Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Irányítástechnika és Informatika Tanszék

Dr. Pilászy György

Hardveralapok
01-02. előadás

(Számok ábrázolása)

Lektorálta: Dr. Horváth Tamás

Korszerű számítógépekben alkalmazott számábrázolási módok

Az alábbi fejezetben röviden összefoglaljuk a legfontosabb tudnivalókat a tízes, kettes és tizenhatos számrendszerekről, előjel kezelésről és a tört számok ábrázolásáról.

Tízes számrendszer

A mindennapi életünkben gyakran találkozunk számokkal. A legtöbb általánosan használt számot úgynevezett tízes számrendszerben ábrázoljuk.

Vegyük például az 5678 decimális számot (öt**ezer**-hat**száz**-hetven-nyolc). Ha a szám értékére vagyunk kíváncsiak, akkor a következő formában határozhatjuk meg:

$$5678 = 5 \cdot 10^3 + 6 \cdot 10^2 + 7 \cdot 10^1 + 8 \cdot 10^0$$

A fenti formában az egyes számjegyeket szoroztuk a számrendszer alapjának (10) megfelelő kitevőjű hatványaival ezeket nevezzük helyi értékeknek. Az egyes számjegyek 0..9 közötti értékeket vehetnek fel. Jelöljük a számrendszer alapját r-el, az egyes számjegyeket h_i-vel, ahol i egy futó indexet jelöl, mely a fenti példánkban 0..3 közötti értékeket vehet fel. Ezen jelölésekkel a következő képen írhatjuk fel tömörebb alakban egy tetszőleges N egész szám értékének kiszámítását:

$$N = \sum_{i=0}^{n} h_i \cdot r^i$$

$$h_i \in \{0,1,2,3,4,5,6,7,8,9\}$$

$$5678 = \sum_{i=0}^{3} h_i \cdot r^i = 5 \cdot 10^3 + 6 \cdot 10^2 + 7 \cdot 10^1 + 8 \cdot 10^0 = 5000 + 600 + 70 + 8$$

Vagyis $h_3=5$, $h_2=6$, $h_1=7$, $h_0=8$; $r^3=10^3=1000$, $r^2=10^2=100$, $r^1=10^1=10$, $r^0=10^0=1$

Kettes számrendszer

A számítástechnikában elterjedten használjuk a kétállapotú jeleket, melyek leírására a kettes számrendszert alkalmazzák. A fenti jelöléseket használva a kettes számrendszer esetén r = 2, és az egyes h_i számjegyek kizárólag 0 vagy 1 értéket vehetnek fel.

$$h_i \in \{0,1\}$$

Átváltás 10→2

Egy tízes számrendszerben ábrázolt számot egyszerűen átalakíthatunk kettes számrendszerbe, ha elkezdjük kettővel osztani majd minden osztásnál külön vesszük az osztási maradékokat. Alakítsuk át a fenti példában szereplő 5678 decimális számot kettes számrendszerbeli számmá.

Osztás	**Maradék**	**Számjegyek (h_i)**	**Helyi értékek (r^i)**
5678:2 = 2839	**0**	= h_0	1
2839:2 = 1419	**1**	= h_1	2
1419:2 = 709	**1**	= h_2	4
709:2 = 354	**1**	= h_3	8
354:2 = 177	**0**	= h_4	16
177:2 = 88	**1**	= h_5	32
88:2 = 44	**0**	= h_6	64
44:2 = 22	**0**	= h_7	128
22:2 = 11	**0**	= h_8	256
11:2 = 5	**1**	= h_9	512
5:2 = 2	**1**	= h_{10}	1024
2:2 = 1	**0**	= h_{11}	2048
1:2 = 0	**1**	= h_{12}	4096

Vagyis $(5678)_{10} = (1011000101110)_2$

Megfigyelhetjük, hogy amíg tízes számrendszerben négy számjegy elegendő volt az ábrázoláshoz, addig kettes számrendszerben összesen tizenhárom számjegyre volt szükségük.

Átváltás 2→10

Kettes számrendszerben ábrázolt számok decimális megfelelőjét többféleképpen is kiszámíthatjuk.

Kevés számjegy esetén talán a legegyszerűbb, ha egyszerűen összeadjuk az 1 értékű számjegyekhez tartozó helyi értékeket:

$(1011000101110)_2 = 4096+1024+512+32+8+4+2 = (5678)_{10}$

Ha nem tudjuk a kettő hatványokat, akkor az úgynevezett Horner szabály segítségével is átválthatjuk:

$(1011000101110)_2 = ((((((((((((1\cdot 2+0)\ 2+1)\ 2+1)\ 2+0)\ 2+0)\ 2+0)\ 2+1)\ 2+0)\ 2+1)\ 2+1)\ 2+1)\ 2+0 = (5678)_{10}$

Itt tulajdonképpen a hatványozás a zárójelek és a 2-es kiemelések segítségével jön létre. Például a legbelső zárójelben szereplő számjegy pont 2^{12}-el lesz megszorozva.

Tizenhatos számrendszer

A kettes számrendszer mellett elterjedten használjuk még a tizenhatos vagy hexadecimális számrendszert. A fenti jelöléseket használva a hexadecimális számrendszer esetén r = 16, és az egyes h_i számjegyek tizenhat különböző értéket vehetnek fel. Mivel a nyelvünk írásjeleiben nincs ennyi számok ábrázolására szolgáló szimbólum, ezért az ABC néhány betűjét használjuk a hiányzó számjegyek leírására. A tizenhatos számrendszerben minden számjegy 0..9, vagy A..F értéket vehet fel.

$$h_i \in \{0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F\}$$

Átváltás 10→16

Egy tízes számrendszerben ábrázolt számot egyszerűen átalakíthatunk tizenhatos számrendszerbe, ha elkezdjük tizenhattal osztani majd minden osztásnál külön vesszük az osztási maradékokat. Alakítsuk át a fenti példában szereplő 5678 decimális számot tizenhatos számrendszerbeli számmá.

Osztás	**Maradék**	**Számjegyek (h_i)**	**Helyi értékek (r^i)**
5678:16 = 354	14→**E**	$= h_0$	1
354:16 = 22	**2**	$= h_1$	16
22:16 = 1	**6**	$= h_2$	256
1:16 = 0	**1**	$= h_3$	4096

Vagyis $(5678)_{10} = (162E)_{16}$

Átváltás 2→16

Ha a legkisebb helyi értéktől kezdve négybites csoportokat képezünk a kettes számrendszerbeli alakból, majd ezeket a csoportokat külön-külön átírjuk tizenhatos számjegyekké, akkor megkapjuk a szám tizenhatos számrendszerbeli alakját. A legmagasabb helyi értéken 0-kal pótoljuk a hiányzó számjegyeket.

$(1011000101110)_2 = (0001)(0110)(0010)(1110)_2 = (162E)_{16}$

Tört számok kezelése kettes számrendszerben

Matematikai számítások során gyakran van igény tört számok kezelésére. Ezt az alkalmazott számrendszer alapjának negatív kitevőjű hatványai (-m..-1) segítségével tudjuk előállítani.

$$N = \sum_{i=-m}^{n} h_i \cdot r^i$$

Ilyenkor külön meghatározzuk a szám egész részét, majd egy kettedes ponttal elválasztva mellé írjuk a tört részét.

Például váltsuk át az alábbi négy+2 bites bináris számot decimális alakra.

$$(1011.11)_2 = (1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 + 1 \cdot 2^{-1} + 1 \cdot 2^{-2})_{10} = (8 + 2 + 1 + 0.5 + 0.25)_{10} = (11.75)_{10}$$

Ezt az ábrázolási módot gyakran hívjuk **fix pontos** ábrázolásnak, mert a számítás során nem változik az egész és a tört rész ábrázolásához használt számjegyek száma.

BCD ábrázolás

A BCD ábrázolás a tízes számrendszerben ábrázolt szám minden számjegyét négy biten, binárisan jeleníti meg. Ez az ábrázolás akkor hasznos, ha tízes számrendszerbeli számokat szeretnénk kezelni bináris formában. A fenti példában ábrázolt számot a következőképen ábrázoljuk BCD formában:

$(5678)_{10}$ = $(0101\ 0110\ 0111\ 1000)_{BCD}$

Kettes komplemens ábrázolás

Az előjeles számok kezelésére szolgáló számábrázolás. Csak előre rögzített bitszámmal működik. Egy n bites ábrázolás esetén az aritmetikai műveletek eredményei moduló 2^n szerint érvényesek (2^n-el vett osztás maradéka).

A **pozitív számok** alakja megegyezik a kettes számrendszerbeli ábrázolással, a **negatív számokat** azonban komplemens formában ábrázoljuk. A komplemens képzése: a negatív szám abszolút értékét kivonjuk a fenti 2^n modulusból. A gyakorlatban a komplemenst egyszerűbb úgy kiszámítani, hogy a negatív szám abszolút értékét (pozitív megfelelőjét) felírjuk binárisan az ábrázolásnak megfelelő bitszámmal (szükség esetén a felső biteken megfelelő mennyiségű nullával kiegészítve), majd bitenként invertáljuk, végül a legkisebb helyi értéken 1-et hozzáadunk.

A példánkban használt számot 16 biten ábrázolva a következőket kapjuk:

$(+5678)_{10}$ = $(0001\ 0110\ 0010\ 1110)_{2kompl}$

$(-5678)_{10}$ = $(1110\ 1001\ 1101\ 0010)_{2kompl}$

A kettes komplemens számábrázolás előnyös tulajdonsága, hogy a legmagasabb helyi értékű bit előjel bitként viselkedik, 1 értékű a negatív és 0 értékű a nem negatív számok esetén. A pozitív és negatív számok kezelésére egységes, bináris aritmetika használható a számok összeadására, kivonására.

A számábrázolási tartomány egyik fele a negatív, másik fele a nem negatív számok (nulla és a pozitív számok) ábrázolására szolgál. Ha n bites kettes komplemens formában egész számokat ábrázolunk, akkor az ábrázolható tartomány határai:

$$-\frac{2^n}{2} \cdots\cdots + \left(\frac{2^n}{2} - 1\right)$$

Ez 8 bit esetén (n=8) azt jelenti, hogy -128…+127 közötti számokat tudunk ábrázolni.

Lebegőpontos számábrázolás (IEEE754)

A nagyon kicsi, illetve nagy értékek ábrázolásakor gyakran alkalmazzuk a számok normál alakját. A normál alak (tízes számrendszerben) annyit jelent, hogy a számot egy olyan kéttényezős szorzat alakjában írjuk fel, ahol az első tényező egy 0 és 10 közé eső szám, a másik tényező pedig a 10 megfelelő egész kitevőjű hatványa.

Pl:

$(5678)_{10} \rightarrow 5.678 \times 10^3$

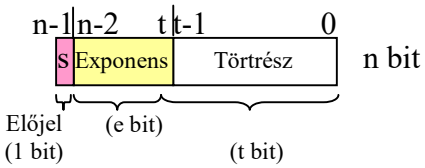
A normál alakot kettes számrendszerben is előnyösen használhatjuk. Ilyenkor a bináris értéket 1.xxx alakra hozzuk, majd ezt szorozzuk a 2 megfelelő egész kitevőjű hatványával.

Pl.:

$(1011000101110)_2 \rightarrow 1.011000101110 \times 2^{12}$

A lebegő pontos számábrázolás a kettes számrendszerbeli normál alakot alkalmazza, de az „1.” részt, vagyis az első egyest és a kettedes pontot nem tárolja (rejtett bit, ezzel tulajdonképpen egy bitet spórolunk a tárolás során). A nagyságrendet (vagyis, hogy hány helyi értékkel kellett léptetni a normálalak eléréséig) külön tároljuk (exponens).

Többféle lebegőpontos formátum van használatban, mi csak az IEE754 jelű szabványos ábrázolással foglalkozunk. A tároláshoz használt bitszám alapján különböző pontosság érhető el. A szakmai terminológiában gyakran használjuk a „feles”, „egyszeres”, „kétszeres”, „négyszeres” pontosság elnevezést. Az 1.ábra mutatja az egyes bitcsoportok jelentését, majd az 1.táblázat röviden összefoglalja a főbb különbségeket.



$$N = (-1)^S \cdot 2^{(\text{Exponens} - \text{eltolás})} \cdot (1.\text{Törtrész})$$

1. ábra

Elnevezés	**Tároláshoz használt bitek száma (n)**	**Exponens bitszáma (e)**	**Exponens Eltolása (o)**	**Törtrész bitszáma (t)**
„Feles”	16	5	15	10
„Egyszeres”	32	8	127	23
„Kétszeres”	64	11	1023	52
„Négyszeres”	128	15	16383	112
*„Intel x86 kiterjesztett” **	*80*	*15*	*16383*	*63*

1. Táblázat

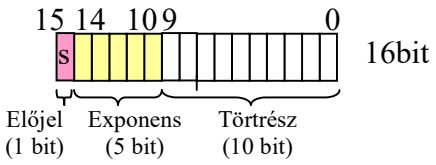
Közös jellemzője a fenti ábrázolási módoknak, hogy bizonyos kombinációknak speciális (szabványban rögzített) jelentése van. Ilyen speciális értéke pl. a végtelen, vagy a NaN (Not a Number) érték, amely a műveletvégzés során fellépő hibák esetén használatos. A végtelen ábrázolásakor az összes exponenst ábrázoló bit 1 értékű, a törtrész pedig zérus. A legmagasabb helyi értékű bit (MSB) megadja a szám előjelét. Amennyiben az MSB értéke 1, akkor az ábrázolt szám negatív.

Az exponens ábrázolását többletkódban oldották meg, ami annyit jelent, hogy a binárisan ábrázolt exponens értékből egy eltolás értéket le kell vonnunk az előjelhelyes érték kiszámításához. Ez az eltolás úgy számítható, hogy meghatározzuk az exponens bitek segítségével ábrázolható számok darabszámát és elosztjuk 2-vel, majd egyet levonunk belőle. (mert a legnagyobb érték eltérő jelentésű). Ez pl.: 8 bites exponens esetén $((2^8)/2)-1 = 127$ eltolás értéket jelent, ennyit kell levonni a decimálissá alakított exponensből az előjelhelyes érték előállításához.

Megjegyzés:

Az 1. táblázat utolsó sorában *-al jelölt ábrázolási módot az Intel processzorok lebegő pontos számításai során elterjedten használják. Ez utóbbi ábrázolás esetén az exponens és a törtrész között még egy vezérlő bit is található, de ezzel részletesen nem foglalkozunk. Ez a kiterjesztett formátum előnyösen használható 64 bites float műveletek során, mert a belső számolás a 80 bit miatt pontosabb lesz, és csak a művelet legvégén kell lekonvertálni 64 bitre. Ezért gyakran használják a „kiterjesztett kétszeres” pontosságú elnevezést is erre a 80 bites formátumra.

„Feles” pontosságú ábrázolás (16 bit) (elsősorban egyszerűbb mikrokontrolleres rendszerekben)



16bit

$$N = (-1)^s \cdot 2^{(Exponens-15)} \cdot (1.Törtrész)$$

Az exponens értéktartománya (15-ös többletkódban van ábrázolva):

$(00001)_2 – (01111)_2$	-14 → x 2^{-14}
$(11110)_2 – (01111)_2$	+15 → x 2^{15}
$(11111)_2$ → végtelen, NaN	
$(00000)_2$ → 0, nem normalizált érték	

Megkülönböztetett számok:

0x0000 → +0 0xEC00 → +végtelen

0x8000 → -0 0xFC00 → - végtelen

A nem normalizált érték esetén az exponens 0, a szám bináris értéke jelenti a számot (nem kell semmilyen nagyságrenddel szorozni. Ezt leginkább konvertáláshoz célszerű használni.

Példa: Alakítsuk át a bevezető részben is használt 5678 decimális számot egyszeres pontosságú lebegőpontos számmá:

Megoldás:

Első lépésben írjuk fel binárisan (ezt már az első feladatok között megtettük):

$(5678)_{10} = (1011000101110)_2$

Második lépésben írjuk fel a szám kettes számrendszerbeli normál alakját

1.011000101110 x 2^{12}

Harmadik lépésben állítsuk elő az exponens értékét 127-es többletkódban

$2^{12} \rightarrow 12+127 = 139 \rightarrow 10001011$

Negyedik lépésként illesszük össze az előjelet, az exponenst és a törtrészt a bevezető **1.** nélkül. A „hiányzó” alsó biteket 0-kkal pótoljuk.

Előjel bit: 0

Exponens: 10001011

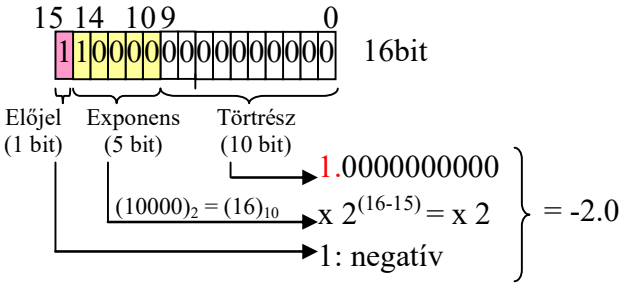
Tört rész: **1.**0110001011100000000000

$(5678)_{10} = (01000101101100010111000000000000)_{float32}$

Példa: Egy 16 bites IEEE754 float változó (5 bites exponens, 10 bites törtrész) hexadecimális értéke 0xC000. Számítsa ki a decimális értékét.

Megoldás:

Átváltjuk a feladatban megadott értéket binárisra, majd csoportosítjuk az egyes mezők szerint. 0xC000 → 1 10000 0000000000



Előjel bit: 1 → negatív az ábrázolt érték

Exponens: 10000 → levonjuk a 15-ös többletértéket $(10000)_2$-$(01111)_2$ = $(00001)_2$ = $(+1)_{10}$

Törtrész: 0000000000 → 1.0000000000 → 1.0

A keresett szám: - 1.0 x 2^{+1} = -2.0