9. Reprezentace čísel a základní dvojkové aritmetické operace v počítači

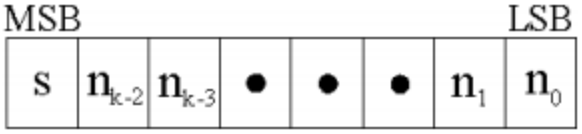
Čísla jsou ve výpočetních systémech reprezentovány v dvojkové (binární) soustavě. To znamená, že pro vyjádření čísel máme k dispozici pouze dvě číslice 0 a 1.

V případě binárních čísel nejnižší významovou číslici (tu nejvíce vpravo) nazýváme též nejnižším významovým bitem (anglicky Least Significant Bit) a značíme LSB. Nejvyšší významovou číslici (nejvíce vlevo) nazýváme též nejvyšším významovým bitem (anglicky Most Significant Bit) a značíme MSB.

Počet bitů *k* binárního čísla určuje rozsah hodnot *H*, kterých může nabývat.
Platí: $H = 2^k$
Endianita je jedním ze základních zdrojů nekompatibility při ukládání a výměně dat v digitální podobě. Při little-endian se na paměťové místo s nejnižší adresou uloží nejméně významný bajt (LSB) a za něj se ukládají ostatní bajty až po nejvíce významný bajt (MSB). Při big-endian se na paměťové místo s nejnižší adresou uloží nejvíce významný bajt (MSB) a za něj se ukládají ostatní bajty až po nejméně významný bajt (LSB) na konci.

Reprezentace čísel se znaménkem

Pro vyjádření čísel záporných je třeba definovat, jak bude příslušné znaménko určeno. Číslicový systém si může informaci o znaménku daného čísla pamatovat obecně různým způsobem. Typicky se však dodržuje konvence - znaménku se vyhradí jeden bit (nejčastěji MSB) příslušného binárního čísla, které má celkem *k* bitů.



Kladná čísla mají ve znaménkovém bitu hodnou 0. Záporná čísla mají ve znaménkovém bitu hodnotu 1. Čísla se znaménkem se reprezentují pomocí různých kódů.

Přímý kód:

Při přirozeném (přímém) binárním kódu hodnota binárního čísla odpovídá převedené hodnotě čísla dekadického. Pokud chceme vyjádřit číslo záporné, prostě jej na místě MSB doplníme znaménkem. Kód má dvě nuly – kladnou a zápornou. Tuto situaci je třeba vhodně ošetřit, což komplikuje počítání v tomto kódu.

Jedničkový doplněk

Jedničkový doplněk lze spočítat prostou negací jednotlivých bitů daného čísla (komplement 1 = 0 a naopak). Taktéž obsahuje kladnou i zápornou nulu. Používá se převážně jako mezioperace při počítání dvojkového doplňku.

Dvojkový doplněk

Je doplněk číselné soustavy o základu r = 2. Lze ho spočítat spočtením jedničkového doplňku daného čísla a přičtením jedničky. Nemá dvě nuly. Kód je nesymetrický - má o jednu více záporných hodnot než kladných (na sudém počtu čísel nelze zobrazit lichý počet hodnot – čísla kladná, záporná a nula). Nesymetričnost není na závadu při provádění většiny aritmetických operací (kromě speciálního případu násobení dvou nejzápornějších čísel). Lze ho relativně snadno realizovat logickými obvody. Představuje základ aritmetiky většiny číslicových systémů.

Sčítání a odčítání

Sčítání se řídí následující tabulkou:

+	**0**	**1**
0	0	1
1	1	(1)0

Součet dvou jedniček dává hodnotu 10, znamená to výsledek 0 s přenosem 1 do vyššího řádu (vyššího významového bitu).

Existují čtyři možnosti součtu (odčítání je přičítaní záporného čísla). (+) + (+) , (-) + (+) , (+) + (-) , (-) + (-). Při operacích typu (+) + (+) a (-) + (-) může dojít k přetečení v případě, že se výsledek dostane mimo rozsah zobrazení daný počtem platných bitů. Při operacích typu (-) + (+) a (+) + (-) k přetečení dojít nemůže.

Násobení

Násobení se řídí následující tabulkou:

x	**0**	**1**
0	0	0
1	0	1

Násobenec je postupně násoben jednotlivými bity násobitele od LSB k MSB. Mezivýsledky se vždy posunou o jeden bit vlevo a nakonec se vše sečte. Výsledek má dvojnásobný počet platných bitů. Při násobení záporných čísel je nutné čísla rozšířit na šířku součinu o znaménkové bity.

Boothův algoritmus

Je algoritmus určený k redukci počtu operací, který využívá mimo operaci přičtení násobence také jeho odečtení. Detekuje a nahrazuje řadu po sobě následujících jedniček v násobiteli a tím redukuje počet operací. Existuje mnoho verzí Boothova algoritmu. Základním je radix 2.

Radix 2 využívá znalosti bitu a jeho pravého souseda (je-li MSB vlevo). Překódování je na základě následující tabulky:

hodnota překódovávaného bitu	hodnota sousedního bitu (bit vpravo)	kód
0	0 / neexistuje	0
0	1	1
1	1	0
1	0 / neexistuje	-1

Neefektivita nastává při osamocených bitech s hodnotou 1. V takovém případě je vhodné použít vyšší radix. Radix 4 využívá dvojnásobnou zápornou a kladnou hodnotu násobence.

hodnota překódovávaných bitů	hodnota sousedního bitu (bit vpravo)	kód radix 2	kód radix 4
00	0	0 0	0
00	1	0 1	1
01	0	1-1	1
01	1	1 0	2
10	0	-1 0	-2
10	1	-1 1	-1
11	0	0-1	-1
11	1	0 0	0

Příklad radix 2 a radix 4:

82 x 125 = 01010010 x 01111101

125: 0 1 1 1 1 1 0 1 → 1 0 0 0 0 -1 1 -1

```
   0 1 0 1 0 0 1 0    (82)
×  1 0 0 0 0-1 1-1    (125)
  1111111110101110    (-82)
  0000000010100100    (+82 << 1)
  1111111010111000    (-82 << 2)
  0010100100000000    (+82 << 7)
  0010100000001010
```

Počet operací se redukoval z 6 na 4

Kontrola: 82x125 = 82x(128 + -4 + 2 + -1) = -82 + 82x2 + -82x4 + 82x128

82 x 125 = 01010010 x 01111101

125: 0 1 1 1 1 1 0 1 → (radix 2) 1 0 0 0 0 -1 1 -1 → (radix 4) 2 0 -1 1

```
   0 1 0 1 0 0 1 0    (82)
×    2   0  -1   1    (125)
  0000000001010010    (+82)
  1111111010111000    (-82 << 2)
  0010100100000000    (+164 << 6)
  0010100000001010
```

Dělení

Při dělení hledáme podíl a zbytek po dělení dělence (2N bitů) a dělitele (N bitů). Nejdřív zarovnáme dělitele podle dělence (posun o N-1 bitů vlevo). Pak se snažíme odečíst posunutý dělitel od průběžného zbytku. Na konci nám zůstane výsledek Q a zbytek R.
Druhou možností je nechat dělitele v pevné poloze a posouvat průběžný zbytek. Tímto způsobem odstraníme nedostatky případné HW realizace.

Dělení 53:7
(posuv d, R_i v pevné poloze, N=4)

D = 53 = 0011 0101_2
d = 7 = 0111_2

```
    00110101   R0 = D
0 x 00111000   q3·2³·d     q3=0
    --------
    00110101   R1

    00110101   R1
-1 x 00011100  q2·2²·d     q2=1
    --------
    00011001   R2

    00011001   R2
-1 x 00001110  q1·2¹·d     q1=1
    --------
    00001011   R3

    00001011   R3
-1 x 00000111  q0·2⁰·d     q0=1
    --------
    00000100   R=0100       Q=0111
```

Dělení 53:7
(d v pevné poloze, posuv R_i, N=3)

D = 53 = 110 101_2
d = 7 = 111_2

```
    1101010   R0 = 2D
-1 x 0111000  q2·2³·d     q2=1
    -------
    0110010   R1

    1100100   2·R1
-1 x 0111000  q1·2³·d     q1=1
    -------
    0101100   R2

    1011000   2·R2
-1 x 0111000  q0·2³·d     q0=1
    -------
    0100000   R=100       Q=111
```

Podobně jako v případě násobení existují algoritmy, které jsou schopny redukovat počet operací dělení zpracováním více bitů najednou a schopny pracovat s čísly se znaménkem. Nejjednodušším přístupem je základní varianta algoritmu SRT (radix 2) pracující po jednom bitu a určující následující operaci dle nejvyšších 3 bitů (včetně znaménkového bitu).

Příklad SRT: 41 : -6 = -6 zbytek 5

```
41=00101001, 6=0 110, -6=1 010
-1   00101001
     1010      +d
     11001001
 1   1001001x  <-
     0110      -d
     1111001x
 0   111001xx  <-
 1   11001xxx  <-
     0110      -d
     00101xxx
-1   0101xxxx  <-
     1010      +d
     1111xxxx  záporný zbytek
     0110       -d (korekce)
     0101      zbytek 5

     Q = -1 1 0 1 -1 =
     -16+8+0+2-1=-7
     po korekci -7+1=-6
```

	d>0		d<0	
Ri	bit podílu	operace	bit podílu	operace
000 111	0	žádná	0	žádná
001 010 011	1	-d	-1	+d
101 110 100	-1	+d	1	-d

Dle tabulky se provede přičtení nebo odečtení dělitele a určí hodnota bitu podílu. Následuje posunutí vlevo (s výjimkou posledního kroku).

V případě záporného zbytku je zapotřebí provést korekci na kladný a **zvýšit** Q o 1 (záporný zbytek zápornou hodnotu zvětšoval).

Pevná a plovoucí řádová čárka

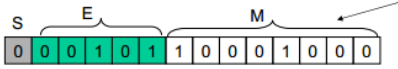
Při pevné řádové čárce (FX) jsou čísla jsou na číselné osa rozložena rovnoměrně. Například čísla bez desetinné části mají v přímém kódu na 8b rozsah 0-255.
Čísla s plovoucí řádovou čárkou (FP) nejsou na číselné ose rozložena rovnoměrně, což umožňuje zvýšit přesnost (více bitů M) nebo rozsah (více bitů E). Číslo X je tedy zadáno:

$$X = (-1)^S M.B^E ,$$

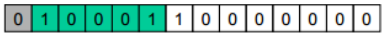
kde M je mantisa, S je znaménko, B je základ a E je exponent.

Příklad FP na 14 bitech

- E: 5 bitů, M: 8 bitů, S: 1 bit, B = 2
- Interpretace: $y = (-1)^S\ 0{,}M \times 2^E$
- Př. $17_{10} = 10001 \times 2^0 = 1000{,}1 \times 2^1 = 100{,}01 \times 2^2 = 10{,}001 \times 2^3 = 1{,}0001 \times 2^4 = \underline{0{,}10001 \times 2^5}$



- Př. $65536 = 2^{16} = 0{,}1 \times 2^{17}$ – se na 14 bitů FX v přímém kódu nevejde, ale v FP to lze



IEEE 754 : Příklad

Jaké číslo je zaznamenáno na 32 bitech v jednoduché přesnosti?
1 1000 0001 0100 0000 0000 0000 0000 000
v poli exponentu je číslo 129
exponent je tedy 129 - 127 = 2
zlomková část $f = ,01_2 = ,25$
mantisa je tedy 1,25
Jde tedy o číslo $-1{,}25 \cdot 2^2 = -5$

Standard IEEE 754

Standard IEEE 754 z roku 1985 definuje B = 2 a jednoduchou a dvojitou přesnost 24 a 53 bitů pro M. Kromě definice B, M, E definuje standard další výjimečné situace:

- nečíselný výsledek, označený zkratkou NaN např. při výpočtu odmocniny z -1
- definice nekonečna, které vznikne podílem 1/0

Norma definuje 4 presnosti vyjadrení:

- Jednoduchá přesnost 24 bitů
- Jednoduchá rozšířená přesnost ≥ 32 bitů
- Dvojitá přesnost 53 bitů
- Dvojitá rozšířená přesnost ≥ 64 bitů

Číslo N se získá z hodnoty E uvedené v poli exponentu a z hodnoty z pole mantisy (zlomková část) podle vzorce:

$$N = (-1)^S (2^{E-BIAS})\ (F0, F1 \ldots F23,\ nebo\ F52,\ nebo\ F63)$$

kde BIAS = 127, 1023, nebo 16 383.
Je důležité si uvědomit, že v poli exponentu je uvedeno číslo zvětšené o odnotu BIAS. Zlomková část f udává číslo menší než 1. Mantisu však získáme součtem 1 + f, což můžeme zapsat 1,f.
K zaokrouhlování dochází v případě, že dané číslo nelze přesně vyjádřit. Norma IEEE zaokrouhluje na číslo, jehož nejnižší číslice je sudá (ve dvojkové soustavě).