January 28, 2002

Important announcements

Class home page
Class syllabus
Class schedule
Homework assignment 1

Two important points from Chapter 1

1) Given enough (tape) storage and enough time, all computers can compute the same set of problems.
2) "Problems" are solving by transformations between abstractions.

Turing machine

A simple "computer" and an infinite "tape".
Read and then write a tape symbol at each computer step.

Universal Turing machine

A Turing machine that can simulate any Turing machine.

Turing's thesis

Every computation can be computed by a Turing machine.

Who was Alan Turing?

Layers of Abstraction

Problem	A natural-language description of the task to be solved
Algorithm	An abstract specification of the computational methods needed to solve the task
Program	What you wrote in CSCI 201
Instruction set architecture	The 1's and 0's used by the computer to execute your program. For example, the x86 machine code.
Microarchitecture	The processor that execute your code. For example, a Pentium II or an AMD Athlon.
Circuits	Hardware "modules", such as adders and multiplexers.
Devices	Transistors, such CMOS or bipolar.

Example of a transformation

From the C program:

```
int Add2Big(int A, int B, int C)
{
  int R ;
  if (A < B)
    R = A ;
  else
    R = B ;
  if (C < R)
    R = C ;
  return(A+B+C-R) ;
} ;
```

The GCC compiler generates the assembly code:

```
Add2Big:
      pushl %ebp
      movl %esp,%ebp
      subl $4,%esp
      movl 8(%ebp),%eax
      cmpl 12(%ebp),%eax
      jge .L2
      movl 8(%ebp),%eax
      movl %eax,-4(%ebp)
      jmp .L3
.L2:
      movl 12(%ebp),%eax
      movl %eax,-4(%ebp)
.L3:
      movl 16(%ebp),%eax
      cmpl -4(%ebp),%eax
      jge .L4
      movl 16(%ebp),%eax
      movl %eax,-4(%ebp)
.L4:
      movl 8(%ebp),%eax
      movl 12(%ebp),%ecx
      leal (%ecx,%eax),%edx
      addl 16(%ebp),%edx
      subl -4(%ebp),%edx
      movl %edx,%eax
      jmp .L1
.L1:
      leave
      ret
```

16(%ebp)	C
12(%ebp)	B
8(%ebp)	A
4(%ebp)	
0(%ebp)	
-4(%ebp)	R

Representation of unsigned numbers

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 → base 10

Multiply and add

Base 10 → 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

Conversion in a spreadsheet

Bit groupings

In decimal

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

Binary to hexadecimal

```
10110110011101100110110011100011
1,0110,1100,1110,1100,1101,1001,1100,0011
16CECD9C3
```

Bases in Java, C, and C++

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

Addition and Subtraction

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

Unsigned addition and subtraction with a spreadsheet

Representing signed numbers

A natural way to represent -34

Subtract 34 from 0 and see what you get!

```
      0000000000000000
    - 0000000000100010
      1111111110111110
```

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** $d_{n-1}(-b^{n-1}) + \sum_{i=0}^{n-2} d_i b^i$

Signed conversion with a spreadsheet

Other negative number representations

Sign-magnitude

One's complement