Binární reprezentace čísel

Číselný rozvoj

V desítkové soustavě jsou cifry čísla `abc,de` zapsány jako $a\times10^2+b\times10^1+c\times10^0+d\times10^{-1}+e\times10^{-2}$. Máme-li např. číslo `314`, jsou to tři stovky, jedna desítka a čtyři jednotky. Budeme-li toto číslo opakovaně dělit 10, zprava budou jako zbytky po dělení vycházet jednotlivé cifry:

```
314 % 10 = 4
31  % 10 = 1
3   % 10 = 3
```

A číslo přečteme zdola nahoru. Obdobně můžeme vyjádřit číslo v jiných soustavách, stačí změnit modulus. 314 v binární soustavě vypočteme stejným algoritmem:

```
314 % 2 = 0
157 % 2 = 1
78  % 2 = 0
39  % 2 = 1
19  % 2 = 1
9   % 2 = 1
4   % 2 = 0
2   % 2 = 0
1   % 2 = 1
```

Výsledek přečtený zdola je tedy `100111010`. Číslo `0.24` můžeme obdobně vyhodnotit násobením základu soustavy, desetinná čísla budou překračovat desetinnou čárku zleva doprava, např. pro `0.24`:

```
0.24 × 10 = 2.4
0.4  × 10 = 4.0
```

Obdobně pro binární soustavu:

```
0.24 × 2 = 0.48   0.72 × 2 = 1.44
0.48 × 2 = 0.96   0.44 × 2 = 0.88
0.96 × 2 = 1.92   0.88 × 2 = 1.76
0.92 × 2 = 1.84   0.76 × 2 = 1.52
0.84 × 2 = 1.68   0.52 × 2 = 1.04
0.68 × 2 = 1.34   0.04 × 2 = 0.08
0.36 × 2 = 0.72   0.08 × 2 = 0.16 …
```

Float

`0.24` je tedy v binární podobě `0.00111101…` a číslo `314.24` je `100111010.00111101011100…`. To je zápis s pevnou desetinnou čárkou, který se z praktických důvodů pro elektronické výpočty nepoužívá. U velkých čísel obvykle není zapotřebí znát tolik desetinných míst, u malých je naopak žádoucí zaznamenat jich více. K tomu se používá úsporný zápis s plovoucí desetinnou čárkou (**float**), který IEEE754 specifikuje pro tvar $M\times2^E$, kde

- `s` (1b) je *signum*, znaménko: 0 pro kladná čísla, 1 pro záporná

- `E` (8b) je exponent (o kolik znaků se posouvá desetinná tečka doleva). U malých čísel se naopak posouvá doprava (záporný exponent) a kóduje se v **posunutém kódu**, kde 0=-127, 1=-126, … 126=-1, 127=0, 128=1, … 254=127, 255=128. Pokud tedy k exponentu přičteme 127, dostaneme jeho reprezentaci v posunutém kódu a naopak, pro dekódování posunutého kódu odečteme 127.
- `M` (23b) je desetinná část (bez úvodní 1)

IEEE754 uvádí ještě typ **double** (dvojitá přesnost, 64b), kde je vyhrazeno 11 b na exponent a 52 b na mantisu

Příklad

Mějme `314.24` je `100111010.00111101011100…`

To můžeme přepsat do exponenciálního tvaru: $1.00111010001111010111100 \times 2^{8}$. Exponent (8) je v doplňkovém kódu 8+127=135 = `10000111`. Binární podoba tohoto čísla je tedy

```
0 10000111 00111010001111010111000
s    E               M
```

V souboru bude toto číslo zabírat následující 4 byty (4×8=32)

```
0100 0011 1001 1101 0001 1110 1011 1000
 4    3    9    D    1    E    B    8
```

Můžeme si ověřit kódem:

```
float number = 314.24f;
uint8_t* b = (uint8_t*)(&number);
printf("%2X %2X %2X %2X\n", b[0], b[1], b[2], b[3]);
```

Uvedenou posloupnost bytů vidíme v obráceném pořadí: `B8 1E 9D 43`. Na x86 architektuře se používá *little endian* kódování, tedy nejméně významné byty se zapisují jako první. Konvertovat můžeme i obráceným směrem:

```
uint8_t bytes[] = { 0xB8, 0x1E, 0x9D, 0x43 };
float* number = (float*)bytes;
printf("%.2f\n", *number);
```

Uvedené operace řeší na počítačích speciální jednotka: *floating point unit*, FPU. Výkonnost stroje se pak uvádí v jednotkách GFLOPS (*giga floating point operations*). Mikrokontrolery obvykle nemají FPU, a proto neumí nativně počítat s typem *float*. Běžně se zde počítá s pevnou desetinnou čárkou (např. 6.7°C můžeme zakódovat jako `67`). Výkonnost mikrokontrolerů proto měříme v jednotkách MIPS (*mega instructions per second*).

Všimněte si také, že číslo `314.24` se nám nepodařilo v tomto formátu zachytit přesně: pokud si číslo oříznuté na 23 binárních desetinných míst převedeme do desítkové soustavy, dostaneme hodnotu 314.239990234375. Vyzkoušejte si:

```
int n = number*1000;
printf("%d\n", n); // 314239
```

Proto při operacích s desetinnými čísly neprovádíme operaci porovnávání. Následující kód vytiskne `wtf???`:

```
float number = 314.24f;
if(number * 1000 == 314240) puts("as expected");
else puts("wtf???");
```