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.

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

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

(!!! finit (rational) numbers may have infinite representation)

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 = (... 5!, 4!, 3!, 2!, 1!) = (... 120, 24, 6, 2, 1)

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

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

    primoradic r = (... 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, ...)

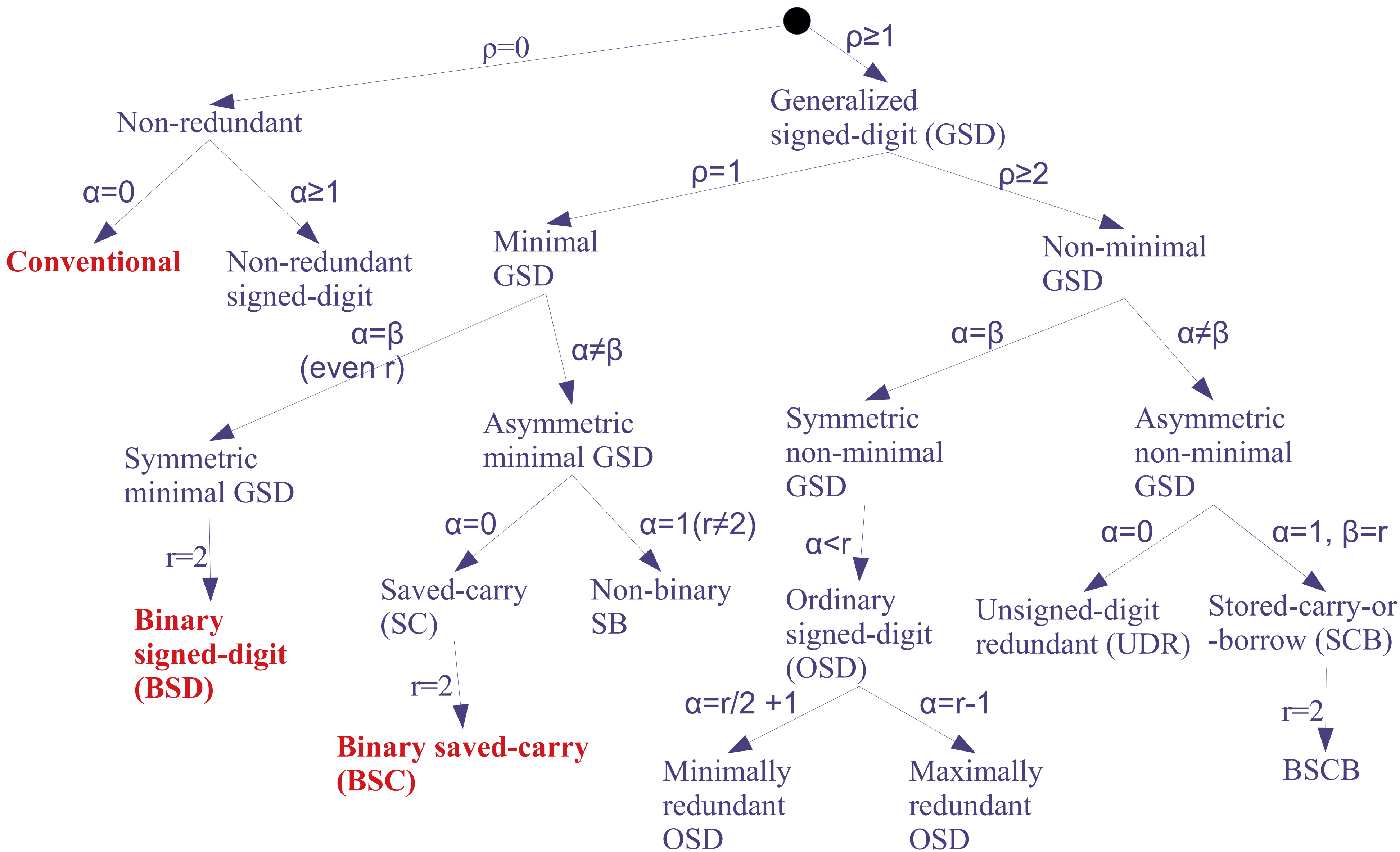
    $54321_{-10} = -462810_{dec}$

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 ♠, ♣, ♥, ♦]
- representation in redundant systems is not unique:
  - binary [0,1,2]: $1000 = 8_{dec}$ or $0120 = 8_{dec}$

Redundant Systems Taxonomy

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



Capacity

- In conventional (non-redundant) *r*-radix system, with n-digit number:
  - the range of representation is 0 ... r^n-1
  - the number of unique representations is r^n
    - e.g. 8-bits binary → range 0...255, with 256 unique values
- Number of digits 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:

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

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

Optimal radix ?

- What would '*the best*' numbering system (in term of ***r***) to represent numbers from a given range 0...max ?
- Criteria for conventional, non-redundant system:
  - high capacity ($\rightarrow$ small n)
  - few symbols-digits ($\rightarrow$ small r)
  - convenient physical realization
- Let's think of mathematical criteria:
  - $E(r) = r \cdot n$
  - ...

(where *r* is system radix for n-digit number)

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.

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

Non-Positional Numerical 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

Non-Positional Numerical Codes

- **BCD** – Binary-Coded Decimal
  - each decimal digit coded with 4 bits, one byte can accommodate positive numbers in range 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]

$$a_{n-1}a_n\ldots a_1a_0=\sum_{i=0}^{n-1}2^i a_i$$

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: result of addition is out of allowed range
  - e.g. 8-bit addition:

```
   11111111
 + 00000001
 ----------
 1 00000000 (9-bits)
```

- Carry-bit (C) signals arithmetic overflow in NBC for unsigned arithmetics
- Carry-bit is always stored by Arithmetical-Logical Units for the purpose of result correctness control

Negative Numbers Coding

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

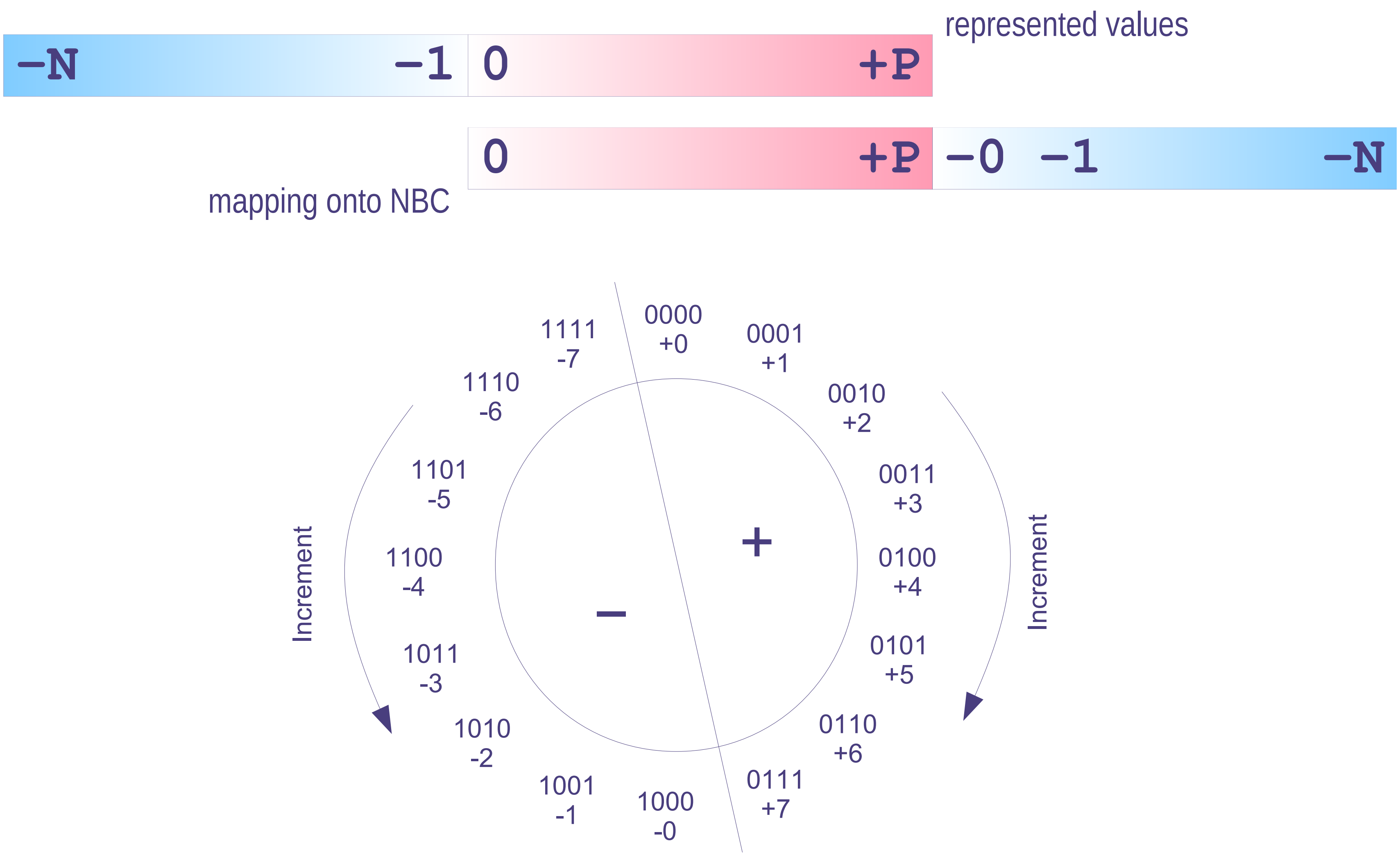
- Signed magnitude coding (SM)
- Biased coding (Bias-N or Excess-N)
- Complement coding (1C, 2C)

Signed-Magnitude (SM)

- Oldest, simplest, but inconvenient
- Binary n-digit SM code:
  - most significant bit 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

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

SM Mapping


represented values
−N
−1
0
+P
0
+P
−0
−1
−N
mapping onto NBC
0000
+0
0001
+1
0010
+2
0011
+3
0100
+4
0101
+5
0110
+6
0111
+7
1000
-0
1001
-1
1010
-2
1011
-3
1100
-4
1101
-5
1110
-6
1111
-7
+
−
Increment
Increment

Biased Coding (Excess-N)

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

  [−4, +11] with bias=4 → [0, 15]

  e.g. −1 → +3
- Advantages:
  - linear mapping – comparison of two numbers is easy
- Disadvantages:
  - 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}
  - most significant bit corresponds to the sign (0 – negative, 1 – positive) → opposite to SM and 2C
  - bias correction(addition/subtraction) is easy for $N=2^{n-1}$, toggling most significant bit
  - negation requires negation of all bits and addition of 1 to the total (same as in 2C)

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

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

Biased Coding – ADD/SUB Correction

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

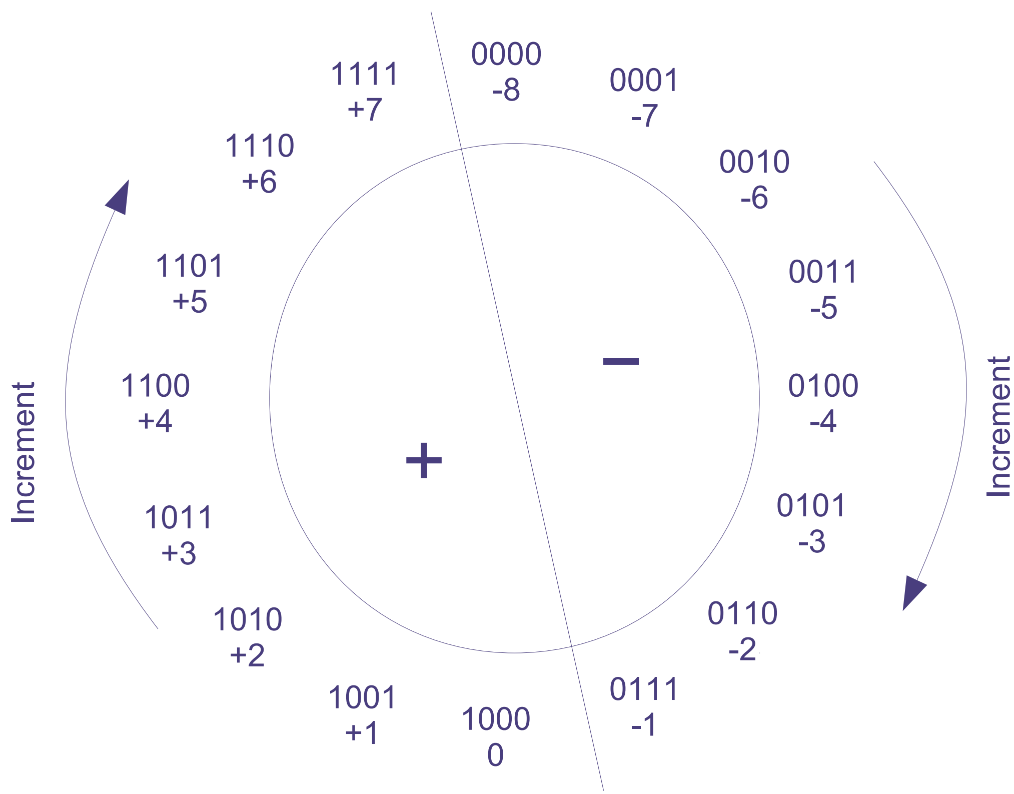
$X = x + \text{bias}$

$Y = y + \text{bias}$

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

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

Biased Coding Mapping



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] with M=16 → [0, 15]

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

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

  $=$ **bit_negation(x) + 1**
- Modulo-M arithmetics:
  - ignoring last carry bit (drop carry-out)

Negation in 2C

- Negation of x:

  a) simple rule (binary level): bit_negation(x) + 1

  b) from definition (all positional): −x → M−x

  c) from weighted-position formula (binary level):

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

sign with negative weight

magnitude in NBC

Arithmetic Overflow in 2C

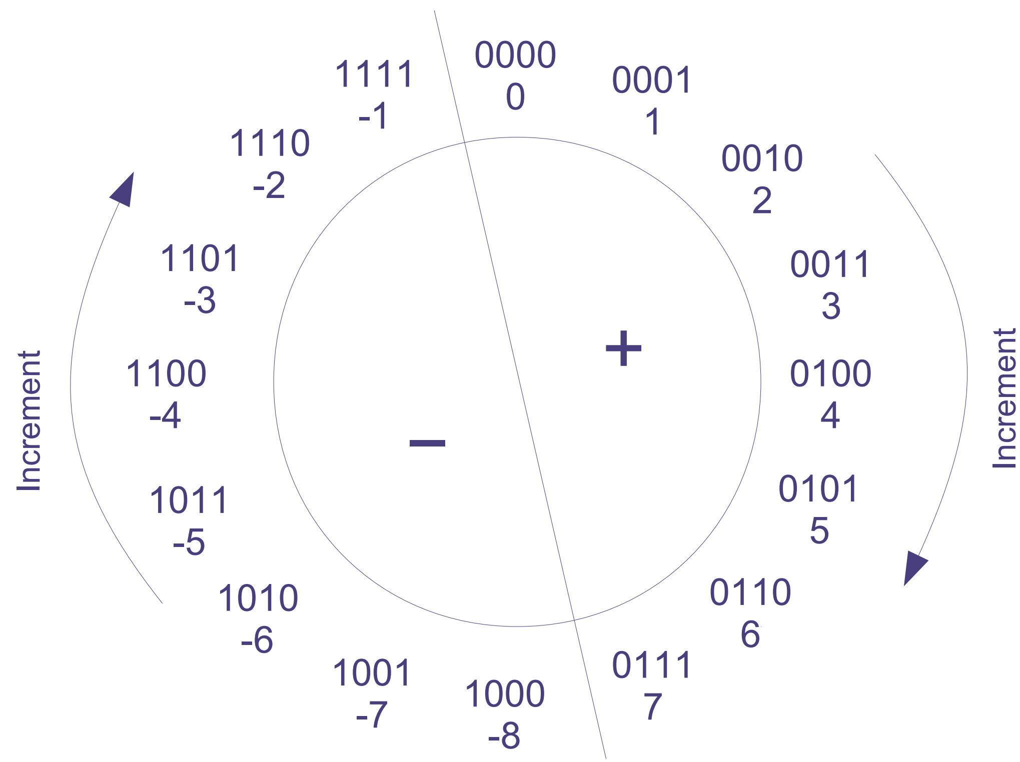
- Overflow: result of operation is out of range:
  - bit V signals arithmetic overflow in 2C (signed-arithmetic)
  - overflow: two operands have the same sign,
    but different than result – comparison of MSB's

  e.g. 8-bit: 01111111 + 00000001 = 10000000
- Overflow bit (V) is always calculated by Arithmetical-Logical Units for the purpose of correctness control
- Carry-bit (C) does not signal arithmetic overflow in 2C

  e.g. 8-bit: 11111111 + 00000001 = 1 00000000
  (result is correct, C is ignored)

2C Mapping

represented values
−N
−1
0
+P
0
+P
−N
−1
mapping onto NBC



0000
0
0001
1
0010
2
0011
3
0100
4
0101
5
0110
6
0111
7
1000
-8
1001
-7
1010
-6
1011
-5
1100
-4
1101
-3
1110
-2
1111
-1
+
−
Increment
Increment

Binary 1's Complement Coding (1C)

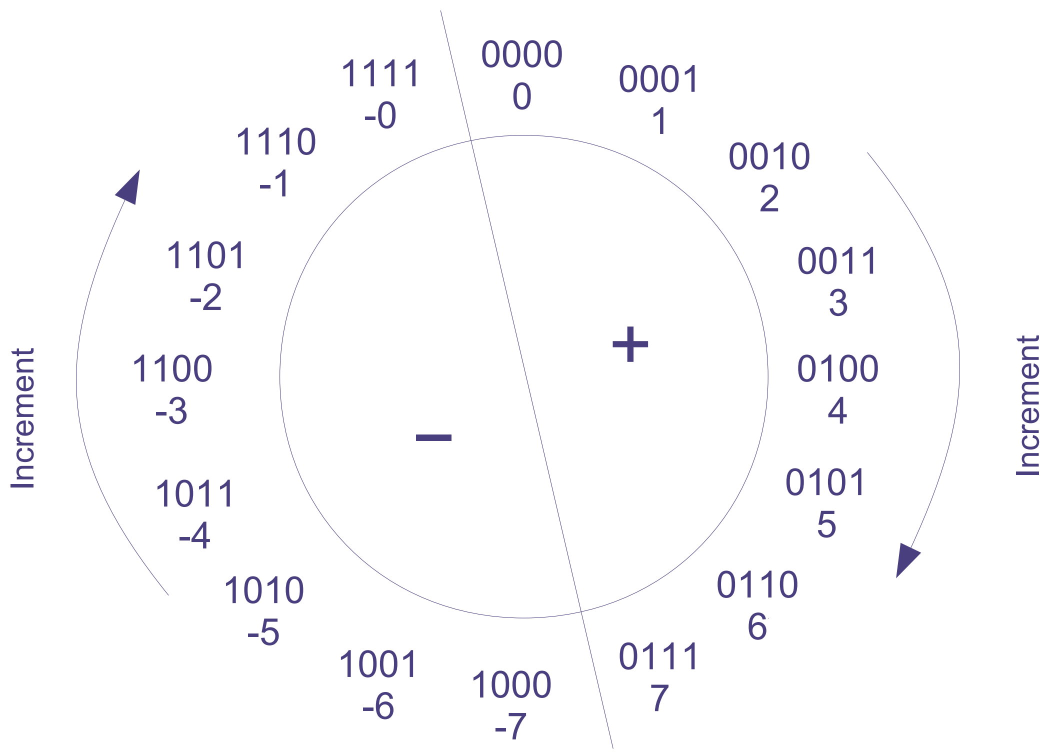
- Range of representation $[-2^{n-1}+1, 2^{n-1}-1]$
- Complementation constant $M = 2^n-1$ (*digit-complement*)
- Most significant bit corresponds to the sign (1 – negative, 0 – positive)
- Double representation of zero
- Negation:

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

- Modulo-M arithmetics
  - correction: adding carry bit from last position to the total (end-around carry)

1C Mapping


represented values
−N
−1
0
+P
0
+P
−N
−1
0
mapping onto NBC


0000
0
0001
1
0010
2
0011
3
0100
4
0101
5
0110
6
0111
7
1000
-7
1001
-6
1010
-5
1011
-4
1100
-3
1101
-2
1110
-1
1111
-0
+
−
Increment
Increment