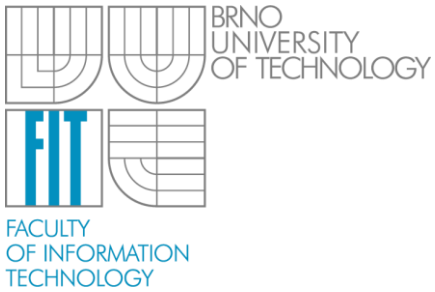
Návrh číslicových systémů (INC)

Otto Fučík

Vysoké učení technické v Brně
Fakulta informačních technologií
Božetěchova 2, 612 66 Brno



Použitá literatura

N. Frištacký, M. Kolesár, J. Kolenička a J. Hlavatý: „Logické systémy", SNTL Praha, 1986
M. Eysselt: „Logické systémy", SNTL Praha, skriptum VUT v Brně, 1985
J. F. Wakerly: „Digital Design. Principles and Practices", Prentice Hall, ISBN 0-13-769191-2, 2000
V. P. Nelson, H.T.Nagle, B.D.Carroll, J.D.Irwin: „Digital Logic Circuit Analysis & Design", ISBN 0-13-463894-8, 1995
T.L.Floyd: „Digital Fundamentals", Prentice Hall, ISBN 0-13-080850-4, 2000

Binární čísla

Číselné soustavy

- Číselné soustavy
  - Čísla se skládají z uspořádané množiny symbolů (číslice)
  - Relace mezi čísly definované pro jednotlivé aritmetické operace (sčítání, odčítání, násobení, dělení, atd.)
- Základ či báze (anglicky radix) číselné soustavy definuje
  - Maximální počet číslic, které máme v dané soustavě k dispozici
- Mezi číselné soustavy nejčastěji používané patří
  - Soustava desítková (dekadická, r = 10)
  - Dvojková (binární, r = 2)
  - Osmičková (oktalová, r = 8)
  - Šestnáctková (hexadecimální, r = 16)
- Každé číslo vyjádřené v těchto soustavách může mít
  - Část celočíselnou (před desetinnou čárkou, či tečkou v anglosaských zemích)
  - Desetinnou část (za desetinnou čárkou)
- Uvedené soustavy řadíme mezi polyadické
  - Číslo se vyjadřuje součtem mocnin základu dané soustavy, které jsou vynásobeny příslušnými platnými číslicemi

Poziční zápis

- V pozičním zápisu
  - Představuje pozice každé číslice v daném čísle její relativní váhu významnosti
  - Obecně platí, že kladné číslo může být zapsáno pozičně následovně:

$$N = (n_{k-1}n_{k-2}\ldots n_1 n_0, n_{-1}n_{-2}\ldots n_{-l})_r$$

desetinná čárka odděluje celou a desetinnou část čísla

r	základ dané číselné soustavy
k	počet číslic celočíselné části
l	počet číslic desetinné části
n	číslice
n_{k-1}	nejvyšší významová číslice
n_{-l}	nejnižší významová číslice

- Příklad
  - Ludolfovo číslo lze, v číselné soustavě o základu 10 a s přesností na 2 desetinná čísla, zapsat pozičně následovně:

$$\pi = (3{,}14)_{10}$$

  - Pozn.: pokud nemůže dojít k záměně, nebudeme závorky a označení základu uvádět

Polynomiální zápis

- Obecně lze číslo v číselné soustavě o základu *r* zapsat polynomem následovně

$$N = \sum_{i=-l}^{k-1} n_i \cdot r^i$$

$$= n_{k-1}r^{k-1} + n_{k-2}r^{k-2} + \ldots n_0 r^0 + n_{-1}r^{-1} + \ldots + n_{-l+1}r^{-l+1} + n_{-l}r^{-l}$$

  - Každá číslice je zde vynásobena vahou, která je dána její pozicí a vyjádřena mocninou o základu *r*
- Příklad
  - Ludolfovo číslo lze, v číselné soustavě o základu 10 a s přesností na 2 desetinná čísla, zapsat polynomem následovně:

$$\pi = (3 \cdot 10^0 + 1 \cdot 10^{-1} + 4 \cdot 10^{-2})_{10}$$

$$= (3 \cdot 1 + 1 \cdot 0{,}1 + 4 \cdot 0{,}01)_{10}$$

Desítková soustava

- V desítkové číselné soustavě je základ r = 10
  - Máme tedy k dispozici deset číslic (0 až 9) pro vyjádření všech čísel
- Příklad
  - Dekadické číslo 327,56 lze zapsat jak pozičně, tak polynomem

$$
\begin{aligned}
&(327{,}56)_{10} \\
&= (3 \cdot 10^2 + 2 \cdot 10^1 + 7 \cdot 10^0 + 5 \cdot 10^{-1} + 6 \cdot 10^{-2})_{10} \\
&= (3 \cdot 100 + 2 \cdot 10 + 7 \cdot 1 + 5 \cdot 0{,}1 + 6 \cdot 0{,}01)_{10} \\
&= (300 + 20 + 7 + 0{,}5 + 0{,}06)_{10}
\end{aligned}
$$

Dvojková soustava

- V binární číselné soustavě je základ r = 2
  - Což znamená, že pro vyjádření čísel máme k dispozici pouze dvě číslice 0 a 1
- Příklad - binární číslo zapsaného pozičně a polynomem:

$$(10011{,}011)_2$$

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

- Jednotlivé binární číslice
  - Nazýváme bity (zkratka "b") a můžeme je sdružovat do skupin, se kterými se pak lépe pracuje
- Nejčastěji používané skupiny jsou
  - Čtveřice bitů (např. 1011) nazývané anglicky „nibble"
  - Osmice bitů (např. 10101010) nazývaná bajt (anglicky byte) a má zkratu "B"
- 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

Příklad: Prvních 16 celých binárních čísel

- Tabulka

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

Binární čísla

- Čísla vyjádřená způsobem uvedeným v předchozí tabulce
  - Jsou zobrazena v tzv. přirozeném (přímém) kódu, ve kterém hodnota binárního čísla přímo odpovídá hodnotě ekvivalentního čísla v jiné soustavě (nejčastěji desítkové)
- V oboru počítačů se v počtu bajtů
  - Vyjadřuje též např. kapacita pamětí, ovšem ne dekadicky, ale pomocí mocnin dvou, např.:

$$2^0, 2^1, 2^2, 2^3, 2^4, 2^5, 2^6, 2^7, 2^8, 2^9, 2^{10}, 2^{11}, \ldots$$
$$= 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048 \ldots$$

  - Hodnota 1024 bitů = kilobajt (anglicky kilobyte) a značí se kB
  - Tisíc kilobajtů = megabajt (MB) a tisíc megabajtů je gigabajt (GB)
- Počet bitů *k* binárního čísla určuje rozsah hodnot *H*, kterých může nabývat:

$$H = 2^k$$

Osmičková a šestnáctková soustava

- Osmičková (oktalová) soustava má základ r = 8
  - Pracuje se zde s osmi číslicemi 0, 1, 2, 3, 4, 5, 6 a 7
  - Příklad oktalového čísla zapsaného pozičně a polynomem:

$$(1234)_8$$
$$= (1 \cdot 8^3 + 2 \cdot 8^2 + 3 \cdot 8^1 + 4 \cdot 8^0)_8$$

- Šestnáctková (hexadecimální) čísla mají základ r = 16
  - Existuje tedy 16 hexadecimálních číslic 0 až 9, A, B, C, D, E a F
  - Příklad hexadecimálního čísla zapsaného pozičně a polynomem

$$(ABCD)_{16}$$
$$= (A \cdot 16^3 + B \cdot 16^2 + C \cdot 16^1 + D \cdot 16^0)_{16}$$

- Čísla oktalová i hexadecimální jsou výhodná
  - Pro zobrazování vícebitových binárních čísel, neboť jejich základ je mocninou dvou (binární číslice sdružujeme do trojic resp. čtveřic)

$$(11{,}01011100\,10)_2 = (0011{,}0101\,1100\,1000)_2 = (3{,}5C8)_{16}$$

Příklad: Čísla vyjádřená v různých soustavách

- Tabulka

Dekadické	*Binární číslo*	*Oktalové*	*Hexadecimální číslo*
0	00000000	0	0
1	00000001	1	1
2	00000010	2	2
3	00000011	3	3
4	00000100	4	4
5	00000101	5	5
6	00000110	6	6
7	00000111	7	7
8	00001000	10	8
9	00001001	11	9
10	00001010	12	A
11	00001011	13	B
12	00001100	14	C
13	00001101	15	D
14	00001110	16	E
15	00001111	17	F
16	00010000	20	10
...	...	...	...
31	00011111	37	1F
32	00100000	40	20
...	...	...	...
63	00111111	77	3F
64	01000000	100	40
...	...	...	...
127	01111111	177	7F
128	10000000	200	80
...	...	...	...
254	11111110	376	FE
255	11111111	377	FF

Převody mezi číselnými soustavami

- Substituční metoda
  - Zápis čísel polynomem je základem pro substituční metodu převodu, podle které se číslo N zapsané v soustavě A převedeno do soustavy B v následujících dvou krocích:
    - 1. číslo se vyjádří polynomem v soustavě A
    - 2. výpočet se provede aritmetikou soustavy B
- Převod čísel z různých soustav do soustavy dekadické
  - Nejprve se číslo, které chceme převést, vyjádří v původní číselné soustavě polynomem
  - Následně se, již v aritmetice cílové soustavy, spočtou mocniny čísel; každá mocnina se vynásobí hodnotou příslušné číslice a vše se sečte
- Příklad převodu čísla binárního na dekadické:

$$(10011{,}011)_2$$
$$= (1\cdot 2^4 + 0\cdot 2^3 + 0\cdot 2^2 + 1\cdot 2^1 + 1\cdot 2^0 + 0\cdot 2^{-1} + 1\cdot 2^{-2} + 1\cdot 2^{-3})_2$$
$$= (16 + 2 + 1 + 0{,}25 + 0{,}125)_{10}$$
$$= (19{,}375)_{10}$$

- Příklad převodu čísla hexadecimálního na dekadické:

$$
\begin{aligned}
&(ABCD)_{16} \\
&= (A \cdot 16^3 + B \cdot 16^2 + C \cdot 16^1 + D \cdot 16^0)_{16} \\
&= (10 \cdot 4096 + 11 \cdot 256 + 12 \cdot 16 + 13 \cdot 1)_{10} \\
&= (43981)_{10}
\end{aligned}
$$

- Příklad převodu čísla oktalového na dekadické:

$$
\begin{aligned}
&(1234)_8 \\
&= (1 \cdot 8^3 + 2 \cdot 8^2 + 3 \cdot 8^1 + 4 \cdot 8^0)_8 \\
&= (1 \cdot 512 + 2 \cdot 64 + 3 \cdot 8 + 4 \cdot 1)_{10} \\
&= (668)_{10}
\end{aligned}
$$

Substituční metoda

- Zkusme však spočítat převod čísla dekadického např. na oktalové:

$$
\begin{aligned}
&(1234)_{10} \\
&= (1 \cdot 10^3 + 2 \cdot 10^2 + 3 \cdot 10^1 + 4 \cdot 10^0)_{10} \\
&= (1 \cdot ? + 2 \cdot ? + 3 \cdot ? + 4 \cdot ?)_8 \\
&= (???)_8
\end{aligned}
$$

- Vidíme, že
  - Vyjádření hodnot mocnin deseti aritmetikou oktalové soustavy může být (pro člověka) obtížné provést přímo
- Substituční metoda je tedy vhodná
  - Pro vzájemné převody mezi soustavou binární, hexadecimální a oktalovou a též pro převod z těchto soustav do soustavy dekadické
- Substituční metoda se však příliš nehodí
  - Pro převod z dekadické do ostatních soustav

Metoda dělení základem (pro celá čísla)

- Tato metoda je určena
  - Pro převod celých (anglicky integer) čísel mezi soustavami
- Uvažujme např. převod čísla v jedné soustavě do ekvivalentního čísla v jiné soustavě
  - Zapišme si celé číslo A, vyjádřené na *i* platných číslic, polynomem následovně:

$$A = \sum_{i=0}^{m-1} a_i \cdot r_A^i = a_{m-1} \cdot r_A^{m-1} + a_{m-2} \cdot r_A^{m-2} + \ldots + a_0 \cdot r_A^0$$

- Vztah přepišme
  - Podle základu soustavy, do které chceme číslo převést tak, aby se základy vyskytovaly bez mocnin:

$$A = ((\ldots(b_{n-1}) \cdot r_B + b_{n-2}) \cdot r_B + \ldots + b_1) \cdot r_B + b_0$$

$$= b_{n-1} b_{n-2} \ldots . b_0 = B$$

- Výsledkem převodu je číslo, které má jednotlivé výsledné číslice zapsané pozičně

Metoda dělení základem (pro celá čísla)

- Příklad ilustrující popsaný převod celého dekadického čísla na binární:

$$
\begin{aligned}
&(109)_{10} \\
&= ((((((1 \cdot 2 + 1) \cdot 2 + 0) \cdot 2 + 1) \cdot 2 + 1) \cdot 2 + 0) \cdot 2 + 1 \\
&= 1 \cdot 2^6 + 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 \\
&= (1101101)_2
\end{aligned}
$$

Metoda dělení základem (pro celá čísla)

- Způsob, jak vyjádřit číslo v požadovaném tvaru
  - Nejnižší významový bit lze nalézt celočíselným vydělením čísla základem:

$$A/r_B = (a_{m-1} \cdot r_A^{m-1} + \ldots + a_1 \cdot r_A^1 + a_0 \cdot r_A^0)/r_B$$
$$= (q_{m-1} \cdot r_A^{m-2} + \ldots + q_1 \cdot r_A^0) + R_0$$

- Příklad převodu

$$(109)_{10}/2 = 54 \;\; zb.\; 1(LSB)$$
$$(54)_{10}/2 = 27 \;\; zb.\; 0$$
$$(27)_{10}/2 = 13 \;\; zb.\; 1$$
$$(13)_{10}/2 = 6 \;\; zb.\; 1$$
$$(6)_{10}/2 = 3 \;\; zb.\; 0$$
$$(3)_{10}/2 = 1 \;\; zb.\; 1$$
$$(1)_{10}/2 = 0 \;\; zb.\; 1(MSB)$$
$$(109)_{10} = (1101101)_2$$

Metoda násobení základem (pro desetinná čísla)

- Tato metoda je určena
  - Pro převod desetinných (anglicky fractional) čísel mezi soustavami
- Uvažujme např. převod desetinného čísla v jedné soustavě do ekvivalentního čísla v jiné soustavě
  - Zapišme si celé číslo vyjádřené na *m* platných číslic polynomem následovně:

$$A_F = \sum_{i=-1}^{-m} a_i \cdot r_A^i = a_{-1} \cdot r_A^{-1} + a_{-2} \cdot r_A^{-2} + \ldots + a_{-l-1} \cdot r_A^{-m-1} + a_{-m} \cdot r_A^{-m}$$

- Vztah přepišme
  - Podle základu soustavy, do které chceme číslo převést tak, aby se základy vyskytovaly bez mocnin:

$$A_F = r_B^{-1} \cdot (a_{-1} + r_B^{-1} \cdot (a_{-2} + \ldots + r_B^{-1} \cdot (a_{-n-1} + r_B^{-1} \cdot a_{-n})))$$

$$= 0, b_{-1} b_{-2} \ldots . b_{-n} = B_F$$

Metoda násobení základem (pro desetinná čísla)

- Výsledkem převodu je
  - Desetinné číslo, které má jednotlivé výsledné číslice zapsané pozičně
- Následující příklad ilustruje převod desetinného dekadického čísla na binární:

$$
\begin{aligned}
&(0{,}625)_{10} \\
&= 2^{-1} \cdot (1 + 2^{-1} \cdot (0 + 2^{-1} \cdot (1 + 2^{-1} \cdot 1))) \\
&= 1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3} \\
&= (0{,}101)_2
\end{aligned}
$$

Metoda násobení základem (pro desetinná čísla)

- Způsob, jak vyjádřit číslo v požadovaném tvaru je následující:
  - Nejvyšší významový bit lze nalézt vynásobením desetinného čísla základem:

$$A \cdot r_B = r_B \cdot (r_B^{-1} \cdot (a_{-1} + r_B^{-1} \cdot (a_{-2} + \ldots + r_B^{-1} \cdot (a_{-m-1} + r_B^{-1} \cdot a_{-m}))))$$

$$= r_B \cdot a_{-1} + r_B \cdot (r_B^{-1} \cdot (a_{-2} + \ldots + r_B^{-1} \cdot (a_{-m-1} + r_B^{-1} \cdot a_{-m})))$$

- Platí
  - 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. Končíme též, pokud je dosaženo požadované přesnosti

Metoda násobení základem (pro desetinná čísla)

- Příklad převodu desetinného čísla:

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

- Zkouška:

$$(0{,}1011)_2$$
$$= (1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3} + 1 \cdot 2^{-4})_2$$
$$= (0{,}5 + 0{,}125 + 0{,}0625)_{10}$$
$$= (0{,}6875)_{10}$$

Metoda násobení základem (pro desetinná čísla)

- Přesnost vyjádření desetinných čísel
  - Je dána počtem platných číslic
- Příklad převodu desetinného čísla na oktalové
  - Vidíme, že výsledek nemusí být vždy možno vyjádřit na konečném počtu číslic (přesnost bude vždy nižší, než v případě původního desetinného):

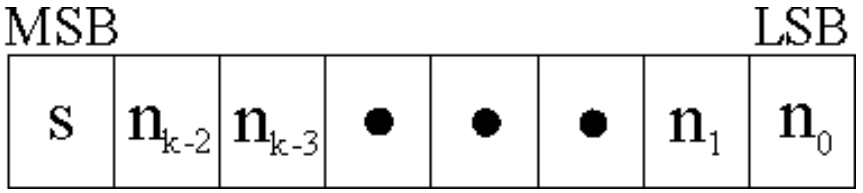
$$(0{,}1285)_{10} \cdot 8 = 1{,}028$$
$$(0{,}028)_{10} \cdot 8 = 0{,}224$$
$$(0{,}224)_{10} \cdot 8 = 1{,}792$$
$$(0{,}792)_{10} \cdot 8 = 6{,}336$$
$$(0{,}336)_{10} \cdot 8 = 2{,}688$$
$$(0{,}688)_{10} \cdot 8 = 5{,}504$$
$$\ldots$$
$$(0{,}1285)_{10} = (0{,}101625\ldots)_8$$

Převody mezi soustavami: Shrnutí

- Pro převod čísla ze soustavy A do soustavy B lze použít:
  - 1) metodu substituční při použití aritmetiky soustavy B
  - 2) pro čísla celá metodu dělení základem a pro čísla desetinná metodu násobení základem
- Uvedený postup lze použít obecně pro převod mezi libovolnými soustavami
- Pro člověka
  - Však může být obtížné pracovat s aritmetikou jiných soustav, než je desítková
  - V takovém případě lze provést
    - Nejdříve převod ze soustavy A do soustavy desítkové pomocí metody 1)
    - Následný převod z desítkové do cílové soustavy B pomocí metody 2)

Reprezentace čísel se znaménkem

- Omezíme se pouze na čísla binární
  - Doposud jsme uvažovali pouze čísla kladná – bez znaménka
  - 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, viz:

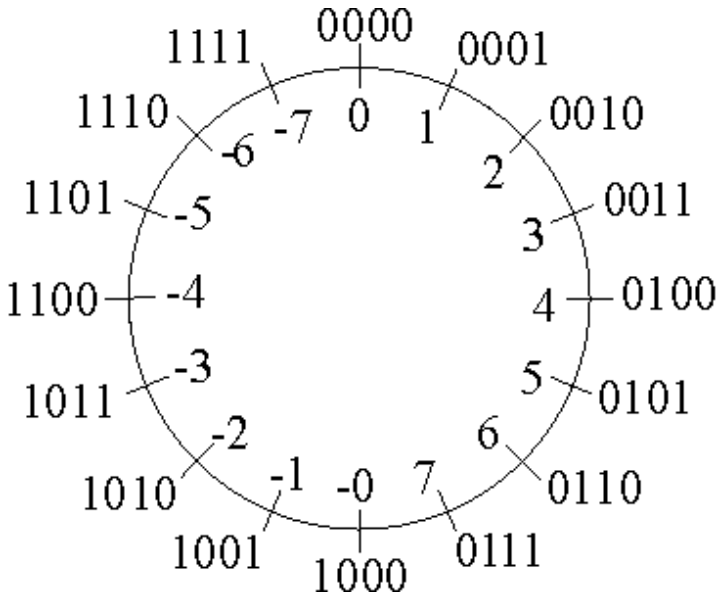


  - Čísla kladná mají ve znaménkovém bitu hodnou 0
  - Čísla záporná mají ve znaménkovém bitu hodnotu 1

Přirozený binární kód se znaménkem

- Přirozený (přímý) binární kód
  - Hodnota binárního čísla odpovídá převedené hodnotě čísla dekadického
  - Pokud chceme vyjádřit číslo záporné, prostě jej na místě MSB doplníme znaménkem
- Příklad
  - Binární číslo se znaménkem reprezentované v přirozeném kódu na čtyřech bitech

- Kód má dvě nuly – kladnou a zápornou
  - Tuto situaci je třeba vhodně ošetřit, což komplikuje počítání v tomto kódu



Doplňkové kódy

- Doplněk číselné soustavy o základu *r*
  - Anglicky radix-complement
  - Platí: $[N]_r = r^n - (N)_r$
  - Rozsah zobrazených čísel: $-r^{n-1}..r^{n-1}-1$
- Doplněk číselné soustavy o základu *r* snížený o 1
  - Anglicky diminished radix-complement
  - Doplněk lze též vyjádřit následovně:

$$[N]_r = r^n - (N)_r = ((r^n - 1) - (N)_r) + 1$$

  - Kde $(r^n - 1) - (N)_r$ je doplněk snížený o 1
  - Často se používá jako mezioperace při výpočtu doplňku přičtením jedničky

Desítkový doplněk

- Doplněk číselné soustavy o základu r = 10
- Příklad:
  - Mějme číslo 356 zobrazené na 3 číslicích
  - Určeme desítkový doplněk

$$[356]_{10} = 10^3 - (356)_{10} = 1000 - 356 = 644$$

- Pro hodnotu 0 platí

$$[0]_{10} = 10^3 - (0)_{10} = 1000 - 0 = 000 = 0$$

  - Na třech číslicích nelze zobrazit číslo 1000 (jedničku zahodíme) – doplňkem nuly je tedy opět nula
- Důležitý výsledek
  - Doplňkové zobrazení čísel nemá dvě nuly

Devítkový doplněk

- Postup výpočtu
  - Platí $(r^n - 1) - (N)_r$
    - Odčítáme vždy číslo menší od většího (nemůže dojít k výpůjčce z vyššího řádu)
    - Rozdíl můžeme počítat pro každou číslici zvlášť (jednoduchá realizace bez přenosů)
- Příklad: $[356]_{10} = ((10^3 - 1) - (356)_{10}) + 1$

$$= (999 - 356) + 1$$

$$= ((9-3)\cdot 10^2 + (9-5)\cdot 10^1 + (9-6)\cdot 10^0) + 1$$

$$= (600 + 40 + 3) + 1 = 643$$

- Desítkový doplněk spočteme přičtením jedničky k devítkovému doplňku

$$[356]_{10} = ([356]_{10} - 1) + 1 = 643 + 1 = 644$$

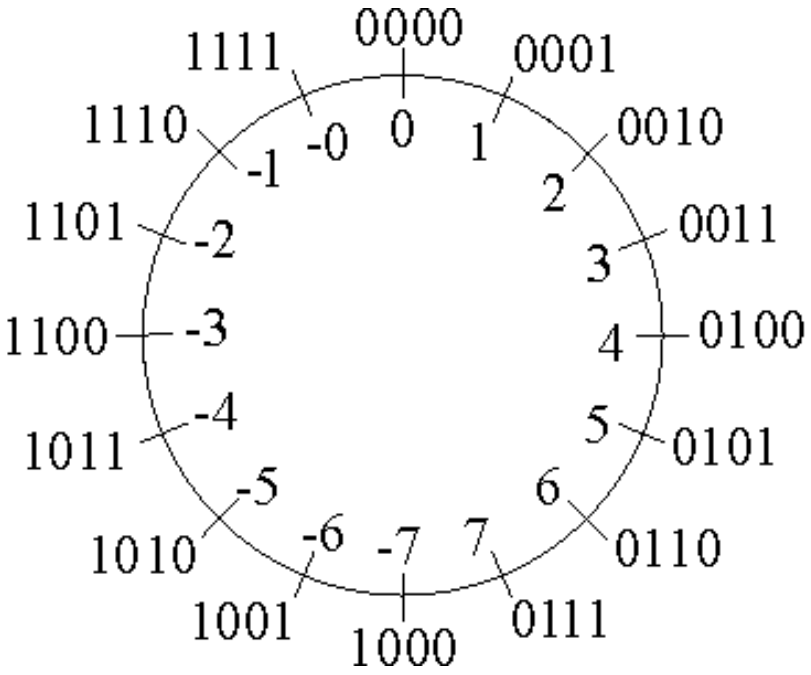
Jedničkový a dvojkový doplněk

- Jedničkový doplněk (anglicky one's complement)
  - Doplněk číselné soustavy o základu r = 2 snížený o 1
  - Lze spočítat prostou negací jednotlivých bitů daného čísla (komplement 1 = 0 a naopak) = odečítání jedničky resp. nuly od jedničky
- Dvojkový doplněk (anglicky two's complement)
  - Doplněk číselné soustavy o základu r = 2
  - Lze spočítat spočtením jedničkového doplňku daného čísla a přičtením jedničky
- Příklad:

$$
\begin{aligned}
[0101]_2 &= ((2^4 - 1) - 0101) + 1 \\
&= ((10000 - 1) - 0101) + 1 \\
&= (1111 - 0101) + 1 \\
&= ((1-0)\cdot 2^3 + (1-1)\cdot 2^2 + (1-0)\cdot 2^1 + (1-1)\cdot 2^0) + 1 \\
&= (1\cdot 2^3 + 0\cdot 2^2 + 1\cdot 2^1 + 0\cdot 2^0) + 1 \\
&= (1010) + 1 \\
&= 1011
\end{aligned}
$$

Jedničkový doplněk

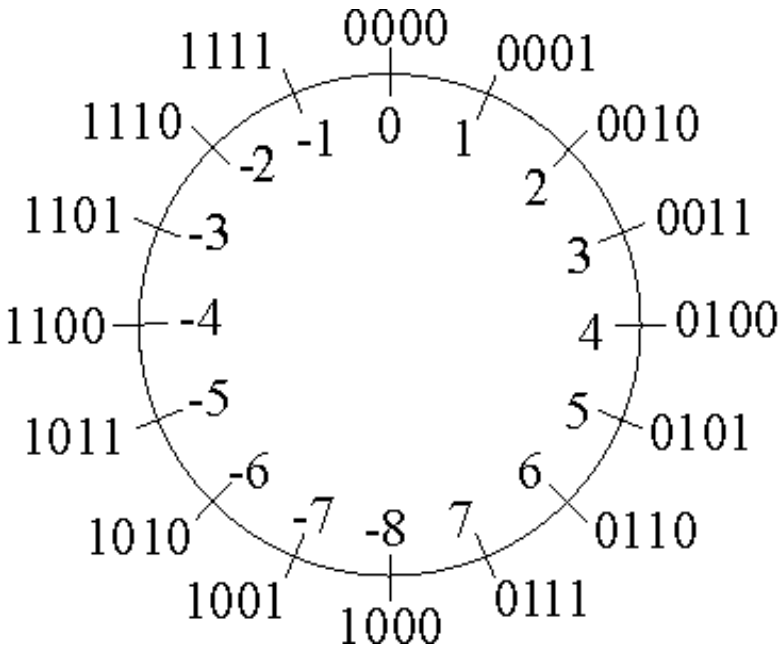
- Zobrazení čtyřbitových čísel
  - Kladná i záporná nula
  - Používá se převážně jako mezioperace při počítání dvojkového doplňku
- Příklad
  - Binární číslo se znaménkem reprezentované v jedničkovém doplňku na čtyřech bitech



Dvojkový doplněk

- Příklad - zobrazení čtyřbitových čísel
  - 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 relativně snadno realizovat logickými obvody
  - Představuje základ aritmetiky většiny číslicových systémů

- Příklad
  - Binární číslo se znaménkem reprezentované ve dvojkovém doplňku na čtyřech bitech



Binární aritmetika

- Aritmetické operace
  - Sčítání, odčítání, násobení a dělení
- Příklad - tabulka součtů dekadických čísel (složité)

+	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9	10
2	2	3	4	5	6	7	8	9	10	11
3	3	4	5	6	7	8	9	10	11	12
4	4	5	6	7	8	9	10	11	12	13
5	5	6	7	8	9	10	11	12	13	14
6	6	7	8	9	10	11	12	13	14	15
7	7	8	9	10	11	12	13	14	15	16
8	8	9	10	11	12	13	14	15	16	17
9	9	10	11	12	13	14	15	16	17	18

- Příklad - tabulka násobení dekadických čísel (složité)

x	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9
2	0	2	4	6	8	10	12	14	16	18
3	0	3	6	9	12	15	18	21	24	27
4	0	4	8	12	16	20	24	28	32	36
5	0	5	10	15	20	25	30	35	40	45
6	0	6	12	18	24	30	36	42	48	54
7	0	7	14	21	28	35	42	49	56	63
8	0	8	16	24	32	40	48	56	64	72
9	0	9	18	27	36	45	54	63	72	81

Sčítání a násobení binárních čísel

- V případě binárních čísel
  - Jsou definiční tabulky mnohem jednodušší, než pro jiné soustavy
  - Sčítání

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

Násobení

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

- Součet dvou jedniček
  - Dává hodnotu dva
  - V případě dekadických čísel musíme, v případě výsledku většího než devět, provádět tzv. přenos (anglicky Carry)
  - U binárních čísel je to stejné, hodnota (1)0 znamená výsledek nula s přenosem do vyššího řádu (do vyššího významového bitu)

Příklad součtu dvou osmibitových čísel

1	1	1		1				
	1	0	1	0	1	0	1	0
	1	1	1	0	1	1	0	0
1	1	0	0	1	0	1	1	0

- Popis
  - V horním řádku jsou znázorněny přenosy z nižších řádů
  - Pod čarou je výsledný součet
  - Výsledek má celkem 9 bitů

$$(10101010)_2 = (170)_{10}$$

$$(11101100)_2 = (236)_{10}$$

- Zkouška

$$(110010110)_2 = (406)_{10} = (170 + 236)_{10}$$

Sčítání binárních čísel se znaménkem

- Existují čtyři možnosti součtu dvou čísel se znaménkem
  - (+) + (+)
  - (-) + (+)
  - (+) + (-)
  - (-) + (-)
- Platí
  - Pokud máme k dispozici libovolný počet bitů, nemusíme tyto případy rozlišovat
  - Pokud máme k dispozici pevný počet bitů, musíme sledovat, zda u výsledku nedošlo k tzv. přetečení (přeplnění) rozsahu (anglicky overflow)
  - Přetečení může nastat tehdy, jestliže nelze výsledek správně zobrazit na daném počtu bitů

Příklad součtu dvou čtyřbitových čísla

- Mějme
  - Dvě kladná binární čísla v přirozeném kódu

$$0011 = (3)_{10}$$

$$0101 = (5)_{10}$$

- Proveďme
  - Převod na čísla záporná spočtením dvojkového doplňku
  - Hranaté závorky značí reprezentaci ve dvojkovém doplňku
- Máme tedy celkem čtyři čísla
  - Dvě kladná (MSB=0)
  - Dvě záporná (MSB=1)

$$[0011]_2 = 1101 = (-3)_{10}$$

$$[0101]_2 = 1011 = (-5)_{10}$$

- Platí
  - U záporných musíme dávat pozor, jak s nimi pracujeme

Součet dvou kladných čísel

$(+3)+(+3)$ $(+5)+(+5)$

	0	0	1	1		0	1	0	1
+	0	0	1	1	+	0	1	0	1
	0	1	1	0		1	0	1	0

- Popis
  - Svislou čarou je naznačena skutečnost, že máme k dispozici jen 4 bity
  - Čárkovanou čarou je oddělen znaménkový bit
- Výsledek
  - V prvním případě nám vyšel správný výsledek
  - Ve druhém případě nám při sčítání dvou kladných čísel vyšlo číslo záporné –6 (MSB = 1)
- Zkouška
  - Výsledek převeďme na kladné číslo nalezením dvojkového doplňku
  - Zapamatujme si znaménko původního výsledku
  - Doplníme k výsledku převodu
  - Výsledk je nesprávný
    - Došlo k přetečení - výsledek je mimo rozsah zobrazení čtyřbitových čísel ve dvojkovém doplňku

$$(+3)+(+3)=(+6)$$

$$[1010]_2 = 0110 = (6)_{10}$$

$$(+5)+(+5)=(+10)$$

$(-8)..(+7)$

Součet dvou záporných čísel

$(-3)+(-3)$ $(-5)+(-5)$

	1	1	0	1
+	1	1	0	1
1	1	0	1	0

	1	0	1	1
+	1	0	1	1
1	0	1	1	0

- Popis
  - Při výpočtu sumy zahazujeme případný přenos mimo zobrazení (carry nás nyní nezajímá – nelze zobrazit)
- Výsledek
  - V prvním případě nám vyšel správný výsledek
  - Ve druhém případě vidíme, že při sčítání dvou záporných čísel vyšlo číslo kladné +6 => došlo k přetečení
  - Výsledek je mimo rozsah zobrazení čtyřbitových čísel reprezentovaných ve dvojkovém doplňku

$$(-3)+(-3)=(-6)$$

$$(-5)+(-5)=(-10)$$

$$(-8)..(+7)$$

Součet čísla kladného se záporným

$(+5)+(-3)$

	0	1	0	1
+	1	1	0	1
1	0	0	1	0

$(+3)+(-5)$

	0	0	1	1
+	1	0	1	1
1	1	1	1	0

- Popis
  - V prvním případě nám vyšel správný výsledek $(+5)+(-3)=(+2)$
  - Ve druhém si pro kontrolu převeďme výsledek na číslo kladné $[1110]_2 = 0010 = (2)_{10}$
  - Vidíme, že i druhý výsledek je správný $(+3)+(-5)=(-2)$

Součet čísla záporného s kladným

$$(-5)+(+3)$$

	1	0	1	1
+	0	0	1	1
1	1	1	1	0

$$(-5)+(+3)=(-2)$$

$$(-3)+(+5)$$

	1	1	0	1
+	0	1	0	1
1	0	0	1	0

$$(-3)+(+5)=(+2)$$

- V obou případech nám opět vyšly správné výsledky

Sčítání čísel ve dvojkovém doplňku: Shrnutí

Součet	Znaménko	Přenos	Přetečení	Podmínka
$(+X)+(+Y)$	0 1	0 0	0 1	$(X+Y) \leq 2^{n-1}-1$ $(X+Y) > 2^{n-1}-1$
$(-X)+(-Y)$	1 0	1 1	0 1	$-(X+Y) \geq -2^{n-1}$ $-(X+Y) < -2^{n-1}$
$(+X)+(-Y)$ $(-X)+(+Y)$	0 1	1 0	0 0	$X \leq Y$ $X > Y$

- 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
- Poznámka
  - Ve většině počítačů se přetečení detekuje příslušným logickým obvodem a uschovává se. V případě, že se přetečení nedetekuje, je na programátorovi, aby zkontroloval, zda nedošlo k přetečení (porovnáním znamének operandů a výsledku)

Násobení binárních čísel bez znaménka

x	0	1
0	0	0
1	0	1

- Násobení binárních čísel
  - Je ještě jednodušší než jejich sčítání – nejsou zde totiž žádné přenosy
- Postup - příklad
  - Násobenec je postupně násoben jednotlivými bity násobitele od LSB k MSB
  - Mezivýsledky se vždy posunou o jeden bit vlevo (neboť jednotlivé bity násobitele mají vždy větší váhu)
  - Nakonec se vše sečte – v praxi se sčítance postupně akumulují (nesčítají se všechny 4 sčítance v jednom kroku)
  - V případě násobení čísel bez znaménka má výsledek dvojnásobný počet platných bitů

$$(1011)_2 = (11)_{10}$$
$$(1101)_2 = (13)_{10}$$

$$\begin{array}{cccccccc}
 & & & & 1 & 0 & 1 & 1 \\
 & & & x & 1 & 1 & 0 & 1 \\
\hline
 & & & & 1 & 0 & 1 & 1 \\
 & & & 0 & 0 & 0 & 0 & \\
 & & 1 & 0 & 1 & 1 & & \\
 & 1 & 0 & 1 & 1 & & & \\
\hline
1 & 0 & 0 & 0 & 1 & 1 & 1 & 1
\end{array}$$

$$(10001111)_2 = (143)_{10} = (11 \cdot 13)_{10}$$

Násobení binárních čísel se znaménkem

- Příklad - čísla ve dvojkovém doplňku
  - Násobení kladného čísla záporným
  - Postupujeme známým způsobem
  - V případě násobení znaménkovým bitem násobitele musíme provést korekci výsledku přičtením dvojkového doplňku násobence
- Výsledek
  - Je zobrazen na sedmi platných číslicích – 2x3 významové číslice + znaménko
  - Poslední mezivýsledek představuje potřebnou korekci

$$1110 = (-2)_{10}$$

$$0010 = (2)_{10} \quad 1101 = (-3)_{10}$$

$$(+2) \cdot (-3)$$

				0	0	1	0
				1	1	0	1
				0	0	1	0
			0	0	0	0	
		0	0	1	0		
	0	0	1	0			
1	1	1	0				
1	1	1	1	1	0	1	0

$$[1111010]_2 = 0000110 = (6)_{10}$$

Násobení binárních čísel se znaménkem

- Popis
  - Při násobení nulovým znaménkovým bitem násobitele neprovádíme korekci výsledku
- Výsledek není správně - očekávané řešení je –6
  - Chyba je způsobena tím, že jednotlivé mezivýsledky nebyly správně reprezentovány
  - Pokud násobíme záporného násobence jednotlivými bity násobitele, musíme též jednotlivé mezivýsledky reprezentovat a akumulovat jako čísla záporná
  - Druhý mezivýsledek je přičten k výsledku jako číslo kladné, nikoliv záporné

$$(-3)\cdot(+2)=(-6)$$

			1	1	0	1	
			0	0	1	0	
			0	0	0	0	
		1	1	0	1		
	0	0	0	0			
0	0	0	0				
0	0	1	1	0	1	0	

$$(0011010)_2=(+26)_{10}$$

1101

Násobení binárních čísel se znaménkem

- Správné řešení
  - Je třeba tzv. šířit znaménko a tím udržet správné zobrazení záporných čísel - stačí znaménko zkopírovat vlevo

$$1101 = 111101 = (-3)$$

				1	1	0	1
				0	0	1	0
0	0	0	0	0	0	0	0
1	1	1	1	0	1		
0	0	0	0	0			
0	0	0	0				
1	1	1	1	0	1	0	

- Zkouška

$$(-3) \cdot (+2) = (-6)$$