Zobrazení čísel v počítači

Def.. 1 slabika = 1 byte = 8 bitů

1 bit = 0 nebo 1 (ve dvojkové soustavě)

Zobrazení celých čísel

Ke zpracování informace v počítači se z důvodu jednoduché realizovatelnosti používá zobrazení číslic nebo celých čísel ve dvojkové soustavě.
Pro zobrazení celých čísel lze v PC použít následujících **7** způsobů zobrazení:

a) dvojkově-desítkový tvar

BCD: Binary Coded Decimal

- do dvojkové soustavy se převádí jednotlivé číslice
- **hodnota každé číslice je uložena v jedné slabice** (1 slabika=8 bitů)
- číslice s nejmenší vahou se uloží do slabiky s nejnižší adresou
- operace sčítání: z paměti se vyberou nejnižší řády čísel , sečtou se a případně se do vyššího řádu přičte přenos atd. (algoritmus sčítání čísla o délce 1 slabika se od algoritmu sčítání čísel o N slabikách liší jen počtem kroků)

Př. $(12345)_{10}$

5	4	3	2	1

5	4	3	2	1
0000 0101	0000 0100	0000 0011	0000 0010	0000 0001

Př. $(742)_{10}$

2	4	7

0000 0010	0000 0100	0000 0111

b) zhuštěný dvojkově-desítkový tvar

packed BCD: Packed Binary Coded Decimal

- do dvojkové soustavy se převádí jednotlivé číslice
- **v jedné slabice se zobrazí dvě číslice** (pro zobrazení jedné číslice jsou vyhrazeny 4 bity, největší číslice v desítkové soustavě je **9**, $(9)_{10} = (1001)_2$ tj. 4 bity pro zobrazení jedné číslice stačí).
- dvojice číslic s nejmenší vahou se uloží do slabiky s nejnižší adresou
- číslice dělíme do dvojic od nejmenší váhy (tj. zprava)
- operace sčítání: z paměti se vyberou nejnižší řády čísel , sečtou se a případně se do vyššího řádu přičte přenos atd. (algoritmus sčítání čísla o délce 1 slabika se od algoritmu sčítání čísel o N slabikách liší jen počtem kroků)

Př. $(12345)_{10}$

45	23	01

4 5	2 3	0 1
0100 0101	0010 0011	0000 0001

Př. $(742)_{10}$

42	07

0100 0010	0000 0111

Pozn. Zobrazení znaménka u způsobu a) resp. b) je záležitostí konkrétního PC

c) binární soustava

viz. minulé cvičení
Tento typ zobrazení je použitelný pouze pro kladná čísla !

d) přímý kód se znaménkem

- číslo zobrazeno jako dvojice → znaménko
  → absolutní hodnota čísla

$$\check{c}_p = \pm|\check{c}|$$

$\check{c}_p$..obraz čísla v přímém kódu

Znaménko: zobrazeno ve znaménkovém bitu (bit s nejvyšší vahou).
Kladné číslo : ve znaménkovém bitu : **0**
Záporné číslo: ve znaménkovém bitu : **1**

Př. $(5)_{10}$ = 00000101 $(-5)_{10}$ = 10000101

znaménkový bit

nevýhoda: hodnota obrazu čísla se pro **zvětšující** hodnotu originálu → **zvětšuje** – kladná čísla
zmenšuje – záporná čísla,

z tohoto důvodu algoritmy sčítání/odčítání velmi složité.

$(4)_{10}$ = **$(\underline{0}000\ 0100)_{PK}$** $(-7)_{10}$ = **$(\underline{1}000\ 0111)_{PK}$**
$(7)_{10}$ = **$(\underline{0}000\ 0111)_{PK}$** $(-4)_{10}$ = **$(\underline{1}000\ 0100)_{PK}$**

Př. V přímém kódu zobrazte (na **osm** bitů) čísla:
a) 55 ***b)*** -55
Výsledek zapište v šestnáctkové soustavě**.**

55:2	27	**1**
27	13	**1**
13	6	**1**
6	3	**0**
3	1	**1**
1	0	**1**

$(55)_{10}$= **$(\underline{0}011\ 0111)_{PK}$ = $(37)_{16}$**
$(-55)_{10}$ = **$(\underline{1}011\ 0111)_{PK}$ = $(B7)_{16}$**

NEPŘÍMÝ KÓD (doplňkový kód, inverzní kód, kód s posunutou nulou)

- se zvětšováním originálu se zvětšuje i jeho obraz
- zavedení báze zobrazení, která je k originálu přičtena
- kladná i záporná čísla jsou zobrazována v oboru kladných čísel
- zobrazení zahrnuje stejně velkou množinu kladných i záporných čísel

e) doplňkový kód

Kladná čísla se zobrazují stejně, pro záporná čísla je volena báze $\mathbf{z^n}$

$$\check{c}_d = \check{c} \quad \Rightarrow \quad 0 \le \check{c} \le \frac{z^n}{2} - 1$$

pro

$$\check{c}_d = z^n + \check{c} \quad \Rightarrow \quad -\frac{z^n}{2} \le \check{c} \le 0$$

$\check{c}_d$……obraz čísla v doplňkovém kódu
n…….počet číslic zobrazení
z…….základ soustavy

Rozsah zobrazení je $-\frac{z^n}{2} \le \check{c} \le \frac{z^n}{2} - 1$

Př. V doplňkovém kódu zobrazte (na 16 bitů) čísla***:***
a) 55 b) -55 c) 1023 d) -1023
Výsledek zapište v šestnáctkové soustavě.

$(55)_{10}$ = **$(0000\ 0000\ 0011\ 0111)_{DK}$ = $(0037)_{16}$**

Trik pro rychlejší výpočet při zobrazování záporných čísel: **2^{16}-55 = 2^{16}-1-55+1**

maximální číslo zobrazitelné v binární soustavě na 16 bitů

2^{16}-1-55: v zápise čísla 55 v binární soustavě prohodíme 1 a 0

$(-55)_{10}$= 1111 1111 1100 1000 +1 = 1111 1111 1100 1001 = **$(1111\ 1111\ 1100\ 1001)_{DK}$ = $(FFC9)_{16}$**
inverze doplněk

Postup pro zobrazování **záporných čísel** v doplňkovém kódu:

1. zobrazit kladné číslo v binární soustavě
2. prohodit 1 a 0 v zápise binárního čísla
3. přičíst 1

$(1023)_{10}$ = **$(0000\ 0011\ 1111\ 1111)_2$ = $(03FF)_{16}$**

$(-1023)_{10}$= 1111 1100 0000 0000 +1 = **$(1111\ 1100\ 0000\ 0001)_2$ = $(FC01)_{16}$**
inverze

f) inverzní kód

podobný doplňkovému kódu, rozdíl jen v bázi posunutí
báze zobrazení: $\mathbf{z^n-1}$

$$\check{c}_i = \check{c} \quad \xrightarrow{\text{pro}} \quad 0 \le \check{c} \le \frac{z^n}{2} - 1$$

$$\check{c}_i = z^n - 1 + \check{c} \quad \Longrightarrow \quad -\frac{z^n}{2} + 1 \le \check{c} \le 0$$

$\check{c}_i$……obraz čísla v inverzním kódu
n…….počet číslic zobrazení
z…….základ soustavy

Rozsah zobrazení je $-\frac{z^n}{2} + 1 \le \check{c} \le \frac{z^n}{2} - 1$

Nula se zobrazí do dvou různých obrazů (kladná a záporná nula)
Pro zobrazení **záporných čísel** v doplňkovém a inverzním kódu zřejmě platí : $\check{c}_d = \check{c}_i + 1$,
tj. při výše popsaném triku neprovádíme krok 3 (přičtení jednotky).
V intervalu nezáporných čísel jsou obě zobrazení (v doplňkovém a inverzním kódu) identická.

g) kód s posunutou nulou

báze zobrazení: $\frac{z^n}{2}$ **, nebo** $\frac{z^n}{2} - 1$

$$\check{c}_{pn} = \frac{z^n}{2} + \check{c} \quad \textbf{, nebo} \quad \check{c}_{pn} = \frac{z^n}{2} - 1 + \check{c}$$

$\check{c}_{pn}$……obraz čísla v kódu s posunutou nulou
n…….počet číslic zobrazení
z…….základ soustavy

Rozsah zobrazení je $-\frac{z^n}{2} \le \check{c} \le \frac{z^n}{2} - 1$, nebo $-\frac{z^n}{2} + 1 \le \check{c} \le \frac{z^n}{2}$

Př: V kódu s posunutou nulou zobrazte (na **osm** bitů) čísla:
a) 55 b) -55 c) 2^5+1.

Výsledek zapište v šestnáctkové soustavě. Báze posunutí (zobrazení) je $\frac{z^n}{2} - 1 = 2^7-1$.

a) $2^7-1 +55 = 128+54=182$
b) $2^7-1 - 55 = 128-56=72$
c) $2^7-1+2^5+1=2^7+2^5=160$

a)

182:2	91	**0**
91	45	**1**
45	22	**1**
22	11	**0**
11	5	**1**
5	2	**1**
2	1	**0**
1	0	**1**

b)

72:2	36	**0**
36	18	**0**
18	9	**0**
9	4	**1**
4	2	**0**
2	1	**0**
1	0	**1**

c)

10000000
+ 00100000
10100000

$(182)_{10}=(1011\ 0110)_2=(B6)_{16}$ $(72)_{10}=(0100\ 1000)_2=(48)_{16}$ $(160)_{10}=(1010\ 0000)_2=(A0)_{16}$

Obecně
zobrazení celých čísel (výše zmíněnými 7 způsoby) je prováděno naprosto přesně. Rozsah zobrazení je dán počtem číslic. Pro zobrazení čísla ve dvojkové soustavě se nejčastěji používá slovo o délce 8, 16, 32, nebo 64 bitů.

Zobrazení čísel v pohyblivé řádové čárce

Zobrazení reálných nebo příliš velkých celých čísel se provádí v **pohyblivé řádové čárce.**
Čísla jsou zobrazena ve tvaru:

$$č = M \cdot z^E$$

kde

M…mantisa čísla, zobrazená v soustavě o základu z
E….exponent
z….základ pro výpočet exponentové části

V PC je pak číslo zobrazováno jako dvojice (M,E).
Přesnost zobrazovaného čísla závisí na počtu číslic mantisy
Rozsah zobrazení závisí na počtu číslic exponentu

Pozn.
Základ soustavy pro zobrazení exponentu i mantisy se většinou volí shodný se základem pro výpočet exponentové části. Některé počítače však z důvodu zvětšení rozsahu zobrazitelného exponentu(a tím i zobrazovaného čísla) zobrazují mantisu i exponent v jiné soustavě , než je základ pro zobrazení exponentové části. Např. **M** a **E** je v binární soustavě ale **z** v hexadecimální.

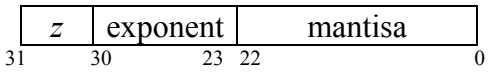
K dosažení co největší přesnosti zobrazení daného čísla se mantisa upravuje na tzv. ***normovaný tvar*** pro který platí:

$$-1 \le M < -\frac{1}{z} \cup 0 \cup \frac{1}{z} \le M < 1$$

kde symbol $\cup$ značí sjednocení intervalů.
Konkrétní způsob zobrazení mantisy a exponentu závisí na typu počítače a jeho aritmetických instrukcích.
Jedním z používaných formátů pro zobrazení čísel v pohyblivé řádové čárce je formát podle standardu IEEE 754(Institute of Electrical and Electronic Engineers) používaný v moderních počítačích.

Reálná čísla jsou zobrazována
- v jednoduché přesnosti (délka slova 32 bitů)
- v dvojnásobné přesnosti (délka slova 64 bitů)
- v rozšířeném formátu (délka slova 80 bitů)

Zobrazení reálného čísla v jednoduché přesnosti:



Mantisa

- je uložena na 23 bitech v přímém kódu se znaménkem
- Znaménkový bit mantisy je označen *z*
- Kladné číslo má znaménkový bit nulový, u záporného čísla je v *z* uložena 1
- Nejvyšší bit mantisy je vždy 1 a nezobrazuje se (mantisa se ukládá počínaje druhým významným bitem-ještě zvyšuje přesnost zobrazení)
- Myšlená desetinná tečka je umístěna za nejvyšším bitem mantisy
- Absolutní hodnota mantisy se tedy zobrazí v intervalu $1 \leq |m| < 2$
- Od normovaného tvaru se upouští pouze tehdy, když výsledek operace je v absolutní hodnotě menší, než je schopen exponent zobrazit. Mantisa se pak zmenší na úkor přesnosti a začne se zobrazovat i nejvyšší bit mantisy.

Exponent

- je uložen na 8 bitech v kódu s posunutou nulou
- Báze posunutí exponentu je $\frac{z^n}{2} - 1$ **(**z = 2, a=8) tj. báze posunutí je 2^7-1=**127**

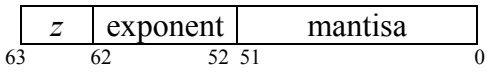
Zobrazení některých hodnot:

Nula zobrazena s obrazem mantisy i exponentu rovným nule (podle hodnoty znaménkového bitu – kladná/záporná nula=>nula má dvě možná zobrazení v kódu IEEE 754)

Nekonečno exponent =128, na hodnotě mantisy nezáleží.(podle hodnoty znaménkového bitu – kladné/záporné nekonečno).

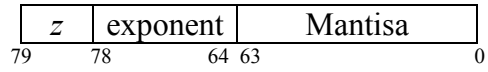
Nenormovaný tvar má hodnotu exponentu nula a nenulovou mantisu. Číslo je uloženo ve čtyřech po sobě jdoucích slabikách.

Zobrazení reálného čísla ve dvojnásobné přesnosti:



Mantisa je uložena na 52 bitech v přímém kódu se znaménkem¨.
Exponent je uložen na 11 bitech v kódu s posunutou nulou s bází posunutí 1023.

Zobrazení reálného čísla v rozšířeném tvaru:



Mantisa je uložena na 64 bitech v přímém kódu se znaménkem¨.
Exponent je uložen na 15 bitech v kódu s posunutou nulou s bází posunutí 16383.

Rozsah zobrazení čísel ve výše uvedených formátech:

Přesnost	Minimum	Maximum
Jednoduchá	$\pm 1.175 \cdot 10^{-38}$	$\pm 3.4 \cdot 10^{38}$
Dvojnásobná	$\pm 2.23 \cdot 10^{-308}$	$\pm 1.8 \cdot 10^{308}$
Rozšířená	$\pm 2 \cdot 10^{-16382}$	$\pm 2 \cdot 10^{16384}$

Př. Zobrazte ve formátu IEEE (na 4 bytech) následující reálná čísla:
a) -258,125 b) 69,1875 c) -0,453125
Výsledek zapište v šestnáctkové soustavě.

Ad a)

$(258)_{10}=(100000010)_2$

$0,125 \cdot 2 = 0,25$ **0**
$0,25 \cdot 2 = 0,5$ **0**
$0,5 \cdot 2 = 1,0$ **1**

$(0,125)_{10}=(0,001)_2$

$(258,125)_{10}=(100000010,001)_2$

norm. tvar: $1,00000010001*2^8$
exp.: $2^7-1+8=2^7+7=10000000+111=$**$(10000111)_{PN}$**

$(258,125)_{10}$= (1100 0011 1000 0001 0001 0000 0000 0000)$_{IEEE}$ =
= (C 3 8 1 1 0 0 0)$_{16}$

Ad b)

$(69)_{10}=(1000101)_2$

$0,1875 \cdot 2 = 0,375$ **0**
$0,375 \cdot 2 = 0,75$ **0**
$0,75 \cdot 2 = 1,5$ **1**
$0,5 \cdot 2 = 1$ **1**

$(0,1875)_{10}=(0,0011)_2$

$(69,1875)_{10}=(1000101,0011)_2$

norm. tvar: $1,0001010011*2^6$
exp.: $2^7-1+6=2^7+5=10000000+101=$**$(10000101)_{PN}$**
$(69,1875)_{10}$= (0100 0010 1000 1010 0110 0000 0000 0000)$_{IEEE}$ =
= (4 2 8 A 6 0 0 0)$_{16}$

Ad c)

$0,453125 \cdot 2 = 0,90625$ **0**
$0,90625 \cdot 2 = 1,8125$ **1**
$0,8125 \cdot 2 = 1,625$ **1**
$0,625 \cdot 2 = 1,25$ **1**
$0,25 \cdot 2 = 0,5$ **0**
$0,5 \cdot 2 = 1$ **1**

$(0,453125)_{10}=(0,011101)_2$

norm. tvar: $1,1101*2^{-2}$
exp.: $2^7-1-2=2^7-3=$**$(01111101)_{PN}$**

$(0,453125)_{10}$=(1011 1110 1110 1000 0000 0000 0000 0000)$_{IEEE}$ =
= (B E E 8 0 0 0 0)$_{16}$

Příklad k procvičení:
Zobrazte ve formátu IEEE (na 4 bytech):
$(-259,5)_{10}$
výsledek: $(1|100\ 0011\ 1|000\ 0001\ 1100\ 0000\ 0000\ 0000)_{IEEE}$

Operace nad celými čísly

Mezi základní celočíselné operace patří:
- sčítání
- odčítání
- násobení
- celočíselné dělení
- určení zbytku celočíselného dělení

Celočíselné dělení
- zavedeme operátor **div**
- pro celá čísla a,b ($b \neq 0$) platí $a \textbf{ div } b = sign\left(\frac{a}{b}\right) \cdot \left\lfloor abs\left(\frac{a}{b}\right)\right\rfloor$

kde
sign znaménko výsledku
abs absolutní hodnota výsledku
$\lfloor x \rfloor$ maximální celé číslo menší nebo rovno x

Určení zbytku celočíselného dělení
- zavedeme operátor **mod**
- pro celá čísla a,b ($b \neq 0$) platí $a \textbf{ mod } b = a - (a \textbf{ div } b) \cdot b$

Počet míst na která čísla v PC zobrazujeme je konečný. Množina zobrazitelných čísel je proto také konečná a výsledky celočíselných operací nemusí být zobrazitelné. Při nevýhodně zvoleném sledu operací může dojít k významné chybě.

1.Problém přetečení
Přetečení – překročení rozsahu zobrazitelného čísla (při operacích + , - , * , /)

- U operace součinu zobrazujeme výsledek na dvojnásobný počet míst
- U operace dělení může dojít k přetečení pouze při dělení nulou.
- Přetečení je technickým vybavením počítače indikováno jako chyba (reakce systému nastavení příznaku přetečení, nebo vyvolání přerušení).
- K přetečení dochází také nevhodným pořadím provádění operací, kdy výsledek je sice menší než maximálně možné zobrazitelné číslo(N_{MAX}), ale mezivýsledek N_{MAX} převyšuje.
- Ze základních zákonů aritmetiky platí : komutativní zákon
  neplatí: asociativní a distributivní zákon

Př. soustava ve které lze v desítkové soustavě zobrazit číslo v rozsahu ± 999
výpočet výrazu (900+500)-800=<u>1400</u>-800 proběhne chybně
900+(500-800)=900-300=600 proběhne správně.

2.Problém zaokrouhlení
Náhodným řazením operací násobení a dělení může dojít vlivem zaokrouhlení k výrazným chybám.
např. výraz (40·80) div 64 = 3200 div 64 = **50**
změníme-li pořadí operací (40 div 64) · 80= 0 · 80= **0**

Operace nad reálnými čísly

Číslo zobrazené v pohyblivé řádové čárce nemusí být zobrazeno přesně
- nepřesnost převodu mezi soustavami (např. 0,1 v binární soustavě periodické)
- omezený počet bitů mantisy

Mezivýsledky operací se musí většinou aproximovat. Aby se aproximace projevila v konečném výsledku co nejméně provádí se výpočty na větší počet platných míst než na který se výsledek nakonec zobrazí.Aproximace se provádí odseknutím přebývajících bitů nebo zaokrouhlením čísla.
Při aproximaci odseknutím lze absolutní chybu výsledku vyjádřit vztahem $\delta = z^E \cdot z^{-n}$
kde δ absolutní chyba
E exponent
n počet míst zobrazení mantisy

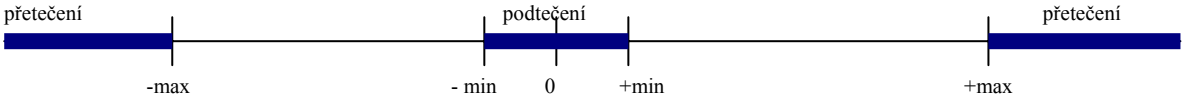
Při aproximaci zaokrouhlení je chyba poloviční.

Porovnání dvou reálných čísel
podmínku if (a = b) nahradíme podmínkou if abs(a-b)< δ
kde δ absolutní chyba porovnání

Problémy při provádění aritmetických operací

1. Přetečení a podtečení čísla
Přetečení i podtečení čísla se týká exponentu.

Je-li hodnota exponentu čísla **větší než maximální** zobrazitelná dochází k **přetečení**
Je-li hodnota exponentu čísla **menší než minimální** zobrazitelná dochází k **podtečení**



Např.
pro zobrazení čísla ve formátu IEEE v pohyblivé řádové čárce s jednoduchou přesností a při zachování normovaného tvaru platí:

$max = 2^{128}$ $\quad min = 2^{-126}$
$-max = -2^{128}$ $\quad -min = -2^{-126}$

Nejmenší zobrazitelné číslo v nenormovaném tvaru je $1{,}4 \cdot 10^{-45}$

2. Zobrazení nuly

Způsob zobrazení nuly má vliv na provádění operací sčítání a násobení nulou. Nula se zobrazuje jako $0 \cdot z^{-min}$, kde „– min“ je minimální zobrazitelná hodnota exponentu.

V případě sčítání dvou čísel dochází nejprve k úpravě čísla s menším exponentem. Jeho exponent se zvětšuje na hodnotu exponentu druhého sčítance a mantisa se zmenšuje tak, aby hodnota čísla byla zachována. Při zmenšování mantisy může dojít k jejímu zaokrouhlení a tím k nepřesnosti zobrazení sčítance. Zobrazíme-li nulu s minimálním exponentem, nebude se nenulový sčítanec upravovat a proto bude vždy platit **a + 0 = a**.

Př. $2^{-10} + 0 = 2^{-10} + 2^{-126} = 2^{-10} + 0 \cdot 2^{-10} = 2^{-10}$

Při násobení dvou čísel se provede součin mantis a součet exponentů. Při násobení nulou, která je vyjádřena navrhovaným způsobem může dojít k překročení rozsahu zobrazení exponentu – k podtečení. Proto algoritmy násobení musí tento případ odlišit a přímo vygenerovat nulový výsledek.
Jestliže případ násobení neodlišíme, může probíhat výpočet následovně.

Př. $2^{-10} \cdot 0 = 2^{-10} \cdot 2^{-126} = 2^{-10} + 0 \cdot 2^{-126} = 2^{-136}$

kde číslo 2^{-136} nemusí být v rozsahu zobrazení exponentu.

3. Neplatnost distributivního a asociativního zákona

Při vyhodnocování aritmetických výrazů záleží na pořadí provádění operací.

Neplatí asociativní zákon pro sčítání : $a + (b + c) \neq (a + b) + c$
Neplatí asociativní zákon pro násobení : $a \cdot (b \cdot c) \neq (a \cdot b) \cdot c$

Např. při zobrazování čísel na tři platná místa a zaokrouhlování mezivýsledků na 3 platná čísla platí:

$a \cdot (b \cdot c) = 0{,}86 \cdot (0{,}56 \cdot 0{,}08) = 0{,}86 \cdot 0{,}0448 =$ **0,0385**

$(a \cdot b) \cdot c = (0{,}86 \cdot 0{,}56) \cdot 0{,}08 = 0{,}482 \cdot 0{,}08 =$ **0,0365**