

11. Kayan Noktalı Sayılar (FloatingPointNumbers)

10 Tabanı:

$$(12.34)_{10} = 12 + \frac{34}{100} \text{ ya da } (12.34)_{10} = 12 + \frac{3}{10} + \frac{4}{100} = 1 \cdot 10^1 + 2 \cdot 10^0 + 3 \cdot 10^{-1} + 4 \cdot 10^{-2}$$

2 Tabanı:

$$(101.11)_2 = 5 + \frac{3}{4} \text{ ya da } (101.11)_2 = 5 + \frac{1}{2} + \frac{1}{4} \quad , \quad (0.1111)_2 = \frac{15}{16} \quad , \quad (0.1)_2 = \frac{1}{2}$$

Virgüllü (noktalı) sayıları bellekte tutmak için akla ilk gelen yöntem sabit noktalı {fixedradix} gösterilirdir.

Sayının noktadan önceki ve sonraki kısımları için sabit uzunlukta yerler ayrılır.

Sabit noktalı gösterilim pratik değildir. Bellekte fazla yer kaplar, işlem yapmak için uygun değildir.

Örneğin 1 trilyon (10^{12}) göstermek için 40 bit gerekir. $10^{12} = 2^{40}$

Benzer şekilde virgülden sonra $1/10^{12}$ hassasiyet için de 40 bit gereklidir. Toplam 80 bit.

Üstel gösterilim (Scientific notation, exponential notation) kullanılır:

$$\pm F \cdot B^{\pm E}$$

F: *Fraction* (Kesir, Mantis)

E: *Exponent* (Üs)

B: *Base* (Taban)

Bellekte: $\pm . F$ ve E tutulur.

Örnek:

$$976,000,000,000,000 = 0.976 \times 10^{15}$$

$$0.000\,000\,000\,000\,976 = 0.976 \times 10^{-12}$$

Normalize Sayı :

Noktanın yerine önceden karar verilir ve bu yer bilgisi bellekte tutulmaz .

Örneğin , noktanın her zaman sıfırdan farklı en yüksek anlamlı sayının solunda

olduğu kabul edilir

$$3.14 = 0.314 \times 10^1$$

Örneğin bellekte $\pm FFF \pm EE$ şeklinde tutulabilir . +314+01

Yükseltilmiş Üs (BiasedExponenet) :

Üs değerinin negatif olmaması için üs değeri bellekte saklanmadan önce belli bir

değer “ökçe” (bias) ile toplanır (üs yükseltilir) .

Böylece üssün işaretinin saklanmasına gerek kalmaz ve aritmetik işlemlerde

(karşılaştırmada) kolaylık sağlanır .

Single (32 bit)

İşaret	E	F
S	Üs	Kesir
1	8	23

Üs 127 yükseltilmiştir

E'deki bit sayısı k olmak üzere üs $(2^{k-1} - 1)$ kadar yükseltilir.

Double (64 bit)

İşaret	E	F
S	Üs	Kesir
1	11	52

Üs 1023 yükseltilmiştir

Güncel standartta
16 bitlik (*half*) ve
128 bitlik (*quadruple*)
sayılar da
bulunmaktadır.

Normalize Sayı (IEEE 754) :

Noktanın her zaman sıfırdan farklı en yüksek anlamlı sayının sağında olduğu kabul edilir .

İkili düzende çalışıldığına göre "0" dan farklı sayı "1" dir .

$$(10110 . 101)_2 = 1 . 0110101 \times 2^4$$

Noktadan önce her zaman 1 olduğu bilindiğinden bu 1 değeri de bellekte tutulmaz .

Buna **gizli 1** (hiddenone) denir .

Örnek:

$$(+22.625)_{10} ?$$

$$(22)_{10} = (10110)_2$$

.625'in 2 tabanındaki karşılığının bulunması:

$$2 \times 0.625 = 1 + 0.25$$

$$2 \times 0.25 = 0 + 0.5 \quad \text{Yüksek anlamlı bit}$$

$$2 \times 0.5 = 1 + 0$$

$$\Rightarrow (0.625)_{10} = (0.101)_2$$

5/8

$$(+22.625)_{10} = (+10110.101)_2 = +1.0110101 \times 2^4 \text{ (Normalize)}$$

Bellekte "Single" olarak:

0	10000011	0110101...000
İşaret	Yükseltilmiş	Mantıs
+	127 + 4 üs	

Mantısın düşük anlamlı bitleri
"0" ile dolduruldu.
Çünkü solunda nokta (.) var.

Float sayıları bit bit yazan program

```
#include <stdio.h>
void printFloatBits(float);

int main()      // floatBits.c
{
    float x;
    printf("Enter a floating-point numbers: ");
    scanf("%f", &x);
    printf("Bits of %f are:\n", x);
    printFloatBits(x);
    putchar('\n');
    return 0;
}

void printBits(unsigned int a){
    static int flag = 0;
    if(flag != 32) {
        ++flag;
        printBits(a/2);
        printf("%d ", a%2);
        --flag;
        if(flag == 31 || flag == 23) putchar(' ');
    }
}

void printFloatBits(float x){
    unsigned int *iP = (unsigned int *)&x;
    printBits(*iP);
}
```

Başka bir program

```
#include <stdio.h>
```

```
union {  
struct {  
    unsigned b0:1;unsigned b1:1;unsigned b2:1;unsigned b3:1;  
    unsigned b4:1;unsigned b5:1;unsigned b6:1;unsigned b7:1;  
    unsigned b8:1;unsigned b9:1;unsigned b10:1;unsigned b11:1;  
    unsigned b12:1;unsigned b13:1;unsigned b14:1;unsigned b15:1;  
    unsigned b16:1;unsigned b17:1;unsigned b18:1;unsigned b19:1;  
    unsigned b20:1;unsigned b21:1;unsigned b22:1;unsigned b23:1;  
    unsigned b24:1;unsigned b25:1;unsigned b26:1;unsigned b27:1;  
    unsigned b28:1;unsigned b29:1;unsigned b30:1;unsigned b31:1;  
    }b;
```

```
float deg;  
}fsayi;
```

```
int main() //  
{  
    float x;  
  
    printf("Enter a floating-point numbers: ");  
    scanf("%f", &x);  
    fsayi.deg=x;  
  
    printf("Bits of %f are:\n", x);  
    printf("%d%d%d%d%d", fsayi.b.b31,fsayi.b.b30,fsayi.b.b29,fsayi.b.b28,fsayi.b.b27);  
    printf("%d%d%d%d%d", fsayi.b.b26,fsayi.b.b25,fsayi.b.b24,fsayi.b.b23,fsayi.b.b22);  
    printf("%d%d%d%d%d", fsayi.b.b21,fsayi.b.b20,fsayi.b.b19,fsayi.b.b18,fsayi.b.b17);  
    printf("%d%d%d%d%d", fsayi.b.b16,fsayi.b.b15,fsayi.b.b14,fsayi.b.b13,fsayi.b.b12);  
    printf("%d%d%d%d%d", fsayi.b.b11,fsayi.b.b10,fsayi.b.b9,fsayi.b.b8,fsayi.b.b7);  
    printf("%d%d%d%d%d", fsayi.b.b6,fsayi.b.b5,fsayi.b.b4,fsayi.b.b3,fsayi.b.b2);  
    printf("%d%d%", fsayi.b.b1,fsayi.b.b0);  
    putchar('\n');  
    return 0;  
}
```