January 17, 2001

Important announcements

[Class home page]
[Homework assignment 1]
[Homework assignment 2]

Positional notation in base (or radix) *b*

$d_{n-1}\ d_{n-2} \ldots d_1\ d_0$ **is really** $\sum_{i=0}^{n-1} d_i b^i$

2001 **is really** $2*10^3 + 0*10^2 + 0*10^1 + 1*10^0$

Conversions to/from decimal (base 10)

Random base $\rightarrow$ base 10

Multiply and add

Base 10 $\rightarrow$ Random base

Divide and take the remainder

Dealing with base 16 (hexadecimal)

Digits are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

Practice in a spreadsheet

Bit groupings

In decimal

6122428867 **is written as** 6,122,428,867

Binary to hexadecimal

```
10110110011101100110110011100001 1
1,0110,1100,1110,1100,1101,1001,1100,0011
16CECD9C3
```

Bases in C and C++

octal	`03721`
decimal	`2001`
hexadecimal	`0x7D1`

Addition and Subtraction

Addition	Subtraction
41573 + 30742 72315	340010 - 817 339193
1000101 +1011101	1000101 - 11001

[More practice with a spreadsheet]

Representing negative numbers

A natural way to represent -17

Subtract 17 from 0 and see what you get!

```
  0000000000000000
- 0000000000010001
```

Two's complement representation

Binary numbers starting with 1 are *negative*

Expressing a negative number in *n*-bit two's complement

Convert to binary	100010
Extend to *n* bits	0000000000100010
Invert all bits Generate the one's complement	1111111111011101
Add in a one Not it's the two's complement	1111111111011101 + 1 1111111111011110

Expressing a positive number in n-bit two's complement

Convert to binary	100010
Extend to *n* bits	0000000000100010

The first bit is the -2^{n-1} place

$d_{n-1}\, d_{n-2} \ldots d_1\, d_0$ **is now really** $d_{n-1}(-b^{n-1}) + \sum_{i=0}^{n-2} d_i b^i$

Other negative number representations

Sign-magnitude

One's complement

Overflow

8-bit numbers

	unsigned	signed
`00000000`	`0`	`0`
`01111111`	`127`	`127`
`10000000`	`128`	`-128`
`11111111`	`255`	`-1`

Range of 8-bit two's complement is –128 to 127
or -2^{8-1} to $2^{8-1}-1$

Range of *n*-bit two's complement is –128 to 127
or -2^{n-1} to $2^{n-1}-1$

What about

```
100 + 50
–100 – 50
100 – 50
```

Detecting overflow

Compare last carry-in with last carry-out