

Számrendszerek

A számábrázoláshoz vezető út első lépése, hogy a számítástechnikában használt számrendszerek közti átváltást elsajátítsuk. Ilyen számrendszerek a következők:

- Bináris (2-es)
- Oktális (8-as)
- Decimális (10-es)
- Hexadecimális (16-os)

Bináris, oktális és hexadecimális számrendszerek közt az átváltás nagyon egyszerű, ugyanis három bináris számjegy megfeleltethető egyetlen oktális számjegynek, négy bináris számjegy pedig egy hexadecimálisnak. Oktális és hexadecimális átváltás során, kézenfekvő közbenső műveletként bináris számrendszerbe átváltani.

bin	okt	bin	hex	bin	hex
000	0	0000	0	1000	8
001	1	0001	1	1001	9
010	2	0010	2	1010	A
011	3	0011	3	1011	B
100	4	0100	4	1100	C
101	5	0101	5	1101	D
110	6	0110	6	1110	E
111	7	0111	7	1111	F

Hogy decimális számokkal is boldoguljunk, elegendő bináris számrendszer és decimális közt átváltani. Innen már az előbb említett egyszerű átalakítással dolgozhatunk.

Tekintsünk most egy tetszőleges a alapú számrendszerben felírt számot:

$$b_n \dots b_1 b_0, b_{-1} \dots b_m \quad \forall i : b_i \in \{0, \dots, a-1\}$$

Ennek decimálissá történő átszámítása egyszerűen a következő összeget jelenti:

$$b_n a^n + \dots + b_1 a^1 + b_0 a^0 + b_{-1} a^{-1} + \dots + b_m a^m = \sum_{i=m}^n b_i a^i$$

Tetszőleges decimális szám a alapú számrendszerbe történő átalakításakor az egész és tizedes jegyeket külön kell kezeljük: legyen dec_0 az egész rész, dec_{m-1} a törtrész. Ekkor a számjegyek meghatározásához a következő táblázat nyújt segítséget.

dec_0	$/a$	$\cdot a$	dec_{m-1}
dec_1	b_0	b_m	dec_m
dec_2	b_1	\vdots	\vdots
\vdots	\vdots	b_{-3}	dec_{-3}
dec_{n-1}	b_{n-1}	b_{-2}	dec_{-2}
0	b_n	b_{-1}	0

$$dec_i = dec_{i+1} * a + b_i$$

$$(b_i + dec_i)/a = dec_{i-1}$$

Mint látható, a fenti megfogalmazás teljesen általános, azonban célszerű csak bináris ($a = 2$) számrendszerbe történő átváltásra alkalmazni, ugyanis így egyszerűen csak 2-vel való osztás és szorzás sorozat adja az eredményt, míg ugyanez 8-cal vagy 16-tal kissé lassabban menne.

Példaként nézzük meg 183,625 átírását bináris, oktális és hexadecimális számmá. A bináris megoldásból azonnal föl tudjuk írni az oktális és hexadecimális alakot is, de ellenőrzésképp megadjuk az oktális alak kiszámítását is.

183		/2		
91		1		
45		1		
22		1		
11		0		
5		1		
2		1		
1		0		
0		1		

·2		0,625
1		0,250
0		0,500
1		0,000

$$183,625_{(10)} = 10110111,101_{(2)} = 267,5_{(8)} = B7, A_{(16)}$$

183		/8		
22		7		
2		6		
0		2		

·8		0,625
5		0,000

Számábrázolás

Amikor a számítógépben egy értéket ábrázolunk, azt mindig adott méretű tárterületen tehetjük. Így nem kezelhetünk akármilyen számokat, az ábrázolás korlátja a felhasználható bitek száma, melyet mindig előre rögzíteni kell. Egy bájt biteit 0-tól 7-ig szokás számozni, ahol 0 a legalacsonyabb helyiértékű bit. Gyakori az a jelölés is, mikor egyetlen bájt megadására két hexadecimális számjegyet használunk.

Fixpontos ábrázolás

Fixpontos szám ábrázolás során az ábrázolás előre rögzített kettedes jegy pontos, azaz a kettedes és egész jegyek száma adott. Ezt általában egész számok ábrázolását jelenti, mikor a kettedes jegyek száma nulla. Szokatlan ugyan, de elképzelhető a valós számok fixpontos ábrázolása.

Nemnegatív egész számok

A decimálisan adott számot binárisra alakítjuk, majd tároljuk, a megadott méretű területen. Hogy ezt valóban kitöltsük, balról nullákkal egészítjük ki a számot. Álljon itt példaként a következő:

$$11852_{(10)} = 10111001001100_{(2)}$$

16	8	0	16	8	0
0	0	1	0	1	1
0	1	1	0	0	1
0	1	1	0	0	0

16	8	0
2	E	4
C		

- A legkisebb ábrázolható szám n biten: 0
- A legnagyobb ábrázolható szám n biten: $2^n - 1$

Egész számok

A kérdés a negatív számok ábrázolása. Erre a következő három megoldás létezik, melyekből az utóbbi kettő használatos.

Előjeles egész szám ábrázolás

Az előjeles ábrázolásnál a legmagasabb helyi értékű bitet előjel jelzésére tartjuk fenn. Megállapodás szerint a pozitív előjelnek a 0, míg a negatívnak az 1-es felel meg. A fennmaradó biteken a szám abszolút értékét tároljuk. Problémát jelent azonban, hogy nulla kétféle is létezik, $+0$ és -0 , valamint a kivonás művelete viszonylag komplikáltan implementálható. Legyen a példa az iméntihez hasonló.

$$-11852_{(10)} = -10111001001100_{(2)}$$

16	8	0	16	8	0
1	0	1	0	1	1
0	1	1	0	0	1
0	1	1	0	0	0

16	8	0
A	E	4
C		

- A legkisebb ábrázolható szám n biten: $-2^{n-1} - 1$
- A legnagyobb ábrázolható szám n biten: $+2^{n-1} - 1$

Kettes komplement

A negatív számok helyett azok kettes komplementének tárolását jelenti. A kettes komplement az egyes komplementnél eggyel nagyobb szám, ahol az egyes komplement egyszerűen a szám bitenkénti negáltja (amennyiben bináris számról van szó). Egy ábrázolt szám legmagasabb helyi értékű bitje pontosan akkor 1, ha a szám negatív volt (bár nem tárolunk előjelet, ez mégis olyan egyszerűen megállapítható, mint az előjeles ábrázolásnál). A negatív számot úgy kapjuk vissza, hogy a kettes komplement képzését visszafelé hajtjuk végre, avagy ismét a kettes komplementét képezzük.

Előnye, hogy a kivonást nem kell implementálni, ez egyszerűen negatív szám hozzáadását jelenti. A kettes komplement előállításánál közben túlsordulás léphet fel, de ekkor a túlsordult bittel nem foglalkozunk.

$$-11852_{(10)} = -0010111001001100_{(2)} \xrightarrow{\text{not}} 1101000110110011_{(2)} \xrightarrow{+1} 1101000110110100_{(2)}$$

16	8	0	16	8	0
1	1	0	1	0	0
1	0	0	1	1	0
1	0	0	1	0	0

- A legkisebb ábrázolható szám n biten: -2^{n-1}
- A legnagyobb ábrázolható szám n biten: $+2^{n-1} - 1$

Feszített előjeles ábrázolás

Szokás eltolásos vagy alapértékes ábrázolásként is emlegetni. A negatív számokat is úgy tároljuk, mintha pozitívak lennének. Ezt úgy érjük el, hogy minden ábrázolni kívánt számhoz hozzáadunk egy előre rögzített c konstans értéket. A legkisebb ábrázolható szám $-c$ lehet. Az összeadást elvégezve, így mindenképp nem negatív számot kapunk, melynek ábrázolása már ismertetésre került.

Előnye, hogy lebegőpontos számításoknál, ha a kitevőt így tároltuk el, az egyes számjegyek valódi helyi értékei könnyen kiszámíthatók. Az előjel megállapításához azonban vissza kell alakítani a számot.

$$-11852_{(10)} = -0010111001001100_{(2)} \xrightarrow{+2^{15}} 1010111001001100_{(2)}$$

16	8	0	16	8	0
1	0	1	0	1	1
0	1	1	0	0	1
1	0	0	1	1	0

- A legkisebb ábrázolható szám n biten $c = 2^{n-1}$ mellett: -2^{n-1}
- A legnagyobb ábrázolható szám n biten $c = 2^{n-1}$ mellett: $+2^{n-1} - 1$

Lebegőpontos ábrázolás

A lebegőpontos szám ábrázolás a számok hatványkitevős felírásán alapszik.

$$e \cdot M \cdot a^k \quad e \in \{-1, +1\}$$

Hogy e felírás egyértelmű legyen, M mindig kisebb kell legyen, mint 1, és az első nem egész jegy nem lehet 0.

$$\frac{1}{a} \leq M < 1$$

A lebegőpontos szám többnyire bináris számok hatványkitevős alakja. Többféle megoldás létezik, melyből kettőt tekintünk most meg, mindkét esetben az egyszeres pontosságú (4 bájtól történő) ábrázolást tekintve:

Hexadecimális alap

Az IBM gépek a bináris számot hexadecimális alakban képezik le, és így végzik el a normálást, azaz a törtrész első hexadecimális jegyének kell nullánál nagyobbak lennie. A legmagasabb helyi értékű bit mindig a szám előjele. Ezt követi a karakterisztika mely most 7 bites, így a maradék három bájtól, a mantissza (M) mint 6 számjegyű hexadecimális szám jelentkezik. A karakterisztika feszített előjeles, ahol $c = 64$.

$$183,625_{(10)} = B7, A_{(16)} = 0, B7A_{(16)} * 16^2$$

$$e = +1 \rightarrow 0, \quad k = 2 \xrightarrow{+64} 66, \quad M = 10110111101$$

32	24	16	8	0	32	24	16	8	0
0	1	0	0	0	0	1	0	1	1
0	0	1	0	1	1	1	1	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
4	2	B	7	A	0	0	0	0	0

Bináris alap

A PDP típusú gépek a bináris számot normálják. Ez azt jelenti, hogy az első kettedes jegy legyen nullánál nagyobb. Mivel bináris számoknál ez mindig az 1-es értéket fogja jelenteni, ennek tárolása felesleges, a fennmaradó egy bittel kibővíthetjük a karakterisztikát, mely így 8 bitesre duzzad. A karakterisztika eltolása $c = 126$ (ezt általában 128-nak kéne venni 8 bites feszített előjeles ábrázolás esetén, a konkrét példa azonban, melyet a manapság is használt valós típus szolgáltat, nem ilyen). Így az ábrázolható számok: $1,5 * 10^{-45} \dots 3,4 * 10^{38}$ tartományba esnek és 7–8 tizedes jegyre pontosak.

$$183,625_{(10)} = 10110111,101_{(2)} = 0,10110111101_{(2)} * 2^8$$

$$e = +1 \rightarrow 0, \quad k = 8 \xrightarrow{+126} 134, \quad M = 10110111101 \rightarrow 0110111101$$

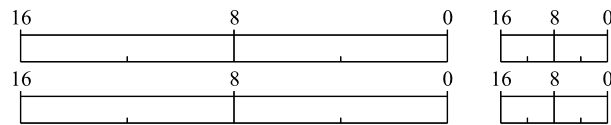
32	24	16	8	0	32	24	16	8	0
0	1	0	0	0	0	1	0	1	1
0	0	1	0	1	1	1	1	0	0
0	0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
4	3	3	7	A	0	0	0	0	0

Decimális számok ábrázolása

A binárisan kódolt decimális szám ábrázolása úgy zajlik, hogy a decimális számot számjegyenként tároljuk le. Ennek módja is különböző a fent említett géptípusok esetében.

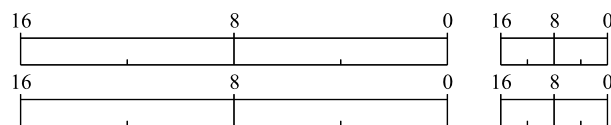
Ábrázolja 16 biten, kettes komplement technikával tárolva a következő számokat! (2 pont)

$$-34165_{(8)}; \quad 12644_{(10)}$$



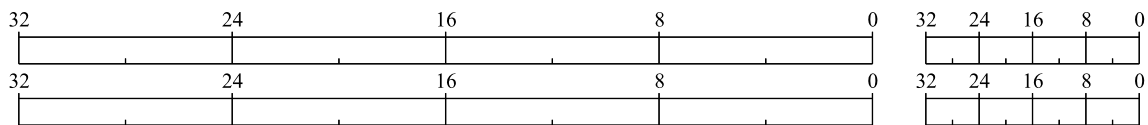
Ábrázolja 16 biten, $c = 2^{15}$ eltolás mellett, feszített előjeles technikával a következő számokat! (2 pont)

$$1235_{(10)}; \quad -1135_{(10)}$$



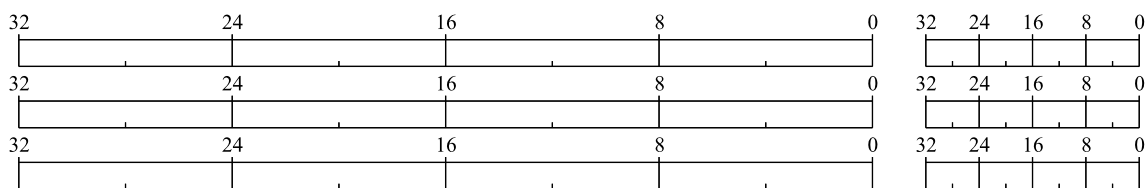
Ábrázolja 32 biten pakolt és zónázott formában a következő számot! (1 pont)

$$5793_{(10)}$$



Ábrázolja lebegőpontosan a következő valós számokat! Az ábrázolás 32 biten, 8 bites karakterisztikával – $c = 126$ eltolás mellett –, és bináris alappal történjen, használja a rejtett egyes technikát! (3 pont)

$$52, 236; \quad -655, 36; \quad -6658, 25$$



Ábrázolja lebegőpontosan a következő valós számokat! Az ábrázolás 32 biten, 7 bites karakterisztikával – $c = 64$ eltolás mellett –, és hexadecimális alappal történjen! (2 pont)

$$276, 44; \quad -300, 3$$

