



# Zajebiste odpowiedzi typie makaki są sztos

**Sieć Aptek**

**i inne mały**

**June 25, 2022**

# Spis treści

<b>I</b>	<b>Pytania - dr. hab. Bogdan Księżopolski</b>	<b>5</b>
1	Sieci i programowanie sieciowe . . . . .	5
1.1	Protokoły TCP i UDP - porównanie i zastosowanie. . .	5
1.2	Protokół IP. . . . .	6
1.3	Modele sieci komputerowych. . . . .	6
1.4	Porównanie protokołów IPv4 i IPv6. . . . .	6
1.5	Format pakietu IP (poszczególne pola, zastosowanie). .	7
1.6	Ethernet. . . . .	7
1.7	Protokoły warstwy aplikacji. . . . .	7
1.8	Charakterystyka modelu OSI i TCP/IP. . . . .	7
1.9	Rodzaje i przykłady nagłówków HTTP. . . . .	7
1.10	Protokół WebSocket. . . . .	8
1.11	Serwer zdarzeniowy, a wielowątkowy. Charakterystyka i porównanie. . . . .	8
2	Bezpieka . . . . .	8
2.1	Infrastruktura klucza publicznego - charakterystyka. . .	8
2.2	Kryptografia symetryczna oraz asymetryczna - charakterystyka. . . . .	8
2.3	Bezpieczeństwo sieci w odniesieniu do warstw modelu TCP/IP. . . . .	10
2.4	Metody kontroli dostępu w systemach IT. . . . .	11
2.5	Atrybuty bezpieczeństwa informacji. . . . .	11
<b>II</b>	<b>Pytania - dr. hab. Grzegorz Wójcik</b>	<b>12</b>
1	Bazy danych . . . . .	12
1.1	Model relacyjny baz danych i języki zapytań. . . . .	12
1.2	Model obiektowo-relacyjny baz danych, inne modele danych. . . . .	12
1.3	Składnia podstawowych zapytań języka SQL. . . . .	12
1.4	Projektowanie baz danych oraz model związków encji. .	12

1.5	Problemy indeksowania baz danych, rodzaje indeksów, indeksy typu B+ drzewo. . . . .	12
1.6	Przetwarzanie transakcyjne OLTP (On-Line Transaction Processing). . . . .	13
2	Paradygmaty . . . . .	13
2.1	Założenia paradygmatu programowania obiektowego. . . . .	13
2.2	Idea dziedziczenia i polimorfizmu w programowaniu. . . . .	13
2.3	Zasady programowania dynamicznego. . . . .	13
2.4	Główne paradygmaty programowania. . . . .	13
2.5	Cechy programowania deklaratywnego. . . . .	13
<b>III Pytania - reszta</b>		<b>14</b>
1	Język C i C++ . . . . .	14
1.1	Instrukcje sterujące w języku C. . . . .	14
1.2	Zarządzanie pamięcią w języku C. . . . .	14
1.3	Budowa, obsługa i formatowanie łańcuchów znakowych w języku C. . . . .	14
1.4	Zasięg i czas życia obiektów w języku C++. . . . .	14
1.5	Obsługa wyjątków w języku C++. . . . .	14
1.6	Definicje obiektu, klasy i szablonu klasy w języku C++. . . . .	14
2	Algorytmy . . . . .	15
2.1	Algorytmy sortujące. . . . .	15
2.2	Algorytmy zachłanne. . . . .	15
2.3	Metoda „dziel i zwyciężaj” konstruowania algorytmów. . . . .	15
2.4	Struktura kopców binarnych. . . . .	15
2.5	Algorytmy wyszukiwania najkrótszej ścieżki w grafie. . . . .	15
2.6	Sposoby implementacji słownika. . . . .	15
2.7	Tablice mieszające. . . . .	15
2.8	Algorytmy Monte Carlo oraz algorytmy Las Vegas. . . . .	15
2.9	Metody rozwiązywania rekurencji. Rekurencje Flawiusza i wieża w Hanoi. . . . .	15
2.10	Algorytmy Euklidesa. Algorytmy faktoryzacji. . . . .	16
2.11	Metody reprezentacji grafów w komputerze. . . . .	16
2.12	Droga i cykl Eulera. Droga i cykl Hamiltona. . . . .	16
2.13	Drzewo spinające graf. . . . .	16
3	Teoria obliczalności czy coś . . . . .	16
3.1	Pojęcia P, NP, NP-zupełne. . . . .	16
4	Automaty i inne takie . . . . .	16
4.1	Deterministyczne i niedeterministyczne automaty skończone. . . . .	16
4.2	Automaty z epsilon przejściami, wyrażenia regularne. . . . .	16

4.3	Kompilacja: gramatyka bezkontekstowa, skaner, parser, błędy. . . . .	16
5	Podstawy komputera i systemy operacyjne . . . . .	17
5.1	Systemy liczbowe i konwersje pomiędzy nimi. . . . .	17
5.2	Sposoby cyfrowej reprezentacji liczby całkowitej i rzeczywistej. . . . .	17
5.3	Wielowarstwowa organizacja oprogramowania komputera. . . . .	20
5.4	Procesy, zasoby i wątki. . . . .	20
5.5	Planowanie przydziału procesora, priorytety, wywłaszczanie oraz planowanie. . . . .	20
5.6	Zarządzanie pamięcią operacyjną. . . . .	20
5.7	Problem zakleszczenia, algorytm Bankiera. . . . .	20
6	Inżynieria Oprogramowania . . . . .	20
6.1	Standardowe metodyki procesu wytwórczego oprogramowania. . . . .	20
6.2	Metodyki zwinne (agile). . . . .	20
6.3	Metody testowania oprogramowania. . . . .	20
6.4	Walidacja i weryfikacja oprogramowania. . . . .	20
6.5	Diagramy UML (przypadków użycia, klas, aktywności, sekwencji, stanów, obiektów, wdrożenia). . . . .	21
6.6	Wzorce projektowe programowania obiektowego. . . . .	21
6.7	Wzorce architektoniczne. . . . .	21
7	Systemy wbudowane i elektronika . . . . .	21
7.1	Różnice pomiędzy obsługą zdarzeń w przerwaniach sprzętowych a obsługą zdarzeń w pętli programowej. . . . .	21
7.2	Stosowalność systemów opartych o mikrokontrolery vs stosowalność typowych komputerów (stacjonarnych i laptopów). . . . .	21
7.3	Dekoder, multiplekser i demultiplekser: budowa, zasada, działania, przeznaczenie/zastosowanie. . . . .	21
7.4	Podstawowe układy budujące system mikroprocesorowy i sposób wymiany informacji pomiędzy nimi. . . . .	21

# Rozdział I

## Pytania - dr. hab. Bogdan Księżopolski

### 1 Sieci i programowanie sieciowe

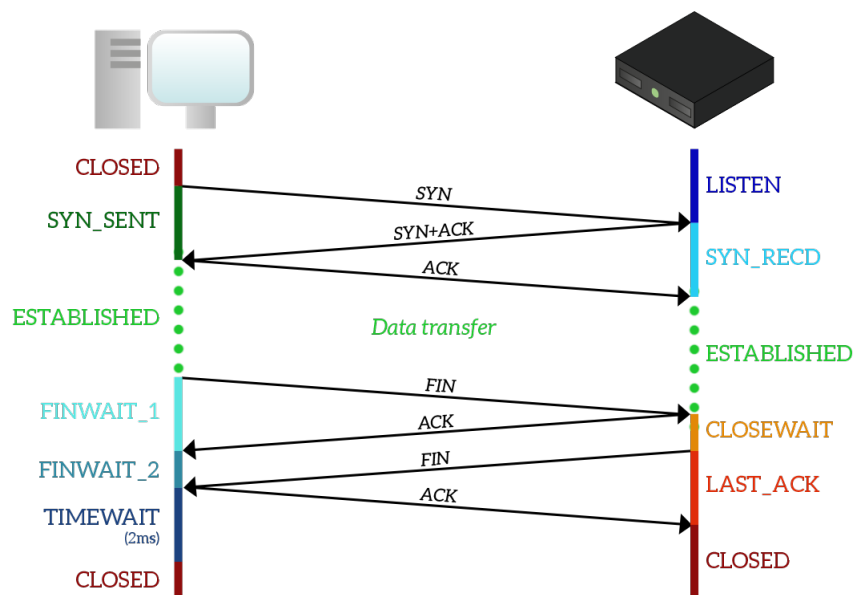
#### 1.1 Protokoły TCP i UDP - porównanie i zastosowanie.

##### TCP

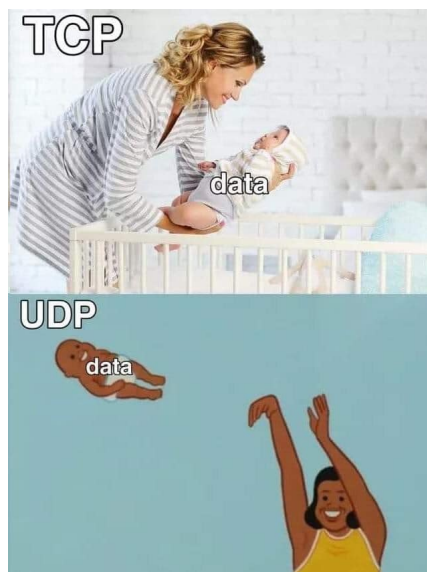
Protokół TCP lub Transmission Control Protocol jest protokołem zorientowanym na połączenie, znajdującym się w warstwie transportowej modelu TCP / IP. Nawiązuje połączenie między komputerem źródłowym a docelowym przed rozpoczęciem komunikacji.

Jest wysoce niezawodny, ponieważ wykorzystuje 3-drożną kontrolę uzgadniania, przepływu, błędów i przeciążenia. Zapewnia to, że dane wysyłane z komputera źródłowego są dokładnie odbierane przez komputer docelowy. Jeśli w przypadku, otrzymane dane nie są w odpowiednim formacie, to TCP ponownie przesyła dane. Poniższe protokoły używają TCP do transmisji danych:

- HTTP
- HTTPS
- FTP
- SMTP



Rysunek I.1: TCP



Rysunek I.2: UDP :D

## UDP

Protokół UDP lub User Datagram Protocol to bezpołączeniowy protokół znajdujący się w warstwie transportowej modelu TCP / IP. Nie ustanawia

połączenia ani nie sprawdza, czy komputer docelowy jest gotowy do odbioru, czy też nie, po prostu przesyła dane bezpośrednio. Protokół UDP służy do przesyłania danych z większą szybkością. Jest mniej niezawodny i dlatego jest używany do przesyłania danych, takich jak pliki audio i wideo. UDP nie gwarantuje ani dostarczenia danych, ani nie przesyła utraconych pakietów.

## 1.2 Protokół IP.

rezerwuje - Rafał

## 1.3 Modele sieci komputerowych.

a

## 1.4 Porównanie protokołów IPv4 i IPv6.

a

## 1.5 Format pakietu IP (poszczególne pola, zastosowanie).

rezerwuje - Rafał

## 1.6 Ethernet.

a

## 1.7 Protokoły warstwy aplikacji.

a

## 1.8 Charakterystyka modelu OSI i TCP/IP.

rezerwuje - Rafał

## 1.9 Rodzaje i przykłady nagłówków HTTP.

a

### 1.10 Protokół WebSocket.

a

### 1.11 Serwer zdarzeniowy, a wielowątkowy. Charakterystyka i porównanie.

a

## 2 Bezpieka

### 2.1 Infrastruktura klucza publicznego - charakterystyka.

a

### 2.2 Kryptografia symetryczna oraz asymetryczna - charakterystyka.

#### Kryptografia symetryczna

W kryptografii symetrycznej szyfrowanie i deszyfrowanie wykonywane jest przy użyciu tego samego klucza. W niektórych algorytmach wykorzystywane są dwa klucze, jednak muszą one być od siebie zależne w taki sposób, że znając jeden z nich, można wygenerować drugi.



Rysunek I.3: Zasada działania kryptografii symetrycznej

W celu zapewnienia bezpiecznej komunikacji, algorytm szyfrowania musi być tak skonstruowany, żeby odtworzenie tekstu jawnego bez znajomości klucza było zadaniem trudnym obliczeniowo. Dodatkowym wymaganiem jest tajność klucza – przed rozpoczęciem wymiany wiadomości, należy opracować protokół uzgadniania lub przekazywania klucza.

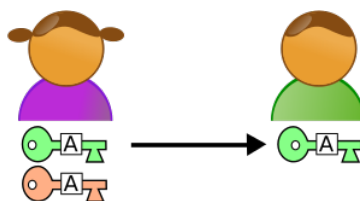


Algorytmy szyfrowania symetrycznego możemy podzielić na algorytmy blokowe i strumieniowe. Pierwsze z nich przekształcają blok danych ustalonej długości, traktując go jako całość, na szyfrogram o tej samej liczbie bitów. Szyfry strumieniowe przyjmują natomiast ciąg (strumień) danych. Algorytmy kryptografii symetrycznej są szybkie, zwykle wymagają też mniejszej mocy obliczeniowej niż algorytmy asymetryczne. Powszechnie stosowanym szyfrem symetrycznych jest **AES**.

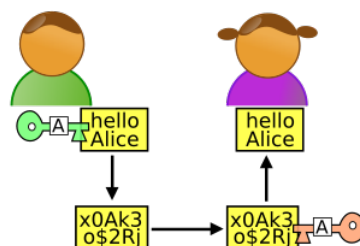
### Kryptografia asymetryczna

Kryptografia asymetryczna to rodzaj kryptografii, w którym jeden z używanych kluczy jest udostępniony publicznie. Każdy użytkownik może użyć tego klucza do zaszyfrowania wiadomości, ale tylko posiadacz drugiego, tajnego klucza może odszyfrować taką wiadomość.

Kryptografia asymetryczna opiera się na funkcjach jednokierunkowych – takich, które da się łatwo wyliczyć w jedną stronę, ale bardzo trudno w drugą. Np. mnożenie jest łatwe, a rozkład na czynniki (z ang. faktoryzacja) trudny (na czym przykładowo opiera się **RSA**). Potęgowanie modulo jest łatwe, a logarytmowanie dyskretne jest trudne (na czym opierają się ElGamal, DSA i **ECC**).



Rysunek I.4: Krok 1: Alice przesyła do Boba swój klucz publiczny



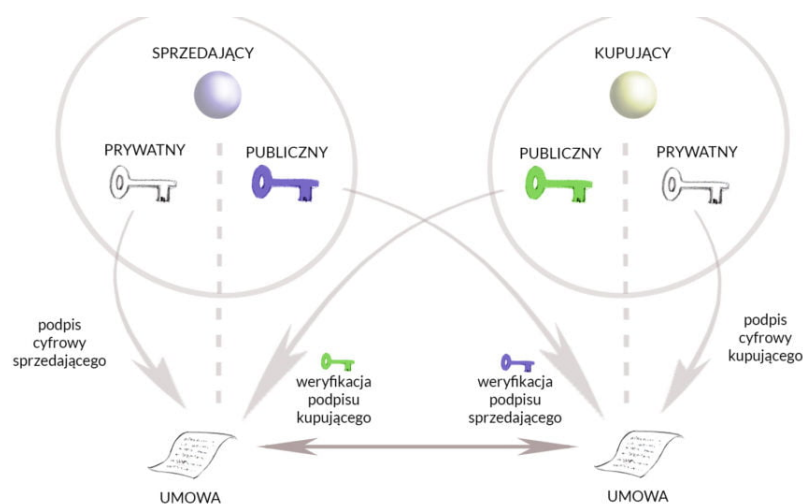
Rysunek I.5: Kroki 2 i 3: Bob szyfruje wiadomość kluczem publicznym Alice, która to następnie otrzymuje zaszyfrowaną wiadomość i rozszyfrowuje ją kluczem prywatnym

Klucz publiczny używany jest do zaszyfrowania informacji, klucz prywatny do jej odczytu. Ponieważ klucz prywatny jest w wyłącznym posiadaniu

adresata informacji, tylko on może ją odczytać. Natomiast klucz publiczny jest udostępniony każdemu, kto zechce zaszyfrować wiadomość.

Ponieważ kryptografia asymetryczna jest o wiele wolniejsza od symetrycznej, prawie nigdy nie szyfruje się wiadomości za pomocą kryptosystemów asymetrycznych (również ze względu na ograniczenie wielkości szyfrowanej wiadomości). Zamiast tego szyfruje się jedynie klucz jakiegoś szyfru symetrycznego, takiego jak np. AES. Takie protokoły, łączące elementy kryptografii symetrycznej i asymetrycznej, nazywa się hybrydowymi.

Nadawcy mogą także używać kluczy prywatnych do cyfrowego podpisywania wiadomości. Te podpisy cyfrowe pozwalają odbiorcom uwierzytelnić tożsamość nadawcy i spać spokojnie, wiedząc, że wiadomości nie zostały zmienione od momentu podpisania. W takim przypadku przesyłane informacje mogą być publiczne, a odbiorca może użyć certyfikatu, który towarzyszy tej informacji, aby zweryfikować integralność i autentyczność podpisanej wiadomości.



Rysunek I.6: Jak działa podpis

## 2.3 Bezpieczeństwo sieci w odniesieniu do warstw modelu TCP/IP.

a

## **2.4 Metody kontroli dostępu w systemach IT.**

a

## **2.5 Atrybuty bezpieczeństwa informacji.**

a

# Rozdział II

## Pytania - dr. hab. Grzegorz Wójcik

### 1 Bazy danych

#### 1.1 Model relacyjny baz danych i języki zapytań.

a

#### 1.2 Model obiektowo-relacyjny baz danych, inne modele danych.

a

#### 1.3 Składnia podstawowych zapytań języka SQL.

a

#### 1.4 Projektowanie baz danych oraz model związków encji.

a

#### 1.5 Problemy indeksowania baz danych, rodzaje indeksów, indeksy typu B+ drzewo.

a

## **1.6 Przetwarzanie transakcyjne OLTP (On-Line Transaction Processing).**

a

## **2 Paradygmaty**

### **2.1 Założenia paradygmatu programowania obiektowego.**

a

### **2.2 Idea dziedziczenia i polimorfizmu w programowaniu.**

a

### **2.3 Zasady programowania dynamicznego.**

a

### **2.4 Główne paradygmaty programowania.**

a

### **2.5 Cechy programowania deklaratywnego.**

a

# Rozdział III

## Pytania - reszta

### 1 Język C i C++

#### 1.1 Instrukcje sterujące w języku C.

a

#### 1.2 Zarządzanie pamięcią w języku C.

a

#### 1.3 Budowa, obsługa i formatowanie łańcuchów znakowych w języku C.

a

#### 1.4 Zasięg i czas życia obiektów w języku C++.

a

#### 1.5 Obsługa wyjątków w języku C++.

a

#### 1.6 Definicje obiektu, klasy i szablonu klasy w języku C++.

a

## 2 Algosy

### 2.1 Algorytmy sortujące.

a

### 2.2 Algorytmy zachłanne.

a

### 2.3 Metoda „dziel i zwyciężaj” konstruowania algorytmów.

a

### 2.4 Struktura kopców binarnych.

a

### 2.5 Algorytmy wyszukiwania najkrótszej ścieżki w grafie.

a

### 2.6 Sposoby implementacji słownika.

a

### 2.7 Tablice mieszające.

a

### 2.8 Algorytmy Monte Carlo oraz algorytmy Las Vegas.

a

### 2.9 Metody rozwiązywania rekurencji. Rekurencje Flawiusza i wieża w Hanoi.

a

**2.10 Algorytmy Euklidesa. Algorytmy faktoryzacji.**

a

**2.11 Metody reprezentacji grafów w komputerze.**

a

**2.12 Droga i cykl Eulera. Droga i cykl Hamiltona.**

a

**2.13 Drzewo spinające graf.**

a

**3 Teoria obliczalności czy coś**

**3.1 Pojęcia P, NP, NP-zupełne.**

a

**4 Automaty i inne takie**

**4.1 Deterministyczne i niedeterministyczne automaty skończone.**

a

**4.2 Automaty z epsilon przejściami, wyrażenia regularne.**

a

**4.3 Kompilacja: gramatyka bezkontekstowa, skaner, parser, błędy.**

a



## 5 Podstawy komputera i systemy operacyjne

### 5.1 Systemy liczbowe i konwersje pomiędzy nimi.

a

### 5.2 Sposoby cyfrowej reprezentacji liczby całkowitej i rzeczywistej.

#### Liczby całkowite

**Kod ZM (kod znak-moduł)** Sprawa w kodzie ZM jest w miarę prosta i klarowna. Najstarszy bit  $b_{n-1}$  dla  $n$ -bitowej liczby jest bitem znaku i określa czy liczba jest dodatnia czy ujemna:

- 0 - liczba dodatnia,
- 1 - liczba ujemna.

Bity od  $b_{n-1}$  do  $b_0$  odpowiadają za kodowanie wartości samej liczby. Wzór na obliczenie wartości liczby zakodowanej w **ZM**:

$$L_{ZM} = (-1)^{b_{n-1}} \cdot (b_{n-2}2^{n-2} + \dots + b_22^2 + b_12^1 + b_02^0)$$

Przykładowe kodowanie liczby na ośmiu bitach w kodzie **ZM**:

$$\begin{aligned} 26 &\longrightarrow 00011010 \\ -26 &\longrightarrow 10011010 \end{aligned}$$

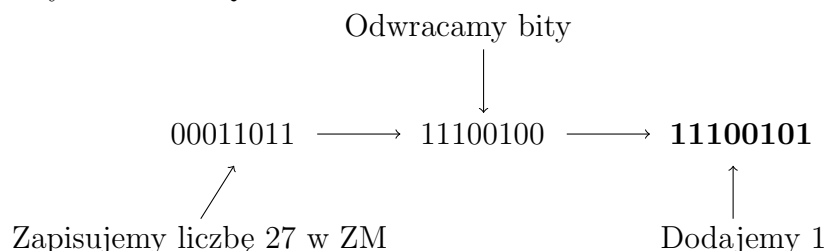
Proste, logiczne, fajne. Pytania, problemy? To jedziemy dalej.

**Kod U2 (kod uzupełnień do 2)** Tutaj sprawa się nieco komplikuje z zapisem liczb ujemnych. Bit  $b_{n-1}$  ma wagę  $-2^{n-1}$  co sprawia, że musimy bitowo tak jakby zapisać odwrotność liczby, którą chcemy reprezentować jako ujemna (i dodać 1, żeby się wszystko zgadzało). W zapisie liczb dodatnich zapis jest identyczny jak w **ZM** - na najstarszym bicie musimy tylko zachować 0.

Istnieje prosty algorytm konwersji na U2 z wykorzystaniem ZM:

1. Zapisać moduł liczby w ZM,
2. Dokonać inwersji bitów (0 na 1 i 1 na 0),
3. Zwiększ wynik dodając 1.

Przykład z liczbą -27 na 8 bitach:



### Liczby rzeczywiste

**Zapis stałopozycyjny** Do zapisu liczby stałoprzecinkowej przeznaczona jest z góry określona liczba bitów, a pozycję przecinka ustala się arbitralnie, w zależności od wymaganej dokładności, wolne bity uzupełniając zerami. Do reprezentacji liczb ze znakiem stosuje także kod U2.

Liczba  $6,25 = 110,01_{(2)}$  zapisana na 8 bitach gdy część ułamkowa zajmuje 3 najmłodsze bity, ma postać:

$$\begin{array}{c}
 \text{część ułamkowa} \\
 \underbrace{00110\overbrace{010}} \\
 \text{część całkowita}
 \end{array}$$

A w reprezentacji U2 będzie miała postać:

$$11001110$$

Część całkowita liczby zachowuje się identycznie jak w przypadku zwykłych liczb całkowitych, natomiast bity w części ułamkowej posiadają wagi  $2^{-1}$ ,  $2^{-2}$ , itd. - czyli  $\frac{1}{2}$ ,  $\frac{1}{4}$ , ..., więc ilość bitów w części ułamkowej wpływa na precyzję zapisu.

**Zapis zmiennopozycyjny** Liczba zmiennoprzecinkowa jest komputerową reprezentacją liczb rzeczywistych zapisanych w postaci wykładniczej o podstawie 2. Przykładowa notacja:

$$(-1)^Z \cdot M \cdot 2^C = (-1)^Z \cdot (1 + m) \cdot 2^{c-BIAS}$$

gdzie:

$(-1)^Z$  - znak liczby

$M = 1 + m$  - znormalizowana mantysa (liczba spełniająca warunek:  $1 \leq M \leq 2$ ). Ponieważ przed przecinkiem stoi zawsze 1, więc można ją przedstawić w postaci  $1 + m$ , gdzie  $m$  jest liczbą ułamkową:  $0 \leq m \leq 1$ )

$C = c - BIAS$  - cecha (liczba całkowita), która dzięki zastosowaniu stałej BIAS pozwoli przedstawić cechę w postaci różnicy  $c - BIAS$  ( $c$  jest liczbą całkowitą dodatnią, tzw spolaryzowana cechę)

$BIAS$  - stała (liczba całkowita BIAS zależna od danej implementacji – rozwiązuje problem znaku cechy)

Kodujemy wyłącznie:

**z** - bit znaku

**m** - mantysę pomniejszoną o 1

**c** - cechę przesuniętą o BIAS

Założmy, że operujemy następującym zmiennopozycyjnym formatem zapisu liczby rzeczywistej:

- na zapis przeznaczamy 16 bitów
- najstarszy bit ( $b_{15}$ ) to bit znaku (będziemy stosować kod ZM)
- kolejne 6 bitów ( $b_9 - b_{14}$ ) to mantysa
- pozostałe bity ( $b_0 - b_8$ ) są przeznaczone na zapis cechy i przyjmijmy, że  $BIAS = 9$

Przedstawimy liczbę  $+0,0224609375$  w powyższym formacie. Naszą liczbę zapisujemy w systemie binarnym w postaci wykładniczej o podstawie 2, przesuwamy przecinek zapisując ją w notacji wykładniczej:

$$0,0224609375 = 0,0000010111_{(2)} = 1,0111_{(2)} \cdot 2^{-6}$$

Z tego wynika, że:

- Znak:  $(-1)^0$
- Mantysa:  $1.\underline{0111}_2$
- Cecha:  $-6 = 3 - 9 = 11_2 - BIAS$

Oto liczba  $0,0224609375$  zapisana w zadanym formacie:

$$\underbrace{0}_{\text{bit znaku}} \overbrace{011100}^{\text{mantysa}} \underbrace{000000011}_{\text{cecha}}$$

**5.3 Wielowarstwowa organizacja oprogramowania komputera.**

a

**5.4 Procesy, zasoby i wątki.**

a

**5.5 Planowanie przydziału procesora, priorytety, wywłaszczanie oraz planowanie.**

a

**5.6 Zarządzanie pamięcią operacyjną.**

a

**5.7 Problem zakleszczenia, algorytm Bankiera.**

a

**6 Inżynieria Oprogramowania**

**6.1 Standardowe metodyki procesu wytwórczego oprogramowania.**

a

**6.2 Metodyki zwinne (agile).**

a

**6.3 Metody testowania oprogramowania.**

a

**6.4 Walidacja i weryfikacja oprogramowania.**

a

**6.5 Diagramy UML (przypadków użycia, klas, aktywności, sekwencji, stanów, obiektów, wdrożenia).**

a

**6.6 Wzorce projektowe programowania obiektowego.**

a

**6.7 Wzorce architektoniczne.**

a

**7 Systemy wbudowane i elektronika**

**7.1 Różnice pomiędzy obsługą zdarzeń w przerwaniach sprzętowych a obsługą zdarzeń w pętli programowej.**

a

**7.2 Stosowalność systemów opartych o mikrokontrolery vs stosowalność typowych komputerów (stacjonarnych i laptopów).**

a

**7.3 Dekoder, multiplekser i demultiplekser: budowa, zasada, działania, przeznaczenie/zastosowanie.**

a

**7.4 Podstawowe układy budujące system mikroprocesorowy i sposób wymiany informacji pomiędzy nimi.**

a