COMP20005 Workshop Week 9

Preparation:

- have LMS and programming tools ready
- have papers & pens ready for taking notes

1	**Discussion 1: Representation of integer** Class Exercise: what are 17, -17, 34, and -34 as 16-bit twos-complement binary numbers, when written as (a) binary digits, and (b) hexadecimal digits?
2	**Discussion 2: Representation of float** Class Exercise: using the same representation as on page 21 of the numericA.pdf slides, what are the floating point representations of -3.125, 17.5, and 0.09375, when written as (a) binary digits, and (b) hexadecimal digits?
LAB	Quiz 2 This Friday! Your choice: • Q&A for quiz 2 • Array exercises: Consolidate your understanding of arrays, by completing any of Exercises 7.1, 7.2, 7.3, 7.4, 7.6, 7.7, 7.8, and 7.15 that you have not yet looked at. *Arrays, and being able to manipulate data stored in them, are the absolute backbone of this subject*. • [If you want a challenge] Write a program that outputs the bit-patterns associated with 64-bit double values.

Looking Ahead

Quiz 2: Will be held 1pm Melbourne time on Friday 7 May. It will cover Chapters 6 and 7 of the textbook: scope, pointers, arrays, two-dimensional arrays, arrays and functions, and strings, including arrays of strings and arrays of pointers to strings. A Practice Quiz is available, *you can try 5 times*.

Ask Questions in the 2nd half of the today's workshop

ARRAY EXERCISES ALISTAIR ADVISED IN LMS UNDER

Arrays, and being able to manipulate data stored in them, are the absolute backbone of this subject.

General: 7.1 (W07)
Sorting: 7.4 (W07) , 7.2, 7.3, 7.6 (W08)
Searching: 7.7, 7.8 (W7X)
Strings: 7.15

Notes: there are many other exercises in W07, W7X, W08, W8X

Numeral Systems

211.39	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}

→ *base* = 10 (decimal)

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

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

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

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

Note: hexadecimal uses 6 additional digits: A, B, C, D, E, F with the values 10-15

Converting between bases 2 and 16 is easy!

- Because 1 hexadecimal digit is equivalent to 4 binary digits. For examples $5_{(16)} \Leftrightarrow 0101(2)$, $C_{(16)} \Leftrightarrow 1100_{(2)}$

Class Exercises:

110101101 $\Leftrightarrow$? $_{(2)}$

AFB5$_{(16)}$ $\Leftrightarrow$? $_{(16)}$

111110011011 $\Leftrightarrow$? $_{(16)}$

3CD$_{(16)}$ $\Leftrightarrow$? $_{(2)}$

Converting Binary → Decimal

Just expand using the base=2:

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

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

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}

Decimal→Binary: Integer Part

Changing integer x to binary (when the solution not obvious):

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: $23 = 10111_{(2)}$ $11 = {}_{(2)}$ $46 = {}_{(2)}$

Decimal→Binary: 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: $0.375 = 0.011_{(2)}$ $0.1 = 0.00011(0011)_{(2)}$

Class Exercise: Converting Decimal->Binary

- $93.875_{(10)}$ =

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 (`unsigned int` and `int` in C)

`unsigned` integers :

- Range: 0.. 2^w-1
- Representation: Just convert to binary, then add `0` to the front to have enough `w` bits.

Example:

if `w`=16, then 6 (which is $110_{(2)}$) will be represented as:

`0000 0000 0000 0110` (or `0006` in hexadecimal format)

Representation of integers (in computers) using w bits

- `signed` integer range: -2^{w-1} to $2^{w-1}-1$

  so: -128 to $+127$ if w= 8

  -2^{31} to $+2^{31}-1$ if w= 32

  Note: $2^{31} = (2^{10})^{3.1} = 1024^{3.1} \approx (10^3)^{3.1} = 10^{9.3}$ ≈ 2x10^9

- To represent `signed` integers `x`:
  - Positive numbers: the binary representation of `x` in `w` bits (the same as in the `unsigned` format, but with smaller range). Note: in this case the first bit is always `0`
  - Negative numbers: using twos-complement of `x` in `w` bit. 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. Do it 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

Exercise: 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?

Decimal	twos-complement in binary digit	twos-complement in hexadecimal digits
17		
-17		
34		
-34		

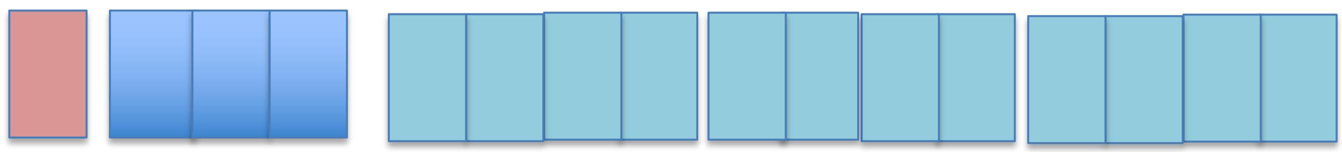
Representation of floats

We learnt 2 different formats:

- one as described in `numericA.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  (numericA.pdf p.22)`.

Representation of floats (as described in numericA.pdf)

- `sign` `e` `m`



1 bit w_e bits w_m bits
sign two-compl of e representation of first n bits of mantissa m

- Convert $|x|$ to binary form, and transform so that:
  $|x| = 0.b_0b_1b_2\ldots \times 2^e$ *where* $b_0 = 1$
- `e` is called exponent, $m = b_0b_1b_2\ldots$ is called mantissa
- x is represented as the triple (`sign, e, m`) as shown in the diagram.
  Note: $w_e = 3$, $w_m = 12$

Class/Group Exercise:

using the above representation, what are the floating point representations of `-3.125`, `17.5`, and `0.09375`, when written as (a) binary digits, and (b) hexadecimal digits?

Do it, and then check your answer.

Exercises

Class/Group Exercise: using the same representation as on page 21 of the numericA.pdf slides, what are the floating point representations of -3.125, 17.5, and 0.09375, when written as (a) binary digits, and (b) hexadecimal digits?

Recall that this representation use:

- 12 bits in total, including
- 1 sign bit
- $w_e = 3$ (in twos-complement format)
- $w_m = 12$

Do it, then check your answer:

-3.125 →

17.5 →

0.09375 →

LAB

Quiz 2: Will be held 1pm Melbourne time on Friday 7 May. It will cover Chapters 6 and 7 of the textbook: scope, pointers, arrays, two-dimensional arrays, arrays and functions, and strings, including arrays of strings and arrays of pointers to strings. A Practice Quiz is available, *you can try 5 times*.

To see **SIX** sample programming questions & solutions, Visit:
https://people.eng.unimelb.edu.au/ammoffat/teaching/20005/quiz2-longq-samp-soln.c.

Ask Questions NOW

Finish your 5 times of doing sample quiz NOW

IMPLENT ARRAY EXERCISES ALISTAIR ADVISED IN LMS:

General: 7.1 (W07)
Sorting: 7.4 (W07) , 7.2, 7.3, 7.6 (W08)
Searching: 7.7, 7.8 (W7X)
Strings: 7.15

Notes: there are many other exercises in W07, W7X, W08, W8X

Additional Material

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

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 $|x| = 1.b_1b_2\ldots \times 2^e$

- 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 $_{(16)}$