Mikroprocesszorok gyorsítása

Tartalomjegyzék

Témakör	Főbb tartalmak	
BCD számábrázolás	A BCD kód alapja, implementációi. BCD összeadás. A BCD kód előnyei és hátrányai.	
Lebegőpontos számábrázolás	Normál alak tízes és kettes számrendszerben. IEEE 754 szerint lebegőpontos formátum. Decimális értékek átírása egyszer pontosságú lebegőpontos formátummá. Egyszeres pontosságú lebegőpontos számok decimális konverziója.	

A számábrázolás problematikája

A digitális rendszerek bitjei IGEN vagy NEM, azaz 1 vagy 0 értéket vehetnek fel. Mint arról már korábban is szó volt az „n” bites vezetékkötegeken (buszokon) 2^n számot vagy szimbólumot különböztethetünk meg.

Amennyiben nem alkalmazunk külön kódolást, akkor egy 8 bites adatvezetéken 2^8, azaz 256 szimbólumot tudunk megkülönböztetni. Ha a szimbólumokat decimális számoknak értelmezzük, akkor 8 bit esetén 0-255 –ig tudunk egész számokat ábrázolni.

Ugyanígy kezelve a szimbólumokat 16 adatbit esetén 2^{16}, azaz 0-65 535 –ig, 32 adatbit esetén 0-4 294 967 295- ig, vagy 64 adatbit esetén 0-18 446 744 073 709 551 615 –ig tudunk egész számokat ábrázolni.

Látszólag a 64 adatbit már igen nagy szám ábrázolására alkalmas, de ha negatív számokat is szeretnénk ábrázolni, akkor a legnagyobb érték már csak a jelzett felel lehet és akkor még mindig csak egész számokról beszélünk.

Irracionális számok ábrázolása esetén a bitszélesség növelése nem célravezető megoldás. Sok lehetőség mellett két elterjedt módszer a binárisan kódolt decimális (BCD) számábrázolás és a lebegőpontos számábrázolás.

A BCD számábrázolás

A binárisan kódolt decimális (Binary Coded Decimal), azaz BCD ábrázolás lényege, hogy a 10-es számrendszerben ábrázolt számok esetén minden helyiértéken 0-9 terjedő szimbólumokat tárolunk a gépen. Ennek megfelelően egy helyiérték ábrázolásához 4 bitre van szükség a jobb oldali táblázat szerint:

A táblázat alapján jól látható, hogy BCD kódolás esetén 8 biten 0-9 –ig (egy helyiérték bájtonként) vagy 0-99 –ig (két helyiérték bájtonként) tudunk számokat ábrázolni, míg bináris kódolás esetén 0-255 –ig. Tehát a BCD kódolás „pazarlóan bánik a bitekkel” mégis a mai napig rendelkezik előnyös tulajdonságokkal.

A BCD ábrázolás mind a mai napig használatos de már a számítógépek megjelenése óta ismert. A kódolásra több módszer használatos. Eleinte jellemző volt az egyes számítógép gyártók által kidolgozott és értelem szerűen a termékcsaládjaikban preferált különféle kódolás. Elsők között az IBM alkalmazta általános jelleggel a különféle BCD kódolásokat.

Bináris szám	Leképező bitek			
	D3	D2	D1	D0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
-	1	0	1	0
-	1	0	1	1
-	1	1	0	0
-	1	1	0	1
-	1	1	1	0
-	1	1	1	1

A BCD számábrázolás

A táblázat alapján látható, hogy a BCD számok négy bites blokkokban tárolhatók és értelmezhetők. A legelterjedtebb kódolásokat az alábbi táblázatok szemléltetik. A tízes számrendszerben ábrázolt 42 értékét BCD kódolásban 8 adatbitbit esetén a következők lehetnek:

Egy bájton két helyiértéket ábrázolt BCD

D7	D6	D5	D4	D3	D2	D1	D0
0	1	0	0	0	0	1	0
4				2			

Több bájton ábrázolt BCD kód az EBCDIC kódolás esetén (IBM). Itt a felső 4 bit értéke mindig „0”.

D7	D6	D5	D4	D3	D2	D1	D0
0	0	1	1	0	0	1	0
2							

D7	D6	D5	D4	D3	D2	D1	D0
0	0	1	1	0	1	0	0
4							

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	1	0
2							

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	1	0	0
4							

Több bájton ábrázolt BCD kód az ASCII kódolás esetén. Itt a felső 4 bit értéke mindig „0011”.

Ebben az esetben az egy bájton ábrázolt számjegyek épp megfelelnek az ASCII táblában a számjegyhez rendelt bináris kóddal.
Például a 00110010 a „2”-es szimbólum ASCII kódja.

Bináris szám	Leképező bitek			
	D3	D2	D1	D0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
-	1	0	1	0
-	1	0	1	1
-	1	1	0	0
-	1	1	0	1
-	1	1	1	0
-	1	1	1	1

A BCD számábrázolás

A fentieken kívül használatos még az úgynevezett pakolt BCD formátum. Ez hasonlít az egy bájton tárolt két helyiértékes kódoláshoz, de a legalsó 4 bit az előjelet tartalmazza. Pozitív a szám 1100 esetén és negatív 1101 esetén.

Ennek megfelelően a +42 az alábbi módon ábrázolható 16 biten:

D7	D6	D5	D4	D3	D2	D0	D0	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	1	0	0	0	0	1	0	1	1	0	0
0				4				2				+			

A -42 pedig az alábbi módon ábrázolható 16 biten:

D7	D6	D5	D4	D3	D2	D0	D0	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	1	0	0	0	0	1	0	1	1	0	1
0				4				2				-			

A pakolt BCD kódolás természetesen hatékonyabb az egy bájtos változatnál, de még mindig erősen pazarló a tisztán bináris kódoláshoz képest.

Bináris szám	Leképező bitek			
	D3	D2	D1	D0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
-	1	0	1	0
-	1	0	1	1
-	1	1	0	0
-	1	1	0	1
-	1	1	1	0
-	1	1	1	1

A BCD összeadás

A BCD kódolással tárolt számok összeadása is binárisan történik, de a műveletet követően szükség van korrekciókra.

Egy egyszerű példa segítségével könnyen érthetővé válik a BCD összeadás.

Számítsuk ki 5+6 értékt. Természetesen az eredmény 11.
Binárisan:

```
  (5) 0101
  (6) 0110
--------------
(11) 1011
```

Az eredmény tehát 11, de ez nem BCD! BCD kódban a decimális 11 a következő képpen alakul: 0001 0001

Ahhoz, hogy BCD kód szerint helyes eredményt kapjunk a bináris összeadás eredményéhez még decimális 6-ot hozzá kell adni!

```
(11)  1011
 (6)  0110
--------------
     1 0001
```

A BCD összeadás

Nagyobb számok esetén is hasonlóan kell eljárni.

Például adjuk össze 24-et 18-al, azaz számítsuk ki 24+18 értékét.

```
  (24) 0010 0100
  (18) 0001 1000
---------------------
       0011 1100
```

Az így kapott eredményhez adjunk hozzá 6-ot.

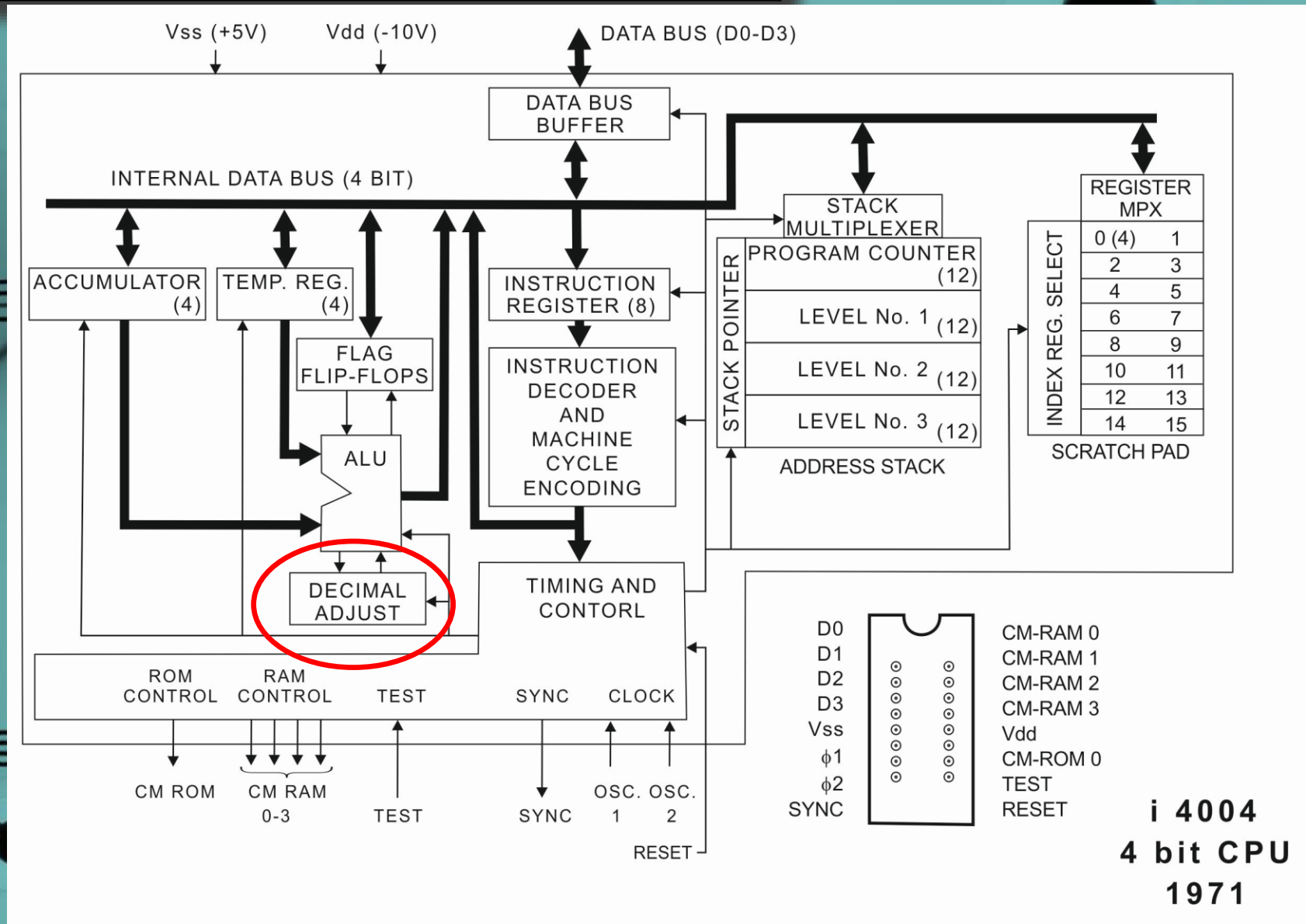
```
       0011 1100
  (6)  0000 0110
--------------------
(42)   0100 0010
```

A processzorok fejlődése során megjelentek a hardveres BCD összeadók, illetve korrektorok. Az Intel processzorok esetében például a DAA (decimal adjust after addition) utasítás segít a korrekcióban, de vannak kimondottan BCD műveletet tartalmazó processzorok is.

A BCD összeadás

Az ábrán az Intel i4004 processzorának felépítése látható. A világ első kereskedelmi forgalomban is elérhető mikroprocesszora már rendelkezett a BCD műveleteket támogató egységgel. Az ábrán piros karika jelzi az ALU-t támogató BCD korrektort.

Ezek a BCD korrektorok nem minden kezdeti processzorban voltak megtalálhatók. Mivel az Intel i4004 processzora kimondottan számológéphez lett kifejlesztve, a hatékony működés érdekében az architektúra részét képezi.



A BCD számábrázolás

A BCD ábrázolás előnyei:

- Bizonyos eseteben pontosabb számolást tesz lehetővé a bináris vagy lebegőpontos számábrázolásokhoz képest. Éppen ezért a mai napig alkalmazzák például pénzügyi vagy szimulációs számítások esetén.
- A számok megjelenítése lényegesen leegyszerűsödik, hiszen a megszokott tízes számrendszer szerint tároltak a helyiértékek.
- A tízes számrendszer szerinti számok bevitele is egyszerű, hiszen minden helyiértéken szereplő szimbólumot (számjegyet) kell csupán kódolni és a memóriában egymást követően tárolni.

A BCD ábrázolás hátrányai:

- Pazaralóan bánik a bitekkel ezért kevésbé hatékony kódolás a tisztán bináris kódolással szemben.
- A BCD-ből binárisra, illetve vissza konverzió összetett, szorzásokat tartalmazó művelet.
- A BCD műveletek valamivel összetettebb hardvert igényelnek a tisztán bináris műveleteket végző áramkörökhöz képest.

A lebegőpontos számábrázolás

Bár a 64 bites adatszélesség első ránézésre már igen nagy szám ábrázolását teszi lehetővé (0-18 446 744 073 709 551 615), mégis ha ezt negatív számok ábrázolására is fel kívánjuk használni, akkor a legnagyobb szám máris a fele lesz és ha mondjuk 10 tizedes pontos tört számokat is ábrászolni szeretnénk, akkor az iménti elfelezett szám tizede lesz a legnagyobb ábrázolható érték.

Az, hogy a 10 tizedesjegy kellen pontos-e a számítás céljától és a kezelt szám nagyságrendjétől függ. Ha például a tizedes vessző előtti érték (egészek) 5 000 000,000 000 000 1 akkor előfordulhat, hogy a példában szereplő 0,000 000 000 1 érték elhanyagolható, de még az is lehet, hogy akár a 0,1 is elhanyagolható.
Ha viszont az egész rész 1 akkor a tört értékek szerepe felértékelődik!

A lebegőpontos számábrázolás előnye, hogy általa jelentősen kiterjeszthető az ábrázolható számokk tartománya és éppen az imént említett nagyságrendhez igazodik a pontosság. Fontos megjegyezni, hogy épp ez utóbbi tulajdonsága sok esetben épp az ábrázolási mód hátránya is!

A lebegőpontos számábrázolás

A lebegőpontos szám ábrázolása megfelel a tízes számrendszerben alkalmazott normál alaknak ($a*10^b$) ahol a „a” az értékes számjegy (0,xxx formátumban), „10” a hatvány alapja és „b” a kitevő.
Például 98765,4321 normál alakja: $0{,}987654321*10^{-5}$

Mivel a számítógépek a kettes számrendszerben tárol adatokkal a leg hatékonyabbak a fenti normál alakot célszerű erre az alapra felírni: $a*2^b$

„a” értékénél a formátum 1,xxx. Mivel „a” értéke minden esetben 1-el kezdődik ezt az értéket nem szükséges eltárolni de természetesen számolunk vele!

A lebegőpontos szám általános felépítése a következő: $(-1)^s*a*2^b$ ahol
s: az előjel bit. 1, ha a szám negatív 0, ha a szám pozitív (1 bit),
a: az értékes számjegy 1,xxx alakban. Ezt nevezzük mantisszának (23 bit),
b: a kettedes vessző helyét meghatározó exponens (8 bit). Értéke -126 -tól 127 –ig terjedhet

Az IEEE 754 szabvány meghatározza a fenti formátumot és az egyes összetevőkhöz rendelt bitek számát.

Mivel a mantissza 8 bites, értéke $-126 \leq m \leq 127$ lehet, az ábrázolható számok 10^{-38} és 10^{38} nagyságrendek közé esnek.

A lebegőpontos számábrázolás

Egy példa a lebegőpontos számábrázolásra.
Legyen a tárolt bitsorozat a következő:

előjel | karakterisztika | mantissza

$2^7;\dots;2^2;2^1;2^0$ | $2^{-1};2^{-2};2^{-3};\dots;2^{-22};2^{-23}$

Mivel az előjel bit 0, a szám pozitív tehát $s=0$.

A karakterisztikát a következők szerint számítjuk: $2^7=128$. $b=2^7-127$, azaz $128-127=1$.

A mantissza a következők szerint alakul: $a=1+2^{-1}+2^{-4}+2^{-7}$, azaz $a=1+0,5+0,0625+0,0078125=1,5703125$.

Végül az ábrázolt szám: $x=(-1)^{s}*a*2^{b}$ azaz $x=(-1)^{0}*1,5703125*2^{1}=3,140625$.

A lebegőpontos számábrázolás

Egy példa egy decimális szám felírása lebegőpontos formátumba.
Legyen a felírandó tízes számrendserben értelmezett szám: 2,75

Első lépésben fel kell írni a 2,71-et bináris formába.

Az egész rész: 2 = 10.
A tizedes rész: 0,75=11. ($2^{-1}+2^{-2}$, azaz 0,5+0,25)
A 2,75 tehát binárisan 10.11

Második lépésben normalizáljuk az 10.11 bináris számot úgy, hogy a kettedes pont előtt legyen az első értékes számjegy, azaz 1.
$10.11 = 1.011 * 2^1$

A karakterisztika tehát 1, azaz b=1.
Ezért a karakterisztika bitsorozata 128 binárisan, azaz 1000 0000 (128-127).

A mantissza a fentiek alapján 0110 0000 0000 0000 0000 0000. Figyelem! A mantisszánál a normalizálás során látható első egész értéket (1-est) elhagyjuk, de a későbbiekben számolunk vele!

Mivel pozitív számot ábrázoltunk s=0.

0	1	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

A lebegőpontos számábrázolás

Az IEEE 754 és annak kiterjesztése több lebegőpontos számábrázolást definiál. Ezek felépítése azonos, csak az ábrázolható számok tartományában és a pontosságban van eltérés.

Megnevezés	előjelbit	Exponens bit (karakterisztika)	Mantissza	Teljes bitszélesség	Exponens eltolás
Fél pontos IEEE 754	1	5	10	16	15
Egyszeres pontosság	1	8	23	32	127
Dupla pontosság	1	11	52	64	1023
X86 kiterjesztett pontosság	1	15	64	80	16383
Négyszeres pontosság	1	15	112	128	16383

Noha a lebegőpontos számábrázolás segítségével jelentősen megnő az ábrázolható számok tartománya a számítások során minden esetben figyelembe kell venni az ábrázolás okozta pontatlanságakat. Ezek egyik szembetűnő esete mikor egy műveletben szerepelnek nagyon nagy és nagyon kis számok, illetve nagy számokkal végzett műveletek esetén a nagyon kis eltéréseket vizsgáljuk.

A lebegőpontos műveletek hagyományos aritmetikával rendkívül lassúak, így a processzorok fejlődése során a mérnökök speciális lebegőpontos aritmetikai egységeket fejlesztettek ki. Ezek az egységek kezdetben önálló chipben, később magában a processzorban kaptak helyet.