06 Binary Numbers

Instructor: Ke Wei（柯韋）

A319 ✆ Ext. 6452 ✉ wke@ipm.edu.mo

http://brouwer.ipm.edu.mo/COMP112/18/

Bachelor of Science in Computing, School of Public Administration, Macao Polytechnic Institute

September 14, 2018

Outline

Decimal Numbers

- We normally use decimal numbers to communicate with others.
- There are ten digits in the decimal number system: $0, 1, 2, 3, 4, 5, 6, 7, 8, 9$.
- When we count to ten, we make a *carry*.
- A decimal number $d_{n-1} \cdots d_1 d_0$ represents an integer value of

$$d_{n-1} \times 10^{n-1} + \cdots + d_1 \times 10^1 + d_0 \times 10^0.$$

For example,

$$12345_{\text{dec}} = 1 \times 10000 + 2 \times 1000 + 3 \times 100 + 4 \times 10 + 5 \times 1.$$

- We can use n decimal digits to represent integer values in the range of 0 to $10^n - 1$.

Numbers in Radix R

- We may observe that in an R-ary number system,
- There are R digits ranging from 0 to $R-1$.
- When we count to R, we make a *carry*.
- An R-ary number $x_{n-1}\cdots x_1 x_0$ represents an integer value of

$$x_{n-1}\times R^{n-1}+\cdots+x_1\times R^1+x_0\times R^0.$$

- We can use n digits to represent integer values in the range of 0 to R^n-1.
- We must have $R\geq 2$.

Binary Numbers

- Many objects are easily having two states. That's why the binary number system is used in computers.
- There are only two digits in the binary number system: 0 and 1.
- When we count to two, we make a carry.
- A binary number $b^{n-1} \cdots b_1 b_0$ represents an integer value of

$$b_{n-1} \times 2^{n-1} + \cdots + b_1 \times 2^1 + b_0 \times 2^0.$$

  For examples, $1101_{\text{bin}} = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 13_{\text{dec}}$.
- We can use n binary digits to represent integer values in the range of 0 to $2^n - 1$.
- A binary digit is often called a *bit*.
- Let's count in the binary number system:

$$\begin{array}{rrrrrrrr} 0, & 1, & 10, & 11, & 100, & 101, & 110, & 111, \\ 1000, & 1001, & 1010, & 1011, & 1100, & 1101, & 1110, & 1111. \end{array}$$

Negative Numbers

- However, we don't have an extra symbol '−', negative integers have to be *encoded*.
- Three systems are widely used to represent both positive and negative integers:

  Sign-and-magnitude, 1's-complement and 2's-complement.
- In all three systems,
  - The left most bit is 0 for positives and 1 for negatives.
  - Positives are identical in all systems.
  - Negative values have different representations.
- If we are using n bits to represent these numbers, then

Sign-and-magnitude	only changing the left most bit.
1's-complement	changing each bit, equivalent to subtracting from $2^n - 1$.
2's-complement	subtracting from 2^n.

2's-complement is the most often used system in computers to represent negative numbers.

The Three Representations (Using 4 Bits)

Binary code

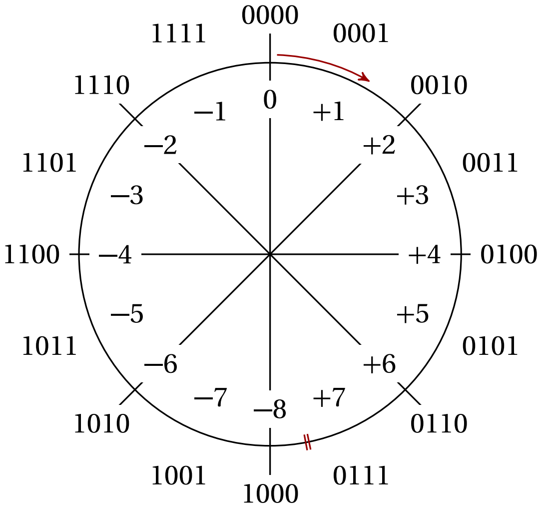
	0111	0110	0101	0100	0011	0010	0001	0000	1000	1001	1010	1011	1100	1101	1110	1111
Sign-and-magnitude	$^+7$	$^+6$	$^+5$	$^+4$	$^+3$	$^+2$	$^+1$	$^+0$	$^-0$	$^-1$	$^-2$	$^-3$	$^-4$	$^-5$	$^-6$	$^-7$
1's-complement	$^+7$	$^+6$	$^+5$	$^+4$	$^+3$	$^+2$	$^+1$	$^+0$	$^-7$	$^-6$	$^-5$	$^-4$	$^-3$	$^-2$	$^-1$	$^-0$
2's-complement	$^+7$	$^+6$	$^+5$	$^+4$	$^+3$	$^+2$	$^+1$	0	$^-8$	$^-7$	$^-6$	$^-5$	$^-4$	$^-3$	$^-2$	$^-1$

Value represented

2’s Complements

- The 2’s-complement system unifies addition and subtraction for both positives and negatives.
- We can obtain the 2’s complement by keeping the right most ‘1’ and the bits right to it, and changing all the bits left to the ‘1’.

Original	2’s Complement
1100	0100
10101111	01010001
1111	0001
1101011001001000	
00111001	
10000000	



Octal and Hexadecimal

- It's too troublesome to write down a non-trivial binary number (too many digits).
- Translations between binary and decimal numbers require heavy calculations, and the two systems are *visually* unrelated.
- We can overcome these by grouping 3 or 4 bits into one digit:

  Octal one digit represents 3 bits, we need 8 digits: 0,1,2,3,4,5,6,7.

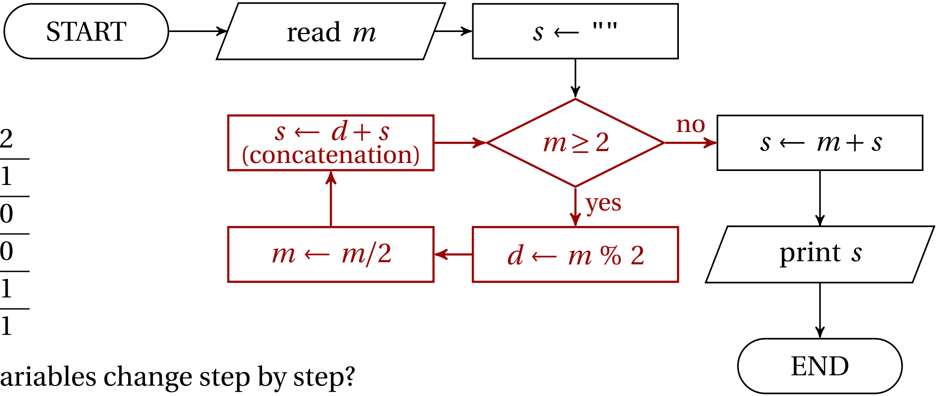
  Hexadecimal one digit represents 4 bits, we need 16 digits: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

  .
- Conversion from binary to octal or hexadecimal is easy:

$$011,111,001,010_{\text{bin}} = 3712_{\text{oct}} \qquad 0111,1100,1010_{\text{bin}} = 7CA_{\text{hex}}.$$

- Since word lengths are usually of power of 2, a number can seldom be wholly divided into groups of 3-bits. Hexadecimal number system is more commonly used.
- An octal literal starts with `0`, a hexadecimal literal starts with `0x`, and a binary literal starts with `0b` (starting from J2SE 7.0, also underscores are allowed between digits since then).

Printing a Number in Binary

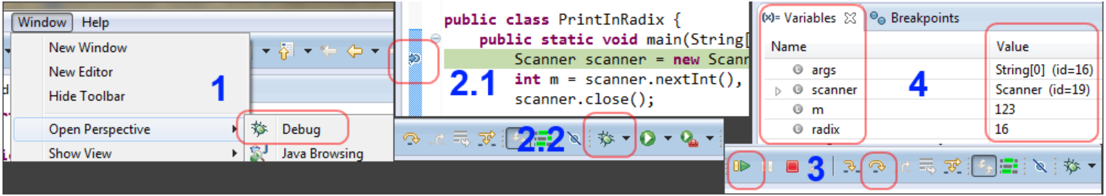


m	$m/2$	$m \% 2$
25	12	1
12	6	0
6	3	0
3	1	1
1	0	1

- How do the variables change step by step?
- In what condition should we exit the loop?
- Will the condition be met?
- How can we collect the digits obtained during the loop in a *correct* order?

Debugging Java Programs in Eclipse

- Eclipse provides a very powerful debugger for Java. We can perform fundamental debugging operations easily, such as *setting breakpoints, single step over statements* and *inspecting variables.*
- To debug your programs, follow these general steps:
  1. Open the "Debug" perspective.
  2. Set an initial breakpoint to pause your program.
  3. When paused, set more breakpoints and continue, or single step over the statements.
  4. When paused, inspect your variables, and think about why they have the current values.



Reading Homework

Textbook

- 3.16, Appendix F.

Internet

- Binary number (`http://en.wikipedia.org/wiki/Binary_number`).
- Breakpoint (`http://en.wikipedia.org/wiki/Breakpoint`).