Informatyka 1

Politechnika Białostocka - Wydział Elektryczny

Elektrotechnika, semestr II, studia stacjonarne I stopnia

Rok akademicki 2018/2019

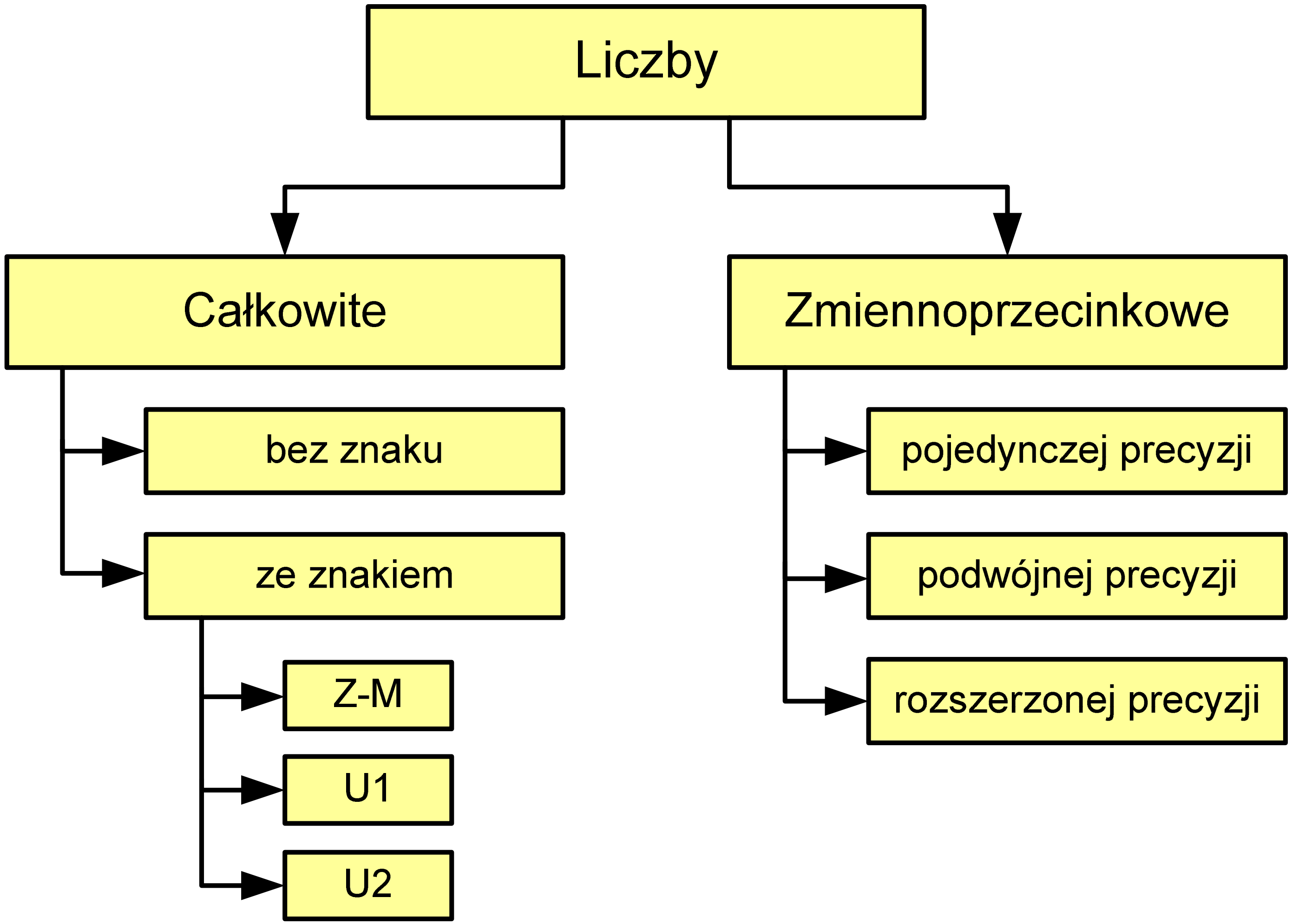
Wykład nr 8 (17.04.2019)

dr inż. Jarosław Forenc

Plan wykładu nr 8

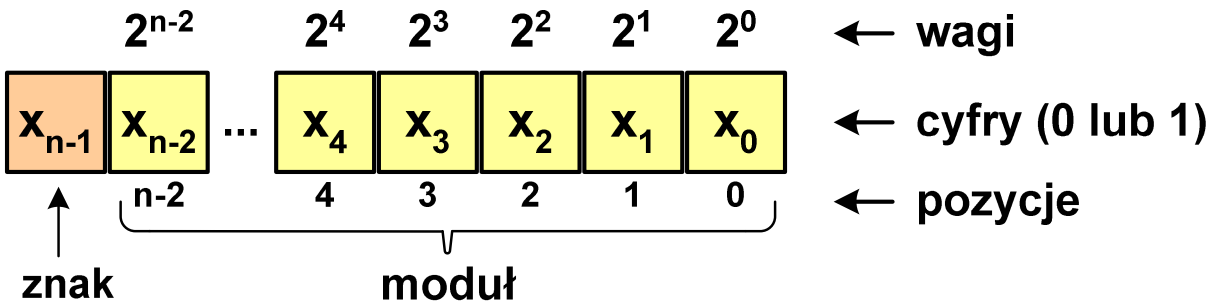
- Reprezentacja liczb całkowitych
  - liczby ze znakiem (ZM, U1, U2)
- Język C
  - pętle while i do…while
- Reprezentacja zmiennoprzecinkowa
  - zapis zmiennoprzecinkowy liczby rzeczywistej, postać znormalizowana
  - zakres liczb zmiennoprzecinkowych
- Standard IEEE 754
  - liczby 32-bitowe

Reprezentacja liczb w systemach komputerowych



Liczby całkowite ze znakiem - kod znak-moduł

- Inne nazwy: ZM, Z-M, SM (Signed Magnitude), S+M
- Najstarszy bit jest bitem znaku liczby: 0 - dodatnia, 1 - ujemna
- Pozostałe bity mają takie same znaczenie jak w NKB



- Wartość liczby:

$$X_{(10)} = \underbrace{(x_0 \cdot 2^0 + x_1 \cdot 2^1 + x_2 \cdot 2^2 + \ldots + x_{n-2} \cdot 2^{n-2})}_{\text{moduł}} \cdot \underbrace{(-1)^{x_{n-1}}}_{\text{znak}} = (-1)^{x_{n-1}} \cdot \sum_{i=0}^{n-2} x_i \cdot 2^i$$

Liczby całkowite ze znakiem - kod znak-moduł

- Liczby 4-bitowe (1 bit - znak, 3 bity - moduł) w kodzie Z-M:

Z-M	dziesiętnie	Z-M	dziesiętnie
0000	+0	1000	-0
0001	1	1001	-1
0010	2	1010	-2
0011	3	1011	-3
0100	4	1100	-4
0101	5	1101	-5
0110	6	1110	-6
0111	7	1111	-7

- dwie reprezentacje zera

$$+0\ (0000_{ZM})$$

$$-0\ (1000_{ZM})$$

- Zakres liczb dla n-bitów:

$$X_{(10)} = \left\langle -2^{n-1}+1,\ 2^{n-1}-1 \right\rangle$$

$$\text{dla 8 bitów}: \quad \left\langle -127 \ldots 127 \right\rangle$$

$$\text{dla 16 bitów}: \quad \left\langle -32767 \ldots 32767 \right\rangle$$

Liczby całkowite ze znakiem - kod znak-moduł

- Zamiana liczby dziesiętnej na kod Z-M:
  - liczba dodatnia

$$93_{(10)} = ?_{(ZM)}$$

  - zamieniamy liczbę na NKB

$$93_{(10)} = 1011101_{(NKB)}$$

  - dodajemy bit znaku

$$93_{(10)} = \boxed{0}1011101_{(ZM)}$$

  - liczba ujemna

$$-93_{(10)} = ?_{(ZM)}$$

  - zamieniamy moduł liczby na NKB

$$\left|-93_{(10)}\right| = 93_{(10)} = 1011101_{(NKB)}$$

  - dodajemy bit znaku

$$-93_{(10)} = \boxed{1}1011101_{(ZM)}$$

Liczby całkowite ze znakiem - kod U1

- Inne nazwy: U1, ZU1, uzupełnień do jedności
- Najstarszy bit jest bitem znaku liczby: 0 - dodatnia, 1 - ujemna
- Wszystkie bity liczby posiadają takie same wagi jak w NKB, oprócz pierwszego bitu, który ma wagę $-2^{n-1} + 1$

$-2^{n-1}+1$	2^{n-2}		2^4	2^3	2^2	2^1	2^0	← **wagi**
x_{n-1}	x_{n-2}	...	x_4	x_3	x_2	x_1	x_0	← **cyfry (0 lub 1)**
n-1	n-2		4	3	2	1	0	← **pozycje**
↑ **znak**								

- Wartość liczby:

$$X_{(10)} = x_0 \cdot 2^0 + x_1 \cdot 2^1 + x_2 \cdot 2^2 + \ldots + x_{n-2} \cdot 2^{n-2} + x_{n-1} \cdot (-2^{n-1} + 1)$$

Liczby całkowite ze znakiem - kod U1

- Liczby 4-bitowe (1 bit - znak, 3 bity - moduł) w kodzie U1:

U1	dziesiętnie
0000	+0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7

U1	dziesiętnie
1111	-0
1110	-1
1101	-2
1100	-3
1011	-4
1010	-5
1001	-6
1000	-7

- liczby <u>dodatnie</u> zapisywane są tak samo jak w NKB
- liczby <u>ujemne</u> otrzymywane są poprzez bitową negację
- dwie reprezentacje zera

- Zakres liczb dla n-bitów:

$$X_{(10)} = \left\langle -2^{n-1}+1,\ 2^{n-1}-1 \right\rangle$$

$$\text{dla 8 bitów}: \quad \left\langle -127\ ...\ 127 \right\rangle$$

$$\text{dla 16 bitów}: \quad \left\langle -32767\ ...\ 32767 \right\rangle$$

Liczby całkowite ze znakiem - kod U1

- Zamiana liczby dziesiętnej na kod U1:

  - liczba dodatnia

$$93_{(10)} = ?_{(U1)}$$

  - zamieniamy liczbę na NKB

$$93_{(10)} = 1011101_{(NKB)}$$

  - dodajemy bit znaku: 0

$$93_{(10)} = \boxed{0}1011101_{(U1)}$$

  - liczba ujemna

$$-93_{(10)} = ?_{(U1)}$$

  - zamieniamy moduł liczby na U1

$$\left|-93_{(10)}\right| = 93_{(10)} = 01011101_{(U1)}$$

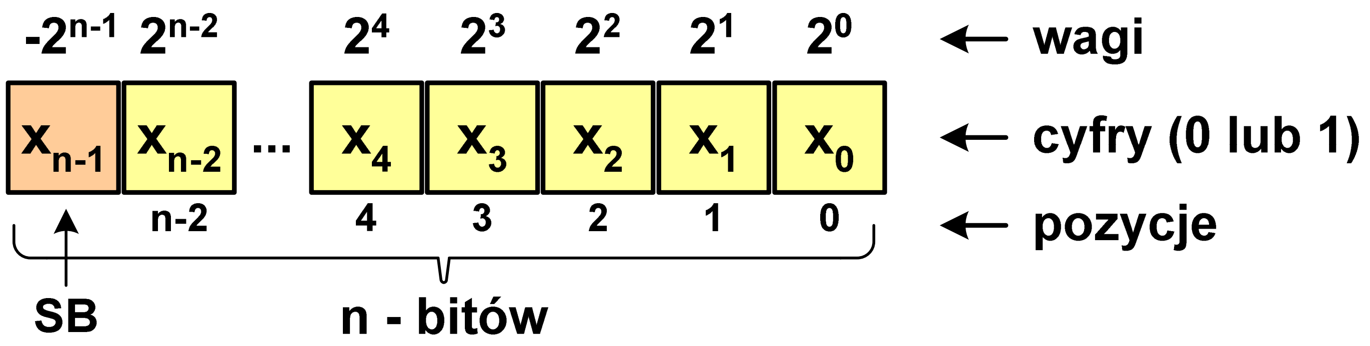
  - negujemy wszystkie bity

$$-93_{(10)} = 10100010_{(U1)}$$

bit znaku

Liczby całkowite ze znakiem - kod U2

- Inne nazwy: ZU2, uzupełnień do dwóch, two's complement
- Najstarszy bit jest bitem znaku liczby: 0 - dodatnia, 1 - ujemna



- Wartość liczby:

$$X_{(10)} = x_0 \cdot 2^0 + x_1 \cdot 2^1 + x_2 \cdot 2^2 + \ldots + x_{n-2} \cdot 2^{n-2} + x_{n-1} \cdot (-2^{n-1})$$

- Kod U2 jest obecnie powszechnie stosowany w informatyce

Liczby całkowite ze znakiem - kod U2

- Liczby 4-bitowe (1 bit - znak, 3 bity - moduł) w kodzie U2:

U2	dziesiętnie
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7

U2	dziesiętnie
1111	-1
1110	-2
1101	-3
1100	-4
1011	-5
1010	-6
1001	-7
1000	-8

- brak podwójnej reprezentacji zera
- liczb ujemnych jest o jeden więcej niż dodatnich
- 00...000 zawsze oznacza $0_{(10)}$
  11...111 zawsze oznacza $-1_{(10)}$

- Zakres liczb dla n-bitów:

$$X_{(10)} = \left\langle -2^{n-1},\ 2^{n-1} - 1 \right\rangle$$

$$\text{dla 8 bitów}: \quad \left\langle -128\ \ldots\ 127 \right\rangle$$

$$\text{dla 16 bitów}: \quad \left\langle -32768\ \ldots\ 32767 \right\rangle$$

Liczby całkowite ze znakiem - kod U2

- Zamiana liczby dziesiętnej na kod U2:
  - liczba dodatnia

$$75_{(10)} = ?_{(U2)}$$

  - zamieniamy liczbę na NKB

$$75_{(10)} = 1001011_{(NKB)}$$

  - dodajemy bit znaku: 0

$$75_{(10)} = \boxed{0}1001011_{(U2)}$$

  - liczba ujemna

$$-75_{(10)} = ?_{(U2)}$$

  - zamieniamy moduł liczby na U2

$$\left|-75_{(10)}\right| = 75_{(10)} = 01001011_{(U2)}$$

  - negujemy wszystkie bity i dodajemy 1

$$\begin{array}{rr} & 01001011 \\ \text{negacja}: & 10110100 \\ +1: & 1 \\ \hline -75_{(10)} = & 10110101_{(U2)} \end{array}$$

Liczby całkowite ze znakiem - kod U2 w języku C

- Typy zmiennych całkowitych ze znakiem stosowane w języku C:

Nazwa typu	Rozmiar (bajty)	Zakres wartości
char	1 bajt	-128 ... 127
short int	2 bajty	-32 768 ... 32 767
int	4 bajty	-2 147 483 648 ... 2 147 483 647
long int	4 bajty	-2 147 483 648 ... 2 147 483 647
long long int	8 bajtów	-9 223 372 036 854 775 808 ... 9 223 372 036 854 775 807

- Przed nazwą każdego z powyższych typów można dodać signed

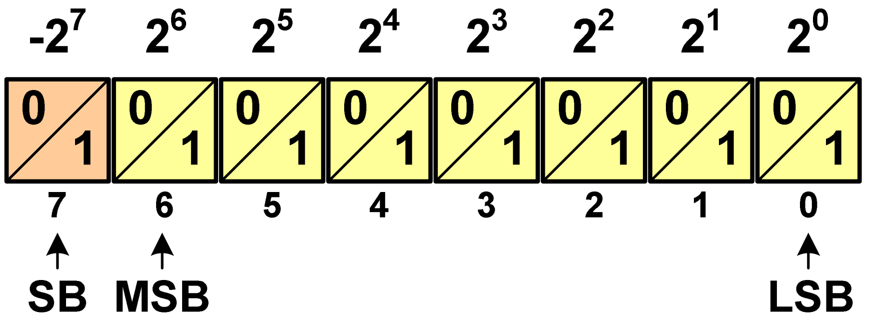
  signed char, signed short int, signed int ...

- W nazwach typów short i long można pominąć słowo int:

  short int → short, long int → long, long long int → long long

Liczby całkowite ze znakiem - kod U2 w języku C

- Typ char / signed char (1 bajt):

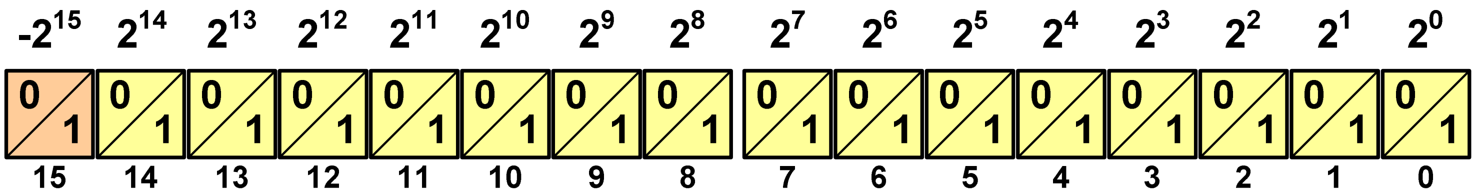


- Zakres wartości:
  - dolna granica: $1000\,0000_{(2)} = -128_{(10)}$
  - górna granica: $0111\,1111_{(2)} = 127_{(10)}$
  - inne wartości: $1111\,1111_{(2)} = -1_{(10)}$

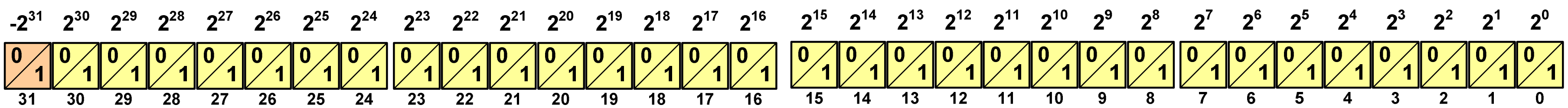
    $0000\,0000_{(2)} = 0_{(10)}$

Liczby całkowite bez znaku w języku C

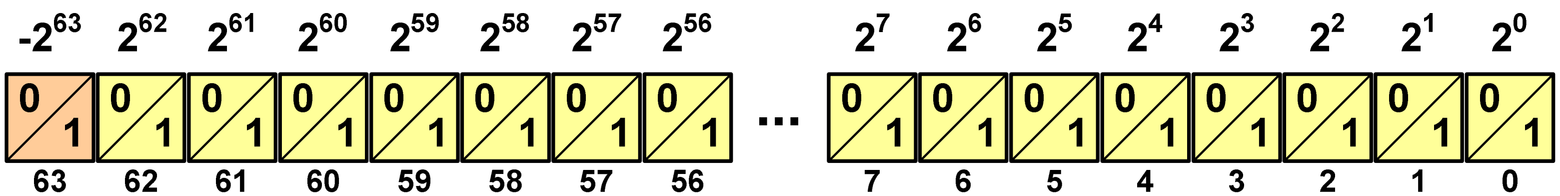
- Typ short / signed short int (2 bajty):



- Typy int / signed int (4 bajty) i long / signed long int (4 bajty):



- Typ long long int / signed long long int (8 bajtów):



Liczby całkowite ze znakiem - kod U2 w języku C

```
short int:     32767 -32768 -32767
int:           2147483647 -2147483648 -2147483647
long int:      2147483647 -2147483648 -2147483647
long long int: 9223372036854775807 -9223372036854775808
```

```
#includ

int main()   /* przepełnienie zmiennej, ang. integer overflow */
{
   short int     si = 32767;
   int            i = 2147483647;
   long int      li = 2147483647;
   long long int lli = 9223372036854775807;

   printf("short int:     %hd %hd %hd\n",si,si+1,si+2);
   printf("int:           %d %d %d\n",i,i+1,i+2);
   printf("long int:      %ld %ld %ld\n",li,li+1,li+2);
   printf("long long int: %lld %lld\n",lli,lli+1);

   return 0;
}
```

Język C - pierwiastek kwadratowy

```
#include <stdio.h>
#include <math.h>

int main(void)
{
    float x, y;

    printf("Podaj liczbe: ");
    scanf("%f",&x);

    if (x>=0)
    {
        y = sqrt(x);
        printf("Pierwiastek liczby: %f\n",y);
    }
    else
        printf("Blad! Liczba ujemna\n");

    return 0;
}
```

```
Podaj liczbe: -3
Blad! Liczba ujemna
```

```
Podaj liczbe: 3
Pierwiastek liczby: 1.732051
```

Język C - pierwiastek kwadratowy (pętla while)

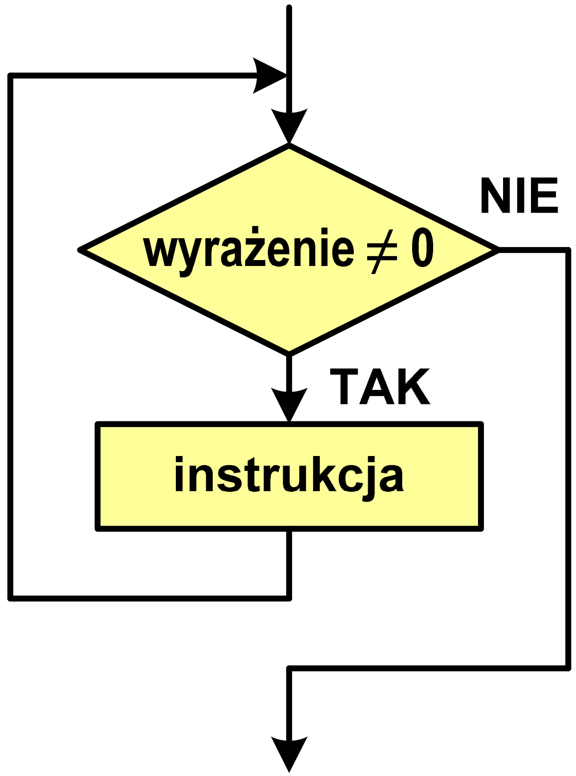
```c
#include <stdio.h>
#include <math.h>

int main(void)
{
    float x, y;

    printf("Podaj liczbe: ");
    scanf("%f",&x);
    while (x<0)
    {
        printf("Blad! Liczba ujemna\n\n");
        printf("Podaj liczbe: ");
        scanf("%f",&x);
    }
    y = sqrt(x);
    printf("Pierwiastek liczby: %f\n",y);

    return 0;
}
```

```
Podaj liczbe: -3
Blad! Liczba ujemna

Podaj liczbe: -5
Blad! Liczba ujemna

Podaj liczbe: 3
Pierwiastek liczby: 1.732051
```

Język C - pętla while

```
while (wyrażenie)
    instrukcja
```

- „dopóki wyrażenie w nawiasach jest prawdziwe wykonuj instrukcję”



- Wyrażenie w nawiasach:
  - prawdziwe - gdy jego wartość jest różna od zera
  - fałszywe - gdy jego wartość jest równa zero
- Jako wyrażenie najczęściej stosowane jest wyrażenie logiczne

Język C - pętla while

```
while (wyrażenie)
    instrukcja
```

- Instrukcja:
  - prosta - jedna instrukcja zakończona średnikiem
  - złożona - jedna lub kilka instrukcji objętych nawiasami klamrowymi

```
int x = 10;
while (x>0)
    x = x - 1;
```

```
int x = 10;
while (x>0)
{
    printf("%d\n",x);
    x = x - 1;
}
```

Język C - suma liczb dodatnich

```c
#include <stdio.h>
#include <math.h>

int main(void)
{
    int x, suma = 0;

    printf("Podaj liczbe: ");
    scanf("%d",&x);

    while(x>0)
    {
        suma = suma + x;
        printf("Podaj liczbe: ");
        scanf("%d",&x);
    }
    printf("Suma liczb: %d\n",suma);

    return 0;
}
```

```
Podaj liczbe: 4
Podaj liczbe: 8
Podaj liczbe: 2
Podaj liczbe: 3
Podaj liczbe: 5
Podaj liczbe: -2
Suma liczb: 22
```

Język C - pętla while

- Program pokazany na poprzednim slajdzie zawiera typowy schemat przetwarzania danych z wykorzystaniem pętli while

```
printf("Podaj liczbe: ");      // wczytanie danych
scanf("%d",&x);

while(x>0)
{
    suma = suma + x;           // operacje na danych

    printf("Podaj liczbe: ");  // wczytanie danych
    scanf("%d",&x);
}
```

- Dane mogą być wczytywane z klawiatury, pliku, itp.

Język C - pętla while (break, continue)

- break i continue są to instrukcje skoku

```
int x=0;

while (x<10)
{
    x++;
    if (x%2==0)
        continue;
    if (x%5==0)
        break;
    printf("%d\n",x);
}
```

- continue przerywa bieżącą iterację
- break przerywa wykonywanie pętli

Język C - pętla while (najczęstsze błędy)

- Postawienie średnika po wyrażeniu w nawiasach powoduje powstanie pętli nieskończonej - program zatrzymuje się na pętli

```
int x = 10;
while (x>0);
    printf("%d ",x--);
```

- Brak aktualizacji zmiennej powoduje także powstanie pętli nieskończonej - program wyświetla wielokrotnie tę samą wartość

```
int x = 10;
while (x>0)
    printf("%d ",x);
```

```
10 10 10 10 10 ...
```

Język C - pętla while (pętla nieskończona)

- W pewnych sytuacjach celowo stosuje się pętlę nieskończoną (np. w mikrokontrolerach)

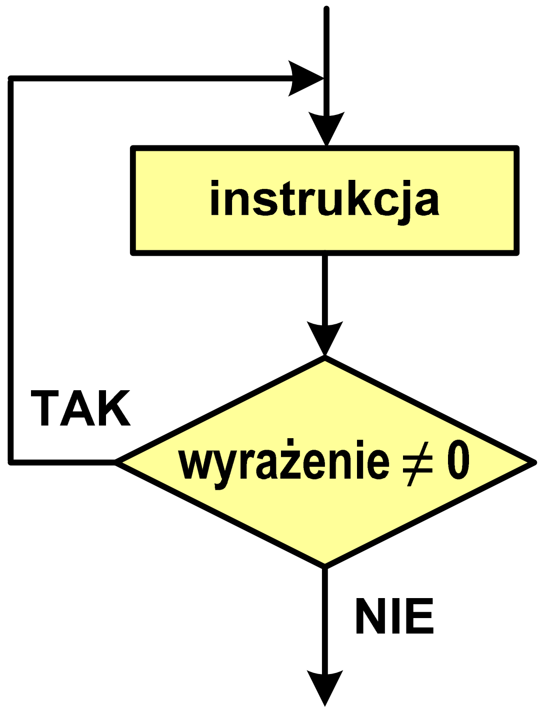
```
while (1)
{
    instrukcja
    instrukcja
    ...
}
```

- W układach mikroprocesorowych program działa aż do wyłączenia zasilania

Język C - pętla do ... while

```
do
    instrukcja
while (wyrażenie);
```

- „wykonuj instrukcję dopóki wyrażenie w nawiasach jest prawdziwe”



- Wyrażenie w nawiasach:
  - prawdziwe - gdy jego wartość jest różna od zera
  - fałszywe - gdy jego wartość jest równa zero

Język C - pętla do … while

```
do
    instrukcja
while (wyrażenie);
```

- Instrukcja:
    - prosta - jedna instrukcja zakończona średnikiem
    - złożona - jedna lub kilka instrukcji objętych nawiasami klamrowymi

```
int x = 10;
do
    x = x - 1;
while (x>0);
```

```
int x = 10;
do
{
    printf("%d\n",x);
    x = x - 1;
}
while (x>0);
```

Język C - pętla do ... while (break, continue)

- break i continue są to instrukcje skoku

```
int x=0;

do
{
    x++;
    if (x%5==0)
        break;
    if (x%2==0)
        continue;
    printf("%d\n",x);
}
while (i<10);
```

- break przerywa wykonywanie pętli
- continue przerywa bieżącą iterację

Zapis zmiennoprzecinkowy liczby rzeczywistej

- Zapis bardzo dużych lub małych liczb wymaga dużej liczby cyfr
- Znacznie prostsze jest przedstawienie liczb w postaci zmiennoprzecinkowej (ang. floating point numbers)
  - 12 000 000 000 000 = $1{,}2 \cdot 10^{13}$
  - 0,000 000 000 001 = $1{,}0 \cdot 10^{-12}$
- Zapis liczby zmiennoprzecinkowej ma postać:

$$L = M \cdot B^{E}$$

gdzie:

L - wartość liczby
M - mantysa
B - podstawa systemu
E - wykładnik, cecha

  - notacja naukowa: 1,2e13 1,2e+13 1,2E13 1,2E+13
  - postać wykładnicza: $1{,}2 \cdot 10^{13}$

Zapis zmiennoprzecinkowy liczby rzeczywistej

$$2{,}43\cdot 10^{3}{}_{(10)} = 2{,}43\cdot 1000 = 2430_{(10)}$$

$$6{,}59\cdot 10^{-2}{}_{(10)} = 6{,}59\cdot 0{,}01 = 0{,}0659_{(10)}$$

$$1{,}011\cdot 10^{101}{}_{(2)} = ?_{(10)}$$

$$M = 1{,}011_{(2)} = 1\cdot 2^{0} + 0\cdot 2^{-1} + 1\cdot 2^{-2} + 1\cdot 2^{-3} = 1{,}375_{(10)}$$

$$B = 10_{(2)} = 0\cdot 2^{0} + 1\cdot 2^{1} = 2_{(10)}$$

$$E = 101_{(2)} = 1\cdot 2^{0} + 0\cdot 2^{1} + 1\cdot 2^{2} = 1 + 4 = 5_{(10)}$$

$$1{,}011\cdot 10^{101}{}_{(2)} = 1{,}375\cdot 2^{5} = 1{,}375\cdot 32 = 44_{(10)}$$

$$3{,}121\cdot 10^{32}{}_{(4)} = ?_{(10)}$$

$$M = 3{,}121_{(4)} = 3\cdot 4^{0} + 1\cdot 4^{-1} + 2\cdot 4^{-2} + 1\cdot 4^{-3} = 3{,}390625_{(10)}$$

$$B = 10_{(4)} = 0\cdot 4^{0} + 1\cdot 4^{1} = 4_{(10)}$$

$$E = 32_{(4)} = 2\cdot 4^{0} + 3\cdot 4^{1} = 2 + 12 = 14_{(10)}$$

$$3{,}121\cdot 10^{32}{}_{(4)} = 3{,}390625\cdot 4^{14} = 910\ 163\ 968_{(10)}$$

Postać znormalizowana zapisu liczby

- Położenie przecinka w mantysie nie jest ustalone i może się zmieniać
- Poniższe zapisy oznaczają tę samą liczbę (system dziesiętny)

$$243 \cdot 10^1 = 24{,}3 \cdot 10^2 = 2{,}43 \cdot 10^3 = 0{,}243 \cdot 10^4$$

- Dla ujednolicenia zapisu i usunięcia wielokrotnych reprezentacji tej samej liczby, przyjęto tzw. postać znormalizowaną zapisu liczby
- W postaci znormalizowanej mantysa spełnia nierówność:

$$B > |M| \geq 1$$

Przykład:

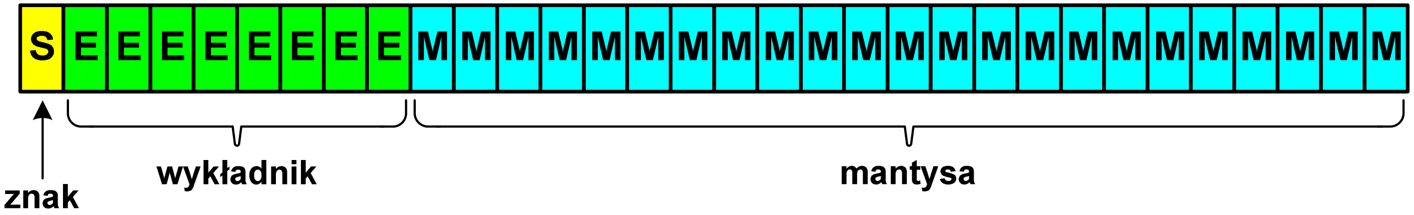
$2{,}43 \cdot 10^3$ - to jest postać znormalizowana, gdyż: $10 > |2{,}43| \geq 1$

$0{,}243 \cdot 10^4$ - to nie jest postać znormalizowana

$24{,}3 \cdot 10^2$ - to nie jest postać znormalizowana

Liczby zmiennoprzecinkowe w systemie binarnym

- Liczba bitów przeznaczonych na mantysę i wykładnik jest ograniczona



- Wartość liczby L:

$$L = (-1)^S \cdot M \cdot B^E$$

gdzie:

S - znak liczby (ang. sign), przyjmuje wartość 0 lub 1
M - znormalizowana mantysa (ang. mantissa), liczba ułamkowa
B - podstawa systemu liczbowego (ang. base)
E - wykładnik (ang. exponent), cecha, liczba całkowita

- W systemie binarnym podstawa systemu jest stała: B = 2

$$L = (-1)^S \cdot M \cdot 2^E$$

Przesunięcie wykładnika

- Wykładnik zapisywany jest z przesunięciem (ang. bias)

$$L = (-1)^{S} \cdot M \cdot 2^{E-BIAS}$$

gdzie:

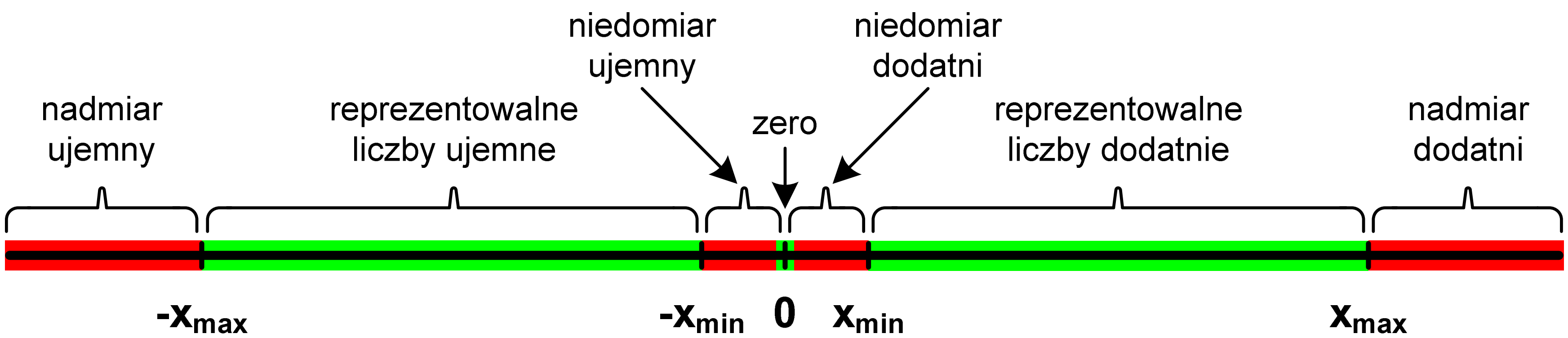
L - wartość liczby   S - znak liczby   M - mantysa
E - wykładnik   BIAS - przesunięcie (nadmiar)

- Typowe wartości przesunięcia (nadmiaru) wynoszą:
  - formatu 32-bitowy: 2^{7}-1 = $127_{(10)}$ = $7F_{(16)}$
  - formatu 64-bitowy: 2^{10}-1 = $1023_{(10)}$ = $3FF_{(16)}$
  - formatu 80-bitowy: 2^{14}-1 = $16383_{(10)}$ = $3FFF_{(16)}$

Zakres liczb zmiennoprzecinkowych

- Zakres liczb w zapisie zmiennoprzecinkowym:

$$\langle -x_{max}, -x_{min} \rangle \cup \{0\} \cup \langle x_{min}, x_{max} \rangle$$



- Największa i najmniejsza wartość liczby w danej reprezentacji:

$$x_{min} = M_{min} \cdot B^{E_{min}} \qquad x_{max} = M_{max} \cdot B^{E_{max}}$$

Standard IEEE 754

- IEEE Std. 754-2008 - IEEE Standard for Floating-Point Arithmetic
- Standard definiuje następujące klasy liczb zmiennoprzecinkowych:

Precyzja	Długość słowa [bity]	Znak [bity]	Wykładnik		Mantysa	
			Długość [bity]	Zakres	Długość [bity]	Cyfry znaczące
Pojedyncza (Single Precision, binary32)	32	1	8	$2^{\pm 127} \approx 10^{\pm 38}$	23	7
Pojedyncza rozszerzona (Single Extended)	≥ 43	1	≥ 11	$\geq 2^{\pm 1023} \approx 10^{\pm 308}$	≥ 31	≥ 10
Podwójna (Double Precision, binary64)	64	1	11	$2^{\pm 1023} \approx 10^{\pm 308}$	52	16
Podwójna rozszerzona (Double Extended)	≥ 79	1	≥ 15	$\geq 2^{\pm 16383} \approx 10^{\pm 4932}$	≥ 63	≥ 19

źródło: Gryś S.: „Arytmetyka komputerów w praktyce”. PWN, Warszawa, 2007 (str. 116).

Standard IEEE 754

- W przypadku liczb:
  - pojedynczej rozszerzonej precyzji (ang. Single Precision)
  - podwójnej rozszerzonej precyzji (ang. Double Precision)

  standard podaje jedynie minimalną liczbę bitów pozostawiając szczegóły implementacji producentom procesorów i kompilatorów

- Bardzo popularny jest 80-bitowy format podwójnej rozszerzonej precyzji (Extended Precision) wprowadzony przez firmę Intel

- W 80-bitowym formacie Intela:
  - długość słowa: 80 bitów
  - znak: 1 bit
  - wykładnik: 15 bitów (zakres: $2^{\pm 16383} \approx 10^{\pm 4932}$)
  - mantysa: 63 bity (cyfry znaczące: 19)

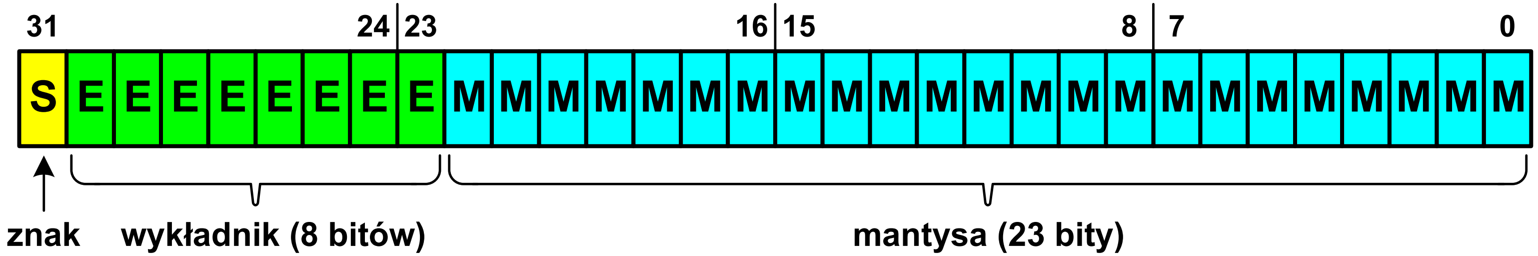
Standard IEEE 754

- Standard IEEE 754 definiuje także dziesiętne typy zmiennoprzecinkowe (operujące na cyfrach dziesiętnych):
  - decimal32 (32 bity, 7 cyfr dziesiętnych)
  - decimal64 (64 bity, 16 cyfr dziesiętnych)
  - decimal128 (128 bitów, 34 cyfry dziesiętnych)

- Standard IEEE 754 definiuje także:
  - sposób reprezentacji specjalnych wartości, np. nieskończoności, zera
  - sposób wykonywania działań na liczbach zmiennoprzecinkowych
  - sposób zaokrąglania liczb

Standard IEEE 754 - liczby 32-bitowe

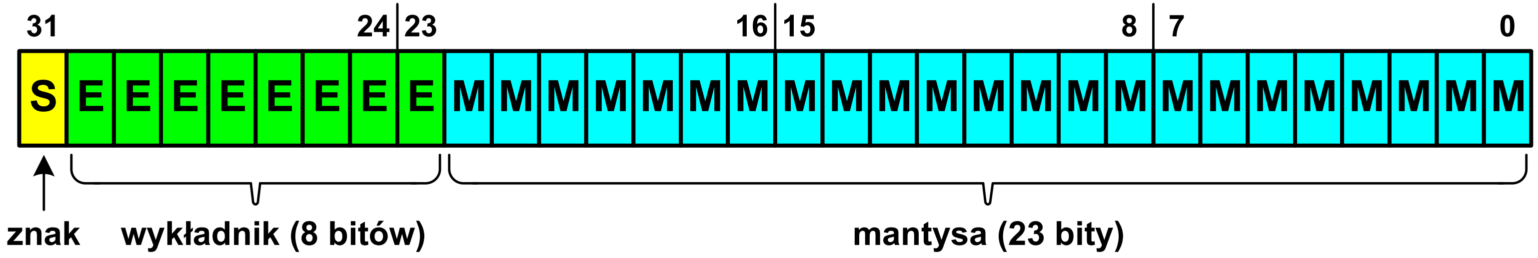
- Liczba pojedynczej precyzji przechowywana jest na 32 bitach:



- Pierwszy bit w zapisie (bit nr 31) jest bitem znaku (0 - liczba dodatnia, 1 - liczba ujemna)
- Wykładnik zapisywany jest na 8 bitach (bity nr 30-23) z nadmiarem o wartości 127
- Wykładnik może przyjmować wartości od -127 (wszystkie bity wyzerowane) do 128 (wszystkie bity ustawione na 1)

Standard IEEE 754 - liczby 32-bitowe

- Liczba pojedynczej precyzji przechowywana jest na 32 bitach:



- Mantysa w większości przypadków jest znormalizowana
- Wartość mantysy zawiera się pomiędzy 1 a 2, a zatem w zapisie liczby pierwszy bit jest zawsze równy 1
- Powyższy bit nie jest zapamiętywany, natomiast jest automatycznie uwzględniany podczas wykonywania obliczeń
- Dzięki pominięciu tego bitu zyskujemy dodatkowy bit mantysy (zamiast 23 bitów mamy 24 bity)

Standard IEEE 754 - liczby 32-bitowe

- Przykład:
  - obliczmy wartość dziesiętną liczby zmiennoprzecinkowej

$$01000010110010000000000000000000_{(IEEE754)} = ?_{(10)}$$

  - dzielimy liczbę na części

$$\underbrace{0}_{S-\text{bit znaku}} \quad \underbrace{10000101}_{E-\text{wykladnik}} \quad \underbrace{10010000000000000000000}_{M-\text{mantysa (tylko czesc ulamkowa)}}$$

  - określamy znak liczby

$$S = 0 \quad -\text{liczba dodatnia}$$

  - obliczamy wykładnik (nadmiar: 127)

$$10000101_{(2)} = 128 + 4 + 1 = 133 \quad \Rightarrow \quad E = 133 - \underbrace{127}_{nadmiar} = 6_{(10)}$$

Standard IEEE 754 - liczby 32-bitowe

- Przykład (cd.):
  - wyznaczamy mantysę dopisując na początku 1, (część całkowita)

$$M = 1{,}10010000000000000000000 =$$

$$= 1 \cdot 2^{0} + 1 \cdot 2^{-1} + 1 \cdot 2^{-4} = 1 + 0{,}5 + 0{,}0625 = 1{,}5625_{(10)}$$

  - wzór na wartość dziesiętną liczby zmiennoprzecinkowej:

$$L = (-1)^{S} \cdot M \cdot 2^{E}$$

  - podstawiając otrzymujemy:

$$S = 0, \qquad E = 6_{(10)}, \qquad M = 1{,}5625_{(10)}$$

$$L = (-1)^{0} \cdot 1{,}5625 \cdot 2^{6} = 100_{(10)}$$

$$01000010110010000000000000000000_{(\mathrm{IEEE754})} = 100_{(10)}$$

Koniec wykładu nr 8

Dziękuję za uwagę!