ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΧΗΜΙΚΩΝ ΜΗΧΑΝΙΚΩΝ
ΥΠΟΛΟΓΙΣΤΙΚΟ ΚΕΝΤΡΟ

Προγραμματισμός και Χρήση Ηλεκτρονικών Υπολογιστών - Βασικά Εργαλεία Λογισμικού

Μάθημα 9ο

Ακρίβεια υπολογισμών (machine epsilon), overflow, underflow
Σφάλματα αποκοπής και στρογγυλοποίησης (round-off errors)
Μεταβλητές απλής και διπλής ακρίβειας
Απαίτηση σε μνήμη (bytes) για την αποθήκευση των μεταβλητών.

Σφάλματα στρογγυλοποίησης (Round off errors)

Numerical errors
Rounding errors
Floating point errors

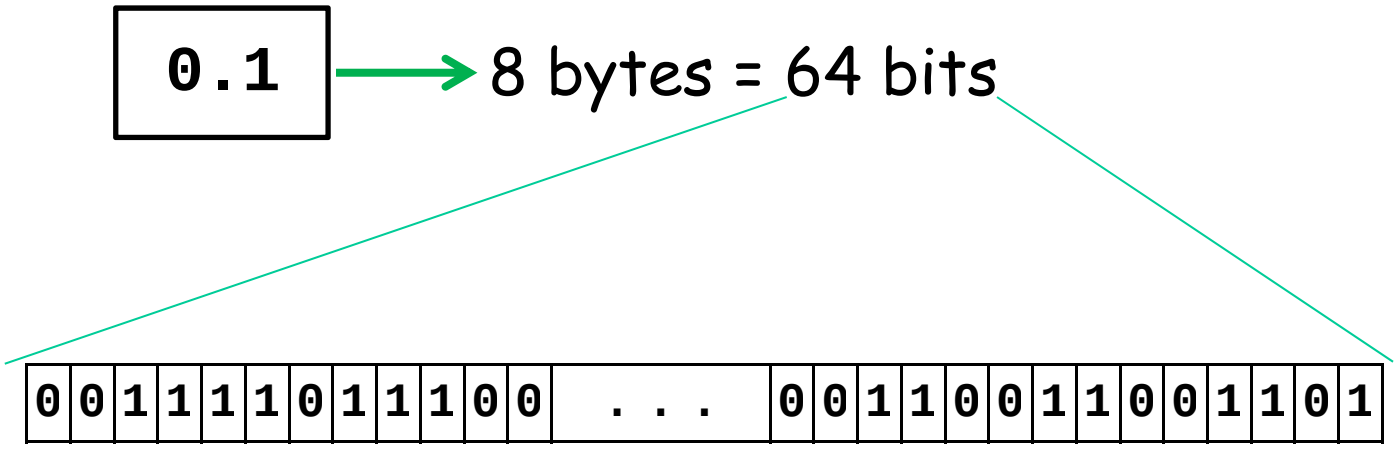
Some disasters caused by numerical errors
http://www-users.math.umn.edu/~arnold//disasters/

Floating-Point Questions Are Endless on stackoverflow.com
http://www.exploringbinary.com/floating-point-questions-are-endless-on-stackoverflow-com

Ορολογία

bit (binary digit): δυαδικό ψηφίο. Τα δυαδικά ψηφία είναι το 0 και το 1
1 byte = 8 bits
word: η θεμελιώδης μονάδα σύμφωνα με την οποία εκπροσωπούνται οι πληροφορίες στον υπολογιστή. Αποτελείται από μια σειρά δυαδικών ψηφίων. Οι πληροφορίες, αποθηκεύονται σε ένα ή περισσότερα words. Το μέγεθος ενός word στους σημερινούς υπολογιστές είναι 32 bit ή 64 bit.

```
my_variable = 0.1
```



Αναπαράσταση αριθμών στο δεκαδικό (decimal) σύστημα

Δεκαδικό ή βάση-10 (base-10)

Η βάση είναι ο αριθμός που χρησιμοποιείται ως αναφορά για την κατασκευή του συστήματος. Το σύστημα βάσης-10 χρησιμοποιεί τα 10 ψηφία 0, 1, 2, 3, 4, 5, 6, 7, 8, και 9 για να αναπαραστήσει τους αριθμούς.

Παράδειγμα

8642.91

$$(8 \times 10^3) + (6 \times 10^2) + (4 \times 10^1) + (2 \times 10^0) + (9 \times 10^{-1}) + (1 \times 10^{-2}) = 8642.91$$

Συμβολισμός θέσης

Αναπαράσταση αριθμών στο δυαδικό (binary) σύστημα

Δυαδικό ή βάση-2 (base-2)

Στους υπολογιστές η αναπαράσταση των αριθμών γίνεται με τη χρήση του δυαδικού συστήματος αρίθμησης. Σύμφωνα με το σύστημα αυτό οι αριθμοί αναπαρίστανται με τα δυαδικά ψηφία (bits) 0 και 1.

Για παράδειγμα,

ο δυαδικός αριθμός 101.1 ή $(101.1)_2$ ισοδυναμεί με:

$$(1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0) + (1 \times 2^{-1}) = 4 + 0 + 1 + 0.5 = 5.5$$

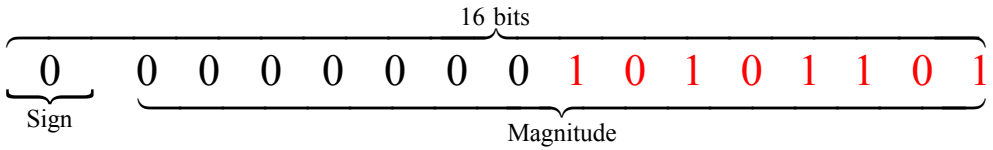
στο δεκαδικό σύστημα ή $(5.5)_{10}$

Αναπαράσταση ακέραιων στους υπολογιστές (1)

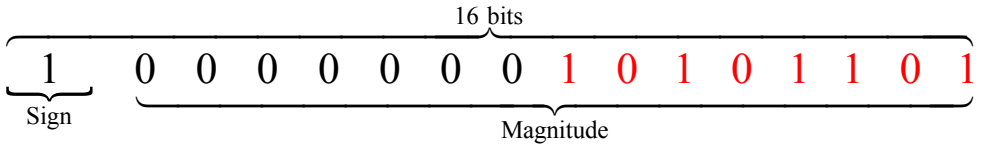
Έστω ο ακέραιος αριθμός 173 που αντιπροσωπεύεται στο δυαδικό ως 10101101

$(10101101)_2 = 2^7 + 2^5 + 2^3 + 2^2 + 2^0 = 128+32+8+4+1 = (173)_{10}$

Σε ένα word με μήκος 16 bits θα αποθηκευτεί ως:



Ο ακέραιος αριθμός -173 θα αποθηκευτεί ως:



Αναπαράσταση ακέραιων στους υπολογιστές (2)

Ο μεγαλύτερος ακέραιος που μπορεί να απεικονιστεί με 16 bits είναι ο

$$(1111111111111111)_2 = (1 \times 2^{14}) + (1 \times 2^{13}) + \ldots + (1 \times 2^{1}) + (1 \times 2^{0}) = 32\ 767$$

$2^{15} - 1 = 32\ 767$

Συνεπώς, ένα word με μήκος 16 bits, μπορεί να αναπαραστήσει τους ακέραιους που κυμαίνονται από -32 767 έως 32 767.

Επιπλέον, επειδή το 0 έχει ήδη οριστεί ως 0000000000000000, είναι περιττό να χρησιμοποιήσουμε τον αριθμό 1000000000000000 για το -0. Έτσι συμβατικά χρησιμοποιείται για να αναπαραστήσει τον αρνητικό αριθμό: -32 768, επεκτείνοντας την περιοχή από -32 768 έως 32 767.

Αναπαράσταση ακέραιων στους υπολογιστές (3)

Ένα word με μήκος n-bit, μπορεί να αναπαραστήσει τους ακέραιους από -2^{n-1} έως $2^{n-1} - 1$

Αν n=32 bit μπορούν να αναπαρασταθούν οι ακέραιοι από -2 147 483 648 έως 2 147 483 647

Αν n=64 bit μπορούν να αναπαρασταθούν οι ακέραιοι από -2^{63} έως $2^{63} - 1$.

Αριθμοί πέρα από αυτά τα όρια δεν μπορούν αναπαρασταθούν.

Οι υπολογιστές έχουν περιορισμένη ικανότητα στην αναπαράσταση ακέραιων αριθμών.

Μεγαλύτεροι είναι οι περιορισμοί στην αποθήκευση και το χειρισμό των πραγματικών αριθμών.

Το σύστημα κινητής υποδιαστολής (floating point)

Αναπαράσταση ενός πραγματικού αριθμού (x) με βάση το 10 στο σύστημα κινητής υποδιαστολής (floating point):

$$x = \pm f \cdot 10^{e}$$

το f ονομάζεται κλασματικό μέρος (fraction / mantissa)
και το e είναι ένας θετικός ή αρνητικός ακέραιος που ονομάζεται εκθέτης (exponent).

Στην αναπαράσταση χρησιμοποιείται μόνο ένα ψηφίο για το ακέραιο μέρος.

Παράδειγμα:

$$3.14 = 3.14 \times 10^{0} = 0.314 \times 10^{1}$$
$$0.000001 = 1.0 \times 10^{-6} = 0.1 \times 10^{-5}$$
$$1941 = 1.941 \times 10^{3} = 0.1941 \times 10^{4}$$

κανονικοποιημένη μορφή με βάση το 10

Ο επιστημονικός συμβολισμός δεν δίνει μια μοναδική αναπαράσταση. Η αναπαράσταση γίνεται μοναδική αν δεχθούμε:
α) το ακέραιο μέρος να είναι μηδέν και
β) το πιο σημαντικό ψηφίο του κλασματικού μέρους να μην είναι μηδέν

Η αναπαράσταση αυτή ονομάζεται κανονικοποιημένη (normalized) μορφή.

Πρότυπο IEEE 754 κινητής υποδιαστολής (1)

κανονικοποιημένη μορφή με βάση το 2

$$x = \pm(1+f)\cdot 2^{e}$$

$f = \frac{i}{2^t}$

i όλοι οι ακέραιοι για τους οποίους ισχύει $0 \le f < 1$

t το πλήθος των bits που δεσμεύονται για την αποθήκευση του f

e όλοι οι ακέραιοι στο διάστημα $[e_{\min}\ e_{\max}]$

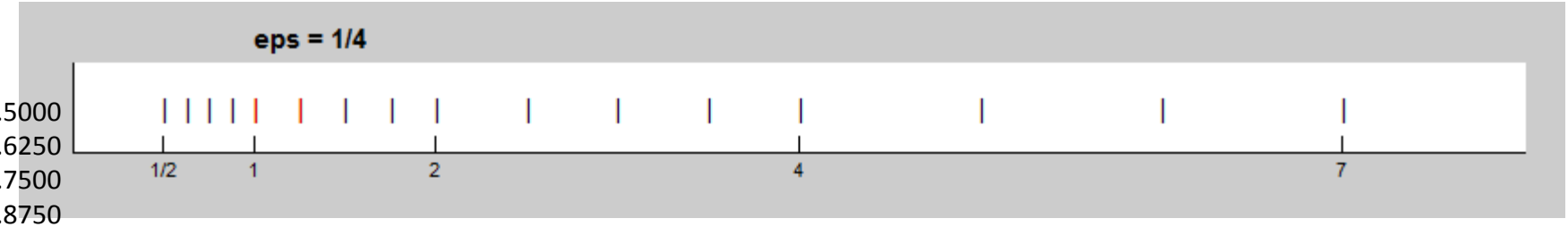
Το e ονομάζεται *exponent* και το 1+f *mantissa*.

Πρότυπο IEEE 754 κινητής υποδιαστολής (2)

Έστω ένας υπολογιστής στον οποίο για την αναπαράσταση των πραγματικών αριθμών χρησιμοποιούνται:

$t = 2, e_{min} = -1, e_{max} = 2$

Σύνολο ***διακριτών τιμών***



0.5000
0.6250
0.7500
0.8750
1.0000
1.2500
1.5000
1.7500
2.0000
2.5000
3.0000
3.5000
4.0000
5.0000
6.0000
7.0000

Θετικοί πραγματικοί αριθμοί μεγαλύτεροι του 7 και μικρότεροι το 0.5 δεν μπορούν να αναπαρασταθούν. Συγκεκριμένα αριθμοί μεγαλύτεροι το 7 προκαλούν *overflow* ενώ μικρότεροι του 0.5 *underflow*.

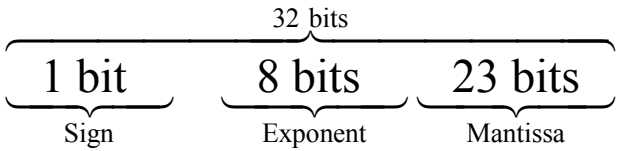
Αριθμοί που είναι μεταξύ 2 διακριτών τιμών του παραπάνω συνόλου δεν μπορούν να αναπαρασταθούν ακριβώς και συνεπώς *στρογγυλοποιούνται* στην πλησιέστερη διακριτή τιμή. Για παράδειγμα ο αριθμός 3.1 θα αποθηκευτεί ως 3. Η διαφορά 3.1 - 3 = 0.1 ονομάζεται σφάλμα στρογγυλοποίησης (round-off error)

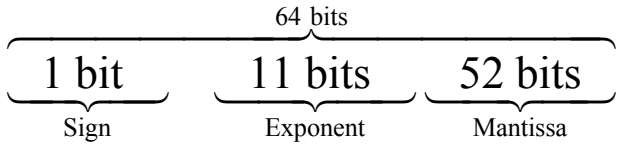
Η απόσταση μεταξύ του αριθμού 1 και του αμέσως μεγαλύτερου αριθμού (στο παράδειγμα μας είναι ο 1.25) ονομάζεται ακρίβεια του υπολογιστή (*machine epsilon* - συμβολίζεται με τη μεταβλητή *eps*)

Πρότυπο IEEE 754 κινητής υποδιαστολής (3)

IEEE Standard 754

	word (bits)	t (bits)	e_{min}	e_{max}
single precision	32	23	-126	127
double precision	64	52	-1022	1023





Πρότυπο IEEE 754 κινητής υποδιαστολής (4)

IEEE Standard 754

Single precision

$\text{realmin} = 2^{e_{\min}} = 2^{-126} = 1.1755\text{e-}38$

$\text{realmax} = (2 - 2^{-t}) \cdot 2^{e_{\max}} = (2 - 2^{-23}) \cdot 2^{127} = 3.4028\text{e+}38$

$eps = 2^{-23} = 1.1921\text{e-}07$

Double precision

$\text{realmin} = 2^{e_{\min}} = 2^{-1022} = 2.2251\text{e} - 308$

$\text{realmax} = (2 - 2^{-t}) \cdot 2^{e_{\max}} = (2 - 2^{-52}) \cdot 2^{1023} = 1.7977\text{e} + 308$

$eps = 2^{-52} = 2.2204\text{e} - 16$

Οι αριθμοί που εμφανίζονται στους υπολογισμούς και κατ΄ απόλυτη τιμή είναι μεγαλύτεροι από την τιμή **realmax** δημιουργούν ***overflow***. Οι αριθμοί αυτοί τίθενται ίσοι με **Infinity** ή **-Infinity** ανάλογα με το αν είναι θετικοί οι αρνητικοί αντίστοιχα.

Οι αριθμοί που εμφανίζονται στους υπολογισμούς και κατ΄ απόλυτη τιμή είναι μικρότεροι από την τιμή **realmin** δημιουργούν **underflow**. Οι αριθμοί αυτοί τίθενται ίσοι με το **μηδέν**.

Ποσότητες που δεν μπορούν να αναπαρασταθούν (π.χ. σε ορισμένα αποτελέσματα μαθηματικών συναρτήσεων) τίθενται ίσοι με **NaN** (Not-a-Number).

Απαίτηση σε υπολογιστική μνήμη

```
>> my_variable = zeros(10000);
>> whos
  Name                  Size                    Bytes  Class     Attributes

  my_variable      10000x10000           800000000  double
```

Γιατί η `my_variable` έχει απαίτηση σε μνήμη 800 MB;

Αναπαράσταση του 0.1 στο δυαδικό (binary) σύστημα

Για να απεικονιστεί ο πραγματικός αριθμός 0.1 στο δυαδικό σύστημα απαιτείται μια άπειρη σειρά:

$$\frac{1}{10} = \frac{1}{2^4} + \frac{1}{2^5} + \frac{0}{2^6} + \frac{0}{2^7} + \frac{1}{2^8} + \frac{1}{2^9} + \frac{0}{2^{10}} + \frac{0}{2^{11}} + \frac{1}{2^{12}} + \dots$$

μετά τον πρώτο όρο η ακολουθία 1,0,0,1 επαναλαμβάνεται συνεχώς.

Συνεπώς η δυαδική αναπαράσταση του πραγματικού αριθμού 0.1 είναι:

$(0.1)_{10} = (0.0001100110011001\dots)_2$

Machine epsilon

Τρέξτε το παρακάτω script:

```
clear;

e=1;

while (true)
  a=1+e;
  if (a==1.); break; end
  e=e/2;
end

myeps = 2*e
```

Τί αντιπροσωπεύει η τιμή που αποθηκεύεται στη μεταβλητή myeps;

Ισότητα αριθμών κινητής υποδιαστολής ή "how close is close"

a == b → **match bit to bit**

Η ερώτηση της ισότητας "γεννάει" μια άλλη ερώτηση : "Τι εννοούμε με τον όρο ισότητα";

Στον προγραμματισμό η ισότητα σημαίνει "αρκετά κοντά".
Ο έλεγχος της "ισότητας" γίνεται ορίζοντας κάποια μικρή απόσταση **e** ως "αρκετά κοντά"

Όμως:
e = 0.00001
είναι κατάλληλο για αριθμούς γύρω από το 1
πολύ μεγάλο για αριθμούς γύρω από το 0.00001
και πολύ μικρό για αριθμούς γύρω από το 10000

Equality test
Όχι: if (a==b)
Ναι: if (abs(a-b)< e)

Πράξεις μεταξύ αριθμών κινητής υποδιαστολής

ΠΡΟΣΟΧΗ

Αφαιρέσεις "κοντινών" αριθμών

Προσθέσεις/Αφαιρέσεις αριθμών με μεγάλη διαφορά

$$1+\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{1000000}=14.39272672286499$$

$$\frac{1}{1000000}+\cdots+\frac{1}{3}+\frac{1}{2}+1=14.39272672286577$$