3. Číselné soustavy, vztahy mezi číselnými soustavami, zobrazení čísel v počítači, principy provádění aritmetických operací.

Číselné soustavy

je způsob reprezentace čísel. Podle způsobu určení hodnoty čísla z dané reprezentace rozlišujeme dva hlavní druhy číselných soustav: poziční číselné soustavy a nepoziční číselné soustavy. V praxi se však také používaly způsoby reprezentace používající postupy z obou těchto druhů. Dnes se obvykle používají soustavy poziční.

Nepoziční číselná soustava

je způsob reprezentace čísel, ve kterém není hodnota číslice dána jejím umístěním v dané sekvenci číslic. Tyto způsoby zápisu čísel se dnes již téměř nepoužívají a jsou považovány za zastaralé.
V nejjednodušším systému stačí sečíst hodnoty jednotlivých číslic. Pokud by například byly hodnoty symbolů následující: A = 1, B = 10, C = 100, D = 1000, pak by vyjádřením čísla 3542 mohl být například řetězec „AABBBBCCCCCDDD“.
Dalším příkladem nepoziční číselné soustavy je počítání na prstech.

Poziční číselné soustavy

je dnes převládající způsob písemné reprezentace čísel – dokonce, pokud se dnes mluví o číselných soustavách, jsou tím obvykle myšleny soustavy poziční. V tomto způsobu zápisu čísel je hodnota každé číslice dána její pozicí v sekvenci symbolů. Každá číslice má touto pozicí dánu svou váhu pro výpočet celkové hodnoty čísla.

Polyadické soustavy

Polyadické soustavy jsou speciálním případem pozičních soustav.

Základ – počet symbolů pro číslice používaných v dané soustavě
Řád -– váha číslice

zápis
číslo = součet mocnin základu vynásobených číslicemi
$A = a_n \cdot z^n + a_{n-1} \cdot z^{n-1} + \cdot \cdot \cdot + a_1 \cdot z^1 + a_0 \cdot z^0$
$A = 1 \cdot 10^2 + 2 \cdot 10^1 + 3 \cdot 10^0$

zhuštěný zápis
běžná je forma zhuštěného zápisu:
$A = a_n a_{n-1} \ldots a_1 a_0$
A = 123
$A = 123_{10}$

• Zobecnění pro racionální číslo (zavedeme záporné mocniny): $A = a_n \cdot z^n + \cdot \cdot \cdot + a_0 \cdot z^0 + a_{-1} \cdot z^{-1} + a_{-2} \cdot z^{-2} + \cdot \cdot \cdot + a_{-m} \cdot z^{-m}$
• Zobecnění pro záporná čísla – přidáním znaménka (pro počítače nevhodné)
• Zobecnění pro komplexní čísla – zavedením imaginární jednotky

Příklad číselné soustavy se základem 2 (tj. dvě číslice 1,0) a výpočet jeho hodnoty:

$\mathbf{(10010)_2 = 0 \cdot 2^0 + 1 \cdot 2^1 + 0 \cdot 2^2 + 0 \cdot 2^3 + 1 \cdot 2^4}$

Soustavy užívané v počítačové praxi

Dvojková (Binární) - z = 2; obsahuje číslice: 0, 1
Osmičková (Oktalová) - z = 8; obsahuje číslice: 0, 1, 2, 3, 4, 5, 6, 7
Šestnáctková (Hexadecimální) - z = 16; obsahuje číslice: 0, 1, . . . 9, A, . . . , F

Převody/Vztahy mezi číselnými soustavami

Jednoduše lze převádět mezi soustavami, kde **základ jedné soustavy tvoří základ mocniny druhé soustavy**. Např. mezi dvojkovou a osmičkovou, a dvojkovou a šestnáctkovou.
Tj. jednoduše lze převádět soustavy o základu z^k a z, každou k-tici číslic nižší soustavy nahradíme jednou číslicí soustavy vyšší.

2 → 8

Např. pro číslo 111010110_2.

Vytvoří se pomocná tabulka:

0	000
1	001
2	010
3	110
4	100
5	101
6	110
7	111

A s ní se číslo převede **111 ~ 7 | 010 ~ 2 | 110 ~ 6 == 726_8**

2 → 16

Např. pro číslo 111001011101_2.

Vytvoří se pomocná tabulka:

0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011

C	1100
D	1101
E	1110
F	1111

A s ní se číslo převede **1110 ~ E | 0101 ~ 5 | 1101 ~ D == $\mathbf{E5D_{16}}$**

8 → 2, 16 → 2

Číslo se jednoduše převede podle zmíněných tabulek po jednotlivých pozicích (číslicích).

2 → 10

Tady je to o něco složitější. Číslo lze převést jako **součet mocnin dvojek např**.

100110_2 se převede jako $0.2^0 + 1.2^1 + 1.2^2 + 0.2^3 + 0.2^4 + 1.2^5 = 2 + 4 + 32 = 38_{10}$

10 → 2

Obrácený převod se provede jako dělení se zbytkem:

38	:	2	=	19	0
19	:	2	=	9	1
9	:	2	=	4	1
4	:	2	=	2	0
2	:	2	=	1	0
1	:	2	=	0	1

Výsledek se čte odspoda (pozpátku): 100110_2

Desetinná čísla

Rozdělíme na celou a desetinnou část, převedeme odděleně. Desetinná část se nedělí 2, ale násobí 2 a výsledek se čte v přímém pořadí (NE obráceném).

- Nejvyšší významový bit lze nalézt vynásobením desetinného čísla základem
- Každá číslice je počítána postupným (iteračním) násobením desetinného výsledku předchozí iterace základem soustavy, do které chceme číslo převést

- Po každém iteračním kroku se sepisuje celočíselná část výsledku a převod končí, když je výsledek násobení roven nule nebo nedosáhneme požadované přesnosti

$1100{,}0011001100_2 = ?_{10}$
Celá část:
$1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0 = 1 \cdot 8 + 1 \cdot 4 + 0 \cdot 2 + 0 \cdot 1 = 12$
Desetinná část:
$0 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3} + 1 \cdot 2^{-4} + 0 \cdot 2^{-5} + 0 \cdot 2^{-6} + 1 \cdot 2^{-7} + 1 \cdot 2^{-8} + \ldots = 0 \cdot 0{,}5 + 0 \cdot 0{,}25 + 1 \cdot 0{,}125 + 1 \cdot 0{,}0625 + \ldots = 0{,}19999\ldots$
Rešení: zaokrouhlení dle poslední číslice rozvoje.

- **Příklad převodu desetinného čísla:**

$$(0{,}6875)_{10} \cdot 2 = 1{,}375 = 1 + 0{,}375 = b_{-1} + 0{,}375 \ (MSB)$$
$$(0{,}375)_{10} \cdot 2 = 0{,}75 = 0 + 0{,}75 = b_{-2} + 0{,}75$$
$$(0{,}75)_{10} \cdot 2 = 1{,}5 = 1 + 0{,}5 = b_{-3} + 0{,}5$$
$$(0{,}5)_{10} \cdot 2 = 1{,}0 = 1 + 0{,}0 = b_{-4} + 0{,}0 \ (LSB)$$
$$(0{,}6875)_{10} = (0{,}1011)_2$$

Zobrazení čísel v počítači

Čísla se mohou v paměti počítače v binárním tvaru zobrazovat buď jako:

- **Little endian** – LSB je na nejnižší adrese.
- **Big endian** – MSB je na nejnižší adrese.

Zobrazení kladných čísel

U kladných čísel buď může nebo nemusí být přítomen tzv. znaménkový bit.

Rozsah pro zobrazení celých kladných čísel je <0, 2^n-1>, n – počet bitů.

Pro 16-bitový integer je to (-32768, 32767)

V případě použití znaménkového bitu je rozsah kladných čísel ovlivněný počtem bitů o jeden méně.

Zobrazení záporných čísel

- **Přímý kód** – jen přidání znaménkového bitu, nevýhodou je záporná nula
- **Inverzní kód** – obrácení jedniček a nul, stále záporná nula
- **Dvojkový doplňkový kód** – obrácení jedniček a nul + přičtení jedničky; rozsah je <-2^n-1, 2^n-1 – 1>
- **Kód s posunutou nulou -** V kódu s posunutou nulou využíváme bázi posunutí (standardně 27 - 1). K této bázi přičteme požadované číslo a výsledek zobrazíme. PRIKLAD: číslo (55) zobrazte na 8 bitů: 2^7 - 1 +55 = 128 - 1 + 55 = 182 = =(10110110) analogicky záporné číslo (-55) na 8 bitů: 2^7 - 1 -55 = 128 - 1 - 55 = 72 = (1001000)

Zobrazení reálných čísel

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

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.

Struktura čísla:

znaménkový bit (1 b) exponent (8 b) mantisa (23 b)

Znaménkový bit

- kladné číslo má znaménkový bit nulový, u záporného čísla je tento bit jedničkový

Exponent

- je uložen na 8 bitech v kódu s posunutou nulou, báze posunutí je $2^7 - 1 = 127$
- příklad: exponent 4_{10} je uložen jako $4 + 127 = 131 = 1000\ 0011_2$

Mantisa

- je znázorněna 24 bity v přímém kódu
- první bit (na nulté desetinné pozici) se neukládá, bere se implicitně jako 1 (mantisa se tedy ukládá počínaje druhým bitem)
- myšlená desetinná tečka je umístěna za nejvyšším bitem mantisy
- příklad: mantisa $1{,}5625_{10}$ je desetinné číslo $1 + 0{,}5 + 0{,}0625 = 2^0 + 2^{-1} + 2^{-4} = 1{,}1001_2$ a uloží se jako 100 1000 0000 0000 0000 0000

Příklad
Zobrazte číslo $(-258{,}125)_{10}$ ve formátu IEEE (na 4 bytech):

$(258)_{10} = (100000010)_2$
$0{,}125 \cdot 2 = 0{,}25\ \mathbf{0}$
$0{,}25 \cdot 2 = 0{,}5\ \mathbf{0}$
$0{,}5 \cdot 2 = 1{,}0\ \mathbf{1}$
$(0{,}125)_{10} = (0{,}001)_2$

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

Nyní je nutné provést normalizaci – pomocí násobku čísla s mocninami dvojky číslo převést do intervalu [1, 2]:

nenormalizované = normalizované $\cdot\ 2^n$, kde n je hledaná mocnina.

norm. tvar: $1{,}00000010001 \cdot 2^8$
exponent: $2^7 - 1 + 8 = 2^7 + 7 = 10000000 + 111 = (10000111)$

Jelikož se předpokládá normální tvar, jednotka na místě před desetinou čárkou se nezapisuje:

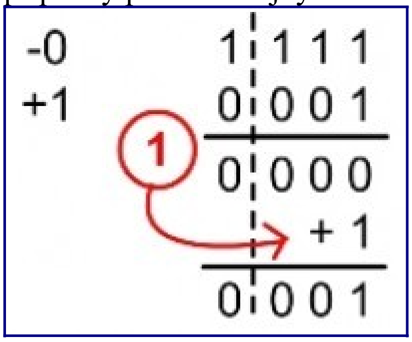
$(-258{,}125)_{10}$ = (1100 0011 1000 0001 0001 0000 0000 0000)$_{IEEE}$

Aritmetické operace

Přetečení – výsledek operace spadá mimo rozsah zobrazení (přenos z nejvyššího bitu)

Součet v doplňkovém kódu – všechny bity se sčítají stejně, včetně znaménkového; přenos ze znaménkového bitu se ignoruje; přetečení nastane, pokud se přenos do znaménkového bitu nerovná přenosu ze znaménkového bitu.

Součet v inverzním kódu – problém dvou nul, nutno provádět tzv. kruhový přenos, tj. případný přenos z nejvyššího řádu přičíst k výsledku.



Hradlo XOR
Hradlo XOR je jedním ze základních kombinačních logických obvodů, jehož výstup je exkluzívní logický součet vstupů („buď A nebo B“). Výstup je log.1 tehdy a jen tehdy pokud se hodnoty vstupů liší.