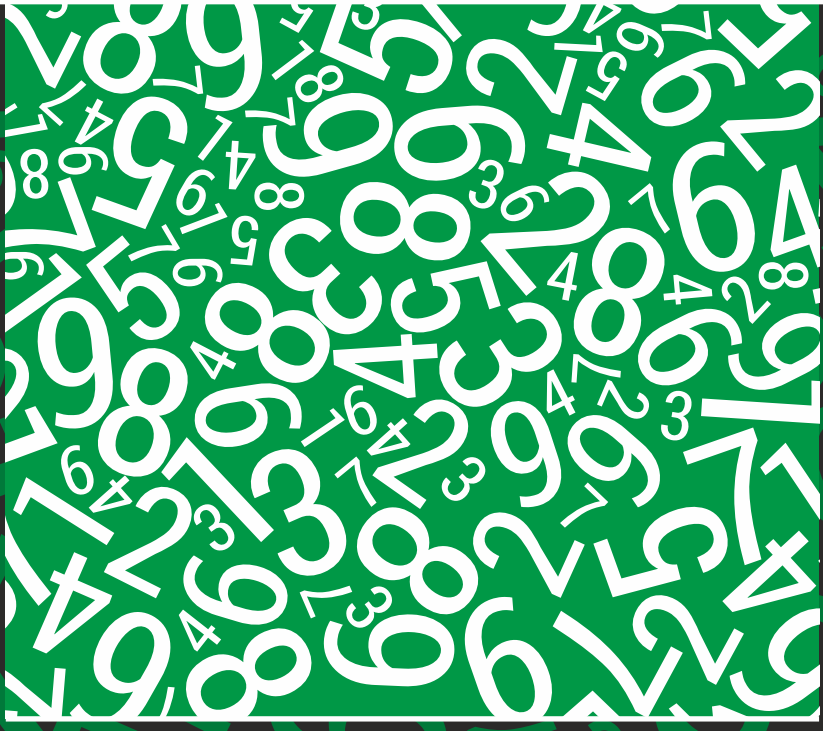
ĆWICZENIA Z PRZEDMIOTU WPROWADZENIE DO INFORMATYKI

ARYTMETYKA ZMIENNOPRZECINKOWA

IRENA BUŁATOWA



Politechnika Białostocka

Irena Bułatowa

ĆWICZENIA Z PRZEDMIOTU WPROWADZENIE DO INFORMATYKI
ARYTMETYKA ZMIENNOPRZECINKOWA

Politechnika Białostocka
OFICYNA WYDAWNICZA POLITECHNIKI BIAŁOSTOCKIEJ
BIAŁYSTOK 2022

Recenzent:
dr inż. Grzegorz Rubin

Redaktor naukowy dyscypliny informatyka techniczna i telekomunikacja:
prof. dr hab. Jarosław Stepaniuk

Korekta językowa:
mgr Natalia Popławska

Skład, grafika i okładka:
Marcin Dominów

ISBN 978-83-67185-42-4
ISBN 978-83-67185-43-1 (eBook)
DOI: 10.24427/978-83-67185-43-1

Druk: PPH Remigraf sp. z o.o.

Oficyna Wydawnicza Politechniki Białostockiej
ul. Wiejska 45C, 15-351 Białystok
e-mail: oficyna.wydawnicza@pb.edu.pl
www.pb.edu.pl

Spis treści

1. Wstęp

Do przechowywania liczb rzeczywistych w systemach komputerowych wykorzystywane są formaty zmiennoprzecinkowe. Zapis zmiennoprzecinkowy oparty jest na przedstawieniu liczby w notacji naukowej (wykładniczej). Przechowywanie liczb rzeczywistych w postaci wykładniczej pozwala w znacznym stopniu rozszerzyć zakres reprezentacji liczb, umożliwia zapis bardzo dużych oraz bardzo małych wartości bezwzględnych przy zapewnieniu wystarczająco dobrej dokładności reprezentacji.

Ze względu na ograniczony rozmiar formatu zmiennoprzecinkowego bardzo często liczby rzeczywiste nie mogą być dokładnie zapamiętane w pamięci komputera. Liczby, które nie mają skończonego rozwinięcia dwójkowego lub zawierają zbyt dużo cyfr znaczących, są reprezentowane w formatach zmiennoprzecinkowych w postaci przybliżonej i sprowadzane do wartości najbliższej liczby maszynowej. Liczby maszynowe to jedyne dokładnie zapisywane liczby rzeczywiste, pozostałe muszą być wyrażane za pomocą liczb maszynowych. Zatem zapis zmiennoprzecinkowy zazwyczaj jest tylko przybliżony, a jedna liczba maszynowa może reprezentować różne liczby rzeczywiste z pewnego zakresu.

Cechą specyficzną arytmetyki zmiennoprzecinkowej jest niedokładność reprezentacji wyników obliczeń. W wielu przypadkach wyniki działań arytmetycznych wymagają zaokrąglenia, sprowadzenia do najbliższej liczby maszynowej, przez co uzyskany wynik może być tylko przybliżony. Z powodu błędów zaokrągleń w arytmetyce zmiennoprzecinkowej nie zawsze zachowywane są zasady łączności i przemienności działań, a kolejność wykonywania operacji przemiennych może mieć wpływ na wynik obliczeń.

W niniejszym skrypcie omówione zostały zasady zmiennoprzecinkowej reprezentacji liczb oraz algorytmy wykonywania operacji arytmetycznych na formatach zmiennoprzecinkowych. Szczegółowo wyjaśniono sposób przeliczania liczb rzeczywistych na zapis zmiennoprzecinkowy, przedstawiono standardowe formaty zmiennoprzecinkowe oraz reguły wykonywania działań arytmetycznych na tych formatach. Zagadnienia teoretyczne zilustrowane zostały dużą liczbą dokładnie skomentowanych przykładów oraz zadań opatrzonych szczegółowymi rozwiązaniami, które ułatwiają zrozumienie i opanowanie materiału.

Poszczególne rozdziały skryptu zostały przygotowane w taki sposób, aby po zapoznaniu się z podstawami teoretycznymi można było dokładnie przyswoić omawiane zagadnienia poprzez analizę licznych zadań z rozwiązaniami, a następnie sprawdzić swoje umiejętności, rozwiązując zadania wymagające samodzielnej pracy.

Rozdział drugi szczegółowo wyjaśnia sposób zapisu liczb rzeczywistych w formatach zmiennoprzecinkowych na przykładzie formatów niestandardowych, które mogą mieć różne rozmiary oraz dowolne sposoby kodowania poszczególnych komponentów liczb. Jedynym wymaganym założeniem jest przeliczenie liczby na zapis zmiennoprzecinkowy z dwójkowej postaci wykładniczej ze znormalizowaną mantysą. Rozdział zawiera liczne przykłady zapisu liczb do formatów o różnych rozmiarach i sposobach kodowania poszczególnych pól oraz przykłady rozkodowywania formatów zmiennoprzecinkowych.

Rozdział trzeci omawia standard zmiennoprzecinkowej reprezentacji liczb IEEE 754, w oparciu o który realizowane są obecnie wszystkie implementacje sprzętowe liczb zmiennoprzecinkowych. Standard ten został opracowany po to, by ujednolicić zapis liczb zmiennoprzecinkowych we wszystkich systemach obliczeniowych. Definiuje on standardowe formaty zmiennoprzecinkowe, sposoby reprezentacji wartości specjalnych oraz reguły wykonywania działań arytmetycznych na formatach tego typu. W tym rozdziale zaprezentowane zostały standardowe formaty zmiennoprzecinkowe, określono rozmiary poszczególnych pól, sposoby kodowania komponentów liczby, zakres i dokładność reprezentacji, a także formaty służące do przechowywania wartości specjalnych. Dodatkowo opisano formaty dziesiętnej reprezentacji zmiennoprzecinkowej wprowadzone w nowej, zmodyfikowanej wersji standardu IEEE 754-2008. Zapis liczb do dziesiętnego formatu zmiennoprzecinkowego wykonywany jest na podstawie dziesiętnej (a nie dwójkowej) postaci wykładniczej liczby rzeczywistej. Zasady zapisu liczb do dziesiętnych formatów zmiennoprzecinkowych zostały dokładnie wyjaśnione z wykorzystaniem licznych przykładów kodowania i dekodowania liczb.

Pozostałe rozdziały skryptu opisują algorytmy wykonywania podstawowych operacji arytmetycznych na liczbach przechowywanych w formatach zmiennoprzecinkowych. Z reguły sprowadzają się one do zasad wykonywania operacji na liczbach w postaci wykładniczej. Rozdział czwarty przedstawia algorytmy operacji dodawania oraz odejmowania liczb zmiennoprzecinkowych. Z kolei rozdział piąty wyjaśnia sposób realizacji algorytmów operacji mnożenia i dzielenia operandów zmiennoprzecinkowych. Rozdział szósty opisuje natomiast wybrane metody wyznaczania pierwiastka kwadratowego z liczby zmiennoprzecinkowej. Wszystkie operacje zostały wyjaśnione i zaprezentowane na czytelnych, dokładnie skomentowanych przykładach wykonywania działań obliczeniowych.

Skrypt przeznaczony jest dla studentów kierunków technicznych jako pomoc w nauce przedmiotów, które podejmują zagadnienia związane z arytmetyką komputerową, kodowaniem i reprezentacją liczb. Z uwagi na fakt, iż materiał zawiera wiele przykładów i zadań, książka może być przydatna do samodzielnego przygotowywania się do zajęć ćwiczeniowych oraz kolokwiów zaliczeniowych.

2. Zmiennoprzecinkowa reprezentacja liczb

2.1. Podstawy teoretyczne

Zmiennoprzecinkowa reprezentacja liczb oparta jest na zapisie liczby w postaci naukowej (wykładniczej). Notacja zmiennoprzecinkowa zawiera dwa podstawowe komponenty wykładniczego zapisu liczby: mantysę M (ang. *significand*) oraz wykładnik E (ang. *exponent*). Wartość liczby zmiennoprzecinkowej obliczana jest zgodnie ze wzorem:

$$F = M \cdot P^E,$$

gdzie P – podstawa systemu liczbowego.

Przy reprezentacji liczb w systemie komputerowym zarówno mantysa M, jak i wykładnik E przedstawiane są w systemie dwójkowym ($P = 2$), a liczba przeliczana jest na postać zmiennoprzecinkową z dwójkowego zapisu wykładniczego:

$$F = M \cdot 2^E.$$

2.1.1. Znormalizowana postać mantysy

W celu ujednolicenia zapisu liczb w formatach zmiennoprzecinkowych przyjęto *znormalizowaną* postać mantysy w reprezentacji zmiennoprzecinkowej. Mantysa w postaci dwójkowej znormalizowanej zawiera się w przedziale [1, 2).

Część całkowita znormalizowanej mantysy zawiera dokładnie jedną cyfrę różną od 0. Ponieważ w systemie dwójkowym taką cyfrą może być tylko 1, to znormalizowana mantysa zawsze ma postać 1,xxx…xx, gdzie x – dowolna cyfra binarna (0 lub 1).

Można zauważyć, że w postaci znormalizowanej nie da się przedstawić liczby 0, gdyż nie można zapisać zera z jedynką w części całkowitej. Z tego powodu liczba 0 w formatach zmiennoprzecinkowych jest reprezentowana jako wartość specjalna (reprezentacja wartości specjalnych omówiona zostanie w podrozdziale 3.1.2).

Aby przedstawić liczbę dziesiętną w dwójkowej postaci wykładniczej ze znormalizowaną mantysą, należy postępować zgodnie z następującym algorytmem:

1. liczbę dziesiętną zamienić na postać dwójkową przy wartości wykładnika $E = 0$;
2. znormalizować mantysę, przenosząc przecinek pozycyjny tak, aby w części całkowitej liczby występowała tylko jedna cyfra 1;

3. przeniesienie przecinka związane jest ze zmianą wartości wykładnika: przy przesunięciu przecinka o n pozycji w lewo zwiększyć wartość wykładnika o n, natomiast przy przesunięciu przecinka o n miejsc w prawo – zmniejszyć wykładnik o n.

Przykład 2.1

Przedstawić liczbę dziesiętną $112{,}625_{(10)}$ *w dwójkowej postaci wykładniczej ze znormalizowaną mantysą.*

Najpierw przeliczymy liczbę na system dwójkowy, zamieniając oddzielnie część całkowitą oraz część ułamkową liczby:

$$\begin{aligned} 112 : 2 &= 56 \quad & c_0 &= 0 \\ 56 : 2 &= 28 \quad & c_1 &= 0 \\ 28 : 2 &= 14 \quad & c_2 &= 0 \\ 14 : 2 &= 7 \quad & c_3 &= 0 \\ 7 : 2 &= 3 \quad & c_4 &= 1 \\ 3 : 2 &= 1 \quad & c_5 &= 1 \\ 1 : 2 &= 0 \quad & c_6 &= 1 \end{aligned}$$

Wynik zamiany części całkowitej:

$$112_{(10)} = 1110000_{(2)}$$

$$\begin{aligned} 0{,}625 \cdot 2 &= 1{,}25 \quad & c_{-1} &= 1 \\ 0{,}25 \cdot 2 &= 0{,}5 \quad & c_{-2} &= 0 \\ 0{,}5 \cdot 2 &= 1{,}0 \quad & c_{-3} &= 1 \end{aligned}$$

Wynik zamiany części ułamkowej:

$$0{,}625_{(10)} = 0{,}101_{(2)}$$

Z powyższych obliczeń otrzymujemy liczbę w postaci dwójkowej:

$$112{,}625_{(10)} = 1110000{,}101_{(2)}.$$

Następnie wykonujemy normalizację liczby i przedstawiamy ją w postaci wykładniczej:

$$1110000{,}101_{(2)} = 1{,}110000101 \cdot 2^6.$$

Podstawę systemu liczbowego (liczbę 2) oraz wykładnik podano tu w systemie dziesiętnym w celu zwiększenia czytelności zapisu. Z wyznaczonej postaci wykładniczej otrzymujemy mantysę $M = 1{,}110000101$ oraz wykładnik $E = 6$.

Przykład 2.2
Przedstawić liczbę $\frac{7}{256}_{(10)}$ *w dwójkowej postaci wykładniczej.*

Zamianę zaczynamy od przeliczenia podanej liczby na system dwójkowy:

$$\frac{7}{256} = \frac{7}{2^8} = 0{,}00000111_{(2)}.$$

Normalizacja liczby wiąże się z przeniesieniem przecinka pozycyjnego o 6 miejsc w prawo (przecinek umieszczany jest po pierwszej jedynce w zapisie liczby), w wyniku czego otrzymujemy postać wykładniczą liczby:

$$0{,}00000111_{(2)} = 1{,}11 \cdot 2^{-6},$$

gdzie mantysa $M = 1{,}11$ oraz wykładnik $E = -6$.

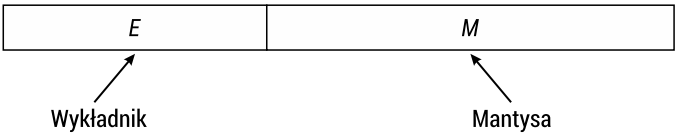
2.1.2. Konwersja liczby dziesiętnej na zapis zmiennoprzecinkowy

W formacie zmiennoprzecinkowym zapisuje się dwa komponenty dwójkowej wykładniczej postaci liczby – wykładnik oraz mantysę. Oba te komponenty są wartościami ze znakiem i można je zakodować za pomocą dowolnego kodu, który umożliwia kodowanie liczb ze znakiem (kod ZM, U2, kod z przesunięciem *bias*). Wykładnik w formacie zmiennoprzecinkowym jest liczbą całkowitą, natomiast mantysa przechowywana jest w postaci ułamka.

Znormalizowana mantysa w systemie dwójkowym zawsze zawiera w części całkowitej jedynkę, wobec tego przyjmuje się, żeby tej jedynki nie zapisywać do formatu zmiennoprzecinkowego. Zamiast tego jeden dodatkowy bit pola mantysy służy do zwiększenia dokładności reprezentacji liczby. Jedynka z części całkowitej mantysy przechowywana jest w sposób niejawny – jest zawsze dodawana do liczby przy jej odczytywaniu z formatu zmiennoprzecinkowego, lecz nie jest zapisywana do formatu w postaci jawnej. W polu mantysy w formacie zmiennoprzecinkowym przechowywana jest wyłącznie część ułamkowa mantysy.

Aby przedstawić liczbę dziesiętną w formacie zmiennoprzecinkowym, należy sprowadzić ją do dwójkowej postaci wykładniczej, a następnie mantysę oraz wykładnik zakodować, uwzględniając wybrane sposoby kodowania i rozmiary poszczególnych pól formatu.

Format zmiennoprzecinkowy może się składać z dwóch pól – pola wykładnika *W* oraz pola mantysy *M* (rys. 2.1). W takim wypadku znak liczby będzie przechowywany w starszym bicie pola mantysy, a na pozostałych bitach tego pola zapisuje się ułamkową część znormalizowanej mantysy. Wykładnik z kolei jest liczbą całkowitą ze znakiem.



RYS. 2.1. Format zmiennoprzecinkowy składający się z dwóch pól

Podział formatu zmiennoprzecinkowego na poszczególne pola wykonywany jest w taki sposób, aby zapewnić szeroki zakres oraz wystarczającą dokładność reprezentacji. Od rozmiaru pola wykładnika zależy zakres reprezentacji liczb, z kolei rozmiar bitowy pola mantysy wpływa na dokładność zapisu liczb w formacie zmiennoprzecinkowym. Przy zachowaniu stałego rozmiaru formatu zwiększenie rozmiaru pola wykładnika prowadzi do rozszerzenia zakresu reprezentacji liczb kosztem zmniejszenia dokładności i odwrotnie.

Przykład 2.3

Przedstawić liczbę dziesiętną $-37{,}75_{(10)}$ *w formacie zmiennoprzecinkowym składającym się z dwóch pól:* 6*-bitowego pola wykładnika oraz* 10*-bitowego pola mantysy. Wykładnik zakodować w kodzie z przesunięciem bias, a mantysę – w kodzie U*2.

Na początku sprowadzamy liczbę do dwójkowej postaci wykładniczej:

$$-37{,}75_{(10)} = -100101{,}11_{(2)} = -1{,}0010111 \cdot 2^5.$$

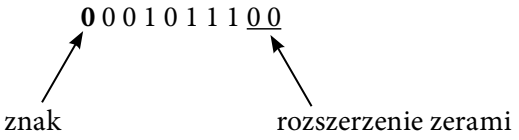
Z tej postaci otrzymujemy wartości mantysy $M = 1{,}0010111$ oraz wykładnika $E = 5$, które należy zakodować zgodnie z wybranym sposobem kodowania.

Wykładnik kodujemy w 6-bitowym kodzie z przesunięciem *bias*:

$$E = 5,\ bias = 2^{6-1} - 1 = 31,\ 5 + 31 = 36_{(10)} = 100100_{(\text{biased})}.$$

Mantysa jest zapisywana do formatu zmiennoprzecinkowego bez wiodącej jedynki, czyli w polu mantysy przechowywane są tylko cyfry z części ułamkowej liczby – 0010111.

W podanym formacie nie zostało wyróżnione specjalne pole do kodowania znaku, więc znak liczby umieszczony zostanie w starszym bicie pola mantysy. Oprócz tego należy rozszerzyć liczbę nieznaczącymi zerami do rozmiaru pola mantysy. W naszym przykładzie mantysa wraz ze znakiem zajmuje 10 bitów:



Dodatkowe zera rozszerzenia dopisywane są z prawej strony, ponieważ mantysa w formacie zmiennoprzecinkowym jest ułamkiem, a zatem dodatkowe zera z prawej strony nie zmieniają wartości liczby.

Ze względu na to, że liczba jest ujemna, mantysa będzie przechowywana w formacie zmiennoprzecinkowym w postaci uzupełnienia U2:

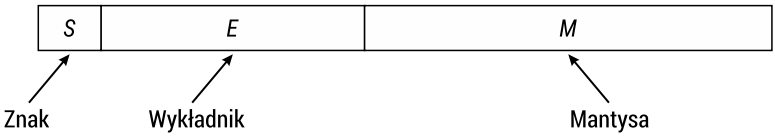
$$\mathbf{0}\ 0\,0\,1\,0\,1\,1\,1\,0\,0_{(2)} = \mathbf{1}\ 1\,1\,0\,1\,0\,0\,1\,0\,0_{(U2)},$$

przy czym starsza jedynka w tym kodzie określa znak liczby.

Ostatecznie otrzymujemy następujący format zmiennoprzecinkowy reprezentujący liczbę $-37{,}75_{(10)}$:

1 0 0 1 0 0	**1** 1 1 0 1 0 0 1 0 0

Znak liczby (znak mantysy) wygodnie jest przechowywać w oddzielnym polu – w starszym bicie formatu zmiennoprzecinkowego (rys. 2.2). Jest to najlepsze miejsce dla znaku, ponieważ łatwo go tam odnaleźć niezależnie od rozmiarów poszczególnych pól formatu. Dodatkowo ułatwia to porównywanie modułów liczb, które w tym wypadku sprowadza się do porównywania binarnej zawartości formatów (z wyłączeniem znaku) jako liczb całkowitych. Z tego powodu znak liczby w standardowych formatach zmiennoprzecinkowych zawsze jest przechowywany w najbardziej znaczącym bicie formatu.



RYS. 2.2. Format zmiennoprzecinkowy składający się z trzech pól

Przykład 2.4

Przedstawić liczbę dziesiętną $-\frac{17}{512}_{(10)}$ *w formacie zmiennoprzecinkowym składającym się z trzech pól:* 1*-bitowego pola znaku,* 5*-bitowego pola wykładnika oraz* 12*-bitowego pola mantysy. Wykładnik zakodować w kodzie z przesunięciem bias, a mantysę – w kodzie ZM.*

Zapis liczby do formatu zmiennoprzecinkowego zawsze zaczynamy od sprowadzenia liczby do dwójkowej postaci wykładniczej ze znormalizowaną mantysą:

$$-\frac{17}{512}_{(10)} = -\frac{17}{2^9}_{(10)} = -0{,}000010001_{(2)} = -1{,}0001 \cdot 2^{-5},$$

z której otrzymujemy mantysę $M = 1{,}0001$ oraz wykładnik $E = -5$.

Zgodnie z przyjętym sposobem kodowania zapiszemy wykładnik liczby w 5-bitowym kodzie U2:

$$E = -5_{(10)} = -00101_{(2)} = 11011_{(U2)}.$$

Znak liczby jest przechowywany w starszym bicie formatu – jedynka w bicie znaku oznacza liczbę ujemną.

Ułamkową część mantysy (bez jedynki z części całkowitej), rozszerzoną nieznaczącymi zerami zapisujemy w postaci modułu w 12-bitowym polu mantysy:

0 0 0 1 0 0 0 0 0 0 0 0.

Otrzymujemy następującą zawartość formatu zmiennoprzecinkowego:

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

2.1.3. Podstawowe właściwości formatów zmiennoprzecinkowych

Cechą charakterystyczną formatów zmiennoprzecinkowych jest zapewnienie szerokiego zakresu reprezentacji liczb. Zapis liczby w postaci wykładniczej pozwala w formatach zmiennoprzecinkowych niewielkiego rozmiaru przechowywać liczby o bardzo dużych i bardzo małych wartościach bezwzględnych.

Zakres reprezentacji liczb zmiennoprzecinkowych zależy od rozmiaru pola wykładnika, czyli jest on ograniczony przede wszystkim minimalną i maksymalną wartością wykładnika, którą można przedstawić w formacie zmiennoprzecinkowym. Zakres reprezentacji liczb zmiennoprzecinkowych obejmuje trzy podzakresy (rys. 2.3):

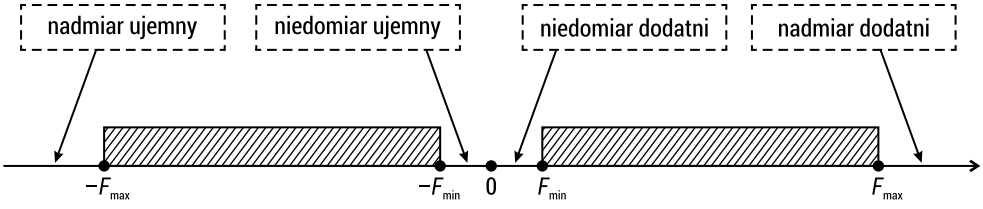
$$[-F_{max}, -F_{min}] \cup \{0\} \cup [F_{min}, F_{max}],$$

gdzie $[-F_{max}, -F_{min}]$ – podzakres liczb ujemnych; $\{0\}$ – zero przechowywane jako wartość specjalna; $[F_{min}, F_{max}]$ – podzakres liczb dodatnich; F_{min} – minimalna wartość bezwzględna reprezentowana w formacie zmiennoprzecinkowym; F_{max} – maksymalna wartość bezwzględna.

Wartości F_{min} oraz F_{max} można opisać za pomocą następujących wzorów:

$$F_{min} = M_{min} \cdot 2^{Emin}, F_{max} = M_{max} \cdot 2^{Emax},$$

gdzie M_{min}, M_{max} – minimalna oraz maksymalna wartość mantysy; E_{min}, E_{max} – odpowiednio minimalna i maksymalna wartość wykładnika.



RYS. 2.3. Zakres reprezentacji liczb zmiennoprzecinkowych

Ponieważ znormalizowana mantysa zawiera się w niewielkim przedziale [1,2), nie wpływa zauważalnie na zakres reprezentacji liczb. Decydującą rolę odgrywają tu minimalna E_{min} i maksymalna E_{min} wartości wykładnika.

Jeśli w wyniku wykonania działań na liczbach zmiennoprzecinkowych otrzymana zostanie liczba zbyt duża lub zbyt mała (wykładnik liczby wychodzi poza zakres reprezentacji $[E_{min}, E_{max}]$), mamy do czynienia z błędem zakresu, który postrzegany jest jako nadmiar lub niedomiar.

Zbyt duża wartość wykładnika powoduje wystąpienie błędu nadmiaru (nadmiar dodatni lub ujemny), w takim wypadku wynik obliczeń przedstawiany jest w formacie zmiennoprzecinkowym w postaci nieskończoności (podrozdział 3.1.2). W sytuacji gdy wynik obliczeń jest liczbą zbyt małą (wykładnik jest mniejszy od E_{min}), mamy do czynienia z błędem niedomiaru (dodatni lub ujemny), a wynik obliczeń w tym wypadku aproksymowany jest przez zero. Przy realizacji algorytmów wykonania działań na liczbach zmiennoprzecinkowych operacje na wykładnikach zawsze są sprawdzane w celu wykrycia błędów nadmiaru czy niedomiaru.

Ze względu na ograniczony rozmiar formatu nie jest możliwe przedstawienie każdej liczby rzeczywistej (wymiernej) w formacie zmiennoprzecinkowym, nawet jeśli znajduje się ona w zakresie reprezentacji dla danego formatu. Nie zawsze można precyzyjnie, dokładnie odwzorować liczbę na zapis zmiennoprzecinkowy, co wynika przede wszystkim z ograniczonego rozmiaru pola mantysy.

Przybliżona reprezentacja liczb jest cechą charakterystyczną formatów zmiennoprzecinkowych. Z tego powodu wprowadzone zostało pojęcie liczby *maszynowej*. Jest to liczba, która reprezentuje liczbę rzeczywistą w formacie zmiennoprzecinkowym, dopasowana jest do rozmiaru formatu, może być zapisana w postaci przybliżonej, na jaką ten format pozwala. Jedna liczba maszynowa reprezentuje różne liczby rzeczywiste z pewnego zakresu. Ze względu na ograniczony rozmiar formatu tylko skończony podzbiór liczb rzeczywistych może być reprezentowany w formacie zmiennoprzecinkowym, a reprezentacja liczby rzeczywistej bardzo często jest przybliżona.

Dokładność reprezentacji liczb w formatach zmiennoprzecinkowych zależy od rozmiaru pola mantysy. Nawet sam zapis liczby do formatu może spowodować utratę dokładności, a wyniki operacji wykonywanych na liczbach zmiennoprzecinkowych są obarczone błędem niedokładności, bardzo często wymagają przybliżenia, sprowadzenia do postaci najbliższej liczby maszynowej.

Istnieją różne schematy zaokrąglania liczb przy ich zapisie do formatów zmiennoprzecinkowych. Przedstawimy dwie podstawowe metody zaokrąglania:

1. zaokrąglanie do najbliższej liczby maszynowej;
2. zaokrąglanie w kierunku zera.

Zaokrąglanie *do najbliższej liczby maszynowej* realizowane jest przez dodanie 1 do pierwszej cyfry dwójkowej, która nie mieści się w polu mantysy formatu zmiennoprzecinkowego. Innymi słowy, w celu zaokrąglenia liczby należy dodać do niej

połowę wartości najmłodszego bitu pola mantysy. Błąd względny zaokrąglenia liczby w tym wypadku wynosi:

$$|\delta| \le \frac{1}{2^{m-1}},$$

gdzie m – rozmiar bitowy pola mantysy w formacie zmiennoprzecinkowym.

Z powyższego wzoru można wywnioskować, że sposobem na zwiększenie dokładności zapisu liczby w formacie zmiennoprzecinkowym jest zwiększenie rozmiaru m pola mantysy.

Schemat zaokrąglania *w kierunku zera* polega na obcięciu części liczby, która nie mieści się w polu mantysy. Nadmiarowe bity są odrzucane.

Przykład 2.5

Przedstawić liczbę dziesiętną $45\frac{15}{32}_{(10)}$ *w formacie zmiennoprzecinkowym składającym się z trzech pól: bitu znaku, 5-bitowego pola wykładnika w kodzie z przesunięciem bias oraz 7-bitowego pola mantysy w kodzie ZM. Zaokrąglić liczbę za pomocą metody: a) zaokrąglenia do najbliższej liczby maszynowej; b) zaokrąglenia w kierunku zera.*

Zapiszemy liczbę w dwójkowej postaci wykładniczej:

$$45\frac{15}{32}_{(10)} = 101101{,}01111_{(2)} = 1{,}0110101111 \cdot 2^5.$$

Wykładnik $E = 5$ w 5-bitowym kodzie z przesunięciem $bias = 2^{5-1}-1 = 15$ zostanie zapisany do formatu zmiennoprzecinkowego w następującej postaci:

$$5 + 15 = 20_{(10)} = 10100_{(biased)}.$$

Podanej liczby nie da się dokładnie przedstawić w formacie zmiennoprzecinkowym z 7-bitowym polem mantysy, więc wykonywane jest zaokrąglenie liczby do 7 cyfr po przecinku.

a) Zaokrąglenie do najbliższej liczby maszynowej polega na dodaniu jedynki do pierwszego bitu mantysy, który nie mieści się w 7-bitowym polu mantysy, czyli do 8 bitu po przecinku, po czym ten bit zostaje odrzucony:

$$\begin{array}{rl|l} & 1,0\,1\,1\,0\,1\,0\,1 & 1\,1\,1 \\ + & & 1 \\ \hline & 1,0\,1\,1\,0\,1\,1\,0 & 0 \end{array}$$

Liczba zostanie zapisana do formatu zmiennoprzecinkowego w postaci przybliżonej:

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

b) Zaokrąglenie w kierunku zera polega na odrzuceniu tych bitów liczby, które nie mieszczą się w 7-bitowym polu mantysy:

$$1,0\,1\,1\,0\,1\,0\,1 \,|\, \sout{1\,1\,1}$$

W takim wypadku format zmiennoprzecinkowy liczby przedstawia się następująco:

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

Dokładność reprezentacji liczb w formatach zmiennoprzecinkowych zwykle jest wyrażana liczbą cyfr dziesiętnych zapamiętywanych w formacie zmiennoprzecinkowym i obliczana jest za pomocą następującego wzoru:

$$d = \frac{m}{\log_2 10} \tag{2.1}$$

gdzie m – liczba bitów pola mantysy w formacie zmiennoprzecinkowym.

Przykład 2.5

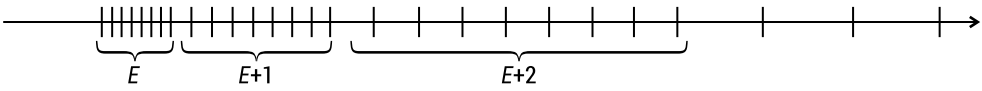
Obliczyć dokładność reprezentacji liczb w formacie zmiennoprzecinkowym, w którym pole mantysy ma rozmiar m = 24 bity.

Zgodnie ze wzorem (2.1) otrzymujemy:

$$d = \frac{24}{\log_2 10} \approx \frac{24}{3{,}321928} \approx 7{,}2247.$$

Dokładność reprezentacji liczb w podanym formacie zmiennoprzecinkowym wynosi 7 cyfr dziesiętnych w zapisie liczby. Oznacza to, że po zapisie, a następnie odczycie liczby z formatu zmiennoprzecinkowego zostanie dokładnie odtworzonych 7 cyfr dziesiętnych liczby. Ostatnia, siódma cyfra może jednak ulec zmianie w wyniku zaokrąglenia liczby.

Liczby reprezentowane w formatach zmiennoprzecinkowych nie są równomiernie rozmieszczone na osi liczbowej (rys. 2.4): dla mniejszych wartości wykładnika są umieszczone gęściej, a przy zwiększeniu wykładnika – coraz rzadziej.



RYS. 2.4. Rozmieszczenie liczb zmiennoprzecinkowych na osi liczbowej

W każdym przedziale (dla stałej wartości wykładnika) znajduje się tyle równomiernie rozłożonych liczb maszynowych, na ile kombinacji pozwala rozmiar pola mantysy. W takim układzie dla mniejszych wartości bezwzględnych wykładnika dokładność reprezentacji liczb jest większa, natomiast obejmowany zakres – mniejszy. Przy wzroście wartości wykładnika obejmowany jest coraz większy zakres, lecz dokładność reprezentacji liczb maleje (jedna liczba maszynowa może tu reprezentować różne liczby rzeczywiste z szerszego zakresu).

2.1.4. Wyznaczanie liczby dziesiętnej z zapisu zmiennoprzecinkowego

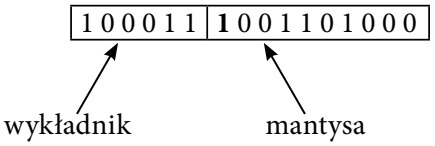
Przeliczenie formatu zmiennoprzecinkowego na liczbę dziesiętną zaczynamy od podziału formatu na odpowiednie pola oraz rozkodowania znaku, wykładnika i mantysy liczby. Na podstawie rozkodowanych komponentów wyznaczamy dwójkową wykładniczą postać liczby, którą następnie przeliczymy na system dziesiętny.

Przy odczycie liczby z formatu zmiennoprzecinkowego należy pamiętać o tym, że jedynka z części całkowitej mantysy nie jest przechowywana w formacie w postaci jawnej. Oznacza to, że tę jedynkę zawsze należy dopisać do odczytanej z formatu ułamkowej części mantysy liczby.

Przykład 2.6

Format zmiennoprzecinkowy składa się z dwóch pól: 6*-bitowego wykładnika zakodowanego w kodzie z przesunięciem bias oraz* 10*-bitowej mantysy* (*wraz ze znakiem*) *w kodzie U*2. *Dwójkowa zawartość formatu wynosi* $1000111001101000_{(2)}$. *Obliczyć liczbę dziesiętną zapisaną w podanym formacie zmiennoprzecinkowym.*

Zgodnie z opisem formatu podzielimy jego zawartość na dwa pola:



Aby rozkodować wykładnik, należy od zawartości pola wykładnika odjąć przesunięcie *bias*, które dla 6-bitowego formatu wynosi $bias = 2^{6-1}-1 = 31_{(10)}$:

$$E = 100011_{(\text{biased})} = 35_{(10)} - 31_{(10)} = 4_{(10)}.$$

Mantysa zapisana jest w podanym formacie wraz ze znakiem w kodzie U2. Starszy bit pola mantysy określa, że liczba jest ujemna, a to oznacza, że jest przechowywana w postaci uzupełnienia U2. Obliczając ponownie uzupełnienie U2 od tego kodu, wyznaczamy moduł mantysy:

$$\mathbf{1}001101000_{(U2)} = \mathbf{0}110011000_{(2)}.$$

Starszy bit w otrzymanym słowie jest bitem znaku, który po prostu odrzucamy. Pozostałe bity tworzą część ułamkową mantysy, do której należy dopisać 1 w części całkowitej, aby otrzymać znormalizowaną postać mantysy:

$$-1{,}110011000_{(2)}.$$

Z rozkodowanych składowych formatu zapisujemy dwójkową postać wykładniczą liczby zmiennoprzecinkowej:

$$-1{,}110011 \cdot 2^4.$$

Mnożenie liczby dwójkowej przez 2^n odpowiada przeniesieniu przecinka pozycyjnego o *n* bitów w dwójkowym zapisie liczby. Po takich przekształceniach i konwersji dwójkowo-dziesiętnej wyznaczamy dziesiętną postać liczby:

$$-1{,}110011 \cdot 2^4 = -11100{,}11_{(2)} = -28{,}75_{(10)}.$$

Przykład 2.7

Format zmiennoprzecinkowy składa się z trzech pól: bitu znaku, 7-bitowego pola wykładnika w kodzie U2 i 6-bitowego pola mantysy w kodzie ZM. Dwójkowa zawartość formatu wynosi $11111100100100_{(2)}$. *Obliczyć dziesiętną wartość liczby zapisanej w podanym formacie.*

Konwersję zaczynamy od podziału formatu zmiennoprzecinkowego na poszczególne pola:

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

Jedynka w starszym bicie formatu informuje o tym, że liczba jest ujemna. Pole wykładnika po rozkodowaniu z kodu U2 przyjmuje następującą wartość:

$$E = 1111100_{(U2)} = -0000100_{(2)} = -4_{(10)}.$$

Mantysa jest przechowywana w postaci modułu, więc do jej zawartości pola wystarczy dopisać jedynkę w części całkowitej, aby otrzymać znormalizowaną mantysę: 1,100100. Następnie z utworzonej dwójkowej postaci wykładniczej liczby wyznaczamy jej wartość dziesiętną:

$$-1{,}100100 \cdot 2^{-4} = -0{,}00011001_{(2)} = -\frac{25}{256}{}_{(10)}.$$

2.2. Zadania z rozwiązaniami

Zadanie 2.1

Liczbę dziesiętną $-56\frac{39}{64}{}_{(10)}$ *przedstawić w formacie zmiennoprzecinkowym składającym się z dwóch pól* (5*-bitowego wykładnika,* 14*-bitowej mantysy*)*, wykorzystując dwa sposoby kodowania: a*) *wykładnik – w kodzie z przesunięciem bias, mantysa –* w kodzie U2*; b*) *wykładnik – w kodzie U2, mantysa – w kodzie ZM.*

Przeliczymy liczbę $-56\frac{39}{64}$ na system dwójkowy i sprowadzimy ją do postaci wykładniczej ze znormalizowaną mantysą:

$$-56\frac{39}{64}_{(10)} = -111000{,}100111_{(2)} = -1{,}11000100111 \cdot 2^5.$$

Komponenty wykładniczego zapisu liczby zakodujemy zgodnie z podanym sposobem kodowania.

a) Wykładnik kodujemy, zwiększając go o przesunięcie *bias* = $2^{5-1}-1 = 15_{(10)}$:

$$E = 5 + 15 = 20_{(10)} = 10100_{(\text{biased})}.$$

W formacie składającym się z dwóch pól znak liczby jest przechowywany w starszym bicie pola mantysy. Ułamkową część mantysy rozszerzamy z prawej strony nieznaczącymi zerami (są podkreślone w liczbie) tak, aby wraz ze znakiem zajmowała 14-bitowe pole. Jedynka z części całkowitej nie jest zapisywana do formatu, zamiast tego w starszym bicie pola mantysy będzie przechowywany znak liczby:

$$0\,1\,1\,0\,0\,0\,1\,0\,0\,1\,1\,1\,\underline{0\,0}_{(2)}.$$

Ponieważ liczba jest ujemna, mantysa w formacie zmiennoprzecinkowym zostanie przedstawiona w postaci uzupełnienia U2:

$$1\,0\,0\,1\,1\,1\,0\,1\,1\,0\,0\,1\,\underline{0\,0}_{(2)}.$$

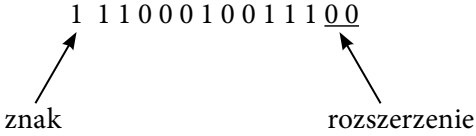
Ostatecznie format zmiennoprzecinkowy ma następującą postać:

1 0 1 0 0	**1** 0 0 1 1 1 0 1 1 0 0 1 0 0

b) Wykładnik jest liczbą dodatnią, więc w kodzie U2 zapisywany jest w zwykłej postaci binarnej:

$$E = 5_{(10)} = 00101_{(2)} = 00101_{(\text{U2})}.$$

W polu mantysy w formacie zmiennoprzecinkowym umieszczona zostanie ułamkowa część mantysy rozszerzona nieznaczącymi zerami z prawej strony, zakodowana w kodzie ZM:



Format zmiennoprzecinkowy przy takim sposobie kodowania ma następującą postać:

0 0 1 0 1	**1** 1 1 0 0 0 1 0 0 1 1 1 0 0

Zadanie 2.2

W formacie zmiennoprzecinkowym składającym się z trzech pól (*bitu znaku,* 6*-bitowego wykładnika w kodzie z przesunięciem bias,* 13*-bitowej mantysy w kodzie ZM*), *zapisać liczbę dziesiętną* $-1\frac{69}{512}_{(10)}$. *Podać szesnastkową reprezentację zawartości* formatu zmiennoprzecinkowego.

Liczba jest przeliczana do formatu zmiennoprzecinkowego z dwójkowej postaci wykładniczej:

$$-1\frac{69}{512} = -1{,}001000101_{(2)} = -1{,}001000101 \cdot 2^0.$$

Wykładnik w kodzie z przesunięciem *bias* ma następującą postać:

$$E = 0 + 31 = 31_{(10)} = 011111_{(\text{biased})}.$$

Ułamkową część mantysy należy rozszerzyć zerami z prawej strony do 14-bitowego rozmiaru pola. W wyniku otrzymujemy następującą zawartość formatu zmiennoprzecinkowego:

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

Wykonując podział dwójkowego zapisu formatu zmiennoprzecinkowego na 4-bitowe grupy cyfr i zastępując je cyframi szesnastkowymi, otrzymamy szesnastkowy zapis zawartości formatu $BE450_{(16)}$.

Zadanie 2.3

Liczbę dziesiętną $0{,}2_{(10)}$ *przedstawić w formacie zmiennoprzecinkowym składającym się z trzech pól: bitu znaku,* 4*-bitowego wykładnika w kodzie z przesunięciem bias,* 10*-bitowej mantysy w kodzie ZM. Przybliżenie liczby do rozmiaru formatu wykonać za pomocą metody zaokrąglenia w kierunku zera.*

Sprowadzamy liczbę do dwójkowego zapisu wykładniczego, zaczynając najpierw od zamiany liczby na system dwójkowy:

$$\begin{aligned}
0{,}2 \cdot 2 &= 0{,}4 & c_{-1} &= 0\\
0{,}4 \cdot 2 &= 0{,}8 & c_{-2} &= 0\\
0{,}8 \cdot 2 &= 1{,}6 & c_{-3} &= 1\\
0{,}6 \cdot 2 &= 1{,}2 & c_{-4} &= 1\\
0{,}2 \cdot 2 &= 0{,}4 & c_{-5} &= 0
\end{aligned}$$

$$0{,}2_{(10)} = 0{,}(0011)_{(2)} = 0{,}001100110011... = 1{,}(1001) \cdot 2^{-3}.$$

W ten sposób widzimy, że liczba $0{,}2_{(10)}$ ma okresowe rozwinięcie dwójkowe i nie może być dokładnie reprezentowana w formacie zmiennoprzecinkowym. Zgodnie z wybraną metodą zaokrąglania (w kierunku zera) zapisujemy do formatu zmiennoprzecinkowego tyle cyfr dwójkowych, ile zmieści się w polu mantysy, a pozostałe cyfry odrzucamy:

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

Zadanie 2.4

W formacie zmiennoprzecinkowym składającym się z trzech pól (*bitu znaku,* 5*-bitowego pola wykładnika w kodzie z przesunięciem bias,* 10*-bitowej mantysy w kodzie ZM*) *przedstawić liczbę dziesiętną* $-22{,}43_{(10)}$. *Zaokrąglić wynik do najbliższej liczby maszynowej. Przedstawić zawartości formatu zmiennoprzecinkowego w postaci szesnastkowej.*

Dokonamy konwersji liczby dziesiętnej na system dwójkowy:

$$\begin{aligned}
0{,}43 \cdot 2 &= 0{,}86 & c_{-1} &= 0\\
0{,}86 \cdot 2 &= 1{,}72 & c_{-2} &= 1\\
0{,}72 \cdot 2 &= 1{,}44 & c_{-3} &= 1\\
0{,}44 \cdot 2 &= 0{,}88 & c_{-4} &= 0\\
0{,}88 \cdot 2 &= 1{,}76 & c_{-5} &= 1\\
0{,}76 \cdot 2 &= 1{,}52 & c_{-6} &= 1\\
0{,}52 \cdot 2 &= 1{,}04 & c_{-7} &= 1\\
0{,}04 \cdot 2 &= 0{,}08 & c_{-8} &= 0
\end{aligned}$$

$$0{,}43_{(10)} \approx 0{,}01101110_{(2)}$$

$$-22{,}43_{(10)} \approx -10110{,}01101110_{(2)}.$$

Część ułamkowa liczby nie ma dokładnego rozwinięcia dwójkowego, więc będzie przedstawiona w formacie zmiennoprzecinkowym w postaci przybliżonej. W tym celu obliczamy tyle cyfr dwójkowych po przecinku, ile zmieści się w polu mantysy formatu zmiennoprzecinkowego oraz dodatkową cyfrę potrzebną do zaokrąglenia.

Zapiszemy liczbę dwójkową w postaci znormalizowanej:

$$-10110{,}01101110_{(2)} = -1{,}011001101110 \cdot 2^4.$$

W polu mantysy formatu zmiennoprzecinkowego zmieści się jedynie 10 cyfr dwójkowych z ułamkowej części mantysy. Dodając 1 do jedenastej cyfry po przecinku, zaokrąglamy mantysę do najbliższej liczby maszynowej:

$$\begin{array}{rl|l}
 & 1,0\,1\,1\,0\,0\,1\,1 & 0\,1\,1 & 1\,0\\
+ & & & 1\\
\hline
 & 1,0\,1\,1\,0\,0\,1\,1 & 1\,0\,0 & 0
\end{array}$$

Po zakodowaniu wykładnika liczby $E = 4_{(10)} = 10011_{(biased)}$ otrzymamy następujący format zmiennoprzecinkowy:

1	10011	0110011100

Zawartość formatu w postaci szesnastkowej wynosi $CD9C_{(16)}$.

Zadanie 2.5

Wyznaczyć wartość dziesiętną liczby zapisanej w formacie zmiennoprzecinkowym, który składa się z dwóch pól: 7-bitowego wykładnika w kodzie z przesunięciem bias oraz 6-bitowej mantysy w kodzie U2. Dwójkowa zawartość formatu wynosi $0111101101100_{(2)}$.

Rozkodujemy zawartości poszczególnych pól formatu:

0111101	101100

Wykładnik liczby otrzymamy w wyniku odejmowania przesunięcia *bias* od zawartości pola wykładnika:

$$bias = 2^{7-1}-1 = 63_{(10)}$$

$$E = 61-63 = -2_{(10)}.$$

Mantysa jest zapisana w formacie zmiennoprzecinkowym w kodzie U2 wraz ze znakiem. W celu wyznaczenia modułu ułamkowej części mantysy wystarczy obliczyć uzupełnienie U2 od zawartości pola mantysy oraz odrzucić bit znaku.

$$1\ 01100_{(U2)} = -\ 0\ 10100_{(NKB)}$$

znak — bit znaku

Warto tu przypomnieć, że jedynka z części całkowitej liczby nie jest przechowywana w formacie zmiennoprzecinkowym i należy ją dopisać do rozkodowanej mantysy w miejscu bitu znaku. W taki sposób otrzymujemy dwójkową postać wykładniczą liczby, którą przeliczamy na system dziesiętny:

$$-1{,}10100 \cdot 2^{-2} = -0{,}01101_{(2)} = -\frac{13}{32}_{(10)}.$$

Zadanie 2.6

Format zmiennoprzecinkowy składa się z trzech pól: bitu znaku, 4-bitowego wykładnika w kodzie z przesunięciem bias oraz 11-bitowej mantysy w kodzie ZM. Wyznaczyć wartość dziesiętną liczby zapisanej w takim formacie, jeśli reprezentacja szesnastkowa zawartości formatu wynosi $6E58_{(16)}$.

Przedstawimy format w postaci dwójkowej i podzielimy go na odpowiednie pola:

$$6E58_{(16)} = 0110\ 1110\ 0101\ 1000_{(2)}$$

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

Po rozkodowaniu składowych zapisu zmiennoprzecinkowego wyznaczamy postać wykładniczą liczby, którą przeliczamy na wartość dziesiętną. Warto tu znów przypomnieć o dopisaniu jedynki do części całkowitej liczby, gdyż jedynka ta nie jest przechowywana w formacie zmiennoprzecinkowym.

$$E = 1101_{(\text{biased})} = 13-7 = 6_{(10)}$$

$$1{,}11001011 \cdot 2^6 = 1110010{,}11_{(2)} = 114{,}75_{(10)}$$

2.3. Zadania do samodzielnego rozwiązania

Zadanie 2.7

Format zmiennoprzecinkowy składa się z dwóch pól: wykładnika zakodowanego w kodzie z przesunięciem bias, mantysy w kodzie U2 (*rozmiary pól są podane w poszczególnych zadaniach*). *Przedstawić w formacie zmiennoprzecinkowym następujące liczby:*

a) $-47{,}40625_{(10)}$, 5-bitowy wykładnik, 14-bitowa mantysa;

b) $0{,}1171875_{(10)}$, 6-bitowy wykładnik, 10-bitowa mantysa;

c) $-\frac{69}{1024}_{(10)}$, 7-bitowy wykładnik, 13-bitowa mantysa;

d) $85\frac{19}{32}_{(10)}$, 4-bitowy wykładnik, 15-bitowa mantysa.

Zadanie 2.8

Format zmiennoprzecinkowy składa się z dwóch pól: wykładnika zakodowanego w kodzie U2, mantysy w kodzie ZM (*rozmiary pól są podane w poszczególnych zadaniach*). *Przedstawić w formacie zmiennoprzecinkowym następujące liczby:*

a) $-0{,}7890625_{(10)}$, 7-bitowy wykładnik, 12-bitowa mantysa;

b) $135{,}875_{(10)}$, 5-bitowy wykładnik, 14-bitowa mantysa;

c) $-14\frac{13}{128}_{(10)}$, 6-bitowy wykładnik, 16-bitowa mantysa;

d) $\frac{215}{512}_{(10)}$, 4-bitowy wykładnik, 11-bitowa mantysa.

Zadanie 2.9

Format zmiennoprzecinkowy składa się z trzech pól: bitu znaku, wykładnika zakodowanego w kodzie z przesunięciem bias, mantysy w kodzie ZM (*rozmiary pól są podane w poszczególnych zadaniach*). *Przedstawić w formacie zmiennoprzecinkowym następujące liczby:*

a) $0{,}796875_{(10)}$, 4-bitowy wykładnik, 10-bitowa mantysa;

b) $-519{,}1640625_{(10)}$, 6-bitowy wykładnik, 16-bitowa mantysa;

c) $\frac{175}{512}_{(10)}$, 8-bitowy wykładnik, 8-bitowa mantysa;

d) $-30_{(10)}$, 5-bitowy wykładnik, 10-bitowa mantysa;

e) $419\frac{3}{8}_{(10)}$, 6-bitowy wykładnik, 15-bitowa mantysa;

f) $-1\frac{79}{256}_{(10)}$, 7-bitowy wykładnik, 12-bitowa mantysa.

Zadanie 2.10

Format zmiennoprzecinkowy składa się z trzech pól: bitu znaku, wykładnika zakodowanego w kodzie z przesunięciem bias, mantysy w kodzie ZM (*rozmiary pól są podane w poszczególnych zadaniach*). *Przedstawić w formacie zmiennoprzecinkowym następujące liczby* (*liczby zaokrąglić za pomocą metody zaokrąglenia w kierunku zera*):

a) $0{,}53_{(10)}$, 4-bitowy wykładnik, 8-bitowa mantysa;

b) $-5{,}214_{(10)}$, 5-bitowy wykładnik, 9-bitowa mantysa;

c) $2{,}6_{(10)}$, 6-bitowy wykładnik, 16-bitowa mantysa;

d) $-411\frac{37}{512}_{(10)}$, 7-bitowy wykładnik, 11-bitowa mantysa.

Zadanie 2.11

Format zmiennoprzecinkowy składa się z trzech pól: bitu znaku, wykładnika zakodowanego w kodzie z przesunięciem bias, mantysy w kodzie ZM (*rozmiary pól są podane w poszczególnych zadaniach*). *Przedstawić w formacie zmiennoprzecinkowym następujące liczby* (*liczby zaokrąglić za pomocą metody zaokrąglenia do najbliższej liczby maszynowej*):

a) $-10{,}8_{(10)}$, 6-bitowy wykładnik, 12-bitowa mantysa;

b) $0{,}324_{(10)}$, 4-bitowy wykładnik, 9-bitowa mantysa;

c) $-537\frac{43}{128}_{(10)}$, 7-bitowy wykładnik, 12-bitowa mantysa;

d) $15{,}27_{(10)}$, 5-bitowy wykładnik, 13-bitowa mantysa.

Zadanie 2.12

Format zmiennoprzecinkowy składa się z dwóch pól: wykładnika zakodowanego w kodzie z przesunięciem bias, mantysy w kodzie U2 (*rozmiary pól są podane w poszczególnych zadaniach*). *Wyznaczyć dziesiętną wartość liczby zapisanej w formacie zmiennoprzecinkowym, jeśli szesnastkowa reprezentacja zawartości formatu wynosi:*

a) $9358_{(16)}$, 6-bitowy wykładnik, 10-bitowa mantysa;

b) $56A0_{(16)}$, 4-bitowy wykładnik, 12-bitowa mantysa;

c) $B498_{(16)}$, 5-bitowy wykładnik, 11-bitowa mantysa;

d) $7D34_{(16)}$, 7-bitowy wykładnik, 9-bitowa mantysa.

Zadanie 2.13

Format zmiennoprzecinkowy składa się z trzech pól: bitu znaku, wykładnika zakodowanego w kodzie z przesunięciem bias, mantysy w kodzie ZM (*rozmiary pól są podane w poszczególnych zadaniach*). *Wyznaczyć dziesiętną wartość liczby zapisanej w formacie zmiennoprzecinkowym, jeśli szesnastkowa reprezentacja zawartości formatu wynosi:*

a) $C96C0_{(16)}$, 6-bitowy wykładnik, 13-bitowa mantysa;

b) $5C00_{(16)}$, 5-bitowy wykładnik, 10-bitowa mantysa;

c) $A6E0_{(16)}$, 4-bitowy wykładnik, 11-bitowa mantysa;

d) $3F9B0_{(16)}$, 8-bitowy wykładnik, 11-bitowa mantysa;

e) $C79DA_{(16)}$, 7-bitowy wykładnik, 12-bitowa mantysa;

f) $B9B0_{(16)}$, 6-bitowy wykładnik, 9-bitowa mantysa.

3. Standard IEEE 754

3.1. Podstawy teoretyczne

Standard IEEE 754 opracowany został w celu ujednolicenia sposobu zapisu liczb w formatach zmiennoprzecinkowych we wszystkich systemach komputerowych. Standard definiuje zasady reprezentacji liczb w formatach zmiennoprzecinkowych, opisuje dostępne formaty, sposób reprezentacji wartości specjalnych, zasady wykonywania operacji na liczbach zmiennoprzecinkowych, w tym reguły zaokrąglania liczb maszynowych.

3.1.1. Standardowe formaty zmiennoprzecinkowe

Zgodnie ze standardem IEEE 754 liczba jest przechowywana w formacie zmiennoprzecinkowym na trzech polach: bicie znaku *S*, polu wykładnika *E* oraz polu mantysy *M*. Wartość liczby zapisanej w formacie zmiennoprzecinkowym wyznaczana jest z następującego wzoru:

$$L = (-1)^{S} \cdot M \cdot 2^{E-bias}.$$

Liczba jest zapisywana do formatu zmiennoprzecinkowego z dwójkowej postaci wykładniczej ze znormalizowaną mantysą (mantysa ma postać 1,xxx…xx i zawiera się w przedziale [1, 2)). Mantysa jest przechowywana w formacie standardowym bez wiodącej jedynki w części całkowitej, czyli występuje jako ułamek. Do kodowania mantysy wybrano kod ZM, oznacza to, że liczba jest przedstawiana w postaci modułu zakodowanego dwójkowo, przy czym znak przechowywany jest oddzielnie od liczby – w starszym bicie formatu.

Wykładnik liczby zmiennoprzecinkowej w formatach standardowych kodowany jest w kodzie z przesunięciem *bias*. Kod ten został wybrany ze względu na jego unikalną właściwość, a mianowicie monotoniczność zapisu – w zakresie zwiększających się wartości kodu (od 00000…00 do 11111…11) występuje monotoniczny zakres wartości liczbowych. Pozwala to w łatwy sposób wyłączyć z zakresu reprezentacji wykładnika wartości skrajne (00000…00 oraz 11111…11), które wykorzystywane są w arytmetyce zmiennoprzecinkowej do reprezentacji tzw. wartości specjalnych (podrozdział 3.1.2).

Standard IEEE 754 definiuje dwa podstawowe standardowe formaty zmiennoprzecinkowe (tabela 3.1):

1. 32-bitowy format pojedynczej precyzji;
2. 64-bitowy format podwójnej precyzji.

Zasady zapisu liczb zmiennoprzecinkowych są takie same dla obu formatów. Różnią się one od siebie jedynie rozmiarem poszczególnych pól, z czego wynikają różnice w zakresie oraz dokładności reprezentacji liczb.

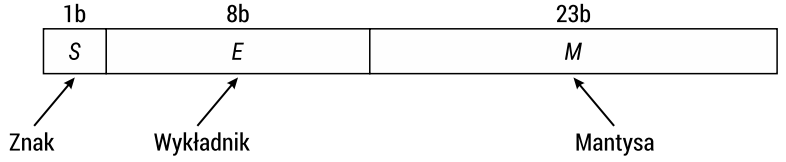
TABELA 3.1. Standardowe formaty zmiennoprzecinkowe

Format	Rozmiar [b]	Wykładnik [b]	Mantysa [b]	Zakres	Dokładność
Pojedynczej precyzji	32	8	23	$2^{\pm 127}$	7
Podwójnej precyzji	64	11	52	$2^{\pm 1023}$	15
Podwójnej rozszerzonej precyzji	80	15	64	$2^{\pm 16383}$	19

W tabeli 3.1 przedstawione zostały parametry standardowych formatów zmiennoprzecinkowych. Wyszczególniono tu rozmiary poszczególnych pól formatów, przybliżony zakres reprezentacji liczb oraz dokładność zapisu liczb zmiennoprzecinkowych wyrażoną liczbą zapamiętywanych cyfr dziesiętnych.

Format podwójnej rozszerzonej precyzji przedstawiony dodatkowo w tabeli 3.1 nie został dokładnie zdefiniowany w standardzie IEEE 754. Zamiast tego sformułowano następujące warunki realizacji takiego formatu: wykładnik oraz mantysa rozszerzonego formatu podwójnej precyzji powinny przyjmować wartości $E \geq 15$ i $M \geq 64$. Zgodnie z tym zaleceniem firma Intel zaimplementowała w swoich procesorach 80-bitowy format (z 15-bitowym polem wykładnika i 64-bitowym polem mantysy), który stał się swoistym standardem, ponieważ inni producenci w celu zachowania kompatybilności również realizowali w swoich układach rozszerzony format w tej samej postaci.

Na rys. 3.1 przedstawiono 32-bitowy standardowy format pojedynczej precyzji, pokazano jego podział bitowy na pola przechowujące podstawowe komponenty zapisu zmiennoprzecinkowego.



RYS. 3.1. Standardowy format zmiennoprzecinkowy pojedynczej precyzji

Najbardziej znaczący bit formatu zawiera znak liczby. Następnie występuje 8-bitowe pole wykładnika zakodowanego w kodzie z przesunięciem *bias*. Zakres

reprezentacji wykładnika w 8-bitowym kodzie z przesunięciem *bias* wynosi od –127 do +128, a samo przesunięcie wykorzystywane do kodowania ma wartość $127_{(10)}$ (lub $01111111_{(2)}$). Najmniejsza (00000000) oraz największa (11111111) wartość wykładnika są wyłączone z zakresu reprezentacji zwykłych liczb, służą one do zapisu wartości specjalnych w formatach zmiennoprzecinkowych (podrozdział 3.1.2).

Pole mantysy formatu zmiennoprzecinkowego pojedynczej precyzji ma rozmiar 23 bity i zapewnia dokładność reprezentacji liczb do 7 cyfr dziesiętnych w zapisie liczby.

Przykład 3.1

Przedstawić liczbę dziesiętną $-117\frac{35}{512}_{(10)}$ *w standardowym formacie zmiennoprzecinkowym pojedynczej precyzji.*

Na początku sprowadzamy liczbę do dwójkowej postaci wykładniczej, z której liczba jest przeliczana na format zmiennoprzecinkowy:

$$-117\frac{35}{512}_{(10)} = -1110101{,}000100011_{(2)} = -1{,}110101000100011 \cdot 2^6.$$

Wykładnik w formacie standardowym kodowany jest w kodzie z przesunięciem *bias*:

$$bias = 2^{8-1} - 1 = 127$$

$$E = 6 + 127 = 133_{(10)} = 10000101_{(\text{biased})}.$$

Wiodąca jedynka z postaci znormalizowanej mantysy nie jest zapisywana do formatu (jest pomijana), a ułamkowa część mantysy rozszerzana jest nieznaczącymi zerami z prawej strony do rozmiaru formatu:

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

Przykład 3.2

Wyznaczyć wartość liczby dziesiętnej przechowywanej w standardowym formacie zmiennoprzecinkowym pojedynczej precyzji $00111101110011000000000000000000_{(\text{IEEE 754})}$.

Zamianę zaczynamy od podziału formatu na odpowiednie pola, z których wyznaczamy części składowe dwójkowej znormalizowanej postaci wykładniczej liczby:

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

Odejmując przesunięcie *bias* = 127 od wartości kodu zapisanego w polu wykładnika ($01111011_{(2)} = 123_{(10)}$), wyznaczamy wykładnik liczby:

$$E = 123 - 127 = -4_{(10)}.$$

Zawartość 23-bitowego pola mantysy stanowi ułamkową część liczby, do której dopisać należy 1 w części całkowitej. W taki sposób otrzymujemy wykładniczą postać liczby i przeliczamy ją na wartość dziesiętną:

$$1{,}10011 \cdot 2^{-4} = 0{,}000110011 = \frac{51}{512}_{(10)}.$$

Przykład 3.3

Obliczyć największą liczbę dziesiętną, którą można zapisać w standardowym formacie zmiennoprzecinkowym pojedynczej precyzji.

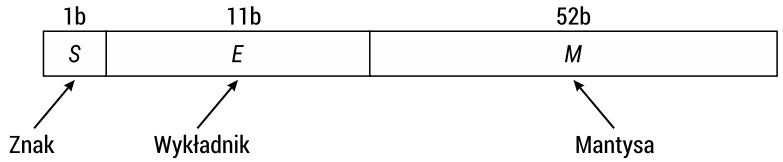
Największa liczba zmiennoprzecinkowa przechowywana jest w formacie standardowym pojedynczej precyzji w następującej postaci:

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

Maksymalna wartość wykładnika wynosi $E = 11111110_{(\text{biased})} = 127_{(10)}$, a największa wartość liczby zmiennoprzecinkowej:

$$1{,}11111111111111111111111 \cdot 2^{127} = \frac{2^{24}-1}{2^{23}} \cdot 2^{127} \approx 3{,}4 \cdot 10^{38}.$$

W 64-bitowym standardowym formacie podwójnej precyzji zdefiniowane są następujące rozmiary poszczególnych pól (rys. 3.2): 1-bitowe pole znaku, 11-bitowe pole wykładnika, 52-bitowe pole mantysy.



RYS. 3.2. Standardowy format zmiennoprzecinkowy podwójnej precyzji

Sposób kodowania i zapisu liczby do formatu podwójnej precyzji jest taki sam, jak w przypadku formatu pojedynczej precyzji, różnica zachodzi jedynie w rozmiarach poszczególnych pól. Wykładnik w formacie standardowym podwójnej precyzji zapisany jest na 11 bitach i zakodowany w kodzie z przesunięciem *bias*, gdzie wartość przesunięcia wynosi 1023 ($2^{11-1} - 1 = 1023$). Zakres reprezentacji wykładnika w takim formacie zmienia się od –1023 do +1024, z tym zastrzeżeniem, że graniczne wartości tego zakresu wykorzystywane są do reprezentacji wartości specjalnych. Taki zakres reprezentacji wykładnika umożliwia zapis liczb w bardzo szerokim zakresie dziesiętnych wartości bezwzględnych – w przybliżeniu od 10^{-308} do 10^{+308}.

Mantysa w formacie podwójnej precyzji zapisana jest na 52 bitach w postaci modułu znormalizowanej liczby dwójkowej bez cyfry 1 stanowiącej część całkowitą

liczby. Zwiększony rozmiar pola mantysy zapewnia dokładność reprezentacji do 15 cyfr dziesiętnych w zapisie liczby:

$$d = \frac{m+1}{\log_2 10} = \frac{52+1}{\log_2 10} \approx \frac{53}{3{,}321928} \approx 15{,}9546.$$

gdzie m – rozmiar pola mantysy w formacie zmiennoprzecinkowym podwójnej precyzji; jedynka dodana do tego rozmiaru uwzględnia cyfrę 1 z części całkowitej liczby przechowywaną w sposób niejawny.

Przykład 3.4

Liczbę dziesiętną –260,125 *przedstawić w standardowym formacie zmiennoprzecinkowym podwójnej precyzji.*

Po znormalizowaniu binarnej postaci liczby otrzymujemy:

$$-260{,}125_{(10)} = -100000100{,}001_{(2)} = -1{,}00000100001 \cdot 2^8.$$

Wykładnik liczby zakodowany w kodzie z przesunięciem *bias* ma postać:

$$E = 8 + 1023 = 1031_{(10)} = 10000000111_{(\text{biased})}.$$

Ułamkową część mantysy zapisujemy w 52-bitowym polu, uzupełniając brakujące bity liczby nieznaczącymi zerami z prawej strony:

1	10000000111	0000010000100000000000000000000000000000000000000000

Przykład 3.5

Obliczyć wartość dziesiętną liczby zmiennoprzecinkowej podwójnej precyzji: 01000000 00101100110000000000000000000000000000000000000000000000.

Kod binarny formatu dzielimy na poszczególne pola zawierające kolejno znak liczby, 11-bitowy wykładnik oraz 52-bitową mantysę:

0	10000000010	1100110000000000000000000000000000000000000000000000

Po rozkodowaniu zawartych w formacie komponentów wyznaczamy dwójkową wykładniczą postać liczby zmiennoprzecinkowej i przeliczamy ją na wartość dziesiętną:

$$E = 1026 - 1023 = 3_{(10)}$$

$$1{,}110011 \cdot 2^3 = 1110{,}011_{(2)} = 14{,}375_{(10)}.$$

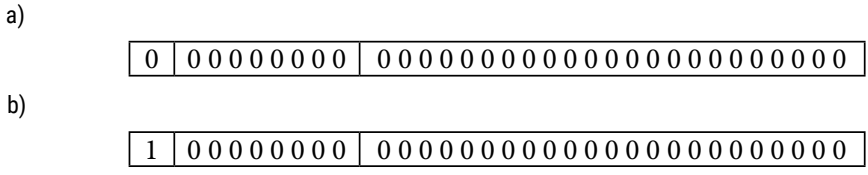
3.1.2. Wartości specjalne

Standard IEEE 754 określa sposób zapisu w formatach zmiennoprzecinkowych tzw. wartości specjalnych, do których należą:

- liczba zero;
- nieskończoność;
- liczby nieznormalizowane;
- nieliczby (*NaN – Not a Number*).

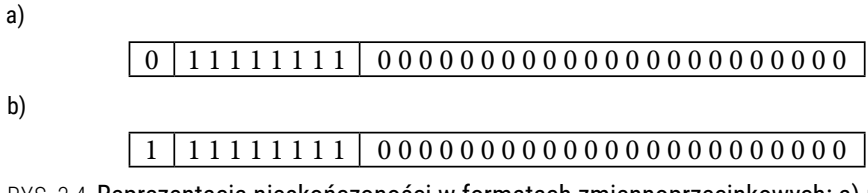
Do reprezentacji wartości specjalnych wykorzystywane są skrajne wartości wykładnika – 00000…00 oraz 11111…11.

Liczba 0 jest specjalnym przypadkiem liczby zmiennoprzecinkowej, ponieważ nie można jej przedstawić w postaci znormalizowanej i przechowywać jak zwykłe liczby. Do reprezentacji liczby 0 używany jest specjalny format, w którym zarówno pole wykładnika, jak i mantysy zawiera wszystkie zera (rys. 3.3). Bit znaku może przyjmować różne wartości (0 lub 1), dlatego też istnieją dwie reprezentacje liczby 0: zero dodatnie i zero ujemne.



RYS. 3.3. Reprezentacja liczby zero w formacie zmiennoprzecinkowym: a) zero dodatnie; b) zero ujemne

W formacie zmiennoprzecinkowym, który reprezentuje nieskończoność, występuje druga skrajna wartość wykładnika – wszystkie jedynki 11111…11 (rys. 3.4). Pole mantysy w takim formacie powinno zawierać same zera. Różne wartości bitu znaku pozwalają zakodować w taki sposób nieskończoność dodatnią (+∞) oraz nieskończoność ujemną (−∞).



RYS. 3.4. Reprezentacja nieskończoności w formatach zmiennoprzecinkowych: a) nieskończoność dodatnia; b) nieskończoność ujemna

W postaci nieskończoności w formatach zmiennoprzecinkowych zapisywane są wyniki działań otrzymywane w przypadku nadmiaru, gdy wynik jest zbyt duży (co do modułu) i wychodzi poza zakres reprezentacji liczb.

Standard IEEE 754 umożliwia także przechowywanie bardzo małych wartości bezwzględnych, których nie da się już przedstawić w postaci znormalizowanej, jednak można jeszcze przechowywać jako liczby nieznormalizowane. Formaty reprezentacji nieznormalizowanych wartości specjalnych przedstawiono na rys. 3.5.

a)

0	0 0 0 0 0 0 0 0	x x x x x x x x x x x x x x x x x x x x x x x

b)

1	0 0 0 0 0 0 0 0	x x x x x x x x x x x x x x x x x x x x x x x

RYS. 3.5. Reprezentacja liczb nieznormalizowanych w formatach zmiennoprzecinkowych: a) liczba dodatnia; b) liczba ujemna

Do reprezentacji liczb nieznormalizowanych zarezerwowano wykładnik o minimalnej wartości (wszystkie zera w polu wykładnika), mantysa natomiast przyjmuje dowolną niezerową wartość. W tej sytuacji liczba nie posiada domyślnej jedynki w części całkowitej, zamiast tego ukrytym bitem liczby jest 0.

Najmniejsza liczba nieznormalizowana, którą da się przedstawić w standardowym formacie zmiennoprzecinkowym pojedynczej precyzji, ma wartość:

$$0{,}00000000000000000000001 \cdot 2^{-126}.$$

Największa liczba nieznormalizowana to $0{,}11111111111111111111111 \cdot 2^{-126}$.

Przykład 3.6

Obliczyć najmniejszą wartość bezwzględną liczby zmiennoprzecinkowej różnej od zera, którą można przechowywać w formacie standardowym pojedynczej precyzji.

Najmniejsza wartość bezwzględna jest przechowywana w formacie zmiennoprzecinkowym pojedynczej precyzji w następującej postaci:

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

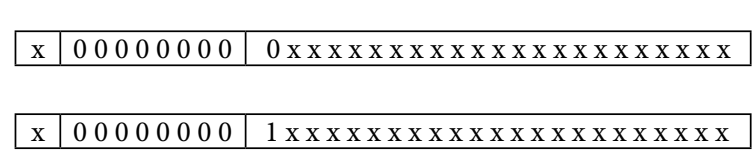
Z tego otrzymujemy liczbę nieznormalizowaną o wartości:

$$0{,}00000000000000000000001 \cdot 2^{-126} = 2^{-23} \cdot 2^{-126} = 2^{-149} \approx 1{,}4 \cdot 10^{-45}.$$

Nieliczby (*NaN*) reprezentują wyniki działań, których nie można przedstawić w postaci liczby zmiennoprzecinkowej. Powstają one w wyniku niedozwolonych operacji, na przykład w przypadku obliczenia pierwiastka z liczby ujemnej lub dzielenia 0 przez 0.

Wyróżniamy dwa rodzaje nieliczb: nieliczby ciche (*quiet NaN – qNaN*) oraz nieliczby sygnalizujące (*signaling NaN – sNaN*). Wystąpienie nieliczby cichej nie generuje wyjątku, natomiast nieliczby sygnalizujące powodują wygenerowanie wyjątku i przerwanie bieżącego programu w celu obsługi wykrytej sytuacji.

Do reprezentacji nieliczb w formatach zmiennoprzecinkowych wykorzystywany jest wykładnik o największej wartości (składający się z samych jedynek), przy czym starszy bit mantysy rozróżnia dwa rodzaje formatów: 1 w starszym bicie oznacza nieliczbę cichą, a 0 – nieliczbę sygnalizującą (rys. 3.6).



RYS. 3.6. Reprezentacja nieliczb w formatach zmiennoprzecinkowych: a) nieliczba sygnalizująca sNaN; b) nieliczba cicha qNaN

Standard IEEE754 dokładnie precyzuje wyniki operacji wykonywanych na wartościach specjalnych, niektóre z nich pokazano w tabeli 3.2.

TABELA 3.2. Wyniki operacji na wartościach specjalnych

Operacja	Wynik
$x / \pm\infty$	0
$\pm\infty \cdot \pm\infty$	$\pm\infty$
$\pm x / 0$	$\pm\infty$
$\infty + \infty$	∞
0 / 0	*NaN*
$\infty - \infty$	*NaN*
$\pm\infty / \pm\infty$	*NaN*
$\pm\infty \cdot 0$	*NaN*

Ponadto każda operacja, której operandem jest nieliczba, również generuje w wyniku nieliczbę.

3.1.3. Standard IEEE 754-2008

Pierwsza wersja standardu IEEE 754 opracowana została w 1985 r. (IEEE 754-1985). W 2008 r. opublikowano rozszerzenie standardu o nazwie IEEE 754-2008. W ramach zmodyfikowanej, rozszerzonej wersji standardu zdefiniowane zostały nowe dodatkowe formaty zmiennoprzecinkowe oraz wprowadzono nowy sposób zapisu liczb oparty na dziesiętnym wykładniczym zapisie liczby.

Dziesiętna reprezentacja zmiennoprzecinkowa pozwala znacznie rozszerzyć zakres reprezentacji liczb przy zachowaniu takiej samej dokładności, jak w przypadku dwójkowej reprezentacji zmiennoprzecinkowej. Oprócz tego zapis dziesiętny eliminuje błędy niedokładności związane z zamianą liczb dziesiętnych na system dwójkowy.

Jego wadą jest natomiast komplikacja algorytmów wykonywania operacji na dziesiętnej postaci liczb.

W tabeli 3.3 przedstawione zostały standardowe formaty zmiennoprzecinkowe zdefiniowane w nowej specyfikacji standardu. Tabela zawiera informacje o rozmiarach poszczególnych pól formatów, przybliżonych zakresach reprezentacji liczb (wyrażonych maksymalną wartością wykładnika w systemie dziesiętnym), a także dokładności reprezentacji (podanej w postaci liczby zapamiętywanych cyfr dziesiętnych).

Formaty przedstawione w tabeli 3.3 możemy podzielić na dwie grupy:

1. formaty dwójkowej reprezentacji zmiennoprzecinkowej (*binary*);
2. formaty dziesiętnej reprezentacji zmiennoprzecinkowej (*decimal*).

TABELA 3.3. Formaty standardowe zgodne ze standardem IEEE 754-2008

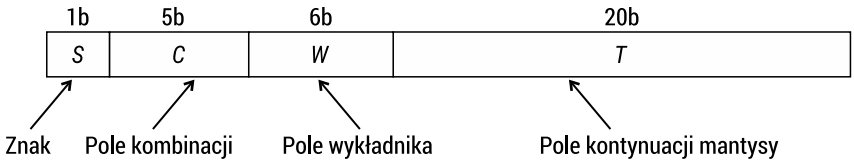
Format	Rozmiar	Wykładnik	Mantysa	Zakres (E_{max})	Dokładność
*binary*16	2B	5b	10b	+5	3
*binary*32	4B	8b	23b	+38	7
*binary*64	8B	11b	52b	+308	15
*binary*128	16B	15b	112b	+4932	34
*decimal*32	4B	6b	20b	+96	7
*decimal*64	8B	8b	50b	+384	16
*decimal*128	16B	12b	110b	+6144	34

Wśród formatów binarnych możemy zauważyć już dobrze nam znane formaty pojedynczej (*binary*32) oraz podwójnej (*binary*64) precyzji. Oprócz tego zdefiniowany został skrócony 16-bitowy format dwójkowy (*binary*16) o bardzo ograniczonych parametrach (zakresie oraz dokładności reprezentacji), przydatny do niezbyt wymagających zastosowań obliczeniowych.

Interesujący wydaje się nowy dostępny 128-bitowy format binarny (*binary*128). W formacie tym zachowano 15-bitowy wykładnik (podobnie jak w 80-bitowym formacie podwójnej rozszerzonej precyzji), co zapewnia bardzo szeroki zakres reprezentacji liczb. Wszystkie dodatkowe bity formatu wykorzystano natomiast do zwiększenia rozmiaru pola mantysy, co pozwoliło osiągnąć bardzo imponującą dokładność reprezentacji liczb – do 34 cyfr dziesiętnych.

Oprócz formatów binarnych standard IEEE 754-2008 definiuje trzy formaty dziesiętnego zmiennoprzecinkowego zapisu liczb (DFP – *Decimal Floating-Point*). Jak widzimy z tabeli 3.3, formaty te oferują znacznie szerszy zakres reprezentacji liczb w porównaniu z formatami binarnymi tego samego rozmiaru przy zachowaniu porównywalnej dokładności.

Omówimy zasady dziesiętnej reprezentacji zmiennoprzecinkowej na przykładzie 32-bitowego dziesiętnego formatu (*decimal*32) przedstawionego na rys. 3.7.



RYS. 3.7. Dziesiętny format zmiennoprzecinkowy decimal32

Dziesiętny format zmiennoprzecinkowy składa się z następujących pól:

- *S* – bit znaku liczby (*sign*);
- *C* – pole kombinacji (*combination field*);
- *W* – pole wykładnika (*biased exponent field*);
- *T* – pole kontynuacji mantysy (*trailing significand field*).

W polu *S* kodowany jest znak liczby w taki sposób, że $S = 1$ oznacza liczbę ujemną, a $S = 0$ – liczbę dodatnią.

Pole kombinacji *C* pozwala między innymi rozróżnić, czy w formacie jest zapisana liczba, czy też wartość specjalna – nieskończoność (±∞) lub nieliczba (*NaN*). W przypadku wartości specjalnej pole *C* zawiera określone kody binarne przedstawione w tabeli 3.4. W ten sposób nieskończoność reprezentowana jest formatem, który w polu C zawiera kod 11110, a różne stany bitu znaku pozwalają przedstawić nieskończoność dodatnią (+∞) oraz nieskończoność ujemną (−∞).

TABELA 3.4. Dekodowanie pola kombinacji C

Pole kombinacji *C*	Typ wartości	Dwa starsze bity wykładnika	Starsza cyfra dziesiętna mantysy
a b c d e	liczba	*a b*	(0)*cde*
1 1 *c d e*	liczba	*c d*	(100)*e*
1 1 1 1 0	±∞	–	–
1 1 1 1 1	*NaN*	–	–

W przypadku nieliczby ($C = 11111$) do rozróżnienia formatów nieliczb cichych oraz sygnalizujących służy starszy bit pola wykładnika *W*: bit $W_0 = 0$ oznacza nieliczbę cichą (*qNaN*), a bit $W_0 = 1$ – nieliczbę sygnalizującą (*sNaN*).

Jeżeli dziesiętny format zmiennoprzecinkowy reprezentuje liczbę, wtedy w polu kombinacji *C* umieszczone są dwa starsze bity wykładnika oraz dwójkowy kod najbardziej znaczącej dziesiętnej cyfry mantysy. Sposób kodowania tych wartości wyjaśniony został w pierwszych dwóch wierszach tabeli 3.4. Para bitów {*a*, *b*} w pierwszym wierszu tabeli, podobnie jak bity {*c*, *d*} w drugim wierszu, mogą przyjmować wyłącznie wartości: 00, 01 oraz 10. Nawiasy { } w powyższym zapisie oznaczają konkatenację pól bitowych.

W pierwszym wierszu tabeli 3.4 opisana została sytuacja, gdy starsza cyfra dziesiętna mantysy znajduje się w zakresie od 0 do 7, wtedy pole kombinacji *C* może

przyjmować wartości 00xxx, 01xxx lub 10xxx. W takim wypadku bity {*a*, *b*} pola kombinacji *C* stanowią dwa starsze bity wykładnika, które w połączeniu z polem *W* tworzą wykładnik liczby przedstawiony w dwójkowym kodzie z przesunięciem *bias*. Pozostałe trzy bity (0)*cde* pola *C*, poprzedzone zerem, przechowują najbardziej znaczącą cyfrę dziesiętną mantysy, która może przyjmować wartości od 0 do 7 (dwójkowo $(0)000_{(2)}$ – $(0)111_{(2)}$). Zero w nawiasie (0) oznacza tu cyfrę dwójkową, która jest przechowywana w formacie zmiennoprzecinkowym w sposób niejawny, czyli nie jest zapisywana do formatu, lecz uwzględniana przy odczycie liczby. Starsza cyfra mantysy zakodowana w polu *C* wraz polem kontynuacji mantysy *T* tworzy mantysę liczby zmiennoprzecinkowej.

Jeśli mantysa liczby rozpoczyna się od cyfry 8 lub 9, wtedy kodowanie odbywa się zgodnie z zawartością drugiego wiersza tabeli 3.4. W tym wypadku kod w polu kombinacji *C* zaczyna się od dwóch jedynek – 11*cde*, gdzie bity {*c*, *d*} stanowią dwa najbardziej znaczące bity wykładnika (mogą przyjmować wartości 00, 01 lub 10), a bit *e* wraz z przedrostkiem 100, czyli (100)*e*, tworzy starszą dziesiętną cyfrę mantysy o wartości 1000 lub 1001 ($8_{(10)}$ lub $9_{(10)}$). Zwróćmy uwagę na to, że bity w nawiasie (100) są przechowywane w sposób niejawny, więc przy odczycie liczby są dopisywane do bitu *e*, tworząc binarny kod starszej cyfry mantysy (100)*e*.

Po połączeniu dwóch najbardziej znaczących bitów wykładnika zapisanych w polu *C* z polem wykładnika *W* otrzymujemy 8-bitowy wykładnik liczby przedstawiony w binarnym kodzie z przesunięciem *bias*. Zakresy reprezentacji wykładników oraz wartości przesunięcia *bias* dla różnych formatów dziesiętnych przedstawiono w tabeli 3.5, gdzie $E_{\min}$ – minimalna wartość wykładnika, $E_{\max}$ – maksymalna wartość wykładnika liczby w postaci znormalizowanej, $E_{\max f}$ – maksymalna wartość wykładnika, która występuje w formacie zmiennoprzecinkowym.

TABELA 3.5. Zakresy reprezentacji wykładników w dziesiętnych formatach zmiennoprzecinkowych

Wartość	*decimal*32	*decimal*64	*decimal*128
$E_{\min}$	96	384	6144
$E_{\max}$	−101	−398	−6176
$E_{\max f}$	90	369	6111
bias	101	398	6176

Wartość przesunięcia *bias* obliczana jest według następującego wzoru:

$$bias = E_{\max} + p - 2,$$

gdzie *p* – liczba cyfr dziesiętnych zapisywanych do formatu zmiennoprzecinkowego, dla formatu *decimal*32 $p = 7$, a $bias = 96 + 7 - 2 = 101$.

Wartość $E_{\max f}$ w tabeli 3.5 oznacza maksymalną wartość wykładnika, którą można zapisać do formatu zmiennoprzecinkowego. Jak możemy zauważyć, jest ona mniejsza od $E_{\max}$. Dzieje się tak, ponieważ maksymalny wykładnik $E_{\max}$ jest stosowany

do wykładniczej postaci liczby, a do formatu dziesiętnego liczba jest zapisywana jako liczba całkowita. Z tego powodu maksymalna wartość wykładnika zapisywanego do formatu zmiennoprzecinkowego zostanie zmniejszona o $(p–1)$:

$$E_{\text{maxf}} = E_{\text{max}} - (p–1),$$

gdzie p – liczba cyfr dziesiętnych przechowywanych w formacie zmiennoprzecinkowym. W ten sposób dla 32-bitowego dziesiętnego formatu standardowego otrzymujemy $E_{\text{maxf}} = 96 - 6 = 90$.

Największa liczba, którą można przedstawić w 32-bitowym dziesiętnym formacie zmiennoprzecinkowym, to $9{,}999999 \cdot 10^{96}$. Ponieważ mantysa liczby zapisywana jest do formatu w postaci liczby całkowitej 9999999, wymaga to przeniesienia przecinka z jednoczesnym zmniejszeniem wykładnika o 6, czyli wartość maksymalnego wykładnika w tym przypadku wynosi 90:

$$9.999999 \cdot 10^{96} = 9999999 \cdot 10^{90}.$$

Mantysa w dziesiętnym formacie zmiennoprzecinkowym jest liczbą całkowitą bez znaku, której starsza cyfra dziesiętna zakodowana jest w polu *C*, a reszta cyfr przechowywana jest w polu kontynuacji mantysy *T*. Rozmiar pola *T* zwykle jest wielokrotnością 10 bitów: w formacie *decimal*32 pole *T* składa się z dwóch 10-bitowych grup, w formacie *decimal*64 – z pięciu 10-bitowych grup, a w formacie *decimal*128 – z jedenastu 10-bitowych grup.

Przy zapisie liczby do dziesiętnego formatu zmiennoprzecinkowego mantysa liczby sprowadzana jest do postaci liczby całkowitej w taki sposób, aby znaczące cyfry zajęły wszystkie dostępne pola mantysy, bez wiodących zer z lewej strony. Rzeczywiste położenie przecinka pozycyjnego w takiej liczbie określa wartość wykładnika. W przypadku niewielkiej liczby cyfr znaczących wiodące zera są dopuszczalne, a liczby takie mogą mieć różne postaci zapisu. Liczba 0 reprezentowana jest w dziesiętnym formacie o zerowej wartości mantysy, przy czym pozostałe bity formatu są bez znaczenia.

Przykład 3.7

Przedstawić liczby 123,4567*,* 0,09575126*,* 1,27513683*,* 457375172,5*,* 157,213 *i* 22 *w postaci zapisywanej do* 32*-bitowego dziesiętnego formatu zmiennoprzecinkowego.*

Każdą z podanych liczb sprowadzamy do postaci 7-cyfrowej liczby całkowitej, przenosząc odpowiednio przecinek dziesiętny z jednoczesną zmianą wartości wykładnika. Wykładnik w tym przypadku jest potęgą liczby 10. Wyniki przekształcenia liczb podano w tabeli 3.6.

Liczby dziesiętne, które mają mniej niż 7 cyfr znaczących, mogą być zapisywane w różnych postaciach, też z wiodącymi zerami z lewej strony. Jeśli znaczących cyfr jest więcej, liczba ulega zaokrągleniu do 7 cyfr w zapisie liczby całkowitej.

TABELA 3.6. Przedstawienie liczb w postaci zapisywanej do formatu dziesiętnego decimal32

Liczba	Mantysa	Wykładnik
123,4567	1234567	−4
0,09575126	9575126	−8
1,27513683	1275137	−6
457375172,5	4573752	2
157,213	0157213	−3
22	0000022	0

Mantysa w dziesiętnym formacie zmiennoprzecinkowym może być zakodowana na jeden z dwóch alternatywnych sposobów: 1) w binarnym kodzie NKB lub 2) w dwójkowo-dziesiętnym kodzie DPD (*Densely Packed Decimal*). Wybór sposobu kodowania nie ma wpływu na zakres oraz dokładność reprezentacji liczb.

Przy wykorzystaniu kodu DPD każda 10-bitowa grupa pola *T* reprezentuje 3 cyfry dziesiętne mantysy. W 20-bitowym polu *T* formatu *decimal*32 umieszczono 6 cyfr dziesiętnych zakodowanych w kodzie DPD, dodatkowo najbardziej znacząca cyfra mantysy zapisana jest w polu kombinacji *C*, więc łącznie format przechowuje 7 cyfr dziesiętnych.

Kodowanie w kodzie DPD jest dość skomplikowane, jednak sprzętowa realizacja kodera i dekodera pozwala wykonywać operacje zamiany bardzo szybko. Kod DPD pozwala zapisać 3 cyfry dziesiętne na 10 bitach (zamiast 12 bitów w kodzie BCD), dlatego jest nazywany kodem upakowanym.

Do kodowania wykorzystano fakt, iż mniejsze wartości cyfr dziesiętnych (0–7) można zakodować na 3 bitach, a do rozróżnienia większych cyfr (8 i 9) wystarczy 1 bit (zerem jest kodowana cyfra 8, a jedynką – 9). W kodzie DPD jednocześnie kodowane są 3 cyfry dziesiętne d_1, d_2, d_3 za pomocą 10-bitowego kodu $\{b_0, b_1, b_2, b_3, b_4, b_5, b_6, b_7, b_8, b_9\}$. Sposób kodowania opisany został w tabeli 3.7.

TABELA 3.7. Kodowanie trzech cyfr dziesiętnych w 10-bitowym kodzie DPD [5]

$d_1[0], d_2[0], d_3[0]$	b_0, b_1, b_2	b_3, b_4, b_5	b_6	b_7, b_8, b_9
0 0 0	$d_1[1:3]$	$d_2[1:3]$	0	$d_3[1:3]$
0 0 1	$d_1[1:3]$	$d_2[1:3]$	1	0, 0, $d_3[3]$
0 1 0	$d_1[1:3]$	$d_3[1:2], d_2[3]$	1	0, 1, $d_3[3]$
0 1 1	$d_1[1:3]$	1, 0, $d_2[3]$	1	1, 1, $d_3[3]$
1 0 0	$d_3[1:2], d_1[3]$	$d_2[1:3]$	1	1, 0, $d_3[3]$
1 0 1	$d_2[1:2], d_1[3]$	0, 1, $d_2[3]$	1	1, 1, $d_3[3]$
1 1 0	$d_3[1:2], d_1[3]$	0, 0, $d_2[3]$	1	1, 1, $d_3[3]$
1 1 1	0, 0, $d_1[3]$	1, 1, $d_2[3]$	1	1, 1, $d_3[3]$

W tabeli 3.7 wymieniono wszystkie możliwe kombinacje występowania cyfr mniejszych (0–7) oraz większych (8 i 9) w 3-cyfrowej sekwencji cyfr dziesiętnych d_1, d_2, d_3. Kombinacje te oznaczone zostały w pierwszej kolumnie tabeli 3.7 za pomocą starszych bitów $\{d_1[0], d_2[0], d_3[0]\}$ kodów BCD cyfr dziesiętnych (starszy bit $d_i[0]$ kodu BCD cyfry pozwala rozróżnić cyfry mniejsze (0–7) i większe (8 i 9)). W tabeli 3.7 wykorzystano odwołania do poszczególnych bitów kodów BCD cyfr dziesiętnych za pomocą indeksów zgodnie z oznaczeniami podanymi na rys. 3.8.

d_1				d_2				d_3			
0	1	2	3	0	1	2	3	0	1	2	3

← cyfry dziesiętne
← indeksy bitów cyfr BCD

Rys. 3.8. Adresowanie poszczególnych bitów cyfr dziesiętnych

Kod BCD każdej cyfry dziesiętnej składa się z 4 bitów – $d_1[0:3]$, $d_2[0:3]$, $d_3[0:3]$, gdzie bit 0 jest najbardziej znaczącym bitem w zapisie dwójkowym cyfry dziesiętnej. Według przedstawionych oznaczeń zapis $d_1[0]$ odpowiada najstarszemu bitowi w kodzie BCD cyfry d_1, oznaczenie $d_2[3]$ – najmniej znaczącemu bitowi cyfry d_2, a zapis $d_3[1:3]$ – 3-bitowej sekwencji bitów od 1 do 3 cyfry d_3. Poszczególne wiersze tabeli 3.7 opisują sposób tworzenia 10-bitowego kodu DPD $\{b_0, b_1, \ldots, b_9\}$ dla każdej kombinacji występowania cyfr mniejszych i większych w 3-cyfrowej sekwencji dziesiętnej.

Przykład 3.8

Zakodować liczbę dziesiętną $945_{(10)}$ *w upakowanym dwójkowo-dziesiętnym kodzie DPD.*

Przedstawimy liczbę $945_{(10)}$ w kodzie BCD:

$$945_{(10)} = 1001\ 0100\ 0101_{(BCD)}.$$

d_1				d_2				d_3			
0	1	2	3	0	1	2	3	0	1	2	3
1	0	0	1	0	1	0	0	0	1	0	1

Kombinacja występowania cyfr dziesiętnych określana jest wartościami starszych bitów cyfr BCD – $\{d_1[0], d_2[0], d_3[0]\} = 100$. Na podstawie tej wartości wybieramy odpowiedni wiersz tabeli 3.7 i postępujemy zgodnie z opisanym w nim sposobem kodowania:

- bity b_0, b_1, b_2 opisane są wzorem $d_3[1:2]$, $d_1[3]$, więc zapisujemy odpowiednią sekwencję bitów $\{b_0, b_1, b_2\} = 101$;
- bity b_3, b_4, b_5 kodowane są za pomocą wartości $d_2[1:3]$, więc otrzymujemy $\{b_3, b_4, b_5\} = 100$;
- bit $b_6 = 1$;
- bity b_7, b_8, b_9 wyznaczane są sekwencją 1, 0, $d_3[3]$, czyli $\{b_7, b_8, b_9\} = 101$.

W wyniku zamiany otrzymujemy następujący 10-bitowy kod liczby $945_{(10)}$:

$$1011001101_{(DPD)}.$$

Algorytm dekodowania kodu DPD opisuje tabela 3.8.

TABELA 3.8. Dekodowanie 10-bitowego kodu DPD na 3 cyfry dziesiętne [5]

b_6, b_7, b_8, b_3, b_4	d_1	d_2	d_3
0 x x x x	$4b_0 + 2b_1 + b_2$	$4b_3 + 2b_4 + b_5$	$4b_7 + 2b_8 + b_9$
1 0 0 x x	$4b_0 + 2b_1 + b_2$	$4b_3 + 2b_4 + b_5$	$8 + b_9$
1 0 1 x x	$4b_0 + 2b_1 + b_2$	$8 + b_5$	$4b_3 + 2b_4 + b_9$
1 1 0 x x	$8 + b_2$	$4b_3 + 2b_4 + b_5$	$4b_0 + 2b_1 + b_9$
1 1 1 0 0	$8 + b_2$	$8 + b_5$	$4b_0 + 2b_1 + b_9$
1 1 1 0 1	$8 + b_2$	$4b_0 + 2b_1 + b_5$	$8 + b_9$
1 1 1 1 0	$4b_0 + 2b_1 + b_2$	$8 + b_5$	$8 + b_9$
1 1 1 1 1	$8 + b_2$	$8 + b_5$	$8 + b_9$

Pierwsza kolumna tabeli 3.8 zawiera wartości bitów kontrolnych b_6, b_7, b_8, b_3, b_4, które odpowiadają różnym kombinacjom rozmieszczenia dużych i mniejszych cyfr dziesiętnych w zakodowanej sekwencji. Pozostałe kolumny tabeli opisują zasady dekodowania poszczególnych cyfr dziesiętnych d_1, d_2 oraz d_3.

Przykład 3.9

Wyznaczyć wartość dziesiętną liczby zakodowanej w kodzie DPD: $1011011110_{(DPD)}$.

Na początku wypisujemy z kodu DPD sekwencję bitów kontrolnych b_6, b_7, b_8, b_3, b_4:

b_0	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	b_9
1	0	1	1	0	1	1	1	1	0

$$\{b_6, b_7, b_8, b_3, b_4\} = 11110.$$

Wyznaczona kombinacja bitów kontrolnych odpowiada przedostatniemu wierszowi tabeli 3.8, stąd otrzymujemy:

$$d_1 = 4b_0 + 2b_1 + b_2 = 5$$
$$d_2 = 8 + b_5 = 9$$
$$d_3 = 8 + b_9 = 8.$$

Zakodowana liczba dziesiętna to $598_{(10)}$.

Zasady dziesiętnej reprezentacji liczb wyjaśnimy na przykładach zapisu liczb do 32-bitowego dziesiętnego formatu zmiennoprzecinkowego.

Przykład 3.10

Przedstawić liczbę $-375{,}2941_{(10)}$ *w 32-bitowym dziesiętnym formacie zmiennoprzecinkowym decimal*32. *Do kodowania mantysy wykorzystać binarny kod NKB.*

Zaczynamy od sprowadzenia mantysy do postaci 7-cyfrowej liczby całkowitej z odpowiednią zmianą wartości wykładnika. Otrzymujemy następującą postać liczby:

$$-375{,}2941_{(10)} = -3752941 \cdot 10^{-4}.$$

Znak liczby jest kodowany w starszym bicie formatu $S = 1$.

Mantysa przechowywana jest w dziesiętnym formacie zmiennoprzecinkowym jako liczba całkowita zakodowana w kodzie NKB:

$$3752941_{(10)} = 11100101000011111101101_{(2)}.$$

Zapiszemy mantysę na 24 bitach:

$$(0)\ \underline{0\ 1\ 1}\ 1\ 0\ 0\ 1\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 1\ 0\ 1\ 1\ 0\ 1_{(2)}.$$

Umieszczony w nawiasie starszy bit mantysy nie jest zapisywany do formatu, przechowywany jest w sposób niejawny. Ponieważ starsze 4 cyfry mantysy (0)011 odpowiadają wartościom dziesiętnym z zakresu 0–7 ($0000_{(2)} - 0111_{(2)}$), to przyjmuje się sposób kodowania przedstawiony w pierwszym wierszu tabeli 3.4. Pole kombinacji *C* zawiera 5 bitów {*a*, *b*, *c*, *d*, *e*}. Do pierwszych dwóch z nich (*a* i *b*) zapisuje się 2 starsze bity przesuniętego wykładnika, a kolejne bity *c*, *d* i *e* przechowują 3 bity mantysy umieszczone po domyślnym zerze (0) (bity te zostały podkreślone w mantysie). Otrzymujemy w ten sposób {*c*, *d*, *e*} = 011.

Zakodujemy wykładnik liczby $E = -4$ w kodzie z przesunięciem *bias*:

$$bias = E_{\max} + p - 2 = 96 + 7 - 2 = 101_{(10)}$$

$$E = -4 + 101 = 97_{(10)} = \underline{01}100001_{(\text{biased})}.$$

Dwa starsze bity wyznaczonego wykładnika przechowywane są w polu kombinacji *C* {*a*, *b*} = 01, pozostałe bity zapisuje się do 6-bitowego pola wykładnika *W*. Pole kontynuacji mantysy *T* zawiera 20 mniej znaczących bitów zakodowanej dwójkowo mantysy.

Po połączeniu wszystkich wyznaczonych komponentów otrzymujemy następujący format zmiennoprzecinkowy:

S	*C*	*W*	*T*
1	0 1 0 1 1	1 0 0 0 0 1	1 0 0 1 0 1 0 0 0 0 1 1 1 1 1 0 1 1 0 1
	a b c d e		

Przykład 3.11

Przedstawić liczbę $0{,}063851347_{(10)}$ *w 32-bitowym dziesiętnym formacie zmiennoprzecinkowym decimal*32. *Do kodowania mantysy wykorzystać dwójkowo-dziesiętny kod DPD.*

Sprowadzamy mantysę do postaci liczby całkowitej, zaokrąglając liczbę do 7 cyfr znaczących:

$$0{,}063851347_{(10)} = 6385135 \cdot 10^{-8}.$$

Ponieważ starsza dziesiętna cyfra mantysy (6) zawiera się w zakresie wartości dziesiętnych 0–7, wybieramy sposób kodowania opisany w pierwszym wierszu tabeli 3.4, czyli bity *a* i *b* pola kombinacji *C* będą przechowywały dwa starsze bity wykładnika, a bity *c*, *d*, *e* – starszą cyfrę mantysy w postaci (0){*c*, *d*, *e*} = (0)110.

Wykładnik zakodujemy w kodzie z przesunięciem *bias*:

$$E = -8 + 101 = 93_{(10)} = \underline{01}011101_{(\text{biased}).}$$

Dwa starsze bity wykładnika przechowywane są w polu kombinacji *C* {*a*, *b*} = 01, a pozostałe bity zawiera pole *W* = 011101. Znak liczby kodowany jest oddzielnie w starszym bicie formatu *S* = 0.

Sześć mniej znaczących cyfr dziesiętnych mantysy zakodujemy w kodzie DPD za pomocą tabeli 3.7. Każde 3 cyfry dziesiętne kodowane są oddzielnie. Najpierw cyfry $385_{(10)}$ zamienimy na 10-bitowy kod DPD:

$$385_{(10)} = 0011\ 1000\ 0101_{(\text{BCD})}$$

d_1				d_2				d_3			
0	1	2	3	0	1	2	3	0	1	2	3
0	0	1	1	1	0	0	0	0	1	0	1

$$\{d_1[0], d_2[0], d_3[0]\} = 010.$$

Do kodowania wybieramy trzeci wiersz tabeli 3.7, który zawiera w pierwszej kolumnie kombinację 010:

$$\{b_0, b_1, b_2\} = d_1[1{:}3] = 011$$

$$\{b_3, b_4, b_5\} = d_3[1{:}2], d_2[3] = 100$$

$$b_6 = 1$$

$$\{b_7, b_8, b_9\} = 0, 1, d_3[3] = 011$$

$$385_{(10)} = 0111001011_{(\text{DPD})}.$$

W podobny sposób kodujemy trzy młodsze cyfry mantysy $135_{(10)}$:

$$135_{(10)} = 0001\ 0011\ 0101_{(BCD)}$$

d_1				d_2				d_3			
0	1	2	3	0	1	2	3	0	1	2	3
0	0	0	1	0	0	1	1	0	1	0	1

$$\{d_1[0], d_2[0], d_3[0]\} = 000.$$

Zgodnie z pierwszym wierszem tabeli 3.7, który zawiera w pierwszej kolumnie kombinację 000, otrzymujemy:

$$\{b_0, b_1, b_2\} = d_1[1{:}3] = 001$$

$$\{b_3, b_4, b_5\} = d_2[1{:}3] = 011$$

$$b_6 = 0$$

$$\{b_7, b_8, b_9\} = d_3[1{:}3] = 101$$

$$135_{(10)} = 0010110101_{(DPD)}.$$

Ostatecznie otrzymujemy następujący format zmiennoprzecinkowy:

S	*C*	*W*	*T*
0	0 1 1 1 0	0 1 1 1 0 1	0 1 1 1 0 0 1 0 1 1 0 0 1 0 1 1 0 1 0 1
	a b c d e		

Szesnastkowa reprezentacja zawartości formatu wynosi $39D72CB5_{(16)}$.

3.2. Zadania z rozwiązaniami

Zadanie 3.1

Przedstawić liczbę dziesiętną $-\frac{25}{128}_{(10)}$ *w standardowym binarnym formacie zmiennoprzecinkowym pojedynczej precyzji.*

Ułamek $-\frac{25}{128}_{(10)}$ ma dokładny zapis dwójkowy, więc liczba będzie reprezentowana dokładnie w formacie zmiennoprzecinkowym. W celu zapisu liczby dziesiętnej do binarnego formatu pojedynczej precyzji należy sprowadzić ją do dwójkowej postaci wykładniczej:

$$-\frac{25}{128}_{(10)} = -0{,}0011001_{(2)} = -1{,}1001 \cdot 2^{-3}.$$

Wykładnik w kodzie z przesunięciem *bias* ma postać:

$$E = -3 + 127 = 124_{(10)} = 01111100_{(\text{biased})}.$$

Mantysa jest zapisywana do formatu w postaci modułu bez wiodącej jedynki z części całkowitej. Dodatkowo mantysa zostaje rozszerzona nieznaczącymi zerami z prawej strony. W wyniku otrzymujemy następujący format zmiennoprzecinkowy:

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

Zadanie 3.2

Przedstawić liczbę dziesiętną $75{,}37_{(10)}$ *w standardowym binarnym formacie zmiennoprzecinkowym pojedynczej precyzji.*

Podana liczba nie ma skończonego rozwinięcia dwójkowego, z tego powodu nie da się przedstawić liczby dokładnie w dwójkowym formacie zmiennoprzecinkowym. W tej sytuacji należy obliczyć tyle bitów dwójkowego rozszerzenia liczby, ile zmieści się w polu mantysy, uwzględniając także dodatkowy bit wykorzystywany do zaokrąglenia liczby:

$$75_{(10)} = 1001011_{(2)}$$

$0{,}37 \cdot 2 = 0{,}74$	$c_{-1} = 0$	$0{,}44 \cdot 2 = 0{,}88$	$c_{-10} = 0$
$0{,}74 \cdot 2 = 1{,}48$	$c_{-2} = 1$	$0{,}88 \cdot 2 = 1{,}76$	$c_{-11} = 1$
$0{,}48 \cdot 2 = 0{,}96$	$c_{-3} = 0$	$0{,}76 \cdot 2 = 1{,}52$	$c_{-12} = 1$
$0{,}96 \cdot 2 = 1{,}92$	$c_{-4} = 1$	$0{,}52 \cdot 2 = 1{,}04$	$c_{-13} = 1$
$0{,}92 \cdot 2 = 1{,}84$	$c_{-5} = 1$	$0{,}04 \cdot 2 = 0{,}08$	$c_{-14} = 0$
$0{,}84 \cdot 2 = 1{,}68$	$c_{-6} = 1$	$0{,}08 \cdot 2 = 0{,}16$	$c_{-15} = 0$
$0{,}68 \cdot 2 = 1{,}36$	$c_{-7} = 1$	$0{,}16 \cdot 2 = 0{,}32$	$c_{-16} = 0$
$0{,}36 \cdot 2 = 0{,}72$	$c_{-8} = 0$	$0{,}32 \cdot 2 = 0{,}64$	$c_{-17} = 0$
$0{,}72 \cdot 2 = 1{,}44$	$c_{-9} = 1$	$0{,}64 \cdot 2 = 1{,}28$	$c_{-18} = 1$

W kolejnym kroku normalizujemy i zaokrąglamy otrzymaną liczbę dwójkową.

$$1001011{,}010111101011100001 = 1{,}00101101011110101011100001 \cdot 2^6.$$

Domyślnym sposobem zaokrąglenia w binarnych formatach standardowych jest zaokrąglenie do najbliższej liczby maszynowej. Takie zaokrąglenie polega na dodaniu 1 do pierwszej cyfry dwójkowej, która nie mieści się w polu mantysy. W przypadku 23-bitowego formatu będzie to wartość 2^{-24}:

$$\begin{array}{rl} 1,0\,0\,1\,0\,1\,1\,0\,1\,0\,1\,1\,1\,1\,0\,1\,0\,1\,1\,1\,0\,0\,0\,0 & 1 \\ + & 1 \\ \hline 1,0\,0\,1\,0\,1\,1\,0\,1\,0\,1\,1\,1\,1\,0\,1\,0\,1\,1\,1\,0\,0\,0\,1 & 0 \end{array}$$

$$1{,}00101101011110101011100001 \cdot 2^6 \approx 1{,}00101101011110101011100001 \cdot 2^6.$$

Wykładnik w kodzie z przesunięciem *bias* ma następującą postać:

$$E = 6 + 127 = 133_{(10)} = 10000101_{(\text{biased})}.$$

Wyznaczone komponenty liczby zapisujemy do formatu zmiennoprzecinkowego:

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

Szesnastkowy zapis zawartości formatu wynosi $4296BD71_{(16)}$.

Zadanie 3.3

Przedstawić liczbę dziesiętną $-15328{,}328125_{(10)}$ *w standardowym binarnym formacie zmiennoprzecinkowym podwójnej precyzji.*

Binarna reprezentacja zmiennoprzecinkowa wymaga zapisu liczby w dwójkowej postaci wykładniczej, więc do takiej postaci sprowadzamy liczbę:

$$15328_{(10)} = 11101111100000_{(2)}$$

$0{,}328125 \cdot 2 = 0{,}65625$	$c_{-1} = 0$
$0{,}65625 \cdot 2 = 1{,}3125$	$c_{-2} = 1$
$0{,}3125 \cdot 2 = 0,\ 625$	$c_{-3} = 0$
$0{,}625 \cdot 2 = 1{,}125$	$c_{-4} = 1$
$0{,}125 \cdot 2 = 0{,}5$	$c_{-5} = 0$
$0,\ 5 \cdot 2 = 1{,}0$	$c_{-6} = 1$

$$0{,}328125_{(10)} = 0{,}010101_{(2)}$$

$$-11101111100000{,}010101_{(2)} = -1{,}1101111100000010101 \cdot 2^{13}.$$

Wykładnik w formacie zmiennoprzecinkowym podwójnej precyzji jest 11-bitowy, dlatego przesunięcie *bias* ma wartość:

$$bias = 2^{(11-1)} - 1 = 1023_{(10)}.$$

Wyznaczamy zakodowaną postać wykładnika:

$$E = 13 + 1023 = 1036_{(10)} = 10000001100_{(\text{biased})}.$$

Zapisujemy wyznaczone komponenty liczby do formatu zmiennoprzecinkowego z uwzględnieniem rozmiarów pól standardowego formatu podwójnej precyzji: 1 bitu znaku, 11-bitowego wykładnika, 52-bitowej mantysy.

1	10000001100	1101111100000010101000000000000000000000000000000000000000

W postaci szesnastkowej zawartość formatu wynosi $C0CDF02A00000000_{(16)}$.

Zadanie 3.4

Wyznaczyć dziesiętną wartość liczby zapisanej w standardowym binarnym formacie zmiennoprzecinkowym pojedynczej precyzji, jeśli zawartość formatu wynosi $C36DC000_{(16)}$.

Zawartość formatu podzielimy na pola przechowujące kolejno znak, wykładnik oraz mantysę liczby:

$$C36DC000_{(16)} = 1100\ 0011\ 0110\ 1101\ 1100\ 0000\ 0000\ 0000_{(2)}$$

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

Jedynka w starszym bicie formatu oznacza, że liczba jest ujemna. Wykładnik można rozkodować, odejmując od wartości kodu przesunięcie *bias* = 127:

$$E = 10000110_{(2)} - bias = 134_{(10)} - 127_{(10)} = 7_{(10)}.$$

Przy odczytywaniu z formatu mantysy należy pamiętać o dopisaniu 1 do części całkowitej liczby. W wyniku otrzymujemy dwójkową postać wykładniczą liczby:

$$-1{,}110110111 \cdot 2^7.$$

Po wykonaniu obliczeń wyznaczamy wartość dziesiętną liczby przechowywanej w formacie zmiennoprzecinkowym:

$$-1{,}110110111 \cdot 2^7 = -11101101{,}11_{(2)} = -237{,}75_{(10)}.$$

Zadanie 3.5

Jaka liczba dziesiętna zapisana jest w binarnym formacie standardowym pojedynczej precyzji, jeśli zawartość formatu wynosi $3F800000_{(16)}$*?*

Zawartość formatu podzielimy na odpowiednie pola:

$$3F800000_{(16)} = 0011\ 1111\ 1000\ 0000\ 0000\ 0000\ 0000\ 0000_{(2)}$$

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

W 8-bitowym polu wykładnika zapisany jest kod $01111111_{(2)}$, który po rozkodowaniu daje wartość wykładnika równą 0:

$$3E = 01111111_{(2)} - bias = 127_{(10)} - 127_{(10)} = 0.$$

Pole mantysy zawiera same zera, lecz nie oznacza to, że mantysa ma wartość zerową. Należy pamiętać o jedynce z części całkowitej mantysy, która jest przechowywana niejawnie i którą należy uwzględnić w zapisie liczby. W wyniku otrzymujemy następującą wartość dziesiętną liczby:

$$1{,}00 \cdot 2^0 = 1_{(10)}.$$

Zadanie 3.6

Wyznaczyć dziesiętną wartość liczby zapisanej w binarnym formacie standardowym podwójnej precyzji, jeśli zawartość formatu wynosi $BFDC800000000000_{(16)}$.

Zawartość formatu podzielimy na odpowiednie pola:

$$BFDC800000000000_{(16)} =$$
$$= 1011\ 1111\ 1101\ 1100\ 1000\ 0000\ 0000\ 0000$$
$$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000_{(2)}$$

1	01111111101	11001000000000000000000000000000000000000000000000000000

Obliczamy wartość wykładnika zakodowanego w formacie podwójnej precyzji w 11-bitowym polu z przesunięciem *bias*:

$$bias = 2^{(11-1)} - 1 = 1023$$

$$01111111101_{(2)} = 1021_{(10)}$$

$$E = 1021 - 1023 = -2_{(10)}.$$

Na podstawie zawartości formatu wyznaczamy dwójkową postać wykładniczą liczby i obliczamy jej wartość dziesiętną:

$$-1{,}11001 \cdot 2^{-2} = -0{,}0111001_{(2)} = -\frac{57}{128}_{(10)} = 0{,}4453125_{(10)}.$$

Zadanie 3.7

Przedstawić liczbę $8{,}875214 \cdot 10^{16}$ *w standardowym 32-bitowym dziesiętnym formacie zmiennoprzecinkowym decimal*32. *Do kodowania mantysy wykorzystać binarny kod NKB.*

Zamianę liczby zaczynamy od przedstawienia mantysy w postaci 7-bitowej liczby całkowitej, pamiętając o odpowiedniej zmianie wykładnika liczby:

$$8{,}875124 \cdot 10^{16} = 8875214 \cdot 10^{10}.$$

Następnie zakodujemy wykładnik liczby w kodzie z przesunięciem $bias = 101_{(10)}$:

$$E = 10_{(10)} + bias = 10_{(10)} + 101_{(10)} + 111_{(10)} = \underline{01}101111_{(\text{biased})}.$$

W naszym przykładzie mantysa będzie przechowywana w kodzie NKB, dlatego dokonujemy konwersji liczby na system dwójkowy:

$$8875214_{(10)} = \underline{1000}\ 01110110110011001110_{(2)}.$$

Ponieważ starsze 4 bity 24-bitowej mantysy przyjmują wartość $1000_{(2)} = 8_{(10)}$, kodowanie liczby odbywa się zgodnie ze schematem opisanym w drugim wierszu tabeli 3.4. Oznacza to, że pole kombinacji *C* formatu będzie miało zawartość 11*cde*, gdzie 11 – stały kod, bity *c* i *d* przechowują 2 starsze bity zakodowanego wykładnika, a bit *e* służy do kodowania starszych 4 bitów mantysy. Dla naszego przykładu {*c*, *d*} = 01, a pozostałe bity wykładnika zapisywane są do pola $W = 101111_{(2)}$.

Bit *e* koduje starszą tetradę mantysy w taki sposób, że pozwala rozróżnić dwie cyfry dziesiętne (100)*e*: przy *e* = 0, będzie to cyfra $(100)0 = 8_{(10)}$, a przy *e* = 1 – $(100)1 = 9_{(10)}$. Przy dekodowaniu liczby bit *e* jest zamieniany na sekwencję (100)*e*, gdzie cyfry umieszczone w nawiasie (100) są przechowywane w sposób niejawny, nie są zapisywane do formatu. Ponieważ w naszym przykładzie starsza tetrada mantysy ma wartość 1000, więc otrzymujemy *e* = 0. Ostatecznie pole kombinacji *C* przyjmuje wartość *C* = 11010.

Pozostałe 20 bitów mantysy zapisuje się do pola *T* kontynuacji mantysy. W ten sposób otrzymujemy następujący format zmiennoprzecinkowy: $6AF76CCE_{(16)}$.

S	*C*	*W*	*T*
0	1 1 0 1 0	1 0 1 1 1 1	0 1 1 1 0 1 1 0 1 1 0 0 1 1 0 0 1 1 1 0
	a b c d e		

Zadanie 3.8

Przedstawić liczbę $0{,}0095378248 \cdot 10^{5}$ *w standardowym 32-bitowym dziesiętnym formacie zmiennoprzecinkowym decimal*32. *Mantysę liczby zakodować w dwójkowo-dziesiętnym kodzie DPD.*

Sprowadzamy mantysę do postaci liczby całkowitej, zaokrąglając ją do 7 cyfr znaczących:

$$0{,}0095378248 \cdot 10^5 = 9537825 \cdot 10^{-4}.$$

Wykładnik w kodzie z przesunięciem ma następującą postać:

$$E = -4_{(10)} + bias = -4_{(10)} + 101_{(10)} + 97_{(10)} = \underline{01}100001_{(\text{biased})}.$$

Ponieważ starsza cyfra mantysy ma wartość $9_{(10)}$, przyjmuje się sposób kodowania przedstawiony w drugim wierszu tabeli 3.4. Wobec tego pole kombinacji ma zawartość $C = 11cde$, gdzie 11 – stały kod, bity $\{c, d\}$ przechowują dwa starsze bity wykładnika, a bit e służy do kodowania starszej cyfry dziesiętnej mantysy. Bit $e = 1$ koduje dwójkowo starszą cyfrę mantysy w postaci $(100)e = (100)1 = 9_{(10)}$, przy czym cyfry umieszczone w nawiasie (100) przechowywane są niejawnie.

Kolejnym krokiem jest kodowanie pozostałych cyfr dziesiętnych mantysy $537825_{(10)}$ w kodzie DPD. Zasady kodowania 3-cyfrowych kombinacji cyfr dziesiętnych opisane zostały w tabeli 3.7. Na podstawie tabeli zakodujemy każdą z 3-cyfrowych sekwencji cyfr dziesiętnych d_1, d_2, d_3 10-bitowym kodem DPD.

$$537_{(10)} = 0101\ 0011\ 0111_{(\text{BCD})}$$

d_1				d_2				d_3			
0	1	2	3	0	1	2	3	0	1	2	3
0	1	0	1	0	0	1	1	0	1	1	1

$$\{d_1[0], d_2[0], d_3[0]\} = 000$$

$$\{b_0, b_1, b_2\} = d_1[1{:}3] = 101$$

$$\{b_3, b_4, b_5\} = d_2[1{:}3] = 011$$

$$b_6 = 0$$

$$\{b_7, b_8, b_9\} = d_3[1{:}3] = 111$$

$$537_{(10)} = 1010110111_{(\text{DPD})}$$

$$825_{(10)} = 1000\ 0010\ 0101_{(\text{BCD})}$$

d_1				d_2				d_3			
0	1	2	3	0	1	2	3	0	1	2	3
1	0	0	0	0	0	1	0	0	1	0	1

$$\{d_1[0], d_2[0], d_3[0]\} = 100$$

$$\{b_0, b_1, b_2\} = d_3[1{:}2], d_1[3] = 100$$

$$\{b_3, b_4, b_5\} = d_2[1{:}3] = 010$$

$$b_6 = 1$$

$$\{b_7, b_8, b_9\} = 1, 0, d_3[3] = 101$$

$$825_{(10)} = 1000101101_{(DPD)}$$

Na podstawie zakodowanych komponentów tworzymy poniższy dziesiętny format zmiennoprzecinkowy:

S	*C*	*W*	*T*
0	11011	100001	1010110111 1000101101
	a b c d e		

Szesnastkowy zapis zawartości formatu zmiennoprzecinkowego to $6E1ADE2D_{(16)}$.

Zadanie 3.9

Przedstawić liczbę $-18{,}75_{(10)}$ *w standardowym* 64*-bitowym dziesiętnym formacie zmiennoprzecinkowym decimal*64*. Mantysę liczby zakodować w dwójkowo-dziesiętnym kodzie DPD.*

Na początku sprowadzamy mantysę do postaci liczby całkowitej. Format zmiennoprzecinkowy *decimal*64 pozwala przechowywać 16 cyfr dziesiętnych mantysy. W tym wypadku, gdy liczba cyfr znaczących mantysy jest niewielka, wiodące zera z lewej strony w zapisie mantysy są dopuszczalne. Otrzymujemy zatem następującą postać liczby:

$$-18{,}75_{(10)} = 0000000000001875 \cdot 10^{-2}.$$

Wykładnik w 64-bitowym formacie zmiennoprzecinkowym jest kodowany za pomocą przesunięcia $bias = 398_{(10)}$:

$$E = -2_{(10)} + bias = -2_{(10)} + 398_{(10)} = 396_{(10)} = \underline{01}10001100_{(biased)}.$$

Z uwagi na to, że starsza cyfra mantysy ma wartość 0 (znajduje się w przedziale 0–7), przyjęty zostaje sposób kodowania opisany w pierwszym wierszu tabeli 3.4. W tej sytuacji pole kombinacji *C* będzie miało zawartość {*a*, *b*, *c*, *d*, *e*}, gdzie bity *a* i *b* przechowują 2 starsze bity wykładnika {*a*, *b*} = 01, a w bitach *c*, *d* i *e* zapisuje się najbardziej znacząca cyfra mantysy – $(0)\{c, d, e\} = (0)000_{(2)}$.

Pozostała część wykładnika zapisana zostanie do 8-bitowego pola wykładnika *W* = 10001100, a reszta cyfr dziesiętnych mantysy zakodowanych w kodzie DPD wypełni pole kontynuacji mantysy *T*.

W kodzie DPD kodowane są następujące 3-cyfrowe kombinacje cyfr dziesiętnych mantysy: 000, 000, 000, 001 i 875. Grupom zerowym odpowiada kod DPD o wartości $0000000000_{(DPD)}$, grupa cyfr $001_{(10)}$ kodowana jest za pomocą kodu $0000000001_{(DPD)}$, natomiast kombinację cyfr $875_{(10)}$ zakodujemy na podstawie tabeli 3.7.

$$875_{(10)} = 1000\ 0111\ 0101_{(\mathrm{BCD})}$$

d_1				d_2				d_3			
0	1	2	3	0	1	2	3	0	1	2	3
1	0	0	0	0	1	1	1	0	1	0	1

$$\{d_1[0], d_2[0], d_3[0]\} = 100$$

$$\{b_0, b_1, b_2\} = d_3[1{:}2], d_1[3] = 100$$

$$\{b_3, b_4, b_5\} = d_2[1{:}3] = 111$$

$$b_6 = 1$$

$$\{b_7, b_8, b_9\} = 1, 0, d_3[3] = 101$$

$$875_{(10)} = 1001111101_{(\mathrm{DPD})}$$

Połączymy wszystkie wyznaczone wartości pól w 64-bitowy dziesiętny format zmiennoprzecinkowy:

S	*C*	*W*	*T*
1	0 1 0 0 0	10001100	0000000000000000000000000000000000000000000011001111101
	a b c d e		

Szesnastkowy zapis zawartości formatu ma postać $\mathrm{A23000000000067D}_{(16)}$.

Zadanie 3.10

Wyznaczyć dziesiętną wartość liczby zapisanej w standardowym 32-bitowym dziesiętnym formacie zmiennoprzecinkowym decimal32, jeśli zawartość tego formatu wynosi $699B6D6A_{(16)}$, *a mantysa zakodowana jest w binarnym kodzie NKB.*

Podzielimy zawartość formatu na odpowiednie pola:

$$699\mathrm{B6D6A}_{(16)} = 0110\ 1001\ 1001\ 1011\ 0110\ 1101\ 0110\ 1010_{(2)}$$

S	C	W	T
0	1 1 0 1 0	0 1 1 0 0 1	1 0 1 1 0 1 1 0 1 1 0 1 0 1 1 0 1 0 1 0
	a b c d e		

Znak liczby zakodowany jest w polu $S = 0$, zerowa wartość bitu znaku wskazuje, że jest to liczba dodatnia. Pole kombinacji odpowiada drugiemu wierszowi tabeli 3.4 (11*cde*), z czego wynika, że starsze 2 bity wykładnika mają wartość $\{c, d\} = 01$, a starsze 4 bity binarnej mantysy to $(100)e = 1000_{(2)}$.

Wykładnik liczby otrzymamy po połączeniu bitów $\{c, d\} = 01$ z zawartością pola W: $\{c, d, W\} = 01011001_{(\mathrm{biased})} = 89_{(10)}$. Po rozkodowaniu wyznaczamy wartość wykładnika liczby: $E = 89 - 101 = -12_{(10)}$.

Mantysę liczby wyznaczamy w wyniku konkatenacji starszej cyfry (100)e z zawartością pola T:

$$\{(100)e, T\} = 10001011011011010110101010_{(2)} = 9137514_{(10)}.$$

W ten sposób uzyskujemy dziesiętną wartość liczby:

$$9137514 \cdot 10^{-12} = 9{,}137514 \cdot 10^{-6}.$$

Zadanie 3.11

Wyznaczyć dziesiętną wartość liczby zapisanej w standardowym 64*-bitowym dziesiętnym formacie zmiennoprzecinkowym decimal*64*, jeśli zawartość tego formatu wynosi* $A234000000000069_{(16)}$*, a mantysa zakodowana jest w dwójkowo-dziesiętnym kodzie DPD.*

Zaczynamy od podziału formatu na odpowiednie pola:

$$A234000000000069_{(16)} =$$
$$= 1010\ 0010\ 0011\ 0100\ 0000\ 0000\ 0000\ 0000$$
$$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0110\ 1001_{(2)}$$

S	C	W	T
1	0 1 0 0 0	10001101	00000000000000000000000000000000000000000000001101001
	$a\ b\ c\ d\ e$		

Z zawartości pola $C = 01000_{(2)}$ możemy wywnioskować, że starsze bity wykładnika mają wartość $\{a, b\} = 01$, a starsza cyfra mantysy to $(0)cde = (0)000 = 0_{(10)}$.

Możemy wtedy obliczyć wartość wykładnika:

$$E = 0110001101_{(\text{biased})} - bias = 397_{(10)} - 398_{(10)} = -1_{(10)}.$$

W celu wyznaczenia mantysy liczby należy rozkodować pięć 10-bitowych kodów DPD, z których każdy zawiera 3 cyfry dziesiętne. Zerowe kody DPD $0000000000_{(\text{DPD})}$ odpowiadają sekwencjom 3 cyfr dziesiętnych o wartości zero – $000_{(10)}$. Aby rozkodować ostatni kod $0001101001_{(\text{DPD})}$, skorzystamy z tabeli 3.8.

b_0	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	b_9
0	0	0	1	1	0	1	0	0	1

Kombinacja bitów kontrolnych $\{b_6, b_7, b_8, b_3, b_4\} = 10011$ odpowiada drugiemu wierszowi tabeli 3.7, stąd otrzymujemy:

$$d_1 = 4b_0 + 2b_1 + b_2 = 0$$
$$d_2 = 4b_3 + 2b_4 + b_5 = 6$$

$$d_3 = 8 + b_9 = 9$$

$$\{d_1, d_2, d_3\} = 069_{(10)}.$$

W ten sposób wyznaczamy mantysę liczby z uwzględnieniem starszej cyfry zakodowanej w polu *C* i obliczamy dziesiętną wartość liczby:

$$M = 0000000000000069_{(10)}$$

$$L = 69 \cdot 10^{-1} = 6{,}9_{(10)}.$$

3.3. Zadania do samodzielnego rozwiązania

Zadanie 3.12

Przedstawić liczby w standardowym dwójkowym formacie zmiennoprzecinkowym pojedynczej precyzji (*binary*32)*:*

a) $75{,}703125_{(10)}$ b) $-296{,}1_{(10)}$ c) $7\frac{99}{1024}_{(10)}$

d) $-361{,}35_{(10)}$ e) $\frac{343}{2048}_{(10)}$ f) $-2{,}37_{(10)}$

Zadanie 3.13

Przedstawić liczby w standardowym dwójkowym formacie zmiennoprzecinkowym podwójnej precyzji (*binary*64)*:*

a) $315{,}45_{(10)}$ b) $-2{,}2_{(10)}$ c) $1{,}087890625_{(10)}$

d) $-\frac{913}{1024}_{(10)}$ e) $1295_{(10)}$ f) $-33{,}8_{(10)}$

Zadanie 3.14

Wyznaczyć dziesiętną wartość liczby zapisanej w standardowym binarnym formacie zmiennoprzecinkowym pojedynczej precyzji (*binary*32)*, jeśli zawartość formatu wynosi:*

a) $C4677800_{(16)}$ b) $3E6E0000_{(16)}$ c) $C3DA0000_{(16)}$

d) $3FB50000_{(16)}$ e) $BD9C0000_{(16)}$ f) $431BC800_{(16)}$

Zadanie 3.15

Wyznaczyć dziesiętną wartość liczby zapisanej w standardowym binarnym formacie zmiennoprzecinkowym podwójnej precyzji (*binary*64)*, jeśli zawartość formatu wynosi:*

a) $C02BB00000000000_{(16)}$ b) $3FDB600000000000_{(16)}$

c) $C09DDD8000000000_{(16)}$ d) $3FCE000000000000_{(16)}$

e) $C077B00000000000_{(16)}$ f) $BFF6780000000000_{(16)}$

Zadanie 3.16

Przedstawić liczby w dziesiętnym 32-bitowym formacie zmiennoprzecinkowym (decimal32) przy kodowaniu mantysy w dwójkowym kodzie NKB:

a) $-317{,}214 \cdot 10^{7}$ b) $8134{,}9806 \cdot 10^{-1}$

c) $-9812385{,}5 \cdot 10^{8}$ d) $598{,}54 \cdot 10^{-8}$

Zadanie 3.17

Przedstawić liczby w dziesiętnym 32-bitowym formacie zmiennoprzecinkowym (decimal32) przy kodowaniu mantysy w dwójkowo-dziesiętnym kodzie DPD:

a) $-37586{,}319 \cdot 10^{5}$ b) $-917{,}924581 \cdot 10^{-10}$

c) $-2{,}35$ d) $8{,}3958983 \cdot 10^{-15}$

Zadanie 3.18

Wyznaczyć dziesiętną wartość liczby zapisanej w standardowym dziesiętnym formacie zmiennoprzecinkowym (decimal32), jeśli mantysa zakodowana jest w binarnym kodzie NKB:

a) $A4F817C7_{(16)}$ b) $A25000B3_{(16)}$

c) $6A4860B9_{(16)}$ d) $A2725849_{(16)}$

Zadanie 3.19

Wyznaczyć dziesiętną wartość liczby zapisanej w standardowym dziesiętnym formacie zmiennoprzecinkowym (decimal32), jeśli mantysa zakodowana jest w dwójkowo-dziesiętnym kodzie DPD:

a) $EA87D6DF_{(16)}$ b) $2250041E_{(16)}$

c) $ED83667C_{(16)}$ d) $B2A7E2E2_{(16)}$

4. Dodawanie i odejmowanie liczb zmiennoprzecinkowych

4.1. Podstawy teoretyczne

Reprezentacja liczb w formatach zmiennoprzecinkowych bardzo często jest przybliżona, co wpływa na własności arytmetyki zmiennoprzecinkowej. Zapis wyników działań na liczbach zmiennoprzecinkowych często wymaga przybliżenia, zaokrąglenia, sprowadzenia wyniku do postaci najbliższej liczby maszynowej, która reprezentuje wynik w formacie ograniczonego rozmiaru. Z tego powodu zaokrąglenie wyniku stanowi jeden z etapów algorytmów wykonania operacji na liczbach zmiennoprzecinkowych. Przybliżenie wyniku jest zwyczajną operacją w arytmetyce zmiennoprzecinkowej, błąd przybliżenia zwykle nie jest sygnalizowany.

Z powodu błędów zaokrągleń arytmetyka zmiennoprzecinkowa nie jest łączna ani rozdzielna, a dokładność obliczeń może zależeć od kolejności wykonywania operacji przemiennych, takich jak dodawanie lub mnożenie.

Błędy zakresu (nadmiar lub niedomiar) są wykrywane i sygnalizowane przez jednostkę zmiennoprzecinkową za pomocą wyjątków. Po wykonaniu działań na wykładnikach liczb zawsze należy sprawdzić, czy nie wystąpiło przepełnienie pola wykładnika, co z kolei oznacza wykroczenie liczby zmiennoprzecinkowej poza zakres reprezentacji w danym formacie. W przypadku otrzymania zbyt dużej wartości wykładnika wynik reprezentowany będzie w postaci nieskończoności. Zbyt mała wartość wykładnika powoduje natomiast to, że wynik przyjmuje wartość 0.

Ważnym etapem algorytmów wykonania operacji na liczbach zmiennoprzecinkowych jest normalizacja wyniku. Podczas niej także może wystąpić błąd zakresu (nadmiar lub niedomiar), ponieważ normalizacja wiąże się z wykonaniem operacji na wykładniku wyniku, co może doprowadzić do przekroczenia zakresu reprezentacji liczby w formacie zmiennoprzecinkowym.

4.1.1. Algorytm dodawania liczb zmiennoprzecinkowych

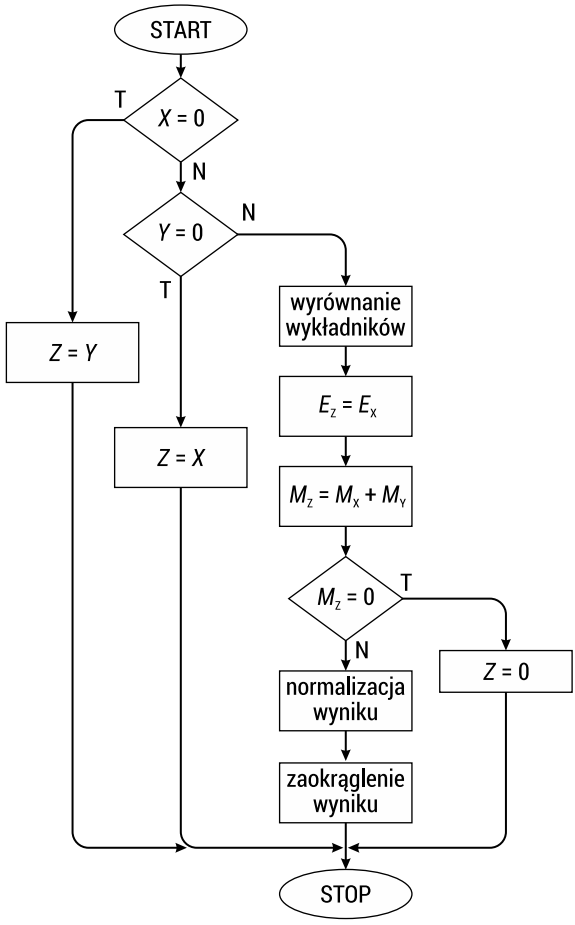
Przy wyjaśnieniu algorytmów wykonania na liczbach zmiennoprzecinkowych odnosimy się do praw arytmetyki obowiązujących dla liczb w postaci wykładniczej (naukowej). W przypadku dodawania liczb zmiennoprzecinkowych $X = M_X \cdot 2^{E_X}$

oraz $Y = M_Y \cdot 2^{E_Y}$ otrzymujemy wynik, który też należy zapisać w formacie zmiennoprzecinkowym, czyli przedstawić w postaci wykładniczej $Z = M_Z \cdot 2^{E_Z}$ ze znormalizowaną mantysą.

Algorytm dodawania dwóch liczb zmiennoprzecinkowych składa się z następujących podstawowych etapów:

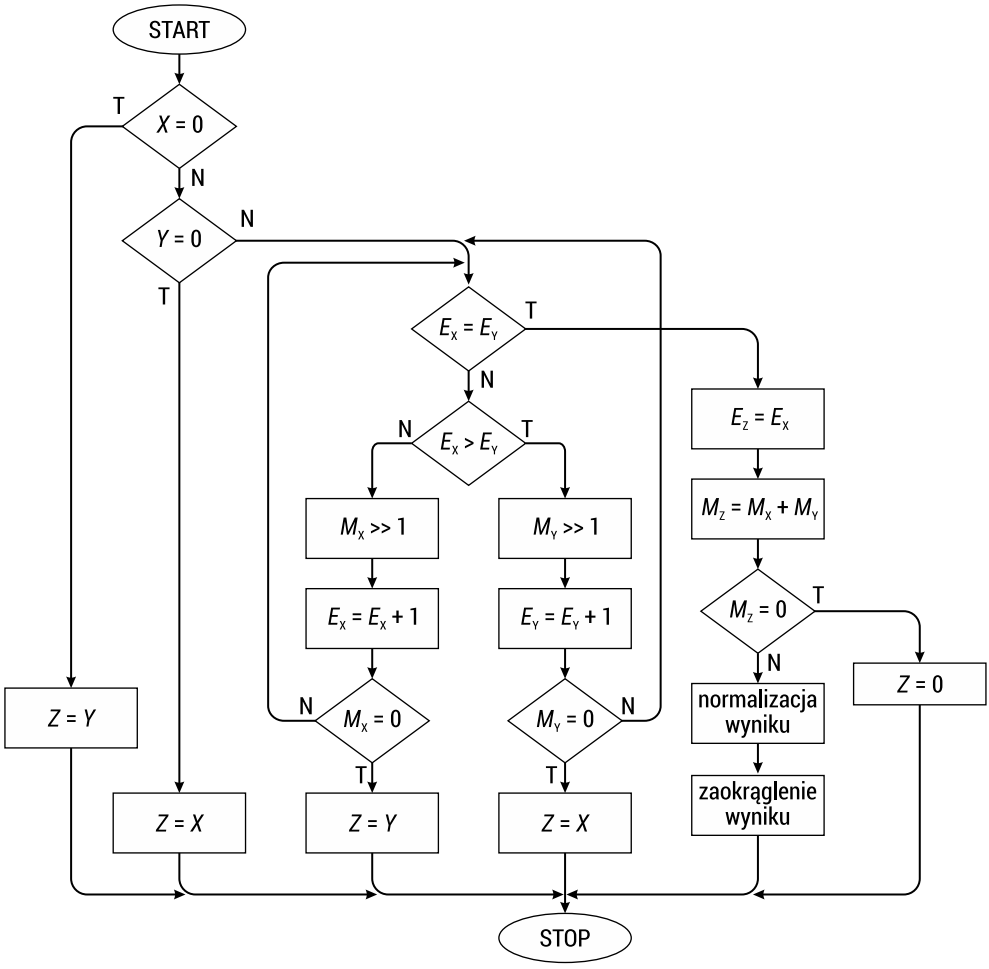
- sprowadzenie liczb do wspólnego wykładnika;
- dodawanie mantys;
- normalizacja wyniku;
- zaokrąglenie wyniku.

Uogólniony schemat blokowy algorytmu dodawania liczb zmiennoprzecinkowych przedstawiono na rys. 4.1. Z rysunku można odczytać, że algorytm zaczyna się od sprawdzenia, czy któryś ze składników, X bądź Y, ma wartość zerową, wtedy wynik przyjmuje wartość drugiego operandu i algorytm zostaje zakończony.



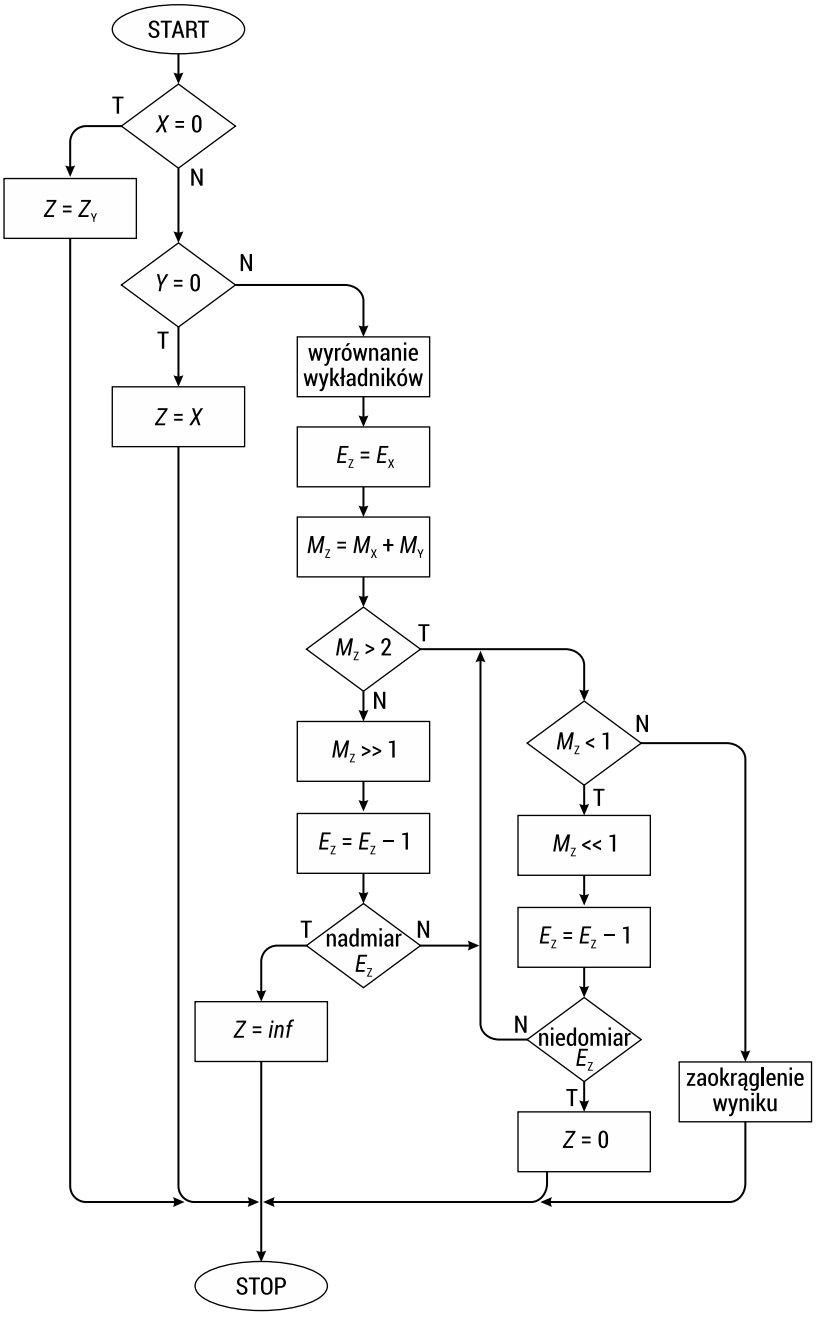
RYS. 4.1. Uogólniony schemat blokowy algorytmu dodawania liczb zmiennoprzecinkowych

Gdy składniki są niezerowe, wykonywany jest etap sprowadzenia liczb do wspólnego wykładnika. Wyrównanie wykładników realizowane jest przez denormalizację liczby mniejszej – przesunięcie mantysy liczby w prawo o tyle bitów, ile wynosi różnica wykładników liczb, z jednoczesnym zwiększeniem wartości wykładnika.



RYS. 4.2. Schemat blokowy algorytmu dodawania liczb zmiennoprzecinkowych z wyszczególnionym etapem wyrównania wykładników

Etap sprowadzania liczb do wspólnego wykładnika został dokładnie przedstawiony na rys. 4.2. Porównywanie wykładników pozwala wybrać liczbę o mniejszej wartości bezwzględnej, która będzie podlegała denormalizacji. Dla tej liczby wykonywane są powtarzające się iteracje przesunięcia mantysy o 1 bit w prawo z jednoczesnym zwiększeniem wykładnika o 1 aż do momentu wyrównania wykładników liczb. W wyniku denormalizacji liczby mniejszej może się okazać, że wielokrotne przesunięcia spowodują utratę wszystkich cyfr znaczących mantysy liczby.



RYS. 4.3. Schemat blokowy algorytmu dodawania liczb zmiennoprzecinkowych z wyszczególnionym etapem normalizacji wyniku

W tym wypadku liczba jest aproksymowana przez zero, a wynik operacji przyjmuje wartość drugiego operandu.

Po wyrównaniu wykładników można dodawać mantysy liczb $M_Z = M_X + M_Y$. Ze względu na to, że mantysy zakodowane są w kodzie ZM, dodawanie nie jest operacją prostą. Należy analizować znaki operandów, a w zależności od kombinacji znaków działanie sprowadzone zostanie do dodawania lub odejmowania. W przypadku gdy znaki operandów są różne, aby obliczyć wynik dodawania, należy od większej liczby odjąć mniejszą oraz wynikowi przypisać znak większego z operandów. Wynikiem dodawania mantys może okazać się wartość zerowa, w tej sytuacji algorytm kończy się z wynikiem $Z = 0$.

Wynik dodawania mantys może wymagać normalizacji (rys. 4.3). W przypadku przepełnienia, czyli otrzymania wartości mantysy $M_Z \geq 2$, należy przesunąć mantysę o 1 bit w prawo oraz zwiększyć wykładnik o 1. Jeśli natomiast otrzymany zostanie wynik $M_Z < 1$, normalizacja wykonuje się iteracyjnie przez przesunięcie mantysy w lewo do momentu otrzymania postaci znormalizowanej. W każdej iteracji mantysa jest przesuwana o 1 bit w lewo, a wykładnik odpowiednio jest zmniejszany o 1.

Ostatnim etapem algorytmu jest zaokrąglenie wyniku do rozmiaru formatu zmiennoprzecinkowego.

Wyjaśnimy działanie omówionego algorytmu na przykładzie dodawania konkretnych liczb zmiennoprzecinkowych.

Przykład 4.1

Obliczyć wynik operacji dodawania liczb zmiennoprzecinkowych $1100111011100_{(2)}$ *i* $1101101101101_{(2)}$ *przedstawionych w formatach składających się z następujących pól: bitu znaku,* 5*-bitowego pola wykładnika w kodzie z przesunięciem bias,* 7*-bitowego pola mantysy w kodzie ZM. Wynik operacji zapisać w formacie zmiennoprzecinkowym.*

Na początku sprowadzimy liczby do dwójkowej postaci wykładniczej. Po rozkodowaniu formatów otrzymamy następujące liczby:

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

$$E_X = 10011_{(\text{biased})} = 19 - 15 = 4_{(10)}$$

$$X = -1{,}1011100 \cdot 2^4$$

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

$$E_Y = 10110_{(\text{biased})} = 22 - 15 = 7_{(10)}$$

$$Y = -1{,}1101101 \cdot 2^7.$$

W celu wyrównania wykładników należy liczbę o mniejszej wartości bezwzględnej (czyli liczbę X) przesunąć o 3 bity w prawo:

$$X = -1{,}1011100 \cdot 2^4 = -0{,}0011011100 \cdot 2^7.$$

Po wyrównaniu wykładników można teraz dodać mantysy liczb. Wymagana jest tu analiza znaków operandów. W naszym przypadku obie liczby są ujemne, co oznacza, że wykonywane będzie dodawanie modułów mantys, a wynik będzie także liczbą ujemną.

$$\begin{array}{r} 0,0\,0\,1\,1\,0\,1\,1\,1\,0\,0 \\ +\quad 1,1\,1\,0\,1\,1\,0\,1 \\ \hline 1\,0,0\,0\,0\,1\,0\,0\,0\,1\,0\,0 \cdot 2^7 \end{array}$$

Otrzymany wynik należy znormalizować:

$$-10{,}0001000100 \cdot 2^7 = -1{,}00001000100 \cdot 2^8.$$

Zaokrąglenie wyniku polega na dodaniu jedynki do 8 bitu po przecinku, czyli do kolejnego bitu, który nie mieści się w 7-bitowym polu mantysy:

$$\begin{array}{r|l} 1,0\,0\,0\,0\,1\,0\,0 & 0\,1\,0\,0 \\ + & 1 \\ \hline 1,0\,0\,0\,0\,1\,0\,0 & 1\,1\,0\,0 \end{array}$$

Wykładnik wyniku ma wartość $E_Z = 8 + 15 = 23_{(10)} = 10111_{(\text{biased})}$. Ostatecznie otrzymujemy następujący format zmiennoprzecinkowy:

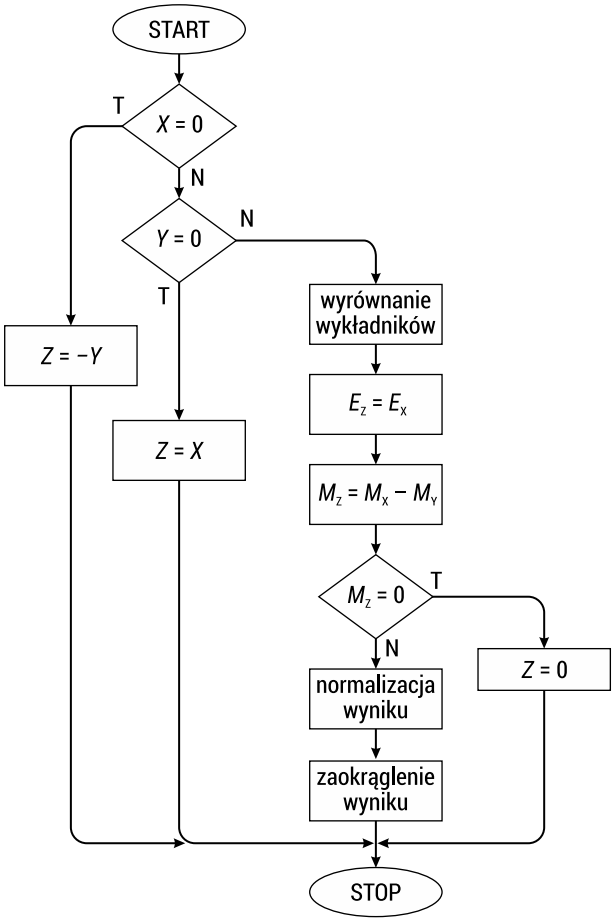
1	1 0 1 1 1	0 0 0 0 1 0 0

Ze względu na ograniczony rozmiar pola mantysy wyniku dodawania nie można dokładnie zapisać w formacie zmiennoprzecinkowym.

4.1.2. Odejmowanie liczb zmiennoprzecinkowych

Algorytm odejmowania liczb zmiennoprzecinkowych $Z = X - Y$ jest bardzo podobny do dodawania, z tą tylko różnicą, że zamiast dodawania wykonuje się odejmowanie mantys. Dla mantys zakodowanych w kodzie ZM odejmowanie jest dość skomplikowaną operacją, wymaga analizy znaków liczb. Jeśli znaki liczb są różne, operacja odejmowania sprowadza się do dodawania mantys, a znak wyniku będzie zgodny ze znakiem odjemnej. W przypadku gdy znaki liczb są takie same, wykonywana jest operacja odejmowania mniejszej (co do modułu) liczby od większej, a znak wyniku będzie zależał od wzajemnej relacji odjemnej X i odjemnika Y. Jeżeli $|X| > |Y|$, znak wyniku będzie zgodny ze znakiem odjemnej. Natomiast przy $|X| < |Y|$ – przeciwny do znaku odjemnej.

Schemat blokowy algorytmu odejmowania liczb zmiennoprzecinkowych przedstawiono na rys. 4.4. Operacja zaczyna się od sprowadzenia liczb do wspólnego wykładnika, co wykonywane jest przez denormalizację (przesunięcie w prawo) liczby o mniejszej wartości bezwzględnej przy odpowiednim zwiększeniu wartości jej wykładnika. Wyrównanie wykładników realizowane jest zgodnie ze schematem przedstawionym na rys. 4.2. Normalizacja wyniku także odbywa się analogicznie, jak w przypadku dodawania liczb (rys. 4.3).



RYS. 4.4. Schemat blokowy algorytmu odejmowania liczb zmiennoprzecinkowych

Przedstawimy działanie algorytmu odejmowania na przykładzie konkretnych liczb zmiennoprzecinkowych.

Przykład 4.2

Obliczyć wynik operacji odejmowania liczb zmiennoprzecinkowych $101100110010111_{(2)}$ *i* $101110101100110_{(2)}$ *przedstawionych w formatach składających się z następujących pól: bitu znaku,* 6*-bitowego pola wykładnika w kodzie z przesunięciem bias,* 8*-bitowego pola mantysy w kodzie ZM. Wynik operacji zapisać w formacie zmiennoprzecinkowym.*

Na początku rozkodujemy formaty zmiennoprzecinkowe, przedstawimy zapisane w nich liczby w dwójkowej postaci wykładniczej:

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

$$E_X = 011001_{(\text{biased})} = 25 - 31 = -6_{(10)}$$

$$X = -1{,}10010111 \cdot 2^{-6}$$

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

$$E_Y = 011101_{(\text{biased})} = 29 - 31 = -2_{(10)}$$

$$Y = -1{,}01100110 \cdot 2^{-2}.$$

Aby wyrównać wykładniki, liczbę o mniejszym wykładniku należy przesunąć o 4 bity w prawo:

$$X = -1{,}10010111 \cdot 2^{-6} = -0{,}000110010111 \cdot 2^{-2}.$$

Operacja odejmowania mantys daje następujący wynik:

$$\begin{array}{r} 1,0\,1\,1\,0\,0\,1\,1\,0\,0\,0\,0\,0 \\ -\quad 0,0\,0\,0\,1\,1\,0\,0\,1\,0\,1\,1\,1 \\ \hline 1,0\,1\,0\,0\,1\,1\,0\,0\,1\,0\,0\,1 \end{array} \cdot 2^{-2}$$

Otrzymany wynik jest znormalizowany, ale wymaga zaokrąglenia:

$$\begin{array}{r|l} 1,0\,1\,0\,0\,1\,1\,0\,0 & 1\,0\,0\,1 \\ + & 1 \\ \hline 1,0\,1\,0\,0\,1\,1\,0\,1 & 0\,0\,0\,1 \end{array}$$

Liczba w tym przypadku przyjmuje znak przeciwny do znaku odjemnej, ponieważ $|X| < |Y|$. Wynik działania zostanie zapisany w formacie zmiennoprzecinkowym w następującej postaci:

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

4.2. Zadania z rozwiązaniami

Zadanie 4.1

Obliczyć wynik operacji dodawania liczb zmiennoprzecinkowych $101111001110011_{(2)}$ *i* $010000011011001_{(2)}$ *zapisanych w formatach składających się z następujących pól: bitu znaku, 7-bitowego pola wykładnika w kodzie z przesunięciem bias, 7-bitowego pola mantysy w kodzie ZM. Wynik przedstawić w formacie zmiennoprzecinkowym.*

Rozkodujemy liczby zapisane w formatach zmiennoprzecinkowych:

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

$$E_X = 0111100_{(\text{biased})} = 60 - 63 = -3_{(10)}$$

$$X = -1{,}1110011 \cdot 2^{-3}$$

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

$$E_Y = 1000001_{(\text{biased})} = 65 - 63 = 2_{(10)}$$

$$Y = 1{,}1011001 \cdot 2^2.$$

Wyrównanie wykładników wykonujemy poprzez przesunięcie mniejszej liczby X o 5 bitów w prawo:

$$X = -1{,}1110011 \cdot 2^{-3} = -0{,}000011110011 \cdot 2^2.$$

Ze względu na to, że znaki liczb są różne, dodawanie mantys sprowadza się do odejmowania modułu mniejszej liczby od większej, a wynik przyjmuje znak większej liczby.

$$\begin{array}{r} 1,\,1\,0\,1\,1\,0\,0\,1\,0\,0\,0\,0\,0 \\ -\quad 0,\,0\,0\,0\,0\,1\,1\,1\,1\,0\,0\,1\,1 \\ \hline 1,\,1\,0\,1\,0\,0\,0\,1\,0\,1\,1\,0\,1 \cdot 2^2 \end{array}$$

Wynik nie wymaga normalizacji, a po zaokrągleniu do 7 cyfr po przecinku otrzymujemy:

$$\begin{array}{rl} 1,\,1\,0\,1\,0\,0\,0\,1 & 0\,1\,0\,0\,1 \\ + & 1 \\ \hline 1,\,1\,0\,1\,0\,0\,0\,1 & 1\,1\,0\,0\,1 \end{array}$$

Wynik operacji dodawania zapisany do formatu zmiennoprzecinkowego ma następującą postać:

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

Zadanie 4.2

Obliczyć wynik operacji dodawania liczb zmiennoprzecinkowych $306_{(16)}$ *i* $AFA_{(16)}$ *zapisanych w formatach składających się z następujących pól: bitu znaku,* 5*-bitowego pola wykładnika w kodzie z przesunięciem bias,* 5*-bitowego pola mantysy w kodzie ZM. Wynik obliczeń przedstawić w formacie zmiennoprzecinkowym.*

Po rozkodowaniu formatów otrzymujemy wykładnicze postaci liczb:

$$306_{(16)} = 0011\ 0000\ 0110_{(2)}$$

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

$$E_X = 01100_{(\text{biased})} = 12 - 15 = -3_{(10)}$$

$$X = 1{,}000110 \cdot 2^{-3}$$

$$\text{AFA}_{(16)} = 1010\ 1111\ 1010_{(2)}$$

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

$$E_Y = 01011_{(\text{biased})} = 11 - 15 = -4_{(10)}$$

$$Y = -1{,}111010 \cdot 2^{-4}.$$

Liczby sprowadzane są do wspólnego wykładnika poprzez przesunięcie liczby Y o 1 bit w prawo:

$$Y = -1{,}111010 \cdot 2^{-4} = -0{,}1111010 \cdot 2^{-3}.$$

Ponieważ znaki liczb są różne, operację dodawania sprowadza się do odejmowania modułów liczb:

$$\begin{array}{rr} & 1,0\,0\,0\,1\,1\,0\,0 \\ - & 0,1\,1\,1\,1\,0\,1\,0 \\ \hline & 0,0\,0\,1\,0\,0\,0\,1\,0 \cdot 2^{-3} \end{array}$$

Otrzymany wynik należy znormalizować, aby można było go zapisać do formatu zmiennoprzecinkowego. Normalizacja polega na przeniesieniu przecinka za pierwszą jedynkę w zapisie liczby:

$$Z = 0{,}001001 \cdot 2^{-3} = 1{,}001 \cdot 2^{-6}.$$

Wynik operacji dodawania przyjmuje znak większego z operandów, czyli jest liczbą dodatnią. Wykładnik kodujemy w kodzie z przesunięciem *bias*: $E = -6_{(10)} + 15_{(10)} = 01001_{(\text{biased})}$. Wynik operacji dodawania jest reprezentowany dokładnie w formacie zmiennoprzecinkowym:

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

Zadanie 4.3

Liczby $-14\frac{5}{16}_{(10)}$ *i* $-2\frac{15}{32}_{(10)}$ *zapisane są w formatach zmiennoprzecinkowych składających się z następujących pól: bitu znaku, 6-bitowego pola wykładnika w kodzie z przesunięciem bias, 7-bitowego pola mantysy w kodzie ZM. Obliczyć wynik dodawania podanych liczb i przedstawić go w formacie zmiennoprzecinkowym.*

Zapiszemy liczby w dwójkowej postaci wykładniczej:

$$X = -14\frac{5}{16}_{(10)} = -1110{,}0101_{(2)} = -1{,}1100101 \cdot 2^3$$

$$Y = -2\frac{15}{32}_{(10)} = -10{,}01111_{(2)} = -1{,}001111 \cdot 2^1.$$

Denormalizacji podlega liczba o mniejszej wartości bezwzględnej:

$$Y = -1{,}001111 \cdot 2^1 = 0{,}01001111 \cdot 2^3.$$

Następnie obliczamy wynik dodawania mantys:

$$\begin{array}{r} 1,1\,1\,0\,0\,1\,0\,1\,0 \\ +\quad 0,0\,1\,0\,0\,1\,1\,1\,1 \\ \hline 1\,0,0\,0\,0\,1\,1\,0\,0\,1 \cdot 2^3 \end{array}$$

Aby zapisać liczbę do formatu zmiennoprzecinkowego, należy ją znormalizować i zaokrąglić:

$$Z = -10{,}00011001 \cdot 2^3 = -1{,}000011001 \cdot 2^4 \approx -1{,}0000110 \cdot 2^4.$$

Wynik zapisany w formacie zmiennoprzecinkowym ma następującą postać:

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

Ze względu na ograniczony rozmiar formatu wynik został zapisany niedokładnie: $-1{,}0000110 \cdot 2^4 = -10000{,}110_{(2)} = -16\frac{3}{4}_{(10)}$ (czyli $-16\frac{24}{32}$ zamiast $-16\frac{25}{32}$).

Zadanie 4.4

Obliczyć sumę dwóch liczb zmiennoprzecinkowych $C1EDC000_{(16)}$ *i* $3DB9A000_{(16)}$ *zapisanych w standardowych formatach pojedynczej precyzji. Wynik obliczeń przedstawić w formacie zmiennoprzecinkowym.*

Na początku rozkodujemy formaty zmiennoprzecinkowe:

$$C1EDC000_{(16)} = 1100\ 0001\ 1110\ 1101\ 1100\ 0000\ 0000\ 0000_{(2)}$$

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

$$E_X = 10000011_{(\text{biased})} = 131 - 127 = 4_{(10)}$$

$$X = -1{,}110110111 \cdot 2^4$$

$$3DB9A000_{(16)} = 0011\ 1101\ 1011\ 1001\ \ 1010\ 0000\ 0000\ 0000_{(2)}$$

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

$$E_Y = 01111011_{(\text{biased})} = 123 - 127 = -4_{(10)}$$

$$Y = 1{,}0111001101 \cdot 2^{-4}.$$

Różnica wykładników wynosi 8, w związku z tym liczba Y zostanie przesunięta o 8 bitów w prawo w celu wyrównania wykładników liczb:

$$Y = 1{,}0111001101 \cdot 2^{-4} = 0{,}000000010111001101 \cdot 2^4.$$

Dodawanie liczb o różnych znakach wymaga odjęcia mniejszego modułu od większego:

$$\begin{array}{rl} & 1,1\,1\,0\,1\,1\,0\,1\,1\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0 \\ - & 0,0\,0\,0\,0\,0\,0\,0\,1\,0\,1\,1\,1\,0\,0\,1\,1\,0\,1 \\ \hline & 1,1\,1\,0\,1\,1\,0\,1\,0\,0\,0\,0\,0\,1\,1\,0\,0\,1\,1 \cdot 2^4 \end{array}$$

Rozmiar formatu pojedynczej precyzji umożliwia dokładne przedstawienie wyniku w formacie zmiennoprzecinkowym:

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

Zadanie 4.5

Obliczyć wynik dodawania dwóch liczb zmiennoprzecinkowych $2EAF2D14_{(16)}$ *i* $6A869307_{(16)}$ *przedstawionych w standardowych* 32*-bitowych dziesiętnych formatach zmiennoprzecinkowych* (*decimal*32) *przy kodowaniu mantysy w dwójkowo-dziesiętnym kodzie DPD. Wynik obliczeń przedstawić w formacie zmiennoprzecinkowym.*

Rozkodujemy liczby zmiennoprzecinkowe, zaczynając od podziału formatu na odpowiednie pola.

$$2EAF2D14_{(16)} = 0010\ 1110\ 1010\ 1111\ 0010\ 1101\ 0001\ 0100_{(2)}.$$

S	*C*	*W*	*T*
0	0 1 0 1 1	1 0 1 0 1 0	1 1 1 1 0 0 1 0 1 1 0 1 0 0 0 1 0 1 0 0
	a b c d e		

Dekodowanie zaczyna się od sprawdzenia zawartości pola kombinacji $C = 01011$. Wartość dwóch pierwszych bitów $\{a, b\} = 01$ świadczy o tym, że a i b to starsze bity wykładnika, które wraz z polem W tworzą dwójkową postać przesuniętego wykładnika:

$$\{a, b, W\} = \underline{01}101010_{(\text{biased})} = 106_{(10)}$$

$$E_X = 106 - 101 = 5_{(10)}.$$

Bity $\{c, d, e\} = 011$ przechowują starszą cyfrę mantysy – $(0)011 = 3_{(10)}$. Pozostałe cyfry mantysy zapisane są w polu kontynuacji mantysy T w postaci zakodowanej w dwójkowo-dziesiętnym kodzie DPD. Rozkodujemy poszczególne 10-bitowe kody DPD za pomocą tabeli 3.8. Starsze 10-bitowe słowo kodu DPD ma wartość:

b_0	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	b_9
1	1	1	1	0	0	1	0	1	1

Kombinacja bitów kontrolnych $\{b_6, b_7, b_8, b_3, b_4\} = 10110$ odwołuje się do trzeciego wiersza (101xx) tabeli 3.8, skąd otrzymujemy:

$$d_1 = 4b_0 + 2b_1 + b_2 = 7$$

$$d_2 = 8 + b_5 = 8$$

$$d_3 = 4b_3 + 2b_4 + b_9 = 5$$

$$\{d_1, d_2, d_3\} = 785_{(10)}.$$

Kolejny 10-bitowy kod ma wartość:

b_0	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	b_9
0	1	0	0	0	1	0	1	0	0

Zgodnie z kodem bitów kontrolnych $\{b_6, b_7, b_8, b_3, b_4\} = 01000$ dekodowanie jest wykonywane na podstawie pierwszego wiersza 0xxxx tabeli 3.8:

$$d_1 = 4b_0 + 2b_1 + b_2 = 2$$

$$d_2 = 4b_3 + 2b_4 + b_5 = 1$$

$$d_3 = 4b_7 + 2b_8 + b_9 = 4$$

$$\{d_1, d_2, d_3\} = 214_{(10)}.$$

W wyniku otrzymujemy pierwszą liczbę o następującej wartości:

$$X = 3785214_{(10)} \cdot 10^5 = 3{,}785214 \cdot 10^{11}.$$

W podobny sposób rozkodowujemy drugą liczbę z formatu zmiennoprzecinkowego.

$6A869307_{(16)} = 0110\ 1010\ 1000\ 0110\ 1001\ 0011\ 0000\ 0111_{(2)}$

S	*C*	*W*	*T*
0	1 1 0 1 0	1 0 1 0 0 0	0 1 1 0 1 0 0 1 0 0 1 1 0 0 0 0 0 1 1 1
	a b c d e		

Pole kombinacji *C* zaczyna się od bitów $\{a, b\} = 11$, co oznacza, że bity $\{c, d\} = 01$ stanowią w tym przypadku starsze bity wykładnika, a starsza cyfra mantysy zakodowana jest za pomocą bitu *e*: $(100)e = 1000_{(2)} = 8_{(10)}$. Po połączeniu bitów $\{c, d\}$ z polem wykładnika *W* otrzymujemy wykładnik liczby w kodzie z przesunięciem *bias*:

$$\{c, d, W\} = \underline{01}101000_{(\text{biased})} = 104_{(10)}$$

$$E_Y = 104 - 101 = 3_{(10)}.$$

Dekodowanie pola *T* wykonujemy za pomocą tabeli 3.8.

b_0	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	b_9
0	1	1	0	1	0	0	1	0	0

Kombinacja bitów kontrolnych $\{b_6, b_7, b_8, b_3, b_4\} = 01001$ kieruje nas do pierwszego wiersza tabeli 3.8 (0xxxx), skąd zgodnie z algorytmem dekodowania otrzymujemy:

$$d_1 = 4b_0 + 2b_1 + b_2 = 3$$
$$d_2 = 4b_3 + 2b_4 + b_5 = 2$$
$$d_3 = 4b_7 + 2b_8 + b_9 = 4$$
$$\{d_1, d_2, d_3\} = 324_{(10)}.$$

W podobny sposób rozkodowujemy drugi 10-bitowy blok kodu DPD:

b_0	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	b_9
1	1	0	0	0	0	0	1	1	1

$$\{b_6, b_7, b_8, b_3, b_4\} = 01100$$
$$d_1 = 4b_0 + 2b_1 + b_2 = 6$$
$$d_2 = 4b_3 + 2b_4 + b_5 = 0$$
$$d_3 = 4b_7 + 2b_8 + b_9 = 7$$
$$\{d_1, d_2, d_3\} = 607_{(10)}.$$

Po połączeniu wszystkich wyznaczonych komponentów otrzymujemy wykładniczą postać drugiej liczby:

$$Y = 8324607_{(10)} \cdot 10^3 = 8{,}324607 \cdot 10^9.$$

Algorytm dodawania liczb zaczyna się od sprowadzenia liczb do wspólnego wykładnika poprzez denormalizację liczby o mniejszej wartości bezwzględnej:

$$X = 3{,}785214 \cdot 10^{11}$$
$$Y = 8{,}324607 \cdot 10^9 = 0{,}08324607 \cdot 10^{11}.$$

Ponieważ znaki liczb są zgodne, wykonywane jest dodawanie mantys:

$$\begin{array}{r} 3,7\,8\,5\,2\,1\,4\,0\,0 \\ +\quad 0,0\,8\,3\,2\,4\,6\,0\,7 \\ \hline 3,8\,6\,8\,4\,6\,0\,0\,7 \end{array} \cdot 10^{11}$$

Po zaokrągleniu otrzymujemy:

$$Z = 3{,}86846007 \cdot 10^{11} \approx 3{,}868460 \cdot 10^{11}.$$

Aby zapisać wynik do formatu zmiennoprzecinkowego, sprowadzamy go do postaci liczby całkowitej:

$$Z = 3{,}868460 \cdot 10^{11} = 3868460 \cdot 10^5.$$

Wykładnik w kodzie z przesunięciem *bias* przyjmuje następującą postać:

$$E_Z = 5 + 101 = 106_{(2)} = \underline{01}101010_{(2)}.$$

Dwa starsze (podkreślone) bity wykładnika zapisane zostaną do pola kombinacji *C*, a pozostałe bity wypełnią 6-bitowe pole wykładnika *W* = 101010.

Ponieważ starsza cyfra mantysy 3 znajduje się w przedziale 0–7, zostanie ona zakodowana w polu kombinacji *C* na 3 bitach {*c*, *d*, *e*} = (0)011, a całe pole kombinacji z uwzględnieniem 2 bitów wykładnika {*a*, *b*} = 01 będzie miało wartość *C* = {*a*, *b*, *c*, *d*, *e*} = 01011.

Następnie kodujemy pozostałe cyfry mantysy w kodzie DPD:

$$868_{(10)} = 1000\ 0110\ 1000_{(BCD)}$$

d_1				d_2				d_3			
0	1	2	3	0	1	2	3	0	1	2	3
1	0	0	0	0	1	1	0	1	0	0	0

$$\{d_1[0], d_2[0], d_3[0]\} = 101$$

$$\{b_0, b_1, b_2\} = d_2[1{:}2], d_1[3] = 110$$

$$\{b_3, b_4, b_5\} = 0, 1, d_2[3] = 010$$

$$b_6 = 1$$

$$\{b_7, b_8, b_9\} = 1, 1, d_3[3] = 110$$

$$868_{(10)} = 1100101110_{(DPD)}$$

$$460_{(10)} = 0100\ 0110\ 0000_{(BCD)}$$

d_1				d_2				d_3			
0	1	2	3	0	1	2	3	0	1	2	3
0	1	0	0	0	1	1	0	0	0	0	0

$$\{d_1[0], d_2[0], d_3[0]\} = 000$$

$$\{b_0, b_1, b_2\} = d_1[1{:}3] = 100$$

$$\{b_3, b_4, b_5\} = d_2[1{:}3] = 110$$

$$b_6 = 0$$

$$\{b_7, b_8, b_9\} = d_3[1{:}3] = 000$$

$$460_{(10)} = 1001100000_{(DPD)}.$$

Po połączeniu wszystkich wyznaczonych komponentów otrzymujemy format zmiennoprzecinkowy przechowujący wynik operacji dodawania:

S	*C*	*W*	*T*
0	0 1 0 1 1	1 0 1 0 1 0	1 1 0 0 1 0 1 1 1 0 1 0 0 1 1 0 0 0 0
	a b c d e		

Szesnastkowy zapis zawartości formatu zmiennoprzecinkowego to $2EACBA60_{(16)}$.

Zadanie 4.6

Obliczyć wynik operacji odejmowania liczb zmiennoprzecinkowych $C35_{(16)}$ *i* $3B9_{(16)}$ *zapisanych w formatach składających się z następujących pól: bitu znaku,* 5*-bitowego pola wykładnika w kodzie z przesunięciem bias,* 6*-bitowego pola mantysy w kodzie ZM. Wynik przedstawić w formacie zmiennoprzecinkowym.*

Liczby przedstawimy w dwójkowej postaci wykładniczej:

$$C35_{(16)} = 1100\ 0011\ 0101_{(2)}$$

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

$$E_X = 10000_{(\text{biased})} = 16 - 15 = 1_{(10)}$$

$$X = -1{,}110101 \cdot 2^1$$

$$3B9_{(16)} = 0011\ 1011\ 1001_{(2)}$$

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

$$E_Y = 01110_{(\text{biased})} = 14 - 15 = -1_{(10)}$$

$$Y = 1{,}111001 \cdot 2^{-1}.$$

Denormalizacji podlega liczba Y:

$$Y = 1{,}111001 \cdot 2^{-1} = -0{,}01111001 \cdot 2^1.$$

Ponieważ znaki liczb są różne, operacja odejmowania sprowadza się do dodawania mantys liczb, przy czym znak wyniku jest zgodny ze znakiem odjemnej, czyli wynik będzie liczbą ujemną.

$$\begin{array}{r} 1,\ 1\ 1\ 0\ 1\ 0\ 1\ 0\ 0 \\ +\quad 0,\ 0\ 1\ 1\ 1\ 1\ 0\ 0\ 1 \\ \hline 1\ 0,\ 0\ 1\ 0\ 0\ 1\ 1\ 0\ 1 \cdot 2^1 \end{array}$$

Po normalizacji i zaokrągleniu wyniku otrzymujemy:

$$-10{,}01001101 \cdot 2^1 = -1{,}001001101 \cdot 2^2 \approx -1{,}001010 \cdot 2^2.$$

Format zmiennoprzecinkowy z wynikiem operacji odejmowania ma następującą postać:

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

Zadanie 4.7

Obliczyć wynik odejmowania dwóch liczb $3DD80000_{(16)}$ *i* $3FEB0000_{(16)}$ *zapisanych w standardowych formatach zmiennoprzecinkowych pojedynczej precyzji. Wynik obliczeń przedstawić w formacie zmiennoprzecinkowym.*

W podanych formatach zapisane są następujące liczby:

$$3DD80000_{(16)} = 0011\ 1101\ 1101\ 1000\ 0000\ 0000\ 0000\ 0000_{(2)}$$

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

$$E_X = 01111011_{(\text{biased})} = 123 - 127 = -4_{(10)}$$

$$X = 1{,}1011 \cdot 2^{-4}$$

$$3FEB0000_{(16)} = 0011\ 1111\ 1110\ 1011\ 0000\ 0000\ 0000\ 0000_{(2)}$$

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

$$E_Y = 01111111_{(\text{biased})} = 127 - 127 = 0$$

$$Y = 1{,}1101011 \cdot 2^{0}.$$

Algorytm odejmowania zaczyna się od wyrównania wykładników liczb:

$$X = 1{,}1011 \cdot 2^{-4} = 0{,}00011011 \cdot 2^{0}.$$

W sytuacji gdy znaki liczb są takie same, wykonywana jest operacja odejmowania modułu mantysy mniejszej liczby od większej. Znak wyniku w tym wypadku będzie przeciwny do znaku odjemnej X, ponieważ $|X| < |Y|$, czyli wynik będzie ujemny:

$$\begin{array}{rl} & 1,1\,1\,0\,1\,0\,1\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0 \\ - & 0,0\,0\,0\,1\,1\,0\,1\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0 \\ \hline & 1,1\,0\,1\,1\,1\,0\,1\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0 \cdot 2^{0} \end{array}$$

Liczba jest znormalizowana i nie wymaga zaokrąglenia, jest reprezentowana dokładnie w poniższym formacie zmiennoprzecinkowym:

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

4.3. Zadania do samodzielnego rozwiązania

Zadanie 4.8

Obliczyć wynik dodawania podanych liczb zmiennoprzecinkowych zapisanych w formatach składających się z następujących pól: bitu znaku, wykładnika w kodzie z przesunięciem bias, mantysy w kodzie ZM (*rozmiary pól wykładnika oraz mantysy są podane w poszczególnych zadaniach*)*:*

a) $10111001110111101_{(2)}$ i $1100001110110110 1_{(2)}$, 6-bitowy wykładnik, 10-bitowa mantysa;

b) $001001110111011101_{(2)}$ i $100111100101101100_{(2)}$, 5-bitowy wykładnik, 12-bitowa mantysa;

c) $00111111111011101_{(2)}$ i $00111101110111011_{(2)}$, 7-bitowy wykładnik, 9-bitowa mantysa;

d) $CECE_{(16)}$ i $3323_{(16)}$, 5-bitowy wykładnik, 10-bitowa mantysa;

e) $BEBB1_{(16)}$ i $C0E79_{(16)}$, 7-bitowy wykładnik, 12-bitowa mantysa;

f) $A7ACC_{(16)}$ i $32EE5_{(16)}$, 5-bitowy wykładnik, 14-bitowa mantysa.

Zadanie 4.9

Obliczyć wynik dodawania podanych liczb zapisanych w standardowych formatach zmiennoprzecinkowych pojedynczej precyzji:

a) $40F97000_{(16)}$ i $3EDEDC00_{(16)}$;

b) $C1595E00_{(16)}$ i $3DB67B40_{(16)}$;

c) $BA5D7300_{(16)}$ i $BCFBDBC0_{(16)}$;

d) $3E4EBCC0_{(16)}$ i $C13ABB40_{(16)}$.

Zadanie 4.10

Obliczyć wynik dodawania podanych liczb zapisanych w standardowych 32*-bitowych dziesiętnych formatach zmiennoprzecinkowych z mantysą zakodowaną w kodzie DPD:*

a) $E9C20FDB_{(16)}$ i $A5E56656_{(16)}$;

b) $E9CECA6D_{(16)}$ i $21B04696_{(16)}$;

c) $2647D61E_{(16)}$ i $327C3DC5_{(16)}$;

d) $2280AD71_{(16)}$ i $AAB778D4_{(16)}$.

Zadanie 4.11

Obliczyć wynik odejmowania podanych liczb zmiennoprzecinkowych zapisanych w formatach składających się z następujących pól: bitu znaku, wykładnika w kodzie z przesunięciem bias, mantysy w kodzie ZM (*rozmiary pól wykładnika oraz mantysy są podane w poszczególnych zadaniach*)*:*

a) $110001110111010_{(2)}$ i $001111111011100_{(2)}$, 5-bitowy wykładnik, 9-bitowa mantysa;

b) $1011100101101011101_{(2)}$ i $1100011110001101001_{(2)}$, 6-bitowy wykładnik, 12-bitowa mantysa;

c) $00111000111011001110111_{(2)}$ i $10111010101111001010111_{(2)}$, 7-bitowy wykładnik, 14-bitowa mantysa;

d) $01001111011101010111101_{(2)}$ i $01000100110110111001101_{(2)}$, 6-bitowy wykładnik, 15-bitowa mantysa.

Zadanie 4.12

Obliczyć wynik odejmowania podanych liczb zapisanych w standardowych formatach zmiennoprzecinkowych pojedynczej precyzji:

a) $B96F35D0_{(16)}$ i $3A76EF40_{(16)}$;

b) $C76F15CC_{(16)}$ i $C45BA360_{(16)}$;

c) $3F7CDD5A_{(16)}$ i $BDF6E1A8_{(16)}$;

d) $415BC5C0_{(16)}$ i $444EDC54_{(16)}$.

5. Mnożenie oraz dzielenie liczb zmiennoprzecinkowych

5.1. Podstawy teoretyczne

Mnożenie i dzielenie liczb zmiennoprzecinkowych odbywa się na zasadach wykonywania operacji na liczbach w postaci wykładniczej. Operacja mnożenia sprowadza się do dodawania wykładników i mnożenia mantys. Dzielenie polega natomiast na odejmowaniu wykładników oraz dzieleniu mantys liczb zmiennoprzecinkowych.

Algorytmy obu działań zaczynają się od wykonania operacji na wykładnikach, ponieważ wykrycie błędu zakresu na tym etapie pozwala zrezygnować z dalszych czynności algorytmu. Wynik operacji w tej sytuacji jest przedstawiany w postaci nieskończoności (w przypadku nadmiaru) lub zera (jeśli wystąpił niedomiar).

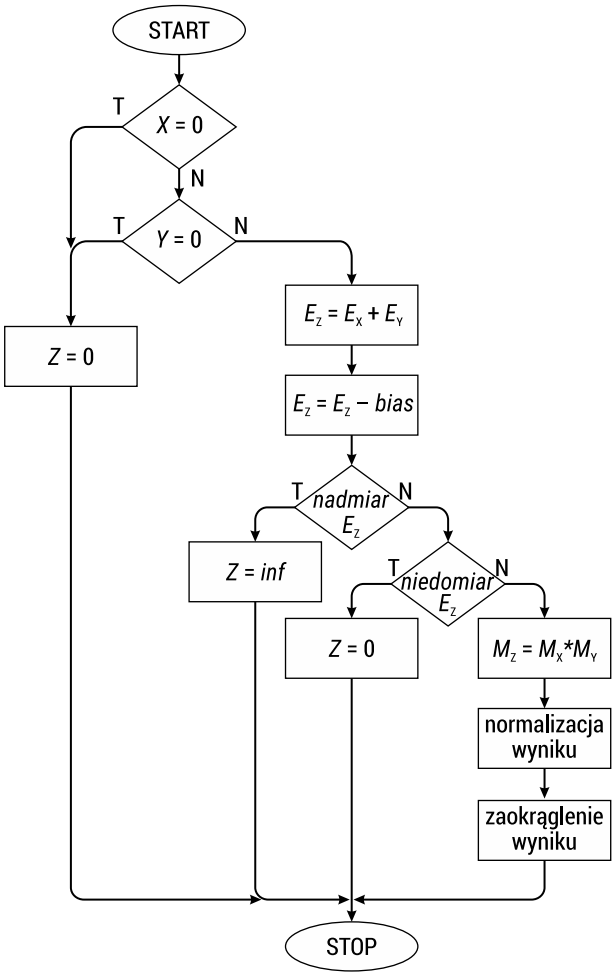
5.1.1. Algorytm mnożenia liczb zmiennoprzecinkowych

Schemat blokowy algorytmu mnożenia liczb zmiennoprzecinkowych przedstawiono na rys. 5.1. W wyniku mnożenia dwóch liczb $X = M_X \cdot 2^{E_X}$ i $Y = M_Y \cdot 2^{E_Y}$ otrzymujemy liczbę $Z = M_Z \cdot 2^{E_Z} = \left(M_X \cdot M_Y\right) \cdot 2^{E_X + E_Y}$.

Jak widzimy na rys. 5.1, algorytm mnożenia zaczyna się od sprawdzenia, czy któryś z mnożników ma wartość zerową, wtedy algorytm można szybko zakończyć z wynikiem $Z = 0$. W przypadku niezerowych operandów pierwszym etapem algorytmu jest dodawanie wykładników liczb. Należy zwrócić uwagę na sposób wykonania tej operacji na wykładnikach zakodowanych w kodzie z przesunięciem *bias* – po binarnym dodawaniu kodów potrzebna jest korekta otrzymanego wyniku. Ponieważ każdy z wykładników zwiększony jest o wartość przesunięcia *bias*, to otrzymana suma będzie zawierała w sobie dwa przesunięcia:

$$E_X + bias + E_Y + bias = (E_X + E_Y) + 2 \cdot bias.$$

Z tego powodu od wyniku binarnego dodawania kodów wykładników należy odjąć jedno przesunięcie *bias*.



RYS. 5.1. Algorytm mnożenia liczb zmiennoprzecinkowych

Przykład 5.1

Obliczyć sumę liczb $10011010_{(biased)}$ *i* $01110100_{(biased)}$ *zakodowanych w kodzie z przesunięciem bias.*

Dodawanie binarne zakodowanych liczb daje następujący wynik:

$$\begin{array}{r} 1\,0\,0\,1\,1\,0\,1\,0 \\ +\quad 0\,1\,1\,1\,0\,1\,0\,0 \\ \hline 1\,0\,0\,0\,0\,1\,1\,1\,0 \end{array}$$

Aby skorygować wynik odejmujemy od niego przesunięcie *bias*:

$$\begin{array}{r} 1\,0\,0\,0\,0\,1\,1\,1\,0 \\ -\quad 0\,1\,1\,1\,1\,1\,1\,1 \\ \hline 1\,0\,0\,0\,1\,1\,1\,1 \end{array}$$

W wyniku otrzymujemy poprawnie zakodowaną sumę liczb – $10001111_{(biased)}$.

Operacje na wykładnikach są zawsze sprawdzane pod względem wystąpienia błędu nadmiaru lub niedomiaru. Jeśli wykładnik wychodzi poza zakres reprezentacji, algorytm mnożenia od razu zostaje zakończony, a wynik przyjmuje postać nieskończoności lub zera – w zależności od tego, czy wystąpił nadmiar, czy niedomiar (rys. 5.1).

Jeśli wykładnik nie wykracza poza zakres reprezentacji, kolejnym krokiem algorytmu jest mnożenie mantys. Operacja ta jest wykonywana jako zwykłe binarne mnożenie modułów liczb. W przypadku mnożenia znormalizowanych mantys wynik mnożenia zawsze będzie się zawierał w przedziale [1, 4). Normalizacja wyniku w tym przypadku będzie polegała na przesunięciu liczby o 1 bit w prawo z jednoczesnym zwiększeniem wykładnika o 1.

Ostatnim etapem algorytmu jest zaokrąglenie iloczynu do najbliższej liczby maszynowej reprezentującej wynik w formacie zmiennoprzecinkowym.

Przykład 5.2

Obliczyć wynik mnożenia liczb $110101111001_{(2)}$ *i* $001100110110_{(2)}$ *przedstawionych w formatach zmiennoprzecinkowych składających się z trzech pól: bitu znaku,* 5*-bitowego wykładnika w kodzie z przesunięciem bias,* 6*-bitowej mantysy w kodzie ZM. Wynik przedstawić w formacie zmiennoprzecinkowym.*

Podzielimy formaty na odpowiednie pola:

1	10101	111001	← X
0	01100	110110	← Y

Operacja mnożenia liczb zaczyna się od dodawania wykładników:

$$\begin{array}{r} 10101 \\ +\ 01100 \\ \hline 00001 \\ +\ 10001 \\ \hline 10010 \end{array}$$

Odejmowanie przesunięcia zastępujemy w tych obliczeniach równoważną operacją dodania liczby przeciwnej do przesunięcia w kodzie U2: $01111_{(2)} = 10001_{(U2)}$. Obliczona wartość wykładnika może jeszcze ulec zmianie w procesie normalizacji wyniku.

Kolejnym etapem algorytmu jest mnożenie mantys:

$$\begin{array}{r} 1,111001 \\ *\ \ 1,110110 \\ \hline 1111001 \\ 1111001 \\ 1111001 \\ 1111001 \\ +\ \ 1111001 \\ \hline 11,011111000110 \end{array}$$

Należy pamiętać o dopisaniu jedynek do części całkowitej mantys, są to jedynki przechowywane niejawnie, nie są zapisane w formacie zmiennoprzecinkowym.

Widzimy, że otrzymany wynik mnożenia wymaga normalizacji, która wykonywana jest poprzez przesunięcie mantysy o 1 bit w prawo i zwiększenie o 1 wykładnika wyniku:

$$11{,}011111000110 = 1{,}1011111000110 \cdot 2^1.$$

$$\begin{array}{r} 1\,0\,0\,1\,0 \\ +\qquad\quad 1 \\ \hline 1\,0\,0\,1\,1 \end{array}$$

Oprócz tego mantysa wyniku podlega zaokrągleniu do wartości najbliższej liczby maszynowej:

$$\begin{array}{r|l} 1,\,1\,0\,1\,1\,1\,1 & 1\,0\,0\,0\,1\,1\,0 \\ + & 1 \\ \hline 1,\,1\,1\,0\,0\,0\,0 & 0 \end{array}$$

Ostatecznie otrzymamy następujący format zmiennoprzecinkowy z wynikiem operacji mnożenia:

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

5.1.2. Dzielenie liczb zmiennoprzecinkowych

Schemat blokowy algorytmu dzielenia liczb zmiennoprzecinkowych przedstawiono na rys. 5.2. Dla dwóch liczb $X = M_X \cdot 2^{E_X}$ i $Y = M_Y \cdot 2^{E_Y}$ operacja dzielenia sprowadza się do odejmowania wykładników oraz dzielenia mantys liczb:

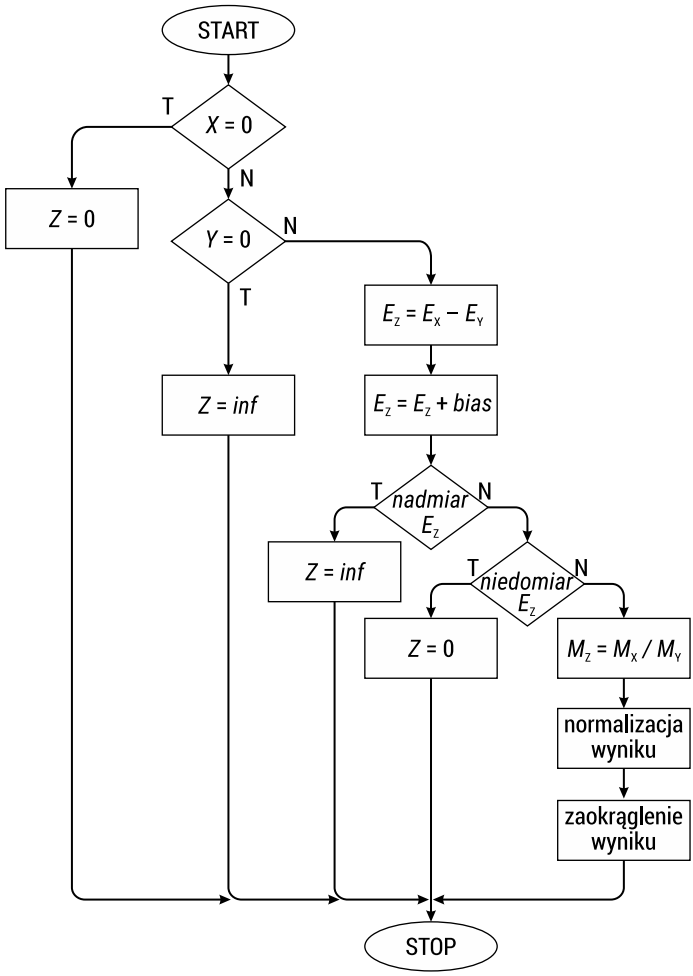
$$Z = M_Z \cdot 2^{E_Z} = (M_X \cdot 2^{E_X}) / (M_Y \cdot 2^{E_Y}) = \left(M_X / M_Y\right) \cdot 2^{E_X - E_Y}.$$

Na początku następuje sprawdzenie, czy operandy nie są zerowe. Jeśli dzielna jest zerem $X = 0$, algorytm można zakończyć z wynikiem Z = 0. W przypadku gdy dzielnik ma wartość zerową $Y = 0$, wynikiem operacji będzie nieskończoność. Tu warto zwrócić uwagę na to, że dzielenie przez 0 w arytmetyce zmiennoprzecinkowej jest operacją dozwoloną, przy czym wynik takiej operacji wyrażany jest nieskończonością.

Podobnie jak w przypadku mnożenia dzielenie liczb zmiennoprzecinkowych zaczyna się od wykonania operacji na wykładnikach. Pozwala to już na początku sprawdzić, czy wynik zawiera się w zakresie reprezentacji liczb. Jeżeli wykryty zostanie błąd zakresu, dalsze operacje na liczbach nie są wykonywane.

Zwróćmy uwagę na operację odejmowania wykładników zakodowanych w kodzie z przesunięciem *bias*. Jest ona wykonywana w dwóch etapach: najpierw zwykłe binarne odejmowanie zakodowanych liczb, a następnie zwiększenie wyniku o wartość przesunięcia *bias*. Poprawka taka jest wymagana, ponieważ po odejmowaniu binarnym skracają się przesunięcia zawarte w kodach: $E_X + bias - (E_Y + bias) = E_X - E_Y$.

W wyniku takiej operacji otrzymujemy różnicę, która nie jest zakodowana, więc należy ją zwiększyć o wartość przesunięcia.



RYS. 5.2. Schemat blokowy algorytmu dzielenia liczb zmiennoprzecinkowych

Przykład 5.3

Obliczyć różnicę liczb $10011011_{(biased)}$ *i* $01111001_{(biased)}$ *zakodowanych w kodzie z przesunięciem bias.*

Po binarnym odejmowaniu kodów wynik obliczeń należy zwiększyć o wartość przesunięcia *bias* w celu uzyskania poprawnie zakodowanego wyniku operacji:

$$\begin{array}{r} 1\,0\,0\,1\,1\,0\,1\,1 \\ -\;\underline{0\,1\,1\,1\,1\,0\,0\,1} \\ 0\,0\,1\,0\,0\,0\,1\,0 \\ +\;\underline{0\,1\,1\,1\,1\,1\,1\,1} \\ 1\,0\,1\,0\,0\,0\,0\,1_{\,(biased)} \end{array}$$

Operacje wykonywane na wykładnikach liczb zawsze są sprawdzane na wypadek wystąpienia błędu nadmiaru lub niedomiaru (rys. 5.2). W przypadku otrzymania zbyt dużej wartości wykładnika, wynik przedstawiany jest w postaci nieskończoności, niedomiar sprawia natomiast, że wynik przyjmuje wartość zerową.

Ponieważ mantysy liczb przechowywane są w formatach zmiennoprzecinkowych w postaci modułów zakodowanych dwójkowo, operacja dzielenia mantys może być wykonana za pomocą zwykłego algorytmu dzielenia binarnego. Przy dzieleniu znormalizowanych mantys wynik może wymagać normalizacji w postaci przesunięcia mantysy o 1 bit w lewo wraz ze zmniejszeniem o 1 wartości wykładnika.

Przykład 5.4

Obliczyć wynik dzielenia liczb $11000010110110_{(2)}$ *i* $00111111010000_{(2)}$ *przedstawionych w formatach zmiennoprzecinkowych składających się z trzech pól: bitu znaku,* 6*-bitowego wykładnika w kodzie z przesunięciem bias,* 7*-bitowej mantysy w kodzie ZM. Wynik przedstawić w formacie zmiennoprzecinkowym.*

Dwójkowe zawartości formatów zmiennoprzecinkowych podzielimy na pola odpowiedniej długości:

1	1 0 0 0 0 1	0 1 1 0 1 1 0	← X
0	0 1 1 1 1 1	1 0 1 0 0 0 0	← Y

Operacja dzielenia liczb zaczyna się od odejmowania wykładników, przy czym uwzględniana jest wymagana poprawka:

$$
\begin{array}{rl}
 & 1\,0\,0\,0\,0\,1 \\
- & \underline{0\,1\,1\,1\,1\,1} \\
 & 0\,0\,0\,0\,1\,0 \\
+ & \underline{0\,1\,1\,1\,1\,1} \quad \leftarrow \text{poprawka} \\
 & 1\,0\,0\,0\,0\,1_{(\text{biased})}
\end{array}
$$

Otrzymana wartość wykładnika może ulec zmianie przy ewentualnej normalizacji wyniku. Dzielenie mantys można wykonać za pomocą dowolnego algorytmu dzielenia liczb binarnych – na przykład metody porównawczej. Na początku porównywana jest pierwsza reszta częściowa (dzielna) z dzielnikiem, a w wyniku porównywania wyznaczana jest cyfra zapisywana do części całkowitej wyniku. Każda kolejna iteracja rozpoczyna się od przesunięcia reszty częściowej o 1 bit w lewo i porównania przesuniętej reszty z dzielnikiem. Jeśli odjęcie dzielnika od bieżącej reszty częściowej nie jest możliwe, do wyniku dopisywane jest 0, w przeciwnym wypadku – 1.

```
          0,1 1 1
    1, 0 1 1 0 1 1 :1, 1 0 1
    1, 1 0 1          q0 = 0
  1 0, 1 1 0 1 1      ← przesunięcie
–   1, 1 0 1          −MY, q−1 = 1
    1, 0 0 1 1 1
  1 0, 0 1 1 1        ← przesunięcie
–   1, 1 0 1          −MY, q−2 = 1
    0, 1 1 0 1
  0 1, 1 0 1          ← przesunięcie
–   1, 1 0 1          −MY, q−3 = 1
    0, 0 0 0
```

Dzielenie jest kontynuowane aż do uzyskania zerowej reszty częściowej lub otrzymania wymaganej dokładności reprezentacji wyniku.

Uzyskany wynik dzielenia należy znormalizować $0{,}111 = 1{,}11 \cdot 2^{-1}$, na skutek czego zmienia się wartość wykładnika:

```
  1 0 0 0 1
–       1
  1 0 0 0 0
```

Ostateczny wynik operacji dzielenia zapisujemy w formacie zmiennoprzecinkowym:

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

5.2. Zadania z rozwiązaniami

Zadanie 5.1

Obliczyć wynik operacji mnożenia liczb $B533_{(16)}$ i $CED9_{(16)}$ przedstawionych w formatach zmiennoprzecinkowych składających się z następujących pól: bitu znaku, 6-bitowego wykładnika w kodzie z przesunięciem bias, 9-bitowej mantysy w kodzie ZM. Wynik zapisać w formacie zmiennoprzecinkowym.

Zaczynamy od podziału formatów na poszczególne pola:

$$X = B533_{(16)} = 1011\ 0101\ 0011\ 0011_{(2)}$$

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

$$Y = CED9_{(16)} = 1100\ 1110\ 1101\ 1001_{(2)}$$

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

Pierwszym etapem algorytmu mnożenia jest dodawanie wykładników:

$$\begin{array}{r} 0\,1\,1\,0\,1\,0 \\ +\quad 1\,0\,0\,1\,1\,1 \\ \hline 0\,0\,0\,0\,0\,1 \\ +\quad 1\,0\,0\,0\,0\,1 \\ \hline 1\,0\,0\,0\,1\,0 \end{array}$$

Następnie wykonujemy binarne mnożenie mantys uzupełnionych jedynkami w części całkowitej:

$$\begin{array}{r} 1,1\,0\,0\,1\,1\,0\,0\,1\,1 \\ *\quad 1,0\,1\,1\,0\,1\,1\,0\,0\,1 \\ \hline 1\,1\,0\,0\,1\,1\,0\,0\,1\,1 \\ 1\,1\,0\,0\,1\,1\,0\,0\,1\,1 \\ 1\,1\,0\,0\,1\,1\,0\,0\,1\,1 \\ 1\,1\,0\,0\,1\,1\,0\,0\,1\,1 \\ 1\,1\,0\,0\,1\,1\,0\,0\,1\,1 \\ +\quad 1\,1\,0\,0\,1\,1\,0\,0\,1\,1 \\ \hline 10,0\,1\,0\,0\,0\,1\,1\,1\,0\,0\,0\,0\,0\,1\,1\,1\,0\,1\,1 \end{array}$$

Po normalizacji i zaokrągleniu wyniku binarnego mnożenia otrzymujemy następującą postać mantysy:

$$10{,}010001110000111011 = 1{,}0010001110000111011 \cdot 2^1 \approx 1{,}001000111 \cdot 2^1.$$

Normalizacja prowadzi do zwiększenia wykładnika liczby:

$$\begin{array}{r} 1\,0\,0\,0\,1\,0 \\ +\qquad\quad 1 \\ \hline 1\,0\,0\,0\,1\,1 \end{array}$$

$$E_Z = 100011_{(\text{biased})}.$$

W rezultacie otrzymujemy następujący format zmiennoprzecinkowy reprezentujący wynik operacji mnożenia:

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

Zadanie 5.2

Obliczyć wynik operacji mnożenia liczb $5A5BD000_{(16)}$ *i* $B99D4000_{(16)}$ *przedstawionych w standardowych formatach zmiennoprzecinkowych pojedynczej precyzji. Wynik zapisać w formacie zmiennoprzecinkowym.*

Podzielimy formaty pojedynczej precyzji na odpowiednie pola standardowego rozmiaru:

$$X = 5A5BD000_{(16)} = 0101\ 1010\ 0101\ 1011\ 1101\ 0000\ 0000\ 0000_{(2)}$$

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

$$Y = \text{B99D4000}_{(16)} = 1011\ 1001\ 1001\ 1101\ 0100\ 0000\ 0000\ 0000_{(2)}$$

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

Najpierw dodawane są wykładniki liczb:

$$\begin{array}{rr} & 1\,0\,1\,1\,0\,1\,0\,0 \\ + & 0\,1\,1\,1\,0\,0\,1\,1 \\ \hline & 0\,0\,1\,0\,0\,1\,1\,1 \\ - & 0\,1\,1\,1\,1\,1\,1\,1 \\ \hline & 1\,0\,1\,0\,1\,0\,0\,0 \end{array}$$

Następnie wykonujemy binarne mnożenie mantys, uwzględniamy tu tylko znaczące cyfry liczb:

$$\begin{array}{rr} & 1,\,1\,0\,1\,1\,0\,1\,1\,1\,1\,0\,1 \\ * & 1,\,0\,0\,1\,1\,1\,0\,1\,0\,1 \\ \hline & 1\,1\,0\,1\,1\,0\,1\,1\,1\,1\,0\,1 \\ & 1\,1\,0\,1\,1\,0\,1\,1\,1\,1\,0\,1 \\ & 1\,1\,0\,1\,1\,0\,1\,1\,1\,1\,0\,1 \\ & 1\,1\,0\,1\,1\,0\,1\,1\,1\,1\,0\,1 \\ & 1\,1\,0\,1\,1\,0\,1\,1\,1\,1\,0\,1 \\ + & 1\,1\,0\,1\,1\,0\,1\,1\,1\,1\,0\,1 \\ \hline & 1\,0,\,0\,0\,0\,1\,1\,1\,0\,0\,0\,0\,0\,1\,0\,1\,1\,0\,0\,0\,0\,1 \end{array}$$

Po normalizacji mantysy okazuje się, że wynik nie wymaga zaokrąglenia, może on być przedstawiony dokładnie w standardowym formacie pojedynczej precyzji:

$$10{,}00011100000101100001 = 1{,}00011100000101100001 \cdot 2^1.$$

W rezultacie normalizacji mantysy wykładnik wyniku zostaje zwiększony o 1:

$$\begin{array}{rr} & 1\,0\,1\,0\,1\,0\,0\,0 \\ + & \qquad\qquad 1 \\ \hline & 1\,0\,1\,0\,1\,0\,0\,1_{\text{(biased)}} \end{array}$$

Ostatecznie otrzymujemy następujący format zmiennoprzecinkowy reprezentujący wynik operacji mnożenia:

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

Zadanie 5.3

Obliczyć wynik mnożenia liczb $6EF76169_{(16)}$ *i* $A2300175_{(16)}$ *przedstawionych w standardowych dziesiętnych* 32*-bitowych formatach zmiennoprzecinkowych* (*decimal*32). *Wynik zapisać w formacie zmiennoprzecinkowym.*

Rozkodujemy formaty zmiennoprzecinkowe:

$$X = \text{6EF76169}_{(16)} = 0110\ 1110\ 1111\ 0111\ 0110\ 0001\ 0110\ 1001_{(2)}$$

S	*C*	*W*	*T*
0	1 1 0 1 1	1 0 1 1 1 1	0 1 1 1 0 1 1 0 0 0 0 1 0 1 1 0 1 0 0 1
	a b c d e		

Ze względu na stan bitów $\{a, b\} = 11$ zakodowana wartość wykładnika liczby X wynosi $\{c, d, W\} = 01101111_{(\text{biased})}$. Starsza cyfra mantysy to $(100)e = 1001_{(2)} = 9_{(10)}$. Pozostałe cyfry mantysy rozkodujemy z 20-bitowego pola T za pomocą tabeli 3.8.

b_0	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	b_9
0	1	1	1	0	1	1	0	0	0

$$\{b_6, b_7, b_8, b_3, b_4\} = 10010$$

$$d_1 = 4b_0 + 2b_1 + b_2 = 3$$

$$d_2 = 4b_3 + 2b_4 + b_5 = 5$$

$$d_3 = 8 + b_9 = 8$$

$$\{d_1, d_2, d_3\} = 358_{(10)}$$

b_0	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	b_9
0	1	0	1	1	0	1	0	0	1

$$\{b_6, b_7, b_8, b_3, b_4\} = 10011$$

$$d_1 = 4b_0 + 2b_1 + b_2 = 2$$

$$d_2 = 4b_3 + 2b_4 + b_5 = 6$$

$$d_3 = 8 + b_9 = 9$$

$$\{d_1, d_2, d_3\} = 269_{(10)}$$

Mantysa pierwszej liczby ma wartość $9358269_{(10)}$.
Analogicznie rozkodujemy drugą liczbę:

$$Y = \text{A2300175}_{(16)} = 1010\ 0010\ 0011\ 0000\ 0000\ 0001\ 0111\ 0101_{(2)}$$

S	*C*	*W*	*T*
1	0 1 0 0 0	1 0 0 0 1 1	0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 0 1 0 1
	a b c d e		

Kombinacja bitów $\{a, b\} = 01$ w polu C oznacza, że wykładnik liczby jest obliczany według schematu $\{a, b, W\} = 01100011_{(\text{biased})}$, a starsza cyfra mantysy wyznaczana jest jako $(0)cde = 0000_{(2)} = 0_{(10)}$.

Pozostałe cyfry mantysy otrzymujemy z pola T. Pierwszy 10-bitowy blok kodu DPD $0000000000_{(\text{DPD})}$ odpowiada 3 cyfrom dziesiętnym o wartości 0: $0000000000_{(\text{DPD})} = 000_{(10)}$. Drugi 10-bitowy blok rozkodujemy za pomocą tabeli 3.8.

b_0	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	b_9
0	1	0	1	1	1	0	1	0	1

$$\{b_6, b_7, b_8, b_3, b_4\} = 01011$$

$$d_1 = 4b_0 + 2b_1 + b_2 = 2$$

$$d_2 = 4b_3 + 2b_4 + b_5 = 7$$

$$d_3 = 4b_7 + 2b_8 + b_9 = 5$$

$$\{d_1, d_2, d_3\} = 275_{(10)}$$

Mantysa drugiej liczby ma wartość $0000275_{(10)}$.

Algorytm mnożenia liczb rozpoczyna się od dodawania wykładników. Należy pamiętać, że wykładniki są zakodowane za pomocą przesunięcia *bias* = $101_{(10)}$, więc poprawka po operacji dodawania polega na odjęciu od sumy wartości $101_{(10)}$:

$$01101111_{(2)} + 01100011_{(2)} - 101_{(10)} = 111_{(10)} + 99_{(10)} - 101_{(10)} = 109_{(10)} = 01101101_{(\text{biased})}.$$

Następnym krokiem algorytmu jest mnożenie mantys:

$$\begin{array}{rl} 9{,}3\,5\,8\,2\,6\,9 & \cdot\ 10^6 \\ * \quad 2{,}7\,5 & \cdot\ 10^2 \\ \hline 4\,6\,7\,9\,1\,3\,4\,5 & \\ 6\,5\,5\,0\,7\,8\,8\,3 & \\ +\ 1\,8\,7\,1\,6\,5\,3\,8 & \\ \hline 2\,5{,}7\,3\,5\,2\,3\,9\,7\,5 & \cdot\ 10^8 \end{array}$$

Po normalizacji i zaokrągleniu otrzymujemy następującą postać mantysy wyniku:

$$25{,}73523975 \cdot 10^8 = 2{,}573523975 \cdot 10^9 \approx 2{,}573524 \cdot 10^9 = 2573524 \cdot 10^3.$$

Normalizacja mantysy prowadzi do zwiększenia wartości wykładnika o 3:

$$\begin{array}{r} 0\,1\,1\,0\,1\,1\,0\,1 \\ +\quad\quad\quad 1\,1 \\ \hline 0\,1\,1\,1\,0\,0\,0\,0_{\,(\text{biased})} \end{array}$$

Sześć mniej znaczących cyfr mantysy należy zakodować w kodzie DPD. W tym celu wykorzystamy schemat kodowania opisany w tabeli 3.7.

$$573_{(10)} = 0101\ 0111\ 0011_{(\text{BCD})}$$

d_1				d_2				d_3			
0	1	2	3	0	1	2	3	0	1	2	3
0	1	0	1	0	1	1	1	0	0	1	1

$$\{d_1[0], d_2[0], d_3[0]\} = 000$$

$$\{b_0, b_1, b_2\} = d_1[1{:}3] = 101$$

$$\{b_3, b_4, b_5\} = d_2[1{:}3] = 111$$

$$b_6 = 0$$

$$\{b_7, b_8, b_9\} = d_3[1:3] = 011$$

$$573_{(10)} = 1011110011_{(DPD)}$$

$$524_{(10)} = 0101\ 0010\ 0100_{(BCD)}$$

d_1				d_2				d_3			
0	1	2	3	0	1	2	3	0	1	2	3
0	1	0	1	0	0	1	0	0	1	0	0

$$\{d_1[0], d_2[0], d_3[0]\} = 000$$

$$\{b_0, b_1, b_2\} = d_1[1:3] = 101$$

$$\{b_3, b_4, b_5\} = d_2[1:3] = 010$$

$$b_6 = 0$$

$$\{b_7, b_8, b_9\} = d_3[1:3] = 100$$

$$524_{(10)} = 1010100100_{(DPD)}$$

Ponieważ starsza cyfra mantysy $2_{(10)}$ znajduje się w przedziale 0–7, pole kombinacji przyjmuje wartość $C = 01010_{(2)}$. Po połączeniu wszystkich wyznaczonych komponentów otrzymujemy następujący format zmiennoprzecinkowy:

S	*C*	*W*	*T*
1	0 1 0 1 0	1 1 0 0 0 0	1 0 1 1 1 1 0 0 1 1 1 0 1 0 1 0 0 1 0 0
	a b c d e		

Zadanie 5.4

Obliczyć wynik dzielenia liczb $11001110110110_{(2)}$ *i* $10110101101000_{(2)}$ *przedstawionych w formatach zmiennoprzecinkowych składających się z następujących pól: bitu znaku,* 5*-bitowego wykładnika w kodzie z przesunięciem bias,* 8*-bitowej mantysy w kodzie ZM. Wynik zapisać w formacie zmiennoprzecinkowym.*

Podzielimy formaty na pola odpowiedniego rozmiaru, wyznaczymy wykładniki oraz mantysy liczb:

1	1 0 0 1 1	1 0 1 1 0 1 1 0	← *X*
1	0 1 1 0 1	0 1 1 0 1 0 0 0	← *Y*

Obliczymy różnicę wykładników w kodzie z przesunięciem *bias*:

$$\begin{array}{r} 1\,0\,0\,1\,1 \\ -\ 0\,1\,1\,0\,1 \\ \hline 0\,0\,1\,1\,0 \\ +\ 0\,1\,1\,1\,1 \\ \hline 1\,0\,1\,0\,1 \end{array}_{(biased)}$$

Wykonując operację dzielenia mantys, obliczamy tyle cyfr wyniku, ile zmieści się w formacie zmiennoprzecinkowym, oraz dodatkowo jedną cyfrę wykorzystywaną do zaokrąglenia liczby:

1, 0 0 1 1 0 1 1 1 0 1	
1, 1 0 1 1 0 1 1 0 : 1, 0 1 1 0 1	
− 1, 0 1 1 0 1	$q_0 = 1$
0, 0 1 0 0 1 1 1 0	
0, 1 0 0 1 1 1 0 0	← przesunięcie, $q_{-1} = 0$
1, 0 0 1 1 1 0 0 0	← przesunięcie, $q_{-2} = 0$
1 0, 0 1 1 1 0 0 0 0	← przesunięcie
− 1, 0 1 1 0 1	$-M_Y$, $q_{-3} = 1$
1, 0 0 0 0 1 0 0 0	
1 0, 0 0 0 1 0 0 0 0	← przesunięcie
− 1, 0 1 1 0 1	$-M_Y$, $q_{-4} = 1$
0, 1 0 1 0 1 0 0 0	
1, 0 1 0 1 0 0 0 0	← przesunięcie, $q_{-5} = 0$
1 0, 1 0 1 0 0 0 0 0	← przesunięcie
− 1, 0 1 1 0 1	$-M_Y$, $q_{-6} = 1$
1, 0 0 1 1 1 0 0 0	
1 0, 0 1 1 1 0 0 0 0	← przesunięcie
− 1, 0 1 1 0 1	$-M_Y$, $q_{-7} = 1$
1, 0 0 0 0 1 0 0 0	
1 0, 0 0 0 1 0 0 0 0	← przesunięcie
− 1, 0 1 1 0 1	$-M_Y$, $q_{-8} = 1$
0, 1 0 1 0 1 0 0 0	
1, 0 1 0 1 0 0 0 0	← przesunięcie, $q_{-9} = 0$

W wyniku dzielenia otrzymujemy znormalizowaną mantysę 1,001101110, która po zaokrągleniu przyjmuje postać 1,00110111.

Na podstawie wykonanych obliczeń możemy przedstawić wynik operacji dzielenia w formacie zmiennoprzecinkowym:

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

Zadanie 5.5

Obliczyć wynik operacji dzielenia liczb $C3A53400_{(16)}$ *i* $41DA0000_{(16)}$ *przedstawionych w standardowych formatach zmiennoprzecinkowych pojedynczej precyzji. Wynik zapisać w formacie zmiennoprzecinkowym.*

Zaczynamy od podziału formatów na pola przechowujące znak, wykładnik oraz mantysę liczb:

$$X = C3A53400_{(16)} = 1100\ 0011\ 1010\ 0101\ 0011\ 0100\ 0000\ 0000_{(2)}$$

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

$$Y = 41DA0000_{(16)} = 0100\ 0001\ 1101\ 1010\ 0000\ 0000\ 0000\ 0000_{(2)}$$

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

Pierwszym etapem algorytmu dzielenia liczb jest odejmowanie wykładników:

$$\begin{array}{rl} & 1\,0\,0\,0\,0\,1\,1\,1 \\ - & 1\,0\,0\,0\,0\,0\,1\,1 \\ & 0\,0\,0\,0\,0\,1\,0\,0 \\ + & 0\,1\,1\,1\,1\,1\,1\,1 \quad \leftarrow \text{poprawka} \\ & 1\,0\,0\,0\,0\,0\,1\,1 \end{array}$$

Kolejny krok algorytmu to dzielenie mantys liczb zmiennoprzecinkowych:

$$\begin{array}{rll} & 0{,}1\,1\,0\,0\,0\,0\,1 & \\ \hline 1{,}0\,1\,0\,0\,1\,0\,1\,0\,0\,1\,1\,0\,1 & : 1{,}1\,0\,1\,1\,0\,1 & \\ 1{,}1\,0\,1\,1\,0\,1 & & q_0 = 0 \\ \hline 1\,0{,}1\,0\,0\,1\,0\,1\,0\,0\,1\,1\,0\,1\,0 & & \leftarrow \text{przesunięcie} \\ - \quad 1{,}1\,0\,1\,1\,0\,1 & & -M_Y,\ q_{-1} = 1 \\ \hline 0{,}1\,1\,1\,0\,0\,0\,0\,0\,1\,1\,0\,1\,0 & & \\ 1{,}1\,0\,0\,0\,0\,0\,0\,1\,1\,0\,1\,0\,0 & & \leftarrow \text{przesunięcie} \\ - \quad 1{,}1\,0\,1\,1\,0\,1 & & -M_Y,\ q_{-2} = 1 \\ \hline 0{,}0\,0\,0\,0\,1\,1\,0\,1\,1\,0\,1\,0\,0 & & \\ 0{,}0\,0\,0\,1\,1\,0\,1\,1\,0\,1\,0\,0\,0 & & \leftarrow \text{przesunięcie},\ q_{-3} = 0 \\ 0{,}0\,0\,1\,1\,0\,1\,1\,0\,1\,0\,0\,0\,0 & & \leftarrow \text{przesunięcie},\ q_{-4} = 0 \\ 0{,}0\,1\,1\,0\,1\,1\,0\,1\,0\,0\,0\,0\,0 & & \leftarrow \text{przesunięcie},\ q_{-5} = 0 \\ 0{,}1\,1\,0\,1\,1\,0\,1\,0\,0\,0\,0\,0\,0 & & \leftarrow \text{przesunięcie},\ q_{-6} = 0 \\ 1{,}0\,1\,1\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0 & & \leftarrow \text{przesunięcie} \\ - \quad 1{,}1\,0\,1\,1\,0\,1 & & -M_Y,\ q_{-7} = 1 \\ \hline 0{,}0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0 & & \end{array}$$

W wyniku dzielenia otrzymujemy mantysę 0,1100001, którą należy znormalizować przed zapisem do formatu zmiennoprzecinkowego:

$$0{,}1100001 = 1{,}100001 \cdot 2^{-1}.$$

Zapis ten oznacza, że wykładnik wyniku należy zmniejszyć o 1:

$$\begin{array}{rl} & 1\,0\,0\,0\,0\,0\,1\,1 \\ - & \qquad\qquad 1 \\ \hline & 1\,0\,0\,0\,0\,0\,1\,0_{\text{(biased)}} \end{array}$$

Wynik obliczeń zapisujemy w standardowym formacie zmiennoprzecinkowym pojedynczej precyzji:

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

Szesnastkowa reprezentacja zawartości formatu wynosi $C1420000_{(16)}$.

5.3. Zadania do samodzielnego rozwiązania

Zadanie 5.6

Obliczyć wynik mnożenia podanych liczb zmiennoprzecinkowych zapisanych w formatach składających się z następujących pól: bitu znaku, wykładnika w kodzie z przesunięciem bias, mantysy w kodzie ZM (*rozmiary pól wykładnika oraz mantysy są podane w poszczególnych zadaniach*)*:*

a) $1011011101010_{(2)}$ i $1101101011011_{(2)}$, 5-bitowy wykładnik, 7-bitowa mantysa;

b) $01010011011100101_{(2)}$ i $10110000010101101_{(2)}$, 6-bitowy wykładnik, 10-bitowa mantysa;

c) $010010111011010_{(2)}$ i $001110011101101_{(2)}$, 7-bitowy wykładnik, 7-bitowa mantysa;

d) $110111000110110101_{(2)}$ i $001101101001101001_{(2)}$, 6-bitowy wykładnik, 11-bitowa mantysa;

e) $D8D3_{(16)}$ i $B88D_{(16)}$, 7-bitowy wykładnik, 8-bitowa mantysa;

f) $B9B8D_{(16)}$ i $3C1DA_{(16)}$, 8-bitowy wykładnik, 11-bitowa mantysa.

Zadanie 5.7

Obliczyć wynik mnożenia podanych liczb zapisanych w standardowych formatach zmiennoprzecinkowych pojedynczej precyzji:

a) $475A9000_{(16)}$ i $B6371000_{(16)}$;

b) $BD9A2C00_{(16)}$ i $B9267000_{(16)}$;

c) $BEDAC000_{(16)}$ i $47BD2000_{(16)}$;

d) $33651C00_{(16)}$ i $47386E00_{(16)}$.

Zadanie 5.8

Obliczyć wynik mnożenia podanych liczb zapisanych w standardowych 32-bitowych dziesiętnych formatach zmiennoprzecinkowych z mantysą zakodowaną w kodzie DPD:

a) $EAE2DFDC_{(16)}$ i $2190917D_{(16)}$;

b) $EDA4D5FB_{(16)}$ i $A211C6A5_{(16)}$;

c) $6AE74BDB_{(16)}$ i $EE68558E_{(16)}$;

d) $6D57C5DC_{(16)}$ i $22700559_{(16)}$.

Zadanie 5.9

Obliczyć wynik dzielenia podanych liczb zmiennoprzecinkowych zapisanych w formatach składających się z następujących pól: bitu znaku, wykładnika w kodzie z przesunięciem

bias, mantysy w kodzie ZM (*rozmiary pól wykładnika oraz mantysy są podane w poszczególnych zadaniach*)*:*

a) $110111010110101_{(2)}$ i $011010011001101_{(2)}$, 6-bitowy wykładnik, 8-bitowa mantysa;

b) $0101011101101001101_{(2)}$ i $1100110010001010100_{(2)}$, 7-bitowy wykładnik, 11-bitowa mantysa;

c) $10110011001101_{(2)}$ i $11011101110110_{(2)}$, 5-bitowy wykładnik, 8-bitowa mantysa;

d) $00111000001011011_{(2)}$ i $10111010101110000_{(2)}$, 7-bitowy wykładnik, 9-bitowa mantysa.

Zadanie 5.10

Obliczyć wynik dzielenia podanych liczb zapisanych w standardowych formatach zmiennoprzecinkowych pojedynczej precyzji:

a) $C9F76700_{(16)}$ i $44048000_{(16)}$;

b) $BB66C9C0_{(16)}$ i $BC9F4000_{(16)}$;

c) $41833940_{(16)}$ i $C66E4000_{(16)}$;

d) $3C4DE970_{(16)}$ i $42596000_{(16)}$;

e) $BEA99E84_{(16)}$ i $44B02800_{(16)}$.

6. Obliczanie pierwiastka kwadratowego z liczb zmiennoprzecinkowych

6.1. Podstawy teoretyczne

Pierwiastkowanie na równi z czterema pozostałymi działaniami jest jedną z podstawowych operacji arytmetycznych. Pierwiastek kwadratowy jest szeroko stosowany w licznych aplikacjach, a jego realizacja sprzętowa ma wiele wspólnego z dzieleniem.

Ponieważ w formatach zmiennoprzecinkowych można przechowywać wyłącznie liczby rzeczywiste, obliczanie pierwiastka kwadratowego z liczby ujemnej w arytmetyce zmiennoprzecinkowej jest operacją niedozwoloną. Liczba pierwiastkowana powinna być dodatnia, inaczej wynik operacji zostanie przedstawiony w formacie zmiennoprzecinkowym w postaci nieliczby (*NaN*).

W przypadku gdy wykładnik liczby zmiennoprzecinkowej jest parzysty, wyznaczenie pierwiastka jest proste – wystarczy obliczyć pierwiastek z mantysy oraz podzielić wykładnik przez 2:

$$\sqrt{M \cdot 2^{2n}} = \sqrt{M} \cdot 2^{n}.$$

Sytuacja się komplikuje w przypadku, gdy wykładnik jest nieparzysty, ponieważ nie możemy w łatwy sposób spierwiastkować liczby 2^{2n+1}. W takim wypadku możemy wykonać przesunięcie mantysy tak, aby otrzymać parzysty wykładnik. Najlepiej jest zmniejszyć wykładnik o 1 (poprzez przesunięcie przecinka o jedno miejsce w prawo), dzięki czemu w wyniku pierwiastkowania otrzymamy już znormalizowaną liczbę.

6.1.1. Obliczanie pierwiastka kwadratowego metodą pisemną

Wyjaśnienie algorytmu obliczania pierwiastka kwadratowego metodą pisemną [8] (ang. *pencil-and-paper*) zaczniemy od powtórzenia tego algorytmu na przykładzie pierwiastkowania liczb dziesiętnych.

Przykład 6.1

Obliczyć pierwiastek kwadratowy z liczby $473{,}0625_{(10)}$ *za pomocą metody pisemnej.*

Na początku należy pogrupować cyfry dziesiętne pierwiastkowanej liczby w pary, zaczynając od przecinka dziesiętnego w obie strony:

q_1	q_0,	q_{-1}	q_{-2}
$\sqrt{\ 0\ 4}$	7 3,	0 6	2 5

Wynik będzie zapisywany nad linią tak, że nad każdą parą cyfr dziesiętnych pojawi się jedna cyfra q_i wyniku. Miejsce przecinka dziesiętnego w wyniku pierwiastkowania odpowiada miejscu przecinka w pierwiastkowanej liczbie.

Obliczenia zaczynamy od pierwszej pary cyfr z lewej strony, w kolejnych iteracjach algorytmu operacje będą wykonywane na kolejnych parach cyfr. Oznacza to, że cyfry wyniku są wyznaczane w kolejności od najstarszej do najmłodszej. W każdej iteracji wyznaczamy częściowy pierwiastek $q^{(i)}$, gdzie i – numer iteracji algorytmu. Przed rozpoczęciem algorytmu przyjmuje się $q^{(0)} = 0$.

W każdej i-tej iteracji algorytmu wyznaczamy kolejną cyfrę q_i wyniku pierwiastkowania w taki sposób, żeby była to największa cyfra, która spełnia poniższy warunek:

$$y^{(i)} = \left(10 \cdot 2q^{(i-1)} + q_j\right) \cdot q_j \le r^{(i)} \tag{6.1}$$

gdzie $q^{(i-1)}$ – pierwiastek częściowy wyznaczony w poprzedniej iteracji algorytmu; q_j – kolejna cyfra dopisywana do wyniku w i-tej iteracji algorytmu (j jest numerem pozycji cyfry w zapisie wyniku pierwiastkowania); $r^{(i)}$ – bieżąca reszta częściowa.

W każdej iteracji algorytmu dobieramy taką cyfrę q_i wyniku, która jest największą cyfrą spełniającą warunek (6.1), odejmujemy obliczoną liczbę $y^{(i)}$ od $r^{(i)}$ i w taki sposób wyznaczamy kolejną resztę częściową $r^{(i+1)}$. Cyfrę q_i zapisujemy nad bieżącą parą cyfr dziesiętnych.

W taki sposób w pierwszej iteracji algorytmu w naszym przykładzie analizujemy pierwszą parę cyfr dziesiętnych 04 i szukamy cyfry q_1 jako największej cyfry spełniającej warunek (6.1). Ponieważ w pierwszej iteracji $q^{(0)} = 0$, warunek (6.1) upraszcza się do postaci $y^{(1)} = q_1 \cdot q_1 \le r^{(1)}$, więc szukamy takiej cyfry q_1, dla której $(q_1)^2 \le 4$.

2	q_0,	q_{-1}	q_{-2}			
$\sqrt{\ 0\ 4}$	7 3,	0 6	2 5			$q^{(0)} = 0$
0 4				$y^{(1)} = 4$	$q_1 = 2$	$q^{(1)} = 2$
0 0	7 3			$r^{(2)}$		

W wyniku otrzymujemy $q_1 = 2$, $y^{(1)} = q_1 \cdot q_1 = 4$.

W drugiej iteracji algorytmu szukamy największej cyfry q_0, która spełnia warunek:

$$(20 \cdot q^{(1)} + q_0) \cdot q_0 \le r^{(2)}$$

$$(40 + q_0) \cdot q_0 \le 73.$$

Cyfrę q_0 można znaleźć, dzieląc resztę $r^{(2)}$ przez $20 \cdot q^{(1)}$, po czym należy sprawdzić, czy otrzymana cyfra jest największą cyfrą spełniającą warunek (6.1). W naszym przypadku otrzymujemy $q_0 = 1$, $y^{(2)} = (40 + 1) \cdot 1 = 41$, $q^{(1)} = 21$, a następnie wyznaczamy kolejną resztę częściową:

2 ¦ 1, ¦ q_{-1} ¦ q_{-2}				
$\sqrt{}$ 04 ¦ 73, ¦ 06 ¦ 25				$q^{(0)} = 0$
04		$y^{(1)} = 4$	$q_1 = 2$	$q^{(1)} = 2$
00 73		$r^{(2)}$		
41		$y^{(2)} = (40 + 1) \cdot 1 = 41$	$q_0 = 1$	$q^{(2)} = 21$
32 06		$r^{(3)}$		

W podobny sposób postępujemy z każdą kolejną parą cyfr dziesiętnych:

2 ¦ 1, ¦ 7 ¦ 5				
$\sqrt{}$ 04 ¦ 73, ¦ 06 ¦ 25				$q^{(0)} = 0$
04		$y^{(1)} = 4$	$q_1 = 2$	$q^{(1)} = 2$
00 73		$r^{(2)}$		
41		$y^{(2)} = (40 + 1) \cdot 1 = 41$	$q_0 = 1$	$q^{(2)} = 21$
32 06		$r^{(3)}$		
29 89		$y^{(3)} = (420 + 7) \cdot 7 = 2989$	$q_{-1} = 7$	$q^{(3)} = 217$
2 17 25		$r^{(4)}$		
2 17 25		$y^{(4)} = (4340 + 5) \cdot 5 = 21725$	$q_{-2} = 5$	$q^{(4)} = 2175$
0 00 00				

Po otrzymaniu zerowej reszty częściowej działanie algorytmu zostaje zakończone. Obliczony pierwiastek z liczby $473,0625_{(10)}$ wynosi $q = 21,75_{(10)}$.

Przy zastosowaniu tego algorytmu do pierwiastkowania liczb dwójkowych obliczenia znacznie się upraszczają. Ponieważ w systemie dwójkowym występują tylko dwie cyfry 0 i 1, wyznaczanie kolejnej cyfry q_j wyniku pierwiastkowania sprowadza się do sprawdzenia, czy wartość $y^{(i)}$ dla $q_j = 1$ jest zawarta w bieżącej reszcie częściowej $r^{(i)}$. Warunek (6.1) zostaje przekształcony dla systemu dwójkowego do postaci [8]:

$$y^{(i)} = \left(2 \cdot 2q^{(i-1)} + q_j\right) \cdot q_j \le r^{(i)} \tag{6.2}$$

Ze wzoru (6.2) wynika, że dla cyfry $q_j = 1$ powinien zostać spełniony warunek $4q^{(i-1)} + 1 \le r^{(i)}$, gdzie wartość $4q^{(i-1)} + 1$ można otrzymać, dopisując do poprzednio wyznaczonej wartości pierwiastka częściowego $q^{(i-1)}$ 2 bity 01 z prawej strony. Dopisanie do liczby dwójkowej bitów 01 odpowiada operacji mnożenia liczby przez 4 oraz dodania 1.

Przykład 6.2

Obliczyć pierwiastek kwadratowy z liczby dwójkowej $10110{,}101101_{(2)}$ *za pomocą metody pisemnej. Wynik obliczyć z dokładnością do* 3 *cyfr po przecinku.*

Liczbę dwójkową należy podzielić na 2-bitowe grupy, zaczynając od przecinka pozycyjnego w obie strony.

q_2	q_1	q_0,	q_{-1}	q_{-2}	q_{-3}				
$\sqrt{\ }$ 01	01	10,	10	11	01				$q^{(0)} = 0$
− 01						$r^{(1)} \geq 01$?	tak	$q_2 = 1$	$q^{(1)} = 1$
0	01					$r^{(2)} \geq 101$?	nie	$q_1 = 0$	$q^{(2)} = 10$
	01	10				$r^{(3)} \geq 1001$?	nie	$q_0 = 0$	$q^{(3)} = 100$
	1	10	10			$r^{(4)} \geq 10001$?	tak	$q_{-1} = 1$	$q^{(4)} = 1001$
−	1	00	01						
		10	01						
		10	01	11		$r^{(5)} \geq 100101$?	tak	$q_{-2} = 1$	$q^{(5)} = 10011$
−		10	01	01					
		0	00	10					
		0	00	10	01	$r^{(6)} \geq 1001101$?	nie	$q_{-3} = 0$	$q^{(6)} = 100110$

W każdej iteracji algorytmu sprawdza się, czy bieżąca reszta częściowa $r^{(i)}$ jest większa lub równa od wartości $4q^{(i-1)} + 1$, czyli wartości poprzednio obliczonego pierwiastka częściowego $q^{(i-1)}$ z dopisanymi bitami 01 – $\{q^{(i-1)}, 0, 1\}$. Jeśli tak, to kolejną cyfrą wyniku będzie 1, inaczej – 0. Ostatecznie obliczony pierwiastek ma wartość $q = 100{,}110_{(2)}$.

Przykład 6.3

Obliczyć pierwiastek kwadratowy z liczby $01001100100001_{(2)}$ *przedstawionej w formacie zmiennoprzecinkowym składającym się z następujących pól: bitu znaku,* 5*-bitowego wykładnika w kodzie z przesunięciem bias,* 8*-bitowej mantysy w kodzie ZM. Wynik przedstawić w formacie zmiennoprzecinkowym.*

Na początku rozkodujemy liczbę zmiennoprzecinkową i przedstawimy ją w dwójkowej postaci wykładniczej:

0	10011	00100001

$$E = 10011_{(\text{biased})} = 19 - 15 = 4_{(10)}$$

$$L = 1{,}00100001 \cdot 2^4.$$

Wykładnik jest liczbą parzystą, więc pierwiastkowanie sprowadza się do obliczenia pierwiastka kwadratowego z mantysy, przy czym wykładnik liczby zostaje podzielony przez 2:

$$\sqrt{1{,}00100001 \cdot 2^4} = \sqrt{1{,}00100001} \cdot 2^2.$$

Obliczymy pierwiastek z mantysy metodą pisemną:

q_0,	q_{-1}	q_{-2}	q_{-3}	q_{-4}				
$\sqrt{}$ 0 1,	0 0	1 0	0 0	0 1				$q^{(0)} = 0$
− 0 1					$r^{(1)} \geq 01$?	tak	$q_0 = 1$	$q^{(1)} = 1$
0	0 0				$r^{(2)} \geq 101$?	nie	$q_{-1} = 0$	$q^{(2)} = 10$
	0 0	1 0			$r^{(3)} \geq 1001$?	nie	$q_{-2} = 0$	$q^{(3)} = 100$
	0	1 0	0 0		$r^{(4)} \geq 10001$?	nie	$q_{-3} = 0$	$q^{(4)} = 1000$
		1 0	0 0	0 1	$r^{(5)} \geq 100001$?	tak	$q_{-4} = 1$	$q^{(5)} = 10001$
−		1 0	0 0	0 1				
		0	0 0	0 0				

W wyniku otrzymujemy wartość pierwiastka $1{,}0001 \cdot 2^2$, która nie wymaga normalizacji i zostanie przedstawiona w formacie zmiennoprzecinkowym w następującej postaci:

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

6.1.2. Obliczanie pierwiastka kwadratowego za pomocą metody odtwarzającej

Podobnie jak dzielenie pierwiastkowanie kwadratowe można przedstawić w postaci sekwencji operacji przesunięcia i odejmowania. Przy pierwiastkowaniu liczb zmiennoprzecinkowych liczba podpierwiastkowa znajduje się w przedziale [1, 4). Wynika to z tego, że mantysa liczby jest albo znormalizowana, albo przesunięta o 1 bit w lewo w celu sprowadzenia do parzystej wartości wykładnika. W ten sposób liczba pierwiastkowana z może być wyrażona postacią $z_1 z_2$, $z_{-1} z_{-2} \ldots z_{-m}$ (znajduje się w przedziale [1, 4)), z kolei pierwiastek q zawsze otrzymujemy w postaci znormalizowanej $1, q_{-1} q_{-2} \ldots q_{-m}$, czyli znajduje się w przedziale [1, 2).

Metoda odtwarzająca pierwiastkowania liczby jest podobna do metody dzielenia. Algorytm obliczania pierwiastka kwadratowego za pomocą metody odtwarzającej [8] możemy opisać następująco:

1. Na początku, w zerowej iteracji algorytmu, wyznaczamy starszą cyfrę pierwiastka, która zawsze będzie jedynką, $q_0 = 1$, pierwiastek częściowy przy tym przyjmuje wartość $q^{(0)} = 1$.
2. W każdej kolejnej i-tej iteracji algorytmu wykonywane jest przesunięcie reszty częściowej o 1 bit w lewo oraz próbne odejmowanie wartości $2q^{(i-1)}+2^{-i}$ od przesuniętej reszty.
3. Następnie analizowany jest znak wyniku odejmowania. Jeśli jest nieujemny, kolejna cyfra wyniku przyjmuje wartość 1, $q_i = 1$. W przeciwnym wypadku – przyjmuje wartość zerową ($q_i = 0$) oraz wykonuje się odtworzenie wartości reszty częściowej sprzed operacji odejmowania.
4. Algorytm jest realizowany do otrzymania zerowej reszty częściowej (przy braku kolejnych cyfr do spisania) lub do osiągnięcia założonej dokładności obliczeń.

Przykład 6.4

Obliczyć pierwiastek kwadratowy z liczby dwójkowej $1{,}110111_{(2)}$ *za pomocą metody odtwarzającej. Obliczyć wynik z dokładnością do* 5 *cyfr po przecinku.*

W zapisie części całkowitej liczby wykorzystujemy trzy cyfry dwójkowe. Dwie cyfry są wymagane, ponieważ reszty częściowe znajdują się w przedziale [1, 4). Dodatkowa starsza cyfra w części całkowitej jest potrzebna, aby pomieścić dodatkowy bit, który pojawia się przy przesunięciu reszty częściowej w lewo. Ten bit działa również jako bit znaku przy wykonaniu próbnego odejmowania.

L	0 1, 1 1 0 1 1 1	$q_0 = 1$	$q^{(0)} = 1,$
$r^{(1)}=L-1$	0 0 0, 1 1 0 1 1 1		
$2r^{(1)}$	0 0 1, 1 0 1 1 1 0		
$-2q^{(0)}+2^{-1}$	− 1 0, 1		
	1 1 1, 0 0 1 1 1 0	$q_{-1} = 0$	$q^{(1)} = 1{,}0$
$r^{(2)}=2r^{(1)}$	0 0 1, 1 0 1 1 1 0	odtwarzanie	
$2r^{(2)}$	0 1 1, 0 1 1 1 0 0		
$-2q^{(1)}+2^{-2}$	− 1 0, 0 1		
$r^{(3)}$	0 0 1, 0 0 1 1 0 0	$q_{-2} = 1$	$q^{(2)} = 1{,}01$
$2r^{(3)}$	0 1 0, 0 1 1 0 0 0		
$-2q^{(2)}+2^{-3}$	− 1 0, 1 0 1		
	1 1 1, 1 1 0 0 0 0	$q_{-3} = 0$	$q^{(3)} = 1{,}010$
$r^{(4)}=2r^{(3)}$	0 1 0, 0 1 1 0 0 0	odtwarzanie	
$2r^{(4)}$	1 0 0, 1 1 0 0 0 0		
$-2q^{(3)}+2^{-4}$	− 1 0, 1 0 0 1		
$r^{(5)}$	0 1 0, 0 0 1 1 0 0	$q_{-4} = 1$	$q^{(4)} = 1{,}010$
$2r^{(5)}$	1 0 0, 0 1 1 0 0 0		
$-2q^{(4)}+2^{-5}$	− 1 0, 1 0 1 0 1		
$r^{(6)}$	0 0 1, 1 0 1 1 1 0	$q_{-5} = 1$	$q^{(5)} = 1{,}01011$
$2r^{(6)}$	0 1 1, 0 1 1 0 0 0		
$-2q^{(5)}+2^{-6}$	− 1 0, 1 0 1 1 0 1		
	0 0 0, 1 0 1 1 1 1	$q_{-6} = 1$	$q^{(6)} = 1{,}010111$

Dodatkowa szósta cyfra po przecinku została wyznaczona w celu zaokrąglenia wyniku:

$$q = 1{,}010111 \approx 1{,}01100_{(2)}.$$

Przykład 6.5

Obliczyć pierwiastek kwadratowy z liczby $0101010011010001_{(2)}$ *przedstawionej w formacie zmiennoprzecinkowym składającym się z następujących pól: bitu znaku,* 5*-bitowego wykładnika w kodzie z przesunięciem bias,* 12*-bitowej mantysy w kodzie ZM. Wynik przedstawić w formacie zmiennoprzecinkowym.*

Sprowadzimy liczbę do dwójkowej postaci wykładniczej:

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

$$E = 10101_{(\text{biased})} = 21 - 15 = 6_{(10)}$$

$$L = 1{,}010011010001 \cdot 2^6$$

$$\sqrt{1{,}010011010001 \cdot 2^6} = \sqrt{1{,}010011010001} \cdot 2^3 .$$

Pierwiastek z mantysy obliczamy za pomocą metody odtwarzającej:

L	0 1, 0 1 0 0 1 1 0 1 0 0 0 1	$q_0 = 1$	$q^{(0)} = 1,$
$r^{(1)}=L-1$	0 0 0, 0 1 0 0 1 1 0 1 0 0 0 1		
$2r^{(1)}$	0 0 0, 1 0 0 1 1 0 1 0 0 0 1 0		
$-2q^{(0)}+2^{-1}$	− 1 0, 1		
	1 1 0, 0 0 0 1 1 0 1 0 0 0 1 0	$q_{-1} = 0$	$q^{(1)} = 1{,}0$
$r^{(2)}=2r^{(1)}$	0 0 0, 1 0 0 1 1 0 1 0 0 0 1 0	odtwarzanie	
$2r^{(2)}$	0 0 1, 0 0 1 1 0 1 0 0 0 1 0 0		
$-2q^{(1)}+2^{-2}$	− 1 0, 0 1		
	1 1 0, 1 1 1 1 0 1 0 0 0 1 0 0	$q_{-2} = 0$	$q^{(2)} = 1{,}00$
$r^{(3)}=2r^{(2)}$	0 0 1, 0 0 1 1 0 1 0 0 0 1 0 0	odtwarzanie	
$2r^{(3)}$	0 1 0, 0 1 1 0 1 0 0 0 1 0 0 0		
$-2q^{(2)}+2^{-3}$	− 1 0, 0 0 1		
$r^{(4)}$	0 0 0, 0 1 0 0 1 0 0 0 1 0 0 0	$q_{-3} = 1$	$q^{(3)} = 1{,}001$
$2r^{(4)}$	0 0 0, 1 0 0 1 0 0 0 1 0 0 0 0		
$-2q^{(3)}+2^{-4}$	− 1 0, 0 1 0 1		
	1 1 0, 0 1 0 0 0 0 0 1 0 0 0 0	$q_{-4} = 0$	$q^{(4)} = 1{,}0010$
$r^{(5)}=2r^{(4)}$	0 0 0, 1 0 0 1 0 0 0 1 0 0 0 0	odtwarzanie	
$2r^{(5)}$	0 0 1, 0 0 1 0 0 0 1 0 0 0 0 0		
$-2q^{(4)}+2^{-5}$	− 1 0, 0 1 0 0 1		
	1 1 0, 1 1 0 1 1 0 1 0 0 0 0 0	$q_{-5} = 0$	$q^{(5)} = 1{,}00100$
$r^{(6)}=2r^{(5)}$	0 0 1, 0 0 1 0 0 0 1 0 0 0 0 0	odtwarzanie	
$2r^{(6)}$	0 1 0, 0 1 0 0 0 1 0 0 0 0 0 0		
$-2q^{(5)}+2^{-6}$	− 1 0, 0 1 0 0 0 1		
	0 0 0, 0 0 0 0 0 0 0 0 0 0 0 0	$q_{-6} = 1$	$q^{(6)} = 1{,}001001$

Pierwiastek kwadratowy obliczony został dokładnie i ma wartość $q = 1{,}001001 \cdot 2^3$. Do formatu zmiennoprzecinkowego zapisany zostanie w następującej postaci:

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

6.2. Zadania z rozwiązaniami

Zadanie 6.1

Za pomocą metody pisemnej obliczyć pierwiastek kwadratowy z liczby $0101100101101_{(2)}$ *przedstawionej w formacie zmiennoprzecinkowym składającym się z następujących pól:*

bitu znaku, 6-bitowego wykładnika w kodzie z przesunięciem bias, 6-bitowej mantysy w kodzie ZM. Wynik zapisać w formacie zmiennoprzecinkowym.

Z formatu zmiennoprzecinkowego otrzymujemy następującą dwójkową postać wykładniczą liczby:

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

$$E = 101100_{(\text{biased})} = 44 - 31 = 13_{(10)}$$

$$L = 1{,}101101 \cdot 2^{13}.$$

Aby ułatwić obliczenie pierwiastka kwadratowego, przekształcamy liczbę do takiej postaci, w której wykładnik jest liczbą parzystą, zmniejszając o 1 wartość wykładnika:

$$1{,}101101 \cdot 2^{13} = 11{,}01101 \cdot 2^{12}.$$

Teraz pierwiastkowanie sprowadza się do wyznaczenia pierwiastka kwadratowego z mantysy liczby:

$$\sqrt{11{,}01101 \cdot 2^{12}} = \sqrt{11{,}01101} \cdot 2^{6}.$$

Obliczymy pierwiastek kwadratowy z dokładnością do 6 cyfr po przecinku, siódma dodatkowa cyfra wyznaczana jest w celu poprawnego zaokrąglenia wyniku obliczeń.

q_0,	q_{-1}	q_{-2}	q_{-3}	q_{-4}	q_{-5}	q_{-6}	q_{-7}				
$\sqrt{1}$ 1,	0 1	1 0	1 0	0 0	0 0	0 0	0 0	$q^{(0)} = 0$			
− 0 1								$r^{(1)} \geq 01$?	tak	$q_0 = 1$	$q^{(1)} = 1$
1 0	0 1							$r^{(2)} \geq 101$?	tak	$q_{-1} = 1$	$q^{(2)} = 11$
− 1	0 1										
1	0 0	1 0						$r^{(3)} \geq 1101$?	tak	$q_{-2} = 1$	$q^{(3)} = 111$
−	1 1	0 1									
	1	0 1	1 0					$r^{(4)} \geq 11101$?	nie	$q_{-3} = 0$	$q^{(4)} = 1110$
	1	0 1	1 0	0 0				$r^{(5)} \geq 111001$?	tak	$q_{-4} = 1$	$q^{(5)} = 11101$
−		1 1	1 0	0 1							
		0 1	1 1	1 1	0 0			$r^{(6)} \geq 1110101$?	tak	$q_{-5} = 1$	$q^{(6)} = 111011$
−		1	1 1	0 1	0 1						
			0 0	0 1	1 1	0 0		$r^{(7)} \geq 11101101$?	nie	$q_{-6} = 0$	$q^{(7)} = 1110110$
			0	0 1	1 1	0 0	0 0	$r^{(8)} \geq 111011001$?	nie	$q_{-7} = 0$	$q^{(8)} = 11101100$

W wyniku obliczeń otrzymujemy wartość pierwiastka $1{,}110110 \cdot 2^6$, która w formacie zmiennoprzecinkowym przedstawia się następująco:

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

Zadanie 6.2

Za pomocą metody pisemnej obliczyć pierwiastek kwadratowy z liczby $32E72000_{(16)}$ *przedstawionej w standardowym formacie zmiennoprzecinkowym pojedynczej precyzji. Wynik zapisać w formacie zmiennoprzecinkowym.*

Szesnastkową postać formatu zamienimy na dwójkową, z której wyznaczymy postać wykładniczą liczby:

$$L = 32E72000_{(16)} = 0011\ 0010\ 1110\ 0111\ 0010\ 0000\ 0000\ 0000_{(2)}$$

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

$$E = 01100101_{(\text{biased})} = 101 - 127 = -26_{(10)}$$

$$L = 1{,}1100111001 \cdot 2^{-26}.$$

Ponieważ wykładnik jest liczbą parzystą, mantysa nie wymaga przesunięcia.

$$\sqrt{1{,}1100111001 \cdot 2^{-26}} = \sqrt{1{,}1100111001} \cdot 2^{-13}.$$

Pierwiastek kwadratowy z mantysy obliczymy za pomocą metody pisemnej:

q_0,	q_{-1}	q_{-2}	q_{-3}	q_{-4}	q_{-5}				
$\sqrt{}$0 1,	1 1	0 0	1 1	1 0	0 1				$q^{(0)} = 0$
− 0 1						$r^{(1)} \geq 01$?	tak	$q_0 = 1$	$q^{(1)} = 1$
0	1 1					$r^{(2)} \geq 101$?	nie	$q_{-1} = 0$	$q^{(2)} = 10$
	1 1	0 0				$r^{(3)} \geq 1001$?	tak	$q_{-2} = 1$	$q^{(3)} = 101$
−	1 0	0 1							
	0	1 1	1 1			$r^{(4)} \geq 10101$?	nie	$q_{-3} = 0$	$q^{(4)} = 1010$
		1 1	1 1	1 0		$r^{(5)} \geq 101001$?	tak	$q_{-4} = 1$	$q^{(5)} = 10101$
−		1 0	1 0	0 1					
		0 1	0 1	0 1	0 1	$r^{(6)} \geq 1010101$?	tak	$q_{-5} = 1$	$q^{(6)} = 101011$
−		1	0 1	0 1	0 1				
			0 0	0 0	0 0				

W wyniku obliczeń otrzymujemy wartość pierwiastka $q = 1{,}01011 \cdot 2^{-13}$. Pierwiastek kwadratowy z liczby zmiennoprzecinkowej ma następującą reprezentację w standardowym formacie pojedynczej precyzji:

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

Zadanie 6.3

Za pomocą metody odtwarzającej obliczyć pierwiastek kwadratowy z liczby $00100010110011_{(2)}$ *przedstawionej w formacie zmiennoprzecinkowym składającym się z następujących pól: bitu znaku,* 5*-bitowego wykładnika w kodzie z przesunięciem bias,* 8*-bitowej mantysy w kodzie ZM. Wynik zapisać w formacie zmiennoprzecinkowym.*

Rozkodujemy format zmiennoprzecinkowy:

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

$$E = 01000_{(\text{biased})} = 8 - 15 = -7_{(10)}$$

$$L = 1{,}10110011 \cdot 2^{-7}.$$

W celu ułatwienia operacji pierwiastkowania zmniejszamy wykładnik o 1, aby otrzymać parzystą wartość wykładnika:

$$1{,}10110011 \cdot 2^{-7} = 11{,}0110011 \cdot 2^{-8}.$$

Pierwiastkowanie sprowadza się wtedy do wyznaczenia pierwiastka kwadratowego z mantysy liczby:

$$\sqrt{11{,}0110011 \cdot 2^{-8}} = \sqrt{11{,}0110011} \cdot 2^{-4}.$$

Wyznaczamy pierwiastek z dokładnością do 8 cyfr po przecinku, co wynika z rozmiaru pola mantysy formatu zmiennoprzecinkowego. Dodatkowa 9 cyfra będzie potrzebna do zaokrąglenia otrzymanego wyniku.

Pierwiastek kwadratowy obliczamy za pomocą metody odtwarzającej:

L	1 1, 0 1 1 0 0 1 1 0 0	$q_0 = 1$	$q^{(0)} = 1,$
$r^{(1)}=L-1$	0 1 0, 0 1 1 0 0 1 1 0 0		
$2r^{(1)}$	1 0 0, 1 1 0 0 1 1 0 0 0		
$-2q^{(0)}+2^{-1}$	− 1 0, 1		
$r^{(2)}$	0 1 0, 0 1 0 0 1 1 0 0 0	$q_{-1} = 1$	$q^{(1)} = 1{,}1$
$2r^{(2)}$	1 0 0, 1 0 0 1 1 0 0 0 0		
$-2q^{(1)}+2^{-2}$	− 1 1, 0 1		
$r^{(3)}$	0 0 1, 0 1 0 1 1 0 0 0 0	$q_{-2} = 1$	$q^{(2)} = 1{,}11$
$2r^{(3)}$	0 1 0, 1 0 1 1 0 0 0 0 0		
$-2q^{(2)}+2^{-3}$	− 1 1, 1 0 1		
	1 1 1, 0 0 0 1 0 0 0 0 0	$q_{-3} = 0$	$q^{(3)} = 1{,}110$
$r^{(4)}=2r^{(3)}$	0 1 0, 1 0 1 1 0 0 0 0 0	odtwarzanie	
$2r^{(4)}$	1 0 1, 0 1 1 0 0 0 0 0 0		
$-2q^{(3)}+2^{-4}$	− 1 1, 1 0 0 1		
$r^{(5)}$	0 0 1, 1 1 0 1 0 0 0 0 0	$q_{-4} = 1$	$q^{(4)} = 1{,}1101$
$2r^{(5)}$	0 1 1, 1 0 1 0 0 0 0 0 0		
$-2q^{(4)}+2^{-5}$	− 1 1, 1 0 1 0 1		
	1 1 1, 1 1 1 1 1 0 0 0 0	$q_{-5} = 0$	$q^{(5)} = 1{,}11010$
$r^{(6)}=2r^{(5)}$	0 1 1, 1 0 1 0 0 0 0 0 0	odtwarzanie	
$2r^{(6)}$	1 1 1, 0 1 0 0 0 0 0 0 0		
$-2q^{(5)}+2^{-6}$	− 1 1, 1 0 1 0 0 1		
$r^{(7)}$	0 1 1, 1 0 0 1 1 1 0 0 0	$q_{-6} = 1$	$q^{(6)} = 1{,}110101$
$2r^{(7)}$	1 1 1, 0 0 1 1 1 0 0 0 0		
$-2q^{(6)}+2^{-7}$	− 1 1, 1 0 1 0 1 0 1		
$r^{(8)}$	0 1 1, 1 0 0 0 1 1 1 0 0	$q_{-7} = 1$	$q^{(7)} = 1{,}1101011$

$2r^{(8)}$	1 1 1, 0 0 0 1 1 1 0 0 0		
$-2q^{(7)}+2^{-8}$	– 1 1, 1 0 1 0 1 1 0 1		
$r^{(9)}$	0 1 1, 0 1 1 0 1 1 1 1 0	$q_{-8} = 1$	$q^{(8)} = 1{,}11010111$
$2r^{(9)}$	1 1 0, 1 1 0 1 1 1 1 0 0		
$-2q^{(8)}+2^{-9}$	– 1 1, 1 0 1 0 1 1 1 0 1		
$r^{(10)}$	0 1 1, 0 0 1 0 1 1 1 1 1	$q_{-9} = 1$	$q^{(9)} = 1{,}110101111$

Po zaokrągleniu wyniku otrzymamy:

$$q = 1{,}110101111 \cdot 2^{-4} \approx 1{,}11011000 \cdot 2^{-4}.$$

Wyznaczony pierwiastek kwadratowy przedstawimy w formacie zmiennoprzecinkowym:

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

Zadanie 6.4

Za pomocą metody odtwarzającej obliczyć pierwiastek kwadratowy z liczby $4E55E400_{(16)}$ *przedstawionej w standardowym formacie zmiennoprzecinkowym pojedynczej precyzji. Wynik zapisać w formacie zmiennoprzecinkowym.*

Z formatu zmiennoprzecinkowego wyznaczamy dwójkową wykładniczą postać liczby:

$$L = 4E55E400_{(16)} = 0100\ 1110\ 0101\ 0101\ 1110\ 0100\ 0000\ 0000_{(2)}$$

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

$$E = 10011100_{(\text{biased})} = 156 - 127 = 29_{(10)}$$

$$L = 1{,}1010101111001 \cdot 2^{29}.$$

Przenosimy przecinek tak, aby ułatwić obliczenie pierwiastka:

$$L = 1{,}1010101111001 \cdot 2^{29} = 11{,}010101111001 \cdot 2^{28}$$

$$\sqrt{11{,}010101111001 \cdot 2^{28}} = \sqrt{11{,}010101111001} \cdot 2^{14}.$$

Pierwiastek kwadratowy z mantysy obliczamy za pomocą metody odtwarzającej:

L	1 1, 0 1 0 1 0 1 1 1 1 0 0 1	$q_0 = 1$	$q^{(0)} = 1,$
$r^{(1)}=L-1$	0 1 0, 0 1 0 1 0 1 1 1 1 0 0 1		
$2r^{(1)}$	1 0 0, 1 0 1 0 1 1 1 1 0 0 1 0		
$-2q^{(0)}+2^{-1}$	– 1 0, 1		
$r^{(2)}$	0 1 0, 0 0 1 0 1 1 1 1 0 0 1 0	$q_{-1} = 1$	$q^{(1)} = 1{,}1$
$2r^{(2)}$	1 0 0, 0 1 0 1 1 1 1 0 0 1 0 0		
$-2q^{(1)}+2^{-2}$	– 1 1, 0 1		

$r^{(3)}$	0 0 1, 0 0 0 1 1 1 1 0 0 1 0 0	$q_{-2} = 1$	$q^{(2)} = 1{,}11$
$2r^{(3)}$	0 1 0, 0 0 1 1 1 1 0 0 1 0 0 0		
$-2q^{(2)}+2^{-3}$	− 1 1, 1 0 1		
	1 1 0, 1 0 0 1 1 1 0 0 1 0 0 0	$q_{-3} = 1$	$q^{(3)} = 1{,}110$
$r^{(4)}=2r^{(4)}$	0 1 0, 0 0 1 1 1 1 0 0 1 0 0 0	odtwarzanie	
$2r^{(4)}$	1 0 0, 0 1 1 1 1 0 0 1 0 0 0 0		
$-2q^{(3)}+2^{-4}$	− 1 1, 1 0 0 1		
$r^{(5)}$	0 0 0, 1 1 1 0 1 0 0 1 0 0 0 0	$q_{-4} = 1$	$q^{(4)} = 1{,}1101$
$2r^{(5)}$	0 0 1, 1 1 0 1 0 0 1 0 0 0 0 0		
$-2q^{(4)}+2^{-5}$	− 1 1, 1 0 1 0 1		
	1 1 0, 0 0 1 0 1 0 1 0 0 0 0 0	$q_{-5} = 0$	$q^{(5)} = 1{,}11010$
$r^{(6)}=2r^{(5)}$	0 0 1, 1 1 0 1 0 0 1 0 0 0 0 0	odtwarzanie	
$2r^{(6)}$	0 1 1, 1 0 1 0 0 1 0 0 0 0 0 0		
$-2q^{(5)}+2^{-6}$	− 1 1, 1 0 1 0 0 1		
	0 0 0, 0 0 0 0 0 0 0 0 0 0 0 0	$q_{-6} = 1$	$q^{(6)} = 1{,}110101$.

Wynik wyznaczenia pierwiastka kwadratowego $q = 1{,}110101 \cdot 2^{14}$ przedstawimy w formacie zmiennoprzecinkowym pojedynczej precyzji:

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

6.3. Zadania do samodzielnego rozwiązania

Zadanie 6.5

Za pomocą metody pisemnej obliczyć pierwiastek kwadratowy z podanej liczby zmiennoprzecinkowej zapisanej w formacie składającym się z następujących pól: bitu znaku, wykładnika w kodzie z przesunięciem bias, mantysy w kodzie ZM (*rozmiary pól wykładnika oraz mantysy są podane w poszczególnych zadaniach*)*:*

a) $00001101111100_{(2)}$, 5-bitowy wykładnik, 8-bitowa mantysa;

b) $00101001011011_{(2)}$, 6-bitowy wykładnik, 7-bitowa mantysa;

c) $0100111010011100 11_{(2)}$, 7-bitowy wykładnik, 10-bitowa mantysa;

d) $0010010001100_{(2)}$, 5-bitowy wykładnik, 7-bitowa mantysa.

Zadanie 6.6

Za pomocą metody pisemnej obliczyć pierwiastek kwadratowy z podanej liczby zmiennoprzecinkowej zapisanej w standardowym formacie pojedynczej precyzji:

a) $64D2C420_{(16)}$ b) $4755E400_{(16)}$ c) $3CA8D200_{(16)}$ d) $33732A40_{(16)}$

Zadanie 6.7

Za pomocą metody odtwarzającej obliczyć pierwiastek kwadratowy z podanej liczby zmiennoprzecinkowej zapisanej w formacie składającym się z następujących pól: bitu znaku, wykładnika w kodzie z przesunięciem bias, mantysy w kodzie ZM (*rozmiary pól wykładnika oraz mantysy są podane w poszczególnych zadaniach*)*:*

a) $59A4_{(16)}$, 5-bitowy wykładnik, 10-bitowa mantysa;

b) $0101101110101_{(2)}$, 6-bitowy wykładnik, 6-bitowa mantysa;

c) $0001110011101_{(2)}$, 5-bitowy wykładnik, 7-bitowa mantysa;

d) $5DC9_{(16)}$, 6-bitowy wykładnik, 9-bitowa mantysa.

Zadanie 6.8

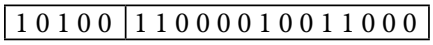
Za pomocą metody pisemnej obliczyć pierwiastek kwadratowy z podanej liczby zmiennoprzecinkowej zapisanej w standardowym formacie pojedynczej precyzji:

a) $3DB77B20_{(16)}$ b) $494CD900_{(16)}$ c) $3208F690_{(16)}$ d) $4AE51D20_{(16)}$

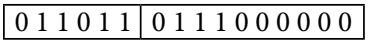
5. Odpowiedzi do zadań

Zadanie 2.7

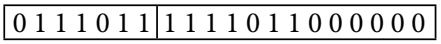
a)



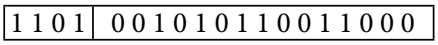
b)



c)

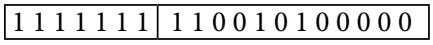


d)

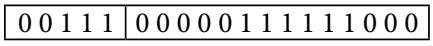


Zadanie 2.8

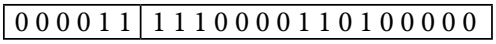
a)



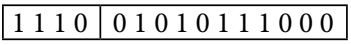
b)



c)

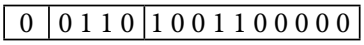


d)

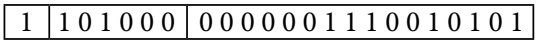


Zadanie 2.9

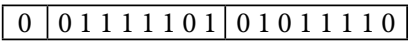
a)



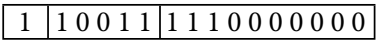
b)



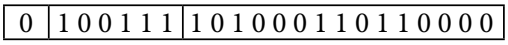
c)



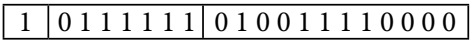
d)



e)



f)



Zadanie 2.10

a)

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

b)

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

c)

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

d)

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

Zadanie 2.11

a)

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

b)

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

c)

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

d)

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

Zadanie 2.12

a) $-42{,}5$; b) $\frac{117}{256}$; c) -237; d) $-\frac{115}{128}$;

Zadanie 2.13

a) $-54{,}75$; b) 256; c) $-\frac{119}{512}$; d) $1\frac{27}{128}$; e) $-413{,}625$; f) $-\frac{59}{256}$;

Zadanie 3.12

a)

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

b)

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

c)

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

d)

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

e)

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

f)

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

Zadanie 3.13

a)

0	10000000111	00111011011100110011001100110011001100110011001100110011

b)

1	10000000000	00011001100110011001100110011001100110011001100110011010

c)

0	01111111111	00010110100000000000000000000000000000000000000000000000

d)

1	01111111110	11001000100000000000000000000000000000000000000000000000

e)

0	10000001001	01000011110000000000000000000000000000000000000000000000

f)

1	10000000100	00001110011001100110011001100110011001100110011001100110

Zadanie 3.14

a) –925,875; b) $\frac{119}{512}$; c) –436; d) $1\frac{53}{128}$; e) $-\frac{39}{512}$; f) $155\frac{25}{32}$;

Zadanie 3.15

a) $-13\frac{27}{32}$; b) $\frac{219}{512}$; c) –1911,375; d) $\frac{15}{64}$; e) –379; f) $-1\frac{207}{512}$;

Zadanie 3.16

a)

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

b)

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

c)

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

d)

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

Zadanie 3.17

a)

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

b)

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

c)

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

d)

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

Zadanie 3.18

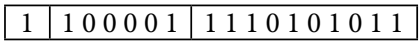
a) $1{,}578951 \cdot 10^{-16}$; b) 179; c) $8{,}937657 \cdot 10^{5}$; d) $-1{,}53673 \cdot 10^{7}$;

Zadanie 3.19

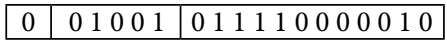
a) $-8{,}737599 \cdot 10^{9}$; b) 1890; c) $-9{,}159874 \cdot 10^{-7}$; d) $-4{,}378562 \cdot 10^{11}$;

Zadanie 4.8

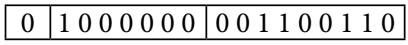
a)



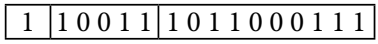
b)



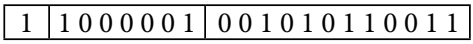
c)



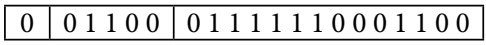
d)



e)

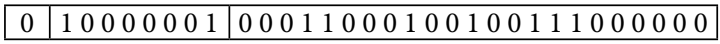


f)

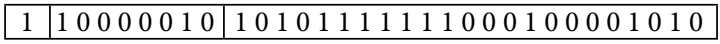


Zadanie 4.9

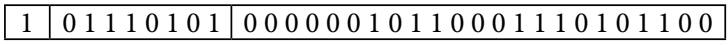
a)



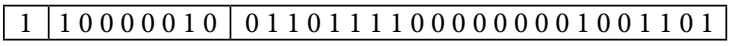
b)



c)

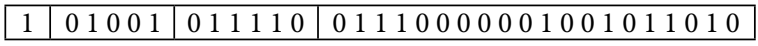


d)

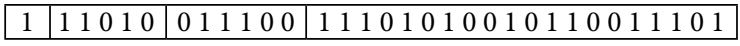


Zadanie 4.10

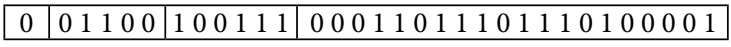
a)



b)



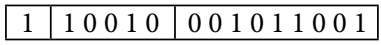
c)



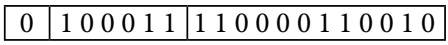
d)

Zadanie 4.11

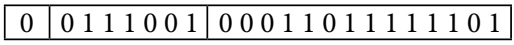
a)



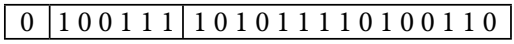
b)



c)

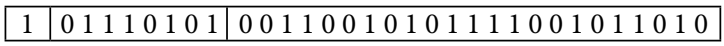


d)

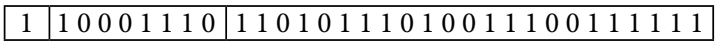


Zadanie 4.12

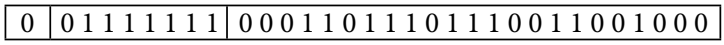
a)



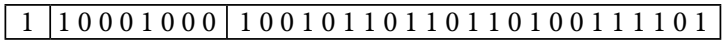
b)



c)

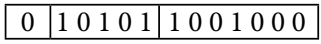


d)

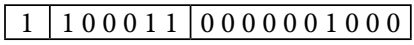


Zadanie 5.6

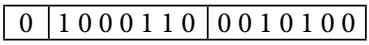
a)



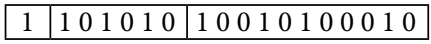
b)



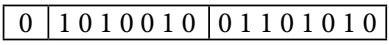
c)



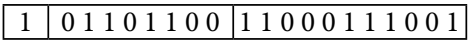
d)



e)

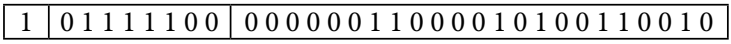


f)

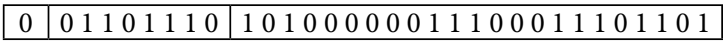


Zadanie 5.7

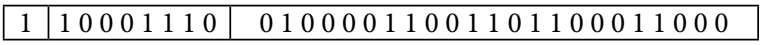
a)



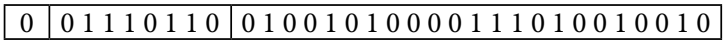
b)



c)



d)



Zadanie 5.8

a)

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

b)

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

c)

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

d)

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

Zadanie 5.9

a)

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

b)

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

c)

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

d)

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

Zadanie 5.10

a)

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

b)

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

c)

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

d)

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

e)

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

Zadanie 6.5

a)

$q_0, q_{-1} q_{-2} q_{-3} q_{-4} q_{-5} q_{-6} q_{-7} q_{-8} q_{-9}$		
$\sqrt{01,\ 01\ 11\ 11\ 00\ 00\ 00\ 00\ 00\ 00}$		$q^{(0)} = 0$
$-\ 01$	$q_0 = 1$	$q^{(1)} = 1$
$0\ 01$	$q_{-1} = 0$	$q^{(2)} = 10$
$01\ 11$	$q_{-2} = 0$	$q^{(3)} = 100$
$1\ 11\ 11$	$q_{-3} = 1$	$q^{(4)} = 1001$
$-\ 1\ 00\ 01$		
$11\ 10\ 00$	$q_{-4} = 1$	$q^{(5)} = 10011$
$-\ 10\ 01\ 01$		
$01\ 00\ 11\ 00$	$q_{-5} = 0$	$q^{(6)} = 100110$
$100\ 11\ 00\ 00$	$q_{-6} = 1$	$q^{(7)} = 1001101$
$-\ 10\ 01\ 10\ 01$		
$10\ 01\ 01\ 11\ 00$	$q_{-7} = 1$	$q^{(8)} = 10011011$
$-\ 100\ 11\ 01\ 01$		
$100\ 10\ 01\ 11\ 00$	$q_{-8} = 1$	$q^{(9)} = 100110111$
$-\ 10\ 01\ 10\ 11\ 01$		
$10\ 00\ 10\ 11\ 11\ 00$	$q_{-9} = 1$	$q^{(10)} = 1001101111$
$-\ 1\ 00\ 11\ 01\ 11\ 01$		
$0\ 11\ 11\ 01\ 11\ 11$		

0	01001	00111000

b)

$q_0, q_{-1} q_{-2} q_{-3} q_{-4} q_{-5} q_{-6} q_{-7} q_{-8}$		
$\sqrt{11,\ 01\ 10\ 11\ 00\ 00\ 00\ 00\ 00}$		$q^{(0)} = 0$
$-\ 01$	$q_0 = 1$	$q^{(1)} = 1$
$10\ 01$	$q_{-1} = 1$	$q^{(2)} = 11$
$-\ 1\ 01$		
$1\ 00\ 10$	$q_{-2} = 1$	$q^{(3)} = 111$
$-\ 11\ 01$		
$1\ 01\ 11$	$q_{-3} = 0$	$q^{(4)} = 1110$
$1\ 01\ 11\ 00$	$q_{-4} = 1$	$q^{(5)} = 11101$
$-\ 11\ 10\ 01$		
$10\ 00\ 11\ 00$	$q_{-5} = 1$	$q^{(6)} = 111011$
$-\ 1\ 11\ 01\ 01$		
$0\ 01\ 01\ 11\ 00$	$q_{-6} = 0$	$q^{(7)} = 1110110$
$1\ 01\ 11\ 00\ 00$	$q_{-7} = 0$	$q^{(8)} = 11101100$
$1\ 01\ 11\ 00\ 00\ 00$	$q_{-8} = 1$	$q^{(9)} = 111011001$
$-\ 11\ 10\ 11\ 00\ 01$		
$10\ 00\ 00\ 11\ 11$		

0	011001	11011101

c)

q_0,	q_{-1}	q_{-2}	q_{-3}	q_{-4}	q_{-5}	q_{-6}	q_{-7}	q_{-8}	q_{-9}	q_{-10}	q_{-11}	
$\sqrt{}$ 1 1,	0 0	1 1	1 0	0 1	1 0	0 0	0 0	0 0	0 0	0 0	0 0	$q^{(0)} = 0$
− 0 1												$q_0 = 1$ $q^{(1)} = 1$
1 0	0 0											$q_{-1} = 1$ $q^{(2)} = 11$
− 1	0 1											
0	1 1	1 1										$q_{-2} = 1$ $q^{(3)} = 111$
−	1 1	0 1										
	0	1 0	1 0									$q_{-3} = 0$ $q^{(4)} = 1110$
		1 0	1 0	0 1								$q_{-4} = 0$ $q^{(5)} = 11100$
		1 0	1 0	0 1	1 0							$q_{-5} = 1$ $q^{(6)} = 111001$
−		1	1 1	0 0	0 1							
		0	1 1	0 1	0 1	0 0						$q_{-6} = 0$ $q^{(7)} = 1110010$
			1 1	0 1	0 1	0 0	0 0					$q_{-7} = 1$ $q^{(8)} = 11100101$
−			1	1 1	0 0	1 0	0 1					
			1	1 0	0 0	0 1	1 1	0 0				$q_{-8} = 1$ $q^{(9)} = 111001011$
−				1 1	1 0	0 1	0 1	0 1				
				1 0	1 0	0 0	0 1	1 1	0 0			$q_{-9} = 1$ $q^{(10)} = 1110010111$
−				1	1 1	0 0	1 0	1 1	0 1			
				0	1 0	1 1	1 0	1 1	1 1	0 0		$q_{-10} = 0$ $q^{(11)} = 11100101110$
					1 0	1 1	1 0	1 1	1 1	0 0	0 0	$q_{-11} = 1$ $q^{(12)} = 111001011101$
−					1	1 1	0 0	1 0	1 1	1 0	0 1	
					1	0 0	1 0	0 0	1 1	0 1	1 1	

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

d)

q_0,	q_{-1}	q_{-2}	q_{-3}	q_{-4}	q_{-5}	q_{-6}	q_{-7}	q_{-8}	
$\sqrt{}$ 0 1,	0 0	0 1	1 0	0 0	0 0	0 0	0 0	0 0	$q^{(0)} = 0$
− 0 1									$q_0 = 1$ $q^{(1)} = 1$
0	0 0								$q_{-1} = 0$ $q^{(2)} = 10$
	0 0	0 1							$q_{-2} = 0$ $q^{(3)} = 100$
	0	0 1	1 0						$q_{-3} = 0$ $q^{(4)} = 1000$
		0 1	1 0	0 0					$q_{-4} = 0$ $q^{(5)} = 10000$
		1	1 0	0 0	0 0				$q_{-5} = 1$ $q^{(6)} = 100001$
−		1	0 0	0 0	0 1				
		0	0 1	1 1	1 1	0 0			$q_{-6} = 0$ $q^{(7)} = 1000010$
			1	1 1	1 1	0 0	0 0		$q_{-7} = 1$ $q^{(8)} = 10000101$
−			1	0 0	0 0	1 0	0 1		
			0	1 1	1 0	0 1	1 1	0 0	$q_{-8} = 1$ $q^{(9)} = 100001011$
−				1 0	0 0	0 1	0 1	0 1	
				0 1	1 0	0 0	0 1	1 1	

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

Zadanie 6.6

a)

	q_0,	q_{-1}	q_{-2}	q_{-3}	q_{-4}	q_{-5}	q_{-6}	q_{-7}	q_{-8}	q_{-9}	
$\sqrt{}$	0 1,	1 0	1 0	0 1	0 1	1 0	0 0	1 0	0 0	0 1	$q^{(0)} = 0$
−	0 1										$q_0 = 1$ $\quad q^{(1)} = 1$
	0	1 0									$q_{-1} = 0$ $\quad q^{(2)} = 10$
		1 0	1 0								$q_{-2} = 1$ $\quad q^{(3)} = 101$
−		1 0	0 1								
		0	0 1	0 1							$q_{-3} = 0$ $\quad q^{(4)} = 1010$
			0 1	0 1	0 1						$q_{-4} = 0$ $\quad q^{(5)} = 10100$
			1	0 1	0 1	1 0					$q_{-5} = 1$ $\quad q^{(6)} = 101001$
−			1	0 1	0 0	0 1					
			0	0 0	0 1	0 1	0 0				$q_{-6} = 0$ $\quad q^{(7)} = 1010010$
				0	0 1	0 1	0 0	1 0			$q_{-7} = 0$ $\quad q^{(8)} = 10100100$
					0 1	0 1	0 0	1 0	0 0		$q_{-8} = 0$ $\quad q^{(9)} = 101001000$
					1	0 1	0 0	1 0	0 0	0 1	$q_{-9} = 1$ $\quad q^{(10)} = 1010010001$
−					1	0 1	0 0	1 0	0 0	0 1	
						0 0	0 0	0 0	0 0	0 0	

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

b)

	q_0,	q_{-1}	q_{-2}	q_{-3}	q_{-4}	q_{-5}	q_{-6}		
$\sqrt{}$	1 1,	0 1	0 1	0 1	1 1	1 0	0 1		$q^{(0)} = 0$
−	0 1							$q_0 = 1$	$q^{(1)} = 1$
	1 0	0 1						$q_{-1} = 1$	$q^{(2)} = 11$
−	1	0 1							
	1	0 0	0 1					$q_{-2} = 1$	$q^{(3)} = 111$
−		1 1	0 1						
		1	0 0	0 1				$q_{-3} = 0$	$q^{(4)} = 1110$
		1	0 0	0 1	1 1			$q_{-4} = 1$	$q^{(5)} = 11101$
−			1 1	1 0	0 1				
			0 0	1 1	1 0	1 0		$q_{-5} = 0$	$q^{(6)} = 111010$
				1 1	1 0	1 0	0 1	$q_{-6} = 1$	$q^{(7)} = 1110101$
−				1 1	1 0	1 0	0 1		
				0 0	0 0	0 0	0 0		

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

c)

q_0,	q_{-1}	q_{-2}	q_{-3}	q_{-4}	q_{-5}	q_{-6}	q_{-7}
$\sqrt{0\,1,}$	0 1	0 1	0 0	0 1	1 0	1 0	0 1
$-\,0\,1$							
0	0 1						
0 1	0 1						
1	0 1	0 0					
$-$	1	0 0	0 1				
		0 0	1 1	0 1			
		1 1	0 1	1 0			
		1 1	0 1	1 0	1 0		
$-$		1 0	0 1	0 0	0 1		
			1	0 0	1 0	0 1	0 1
$-$			1	0 0	1 0	0 1	0 1
			0	0 0	0 0	0 0	0 0

$q^{(0)} = 0$

$q_0 = 1$ $\quad q^{(1)} = 1$

$q_{-1} = 0$ $\quad q^{(2)} = 10$

$q_{-2} = 0$ $\quad q^{(3)} = 100$

$q_{-3} = 1$ $\quad q^{(4)} = 1001$

$q_{-4} = 0$ $\quad q^{(5)} = 10010$

$q_{-5} = 0$ $\quad q^{(6)} = 100100$

$q_{-6} = 1$ $\quad q^{(7)} = 1001001$

$q_{-7} = 1$ $\quad q^{(8)} = 10010011$

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

d)

q_0,	q_{-1}	q_{-2}	q_{-3}	q_{-4}	q_{-5}	q_{-6}	q_{-7}	q_{-8}
$\sqrt{1\,1,}$	1 1	0 0	1 1	0 0	1 0	1 0	1 0	0 1
$-\,0\,1$								
1 0	1 1							
$-$ 1	0 1							
1	1 0	0 0						
$-$	1 1	0 1						
	1 0	1 1	1 1					
$-$	1	1 1	0 1					
	1	0 0	1 0	0 0				
$-$		1 1	1 1	0 1				
		0 0	1 0	1 1	1 0			
			1 0	1 1	1 0	1 0		
			1 0	1 1	1 0	1 0	1 0	
$-$			1	1 1	1 1	0 0	0 1	
			0	1 1	1 1	1 0	0 1	0 1
$-$				1 1	1 1	1 0	0 1	0 1
				0 0	0 0	0 0	0 0	0 0

$q^{(0)} = 0$

$q_0 = 1$ $\quad q^{(1)} = 1$

$q_{-1} = 1$ $\quad q^{(2)} = 11$

$q_{-2} = 1$ $\quad q^{(3)} = 111$

$q_{-3} = 1$ $\quad q^{(4)} = 1111$

$q_{-4} = 1$ $\quad q^{(5)} = 11111$

$q_{-5} = 0$ $\quad q^{(6)} = 111110$

$q_{-6} = 0$ $\quad q^{(7)} = 1111100$

$q_{-7} = 1$ $\quad q^{(8)} = 11111001$

$q_{-8} = 1$ $\quad q^{(9)} = 111110011$

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

Zadanie 6.7

a)

L	1 0, 1 1 0 1 1 0 0 1	$q_0 = 1$	$q^{(0)} = 1,$
$r^{(1)}=L-1$	0 0 1, 1 1 0 1 1 0 0 1		
$2r^{(1)}$	0 1 1, 1 0 1 1 0 0 1 0		
$-2q^{(0)}+2^{-1}$	− 1 0, 1		
$r^{(2)}$	0 0 1, 0 0 1 1 0 0 1 0	$q_{-1} = 1$	$q^{(1)} = 1,1$
$2r^{(2)}$	0 1 0, 0 1 1 0 0 1 0 0		
$-2q^{(1)}+2^{-2}$	− 1 1, 0 1		
	1 1 1, 0 0 1 0 0 1 0 0	$q_{-2} = 0$	$q^{(2)} = 1,10$
$r^{(3)}$	0 1 0, 0 1 1 0 0 1 0 0	odtwarzanie	
$2r^{(3)}$	1 0 0, 1 1 0 0 1 0 0 0		
$-2q^{(2)}+2^{-3}$	− 1 1, 0 0 1		
$r^{(4)}$	0 0 1, 1 0 1 0 1 0 0 0	$q_{-3} = 1$	$q^{(3)} = 1,101$
$2r^{(4)}$	0 1 1, 0 1 0 1 0 0 0 0		
$-2q^{(3)}+2^{-4}$	− 1 1, 0 1 0 1		
$r^{(5)}$	0 0 0, 0 0 0 0 0 0 0 0	$q_{-4} = 1$	$q^{(4)} = 1,1011$

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

b)

L	0 1, 1 1 0 1 0 1 0	$q_0 = 1$	$q^{(0)} = 1$
$r^{(1)}=L-1$	0 0 0, 1 1 0 1 0 1 0		
$2r^{(1)}$	0 0 1, 1 0 1 0 1 0 0		
$-2q^{(0)}+2^{-1}$	− 1 0, 1		
	1 1 1, 0 0 1 0 1 0 0	$q_{-1} = 0$	$q^{(1)} = 1,0$
$r^{(2)}$	0 0 1, 1 0 1 0 1 0 0	odtwarzanie	
$2r^{(2)}$	0 1 1, 0 1 0 1 0 0 0		
$-2q^{(1)}+2^{-2}$	− 1 0, 0 1		
$r^{(3)}$	0 0 1, 0 0 0 1 0 0 0	$q_{-2} = 1$	$q^{(2)} = 1,01$
$2r^{(3)}$	0 1 0, 0 0 1 0 0 0 0		
$-2q^{(2)}+2^{-3}$	− 1 0, 1 0 1		
	1 1 1, 1 0 0 0 0 0 0	$q_{-3} = 0$	$q^{(3)} = 1,010$
$r^{(4)}=2r^{(3)}$	0 1 0, 0 0 1 0 0 0 0	odtwarzanie	
$2r^{(4)}$	1 0 0, 0 1 0 0 0 0 0		
$-2q^{(3)}+2^{-4}$	− 1 0, 1 0 0 1		
$r^{(5)}$	0 0 1, 1 0 1 1 0 0 0	$q_{-4} = 1$	$q^{(4)} = 1,0101$
$2r^{(5)}$	0 1 1, 0 1 1 0 0 0 0		
$-2q^{(4)}+2^{-5}$	− 1 0, 1 0 1 0 1		
$r^{(6)}$	0 0 0, 1 0 1 1 1 0 0	$q_{-5} = 1$	$q^{(5)} = 1,01011$
$2r^{(6)}$	0 0 1, 0 1 1 1 0 0 0		
$-2q^{(5)}+2^{-6}$	− 1 0, 1 0 1 1 0 1		
	1 1 0, 1 0 1 1 1 1 0	$q_{-6} = 0$	$q^{(6)} = 1,010110$
$r^{(7)}=2r^{(6)}$	0 0 1, 0 1 1 1 0 0 0	odtwarzanie	

$2r^{(7)}$	0 1 0, 1 1 1 0 0 0 0		
$-2q^{(6)}+2^{-7}$	− 1 0, 1 0 1 1 0 0 1		
$r^{(8)}$	0 0 0, 0 0 1 0 1 1 1	$q_{-7} = 1$	$q^{(7)} = 1{,}0101101$

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

c)

L	0 1, 0 0 1 1 1 0 1 0	$q_0 = 1$	$q^{(0)} = 1,$
$r^{(1)}=L-1$	0 0 0, 0 0 1 1 1 0 1 0		
$2r^{(1)}$	0 0 0, 0 1 1 1 0 1 0 0		
$-2q^{(0)}+2^{-1}$	− 1 0, 1		
	1 0 1, 1 1 1 1 0 1 0 0	$q_{-1} = 0$	$q^{(1)} = 1{,}0$
$r^{(2)}=2r^{(1)}$	0 0 0, 0 1 1 1 0 1 0 0	odtwarzanie	
$2r^{(2)}$	0 0 0, 1 1 1 0 1 0 0 0		
$-2q^{(1)}+2^{-2}$	− 1 0, 0 1		
	1 1 0, 1 0 1 0 1 0 0 0	$q_{-2} = 0$	$q^{(2)} = 1{,}00$
$r^{(3)}=2r^{(2)}$	0 0 0, 1 1 1 0 1 0 0 0	odtwarzanie	
$2r^{(3)}$	0 0 1, 1 1 0 1 0 0 0 0		
$-2q^{(2)}+2^{-3}$	− 1 0, 0 0 1		
	1 1 1, 1 0 1 1 0 0 0 0	$q_{-3} = 0$	$q^{(3)} = 1{,}000$
$r^{(4)}=2r^{(3)}$	0 0 1, 1 1 0 1 0 0 0 0	odtwarzanie	
$2r^{(4)}$	0 1 1, 1 0 1 0 0 0 0 0		
$-2q^{(3)}+2^{-4}$	− 1 0, 0 0 0 1		
$r^{(5)}$	0 0 1, 1 0 0 1 0 0 0 0	$q_{-4} = 1$	$q^{(4)} = 1{,}0001$
$2r^{(5)}$	0 1 1, 0 0 1 0 0 0 0 0		
$-2q^{(4)}+2^{-5}$	− 1 0, 0 0 1 0 1		
$r^{(6)}$	0 0 0, 1 1 1 1 1 0 0 0	$q_{-5} = 1$	$q^{(5)} = 1{,}00011$
$2r^{(6)}$	0 0 1, 1 1 1 1 0 0 0 0		
$-2q^{(5)}+2^{-6}$	− 1 0, 0 0 1 1 0 1		
	1 1 1, 1 0 1 1 1 1 0 0	$q_{-6} = 0$	$q^{(6)} = 1{,}000110$
$r^{(7)}=2r^{(6)}$	0 0 1, 1 1 1 1 0 0 0 0		
$2r^{(7)}$	0 1 1, 1 1 1 0 0 0 0 0		
$-2q^{(6)}+2^{-7}$	− 1 0, 0 0 1 1 0 0 1		
$r^{(8)}$	0 0 1, 1 0 1 0 1 1 1 0	$q_{-7} = 1$	$q^{(7)} = 1{,}0001101$
$2r^{(8)}$	0 1 1, 0 1 0 1 1 1 0 0		
$-2q^{(7)}+2^{-8}$	− 1 0, 0 0 1 1 0 1 0 1		
$r^{(9)}$	0 0 1, 0 0 1 0 0 1 1 1	$q_{-8} = 1$	$q^{(8)} = 1{,}00011011$

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

d)

L	1 1, 1 1 0 0 1 0 0 1 0 0	$q_0 = 1$	$q^{(0)} = 1$
$r^{(1)}=L-1$	0 1 0, 1 1 0 0 1 0 0 1 0 0		
$2r^{(1)}$	1 0 1, 1 0 0 1 0 0 1 0 0 0		
$-2q^{(0)}+2^{-1}$	− 1 0, 1		
$r^{(2)}$	0 1 1, 0 0 0 1 0 0 1 0 0 0	$q_{-1} = 1$	$q^{(1)} = 1{,}1$
$2r^{(2)}$	1 1 0, 0 0 1 0 0 1 0 0 0 0		
$-2q^{(1)}+2^{-2}$	− 1 1, 0 1		
$r^{(3)}$	0 1 0, 1 1 1 0 0 1 0 0 0 0	$q_{-2} = 1$	$q^{(2)} = 1{,}11$
$2r^{(3)}$	1 0 1, 1 1 0 0 1 0 0 0 0 0		
$-2q^{(2)}+2^{-3}$	− 1 1, 1 0 1		
$r^{(4)}$	0 1 0, 0 0 1 0 1 0 0 0 0 0	$q_{-3} = 1$	$q^{(3)} = 1{,}111$
$2r^{(4)}$	1 0 0, 0 1 0 1 0 0 0 0 0 0		
$-2q^{(3)}+2^{-4}$	− 1 1, 1 1 0 1		
$r^{(5)}$	0 0 0, 1 0 0 0 0 0 0 0 0 0	$q_{-4} = 1$	$q^{(4)} = 1{,}1111$
$2r^{(5)}$	0 0 1, 0 0 0 0 0 0 0 0 0 0		
$-2q^{(4)}+2^{-5}$	− 1 1, 1 1 1 0 1		
	1 0 1, 0 0 0 1 1 0 0 0 0 0	$q_{-5} = 0$	$q^{(5)} = 1{,}11110$
$r^{(6)}=2r^{(5)}$	0 0 1, 0 0 0 0 0 0 0 0 0 0	odtwarzanie	
$2r^{(6)}$	0 1 0, 0 0 0 0 0 0 0 0 0 0		
$-2q^{(5)}+2^{-6}$	− 1 1, 1 1 1 0 0 1		
	1 1 0, 1 0 0 1 1 1 0 0 0 0	$q_{-6} = 0$	$q^{(6)} = 1{,}111100$
$r^{(7)}=2r^{(6)}$	0 1 0, 0 0 0 0 0 0 0 0 0 0	odtwarzanie	
$2r^{(7)}$	1 0 0, 0 0 0 0 0 0 0 0 0 0		
$-2q^{(6)}+2^{-7}$	− 1 1, 1 1 1 0 0 0 1		
$r^{(8)}$	0 0 0, 0 0 0 1 1 1 1 0 0 0	$q_{-7} = 1$	$q^{(7)} = 1{,}1111001$
$2r^{(8)}$	0 0 0, 0 0 1 1 1 1 0 0 0 0		
$-2q^{(7)}+2^{-8}$	− 1 1, 1 1 1 0 0 1 0 1		
	1 0 0, 0 1 0 1 0 1 1 1 0 0	$q_{-8} = 0$	$q^{(8)} = 1{,}11110010$
$r^{(9)}=2r^{(8)}$	0 0 0, 0 0 1 1 1 1 0 0 0 0	odtwarzanie	
$2r^{(9)}$	0 0 0, 0 1 1 1 1 0 0 0 0 0		
$-2q^{(8)}+2^{-9}$	− 1 1, 1 1 1 0 0 1 0 0 1		
	1 0 0, 1 0 0 1 0 0 1 1 1 0	$q_{-9} = 0$	$q^{(9)} = 1{,}111100100$
$r^{(10)}=2r^{(9)}$	0 0 0, 0 1 1 1 1 0 0 0 0 0	odtwarzanie	
$2r^{(10)}$	0 0 0, 1 1 1 1 0 0 0 0 0 0		
$-2q^{(9)}+2^{-10}$	− 1 1, 1 1 1 0 0 1 0 0 0 1		
$r^{(11)}$	1 0 1, 1 0 0 0 1 0 1 1 1 1	$q_{-10} = 0$	$q^{(10)} = 1{,}1111001000$

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

Zadanie 6.8

a)

L	0 1, 0 1 1 0 1 1 1 0 1 1 1 1 0 1 1 0 0 1	$q_0 = 1$	$q^{(0)} = 1$
$r^{(1)}=L-1$	0 0 0, 0 1 1 0 1 1 1 0 1 1 1 1 0 1 1 0 0 1		
$2r^{(1)}$	0 0 0, 1 1 0 1 1 1 0 1 1 1 1 0 1 1 0 0 1 0		
$-2q^{(0)}+2^{-1}$	− 1 0, 1		
	1 1 0, 0 1 0 1 1 1 0 1 1 1 1 0 1 1 0 0 1 0	$q_{-1} = 0$	$q^{(1)} = 1{,}0$
$r^{(2)}=2r^{(1)}$	0 0 0, 1 1 0 1 1 1 0 1 1 1 1 0 1 1 0 0 1 0	odtwarzanie	
$2r^{(2)}$	0 0 1, 1 0 1 1 1 0 1 1 1 1 0 1 1 0 0 1 0 0		
$-2q^{(1)}+2^{-2}$	− 1 0, 0 1		
	1 1 1, 0 1 1 1 1 0 1 1 1 1 0 1 1 0 0 1 0 0	$q_{-2} = 0$	$q^{(2)} = 1{,}00$
$r^{(3)}=2r^{(2)}$	0 0 1, 1 0 1 1 1 0 1 1 1 1 0 1 1 0 0 1 0 0	odtwarzanie	
$2r^{(3)}$	0 1 1, 0 1 1 1 0 1 1 1 1 0 1 1 0 0 1 0 0 0		
$-2q^{(2)}+2^{-3}$	− 1 0, 0 0 1		
$r^{(4)}$	0 0 1, 0 1 0 1 0 1 1 1 1 0 1 1 0 0 1 0 0 0	$q_{-3} = 1$	$q^{(3)} = 1{,}001$
$2r^{(4)}$	0 1 0, 1 0 1 0 1 1 1 1 0 1 1 0 0 1 0 0 0 0		
$-2q^{(3)}+2^{-4}$	− 1 0, 0 1 0 1		
$r^{(5)}$	0 0 0, 0 1 0 1 1 1 1 1 0 1 1 0 0 1 0 0 0 0	$q_{-4} = 1$	$q^{(4)} = 1{,}0011$
$2r^{(5)}$	0 0 0, 1 0 1 1 1 1 1 0 1 1 0 0 1 0 0 0 0 0		
$-2q^{(4)}+2^{-5}$	− 1 0, 0 1 1 0 1		
	1 1 0, 0 1 0 1 0 1 1 0 1 1 0 0 1 0 0 0 0 0	$q_{-5} = 0$	$q^{(5)} = 1{,}00110$
$r^{(6)}=2r^{(5)}$	0 0 0, 1 0 1 1 1 1 1 0 1 1 0 0 1 0 0 0 0 0	odtwarzanie	
$2r^{(6)}$	0 0 1, 0 1 1 1 1 1 0 1 1 0 0 1 0 0 0 0 0 0		
$-2q^{(5)}+2^{-6}$	− 1 0, 0 1 1 0 0 1		
	1 1 1, 0 0 0 1 1 0 0 1 1 0 0 1 0 0 0 0 0 0	$q_{-6} = 0$	$q^{(6)} = 1{,}001100$
$r^{(7)}=2r^{(6)}$	0 0 1, 0 1 1 1 1 1 0 1 1 0 0 1 0 0 0 0 0 0	odtwarzanie	
$2r^{(7)}$	0 1 0, 1 1 1 1 1 0 1 1 0 0 1 0 0 0 0 0 0 0		
$-2q^{(6)}+2^{-7}$	− 1 0, 0 1 1 0 0 0 1		
$r^{(8)}$	0 0 0, 1 0 0 1 1 0 0 1 0 0 1 0 0 0 0 0 0 0	$q_{-7} = 1$	$q^{(7)} = 1{,}0011001$
$2r^{(8)}$	0 0 1, 0 0 1 1 0 0 1 0 0 1 0 0 0 0 0 0 0 0		
$-2q^{(7)}+2^{-8}$	− 1 0, 0 1 1 0 0 1 0 1		
	1 1 0, 1 1 0 0 1 1 0 1 0 1 0 0 0 0 0 0 0 0	$q_{-8} = 0$	$q^{(8)} = 1{,}00110010$
$r^{(9)}=2r^{(8)}$	0 0 1, 0 0 1 1 0 0 1 0 0 1 0 0 0 0 0 0 0 0		
$2r^{(9)}$	0 1 0, 0 1 1 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0		
$-2q^{(8)}+2^{-9}$	− 1 0, 0 1 1 0 0 1 0 0 1		
$r^{(10)}$	0 0 0, 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	$q_{-9} = 1$	$q^{(9)} = 1{,}001100101$

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

b)

L	1 1, 0 0 1 1 0 0 1 1 0 1 1 0 0 1	$q_0 = 1$	$q^{(0)} = 1,$
$r^{(1)}=L-1$	0 1 0, 0 0 1 1 0 0 1 1 0 1 1 0 0 1		
$2r^{(1)}$	1 0 0, 0 1 1 0 0 1 1 0 1 1 0 0 1 0		
$-2q^{(0)}+2^{-1}$	− 1 0, 1		

$r^{(2)}$	0 0 1, 1 1 1 0 0 1 1 0 1 1 0 0 1 0	$q_{-1} = 1$	$q^{(1)} = 1{,}1$
$2r^{(2)}$	0 1 1, 1 1 0 0 1 1 0 1 1 0 0 1 0 0		
$-2q^{(1)}+2^{-2}$	− 1 1, 0 1		
$r^{(3)}$	0 0 0, 1 0 0 0 1 1 0 1 1 0 0 1 0 0	$q_{-2} = 1$	$q^{(2)} = 1{,}11$
$2r^{(3)}$	0 0 1, 0 0 0 1 1 0 1 1 0 0 1 0 0 0		
$-2q^{(2)}+2^{-3}$	− 1 1, 1 0 1		
	1 0 1, 0 1 1 1 1 0 1 1 0 0 1 0 0 0	$q_{-3} = 0$	$q^{(3)} = 1{,}110$
$r^{(4)}=2r^{(3)}$	0 0 1, 0 0 0 1 1 0 1 1 0 0 1 0 0 0	odtwarzanie	
$2r^{(4)}$	0 1 0, 0 0 1 1 0 1 1 0 0 1 0 0 0 0		
$-2q^{(3)}+2^{-4}$	− 1 1, 1 0 0 1		
	1 1 0, 1 0 1 0 0 1 1 0 0 1 0 0 0 0	$q_{-4} = 0$	$q^{(4)} = 1{,}1100$
$r^{(5)}=2r^{(4)}$	0 1 0, 0 0 1 1 0 1 1 0 0 1 0 0 0 0	odtwarzanie	
$2r^{(5)}$	1 0 0, 0 1 1 0 1 1 0 0 1 0 0 0 0 0		
$-2q^{(4)}+2^{-5}$	− 1 1, 1 0 0 0 1		
$r^{(6)}$	0 0 0, 1 1 1 0 0 1 0 0 1 0 0 0 0 0	$q_{-5} = 1$	$q^{(5)} = 1{,}11001$
$2r^{(6)}$	0 0 1, 1 1 0 0 1 0 0 1 0 0 0 0 0 0		
$-2q^{(5)}+2^{-6}$	− 1 1, 1 0 0 1 0 1		
	1 1 0, 0 0 1 1 0 1 0 1 0 0 0 0 0 0	$q_{-6} = 0$	$q^{(6)} = 1{,}110010$
$r^{(7)}=2r^{(6)}$	0 0 1, 1 1 0 0 1 0 0 1 0 0 0 0 0 0		
$2r^{(7)}$	0 1 1, 1 0 0 1 0 0 1 0 0 0 0 0 0 0		
$-2q^{(6)}+2^{-7}$	− 1 1, 1 0 0 1 0 0 1		
$r^{(8)}$	0 0 0, 0 0 0 0 0 0 0 0 0 0 0 0 0 0	$q_{-7} = 1$	$q^{(7)} = 1{,}1100101$

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

c)

L	1 0, 0 0 1 0 0 0 1 1 1 1 0 1 1 0 1 0 0 1	$q_0 = 1$	$q^{(0)} = 1{,}$
$r^{(1)}=L-1$	0 0 1, 0 0 1 0 0 0 1 1 1 1 0 1 1 0 1 0 0 1		
$2r^{(1)}$	0 1 0, 0 1 0 0 0 1 1 1 1 0 1 1 0 1 0 0 1 0		
$-2q^{(0)}+2^{-1}$	− 1 0, 1		
	1 1 1, 1 1 0 0 0 1 1 1 1 0 1 1 0 1 0 0 1 0	$q_{-1} = 0$	$q^{(1)} = 1{,}0$
$r^{(2)}=2r^{(1)}$	0 1 0, 0 1 0 0 0 1 1 1 1 0 1 1 0 1 0 0 1 0	odtwarzanie	
$2r^{(2)}$	1 0 0, 1 0 0 0 1 1 1 1 0 1 1 0 1 0 0 1 0 0		
$-2q^{(1)}+2^{-2}$	− 1 0, 0 1		
$r^{(3)}$	0 1 0, 0 1 0 0 1 1 1 1 0 1 1 0 1 0 0 1 0 0	$q_{-2} = 1$	$q^{(2)} = 1{,}01$
$2r^{(3)}$	1 0 0, 1 0 0 1 1 1 1 0 1 1 0 1 0 0 1 0 0 0		
$-2q^{(2)}+2^{-3}$	− 1 0, 1 0 1		
$r^{(4)}$	0 0 1, 1 1 1 1 1 1 1 0 1 1 0 1 0 0 1 0 0 0	$q_{-3} = 1$	$q^{(3)} = 1{,}011$
$2r^{(4)}$	0 1 1, 1 1 1 1 1 1 1 0 1 1 0 1 0 0 1 0 0 0 0		
$-2q^{(3)}+2^{-4}$	− 1 0, 1 1 0 1		
$r^{(5)}$	0 0 1, 0 0 1 0 1 1 0 1 1 0 1 0 0 1 0 0 0 0	$q_{-4} = 1$	$q^{(4)} = 1{,}0111$
$2r^{(5)}$	0 1 0, 0 1 0 1 1 0 1 1 0 1 0 0 1 0 0 0 0 0		
$-2q^{(4)}+2^{-5}$	− 1 0, 1 1 1 0 1		
	1 1 1, 0 1 1 1 0 0 1 1 0 1 0 0 1 0 0 0 0 0	$q_{-5} = 0$	$q^{(5)} = 1{,}01110$

$r^{(6)}=2r^{(5)}$	0 1 0, 0 1 0 1 1 0 1 1 0 1 0 0 1 0 0 0 0 0	odtwarzanie	
$2r^{(6)}$	1 0 0, 1 0 1 1 0 1 1 0 1 0 0 1 0 0 0 0 0 0		
$-2q^{(5)}+2^{-6}$	− 1 0, 1 1 1 0 0 1		
$r^{(7)}$	0 0 1, 1 1 0 1 0 0 1 0 1 0 0 1 0 0 0 0 0 0	$q_{-6}=1$	$q^{(6)}=1{,}011101$
$2r^{(7)}$	0 1 1, 1 0 1 0 0 1 0 1 0 0 1 0 0 0 0 0 0 0		
$-2q^{(6)}+2^{-7}$	− 1 0, 1 1 1 0 1 0 1		
$r^{(8)}$	0 0 0, 1 0 1 1 1 0 1 1 0 0 1 0 0 0 0 0 0 0	$q_{-7}=1$	$q^{(7)}=1{,}0111011$
$2r^{(8)}$	0 0 1, 0 1 1 1 0 1 1 0 0 1 0 0 0 0 0 0 0 0		
$-2q^{(7)}+2^{-8}$	− 1 0, 1 1 1 0 1 1 0 1		
	1 1 0, 1 0 0 0 1 0 0 1 0 1 0 0 0 0 0 0 0 0	$q_{-8}=0$	$q^{(8)}=1{,}01110110$
$r^{(9)}=2r^{(8)}$	0 0 1, 0 1 1 1 0 1 1 0 0 1 0 0 0 0 0 0 0 0	odtwarzanie	
$2r^{(9)}$	0 1 0, 1 1 1 0 1 1 0 0 1 0 0 0 0 0 0 0 0 0		
$-2q^{(8)}+2^{-9}$	− 1 0, 1 1 1 0 1 1 0 0 1		
$r^{(10)}$	0 0 0, 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	$q_{-9}=1$	$q^{(9)}=1{,}011101101$

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

d)

L	0 1, 1 1 0 0 1 0 1 0 0 0 1 1 1 0 1 0 0 1	$q_0=1$	$q^{(0)}=1$
$r^{(1)}=L-1$	0 0 0, 1 1 0 0 1 0 1 0 0 0 1 1 1 0 1 0 0 1		
$2r^{(1)}$	0 0 1, 1 0 0 1 0 1 0 0 0 1 1 1 0 1 0 0 1 0		
$-2q^{(0)}+2^{-1}$	− 1 0, 1		
	1 1 1, 0 0 0 1 0 1 0 0 0 1 1 1 0 1 0 0 1 0	$q_{-1}=0$	$q^{(1)}=1{,}0$
$r^{(2)}=2r^{(1)}$	0 0 1, 1 0 0 1 0 1 0 0 0 1 1 1 0 1 0 0 1 0	odtwarzanie	
$2r^{(2)}$	0 1 1, 0 0 1 0 1 0 0 0 1 1 1 0 1 0 0 1 0 0		
$-2q^{(1)}+2^{-2}$	− 1 0, 0 1		
$r^{(3)}$	0 0 0, 1 1 1 0 1 0 0 0 1 1 1 0 1 0 0 1 0 0	$q_{-2}=1$	$q^{(2)}=1{,}01$
$2r^{(3)}$	0 0 1, 1 1 0 1 0 0 0 1 1 1 0 1 0 0 1 0 0 0		
$-2q^{(2)}+2^{-3}$	− 1 0, 1 0 1		
	1 1 1, 0 0 1 1 0 0 0 1 1 1 0 1 0 0 1 0 0 0	$q_{-3}=0$	$q^{(3)}=1{,}010$
$r^{(4)}=2r^{(3)}$	0 0 1, 1 1 0 1 0 0 0 1 1 1 0 1 0 0 1 0 0 0		
$2r^{(4)}$	0 1 1, 1 0 1 0 0 0 1 1 1 0 1 0 0 1 0 0 0 0		
$-2q^{(3)}+2^{-4}$	− 1 0, 1 0 0 1		
$r^{(5)}$	0 0 1, 0 0 0 1 0 0 1 1 1 0 1 0 0 1 0 0 0 0	$q_{-4}=1$	$q^{(4)}=1{,}0101$
$2r^{(5)}$	0 1 0, 0 0 1 0 0 1 1 1 0 1 0 0 1 0 0 0 0 0		
$-2q^{(4)}+2^{-5}$	− 1 0, 1 0 1 0 1		
	1 1 1, 0 1 1 1 1 1 1 1 0 1 0 0 1 0 0 0 0 0	$q_{-5}=0$	$q^{(5)}=1{,}01010$
$r^{(6)}=2r^{(5)}$	0 1 0, 0 0 1 0 0 1 1 1 0 1 0 0 1 0 0 0 0 0	odtwarzanie	
$2r^{(6)}$	1 0 0, 0 1 0 0 1 1 1 0 1 0 0 1 0 0 0 0 0 0		
$-2q^{(5)}+2^{-6}$	− 1 0, 1 0 1 0 0 1		
$r^{(7)}$	0 0 1, 1 0 1 0 1 0 1 0 1 0 0 1 0 0 0 0 0 0	$q_{-6}=1$	$q^{(6)}=1{,}010101$
$2r^{(7)}$	0 1 1, 0 1 0 1 0 1 0 1 0 0 1 0 0 0 0 0 0 0		
$-2q^{(6)}+2^{-7}$	− 1 0, 1 0 1 0 1 0 1		
$r^{(8)}$	0 0 0, 1 0 1 0 1 0 1 1 0 0 1 0 0 0 0 0 0 0	$q_{-7}=1$	$q^{(7)}=1{,}0101011$

$2r^{(8)}$	0 0 1, 0 1 0 1 0 1 1 0 0 1 0 0 0 0 0 0 0 0		
$-2q^{(7)}+2^{-8}$	− 1 0, 1 0 1 0 1 1 0 1		
	1 1 0, 1 0 1 0 1 0 0 1 0 1 0 0 0 0 0 0 0 0	$q_{-8} = 0$	$q^{(8)} = 1{,}01010110$
$r^{(9)}=2r^{(8)}$	0 0 1, 0 1 0 1 0 1 1 0 0 1 0 0 0 0 0 0 0 0		
$2r^{(9)}$	0 1 0, 1 0 1 0 1 1 0 0 1 0 0 0 0 0 0 0 0 0		
$-2q^{(8)}+2^{-9}$	− 1 0, 1 0 1 0 1 1 0 0 1		
$r^{(10)}$	0 0 0, 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	$q_{-9} = 1$	$q^{(9)} = 1{,}010101101$

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

Literatura

1. Biernat J., *Metody i układy arytmetyki cyfrowej*, Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław 2001.
2. Brent R.P., Zimmermann P., *Modern Computer Arithmetic*, Cambridge University Press, Cambridge 2010.
3. Flynn M., Oberman S., *Advanced Computer Arithmetic Design*, Wiley, New York 2001.
4. *IEEE Standard for Binary Floating-Point Arithmetic*, ANSI/IEEE Std 754-1985, 1985, DOI: 10.1109/IEEESTD.1985.82928.
5. *IEEE Standard for Floating-Point Arithmetic*, IEEE Std 754-2008, 2008, DOI: 10.1109/IEEESTD.2008.4610935.
6. Koren I., *Computer Arithmetic Algorithms*, A.K. Peters Ltd., Natick, Massachusetts 2002.
7. Ogrodzki J., *Wstęp do systemów komputerowych*, Oficyna Wydawnicza Politechniki Warszawskiej, Warszawa 2005.
8. Parhami B., *Computer Arithmetic. Algorithms and Hardware Design*, Oxford University Press, New York 2010.
9. Pochopień B., *Arytmetyka w systemach cyfrowych*, EXIT, Warszawa 2004.
10. Pochopień B., Stańczyk U., Wróbel E., *Arytmetyka systemów cyfrowych w teorii i praktyce*, Wydawnictwo Politechniki Śląskiej, Gliwice 2012.
11. Vladutiu M., *Computer Arithmetic: Algorithms and Hardware Implementations*, Springer-Verlag, Berlin Heidelberg 2012.
12. Wantuch E., Drabowski M., *Wstęp do informatyki*, Wydawnictwo Politechniki Krakowskiej, Kraków 2006.

Spis tabel

Spis rysunków

Streszczenie

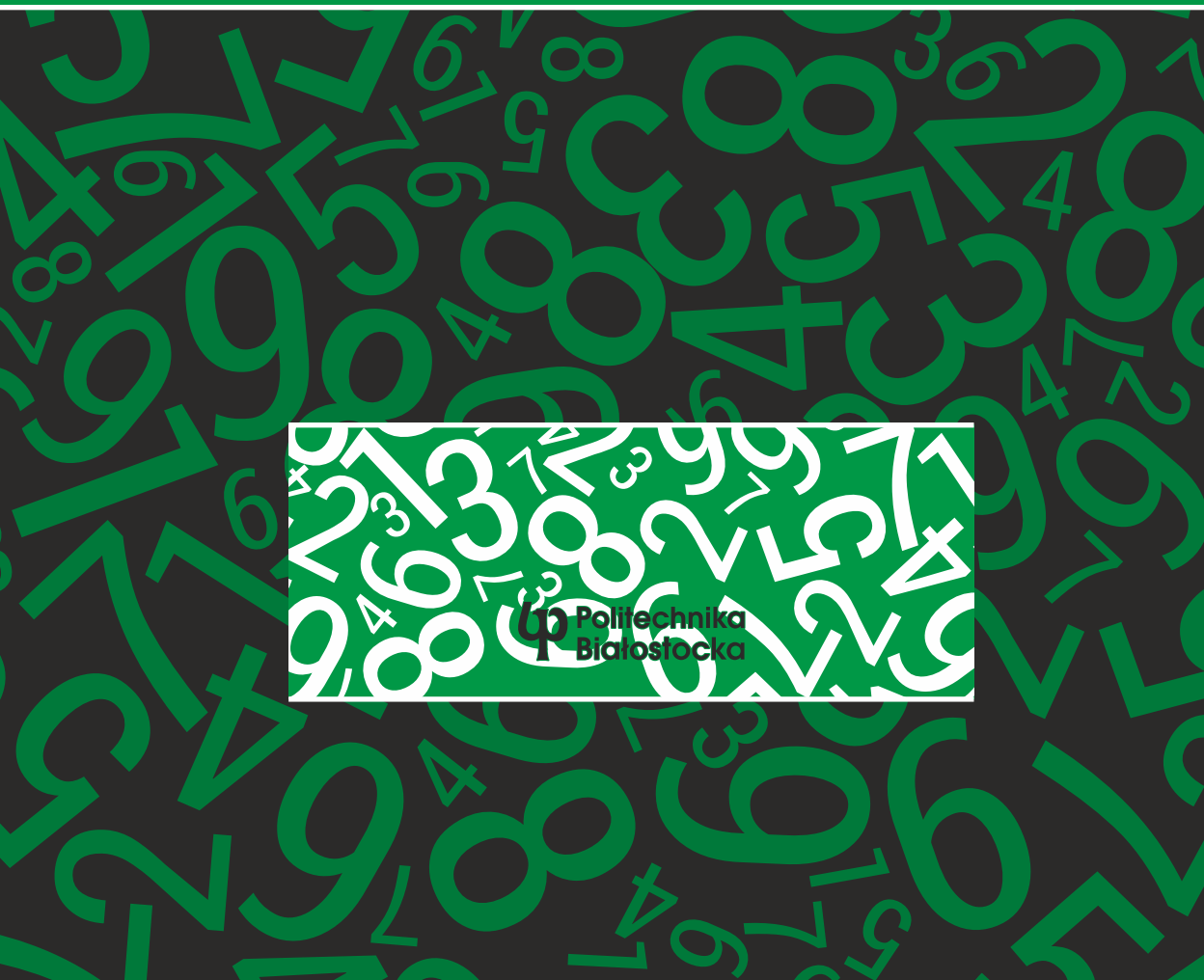
W książce omówione zostały zagadnienia dotyczące zmiennoprzecinkowej reprezentacji liczb w systemach komputerowych oraz arytmetyki zmiennoprzecinkowej. Materiał zawarty w skrypcie dokładnie opisuje zasady reprezentacji liczb w formatach zmiennoprzecinkowych, przedstawia standardowe formaty binarnej oraz dziesiętnej reprezentacji zmiennoprzecinkowej zgodne ze standardem IEEE 754-2008, prezentuje i wyjaśnia algorytmy wykonywania podstawowych operacji arytmetycznych na liczbach przechowywanych w formatach zmiennoprzecinkowych. Książka zawiera wiele dokładnie skomentowanych przykładów oraz zadań z rozwiązaniami opatrzonych wskazówkami i obszernymi komentarzami wyjaśniającymi, które ułatwiają dogłębne zrozumienie i przyswojenie materiału. Zaproponowano także zestawy zadań do samodzielnego rozwiązywania.

Skrypt przeznaczony jest dla studentów uczelni technicznych, stanowi materiał pomocniczy do ćwiczeń z przedmiotów, które realizują między innymi zagadnienia związane z arytmetyką komputerową, kodowaniem i reprezentacją liczb. Ze względu na liczne przykłady i szczegółowe wyjaśnienia, książka może być wykorzystana do samodzielnej pracy w celu przygotowania się do zajęć, sprawdzianów i kolokwiów zaliczeniowych. Może być też przydatna szerokiemu gronu czytelników zainteresowanych podstawami reprezentacji, kodowania i przetwarzania danych w systemie komputerowym.

Abstract

The book discusses issues related to the floating-point number representation and floating-point arithmetic. The book describes in detail the methods of floating-point representation, presents the standard formats of binary and decimal floating-point representations in accordance with the IEEE 754-2008 floating-point arithmetic standard, describes and explains the working of algorithms realizing basic arithmetic operations on floating-point numbers. The book contains a large number of carefully commented examples and solved exercises to facilitate in-depth understanding and assimilation of the material. Sets of tasks for self-solving were also proposed.

The book is addressed to students of technical universities, it can be helpful in learning subjects related to computer arithmetic, encoding and number representation. Due to the numerous examples and detailed explanations, the book is well suited for self-preparation for classes and tests. It can also be useful to a wide group of readers interested in the basics of data representation, encoding and computer arithmetic.


Politechnika
Białostocka