POLITECNICO DI TORINO
Dipartimento di Automatica e Informatica

Unit PX: How do we count?

NUMBERS IN THE DIGITAL WORLD

1

Positional numbering system in base B

Characteristics:

- Digits: { 0, 1, 2, ..., B-1 }
- Weight of i-th digit: B^i
- Representation (natural numbers) on N digits

$$A = \sum_{i=0}^{N-1} a_i \cdot B^i$$

2

B = 10?

3

Positional?

4

How do we count?

7 by most Chinese
8 by Guangdong & Hong Kong locals
5 by Malaysian or Singaporean

"A dead Italian" (popular Joke)

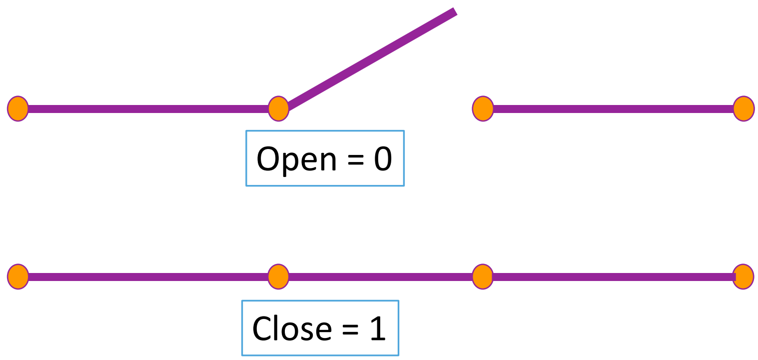
8 by most Chinese, but
7 in Guangdong, Fujian, Hong Kong and Taiwan
(and by Malaysian or Singaporean Chinese)

How do we count?

0 by most Chinese
10 by some?

Bit and switches

A switch has two states
(Open/Close, OFF/ON)



7

7

Some binary numbers

0	...	0	1000	...	8
1	...	1	1001	...	9
10	...	2	1010	...	10
11	...	3	1011	...	11
100	...	4	1100	...	12
101	...	5	1101	...	13
110	...	6	1110	...	14
111	...	7	1111	...	15

8

8

Some powers of 2

2^0	…	1	2^9	…	512
2^1	…	2	2^{10}	…	1024
2^2	…	4	2^{11}	…	2048
2^3	…	8	2^{12}	…	4096
2^4	…	16	2^{13}	…	8192
2^5	…	32	2^{14}	…	16384
2^6	…	64	2^{15}	…	32768
2^7	…	128	2^{16}	…	65536
2^8	…	256			

Binary to decimal conversion

- Apply the definition of the weighted sum of the binary numbers:
- $1101_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$

  $= 8 + 4 + 0 + 1$

  $= 13_{10}$

Decimal to binary conversion

- Algorithm for finding the binary representation of a positive integer:
- Divide by 2 the original value and record the remainder
- If the quotient is not zero, continue to divide the new quotient by 2 and record the remainder
- Once the quotient equals 0, the binary value consists of the remainders listed from right to left in the order they were recorded.

	6	3	1	0
1	0	1	1	

$13_{10} = 1101_2$

Binary addition

Basic rules :

0 + 0 = 0

0 + 1 = 1

1 + 0 = 1

1 + 1 = 0 (carry = 1)

Binary Subtraction

- Basic rules:

  0 - 0 = 0
  0 - 1 = 1 (borrow = 1)
  1 - 0 = 1
  1 - 1 = 0

Overflow

- Term **overflow** indicates the error that occurs, in an automatic calculation system, when the result of an operation can not be represented with the same code and number of bits of the operands.

Overflow - example

- *Assumption*: Operation on numbers represented with 4 bits,

  in plain binary:

0101 +

1110

----

overflow → **10011**

15

15

16

Octal system

- base = 8
  (Sometimes indicated with Q as octal)
- digits = { 0, 1, 2, 3, 4, 5, 6, 7 }
- Useful to write binary numbers in compact form (3:1)

$$\underbrace{1\ \ 0\ \ 1}_{2}\ \underbrace{1\ \ 1\ \ 1}_{7}\ \underbrace{0\ \ 0\ \ 1}_{1}{}_{2} \rightarrow 271_{8}$$

17

17

Hexadecimal system

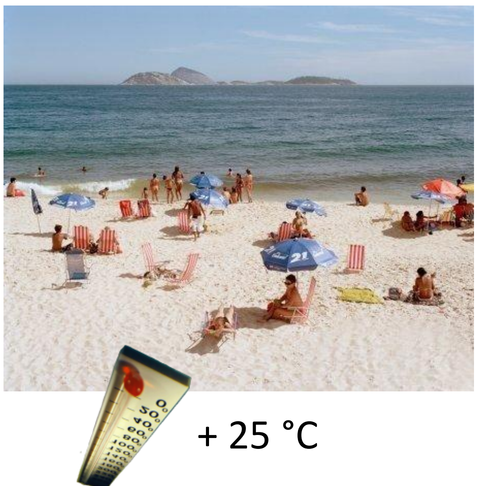
- base = 16
  (Sometimes indicated with H for Hexadecimal)
- digits= { 0, 1, ..., 9, A, B, C, D, E, F }
- Useful to write binary numbers in compact form (4:1)

$$\underbrace{1\ \ 0\ \ 1\ \ 1}_{B}\ \ \underbrace{1\ \ 0\ \ 0\ \ 1}_{9}{}_{2} \rightarrow B9_{16}$$

18

18

Realtive numbers representation



+ 25 °C

- 9 °C

Signed numbers

- A number can be:
  - positive (+)
  - negative (–)

- Representing the sign in binary is easy, but…
- the simpler solution is not always the best solution!
  - Sign and Magnitude (S&M)
  - One’s Complement (1C)
  - Two’s Complement (2C)

Sign and magnitude representation

- One bit for the sign (usually the MSB):
  - 0 = positive sign (+)
  - 1 = negative sign (–)
- N-1 bits for the *absolute value* (also called *magnitude*)

Sign	*Magnitude*
1 bit	N-1 bit

Sign and Magnitude: Examples

Using a 4 bit representation :

$+\,3_{10} \rightarrow 0011_{S\&M}$

$-\,3_{10} \rightarrow 1011_{S\&M}$

$0000_{S\&M} \rightarrow +\,0_{10}$

$1000_{S\&M} \rightarrow -\,0_{10}$

Sign and Magnitude

Issues:
- Double representation of zero (+ 0, - 0)

- Complexity of operations
  - Example: A+B

	A > 0	A < 0
B > 0	A + B	B - |A|
B < 0	A - |B|	- (|A| + |B|)

Sign and Magnitude: Limits

In a S&M representation on N bit:

$$-(2^{N-1}-1) \leq x \leq +(2^{N-1}-1)$$

Example:
- 8 bit = [-127 ... +127]
- 16 bit = [-32,767 ... +32,767]

One’s Complement

Given a binary number A of N bits, the one’s complement can be defined as :

$$\overline{A} = (2^N - 1) - A$$

Simply called *complement*.

It is usually indicated with an horizontal line upon the number or with the apostrophe (A’).

25

One’s Complement

Practical Rule:

One’s complement of a binary number A can be obtained by complementing (i.e., inverting) all of its bits

Example:

$$A = 1011 \rightarrow \overline{A} = 0100$$

26

Two's Complement

Given a binary number A of N bits, two's complement (2C) can be defined as :

$$\overline{\overline{A}} = 2^N - \overline{A} = A + 1$$

27

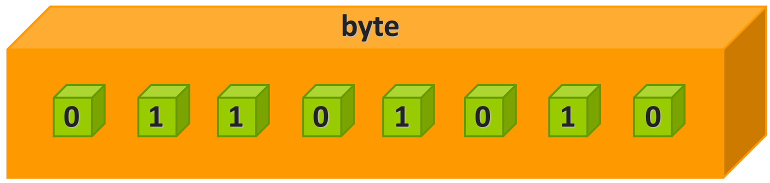
Encoding a value in two's complement

- in a two's complement number of N bits:
  - the MSB has a negative weight (-2^{N-1})
  - the rest of the bits have a positive weight according to its position
- in this way, the MSB determines the number sign:
  - 0 = **+** 1 = **–**
- examples (two's complement numbers on 4 bits):
  - $1000_{CA2} = -2^3 = -8_{10}$
  - $1111_{CA2} = -2^3 + 2^2 + 2^1 + 2^0 = -8 + 4 + 2 + 1 = -1_{10}$
  - $0111_{CA2} = 2^2 + 2^1 + 2^0 = 7_{10}$

28

28

Two’s complement



-2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0

29

Two’s complement (example)

101101

2^5 2^4 2^3 2^2 2^1 2^0

$$-32+8+4+1= -19$$

30

Two's Complement

Practical Rule:

Two's complement of a binary number A can be obtained by adding one to its one's complement

Example:

$$A = 1011 \rightarrow \overline{A} = 0100 \rightarrow \overline{\overline{A}} = \overline{A} + 1 = 0101$$

31

Two's Complement (bis)

Practical Rule (b):

The two's complement of a binary number is obtained starting from the LSB and copying all the bits up to the first "1" and complementing the remaining bits.

Example:

$$A = 10110 \rightarrow \underline{\underline{A}} = 01010$$

32

Two's Complement

- The two's complement representation is the most adopted in today's processors, since symplifies the circuit implementation for the basic arithmetic operations
- It is possible to apply the binary rules to all of the bits composing the number

Addition and Subtraction in 2's Complement

- The addition and subtraction are done directly, without considering the signs of the operands

  - $A_{2C} + B_{2C} \rightarrow A_{2C} + B_{2C}$
  - $A_{2C} - B_{2C} \rightarrow A_{2C} - B_{2C}$

Overflow two's complement addition

- Operands with different signs: overflow never occurs
- Operands with the same sign: Overflow occurs if the result has different sign
- In any case, carry is always neglected on MSB
- Note: it is not needed to define the overflow for the subtraction in two's complement because it is always computed with an addition.

35

35

Overflow in two's complement

```
0101 +            1110 +
0100 =            1101 =
---------        ----------
1001             11011 =
                  1011
```

overflow! — (5 + 4 = 9)

Impossible in 2C on 4 bits

OK — (-2 + -3 = -5)

36

36

Real numbers representation (Floating Point)

Scientic notation

3.14	**$0.314 \times 10^{+1}$**
0.0001	**0.1×10^{-3}**
137	**$0.137 \times 10^{+3}$**

S = sign

M = mantissa

E = exponent

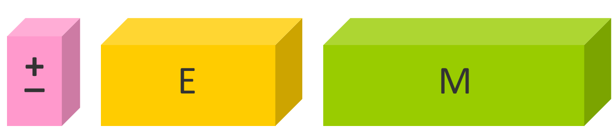
$$N = \pm\ M \times 2^{E}$$

37

Real numbers representation (Floating Point)

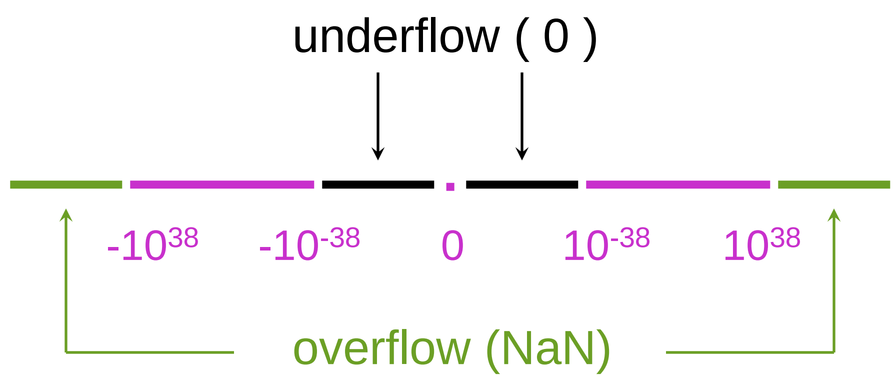
The computer system stores the following elements:

- Sign
- Exponent
- Mantissa



38

IEEE-754 SP



39

Problems

- Fix number of encoding bits
- Numbers are represented as sequences of digits
- Problems:
  - Representation interval
  - Overflow
  - Precision

40