CSE 541

ELEMENTARY NUMERICAL METHODS

Computer (finite)
Representation of numbers

Number Bases & Conversions

$12345 = 5*10^0 + 4*10^1 + 3*10^2 + 2*10^3 + 1*10^4$

$12345_8 = 5*8^0 + 4*8^1 + 3*8^2 + 2*8^3 + 1*8^4$

conversion of arbitrary base to decimal

conversion of decimal to arbitrary base

conversion of arbitrary base to arbitrary base

convert to decimal: do base math in decimal
convert to arbitrary base: repeated divide by base: generates numbers in right to left order
divide by largest power of base less than number: generates numbers in left to right order
usually convert base1 to decimal and covnert decimal to base2

Fractional Conversions

$.12345 = 1*10^{-1} + 2*10^{-2} + 3*10^{-3} + 4*10^{-4} + 5*10^{-5}$

$.12345_8 = 1*8^{-1} + 2*8^{-2} + 3*8^{-3} + 4*8^{-4} + 5*8^{-5}$

conversion of arbitrary base to decimal

conversion of decimal to arbitrary base

conversion of arbitrary base to arbitrary base

(same idea - just use negative powers of base)
do math in decimal
repeatedly divide by negative powers of base to generate left to right
convert to and from decimal

Binary Number System

$$10110 = 0*2^0 + 1*2^1 + 1*2^2 + 0*2^3 + 1*2^4$$

Notations of Binary Numbers

hexadecimal

1011	0011	0101	0110	1001	0011
D	C	5	6	9	3

octal

101	100	110	101	011	010	010	011
7	4	6	5	3	2	2	3

used as notation to save ink/space - not really used as a base for computations

Conversion to binary using octal

12345 / 8 = 1543 remainder: 1
1543 / 8 = 192 remainer: 7
192 / 8 = 24 remainer: 0
24 / 8 = 3 remainder: 0
3 / 8 = 0 remainder: 3
$12345 = 30071_8$
$12345 = 011\ 000\ 000\ 111\ 001_2$

Signed Integer numbers

32

n

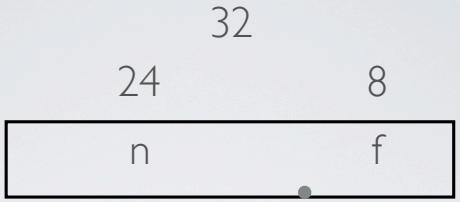
two's complement

$-2^{31} < n < 2^{31} - 1$

1111	15
1110	14
1101	13
1100	12
1011	11
1010	10
1001	9
1000	8
0111	7
0110	6
0101	5
0100	4
0011	3
0010	2
0001	1
0000	0

0111	7
0110	6
0101	5
0100	4
0011	3
0010	2
0001	1
0000	0
1111	-1
1110	-2
1101	-3
1100	-4
1011	-5
1010	-6
1001	-7
1000	-8

Fixed Point numbers



2's complement integer

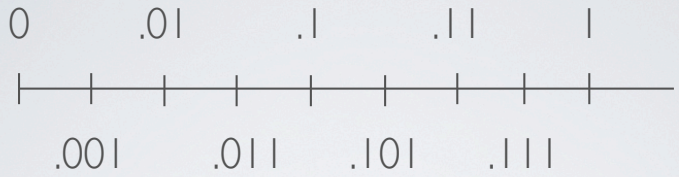
no hardware support - implement in software

$-2^{23} < n < 2^{23} - 1$ $\quad 0 < f <= .11111111_2$

$0 < f <= .99609375$

in software - how to add
how to multiply - divide out fractional part
how to divide - multiply by base, convert remainder to fraction

Binary Fraction



more fractional bits distributes numbers throughout number line

Floating point numbers

		normalized
123.45	123.45×10^{0}	$.12345 \times 10^{3}$
.000012345	$.000012345 \times 10^{0}$	$.12345 \times 10^{-4}$

$+/- \; r \times 10^{n} \qquad 1/10 <= r < 1$

binary floating point numbers

$+/- \; q \times 2^{n} \qquad 1/2 <= q < 1$

Computer representation of floating point numbers

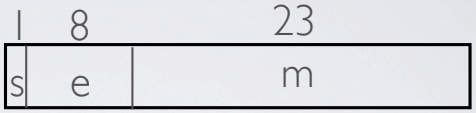
finite precision irrational numbers

$0.1 = (0.0001100110011001100110011001100110011...)_2$

see footnote on page 55 of book

Computer representation of floating point numbers

IEEE standard



normalized form: shift left so leftmost bit is '1'

exponent in excess notation +/- 2^{e-127} x 1.m

underflow overflow

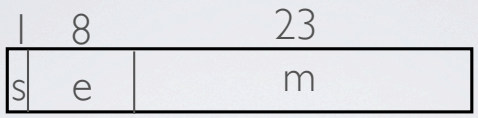
note: why 1.m for mantissa ?
note exponent is excess notation, not 2's complement
double precision expands mantissa
how to represent 1?

Double Precision

IEEE standard

normalized form +/- 2^{e-1023} x 1.m

Errors roundoff error



0.0010

0.000110000000000000000001

0.00100111111111111111111111

.0010 .001001111111111111 : .00011000000000000000001

Errors loss of significance

- Consider the error for *x*-*y* using 5 decimal digits of precision:

  x = .3721448693
  y =. 3720214371
  x' = .37214
  y' =. 37202
  x'-y' = .00012
  x-y = .0001234322

.0010 .001001111111111111 : .00011000000000000000001

Errors loss of significance

- The relative error is:

$$3 * 10^{-2}$$

- However, the relative error of *x'* and *y'* is only $1.3*10^{-5}$.
- We lost 3 significant digits.

The closer the two numbers, the greater the loss of significance

.0010 .001001111111111111 : .00011000000000000000001

Avoiding loss of significance

Use double precision or higher

Modify the calculations to remove subtraction of numbers close together

Consider: $f(x) = \sqrt{x^2+1} - 1$ as x approaches 0

Reorder to remove the subtraction

$$f(x) = (\sqrt{x^2+1} - 1)\left(\frac{\sqrt{x^2+1}+1}{\sqrt{x^2+1}+1}\right) = \frac{x^2}{\sqrt{x^2+1}+1}$$

subtraction of similar numbers can result in loss of precision

- Look at the C include file *float.h*

```
#define DBL_DIG         15                      /* # of decimal digits of precision */
#define DBL_EPSILON     2.2204460492503131e-016 /* smallest such that 1.0+DBL_EPSILON != 1.0
    */
#define DBL_MANT_DIG    53                      /* # of bits in mantissa */
#define DBL_MAX         1.7976931348623158e+308 /* max value */
#define DBL_MAX_10_EXP  308                     /* max decimal exponent */
#define DBL_MAX_EXP     1024                    /* max binary exponent */
#define DBL_MIN         2.2250738585072014e-308 /* min positive value */
#define DBL_MIN_10_EXP  (-307)                  /* min decimal exponent */
#define DBL_MIN_EXP     (-1021)                 /* min binary exponent */
#define _DBL_RADIX      2                       /* exponent radix */
#define _DBL_ROUNDS     1                       /* addition rounding: near */

#define FLT_DIG         6                       /* # of decimal digits of precision */
#define FLT_EPSILON     1.192092896e-07F        /* smallest such that 1.0+FLT_EPSILON != 1.0
    */
#define FLT_GUARD       0
#define FLT_MANT_DIG    24                      /* # of bits in mantissa */
#define FLT_MAX         3.402823466e+38F        /* max value */
#define FLT_MAX_10_EXP  38                      /* max decimal exponent */
#define FLT_MAX_EXP     128                     /* max binary exponent */
#define FLT_MIN         1.175494351e-38F        /* min positive value */
#define FLT_MIN_10_EXP  (-37)                   /* min decimal exponent */
#define FLT_MIN_EXP     (-125)                  /* min binary exponent */
```

subtraction of similar numbers can result in loss of precision

For Tuesday

Read Chapter 3

Homework #1

Makefile

```
hw1:
    gcc -lm hw1.c -o hw1
```

use g++ for C++

submit

submit c541aa hw1 hw1.c Makefile

submit c541aa hw1 myhw1/

hw1

```
#define Real float
#include <math.h>

int  main()
{
    Real x,y,z;
    y = 0.1;
    z = 1000.0;

    x = y+z;
    x = x-z;

    printf("x: %20.17f; y: %20.17f\n",x,y);

    if (x == y) printf("yes\n");
    else printf("no\n");
    return 0;
}
```