Mathematics for Computing 2016-2017

Lecture 1:
Course Introduction and Numerical Representation

Dr Andrew Purkiss
The Francis Crick Institute
or
Dr Oded Lachish, Birkbeck, University of London

Topics 2016-17

- Number Representation
- Logarithms
- Logic
- Set Theory
- Relations & Functions
- Graph Theory

Assessment

- In Class Test (Partway through term, 31/10)
  (20% of marks)
- ‘Homework’ (3 parts for 10% of marks)
- Two hour unseen examination in May/June 2017
  (70% of marks)

Lecture / tutorial plans

- Lecture every week 18:00 for 18:10 start. 1 – 2½ hours (with break)
- Tutorials (problems and answers) one week in two (~1½ hours)
- Compulsory In-Class Test, October 31st
- Lecture Notes etc. will appear on Moodle
- Class split in two rooms

Provisional Timetable

Date	Week	Lecture Topic / Tutorial
3/10/16	1	Introduction and Numerical Rep
10/10/16	2	Logarithms & Indices / Tutorial 1 (Number Rep / Indices)
17/10/16	3	Logic 1
24/10/16	4	Logic 2 / Tutorial 2 (Logs & Logic)
31/10/16	5	In Class Test
7/11/16	6	Set Theory 1
14/11/16	7	Set Theory 2 / Tutorial 3 (Sets 1)
21/11/16	8	Set Theory 3 (Relations / Functions 1)
28/11/16	9	Set Theory 4 / Tutorial 4 (Sets 2)
5/12/16	10	Graph Theory 1
12/12/16	11	Graph Theory 2 / Tutorial 5 (Graph Theory)

Course Textbook

- Schaum's Outlines Series
  Essential Computer Mathematics
- Author: Seymour Lipschutz
  ISBN 0-07-037990-4

Maths Support

- http://www.bbk.ac.uk/business/current-students/learning-co-ordinators/eva-szatmari
- See separate powerpoint file.

Lecture 1

- Rule 1

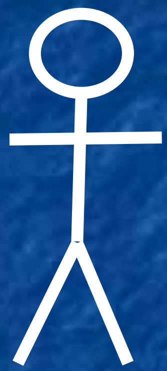
≠

Communication is not easy,

How do you tell a computer what to do?

Welcome

- Rule 1

 $\neq$ 

- We want to get the computer to do <u>NEW</u> complicated things
- We start by learning the basics of its language, Numerical Representation, Logic …

Memory for numbers

- We don't know how our memory stores numbers
- We know how a computer does (we designed it)
- Full glass is 1, empty is 0

- Great, we know how to store 1 and 0 in the computer memory
- How do we store 0,1,2,3?
- We use two cups!

The numbers in the way the computer sees them. Base 2 (binary).

The numbers in the way we are used to see them. Base 10 (decimal).

0 = 0 0 =

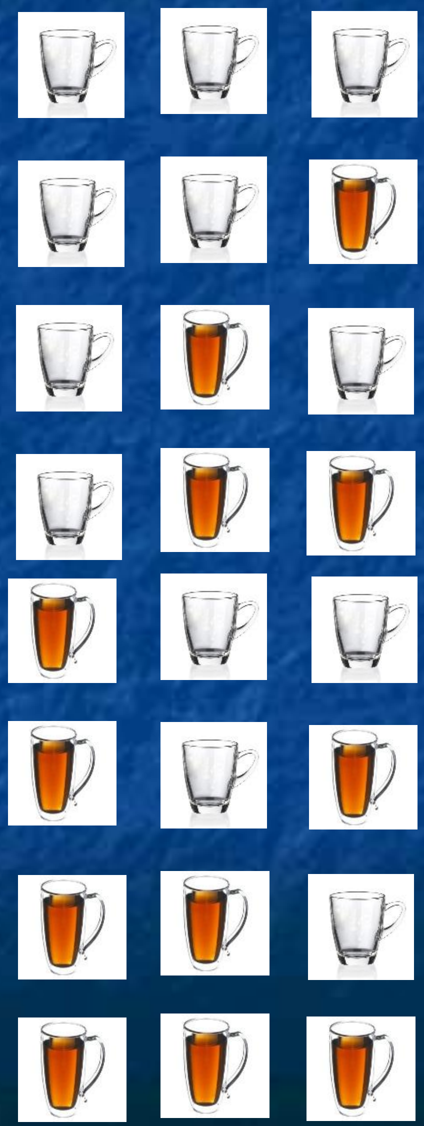
1 = 0 1 =

2 = 1 0 =

3 = 1 1 =

- If we want extra numbers we add an extra cup!
- Each cup we add doubles the number of values we can store

0	=	0	0	0	=
1	=	0	0	1	=
2	=	0	1	0	=
3	=	0	1	1	=
4	=	1	0	0	=
5	=	1	0	1	=
6	=	1	1	0	=
7	=	1	1	1	=



- We don’t need the cups now.
- Let’s understand how this works
- We shall look for repetitive patterns and see what they mean

Same

$$0 = 0\,0 = 0 + 0$$

$$1 = 0\,1 = 0 + 1$$

$$2 = 1\,0 = 2 + 0$$

$$3 = 1\,1 = 2 + 1$$

- The repetitive pattern here tells us whether the number is odd or even (add 0 or 1)

- The repetitive pattern here tells us whether to add 0 or 2

Same

$$0 = 000 = 0 + 0 + 0$$
$$1 = 001 = 0 + 0 + 1$$
$$2 = 010 = 0 + 2 + 0$$
$$3 = 011 = 0 + 2 + 1$$
$$4 = 100 = 4 + 0 + 0$$
$$5 = 101 = 4 + 0 + 1$$
$$6 = 110 = 4 + 2 + 0$$
$$7 = 111 = 4 + 2 + 1$$

Convert from Binary to Decimal

- When we translate from the binary base (base 2) the decimal base (base 10 – ten fingers)

- The first binary digit tells us whether to add 1
- The second binary digit tells us whether to add 2
- The third binary digit tells us whether to add 4
- The fourth binary digit tells us whether to add ??

1 0 1 1

Convert from Binary to Decimal

- When we translate from the binary base to the decimal base
- The first binary digit tells us whether to add 1
- Every digit afterwards tells us whether to add exactly two times as much a the previous digit

Lets try this out

1 0 1 1 1 0 1 =

1*64+0*32+1*16+1*8+1*4+0*2+1*1 =

<u>83</u>

The binary system (computer)

- The way the computer stores numbers
- Base 2
- Digits 0 and 1
- Example:

  11011011_2

  ↑ ↑

  msd lsd

(most significant digit) (least significant digit)

The decimal system (ours)

- Probably because we started counting with our fingers
- Base 10
- Digits 0,1,2,3,4,5,6,7,8,9
- Example:
  76413219_{10}
  ↑ ↑
  msd lsd

Significant Figures

- Significant Figures:
  Important in science for precision of measurements.
- All non-zero digits are significant
- Leading zeros are not significant
- e.g. $\pi = 3.14$ (to 3 s.f.) = 3.142 (to 4 s.f.) = 3.1416 (to 5 s.f.)

Some binary numbers!!!

Binary	Decimal	Binary	Decimal	Binary	Decimal
0_2	0	111_2	7	1110_2	14
1_2	1	1000_2	8	1111_2	15
10_2	2	1001_2	9	10000_2	16
11_2	3	1010_2	10	10001_2	17
100_2	4	1011_2	11	10010_2	18
101_2	5	1100_2	12	10011_2	19
110_2	6	1101_2	13	10100_2	20

Convert from Binary to Decimal

Lets make this more mathematical,

We now use powers of 2 to represent 1,2,4,8,…

$$1 \quad 0 \quad 1 \quad 1 \quad 1 \quad 0 \quad 1 =$$

$$1*64+0*32+1*16+1*8+1*4+0*2+1*1 =$$

$$1*2^6+0*2^5+1*2^4+1*2^3+1*2^2+0*2^1+1*2^0 =$$

$$\underline{93}_{10}$$

Note that the power is the ***index of the digit***, when the indices start from 0 (first index is 0)

(digit with index 6 corresponds to 2^6)

Convert from Binary to Decimal

Example of how to use what we learned to convert from binary to decimal

Digit index	6	5	4	3	2	1	0	Total
Binary Number	1	1	0	1	1	0	1	
Power of 2 To Add	$1*2^6$	$1*2^5$	$0*2^4$	$1*2^3$	$1*2^2$	$0*2^1$	$1*2^0$	
Actual values	**64**	**32**	**0**	**8**	**4**	**0**	**1**	**109**

$1101101_2 = 1*2^6+1*2^5+0*2^4+1*2^3+1*2^2+0*2^1+1*2^0$

$= 64+32+0+8+4+0+1 = 109_{10}$

Idea for Converting Decimal to Binary

- Digit at position 0 is easy.
- It is 1 if the number is even and 0 otherwise
- Why?
- In a binary number only the least significant digit (2^0=1)

Digit index	6	5	4	3	2	1	0	Total
Binary Number								
Power of 2 To Add	$*2^6$	$*2^5$	$*2^4$	$*2^3$	$*2^2$	$*2^1$	$*2^0$	
Actual values								

Convert from Decimal to Binary

Number	Remainder when dividing by 2
43	1
21	1
10	0
5	1
2	0
1	1

Divide by 2 and remember remainder

Number is given from bottom to top

1	0	1	0	1	1

What Happens when we Convert from Decimal to Binary

Divide by 2 and remember remainder

Number	5	4	3	2	1	0
43	1	0	1	0	1	1
21		1	0	1	0	1
10			1	0	1	0
5				1	0	1
2					1	0
1						1

1
1
0
1
0
1

Same

Number is given from bottom to top

101011_2

The empty cells are 0

Decimal to Binary conversion Algorithmically: Natural Numbers

- 1. Input *n* (natural no.)
  2. **Repeat**
     2.1. Output *n* mod 2
     2.2. $n \leftarrow n$ div 2
     until $n = 0$

Example: 11_{10}

Step	n	output
1	11	-
2.1	11	1
2.2	5	-
2.1	5	1
2.2	2	-
2.1	2	0
2.2	1	-
2.1	1	1
2.2	0	-

Number is given from bottom to top

Convert from Decimal to Binary

Divide by 2 and remember remainder

Number	Remainder when dividing by 2
56	0
28	0
14	0
7	1
3	1
1	1

Number is given from bottom to top

1	1	1	0	0	0

Numbers we can already represent

- Natural numbers: 1, 2, 3, 4, …
- <u>Alternative versions of the number six</u>

  Decimal: 6

  Alphabetically: six

  Roman: VI

  Tallying: 

What’s still missing

- Fractional numbers (real numbers)
- Versions of one and a quarter

  Mixed number: 1¼,

  Improper fraction: 5/4,

  Decimal: 1.25

Decimal numbers (base 10)

- String of digits
- - symbol for negative numbers
- Decimal point
- A positional number system, with the index giving the 'value' of each position.
  Example:
  $3583.102 = 3 \times 10^3 + 5 \times 10^2 + 8 \times 10^1 + 3 \times 10^0 + 1 \times 10^{-1} + 0 \times 10^{-2} + 2 \times 10^{-3}$

Representing Decimal numbers in Binary

- We can use two binary numbers to represent a fraction by letting the first number be the enumerator and the other be denominator

Problem: we want operation such as addition and subtraction to execute fast. This representation is not optimal.

Representing Fractions in Binary

- Use a decimal point like in decimal numbers
- There are two binary numbers the first is the number before the (radix) point and the other after the point

Representing decimal numbers in binary

Digit Position	2	1	0	-1	-2	-3	Total
Binary Number	1	1	0	1	0	1	
Power of 2 To Add	$1*2^2$	$1*2^1$	$0*2^0$	$1*2^{-1}$	$1*2^{-2}$	$0*2^{-3}$	
Power of 2 in actual numebers	1*4	1*2	0*1				
Actual values	4	2	0	0.5	0	0.125	6.625

$110.101_2 = 1*2^2+1*2^1+0*2^0+1*2^{-1}+1*2^{-2}+0*2^{-3} = 4+2+0+0.5+0.125 = 6.625$

Convert fractional part from Decimal to Binary

- To convert the decimal part:

.	1	1	0	1

Multiply by 2, remove and remember the integer part, which can be either 0 or 1.

(Continue until we reach 1.0)

Number	Number*2	Integer part when multiplying by 2
0.8125	1.625	1
0.625	1.25	1
0.25	0.5	0
0.5	1.0	1

Number is given from top to bottom, because this time we multiplied

Negative numbers

- First bit (MSB) is the sign bit
  - If it is *0* the number is positive
  - If it is *1* the number is negative

Goal when definition was chosen:

1. Maximize use of memory
2. Make computation easy

Negative Numbers – Calculate two’s Complement

- The generate two’s complement
  Write out the positive version of number,
  Write complement of each bit
  (0 becomes 1 and 1 becomes 0)
  Add 1
  The result is the two’s complement and the negative version of the number

Negative Numbers – Two's Complement (examples)

- 3bit 8bit

  011 3_{10} 00011101 29_{10} number

  100 11100010 complement

  +

  001 00000001 +1

  === =========

  101 -3_{10} 11100011 -29_{10} 2's complement

Negative numbers – Two's Complement(3 bits)

- First bit (MSB) is the sign bit
  - If it is *0* the number is positive
  - If it is *1* the number is negative

Goal when definition was chosen:

1. Maximize use of memory
2. Make computation easy

- None of the numbers repeat themselves – memory efficiency
- If you add the binary numbers the sum up properly

3 bit number			2's comp.
0	0	1	1
	+		+
1	1	0	-2
	=		=
1	1	1	-1

3 bit number			Two's complement value
sign	bits		
0	1	1	3
0	1	0	2
0	0	1	1
0	0	0	0
1	1	1	-1
1	1	0	-2
1	0	1	-3
1	0	0	-4

- Table of two's complement for 3 bit numbers.

Negative numbers – Two's Complement (4 bits)

4 bits number				2's comp.
0	1	0	0	4
+				+
1	1	0	1	-3
=				=
0	0	0	1	1

- Binary addition is done in the same way as decimal, using carry
- The last carry here doesn't matter
- When adding large numbers this has a wraparound (computers are equipped to deal with this)

4 bits number				Two's complement value
sign	bits			
0	1	1	1	7
0	1	1	0	6
0	1	0	1	5
0	1	0	0	4
0	0	1	1	3
0	0	1	0	2
0	0	0	1	1
0	0	0	0	0
1	1	1	1	-1
1	1	1	0	-2
1	1	0	1	-3
1	1	0	0	-4
1	0	1	1	-5
1	0	1	0	-6
1	0	0	1	-7
1	0	0	0	-8

Computer representation

- Fixed length
- Integers
- Real
- Sign

Bits, bytes, words

- Bit: a single binary digit
- Byte: eight bits
- Word: Depends!!!
- Long Word: two words

Integers

- A two byte integer
- 16 bits
- 2^{16} possibilities $\rightarrow$ 65536
- $-32768 \leq n \leq 32767$ or $0 \leq n \leq 65535$

Signed integers

- First bit is sign bit. $n \geq 0$, 0; $n < 0$, 1
- For $n \geq 0$, 15 bits are binary n
- For $n < 0$, 15 bits are binary $(n + 32768)$
- Example: -6772_{10} (-001101001110100_2)

$$
\begin{array}{r}
1000000000000000_2 \\
-001101001110100_2 \\
\hline
110010110001100_2
\end{array}
$$

Real numbers

- ‘Human’ form: 4563.2835
- Exponential form: 0.45632835×10^4

- General form: $\pm m \times b^e$
- Normalised binary exponential form: $\pm m \times 2^e$

Real numbers

- Conversion from Human to Exponential and back

655.54 = 0. 65554 * 10^3

0.000545346 = 0. 545346 *10^{-3}

0.523432 * 10^5 = 52343.2

0.7983476 * 10^{-4} = 0.00007983476

If the exponent is positive then it is the number of digits after the decimal point (first must be non zero). If it is negative its absolute value is the number of digits after the decimal point.
You can use this to do both conversions

Real numbers 2

For a 32 bit real number

- Sign, 1 bit
- Significand, 23 bits
- Exponent, 8 bits

Types of numbers

- Integers: …, -3, -2, -1, 0, 1, 2, 3, …
- Rational numbers: m/n, where m and n are integers and $n \neq 0$.
  Examples: ½, 5/3, ¼ = 0.25 1/3 = 0.3333…
- Irrational numbers,
  examples: $\sqrt{2} \approx 1.414$, $\pi \approx 22/7 \approx 3.14159$
  $e \approx 2.718$.

Other representations

- Base Index form
  Number = $\text{base}^{\text{index}}$
  e.g. $100 = 10^2$
- Percentage form
  Percentage = number/100
  e.g. 45% = 45/100 = 0.45
  20% = 20/100 = 0.2
  110% = 110/100 = 1.1

Other number systems

- Bases can be any natural number except 1.
- Common examples are :
  Binary (base 2)
  Octal (base 8)
  Hexadecimal (base 16)
- We'll show what to do with base 5 and 7 and then deal with the octal and hexadecimal bases

Convert from Decimal to Base 7

Number	3	2	1	0
779	2	1	6	2
111		2	1	6
15			2	1
2				2

2
6
1
2

Divide by 7 and remember remainder

Same

Number is given from bottom to top

2162_7

Convert from Base 7 to Decimal

Digit index	3	2	1	0	Total
Base 7 Number	2	1	6	2	
Power of 2 To Add	$2*7^3$	$1*7^2$	$6*7^1$	$2*7^0$	
	2*343	1*49	6*7	2*1	
Actual values	**686**	**49**	**42**	**2**	**779**

$2162_7 = 2*7^3+1*7^2+6*7^1+2*7^0 = 686+49+42+2=779_{10}$

Convert from Decimal to Base 5 and back

Divide by 5 and remember remainder

Number	Remainder when dividing by 5
<u>**996**</u>	1
199	4
39	4
7	2
1	1

1	2	4	4	1

Digit Position	4	3	2	1	0	Total
Binary Number	1	2	4	4	1	
Power of 5 to Add	$1*5^4$	$2*5^3$	$4*5^2$	$4*5^1$	$1*5^0$	
Actual values	**625**	**250**	**100**	**20**	**1**	<u>**996**</u>

$13441_5 = 1*5^4+2*5^3+4*5^2+4*5^1+1*5^0= 625+250+100+20+1=\underline{996}_{10}$

Octal

- Base eight
- Digits 0,1,2,3,4,5,6,7
- Example: $12_{10} = 14_8 = 1100_2$
- 10011011110_2 Binary

2 3 3 6 $= 2336_8$ Octal

Conversion from binary to octal

Convert from Binary to Octal and back

Index binary	12	11	10	9	8	7	6	5	4	3	2	1	0
Binary	1	1	1	1	1	0	0	0	1	1	1	0	1
Base 8	1		7				4			3		5	
Index octal	4		3				2			1		0	

$1111100011101_2 = 17435_8$

- When converting from binary to octal every three binary digits are converted to one octal digit as in the table
- When converting from octal to binary every octal digit is converted to three binary digits as in the table
- The actual conversion can be done using the conversion table

Conversion table			
binary			octal
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

Hexadecimal

- Base sixteen
- Digits 0,1,2,3,4,5,6,7,8,9,A(10), B(11), C(12),D(13),E(14),F(15).
- Example $B3_{16} = 179_{10} = 10110011_2$
- 11010101_2 Binary

D 5 Hexadecimal

Conversion from binary to hexadecimal

Convert from Binary to Hexadecimal and back

Index binary	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	1	1	1	1	0	0	0	1	1	1	0	1
	1	F				1				D			
Index Hexadecimal	3	2				1				0			

$111110001110 1_2 = 1F1D_{16}$

- When converting from binary to hexadecimal every four binary digits are converted to one hexadecimal digit as in the table
- When converting from hexadecimal to binary every hexadecimal digit is converted to four binary digits as in the table
- The actual conversion can be done using the conversion table which can be written down in less than a minute

binary				Hexa decimal
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	A
1	0	1	1	B
1	1	0	0	C
1	1	0	1	D
1	1	1	0	E
1	1	1	1	F

Writing down the hexadecimal conversion table

- Create the table with a ruler need to be 5 columns and 16 rows
- The binary LSB column is 01 repeated from top to bottom
- The second binary index is 0011 repeated from top to bottom
- The patterns should be obvious for the other digits
- For the hexadecimal just start with 0 at the top and continue in increments of 1 until 9 is reached, then proceed with the letters of the alphabet

binary				Hexa decimal
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	A
1	0	1	1	B
1	1	0	0	C
1	1	0	1	D
1	1	1	0	E
1	1	1	1	F

Extra Slides

$$
\begin{array}{rcccccccc}
 & 1 & 1 & 1 & & 1 & 1 & 1 & \\
 & & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\
+ & & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\
\hline
 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0
\end{array}
$$

May have an extra 0, but that doesn't matter

$1_2+1_2= 10_2$ → 0 with carry 1

$1_2+1_2+1_2= 10_2$ → 1 with carry 1

All other options don't have carry

End of Lecture

Extra Slides

- The following slides present the same information already appearing in other slides, in a different manner.

Decimal to Binary conversion 1: Mathematical Operations

- n div 2 is the quotient.
- n mod 2 is the remainder.

- For example:
  14 div 2 = 7, 14 mod 2 = 0
  17 div 2 = 8, 17 mod 2 = 1

Decimal to Binary conversion 2: Natural Numbers

- 1. Input *n* (natural no.)
  2. **Repeat**
     2.1. Output *n* mod 2
     2.2. $n \leftarrow n$ div 2
  until $n = 0$

Example: 11_{10}

Step	n	output
1	11	-
2.1	11	1
2.2	5	-
2.1	5	1
2.2	2	-
2.1	2	0
2.2	1	-
2.1	1	1
2.2	0	-

Decimal to Binary conversion 3: Fractional Numbers

- 1. Input n
  2. **Repeat**
     2.1. $m \leftarrow 2n$
     2.2. Output $\lfloor m \rfloor$
     2.3. $n \leftarrow \text{frac}(m)$
     until $n = 0$
- $\lfloor m \rfloor$ is the integer part of m
- frac(m) is the fraction part.

Example: 0.375_{10}

Step	m	n	output
1	-	0.375	-
2.1	0.75	0.375	-
2.2	0.75	0.375	0
2.3	0.75	0.75	-
2.1	1.5	0.75	-
2.2	1.5	0.75	1
2.3	1.5	0.5	-
2.1	1	0.5	-
2.2	1	0.5	1
2.3	1	0	-

Some hexadecimal (and binary) numbers!!!

Binary	Decimal	Hex	Binary	Decimal	Hex
0000_2	0	0_{16}	1000_2	8	8_{16}
0001_2	1	1_{16}	1001_2	9	9_{16}
0010_2	2	2_{16}	1010_2	10	A_{16}
0011_2	3	3_{16}	1011_2	11	B_{16}
0100_2	4	4_{16}	1100_2	12	C_{16}
0101_2	5	5_{16}	1101_2	13	D_{16}
0110_2	6	6_{16}	1110_2	14	E_{16}
0111_2	7	7_{16}	1111_2	15	F_{16}

End