Technologie Informacyjne i Komunikacyjne 2023/2024

dr Magdalena Posiadała-Zezula,

Wykład 3:
Reprezentacje liczb

Photo Credit: Piotr Mijakowski

Technologie Informacyjne i Komunikacyjne

- Dr Magdalena Posiadala-Zezula, email mposiada@fuw.edu.pl
- Strona przedmiotu : https://www.fuw.edu.pl/~mposiada/dydaktyka/tik23/index.html

Reprezentacje liczb i znaków

- Liczby:
  - Reprezentacja **„naturalna”** – nieujemne liczby całkowite – naturalny system dwójkowy.
  - Reprezentacje **„umowne”** – liczby ujemne, liczby niecałkowite
- Znaki:
  - Tylko reprezentacje „umowne” – zbiory znaków (ang. character set).
  - ASCII (m. in. $1000001_{bin} = A$, $1000010_{bin} = B$ itd.)
  - Strony kodowe, standardy ISO-8859, Unicode.

Zapis binarny (1)

- Komputery przechowują rozkazy, a także liczby i znaki w postaci ciągów **cyfr 0 i 1**, zwanych **cyframi binarnymi lub bitami (ang. BInary digiT).**
- Cyfry binarne łatwo jest przechowywać. Weźmy dowolny układ o dwóch (łatwo rozróżnialnych) stanach.
- Oznaczmy te stany jako „**↑**” **i „↓”.**
- Wprowadzając układ w stan „**↓**” **zapisujemy cyfrę 0.**
- Wprowadzając układ w stan „**↑**” **zapisujemy cyfrę 1.**
- Zbiór takich układów może reprezentować ciąg cyfr 0 i 1.
- Z takich układów zbudowana jest pamieć komputera.
- Ciągi cyfr binarnych (bitów) sa nazywane słowami (ang. word).

Zapis binarny (2)

- Wektor znaków binarnych:
  - **1 – elementowy – bit**
  - 4 – elementowy – połowa bajtu
  - **8 – elementowy – bajt**
  - 16 – elementowy – słowo 16 bitowe
  - 32 – elementowy – słowo 32 bitowe, itd

Naturalny system dwójkowy (1)

- **Naturalny system dwójkowy** (ang. **NBS - Natural Binary System**) jest najprostszym systemem pozycyjnym, w którym podstawa wynosi 2.
- System posiada dwie cyfry 0 i 1.

Wartość dziesiętna liczby zapisanej w naturalnym kodzie binarnym

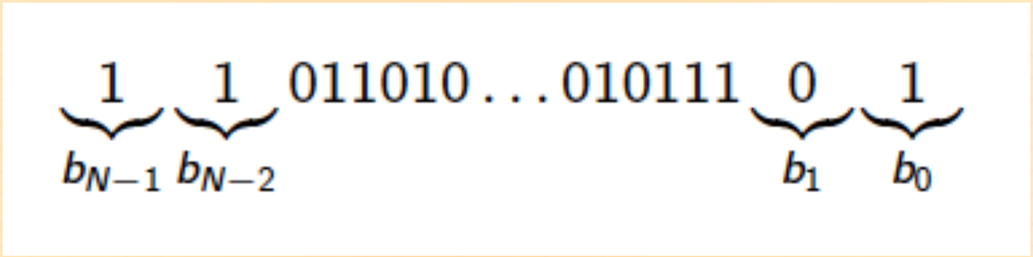
$$b_{n-1}b_{n-2}\ldots b_2b_1b_0 = b_{n-1}2^{n-1} + b_{n-2}2^{n-2} + \ldots + b_2 2^2 + b_1 2^1 + b_0 2^0$$

gdzie

b - bit, cyfra dwójkowa 0 lub 1
n - liczba bitów w zapisie liczby

Naturalny system dwójkowy (2)

Dla słowa **N-bitowego**:



- **b_j – bit (cyfra binarna) na pozycji j = 0, 1, . . . ,N – 1**
- Waga bitu odpowiada jego pozycji w słowie:
  - **b_0 – najmniej znaczący (najmłodszy) bit.**
  - **b_{N-1} – najbardziej znaczący (najstarszy) bit.**

Naturalny system dwójkowy (3) -przykłady

Obliczyć wartość liczby dwójkowej $11100101_{(2)}$.

$$11100101_{(2)} = 1 \times 2^7 + 1 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$$

$$11100101_{(2)} = 1 \times 128 + 1 \times 64 + 1 \times 32 + 0 \times 16 + 0 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1$$

$$11100101_{(2)} = 128 + 64 + 32 + 4 + 1$$

$$11100101_{(2)} = 229_{(10)}$$

Zakres liczby dwójkowej (1)

- Jaką największą liczbę dwójkową możemy zapisać za pomocą **n bitów**?
- Największa liczba musi posiadać same cyfry 1, czyli w wartości liczby muszą uczestniczyć wszystkie wagi pozycji.

dla 1b mamy $1_{(2)} = 1_{(10)}$
dla 2b mamy $11_{(2)} = 2 + 1 = 3_{(10)}$
dla 3b mamy $111_{(2)} = 4 + 2 + 1 = 7_{(10)}$
dla 4b mamy $1111_{(2)} = 8 + 4 + 2 + 1 = 15_{(10)}$
...

dla 1b mamy 1
dla 2b mamy 3
dla 3b mamy 7
dla 4b mamy 15
...

Zakres liczby dwójkowej (2)

- Liczby te tworzą ciąg liczbowy:

dla 1b mamy $1 = 2^1 - 1$
dla 2b mamy $3 = 2^2 - 1$
dla 3b mamy $7 = 2^3 - 1$
dla 4b mamy $15 = 2^4 - 1$
...

- **Wykładnik potęgowy liczby 2 jest równy ilości bitów**, zatem dla n bitów otrzymujemy wzór:

$$2^n - 1$$

Zakres liczby dwójkowej (3) - przykład

- Jaką największą liczbę dziesiętną można przedstawić za pomocą 34 bitów?

  - $L=2^{34}-1= 17179869184 -1= 17179869183$

Dodawanie dwójkowe

- W systemie binarnym mamy tylko dwie cyfry 0 i 1, zatem tabliczka dodawania składa się tylko z 4 pozycji:
  - **0 + 0 = 0**
  - **0 + 1 = 1**
  - **1 + 0 = 1**
  - **1 + 1 = 10**

Dodawanie dwójkowe -przykład

- Zsumować liczby binarne $1111001_{(2)}$ oraz $10010_{(2)}$.
- Sumowane liczby zapisujemy jedna pod drugą tak, aby w kolejnych kolumnach znalazły się cyfry stojące na pozycjach o tych samych wagach (identycznie postępujemy w systemie dziesiętnym zapisując liczby w słupkach przed sumowaniem):

```
    11
  1111001
+ 10010
-------
= 10001011
```

Dodawanie dwójkowe- UWAGA (1)

- W pamięci komputera liczby binarne przechowywane są w postaci ustalonej ilości bitów (np. 8, 16, 32 bity).
- Jeśli wynik sumowania np. dwóch liczb 8 bitowych jest większy niż 8 bitów, to najstarszy bit (dziewiąty bit) zostanie utracony.
- Sytuacja taka nazywa się **nadmiarem (ang. overflow) i występuje zawsze, gdy wynik operacji arytmetycznej jest większy niż górny zakres danego formatu liczb binarnych (np. dla 8 bitów wynik większy od $2^8 - 1$, czyli większy od 255):**
- $\mathbf{11111111_{(2)} + 00000001_{(2)} = 1|00000000_{(2)}}$ **(255 + 1 = 0)**

Dodawanie dwójkowe- UWAGA (2)

- Przy nadmiarze otrzymany wynik jest zawsze błędny!!!
- Należy wtedy **ustalić typy danych** zgodnie z przewidywanym zakresem wartości otrzymywanych wyników.
- Np. w C++ mamy zmienne typu int (integer - całkowite), które mają rozmiar minimum 4 bajty .

Odejmowanie dwójkowe

- Przy odejmowaniu korzystamy z tabliczki odejmowania, która w systemie binarnym jest bardzo prosta:
  - 0 - 0 = 0
  - **0 - 1 = 1 i pożyczka do następnej pozycji**
  - 1 - 0 = 1
  - 1 - 1 = 0
- Odejmując 0 - 1 otrzymujemy wynik 1 i pożyczkę (ang. borrow) do następnej pozycji. **Pożyczka oznacza konieczność odjęcia 1 od wyniku odejmowania cyfr w następnej kolumnie**. Identycznie postępujemy w systemie dziesiętnym, tyle że tam jest to o wiele bardziej skomplikowane.

Odejmowanie dwójkowe- przykład (1)

- Wykonać odejmowanie w systemie binarnym
  - $1101110_{(2)} - 1111_{(2)}$.
- Obie liczby umieszczamy jedna pod drugą tak, aby ich cyfry znalazły się w kolumnach o tych samych wagach

```
  1101110
-    1111
```

- Odejmowanie rozpoczynamy od cyfr ostatniej kolumny. Wyniki zapisujemy pod kreską. W tym przykładzie odjęcie ostatnich cyfr 0 - 1 daje wynik 1 oraz pożyczkę do następnej kolumny. Pożyczki zaznaczamy kolorem czerwonym.

```
      1
  1101110
-    1111
-------
       1
```

Odejmowanie dwójkowe- przykład (2)

- Odjęcie cyfr w drugiej od końca kolumnie daje wynik 1 - 1 = 0. Od tego wyniku musimy odjąć pożyczkę 0 - 1 = 1 i pożyczka do następnej kolumny.

```
    1 1
  1101110
-    1111
---------
      11
```

- Według tych zasad kontynuujemy odejmowanie cyfr w pozostałych kolumnach. Pamiętaj o pożyczkach! Jeśli w krótszej liczbie zabraknie cyfr, to możemy kolumny wypełnić zerami:

```
  1 1 1 1 1
  1101110
- 0001111
---------
  1011111
```

Odejmowanie dwójkowe -niedomiar

- Jeśli od liczby mniejszej odejmiemy większą, to wynik będzie ujemny. Jednakże w naturalnym systemie binarnym nie można zapisywać liczb ujemnych. Zobaczmy zatem co się stanie, gdy od liczby 0 odejmiemy 1, a wynik ograniczymy do 8 bitów.

```
 11111111
  00000000
- 00000001
 ----------
  11111111
```

- Otrzymujemy same jedynki, a pożyczka nigdy nie zanika. Sytuacja taka nazywa się **niedomiarem (ang. underflow) i występuje zawsze, gdy wynik operacji arytmetycznej jest mniejszy od dolnego zakresu formatu liczb binarnych (dla naturalnego kodu dwójkowego wynik mniejszy od zera).**

Kodowanie liczb ze znakiem (1)

- Dotychczas przedstawiliśmy naturalny system binarny (NBS), który pozwala na zapisywanie liczb dodatnich oraz zera w postaci bitów o wartościach 0,1.
- W przypadku liczb ujemnych mamy problem, który polega na wymyślenia takiego sposobu kodowania, aby za pomocą bitów można było zapisywać wartości ujemne – czyli też i znak liczby.

Kodowanie liczb ze znakiem (2)

- Wybrane metody kodowania liczb ze znakiem:

1. System znak-moduł (ZM)
2. Zapis w systemie uzupełnień do 2 - U2

Zapis w systemie znak – moduł (ZM)

- W zapisie **znak-moduł (ZM**- ang. SM Signed Magnitude) liczba składa się **z dwóch części – bitu znaku oraz bitów wartości liczby (modułu)**
- $b_{n-1}b_{n-2}b_{n-3}\ldots b_2b_1b_0$ gdzie:
- **b_{n-1} – bit znaku liczby**
- $b_{n-2\ldots}b_1b_0$ – bity modułu liczby.
- **Dla liczb dodatnich bit znaku =0,**
- **Dla ujemnych i zera bit znaku =1**.

Zapis ZM

- Moduł liczby ZM jest zapisany w naturalnym kodzie dwójkowym NBS, zatem w celu obliczenia jej wartości moduł mnożymy przez 1, gdy bit znaku wynosi 0 lub przez -1, gdy bit znaku wynosi 1. Wzór jest następujący:

$$\text{Liczba}_{ZM} = (-1)^{\text{bit znaku}} \cdot \text{modul liczby}$$

$$b_{n\text{-}1}b_{n-2}..b_2b_1b_0 = (-1)^{b_{n-1}} \cdot (b_{n-2}2^{n-2} + ... + b_2 2^2 + b_1 2^1 + b_0 2^0)$$

gdzie :

b - bit, cyfra dwójkowa 0 lub 1,

n - liczba bitów w zapisie liczby

Zapis ZM- przykłady

Tabela przedstawia wszystkie możliwe do utworzenia liczby w zapisie ZM dla 3 bitów.

Liczbę 0 realizuje się przed dwa słowa kodowe : 0000 oraz 1000.

Kod ZM	Wyliczenia	Wartość
1011	$(-1)^1 x (2^1+2^0)$	(-3)
1010	$(-1)^1 x (2^1)$	(-2)
1001	$(-1)^1 x (2^0)$	(-1)
0000 lub 1000	$(-1)^1 \times 0$	0
0001	$(-1)^0 x (2^0)$	1
0010	$(-1)^0 x (2^1)$	2
0011	$(-1)^0 x (2^1+2^0)$	3

Zapis ZM- zadanie

- Oblicz wartość dziesiętną liczby $100111_{(ZM)}$.
- Rozwiązanie:
  - Pierwszy bit zapisu **ZM jest bitem znaku. Wartość 1 informuje nas, iż jest to liczba ujemna.**
  - $100111_{(ZM)} = (-1)^1 \times (0x2^4+0x2^3+1x2^2+1x2^1+1x2^0)$
  - $100111_{(ZM)} = -(4+2+1) = -7$

Przeliczanie liczb dziesiętnych na liczby ZM

- Wyznaczanie zapisu **ZM** dla danej liczby dziesiętnej:

1. Jeśli liczba jest dodatnia, to bit znaku ma wartość 0. W przeciwnym razie bit znaku ma wartość 1.
2. Obliczamy wartość absolutną liczby, czyli jej moduł.Wyznaczamy bity modułu według metody przeliczania liczb dziesiętnych na system dwójkowy.
3. Aby otrzymać zadana w formacie liczbę bitów dla modułu do otrzymanych bitów modułu dopisujemy bity o wartości 0.
4. Na koncu do bitów modułu dodaje się bit dla znaku i mamy zapis ZM.

Przeliczanie liczb dziesiętnych na liczby ZM - przykład

- Przedstawić w 8 bitowym kodzie ZM liczbę o wartości dziesiętnej -7.
  - Rozwiązanie:
    - Wyznaczamy bit znaku. Liczba -7 jest ujemna, zatem $b_7 = 1$ (najstarszy bit).
    - Wartość absolutna, czyli moduł z -7 to 7 (po prostu opuszcza się znak -).
    - Obliczamy zapis dwójkowy modułu 7=111.
    - Moduł 8 bitowej liczby ZM ma 7 bitów (numeracja bitów od zera) dlatego do otrzymanego wyniku dodajemy 4 zera 0000111
    - Na końcu wpisujemy jeszcze bit znaku czyli 1 i mamy $10000111_{(ZM)}$=-7

Zapis ZM - wady

- Przedstawiony system ZM zapisu liczb ze znakiem był prosty, lecz stwarzał poważne problemy przy wykonywaniu operacji arytmetycznych.
- **Liczba w zapisie ZM nie jest jednorodna.**
- **Bit znakowy posiada zupełnie inne znaczenie od pozostałych bitów** i nie uczestniczy bezpośrednio w operacjach arytmetycznych.

Kodowanie liczb ze znakiem

- Wybrane metody kodowania liczb ze znakiem:

1. System znak-moduł (ZM)
2. **Zapis w systemie uzupełnień do 2 - U2**

Reprezentacja uzupełnień do 2 – U2

- Liczbę w systemie U2 kodujemy zgodnie z poniższą formułą:

$$b_{n-1}b_{n-2}..b_2b_1b_0 = b_{n-1}(-2)^{n-1} + b_{n-2}2^{n-2} + ...+ b_2 2^2 + b_1 2^1 + b_0 2^0$$

gdzie :

b - bit, cyfra dwójkowa 0 lub 1,

n - liczba bitów w zapisie liczby

Reprezentacja uzupełnień do 2 – U2 - przykłady

W systemie U2 zero ma tylko jedną wartość w odróżnieniu od ZM.

Najstarszy bit znowu określa znak liczby.

Jeśli jest równy 0, liczba jest dodatnia i resztę zapisu możemy potraktować jak liczbę w NBS.

Kod U2	Wyliczenia	Wartość
1101	$(-2^3)+ 2^2+ 2^0$	-3
1110	$(-2^3)+ 2^2+2^1$	-2
1111	$(-2^3)+ 2^2+2^1+ 2^0$	-1
0000	0	0
0001	2^0	1
0010	2^1	2
0011	2^1+2^0	3

Jeśli bit znaku ustawiony jest na 1, to liczba ma wartość ujemną.
Bit znaku ma wagę (-2^n-1), gdzie n oznacza liczbę bitów w wybranym formacie **U2.**

Przeliczanie liczb dziesiętnych na liczby U2 - przykład

Zasada: Przeliczanie ujemnej liczby dziesiętnej na zapis **U2:** Jeśli do liczby 2^n (n - ilość bitów w formacie U2) dodamy przetwarzaną liczbę dziesiętną, to w wyniku otrzymamy wartość kodu dwójkowego równoważnego bitowo (tzn. o takiej samej postaci) kodowi **U2 przetwarzanej liczby. Wynik dodawania wystarczy zapisac w postaci kodu NBS.**

- Przykład:
- Wyznaczyć 8 bitowy kod **U2 dla liczby dziesiętnej $(-7)_{(10)}$.**
- **$2^8 + (-7) = 256 - 7 = 249 = 11111001_{(2)}$.**
- **Stąd$(-7)_{(10)} = 11111001_{(U2)}$.**

Reprezentacja liczb rzeczywistych- ułamki

$3.245_{(10)} = 3x10^{0} + 2x10^{-1} + 4x10^{-2} + 5x10^{-3}$ - system dziesiętny

$5.25_{(10)} = 101.01_{(2)}$

$101.01 = \mathbf{1x2^{2}+0x2^{1}+1x2^{0}+0x2^{-1}+1x2^{-2}}$

część całkowita

część ułamkowa

Reprezentacja liczb rzeczywistych - przeliczanie

$125_{(10)}=?_{(2)}$

Liczby całkowite :
Operacja modulo % – reszta z dzielenia:

125%2=62 reszta 1
62% 2= 31 reszta 0
31% 2=15 reszta 1
15%2=7 reszta 1
7%2=3 reszta 1
3%2=1 reszta 1
1%2= 0 reszta 1

Część całkowita: spisujemy **od DOŁU !!!**
$125_{(10)}=1111101_{(2)}$!!!

$125.40625_{(10)}=?_{(2)}$

Część całkowita- operacja modulo %
Część ułamkowa- operacja mnożenia * 2

0.40625*2 = **0**.8125
0.8125*2 = **1**.625
0.625 *2 = **1**.25
0.25*2 = **0**.5
0.5*2 = **1**.0

Część ułamkowa: spisujemy **od GÓRY!!!**

$125.40625_{(10)}=1111101.01101$!!!

Zadania

Zad 1. Wyrazić liczbę w systemie dziesiętnym: $10111101_{(U2)}$?

Zad2. Wyrazić liczbę w systemie dwójkowym: 240?

Zad3. Wyrazić liczbę w systemie dwójkowym 347,2345?


Thank you!
Photo Credit: Piotr Mijakowski
ν_e
ν_μ
ν_τ