Celá a reálná čísla

Čísla, s nimiž pracujeme v matematice, mohou být z různých množin –

- $\mathbb{N}$ přirozená,
- $\mathbb{Z}$ celá,
- $\mathbb{Q}$ racionální,
- $\mathbb{R}$ reálná,
- $\mathbb{C}$ komplexní,

atd.

Tyto množiny jsou v inkluzi: $\mathbb{N} \subset \mathbb{Z} \subset \mathbb{Q} \subset \mathbb{R} \subset \mathbb{C}$.

Čísla v počítači

Programy v počítači mohou pracovat s čísly z libovolné číselné množiny.
Záleží na softwarových knihovnách, které jsou v programech použity.

Samotný procesor, do jehož instrukcí se programy překládají, však zpravidla podporuje jen tři číselné typy:

- *Celá čísla beze znaménka* – nezáporná celá čísla,
- *Celá čísla se znaménkem* – celá čísla, kladná i záporná,
- *Necelá čísla v pohyblivé řádové čárce* – reálná čísla tvaru $\pm m \cdot 2^e$

  V tomto číselném typu je mantisa m v kódu *dvojí nuly*
  a exponent e je v kódu *posunuté nuly*.

Velikost a přesnost čísel, s nimiž může procesor pracovat, je omezena délkou slova.

Nezáporná celá čísla

Celá čísla bez znaménka jsou v počítači uložena ve dvojkové soustavě přímo.
Tato reprezentace se nazývá *přímý kód*.

Velikost zobrazitelného čísla je omezena délkou slova, s nímž pracuje procesor.

Například 32bitový procesor pracuje s celými čísly bez znaménka z intervalu $\langle 0,\ 2^{32}-1 \rangle$,
64bitový procesor s čísly z intervalu $\langle 0,\ 2^{64}-1 \rangle$.

V programovacích jazycích mají celá čísla bez znaménka svůj typ,
např. v jazyce C je to typ `unsigned int`.

Nezáporná celá čísla v přímém kódu

Příklad: Kdyby byla délka slova jen 3 bity (místo obvyklých 64 bitů), pracovali bychom s osmi celými čísly bez znaménka, od 0 do 7.

	přímý kód
7	1 1 1
6	1 1 0
5	1 0 1
4	1 0 0
3	0 1 1
2	0 1 0
1	0 0 1
0	0 0 0

Cvičení: Sečtěte ve dvojkové soustavě $2 + 5$. Čísla jsou v přímém kódu (jako celá bez znaménka) ve tříbitovém slově.

Celá čísla

Celá čísla se znaménkem, kladná i záporná.
Jejich velikost je omezena délkou slova procesoru na interval, v jehož středu je nula.

Například 32bitový procesor pracuje s celými čísly se znaménkem z intervalu $\langle -2^{31},\ 2^{31}-1 \rangle$,
64bitový procesor s čísly z intervalu $\langle -2^{63},\ 2^{63}-1 \rangle$.

Celá čísla v přímém kódu s dvojí nulou

Nejvyšší bit čísla má význam znaménka a zbývající bity jsou dvojkové číslice.
Znaménko plus u kladného čísla je 0, číslo je za ním v přímém kódu.
Znaménko minus u záporného čísla je bit 1, absolutní hodnota čísla je za ním v přímém kódu.
Nula může mít obojí znaménko.

To však znamená dvě různé reprezentace nuly:

kladná nula $+0 = 0\,000\ldots000$
záporná nula $-0 = 1\,000\ldots000.$

Celá čísla v přímém kódu s dvojí nulou

Příklad:

Ve slově o délce 3 bity jsou čísla od -3 do 3 v přímém kódu se znaménkem a dvojí nulou.

	dvojí nula
3	011
2	010
1	001
+0	000
−0	100
−1	101
−2	110
−3	111

Instrukce počítající s celými čísly musí být co nejefektivnější. Přímý kód se dvojí nulou je pro ně nevýhodný, protože při každé operaci a porovnávání musí ošetřovat více případů.
Místo kódu se dvojí nulou se používá *kód dvojkového doplňku*.

Přímý kód s dvojí nulou se používá u *mantisy* čísel v *pohyblivé řádové čárce*.
Instrukce počítající s desetinnými čísly jsou pomalejší a složitější zacházení s nulou se na jejich efektivnosti neprojeví.

Celá čísla v kódu dvojkového doplňku

Procesory počítají s celými čísly v tzv. *kódu dvojkového doplňku*:

Nezáporná čísla jsou zapsána stejně jako v přímém kódu,

-1 má všechny bity jedničkové,

-2 vznikne odečtením jedničky od -1,

-3 vznikne odečtením jedničky od -2,

atd., odečítá se po jedničce, ...,

nejmenší zobrazitelné číslo má nejvyšší (znaménkový) bit jedničku a jinak samé nuly.

To znamená, že nejvyšší bit je opět znaménkový:
nezáporná čísla mají znaménkový bit `0`,
záporná čísla mají znaménkový bit `1`.

V programovacích jazycích mají celá čísla bez znaménka svůj typ,
např. v jazyce C je to typ `signed int`.

Kód dvojkového doplňku

Příklad: Kdyby byla délka slova jen 3 bity (místo obvyklých 64 bitů), pracovali bychom s osmi celými čísly od -4 do 3.

	dvojkový doplněk
3	011
2	010
1	001
0	000
-1	111
-2	110
-3	101
-4	100

Dvojkový doplněk má výhodu při sčítání a odčítání záporných a kladných čísel:
Nemusíme se starat o znaménko a můžeme se zápornými čísly zacházet stejně jako s kladnými.
Se znaménkovým bitem pracujeme stejně, jako by to byl bit v čísle.

Příklad: Ve tříbitovém slově odčítáme $1 - 3$:

$$\begin{array}{rcl} 1 & = & 001 \\ -3 & = & 101 \\ \hline -2 & = & 110 \end{array}$$

Kód dvojkového doplňku

Cvičení: Ve čtyřbitových slovech jsou uložena celá čísla od -8 do 7 v kódu dvojkového doplňku. Napište jejich kódy.

Cvičení: Historický počítač PDP 11 měl délku slova 16 bitů a čísla typu `signed int` měl uložená v jednom slově v kódu dvojkového doplňku. Určete jejich rozsah.

Cvičení: S číslem uloženým v kódu dvojkového doplňku provedeme postupně tyto operace:

1. všechny jedničkové bity změníme na nulové a naopak (včetně znaménkového bitu)
2. k výsledku přičteme číslo 1

Co se po těchto dvou operacích stane s původním celým číslem?

Kód dvojkového doplňku

Při sčítání a odčítání se se znaménkovým bitem zachází stejně jako s číslicemi za ním.

Cvičení: Ve čtyřbitovém slově sečtěte po bitech dvojice čísel:
$2 + 4$, $(-2) + (-5)$, $(-5) + 6$, $2 + (-5)$.
Výsledná čtyřbitová slova převeďte zpět na čísla v desítkové soustavě.

Přetečení rozsahu lze rozpoznat a ošetřit, ale lze je také ignorovat.
Při ignorování přetečení dostáváme korektní výsledky v *modulární aritmetice*.

Cvičení: Ve čtyřbitovém slově sečtěte po bitech dvojice čísel:
$3 + 6$, $(-5) + (-7)$
Výsledná čtyřbitová slova převeďte zpět na čísla v desítkové soustavě.

Poznámka: Hodnotu čísla zapsaného w bity v kódu dvojkového doplňku lze spočítat stejně jako u čísla v přímém kódu, ale hodnotu nejvyššího bitu bereme záporně, tedy -2^{w-1}.

Například -3 ve třech bitech je $101 = 1 \cdot (-2^2) + 0 \cdot 2^1 + 1 \cdot 2^0$.

Celá čísla v kódu posunuté nuly

Kód posunuté nuly je podobný přímému kódu, ale nula je posunuta na pozici nějakého kladného čísla (obvykle do středu intervalu).

Příklad: Nula posunutá o $+3$ ve slově o délce 3 bity.

	posunutá nula
4	1 1 1
3	1 1 0
2	1 0 1
1	1 0 0
0	0 1 1
-1	0 1 0
-2	0 0 1
-3	0 0 0

Posunutá nula se používá relativně málo, ale důležité je její využití při zápisu *exponentu* v zobrazení reálných čísel v *pohyblivé čárce*.

Dvojkový doplněk a posunutá nula

Výhody kódu dvojkového doplňku:

- Sčítat a odčítat čísla lze přímo, jako by znaménko bylo dvojkovou číslicí. Při případném *přetečení rozsahu* je získaný výsledek platným součtem/rozdílem v *modulární aritmetice*.
- nezáporná čísla jsou reprezentována stejně jako přímé zobrazení bezeznaménkového čísla.

Výhoda posunuté nuly:

- Porovnání čísel podle velikosti funguje stejně jako v přímém kódu a se znaménkem se pracuje jako s dvojkovou číslicí: nezáporná čísla jsou větší než záporná čísla.

Ve čtyřbitovém slově

- přímý kód,
- přímý kód s dvojí nulou,
- posunutá nula o 8
- dvojkový doplněk

vypadají takto:

	přímý kód
15	1111
14	1110
13	1101
12	1100
11	1011
10	1010
9	1001
8	1000
7	0111
6	0110
5	0101
4	0100
3	0011
2	0010
1	0001
0	0000

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

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

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

Reálná čísla

Reálná čísla obecně nelze uložit zcela přesně, ani když je bereme jen z omezeného intervalu. Stačí-li nám však pracovat s reálnými čísly, která jsou nejen z omezeného intervalu, ale navíc jsou zaokrouhlená jen na určitou přesnost, bude nám k jejich vyjádření stačit jedno počítačové slovo.

Například podle normy *IEEE 754* lze do počítačového slova uložit

- nulu,
- nenulová reálná čísla
  - ležící v předem daném (dostatečně velkém) intervalu,
  - zaokrouhlená na určitý počet platných číslic,
- i zvláštní nečíselné hodnoty $+\infty$, $-\infty$, $\perp$.

Toto uložení čísel v počítači se nazývá *pohyblivá řádová čárka*
a je datovým tvarem vyjádření čísla pomocí *mantisy* a *exponentu*.

Reálná čísla v pevné řádové čárce v počítači

Racionální čísla z pevného rozsahu a omezené přesnosti lze zobrazit i v *pevné řádové čárce*.
Musí být předem dáno, mezi kterými bity procesorového slova leží řádová čárka.

Příklad: Ve 32bitovém slově

- je nejvyšší bit znaménkový,
- v dalších 19 bitech je celá část mantisy (v přímém kódu)
- ve zbývajících 12 bitech je zlomková část mantisy.

Takto lze zobrazit čísla

od	$-2^{19} + 2^{-12}$	$\doteq$	$-524287{,}99976$	1 11111111111111111111 111111111111
do	$2^{19} - 2^{-12}$	$\doteq$	$524287{,}99976$	0 11111111111111111111 111111111111
krokem	2^{-12}			0 00000000000000000000 000000000001

Reálná čísla v pevné řádové čárce

Čísla v pevné čárce se používají málo a pokud se používají, tak hlavně u čísel v desítkové soustavě (BCD) při práci s finančními částkami.

Nevýhody pevné řádové čárky:

- příliš omezený rozsah,
- příliš omezená přesnost.

Reálná čísla ve tvaru mantisa–exponent

Desetinná čísla zapíšeme ve tvaru

$$m \cdot z^e$$

kde z je *základ* soustavy – přirozené číslo větší nebo rovné 2,
e je celočíselný *exponent*,
m je *mantisa* – desetinné číslo zaokrouhlené na předem daný počet míst.

Příklad: V čísle $6{,}022\,140\,76 \cdot 10^{23}$ je mantisa 6,022 140 76, exponent je 23 a základ soustavy je 10.

Poznámka: Pro zápis čísla ve tvaru mantisa–exponent se někdy používá doslovný překlad *vědecký zápis* (*scientific notation*).

Cvičení: Jaký je fyzikální význam čísla $6{,}022\,140\,76 \cdot 10^{23}$?
Vyhledejte.

Reálná čísla ve tvaru mantisa–exponent

Stejné číslo lze pomocí mantisy a exponentu zapsat více způsoby:

$$6{,}022\,14 \cdot 10^{23} \quad = \quad 602\,214 \cdot 10^{18} \quad = \quad 0{,}006\,022\,14 \cdot 10^{26} \quad = \quad 0{,}602\,214 \cdot 10^{24}$$

Je-li celá část mantisy nenulové jednociferné číslo,
tj. stojí-li před řádovou čárkou jedna nenulová číslice,
říkáme, že zápis mantisa-exponent je v *normalizovaném tvaru*.

Cvičení: Který z uvedených zápisů Avogadrova čísla je v normalizovaném tvaru?

Cvičení: Převeďte do normalizovaného tvaru čísla:

- $\mathrm{h} = 662{,}607 \cdot 10^{-36}\ \mathrm{J\,s}$
- $\mathrm{c} = 299\,792\,458\ \mathrm{m\,s^{-1}}$
- $\gamma = 0{,}000\,000\,000\,066\,74\ \mathrm{m^3\,kg^{-1}\,s^{-2}}$

Reálná čísla v pohyblivé řádové čárce v počítači

Reprezentace čísel v pohyblivé řádové čárce vychází ze zápisu mantisa–exponent ve dvojkové soustavě. Exponent je ve vyšších bitech, mantisa za ním.

- Základ soustavy je 2.
- V nejvyšším bitu je znaménko mantisy (bit `0` pro plus, bit `1` pro minus),
- následuje exponent v kódu posunuté nuly,
- následují dvojkové číslice mantisy *za řádovou čárkou*; mantisa je normalizovaná a číslice 1 před řádovou čárkou se *nezapisuje*

Reálná čísla v pohyblivé řádové čárce v počítači

Příklad: Podle normy IEEE 754 je v 64bitovém slově:

- 1 bit pro znaménko mantisy,
- 11 bitů pro exponent v kódu posunuté nuly (posun je o 1023 bity),
- 52 bitů pro číslice normalizované mantisy, *bez jedničky před čárkou*.

Poznámka: Čísla v pohyblivé řádové čárce mají v programovacích jazycích také svůj typ, např. v jazyce C jsou to typy `float` a `double`.

Reálná čísla v pohyblivé řádové čárce v počítači

Příklad: Číslo 0,3125 ve dvojkové soustavě je

$$0{,}3125 = 2^{-2} + 2^{-4} = 0{,}0101_2 = 1{,}01_2 \cdot 2^{-2}$$

$$0{,}3125 = \begin{array}{ccccc} 0 & 01111111101 & & 010000\ldots\ldots0 \\ + & & -2 & 1{,}25 \end{array}$$

$$0{,}3125 = 1{,}25 \cdot 2^{-2}$$

- Znaménko mantisy je plus, tedy bit 0 na nejlevější pozici.
- Exponent -2 v kódu nuly posunuté o $+1023$ je roven dvojkově 01111111101.
- Jednička z mantisy před binární čárkou se nezobrazuje, protože v normalizovaném čísle je vždy.
- Zbytek mantisy za binární čárkou je $010000\ldots0$.

Poznámka: Exponent 64bitového čísla v pohyblivé čárce je uložen v 11 bitech (bity 52 až 62) v kódu posunuté nuly. Posunutí je o 1023 a používají se exponenty -1022 až 1023. Čísla -1023 a 1024 na místě exponentu mají speciální význam.

exponent		význam
1024	11111111111	mantisa $+0 \Rightarrow +\infty$ mantisa $-0 \Rightarrow -\infty$ mantisa nenulová $\Rightarrow$ `NaN` – nedefinováno
1023	11111111110	
1022	11111111101	
⋮	⋮	
1	10000000000	
0	01111111111	$(\textit{znaménko mantisy}) \cdot (1 + \lvert\textit{mantisa}\rvert) \cdot 2^{\textit{exponent}}$
-1	01111111110	
⋮	⋮	
-1022	00000000001	
-1023	00000000000	mantisa $+0 \Rightarrow$ kladná nula mantisa $-0 \Rightarrow$ záporná nula mantisa nenulová $\Rightarrow$ denormalizované číslo

Čísla s denormalizovanou mantisou

Poznámka:

- Standardní nenulová čísla v pohyblivé čárce mají exponent od 00...0001 do 11...1110.
- Exponent 00...000 má číslo nula (s kladným i záporným znaménkem mantisy).
- Exponent 00...000 mohou mít také čísla velmi blízká nule, která výjimečně *nemají normalizovanou mantisu*. U nich se předpokládá nejmenší možný exponent (u 64bitových slov -1023) a na místě nezobrazeného bitu mantisy (před řádovou čárkou) je nula. Například v 64bitovém slově jsou to čísla $2^{-1074} \leqslant x < 2^{-1022}$ a čísla $-2^{-1022} < x \leqslant -2^{1074}$.

Cvičení: V 16bitovém slově jsou čísla v pohyblivé čárce uložena takto:

- nejvyšší bit je znaménko mantisy,
- v dalších 5 bitech je exponent s nulou posunutou o 15,
- v nejnižších 10 bitech jsou bity normalizované mantisy (bez jedničky před řádovou čárkou)

Které je nejmenší a které je největší zobrazitelné číslo s normalizovanou mantisou? Určete, jak jsou uložena čísla

1

-1

$-0{,}5$

1,0625

11,78125

Zaokrouhlovací chyby

Mnoho racionálních čísel nelze v pohyblivé čárce uložit přesně a musí se zaokrouhlit.

Například, neuvažujeme-li zvláštní nečíselné hodnoty, nulu a denormalizovaná čísla, lze do 64bitového slova uložit přesně a bez zaokrouhlování jen racionální čísla, jejichž absolutní hodnota je

$$(1 + k \cdot 2^{-52}) \cdot 2^e$$

pro celá čísla k, e, $0 \leqslant k < 2^{52}$, $-1022 \leqslant e \leqslant 1023$.

Ostatní reálná čísla z intervalů $(-2^{1024}, -2^{-1022}\rangle$ a $\langle 2^{-1022}, 2^{1024})$ je potřeba zaokrouhlit na nejbližší zobrazitelnou hodnotu.

Shrnutí

Na úrovni instrukcí procesoru se rozlišují tři způsoby reprezentace přirozených, celých a desetinných čísel:

- přímý kód pro přirozená čísla,
- dvojkový doplněk pro celá čísla,
- pohyblivá řádová čárka pro reálná čísla; v ní je
  - mantisa v přímém kódu se dvojí nulou
  - a exponent v kódu posunuté nuly.

Cvičení: Množiny všech přirozených, celých a reálných čísel jsou v inkluzi: $\mathbb{N} \subseteq \mathbb{Z} \subseteq \mathbb{R}$. Proč tedy procesor nepracuje jen s čísly v pohyblivé řádové čárce jako jediným způsobem reprezentace čísel?