Lebegőpontos számok 6 bájtos alakja memóriacím szerint sorban:

0	1	2	3	4	5

0. bájt: exponens+129

5. bájt: mantissza előjele és a bináris vessző utáni 7 bit

4. bájt: mantissza következő 8 bitje

3. bájt: mantissza következő 8 bitje

2. bájt: mantissza következő 8 bitje

1. bájt: mantissza utolsó 8 bitje

Vagyis jelöléssel:

eeee eeee _ mmmm mmmm _ mmmm mmmm _ mmmm mmmm _ mmmm mmmm _ smmm mmmm,

Ahol s, m és e rendre a szám előjelét, mantisszáját és exponensét határozzák meg.

```
    0:00 00 00 00 00 00
    1:81 00 00 00 00 00
    2:82 00 00 00 00 00
    3:82 00 00 00 00 40
    4:83 00 00 00 00 00
    5:83 00 00 00 00 20
   10:84 00 00 00 00 20
    6:83 00 00 00 00 40
   -1:81 00 00 00 00 80
   -6:83 00 00 00 00 c0
    1:80 00 00 00 00 00
    0:7f 00 00 00 00 00
61666:90 00 00 00 e2 70
```

Példa: Vizsgáljuk meg a következő kifejezés értékeként előálló számot, majd annak kettes számrendszer beli normál alakját:

828+1/4+1/4096+1/65536/2+1/65536/16384 = 828,25025177d =

1100111100,01000000000100001000000000000001 b =

1,1001111000100000000001000010000000000001 b * 2d^9d

E számot a *Borland Pascal* féle Real típusban tárolva a következő 6 bájtos adatot kapjuk:

8ah 01h 20h 04h 10h 4fh, vagyis bináris alakban:

1000 1010 _ 0000 0001 _ 0010 0000 _ 0000 0100 _ 0001 0000 _ **0**100 1111

A fenti normál alak alapján számoljuk végig, hogy hogyan áll elő ez a hat bájt! A szám pozitív, így az előjelet leíró bit értéke 0 (ha negatív lenne, akkor 1 lenne), ez a bit van tárolva az utolsó bájt, vagyis a 4fh legbaloldalibb bitjében (lásd fent félkövérrel szedve). A következő hét bit (1001111 adja a normál alakból a mantissza bináris-vessző utáni hét bitjét (lásd a normál alakban az egyszeresen aláhúzott részt). A mantissza következő 7 bitje (lásd a pont-vonal aláhúzású részt) van tárolva a 4fh-et megelőző bájton, vagyis az 10h-n. A mantissza következő 8 bitje (lásd duplán aláhúzott rész) a 04h-n, az ez utáni 8 bit (lásd hullámos aláhúzású) a 20h-n, végül a mantissza maradék 8 bitje a 01h-n van tárolva. A mantissza legelső jegye (amely a bináris vessző előtt áll) mindig 1-es, ezért nem kell tárolni, így a helye felszabadul: ezen a biten a szám előjelét tároljuk (0 ha pozitív, és 1 ha negatív a szám). Ez a bit volt a hat bájt közül az utolsónak (azaz az 5.-nek) a legbaloldalibb bitje. Ezzel a mantissza tárolásával végeztünk.

Az exponens tárolásához csak azt kell tudni, hogy mindig az exponens értékénél 129d-cel nagyobb szám kerül tárolásra, vagyis mivel a normál alakban a kitevő értéke 9d, ezért 129d+9d = 138d = 8ah lesz tárolva.

Láthattuk, hogy a fenti példában a mantissza legutolsó bitjét is kihasználtuk (01h-ból az 1-est). Ez azt jelenti, hogy a fenti számhoz újabb bináris jegyeket fűzve nem fogjuk tudni tárolni az eltérést, vagyis 828+1/4+1/4096+1/65536/2+1/65536/16384 = 828+1/4+1/4096+1/65536/2+1/65536/16384+**1/65536/32768**. Az ilyen módon hozzáadott plussz érték már olyan alacsony helyiértéken szerepel, hogy a Real típus számára rendelkezésre álló 6 bájtban, pontosabban az 5 bájtnyi mantisszában nincs hely már neki. Ez a véges pontosság hátránya…

Nézzünk meg egy másik példát! Legyen a tárolandó szám a –5,125, és próbáljuk meg kitalálni, hogy a 6 bájt hogyan fog alakulni.

Ehhez fel kell írni a kettes számrendszer beli normál alakját:

-5,125d = -101,001b = -1,01001b * 2d^2d

Mivel ez negatív szám, ezért az utolsó bájt bal oldali bitje 1-es lesz, majd ezt követi a mantissza tört részéből az első 7 bit (vagyis a vessző utáni 7 bit): 01001b. Mivel a mantisszának a végére értünk (nincs több bit), ezért a hiányzó helyekre 0-kat írunk. Így a az utolsó bájt értéke **1**010 0100 vagyis a4h lesz, amelyből a félkövérrel szedett bit a fent említett előjel bit. A mantissza többi bitje csupa 0, vagyis a hat bájtból az utolsó 5 így néz ki: 00h 00h 00h 00h a4h. Már csak a nulladik bájt hiányzik, ehhez az exponensnél 129-cel nagyobb értéket kell tárolnunk. Mivel az exponens értéke 2d, ezért 129d+2d = 131d = 83h lesz, vagyis a teljes szám: 83h 00h 00h 00h 00h a4h.

Még egy fontos megállapítás a Real típusról: A legnagyobb ábrázolható szám a maximális exponens esetén, vagyis ffh esetén érhető el. Ez ffh – 129d = 255d –129d =127d-t jelent. Így a legnagyobb ábrázolható Real típusú szám nagyságrendileg a 2d*2^127d, vagyis kb. 1,7d*10^38d. Ez ugyan 39 jegyű szám, de ne felejtsük el, hogy nem tudunk minden jegyet tárolni: mivel a mantissza csak 5*8-1=39 bitje kerül tárolásra, ez kb. 12 decimális jegynek felel meg, így a többi számjegy mindenképpen 0 lesz. Ez azt jelenti, hogy a 10^39 nagyságrendű számból csak a 10^28-nál nagyobb helyiértékek tárolhatók, ezért a 10^39+1 kifejezés értékét Real típusú változóban tárolva ugyanazt a 6 bájtot fogjuk kapni, mintha a +1 nem szerepelne, vagyis *egyenlőnek fogja tekinteni ezt a két értéket a gép!*

Most vizsgáljuk meg, hogy a Real típus helyett Single-vel dolgozva milyen bájtokat kapunk! Ehhez a következő ismeretekre van szükség:

1. a Single típus nem 6, hanem csak 4 bájtot foglal el

2. az előjel itt is az utolsó (tehát a 3.) bájt legbaloldalibb bitjén van tárolva

3. az exponens tárolására 8 bit áll rendelkezésre, amelyet úgy kaphatunk meg, hogy a 3. bájt jobboldali 7 bitjét, majd a 2. bájt legbaloldalibb bitjét egymás mellé írjuk (ilyen sorrendben!), az így kapott szám 127-tel nagyobb, mint az exponens.

4. A mantissza bináris-vessző utáni első 7 bitje a 2. bájt jobb oldali hét bitjén, a következő 8 bitje az 1. bájt bitjein végül az utolsó 8 bit a 0. bájt bitjein helyezkedik el.

A korábbi jelölést alkalmazva:

mmmm mmmm _ mmmm mmmm _ emmm mmmm _ seee eeee.

Példa:

Írjuk fel a 12768,125d-t a Single adattípus bájtjainak alakjában!

Ennek a számnak a kettes számrendszer beli normál alakja:

12768,125d = 11000111100000,001b = 1,1000111100000001 b * 2d ^13d.

Innen a mantisszából a vessző utáni első 7 bit (lásd az aláhúzottakat) kerül a 2. bájt jobb oldali 7 bitje helyére, a következő 8 bit (lásd pontozott aláhúzásúak) kerülnek az 1. bájt bitjeibe, végül a fennmaradó bitek (most az 1000 0000, vagyis a dupla aláhúzású 1 a hiányzó 0-kkal kiegészítve) kerülnek a 0. bájtba. A fenti jelöléseket alkalmazva azt már tudjuk, hogy a mantissza bitjei így lesznek tárolva: 1000 0000 _ 1000 0000 _ e100 0111 _ seee eeee.

Mivel a szám előjele pozitív, tehát a 3. bájt legbaloldalibb bitje 0 lesz: 1000 0000 _ 1000 0000 _ e100 0111 _ 0eee eeee.

Ahhoz, hogy a hiányzó bitek értékét megkapjuk, az exponens értékéhez hozzá kell adni 127-et, majd ennek a nyolcbites alakjából a legjobboldalibb bitet a 2. bájt bal szélső bitje helyére tenni, és a többi 7 bitet a 3. bájt jobb oldali bitjei helyére: 13d + 127d = 140d = 1000110**0**b, vagyis a (félkövérrel szedett) 0 kerül a 2 bájtba, és az 1000110 kerül a 3. bájt jobb oldali 7 bitje helyére.

Így a végeredmény: 1000 0000 _ 1000 0000 _ 0100 0111 _ 0100 0110, vagyis 80h 80h 47h 46h.

Másik példa: -95634,625. Melyek az őt leíró Singe adattípusú változó bájtjai?

-95634,625d = 10111010110010010,101b = 1,0111010110010010101b *2d^16d. Innen a mantissza vessző utáni bitjeit megfelelően elhelyezve kapjuk:

0101 0000 _ 1100 1001 _ e011 1010 _ seee eeee.

A szám negatív, így az előjel bit 1 lesz:

0101 0000 _ 1100 1001 _ e011 1010 _ 1eee eeee.

Az exponens pedig: 16d+127d = 143d = 10001111b. Ennek bitjeit megfelelő sorrendben szétosztva az „e” karakterek között:

0101 0000 _ 1100 1001 _ 1011 1010 _ 1100 0111, vagyis 50h c9h bah c7h.

Harmadik példa: -0,03515625d.

-0,03515625d = -0,00001001b = -1,001b * 2d ^(-5d)

Innen a mantissza bitjei:

0000 0000 _ 0000 0000 _ e001 0000 _ seee eeee.

A szám előjele negatív (mellesleg az exponens is negatív, de az minket most nem érdekel, hanem csak az eredeti szám előjele), ezért:

0000 0000 _ 0000 0000 _ e001 0000 _ 1eee eeee.

Az exponens értékére pedig: -5d+127d = 122d = 01111010b*, vagyis:

0000 0000 _ 0000 0000 _ 0001 0000 _ 1011 1101, hexadecimális alakban pedig: 00h 00h 10h bdh.

Végezetül álljon itt néhány példa a gyakorláshoz:

	Single	Real
3.713623047	00 ac 6d 40	82 22 00 00 ac 6d
7235.2109375	b0 19 e2 45	8d 00 00 b0 19 62
-83662.390625	32 67 a3 c7	91 00 00 32 67 a3
-0.017333984	00 00 8e bc	7b 76 cc ff ff 8d
-5362.267333984	23 92 a7 c5	8d 00 80 23 92 a7