Számítógépes számábrázolás

Számrendszer

Jelek továbbá jelölési és logikai elvek összessége, melyek segítségével a *végtelen sok mennyiség* bármelyikének egyedi nevet adhatunk egy *véges jelkészlet*tel. Mindeközben a mennyiségek formális nevei alkalmassá válnak számolások könnyű elvégzésére. Ma már kizárólag *helyértékes számrendszer*eket alkalmazunk. A mindennapi életben általános a tízes számrendszer, a számítástechnikában a bináris (kettes) illetve a hexadecimális (tizenhatos) számrendszer.

A helyértékes számrendszer nemcsak egész, hanem tört és irracionális számok jelölésére is alkalmas.

A számrendszereket az egy helyértéken ábrázolható különböző mennyiségek száma alapján nevezzük el. Ez a szám, figyelembe véve a *csoportképzési elv*et, bármely 1-nél nagyobb egész szám lehet. Pl.: a 10-es számrendszer alapja 10, hiszen 10 különböző mennyiséget nevezünk meg különböző jelekkel(0, 1, 2…9), a 2-es számrendszer alapja 2 (0, 1).

A számítógép minden bevitt jelet, adatot bináris számokká alakít, és e számokkal végez műveleteket (logikai, aritmetikai). A kettes számrendszer az elektronika által legkönnyebben használható számrendszer, mivel két stabil állapotot (0: nincs áram, 1: van áram) használ. Kényelmi okokból néha hexadecimális (tizenhatos) számrendszert is használunk, főleg a tömörebb írás miatt, mivel a sok bináris jegy nehezen írható és áttekinthető az ember számára. (Nehéz mennyiség képzetet asszociálni a 0-1 jelsorozathoz.)

2-es számrendszer, átváltás kettesből tízes számrendszerbe (2→10)

A kettes rendszerben a helyértékeket a kettő egész kitevős hatványai jelentik.

Példa: Nézzük meg mekkora tízes számrendszerbeli számnak felel meg az alábbi 8 bites kettes számrendszerbeli szám: 10010110?

2^7=128	2^6=64	2^5=32	2^4=16	2^3=8	2^2=4	2^1=2	2^0=1
1	0	0	1	0	1	1	0

$10010110_2 = 1\cdot2^7 + 0\cdot2^6 + 0\cdot2^5 + 1\cdot2^4 + 0\cdot2^3 + 1\cdot2^2 + 1\cdot2^1 + 0\cdot2^0 = 128+0+0+16+0+4+2+0 = 150_{10}$

Tudjuk, hogy az adattárolás és információ feldolgozás alapegysége a bájt ami 8 bitet jelent. Egy bájtnyi helyen tárolható legnagyobb bináris szám:

1	1	1	1	1	1	1	1

$11111111_2 = 1\cdot2^7+1\cdot2^6+1\cdot2^5+1\cdot2^4+1\cdot2^3+1\cdot2^2+1\cdot2^1+1\cdot2^0 = 128+64+32+16+8+4+2+1= 255$

Tehát egy bájtnyi helyen 256 darab szám (0-255) tárolható. Az alfanumerikus jelek (betű-, szám-, írás-, műveleti-jelek valamint a nyelvi szintaxis jelei) tárolására elegendő az 1 bájtnyi adatterület, ezért ezeket a jeleket 1 bájt hosszúságú kódokkal tároljuk a számítógépen (ASCII – American Standard Code For Information Interchange kódrendszer). Pl. az „*A*” betű decimális ASCII kódja 65, mely a kettes számrendszerben 01000001 jelsorozatnak felel meg. Az első nullát vezető nullának nevezzük, melyet a matematikai számításokban megállapodás szerint nem kell kiírnunk, de a számítógép tároló regiszterét mindenképpen tájékoztatni kell a legmagasabb helyérték nagyságáról (0 vagy 1).

Átváltás decimális számrendszerből kettes számrendszerbe (10→2):

Vegyük a decimális számot, majd osszuk el 2-vel; ezt folytassuk, amíg a hányados nulla nem lesz! A maradékokat fordított sorrendben visszaírva megkapjuk a hexadecimális számot!

2634	0	2634:2 = 1317 maradék 0
1317	1	1317:2 = 658 maradék 1
658	0	
329	1	
164	0	
82	0	
41	1	
20	0	
10	0	
5	1	
2	0	
1	1	

A maradékokat a második oszlopban tüntettük fel.

Tehát a 2634_{10} szám bináris értéke: 101001001010_2. Természetesen egy ekkora szám tárolására már 2 bájtnyi (8+8=16 bit) adatterület szükséges. (az első négy vezető nullát nem írtuk ki)

16-os számrendszer, átváltás tizenhatos számrendszerből tízesbe (16→10)

Tizenhat alapjellel dolgozunk (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, ahol A=10, B=11, C=12, D=13, E=14, F=15).

Például:

$A8EC_{16} = 10 \cdot 16^3 + 8 \cdot 16^2 + 14 \cdot 16^1 + 12 \cdot 16^0 = 10 \cdot 4096 + 8 \cdot 256 + 14 \cdot 16 + 12 \cdot 1 = 43244_{10}$

Átváltás decimális számrendszerből hexadecimálisba (10→16)

Vegyük a decimális számot, majd osszuk el 16-al; ezt folytassuk amíg a hányados nulla nem lesz! A maradékokat fordított sorrendben visszaírva megkapjuk a hexadecimális számot!

43244	12	43244 : 16 = 2702 maradék 12
2702	14	
168	8	
10	10	

Tehát $43244_{10} = A8EC_{16}$, így visszakaptuk a várt eredményt.

Közvetlen átváltás hexadecimális és bináris számrendszerek között (2↔16)

Az elv a *tetrád képzés módszere*. A módszer mindkét irányban működik.

Vegyünk egy számot a 16-os számrendszerben: $28EC_{16}$

Vegyük a helyértékek bináris értékét

2	8	E	C
0010	1000	1110	1100

Írjuk ezeket egymás mellé! Így ki is jött a szám bináris értéke:| $10|1000|1110|1100|_2$

Aritmetikai műveletek a számítógépen 2-es számrendszerben

1. **Összeadás:**

Bitenként adjuk össze a számokat, és figyelembe vesszük a *keletkező átvitel*eket.

Az egyes bitösszegeket az összeadandó bitek kizáró-vagy (XOR) kapcsolata adja meg.

$0 \oplus 0 = 0$,
$0 \oplus 1 = 1$,
$1 \oplus 0 = 1$,
$1 \oplus 1 = 0$,

```
  101101
+   1110
--------
  111011
```

a 3. és 4. helyértékeken:
1+1 = 10, leírom a 0-t,
marad az 1 (átvitel)
1+1+1 = 11, leírom az 1-et,
marad az 1 (átvitel)

Leellenőrizhető a művelet tízes számrendszerben is:
$101101_2 = 45_{10}$
$1110_2 = 14_{10}$
45+14 = 59
$59_{10} = 111011_2$

A számítógép a 45 + 14 művelet argumentumait tehát a következőképpen tárolja:

Összeadandók bájtjai:

0	0	1	0	1	1	0	1

0	0	0	0	1	1	1	0

Eredmény:

0	0	1	1	1	0	1	1

2. **Kivonás:**

A kivonás az összeadásra vezethető vissza az A - B = A + (- B) összefüggés alapján. Azaz a kisebbítendőhöz hozzáadjuk a kivonandó ellentettjét.

Kettes számrendszerben egy *A* szám *–A* ellentettjét *kettes komplemens*nek nevezzük. Ezt a módszert Konrad Zuse és Neumann János dolgozták ki.

Kettes komplemens előállítása:

Először képezzük a szám egyes komplemensét, ezt úgy tesszük, hogy a számot bitenként invertáljuk. Majd az így kapott egyes komplemenshez hozzáadunk 1-et. Így kapjuk a kettes komplemenst.

Példa:

Állítsuk elő 1011101-nek a kettes komplemensét! A szám egy bájtos alakja: 01011101 (a legnagyobb helyértékre egy vezető nullát írunk). Az egyes komplemens invertálással: 10100010.

A kettes komplemenst megkapjuk, ha ehhez hozzáadunk egyet:

```
  10100010
+         1
----------
  10100011
```

Tehát 1011101 kettes komplemense 10100011. Ha a kivonás szabálya helyes, akkor egy szám és a szám kettes komplemensének összege nulla kell legyen. Ellenőrizzük le!

```
    101|1101
+ 1010|0011
-----------
 1|0000|0000
```

Látszik, hogy a jobboldalon nyolc nullás van, ami azt jelenti, hogy a művelet eredménye nulla lesz… feltéve, hogy egy bájtos ábrázolást használunk. Ekkor a legnagyobb helyértéken keletkező 1-es átvitel, úgymond, a semmibe fordul. Ezt *túlcsordulás*nak nevezik.

Az említett feltevés nem megy az általánosság rovására, ugyanis minden fizikailag létező számítógép véges méretű kijelzővel rendelkezik. A komplemensképzés segítségével összeadásként elvégzett kivonás függetlenül a használt kijelző bitszámától helyes eredményt ad, amennyiben a ténylegesen létező legnagyobb bitszámra képezzük a komplemenst.

Példa:

Végezzük el a $22 - 13 = ?$ műveletet a számok bináris kódjával!

$22_{10} = 10110_2$

0	0	0	1	0	1	1	0

$13_{10} = 1101_2$

0	0	0	0	1	1	0	1

A 13-as kettes komplemens kódja (8 bites kijelző esetén):

Invertáljuk a biteket $\rightarrow$ 11110010

Hozzáadunk 1-et $\rightarrow$ kettes komplemens kód: 11110011

1	1	1	1	0	0	1	1

$22 - 13 = 22 + (-13) =$

$$\begin{array}{r} 0001|0110 \\ +\ 1111|0011 \\ \hline 1|0000|1001 \end{array}$$

Az eredménybájt tehát:

0	0	0	0	1	0	0	1

$1001_2 = 9_{10}$. Az eredmény tehát helyes.

Az egész számok leírt ábrázolási módját *fixpontos ábrázolásnak* nevezzük. A tizedes számoknak is megvan az ábrázolási módszerük, ezt nevezzük *lebegőpontos ábrázolásnak*. A számokat általában páros számú bájtokon ábrázolják (jellemzően 4, 8, 16 bájt).

A számítógép hasonlóképpen végzi el a szorzás és osztás műveleteket is, erre bonyolultságuk miatt nem térünk ki.

Feladat: Mekkora az a legnagyobb pozitív egész szám mely 16 bájton ábrázolható?

16B = 16·8 bit = 128 bit

Ha a 128 bit mindegyike egyes, akkor a szám értéke 2^{128} -1. Ez a szám nagyon nagy! ($\approx 3{,}4 \cdot 10^{38}$)