COMP10002 Workshop Week 12

Outlook:

1	Representation of integers and floats
2	Ex. 13.1 and 13.2
3	Working with text files Design & implement 11.3
4	Design 9.3 Implement 9.3 and/or 9.11
5	Q&A

Numeral Systems

214.39	2	1	1	.	3	9
Position	2	1	0	Dot	-1	-2
Value	2×10^2	1×10^1	4×10^0		3×10^{-1}	9×10^{-2}

2

→ *base* = 10 (decimal)

Other bases: binary (base= 2), octal (base= 8), hexadecimal (16)

$21.3_{(10)} = 2 \times 10^1 + 1 \times 10^0 + 3 \times 10^{-1}$

$1001_{(2)} = 1x2^3 + 0x2^2 + 0x2^1 + 1x2^0 = 9_{(10)}$

$5B_{(16)} = 5x16^1 + 11x16^0 = 91_{(10)}$

Converting between bases 2 and 16 is easy!

Because 1 hexadecimal digit is equivalent to 4 binary digits.

Converting Binary → Decimal

Just expand using the base=2:

$$\ldots\ b_3\ \ b_2\ \ b_1\ b_0\ .\ b_{-1}\ \ b_{-2}\ \ldots\quad (2)$$

is $\ldots\ b_3 x 2^3 + b_2 x 2^2 + b_1 x\ 2^1 + b_0\ +\ b_{-1} x 2^{-1} + b_{-2} x 2^{-2} \ldots$

Examples: 1101 →

1.011 →

Practical advise: remember

128	64	32	16	8	4	2	1	0.5	0.25	0.125
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}

Converting Decimal → Binary: Method 1 (lectured)

It's better to convert the integer and the fractional parts separately. Remember and apply the power-of-two sequence:

	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	.	2^{-1}	2^{-2}	2^{-3}
	128	64	32	16	8	4	2	1		.5	.25	.125
13.625_{10} → 1101.101_2					8	4		1		.5		.125
109_{10} →		64	32		8	4		1				
19												
1.3125												

Decimal→Binary. Method 2A: Integer Part

Changing integer x to binary: Just divide x and the subsequent quotients by 2 until getting zero. The sequence of remainders, in reverse order of appearance, is the binary form of x.

Example: 23

operation	quotoion	remainder
23 :2	11	1
11:2	5	1
5:2	2	1
2:2	1	0
1:2	0	1

So: $23 = 10111_{(2)}$

Decimal→Binary. Method 2B: Fraction Part

Fraction: Multiply it, and subsequent fractions, by 2 until getting zero. Result= sequence of integer parts of results, in appearance order. Examples:

0.375				0.1		
operation	int	fraction		operation	int	fraction
.375 x 2	0	.75		.1 x 2	0	.2
.75 x 2	1	.5		.2 x 2	0	.4
.5 x 2	1	.0		.4 x 2	0	.8
				.8 x 2	1	.6
				.6 x 2	1	.2

So: $0.375 = 0.011_{(2)}$ $0.1 = 0.00011(0011)_{(2)}$

Now try convert: 6.875 to binary

Converting Decimal->Binary

$7_{(10)} = ?_{(2)}$

$130_{(10)} =$

$6.375_{(10)} =$

$9.2_{(10)} =$

Representation of integers (in computers) using w bits

- Note that we use a fixed amount of bits `w`
- Make difference between `unsigned` and `signed` integers

- To represent `unsigned` integers : Just convert to binary, then add `0` to the front until having `w` bits. Largest value: $2^{w}-1$

- To represent `signed` integers `x`:
  - Positive integers: a `0-bit`, followed by the binary representation of `x` in `w-1` bits. Largest value: $2^{w-1}-1$
  - Negative integers: using twos-complement of `x` in `w` bit. Smallest value: -2^{w-1}. The first bit will always be `1`.

Finding twos-complement representation in w bits for negative numbers in 3 step

Suppose that we need to find the twos-complement representation of −`x`, *where* `x` *is positive, in* `w=16` *bits. It can be done easily in 3 steps:*

1) *Write binary representation of* |`x`| *in* `w` *bits*
2) *Find the rightmost* `one-bit`
3) *Inverse (ie. flip 1 to 0, 0 to 1) all bits on the left of that rightmost* `one-bit`

find the 2-comp repr of **−40**	**Bit sequence**			
1) bin repr of 40 in 16 bits	0000	0000	0010	1000
2) find the `rightmost 1`	0000	0000	0010	1000
3) inverse its left	1111	1111	1101	1000

Examples

w= 8, then 2^{w-1}= 128, and only numbers from -128 to +127 can be represented.

Decimal				Representation		Hexa equivalent
2	→	10	→	0000 0010	=	02
93	→	1011101	→	0101 1101	=	5D
-93	→		→	1010 0011	=	A3
92	→					
-92	→					
130	→					
-128	→					

Ex: 2-complement representation in w=16 bits

Q: What are 17, -17, 34, and -34 as 16-bit twos-complement binary numbers, when written as (a) binary digits, and (b) hexadecimal digits?

Ex. 13.1

Suppose that a computer uses w= 6 bits to represent integers. Calculate the two-complement representations for 0, 4, 19, -1, -8, and -31; Verify that 19-8 = 11;

Quiz

What is the binary form of $255_{(10)}$?

A. `1000 0000`

B. `1111 1111`

C. `1 0000 0000`

D. `1 1111 1111`

Quiz

What is the binary form of $13.625_{(10)}$?

A. 1101.101

B. 1011.101

C. 1101.11

D. 1011.11

Quiz

What is your tutor's name?

A. Alistair

B. Anh

C. Artem

D. Ahn

5 min break for qoct : quality of casual teaching survey

- Please do the survey right now
- To access the survey, just google:

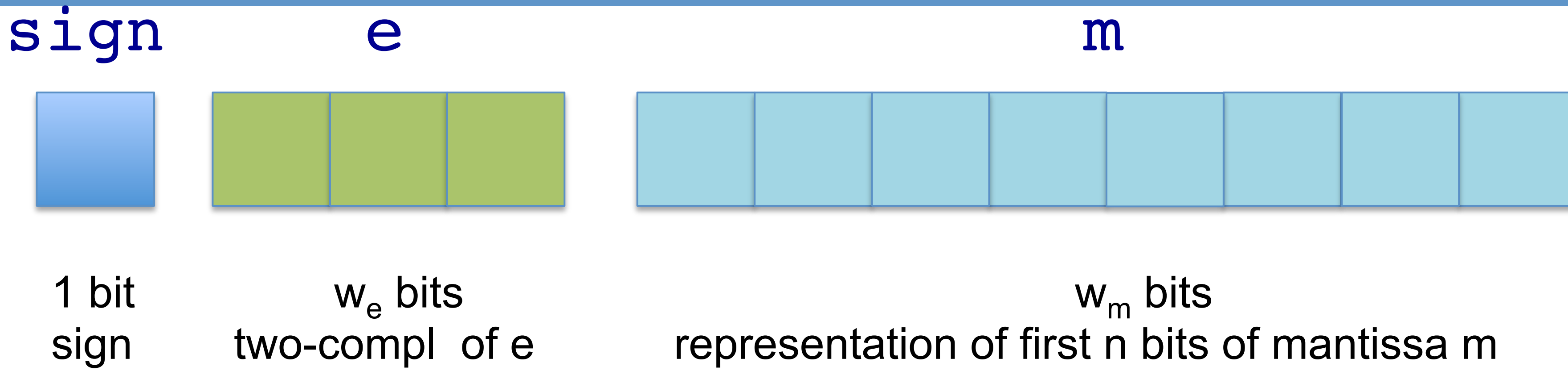
  qoct

Representation of floats

We learnt 2 different formats:

- one as described in `lec09.pdf` and in the text book
- another is an `IEEE standard`, which is:
  - employed in most of modern computers,
  - demonstrated in the lecture, and
  - you can find/experiment with using the program `floatbits.c`.

Representation of floats (as described in lec09.pdf)



Convert $/x/$ to binary form, and transform so that:

$$|x| = \ 0.b_0b_1b_2\ldots \ \times\ 2^e \quad \textit{where } b_0 = 1$$

e is called exponent, $m = b_0b_1b_2\ldots$ is called mantissa

Three components: `sign, e, m` are represented as in the diagram.

Ex. 13.2

Suppose w_s=1, w_e=3, w_m=12, what's the representation of 2.0, -2.5, 7.875 ?

Representation of 32-bit float: (IEEE 754, as in floatbits.c)

w_s=1, w_e=8, w_m=23 $|x|$= $1.b_1b_2...$ x 2^e [Note the *different transformation* of $|x|$]

----------------------------------Bit Position --

1 2 9 10 32

Sign ----------------------------------- --

e+127 b_1 b_2 . . . $b_{22}b_{23}$

That is:

- The sign bit is 0 or 1 as in the previous case
- e is represented in excess-127 format, which means e is represented as the unsigned value e+127 in w_e bits
- The first bit of the mantissa is omitted from the representation, and the mantissa is just $b_1b_2...b_{23}$

Note: Valid e is -126 → +127, corresponding to values 1 →254 . Value 0 used for representing 0.0, value 255 used to represent infinity. And, zero is all 32 zero-bit, and infinity is all 32 one-bit.

Representation of 32-bit float: (IEEE 754, as in floatbits.c)

w_s=1, w_e=8, w_m=23 $\qquad |x| = 1.b_1b_2\ldots \times 2^e$

Example: x= 3.5

In binary: x= 11.1= 1.11 x 2^1

→ sign bit: 0

→ e=1 is represented as e+127= 128 in 8 bits

→ e is represented as 1000 0000

→ mantissa: 110 0000 0000 0000 0000 0000

→ Final representation:

0100 0000 0110 0000 0000 0000 0000 0000

or 4 0 6 0 0 0 0 0 $_{(16)}$

Working with text files. Ex 11.3

The Unix `tee` command writes its `stdin` through to `stdout` in the same way that the `cat` command does. But it also creates an additional copy of the file into each of the filenames listed in the command-line when it is executed.

Implement a simple version of this command.

Hint: you will need an array of files all opened for writing.

Design 9.3

Write a program that deals four random five-card poker hands from a standard 52-card desk:

```
$ ./poker
player 1:   3-S, Ac-C, Qu-D, 4-H, Qu-H
```

. . . (all 4 players)

Then, modify your program to allow you to estimate the probability that a player obtains a simple pair (2 cards with the same face value) in their initial hand. Compute that probability using 40,000 hands dealt from 10,000 shuffled desks.

. . .

Design 9.3

Implement: 11.2 ,   9.3 OR   9.11

Possible follow-up subjects next year:

sem1: `comp20007` – Design of Algorithms

sem2: `comp20003` – Algorithms & Data Structures

Good Luck!