Introduction to Computer Science

Iain Styles
I.B.Styles@cs.bham.ac.uk

Organisation

- I am coordinating initially and will give first three weeks lectures.
  - This *may* change
- Two lectures per week
  - Tuesday 4-5pm
  - Wednesday 9-10am
  - Rooms change quite a lot – check carefully
- All materials on Canvas

Assessment

- 80% by unseen written examination in May
- 20% by Coursework
  - ???
- Module pass mark is 50% for MSc, 40% for Year in CS

Learning Outcomes

On successful completion of this module, you should be able to:

- Demonstrate knowledge of the fundamentals of computer hardware and architectures
- Explain the relation between high level object-oriented code and low level code execution.
- Explain and apply basic principles for reasoning about high level object-oriented code.
- Reflect on the significance of computer science in other disciplines.

Outline Syllabus

- Representing numbers in computers
- Organising principles of computer hardware
- Interaction between software and hardware
- Java and its relationship to the hardware
- Reasoning about computer programs
  - Invariants (instance and loop)
- Recursion

What are computers made of?

- A collection of “switches” called transistors
- Can either be “on” or “off”, corresponding to a particular electrical state (conducting or not).
- Forces “binary” representation
  - “off” $\rightarrow 0$
  - “on” $\rightarrow 1$
- All data must be represented as sequences of ones and zeros – binary digits – **bits**
  - As must the computer's “instructions”

Representing Text

- Every text character represented as a “binary string”
  - A: 1000001
  - B: 1000010
  - C: 1000011
- Words represented by concatenating strings
- Numbers are more subtle
  - Need to worry about arithmetic

Revisiting Decimal Numbers

- Understand decimal $\rightarrow$ understand binary

$$7265$$

$$7000+200+60+5$$

$$7x1000 + 2x100 + 6x10 + 5x1$$

$$7x10^3 + 2x10^2 + 6x10^1 + 5x10^0$$

- Character position denotes power of **ten**
- **Ten** possible symbols at each position (0–9)
  - Larger values “transfer” into next column left

Whole numbers in Binary

- Position denotes powers of **two**
- **Two** possible symbols at each position: {0,1}

$$101011$$

$$1 \text{x} 2^5 + 0 \text{x} 2^4 + 1 \text{x} 2^3 + 0 \text{x} 2^2 + 1 \text{x} 2^1 + 1 \text{x} 2^0$$

$$1 \text{x} 32 + 0 \text{x} 16 + 1 \text{x} 8 + 0 \text{x} 4 + 1 \text{x} 2 + 1 \text{x} 1$$

$$43$$

- 11001?
- 00110?
- 10101?
- With N “bits”, can represent $0 \rightarrow 2^N$-1

Hexadecimal

- Binary numbers are too long for us to remember
- Frequently use *hexadecimal* (base 16) instead
- Shorter, easier to remember, maps nicely onto binary
- 16 symbols {0-9, A-F}
- Each group of four bits maps to one hex character

1010 1001 0101 1100

A 9 5 C

- Frequently used to represent image colours
- Try it:
  - 0101 1111 0100 1011

Decimal to Binary

- What is 37 in binary?
- Odd number – right-most digit **must** be one
- Divide by two – remainder gives us right-most, or **least significant** bit.
- Apply this sequentially:

37/2 = 18r**1**

18/2 = 9r**0**

9/2 = 4r**1**

4/2 = 2r**0**

2/2 = 1r**0**

1/2 = 0r**1**

so $37_{10} = 100101_2$

- 24?
- 57?

Extending to the (positive) real numbers

- We can do 0, 1, 2, 3, … , 43, … , 12345, … etc
  - 3.14159? 2.71828?
- What comes after the “decimal” (radix) point?
- Negative powers of ten
  - $0.1 = 1/10 = 10^{-1}$
  - $0.01 = 1/100 = 1/(10^{2}) = 10^{-2}$

$$3.14$$

$$3x1 + 1x0.1 + 4x0.01$$

$$3x10^{0} + 1x10^{-1} + 4x10^{-2}$$

Fixed Point Binary

- Allocate subset of bits to integer part, and the remainder to the non-integer part. For example, 4+4 bits:

1101.0101

$$1x2^{3} + 1x2^{2} + 1x2^{0} + 1x2^{-2} + 1x2^{-4}$$

$$8 + 4 + 1 + 0.25 + 0.0625$$

11.3125

- 101.110?
- 010.001?

Fixed Point Decimal to Binary

- Integer part convert as before (repeated division by 2)
- Non-integer part follows opposite process
- Repeated multiplication by 2, keeping **integer** part

0.537 x 2 = **1**.074

0.074 x 2 = **0**.148

0.148 x 2 = **0**.296

0.296 x 2 = **0**.592

0.592 x 2 = **1**.184

0.184 x 2 = **0**.368

So $0.537_{10} = 0.100010_2$

- Or is it?.... try the following:
- 0.625?
- 0.512?

Numerical Precision

- Fixed point is convenient and intuitive but has two problems
- Numerical precision
  - only values that are an integer multiple of the smallest power of two can be represented exactly
- Numerical range
  - Increased precision of non-integer part is at the expense of numerical range
- Next lecture we will look at so-called floating point representations

Summary

- Representing numbers in the computer
  - Whole numbers in binary and hexadecimal notation
  - Positive real numbers in fixed-point binary
- Next time:
  - Negative numbers
  - Arithmetic
  - Floating point