Imię i nazwisko:

Maciej Mysków 272794

Kacper Kostrzewa 272855

Projekt: Organizacja i Architektura Komputerów

Temat: Liczby typu Posit

Spis treści:

1. Wstęp

Liczby Posit to koncepcja reprezentacji liczb w komputerach, zaproponowana przez Johna Gustafsona. Jest to sposób na reprezentację liczb zmiennoprzecinkowych, który ma na celu eliminację błędów zaokrągleń charakterystycznych dla standardowej reprezentacji zmiennoprzecinkowej w komputerach.

W przeciwieństwie do standardowej reprezentacji zmiennoprzecinkowej, gdzie liczby są zapisywane jako mantysa i cecha, liczby Posit mają trzy główne cechy:

- **Unormowane wartości**: Podobnie jak w reprezentacji zmiennoprzecinkowej, liczby są unormowane, co oznacza, że są przechowywane zgodnie z najwyższym stopniem precyzji, jaki można osiągnąć.
- **Elastyczność:** Liczby Posit są elastyczne, co oznacza, że można nimi reprezentować zarówno bardzo małe, jak i bardzo duże liczby.
- **Błąd reprezentacji**: System liczbowy Posit ma na celu zminimalizowanie błędów reprezentacji występujących w tradycyjnych systemach zmiennoprzecinkowych. Dzięki czemu na różnych maszynach otrzymujemy ten sam wynik a nie jak w przypadku standardu IEEE - 754, gdzie przy dużych operandach mieliśmy rozbieżności pomiędzy wynikami, wartościami.

Koncepcja liczb Posit zdobywa coraz większą uwagę, ponieważ może prowadzić do bardziej niezawodnych i precyzyjnych obliczeń numerycznych. Jednak nadal pozostaje wiele kwestii do rozważenia i implementacji praktycznych rozwiązań związanych z tą koncepcją.

2. Charakterystyka Liczb Posit - zalety/wady

ZALETY	WADY
• **Elastyczność w zakresie reprezentacji:** Liczby Posit pozwalają na reprezentację zarówno bardzo małych, jak i bardzo dużych liczb z wysoką precyzją. Ich elastyczna struktura umożliwia dynamiczne dostosowanie się do zakresu wartości. • **Minimalizacja błędów zaokrągleń:** Dzięki specjalnej strukturze, liczby Posit eliminują wiele problemów związanych z	• **Brak standardu:** W przeciwieństwie do tradycyjnych systemów zmiennoprzecinkowych, liczby Posit nie mają ustalonego standardu, co może prowadzić do problemów z interoperacyjnością między różnymi platformami i aplikacjami. • **Konieczność dalszych badań i rozwoju:** Mimo obiecujących cech, liczby Posit wciąż

błędami zaokrągleń, które występują w tradycyjnych systemach zmiennoprzecinkowych ● **Brak zerowego dzielenia**: W odróżnieniu od systemów zmiennoprzecinkowych, liczb Posit nie występuje problem zerowego dzielenia, co może pomóc w uniknięciu błędów i zapobieganiu awariom w obliczeniach.	wymagają dalszych badań i rozwoju, aby pełni wykorzystać ich potencjał i rozwiązać ewentualne problemy i wyzwania związane z ich implementacją i stosowaniem. ● **Brak powszechnej akceptacji:** Ponieważ liczby Posit to stosunkowo nowa koncepcja, mogą istnieć opory wobec ich przyjęcia w niektórych środowiskach, co może opóźnić ich powszechne stosowanie i rozwój. ● **Nowy standard Posit jest bardziej zasobożerny:** wymaga większej mocy obliczeniowej co skutkuje zwiększeniem kosztów eksploatacji.

3. Różnice między Posit a IEEE-754

- W przeciwieństwie do standardu IEEE 754, w liczbach typu Posit wprowadzono dodatkowe pole typu Regime, które kończy się w momencie napotkanie pierwszego przeciwnego bitu. Służy do osiągnięcia większej dokładności wyniku
- W liczbach typu posit, zakres wartości jest zwykle większy niż w IEEE 754 dzięki innej strukturze reprezentacji.
- Liczby typu posit mogą być bardziej efektywne pod względem pamięci, ponieważ mogą reprezentować szeroki zakres wartości przy mniejszej liczbie bitów w niektórych przypadkach w porównaniu do standardu IEEE 754.
- Standard IEEE 754 jest szeroko stosowany i zaimplementowany w wielu platformach i systemach obliczeniowych. Z kolei liczby typu posit, choć

rozwijane i badane, mogą nie być jeszcze tak powszechnie dostępne i zaimplementowane w porównaniu do IEEE 754.

4. Implementacja matematyczna

- Zrozumienie notacji

Notacja liczby w standardzie Posit to: ***Posit<n, es>***, gdzie:
n - *długość liczby w bitach*
es - *część liczby na zakodowanie wykładnika*

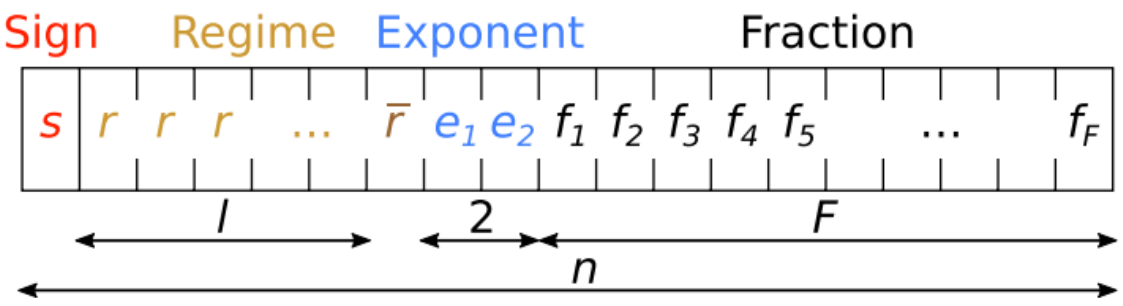
W systemach cyfrowych es jest stała i wynosi 2. Stąd możemy zapisać:

Posit <n,2>

Upraszcza to konwersje między różnymi rozmiarami pozycji i redukuje konstrukcję układy/sprzętu arytmetycznego.

W omawianym standardzie mamy do dyspozycji takie same pola jak w IEEE-754: znak (s), wykładnik (e/es), mantysa (m/f) plus dodatkowe pole, które **nie** występuje w IEEE-754 pole: **Regime.** Jako że długość tego pola nie jest z góry określona, to do wyznaczenia momentu zaczęcia się części na wykładnik stosuje się technikę wykrywania napotkania przeciwnego bitu.

Powyższe rozważanie ilustruje poniższa grafika:



rys.1 *Generic n-bit posit binary encoding. - Artykuł*

- Wzór dekodowania na dziesiętny i opis zmiennych

Wzór na dekodowanie liczby Posit występuje w dwóch formach. W formie zaprezentowanej w artykula oraz druga, ogólniejsza forma. W projekcie posługujemy się formą drugą, gdyż jest prostsza do dalszych obliczeń i implementacji. Jednak na potrzeby przedstawienia raportu z dotychczasowych prac projektowych zostaną zamieszczone obydwie formy a następnie po konsultacji z prowadzącym zostanie wybrana jedna. Jednak omówiona będzie tylko ogólniejsza forma.

Wzór 1 (forma ogólna): (1)

$$L = (-1)^{s} * u^{k} * 2^{es} * m$$

Wzór 2 (forma z artykułu): (2)

$$p = (1 - 3s + f) * 2^{(1-2s)*(4k+e+s)}$$

gdzie:

l – liczba identycznych bitów w części Regime

$k = -l$, gdy Regime składa się z p zer;

$k = l + 1$, gdy Regima składa się z p jedynek

es lub e – długość pola wykładnika

s lub z – znak

$u = 2^{2^{es}}$

f lub m – mantysa

5. Przykład

Przed przystąpieniem do dekodowania upewniamy się, czy przedstawiona liczba nie jest ujemna **(S = 1)**. Jeśli tak, to przed dekodowaniem, każdy bit (poza znakiem) uzupełniamy do 2. Dopełnienie do 2 na tym etapie oszczędza procesor przez redukcję rozkazów o kosztowne rozkazy porównań. Następnie wyznaczamy składowe ***z***, ***u***, ***k***, ***l,*** **w** oraz ***m***, co pozwoli nam podstawić pod wzór ogólny. Poniżej

zamieszczony przykład z pliku Excel, który będzie udostępniony w załączniku do niniejszego sprawozdania.

Wprowadz Liczbę w foramcie Posit <16, 2>; gdzie n - długość bitów liczby, 2 - długość wykładnika. Długość Regime to 4 bity

S	R				ES		M								
1	1	1	1	0	1	0	1	1	0	1	0	1	0	1	1

REZULTAT:

Konwersja na U2 (bez znaku)

S	R				ES		M								
1	0	0	0	1	0	1	0	0	1	0	1	0	1	0	1

Wyjaśnienie zmiennych
l - Liczba identycznych bitów w polu Regime
k - gdy Regime z zamych "0", to -l; gdy Regime z samych "1"' to l - 1
ES - długość pola wykładnika
M - mantysa dziesiętni
s - znak

Wartości zmiennych z wprowadzonej liczby:	
l	3
k	-3
ES dzisiętnie	1
M dziesiętnie	0,175781
Długość ES	2
u	16

Wzór:

$$L = (-1)^s * u^k * 2^E * m$$

Wynik:
-8,58E-0

rys.2 Dekodowanie Posit - przykład: P<16,2> = 1111010110101011

Zatem przedstawione dekodowanie liczby typu Posit jest bardzo podobne do procesu dekodowanie standardu **IEEE-754**, występują podobne składowe w tym procesie: znak, mantysa i cecha. Różnicą jest nowe pole typu **Regime** oraz konieczność dopełnienia do 2 jeśli przedstawiona liczba jest ujemna w celu zaoszczędzenie operacji porównań.

6. Implementacja w C++

- Opis działania algorytmu - pseudokod (z artykułu)

  Wejście:
  Algorytm przyjmuje na wejściu liczbę x typu PositN, która reprezentuje wartość w formacie pozycyjnym. Typ PositN jest liczbą całkowitą bez znaku o wielkości jednego bajtu.

Wyjście:

Na wyjściu algorytm zwraca liczbę y typu FloatEM, która reprezentuje tę samą wartość co x, ale w formacie float zgodnym ze standardem IEEE 754. Typ FloatEM składa się z pól: znaku, wykładnika i mantysy.

Działanie algorytmu:

1. Algorytm najpierw analizuje najbardziej znaczący bit x w celu określenia znaku liczby. Pozostałe bity są traktowane jako wartość liczby.
2. Sprawdzane jest, czy wartość x jest równa zero. Jeśli tak, algorytm zwraca 0 (jeśli x jest dodatnie) lub NaN (jeśli x jest ujemne), w zależności od znaku x.
3. Jeśli wartość x jest różna od zera, algorytm oblicza jej moduł i wyciąga z niego pola: regime, exp - wykładnik i f/m - mantysa.
4. Następnie obliczany jest znormalizowany wykładnik biased_exp, dodając do sumy pól regime, exp oraz stałej wartości bias (przesunięcie).
5. Na koniec tworzony jest wynik y, składający się z: znaku x, znormalizowanego wykładnika oraz pola mantysy. W razie potrzeby y jest uzupełniane zerami z prawej strony.