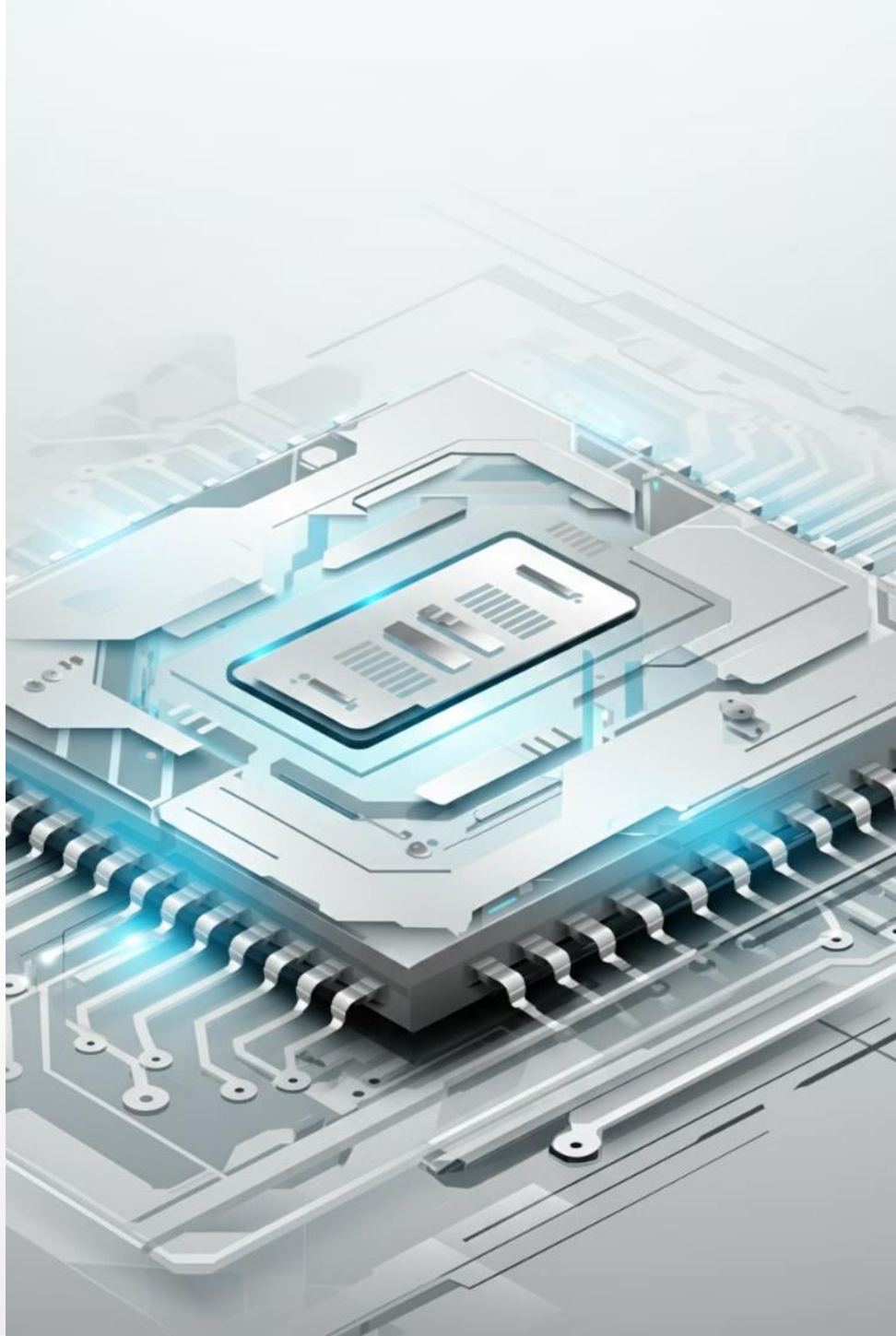
Kodowanie liczb

dr inż. Marta Chodyka



Wprowadzenie do kodów liczbowych

1 Reprezentacja binarna

Komputery operują na systemie binarnym. Każda liczba jest zapisywana jako ciąg zer i jedynek

2 Kody liczbowe

Istnieją różne metody kodowania liczb
Trzy podstawowe: ZM, U1 i U2

3 Praktyczne zastosowanie

Zrozumienie kodów liczbowych jest kluczowe dla efektywnego programowania i debugowania

Kod ZM (Znak-Moduł)

Struktura

Pierwszy bit oznacza znak
(0 dla +, 1 dla -)
Pozostałe bity to wartość bezwzględna

Zalety

Prosta interpretacja.
Łatwe odczytanie znaku i wartości liczby

Wady

Dwie reprezentacje zera ("+0" i "-0")
Utrudnione operacje arytmetyczne

Kod U1 (Uzupełnienie do 1)

1

Liczby dodatnie

Reprezentowane jak w kodzie ZM. Pierwszy bit to 0, reszta to wartość liczby

2

Liczby ujemne

Inwersja bitów liczby dodatniej. Zamiana każdego 0 na 1 i odwrotnie

3

Zero

Ma dwie reprezentacje, podobnie jak w kodzie ZM

Kod U2 (Uzupełnienie do 2)

Liczby dodatnie

Identyczne jak w U1

Pierwszy bit to 0, pozostałe to wartość liczby

Liczby ujemne

Inwersja bitów liczby dodatniej i dodanie 1 do wyniku

Jedna reprezentacja zera

U2 rozwiązuje problem podwójnego zera

Ma tylko jedną reprezentację: 00...0

Efektywność

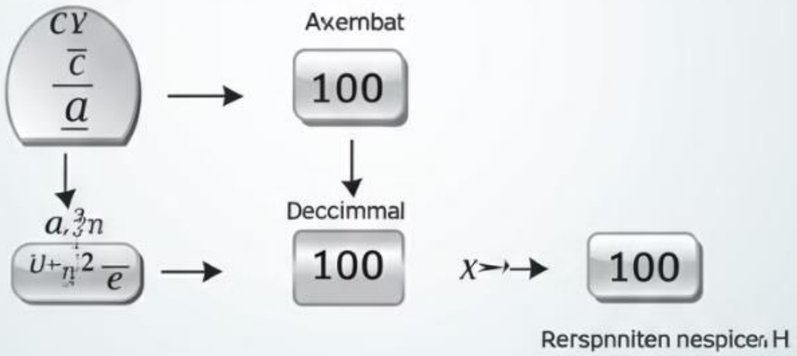
Upraszcza operacje arytmetyczne

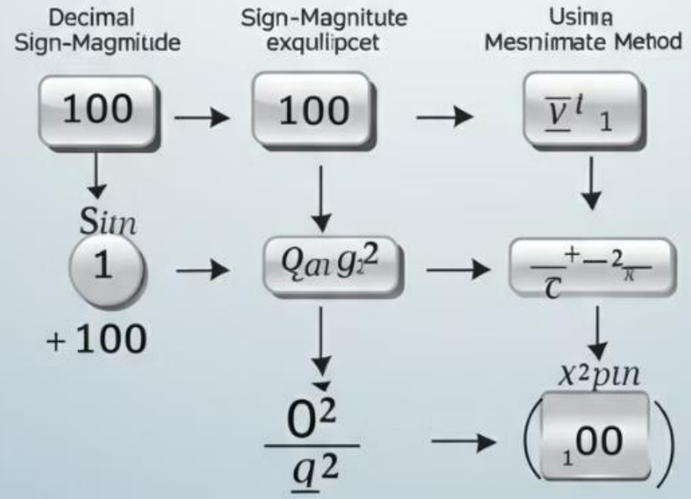
Powszechnie stosowany w współczesnych komputerach

Porównanie kodów ZM, U1 i U2

Kod	Reprezentacja +3	Reprezentacja -3	Reprezentacja 0
ZM	0011	1011	0000 lub 1000
U1	0011	1100	0000 lub 1111
U2	0011	1101	0000

Przekształcanie liczb w kod ZM





1

Krok 1: Określenie znaku

Ustal pierwszy bit. 0 dla liczb dodatnich, 1 dla ujemnych

2

Krok 2: Konwersja na binarny

Przekształć wartość bezwzględną liczby na system dwójkowy

3

Krok 3: Uzupełnienie zerami

Dodaj zera wiodące, aby uzyskać wymaganą długość słowa

4

Krok 4: Połączenie

Połącz bit znaku z binarną reprezentacją wartości bezwzględnej

Przykład przekształcenia w kod ZM

1. **Liczba: -13**

   Znak: ujemny (1). Wartość bezwzględna: 13

2. **Konwersja 13 na binarny**

   13 w systemie dwójkowym to 1101

3. **8-bitowa reprezentacja**

   Uzupełniamy zerami: 00001101

4. **Wynik końcowy**

   -13 w kodzie ZM: 10001101

Przekształcanie liczb w kod U1

1 **Liczby dodatnie**

Proces identyczny jak w kodzie ZM. Pierwszy bit to 0

2 **Liczby ujemne**

Najpierw zapisz liczbę jak dodatnią, potem zaneguj wszystkie bity

3 **Negacja bitów**

Zamień każde 0 na 1 i każde 1 na 0

4 **Uwaga na zero**

Zero ma dwie reprezentacje: 00...0 i 11...1

Przykład przekształcenia w kod U1

1

Liczba: -7

Zaczynamy od dodatniej reprezentacji 7

2

7 w binarnym

0111 (dla 8-bitowej reprezentacji: 00000111)

3

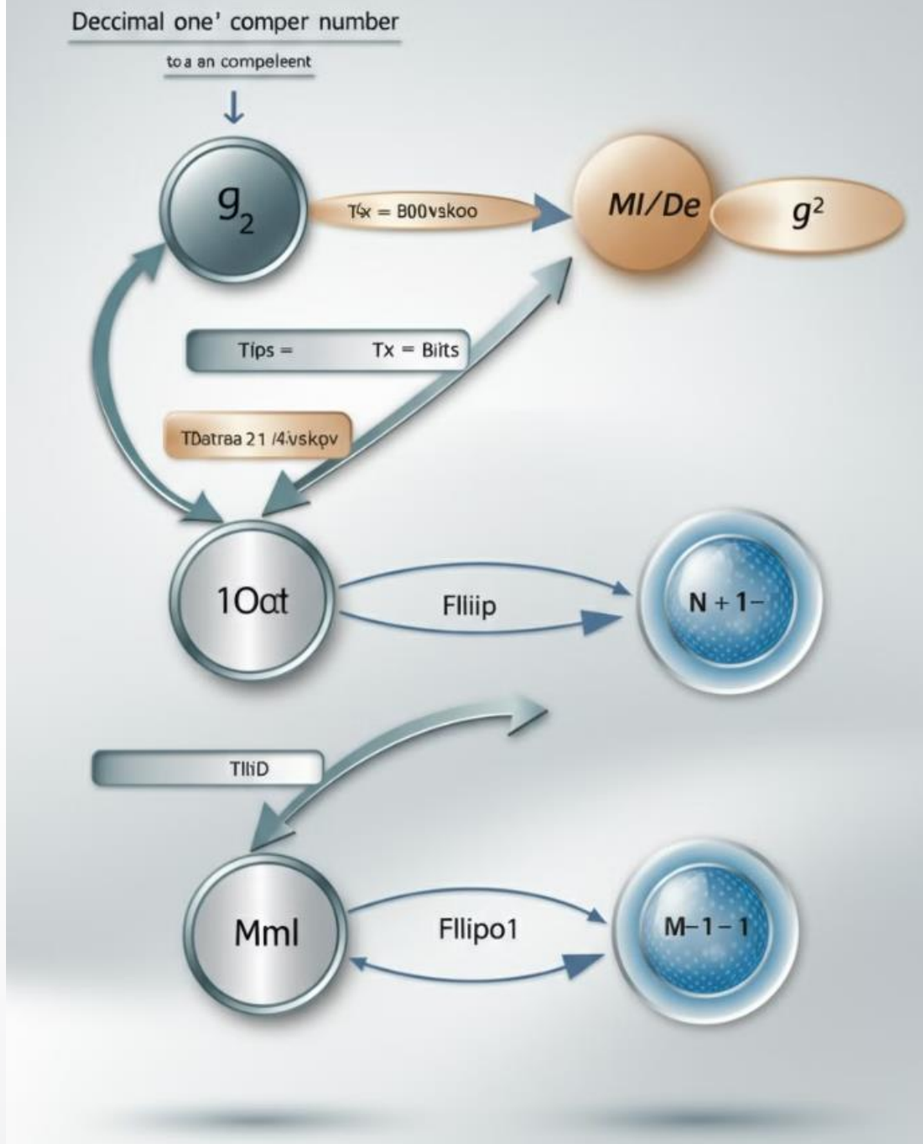
Negacja bitów

Zamieniamy każdy bit: 11111000

4

Wynik

-7 w kodzie U1: 11111000



Przekształcanie liczb w kod U2

1 **Liczby dodatnie**

Proces identyczny jak w U1 i ZM. Pierwszy bit to 0

2 **Liczby ujemne - krok 1**

Zapisz liczbę jak dodatnią i zaneguj wszystkie bity (jak w U1)

3 **Liczby ujemne - krok 2**

Do negacji dodaj 1 (w systemie binarnym)

4 **Zaleta**

Zero ma tylko jedną reprezentację: 00...0

Przykład przekształcenia w kod U2

1

Liczba: -5

Zaczynamy od dodatniej reprezentacji 5

2

5 w binarnym

0101 (dla 8-bitowej reprezentacji: 00000101)

3

Negacja bitów

11111010 (jak w U1)

4

Dodanie 1

11111010 + 1 = 11111011

5

Wynik

-5 w kodzie U2: 11111011

Obliczenia na liczbach w kodzie ZM

Dodawanie liczb o tym samym znaku

Dodaj wartości bezwzględne. Zachowaj wspólny znak

Dodawanie liczb o różnych znakach

Odejmij mniejszą wartość od większej. Znak wyniku zależy od większej liczby

Odejmowanie

Zmień znak odjemnika i wykonaj dodawanie

Komplikacje

Operacje wymagają analizy znaków. Możliwe błędy przy przepełnieniu

Przykład obliczeń w kodzie ZM

1. **Zadanie: 5 + (-3)**

   5 w ZM: 00000101, -3 w ZM: 10000011

2. **Analiza znaków**

   Różne znaki. Odejmujemy mniejszą wartość od większej

3. **Obliczenie**

   5 - 3 = 2. Znak będzie dodatni (większa liczba)

4. **Wynik**

   2 w ZM: 00000010

Obliczenia na liczbach w kodzie U1

Dodawanie

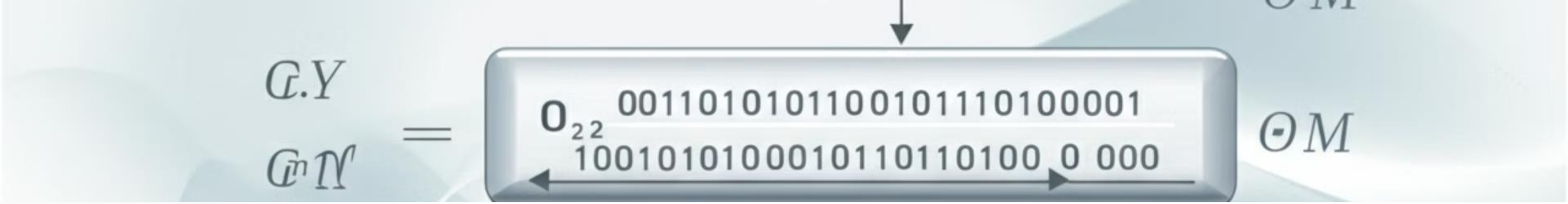
Dodaj bity normalnie. Jeśli wystąpi przeniesienie, dodaj 1 do wyniku

Odejmowanie

Zamień odjemnik na jego U1 i wykonaj dodawanie

Przeniesienie cykliczne

Przeniesienie z najbardziej znaczącego bitu dodaj do najmniej znaczącego



Przykład obliczeń w kodzie U1

1

Zadanie: 4 + (-2)

4 w U1: 00000100, -2 w U1: 11111101

2

Dodawanie

00000100 + 11111101 = 11111001 (z przeniesieniem)

3

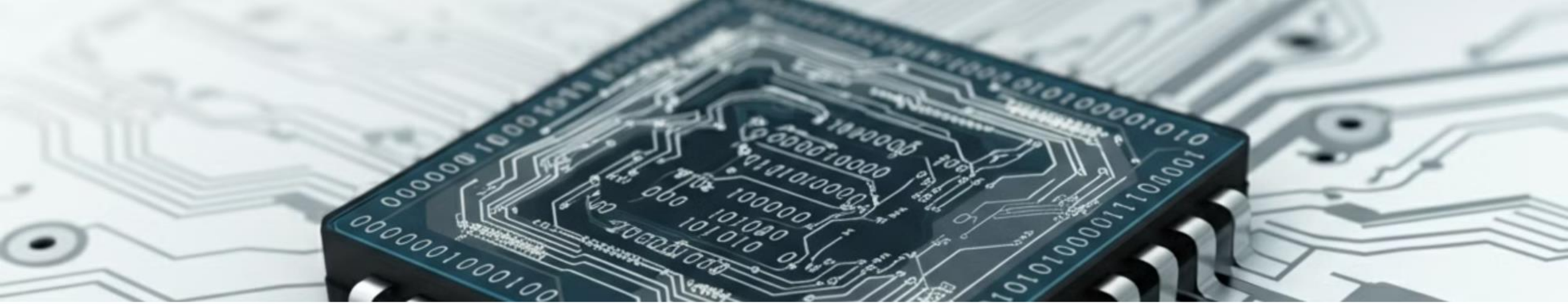
Przeniesienie cykliczne

Dodajemy 1 do wyniku: 11111001 + 1 = 00000010

4

Interpretacja

Wynik to 2 w kodzie U1



Obliczenia na liczbach w kodzie U2

1 Dodawanie

Dodaj bity normalnie. Ignoruj przeniesienie z najbardziej znaczącego bitu

2 Odejmowanie

Zamień odjemnik na jego U2 i wykonaj dodawanie

3 Uproszczenie

Nie wymaga przeniesienia cyklicznego
Operacje są prostsze niż w U1

4 Przepełnienie

Łatwe do wykrycia.
Występuje, gdy znak wyniku jest niepoprawny

Przykład obliczeń w kodzie U2

1. **Zadanie: 6 + (−3)**

   6 w U2: 00000110, -3 w U2: 11111101

2. **Dodawanie**

   00000110 + 11111101 = 00000011

3. **Interpretacja**

   Wynik to 3 w kodzie U2

4. **Sprawdzenie przepełnienia**

   Brak przepełnienia. Znak wyniku jest poprawny

Porównanie efektywności kodów

Operacja	ZM	U1	U2
Dodawanie	Skomplikowane	Średnio złożone	Proste
Odejmowanie	Skomplikowane	Średnio złożone	Proste
Wykrywanie przepełnienia	Trudne	Średnio trudne	Łatwe

Zastosowania praktyczne kodów liczbowych

Kalkulatory

Wykorzystują kody U2 do efektywnych obliczeń na liczbach całkowitych

Procesory

Operują na liczbach w kodzie U2 dla szybkich operacji arytmetycznych

Sieci komputerowe

Wykorzystują kody do przesyłania i przetwarzania danych liczbowych

Bazy danych

Przechowują i przetwarzają dane liczbowe w efektywnych formatach binarnych

Wyzwania związane z kodami liczbowymi

Ograniczony zakres

Liczba bitów determinuje maksymalną i minimalną reprezentowalną wartość

Przepełnienie

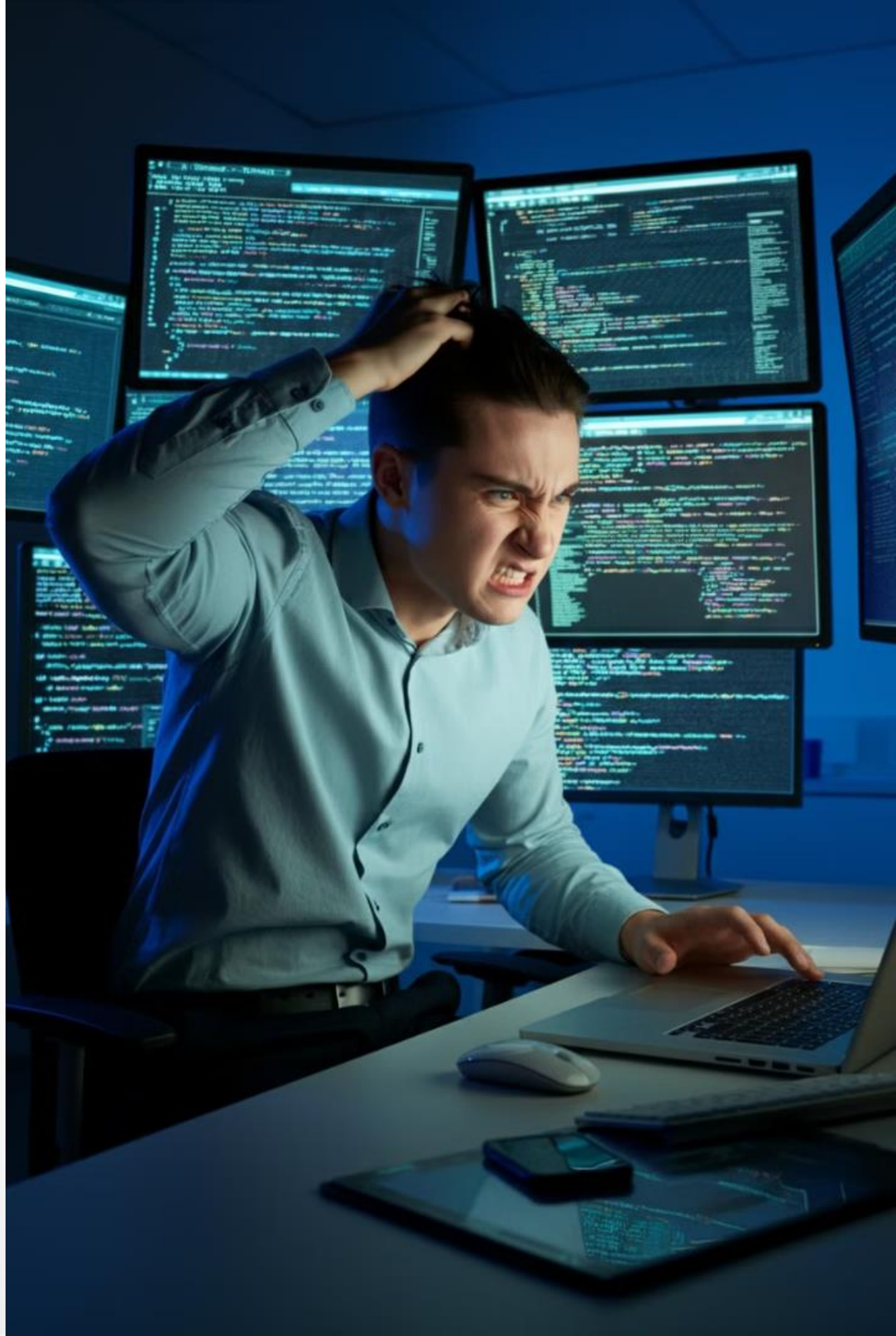
Może prowadzić do nieoczekiwanych wyników, gdy wynik przekracza dostępny zakres

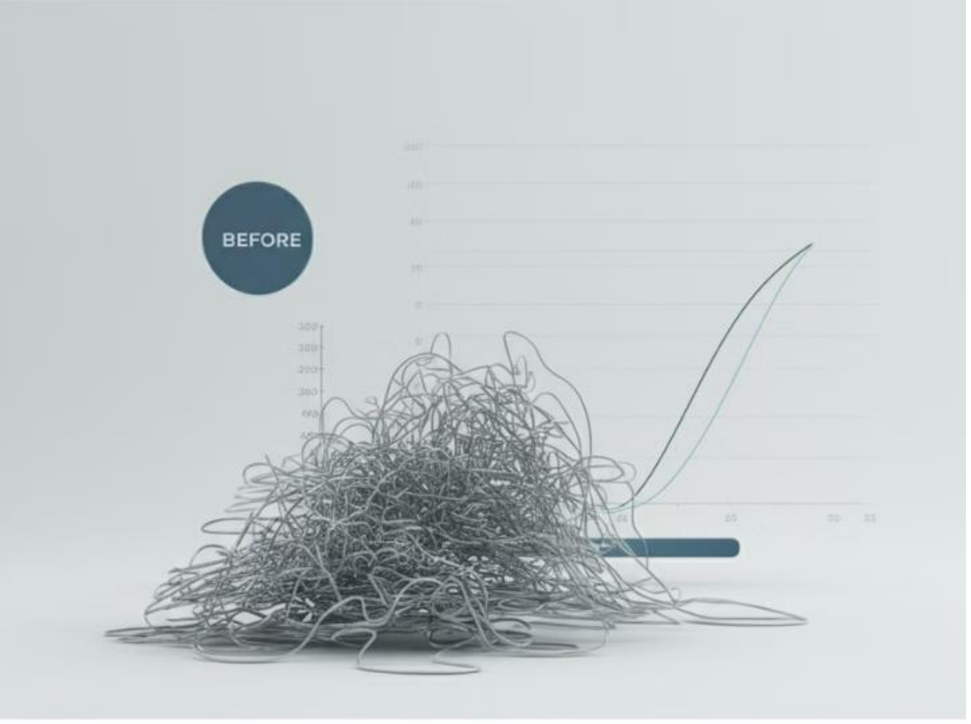
Precyzja

Ograniczona dla liczb zmiennoprzecinkowych, co może prowadzić do błędów zaokrągleń

Złożoność debugowania

Trudności w interpretacji błędów wynikających z niepoprawnych operacji na kodach.





Optymalizacja kodu z wykorzystaniem wiedzy o kodach liczbowych

1

Analiza zakresu

Wybierz odpowiedni typ danych, uwzględniając zakres wartości w programie

2

Operacje bitowe

Wykorzystaj operacje na poziomie bitów dla szybszych obliczeń

3

Unikanie przepełnienia

Implementuj kontrole zapobiegające nieoczekiwanym wynikom przy przepełnieniu

4

Optymalizacja pamięci

Minimalizuj zużycie pamięci poprzez efektywne kodowanie danych liczbowych

Przyszłość kodów liczbowych

1 **Kwantowe obliczenia**

Nowe paradygmaty reprezentacji liczb w komputerach kwantowych

2 **Rozszerzone formaty**

Rozwój formatów o większej precyzji i szerszym zakresie dla zaawansowanych zastosowań

3 **Sztuczna inteligencja**

Specjalizowane formaty liczbowe optymalizowane pod kątem algorytmów AI

4 **Energooszczędność**

Nowe kody zoptymalizowane pod kątem minimalnego zużycia energii w obliczeniach