Kettes számrendszer véges ábrázolásai

Integrált szakdolgozat

Írta: Harmath Zsolt

Matematika tanár szak

Témavezető:

Hermann Péter, egyetemi docens

Algebra és Számelmélet Tanszék
Eötvös Loránd Tudományegyetem, Természettudományi Kar

Eötvös Loránd Tudományegyetem
2013

NYILATKOZAT

Név: Harmath Zsolt

ELTE Természettudományi Kar, szak: matematika-informatika tanár

ETR azonosító: HAZLABT.ELTE

NEPTUN azonosító: CI0SWY

Szakdolgozat címe: Kettes számrendszer véges ábrázolásai

A **szakdolgozat** szerzőjeként fegyelmi felelősségem tudatában kijelentem, hogy a dolgozatom önálló munkám eredménye, saját szellemi termékem, abban a hivatkozások és idézések standard szabályait következetesen alkalmaztam, mások által írt részeket a megfelelő idézés nélkül nem használtam fel.

Budapest, 2013. május 31.

_____

a hallgató aláírása

Eötvös Loránd Tudományegyetem
Természettudományi Kar

Matematikai Intézet

Hozzájárulás szakdolgozat benyújtásához

Alulírott……(témavezető)
igazolom, hogy …… ……(hallgató)
az előre meghatározott gyakoriságú szakdolgozati konzultáción részt vett, és hozzájárulok szakdolgozatának benyújtásához.

Budapest, 20….

(témavezető aláírása)

Tartalomjegyzék

Előszó **6**
1. Természetes számok **7**
1.1 Oszthatóság ***8***
1.2 Számrendszerek közti átváltás ***8***
1.2.1 Átváltás különböző alapú számrendszerek között 9
1.3 Természetes számokkal végzett alapműveletek ***11***
1.3.1 Összeadás 11
1.3.2 Kivonás 12
1.3.3 Szorzás 13
1.3.4 Osztás 14
1.3.4 Gyűrű-tulajdonságok 15
2. Negatív számok **16**
2.1 Komplemens ábrázolás ***16***
2.2 Egész számok halmaza ***17***
2.3 Műveletek egész számokkal ***18***
2.3.1 Túlcsordulás előjelről 18
2.3.2 Túlcsordulás előjelre 19
2.3.3 Asszociativitás 21
2.3.4 Disztributivitás 21
3. Bitműveletek **23**
3.1 Egy és kétváltozós bitműveletek ***23***
3.1.1 Tagadás 23
3.1.2 Konjunkció (ÉS) 23
3.1.3 Diszjunkció (megengedő VAGY) 24
3.1.4 Implikáció 24
3.1.5 Ekvivalencia 24
3.1.6 Antivalencia (kizáró VAGY) 25
3.1.7 NOR (Not OR) 25
3.1.8 Sheffer-művelet, NAND (Not AND) 25
3.2 A teljes igazság ***26***
3.3 Bitműveletek tételei ***27***
3.3.1 Tételek tagadásra, konjunkcióra, diszjunkcióra 27
3.3.2 A Sheffer-féle NAND teljessége 28
3.3.3 Bitenkénti logikai műveletek 29

3.3.4 Megoldás az egész számok túlcsordulási hibáira 29
3.3.5 Relációk 30
4. A valós számok 32
4.1 A gépi számok numerikus modellje 32
4.2 Szabványos lebegőpontos számábrázolás 33
4.2.1 Műveletek lebegőpontos számokkal 35
5. Véges ábrázolás alkalmazásai 36
5.1 Excel szorzás bug (2007) 36
5.2 Lebegőpontos ábrázolás matematikai szoftverekben 37
5.2.1 Matlab, Octave 37
5.2.2 Maple 38
5.2.3 R (statisztika) 38
6. A kettes számrendszer tanítása szakközépiskola 9. évfolyamán 39
6.1 1. óra: Bevezető óra 39
6.2 2. óra: Kettes és más alapú számrendszerek értelmezése 39
6.3 3. óra: Decimális számok átváltása 40
6.4 4. óra: Számrendszerek megtapasztalása IKT eszközökkel 42
6.5 Dolgozat: Számrendszerek 43

Előszó

A kettes számrendszernek az elmúlt 70 évben különleges jelentősége lett. Az elektronikus számítógép megjelenése forradalmasította az informatikát, elindítva azon az úton, mely ma az információs társadalom kibontakozásához és a lehető legteljesebb információáramláshoz vezetett.

Nem is találhatunk jobb leírást a kettes számrendszer jelentőségéről, mint Neumann János First Draft munkáját, melyben lefektette azon alapelveket, amelyekből megszületett a számítógép.

Részlet a First Draft-ból:

„A consistent use of the binary system is also likely to simplify the operations of multiplication and division considerably. Specifically it does away with the decimal multiplication table, or with the alternative double procedure of building up the multiples of each multiplier or quotient digit by additions first, and then combining these (according to positional value) by a second sequence of additions or subtractions. In other words: Binary arithmetics has a simpler and more one-piece logical structure than any other, particularly than the decimal one.” (John von Neumann [1])

A kettes számrendszer következetes használata jelentősen megkönnyíti a szorzás és osztás műveletét. Konkrétan, nem lenne feltétlenül szükség tízes alapú szorzótáblára vagy olyan eljárások megalkotására, melyekben újabb számjegyekkel kell kiegészíteni a tényezőket, majd (helyiértékek alapján) részszorzatokat kell összeadni vagy kivonni. Más szavakkal: A kettes számrendszer aritmetikája egységesebb és egyszerűbb logikai szerkezet, mint bármely más számrendszeré, akár a tízesé.

Számos dolgozat [2], publikáció [3], disszertáció [4] született a témában. Dolgozatomban arra vállalkoztam, hogy elsősorban a véges ábrázolásból eredő érdekes vagy meglepő következtetésekre felhívjam a figyelmet, továbbá a témához kapcsolódó, gyakorlati jelentőséggel bíró jelenségeket bemutassam.

Az integrált tanári dolgozat részét képezi az utolsó fejezetben tárgyalt, „A kettes számrendszer tanítása szakközépiskola 9. évfolyamán” rész is.

Számok fix- és lebegőpontos véges ábrázolása nem csak kizárólag a kettes számrendszer sajátossága. Más szabványok véges számábrázolásra kidolgozott aritmetikával például:

1. BCD (Binary-Coded Decimal, binárisan kódolt számok) [5]
2. EBCDIC (Extended Binary Coded Decimal Interchange Code) [6]

A továbbiakban számok véges ábrázolásával a kettes számrendszer keretein belül foglalkozunk, ahogy azt Neumann János is javasolta [1] 1945-ben a First Draft-ban kidolgozott elveiben, melyeket ma Neumann-elvekként ismerünk.

1. Természetes számok

A természetes számok kettes számrendszerbeli ábrázolása középiskolából ismert. Ezúttal véges sok számjegyen vizsgáljuk a számok ábrázolását.

Definíció (*bitmélység*): $n \in \mathbb{N}$ a számjegyek (maximális) száma.

Ismert: bármely $s \in \mathbb{N}$ felírható $s = \sum_{i=0}^{n-1} s_i 2^i$alakban, ahol s_i jelöli az s szám i-edik kettes számrendszerbeli számjegyét.

Jelölés: $s_{n-1}s_{n-2} \dots s_{0②}$.

Ha másképp nincs jelölve, a számok a példákban kettes számrendszerben vannak megadva.

A tízes számrendszer jele az alsó indexbe tett ⑩.

Definíció (*bit, BInary Digit*): kettes számrendszerbeli számjegy.

Minden bit vagy 0, vagy 1 értékű, más nincs.

Szokás még magát a helyiértéket is bitnek nevezni.

A példák egyszerűség kedvéért $n = 8$ bites ábrázolásúak.

Példa:

A 00100101 számban 5 db 0-bit és 3 db 1-bit van, és a 0. bitje 1, az 5. bitje is 1.

Az i. bit tehát a fenti képletben a 2^i együtthatója, 0 vagy 1 értékkel.

Megjegyzés: Szilárd Leó másféleképp definiálta a bitet;

Egy ***S*** hírforrás valamely *p* valószínűséggel (relatív gyakorisággal) kibocsátott *h* hírének az információtartalma $I(h)=-\log_2 p$ bit.

Shannon 2. tétele (tétel az információ összenyomhatatlanságáról) szerint egy bit információ továbbításához legalább egy bináris számjegy szükséges.

Definíció (összes természetes számok halmaza): M:=[00…00; 11…11]∩ℕ.

Definíció (*LSB, less significant bit*): legkisebb helyiértékű számjegy.

Definíció (*MSB, most significant bit*): legnagyobb helyiértékű számjegy.

Itt LSB a 0. bit, MSB az (n-1). bit.

Vizsgáljuk meg, hány darab természetes szám van!

Könnyen látható: n biten 2^n különböző szám ábrázolható, a legnagyobb szám a $2^n - 1$.

Példa:

Legkisebb szám: $0 = 00000000$

Legnagyobb szám: $2^8 - 1 = 255 = 11111111$

Középiskolából ismert, hogyan lehet felírni a számokat kettes számrendszerben, számrendszerek közötti átváltással.

1.1 Oszthatóság

Definíció (*páros szám*): LSB=0.

A legkisebb szám, a 00000000 is páros szám.

Definíció (*páratlan szám*): LSB=1.

Érdekesség, hogy a legnagyobb szám mindig páratlan.

Megjegyzés: a 2^k-nal osztható számok utolsó (legkisebb) k db bitje 0.

Itt tehát a 2-hatványok a kerek számok.

Következmény (*2-hatvány*): $2^k{}_{⑩} = 100 \ldots 0_{②}$, ahol k db 0-bit szerepel.

Definíció (*paritás*): 1-es bitek számának páros/páratlan volta.

Páros paritású a szám, ha 1-es bitjei száma páros.

Páratlan paritású a szám, ha 1-es bitjei száma páratlan.

Ezek szerint egy szám vagy páros paritású, vagy páratlan paritású, egyszerre nem lehet mindkettő.

n=8 esetén minden páros paritású szám 0-bitjeinek száma is páros, valamint minden páratlan paritású szám 0-bitjeinek száma is páratlan.

1.2 Számrendszerek közti átváltás

Példa: (*átváltás 2-esből 10-esbe*):

01100100=?

bit sorszám	7.	6.	5.	4.	3.	2.	1.	0.
helyiérték	$2^7 = 128$	$2^6 = 64$	$2^5 = 32$	$2^4 = 16$	$2^3 = 8$	$2^2 = 4$	$2^1 = 2$	$2^0 = 1$
	0	1	1	0	0	1	0	0

$01100100 = 0 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0 = 100_{⑩}$.

Példa: (*átváltás 10-esből 2-esbe egyszerűen*):

Algoritmus: Ha a szám maga a 0, akkor 0=00000000, kész.

Ha nem nulla: kihasználjuk, hogy 2-vel való osztáskor csak 0 vagy 1 maradék keletkezhet. A számot addig osztjuk maradékosan 2-vel, amíg 1-et nem kapunk. Ez biztosan bekövetkezik. Végül az 1-et is elosztjuk 2-vel, utolsó maradék az 1.

(Triviális következmény: minden nem-nulla szám első értékes jegye 1.)

A keletkező maradékokat írjuk fel fordított sorrendben, és kész.

Mivel véges az ábrázolás, a megfelelő 0-biteket balról pótolni kell.

100	0
50	0
25	1
12	0
6	0
3	1
1	1
0	

$100_{⑩}=1100100_{②}=01100100$

1.2.1 Átváltás különböző alapú számrendszerek között

Középiskolában A alapú számrendszerbeli számot úgy váltottunk át B alapúra, hogy először a fenti módszerrel áttértünk decimális rendszerbe, majd a részeredményt kellett váltani B alapúra.

Létezik algoritmus, amely hatékonyan átvált közvetlenül A alapúról B alapúra, például a következő algoritmus helyes.

Bemenet:

A∈ℕ a kezdeti számrendszer alapszáma;

B∈ℕ a cél-számrendszer alapszáma;

S∈ℤ az átváltandó, A alapú szám

Kimenet: U∈ℤ az átváltott, B alapú szám.

Előfeltétel: léteznek A, B alapú számrendszerek és S A alapú

Utófeltétel: U B alapú és U=S

```
Eljárás SzámrendszerÁtváltásAB(A, B, S, U)
  Ha előjeles(S) akkor st:=1 különben st:=0
  hossz:=JegyeiSzáma(S)
  Ciklus i:=st-től hossz-ig
    Ha S[i]∈[0; 9] akkor Jegy[i-st]:=S[i]-"0"
    különben Jegy[i-st]:=S[i]-"A"+10
    Elágazás vége
    hossz:=hossz-(1+st)
    i:=0
    Ciklus amíg 2*i≤hossz
      Csere(Jegy[i], Jegy[hossz-i])
    Ciklus vége
    újhossz:=0
    Ciklus
      maradék:=0
      Ciklus i:=hossz-tól 0-ig (-1)-esével
        maradék:=maradék*A+Jegy[i]
        Jegy[i]:=maradék/B
        maradék:=maradék mod B
      Ciklus vége
      U[újhossz]:=maradék
      újhossz:=újhossz+1
      Ciklus amíg hossz≥0 és Jegy[hossz]=0
        hossz:=hossz-1
      Ciklus vége
    Ciklus vége amíg hossz≥0
  Ciklus vége
  Ha sgn(S)=-1 akkor U:=U*(-1)
Eljárás vége.
```

1.3 Természetes számokkal végzett alapműveletek

1.3.1 Összeadás

A középiskolából ismert módon történik az összeadás: összeadótábla alapján, átvitellel.

Összeadótábla:

	0	1
0	(0)0	(0)1
1	(0)1	(1)0

A zárójelbe tett érték az átvitel eggyel magasabb bitre.

Továbbá: 1+1+1=(1)1.

Példa:

```
 01001101   77
+00101101  +45
 01111010  122
```

Mi történik, ha nem elég a jegyek száma?

Példa (n=8):

```
 11111111  255
+00000001 +  1
100000000    0
```

Magyarázat: 100000000=00000000=0 (nincs 8. bit!)

Vegyük észre, hogy amikor a 7. bit fölötti összes bitet „levágjuk”, akkor tulajdonképpen vettük a szám $2^8 = 256$-tal vett osztási maradékát!

Ebben nincs semmi meglepő, hiszen az eddig tárgyalt természetes számok halmaza megfelel a számelméletből ismert „mod 256” maradékosztálynak.

Vagyis, az eredmény helyes, hiszen 0≡256(mod 256).

Példa:

```
 11111111  255
+00000010 +  2
100000001    1
```

1≡257(mod 256).

Példa:

```
 11111110  254
+00000010 +  2
100000000    0
```

0≡256(mod 256).

Ezen egyszerű példákból is látszik, hogy a túlcsordult összegek elvesznek, de ami még érdekesebb, ötletet adhat a negatív számok tárolására, hiszen ha 254+2=0, akkor a 254 lehetne a 2 ellentettje …

1.3.2 Kivonás

A kivonás sem különbözik érdemben a középiskolából ismert művelettől. A kérdés ismét a számhalmazból való kilépés problémája.

Ennyit kell tudni a kivonáshoz:

0-0=0
1-0=1
1-1=0
10-1=01

Példa:

```
 01001011  75
-00100001 -33
 00101010  42
```

Vizsgáljuk meg, mi történik, ha egy számból kivonunk egy nála nagyobbat!

Példa:

```
 00000000   0
-00000001 - 1
111111111 255
```

Vegyük észre, hogy az itt kapott eredmény végtelen sok 1-bitet kellene tartalmazzon! Mivel n=8, így az értékes 11111111=255 eredményt kapjuk, ami jó, mert megfelel az összeadásnál leírtaknak, hiszen 255+1=0 volt. Ez ötletet adhat a 2. fejezetben tárgyalt negatív számok tárolásához …

1.3.3 Szorzás

Szorozhatunk ismételt összeadással, de nem érdemes, hiszen vegyük észre, hogy kettes számrendszerben szorozni nagyon könnyű: egy szám 1-szerese önmaga, 0-szorosa pedig 0. Így a rész-szorzatok nagyon könnyen felírhatók.

Példa:

```
 1101·101
 1101
  0000
+  1101
1000001
```

Ugyanakkor figyelni kell arra, hogy nagy számok szorzásakor nem feltétlenül kapjuk meg a helyes eredményt!

Példa:

20·30=600 kellene legyen. Ehelyett:

```
 10100·11110
 10100
  10100
   10100
    10100
+    00000
1001011000
```

Itt az eredmény 01011000=88, vagyis 20·30=88.

Ez meglepő lehet első ránézésre, pedig nem az: a 88 valóban a 600-nak 256-tal vett osztási maradéka, hiszen a 7. bit fölöttiek levágásával keletkezett. 88≡600(mod 256).

Tízes számrendszerben megszokhattuk, hogy ha kerek számmal szorzunk, akkor csak a számhoz kell megfelelő számú 0 jegyet írni, hogy megkapjuk a megfelelő nagyságrendet.

Ez kettes számrendszerben is így van: 2-hatvánnyal szorozva csak a helyiértékeket kell mozgatni, a megfelelő darabszámú 0-bit beírásával.

Példa:

111·10000=1110000

Következmény: egy szorzat értéke lehet úgy nulla, hogy egyik szorzótényező sem volt nulla.

Példa:

00000010·10000000=100000000=00000000, vagyis: 2·128=256≡0(mod 256).

A szorzás algoritmusának hatékonyságán lehet javítani.

Karatsuba-Ofman algoritmus nagy számok szorzására (1962)

Az ötlet Charles Babbage-től [7] származott, Karatsuba [8] dolgozta ki az algoritmust. Az eredményt Bernstein [9] publikálta.

Legyen adott x és y nagy (n-jegyű) B számrendszerben felírt számok, melyeknek szorzatát kell kiszámítani. (Ha az egyik szám nem n-jegyű, nullás számjegyekkel pótoljuk a jegyeket.)

Bármely m<n –re (gyakran speciálisan ha n=2k akkor m=k) felírható

$x=aB^m+b$

$y=cB^m+d$

ahol $b, d < B^m$.

$xy=(aB^m+b)(cB^m+d)=(ac)B^{2m}+(ad)B^m+(bc)B^m+(bd)=(ac)B^{2m}+(ad+bc)B^m+(bd)$.

Vegyük észre, hogy $ad+bc=(a+b)(c+d)-ac-bd$

Ezt behelyettesítve kapjuk, hogy

$xy=(ac)B^{2m}+[(a+b)(c+d)-ac-bd]B^m+bd$.

Ez viszont már csak 3 szorzást tartalmaz (4 helyett!): ac, bd, (a+b)(c+d).

A módszer rekurzióval tovább ismételhető ezen három szorzatra is, amíg csak 1-jegyű számokat kell szorozni, ami a szorzótábla ismeretében triviális.

Nem használtuk ki a számrendszer alapszámát. Egy kis továbbgondolással látható, hogy a Karatsuba-Ofman algoritmus használható polinomok szorzására is! (x:=B)

1.3.4 Osztás

Osztani lehet sorozatos kivonással, vagy az általános iskolában tanult módszerrel, kettes számrendszerbe átültetve.

Példa:

```
01110111:101=00010111
0111
  1001
   1001
    1001
     100
```

119:5=23, maradék a 4.

Következmény: számokat ugyanúgy osztunk maradékosan, mint polinomokat, ahol az együtthatók játszák a bitek szerepét: mind 0 vagy 1. Érdemes arra odafigyelni, hogy a 0 együtthatójú tagokat is kiírjuk, nehogy osztás közben eltévesszük a helyiértéket!

Érdekesség: a (maradékos) osztás az egyetlen az alapműveletek közül, amellyel nem tudunk kilépni a halmazból: nem kell a számokat a többi műveletnél ismertetett módon „vágni”.

2-hatvánnyal könnyű osztani, csak a kettedesvesszőt kell „balra” mozdítani a megfelelő jegyek számával.

Példa:

10110100:00000100=00101101

10110100:00001000=00010110, és maradék az 1.

Hatványozás: történjen ismételt szorzással! A gyökvonást vagy formálisan tároljuk, vagy használjunk egy sorozat szerinti közelítést a kellő pontossággal.

1.3.4 Gyűrű-tulajdonságok

Vegyük észre, hogy a fenti értelemben vett ábrázolásban a számok halmaza kommutatív gyűrűt alkot a „mod 2^n” összeadásra és szorzásra. Láttuk azonban, hogy ez a gyűrű nem nullosztómentes!

2. Negatív számok

2.1 Komplemens ábrázolás

A példák továbbra is n=8 biten vannak értelmezve.

Definíció (*egyes komplemens kód*): minden bit megfordításával kapott szám.

Minden 0-bit 1-bitté válik, és minden 1-bit 0-vá válik.

Jelölés (*egyes komplemens kód*): ~

Ha tehát $x=s_7s_6s_5s_4s_3s_2s_1s_0②$, akkor $\sim x=(1-s_7)(1-s_6)(1-s_5)(1-s_4)(1-s_3)(1-s_2)(1-s_1)(1-s_0)②$.

Példa: ~01001101=10110010

```
~01001101
 10110010
```

Megjegyzés: egy szám és egyes komplemens kódjának összege a csupa 1-es szám; n=8 esetén 11111111.

Definíció (*kettes komplemens kód*): egy x szám kettes komplemens kódja az a szám, amelyet úgy kapunk, hogy x egyes komplemens kódjához 1-et hozzáadunk.

Jelölés (*kettes komplemens kód*): -

Példa:

-01001101=10110011, mert:

```
~01001101
 10110010
+00000001
 10110011
```

Mi köze lehet egy számnak a kettes komplemenséhez? Adjuk össze őket!

```
 01001101   77
+10110011 +179
100000000    0
```

Vagyis, a kettes komplemens kód mindig pont azt a számot állítja elő, amivel az eredeti számot összeadva 00000000-t kapunk.

Persze, negatív számok tárolásához csak akkor használhatjuk a kettes komplemens kódot, ha tudjuk, hogy egy x szám kettes komplemens kódjának kettes komplemens kódja maga az x szám, azaz: bármely x szám bitjeit megfordítjuk, hozzáadunk 1-et, az eredmény bitjeit megfordítjuk, végül hozzáadunk 1-et, a végeredmény x.

Definíció (*ellentett*): egy x szám ellentettje legyen az x kettes komplemens kódja. (Neumann János javaslata)

Következmény: minden számnak van ellentettje.

Egy szám ellentettjének ellentettje maga a szám.

Megjegyzés: ezzel még nincs definiálva, melyek a pozitív és melyek a negatív számok.

Vegyük észre, hogy nincs mínusz nulla: -0=-00000000=100000000=00000000=0.

A nulla számtól eltekintve, észrevehető, hogy kettes komplemens kód számításakor MSB mindig megváltozik.

Definíció (S, Sign, *előjelbit*): MSB.

Definíció (*pozitív szám*): S=0.

Definíció (*negatív szám*): S=1.

Következmény: a 00000000 pozitív.

Mivel S csak 0 vagy 1 lehet, az egész számok halmaza a pozitív számok halmazának és a negatív számok halmazának diszjunkt uniójából áll.

Vizsgáljuk meg, hány pozitív és hány negatív szám van, n=8 esetén!

Adott a 0=00000000 legkisebb pozitív szám. A legnagyobb: 01111111=127.

11111111 most nem a legnagyobb szám, hiszen negatív, egészen pontosan -1, így ő a legnagyobb negatív szám.

Könnyen látható, hogy 00000000 kettes komplemense önmaga: 00000000. Ez azonban igaz az 10000000 számra is! ~10000000=01111111, így -10000000=10000000! Emiatt szokás ezt a számot nem nullának tekinteni, hanem a legkisebb negatív számnak. n=8 esetén tehát 10000000=-128. Ezért szokás kikötni, hogy 10000000 kettes komplemens kódját ne tekintsük az ő ellentettjének, hogy ne lépjen fel az a probléma, hogy -(-128)=-128 lenne, mivel +128 nincs. Nullára szimmetrikus számhalmaznál nem kellene ilyen kikötéseket tenni.

Itt tehát a negatív számok: -128 … -1: 128 db, a pozitív számok: 0 … 127: 128 db.

Általában is: 2^{n-1} db negatív és 2^{n-1} db pozitív szám van, de mint láttuk, a számhalmaz nullára nem szimmetrikus!

2.2 Egész számok halmaza

Fentiek alapján fel lehet sorolni az egész számokat; néhány egész szám következik. (n=8)

10000000	-128
…	…
11111101	-3
11111110	-2
11111111	-1
00000000	0
00000001	1
00000010	2
00000011	3
…	…
01111111	127

2.3 Műveletek egész számokkal

Láttuk, hogy egy szám és ellentettje összege 00000000, ahogy megszoktuk. Azonban a kettes komplemens kódú ábrázolás tartogat néhány meglepetést, ha az alapműveleteket a már ismert módon alkalmazzuk.

2.3.1 Túlcsordulás előjelről

Az összeadás művelete nincs tekintettel arra, hogy előjelként definiáltuk MSB-t. Az előjelbitek összeadása túlcsordulást vonhat maga után.

Ha összeadunk két n-bites számot, akkor két eset lehetséges.

1. eset: az eredmény is elfér n biten.

2. eset: (n+1)-bites eredményt kapunk, mert legalább egy 1-bit átvitel keletkezett MSB-ről. Ekkor az (n+1). bit 1-es.

1. példa:

```
 00100001  33
+00001000 + 8
 00101001  41
```

Két „kis abszolútértékű” számot adtunk össze, pozitív számok összege pozitív.

2. példa:

```
 11001011  -53
+10101101 +-83
101111000  120
```

Itt az első meglepő eredmény: két negatív szám összege pozitív lett! Mivel S-ről történt az 1-bit túlcsordulása, ezt **előjelről történő túlcsordulás**-nak nevezzük.

3. példa:

```
 11001011  -53
+11101110 +-18
110111001  -71
```

Az eredmény helyes (n=8), de mivel mindkét szám negatív volt, keletkezett egy előjelről történő túlcsordulás.

Mivel vannak példák helyes és helytelen eredményekre is előjelről történő túlcsordulás után, szükség lesz a két esetet szétválasztani, hiszen abból, hogy előjelről történő túlcsordulás történt, még nem következik, hogy az eredmény hibás lenne.

4. példa:

```
 00110101   53
+11101110 +-18
100100011   35
```

Mint láthatjuk, az eredmény előjelhelyes, de az összeadás ismét előjelről történő túlcsordulást eredményezett.

5. példa:

Tekintsünk egy olyan esetet, mikor egy számot éppen az ellentettjével adjuk össze!

```
 10101010  -86
+01010110 + 86
100000000    0
```

Az eredmény valóban 0, de ismét előjelről történő túlcsordulás keletkezett.

2.3.2 Túlcsordulás előjelre

Nem csak a legmagasabb helyiértékekről lehet túlcsordulást előállítani. Gyakran előfordul, hogy kis számok összege vált rosszul előjelet.

6. példa:

```
 01111111  127
+00000001 +  1
 10000000 -128
```

Előjelezés nélkül helyes lenne, de a korábbiak alapján ez nem jó eredmény!

Előjelre történő túlcsordulás-nak nevezzük azt a jelenséget, amikor az eredmény az összeadás folytán előjelhibás lesz.

7. példa:

```
 01101000  104
+01101001 +105
 11010001  -47
```

Két pozitív szám összege ismét negatív lett.

8. példa:

Egy negatív és egy kis pozitív szám összege pozitív lehet:

```
 11001001  -55
+00111010 + 58
100000011    3
```

9. példa:

```
 11010001  -47
+01001001 + 73
100011010   26
```

Itt az eredmény helyes és előjelhelyes, és MSB-ről csordult túl, ezért megfelel a 4. és 5. példáknak.

10. példa:

```
 11010001  -47
+11001001 +-55
110011010 -102
```

Az eredmény szintén helyes és előjelhelyes; MSB-ről csordult túl, hasonlóan a 3. példához.

Láthattuk tehát, miképp csordulhatnak túl az értékek kettes komplemens kódú ábrázolásnál. Vizsgáljuk meg, mik következnek mindezekből!

2.3.3 Asszociativitás

Könnyen látható, hogy az összeadás asszociatív (akárhány biten), hiszen nem csinálunk mást, mint kettes számrendszerben adjuk össze a számokat. Az egyetlen pont, ahol elvileg elromolhatna, a túlcsordulás.

Vizsgáljuk meg az összes esetet, vajon mi történik, ha az előjelbit körül túlcsordulás történik a különböző sorrendben vett összeadások során!

Kérdés tehát, hogy ha adott *a, b, c* n-bites egész szám, akkor mindig igaz-e, hogy $(a+b)+c=a+(b+c)$.

Mivel máshol nem lehet baj, elegendő és szükséges az előjelbit alatti bittől vizsgálni az értékeket. Ez $2^6+4=64+4=68$ lehetőség, hiszen ha egymás alá írjuk a három összeadandót, 2^3-féle bit állhat az előjelbit alatti helyiértéken, tőlük függetlenül 2^3-féle előjelbit fordulhat elő, és még érkezhet egy +1 átvitel az eleve 1+1 összeghez, amihez vagy 0-t, vagy 1-et kell adnunk, mindkét sorrendben.

Az összes számítás n=8 bitre megtalálható e dokumentum DVD mellékletén, a kiszámító programmal, melynek tetszőleges n>3 bitmélységet megadhatunk.

A szorzás asszociativitása elsőre nehezebbnek tűnhet, hiszen nagyon érdekes szorzatok állhatnak elő ebben a számábrázolásban, ugyanakkor a fentiekhez hasonlóan be lehet látni, hogy bármely *a, b, c* n-bites egészekre $(ab)c=a(bc)$. Ne lepődjünk meg, ha a legtöbb kiszámolható szorzat túlcsordul!

A szorzat asszociativitásának n=8 esete valamint a program megtalálható a DVD mellékleten.

Példa:

```
(-81·83)·126=6
-81·(83·126)=6
```

Példa:

```
(-56·44)·(-80)=0
-56·(44·(-80))=0
```

2.3.4 Disztributivitás

Meglepő eredmény, de a disztributivitás is teljesül! Legalább annyira meglepő példákkal, mint az eddigiek.

Példa:

```
(26-62)·14=8
26·14+(-62)·14=8
```

Példa:

(-64-68)·64=0

-64·64-68·64=0

További példák a DVD mellékleten.

3. Bitműveletek

Tekintsük az L={0; 1} halmaz önmagába való leképezéseit. A bitek halmazából saját magába képező függvényeket bitműveleteknek nevezzük.

A bitműveletek hasonlóságot mutatnak a matematikai logikában tanult logikai műveletekhez.

3.1 Egy és kétváltozós bitműveletek

3.1.1 Tagadás

Definíció (tagadás, NEM)**:** Legyen ┐ : L→L.

┐(0)=1;

┐(1)=0.

Ezt a ┐ jellel jelölt bitműveletet **tagadás**nak nevezzük.

Az elnevezés onnan ered, hogy az argumentumként kapott bitet megfordítja.

A bitműveleteket, mint sok függvényt, megadhatunk táblázattal, a megfelelő hozzárendeléseket jelölve. Ennek a neve **igazságtábla**. Hasonló a szorzótáblához, csak szorzás helyett a megadott bitműveletet kell alkalmazni.

A tagadás igazságtáblája:

A	┐A
0	1
1	0

A tagadás tehát egyváltozós művelet.

A két „nullaváltozós” konstans bitművelet:

Definíció (azonosan igaz bitművelet): I: L→L; I(0)=1; I(1)=1.

Definíció (azonosan hamis bitművelet): H: L→L; H(0)=0; H(1)=0.

Mint később látni fogjuk, ezek a műveletek egyébként mind kifejezhetők a következő bitműveletekkel: ∧(és), ∨(vagy), ┐(nem).

A továbbiakban kétváltozós bitműveleteket vizsgálunk. Ezek az f: L×L→L függvények. Ebből pontosan 16 különböző létezik, közülük néhánynak különös jelentősége van.

3.1.2 Konjunkció (ÉS)

Definíció (konjunkció, ÉS): bitművelet, amely csak akkor ad 1-et, ha minden argumentuma 1.

Jelölés: ∧, infix alakban szokás írni.

Igazságtáblája:

A	B	A∧B
0	0	0
0	1	0
1	0	0
1	1	1

3.1.3 Diszjunkció (megengedő VAGY)

Definíció (diszjunkció, megengedő VAGY): bitművelet, amely csak akkor ad 0-át, ha minden argumentuma 0.

Jelölés: ∨, infix alakban szokás írni.

Igazságtáblája:

A	B	A∨B
0	0	0
0	1	1
1	0	1
1	1	1

3.1.4 Implikáció

Definíció (implikáció): kétváltozós bitművelet az alábbi igazságtáblával:

A	B	A→B
0	0	1
0	1	1
1	0	0
1	1	1

Jelölés: →, infix alakban szokás írni.

3.1.5 Ekvivalencia

Definíció (ekvivalencia): kétváltozós bitművelet, amely csak akkor ad 1-et, ha mindkét argumentuma megegyezik.

Jelölés: ↔, infix alakban szokás írni.

Igazságtáblája:

A	B	A↔B
0	0	1
0	1	0
1	0	0
1	1	1

Az előbbi tagadása:

3.1.6 Antivalencia (kizáró VAGY)

Definíció (antivalencia, kizáró VAGY, eXclusive OR, XOR): kétváltozós bitművelet, amely csak akkor ad 1-et, ha a két argumentum különböző.

Jelölés: ⊕, infix alakban szokás írni.

Igazságtáblája:

A	B	A⊕B
0	0	0
0	1	1
1	0	1
1	1	0

Vegyük észre, hogy a XOR művelet hasonlít az n=1 bites számok összeadására!

3.1.7 NOR (Not OR)

Definíció (NOR): a NOR bitművelet a VAGY művelet eredményének tagadása.

Igazságtáblája:

A	B	A NOR B
0	0	1
0	1	0
1	0	0
1	1	0

3.1.8 Sheffer-művelet, NAND (Not AND)

Definíció (NAND): A Henry M. Shefferről elnevezett kétváltozós bitművelet az ÉS művelet eredményének tagadása.

Jelölés: |, infix alakban szokás írni.

Igazságtáblája:

A	B	A|B
0	0	1
0	1	1
1	0	1
1	1	0

3.2 A teljes igazság

Mind a 16 különböző bitművelet igazságtáblája következik egyben:

A	B	M_0	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}	M_{11}	M_{12}	M_{13}	M_{14}	M_{15}
0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
0	1	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0
1	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
1	1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0

Ahol:

M_0=I

M_1=A|B

M_2=A→B

M_3=┐A

M_4=B→A

M_5=┐B

M_6=A↔B

M_7=A NOR B

M_8=A∨B

M_9=A⊕B

M_{10}=B

M_{11}=┐(B→A)

M_{12}=A

M_{13}=┐(A→B)

M_{14}=A∧B

M_{15}=H

3.3 Bitműveletek tételei

Legyenek A, B, C bitek, azaz L={0; 1} halmazbeliek. A tételek bizonyítása egyszerűen elvégezhető például az igazságtáblák alapján.

3.3.1 Tételek tagadásra, konjunkcióra, diszjunkcióra

Tétel (tagadás tagadása): ┐(┐A)=A.

Tétel (a konjunkció kommutatív): A∧B=B∧A.

Tétel (a diszjunkció kommutatív): A∨B=B∨A.

Tétel (a konjunkció asszociatív): A∧(B∧C)=(A∧B)∧C.

Tétel (a diszjunkció asszociatív): A∨(B∨C)=(A∨B)∨C.

Tétel (a konjunkció disztributív a diszjunkcióra): A∧(B∨C)=(A∧B)∨(A∧C).

Tétel (a diszjunkció disztributív a konjunkcióra): A∨(B∧C)=(A∨B)∧(A∨C).

Elnyelési tételek: a konjunkcióra és a diszjunkcióra teljesülnek az alábbi elnyelési tulajdonságok.

***Tétel*:** A∨(A∧B)=A.

Tétel: A∧(A∨B)=A.

De Morgan azonosságok:

Tétel: ┐(A∧B)=(┐A)∨(┐B).

Tétel: ┐(A∨B)=(┐A)∧(┐B).

Következmény: A∧B=┐((┐A)∨(┐B)).

Következmény: A∨B=┐((┐A)∧(┐B)).

Tétel (komplementer képzés): A∨(┐A)=1.

Tétel (komplementer képzés): A∧(┐A)=0.

Továbbá nyilván ┐0=1 és ┐1=0.

További következmények:

Idempotencia:

Tétel: A∨A=A.

Tétel: A∧A=A.

Korlátosság:

Tétel: A∨0=A.

Tétel: A∨1=1.

Tétel: A∧1=A.

Tétel: A∧0=0.

3.3.2 A Sheffer-féle NAND teljessége

Lemma: minden kétváltozós bitművelet kifejezhető a ┐, ∧, ∨ műveletekkel.

Bizonyítás: elég megmutatni, hogy a következő kétváltozós műveletek kifejezhetők:

- azonosan igaz: I
- azonosan hamis: H
- implikáció: →
- ekvivalencia: ↔
- kizáró vagy: XOR
- NOR

Legyenek A, B bitek. Ekkor:

I=A∨(┐A).

H=A∧(┐A).

A→B=┐(A∧(┐B)).

A↔B=┐(A⊕B), és:

A⊕B=(A∧(┐B))∨(B∧(┐A)) (szimmetrikus differencia), tehát

A↔B=┐((A∧(┐B))∨(B∧(┐A))).

A NOR B=┐(A∨B).

Tétel: minden kétváltozós bitművelet kifejezhető a Sheffer-féle NAND (|) művelettel.

Bizonyítás: elég megmutatni, hogy ┐, ∧, ∨ kétváltozós bitműveletek kifejezhetők, mivel a Lemma alapján minden más műveletet már ezekkel ki tudtunk fejezni.

Legyenek A, B bitek. Ekkor:

┐A=A|A.

A∧B=┐(A|B)=(A|B)|(A|B).

A∨B kifejezéséhez felhasználhatjuk azt a 3.3.1 alatti következményt, hogy A∨B=┐((┐A)∧(┐B)).

Így:

A∨B=┐((┐A)∧(┐B))=┐(((┐A)|(┐B))|((┐A)|(┐B)))=┐(((A|A)|(B|B))|((A|A)|(B|B)))=
(((A|A)|(B|B))|((A|A)|(B|B)))|(((A|A)|(B|B))|((A|A)|(B|B))).

Definíció (háló): Az (M, ∧, ∨) mint algebrai struktúrát hálónak nevezzük, ha tetszőleges a, b, c∈M számokra teljesülnek a következők:

1. a∨b=b∨a, a∧b=b∧a (kommutativitás),
2. (a∨b)∨c=a∨(b∨c), (a∧b)∧c=a∧(b∧c) (asszociativitás),
3. a∧(a∨b)=a, a∨(a∧b)=a (elnyelési azonosságok).

Tétel: Ha M={egész számok halmaza}\{100…00}, ∨ a megengedő logikai VAGY művelet, valamint ∧ az ÉS művelet, akkor az (M, ∧, ∨) struktúra egységelemes háló, amely ráadásul disztributív.

Az így alkotott logikai rendszer neve: **Boole-algebra**.

Sőt:

Tétel: Az így megadott M halmaz minden a elemére teljesül, hogy létezik olyan $\bar{a}$ elem, hogy $a \vee \bar{a} = 1$ és $a \wedge \bar{a} = 0$.

Definíció (*komplementer bit*): $\bar{a}$-t az a komplementerének nevezzük.

Könnyű belátni, hogy:

Tétel: $a = \bar{b}$ pontosan akkor, ha $a = \neg$ b. Hasonlóképpen $\bar{a} = b$ pontosan akkor, ha ¬ a = b.

3.3.3 Bitenkénti logikai műveletek

Vezessük be az összes, előzőekben tárgyalt műveletet bitenként végrehajtva két tetszőleges egész számra!

Példa az ∧ mint bitenkénti műveletre:

```
 01001101   77
∧00101101  ∧45
 00001101   13
```

3.3.4 Megoldás az egész számok túlcsordulási hibáira

A fentiekben már bevezetett S (Sign) előjelbit mellé vezessünk be még néhány fontos jelzőbitet. [10]

Definíció (Z, Zero, *nullát jelző bit*): értéke legyen 1, ha a művelet eredménye ≡0(mod 2^n).

Definíció (C, Carry, *átvitelbit*):

Összeadás esetén: Értéke legyen 1, ha a művelet elvégzése során átvitel keletkezik MSB-n.

Kivonás esetén: A Carry bit törlődjön (azaz legyen 0), ha egy művelet elvégzése során áthozatal történik MSB-n (előjel nélküli alulcsordulás, az eredmény <0, azaz nagyobb számot vonunk ki kisebb számból). A Carry bit értéke 1 legyen, ha nem történt áthozatal.

Következmény: Egyszerre Z=1 és C=0 állapot nem következhet be, mivel nullát csak akkor kaphatunk eredményül, ha két egyenlő számot vonunk ki egymásból, ilyenkor viszont nem keletkezik átvitel.

Definíció (O, Overflow, *túlcsordulást jelző bit*): Ha két pozitív szám összege negatív, legyen O=1; illetve ha két negatív szám összege pozitív, legyen O=1. Minden más esetben legyen O=0.

Következmény: Vegyük észre, hogy különböző előjelű számok összeadása esetén az eredmény helyes lesz, így az Overflow bit nulla marad.

A megadott definíció szerint tehát O=1 esete jelzi, ha hiba történt és az eredmény korrekcióra vár.

Tétel: O=S⊕C. (triviális)

A kizáró vagy (XOR) művelet pontosan azt adja, amit a túlcsordulást jelző bit definíciója kíván.

Vigyázat! A tétel csak kettes számrendszerben teljesül minden O, S, C hármasra! Magasabb alapú számrendszerben az imént bevezetett átvitelbit nem keletkeztet előjelváltást az (MSB-1). számjegyről!

Túlcsordulás megoldása tehát: O=1 esetén vegyük az eredmény ellentettjét!

Másképpen: a helyes előjelbit túlcsordulás esetén S⊕O.

3.3.5 Relációk

Az egyenlőség, nem-egyenlőség relációk a már megszokott értelemben használhatók.

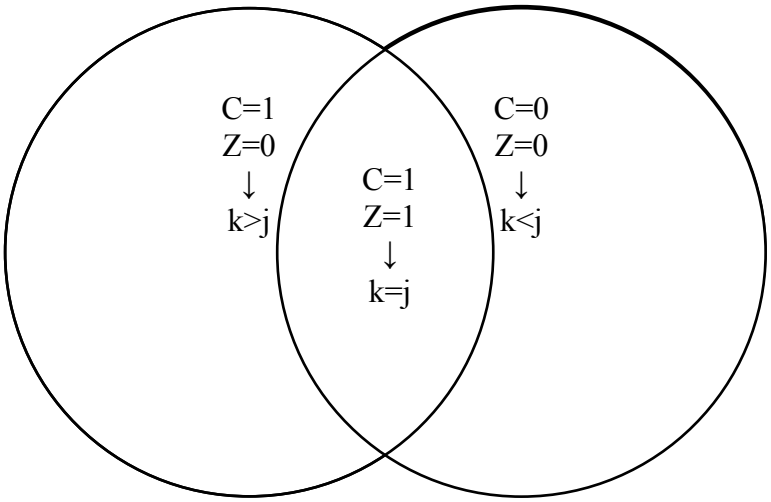
Két szám egyenlő, ha minden bit ugyanazon sorrendben felsorolva egyenlő és az előjelbitek is megegyeznek. Ha bármelyik feltétel nem teljesül, a két szám nem lehet egyenlő.

Rendezési relációk

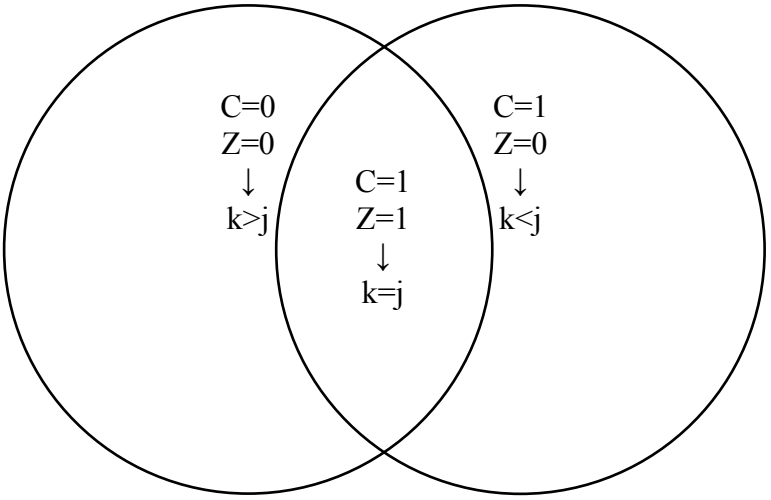
A kisebb (<), nagyobb (>), kisebb (megengedő-)vagy egyenlő (≤), nagyobb (megengedő-)vagy egyenlő (≥) relációkat kivonással vizsgáljuk.

Legyen adott a k és j szám. Tekintsük két esetnek a k-j, majd a j-k különbségek elvégzésekor kapott jelzőbitek állását. [11]

k-j



j-k



Az ábrákon értelemszerűen kiolvashatók az egyes esetek.

4. A valós számok

Valós számok véges ábrázolásának lehetőségeit számba véve alapvető akadályokba ütközhetünk. Például, bizonyos axiómák nem teljesülnek!

A továbbiakban feltételezzük, hogy a használt számrendszer alapszáma a B=2. (Neumann-elv)

4.1 A gépi számok numerikus modellje

A számítógépek lebegőpontos számábrázolással ábrázolják leggyakrabban a valós számok véges halmazát. Ennek egy lehetséges modellje a numerikus analízis tárgyalásában (Krebsz [12]):

Definíció (*normalizált lebegőpontos szám*): Legyen $t \in \mathbb{N}$, $k \in \mathbb{Z}$, $m = \sum_{i=1}^{t} m_i \cdot 2^{-i}$, ahol $m_1=1$, $m_i \in \{0; 1\}$ (i=1, 2, …, t). Ekkor az $a = \pm\left(\sum_{i=1}^{t} m_i \cdot 2^{-i}\right) \cdot 2^k = \pm m \cdot 2^k$ alakú számot normalizált **lebegőpontos szám**nak nevezzük.

A t szám az m-mel jelölt **mantissza** hossza, míg k a szám **karakterisztikája**.

Jelölés: $a = \pm[m_1 \dots m_t \mid k]$

A gépi számok halmazát M=M(t, k^-, k^+) jelöli, ahol

$M = \{a | a = \pm m \cdot 2^k, k^- \leq k \leq k^+\} \cup \{0\}$.

Fontos **következmények**:

1. Létezik legnagyobb valós szám! $\max M = M_\infty = [11 \dots 1|k] = (1 - 2^{-t}) \cdot 2^{k^+}$
2. Létezik legkisebb pozitív szám! $\varepsilon_0 = \frac{1}{2} \cdot 2^{k^-} = 2^{k^- - 1}$

Mindezen axióma-sértések ellenére megkíséreljük megfeleltetni a valós számoknak lebegőpontos számokat.

Definíció (*input-függvény*): Az fl: $\mathbb{R} \to M$ függvény input-függvény, ha

$$fl(x) = \begin{cases} M_\infty, ha\ x > M_\infty, \\ -M_\infty, ha\ x < -M_\infty, \\ az\ x - hez\ legközelebbi\ gépi\ szám\ a\ kerekítés\ ismert\ szabálya\ szerint, ha\ |x| \leq M_\infty \end{cases}$$

Sajnos ez a megfeleltetés is hibát generál.

Tétel (*input hiba*): Minden $|x| < M_\infty$ valós szám esetén

$$|x - fl(x)| \leq \begin{cases} \varepsilon_0, ha\ |x| < \varepsilon_0 \\ \frac{1}{2}|x| \cdot 2^{1-t}, ha\ |x| \geq \varepsilon_0 \end{cases}$$

$|x| \geq \varepsilon_0$ esetén az ábrázolt szám relatív hibája 2^{-t}, azaz csak a mantissza hosszától függ.

Példa

Vizsgáljuk meg az M=M(8, -4, +4) lebegőpontos számok halmazát!

m_1=1; m_2, m_3, … , $m_8 \epsilon \{0; 1\}$, tehát 2^7=128 féle mantissza lehetséges és -4, -3, … , 3, 4 lehetnek a karakterisztikák (9-féle). Így negatív számokkal együtt 2·128·9+1=2305 különböző eleme lehet az M halmaznak a nullát is beleszámítva.

Ebben a rendszerben a legnagyobb valós szám és a legkisebb pozitív szám:

$$M_\infty = [11111111|4] = (1 - 2^{-8}) \cdot 2^4 = \frac{255}{256} \cdot 16 = \frac{255}{16} = 15{,}9375$$

$$\varepsilon_0 = [10000000| - 4] = \frac{1}{2} \cdot 2^{-4} = \frac{1}{32} = 0{,}03125$$

Egyáltalán nem mindegy, a mantisszára és a karakterisztikára hány bit van felosztva.

Egy másik **példa**:

M=M(3, -2, 2) gépi számok halmazában határozzuk meg, a π számhoz mely lebegőpontos szám tartozik az input-függvény szerint?

$$[110|2] = \left(\frac{1}{2} + \frac{1}{4}\right) \cdot 2^2 = 3 < \pi$$

$$[111|2] = \left(\frac{1}{2} + \frac{1}{4} + \frac{1}{8}\right) \cdot 2^2 = 3{,}5 > \pi$$

Mivel $\pi - 3 < 3{,}5 - \pi$, ezért $fl(\pi) = 3$.

Input-hiba: $|\pi - 3| \approx 0{,}14$, jelentősnek mondható.

De ami még sokkal nagyobb probléma az ábrázolásban:

Tétel: A gépi számok körében az **összeadás nem asszociatív** művelet.

Például az előbbi M=M(3, -2, 2) halmazban $[110|2] = 3, [100| - 1] = \frac{1}{4}$.

$fl\left(3 + \frac{1}{4}\right) = [110|2] = 3$, tehát $fl\left(\left(3 + \frac{1}{4}\right) + \frac{1}{4}\right) = 3$.

$fl\left(\frac{1}{4} + \frac{1}{4}\right) = [100|0] = \frac{1}{2}$, így $fl\left(3 + \left(\frac{1}{4} + \frac{1}{4}\right)\right) = [111|2] = 3{,}5$.

4.2 Szabványos lebegőpontos számábrázolás

Elsősorban az informatika nyomására és a hibák minimalizásáért a valós számok véges ábrázolását az Institute of Electronics Engineers szabványosította, neve **IEEE 754**. [13]

A szabvány háromféle pontosságot definiál:

1. egyszeres (32 bites)
2. dupla (64 bites)
3. kiterjesztett (80 bites)

A bitek kiosztása a következő táblázat szerint működik:

	előjel	karakterisztika	mantissza
single real	1 bit (Sign)	8 bit	23 bit
double float	1 bit (Sign)	11 bit	52 bit
extended comp.	1 bit (Sign)	15 bit	64 bit

Példa (*single real*):

Egyszeres pontosság esetén a 32 bit a szabvány szerint tehát a következőképp van kiosztva:

```
S KKKKKKKK MMMMMMMMMMMMMMMMMMMMMMM
```

Például az 1 szám így néz ki:

```
0 01111111 00000000000000000000000
```

Sign=0 (pozitív szám); a karakterisztika 127, ha ebből kivonjuk a nagyságrend szerinti eltolást (127), éppen nullát kapunk. A mantissza nulla.

A szabványbeállítások is tartalmaznak hibát. Például az 1,3 végtelen kettedestört alakban 1010011001100110011001100110011001100110011001100 … , azonban a mantissza csak véges hosszú. Kerekítéssel 01001100110011001100110 adódik, ami decimálisan 1,2999999523162842. 6 tizedesjegy pontossággal 1,3-at kapunk, de például 8 tizedesjegyre 1,29999995. Megfordítva, a 8. tizedesjegytől már nem kaphatunk pontos értékeket a szükséges kerekítés miatt. Másszóval az egyszeres pontosság – értéktől függően – 6-7 tizedesjegyig pontos. Dupla pontosság esetén ez 15-16 tizedesjegy.

A kerekítések további anomáliákat okozhatnak, például jegyenként nem lehet összehasonlítással megállapítani, hogy két valós szám egyenlő-e? $0{,}1 \cdot 10 \neq 1$?

Gyakori módszer ehelyett, hogy a két szám különbségének és/vagy hányadosának abszolútértékét hasonlítjuk egy kellően alacsony küszöbértékkel.

Speciális lebegőpontos számok és nem-számok

Jelölés (*nem-szám*): NaN (Not a Number)

Egyszeres pontosság esetén a következő számok különleges jelentőséggel bírnak:

```
0 11111111 00000000000000000000000 - +∞ (pozitív végtelen)
1 11111111 00000000000000000000000 - -∞ (negatív végtelen)
S 11111111 MMMMMMMMMMMMMMMMMMMMMMM – NaN (ha a mantissza nem 0)
0 00000000 00000000000000000000000 - +0 (pozitív nulla)
1 00000000 00000000000000000000000 - -0 (negatív nulla)
S 00000000 MMMMMMMMMMMMMMMMMMMMMMM – denormált NaN (ha a mantissza nem 0)
```

4.2.1 Műveletek lebegőpontos számokkal

A közönséges (nem-speciális) lebegőpontos számokkal ugyanazon műveletek végezhetők, mint a gépi számok numerikus modelljében tárgyaltak.

A szokásos műveleti tulajdonságok megtartása érdekében a következőket mondjuk ki:

- $NaN \pm x = NaN$
- $NaN \cdot x = NaN$
- $NaN / x = NaN$
- $x / NaN = NaN$
- $\infty + \infty = \infty$
- $\infty - \infty = NaN$
- $\infty / (\pm\infty) = NaN$
- $\infty \cdot \infty = \infty$
- $\infty \cdot (-\infty) = NaN$
- $(-\infty) \cdot (-\infty) = \infty$
- $x / (\pm 0) = NaN$

5. Véges ábrázolás alkalmazásai

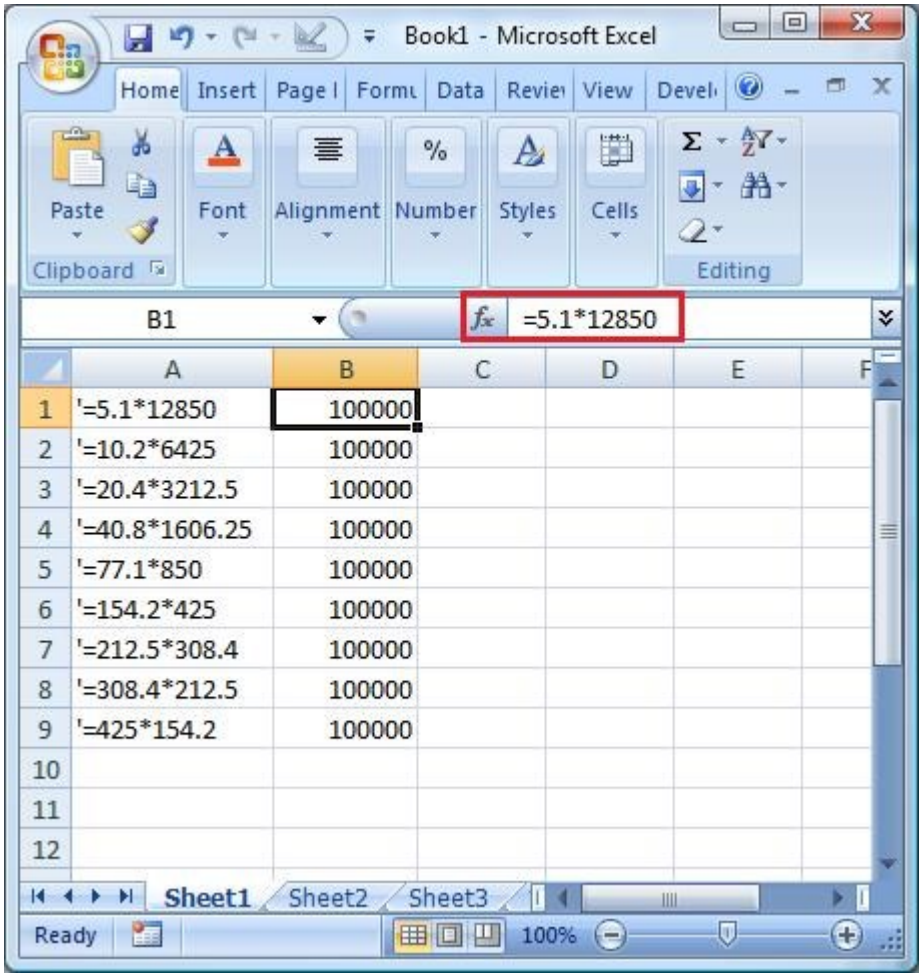
5.1 Excel szorzás bug (2007)

2007-ben, a Microsoft friss kiadású irodai programcsomagjában egy rendkívül érdekes hibajelenségre figyeltek fel a felhasználók. A táblázatkezelő cellájába a következő képletet írva:

```
=850*77.1
```

a helyes 65535 helyett a kapott eredmény **100000**.

Nem csak ez az egyetlen szorzás adott fals eredményt.



[14]

Hogyan is lehet ez?

A probléma leírása és a megoldás az Office hivatalos blogjában olvasható [15]:

„Yesterday evening we were alerted to an issue in Excel 2007 (and Excel Services 2007) involving calculation of numbers around 65,535. The first example that we heard about was =77.1*850, but it became clear from our testing as well as additional reports that this was just one instance where Excel 2007 would return a value of 100,000 instead of 65,535.”

„So what, specifically, are the values that cause this display problem? Of the 9.214*10^18 different floating point numbers […] that Excel 2007 can store, there are 6 floating point numbers (using binary representation) between 65534.99999999995 and 65535, and 6 between 65535.99999999995 and 65536 that cause this problem.”

Azaz: a lebegőpontos számábrázolás pontatlansága miatt a fejlesztők nem vették figyelembe a 11. tizedesjegytől a pontos értékeket!

Természetesen a megoldás azóta megszületett, az Office 2007 SP1 javítja a hibát.

5.2 Lebegőpontos ábrázolás matematikai szoftverekben

A matematikai célú szoftverek esetén különösen fontos, hogy a megszokott valós számokat milyen formában, milyen pontossággal tároljuk.

5.2.1 Matlab, Octave

Matlab-ban és a vele kompatibilitást vállaló Octave rendszerben a következő egész és valós típusok, konstansok, konverziók léteznek:

Azonosító	**Leírás**
double	szabványos duplapontosságú lebegőpontos szám
single	szabványos szimpla pontosságú lebegőpontos szám
int8	8 bites egész szám
int16	16 bites egész szám
int32	32 bites egész szám
int64	64 bites egész szám
uint8	8 bites természetes szám
uint16	16 bites természetes szám
uint32	32 bites természetes szám
uint64	64 bites természetes szám
Inf	végtelen (előjeles)
NaN	nem-szám (előjel nélküli)
realmax	legnagyobb lebegőpontos szám
realmin	legkisebb pozitív lebegőpontos szám

5.2.2 Maple

A Maple háromféle számtípust különböztet meg:

1. integer: 158 jegyű egész szám (például a 100! száz faktoriális pont kifér)
2. float/real: 15 tizedesjegy pontosságú lebegőpontos szám (duplapontosságúnak tűnik)
3. complex: real és imaginary részek komplex számok ábrázolásához

5.2.3 R (statisztika)

Az R külön kezel 10-es és külön 2-es számrendszerbeli számokat!

Decimálisan jegyenként tárolt számok típusa: numeric (BCD).

Bináris számok: integer.

Létezik Complex típus komplex számokhoz, valamint Logical bit és bitműveletek.

6. A kettes számrendszer tanítása szakközépiskola 9. évfolyamán

Tekintettel arra, hogy kilencedikben még nem feltétlenül ismerik a tanulók a polinomokat, más megközelítésben kell tárgyalni a számrendszereket.

6.1 1. óra: Bevezető óra

A bevezető órán a következő tartalmaknak kell elhangoznia:

Egy tetszőleges mennyiség mérőszámát többféle számrendszerben is megadhatjuk.
Napjainkban a 10-es (decimális) számrendszert használjuk. Elterjedésében közrejátszhatott 10 ujjunk, melyek mindig kéznél voltak és segítséget nyújtottak alapvető matematikai műveletek elvégzésében.
Ismeretes, hogy a decimális számrendszer helyiértékes, mivel ugyanaz a számjegy más-más értékű aszerint, hogy hol helyezkedik el a számban.
Tízes számrendszerben tíz számjegyet használunk: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. A felsorolás egyben a számok alaki értéke.
A számjegy tényleges értéke helyiértéke attól függ, hogy a szám melyik pozíciójában áll, mert az alaki érték még megszorzódik az alapszám (10-es számrendszer esetén: 10) adott pozíciója szerint hatványával.
Példa:
125=1*100+2*10+5
Ugyanez hatvány alakban:
$125=1*10^2+2*10^1+5*10^0$
Feladatok: (ilyen jellegűekből sokat)

1. Írd be a hiányzó számokat! 9876=□*1000+□*100+□*10+□*1
2. 2134=2*□+1*□+3*□+4*□
3. $3012=□*10^{□}+□*10^{□}+□*10^{□}+□*10^{□}$

6.2 2. óra: Kettes és más alapú számrendszerek értelmezése

A kettes számrendszer is helyiértékes. Ugyanaz a számjegy más-más értékű aszerint, hogy hol helyezkedik el a számban.
A bináris (2-es) számrendszer alapszáma a 2.

A kettes számrendszerben két számjegyet használunk: 0 és 1.

A számjegy tényleges értéke helyiértéke attól függ, hogy a szám melyik pozíciójában áll, mert az alaki érték még megszorzódik a alapszám (2-es számrendszer esetén: 2) adott pozíciója szerint hatványával.

Példa:

1011 = 1*8 + 0*4 + 1*2 + 1*1.

Ugyanez hatvány alakban:

$1011 = 1*2^3 + 0*2^2 + 1*2^1 + 1*2^0$

Egy számjegyet 1 bit-nek is hívnak. A helyiértékek kettő hatványaiként írhatók le.

A helyiértékek elnevezése 2-es számrendszerben: egyesek, kettesek, négyesek, nyolcasok, …

Átváltás 2-esből 10-es számrendszerbe

Egy kettes számrendszerbeli számot hatvány alakból egyszerűen átalakíthatunk 10-e számrendszerbe.

Az előbbi példában szereplő 1011 számot például így alakíthatjuk át:

```
1 db egyes   => 1*1 = 1
1 db kettes  => 1*2 = 2
0 db négyes  => 0*4 = 0
1 db nyolcas => 1*8 = 8
----------------------
Összesen:            11 (tízesben)
```

Feladatok:

1. Írd be a hiányzó számokat! $1010111=\square*2^{\square}+\square*2^{\square}+\square*2^{\square}+\square*2^{\square}$
2. Váltsd át a következő bináris számokat tízes számrendszerbe! 100111=_____ …

Más (B) alapú számrendszerek: hasonlóképpen kell értelmezni a B alapú számokat, mint a binárisokat, csak itt a helyiértékek a B hatványai!

B>10 esetén a 9-nél nagyobb számjegyeket jelöljük a következő megfeleltetéssel:

A:=10, B:=11, C:=12, D:=13, E:=14, F:=15, …

6.3 3. óra: Decimális számok átváltása

Próbáljuk a tízes számrendszerbeli számokat átváltani más alapú számrendszerekbe!

Megoldás:

1. Keressük meg az alapszám legnagyobb hatványát, amely még megvan az átváltandó számban, maradékos osztással (azaz a hányados nem nulla).
2. Vonjuk le ezt a hatványt az átváltandó számból, és ahányszor ezt megtehetjük (maradékos osztás hányadosa!), írjuk le a megoldás egy újabb számjegyeként!
3. Ha a maradék nulla, kész vagyunk. Ha nem nulla, folytassuk az 1. számú lépéssel újra!
4. Az esetlegesen kihagyott helyiértékeken az alaki érték nulla!

Példa:

Váltsuk át az 1000 számot 5-ös számrendszerbe!

Megoldás:

$5^5=625$, de $5^6=3125$ (sok).

1000:625=**1**

375

$5^4=125$

375:125=**3**

0

Kimaradt még az 5-nek 3., 2., 1., 0. hatványa, ez tehát még ennyi 0 jegy.

Eredmény: $1000_{10}=13000_5$

Ha megcsinálnak a tanulók egy másik példát kifejezetten 2-es számrendszerbe váltással, mutatható a következő, egyszerűbb módszer decimálisból bináris rendszerbe váltásra:

Az átalakítás 2-vel való sorozatos osztással is elvégezhető. Az első osztásnál kapott **maradék** (0 vagy 1) adja a legkisebb helyiértékű bináris számjegyet (bitet). A következő osztás **maradéka** a következőt ... A sorozatos osztást addig kell folytatni, amíg a hányados 0 lesz.

Példa:

$50_{10}=?_2$

50 | 0

25 | 1

12 | 0

6 | 0

3 | 1

1 | 1

0 |

Nagyon fontos! Az eredményt lentről felfelé, tehát fordított sorrendben látjuk bitenként!

$50_{10}=110010_2$

6.4 4. óra: Számrendszerek megtapasztalása IKT eszközökkel

Gyakorlásként sok ilyen példát el lehet végeztetni. Számítógépteremben vagy interaktív táblánál érdekes feladatok készíthetők, próbálhatók ki a gyerekekkel.

(Egy ilyen flash alkalmazás fellelhető az interneten például itt: [16])

Vigyük be a tanulókat számítógépterembe! Készítsünk Moodle-tesztet a témakörben!

A Sulinet Digitális Tudásbázis [17] megfelelő anyagainak bemutatása után a tanulók csinálják meg a tesztet, amit a rendszer kiértékel és azonnal láthatják az eredményüket.

Rendkívül látványos prezentációk és videók is készültek a témában kifejezetten kilencedikes tanulók részére, vetítésre vagy megosztásra alkalmas teremben mindenképp érdemes rászánni 5-10 percet. (Például [18])

6.5 Dolgozat: Számrendszerek

A megoldásra fordítható idő: 40 perc. Segédeszköz nem használható.

1. Váltsd át a 496_{10} decimális számot bináris számrendszerbe, és ellenőrizd az eredményt!
2. Mekkora a legnagyobb, 5 biten ábrázolható természetes szám?
3. Írd fel az alábbi 16-os számrendszerbeli számot decimális alakban! 3F08=
4. Váltsd át az 511_{10} decimális számot 8-as számrendszerbe!
5. Hogyan lehet megállapítani egy 5-ös számrendszerbeli számról, hogy páratlan? Válaszodat röviden indokold!

Eredményes munkát kívánok!

A kitűzött feladatok megoldása és egy lehetséges értékelés:

1.

```
496|0
248|0
124|0
 62|0
 31|1
 15|1
  7|1
  3|1
  1|1
  0|
```

(2 pont ha jó, 1 pont ha csak elszámolta)

$496_{10}=111110000_2$. (1 pont)

Ellenőrzés: $1*2^8+1*2^7+1*2^6+1*2^5+1*2^4=496$ (1 pont)

2.

$2^5=32$, de mivel $0 \in \mathbb{N}$, a keresett szám a max=31. (2 pont, ha a gondolatmenet ugyanerre vezet)

3.

F=15. (1 pont ha kiderül a megoldásból)

$3F08_{16}=3*16^3+15*16^2+0*16^1+8*16^0=12288+3840+0+8=16136_{10}$. (1 pont)

4.

8^2=64; 8^3=512. (1 pont ha kiderül a megoldásból)

511:64=**7**

63

63:8=**7**

7

7:1=**7**

0

(1 pont a számolásért)

$511_{10}=777_8$ (1 pont)

5.

Mint bármely más számrendszerben; ha az utolsó (legkisebb helyiértéken álló) számjegy értéke páratlan, akkor maga a szám páratlan. (1 pont)

Összes pont: legfeljebb 12 pont.

Értékelés: osztályzással.

11-12 pont	jeles(5)
9-10 pont	jó(4)
7-8 pont	közepes(3)
5-6 pont	elégséges(2)
0-4 pont	elégtelen(1)

Irodalomjegyzék

[1] Neumann János, „First Draft of a Report on the EDVAC," Moore School of Electrical Engineering, University of Pennsylvania, 1945.

[2] Tózsa Judit, „Számelméleti versenyfeladatok a középiskolában," 2007. [Online]. Available: http://dea.unideb.hu/dea/bitstream/2437/2342/1/szakdolgozat.pdf. [Hozzáférés dátuma: 31 május 2013.].

[3] Dr Boros Zoltán, „Regular functions that preserve digital representation," Debreceni Egyetem, Matematikai Intézet, Analízis Tanszék, Debrecen, 1998.

[4] Farkas Gábor, Általánosított számrendszerek vizsgálata algebrai testbővítésekben, Budapest: ELTE, 2001.

[5] A. Kaivani et al., „Reversible Implementation of Densely-Packed-Decimal Converter to and from Binary-Coded-Decimal Format Using in IEEE-754R," in *IEEE*, Bhubaneswar, 2006.

[6] IBM, „AS/400," 2005. [Online]. Available: http://publib.boulder.ibm.com/cgi-bin/bookmgr/BOOKS/QB3AQ501/F.0. [Hozzáférés dátuma: 31. május 2013.].

[7] C. Babbage, Of the Analytical Engine, Larger Numbers Treated, London: Longman Green, 1864.

[8] A. K. -. Y. Ofman, Multiplication of Many-Digital Numbers by Automatic Computers, USSR: Academy of Sciences, 1962.

[9] Berechnungen und die Kompliziertheit von Beziehungen, Németország: Kybernetik, 1975.

[10] I. D. Allen, „The CARRY flag and OVERFLOW flag in binary arithmetic," [Online]. Available: http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt. [Hozzáférés dátuma: 31. május 2013.].

[11] Cserny István, „PICula projekt," 2009. [Online]. Available: http://esca.atomki.hu/PICula/ch4.html. [Hozzáférés dátuma: 31. május 2013.].

[12] Krebsz Anna, „Numerikus analízis," 30. május 2013. [Online]. Available: http://numanal.inf.elte.hu/~krebsz/jegyzet.html. [Hozzáférés dátuma: 31. május 2013.].

[13] Schoffhauzer Péter, „32 bites standard IEEE 754 lebegőpontos számábrázolás," [Online]. Available: http://numanal.inf.elte.hu/~krebsz/float.pdf. [Hozzáférés dátuma: 31. május 2013.].

[14] M. Wilson, „Multiplication bug in Excel 2007," [Online]. Available: http://www.mikewilson.cc/wp-content/uploads/2007/10/excel2007calculationerror.jpg. [Hozzáférés dátuma: 31. május 2013.].

[15] D. M. Oppenheimer, „Excel Blog - Calculation Issue Update," 25. szeptember 2007. [Online]. Available: http://blogs.office.com/b/microsoft-excel/archive/2007/09/25/calculation-issue-update.aspx. [Hozzáférés dátuma: 31. május 2013.].

[16] „Matematika," [Online]. Available: http://fitodi.byethost24.com/Matematika/. [Hozzáférés dátuma: 31. május 2013.].

[17] Educatio, „Sulinet Digitális Tudásbázis," Educatio, [Online]. Available: http://tudasbazis.sulinet.hu/hu/matematika/matematika/matematika-9-osztaly/a-szamrendszerekrol/szamrendszerek. [Hozzáférés dátuma: 31. május 2013.].

[18] Habóczki Károly, „YouTube," Google, 15. február 2013. [Online]. Available: http://www.youtube.com/playlist?list=PLjsWhXEUT7gg7MyeQJydQYubMb2C8wwfC. [Hozzáférés dátuma: 31. május 2013.].