Liczby zmiennoprzecinkowe

1 Liczby zmiennoprzecinkowe

Najprostszym sposobem reprezentowania liczb rzeczywistych byłaby reprezentacja stałopozycyjna: zakładamy, że mamy n bitów na część całkowitą oraz m na część ułamkową. Wadą takiego rozwiązania jest stosunkowo niewielki przedział, z którego liczby możemy reprezentować: często podczas obliczeń używamy zarówno wartości bardzo dużych jaki i bardzo małych. Dlatego w komputerach przechowuje się liczby rzeczywiste w postaci zmiennoprzecinkowej (ang. *floating point*): $\pm m \cdot b^e$. Pamiętamy osobno znak, **mantysę** m oraz **wykładnik** e. Podstawa b jest ustalona (zazwyczaj 2) i nie jest jawnie przechowywana.

W konkretnej reprezentacji na mantysę i wykładnik przeznaczone są odpowiednie, ustalone liczby bitów. Zatem potrafimy reprezentować skończoną liczbę wartości. Im dłuższa mantysa, z tym większą dokładnością możemy reprezentować liczby. Z kolei im dłuższy wykładnik, tym większy przedział z jakiego liczby potrafimy reprezentować.

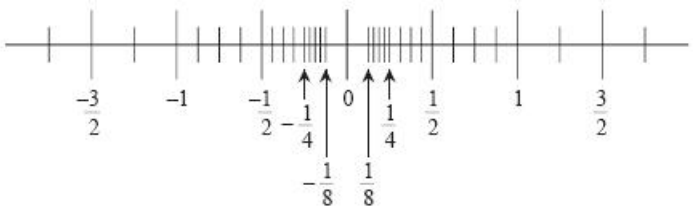
Każdą liczbę można zapisać w postaci zmiennopozycyjnej na wiele sposobów. W konkretnych reprezentacjach ustala się postać jednoznaczną, tzw. **znormalizowaną**. Zazwyczaj przyjmuje się, że przecinek w mantysie ustawiony jest bezpośrednio przed pierwszą cyfrą znaczącą lub za nią. W przypadku podstawy reprezentacji $b = 2$ pierwsza cyfra znacząca – 1 nie jest zazwyczaj jawnie pamiętana. Mówimy wtedy o **ukrytej 1**. Zauważmy, że przy takim założeniu nie można reprezentować liczby 0. Dlatego 0 jak i kilka innych wartości traktowanych jest wyjątkowo i przypisywane są im specjalne ciągi bitów.

1.1 Prosta modelowa reprezentacja

Rozważmy prostą modelową reprezentację, w której liczby pamiętane są jako ciągi sześciobitowe *zeemmm*, gdzie z oznacza bit znaku (0 - plus, 1 – minus), ee to dwubitowy wykładnik pamiętany z przesunięciem o 2 (tzn. liczba x jest pamiętana jako naturalny kod binarny liczby $x + 2$; reprezentacja z przesunięciem jest standardem dla wykładników), a mmm to trzy bity znormalizowanej mantysy. Zakładamy, że pierwsza jedynka w mantysie nie jest ukryta, a zero reprezentowane jest jako specjalny ciąg 000000.

Reprezentowalne wartości przedstawione są na rysunku 1. Najmniejsza reprezentowalna wartość dodatnia to $\frac{1}{8}$ – wstawiamy najmniejszą możliwą mantysę: 0.100 oraz najmniejszy wykładnik: -2. Odpowiedni ciąg bitów to 000100. Podobnie, największą wartością jest $1\frac{3}{4}$: 011111. Zauważmy, że pomiędzy reprezentowalnymi liczbami pojawiają się różne odstępy – im większe wartości, tym większe odstępy. I tak najmniejszy odstęp wynosi $\frac{1}{32}$ (gdy wykładnik jest równy -2 i zmieniamy mantysę o $\frac{1}{8}$, a największy – $\frac{1}{4}$, gdy wykładnik jest równy 2. Za to mniej więcej stała jest względna odległość: stosunek wartości liczby do wartości jej sąsiada. Zatem możemy powiedzieć, że w naszej reprezentacji mamy mniej więcej stały względny błąd przybliżenia.

Zauważmy teraz, że nie wszysktkie ciągi bitów w naszej reprezentacji są poprawne: niepoprawne są te, które mają 0 jako pierwszą cyfrę mantysy (z wyjątkiem ciągu 000000). Zatem nasza reprezentacja pozwala przechowywać 33 różne wartośći.



Rysunek 1: Liczby reprezentowalne w naszym modelu

Jeszcze jedną charakterystyczną cechą reprezentacji zmiennopozycyjnej (znormalizowanej) jest stosunkowo duży odstęp pomiędzy zerem a pierwszą reprezentowalną wartością. Przedział pomiędzy zerem a pierwszą wartością reprezentowalną nazywany jest **niedomiarem** (odpowiednio dodatnim lub ujemnym). Mówimy także o **nadmiarze** (również dodatnim lub ujemnym) – jest to przedział powyżej (poniżej) największej (najmniejszej) reprezentowalnej wartości.

O arytmetyce zmiennopozycyjnej będziemy mówić nieco dalej, teraz spróbujmy wykonać w naszej reprezentacji proste działanie $a + b$ dla $a = 0.2$ i $b = 0.7$. Przekształćmy nasze ułamki na system binarny: $a = 0.00110011... \approx 0.110 \cdot 2^{-2}, b = 0.10110... \approx 0.101 \cdot 2^0$. Ponieważ możemy pamiętać tylko trzy bity mantysy, to już na początku tracimy precyzję. Wyrównujemy wykładniki zwiększając pierwszy do 0: $a = 0.001 \cdot 2^0$. Ponownie tracimy precyzję (w rzeczywistości może być ciut lepiej, bo obliczenia pośrednie wykonywane są zazwyczaj na rozszerzonej reprezentacji – zawierającej dodatkowe bity). Dodajemy mantysy: $a + b = 0.110 \cdot 2^0$. Otrzymujemy zatem 0.75.

Podobne błędy napotykamy w rzeczywistości. Oto prosty przykład ilustrujący błąd wynikający z braku dokładnej reprezentacji dla pewnych liczb. Uruchom następujące programy w języku C:

```
int main{}
{
   float suma=0;
   long i;

   for (i=0; i<100000; ++i)
     suma=suma+0.6;

   printf{"%f", suma);
}


int main{}
{
   float suma=0;
   long i;

   for (i=0; i<100000; ++i)
     suma=suma+0.5;
```

```
    printf{"%f", suma);
}
```

W pierwszym przypadku wynik odbiega od oczekiwanego, w drugim jest poprawny. Wynika to z faktu, że 0.6, w przeciwieństwie do 0.5 nie ma dokładnej reprezentacji w systemie dwójkowym (z ograniczoną liczbą bitów po przecinku).

2 Standard IEEE 754

Norma IEEE 754 jest powszechnie obowiązującym standardem w jakim przechowywane są we współczesnych komputerach liczby zmiennopozycyjne. Oprócz formatu danych określa on też pewne zasady wykonywania obliczeń arytmetycznych, dzięki czemu można założyć, że ten sam program, napisany np. w języku C, uruchomiony na różnych maszynach da te same rezultaty.

Mamy dwa formaty: 32-bitowy – pojedynczej precyzji (`float` w C) i 64-bitowy (podwójnej precyzji) (`double` w C). Dodatkowo definiowane są formaty pomocnicze: rozszerzony pojedynczej precyzji i rozszerzony podwójnej precyzji. Służą one do wykonywania obliczeń pośrednich.
Format pojedynczej precyzji Mantysa: 23 bity (znormalizowana, ukryta 1 **przed** przecinkiem), wykładnik: 8 bitów (przesunięcie 127), zakres liczb dodatnich: $10^{-38}, 10^{+38}$, liczba reprezentowalnych wartości: $1,98 \cdot 2^{31}$.
Format podwójnej precyzji Mantysa: 52 bity (znormalizowana, ukryta 1 **przed** przecinkiem), wykładnik: 11 bitów (przesunięcie 1023), zakres liczb dodatnich: $10^{-308}, 10^{+308}$, liczba reprezentowalnych wartości: $1,99 \cdot 2^{63}$.

Niektóre sekwencje bitów są interpretowane w specjalny sposób. Są to sekwencje z wykładnikiem składającym się z samych zer lub samych jedynek:

- same 0 w wykładniku, same zera w mantysie: reprezentują 0 (dodatnie lub ujemne...)
- same 0 w wykładniku, niezerowa mantysa: liczb zdenormalizowana (bit na lewo od przecinka jest zerem, wykładnik wynosi -126 lub -1022); pomysł na redukcję odstępu pomiędzy zerem a najmniejszą dodatnią (ujemną) liczbą reprezentowalną.
- same 1 w wykładniku, same 0 w mantysie: plus lub minus nieskończoność
- same 1 w wykładnkiu, niezerowa mantysa: **NaN** (not a number) sytuacja wyjątkowa

3 Arytmetyka zmiennoprzecinkowa

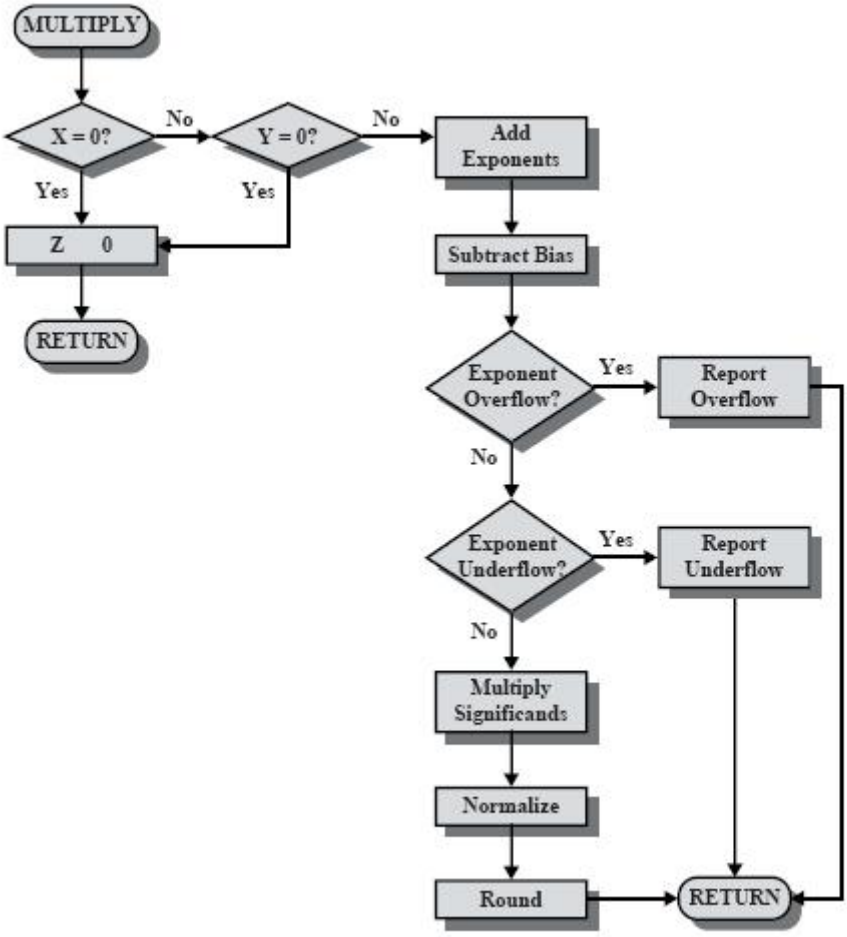
Ze względu na zupełnie inną reprezentację za operacje arytemtyczne na liczbach zmiennopozycyjnych odpowiedzialne są zupełnie inne obwody procesora niż za operacje całkowitoliczbowe.

3.1 Dodawanie i odejmnowanie

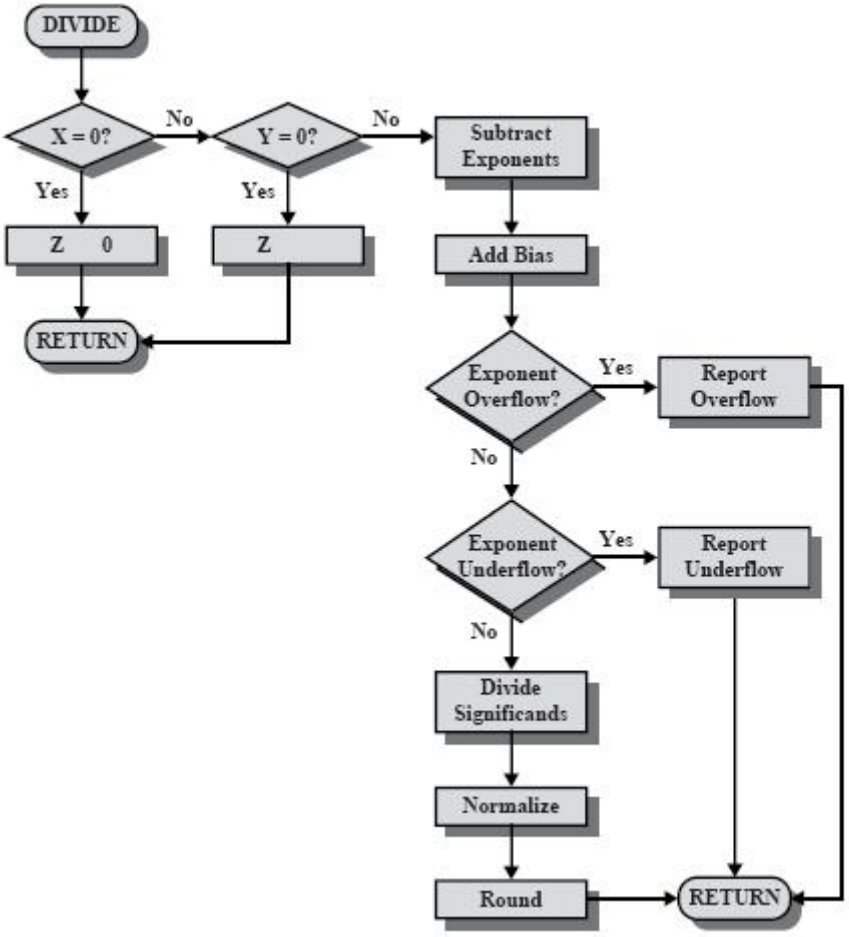
1. Sprawdzanie zer.
2. Wyrównywanie wykładników
3. Dodawanie lub odejmnowanie mantys
4. Normalizowanie wyniku.

3.2 Mnożenie i dzielenie

1. Sprawdzanie zer.
2. Dodawanie lub odejmowanie wykładników
3. Mnożenie lub dzielenie mantys, ustawianie znaku
4. Normalizowanie wyniku
5. Zaokrąglanie



Rysunek 2: Mnożenie zmiennopozycyjne



Rysunek 3: Dzielenie zmiennopozycyjne