AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA W KRAKOWIE
AGH UNIVERSITY OF SCIENCE
AND TECHNOLOGY

Analiza użycia architektury SIMD oraz biblioteki Vector API w celu implementacji algorytmów kompresji danych

Autor: Maciej Pabin
Opiekun pracy: dr hab. inż. Piotr Chołda

Cel pracy

Analiza wydajności działania algorytmów kompresji liczb całkowitych BinaryPacking oraz FastPFOR, zaimplementowanych w oparciu o wykorzystanie potencjału architektury SIMD za pomocą biblioteki Vector API.

Kompresja danych

Dzięki kompresji danych możliwym staje się ich szybsze przesyłanie przez sieć przy jednoczesnym zaoszczędzeniu miejsca potrzebnego do ich przechowywania. Bardzo ważnym jest aby proces kompresji oraz dekompresji danych przebiegał jak najszybciej. W pracy skupiono się na technikach kompresji liczb całkowitych: BinaryPacking oraz FastPFOR.

Kompresja danych

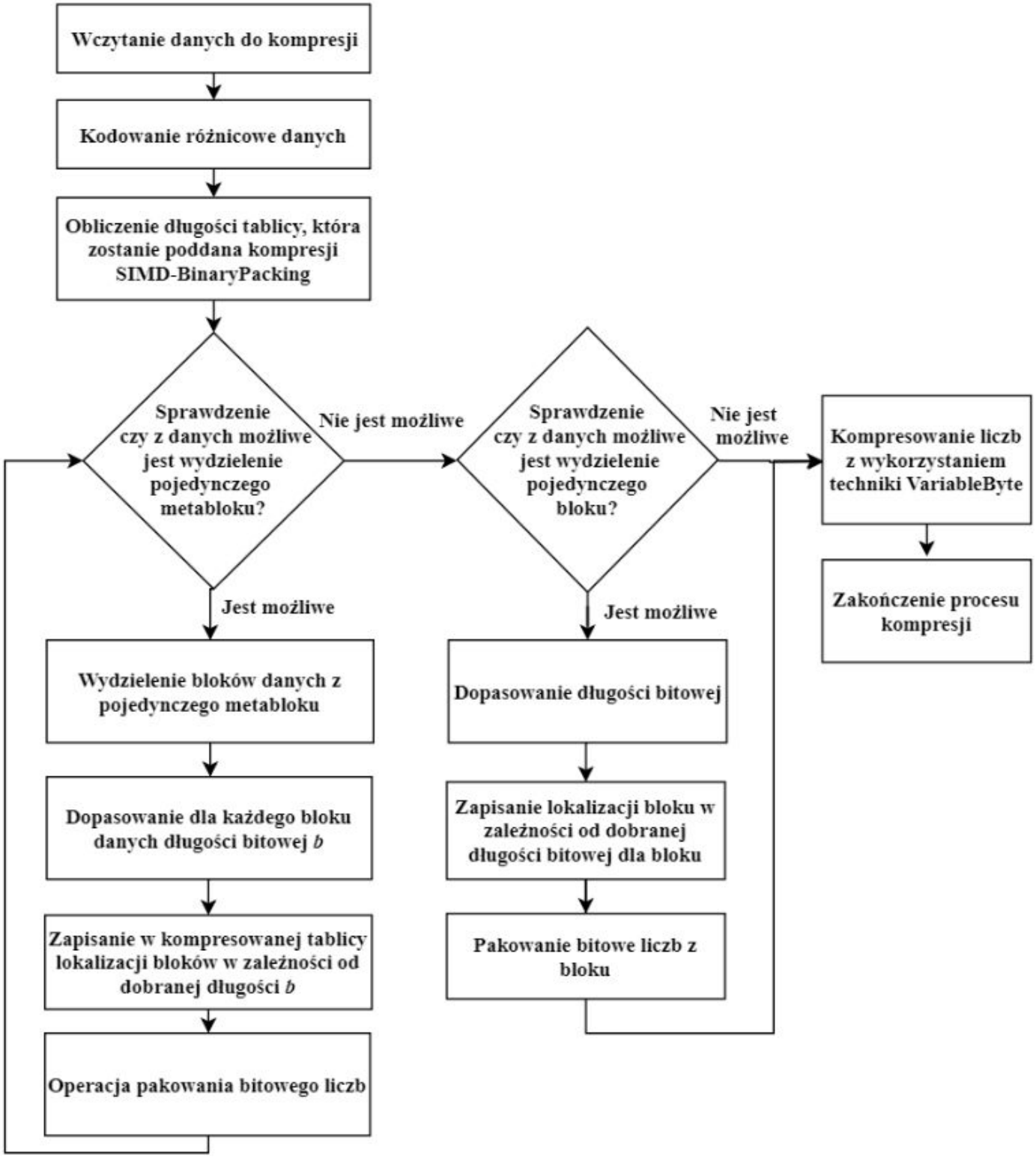
- Kompresja liczb całkowitych jest bezstratną kompresją.

  Dane sprzed procesu kompresji muszą być takie same jak uzyskane dane w wyniku dekompresji
- Kompresję charakteryzować mogą następujące czynniki:
  - stopień kompresji
  - czas kompresji
  - czas dekompresji

Technika kompresji BinaryPacking

BinaryPacking to niezwykle szybka technika kompresji i dekompresji danych. Jej działanie polega na formowaniu z danych, które mają zostać poddane procesowi kompresji, odpowiednich struktur danych, nazywanych meta-blokami oraz blokami.

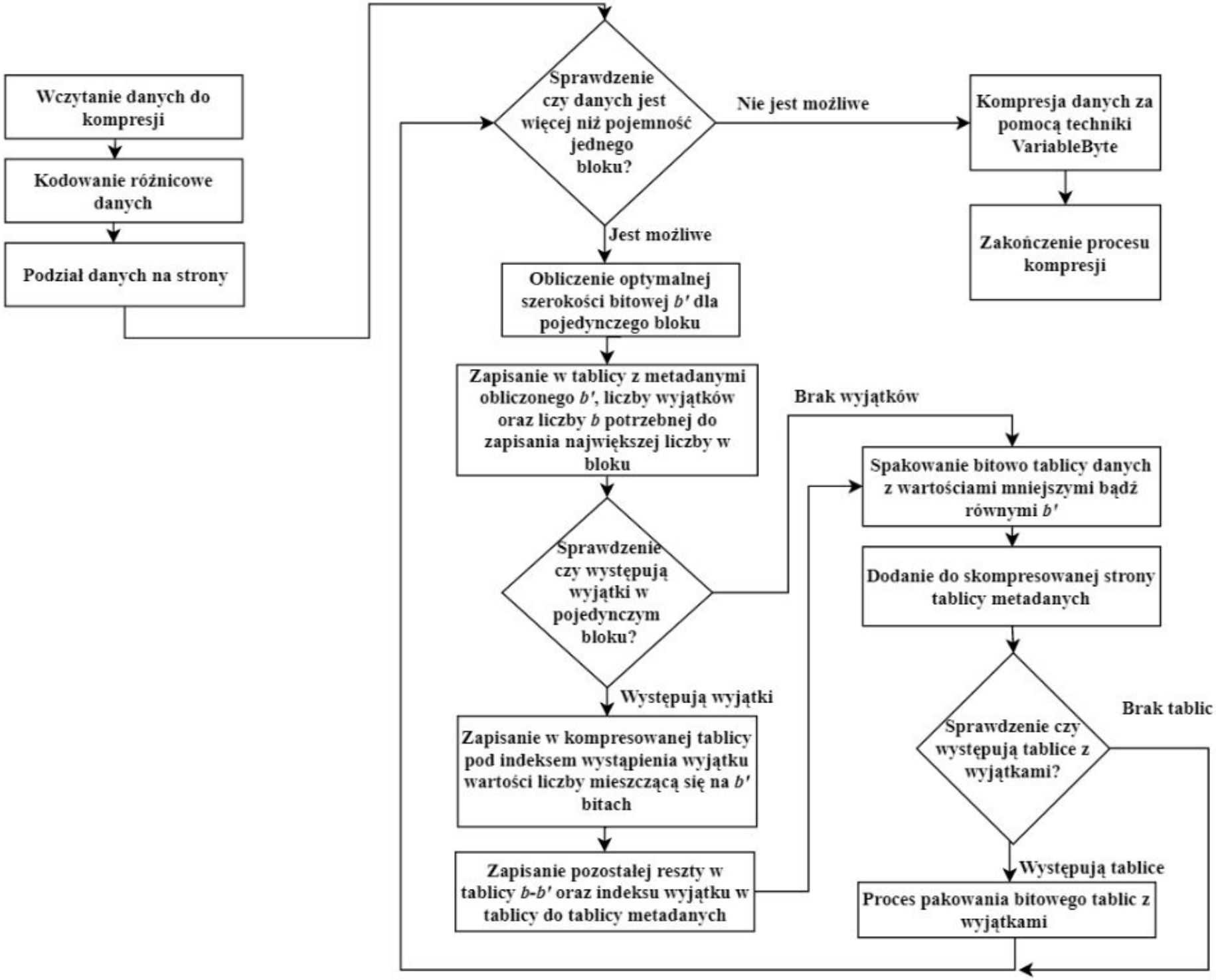
Dane zawarte w bloku danych są pakowane bitowo z długością bitową potrzebną do zapisania największej liczby występującej w bloku. Rozmiar bloku zależy od rozmiaru rejestrów wspieranych przez procesor maszyny, na której uruchomiono kompresję liczb.



Technika kompresji FastPFOR

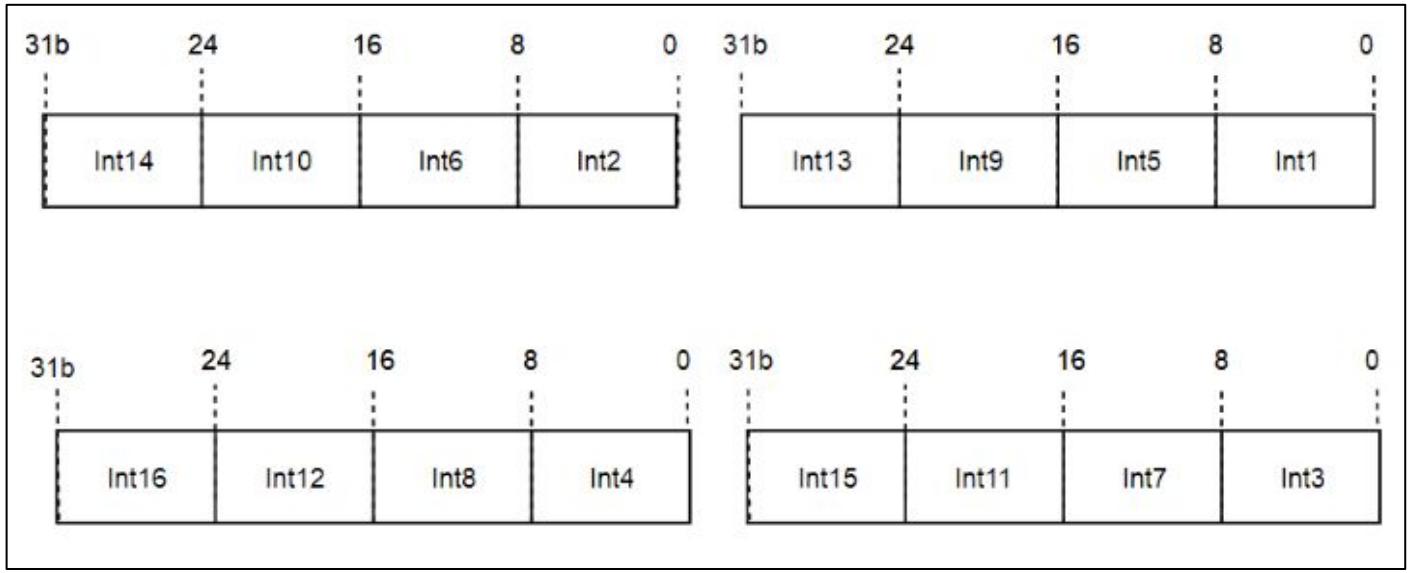
FastPFOR to technika kompresji, która dzięki przeprowadzaniu skomplikowanego procesu obliczania ilości bitów potrzebnych do przechowania danych, pozwala na osiągnięcie wysokiego stopnia kompresji danych.

Dane, podobnie jak w technice BinaryPacking, poddawane są procesowi pakowania bitowego liczb zgodnie z wybraną długością bitową wystarczającą na zapisanie wszystkich liczb w bloku danych.



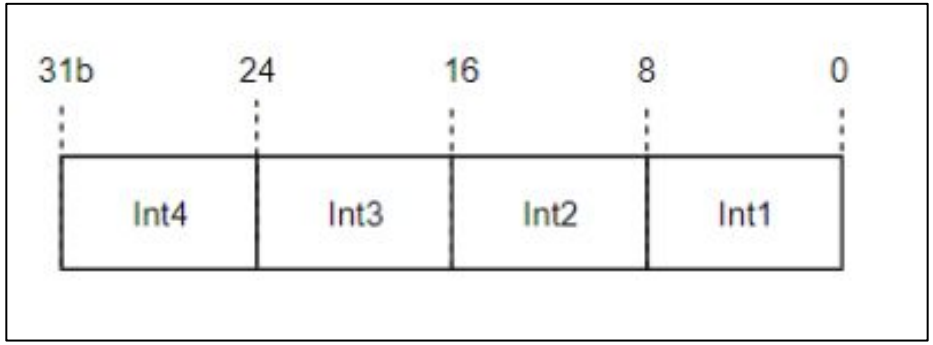
Pakowanie i rozpakowywanie bitowe liczb

- **Wektorowy** proces pakowania i rozpakowywania bitowego liczb stosowany jest gdy ilość liczb w tablicy jest większa lub równa rozmiarowi pojedynczego bloku danych.
- Liczby w postaci binarnej dla wektorowego procesu pakowania bitowego są zapisywane w pamięci w postaci **układu wertykalnego**.



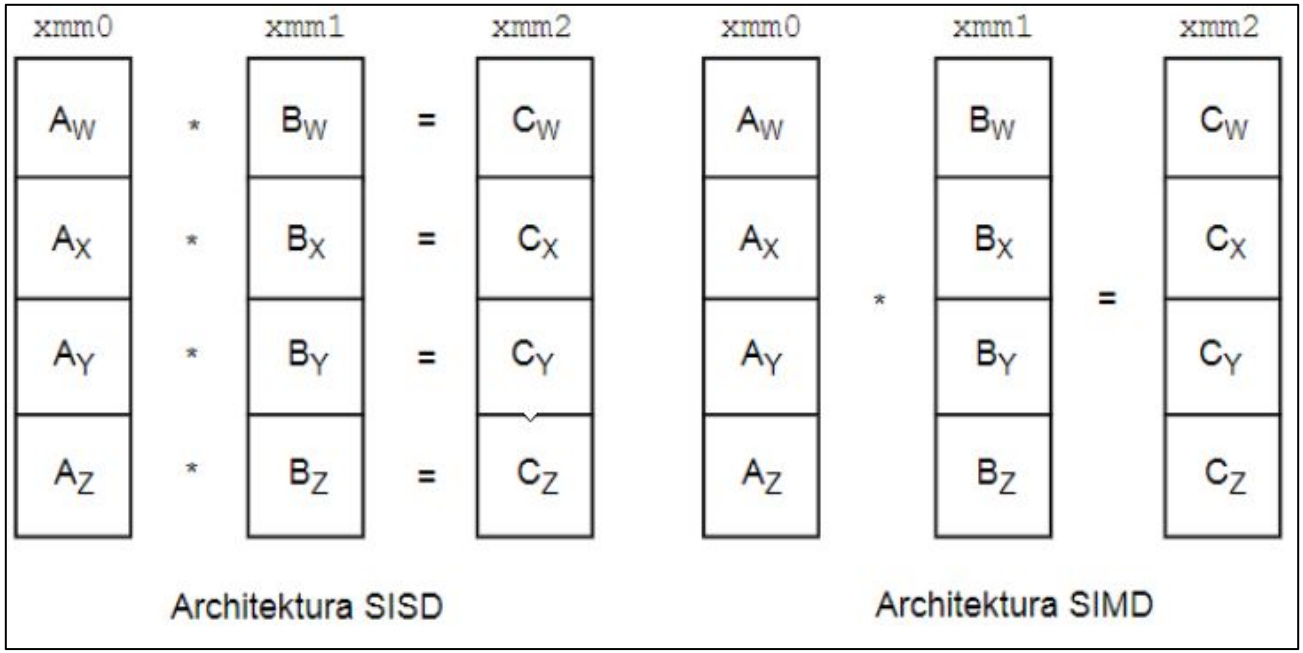
Pakowanie i rozpakowywanie bitowe liczb

- **Skalarny** proces pakowania i rozpakowywania bitowego liczb stosowany jest gdy ilość liczb w tablicy jest mniejsza od rozmiaru pojedynczego bloku danych.
- Liczby w postaci binarnej dla skalarnego procesu pakowania bitowego są zapisywane w pamięci w postaci **układu horyzontalnego.**



Architektura SIMD

SIMD (ang. *Single Instruction Multiple Data*) pozwala na przetwarzanie wielu strumieni danych równocześnie za pomocą pojedynczej instrukcji. To jedna z metod zrównoleglania przetwarzanych danych. Pozwala na znaczące skrócenie czasu wykonywania obliczeń.



Architektura SIMD

- Ilość operacji jaka może zostać zastąpiona dzięki użyciu architektury SIMD zależy od **zestawów instrukcji** wspieranych przez procesor
- Obecnie zestawy instrukcji **SSE**, **AVX/AVX2**, **AVX512** dostarczają 128-,256, 512-bitowe rejestry. Przykładowo, dzięki zbiorowi instrukcji SSE możliwym jest zastąpienie 4 operacji mnożenia MUL 32-bitowych liczb całkowitych typu int, za pomocą pojedynczej operacji MULPS

Zastosowanie SIMD w JVM

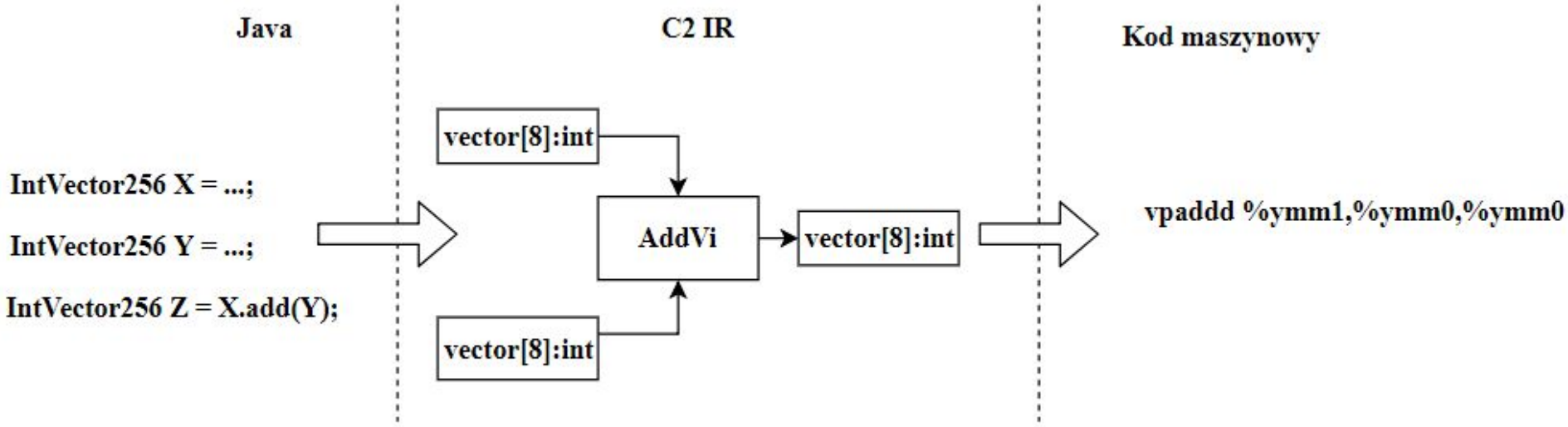
- **Kompilator JIT C2** w czasie procesu kompilacji dokonuje dogłębnej analizy kodu bajtowego w celu, dzięki czemu zostanie on w możliwych miejscach zoptymalizowany. W skutek tego wygenerowany kod maszynowy może zostać wykonany przez odpowiednie wektorowe instrukcje procesora co pozwala na znaczne przyspieszenie działania skompilowanego kodu.
- Brak możliwości tworzenia implementacji w języku Java mogącej bezpośrednio korzystać z architektury SIMD.

Biblioteka Vector API

- W ramach OpenJDK powstaje **projekt Panama** dostarczający bibliotekę **Vector API**.
- Biblioteka Vector API zapewnia zbiór klas i metod, pozwalających na wykonywanie implementacji w języku Java, która zostanie skompilowana do kodu maszynowego, mogącego zostać wykonanym przez wektorowe instrukcje procesora

Biblioteka Vector API

Za pomocą wykonanych w ramach projektu Panama odpowiednich **funkcji intrinsics**, kompilator C2 w czasie kompilacji jest w stanie użyć odpowiedniej operacji w zależności od typu wektora oraz jego rozmiaru.



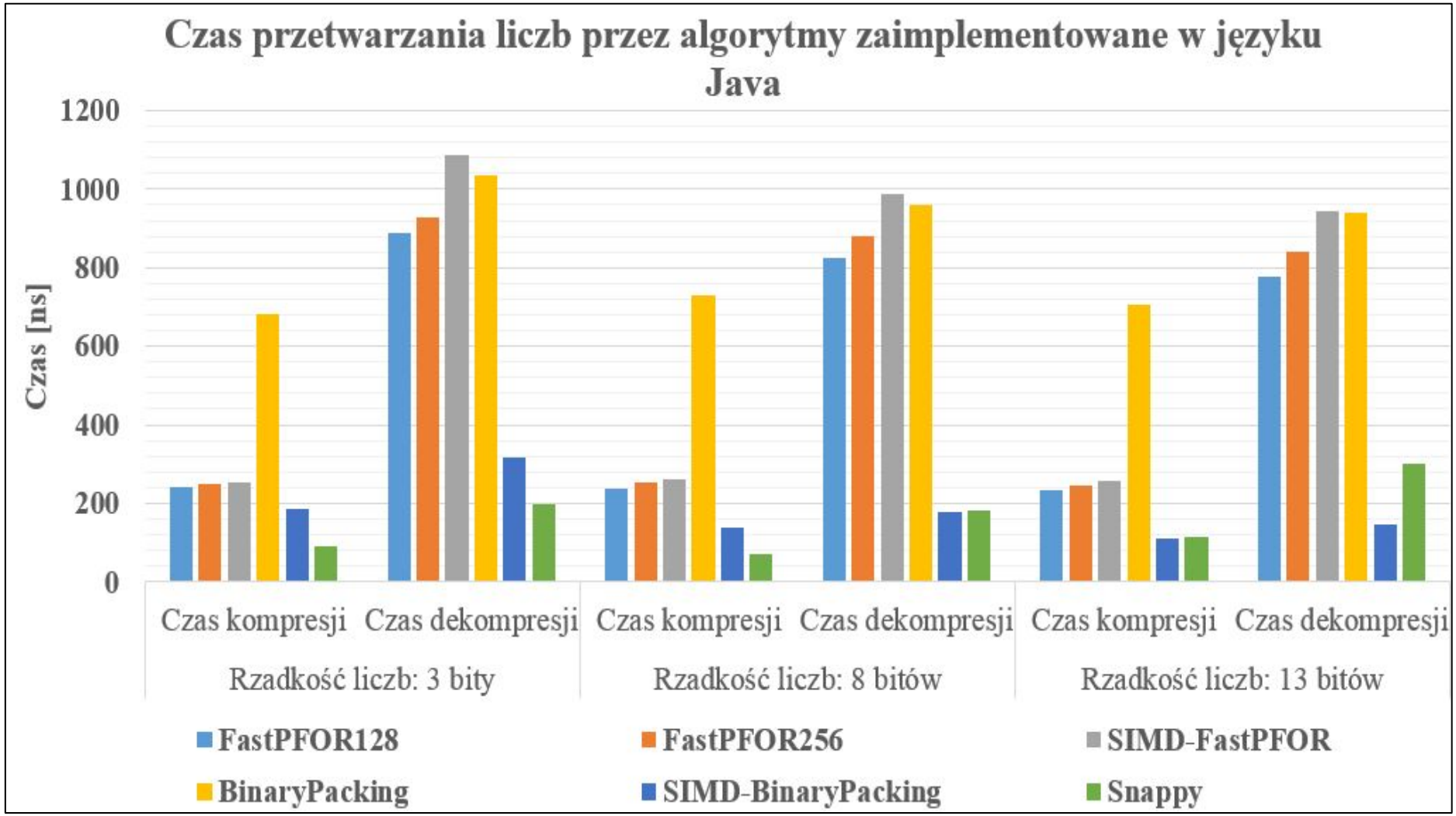
Przygotowanie do badań

- Wykonano implantację technik kompresji **SIMD-BinaryPacking** oraz **SIMD-FastPFOR** w oparciu o istniejące już implementacje technik BinaryPacking oraz FastPFOR.
- Algorytm pakowania i rozpakowywania bitowego liczb zaimplementowano w oparciu o bibliotekę Vector API.

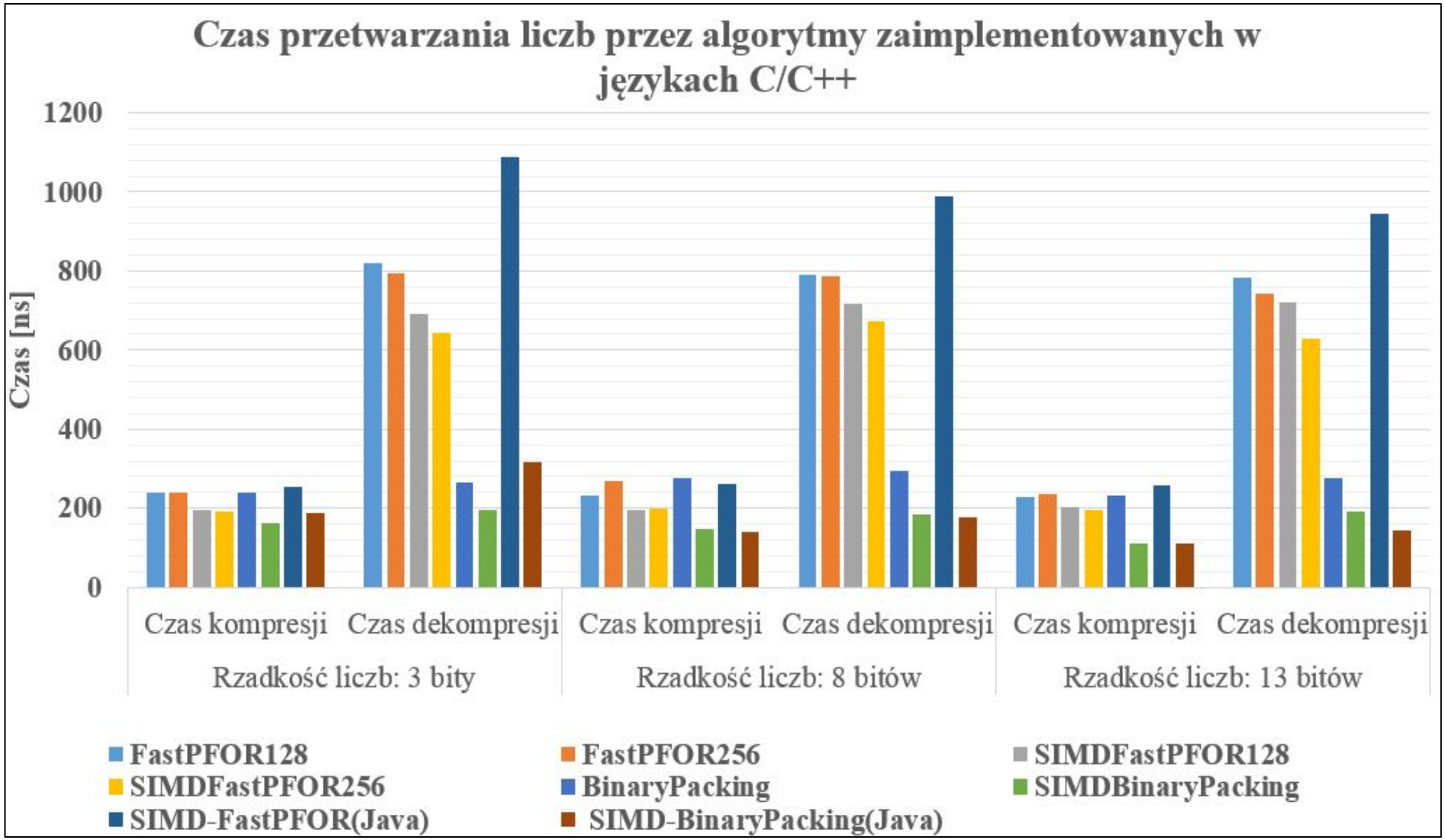
Przygotowanie do badań

- Uruchomiono wirtualną maszynę w chmurze obliczeniowej AWS, zawierającą zaimplementowane techniki SIMD-BinaryPacking oraz SIMD-FastPFOR, a także inne podobne techniki kompresji liczb, ogólnodostępnych, napisanych w językach Java, C i C++.
- Przygotowano generator liczb pseudolosowych, dostarczający tablice liczb, których kolejne elementy były generowane zgodnie z parametrem rzadkości liczb wynoszącym odpowiednio 3, 8 oraz 13 bitów.

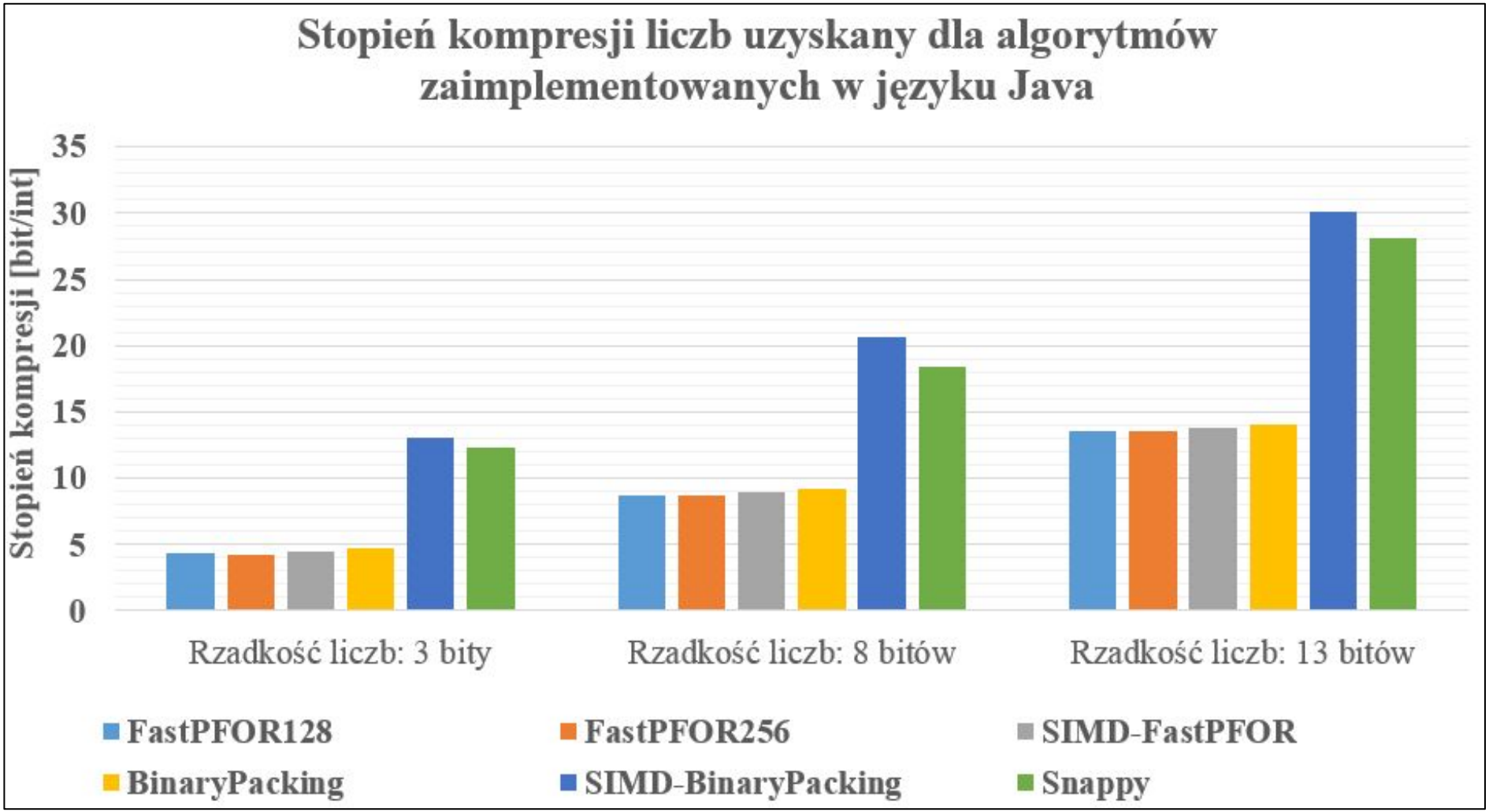
Otrzymane wyniki



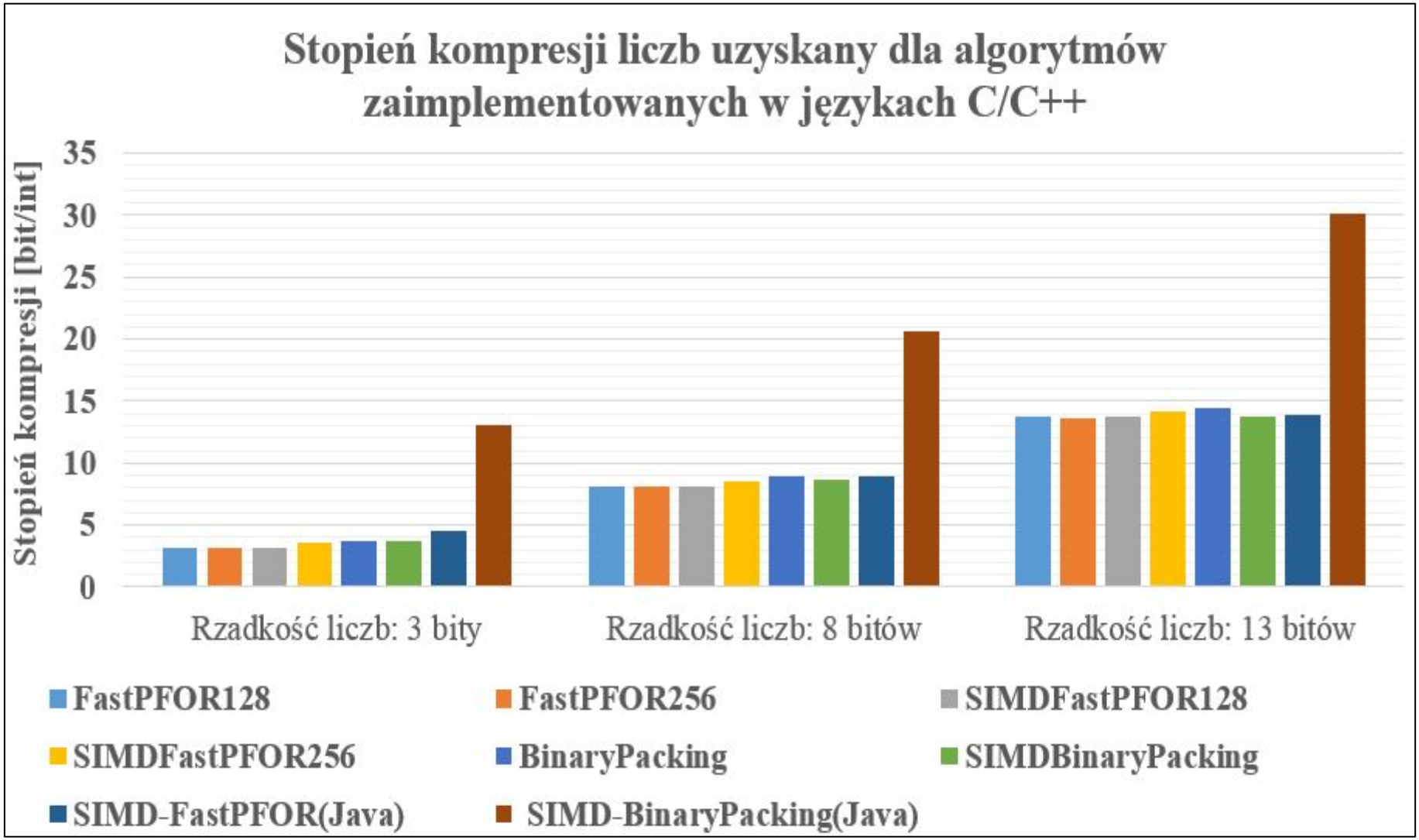
Otrzymane wyniki



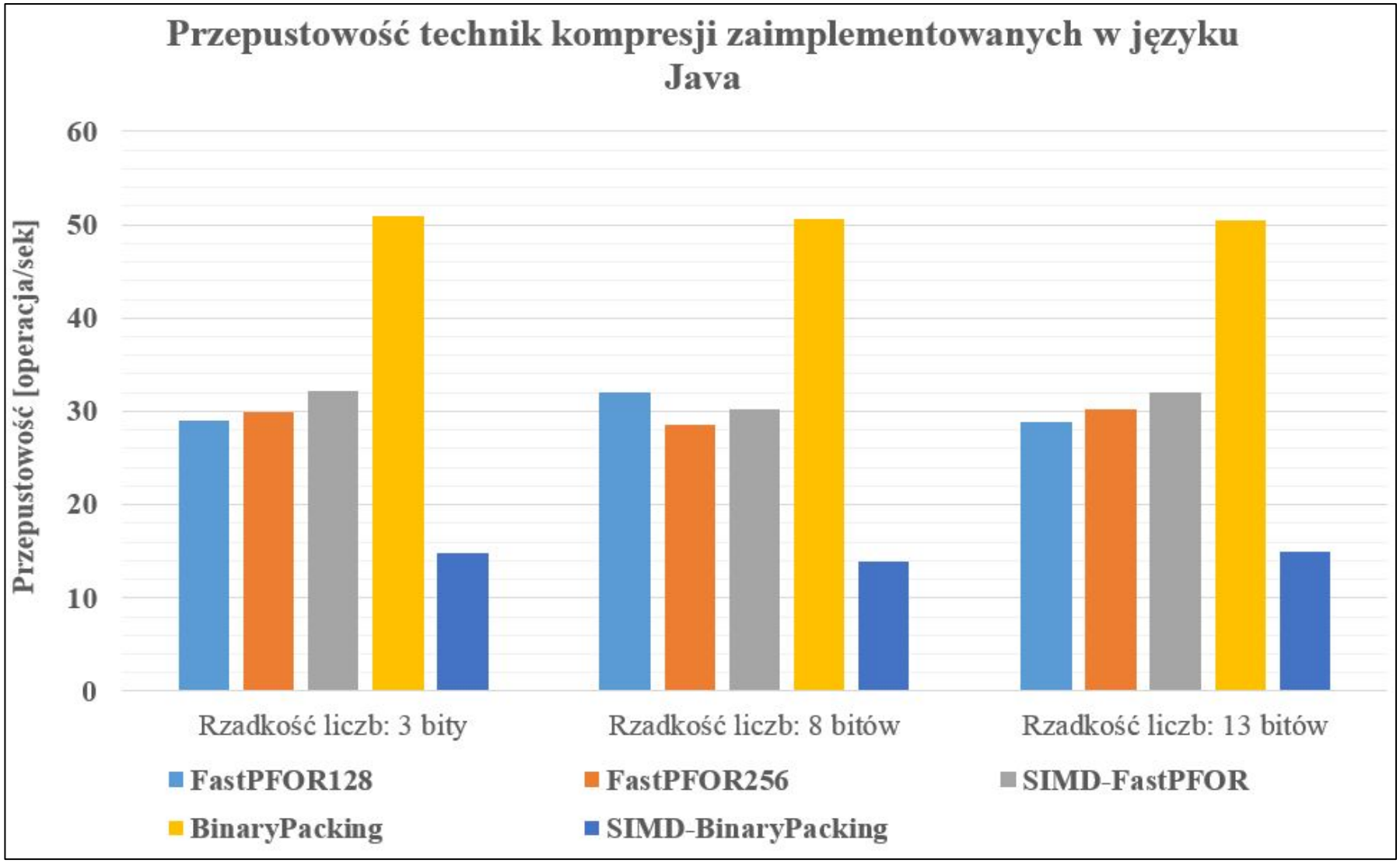
Otrzymane wyniki



Otrzymane wyniki



Otrzymane wyniki



Wnioski

- Warto używać operacji zrównoleglania przetwarzania danych.
- Implementację należy tworzyć w generyczny sposób, tak aby była niezależna od architektury procesora
- Biblioteka Vector API pozwala na osiąganie konkurencyjnych wyników implementacji wykonanych w języku Java w porównaniu z językami C/C++ oraz z biblioteką Snappy