Representation of Integer Numbers in Computer Systems

Positional Numbering System

- Additive Systems – history but ... Roman numerals
- Positional Systems:

$$A = \sum_{i=-\infty}^{+\infty} r_i^i a_i$$

r – system base (***radix***)

- *A* – number value
- *a* - digit
- *i* – digit position

e.g.

$-11{,}3125_{dec} = -1101{,}0101_{bin}$

$0{,}1_{dec} = -0{,}0(0011)_{bin}$

(!!! numbers may have infinite representation for some bases)

Base

- System Base *r* (***radix***)
  - constant value for all digit positions (***fixed-radix***)

    decimal, hexadecimal, octal, binary
  - may have different values for digit positions (***mixed-radix***)

    time: hour, minute, second $r = (24,60,60)$

    angle: degree, minute, second $r = (360,60,60)$

    factoradic $r = (\ldots 5!, 4!, 3!, 2!, 1!) = (\ldots 120, 24, 6, 2, 1)$

    $54321_{factoradic} = 719_{dec}$

    $5\times5! + 4\times4! + 3\times3! + 2\times2! + 1\times1! = 719$

    primoradic $r = (\ldots 11, 7, 5, 3, 2, 1)$

    $54321_{primoradic} = 69_{dec}$

    $5\times7 + 4\times5 + 3\times3 + 2\times2 + 1\times1 = 69$
  - may be other than natural number (negative, rational, complex, ...)

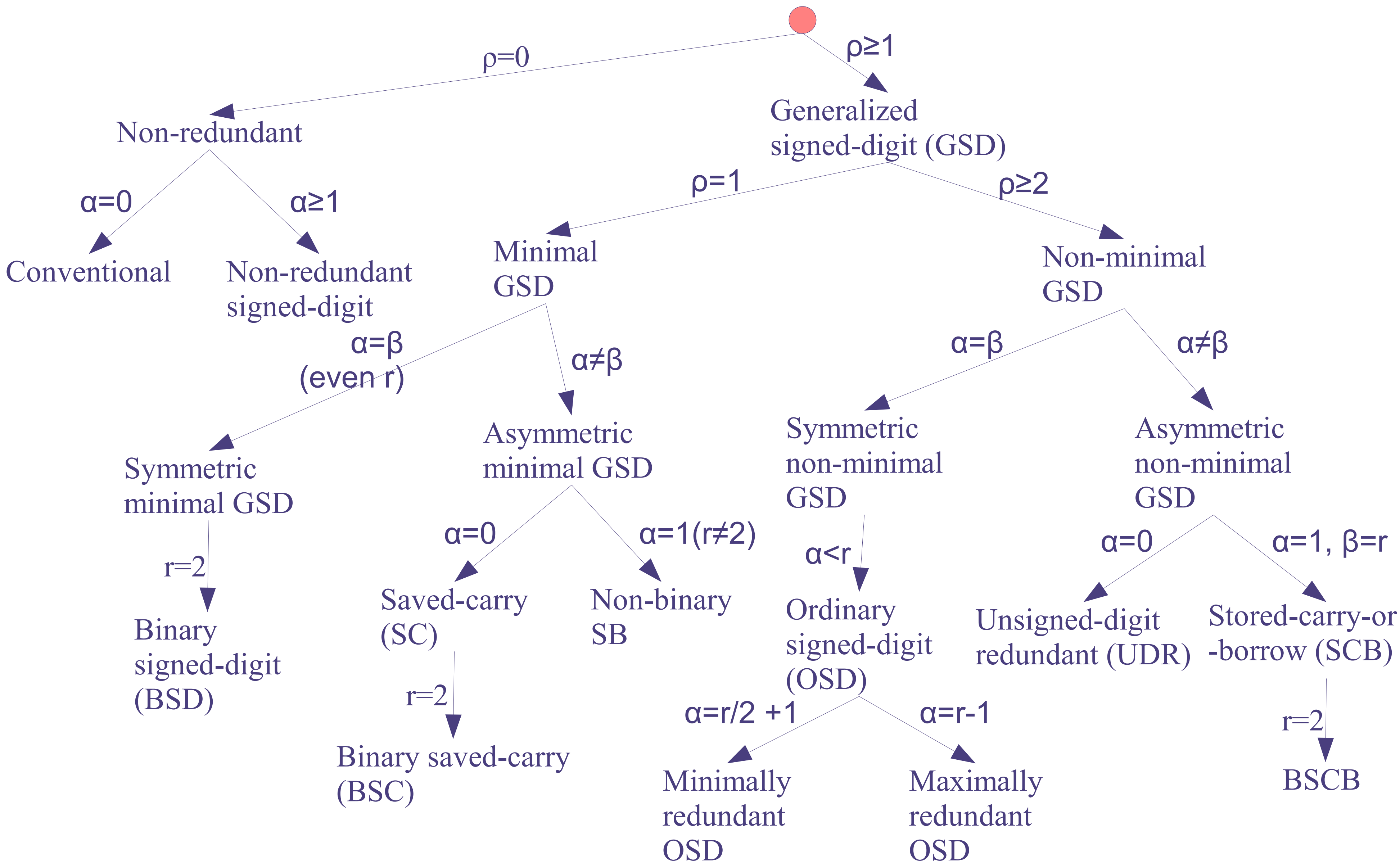
    $54321_{-10} = -462810_{dec}$

Digits

- *r*-radix system using standard digit set [0..*r*-1] is non-redundant
  - binary → 0, 1
  - decimal → 0... 9
  - hexadecimal → 0... F
- system using more digits than radix *r* is redundant
  - binary → 0,1,2 or -1,0,1
  - decimal → 0... F
  - decimal → 0... 9 ♠, ♣, ♥, ♦
- number representation in redundant systems is not unique
  - binary [0,1,2]: 1000 = 8_{dec} and   0120 = 8_{dec}

Numbering System Taxonomy

Positional fixed-radix systems with $[-\alpha, \beta]$ digit set → redundancy $\rho=\alpha+\beta+1-r$



Capacity

- In standard (non-redundant) *r*-radix numbering system, with n-digit number:
  - only numbers in range [0 ... r^n-1] can be represented
  - the number of unique representations is r^n
    - binary: 8-bits, range 0...255, unique values 256
- How many digits are needed to accommodate numbers from arbitrary range [0 ... max]?

$$n = floor[(\log_r max)] + 1 = ceil[\log_r(max + 1)]$$

e.g. for 50000 numbers (representations) in binary:

$\log_2 49999+1 = 16.61 \rightarrow$ (floor) $\rightarrow$ 16 digits (bits)

$\log_2 50000 = 15.61 \rightarrow$ (ceil) $\rightarrow$ 16 digits (bits)

Optimal radix

- Criteria (for standard, non-redundant):
  - short representation (small n) and few digits (small r)
  - convenient physical realization and handling
  - simple arithmetic algorithms (?)

- What is "the best" numbering system (i.e. radix) to represent numbers in a given range[0...max] ?
- Mathematical criterion: $E(r) = r * n$

  where *r* is numbering system radix for n-digit number

(one out of many possible criteria)

Optimal radix

- Looking for maximum of function: E(r) = r * n

$$E(r)=r*n=r*\log_r(max+1)=r*\frac{\ln(max+1)}{\ln(r)}=\ln(max+1)*\frac{r}{\ln(r)}$$

$$\frac{dE}{dr}=\ln(max+1)*\frac{\ln(r)-1}{\ln^2(r)}=0$$

$$r_{optimal}=e=2.71$$

Optimal (according to E(r) criterion) radix is 3, but 2 is almost as good and offers better physical implementation possibilities.

$$\frac{E(2)}{E(3)}=1.056,\ \frac{E(10)}{E(2)}=1.5$$

Special Codes

- Gray code – non-positional binary code
  - codes of every two successive values differ in only one bit
  - codes for first and last represented values also differ in only one bit (cyclic code)
  - applications: hazard-free digital electronics (counters, A/D converters, angle/position sensors, etc.)

value	Gray
0	0 0 0
1	0 0 1
2	0 1 1
3	0 1 0
4	1 1 0
5	1 1 1
6	1 0 1
7	1 0 0

Special Codes

- BCD – Binary-Coded Decimal
  - each decimal digit coded with 4 bits, one byte can accommodate positive numbers in ranage 0..99
  - applications:
    - communication with digital 7-segment LED displays
    - direct operations on decimal numbers in binary code – no problems with decimal/binary/decimal conversions

digit	BCD
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1

$$5127_{DEC} = 0101000100100111_{BCD}$$

Natural Binary Code (NBC)

- NBC features:
  - fixed radix-2 with two digits 0 and 1 ([0,1] digit set)
  - n-bit representation of non-negative values [0 ... 2^n-1]

$$a_{n-1}a_{n}\ldots a_{1}a_{0}=\sum_{i=0}^{n-1}2^{i}a_{i}$$

4-bits:	range 0 ... 2^4-1 → 0 ... 15
8-bits:	range 0 ... 2^8-1 → 0 ... 255
16-bits:	range 0 ... 2^{16}-1 → 0 ... 65 535
32-bits:	range 0 ... 2^{32}-1 → 0 ... 4 294 967 295
64-bits:	range 0 ... 2^{64}-1 → 0 ... 18 446 744 073 709 551 615

- NBC cannot represent negative values

Arithmetic Overflow in NBC

- Overflow: results of operation out of range
  - e.g. 8-bit: 11111111 + 00000001 = 1 00000000
- Carry-bit signals arithmetic overflow in NBC (unsigned arithmetics)
- Carry-bit is always stored by Arithmetical-Logical Units for the purpose of correctness control

Negative Numbers Coding

- Mapping negative numbers on range of positive rep.
- Simple arithmetic operations (addition/subtraction)
- Intuitive representation (?)

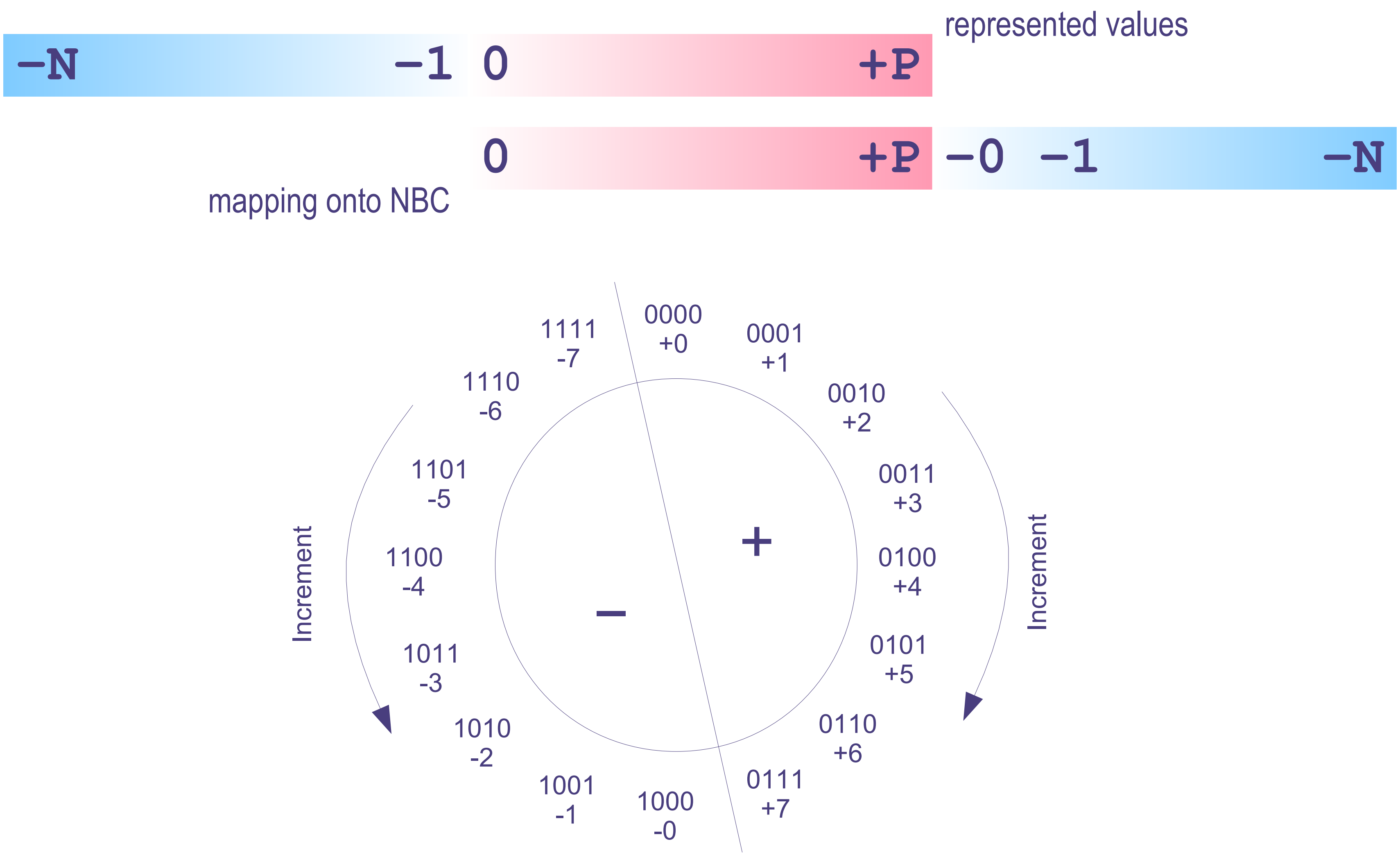
- Signed-magnitude coding (S-M)
- Biased coding (or excess-N)
- Complement coding (1's, 2's)

Signed-Magnitude (S-M)

- Oldest and simplest solution
- Binary n-digit S-M code:
  - most significant bit (MSB) represents the sign of the number (1 – negative, 0 – positive)
  - range of representation is symmetrical $[-2^{n-1}+1, 2^{n-1}-1]$
- Advantages:
  - intuitive representation
  - symmetrical range
  - simple negation
- Disadvantages:
  - complex arithmetical operations (addition/subtraction) !!!
  - double representation of zero

$$49_{DEC} = 00110001_{ZM}$$
$$-49_{DEC} = 10110001_{ZM}$$
$$+0_{DEC} = 00000000_{ZM}$$
$$-0_{DEC} = 10000000_{ZM}$$

Signed-Magnitude (S-M)



Biased Coding (Excess-N)

- Range [−N, +P] is mapped onto positive [0, N+P]
- Conversion requires addition of a bias value

[−4, +11] → [0, 15], bias = 4
−1 → 3

- Advantages:
  - quite intuitive representation
  - linear mapping – comparison of two numbers is easy
- Disadvantages:
  - result of addition/subtraction requires correction
  - multiplication/division is difficult

Biased Coding

- Binary n-digit Excess-N code:
  - range of representation $[-2^{n-1}, 2^{n-1}-1]$
  - bias (N) amounts to 2^{n-1}
  - MSB corresponds to sign of the coded value (0 – negative, 1 – positive) – opposite to S-M
  - bias correction(addition/subtraction) is easy for $N=2^{n-1}$, operation on MSB only
  - negation requires negation of all bits and addition of 1 to the total (similar to 2's complement negation)

$$16_{DEC} \rightarrow 16_{DEC} + \text{bias} = 16_{DEC} + 128_{DEC} = 10010000_{Excess128}$$
$$-16_{DEC} \rightarrow -16_{DEC} + \text{bias} = -16_{DEC} + 128_{DEC} = 01110000_{Excess128}$$

Biased Coding – Correction

- Addition/subtraction can be performed according to the same rules as NBC
- Result of addition/subtraction operations requires a correction:

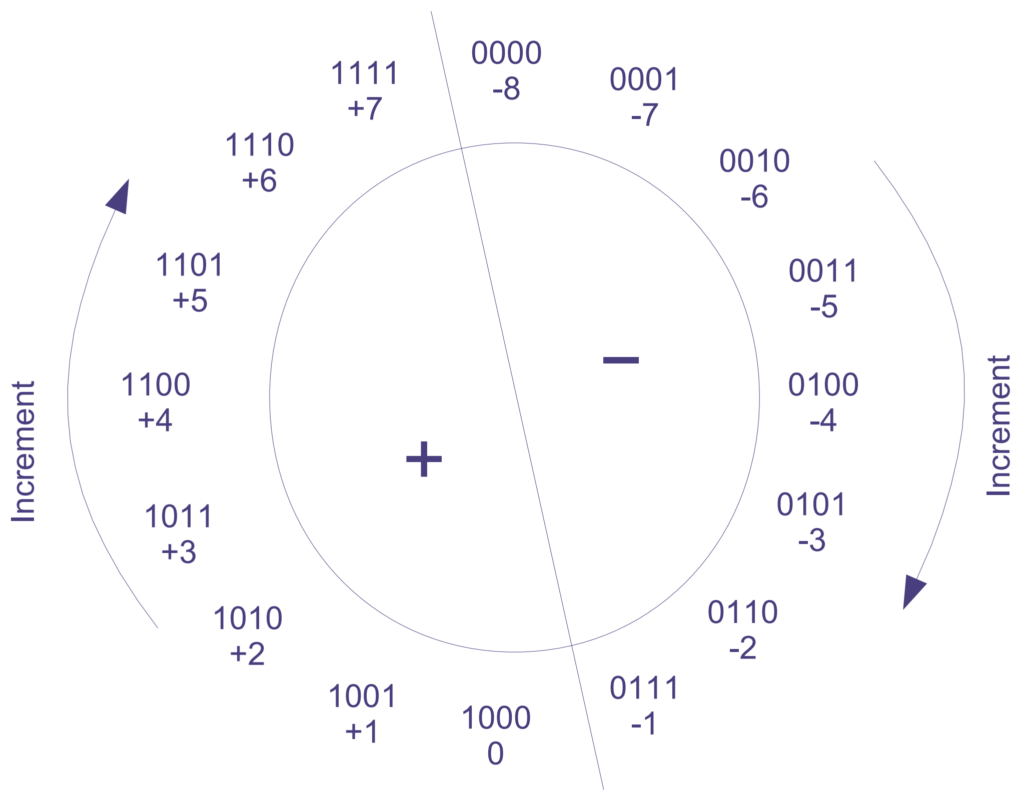
$X = x + bias$

$Y = y + bias$

$X + Y \rightarrow x + bias + y + bias = x + y + 2*bias \rightarrow X + Y \mathbf{- bias}$

$X - Y \rightarrow x + bias - y - bias = x - y + 0*bias \rightarrow X - Y \mathbf{+ bias}$

Biased Coding



Complement Coding

- Range [−N, +P] is mapped onto [0, N+P]
- Positive numbers are identical with NBC
- Representation of negative numbers is calculated as complement to a constant M = N+P+1

  $-x = M - x$

  $[-4, +11] \rightarrow [0, 15]$, $M = 16$
  $-1 \rightarrow 15$

- Advantages:
  - simple arithmetic operations – identical as in NBC !!!
- Disadvantages:
  - non-intuitive representation (but not for computers...:)

Binary 2's Complement Coding (2C)

- Range of representation $[-2^{n-1}, 2^{n-1}-1]$
- Complement constant $M = 2^n$ (*radix-complement*)
- MSB corresponds to the sign of the number (1 – negative, 0 – positive)
- Negation:

$$-x = 2^n - x = (2^n - 1) - x + 1 = 11...1_{BIN} - x + 1 =$$

$$= \text{bit\_negation}(x) + 1$$

- Modulo-M arithmetics:
  - ignoring carry bit (C – Carry) from last (n-1) position (drop carry-out)

2's Complement Negation

- Negation of a number x:
  - a) simple rule (binary): bit_negation(x) + 1
  - b) from definition (all positional) −x = M−x
  - c) from weighted-position formula (binary)

$$A_{U2} = -2^{n-1} a_{n-1} + \sum_{i=0}^{n-2} 2^i a_i$$

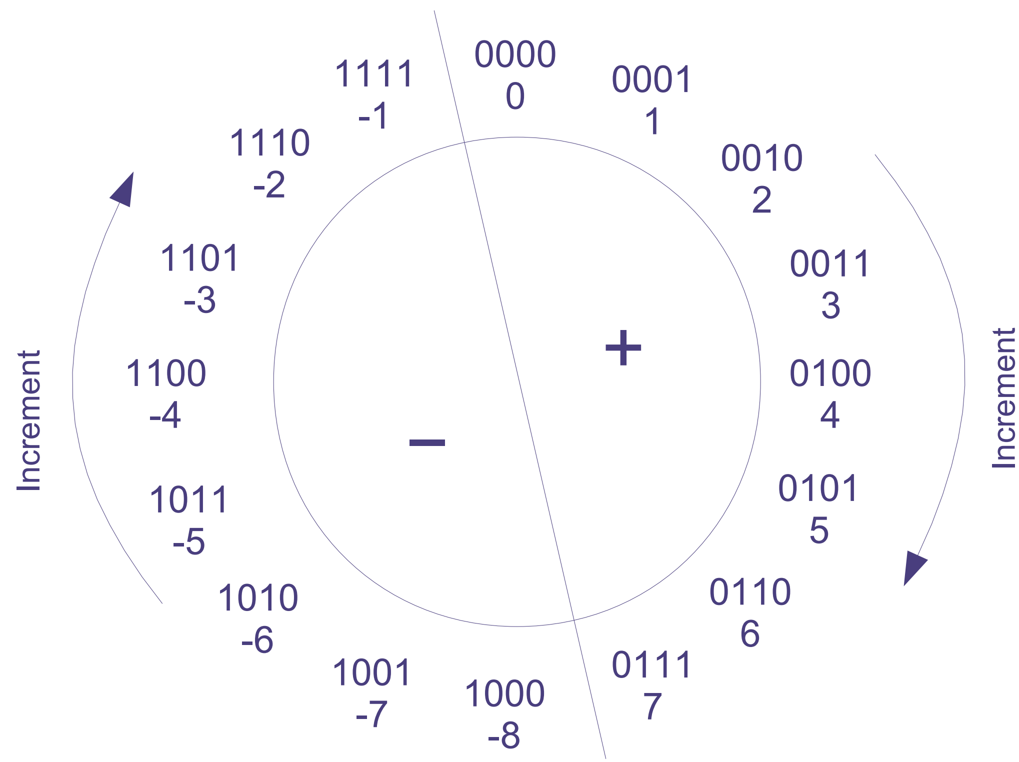
sign with negative weight

magnitude in NBC

Arithmetic Overflow in 2C

- Overflow: results of operation out of range
  - e.g. 8-bit: 01111111 + 00000001 = 10000000
  - overflow if two operands have the same sign, but different than result – simple comparison of MSB's
- Carry-bit does not signal arithmetic overflow in 2C
  - e.g. 8-bit: 11111111 + 00000001 = 1 00000000 (correct result, Carry is just a side-effect)
- Overflow bit (V) is always calculated by Arithmetical-Logical Units for the purpose of correctness control
- V-bit signals arithmetic overflow in 2C (signed-arithmetic)

2's Complement Coding



Binary 1's Complement Coding (1C)

- Range of representation $[-2^{n-1}+1, 2^{n-1}-1]$
- Complementation constant $M = 2^n-1$ (*digit-complement*)
- MSB corresponds to the sign of the number (1 – negative, 0 – positive)
- Double representation of zero
- Negation:

  $- x = 2^n - 1 - x = 11...1_{BIN} - x = \text{bit\_negation}(x)$
- Modulo-M arithmetics
  - adding carry bit (C – Carry) from last (n-1) position to the total (end-around carry)

1's Complement Coding (U1)