

## Computing Tutorial: Week 4

Binary

Tony Chung a.chung@lancaster.ac.uk

### Objectives Today

- Interpretation (5 min)
  - Bits and Bytes (10 mins)
  - Hexadecimal (15 mins)
  - Limits (5 mins)
  - Sign and Magnitude (15 mins)
  - Binary Addition (20 mins) Hopefully
- Today's slides are based on topics suggested by Andrew Scott.
  - **You can suggest topics to cover at any time over email or at the end of the class.**

### Interpretation

- What is stored here?
  - 1100100100000001111111011001001
- Discuss

### Interpretation (A)

- What is stored here?
  - 1100100100000001111111011001001
- Raw data has no meaning unless you give it context.
- There are several ways to give raw data meaning:
  - Define types.
    - Bit length.
  - Write a program.
    - Memory organisation.

### Bits and Bytes

- How do we represent the following decimal numbers in binary?
  - 1
  - 8
  - 256
  - 511
  - 22
- What is the difference between big endian and little endian?
- How do the above numbers appear in a 4-byte binary representation in little endian and big endian?

### Bits and Bytes (A)

- How do we represent the following decimal numbers in binary?
  - Write down the bit values, from right to left, by doubling from 1.
  - 1 1
  - 8 1000
  - 256 100000000
  - 511 111111111
  - 22 00010110
- What is the difference between big endian and little endian?
  - Big endian: Like in decimal. Most significant **byte** first.
  - Little endian: Least significant **byte** first.
- How do the above numbers appear in a 4-byte binary representation in little endian and big endian?
  - 1 Big-endian: 0,0,0,1 Little-endian: 1,0,0,0
  - 511 Big-endian: 0,0,1,11111111 Little-endian: 11111111,1,0,0

## Hexadecimal

- You have just seen that it is less ideal to convert from decimal to binary and vice versa.
- This is because base-10 digits do not map directly to base-2 digits.
- Hexadecimal is an improvement.
  - Instead of 0–9, it is 0–F (A:10, B:11, C:12, D:13, E:14, F:15).
  - Each hex digit thus maps to 4 binary digits (1111 = 15).
  - Each decimal digit does **not** map to binary digits (9 = 1001).
- Convert these (in all cases, assume big-endian):
  - binary 11001001 to hex
  - hex FE to binary
  - decimal 64 to hex
  - hex E4D to decimal

## Hexadecimal A

- Convert these (in all cases, assume big-endian):
  - binary 11001001 to hex
    - 1100 = 12 = C, 1001 = 9 = 9. Answer: C9
    - Easy way: Use a look up table.
  - hex FE to binary
    - F = 15 = 1111, E = 14 = 1110 = 11111110
    - Again, could use look up table.
  - decimal 64 to hex
    - 64 = 01000000. Split. 0100 and 0000. 0100 = 4. 0000 = 0. Ans: 40.
  - hex E4D to decimal
    - E4D = 0E4D. Split. 0E = 14. 4D = 01001101 = 77.
    - 14 \* 256 = 3584. 77 \* 1 = 77. 3584 + 77 = 3661
    - Use Google to confirm.... "0xE4D to decimal".

## Limits

- What is the maximum value that can be stored in 1-byte?
- What happens if we add 10 to 249?

## Sign and Magnitude

- Plain binary starts from 0 and goes upward to the limit.
- Negative numbers are not handled.
- The first bit could be used as a Sign Bit.
- Two methods:
  - Excess-N
  - Two's Complement
- Very Suggested Reading:
  - [http://en.wikipedia.org/wiki/Signed\\_number\\_representations](http://en.wikipedia.org/wiki/Signed_number_representations)
- How is zero represented in each?
- What about 1, 127 and -127 in each?

## Sign and Magnitude A

- Excess-N
  - The lowest number (-N) is at binary 0.
  - Decimal 0 is therefore at the binary of N.
  - The biggest number is therefore the maximum possible in the bytes, minus N.
  - (The sign bit is not used properly...)
- Two's Complement
  - Decimal 0 is binary 0.
  - Positive numbers go up.
  - Negative numbers go down from 11111111.
  - Negatives are thus the compliment of positive - 1.
  - Sign bit **does** make sense!

## Binary Addition (Time Permitting)

- Convert these numbers to one of the formats, and then do an addition in binary:
  - 12
  - 65
  - If you chose Excess-N, N will be 127.

## Binary Addition A

- Excess-N (Excess-127 in this case)
  - $0 = 01111111$  (127 in decimal)
  - $0-12 = 01111111 - 00001100$  ( $127 - 12$ ) =  $01110011$  (115)
  - $0+65 = 01111111 + 01000001$  ( $127 + 65$ ) =  $11000000$  (192)
  - $01110011 + 11000000 = 100110011$  (307)
  - It is over 255, but Excess-N does not impose an upper bound.
  - To convert back to decimal, just subtract  $254$  (two numbers)
- Two's compliment
  - $12 = 1100$
  - $-12$  is  $1100 - 1$  (1011) complimented:  $11110100$
  - $65$  is  $65$  in standard binary:  $01000001$
  - Can add through 0 abusing the overflow, so:
  - $11110100 + 01000001 = \underline{1}00110101$  (chop off the first 1) Ans:  $63$
- Go and think about this!!

## That's All Folks...

- Any questions?
- Slides will go online to <http://www.tonychung.net/>
- Questions and tutorial suggestions to [a.chung@lancaster.ac.uk](mailto:a.chung@lancaster.ac.uk)