Számrendszerek

Az informatika fejlődésével a megszokott tízes számrendszer mellé újak születtek. Ilyen például a kettes, nyolcas vagy a tizenhatos.

A tízes (decimális) számrendszer:

A mindennapi életben a helyiértékes rendszerű tízes, más néven decimális számrendszert használjuk. Ez tíz számjegyből és a 10 hatványainak megfelelő helyiértékekből áll. Számjegyei: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, helyiértékei pedig például: egyes, tízes, százas, ezres…

A kettes (bináris) számrendszer:

A kettes számrendszer az egész informatika alapja ilyen téren. Minden hardveri vagy szoftveri utasítás ugyanis ezen alapszik – 0 és 1 szerepel bennük és ezek megfelelő sorozatát fordítják le. Erre épül továbbá minden logikai utasítás is – a 0 a nem, az 1 pedig az igen logikai értéket jelöli. Hardveri téren az áram folyásának állapotát is 0-val vagy 1-el jelölik – folyik áram vagy sem. Tehát a kettes számrendszer számjegyei a 0 és 1, alapszáma pedig a 2.

Nézzünk egy példát, amelyben egy bináris számot decimálissá alakítunk át.

$$11100110_2 = 2^7 + 2^6 + 2^5 + 2^2 + 2^1 = 128 + 64 + 32 + 4 + 2 = 230_{10}$$

A tízes számrendszerbeli egész számot kettes számrendszerbe az **ismételt osztás** módszerével alakíthatjuk át. A decimális számokat elosztjuk kettővel, az eredményt a szám alá, a maradékot (ami csak 1 vagy 0 lehet) pedig mellé írjuk. Ez a maradék lesz a legkisebb (2^0) helyérték. Ezután a hányadost ismét elosztjuk kettővel, és megkapjuk a következő helyértéket, stb. Az osztást addig végezzük, amíg az eredmény 0 nem lesz.

230	0	2^0
115	1	2^1
57	1	2^2
28	0	2^3
14	0	2^4
7	1	2^5
3	1	2^6
1	1	2^7
0		

Tehát $230_{10} = 11100110_2$

A tízes számrendszerbeli törtszámot kettes számrendszerbe az **ismételt szorzás** módszerével alakíthatjuk át. A decimális törtszámot megszorozzuk kettővel, az eredményt a szám alá, az eredmény egész részét (ami csak 1 vagy 0 lehet) pedig mellé írjuk. Ez az egész rész lesz a legnagyobb (2^{-1}) helyértéke a törtnek. Ha a szorzat egynél nagyobb szám lett, akkor egyet levonunk belőle, majd az így kapott számot ismét megszorozzuk kettővel, és megkapjuk a következő helyértéket, stb. A szorzást addig kellene ismételnünk, amíg a szorzat 1 nem lesz. A véges tizedes

törtek azonban leggyakrabban végtelen kettedes törtnek felelnek meg. Minél többször ismételjük meg a szorzást az eredmény annál pontosabb lesz.

0,181	0	2^{-1}
0,362	0	2^{-2}
0,724	1	2^{-3}
1,448	0	2^{-4}
0,896	1	2^{-5}
1,792	1	2^{-6}
1,584	1	2^{-7}
1,168		

Tehát $0,181_{10} = 0,0010111_2$

A nyolcas (oktális) számrendszer:

A tizenhatos számrendszerrel együtt a kevésbé használatos de legalább annyira fontos számrendszerek közé tartoznak. A nyolcas számrendszer számjegyeinek száma tehát 8 (0, 1, 2, 3, 4, 5, 6, 7), alapszáma szintén 8.

A bináris - oktális átalakításhoz csak az oktális számjegyek bináris megfelelőjét kell ismernünk. A nyolcas számrendszerbeli szám átalakítása kettes számrendszerbe úgy történik, hogy az oktális számjegyeket átalakítjuk binárissá, és egymás után leírjuk. Fordítva pedig a kettes számrendszerbeli számot hármas csoportokra bontjuk a kettedes vesszőtől indulva balra, majd jobbra, és a bitcsoportokhoz a megfelelő sorrendben odaírjuk az oktális megfelelőjét.

Bináris	Oktális
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

$217,065_8 = 010\ 001\ 111\ ,\ 000\ 110\ 101_2$
$011\ 100\ 110\ ,\ 001\ 101\ 111_2 = 346,157_8$

A tizenhatos (hexadecimális) számrendszer:

A tizenhatos számrendszer számjegyeit tekintve már kissé bonyolultabb, mint az eddig megismert számrendszerek, ugyanis mivel 16 számjegye van, mi azonban csak tízet jelölünk valahogy, ezért a maradék hatot az angol ABC betűivel jelölik a következőképpen: 11-A, 12-B, 13-C, 14-D, 15-E, 16-F.
Így tehát a számjegyek: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. A számrendszer alapszáma 16.

A bináris - hexadecimális átalakításához ismernünk kell a hexadecimális számjegyek bináris megfelelőjét. A tizenhatos számrendszerbeli szám átalakítása kettes számrendszerbe úgy történik, hogy a hexadecimális számjegyekkel átalakítjuk binárissá, és egymás után leírjuk. Fordítva a kettes számrendszerbeli számot négyes csoportokra bontjuk a kettedes vesszőtől jobbra, majd balra indulva, és a bitcsoportokhoz a megfelelő sorrendben odaírjuk a hexadecimális megfelelőjét., úgy mint a nyolcas számrendszernél láthattuk.

Bináris	Hexadecimális
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

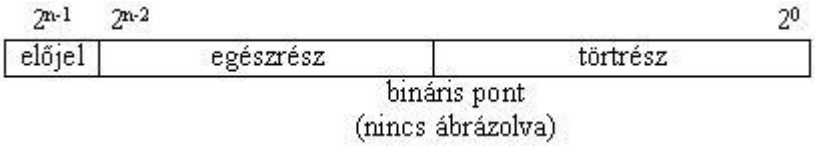
$BD,3C_{16} = 1011\ 1101\ ,\ 0011\ 1100_2$

$1110\ 0110\ ,\ 1001\ 1010_2 = E6,9A_{16}$

Számábrázolás fajtái

Fixpontos számábrázolás

Ebben az ábrázolási módban a bináris pont helye - ami a baloldalon található egészeket elválasztja a jobb oldalon lévő törtektől - rögzített, és a számokat többnyire kettes komplemens kódban ábrázolják.



A számok ábrázolásának két fontos jellemzője van a felhasználás szempontjából:

- az ábrázolandó számok nagysága
- az ábrázolás pontossága

A két jellemző az alkalmazott regisztermérettől és bináris pont helyétől függ. Ha a bináris pontot balra toljuk el, akkor

- a számok ábrázolási tartománya csökken
- az ábrázolás pontossága nő
- ha bináris pont a regiszter bal szélén van, akkor a szám fixpontos tört.

Ha pedig a bináris pont jobbra mozdul, akkor

- a számok ábrázolási tartománya nő
- az ábrázolás pontossága csökken
- ha a bináris pont a regiszter jobb szélén van, akkor a szám fixpontos egész.

Az ábrázolható számok tartományát a lenti ábra mutatja. Itt látható, hogy a számegyenesen nem minden tartományt tudunk ábrázolni, az abszolút értékben túl nagy (overflow) vagy nagyon kicsi (underflow) számok nem ábrázolhatók egy adott regiszter méret mellett, illetve az ábrázolandó számok tartományában sem minden valós szám ábrázolható.

Lebegőpontos számábrázolás

A számokat ebben az esetben normalizált alakban használjuk.

A mantisszát leggyakrabban előjeles abszolút értékes formátumban tárolják. A normalizálásra kétféle gyakorlat terjed el.

Törtre normalizálás

A bináris pontot addig toljuk el, amíg a mantissza értéke 1/2 és 1 közötti értékű nem lesz.
$N_2 = 0, 00010110012 = 0, 1011001 \cdot 2^{-3}$

Mivel a 2^{-1} helyértéken lévő bit mindig 1 értékű, ezért a szám eltárolása előtt kiveszik. Ezt **implicitbit**nek hívják. Így a tárolt mantissza (m) értéke: 011001000...

Egészre normalizálás

Ez esetben a normalizált mantissza értéke 1 és 2 közé esik.
A karakterisztikához egy egész számot adnak hozzá, és így tárolják. Ezt a megoldást **eltolt vagy ofszet karakterisztiká**nak hívják. Az eltolásra azért van szükség, hogy a karakterisztikát eltoljuk a pozitív számok tartományába, és így nem kell az előjelét ábrázolni.

A lebegőpontos számok ábrázolásának egységesítésére született az ANSI/IEEE 754 szabvány, amelyet a nagy processzorgyártók (INTEL, MOTOROLA, stb.) is használnak. Ez a szabvány háromféle lebegőpontos formát ír elő:

- szimpla pontosság 32 bit
- dupla pontosság 64 bit
- belső pontosság 80 bit