PRG1: A számítógépes adattárolás alapjai

1. lecke: A számítógép és a kettes számrendszer

1. A lecke tartalma

- Helyiértékes számrendszerek
- A kettes számrendszer
- Átváltás tízes és kettes számrendszer között
- Egész számok ábrázolása a számítógépben

2. Számrendszerek

Amióta az ember mennyiségeket akar nyilvántartani, azóta van szükség a számokra. Azonban nem mindegy, ezeket milyen szabályok szerint írjuk le. A történelem során többféle számrendszer alakult ki. Ezek között az egyik közös, hogy majdnem mind helyiértékes számrendszerek.

Matematikából már biztosan ismerős a helyiérték és a helyiértékes számrendszer fogalma, így itt elég egy rövid összefoglalás, mielőtt a minket – pontosabban a számítógépet – közelebbről érdeklő kettes számrendszerre rátérnénk:

- Minden helyiértékes számnak van egy alapszáma.
- A számrendszerben annyi különböző jelre (számjegyre) van szükség az értékek megkülönböztetésére, amennyi az alapszám. Ha mondjuk *b*-vel jelöljük az alapszámot, akkor elmondhatjuk a legtöbb számrendszerről, hogy a számjegyek a 0, 1, … *b*-1 értékeket jelölik.
- A számrendszer attól helyiértékes, hogy jobbról balra haladva a leírt számjegyek értéke éppen az alapszám soron következő hatványával nő, így a helyiérték rendre b^0, b^1, b^2, … lesz.
- A leírt szám értékének kiolvasásához a helyiértéket és az adott helyiértéken levő számjegyet kell összeszorozni, majd az így kapott értékeket összeadni.
- A fentiekből következik, hogy minden számrendszer esetén az alapszámot 10 alakban kell írni, hiszen az alapszám első hatványából van egy.

A fentiek alapján gondoljuk végig mielőtt tovább lépnénk, hogy a történelem folyamán milyen számrendszereket használt az emberiség, amelyek a fentieknek megfelelnek.

A római számok vajon helyiértékes számrendszert alkotnak? Gondoljunk arra, hogy a használt jelek az 1, 5, 10, 50 stb. értékekhez tartoznak, a nullát nem lehet ábrázolni (a rómaiak még nem ismerték a nulla fogalmát), viszont az abakusszal éppen a római számrendszerben lehet úgy számolni, ahogy helyiértékes számokkal szoktunk...

3. A kettes számrendszer

Bennünket a továbbiakban a legegyszerűbb helyiértékes számrendszer fog érdekelni. Ennek alapszáma a kettő, tehát összesen két számjegyet használhatunk benne. A későbbiekben majd ennek is egy egyszerűsített, de pontatlan változatát ismerjük meg, amelyet a számítógép az adatok ábrázolására használ. Ebben ténylegesen csak a 0 és 1 jelek, illetve ezek fizikai reprezentációja létezhet, előjel például már nem. De előtte nézzük meg magát a kettes számrendszer!

A fentiekből már tudjuk, hogy egy kettes számrendszerben felírt szám helyiértékei jobbról balra haladva a kettő hatványai:

0. $2^0 = 1$
1. $2^1 = 2$
2. $2^2 = 4$
3. $2^3 = 8$
4. $2^4 = 16$
5. $2^5 = 32$
6. $2^6 = 64$
7. $2^7 = 128$
8. $2^8 = 256$
9. $2^9 = 512$
10. $2^{10} = 1024$

Tehát például a 1011 értéke 1 + 2 + 0 + 8 = 11, a 10101 értéke pedig 1 + 0 + 4 + 0 + 16 = 21.

A két fenti példában látszólag kimaradt az átszámítás egy lépése, a helyiértékkel való beszorzás. De ez csak látszólagos. A kettes számrendszer egyik nagy előnye a többi számrendszerrel szemben ugyanis éppen az, hogy a számjegy vagy nulla és akkor az összeadásban nem kell figyelembe venni, hiszen a résszorzat is nulla, vagy egy, amely esetben a résszorzat pedig megegyezik a helyiértékkel. Így valójában elég a szám azon helyiértékeit összeadni, ahol a számjegy 1. Ez látható a fenti példákban.

Ezzel már meg is adtam a kettes számrendszerből az általunk a hétköznapi életben használt tízes számrendszerbe történő átváltás módját:

Átváltás kettesből tízes számrendszerbe: Jobbról balra haladva írd fel azoknak a helyiértékeknek, azaz a kettő hatványainak (nullától indulva) az értékeit, ahol az 1 számjegy szerepel, a kapott értékeket add össze!

A fenti eljárásban a „jobbról balra haladva” elvileg felesleges, hiszen haladhatunk balról jobbra is, ha tudjuk, melyik helyiérték mennyit ér. Azonban sokkal egyszerűbb jobbról indulni, hiszen így folyamatosan nő a kettő hatványok kitevője.

Másik lehetséges módszert az ellenkező irányú átváltás után nézzük meg.

3.1. Átváltás tízesből kettes számrendszerbe

Ha egy tízes számrendszerbeli szám kettes számrendszerbeli alakját szeretnénk megkapni, akkor nyilván meg kell keresnünk azoknak a kettő hatványoknak az összegét, amelyek eredményeként a

számunk adódik. Ez elég próbálgatós módszernek hangzik ahhoz, hogy keressünk egyszerűbb módszert helyette. Íme:

1. Írjuk fel a számot egy függőleges vonal baloldalára.
2. Osszuk el a számot kettővel, az eredményt írjuk a szám alá, a maradékot pedig a szám mellé, a vonal jobboldalára.
3. Az előző pontot ismételjük meg mindaddig, amíg a vonal baloldalán meg nem kapjuk a nulla értéket eredményül.

Néhány megjegyzést érdemes a fenti eljáráshoz hozzáfűzni: Először is a fenti eljárást más számrendszerekre történő átváltásnál is ugyanígy kell alkalmazni, ezért talán matematika óráról már ismerős is. Mindig a cél-számrendszer alapszámával kell osztani.

Második megjegyzés, hogy a fenti eljárást a programozók – és majd mi is, amikor már programozni fogunk – úgy hívják, hogy algoritmus. Az algoritmus szót ugyanannak az embernek köszönhetjük, akinek azt is, hogy már nem római számokkal kell szenvednünk, amikor számolunk, hanem az Indiában kifejlődött számokat használjuk. Ezeket a számokat mi arab számokként ismerjük, mert hozzánk arab közvetítéssel érkezett el ennek a bagdadi tudósnak a műve. Úgy hívták: IBN MUSA AL-KWARIZMI. A számítástechnika történetének kapcsán még lesz szó róla és művéről bővebben.

Harmadik megjegyzés: Az osztás minden esetben egész osztást jelent. Kettes számrendszer esetében ezt az osztást lehet úgy is értelmezni, hogy a szám mellé írjuk egy biten kódolva, hogy igaz-e, hogy a szám páratlan, alá pedig a számnak, vagy ha páratlan a nála egyel kisebb páros számnak a felét írjuk.

A fenti algoritmus végrehajtása után a vonal jobb oldalán levő 0,1 jelekből álló sorozat nem más, mint a szám kettes számrendszerbeli – más számrendszerekben ez is analóg módon történik – alakja. A legalsó számjegy (amely mindig 1) a legbaloldalon levő, a legfelső pedig a jobboldalon levő, tehát legkisebb helyiértékű számjegy lesz.

3.2. Kettesből tízesbe átváltás másik módon

A fenti átváltások elvileg matematikából is szerepelnek általános alapszám esetén. A következő módszer azonban nem feltétlenül szerepel, pedig a tízesből történő átszámolás egyszerű analógiája:

1. Írd fel lentről felfelé haladva a kettes számrendszerbeli szám számjegyeit balról jobbra haladva a függőleges vonal jobb oldalára!
2. A vonal baloldalán, az első üresen maradt sorba írj egy nullát!
3. Ezek után mindaddig, amíg fel nem érsz a számsor tetejére ismételd meg a következőket:
    1. A baloldali számhoz add hozzá a felette levő sorban a jobboldalon levő számjegyet (0 vagy 1).
    2. A kapott számot szorozd meg kettővel.
    3. Az így kapott számot írd az üres helyre a vonal baloldalára.

Amikor eljutsz a legfelső sor baloldalára, az odaírt szám lesz a tízes számrendszerbeli megfelelője annak a számnak, aminek a balról jobbra levő számjegyei a vonal jobboldalán vannak lentről kezdve felírva. Bizonyíték? Végezd el az előző pontban leírt algoritmust! Ez pont annak az ellenkező irányú végrehajtása.

Igazából amikor – például szorzásnál – a számítógépnek át kell váltania a számot tízes számrendszerbe, akkor ezt a módszert használja.

3.3. Egész számok ábrázolása számítógépen

A fentiekben csupán egész számok ábrázolását néztük meg kettes számrendszerben. Azonban ha az egyes helyiértéktől tovább haladunk jobbra, akkor ott a kettő negatív kitevős hatványaival is lehetne folytatni. Itt azonban felmerül a kérdés, hogy meddig folytassuk.

Egyelőre semeddig! Később azonban majd megnézzük azt is, hogy tört számokat hogyan lehet ábrázolni. Jelenleg azonban elégedjünk meg egész számokkal!

A számítógépben tárolható bitek száma ugyanis véges. Emiatt már a kezdetekben – igaz többféleképpen – meghatározták, hogy egy-egy elemi adat tárolására hány bitet használjon a számítógép. Mivel a ma használatos számítógépek angol nyelvterületről származik, így talán nem meglepő, hogy az angol nyelvben használt jelek tárolására tervezték őket eredetileg. Így alakult ki az a gyakorlat, hogy a 128 lehetőséget adó 7 bites tárolási egységeket kezdték el használni, melléjük téve az adat helyességének ellenőrzésére még egy 8. bitet is. Később ez a nyolcadik bit is hasznos bitté vált, így alakult ki a 8 bitből álló egység, a byte. Aztán később ennek többszöröseire is hamar szükség vált, hiszen egy byte összesen 256 féle adatot tud tárolni.

A tényleges legkisebb tárolási egység tehát valahány byte – jellemzően ez is kettő hatvány. Ezt nevezzük *szó*nak, ami gépenként változik. A szó mérete meghatározza azt is, hogy mekkora méretű számokat lehet tárolni.

Alapesetben nem történik más, mint a szám kettes számrendszerbeli alakját egyszerűen eltároljuk a szóban. Persze a legtöbbször nem annyi számjegyből álló a szám, mint amennyi bit rendelkezésre áll. Ezért alapszabály, hogy a szót alkotó egy vagy több byte sorba rakása után a jobboldalon levő bit a szám jobboldalán levő számjegyet kapja, azaz a legkisebb bit tárolja az egyes helyiértéket. A szám tárolásához már nem szükséges bitek értéke pedig nulla lesz. Így az átszámítás tízes számrendszerbe ugyanúgy történik, mint normálisan.

Például a korábban már látott tizenegy értékű 1011 számot egy byte méretű szó esetén így tárolja a számítógép: 00001011, a két byte méretű szóban 00000000 00001011 formában fog szerepelni.

A másik eset, amikor több számjegyű a szám, mint amennyi bitből áll a szó, már nem ilyen egyszerű. Ekkor ugyanis a számnak csak a legkisebb n számjegye lesz tárolható (n a szóban levő bitek száma). A szám többi számhegye már nem fér el a szóban. Ez alaphelyzetben egyszerűen elveszik. Ezt az esetet hívják *túlcsordulás*nak.

Túlcsordulás: Számábrázolási hiba, amikor az ábrázolandó szám tárolására nem áll elegendő bit rendelkezésre és a szám legmagasabb helyiértékű számjegyei tárolásra használható bit hiányában elvesznek. Amikor túlcsordulás keletkezik, akkor az eltárolt szám helyett valójában egy másik szám tárolódik el, amely természetesen számítási hibát fog eredményezni.

Bizonyos esetekben még a túlcsordulást okozó méretű számok is tárolhatóak. Ehhez viszont az ilyen számokkal dolgozó programot kell erre megfelelően felkészíteni. Túlcsorduláskor ugyanis a CPU ezt jelezni tudja, így a program képes rá, hogy a számot több részre bontva, több szóban helyezze el. Ekkor azonban a CPU önmagától nem tud ezekkel a számokkal helyesen műveleteket végezni, azt is a programban kell megoldani.

3.4. Számábrázolási tartomány

Ahhoz, hogy a túlcsordulásból eredő problémákat elkerülhessük, tudnunk kell, mekkora méretű szóban mekkora számok tárolhatóak. Annak, aki ismeri a kettő hatványokat, ez nem okozhat túlzott problémát.

Nemnegatív egész számokról van szó jelenleg, így a legkisebb ábrázolható szám nyilvánvalóan a 0 lesz, amit a számítógép úgy fog ábrázolni, hogy minden bit értéke 0 lesz.

Ha végig gondoljuk, milyen formája van a számoknak kettes számrendszerben,[1] akkor hamar rájövünk, hogy a legnagyobb számot akkor kapjuk meg, ha minden bit értéke 1. Mennyi ez a szám, ezt kell meghatározni. Nyilván egyel kisebb annál a számnál, amelyben ezeken a helyiértékeken 0 lenne, a következő helyiértéken pedig 1. Ez az első olyan szám, amit a túlcsordulás miatt már nem lehet ábrázolni (történetesen nullának ábrázolná a gép).

Ha felidézzük, mi a helyiértéke értéke, akkor láthatjuk, hogy az első olyan szám, amely a túlcsordulás miatt nem ábrázolható n bit esetén 2^n. Mivel ez egyel nagyobb a csupa egyessel ábrázolható legnagyobb számnál, a legnagyobb ábrázolható szám 2^n-1 lesz, ha a n bit áll rendelkezésre.

Tehát a számítógép legalapvetőbb ábrázolási módja esetén, amikor nemnegatív egész számokat ábrázolunk, a számábrázolási határok így néznek ki:

1. 1 byte esetén: $0 \leq$ szám ≤ 127 ($2^8 - 1$)
2. 2 byte esetén: $0 \leq$ szám ≤ 255 ($2^{16} - 1$)
3. n bites szó esetén: $0 \leq$ szám $\leq 2^n - 1$.

A negatív számok, illetve a tört számok ábrázolására majd később térünk vissza.

1 *Akinek ez nem megy, majd a következő lecke után újra megpróbálhatja, amikor már tud kettes számrendszerben összeadni...*