CMPSCI 105: Lecture #3
Base Conversions

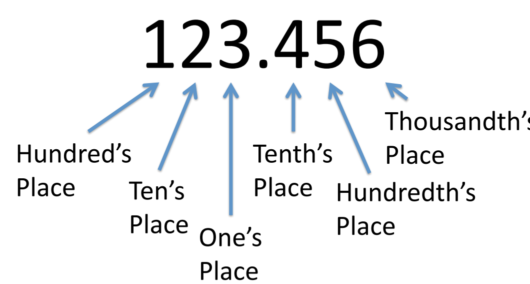
Base Conversions

- What are bases?
  - Labels for numbers using different sets of symbols.
- Why do we care?
  - We use base 10 (decimal) but it isn't special,
  - Understand how computers work internally,
  - Used in color specifications for Web design,
  - Used in file permissions on Web servers (UNIX),
  - Used in many other places.

Ordinary Decimal Numbers

- Uses the positional notation developed by Arab and Hindu mathematicians of a thousand years ago.
- Requires the invention of "zero" to work (so that 12, 120, 102, 10.2, 100.02 are all distinct).
- Contrast with Roman numerals (I=1, V=5, X=10, L=50, C=100, D=500, M=1000).

Ordinary Decimal Numbers

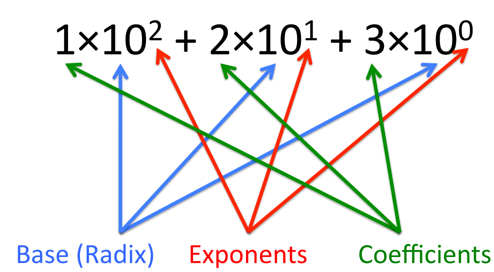


Decimal Expansion of 123.456

$= 1\times100 + 2\times10 + 3\times1 + 4\times0.1 + 5\times0.01 + 6\times0.001$

$= 1\times10^2 + 2\times10^1 + 3\times10^0 + 4\times10^{-1} + 5\times10^{-2} + 6\times10^{-3}$

Decimal Expansion of 123



Definitions

- Base (Radix):
  - Number of distinct symbols used for counting
- Exponents:
  - Powers of the base for each digit,
  - Increase linearly going to the left, decrease to the right (..., 3, 2, 1, 0, -1, -2, -3, ...),
  - Determines the weight/contribution of each digit
- Coefficients:
  - Digits of the number
  - Range is always from 0 to Base-1

 6

What if the Base isn't 10?

- So what? Math still works.
- Changing a base changes only the *labeling* of a number, not its *value*.
- Historical precedent:
  - Sumerians & Babylonians used base 60 (so do we in our clocks and angle measures!)
  - Mayans/Aztecs/Africans/some Europeans used base 20.

 7

Case #1: Base 4

- Only four symbols used for counting: 0, 1, 2, 3,
- No coefficient digit value can exceed 3,
- Exponents are powers of 4,
- Everything else is the same.

 8

Case #1: Base 4

- What, then does $\mathbf{1203_4}$ mean? (Notice the subscript indicating the base)
- Using the same expansion technique:
  - $1{\times}4^3 + 2{\times}4^2 + 0{\times}4^1 + 3{\times}4^0$ =
  - $1{\times}64 + 2{\times}16 + 0{\times}4 + 3{\times}1$ =
  - $64 + 32 + 0 + 3$ =
  - 99_{10}
- Therefore, 1203_4 is the *same number* as 99_{10}

 9

What Happened?

- How did we get to Base 10 all of a sudden?
- When we wrote out the expansion of 1203_4 as $1{\times}4^3 + 2{\times}4^2 + 0{\times}4^1 + 3{\times}4^0$, we wrote it down using base 10 (decimal) notation.
- The rest is just arithmetic!
  - Powers first,
  - Multiplications second,
  - Additions third.

 10

Case #2: Base 2 (Binary)

- Only two symbols used for counting: 0, 1,
- No coefficient digit value can exceed 1,
- Exponents are powers of 2,
- Everything else is the same.

 11

Case #2: Base 2 (Binary)

- What, then does **110101_2** mean?
- Using the same expansion technique:
  - $1{\times}2^5 + 1{\times}2^4 + 0{\times}2^3 + 1{\times}2^2 + 0{\times}2^1 + 1{\times}2^0 =$
  - $1{\times}32 + 1{\times}16 + 0{\times}8 + 1{\times}4 + 0{\times}2 + 1{\times}1 =$
  - $32 + 16 + 0 + 4 + 0 + 1 =$
  - 53_{10}
- Therefore, 110101_2 is the *same* as 53_{10}
- Notice that multiplications are only by 0 or 1.

A Shortcut for Binary

- Write down the powers-of-two placeholder weights above each digit.
- Wherever there is a 1, add the weight to the total, ignore zeroes.

32 16 8 4 2 1

110101

Generalizations

- Conversion techniques presented so far work for converting numbers in bases 2, 3, 4, 5, 6, 7, 8, 9, 10 back into base 10.
- But what about bases greater than 10?
- We will need more symbols.
- By tradition, we use the letters from the Roman alphabet:
  - 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, G, H, …
- Allows up to base 36 (10 digits, 26 letters).

Case #3: Base 16 (Hexadecimal)

- Sixteen symbols used for counting: 0-9 & A-F,
- A=10, B=11, C=12, D=13, E=14, F=15,
- No coefficient digit value can exceed F=15,
- Exponents are powers of 16,
- Everything else is the same.

Case #3: Base 16 (Hexadecimal)

- What, then does **$1AE_{16}$** mean?
- Using the same expansion technique:
  - $1{\times}16^2 + A{\times}16^1 + E{\times}16^0 =$
  - $1{\times}16^2 + 10{\times}16^1 + 14{\times}16^0 =$
  - $1{\times}256 + 10{\times}16 + 14{\times}1 =$
  - $256 + 160 + 14 =$
  - 430_{10}
- Therefore, $1AE_{16}$ is the *same* as 430_{10}

Converting Integers From Base 10

- We now know how to convert to base 10…
- …but what about converting from base 10 to another base?
- Can we know ahead of time how many digits in the target base we will need?
- Two Methods to know:
  - #1: Simple, understandable, lengthy computations
  - #2: Fast, efficient, hard to explain why it works

How Many Digits in Target Base?

- We need to compute logarithm in the target base B of decimal number N as: **$\log_B(N)$**.
- That is, *what power of B gives us N?*
- Unfortunately, logarithms in base B aren't usually available, so we can compute it with…
- …**$\log_A(N) \div \log_A(B)$** for existing log in base A.
- Finally, take the ⌈ceiling⌉ of the result
  - Smallest integer no smaller than the result.

Example: 99_{10} to Base 4

- Want: $\lceil \log_A(99) \div \log_A(4) \rceil$
- If A=10 (common log):
  - $\lceil \log_{10}(99) \div \log_{10}(4) \rceil$
  - $\lceil 1.99563519459755 \div 0.602059991327962 \rceil$
  - $\lceil 3.3146783100398 \rceil$
  - 4
- Base 4 answer needs 4 digits.

Example: 99_{10} to Base 4

- Want: $\lceil \log_A(99) \div \log_A(4) \rceil$
- If A=*e* (natural log):
  - $\lceil \log_e(99) \div \log_e(4) \rceil$
  - $\lceil 4.59511985013459 \div 1.386294361119891 \rceil$
  - $\lceil 3.3146783100398 \rceil$
  - 4
- Base 4 answer still needs 4 digits.

Method #1

- Write out "enough" powers of the target base to cover the job, but no more,
- Distribute values from base 10 number to largest digit (without exceeding maximum value in target base),
- Repeat with next smallest digit, continue,
- After filling out rightmost digit there should be nothing left.

99_{10} to Base 4 (should be 1203_4)

- Write out "enough" powers of 4, compute:
  - … 4^6 4^5 4^4 4^3 4^2 4^1 4^0
  - … 4096 1024 256 | 64 16 4 1
  - Don't need anything bigger than 64 (four digits)
- Distribute:
  - How many 64 are in 99? **1**, with 35 left over,
  - How many 16 are in 35? **2**, with 3 left over,
  - How many 4 are in 3? **0**, with 3 left over,
  - How many 1 are in 3? **3**, with 0 left over.

430_{10} to Base 16 (should be $1AE_{16}$)

- Write out "enough" powers of 16, compute:
  - … 16^4 16^3 16^2 16^1 16^0
  - … 65536 4096 | 256 16 1
  - Don't need anything bigger than 256 (three digits)
- Distribute:
  - How many 256 are in 430? **1**, with 174 left over,
  - How many 16 are in 174? **10**, with 14 left over,
  - How many 1 are in 14? **14**, with 0 left over.
  - Replace digits>9 with letters: **1AE**

Method #2

- Divide decimal number by the target base, save remainder,
- Repeat process with quotient,
- Stop when quotient is zero.
- Sequence of remainders is answer in right-to-left order.

99_{10} to Base 4 (should be 1203_4)

- 99 ÷ 4 = 24 R **3**,
- 24 ÷ 4 = 6 R **0**,
- 6 ÷ 4 = 1 R **2**,
- 1 ÷ 4 = 0 R **1**. (Quotient=0 means stop!)
- Result = 1203_4.

53_{10} to Base 2 (should be 110101_2)

- 53 ÷ 2 = 26 R **1**,
- 26 ÷ 2 = 13 R **0**,
- 13 ÷ 2 = 6 R **1**,
- 6 ÷ 2 = 3 R **0**,
- 3 ÷ 2 = 1 R **1**,
- 1 ÷ 2 = 0 R **1**. (Quotient=0 means stop!)
- Result = 110101_2.

430_{10} to Base 16 (should be $1AE_{16}$)

- 430 ÷ 16 = 26 R 14, (replace 14 with **E**),
- 26 ÷ 16 = 1 R 10, (replace 10 with **A**),
- 1 ÷ 16 = 0 R **1**. (Quotient=0 means stop!)
- Result = $1AE_{16}$.

Back to Bytes

- Decimal values 0…255,
- Binary values 00000000…11111111,
- Hexadecimal values 00…FF.
  - $F \times 16^1 + F \times 16^0 =$
  - $15 \times 16 + 15 \times 1 =$
  - $240 + 15 =$
  - 255
- One Byte is *exactly two* hexadecimal digits!
- Therefore, each hex digit is exactly 4 bits.

Base 2 to Base 2^N for any N

To convert a binary number to any base which is a power of two:

- Start from the right…
- …and partition the binary number into packets of N bits per packet,
- If leftmost packet contains fewer than N bits, pad on left with 0s, and then…
- …convert each packet as a separate problem.

Base 2 to Base 16 (Hexadecimal)

- Binary number **10110101101111**,
- Partition into groups of 4 bits ($2^4 = 16^1$),
  - 0010 1101 0110 1111
- Convert each packet separately:
  - 0010 = 2, 1101 = 13 = D, 0110 = 6, 1111 = 15 = F
- Hexadecimal result is **2D6F**

 30

Base 2 to Base 8 (Octal)

- Binary number **10110101101111**,
- Partition into groups of 3 bits ($2^3 = 8^1$),
  - 010 110 101 101 111
- Convert each packet separately:
  - 010 = 2, 110 = 6, 101 = 5, 101 = 5, 111 = 7
- Octal result is **26557**

 31