A kettes számrendszer

A számítógép, mint már említettük egy két jelből álló jelkészlettel dolgozik. Ez a két jel a számítógép számára két különböző feszültségszint (az egyik általában a feszültség hiánya szokott lenni), azonban ezeket 0 és 1 jelekkel szoktuk szemléltetni, mivel a kettes számrendszerben is ezt a két számjegyet használjuk. Mint a következőkből kiderül, a kettes számrendszerben számolni is sokkal könnyebb, mint a hétköznapi életben használt tizes számrendszerben.

Az emberiség története során már sokféle számrendszer előfordult. A régi babiloniak például a hatvanas és a tizenkettes számrendszert is használták. Innen származik az időmérésben és a szögmérésben a mai napig használt 60-as váltószám és a nap 24 órája egyaránt.

Ugyanakkor a tucat szó, amely 12-t jelent, szintén a tizenkettes számrendszer használatára utal.

A számítástechnikában is használnak néhány számrendszert: kettes, nyolcas, tizes és tizenhatos számrendszert. A nyolcas és a tizenhatos számrendszer jelentőségére még visszatérünk a későbbiekben.

Átváltás kettes és tizes számrendszer között

Ahhoz, hogy tudjunk kettes számrendszerben számolni, előbb fel kell tudnunk írni a hétköznapi életben használt tizes számrendszerből a számokat kettes számrendszerbe. Ehhez először nézzük meg, hogy hogyan is néznek ki a kettes számrendszerben a számok.

A kettes számrendszerbeli számok felépítése

Minden helyiértékes számábrázolási rendszerben, így a kettes vagy a tizes számrendszerben is a számot valahány különböző számjegyből gazdálkodva írjuk fel. A rendelkezésre álló számjegyek száma valójában a számrendszer alapszáma.

Amikor a számjegyeket egymás mellé írjuk, akkor minden számjegy azt mondja meg, hogy az adott helyiértékből mennyi van a számban. Hogy mi a helyiérték? Nos az szintén az alapszámtól függ. A helyiértékek mindig jobbról balra nőnek. A legbaloldali számjegy mindig az egyesek számát adja meg, majd tőle balra haladva minden számjegyet egyel többször kell az alapszámmal megszorozni, mint a tőle jobbra levőt. Tehát a helyiértékek nem mások, mint az alapszám hatványai a nulladik hatványtól (ami 1 minden szám esetén) kezdve.

Ennek megfelelően tehát kettes számrendszerben a 100101 valójában a következő:

- 2^0=1-ből van egy darab
- 2^1=2-ből van nulla darab
- 2^2=4-ből van egy darab
- 2^3=8-ból és 2^4=16-ból van nulla darab
- és 2^5=32-ből van egy darab

Ha összeadjuk a nem nulla értékeket, akkor megkapjuk a szám értékét, ami éppen 37. Ezzel tulajdonképpen kaptunk egy átváltási módot kettes számrendszerből tizes számrendszerbe. Másrészt az is látható, hogy ha valaki tud kettővel szorozni, és így fel tudja írni a kettő hatványait, akkor az átváltás nem lehet gond, mivel a kettes számrendszerben egy adott helyiértéket vagy figyelembe kell venni (mert 1 az értéke), vagy nem kell figyelembe venni (mert 0 az értéke), így

csak bizonyos kettő-hatványok összeadására van szükség. Egyetlen más számrendszerre sem igaz ez.

Átváltás kettes számrendszerre

Most már tudjuk, hogy mit kell kapnunk: olyan felbontását a számnak, amelyben kettő hatványainak összegére bontottuk a számot. Hogy lehet ezt a legkönnyebben meghatározni?

Ez majdnem olyan, mint amikor valamilyen pénzösszeget kell meghatározott címletekben minél kevesebb érmével kifizetni. Most az egyes címletek megkaphatók kettővel osztás illetve szorzás révén.

A legkisebb címlet nyilván akkor kell, ha az egyel nagyobb címlettel nem osztható a szám, a következő címlet akkor, ha az egyel nagyobbal nem osztható a megmaradt összeg és így tovább.

Az átváltás módszere ennek megfelelően a következő: Osszuk el a számot kettővel, és írjuk fel a maradékot (ez lesz az egyes helyiérték). Az osztás eredményével ismételjük meg a műveletet mindaddig, amíg az nulla nem lesz. Így jobbról balra haladva mindig egy-egy helyiértéket kapunk meg a szám kettes számrendszerbeli alakjából.

A gyakorlatban az átváltást egy egyszerű, nagyon mechanikus módon lehet elvégezni:

> A módszer természetesen nem csak a kettes számrendszerbe történő átváltásra igaz, hanem bármely más számrendszer esetén. Viszont mindig annyiféle maradék van, amennyi a számrendszer alapszáma, és kettővel könnyebb osztani, mint a nagyobb számokkal.

1. Írjuk fel a számot egy függőleges vonal bal oldalára
2. Ha a szám páratan, írjunk mellé a vonal túloldalára egyet, alá a számnál egyel kisebb páros szám felét
3. Ha a szám páros, írjunk mellé a vonal túloldalára nullát, alá pedig a szám felét
4. Ismételjük meg az előző két lépést a legalsó baloldali számmal mindaddig, amíg az nullává nem válik (tovább nincs értelme, mert mindig nulla marad)
5. Ezután alulról felfelé haladva a jobboldali számjegyeket írjuk egymás mellé balról jobbra haladva. A kapott szám a kettes számrendszerbeli alakja a legelőször felírt számnak.

Az alábbiakban néhány példa látható a számolás elvégzésére:

37	1	23	1	52	0
18	0	11	1	26	0
9	1	5	1	13	1
4	0	2	0	6	0
2	0	1	1	3	1
1	1	0		1	1
0				0	

100101 10111 110100

Átváltás kettes számrendszerből tizes számrendszerbe

A kettes számrendszerbeli számok tizesbe történő átszámolására már láttunk egy módszert: Írjuk fel a számjegyek helyiértékét, és ahol egyes van, azokat a helyiértékeket adjuk össze. Ez egy működőképes módszer, de van egy másik módszer is, amely tulajdonképpen ugyanazon az elven működik, de néha jobban, néha kevésbé jól használható, ezért mindkét módszert érdemes ismerni.

Ennél a módszernél nincs szükség a helyiértékek ismeretére, csupán összeadni és kettővel szorozni kell tudni. A módszer a szám legmagasabb helyiértékű számjegyétől halad a legkisebb helyiérték felé a következőképpen:

1. Szorozzuk meg a legmagasabb helyiértéken levő számjegyet kettővel;
2. Adjuk hozzá a következő (egyel kisebb helyiértéken levő) számjegy értékét;
3. Amennyiben az éppen hozzáadott számjegy nem a legkisebb (egyes) helyiértékű számjegy volt, akkor az eredményt szorozzuk meg kettővel, és folytassuk a műveletet az előző ponttal, különben megkaptuk az eredményt.

Ez a módszer is működik más számrendszerre is, de kettő helyett mindig a számrendszer alapszámával kell a szorzásokat elvégezni, és a lehetséges számjegy is többféle lehet.

Hogy ez a módszer ugyanúgy helyes, mint a helyiértékek összeadása, az könnyen belátható: A folyamatos szorzások során az egyes helyiértékek minden lépésben egyel többször kerülnek szorzásra, mint az előző lépésben. Az abcde szám átszámolása során a következő történik:

((((2*a+b)*2+c)*2+d)*2+e

Ha felbontjuk a zárójeleket, akkor a következőt kapjuk:

((((2*a+b)*2+c)*2+d)*2+e = e + 2*d + 4*c + 8*b + 16*a = $e*2^0 + d*2^1 + c*2^2 + b*2^3 + a*2^4$

Vagyis a másik módszer szerinti átváltása a számnak...

A számok ábrázolása

A számítógép a számokat mindig meghatározott számú számjeggyel ábrázolja. Mint később látni fogjuk, ez a számolás során is előnyös lesz. Igen ám, de a számok, amelyeket ábrázolni fogunk nem mindig ugyanannyi jegyűek. A megoldás erre nagyon egyszerű: egészítsük ki balról a számot nullákkal, hogy a kívánt mennyiségű számjegyből álljon.

Bit, Byte, Szó

A számok (és a számok formájában tárolt bármely adat) ilyen adott számjegyen történő ábrázolása vezetett az *információ mértékegysége*inek is tekintett mértékegységek kialakulásához.

A **bit**ről már említettük, hogy az információ legkisebb egysége, amely két különböző információt tud egymástól megkülönböztetni, mivel két értéke lehet. Ez a kettes számrendszer egy számjegyének felel meg. Egy biten tehát egy egyjegyű számot tudunk megadni.

Két biten már 2*2 = 2^2= 4 számot, három biten 2^3=8 számot, négy biten 2^4=16 számot stb. lehet megadni. *n* biten tehát 2^n féle számot lehet tárolni. A lehetséges értékek pedig a csupa nullával megadott 0 értéktől a csupa egyel megadott 2^n-1 értékig terjedhetnek.

Eredetileg 6, majd később 7 bitet használtak együtt. Később a 8 bit vált elterjedté, és ez önálló nevet is kapott: **byte**. Az elnevezés onnan ered, hogy az angolban azt hogy a biteket nyolcasával csoportosítjuk a *by eight* kifejezéssel lehet megadni. Ez az amerikai angol szokásos beszédstílusában, amikor a gyors beszédben a szó felét elharapják, éppen úgy hangzik, mint ahogyan a *byte* szót kell ejteni.

Bár ma már 8 bitnél nagyobb méretű számokkal dolgoznak a számítógépek, az **információ alapegysége** továbbra is a **byte**, amely tehát nyolc bitet jelent. A rendelkezésre álló memória, vagy egyéb tárolóegységek kapacitását byte-ban szokás megadni.

Azonban a tényleges számméret megadása során szokás használni a **szó** fogalmát is, amely gépenként változhat, és

A jelenleg érvényes magyar helyesírási szabályok a *byte* szót és egy másik a számítástechnikában használt, egyébként rendes magyar jelentéssel rendelkező szót, „magyarította”, fonetikus írásmódot írva elő. Azonban a bájt illetve a másik szó esetén a fájl írásmód valójában nem megfelelő. Ez utóbbi nem magyar szó, így a fonetikus írásmódot semmi nem indokolja, magyar jelentése állomány, ami sokkal megfelelőbb helyette. A

az együtt kezelt, egyszerre feldolgozható bitek számát adja meg. Ha tehát például egy processzorra azt mondjuk, hogy 32 bites, az azt jelenti, hogy 32 bites szavakkal képes műveleteket végezni, vagyis legfeljebb 32 bit hosszúságú számok kezelésére képes.

A továbbiakban általában 8 bites, vagyis 1 byte-os számokkal fogunk foglalkozni, de minden egyéb szóméret esetén is igazak lesznek az állítások.

byte fonetikus írása ellen pedig az szól, hogy más idegen eredetű mértékegységet, mint például a *newton*t sem írjuk fonetikusan. Ha következetesek akarunk lenni, akkor írjuk valamennyi mértékegységünket egyformán! Ha az erő mértékegysége *newton*, akkor az információé *byte*, ha az információ mértékegysége *bájt*, akkor legyen az erőé *nyúton*!

A 16-os és a nyolcas számrendszer jelentősége

A számítógép belül tehát a számokat kettes számrendszerben tárolja. Azonban elég nehéz a kettes számrendszerbeli számokat olvasni, vagy leírni. Ezért a számítógéppel mélyebb szinten foglalkozók, akiknek a közvetlen számértékekkel is kapcsolatba kell kerülniük, jobban szerették volna, ha nem kettes számrendszerbeli számokkal kell dolgozniuk, de az átváltás sem jelent problémát.

A tizes számrendszer nem a legmegfelelőbb, mivel a tíz nem kettő-hatvány. Viszont van két olyan számrendszer, amelynek alapszáma kettő hatványa, és így a kettes számrendszerbeli számok könnyebb ábrázolását teszi lehetővé. Az egyik a régebben, a 6 bites szavak idején használt 8-as számrendszer, a másik a 16-os számrendszer. Az alábbi táblázat egyszerű átváltási útmutatást ad a kettes, a tizes, a nyolcas és a tizenhatos számrendszer között. A számrendszer megnevezésénél szerepel a számrendszer idegen eredetű, gyakran használt neve is.

Tízes (decimális) számrendszer	Kettes (bináris) számrendszer	nyolcas (oktális) számrendszer	Tizenhatos (hexadecimáls) számrendszer
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10

Az informatikában használt számrendszerek és kapcsolatuk

Ebből már látható, hogy mire használható a nyolcas illetve a tizenhatos számrendszer: Mivel alapszáma a kettő valamely hatványa, így a kettes számrendszerbeli számot könnyen, és rövidebben fel lehet írni ezekben a számrendszerekben. Három kettes számrendszerbeli számjegy helyett lehet egy nyolcas vagyis oktális számrendszerbeli számjegyet írni. Négy kettes számrendszerbeli számjegy helyett pedig egy tizenhatos számrendszerbeli számjegyet írhatunk. Így egy byte értékét pontosan két tizenhatos számrendszerbeli számjeggyel lehet felírni, ami időnként sokkal kényelmesebb lehet a nyolc darab bit felírása helyett.

> Egy byte, azaz nyolc bit esetén még talán nem tűnik túl nagy gondnak a nyolc számjegy leírása, de több byte hosszú értékek megadásánál már fárasztó a sok egyes és nulla leírása és olvasása egyaránt. Például a színek kódolására szokás három byte-ot alkalmazni, amelyeknek byte-onként is van külön jelentésük. Itt már valóban kényelmetlen lenne, ha huszonnégy bitet kellene leírni. Helyette rövidebb és olvashatóbb a hat tizenhatos számrendszerbeli, azaz hexadecimális számjegy.

A fenti táblázatból az is kiderül, hogy hogyan szokás megoldani azt a problémát, hogy összesen csak tíz számjegyünk van, mivel a hétköznapok során csak a tizes számrendszerben használt számjegyekre van szükség. A tizes értéktől a tizenötös értékig az angol ábécé első hat betűjét használjuk számjegyként. Az, hogy kis vagy nagy betűváltozatot használunk, az mindegy, de lehetőleg következetesen kell vagy az egyiket, vagy a másikat használni.

Példaként az átváltásra:

kettesben:	00110101	11010001	11111111
tizenhatosban	35	D1	FF
nyolcasban	065	321	377

Természetesen a példáknál a nyolcas számrendszerben a lemagasabb helyiértékű jegy 3-nál nem lehet nagyobb, mivel csak két bit jut erre a jegyre...

Számolás kettes számrendszerben

Most már tudjuk a számokat felírni kettes számrendszerben, és értelmezni is tudjuk a kettes számrendszerbeli számokat. A következőkben megtanuljuk, hogyan kell ezekkel a számokkal összeadni, kivonni valamint szorozni. Az osztás még kettes számrendszerben is bonyolult műveletnek számít, amelyet a számítógép is több lépésben végez el, így azzal nem foglalkozunk. Azonban mint látni fogjuk, a szorzás kettes számrendszerben sokkal egyszerűbb, mint bármely másik számrendszerben.

A számolás során mindig olyan számokkal fogunk foglalkozni, amelyek 1 byte-on tárolhatók, és ezeket pontosan egy byte-on, vagyis 8 biten fogjuk ábrázolni. A szükséges számú bit elérése mindig a szám balról nullákkal történő kiegészítésével történik.

Elöljáróban még annyit, hogy a kettes számrendszerbeli számolás könnyűségét az alkalmazandó szabályok alacsony száma okozza, ami abból adódik, hogy kevés számjeggyel kell dolgozni, így a számolás során kevés lehetőséget kell figyelembe venni. Először mindig a szabályokat adjuk meg egy művelettábla segítségével, és azután megnézzük, hogyan is kell a gyakorlatban a szabályokat alkalmazni.

Összeadás

Az összeadás művelettáblája a következő:

+	**0**	**1**
0	0	1
1	1	0^*

A csillag jelentése itt az, hogy a művelet eredménye kihat a következő helyiértékre is. Ezt a hatást nevezzük **átvitel**nek. Az átvitel azt mondja meg, hogy az összeadás eredménye nem fér el az adott helyiértéken, így a következő helyiértéken az összeadást úgy kell végrehajtani, hogy az eredményhez még egy egyest hozzá kell adni.

Ebből viszont az is következik, hogy nem ilyen egyszerűek a szabályok, hiszen azt is meg kell vizsgálni, amikor három számjegyet adunk össze. A lehetséges esetek tehát a következők:

1. Minden jegy 0: az eredmény 0, és nincs átvitel (0+0=0).
2. Egy egyes jegy van: az eredmény 1, és nincs átvitel (0+1=1+0=1).
3. Két egyes van: az eredmény 0, és van átvitel (1+1=10).
4. Három egyes van: az eredmény 1, és van átvitel (1+1+1=11).

Ezt persze lehetne a fenti művelettáblával is szemléltetni, csak kell egy külön művelettábla arra az esetre is, amikor egy átvitelt is figyelembe kell venni. Jelöljük ezt a műveletet $+^*$-gal, és írjuk fel a táblába az esetlegesen keletkező átvitelt is:

+	**0**	**1**
0	00	01
1	01	10

$+^*$	**0**	**1**
0	01	10
1	10	11

Ezek után nézzük, hogy a gyakorlatban hogyan kell két kettes számrendszerbeli szám összeadását elvégezni! Tulajdonképpen ugyanúgy, ahogyan tizes számrendszerben is csináljuk, ha csak papír és ceruza áll rendelkezésünkre (géppel nem is lehetne tizes számrendszerbeli számokat összeadni, mert a számológép is kettes számrendszerben számol):

Írjuk egymás alá a két számot, majd a legkisebb helyiértéktől haladva a fenti szabályok szerint minden helyiértéken számoljuk ki az eredményt, és írjuk az összeadandó számok alá. Amikor az utolsó, legnagyobb helyiértéken is elvégeztük az összeadást, az esetleges átvitelt is írjuk még a szám elé. Az így kapott szám lesz az összeg.

Íme néhány elvégzett számítás:

```
  00101110      00100001      00110111
+ 00001111    + 00001111    + 00101111
  00111101      00110000      01100110
```

A pöttyök minden esetben az átvitelre utalnak.

A túlcsordulás fogalma

A számítógép mindig meghatározott számú biten dolgozik. Példáinknál ez nyolc bit, vagyis egy byte. Azonban az összeadás eredménye a legmagasabb helyiértéken esetleg keletkező átvitel miatt lehet egyel több, példánkban 9 bit hosszúságú is. Ez a tárolásnál problémát jelenthet.

Ezt az utolsó átvitelként létrejövő többlet bitet nevezzük **túlcsordulás**nak. Látszólag ez problémát okoz, mivel nem tudunk vele mit kezdeni, azonban lehet hasznos is.

Az egyik eset, amikor ezt felhasználják, a több szót elfoglaló, nagyobb számok összeadása. Mivel az összeadás után ugyan ez nem tárolódik el az eredménnyel, azonban rendelkezésre áll az információ, hogy túlcsordulás történt, így fel lehet használni átvitelként a következő szóban található számrészek összeadásához. Így tulajdonképpen egy olyan összeadást lehet

megvalósítani, ahol a számjegyek egy-egy szóban vannak tárolva, a számrendszer alapszáma pedig az egy szóban tárolható értékek száma.

Ennél sokkal nagyobb jelentősége lesz azonban a mi számunkra a túlcsordulásnak a negatív számok ábrázolásánál és a negatív számokkal való számolásnál...

Gyakorló feladatok:

Add össze a következő kettes számrendszerbeli számokat:

- 1011+1001=
- 10011+10110=
- 1111+1010=

Szorzás

A szorzás legalább olyan egyszerű kettes számrendszerben, vagy talán még egyszerűbb is, mint az összeadás. Összesen csak két dolog kell hozzá: a szorzótábla, és annak ismerete, hogy minden számrendszerben ha egy számot a számrendszer alapszámával szorzunk, az azt jelenti, hogy egy nullát kell a szám végére írni, vagyis minden helyiérték egyel nő.

A szorzótábla kettes számrendszerben így néz ki:

*	**0**	**1**
0	0	0
1	0	1

Vagyis, ha két egyest szorzunk össze, akkor az eredmény egy lesz, különben pedig 0. Szerencsére most nincs semmiféle átvitel.

Ezzel a szorzótáblával egy kettes számrendszerbeli számot egy egy bites kettes számrendszerbeli számmal már össze tudunk szorozni: ha ez utóbbi egy, akkor az eredmény a másik szám, ha nulla, akkor az eredmény nulla. Azonban mi két több bites számot akarunk összeszorozni.

Ehhez kell az az információ, hogy ha kettővel szorzunk egy számot, az nem más, mint a szám egy helyiértékkel feljebb tolása, és az egyes helyiértékre nulla írása. Ennek ismeretében nem kell mást tenni, mint a két összeszorzandó szám egyikét felírni kettő hatványok összegeként. Ez bonyolultan hangzik, pedig nem más, mint amit az átváltásnál is alkalmaztunk: minden helyiérték kétszerese az előzőnek.

Magyarul arról van szó, hogy szorozzuk meg az egyik számot mindig a másik szám helyiértékével, majd adjuk össze a kapott eredményeket de úgy, hogy a helyiértékeire bontott számban a helyiértékkel még meg kell majd az eredményt az összeadás előtt szorozni.

Még érthetőbben ez annyit jelent, hogy ha a szorzó számjegyein a legkisebb helyiértéktől haladunk, akkor a részszorzatot minden esetben egyel toljuk balra az előzőhöz képest (kettővel szorzás), és így adjuk össze a részszorzatokat.

Ez tulajdonképpen ismét a már tizes számrendszerből is ismert szorzás, csak azzal a különbséggel, hogy most sokkal egyszerűbb szorzótáblát kell használnunk: ha a szorzó helyiértéke egy, akkor a részszorzatnak magát a szorzandót kell leírni, ha pedig nulla, akkor a részszorzat nulla.

A részszorzatok összeadását kettesével végezve pedig az előzőekben ismertetett módszert alkalmazhatjuk.

Egy dologra azonban a részszorzatok összeadása kapcsán még figyelni kell: Mivel ha egy *n* bites számot megszorzunk egy *m* bites számmal, akkor összesen *m* darab *n* bites szorzat fog

keletkezni, amelyek mind egy-egy helyiértékkel eltolva adandók össze. Ez azt jelenti, hogy az eredmény *m+n* hosszú lesz. Vagyis két 1 byte-os szám szorzataként egy 2 byte-os számot kapunk. Általában is igaz, hogy két egyforma hosszúságú bináris szám szorzata kétszer annyi bitet igényel, aminek megfelelően a számítógép a szorzás eredményét mindig a szorzandóknál kétszer hosszabb helyen tárolja.

Szorozd össze a fent leírtakat használva a következő számokat:

- 10011 * 10100
- 10011101*10001
- 10101011*10111
- 101010*10110

Negatív számok ábrázolása

A negatív számok ábrázolására olyan megoldásra van szükség, amelyben egyértelműen megállapítható a számról, hogy negatív vagy pozitív, és amely ábrázolás mellett a matematikai műveletek is könnyen elvégezhetők maradnak.

Ez utóbbi feltétel azt jelenti, hogy olyan ábrázolási módot kell alkalmaznunk, amely esetén az összeadásnál nem kell további szabályokat megadni ha valamelyik vagy mindkét szám negatív. Vagy ha mégis kénytelenek vagyunk új szabályt is bevezetni ehhez, minél egyszerűbben alkalmazható legyen.

Amennyiben találnánk egy olyan ábrázolási módot, amelyben az összeadás szabályait egy negatív számra és annak abszolút értékére alkalmazva az összeadás eredménye nulla lenne, akkor ez egy olyan ábrázolási mód lenne, ahol nincs szükség semmilyen új szabály bevezetésére.

A jó hír, hogy van ilyen számábrázolási mód, mégpedig a *kettes komplemens* alak használata. Lássuk először, hogy ez hogyan is néz ki, majd ezután megvizsgáljuk, hogy ez tényleg megfelel-e a céljainknak!

A kettes komplemens alak

A pozitív számok ábrázolása ugyanaz, mint eddig volt, így azzal nem kell foglalkozni. A negatív számoknál pedig a következő módszert kell alkalmazni:

1. Írjuk fel kettes számrendszerben, az ábrázolásban használt biteknek megfelelő számú számjeggyel a szám abszolút értékét.
2. Fordítsuk meg a biteket: a 0 helyett írjunk 1-et, az 1 helyett pedig 0-át.
3. Adjunk hozzá a kapott számhoz 1-et.

Az így kapott számot nevezzük a szám **kettes komplemens ábrázolásá**nak.

Megjegyzendő, hogy azért hívják kettes komplemens alaknak, mert az átalakítás során első lépésként (a 2. pont) végzett bit-fordítás (idegen szóval *bitenkénti negálás*) tulajdonképpen a szám *egyes komplemense*.

A kettes komplemens tulajdonságai

Most már bármely negatív számot tudunk kettes komplemensben ábrázolni. A kérdés csupán az, hogy ez az ábrázolási mód megfelel-e a kívánalmainknak.

Ehhez először nézzük meg, mit kapunk, ha a -1 és abszolút értéke, azaz az 1 összegét a tanult módon kiszámoljuk. Először a -1 kettes komplemens alakját kell megkapni. Eddigi példáinkhoz

hasonlóan most is nyolc bites számokat használunk. Ekkor az 1 alakja: 00000001. Ennek egyes komplemense, vagyis a bitek megfordítása: 11111110. Ehhez egyet hozzáadva 11111111-t kapunk, vagyis ez a -1 kettes komplemensbeli alakja.

Adjuk most össze ezt és a 00000001-t vagyis az 1-et! Rövid számolás után látható, hogy az eredmény 100000000. Azonban a kilencedik bitet nem ábrázoljuk, vagyis az összeadás eredménye 00000000, ami a nulla alakja. Ugyanezt alkalmazva más számokra is, látható, hogy mindig ezt az eredményt kapjuk, vagyis ha figyelembe vesszük, hogy a túlcsorult egyest nem fogjuk tárolni, akkor valóban nullát kapunk minden szám és ellentettje összeadásánál.

Figyeljük most meg azt is, hogy hogyan néznek ki a kettes komplemens ábrázolásban a negatív és a pozitív számok! A negatív számoknál a legmagasabb helyiértéknek megfelelő bit mindig 1 lesz, míg a pozitív számok esetén és a nulla esetén ez a bit mindig 0. Éppen ezért ezt a bitet **előjelbit**nek nevezzük. Ez egyértelműen megadja, hogy a számunk milyen előjelű. Természetesen a nullát pozitív számnak mutatja ez a bit.

Ebből már az is következik, hogy a számok előjeles ábrázolásánál majdnem pontosan 50-50% eloszlással ábrázolhatók negatív és pozitív számok, hiszen egy bit értéke éppen a felét adja az ábrázolható számoknak. Igen ám, de a nulla a pozitív számok közül egynek a helyét veszi el. Akkor ezek szerint egyel több negatív számot tudunk ábrázolni, mint pozitívat. Vajon melyik ez a szám?

Mielőtt ezt megkeresnénk, még egy tulajdonságát nézzük meg a kettes komplemens alaknak: A -1 kettes komplemens alakjából hogyan tudnánk visszakapni az abszolút értékét? A megoldás nagyon egyszerű: képezzük ennek is a kettes komplemens alakját: 11111111 egyes komplemens alakja: 00000000, ehhez egyet hozzáadva pedig 00000001-et kapunk.

Ugyanez igaz minden negatív számra is, vagyis egy negatív számból az abszolút értékét ugyanazzal a módszerrel kapjuk, amivel az abszolút értékből a kettes komplemens alakját kaptuk.

Most pedig térjünk vissza az előző kérdésre: melyik az a szám, amit negatív számként ábrázolni tudunk, de pozitívként már nem? Válaszként egy feladat: számoljuk ki az 10000000 negatív szám abszolút értékét!

Ha ennek a számnak vesszük a kettes komplemens alakját, akkor 10000000-t kapunk. Ez azonban most már pozitív szám, amit átszámolva 128-at kapunk. Mivel a legnagyobb nyolc biten előjelesen ábrázolható pozitív szám (vagyis aminek az előjelbite 0) a 01111111, ami 127, ebből látható, hogy az ábrázolható számok 127 és -128 közé esnek. Ez *n* bit esetén is igaz. Ekkor az előjelesen ábrázolható számok -2^{n-1} és $2^{n-1}-1$ közé esnek ellentétben az előjelnélküli esettel, amikor ez a tartomány 0 és 2^n-1 között volt.

Számolás előjeles számokkal

Az előjeles számokkal az összeadás és a kivonás nagyon egyszerű: összeadásnál egyszerűen összeadjuk őket, a túlcsordulást figyelmen kívül hagyva. Kivonásnál előbb a kivonandó ellentetjét kell venni, majd ezután összeadást végzünk.

Egy dologra azért érdemes figyelni: ha az eredmény az ábrázolási tartományon kívül esik, akkor hibás eredményt fogunk kapni. Például előfordulhat, hogy két pozitív szám összeadásának eredménye negatív szám lesz, vagy két negatív szám összege pozitív lesz. Ilyenkor az előjelbit nyilván a két szám összeadásakor keletkezett túlcsorulás miatt lesz hamis.

Szorzásnál egyszerűbb a helyzet, mivel nem kell negatív számokkal foglalkozni. Egyszerűen összeszorozzuk a két szám abszolút értékét, majd a matematika szabályai alapján megállapítjuk az előjelet, és ha negatív az eredmény, akkor képezzük a kettes komplemensét.

Összefoglaló feladatok:

1. Végezd el kettes számrendszerben a következő műveleteket:

- 25+31
- 35-12
- 18-51
- 21*15
- -4*30
- 50-110

2. Mennyi a legnagyobb és a legkisebb ábrázolható szám 16 biten, előjeles számábrázolásnál illetve előjel nélküli számábrázolás esetén?
3. Két pozitív szám összeadása után -15 lett az eredmény. Mi ennek az oka?

Következő: A számítógép felépítése
Vissza: Bevezetés az informatikába akkor az