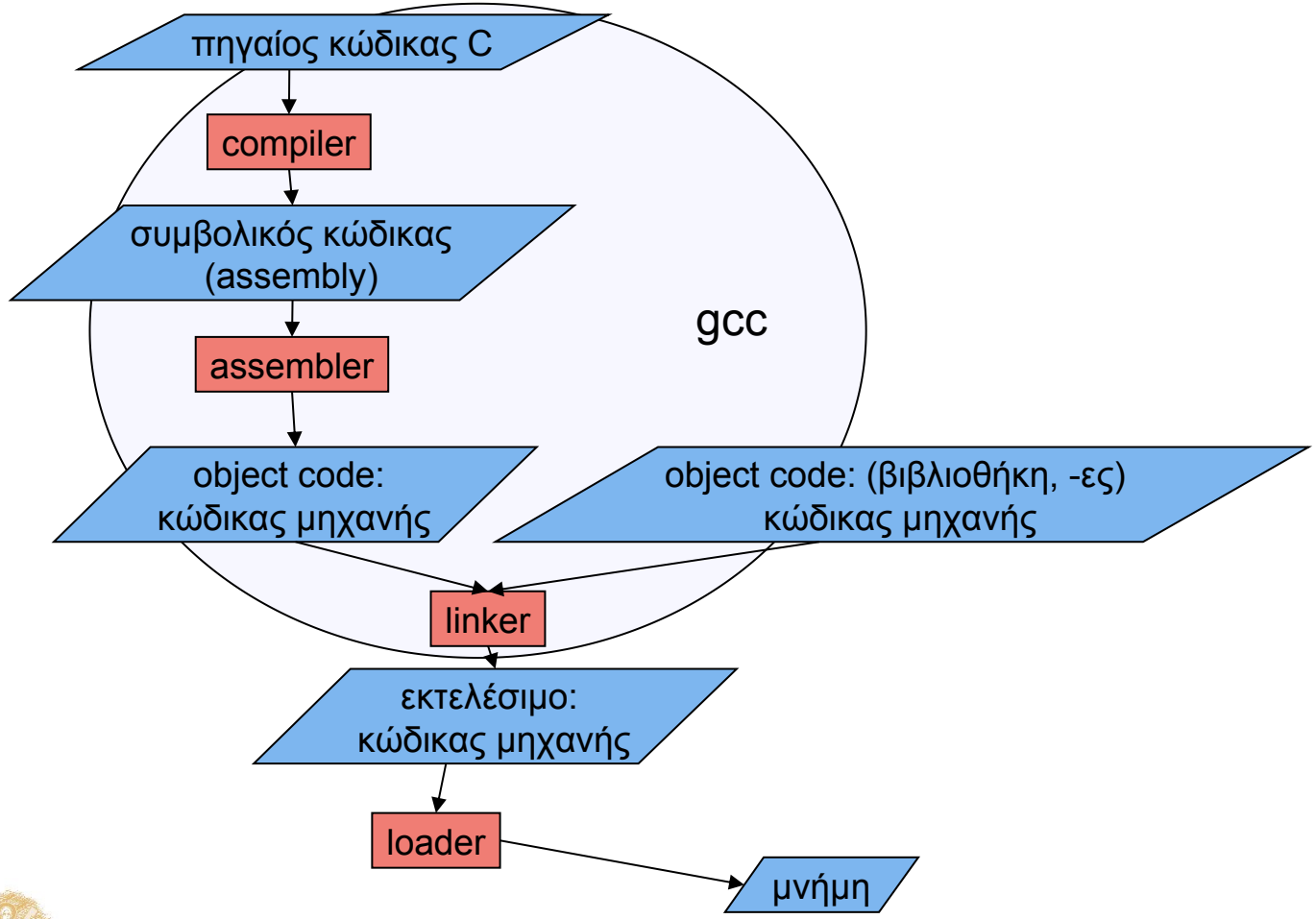
**Χρόνος:** 21.0824803152 sec

# Εκτέλεση σε C

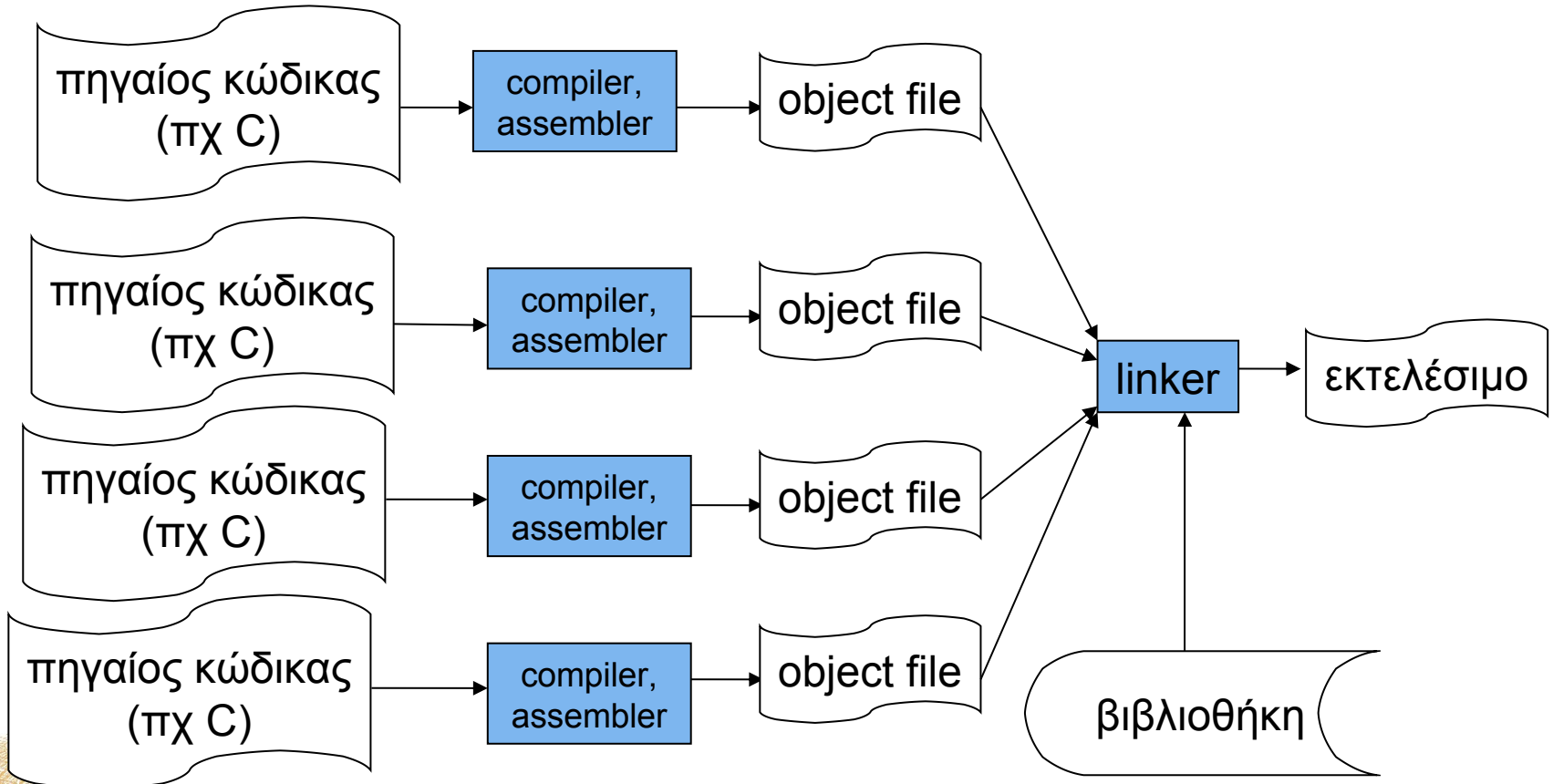
```
#include <stdio.h>
#include <time.h>
int a = 1<<31;

int test() {
 int j;
 int b ;
 for (j=0;j<=50;j++)
        b = a + 1;
}

int main(int argc, char *argv[]) {
 int j;
 clock_t t1, t2;
 float ratio ;
 ratio = 1./CLOCKS_PER_SEC;
 t1 = clock();
 for (j=0;j<1000000;j++)
     test();
 t2 = clock();
printf("Time = %f\n", ratio*(long)t1 + ratio*(long)t2 );
}
```

**Χρόνος:** 0.46 sec

# Βήματα μετάφρασης κώδικα C




# Από γλώσσα υψηλού επιπέδου σε εκτελέσιμο



# Η γλώσσα C ως αφαίρεση

- Οι γλώσσες υψηλού επιπέδου είναι μια μορφή αφαίρεσης.
- Η πολυπλοκότητα του κώδικα μηχανής αποκρύπτεται από τον προγραμματιστή.
- Ο μεταγλωττιστής (compiler) αντιστοιχίζει στον κώδικα C σύνολα εντολών κώδικα μηχανής

# Βρόχος for σε C

```
#include <stdio.h>

main () {

    register int i, sum = 0;

    for ( i = 0; i < 10; i++)
        sum =  sum + i;

    printf ("\ti: %d\n", sum);

}
```

# Το ίδιο πρόγραμμα σε assembly (για επεξεργαστή Pentium)

```
_main:
        pushl %ebp
        movl %esp,%ebp
        subl $16,%esp
        pushl %esi
        pushl %ebx
        call ___main
        xorl %esi,%esi
        xorl %ebx,%ebx
L11:
        cmpl $9,%ebx
        jle L14
        jmp L12
L14:
        addl %ebx,%esi
L13:
        incl %ebx
        jmp L11
```

```
L12:
        addl $-8,%esp
        pushl %esi
        pushl $LC0
        call _printf
        addl $16,%esp
L10:
        leal -24(%ebp),%esp
        popl %ebx
        popl %esi
        movl %ebp,%esp
        popl %ebp
        ret
```

# Αντιστοιχίζοντας κώδικα assembly σε κώδικα μηχανής Παράδειγμα η εντολή ADD

| | | |
|---|---|---|
| 04 ib | ADD AL, imm8 | Add imm8 to AL |
| 05 iw | ADD AX, imm16 | Add imm16 to AX |
| 05 id | ADD EAX, imm32 | Add imm32 to EAX |
| 80 /0 ib | ADD r/m8,imm8 | Add imm8 to r/m8 |
| 81 /0 iw | ADD r/m16,imm16 | Add imm16 to r/m16 |
| 81 /0 id | ADD r/m32,imm32 | Add imm32 to r/m32 |
| 83 /0 ib | ADD r/m16,imm8 | Add sign-extended imm8 to r/m16 |
| 83 /0 ib | ADD r/m32,imm8 | Add sign-extended imm8 to r/m32 |
| 00 / r | ADD r/m8,r8 | Add r8 to r/m8 |
| 01 / r | ADD r/m16,r16 | Add r16 to r/m16 |
| 01 / r | ADD r/m32,r32 | Add r32 to r/m32 |
| 02 / r | ADD r8,r/m8 | Add r/m8 to r8 |
| 03 / r | ADD r16,r/m16 | Add r/m16 to r16 |
| 03 / r | ADD r32,r/m32 | Add r/m32 to r32 |

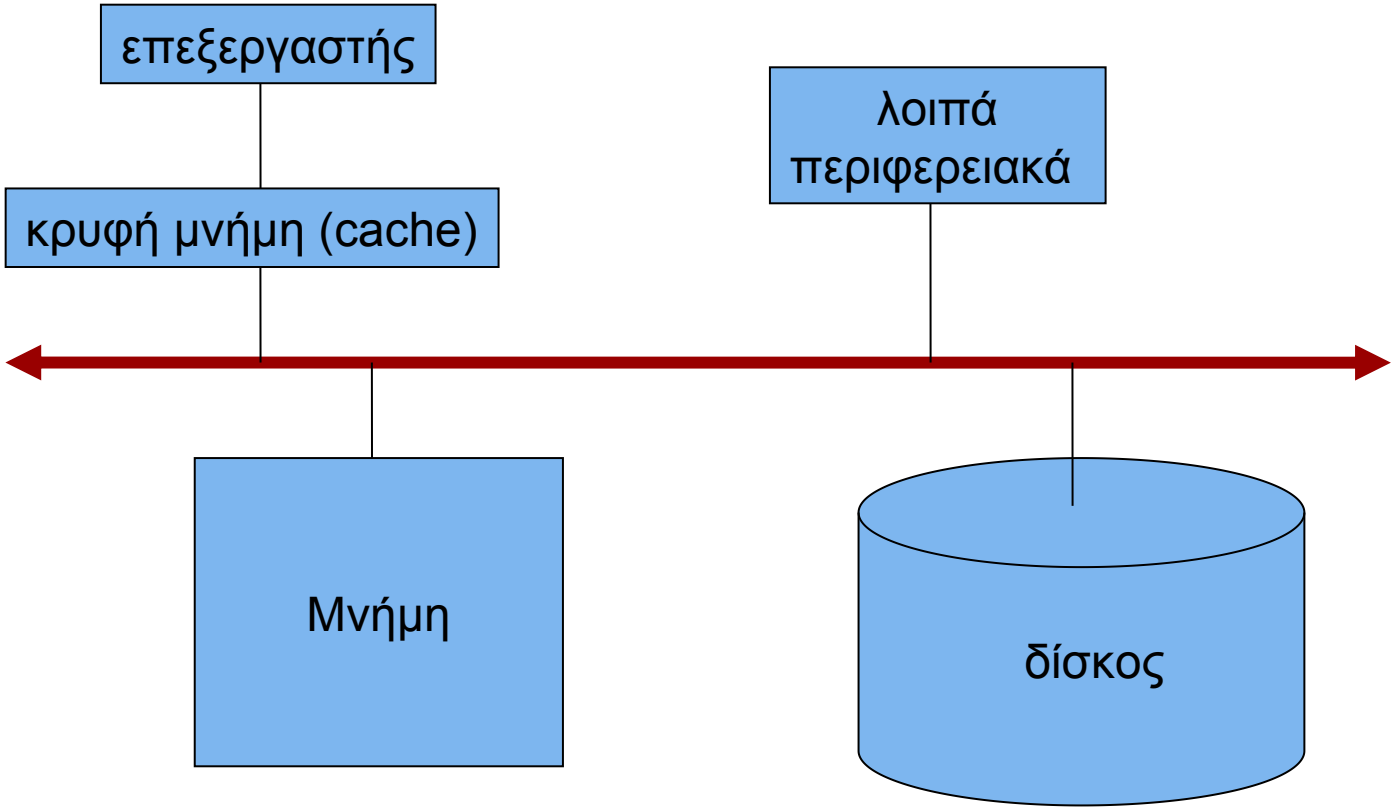
# ...κώδικας μηχανής (ενδεικτικά)

1011001110010101 0
0111010101010111 0
0101010101010101 1
1101010111011100 1
0001010101110111 0
1111110101010101 0

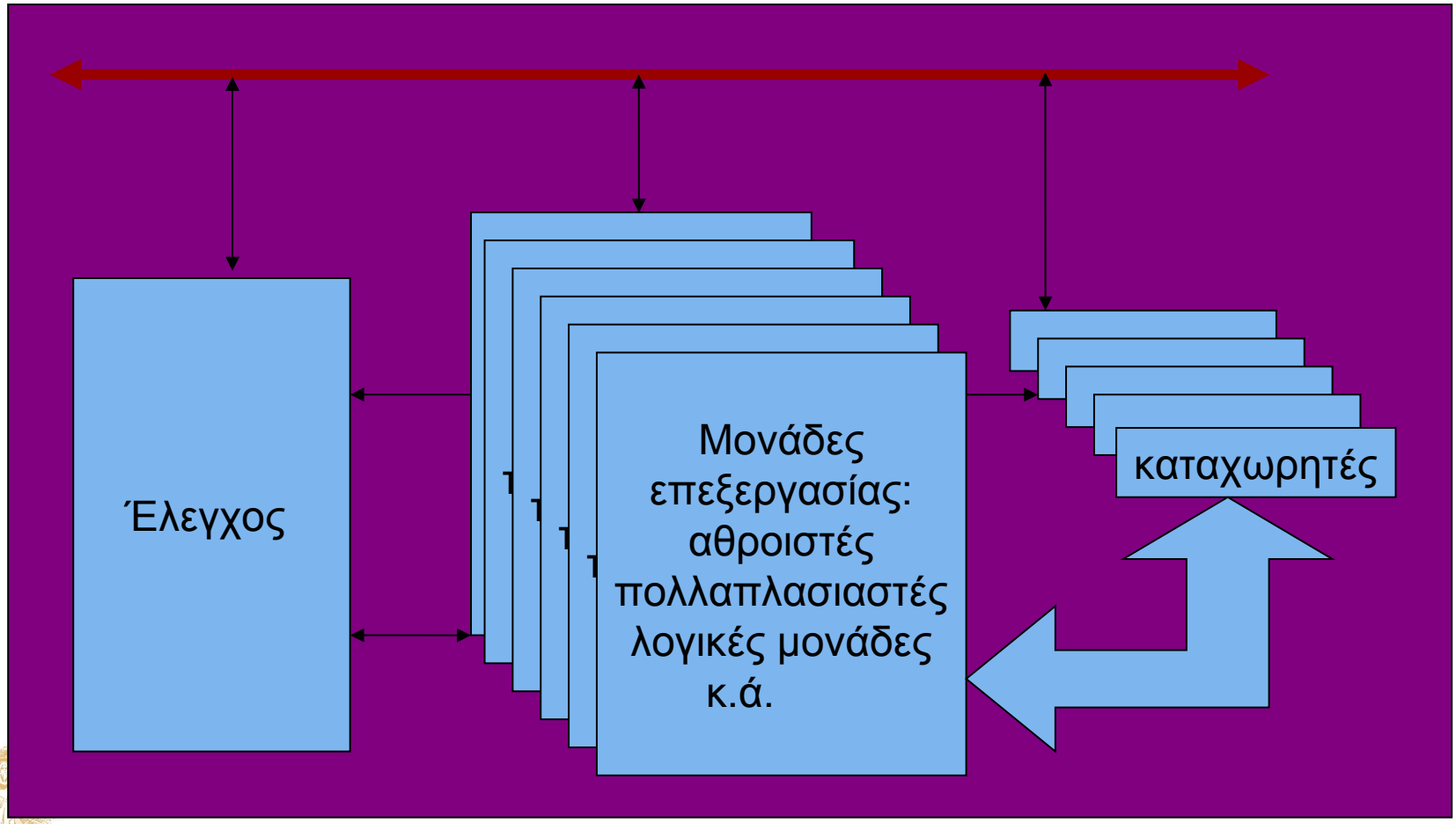
# Χαρακτηριστικά γλωσσών assembly

- Το πρόγραμμα εξαρτάται από το είδος του επεξεργαστή στον οποίο εκτελείται
- το προγράμματα είναι μεγαλύτερα
  - τι σημαίνει μεγαλύτερα; μονάδα μέτρησης οι «Γραμμές κώδικα»
  - περισσότερος χρόνος για να γραφούν
  - ευαίσθητα σε λάθη, δυσνόητα
- έλλειψη δομής (structure)
  - ακόμα και οι απλούστεροι βρόχοι, υπό συνθήκη διακλαδώσεις κτλ δομούνται με jumps
  - ιστορική εξέλιξη => δομημένος προγραμματισμός (structured programming)
- Πλεονεκτήματα
  - ταχύτητα, μικρό μέγεθος κώδικα σε bytes...

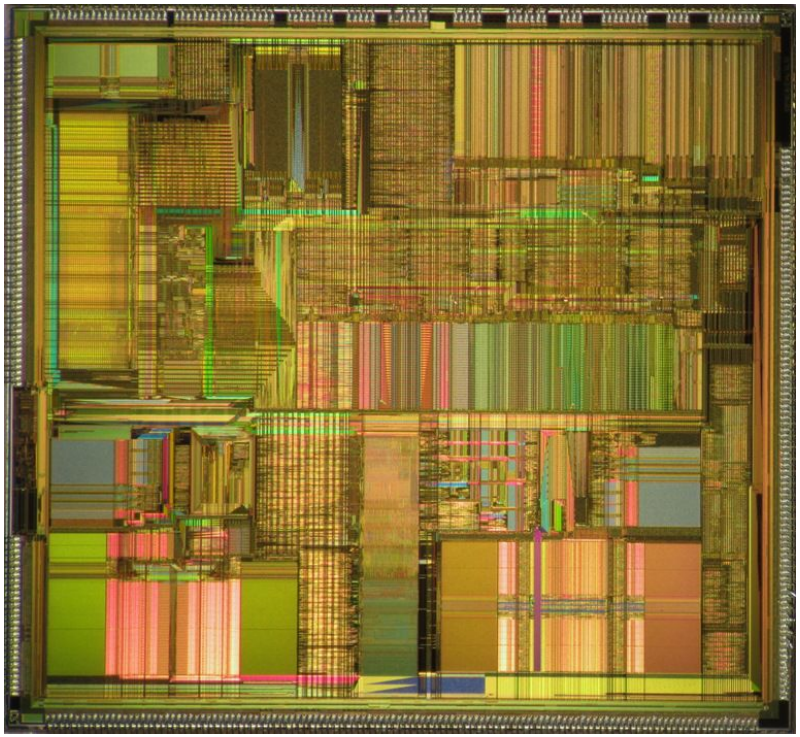
# Στοιχειώδης οργάνωση υπολογιστή (από τα χαμηλά στα ψηλά => bottom up)




# Στοιχειώδης Οργάνωση Επεξεργαστή

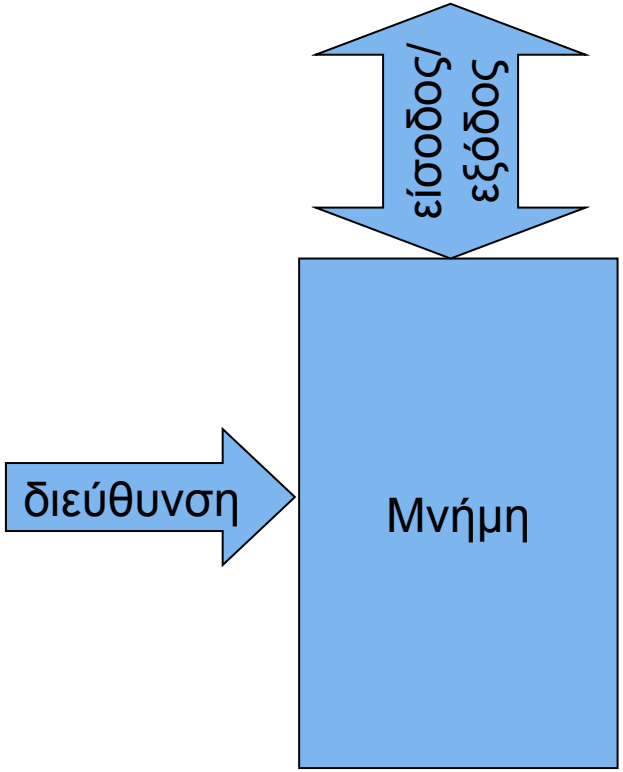


# dual core μPs



Πηγή:http://en.wikipedia.org/wiki/P5_(microarchitecture)

# Η μνήμη

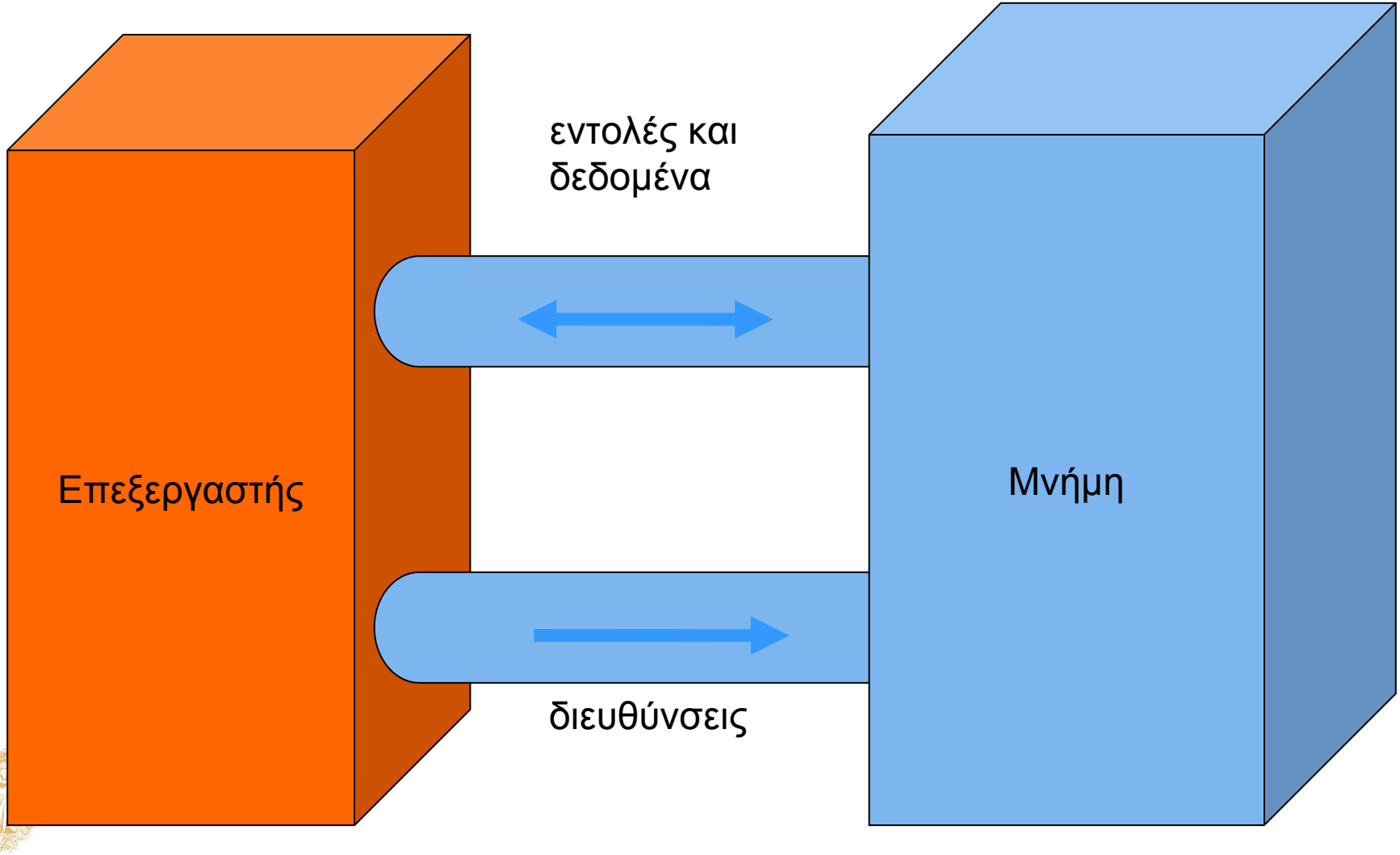



| Διεύθυνση (address) | Περιεχόμενα |
|---|---|
| 0x0000 | 00111111 |
| 0x0001 | 10110101 |
| 0x0002 | 11011110 |
| 0x0003 | 10111100 |
| … | |
| 0x00FF | 00111101 |
| 0x0100 | 10101110 |
| … | |

Πόσα bytes διαθέτονται για κάθε μεταβλητή;

Περιεχόμενα: Δεδομένα και εντολές

# Το προγραμματιστικό μοντέλο



# Περιβάλλοντα προγραμματισμού

- dev-c++ (http://www.bloodshed.net)
- Orwell Dev-C++ (http://orwelldevcpp.blogspot.gr/)
- gcc σε Cygwin ή linux
- Eclipse IDE for C/C++ developers

# Το αντικείμενο του μαθήματος

- Πώς ξεκινώντας από την περιγραφή ενός προβλήματος θα καταλήξουμε σε καλό πρό γραμμα;
  - Problem solving
- Πώς θα διαχειριστούμε την πολυπλοκότητα;
- Βασικές προγραμματιστικές αρχές
  - Μαθαίνουμε να τις εφαρμόζουμε στην πράξη!
  - Χρησιμοποιούμε κυρίως τη γλώσσα C
- Τεχνικές
  - Αφαιρετικότητα (abstraction)
  - Έλεγχος ορθής λειτουργίας και διασφάλιση ποιότητας
    - έλεγχος τύπων
    - πώς διασφαλίζουμε ότι το πρόγραμμα λειτουργεί σωστά;
    - τι θα πει «σωστή» λύση;
  - Αξιοποίηση διαθέσιμων δομικών στοιχείων (reusability)
  - Δυνατότητες επέκτασης – τροποποίησης του κώδικα
  - Οι τεχνικές δεν αφορούν μόνο το λογισμικό!
    - Εφαρμόζονται στο σύνολο των συστημάτων που σχεδιάζει ο μηχανικός.
- Πολλές στρατηγικές προγραμματισμού
  - Διαδικαστικός (procedural) προγραμματισμός
  - Αντικειμενοστρεφής (object-oriented) προγραμματισμός
    - συναρτησιακός (functional),
    - λογικός (logic),....

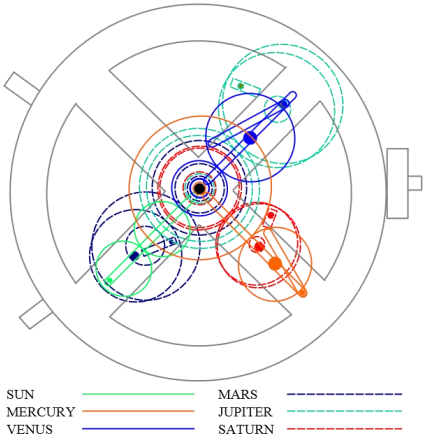
# Ιστορικά στοιχεία

Πηγή:wikipedia.org

Πηγή:wikipedia.org

~2000πΧ

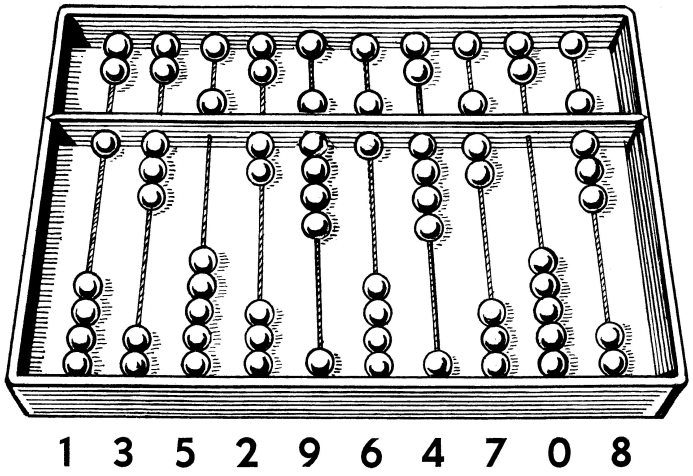
Πηγή:wikipedia.org




Πηγή:http://commons.wikimedia.org/

100 – 150 πΧ

# Ιστορικά στοιχεία




άβακας 300 πΧ – 1200 μΧ

Πηγή:wikipedia.org

# Ιστορικά στοιχεία

Blaise Pascal

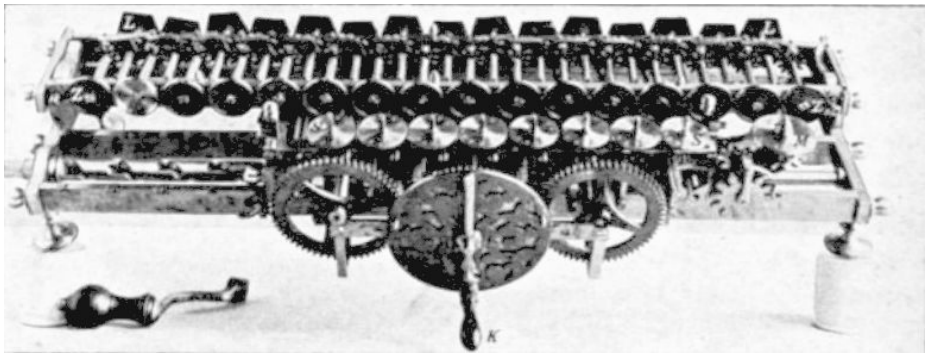
Πηγή:wikipedia.org

Πηγή:wikipedia.org

The Pascaline Automatic Calculator 1642

# Ιστορικά στοιχεία

Πηγή:wikipedia.org



Πηγή:wikipedia.org

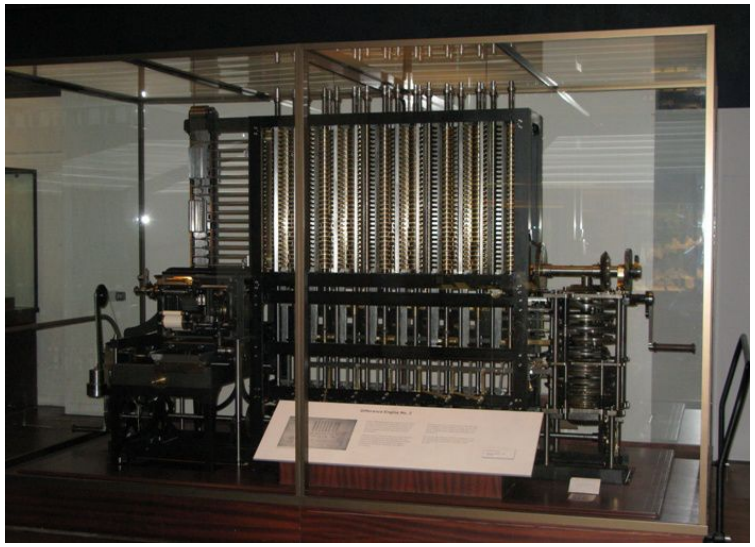
1673 - 1694: The Leibniz Calculator

# Ιστορικά στοιχεία

Πηγή:wikipedia.org

Αριθμόμετρο - Charles Xavior Thomas - 1820

# Ιστορικά στοιχεία



Πηγή:wikipedia.org

The difference engine

*Ada Augusta King*

Πηγή:wikipedia.org

*Charles Babbage*

Πηγή:wikipedia.org

# Ιστορικά στοιχεία

Πηγή:wikipedia.org

ENIAC

Πηγή:wikipedia.org

John von Neumann

# Το πρώτο bug


Πηγή:wikipedia.org

«Γιατί κόλλησε;» (υπολογιστής Mark I, 1945)

Πηγή:wikipedia.org

*Grace Murray Hopper*

# Σύγχρονοι υπολογιστές

Πηγή:wikipedia.org

IBM_Blue_Gene_P_supercomputer


Πηγή:wikipedia.org

Intel Core i7 2600K

DreamWorks Animation studio is powered by HP workstations and Redhat Linux distributions (image credit Linux Journal).

# Γλώσσα προγραμματισμού

- Γλώσσα προγραμματισμού
- «...είναι μια συστηματική σημειογραφία με την οποία περιγράφουμε υπολογιστικές διεργασίες...»
- Οι αρχές προγραμματισμού είναι ανεξάρτητες της γλώσσας
- μια σύγχρονη γλώσσα θα πρέπει να διευκολύνει την εφαρμογή τους.

# Γλώσσα προγραμματισμού

- Διαδικαστικός προγραμματισμός
- Εφαρμογές με γλώσσα C
- Η C++ ως καλύτερη C
- Αντικειμενοστρεφής προγραμματισμός

# Βιβλιογραφικές πηγές


- C How to program (Introducing C++ and Java), Deitel & Deitel, Prentice Hall.
- Programming Language Pragmatics, M. Scott, Morgan – Kaufmann.
- The C Programming Language, B. Kernighan και D. Ritchie, Prentice Hall.
- The Practice of Programming, B. Kernighan και R. Pike, Addison – Wesley.
- Από τη C στη Java, Τόμοι Α και Β, Κ. Θραμπουλίδης
- Structure and Interpretation of Computer Programs, Abelson, Sussman, and Sussman, MIT Press (διαθέσιμο στο internet: http://mitpress.mit.edu/sicp/ )
- Memory as a programming concept in C and C++, Frantisec Franek, Cambridge University Press.

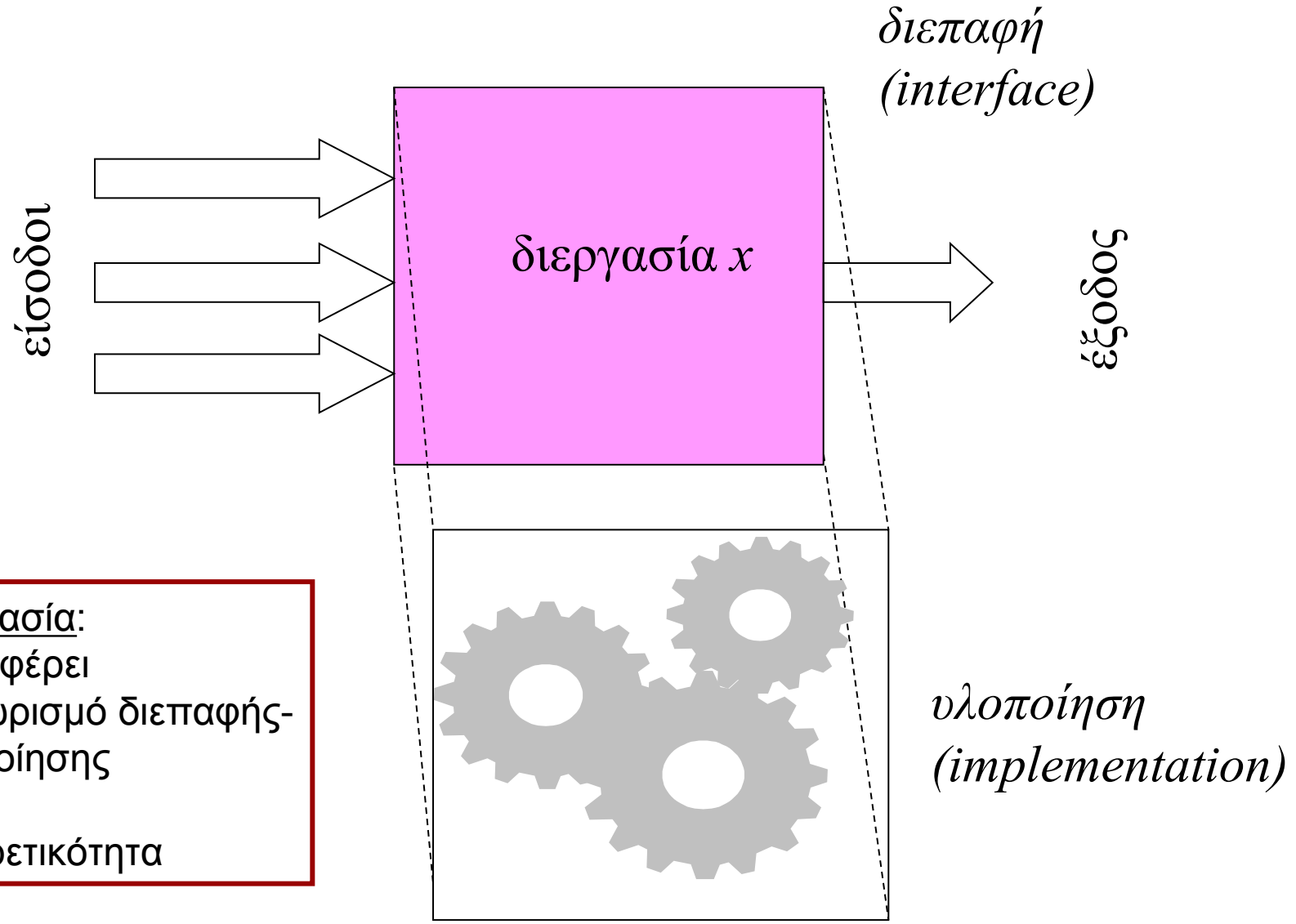

# Καθημερινότητα...

- Προδιαγραφές και εξομοιώσεις
  σε C/C++, matlab, SystemC,...
- Περιγραφή υλικού σε VHDL, verilog,...
  - γράφουμε απευθείας μοντέλα ή μέσω scripts (perl,...) αλλά και matlab ή C, mathematica
  - παράγουμε οδηγίες προς τον εξομοιωτή με script
- Επεξεργασία μοντέλων
  με EDA CAD,
  - οδηγίες με γλώσσα
    χαρακτηριστική του εργαλείου
- κτλ.

# Διαδικαστικός προγραμματισμός: Η Διεργασία

- Από ψηλά στα χαμηλά
  - αντιμετώπιση top down
- Τμήμα μιας ευρύτερης λύσης, με σαφώς καθορισμένη σχέση εισόδων - εξόδων
  - όταν είναι δυνατόν να εκτελεστεί από υπολογιστή $\Rightarrow$ *υπολογιστική διεργασία*
- Βασικό εργαλείο στο χειρισμό της πολυπλοκότητας $\Rightarrow$ αφαιρετικότητα

# Διεργασία




Διεργασία:
προσφέρει
διαχωρισμό διεπαφής-
υλοποίησης

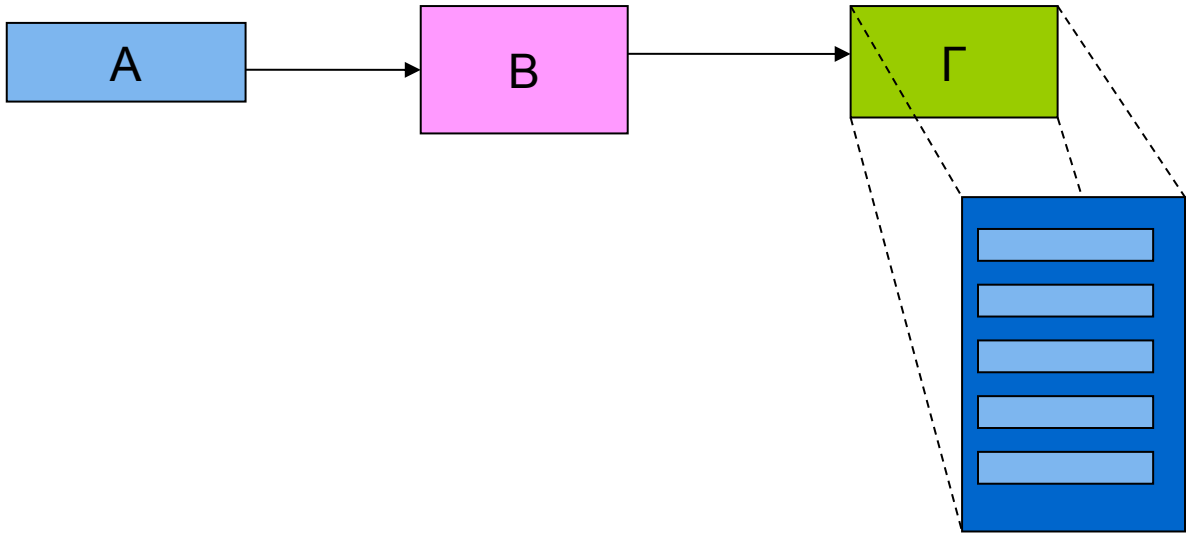
⇒

αφαιρετικότητα

# Συναρτήσεις και Διεργασίες

- Στη γλώσσα C
- Συναρτήσεις ↔ υπολογιστικές διεργασίες
- χρήση της printf στο παράδειγμα

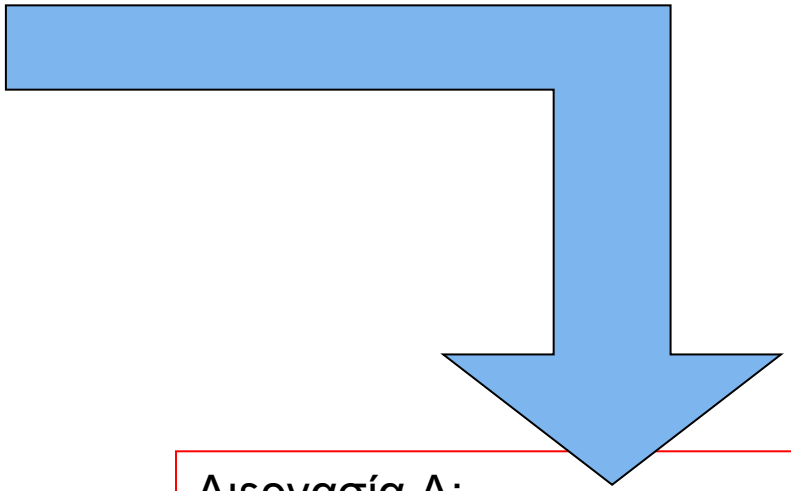
# Οργάνωση λύσης




# "Προστακτικός" προγραμματισμός

- Εκτέλεσε τη διεργασία Α
- Εκτέλεσε τη διεργασία Β
- Εκτέλεσε τη διεργασία Γ
- Εκτέλεσε τη διεργασία Δ

Απεικόνιση σε οδηγίες προς τον υπολογιστη



Διεργασία Α:
Εκτέλεσε τη διεργασία Α1
Εκτέλεσε τη διεργασία Α2

Πώς γνωρίζει το σύστημα τι πρέπει να κάνει για να εκτελέσει τη διεργασία Α;

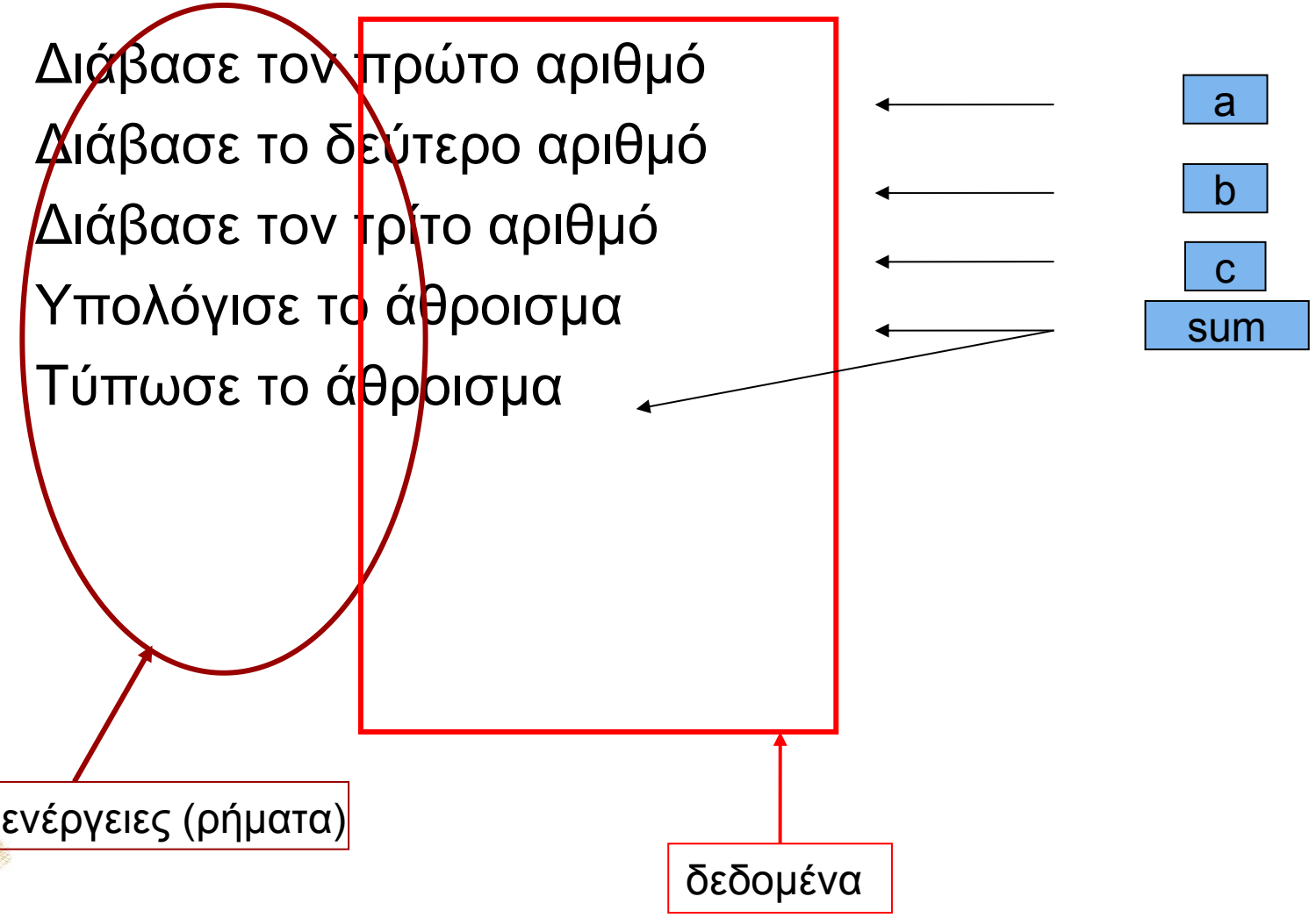
# Αφαιρετικότητα με διεργασίες

```c
#include <stdio.h>
int computeGCD(int, int);

int main ( ) {
 int a, b;
 int gcd;

 scanf ("%d %d", &a, &b) ;

 gcd = computeGCD (a, b);

 printf("%d\n", gcd);
}
```

```c
int computeGCD(int a , int b) {
int i = a, j = b;

 while (i != j) {
   if ( i > j)  {
        i = i - j ;
        }
     else
        j = j - i;
   }

return i;
}
```

# Βήμα 1

- Διάβασε τον πρώτο αριθμό
- Διάβασε το δεύτερο αριθμό
- Διάβασε τον τρίτο αριθμό
- Υπολόγισε το άθροισμα
- Τύπωσε το άθροισμα



# Ρήματα ⇒ διεργασίες
# ⇒ κλήσεις συναρτήσεων

- Διάβασε το a
- Διάβασε το b
- Διάβασε το c
- Υπολόγισε το sum
- Τύπωσε το sum

```
int a, b, c, sum;
```

```
scanf("%d", &a);
scanf("%d", &b);
scanf("%d", &c);
sum = a + b + c;
printf("the sum is: %d\n", sum);
```

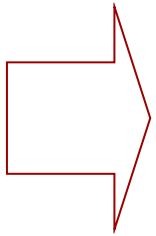
# Δεύτερο Πρόγραμμα σε C

```
#include <stdio.h>

main ( ) {
    int a, b, c, sum;

    scanf("%d", &a);
    scanf("%d", &b);
    scanf("%d", &c);

    sum = a + b + c;

    printf ("sum is %d", sum);
}
```

τι κάνει;

# Ευανάγνωστος Κώδικας

```
i= 120;
if (i>j)
func1();
else
func2();
```



```
velocity = 120;
if (velocity > max_velocity)
    decrease_velocity( );
else
    increase_velocity( ) ;
```

# Μέχρι τώρα...

- Αφαιρετικότητα
  - ως εργαλείο για την αντιμετώπιση της πολυπλοκότητας
  - Σχεδιασμός
  - *Αναφέραμε* την Αυξητική Ανάπτυξη Προγράμματος
    - διευκολύνει τον έλεγχο του τι κάνουμε

# Σημείωμα αναφοράς

- Copyright Πανεπιστήμιο Πατρών, Παλιουράς Βασίλειος , Δερματάς Ευάγγελος «Αρχές Προγραμματισμού ». Έκδοση: 1.0. Πάτρα 2015
- Διαθέσιμο από τη δικτυακιακή διέυθυνση https://eclass.upatras.gr/modules/