Ministerstwo Nauki
i Szkolnictwa Wyższego

Regionalna Inicjatywa Doskonałości w Dyscyplinach
Informatyki, Elektrotechniki, Elektroniki, Automatyki i Robotyki
na Politechnice Częstochowskiej

Projekt w ramach programu Regionalna Inicjatywa Doskonałości, decyzja nr 020/RID/2018/19

Architektura Systemów Komputerowych

Binarne reprezentacje danych oraz arytmetyka systemów komputerowych

dr hab. inż. Łukasz Szustak, prof. PCz
lszustak@icis.pcz.pl

Katedra Informatyki
Wydział Inżynierii Mechanicznej i Informatyki
Politechnika Częstochowska

Ministerstwo Nauki
i Szkolnictwa Wyższego

Regionalna Inicjatywa Doskonałości w Dyscyplinach
Informatyki, Elektrotechniki, Elektroniki, Automatyki i Robotyki
na Politechnice Częstochowskiej

Projekt w ramach programu Regionalna Inicjatywa Doskonałości, decyzja nr 020/RID/2018/19

Zamieszczane materiały służą wyłącznie do celów indywidualnego kształcenia. Nie wyrażam zgody na ich utrwalanie przekazywanie osobom trzecim ani rozpowszechnianie.

dr hab. inż. Łukasz Szustak, prof. PCz
lszustak@icis.pcz.pl

Katedra Informatyki
Wydział Inżynierii Mechanicznej i Informatyki
Politechnika Częstochowska

Wydział Inżynierii Mechanicznej i Informatyki
The Faculty of Mechanical Engineering and Computer Science

Binarna reprezentacja danych

- Komputery działają w systemie binarnym wykonując operacje na grupach bitów, tworzących liczby binarne
- Współczesne komputery operują najczęściej na słowach binarnych, których długość jest wielokrotnością bitów najczęściej 2^i gdzie i=3,4,… (np.. 8, 16, 32, 64)
- Wszystkie dane, na których operuje komputer, są zapisane w postaci ciągów cyfr binarnych – bitów, interpretowanych najczęściej jako liczby binarne
- Dane nieliczbowe (znaki, sygnały) są zapisywane (mapowane) przy użyciu liczb

Podstawowe typy danych

- Podstawowe typy danych:
  - Wartości logiczne (prawda/fałsz)
  - Znaki pisarskie (znakowy/tekstowy)
  - Liczby
    - Całkowite
    - Stałopozycyjne
    - Zmiennopozycyjne
  - Wskaźnikowy/Referencyjny
  - Dźwięki i inne sygnały jednowymiarowe
  - Obrazy
  - Filmy
- **Wszystkie typy danych przechowywane są w postaci binarnej, a następnie są one konwertowane do postaci czytelnej dla użytkownika**

Dane alfanumeryczne

- Każdy znak pisarski jest reprezentowany przez liczbę, stanowiącą jego numer w tablicy kodowej
- Najczęściej używane kody:
  - ASCII - 128 pozycji, w tym małe i wielkie litery alfabetu łacińskiego
  - rozszerzone kody 256-pozycyjne na bazie ASCII - pierwsze 128 pozycji jak w ASCII. następne 128 pozycji zawiera znaki narodowe lub inne symbole
  - problem: różne kody dla różnych części świata
- UNICODE
  - pierwotnie 216, obecnie do 232 możliwych pozycji
  - reprezentacja wszystkich znaków używanych na świecie

Kody ASCII

- Opracowany dla urządzeń dalekopisowych, później przyjęty dla komputerów
- 128 pozycji, w tym 95 znaków widocznych i 33 niewidoczne
- znaki niewidoczne:
  - spacja (kod 32)
  - odstępy i inne kody formatujące
  - kody sterujące transmisją i urządzeniami
- znaki widoczne:
  - Cyfry
  - wielkie i małe litery alfabetu łacińskiego
  - znaki interpunkcyjne
  - podstawowe symbole matematyczne

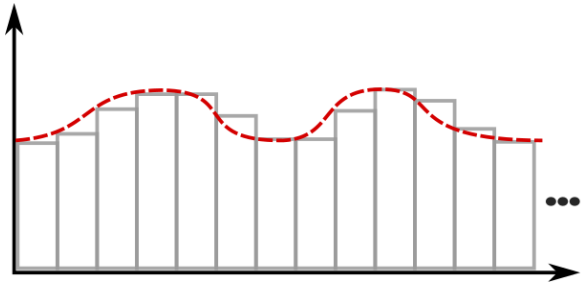
Ciekawostki

- Kody sterujące zajmują pozycje od 0 do 31, w tym
  - CR - powrót na początek wiersza – 13
  - LF - przejście do następnego wiersza – 10
  - inne ważne: HT, FF, BSP, BEL
- spacja – kod 32
- cyfry 0..9 - kody od 48 do 57 (0x30..0x39)
- litery w kolejności alfabetycznej
  - wielkie - 65..90 (0x41..0x5a)
  - małe - 97..122 (0x61..0x7a)
  - dostęp pomiędzy małą i wielką literą wynosi 32 (0x20)
  - pozostałe znaki widoczne zajmują pozycje pomiędzy 32 i 127
- 127 - kod specjalny (kasowanie znaku)

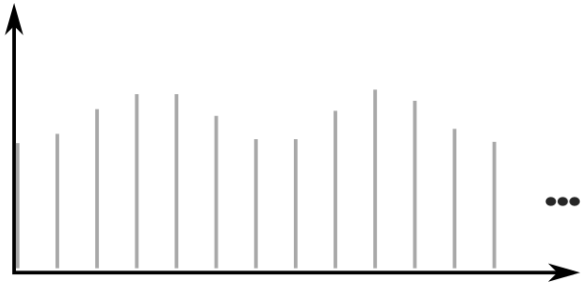
Reprezentacja dźwięków i obrazów

- Sygnał dźwiękowy przetwarzany jest z postaci naturalnej, ciągłej, do reprezentacji numerycznej, czyli ciągu dyskretnych wartości liczbowych:
- Przetwarzanie dźwięku składa się z trzech podstawowych procesów:

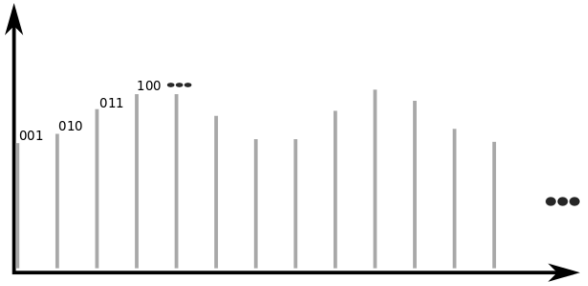
Próbkowania



Kwantyzacji



Kodowania

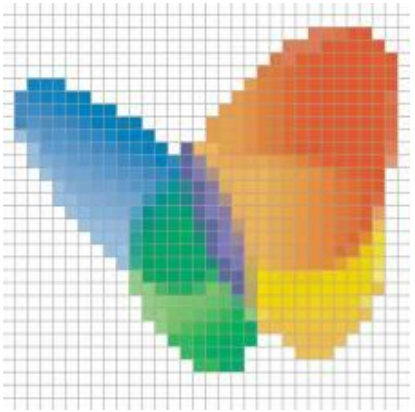


- W efekcie dźwięk zapisany jest w postaci binarnej

Reprezentacja dźwięków i obrazów

- Przykładowo, obraz rastrowy
  - Jest zapisany w postaci prostokątnej macierzy punktów (pikseli)
  - każdemu pikselowi odpowiada jeden kolor
  - kolor reprezentowany w postaci trzech składowych
  - jasności świateł podstawowych (R,G,B)
  - wartości jasności zapisane w postaci liczb

- **W efekcie obraz zapisany jest w postaci binarnej**



Jednostki Informacji

- bit (Binary digIT) - skrót „b"- najmniejsza jednostka informacji, odpowiada informacji TAK-NIE, 1-0, PRAWDA-FAŁSZ
- bajt (byte) - skrót „B" - najmniejsza jednostka informacji **adresowana** przez procesor - obecnie 8 bitów
- słowo (word) - jednostka informacji, na której operuje komputer (1, 2, 4, 8,16 bajtów)
- słowo procesora - jednostka informacji o długości naturalnej dla danego procesora
  - Zazwyczaj rozmiar słowa procesora odpowiada długości rejestrów - obecnie 32 lub 64 bity
- słowo pamięci - jednostka informacji możliwa do przetransmitowania w jednym cyklu transmisji do lub z pamięci (większa długość = szybsza transmisja danych)

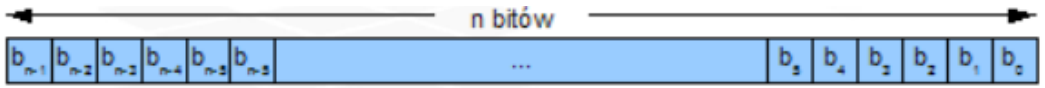
Format danych

- Komputery operują na słowach - ciągach bitów o długościach wyrażonych najczęściej potęgami liczby 2
  - typowe długości słów wynoszą: 8, 16, 32, 64, 128 bitów
  - komputery specjalizowane mogą używać innych formatów danych np. 24 bity
- Niektóre komputery mogą ponadto operować na pojedynczych bitach i ciągach bitowo dowolnych, niewielkich długościach - tzw. polach bitowych
- Dane są najczęściej zapisywane w postaci słów o długości odpowiedniej dla danego typu komputera

Zapisy danych logicznych (Boolowskich)

- Do zapisu danej logicznej wystarcza jeden bit
  - w komputerach obsługujących dane bitowe używa się zapisu jednobitowego
  - Jednakże, w większości komputerów dane logiczne są reprezentowane przez słowa o długości naturalnej dla danego komputera tak samo, jak dane całkowitoliczbowe, np. 32 bity
- Reprezentacja zależą od środowiska i języka programowania
  - wartość "fałsz" jest zwykle reprezentowana przez słowo o wszystkich bitach równych 0
  - "prawda" może być reprezentowana przez:
    - liczbę całkowitą o wartości 1 (język C - wynik operacji)
    - liczbę całkowitą o dowolnej wartości różnej od zera (język C - argument operacji)
- **Różne wzorce bitowe używane w różnych językach oraz korzystanie z operatorów (np. negacji) bitowych zamiast logicznych mogą być przyczyną błędów w programach, w których poszczególne moduły są pisane w różnych językach**

Zapis liczb całkowitych nieujemnych



- Naturalny kod binarny – NKB

$$\nu_{NKB} = \sum_{i-0}^{n-1} b_i \cdot 2^i$$

- W naturalnym kodzie binarnym numer bitu jest równy wykładnikowi jego wagi binarnej

Zapis liczb całkowitych ze znakiem

- U2 - kod uzupełnieniowy do dwóch:

$$v_{u2} = -b_{n-1} \cdot 2^{n-1} + \sum_{i-0}^{n-2} b_i \cdot 2^i$$

- U1 - kod uzupełnieniowy do jedności:

$$v_{u1} = -b_{n-1} \cdot (2^{n-1} - 1) + \sum_{i-0}^{n-2} b_i \cdot 2^i$$

- Znak-moduł:

$$v_{Z-M} = -1^{b_{n-1}} \cdot \sum_{i-0}^{n-1} b_i \cdot 2^i$$

- Zapis spolaryzowany (biased):
  - zwykle przyjmuje się
  - BIAS = 2n-1-1

$$v_B = -BIAS + \sum_{i-0}^{n-1} b_i \cdot 2^i$$

Zapis liczb całkowitych ze znakiem

Zapis	Wartości bitowe			Wartości bajtowe		16bitów		32bity	
	min	zero	max	min	max	min	max	min	max
NKB	00...0	00...0	11...1	0	255	0	65535	0	4294967295
U2	10...0	00...0		-128	127	-32768	32767	-2147483648	2147483647
Z-M	11...1	00...0 10...0	01...1	-127	127	-32767	32767	-2147483647	2147483647
U1	10...0	00...0 11...1	01...1	-127	127	-32768	32767	-2147483647	2147483647
Biased	00...0	01...1	11...1	-127	128	-32768	32768	-2147483648	2147483648

Zapis liczb całkowitych ze znakiem: właściwości

- Reprezentacja zera
  - dwie reprezentacje w kodach U1 i znak-moduł
  - łatwość wykrywania wartości 0
- Symetryczność zakresu (dla liczb ze znakiem)
- Reprezentacja znaku liczby - łatwość rozróżniania znaku
- Zmiana znaku liczby
  - U1 - negacja bitowa
  - U2 - negacja i inkrementacja
  - znak-moduł - negacja bitu znaku
- Łatwość wykonywania operacji arytmetycznych: dodawanie i odejmowanie w U2 realizuje się tak samo, jak w NKB

Zapis zmiennopozycyjny: wprowadzenie

- Zapis dużych liczb lub bardzo małych w normalnej notacji pozycyjnej jest niewygodny, gdyż wymaga sporej ilości cyfr
- Z tego względu takie liczby zapisuje się w sposób następujący:

$$\mathbf{3{,}25 \times 10^{33}}$$

- Powyższy zapis składa się z trzech elementów:
  - **m** mantysy, u nas równej 3,25
  - **p** podstawy systemu, u nas równej 10
  - **c** cechy, u nas równej 33

Zapis zmiennopozycyjny: wprowadzenie

- Wartość liczby zmiennoprzecinkowej obliczamy zgodnie ze wzorem:

$$L = m \times p^{C}$$

- Wzór pozwala obliczyć wartość liczby zmiennoprzecinkowej zapisanej w dowolnym systemie pozycyjnym, a nie tylko dziesiętnym
- Ponieważ położenie przecinka w mantysie nie jest ustalone i może się dowolnie zmieniać, liczbę możemy zapisać na wiele sposobów:

$$325 \times 10^{20} = 32{,}5 \times 10^{21} = 3{,}25 \times 10^{22} = 0{,}325 \times 10^{23}, \text{ itd...}$$

- Oczywiście zmiana położenia przecinka w mantysie wpływa na wartość cechy liczby

Zapis zmiennopozycyjny: wprowadzenie

- Ponieważ liczbę zmiennoprzecinkową można zapisywać w różny sposób, przyjęto tzw. postać znormalizowaną
- Znormalizowana liczba zmiennoprzecinkowa to taka, w której mantysa spełnia nierówność:

$$p > |m| \geq 1$$

- Według tej definicji, który z poniższych zapisów zmiennoprzecinkowych reprezentuje postać znormalizowaną?
  $325\times10^{20} = 32{,}5\times10^{21} = \mathbf{3{,}25\times10^{22}} = 0{,}325\times10^{23}$
- Zera nie da się zapisać w postaci znormalizowanej

Zapis zmiennopozycyjny: wprowadzenie

- Ponieważ liczbę zmiennoprzecinkową można zapisywać w różny sposób, przyjęto tzw. postać znormalizowaną
- Znormalizowana liczba zmiennoprzecinkowa to taka, w której mantysa spełnia nierówność:

$$p > |m| \geq 1$$

- Według tej definicji, który z poniższych zapisów zmiennoprzecinkowych reprezentuje postać znormalizowaną?
  $325\times10^{20} = 32{,}5\times10^{21} = \mathbf{3{,}25\times10^{22}} = 0{,}325\times10^{23}$
- Zera nie da się zapisać w postaci znormalizowanej

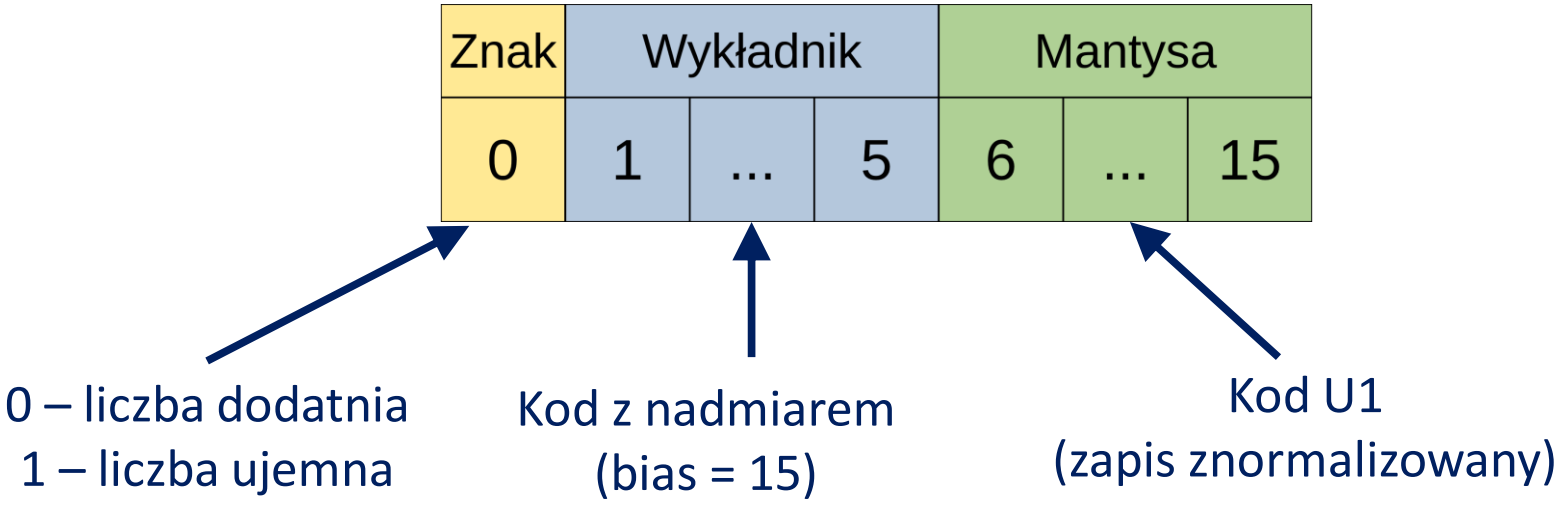
IEEE754

- IEEE754 jest powszechnie stosowanym standardem reprezentacji binarnej oraz operacji zmiennoprzecinkowych:
  - Pół precyzja (half precision) – 16 bitów

  - Pojedyncza precyzja (single precision) – 32 bity

  - Podwójna precyzja (double precision) – 64 bity

IEEE754 – half precision



$$L = (-1)^z \cdot 2^w \cdot m$$

IEEE754 – single precision

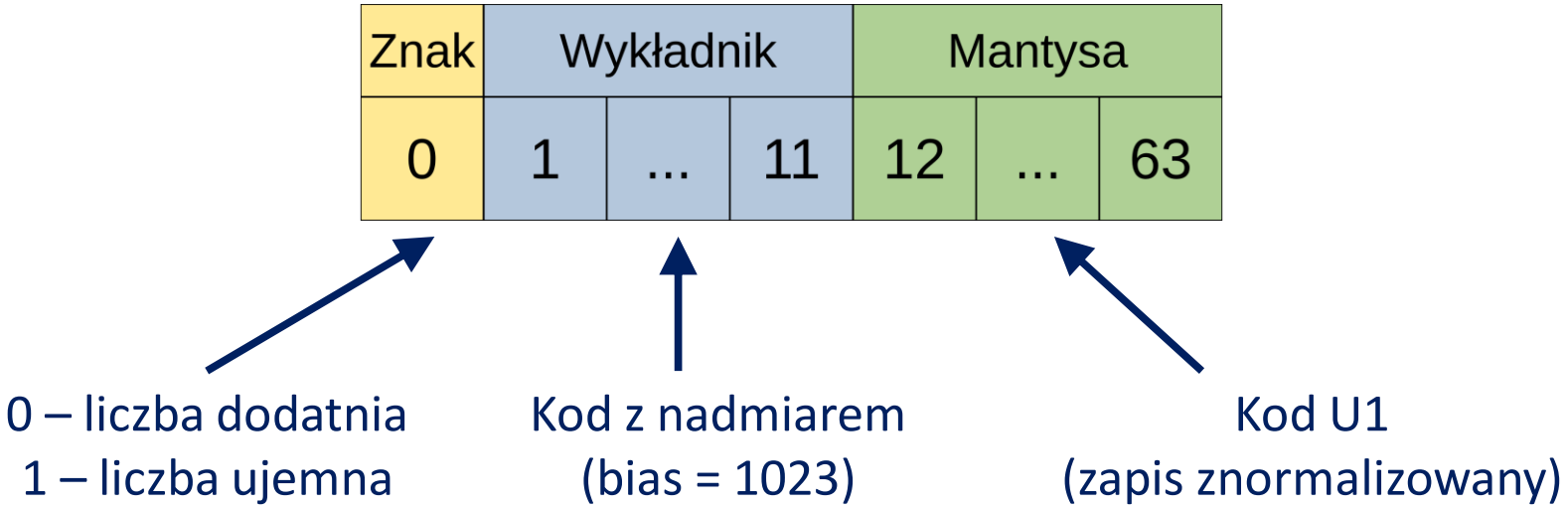
0 – liczba dodatnia
1 – liczba ujemna

Kod z nadmiarem
(bias = 127)

Kod U1
(zapis znormalizowany)

$$L = (-1)^z \cdot 2^w \cdot m$$

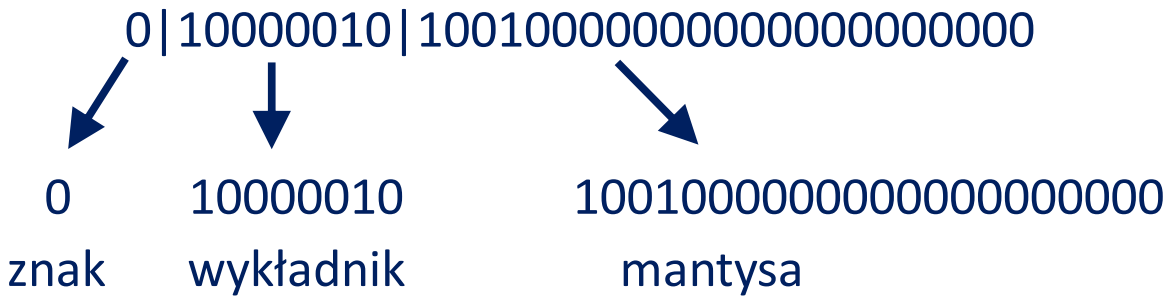
IEEE754 – double precision



$$L = (-1)^z \cdot 2^w \cdot m$$

IEEE754 - przykład

- Znaleźć wartość dziesiętną liczby zmiennoprzecinkowej 01000001010010000000000000000000 zapisanej w standardzie IEEE754:



$$z = 0$$

$$w = 10000010 = 128 + 2 - 127(BIAS) = 3$$

$$m = 01{,}10010000000000000000000_{U1} = 2^0 + 2^{-1} + 2^{-4} = 1 + 0.5 + 0.0625 = 1.5625$$

$$L = (-1)^z \cdot 2^w \cdot m = (-1)^0 \cdot 2^3 \cdot 1.5625 = 8 \cdot 1.5625 = 12{,}5$$

IEEE754 – zakres liczb

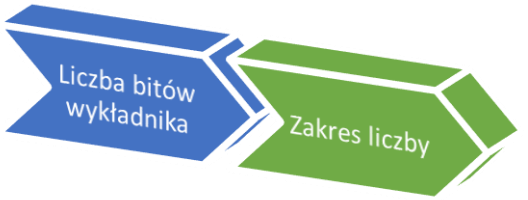
Kiedy zakodowana liczba będzie największa?

NaN

0|11111111|11111111111111111111111

0|11111110|11111111111111111111111

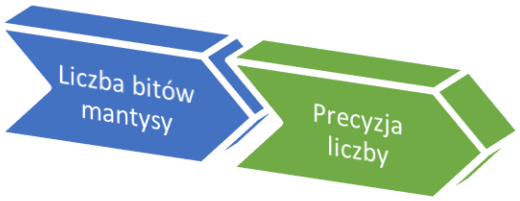
0	11111110	11111111111111111111111
znak	wykładnik	mantysa



$$L = (-1)^z \cdot 2^w \cdot m = 2^{127} \cdot \frac{2^{24} - 1}{2^{23}} \approx 3{,}4 \cdot 10^{38}$$

- Pół precyzja $\max_{16} \approx 6.6 \cdot 10^4$
- Pojedyncza precyzja $\max_{32} \approx 3.4 \cdot 10^{38}$
- Podwójna precyzja $max_{64} \approx 1.8 \cdot 10^{308}$

IEEE754 – precyzja



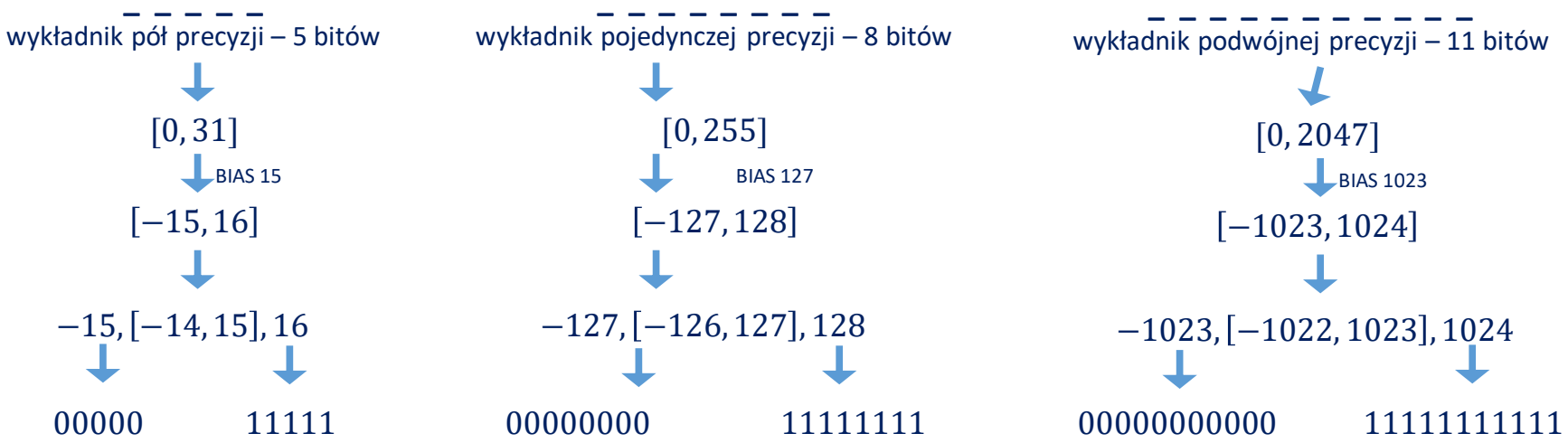
Epsilon maszynowy – największa liczba nieujemna ϵ, która spełnia warunek $1 + \epsilon = 1$

00000000000000000000001
mantysa

- Pół precyzja – około 4 cyfr znaczących
  $2^{-10} \approx 9.8 \cdot 10^{-4}$
- Pojedyncza precyzja – około 7 cyfr znaczących
  $2^{-23} \approx 1.2 \cdot 10^{-7}$
- Podwójna precyzja – około 16 cyfr znaczących
  $2^{-52} \approx 2.2 \cdot 10^{-16}$

IEEE754 – wyjątki

- Wykładniki składające się z samych zer lub jedynek są zarezerwowane dla szczególnych przypadków.



IEEE754 – 0 i ∞

- W przypadku gdy wszystkie bity wykładnika i mantysy są równe 0, liczba jest interpretowana jako 0.

0	00000000	00000000000000000000000	⟶	**+ 0**
1	00000000	00000000000000000000000	⟶	**- 0**
znak	wykładnik	mantysa		

- W przypadku gdy wszystkie bity wykładnika są równe 1 i wszystkie bity mantysy są równe 0, liczba jest interpretowana jako nieskończoność.

0	11111111	00000000000000000000000	⟶	**+ ∞**
1	11111111	00000000000000000000000	⟶	**- ∞**
znak	wykładnik	mantysa		

IEEE754 – wartości zdenormalizowane

- W przypadku gdy wszystkie bity wykładnika są równe 0, **mantysa nie zawiera domyślnej liczby całkowitej 1.** Umożliwia to zakodowanie najmniejszych liczb zmiennoprzecinkowych.

0	00000000	00000000000000000001
znak	wykładnik	mantysa

$$m_{16} = 0\mathbf{0},0000000001_{U1} = 2^{-10}$$

$$m_{32} = 0\mathbf{0},00000000000000000001_{U1} = 2^{-23}$$

$$m_{64} = 0\mathbf{0},0000000000000000000000000000000000000000000000000001_{U1} = 2^{-52}$$

$$min_{16} = (-1)^z \cdot 2^w \cdot m_{16} = (-1)^0 \cdot 2^{-14} \cdot 2^{-10} = -2^{-24} \approx 6.0 \cdot 10^{-8}$$

$$min_{32} = (-1)^z \cdot 2^w \cdot m_{32} = (-1)^0 \cdot 2^{-126} \cdot 2^{-23} = -2^{-149} \approx 1.4 \cdot 10^{-45}$$

$$min_{64} = (-1)^z \cdot 2^w \cdot m_{64} = (-1)^0 \cdot 2^{-1022} \cdot 2^{-52} = -2^{-1074} \approx 4.9 \cdot 10^{-324}$$

IEEE754 – Not a Number

- W przypadku gdy wszystkie bity wykładnika są równe 0
  a mantysa zawiera przynajmniej jeden niezerowy bit, wartość jest interpretowana jako nie-liczba (Not a Number, Nan).

0/1	11111111	XXXXXXXXXXXXXXXXXXXXXXX	
znak	wykładnik	mantysa	
0/1	11111111	1XXXXXXXXXXXXXXXXXXXXXX →	**QNan**
0/1	11111111	0XXXXXXXXXXXXXXXXXXXXXX →	**SNan**

Ciche nie-liczby

Nie wywołują żadnych wyjątków. Działanie programu nie jest zatrzymywane.

np. pierwiastek kwadratowy z liczby ujemnej

Sygnalizujące nie-liczby

Mogą zgłaszać wyjątek.

Używane np. do niezainicjalizowanych danych w przypadku ich użycia

IEEE754 – podsumowanie

Wartość	Znak	Wykładnik	Mantysa
± 0	0 / 1	0 … 0	0 … 0
liczby zdenormalizowane	0 / 1	0 … 0	różna od 0 … 0
$\pm \infty$	0 / 1	1 … 1	0 … 0
Snan (sygnalizujące nieliczby)	0 / 1	1 … 1	różna od 0 … 0, pierwszy bit 0
Qnan (ciche nieliczby)	0 / 1	1 … 1	różna od 0 … 0, pierwszy bit 1
liczby znormalizowane	0 / 1	różna od 0 … 0, różna od 1 … 1	dowolna

$$2 \cdot 2^{52} - 2 = 9\,007\,199\,254\,740\,990$$

sposobów na powiedzenie, że „x” nie jest liczbą...

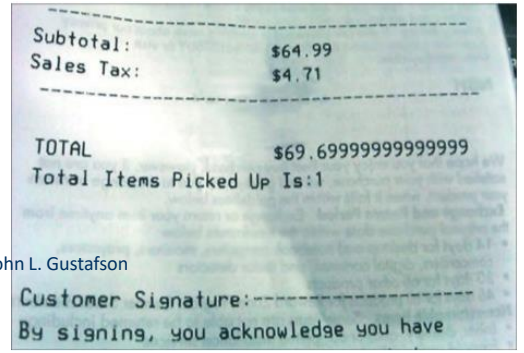
IEEE754 – główne wady

- Brak gwarancji jednakowego wyniku na różnych systemach obliczeniowych.
- Nieprzestrzeganie podstawowych własn[illegible]ń.

$$(a + b) + c = a + (b + c)$$
$$a \cdot (b + c) = a \cdot b + a \cdot c$$

```
float a, b, c, w1, w2;

a = 0.2;
b = 0.2;
c = 0.3;

w1 = (a + b) + c;
w2 = a + (b + c);

cout << w1 << endl;
cout << w2 << endl;

w1 = a*(b+c);
w2 = a*b+a*c;

cout << w1 << endl;
cout << w2 << endl;
```

```
0.7000000477
0.6999999881

0.1000000015
0.1000000089
```



źródło: John L. Gustafson

- Dodatnie i ujemne 0.
- Problem niedomiaru – gdy dokładny wynik jest różny od zera ale mniejszy niż najmniejsza znormalizowana liczba. Sytuacja łagodzona poprzez zaokrąglanie.
- W podwójnej precyzji **ponad 9 biliardów** kombinacji bitowych wykorzystanych na NaN.
- Liczba bitów wykładnika i mantysy jest stała. Brak możliwości zdecydowania co jest ważniejsze: zakres czy precyzja?

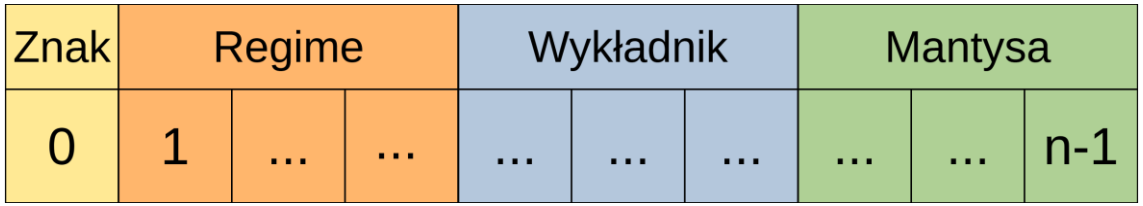
Arytmetyka zmiennopozycyjna

- Reprezentacja zmiennopozycyjna jest reprezentacją przybliżoną
  - wyniki działań arytmetycznych są również przybliżone
- Wynik może zależeć od kolejności działań
  - dodawanie i odejmowanie wielu argumentów należy wykonywać w kolejności rosnącej wartości bezwzględnej
  - jeśli |a| jest znacznie mniejsze od |b| to a + b daje w wyniku b
- Nie należy używać relacji równości
  - zamiast tego należy używać formuł takich jak
    abs(a-b) < delta

Posit – następca IEEE754?

- Idea została zaprezentowana w 2017 r. przez Johna Gustafsona
- Jest rozwinięciem wcześniejszej koncepcji Gustafsona – Unums (Universal Numbers)
- W przeciwieństwie do Unums, łatwiejszym do zrealizowania od strony sprzętowej
- Główną zaletą posit jest możliwość uzyskania większej precyzji lub zakresu przy tej samej liczbie użytych bitów
- Idea polega na prezentowaniu liczb ze zmienną dokładnością
- Liczby z małymi wykładnikami są reprezentowane dokładniej niż liczby w dużymi wykładnikami
- W porównaniu do IEEE754 występuje dodatkowa kategoria bitów: „regime”
- Liczba bitów poświęconych na wykładnik, mantysę oraz regime jest dobierana w zależności od potrzeb

Posit – sposób kodowania



- Struktura posit definiowana jest za pomocą dwóch liczb

$$posit < n, es >$$

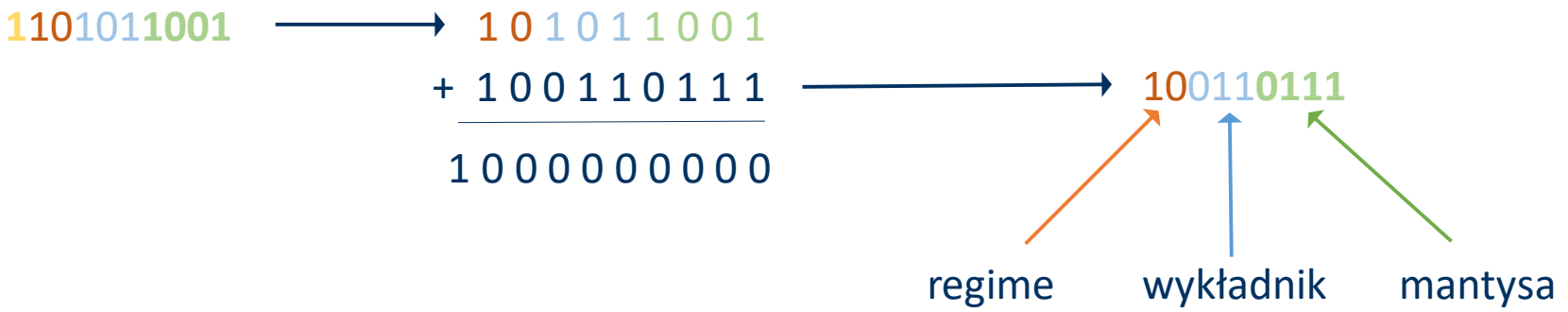
całkowita liczba bitów

maksymalna liczba bitów poświęconych wykładnikowi

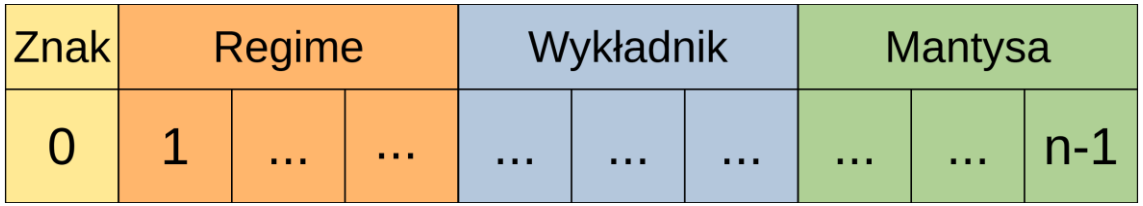
Posit – sposób kodowania

Znak	Regime			Wykładnik			Mantysa		
0	1	...	...	...	...	...	...	...	n-1

- **Znak** kodowany jest tak samo w standardzie IEEE754:
  - 0 – liczba dodatnia
  - 1 – liczba ujemna
- Jeżeli bit **znaku** jest równy 1, to przed dalszym dekodowaniem, pozostałe bity należy uzupełnić do 2.



Posit – sposób kodowania



- **Regime** składa się z samych zer lub jedynek.
- Może mieć od 1 do $n-1$ bitów.
- Skąd więc wiadomo, że obszar regime się zakończył?
  - napotkanie przeciwnego bitu
  - koniec bitów

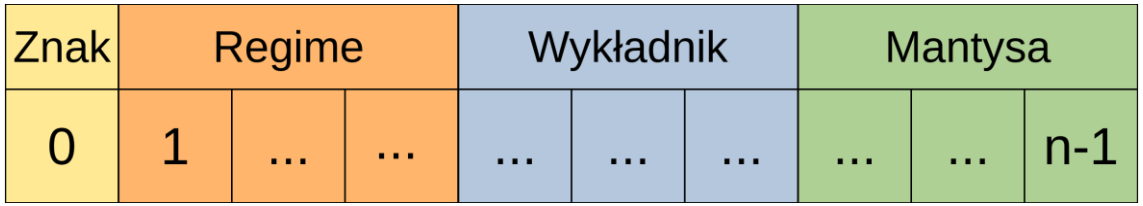
Przeciwny bit jest zakończeniem obszaru regime. Nie wchodzi w skład wykładnika

0111110
...

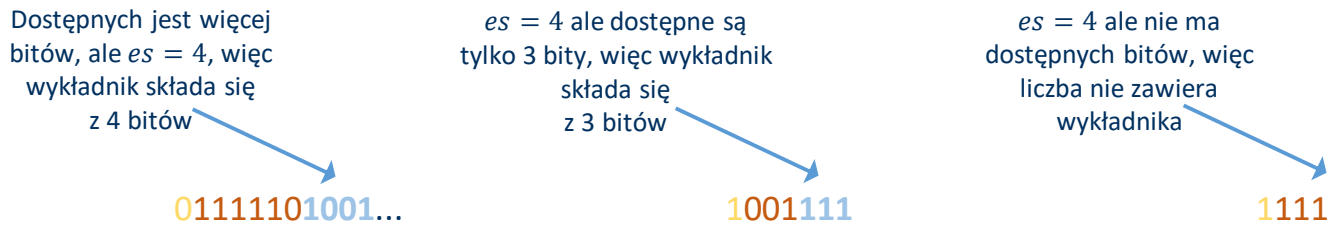
1001
...

1111

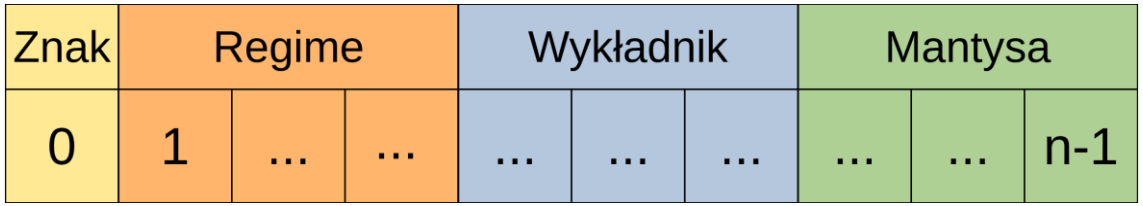
Posit – sposób kodowania



- **Wykładnik** zawiera od 0 do es bitów.
- Jest zapisany w naturalnym kodzie dwójkowym.
- Jeżeli po zakodowaniu znaku i regime, zostały dostępne bity, wchodzą one w skład wykładnika (maksymalnie es bitów)

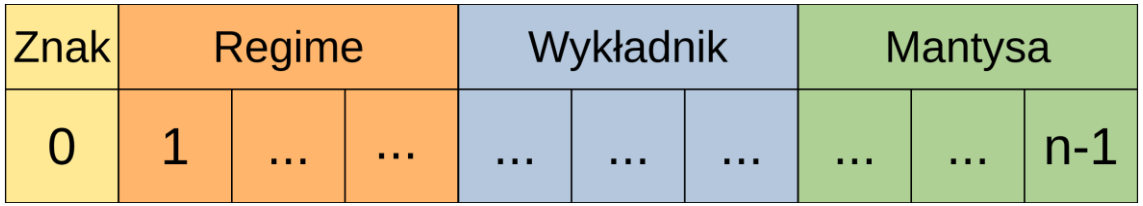


Posit – sposób kodowania



- Jeżeli po zakodowaniu znaku, regime i wykładnika, zostały dostępne bity, wchodzą one w skład **mantysy**.
- Mantysa jest znormalizowana, zawiera domyślną jedynkę.

Posit – sposób kodowania



p – liczba identycznych bitów występujących w części regime

$$k = \begin{cases} -p & \text{gdy regime składa się z } p \text{ zer} \\ p-1 & \text{gdy regime składa się z } p \text{ jedynek} \end{cases}$$

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

$$L = (-1)^z \cdot u^k \cdot 2^w \cdot m$$

Posit – przykłady

- Znaleźć wartość dziesiętną liczby zmiennoprzecinkowej 1111010110101O1 zapisanej przy użyciu $posit$<15,3>:

Znak	Regime				Wykładnik			Mantysa						
1	1	1	1	0	1	0	1	1	0	1	0	1	0	1

Bit znaku = 1, dlatego najpierw należy uzupełnić pozostałe bity do 2.

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

\+ 0 0 0 1 0 1 0 0 1 0 1 0 1 1 ⟶ 0 0 0 1 0 1 0 0 1 0 1 0 1 1

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

$$k = -p = -3$$

$$u = 2^{2^{es}} = 2^{2^3} = 256$$

$$w = 2$$

$$L = (-1)^z \cdot u^k \cdot 2^w \cdot m$$

$$L = (-1)^1 \cdot 256^{-3} \cdot 2^2 \cdot (1 + \frac{1}{4} + \frac{1}{16} + \frac{1}{64} + \frac{1}{128})$$

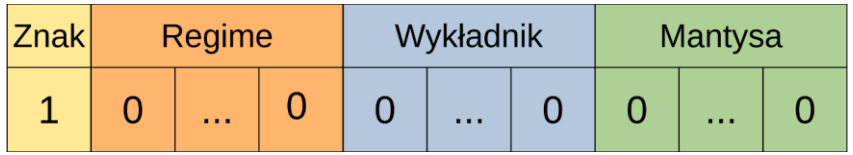
$$L \approx -3.19 \cdot 10^{-7}$$

Posit – wyjątki

- Arytmetyka posit wyróżnia **tylko** 2 wyjątki:
- **Przypadek 1:** wszystkie bity są równe 0, wartość jest interpretowana jako 0

$\rightarrow 0$

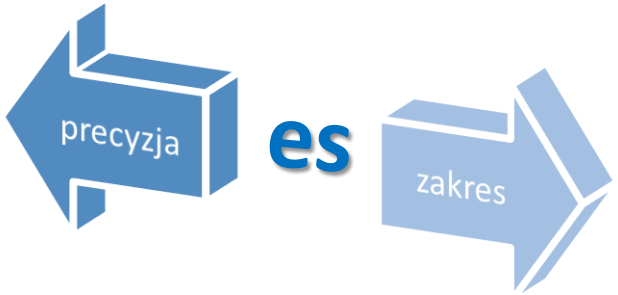
- **Przypadek 2:** bit znaku jest równy 1, a wszystkie pozostałe bity są równe 0, wartość jest interpretowana jako nieskończoność lub nie-liczba.



$\rightarrow \pm\infty, NaN$

Posit – zakres i precyzja

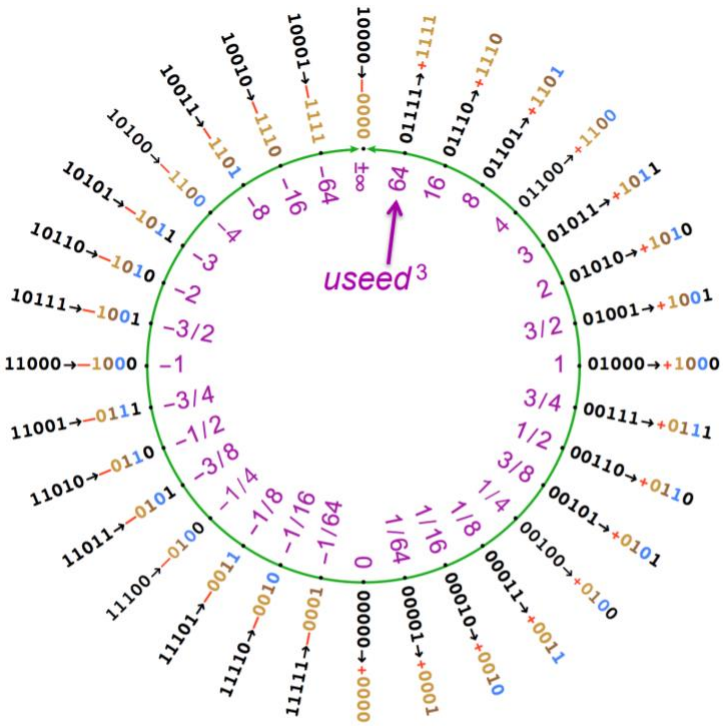
- Zakres liczby posit zależy nie tylko od liczby bitów n, ale także od wartości es czyli maksymalnej liczby bitów wykładnika
- Większe wartości es oznaczają więcej bitów dla części regime i wykładnika, a co za tym idzie większy zakres.
  Z drugiej strony, im większa wartość es, tym mniej bitów przypada na mantysę, odpowiedzialną za precyzję



- Modyfikowanie wartości es pozwala więc na dostosowanie arytmetyki do potrzeb i znalezienie kompromisu pomiędzy zakresem a dokładnością obliczeń

Posit – zakres i precyzja

- Niezależnie od ilości wykorzystanych bitów, liczby
  z zakresu $[-1,1]$ zajmują około **połowy** wszystkich dostępnych kombinacji bitowych
- Liczby bliskie 0 mają więc dużą precyzję. Jest to sytuacja pożądana, ponieważ zdecydowana większość obliczeń dotyczy małych liczb
- Tak duża precyzja nie jest potrzebna w przypadku bardzo dużych oraz bardzo małych liczb, dlatego one przedstawiane są z mniejszą precyzją



Posit – dynamiczny zakres

- Dynamiczny zakres jest miarą określającą stosunek pomiędzy największą i najmniejszą wartością jaką może przyjąć określona wielkość. Obliczany jest za pomocą wzoru: $\log_{10} \frac{max}{min}$
- Największa liczba jaką można przedstawić w arytmetyce posit to u^k, gdzie u=2^(2^es).
- Liczba k jest zdefiniowana: $k = \begin{cases} -p & \text{gdy regime składa się z } p \text{ zer} \\ p-1 & \text{gdy regime składa się z } p \text{ jedynek} \end{cases}$
- k będzie więc największe, gdy wszystkie bity po bicie znaku będą jedynkami. Wtedy regime będzie się składał z $p = n - 1$ jedynek, więc $k = p - 1 = n - 2$.

$$max = u^k = \left(2^{2^{es}}\right)^{n-2} \qquad min = u^{-k} = \left(2^{2^{es}}\right)^{2-n} = 1/max$$

$$\log_{10} \frac{max}{min} = \log_{10} \frac{max}{\frac{1}{max}} = \log_{10} \qquad max^2 = \log_{10}\left(2^{2^{es}}\right)^{2n-4} = (2n-4)\, 2^{\mathrm{es}} \log_{10} 2$$

Porównanie Posit – IEEE754

IEEE 754 : pojedyncza precyzja

$$\log_{10}\frac{max}{min} = \log_{10}\frac{3.4 \cdot 10^{38}}{1.4 \cdot 10^{45}} \approx 83$$

posit <32,3> :

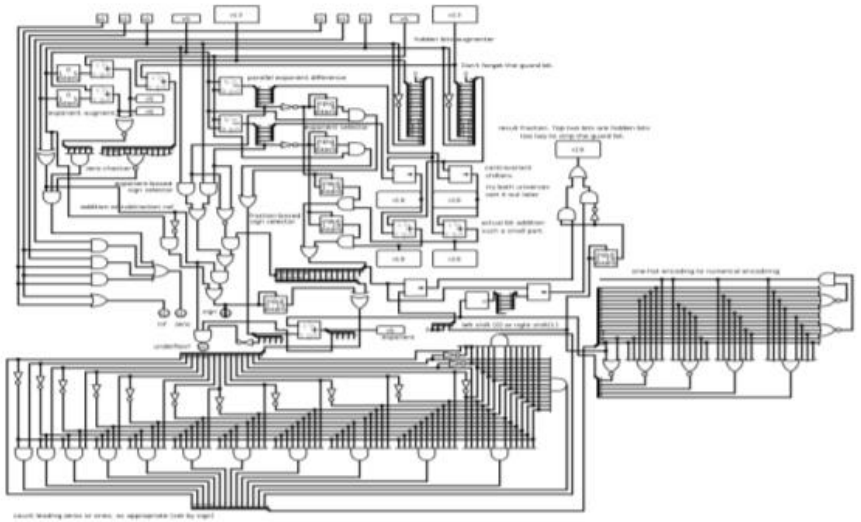
$$\log_{10}\frac{max}{min} = (2n-4)\,2^{es}\log_{10} 2 \approx 144$$

liczba bitów		Wykładnik	Mantysa	Dynamiczny zakres
16	IEEE	5	10	12
	posit	1	12	17
32	IEEE	8	23	83
	posit	3	26	144
64	IEEE	11	52	632
	posit	5	56	1194
128	IEEE	15	112	9897
	posit	8	117	19420

Posit – Podsumowanie

- W przeciwieństwie do IEEE754, posit jest system poprawnym od strony matematycznej: przestrzeganie praw wykonywania działań, jedna wartość symbolizująca 0
- Niezależnie od liczby bitów – tylko 2 wyjątki: 0 i ∞
- Dzięki zmiennej liczbie bitów poświęconych wykładnik, możliwy wybór priorytetów: zakres czy precyzja?
- Zwężana dokładność. Liczby, które występują w obliczeniach najczęściej, kodowane są z dużą precyzją w przeciwieństwie do ekstremalnie dużych liczb

Układ logiczny realizujący dodawanie w arytmetyce posit



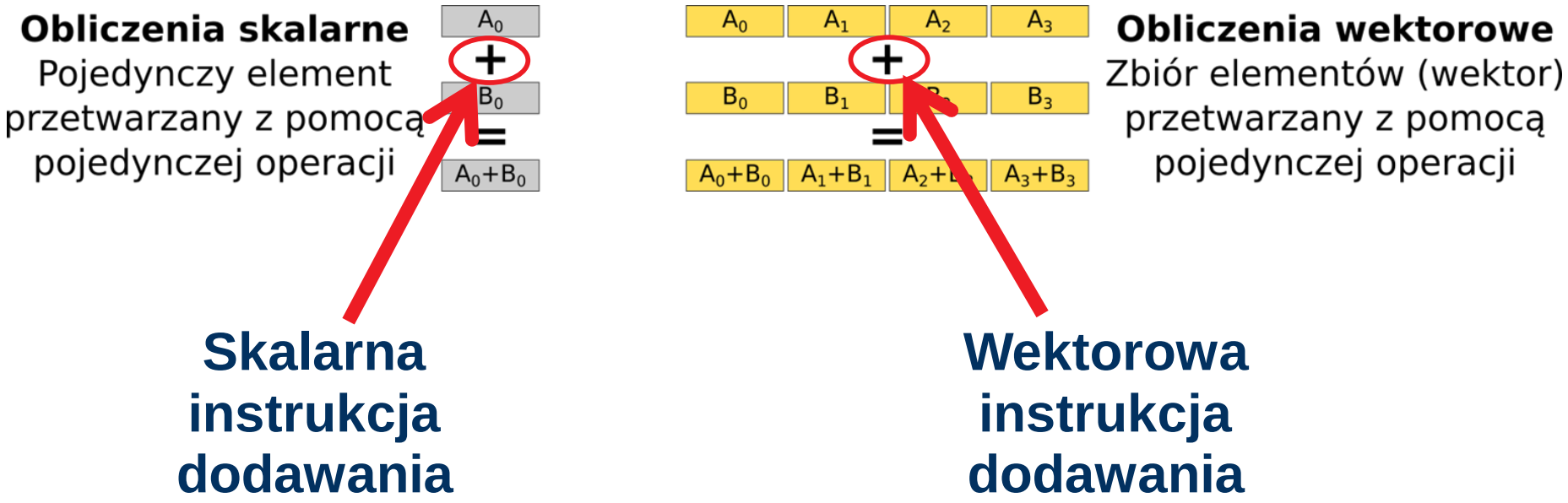
- Istnieją implementacje softwarowe w różnych językach m.in. C, C++
- Trwają prace nad budową układu scalonego korzystającego z arytmetyki posit
- Ostatnie badania pokazują, że zastosowanie arytmetyki posit w sieciach neuronowych może poprawić ich wydajność
- Mimo wielu zalet, arytmetyka posit ma jedną znaczącą wadę. Zmienna liczba bitów w wykładniku i mantysie powoduje, że dekodowanie może być wykonywane tylko sekwencyjnie. Format IEEE754 zakłada stałą liczbę bitów na wykładnik oraz mantysę, dzięki czemu możliwe jest dekodowanie w sposób równoległy

Dane wektorowe

- Współczesne architektury procesorów mogą operować na długich słowach danych, o rozmiarze większym niż 8B (64bity), przykładowo:
  - x86:
    - SSE 128 bit, AVX(2) 256 bit, AVX-512 512 bit
  - ARM:
    - NEON 128 bit
  - POWER:
    - AltiVec/VMX/VSX 128 bit, QPX 256 bit
  - SPARC:
    HPC-ACE 128 bit, HPC-ACE2 256 bit
- Wprowadzenie wektorowego formatu danych - wielu krótkich danych zapisanych w pojedynczym słowie - umożliwia lepsze wykorzystanie możliwości procesora

Dane wektorowe

- Wprowadzenie wektorowego formatu danych - **wielu krótkich danych zapisanych w pojedynczym słowie** - umożliwia lepsze wykorzystanie możliwości procesora



Formaty danych wektorowych: SSE

- Wektorowy typ danych bazujący na standardzie SSE pozwala na przechowywanie do 16 elementów w rejestrach
- Słowo danych wynosi 128 bitów i obsługuje następujące formaty:
  - 4 typy stało pozycyjne 32-bitowe
  - 2 typy stało pozycyjne 64-bitowe
  - 4 typy zmiennopozycyjne 32-bitowe IEEE single
  - 2 typy zmiennopozycyjne 64-bitowe IEEE double

Formaty danych wektorowych: AVX

- Wektorowy typ danych bazujący na standardzie AVX pozwala na przechowywanie do 32 elementów w rejestrach
- Słowo danych wynosi 256 bitów i obsługuje następujące formaty:
  - 8 typów stało pozycyjne 32-bitowe
  - 4 typy stało pozycyjne 64-bitowe
  - 8 typów zmiennopozycyjne 32-bitowe IEEE single
  - 4 typy zmiennopozycyjne 64-bitowe IEEE double

Formaty danych wektorowych: AVX-512

- Wektorowy typ danych bazujący na standardzie AVX pozwala na przechowywanie do 64 elementów w rejestrach
- Słowo danych wynosi 512 bitów i obsługuje następujące formaty:
  - 16 typów stało pozycyjne 32-bitowe
  - 8 typów stało pozycyjne 64-bitowe
  - 16 typów zmiennopozycyjne 32-bitowe IEEE single
  - 8 typów zmiennopozycyjne 64-bitowe IEEE double

Przedrostki w informatyce

- Przedrostki kilo, mega, giga, tera oznaczają odpowiednio wielokrotności 10^3 , 10^6 , 10^9 , 10^{12} jednostek podstawowych
- W informatyce mogą one też oznaczać odpowiednio 2^{10} , 2^{20} , 2^{30}, 2^{40}
- Zdarza się też mieszanie potęg 2 i 10

Przedrostki w informatyce

- Obowiązują pewne zwyczajowe reguły:
  - Wielkość pamięci określa się, stosując potęgi 2
  - Przepływność/przepustowość określa się, stosując na ogół potęgi 10: w modemie 56 kb/s to 56000 b/s, w Ethernecie 100Mb/s to 10^8 b/s
  - Pojemności dysków twardych określa się, stosując potęgi 10
  - Przy określaniu pojemności innych pamięci masowych panuje „bałagan”:
    - dla dyskietki 1,44 MB mega oznacza $10^3 \cdot 2^{10}$,
    - dla płyty CD mega jest bliższe 2^{20},
    - dla płyty DVD giga jest bliższe 10^9

Przedrostki w informatyce

- Poprzez dodanie po znaku mnożnika litery ***i***, i zastąpienie drugiej sylaby nazwy mnożnika przez bi (od binarny): KiB, czyli kibibajt (1024 bajty), w odróżnieniu od kB (1000 bajtów)

IEC		podstawa						SI	
nazwa	symbol	2	16		różnica	10		nazwa	symbol
kibi	Ki	2^{10}	$16^{2.5}$	400_{16}	2,40%	1 024	$> 10^{3}$	kilo	k
mebi	Mi	2^{20}	16^{5}	$10\ 0000_{16}$	4,86%	1 048 576	$> 10^{6}$	mega	M
gibi	Gi	2^{30}	$16^{7.5}$	$4000\ 0000_{16}$	7,37%	1 073 741 824	$> 10^{9}$	giga	G
tebi	Ti	2^{40}	16^{10}	$100\ 0000\ 0000_{16}$	9,95%	1 099 511 627 776	$> 10^{12}$	tera	T
pebi	Pi	2^{50}	$16^{12.5}$	$4\ 0000\ 0000\ 0000_{16}$	12,59%	1 125 899 906 842 624	$> 10^{15}$	peta	P
eksbi	Ei	2^{60}	16^{15}	$1000\ 0000\ 0000\ 0000_{16}$	15,29%	1 152 921 504 606 846 976	$> 10^{18}$	eksa	E
zebi	Zi	2^{70}	$16^{17.5}$	$40\ 0000\ 0000\ 0000\ 0000_{16}$	18,06%	1 180 591 620 717 411 303 424	$> 10^{21}$	zetta	Z
jobi	Yi	2^{80}	16^{20}	$1\ 0000\ 0000\ 0000\ 0000\ 0000_{16}$	20,89%	1 208 925 819 614 629 174 706 176	$> 10^{24}$	jotta	Y

Dziękuję za uwagę

Kontakt:
dr hab. inż. Łukasz Szustak, prof. PCz
lszustak@icis.pcz.pl
Katedra Informatyki
Wydział Inżynierii Mechanicznej i Informatyki
Politechnika Częstochowska