



Kodowanie liczb całkowitych w systemach komputerowych

System pozycyjny

- Systemy addytywne – znaczenie historyczne
- Systemy pozycyjne

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

r – podstawa systemu liczbowego (**radix**)

- A – wartość liczby
- a - cyfra
- i – pozycja cyfry

np.

$$-11,3125_{\text{dziesiętnie}} = -1101,0101_{\text{dwójkowo}}$$

Reprezentacja części ułamkowej

- Liczby, które mają skończoną postać w jednym systemie, mogą mieć nieskończone rozwinięcie w innych systemach (!)

$$\text{np. } 0,1_{\text{dziesiętnie}} = -0,0(0011)_{\text{dwójkowo}}$$

Jaki będzie efekt działania tego programu?
Na czym polega błąd programisty?

```
#include <stdio.h>

int main()
{
    double i;
    for(i=0; i!=1; i+=0.1) printf("%f\n",i);
}
```

Podstawa systemu liczbowego

Podstawa systemu r (*radix*)

- ma stałą wartość dla wszystkich pozycji cyfry (**fixed-radix**)
dziesiętny, szesnastkowy, ósemkowy, dwójkowy
- może mieć różną wartość dla różnych pozycji (**mixed-radix**)

czas: godzina, minuta, sekunda $r = (24, 60, 60)$

kąt: stopnie, minuty, sekundy $r = (360, 60, 60)$

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

$$54321_{\text{factoradic}} = 719_{\text{dziesiętnie}}$$

$$5 \times 5! + 4 \times 4! + 3 \times 3! + 2 \times 2! + 1 \times 1! = 719$$

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


$$54321_{\text{primoradic}} = 69_{\text{dziesiętnie}}$$


$$5 \times 7 + 4 \times 5 + 3 \times 3 + 2 \times 2 + 1 \times 1 = 69$$


- nie musi być liczbą naturalną (liczby ujemne, wymierne, zespolone)

$$54321_{-10} = -462810_{\text{dziesiętnie}}$$

Cyfry systemu liczbowego

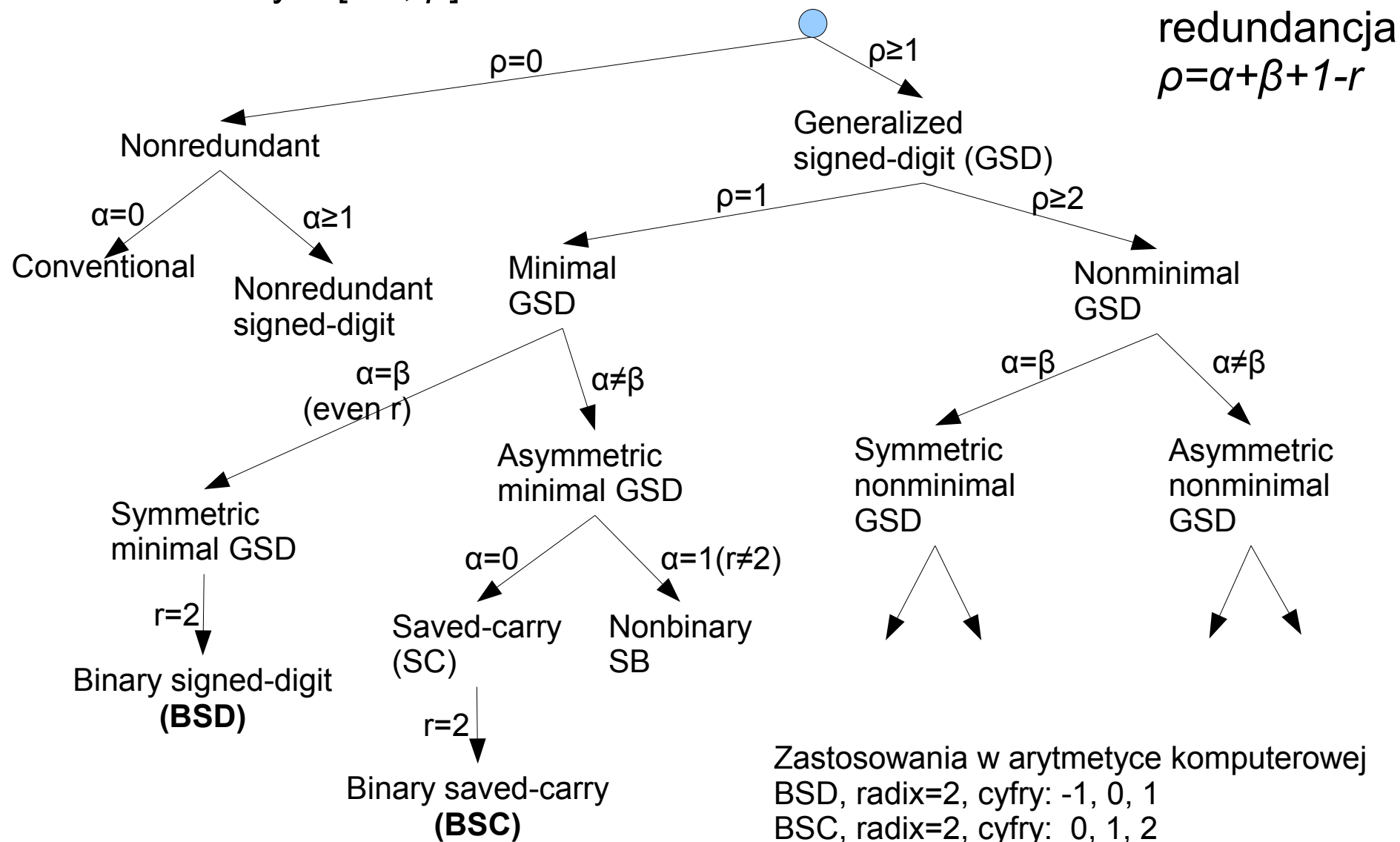
-  System o podstawie r , który wykorzystuje standardowy zestaw cyfr $[0..r-1]$ to system nieredundantny
 - dwójkowy $\rightarrow 0, 1$
 - dziesiętny $\rightarrow 0... 9$
 - szesnastkowy $\rightarrow 0... F$

-  System, który posiada więcej cyfr niż r jest systemem redundantnym
 - dwójkowy $\rightarrow 0, 1, 2$ lub $-1, 0, 1$
 - dziesiętny $\rightarrow 0... 9$ ♠, ♣, ♥, ♦







-  W systemach redundantnych reprezentacja liczby nie jest unikalna
 - dwójkowy $[0, 1, 2]$: $1000 = 8$ _{dziesiętnie} i $0120 = 8$ _{dziesiętnie}

Klasyfikacja (kryterium: redundancja)

Systemy pozycyjne o stałej podstawie r (r – liczba naturalna)
i zestawie cyfr: $[-\alpha, \beta]$



Kodowanie liczb całkowitych

-  W standardowym systemie liczbowym o podstawie r , za pomocą n -cyfrowej liczby:
 -  można reprezentować liczby z zakresu $[0 \dots r^n - 1]$
 -  liczba różnych reprezentacji wynosi r^n
 -  system dwójkowy: 8-bitów, zakres $0 \dots 255$, różnych liczb 256
 -  system piątkowy: 3-cyfry, zakres $0 \dots 124$, różnych liczb 125
-  Ile cyfr potrzeba na zapis liczb w zakresie $[0 \dots \max]$?


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


np. do zapisania 50000 liczb w kodzie dwójkowym potrzeba:



$$\log_2 50000 = 15.61 \rightarrow (\text{ceil}) \rightarrow 16 \text{ cyfr (bitów)}$$

$$\log_2 49999 + 1 = 16.61 \rightarrow (\text{floor}) \rightarrow 16 \text{ cyfr (bitów)}$$

Wybór optymalnej podstawy

-  Kryteria:
 - możliwie krótka reprezentacja (małe n) i mało cyfr (małe r)
 - wygodna implementacja fizyczna
 - „proste“ algorytmy arytmetyczne

-  Jaki system liczbowy (o jakiej podstawie) będzie „najlepszy” do kodowania liczb w zakresie $[0...max]$?

-  Kryterium matematyczne (jedno z wielu):
 -  $E(r) = r * n$

gdzie r – podstawa systemu dla n -cyfrowej liczby

Wymagany jest kompromis pomiędzy małymi wartościami r (łatwa implementacja) i n (efektywny zapis)

Wybór optymalnej podstawy (c.d.)

- Szukamy minimum funkcji $E(r) = r * n$ dla $r > 1$

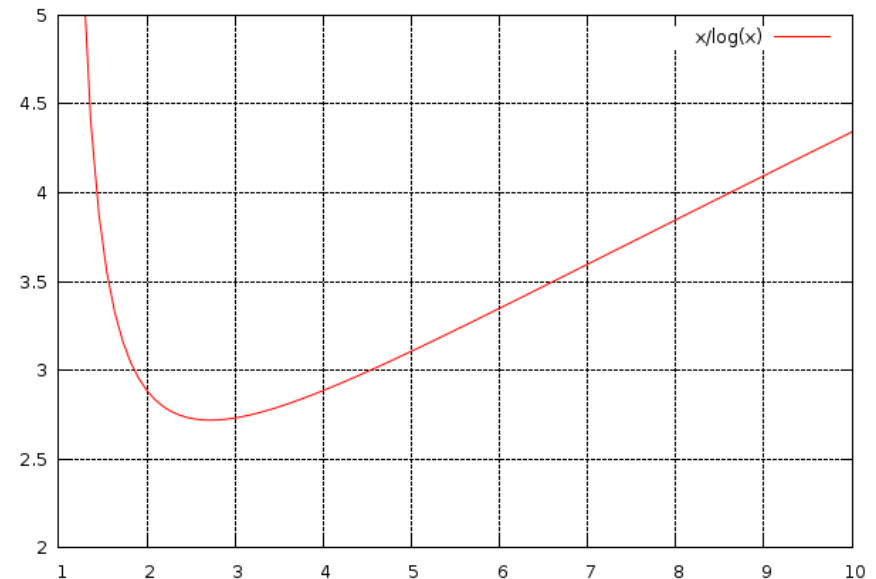
$$E(r) = r * n = r * \log_r(max + 1) = r * \frac{\ln(max + 1)}{\ln(r)} = \ln(max + 1) * \frac{r}{\ln(r)}$$

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

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

Optymalna, w sensie $E(r)$, podstawa to 3, ale podstawa 2 jest niewiele gorsza i dużo bardziej praktyczna ze względu na realizację fizyczną

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



Naturalny kod binarny (NKB, NBC)

- ☉ Zakładamy, że pod pojęciem NKB będziemy rozumieli
 - ☉ system stało-pozycyjny o podstawie 2 i cyfrach 0 i 1
 - ☉ n-bitową reprezentację liczb nieujemnych [0 ... max]

$$a_{n-1} a_n \dots a_1 a_0 = \sum_{i=0}^{n-1} 2^i a_i$$

4-bity: zakres 0 ... $2^4-1 \rightarrow 0 \dots 15$ (1 cyfra hex)

8-bitów: zakres 0 ... $2^8-1 \rightarrow 0 \dots 255$ (2 cyfry hex)

16-bitów: zakres 0 ... $2^{16}-1 \rightarrow 0 \dots 65\,535$

32-bity: zakres 0 ... $2^{32}-1 \rightarrow 0 \dots 4\,294\,967\,295$

64-bity: zakres 0 ... $2^{64}-1 \rightarrow 0 \dots 18\,446\,744\,073\,709\,551\,615$

- ☉ NKB nie pozwala na zapis liczb ujemnych

Kody specjalne

- 🕒 Kod Graya – dwójkowy kod niepozycyjny
 - 🕒 że każde dwa kolejne słowa kodowe różnią się tylko stanem jednego bitu.
 - 🕒 kod cykliczny – ostatni i pierwszy wyraz tego kodu także spełniają zasadę różnicy tylko na jednym bicie
 - 🕒 zastosowanie: unikanie stanów przejściowych w układach elektroniki cyfrowej (przetworniki analogowo-cyfrowych, czujniki położenia/obrotu, etc.)

wartość	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

Kody specjalne

● Kod BCD – Binary-Coded Decimal

- kodowanie cyfr dziesiętnych za pomocą czterech bitów, w jednym bajcie można zapisać liczbę z zakresu 0...99
- zastosowania:
 - do sterowania wyświetlaczami cyfrowymi w urządzeniach elektronicznych
 - do przechowywania i obliczeń bezpośrednio na liczbach dziesiętnych, bez konwersji do kodu dwójkowego.

cyfra	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

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

Kodowanie liczb ujemnych całkowitych

- Mapowanie liczb ujemnych na zakres NKB
- Intuicyjna reprezentacja liczb ujemnych
- Proste operacje arytmetyczne (dodawanie i negacja)

- Kod znak-moduł: ZM, (ang. *SM*)
- Kod z przesunięciem (ang. *Bias*, *Excess-N*)
- Kod uzupełnieniowy: U2, U1, (ang. *1C*, *2C*)

Kod znak-moduł (ZM)

- Najstarsze i najprostsze rozwiązanie
- W kodzie dwójkowym:
 - najbardziej znaczący bit liczby służy do kodowania znaku (1 – ujemna, 0 – dodatnia)
 - reprezentowane są liczby z zakresu $[-2^{n-1}+1, 2^{n-1}-1]$

● Zalety:

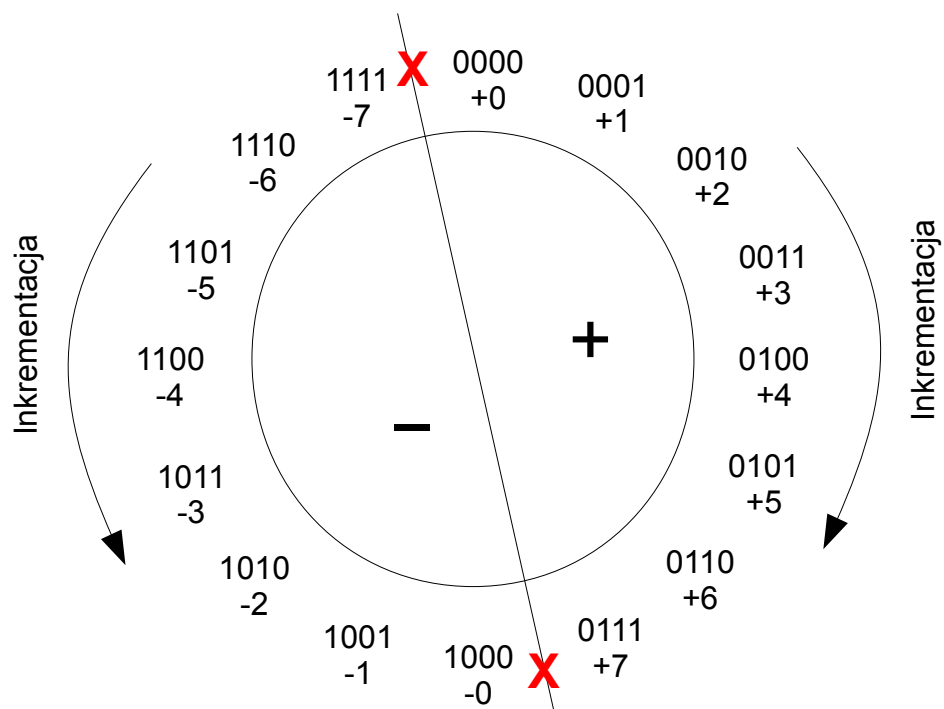
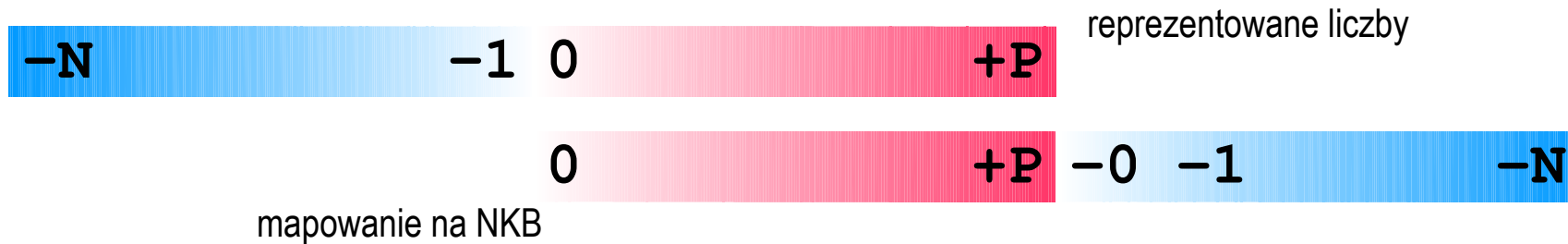
- intuicyjna reprezentacja
- symetryczny zakres
- proste operacja negacji

● Wady:

- skomplikowane dodawanie !!!
- podwójna reprezentacja zera

$$\begin{aligned}
 49_{\text{DEC}} &= 00110001_{\text{ZM}} \\
 -49_{\text{DEC}} &= 10110001_{\text{ZM}} \\
 +0_{\text{DEC}} &= 00000000_{\text{ZM}} \\
 -0_{\text{DEC}} &= 10000000_{\text{ZM}}
 \end{aligned}$$

Kod znak-moduł (ZM)



Kody z przesunięciem (Bias, Excess-N)

- Zakres $[-N, +P]$ jest mapowany na $[0, N+P]$
- Polega na dodaniu przesunięcia ($\text{bias}=N$) do liczby

$$\begin{aligned} [-4, +11] &\rightarrow [0, 15], \text{ bias} = 4 \\ -1 &\rightarrow 3 \end{aligned}$$

- Zalety:

- intuicyjna reprezentacja
- liniowe mapowanie – łatwe operacje porównania

- Wady:

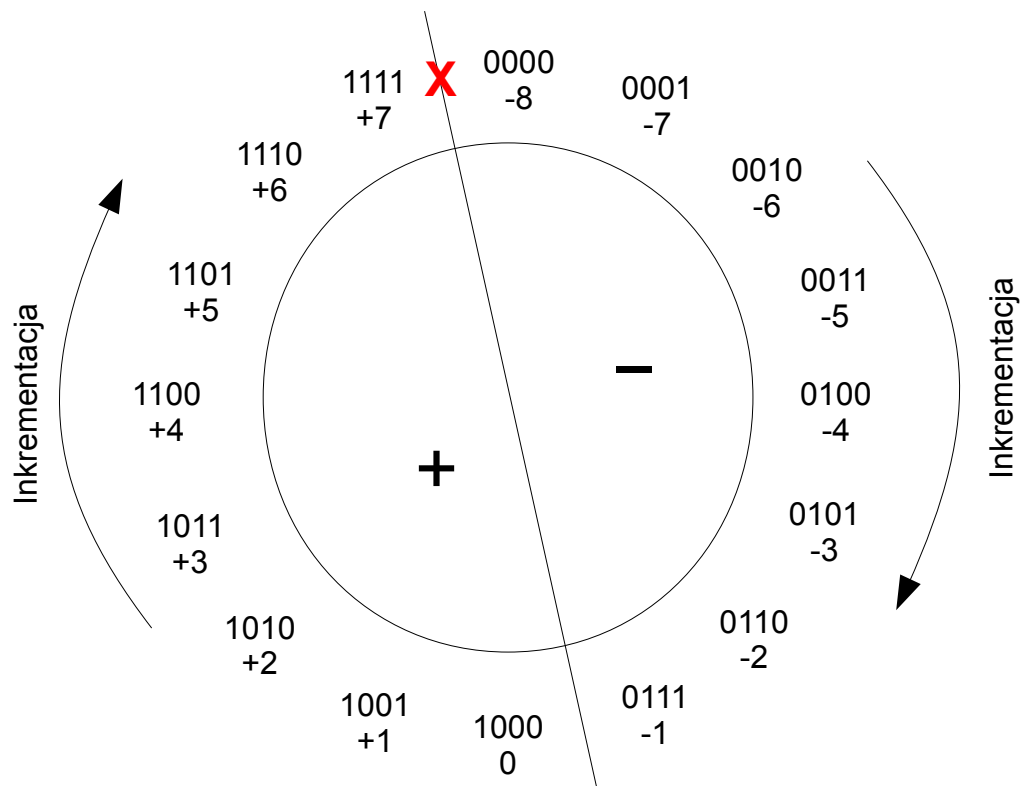
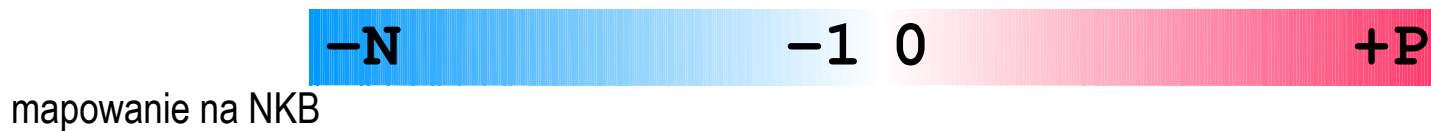
- dodawanie/odejmowanie wymaga korekcji wyniku
- mnożenie i dzielenie dość skomplikowane

n-bitowy kod Excess- 2^{n-1}

- W n-bitowym kodzie dwójkowym Excess- 2^{n-1} :
 - reprezentowane są liczby z przedziału $[-2^{n-1}, 2^{n-1}-1]$
 - przesunięcie (bias) wynosi 2^{n-1} (MSB = 1, pozostałe bity 0)
 - najbardziej znaczący bit liczby pozwala poznać znak (0 – ujemna, 1 – dodatnia lub 0) – przeciwnie do ZM
 - dodanie lub odjęcie przesunięcia wymaga jedynie operacji na najbardziej znaczącym bicie liczby (korekta na MSB)
 - negacja liczby polega na negacji wszystkich bitów i dodaniu jedynki do całości

$$\begin{aligned}
 16_{\text{DEC}} &\rightarrow 16_{\text{DEC}} + \text{bias} = 16_{\text{DEC}} + 128_{\text{DEC}} = 10010000_{\text{excess-128}} \\
 -16_{\text{DEC}} &\rightarrow -16_{\text{DEC}} + \text{bias} = -16_{\text{DEC}} + 128_{\text{DEC}} = 01110000_{\text{excess-128}}
 \end{aligned}$$

n-bitowy kod Excess- 2^{n-1}



Kody z przesunięciem – arytmetyka

- Operacje dodawania i odejmowania wymagają korekcji wyniku:

$$X = x + \text{bias}$$

$$Y = y + \text{bias}$$

$$X + Y \rightarrow x + \text{bias} + y + \text{bias} = x + y + 2 * \text{bias} \rightarrow (X + Y) - \text{bias}$$

$$X - Y \rightarrow x + \text{bias} - y - \text{bias} = x - y + 0 * \text{bias} \rightarrow (X - Y) + \text{bias}$$

Operacje $\pm \text{bias}$ to zmiana tylko bitu MSB

Uwaga na przekroczenie zakresu!

Kody uzupełnieniowe

- ☉ Zakres $[-N, +P]$ jest mapowany na $[0, N+P]$
- ☉ Reprezentacja liczb dodatnie jest bez zmian
- ☉ Postać liczb ujemnych oblicza się jako uzupełnienie do stałej uzupełniania $M \rightarrow M = N+P+1$

$$-x = M - x$$

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


$$-1 \rightarrow 15$$

- ☉ Zalety:
 - ☉ proste operacje arytmetyczne, identyczna jak w NKB !!!
- ☉ Wady:
 - ☉ mało intuicyjna reprezentacja (?)

Kod uzupełnienia dwójkowego (U2)

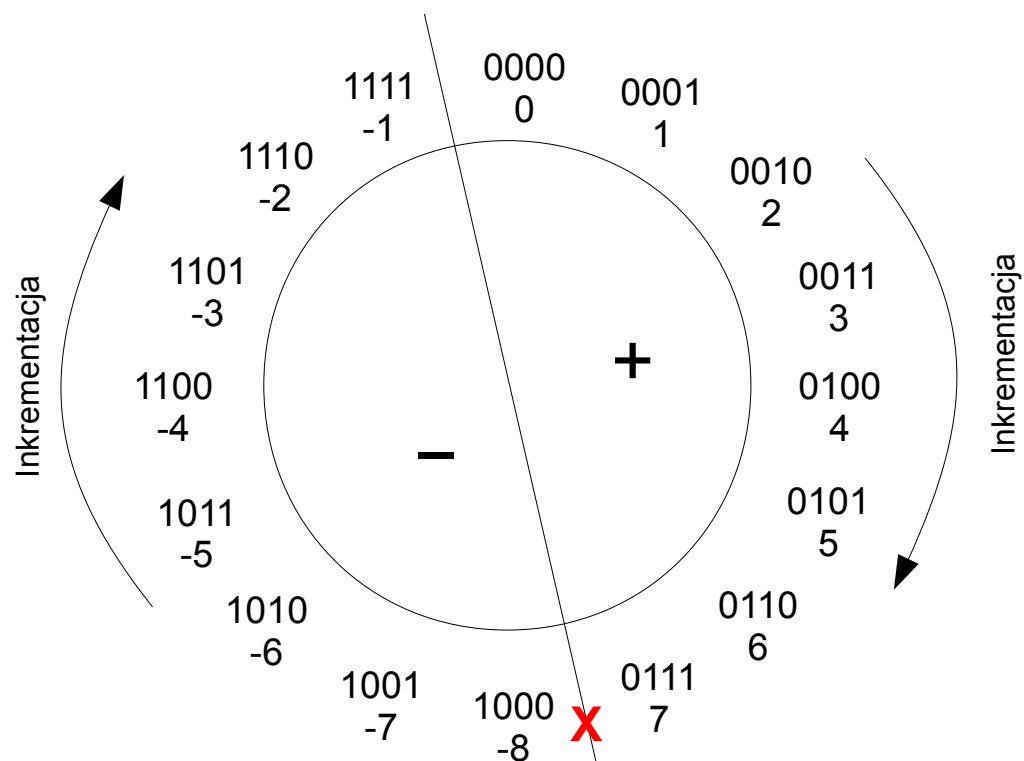
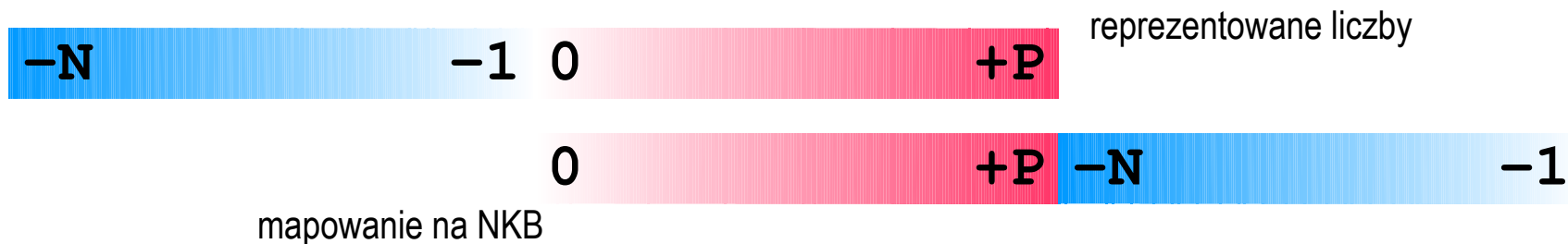
- reprezentowane są liczby z przedziału $[-2^{n-1}, 2^{n-1}-1]$
- stała uzupełnienia M wynosi 2^n (*radix-complement*)
- najbardziej znaczący bit liczby pozwala poznać znak (1 – ujemna, 0 – dodatnia)
- Negacja liczby:
 - $x = M - x = 2^n - x = (2^n - 1) - x + 1 = 11\dots1_{\text{BIN}} - x + 1 =$
 - = negacja_bitowa(x) + 1
- Arytmetyka modulo-M
 - w kodzie dwójkowym: ignorowanie bitu przeniesienia (C – Carry) z pozycji n-1 (*drop carry-out*)

Kod uzupełnienia dwójkowego (U2)

-  Obliczanie postaci/wartości liczby ujemnej
 - a) negacja_bitowa(x) + 1
 - b) ze wzoru definicyjnego $-x = M-x$
 - c) ze wzoru pozycyjno-wagowego

$$A_{U2} = \underbrace{-2^{n-1} a_{n-1}}_{\text{znak z wagą ujemną}} + \underbrace{\sum_{i=0}^{n-2} 2^i a_i}_{\text{moduł w NKB}}$$

Kod uzupełnienia dwójkowego (U2)



Kod uzupełnienia jedynkowego (U1)

- reprezentowane są liczby z przedziału $[-2^{n-1}+1, 2^{n-1}-1]$
- stała uzupełnienia M wynosi 2^n-1 (*digit-complement*)
- najbardziej znaczący bit liczby pozwala poznać znak (1 – ujemna, 0 – dodatnia)
- Dwie reprezentacje zera
- Negacja liczby:
 - $x = M - x = 2^n - 1 - x = 11\dots1_{\text{BIN}} - x = \text{negacja_bitowa}(x)$
- Arytmetyka modulo- M
 - w kodzie dwójkowym: dodawanie bitu przeniesienia (C – Carry) z pozycji $n-1$ do wyniku działania (*end-around carry*)

Kod uzupełnienia jedynekowego (U1)

