D-010	Szalas	Natalia
D-011	Tomasik	Kacper

Liczby binarne

Temat 5

Next

Czym jest system Binarny?

Tak jak powszechnie stosowany system liczbowy dziesiętny, system binarny (zwany także dwójkowym) jest systemem pozycyjnym. Co to znaczy system pozycyjny dowiemy się BARDZO dobrze na lekcjach z algebry i logiki, na razie wystarczy nam wiedza, iż posługuje się takimi samymi regułami jak system dziesiętny.
Jedna wartość jest nazywana **bit**em. Jeden **byte** to 8 bitów, może przechowywać wartości do 2^8 (maksymalna wartość jednego byte'a to 255).
W programowanie liczby binarne zaznaczane są przedrostkiem `0b`.

2^3	2^2	2^1	2^0	2^{-1}
1	1	0	1	1
8	4	0	1	0.5

Każda kolejna liczba przyjmuje wartość dwa razy większą niż poprzednia.
Żeby przekształcić liczbę dwójkową na dziesiętną, wystarczy pomnożyć każdą cyfrę przez potęgę dwójki odpowiadającą miejscu (miejsce przed przecinkiem zaliczamy jako 2^0), a następnie, dodać całość; otrzymamy w ten sposób liczbę w systemie dziesiętnym. Możemy w taki sposób dodać:

8	+	4	+	0	+	1	+	0.5	=	13.5

W naszym przypadku, otrzymana liczba to 13.5

Prev Next

Przekształcanie liczb dziesiętnych na dwójkowe

Przekształcanie liczb na dwójkowe może przebiegać analogicznie. Znajdujemy największą potęgę liczby **2** mniejszą niż dana liczba i odejmujemy ją od liczby dziesiętnej oraz zapisujemy 1.
Zmniejszamy potęge **dwójki** o 1, jeśli ta potęga jest większa niż nasza liczba, zapisujemy 0. Proces powtarzamy; jeżeli potęga **dwójki** jest mniejsza niż nasza liczba, potęgę tą odejmujemy i zapisujemy 1. Proces ten możemy powtarzać aż nie otrzymamy 0 jako liczby końcowej lub wystarczającego przybliżenia. Po dotarciu do potęgi 2^0, stawiamy przecinek.

47.25	-	2^5	=	15.25	→	1
15.25	<	2^4	=	15.25	→	0
15.25	-	2^3	=	7.25	→	1
7.25	-	2^2	=	3.25	→	1
3.25	-	2^1	=	1.25	→	1
1.25	-	2^0	=	0.25	→	1.
0.25	<	2^{-1}	=	0.25	→	0
0.25	-	2^{-2}	=	0	→	1

47.25 = 101111.01

System Oktalny

System Oktalny (zwany także systemem ósemkowym) jest systemem opartym na liczbie 8.
Jest używany głównie żeby uprościć zapis liczb binarnych; liczba $7_{(8)}$ to 3 pełne bity.
W programowanie liczby binarne zaznaczane są przedrostkiem `0o`.
Na przykład:

1	7	(8)

=

1	1	1	1	(2)

System Heksadecymalny

System Hexadecymalny to zapis korzystający z cyfr od 0 do 16... Nie mamy pojedyńczych cyfr oznaczających wartości większe od 9, więc używamy do tego celu pierwszych liter alfabetu: A-10, B-11, ... F-16.
Dwie cyfry heksadecymalne zawierają wartość jednego byte'a.
W programowanie liczby binarne zaznaczane są przedrostkiem `0x`.

9	f	(16)

=

1	0	0	1	1	1	1	1	(2)

Operacje logiczne

Operacje logiczne łączą **wartości logiczne**.

Iloczyn logiczny (*koniunkcja*)

Iloczyn logiczny oznaczamy symbolami: && (C++), `and` (Python) lub symbolem ∧ (Logika).
Wynik sumy logicznej jest prawdziwy tylko gdy wartości po obu stronach operacji są prawdziwe.

∧	**1**	**0**
1	1	0
0	0	0

Suma logiczna (*alternatywa*)

Sumę logiczną oznaczamy symbolami: || (C++), `or` (Python) lub symbolem ∨ (Logika).
Wynik sumy logicznej jest prawdziwy tylko gdy wartości po obu stronach operacji są prawdziwe.

∨	**1**	**0**
1	1	1
0	1	0

Negacja

Negację oznaczamy symbolami: ! (C++), `not` (Python) lub symbolem ¬ (Logika).
Jest ona jednoargumentowym działaniem. Wynik negacji to odwrotność wartości logicznej.

¬	**1**	**0**
=	0	1

Alternatywa rozłączna (*różnica symetryczna*, *XOR*)

Alternatywę rozłączną oznaczamy symbolami: ^ (C++ oraz Python) lub symbolem ⊻ (Logika).
Wynik alternatywy logicznej jest prawdziwy tylko jeśli jedna wartość jest prawdziwa.

⊻	**1**	**0**
1	0	1
0	1	0

W programowaniu, XOR można zapisać również jako `p!=q`

Implikacja

Implikację oznaczamy symbolem: symbolem ⇒ (Logika).
Implikacja nie posiada własnego symbolu w programowaniu, lecz można ją zapisać jako `!p || q`.
Wynik implikacji jest nieprawdziwy tylko jeśli pierwsza wartość jest fałszywa a druga prawdziwa.

p	q	p⇒q
0	0	**1**
0	1	**0**
1	0	**1**
1	1	**1**

Równoważność

Równoważność oznaczamy symbolami: `==` (Programowanie) lub ⇔ (Logika).
Wynik równoważności jest prawdziwy jeśli po obu stronach znajduje się taka sama wartość.

p	q	p⇔q
0	0	**1**
0	1	**0**
1	0	**0**
1	1	**1**

Operacje na liczbach binarnych

Działania na liczbach innych systemów numerycznych przebiegają analogiczne jak na liczbach dziesiętnych.

Dodawanie

		1	1	0	0	+
	1	1	0	0	1	=

1	0	0	1	0	1

1

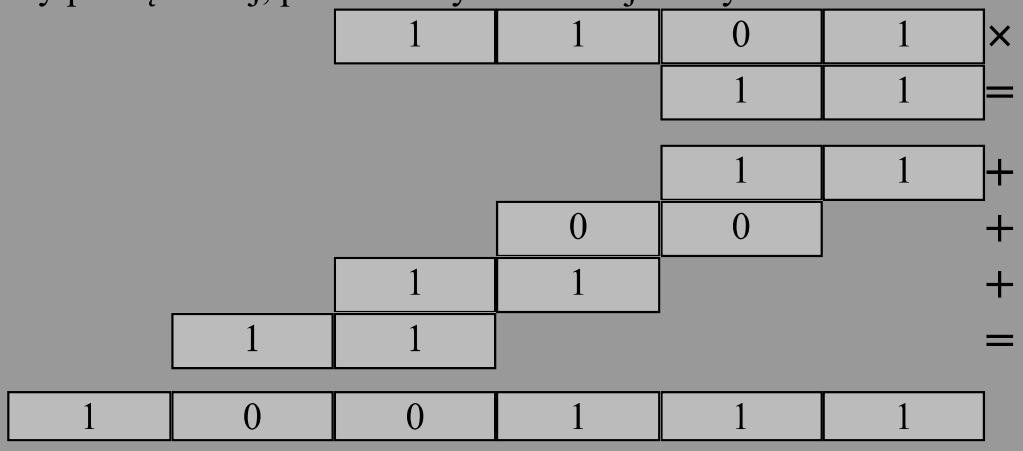
Odejmowanie

1	1	1	1	0	0	-
	1	1	0	1	0	=

1	0	0	0	1	0

Mnożenie

Mnożenie jest bardzo proste, mnożymy mnożnik przez mnożną i dodajemy do liczby początkowej, powtarzamy dla każdej liczby mnożnika.



Prev Next

Dzielenie

Dzielenie zaczynamy biorąc tyle cyfr z licznika (zaczynając od końca) ile ma mianownik i sprawdzamy czy jest większy, jeśli tak nie jest, musi dobrać jeszcze jedną cyfrę.

1	0	1	1	0	1	÷
			1	1	0	=

1	0	1	<	1	1	0

Więc musimy dobrać jeszcze jedną cyfrę i zapisać 0. Następnie odejmujemy od naszego okrojonego mianownika licznik.

1	0	1	1	-
	1	1	0	=
	1	0	1	

Zapisujemy 1 oraz powtarzamy działanie dobierając kolejną cyfrę

1	0	1	0		-
	1	1	0		=
	1	0	0	1	-
		1	1	0	=
			1	1	

Po dotarciu do końca liczby stawiamy przecinek i dobieramy 0 do końca liczby.

1	1	0	-
1	1	0	

Dzielenie kończymy gdy otrzymujemy resztę 0. Wynik to 0111.1

Jak zapisywane są liczby ujemne w komputerze?

Pierwszy bit przechowuje informacje o znaku liczby; 0 oznacza liczbę dodatnią, 1 oznacza liczbę ujemną.
Aby odejmować i dodawać liczby ujemne, musimy te liczby przekształcić odejmując 1, a następnie negując każdą cyfrę.
Na przykładzie liczby 92:

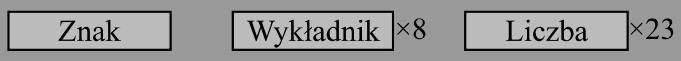
0	1	0	1	1	1	0	0	-
							1	=
0	1	0	1	1	0	1	1	

Liczbę teraz odwracamy i otrzymujemy -92:

1	0	1	0	0	1	0	0

Jak zapisywane są liczby dziesiętne w komputerze?

Komputery do tego zapisu używają potęgi liczby 10, tak jak w zapisie naukowym (Np. 3×10^{-8}).
Są one zapisywane na 32 bitach:



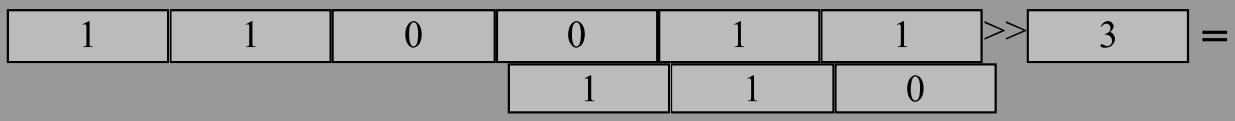
Bit "Znak" oznacza znak liczby, tak jak zostało to wytłumaczone wyżej.
Następne bity "Wykładnik" przedstawiają potęgę do której podnosimy liczbę 10 (to jest normalna liczba typu integer, więc może być ujemna).
Bity pod nazwą "Liczba" są normalnym zapisem liczby.

Zastosowania w komputerach

Liczby binarne są stosowane w komputerach ze względu na ich powiązanie z logiką; można w prosty sposób przedstawić 1 jako uruchomiony przełącznik i 0 jako nieuruchomiony przełącznik. Te przełączniki nazywamy **tranzystorami**, komputery potrafią odczytać taki zapis dzięki prądu który przez te tranzystory przebiega.
W dzisiejszych czasach, korzystamy z tranzystorów silikonowych, ich średnica jest o wiele mniejsza niż ta ludzkiego włosa, komputery używają ich w bilionach.

Komputery używają **bramek logicznych** żeby porównywać wartości przechowywane w tranzystorach. Wystarczą 3 bramki aby uzyskać dowolną funkcję logiczną, te bramki to:
-AND
-OR
-NOT
Często używaną bramką jest również bramka **XOR**.

Istnieje jeszcze jedno działanie na bitach nazywające się Bit shift, oznaczane jako << lub >> w Pythonie. Pozwala ono przemiścić bity o parę miejsc w lewo lub w pravo uzupełniając dodane miejsca o 0. Jeśli przesuwamy wartość w prawo o x miejsc, ostatnie x cyfr zostaje usunięte.



Jak działania arytmetyczne są zapisywane w komputerach

Działania arytmetyczne są zapisywane dzięki działaniom logicznym przedstawionych wcześniej.

Dodawanie

Mamy dwa Bity A i B, suma jest zapisywana jako A XOR B, natomiatst *carry* (przejście dalej) jest zapisywane jako A AND B.
Wynik A AND B jest dodawany do bitu na następnym miejscu.

Odejmowanie

Różnica jest zapisywana jako ((A NAND B) NAND B) NAND (A NAND B) NAND B), a *borrow* (...) jest zapisywane jako !((A NAND B) NAND B).*

**NAND oznacza po prostu Not AND, wynik NAND'u jest odwrotny do wyniku AND'u z tymi samymi wartościami.*

Funfacts

Nazwa jednostki **Bit** pochodzi od słów **B**inary dig**it**.

Byte powstał jako liczba bitów potrzebna by zakodować jeden znak (8 bitów), w dzisiejszych czasach możemy jednak używać formatów tekstu które pochłaniają o wiele więcej bitów.
ASCII używa tradycyjnie jednego byte'a, lecz jeden znak w formacie UTF-8 może wykorzystywać do 4 byte'ów (32 bity).

Koniec

Kacper Tomasik D-11

Natalia Szalas D-10

Prev