Informatyka 1

Politechnika Białostocka - Wydział Elektryczny

Elektrotechnika, semestr II, studia niestacjonarne I stopnia

Rok akademicki 2017/2018

Wykład nr 4/5 (13.04.2018)

dr inż. Jarosław Forenc

Plan wykładu nr 4/5

- Język C
  - pętla for, operatory ++ i --
- Reprezentacja zmiennoprzecinkowa
  - zapis zmiennoprzecinkowy liczby rzeczywistej, postać znormalizowana
  - zakres liczb zmiennoprzecinkowych
- Standard IEEE 754
  - liczby 32-bitowe i 64-bitowe, zakres i precyzja liczb
  - wartości specjalne, operacje z wartościami specjalnymi
- Klasyfikacja systemów komputerowych (Flynna)
- Architektura von Neumanna i architektura harwardzka
- Budowa komputera
  - jednostka centralna, płyta główna, procesory, moduły pamięci obudowa (AT, ATX)

Język C - suma kolejnych 10 liczb: 1+2+...+10

```c
#include <stdio.h>

int main(void)
{
    int suma;

    suma = 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10;

    printf("Suma wynosi: %d\n",suma);

    return 0;
}
```

```
Suma wynosi: 55
```

Język C - suma kolejnych 100 liczb: 1+2+...+100

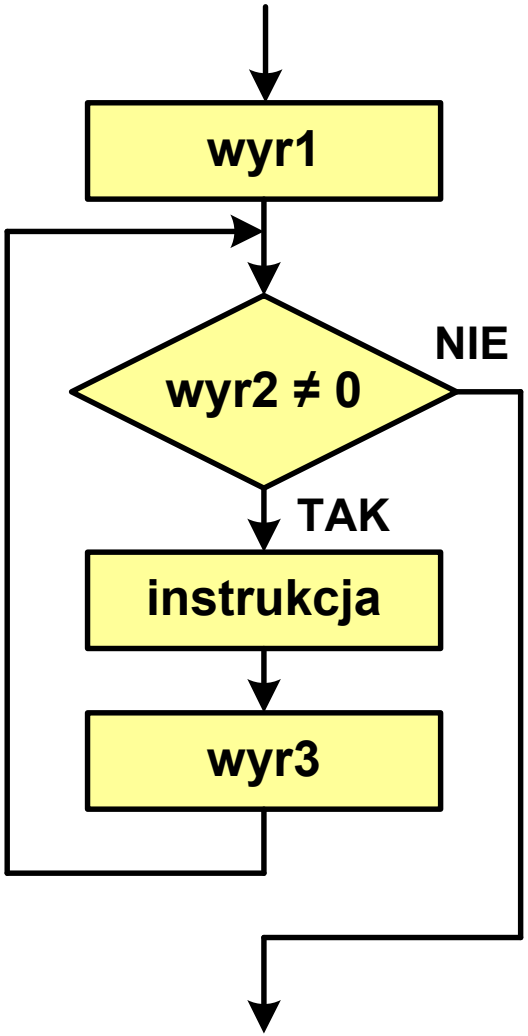
```c
#include <stdio.h>

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

    for (i=1; i<=100; i=i+1)
        suma = suma + i;

    printf("Suma wynosi: %d\n",suma);

    return 0;
}
```

```
Suma wynosi: 5050
```

Język C - pętla for

```
for (wyr1; wyr2; wyr3)
    instrukcja
```

- wyr1, wyr2, wyr3 - dowolne wyrażenia w języku C
- Instrukcja:
  - prosta - jedna instrukcja zakończona średnikiem
  - złożona - jedna lub kilka instrukcji objętych nawiasami klamrowymi



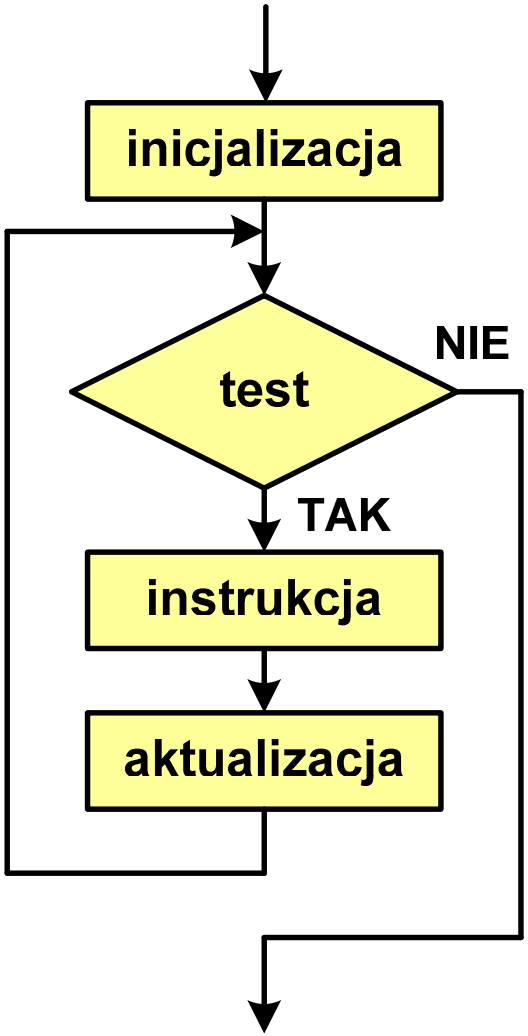
Język C - pętla for

- Najczęściej stosowana postać pętli for

```
int i;
for (i = 0; i < 10; i = i + 1)
    instrukcja
```

- Instrukcja zostanie wykonana 10 razy (dla i = 0, 1, 2, ... 9)
- Funkcje pełnione przez wyrażenia

```
for (inicjalizacja;test;aktualizacja)
    instrukcja
```



Język C - pętla for (wyświetlenie tekstu)

```c
#include <stdio.h>

int main(void)
{
    int i;

    for (i=0; i<5; i=i+1)
        printf("Programowanie nie jest trudne\n");

    return 0;
}
```

```
Programowanie nie jest trudne
Programowanie nie jest trudne
Programowanie nie jest trudne
Programowanie nie jest trudne
Programowanie nie jest trudne
```

Język C - pętla for (suma liczb: 1 + 2 + ... + N)

```
#include <stdio.h>
#define N 1234

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

    for (i=1; i<=N; i++)
        suma = suma + i;

    printf("Suma %d liczb to %d\n", N, suma);

    return 0;
}
```

```
Suma 1234 liczb to 761995
```

Język C - pętla for (przykłady)

```
for (i=0; i<10; i++)
    printf("%d ",i);
```

```
0 1 2 3 4 5 6 7 8 9
```

```
for (i=0; i<10; i++)
    printf("%d ",i+1);
```

```
1 2 3 4 5 6 7 8 9 10
```

```
for (i=1; i<=10; i++)
    printf("%d ",i);
```

```
1 2 3 4 5 6 7 8 9 10
```

Język C - pętla for (przykłady)

```
for (i=1; i<10; i=i+2)
    printf("%d ",i);
```

```
1 3 5 7 9
```

```
for (i=10; i>0; i--)
    printf("%d ",i);
```

```
10 9 8 7 6 5 4 3 2 1
```

```
for (i=-9; i<=9; i=i+3)
    printf("%d ",i);
```

```
-9 -6 -3 0 3 6 9
```

Język C - pętla for (break, continue)

- W pętli for można stosować instrukcje skoku: break i continue

```
int i;

for (i=1; i<10; i++)
{
    if (i%2==0)
        continue;
    if (i%7==0)
        break;
    printf("%d\n",i);
}
```

```
1 3 5
```

- continue przerywa bieżącą iterację i przechodzi do obliczania wyr3
- break przerywa wykonywanie pętli

Język C - pętla for (najczęstsze błędy)

- Postawienie średnika na końcu pętli for

```
int i;
for (i=0; i<10; i++);
    printf("%d ",i);
```

```
10
```

- Przecinki zamiast średników pomiędzy wyrażeniami

```
int i;
for (i=0, i<10, i++)
    printf("%d ",i);
```

Błąd kompilacji!

```
error C2143: syntax error : missing ';' before ')'
```

Język C - pętla for (najczęstsze błędy)

- Błędny warunek - brak wykonania instrukcji

```
int i;
for (i=0; i>10; i++)
    printf("%d ",i);
```

- Błędny warunek - pętla nieskończona

```
int i;
for (i=1; i>0; i++)
    printf("%d ",i);
```

```
1 2 3 4 5 6 7 8 9 ...
```

Język C - pętla nieskończona

```
for (wyr1; wyr2; wyr3)
    instrukcja
```

- Wszystkie wyrażenia (wyr1, wyr2, wyr3) w pętli for są opcjonalne

```
for ( ; ; )
    instrukcja
```

- pętla nieskończona

- W przypadku braku wyr2 przyjmuje się, że jest ono prawdziwe

Język C - zagnieżdżanie pętli for

- Jako instrukcja w pętli for może występować kolejna pętla for

```
int i, j;
for (i=1; i<=3; i++)          // pętla zewnętrzna
    for (j=1; j<=2; j++)      // pętla wewnętrzna
        printf("i: %d   j: %d\n",i,j);
```

```
i: 1   j: 1
i: 1   j: 2
i: 2   j: 1
i: 2   j: 2
i: 3   j: 1
i: 3   j: 2
```

Język C - operator inkrementacji (++)

- Jednoargumentowy operator ++ zwiększa wartość zmiennej o 1 (nie wolno stosować go do wyrażeń)
- Operator ++ może występować jako przedrostek lub przyrostek

Zapis	Nazwa	Znaczenie
`++x`	preinkrementacji	wartość zmiennej jest modyfikowana przed jej użyciem
`x++`	postinkrementacji	wartość zmiennej jest modyfikowana po użyciu jej poprzedniej wartości

Język C - operator inkrementacji (++)

- Przykład

```
int x = 1, y;
y = 2 * ++x;
```

```
int x = 1, y;
y = 2 * x++;
```

- Kolejność operacji

```
++x                 x = 2
2 * ++x             2 * 2
y = 2 * ++x         y = 4
```

```
2 * x               2 * 1
y = 2 * x           y = 2
x++                 x = 2
```

- Wartości zmiennych

```
x = 2     y = 4
```

```
x = 2     y = 2
```

Język C - operator inkrementacji (++)

- Miejsce umieszczenia operatora ++ nie ma znaczenia w przypadku instrukcji typu:

```
x++;
++x;
```

równoważne

```
x = x + 1;
```

- Nie należy stosować operatora ++ do zmiennych pojawiających się w wyrażeniu więcej niż jeden raz

```
x = x++;
x = ++x;
```

- Zgodnie ze standardem języka C wynik powyższych instrukcji jest niezdefiniowany

Język C - operator dekrementacji (--)

- Jednoargumentowy operator -- zmniejsza wartość zmiennej o 1 (nie wolno stosować go do wyrażeń)
- Operator -- może występować jako przedrostek lub przyrostek

Zapis	Nazwa	Znaczenie
`--x`	predekrementacji	wartość zmiennej jest modyfikowana przed jej użyciem
`x--`	postdekrementacji	wartość zmiennej jest modyfikowana po użyciu jej poprzedniej wartości

Język C - priorytet operatorów ++ i --

Priorytet	Operator / opis
1	`++` `--` (przyrostki) `()` `[]` `.` `->`
2	`++` `--` (przedrostki) `sizeof` `(typ)` `+` `-` `!` `~` `*` `&` (jednoargumentowe)
3	`*` `/` `%`
4	`+` `-` (dwuargumentowe)
5	`<<` `>>`
6	`<` `>` `<=` `>=`
7	`==` `!=`
8	`&` (bitowy)
9	`^`

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:

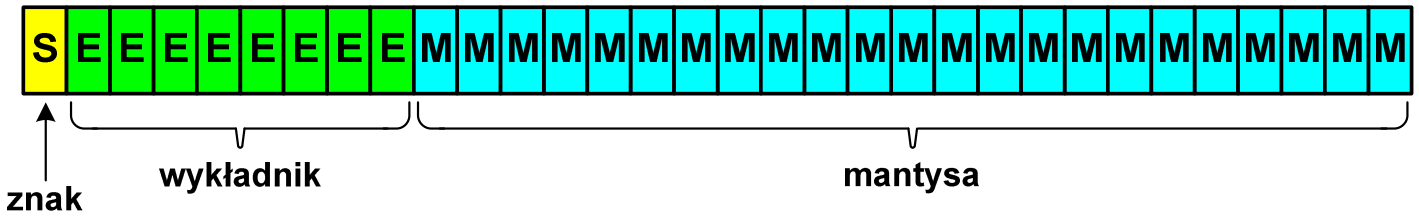
$2{,}43 \cdot 10^3$ - to jest postać znormalizowana, gdyż: $10 > |2{,}43| \geq 1$

$0{,}243 \cdot 10^4$ - 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:

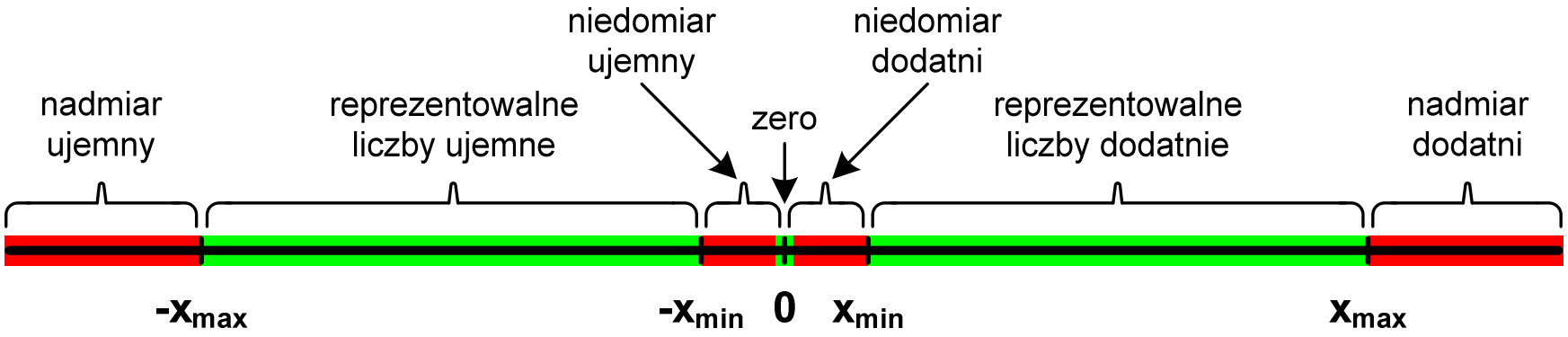
L - wartość liczby S - znak liczby M - mantysa
E - wykładnik BIAS - przesunięcie (nadmiar)

- 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)

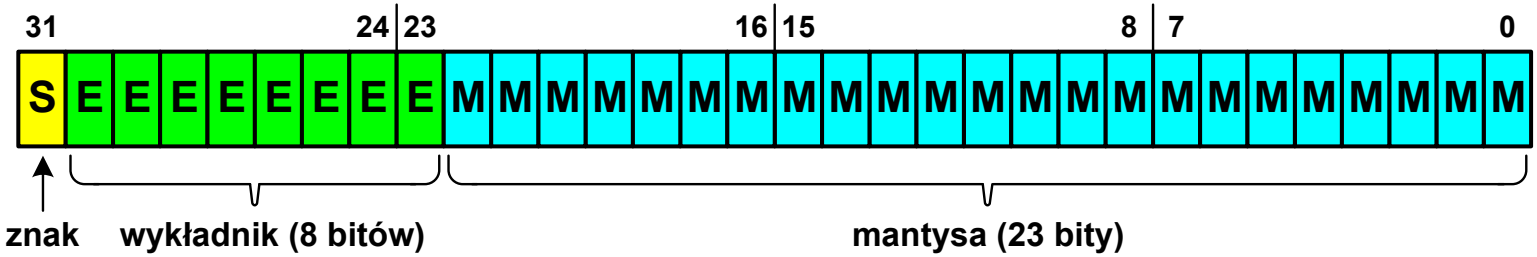
Standard IEEE 754

- 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

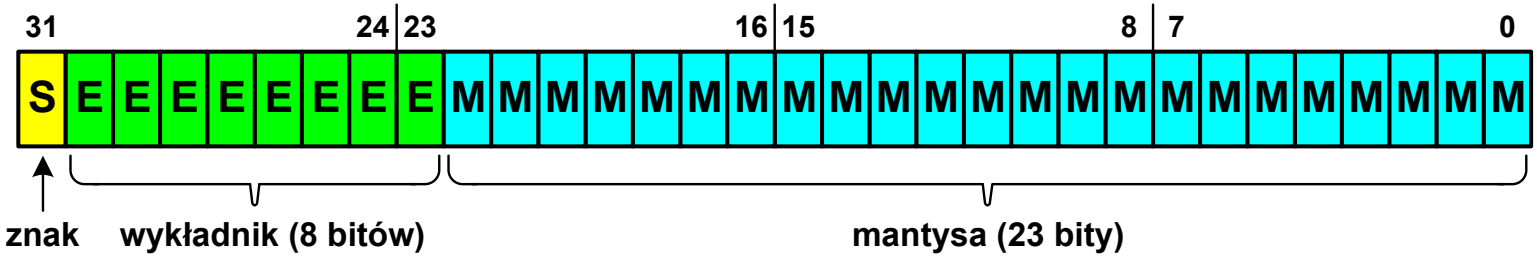
- Liczba pojedynczej precyzji przechowywana jest na 32 bitach:



- 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

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

  - dzielimy liczbę na części

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

  - określamy znak liczby

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

  - obliczamy wykładnik (nadmiar: 127)

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

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:

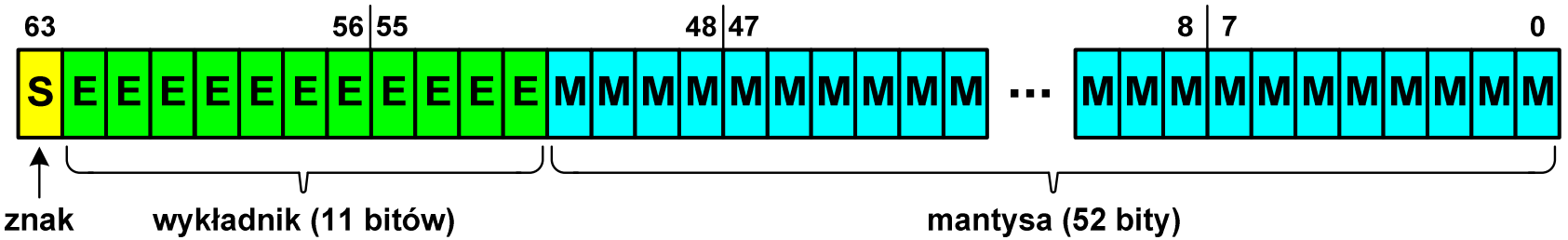
$$S = 0, \qquad E = 6_{(10)}, \qquad M = 1,5625_{(10)}$$

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

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

Standard IEEE 754 - liczby 64-bitowe

- Liczba podwójnej precyzji przechowywana jest na 64 bitach:



- Pierwszy bit w zapisie (bit nr 63) jest bitem znaku (0 - liczba dodatnia, 1 - liczba ujemna)
- Wykładnik zapisywany jest na 11 bitach (bity nr 62-52) z nadmiarem o wartości 1023
- Wykładnik może przyjmować wartości od -1023 (wszystkie bity wyzerowane) do 1024 (wszystkie bity ustawione na 1)
- Mantysa zapisywana jest na 52 bitach (pierwszy bit mantysy, zawsze równy 1, nie jest zapamiętywany)

Standard IEEE 754 - zakres liczb

- Pojedyncza precyzja:
  - największa wartość: $\approx 3{,}4 \cdot 10^{38}$
  - najmniejsza wartość: $\approx 1{,}4 \cdot 10^{-45}$
  - zakres liczb: $<-3{,}4 \cdot 10^{38} \ldots -1{,}4 \cdot 10^{-45}> \cup \{0\} \cup <1{,}4 \cdot 10^{-45} \ldots 3{,}4 \cdot 10^{38}>$
- Podwójna precyzja:
  - największa wartość: $\approx 1{,}8 \cdot 10^{308}$
  - najmniejsza wartość: $\approx 4{,}9 \cdot 10^{-324}$
  - zakres liczb: $<-1{,}8 \cdot 10^{308} \ldots -4{,}9 \cdot 10^{-324}> \cup \{0\} \cup <4{,}9 \cdot 10^{-324} \ldots 1{,}8 \cdot 10^{308}>$
- Podwójna rozszerzona precyzja:
  - największa wartość: $\approx 1{,}2 \cdot 10^{4932}$
  - najmniejsza wartość: $\approx 3{,}6 \cdot 10^{-4951}$
  - zakres liczb: $<-1{,}2 \cdot 10^{4932} \ldots -3{,}6 \cdot 10^{-4951}> \cup \{0\} \cup <3{,}6 \cdot 10^{-4951} \ldots 1{,}2 \cdot 10^{4932}>$

Standard IEEE 754 - precyzja liczb

- Precyzja - liczba zapamiętywanych cyfr znaczących w systemie (10)

  4,86452137846 → 4,864521 - 7 cyfr znaczących

- Precyzja liczby zależy od liczby bitów mantysy
- Liczba bitów potrzebnych do zakodowania 1 cyfry dziesiętnej:

$$10^1 = 2^n \quad \rightarrow \quad n = \log_2(10) \approx 3{,}321928$$

- Liczba cyfr dziesiętnych (d) możliwa do zakodowania na m bitach:

  $\log_2(10)$ bitów - 1 cyfra dziesiętna
  m bitów - d cyfr dziesiętnych

$$d = \frac{m}{\log_2(10)}$$

Standard IEEE 754 - precyzja liczb

- Dla formatu pojedynczej precyzji:
  - mantysa: 23 + 1 = 24 bity
  - cyfry znaczące: 7

$$d = \frac{24}{\log_2(10)} = \frac{24}{3{,}321928} = 7{,}2247 \approx 7$$

- Dla formatu podwójnej precyzji:
  - mantysa: 52 + 1 = 53 bity
  - cyfry znaczące: 16

$$d = \frac{53}{\log_2(10)} = \frac{53}{3{,}321928} = 15{,}9546 \approx 16$$

- Dla formatu podwójnej rozszerzonej precyzji:
  - mantysa: 63 + 1 = 64 bity
  - cyfry znaczące: 19

$$d = \frac{64}{\log_2(10)} = \frac{64}{3{,}321928} = 19{,}2659 \approx 19$$

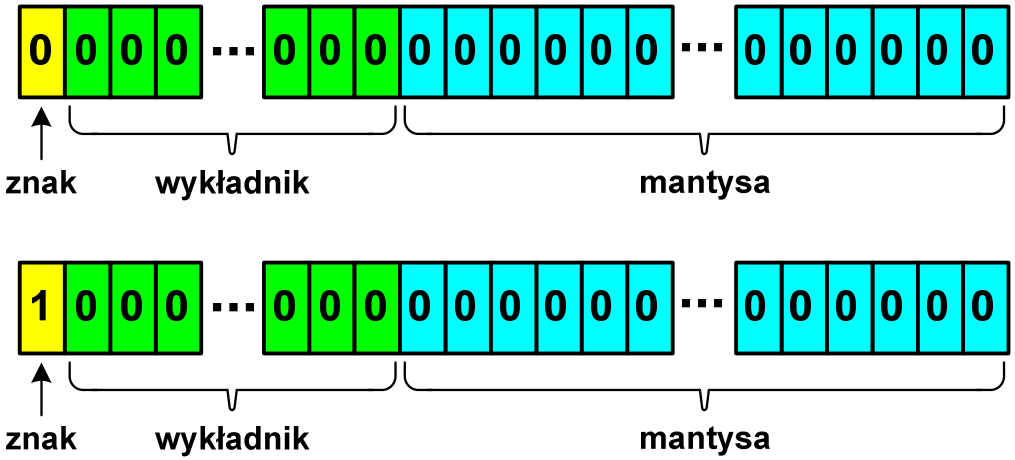
Standard IEEE 754 - precyzja liczb

```
#include <stdio.h>

int main()
{
    float x;
    double y;

    x = 1234567890.0; /* 1.234.567.890 */
    y = 1234567890.0; /* 1.234.567.890 */
    printf("float  -> %f\n",x);
    printf("double -> %f\n\n",y);

    y = 12345678901234567890.0;
    printf("double -> %f\n",y);

    return 0;
}
```

```
float  -> 1234567936.000000
double -> 1234567890.000000

double -> 12345678901234567000.000000
```

Standard IEEE 754 - wartości specjalne

- Zero:



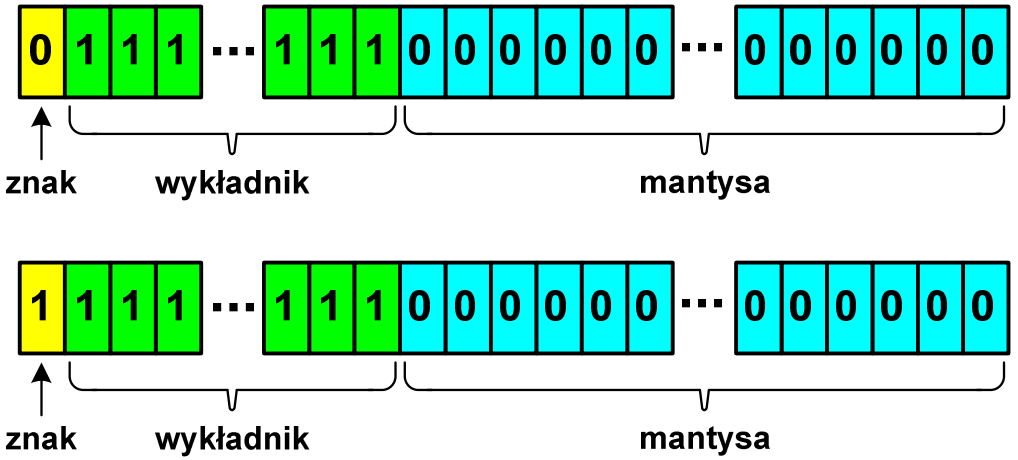
- zero dodatnie

- zero ujemne

- Podczas porównań zero dodatnie i ujemne są traktowane jako równe sobie

Standard IEEE 754 - wartości specjalne

- Nieskończoność:



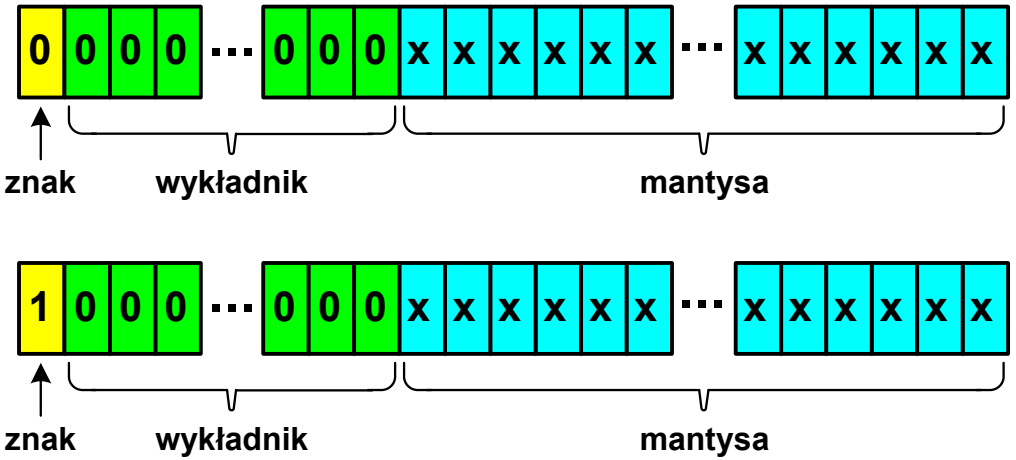
- nieskończoność dodatnia

- nieskończoność ujemna

- Nieskończoność występuje w przypadku wystąpienia nadmiaru (przepełnienia) oraz przy dzieleniu przez zero

Standard IEEE 754 - wartości specjalne

- Liczba zdenormalizowana:

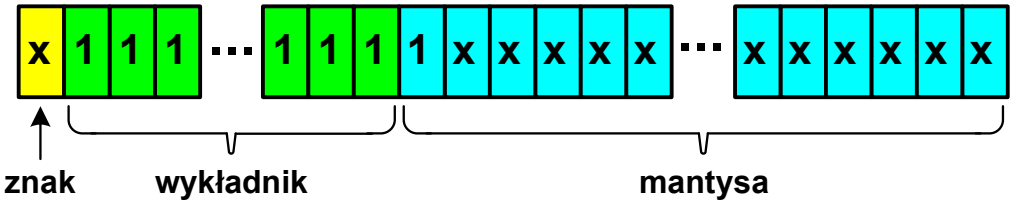


- Pojawia się, gdy występuje niedomiar (ang. underflow), ale wynik operacji można jeszcze zapisać denormalizując mantysę
- Mantysa nie posiada domyślnej części całkowitej równej 1, tzn. reprezentuje liczbę o postaci 0,xxx…xxx, a nie 1,xxx…xxx

Standard IEEE 754 - wartości specjalne

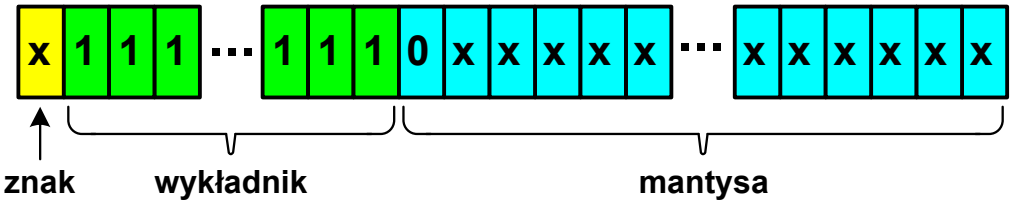
- Nieliczby - NaN (Not A Number) - nie reprezentują wartości liczbowej
- Powstają w wyniku wykonania niedozwolonej operacji

- QNaN (ang. Quiet NaN) - ciche nieliczby



- „przechodzą” przez działania arytmetyczne (brak przerwania wykonywania programu)

- SNaN (ang. Signaling NaN) - sygnalizujące, istotne, głośne nieliczby



- zgłoszenie wyjątku (przerwanie wykonywania programu)

Standard IEEE 754 - wartości specjalne

- Standard IEEE 754 definiuje dokładnie wyniki operacji, w których występują specjalne argumenty

Operacja	Wynik
x / ±∞	0
±∞ · ±∞	±∞
±wart_niezer / 0	±∞
∞ + ∞	∞
±0 / ±0	NaN
∞ - ∞	NaN
±∞ / ±∞	NaN
±∞ · 0	NaN

Język C - operacje z wartościami specjalnymi

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

int main()
{
  float x = 0.0;
  printf("1.0/0.0     = %f\n",1.0/x);
  printf("-1.0/0.0    = %f\n",-1.0/x);
  printf("0.0/0.0     = %f\n",0.0/x);
  printf("sqrt(-1.0)  = %f\n",sqrt(-1.0));
  printf("1.0/INF     = %f\n",1.0/(1.0/x));
  printf("0*INF       = %f\n",0.0*(1.0/x));

  return 0;
}
```

```
1.0/0.0     =  1.#INF00
-1.0/0.0    = -1.#INF00
0.0/0.0     = -1.#IND00
sqrt(-1.0)  = -1.#IND00
1.0/INF     =  0.000000
0*INF       = -1.#IND00
```

Operacja	Wynik
$x / \pm\infty$	0
$\pm\infty \cdot \pm\infty$	$\pm\infty$
$\pm$wart_niezer / 0	$\pm\infty$
$\infty + \infty$	∞
$\pm 0 / \pm 0$	NaN
$\infty - \infty$	NaN
$\pm\infty / \pm\infty$	NaN
$\pm\infty \cdot 0$	NaN

- Środowisko: Microsoft Visual C++ 2008 Express Edition

Reprezentacja liczb zmiennoprzecinkowych w C

- Typy zmiennoprzecinkowe w języku C:

Nazwa typu	Rozmiar (bajty)	Zakres wartości	Cyfry znaczące
float	4 bajty	$-3{,}4\cdot10^{38}$ ... $3{,}4\cdot10^{38}$	7-8
double	8 bajtów	$-1{,}8\cdot10^{308}$ ... $1{,}8\cdot10^{308}$	15-16
long double	10 bajtów	$-1{,}2\cdot10^{4932}$ ... $1{,}2\cdot10^{4932}$	19-20

- Typ long double może mieć także inny rozmiar:

Środowisko	Rozmiar (bajty)
MS Visual C++ 2008 EE	8 bajtów
Borland Turbo C++ Explorer	10 bajtów
Dev-C++	12 bajtów

Reprezentacja liczb zmiennoprzecinkowych w C

```
#include <stdio.h>

int main()
{
    float       sf  = 0.0f;
    double      sd  = 0.0;
    long double slg = 0.0L;
    int i;

    for(i=0; i<10000; i++)
    {
        sf  = sf  + 0.01f;
        sd  = sd  + 0.01;
        slg = slg + 0.01L;
    }

    printf("float:       %.20f\n",sf);
    printf("double:      %.20f\n",sd);
    printf("long double: %.20Lf\n",slg);

    return 0;
}
```

Reprezentacja liczb zmiennoprzecinkowych w C

- Microsoft Visual C++ 2008 Express Edition (long double - 8 bajtów)

```
float:       100.00295257568359000000
double:      100.00000000001425000000
long double: 100.00000000001425000000
```

- Borland Turbo C++ Explorer (long double - 10 bajtów)

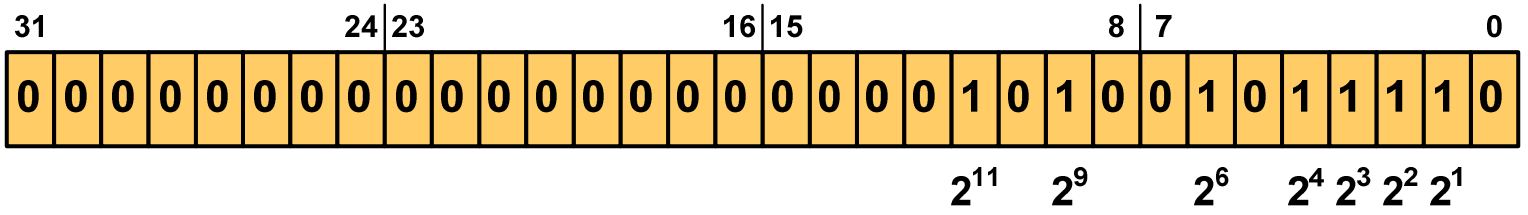
```
float:       100.00295257568359375000
double:      100.00000000001425349000
long double: 100.00000000000001388000
```

- Dev-C++ (long double - 12 bajtów)

```
float:       100.00295257568359000000
double:      100.00000000001425000000
long double: -6805647338419354100000000000000000000.0000000000000
```

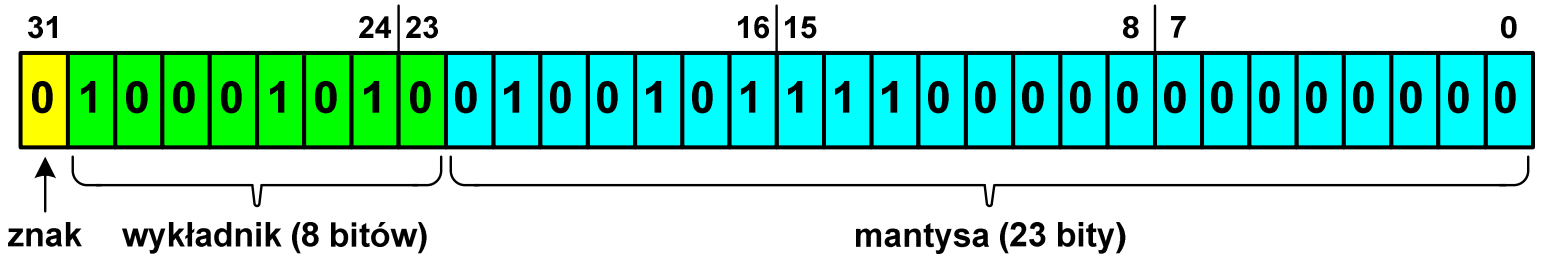
Liczba $2654_{(10)}$ jako całkowita i rzeczywista w C

- int (4 bajty): $2654_{(10)} = 00\ 00\ 0A\ 5E_{(16)}$



$$2^{11} + 2^9 + 2^6 + 2^4 + 2^3 + 2^2 + 2^1 = 2048 + 512 + 64 + 16 + 8 + 4 + 2 = 2654_{(10)}$$

- float (4 bajty): $2654_{(10)} = 45\ 25\ E0\ 00_{(IEEE\ 754)}$



$+ \quad 138 - 127 = 11_{(10)}$ $\qquad 1\ .0100101111_{(2)} = 1.2958984_{(10)}$

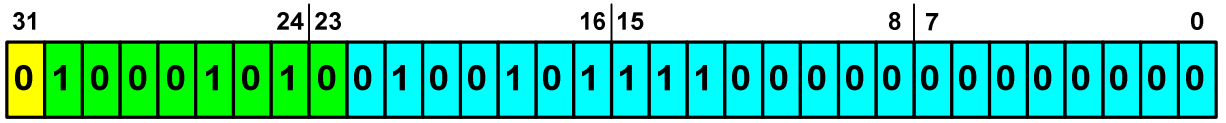
$$1.2958984 \cdot 2^{11} = 2654_{(10)}$$

Język C - nieprawidłowy specyfikator formatu

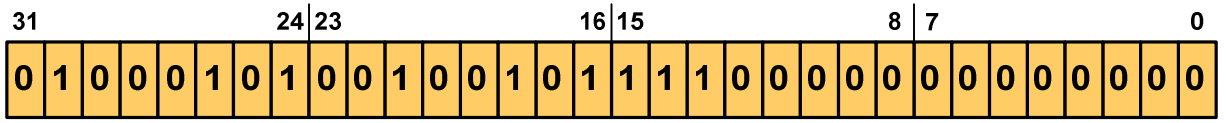
```
int x;
printf("x (%%f) = "); scanf("%f",&x);
printf("x (%%d) = %d\n",x);
printf("x (%%f) = %f\n",x);
printf("x (%%e) = %e\n",x);
```

```
x (%f) = 2654
x (%d) = 1160110080
x (%f) = 0.000000
x (%e) = 5.731705e-315
```

- Zgodnie ze standardem języka C wynik jest niezdefiniowany
- Zapamiętana wartość:



- Wyświetlona wartość przy wykorzystaniu %d:



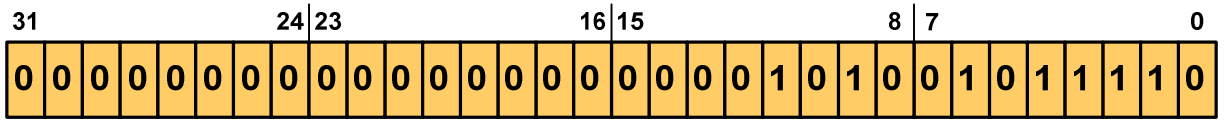
$$2^{30} + 2^{26} + 2^{24} + 2^{21} + 2^{18} + 2^{16} + 2^{15} + 2^{14} + 2^{13} = 1.160.110.080_{(10)}$$

Język C - nieprawidłowy specyfikator formatu

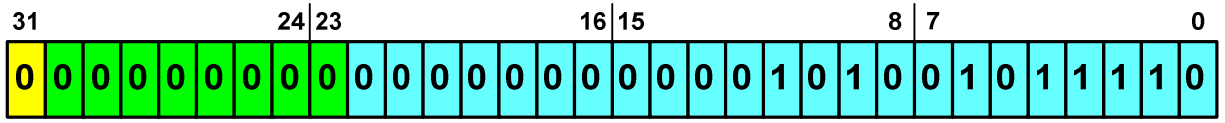
```
float x;
printf("x (%%d) = "); scanf("%d",&x);
printf("x (%%d) = %d\n",x);
printf("x (%%f) = %f\n",x);
printf("x (%%e) = %e\n",x);
```

```
x (%d) = 2654
x (%d) = 0
x (%f) = 0.000000
x (%e) = 3.719046e-042
```

- Zgodnie ze standardem języka C wynik jest niezdefiniowany
- Zapamiętana wartość:



- Wyświetlona wartość przy wykorzystaniu %e:

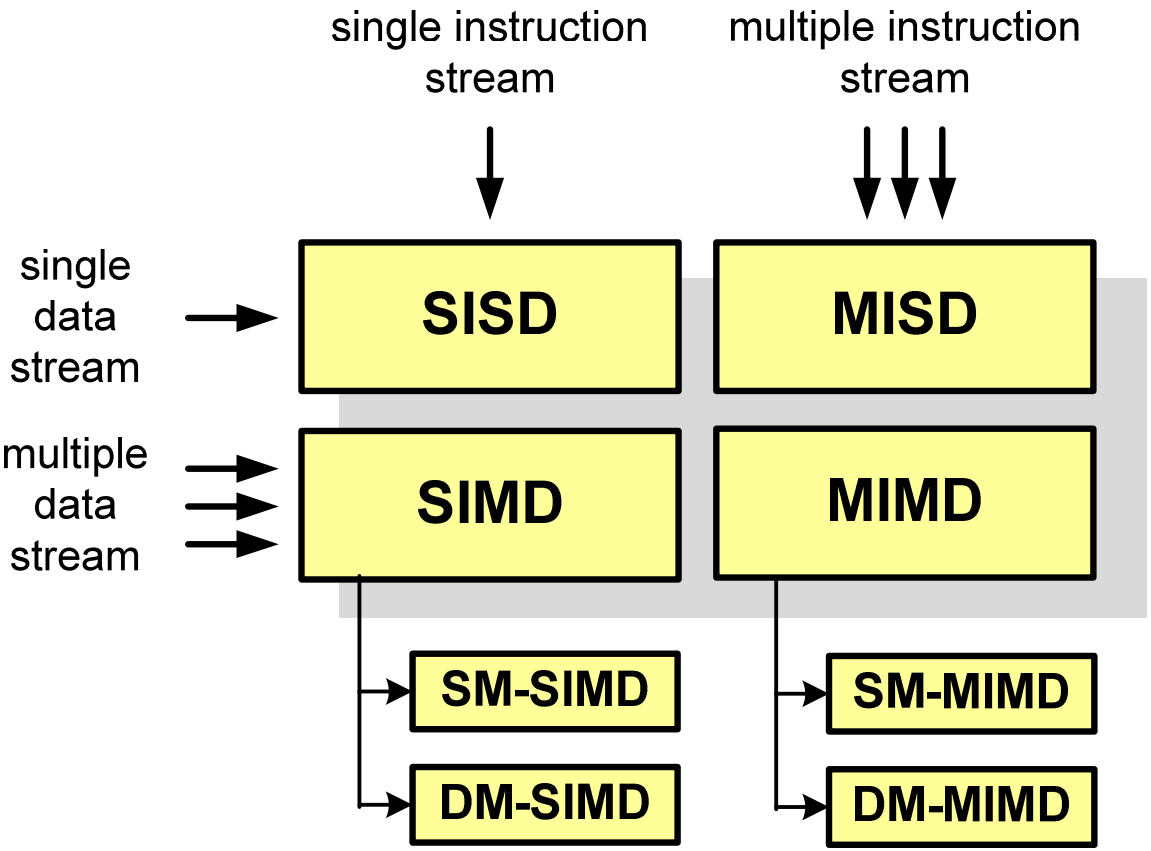


Liczba zdenormalizowana: 3,719046E-42

Klasyfikacja systemów komputerowych

- Taksonomia Flynna - pierwsza, najbardziej ogólna klasyfikacja architektur komputerowych (1972):
  - Flynn M.J.: „Some Computer Organizations and Their Effectiveness”, IEEE Transactions on Computers, Vol. C-21, No 9, 1972.
- Opiera się na liczbie przetwarzanych strumieni rozkazów i strumieni danych:
  - strumień rozkazów (Instruction Stream) - odpowiednik licznika rozkazów; system złożony z n procesorów posiada n liczników rozkazów, a więc n strumieni rozkazów
  - strumień danych (Data Stream) - zbiór operandów, np. system rejestrujący temperaturę mierzoną przez n czujników posiada n strumieni danych

Taksonomia Flynna



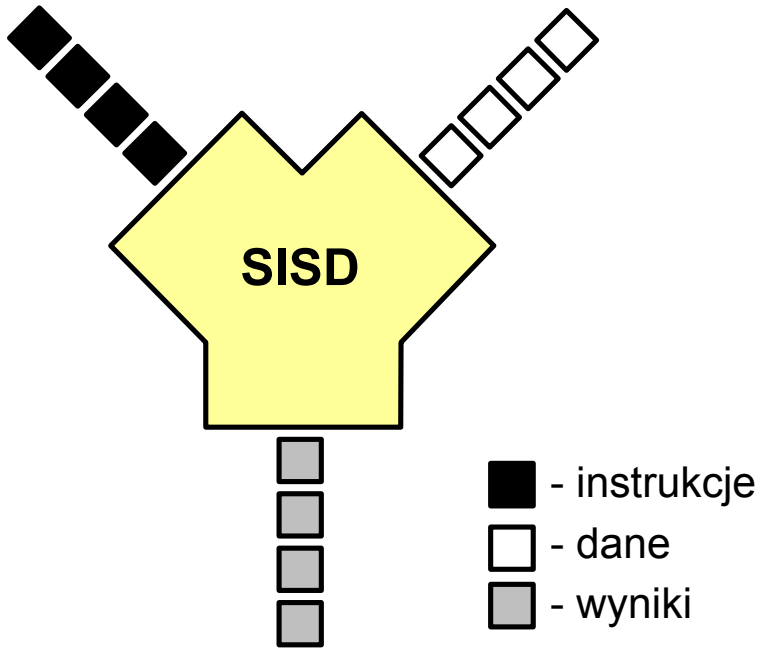
SI - Single instruction
MI - Multiple Instruction

SD - Single Data
MD - Multiple Data

SM - Shared Memory
DM - Distributed Memory

SISD (Single Instruction, Single Data)

- Jeden wykonywany program przetwarza jeden strumień danych
- Klasyczne komputery zbudowane według architektury von Neumanna
- Zawierają:
  - jeden procesor
  - jeden blok pamięci operacyjnej zawierający wykonywany program.



SISD (Single Instruction, Single Data)

Komputer IBM PC/AT

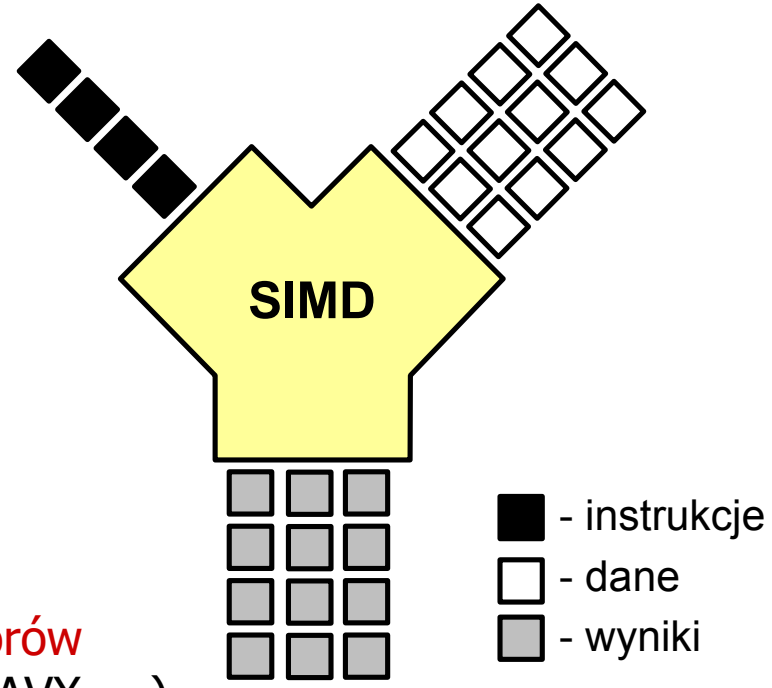
Komputer PC

Komputer PC

Laptop

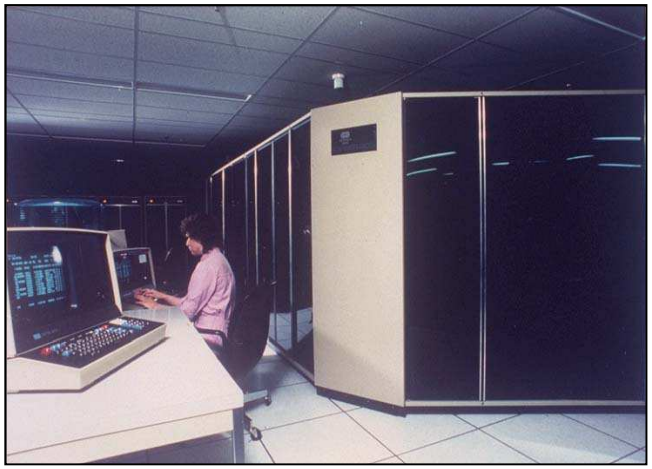
SIMD (Single Instruction, Multiple Data)

- Jeden wykonywany program przetwarza wiele strumieni danych
- Te same operacje wykonywane są na różnych danych
- Podział:
  - SM-SIMD (Shared Memory SIMD):
    - komputery wektorowe
    - rozszerzenia strumieniowe procesorów (MMX, 3DNow!, SSE, SSE2, SSE3, AVX, ...)
  - DM-SIMD (Distributed Memory SIMD):
    - tablice procesorów
    - procesory kart graficznych (GPGPU)



SM-SIMD - Komputery wektorowe

CDC Cyber 205 (1981)



Cray-1 (1976)

Cray-2 (1985)

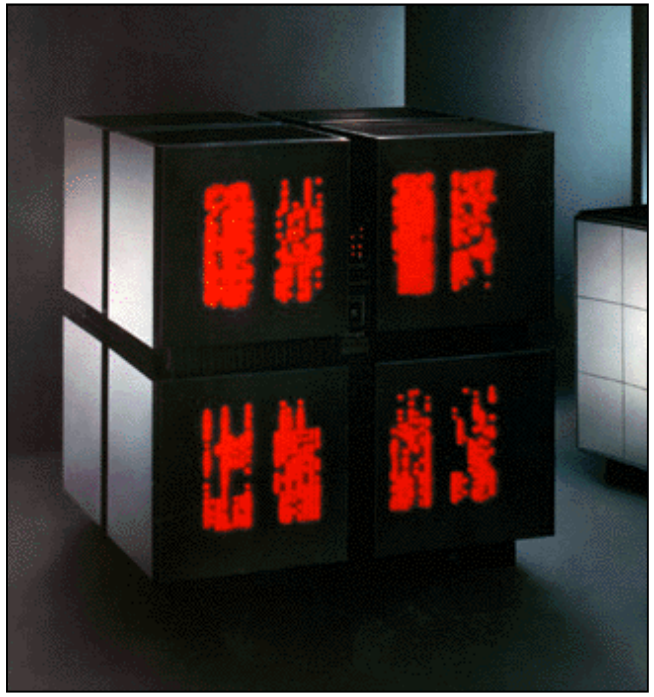
Hitachi S3600 (1994)

DM-SIMD - Tablice procesorów

Illiac IV
(1976)

Illiac IV
(1976)

MasPar
MP-1/MP-2
(1990)



Thinking
Machines
CM-2
(1987)

DM-SIMD - Procesory graficzne (GPU)

GeForce
GTX Titan X

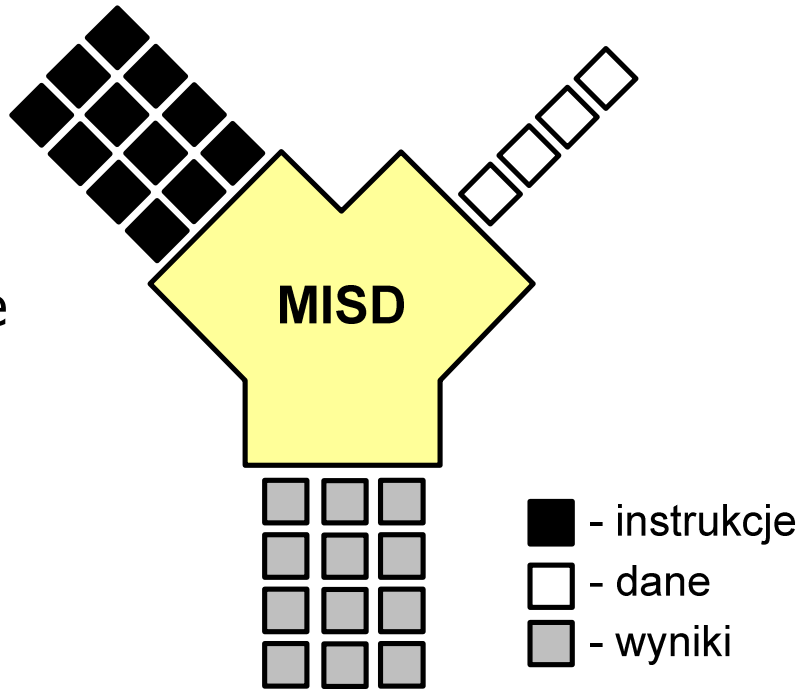
Tesla
V100

DGX-1
Volta

Tesla
D870

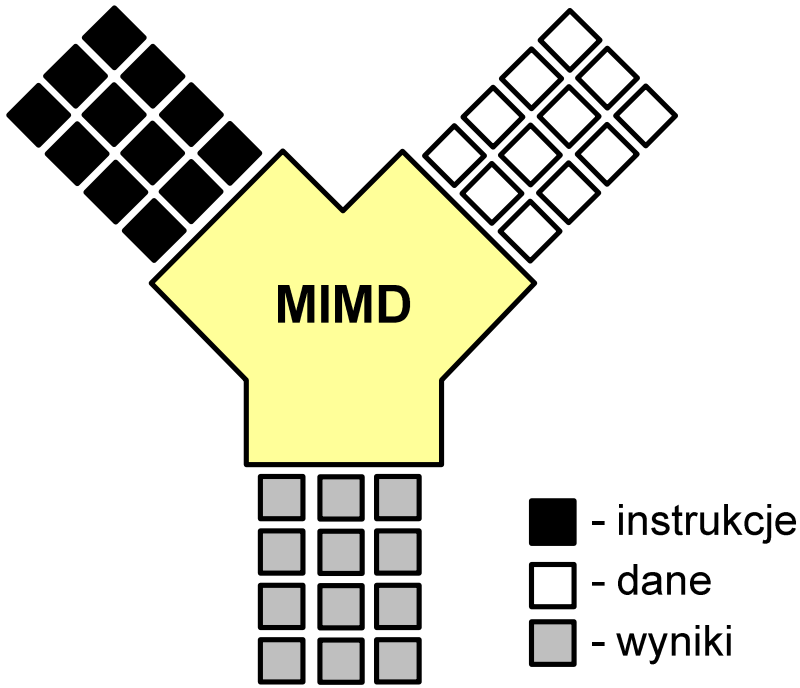
MISD (Multiple Instruction, Single Data)

- Wiele równolegle wykonywanych programów przetwarza jednocześnie jeden wspólny strumień danych
- Systemy tego typu nie są spotykane



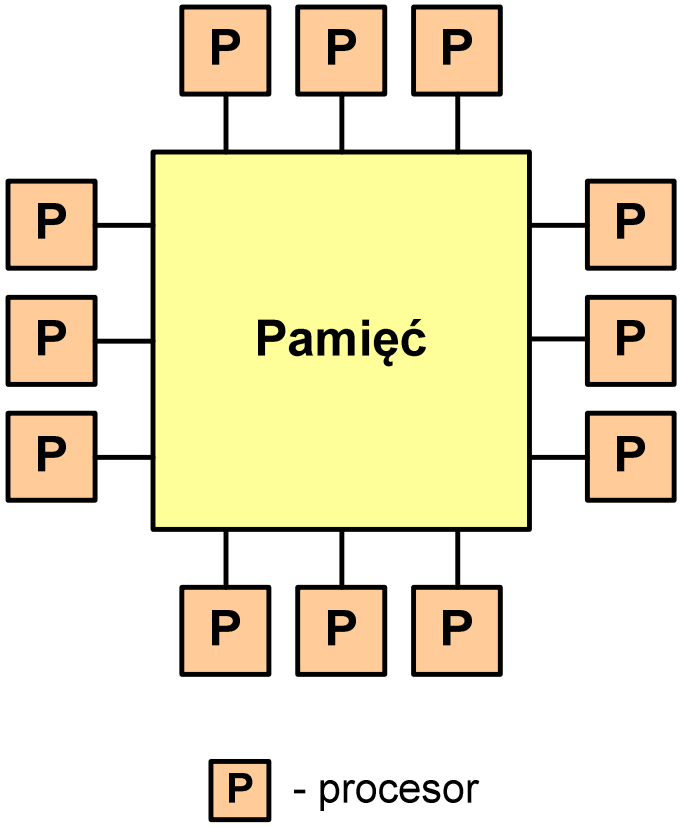
MIMD (Multiple Instruction, Multiple Data)

- Równolegle wykonywanych jest wiele programów, z których każdy przetwarza własne strumienie danych
- Podział:
  - SM-MIMD (Shared Memory):
    - wieloprocesory
  - DM-MIMD (Distributed Memory):
    - wielokomputery
    - klastry
    - gridy



SM-MIMD - Wieloprocesory

- Systemy z niezbyt dużą liczbą działających niezależnie procesorów
- Każdy procesor ma dostęp do wspólnej przestrzeni adresowej pamięci
- Komunikacja procesorów poprzez uzgodniony obszar wspólnej pamięci
- Do SM-MIMD należą komputery z procesorami wielordzeniowymi
- Podział:
  - UMA (Uniform Memory Access)
  - NUMA (NonUniform Memory Access)
  - COMA (Cache Only Memory Architecture)



SM-MIMD - Wieloprocesory

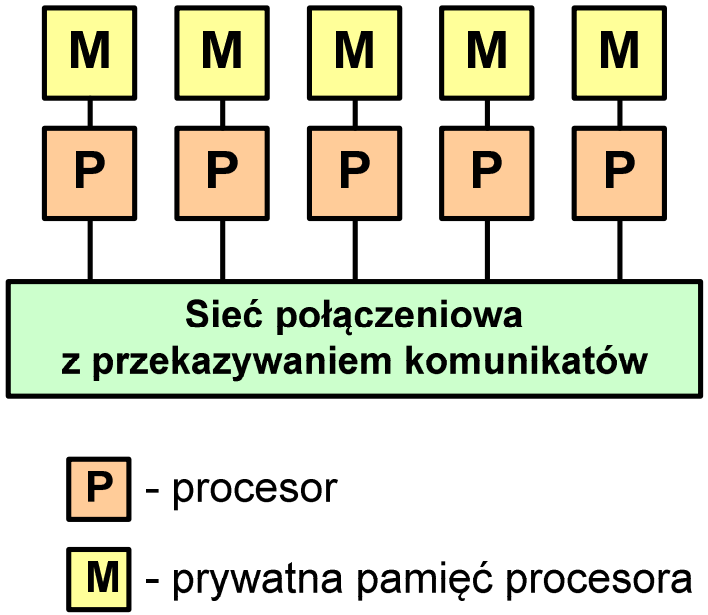
Cray YM-P
(1988)

Cray
CS6400
(1993)

Cray J90
(1994)

DM-MIMD - Wielokomputery

- Każdy procesor wyposażony jest we własną pamięć operacyjną, niedostępną dla innych procesorów
- Komunikacja między procesorami odbywa się za pomocą sieci poprzez przesyłanie komunikatów
- Biblioteki komunikacyjne:
  - MPI (Message Passing Interface)
  - PVM (Parallel Virtual Machine)



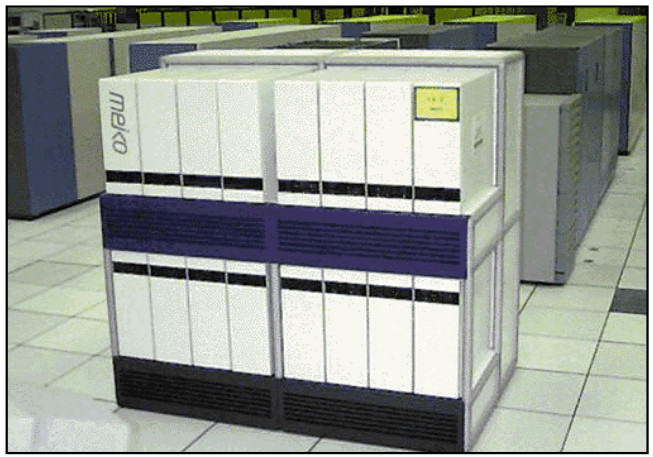
DM-MIMD - Wielokomputery

Cray T3E
(1995)

Thinking Machines CM-5
(1991)

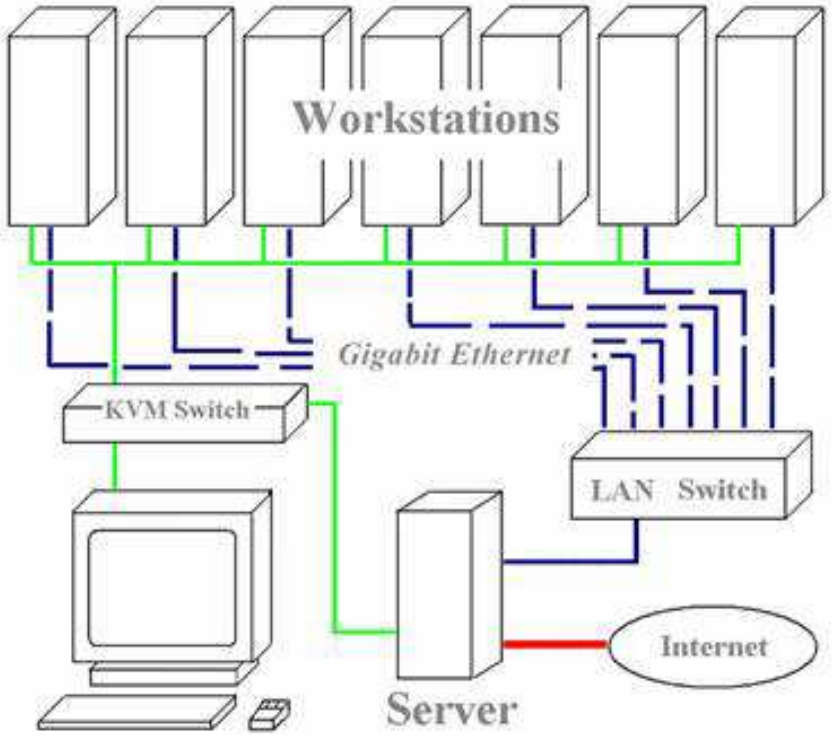
nCube 2s
(1993)

Meiko CS-2
(1993)



DM-MIMD - Klastry

- Klaster (cluster):
  - równoległy lub rozproszonego system składający się z komputerów
  - komputery połączone są siecią
  - używany jest jako pojedynczy, zintegrowany zespół obliczeniowy
- Węzeł (node) - pojedynczy komputer przyłączony do klastra i wykonujący zadania obliczeniowe



źródło:
http://leda.elfak.ni.ac.rs/projects/SeeGrid/see_grid.htm

KVM - Keyboard, Video, Mouse

DM-MIMD - Klastry

- Klastry Beowulf budowane były ze zwykłych komputerów PC

Odin II Beowulf Cluster Layout, University of Chicago, USA

DM-MIMD - Klastry

- Klastry Beowulf budowane były ze zwykłych komputerów PC

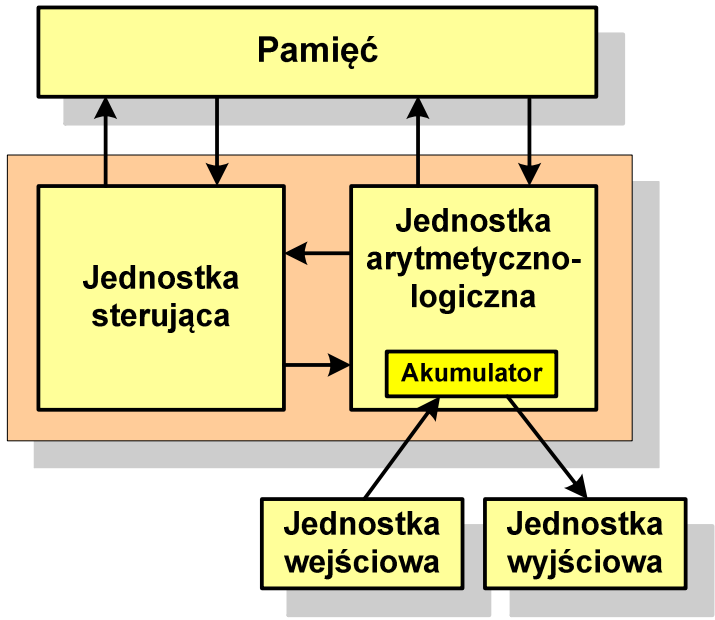
NASA 128-processor Beowulf cluster: A cluster built from 64 ordinary PC's

DM-MIMD - Klastry

Early Aspen Systems Beowulf Cluster With RAID

Architektura von Neumanna

- Rodzaj architektury komputera, opisanej w 1945 roku przez matematyka Johna von Neumanna
- Inne spotykane nazwy: architektura z Princeton, store-program computer (koncepcja przechowywanego programu)
- Zakłada podział komputera na kilka części:
  - jednostka sterująca (CU - Control Unit)
  - jednostka arytmetyczno-logiczna (ALU - Arithmetic Logic Unit)
  - pamięć główna (memory)
  - urządzenia wejścia-wyjścia (input/output)

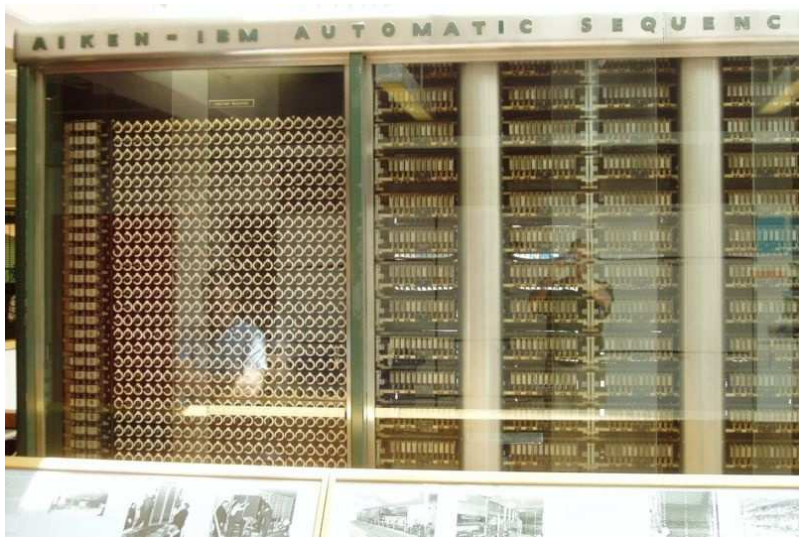


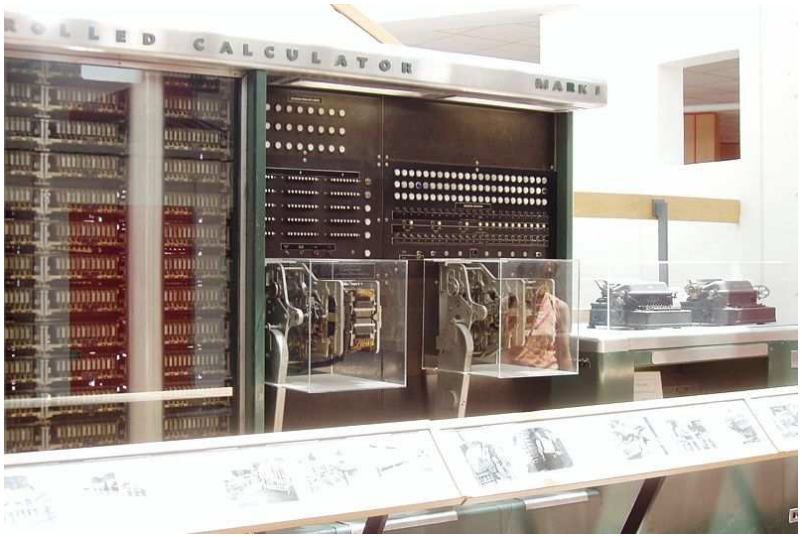
Architektura von Neumanna - podstawowe cechy

- Informacje przechowywane są w komórkach pamięci (cell) o jednakowym rozmiarze, każda komórka ma numer - adres
- Dane oraz instrukcje programu (rozkazy) zakodowane są za pomocą liczb i przechowywane w tej samej pamięci
- Praca komputera to sekwencyjne odczytywanie instrukcji z pamięci komputera i ich wykonywanie w procesorze
- Wykonanie rozkazu:
  - pobranie z pamięci słowa będącego kodem instrukcji
  - pobranie z pamięci danych
  - wykonanie instrukcji
  - zapisanie wyników do pamięci
- Dane i instrukcje czytane są przy wykorzystaniu tej samej magistrali

Architektura harwardzka

- Architektura komputera, w której pamięć danych jest oddzielona od pamięci instrukcji
- Nazwa architektury pochodzi komputera Harward Mark I:
  - zaprojektowany przez Howarda Aikena
  - pamięć instrukcji - taśma dziurkowana,
    pamięć danych - elektromechaniczne liczniki



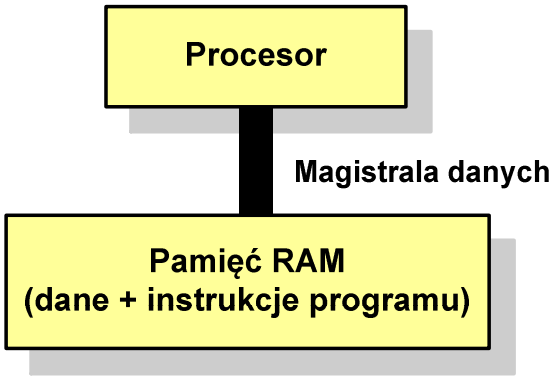


Architektura harwardzka

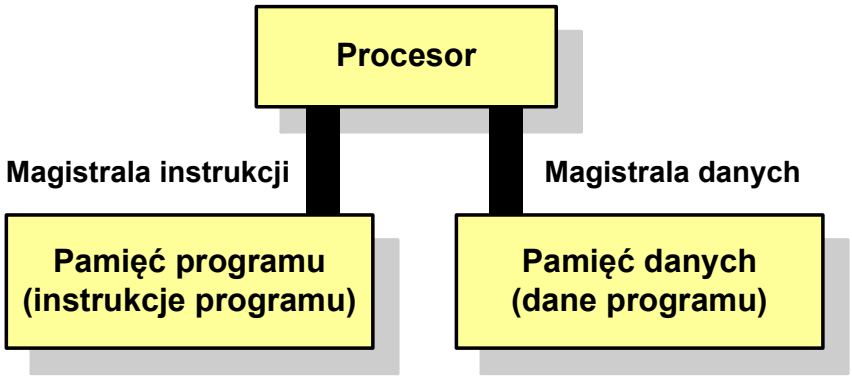
- Pamięci danych i instrukcji mogą różnić się:
  - technologią wykonania
  - strukturą adresowania
  - długością słowa
- Przykład:
  - ATmega16 - 16 kB Flash, 1 kB SRAM, 512 B EEPROM
- Procesor może w tym samym czasie czytać instrukcje oraz uzyskiwać dostęp do danych

Architektura harwardzka i von Neumanna

- W architekturze harwardzkiej pamięć instrukcji i pamięć danych:
  - zajmują różne przestrzenie adresowe
  - mają oddzielne szyny (magistrale) do procesora
  - zaimplementowane są w inny sposób



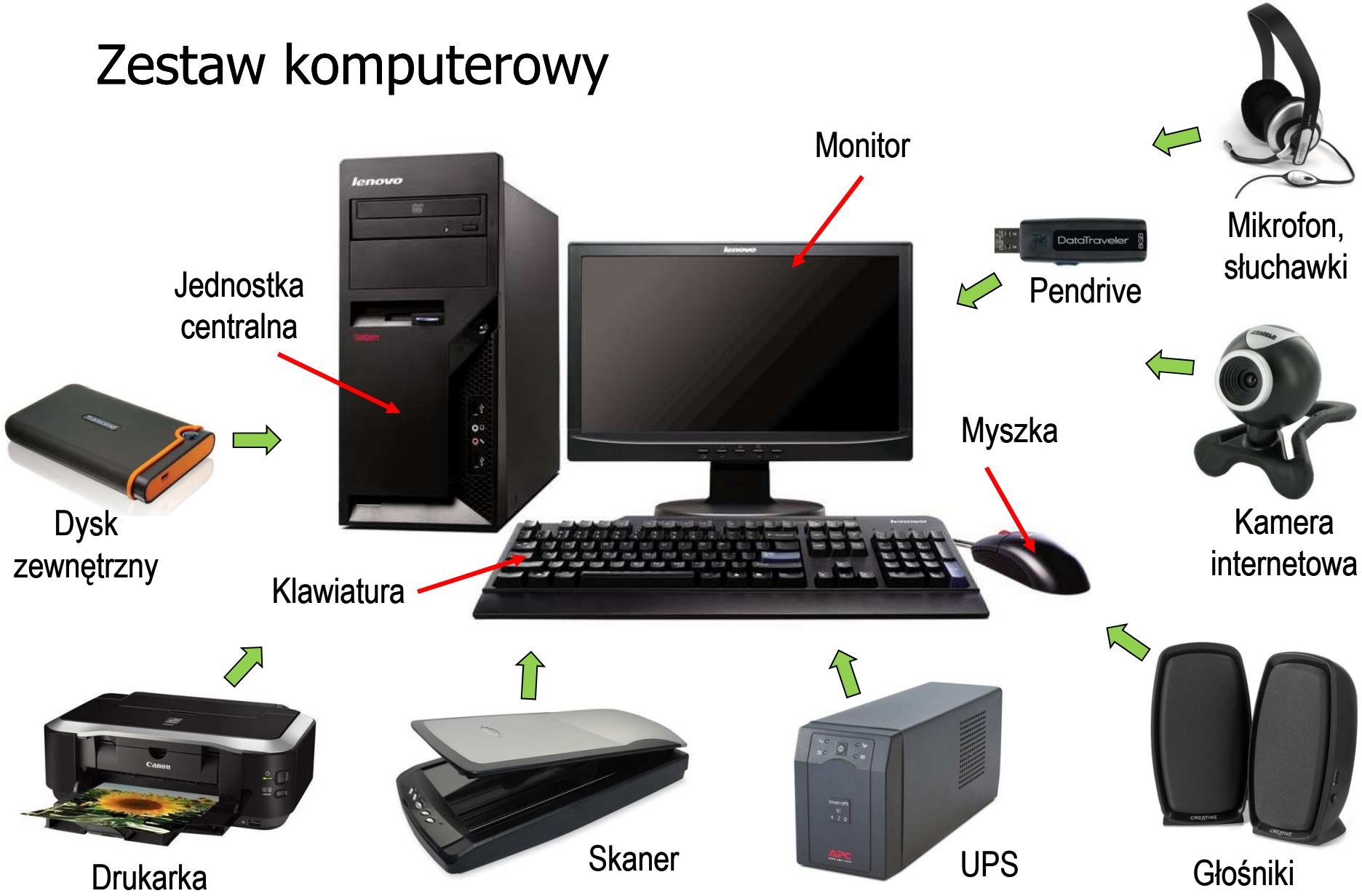
Architektura von Neumanna



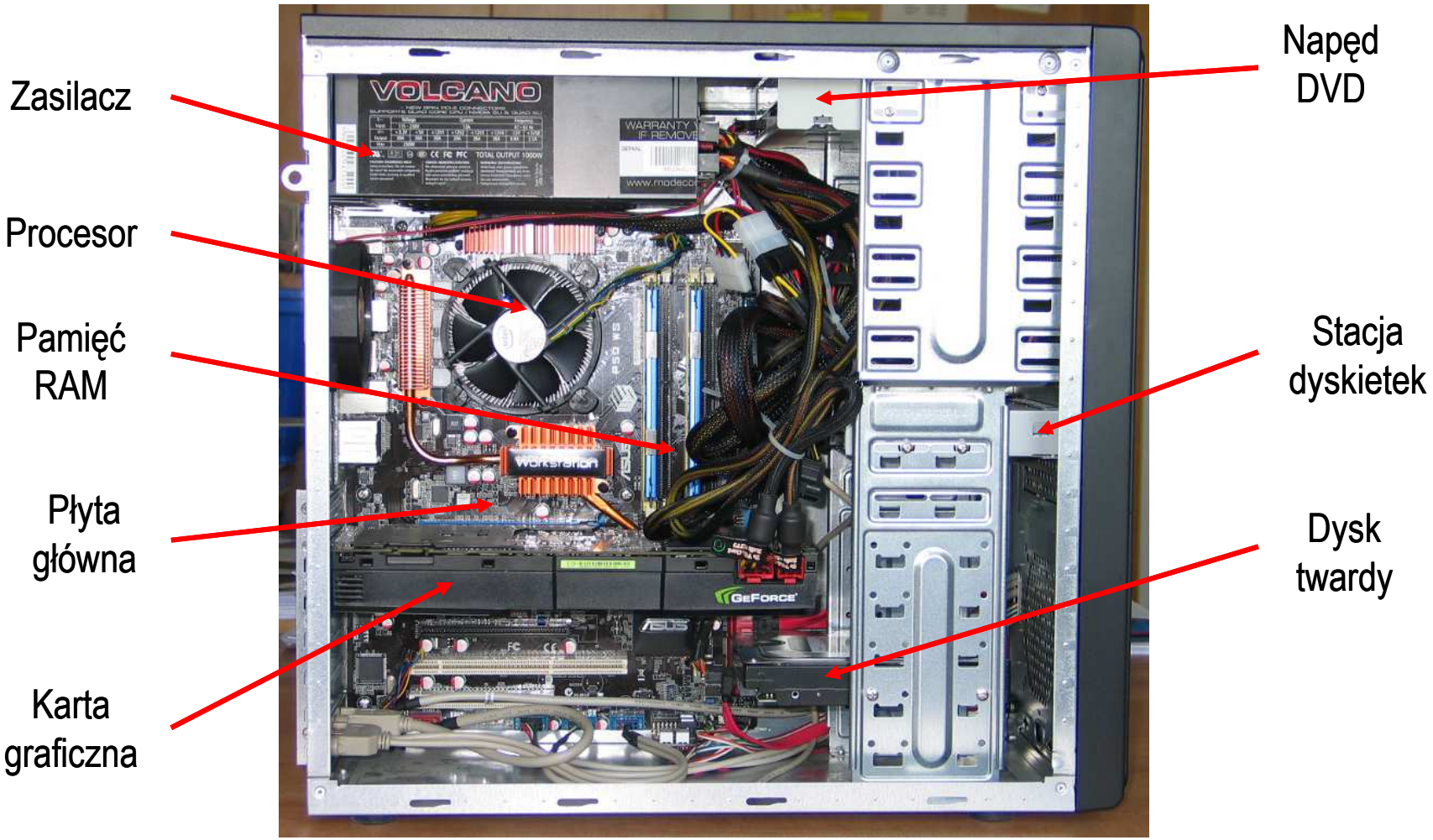
Architektura harwardzka

- Zmodyfikowana architektura harwardzka:
  - oddzielone pamięci danych i rozkazów, lecz wykorzystujące wspólną magistralę

Zestaw komputerowy



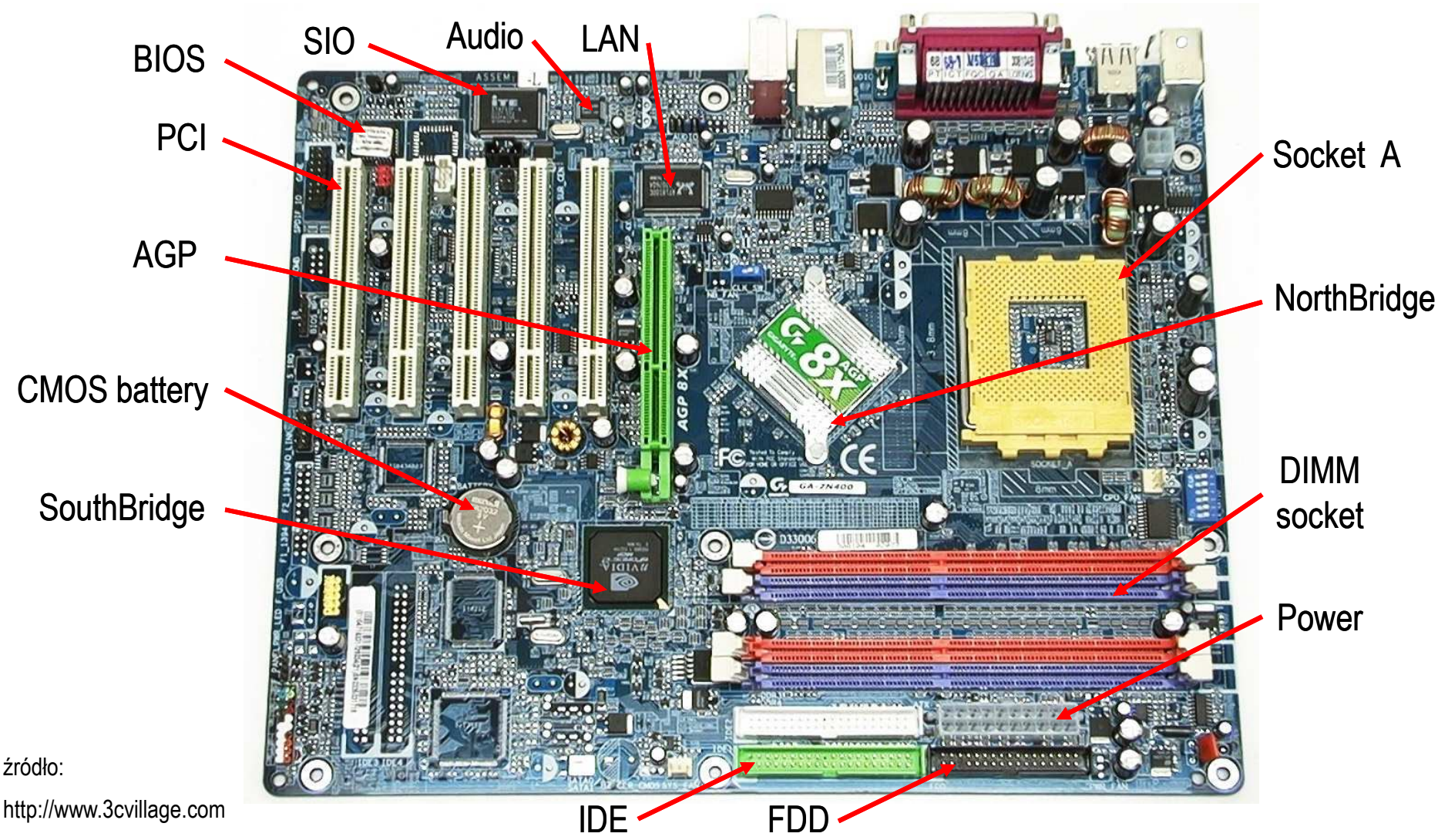
Jednostka centralna



Płyta główna (motherboard) - przykłady

Model	Gigabyte GA-7N400-L	Gigabyte GA-X58A-UD5	Gigabyte G1-Assassin 2
Rok	2003	2009	2011
Gniazdo	Socket A	Socket 1366	Socket 2011
Procesor	AMD Athlon, Athlon XP	Intel Core i7	Intel Core i7
Northbridge	nVIDIA nForce 2 Ultra 400	Intel X58 Express Chipset	Intel X79
Southbridge	nVIDIA nForce 2 MCP	Intel ICH10R	
Pamięć	4 x 184-pin DDR DIMM sockets, max. 3 GB	6 x 1.5V DDR3 DIMM sockets, max. 24 GB	4 x 1.5V DDR3 DIMM sockets, max. 32 GB
Format	ATX	ATX	ATX
Inne	AGP, 5 × PCI, 2 × IDE, FDD, LPT, 2 × COM, 6 × USB, IrDA, RJ45, 2 × PS/2	4 × PCIe x16, 2 × PCIe x1, PCI, 8 × SATA II 3 Gb/s, 2 × SATA II 6 Gb/s, 2 × eSATA, IDE, FDD, 2 × RJ45, 10 × USB 2.0, 2 × USB 3.0, 2 × PS/2	3 × PCIe x16, 2 × PCIe x1, PCI, 4 × SATA II 3 Gb/s, 4 × SATA III 6 Gb/s, 2 × eSATA, RJ45, 9 × USB 2.0, 3 × USB 3.0, PS/2

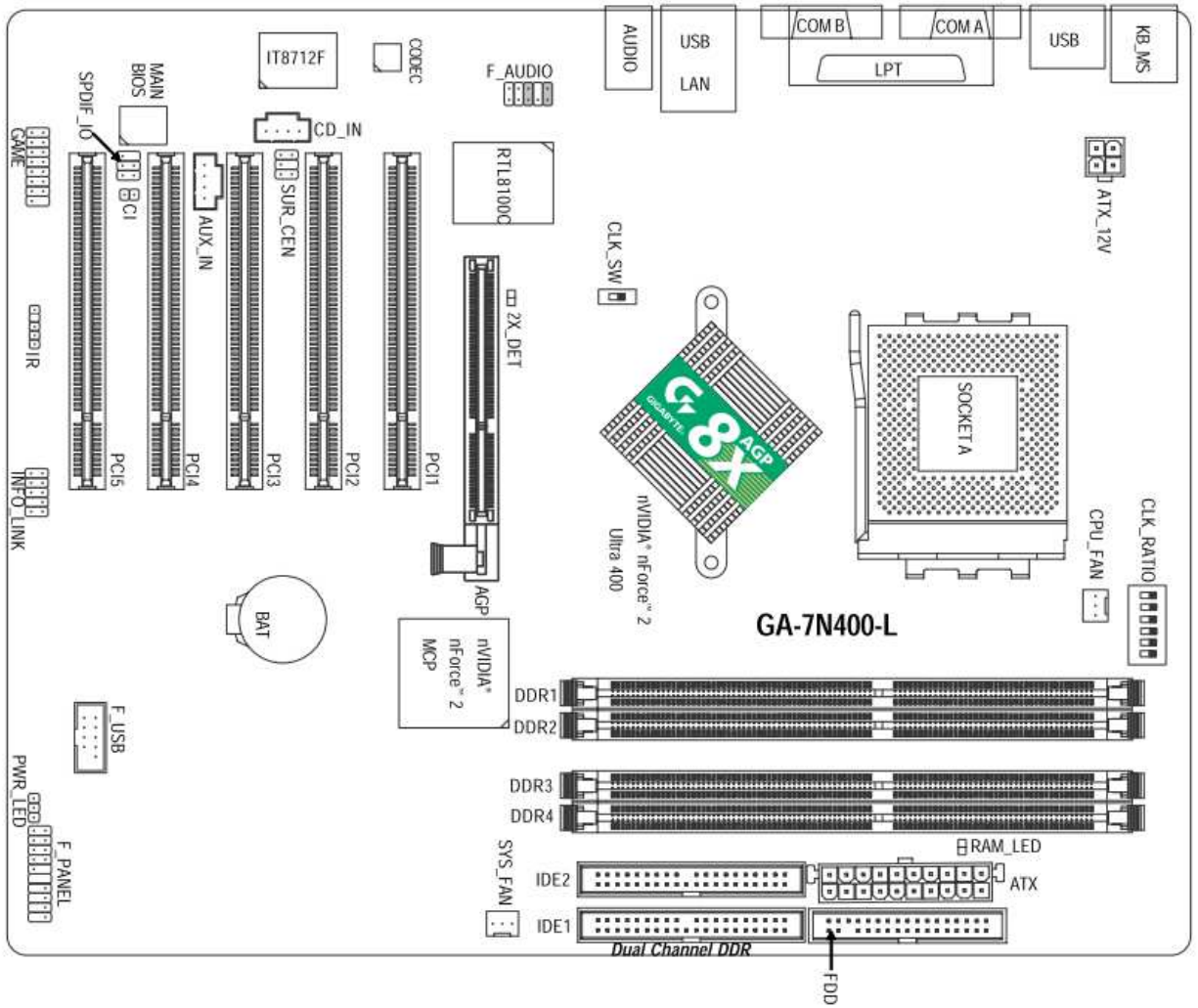
Gigabyte GA-7N400-L



źródło:

http://www.3cvillage.com

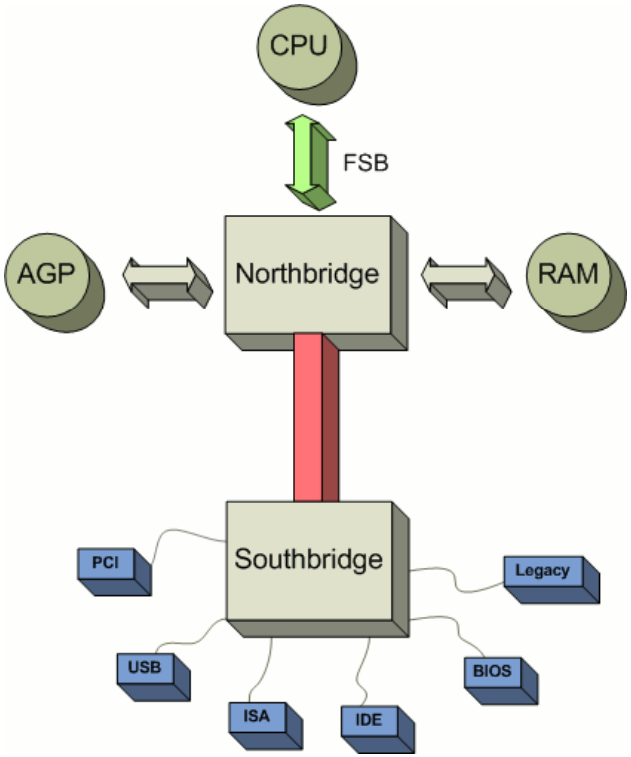
Gigabyte GA-7N400-L

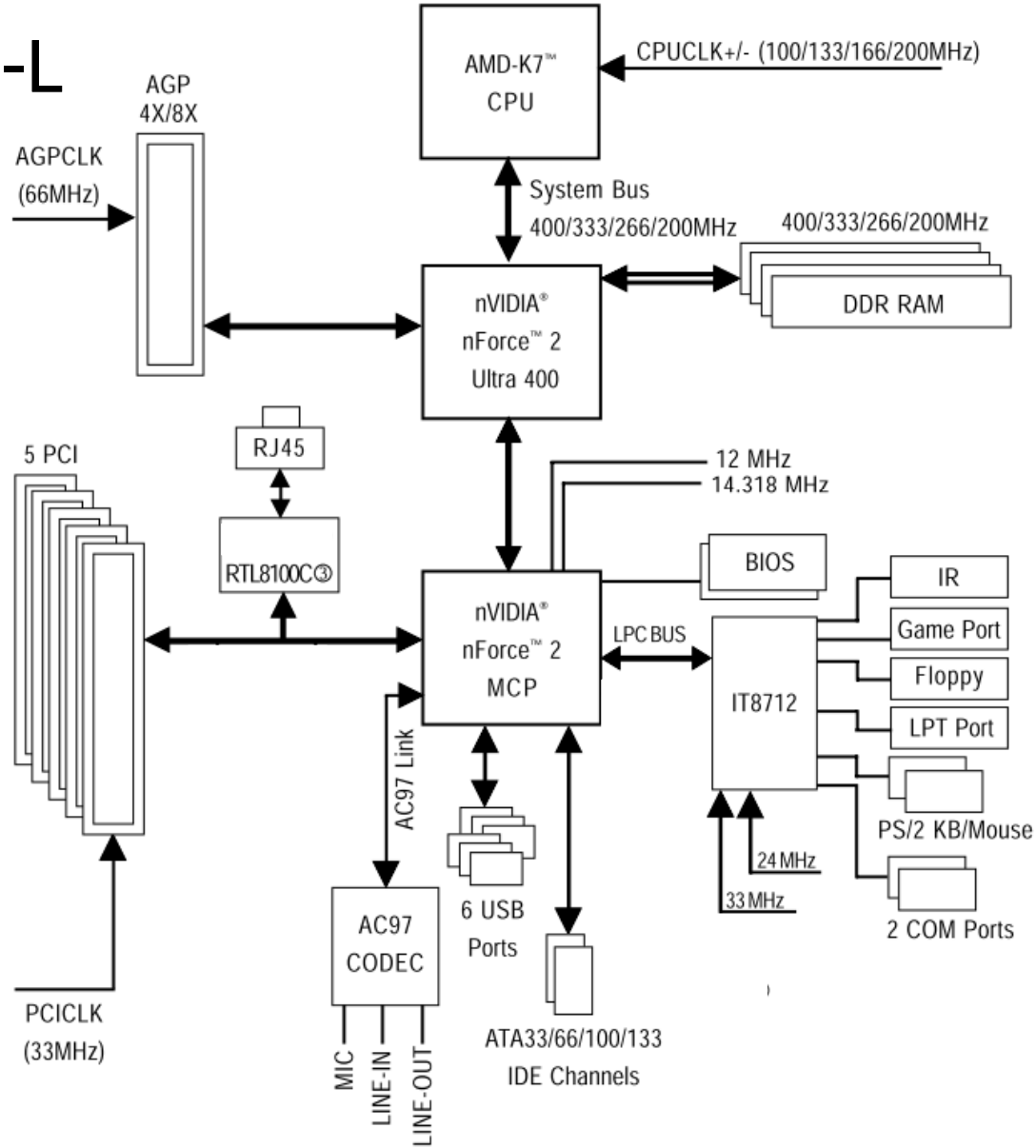


źródło:

GA-7N400 Pro2 / GA-7N400 / GA-7N400-L
AMD Socket A
Processor Motherboard
User's Manual

Gigabyte GA-7N400-L

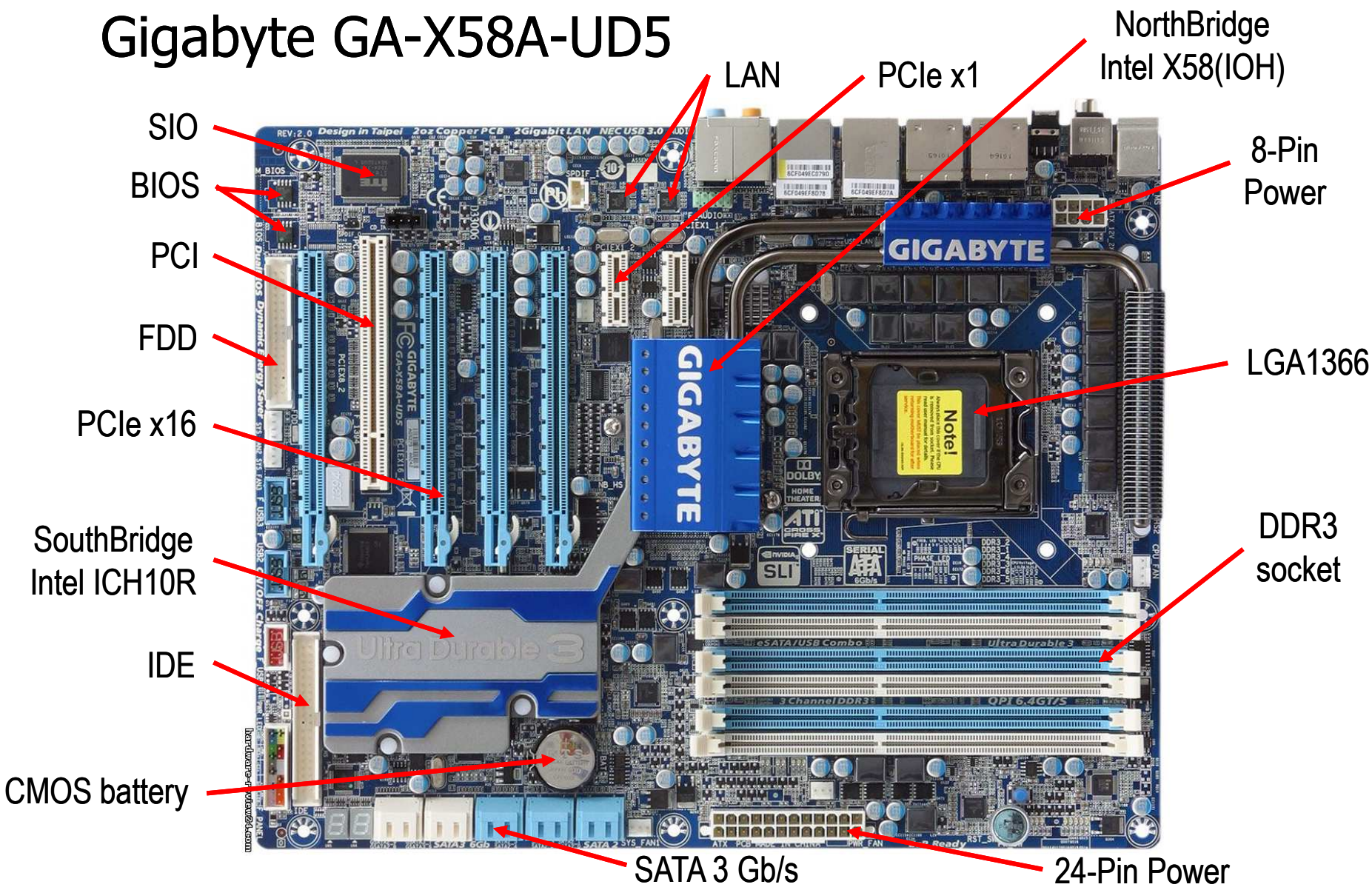




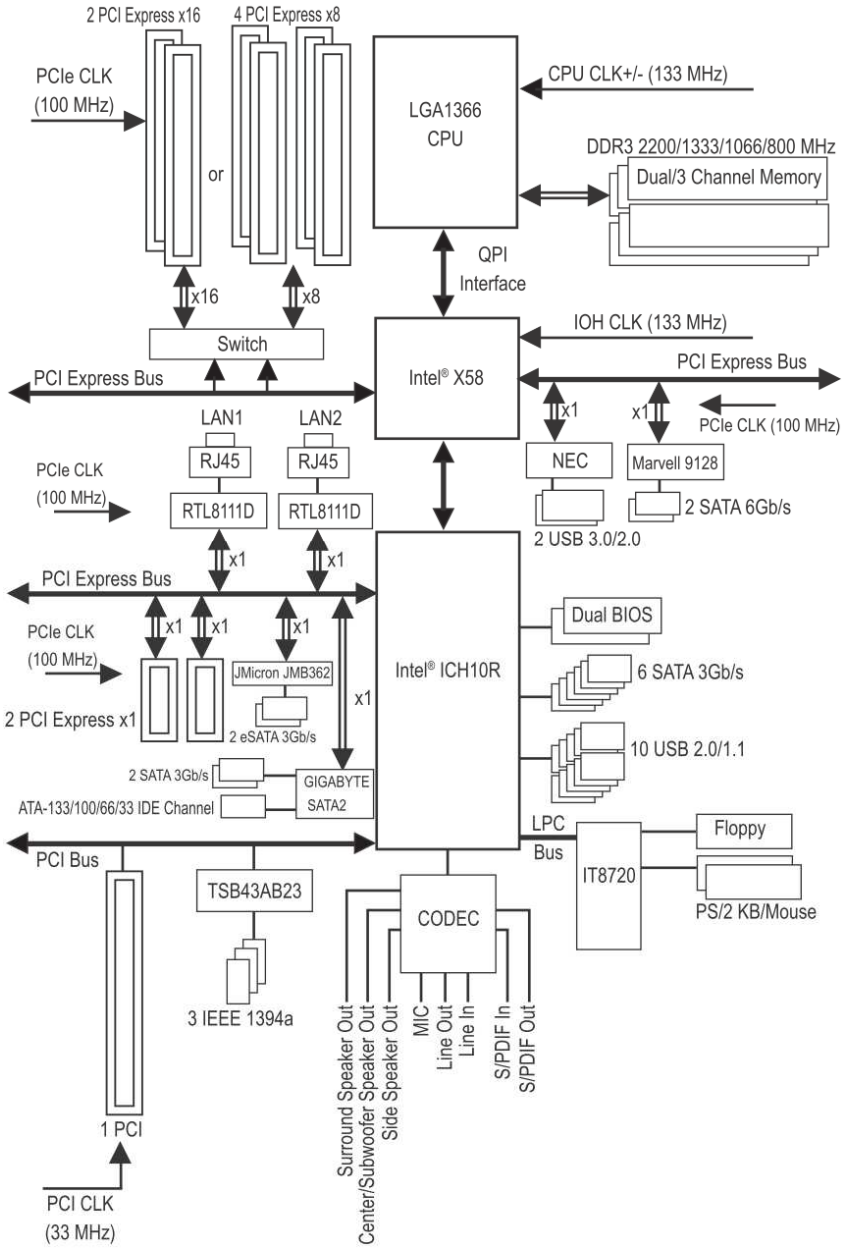
źródło:

GA-7N400 Pro2 / GA-7N400 / GA-7N400-L
AMD Socket A Processor Motherboard
User's Manual

Gigabyte GA-X58A-UD5



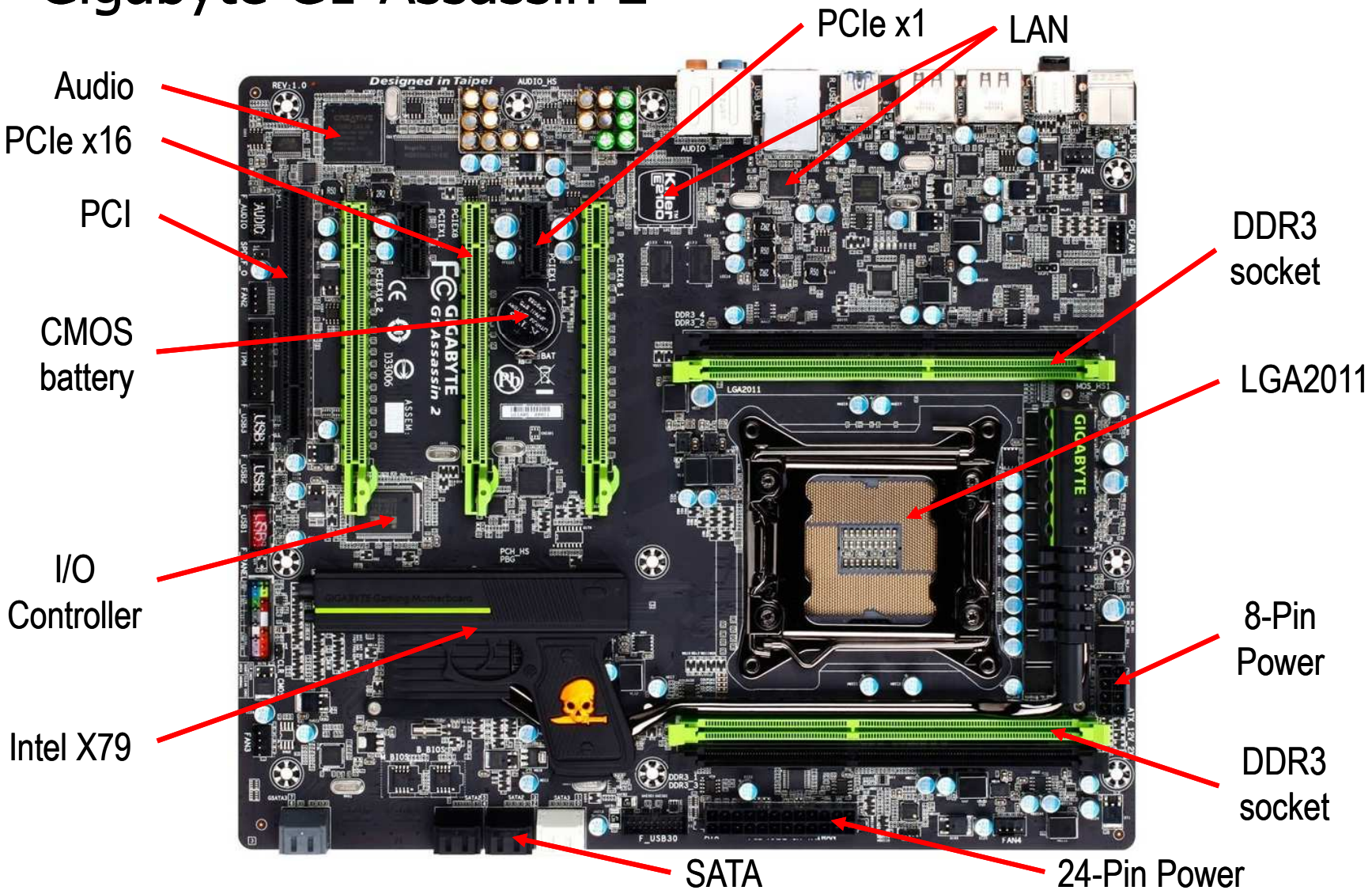
Gigabyte GA-X58A-UD5



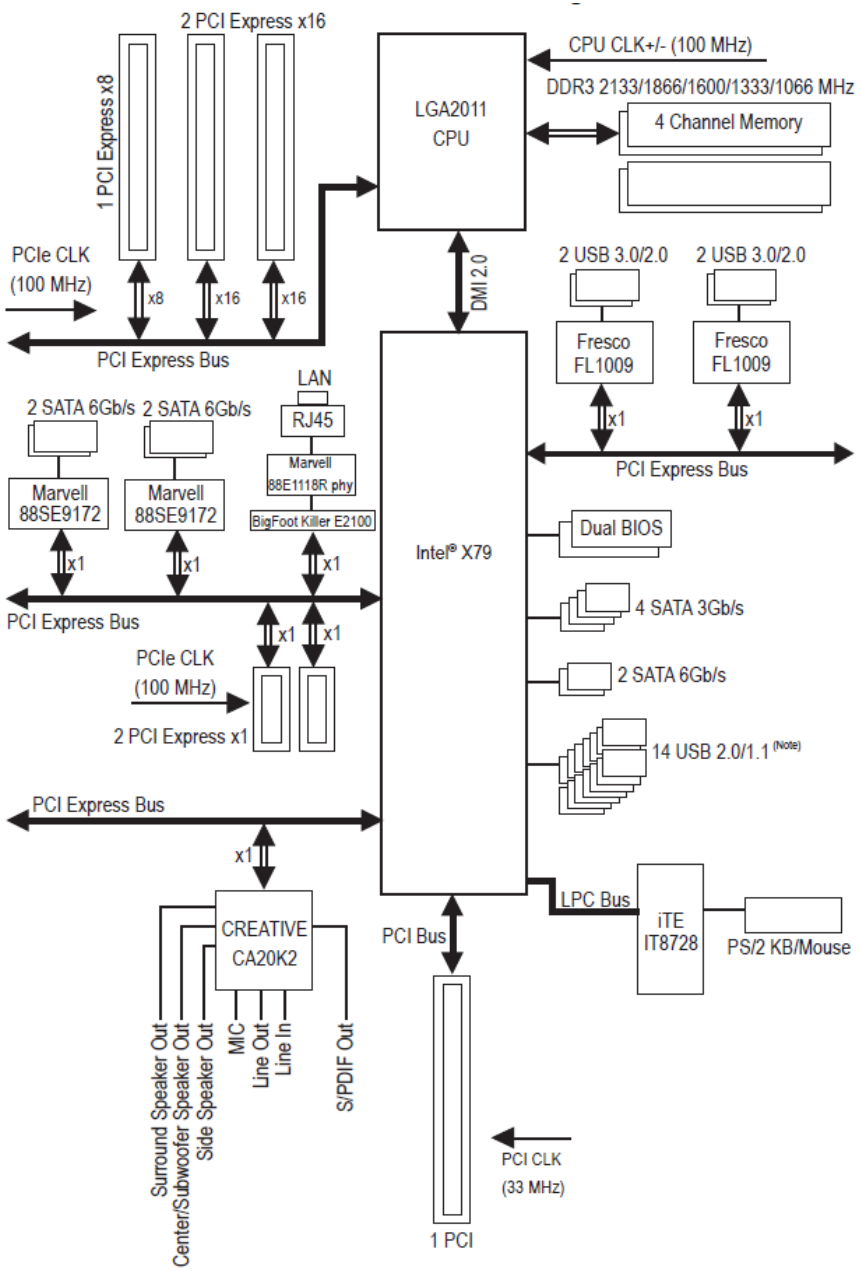
źródło:

GA-X58A-UD5
LGA1366 socket motherboard for Intel ® Core ™ i7 processor family
User's Manual

Gigabyte G1-Assassin 2



Gigabyte G1-Assassin 2



źródło:

Gigabyte G1.Assassin 2, User's Manual, Rev. 1001

Płyty główne - standardy

Standard	Rok	Wymiary
AT	1984 (IBM)	12 × 11–13 in 305 × 279–330 mm
Baby-AT	1985 (IBM)	8.5 × 10–13 in 216 × 254–330 mm
ATX	1996 (Intel)	12 × 9.6 in 305 × 244 mm
Micro-ATX	1996	9.6 × 9.6 in 244 × 244 mm
Mini-ITX	2001 (VIA)	6.7 × 6.7 in 170 × 170 mm max.
Nano-ITX	2003 (VIA)	4.7 × 4.7 in 120 × 120 mm
Pico-ITX	2007 (VIA)	100 × 72 mm max.

źródło: http://en.wikipedia.org

Płyty główne - standardy

Standard-ATX

Micro-ATX

Mini-ITX

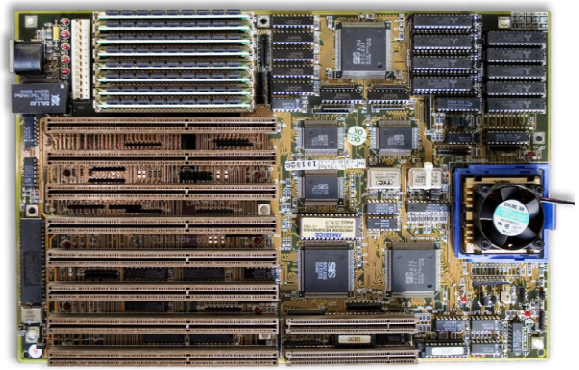
Nano-ITX

Pico-ITX

ATX (Advanced Technology Extended)

źródło:

http://en.wikipedia.org



Baby-AT

AT (Advanced Technology)

Procesory Intel - mikroarchitektury

- Mikroarchitektura - organizacja procesora

Proces	Mikroarchitektura	Nazwa kodowa	Data	Procesory
65 nm	P6, Netburst	Presler, Cedar Mill, Yonah	2006-01-05	Presler, Cedar Mill, Yonah
	Core	Merom	2006-07-27	Clovertown, Kentsfield, Conroe, Merom
45 nm		Penryn	2007-11-11	Harpertown, Yorkfield, Wolfdale, Penryn
	Nehalem	Nehalem	2008-11-17	Bloomfield, Lynnfield, Clarksfield
32 nm		Westmere	2010-01-04	Westmere-EX, -EP, Gulftown, Clarkdale
	Sandy Bridge	Sandy Bridge	2011-01-09	Sandy Bridge-EP, -E, -M, Sandy Bridge
22 nm		Ivy Bridge	2012-04-29	Ivy Bridge-EX, -EP, -E, -M, Ivy Bridge
	Haswell	Haswell	2013-06-02	Haswell-EX, -EP, -E, -DT, -MB, -LP
14 nm		Broadwell	2014-09-05	Broadwell-EX, -EP, -E
	Skylake	Skylake	2015-08-05	Skylake-EX, -EP
		Kaby Lake	2017-01-03	Kabylake-X
		Coffee Lake	2017-10-05	CoffeLake-DT/H
10 nm		Cannonlake	2018	
	Icelake	Icelake	2018 / 2019	
		Tigerlake	2019	

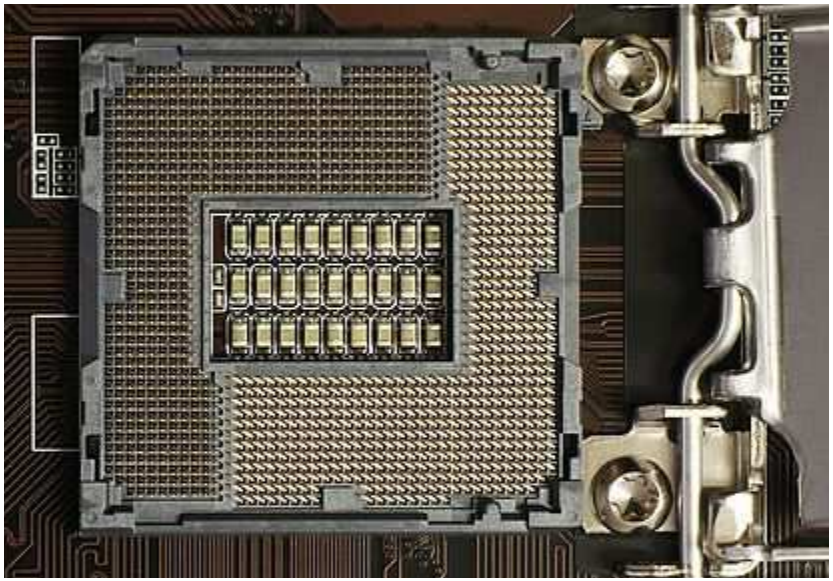
Procesory Intel - mikroarchitektury

- Mikroarchitektura - organizacja procesora

Proces	Mikroarchitektura	Nazwa kodowa	Nazwa marketingowa
65 nm	P6, Netburst	Presler, Cedar Mill, Yonah	Core, Pentium 4, Pentium D, Pentium M, Celeron, Xeon, …
45 nm	Core	Merom	Core 2, Pentium Dual-Core, Pentium, Celeron Dual-Core, Celeron, Celeron M, Xeon
		Penryn	
32 nm	Nehalem	Nehalem	Core i3, i5, i7, Pentium, Celeron, Xeon
		Westmere	
22 nm	Sandy Bridge	Sandy Bridge	Core i3, i5, i7 (2 gen.), Pentium, Celeron, Xeon
		Ivy Bridge	Core i3, i5, i7 (3 gen.), Pentium, Celeron, Xeon
14 nm	Haswell	Haswell	Core i3, i5, i7 (4 gen.), Pentium, Celeron, Xeon
		Broadwell	Core i3, i5, i7 (5 gen.), Core M, Pentium, Celeron, Xeon
	Skylake	Skylake	Core i3, i5, i7 (6 gen.), Core M, Pentium, Celeron, Xeon
		Kaby Lake	Core i3, i5, i7, Celeron, Pentium, Xeon
		Coffee Lake	Core i3, i5, i7, i9, Celeron, Pentium Gold
10 nm		Cannonlake	
	Icelake	Icelake	
		Tigerlake	

Procesory Intel - LGA 1150 (Socket H3)

- LGA (Land Grid Array) - na procesorze złocone, miedziane, płaskie styki, dociskane do pinów w gnieździe na płycie głównej
- czerwiec 2013 roku, liczba pinów: 1150
- procesory:
  - Haswell (22 nm): Celeron, Pentium, Core i3 / i5 / i7
  - Broadwell (14 nm): Core M, Celeron, Pentium, Core i3 / i5 / i7
- chipsety:
  - Haswell: H81, B85, Q85, Q87, H87, Z87
  - Broadwell: Z97, H97

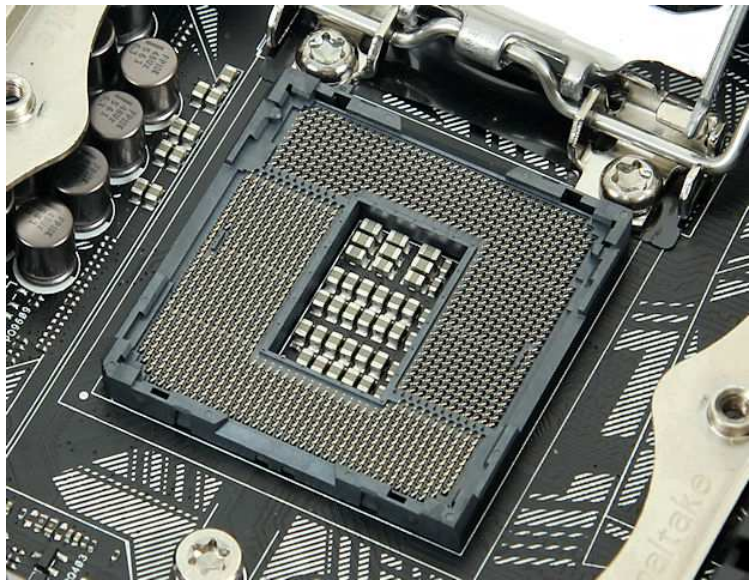


LGA 1150

Procesory Intel - LGA 1151 (Socket H4)

- sierpień 2015 roku, liczba pinów: 1151
- procesory Skylake (14 nm) i Kaby Lake (14 nm)
- wsparcie dla pamięci RAM: DDR4, DDR3(L)

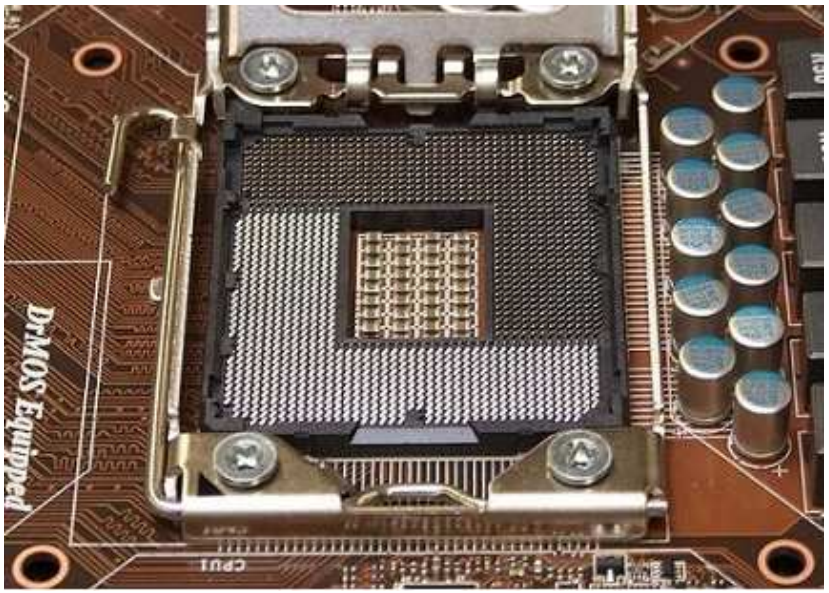
Core i7-6700K



LGA 1151

Procesory Intel - LGA 2011 (Socket R)

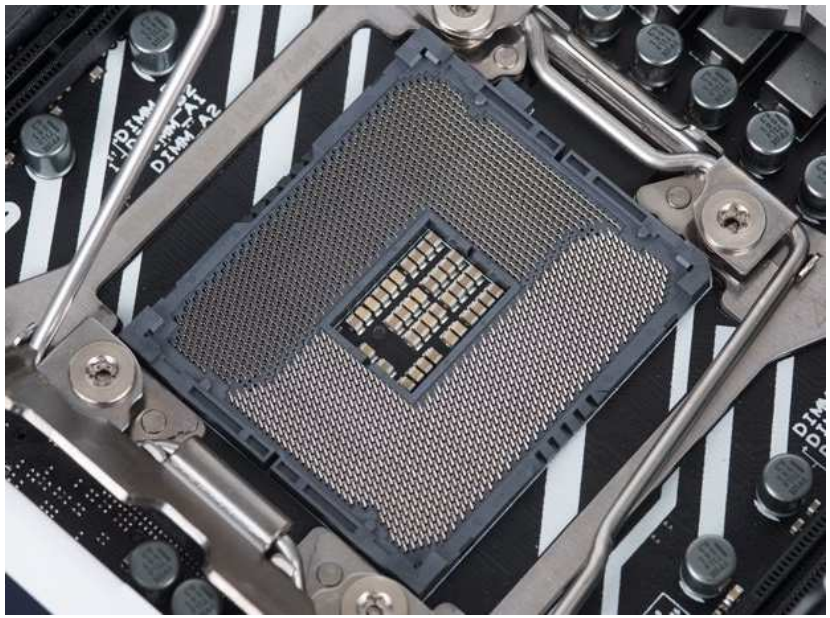
- listopad 2011 roku, liczba pinów: 2011
- procesory:
  - Sandy Bridge-E/EP (22 nm): Core i7, Xeon
  - Ivy Bridge-E/EP (14 nm): Core i7, Xeon
  - Haswell-E (22 nm): Core i7
- chipsety: Intel X79, X99
- 4-kanałowy kontroler pamięci
- PCI Express 3.0
- inne wersje:
  - LGA 2011-1 (luty 2014)
  - LGA 2011-v3 (sierpień 2014)



LGA 2011

Procesory Intel - LGA 2066 (Socket R4)

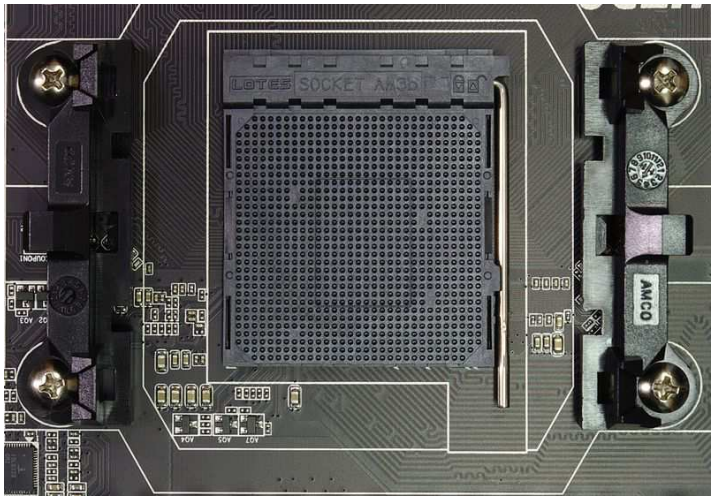
- czerwiec 2017, liczba pinów: 2066
- procesory:
  - Skylake-X
  - Kaby Lake-X
  - Skylake-SP
  - Cascade Lake-X
- chipsety: Intel X299



LGA 2066

Procesory AMD - Socket AM3+

- PGA-ZIF - nóżki znajdują się na procesorze
- 2011 rok, liczba kontaktów: 942
- mikroarchitektura Bulldozer
- procesory: Athlon II, Phenom II, FX

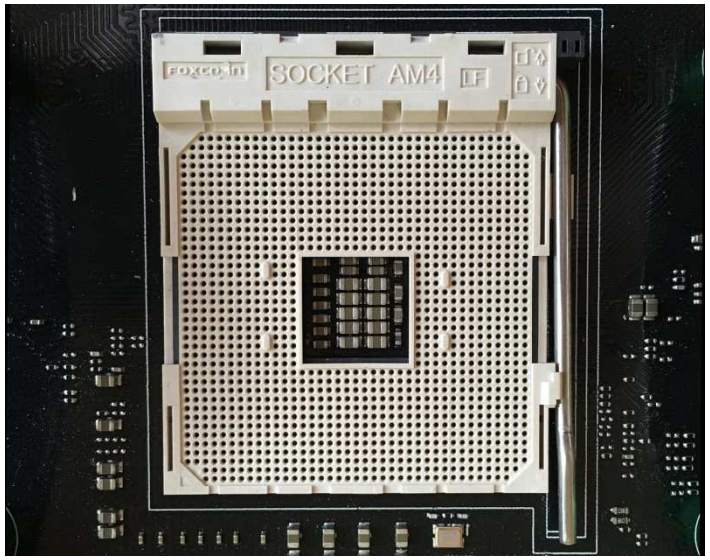


Socket AM3+

AMD Phenom II

Procesory AMD - Socket AM4

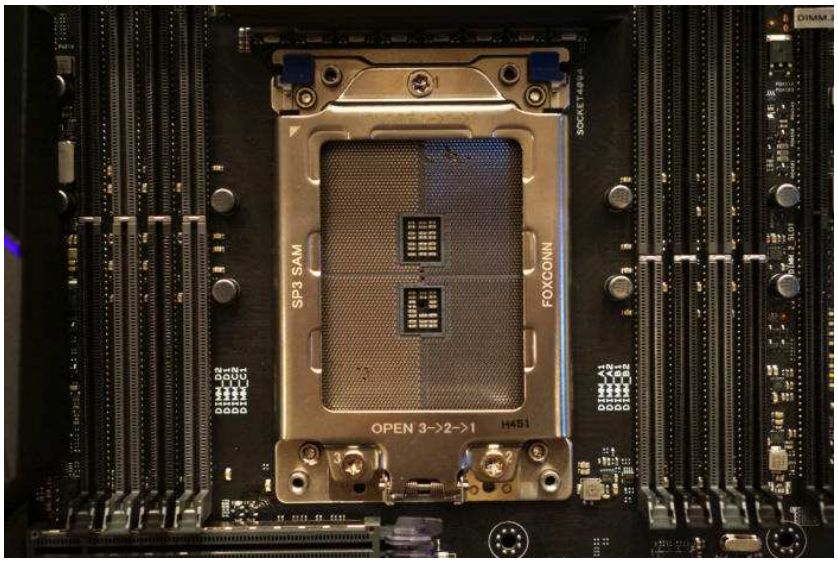
- 2017 rok, liczba kontaktów: 1331
- mikroarchitektura: Zen, Excavator
- obsługa: DDR4 Memory, PCIe Gen 3, USB 3.1 Gen2 10Gbps, NVMe
- procesory: Bristol Ridge, Summit Ridge, Raven Ridge



Socket AM4

Procesory AMD - Socket TR4

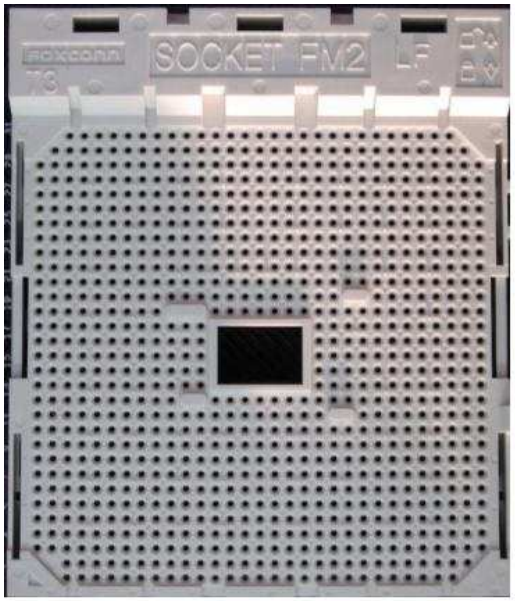
- 10 sierpnia 2017 r., liczba pinów: 4094
- inne nazwy: Socket Threadripper 4, Socket SP3r2
- procesory: Zen, Ryzen Threadripper
- pierwsza podstawka LGA przeznaczona na rynek konsumencki



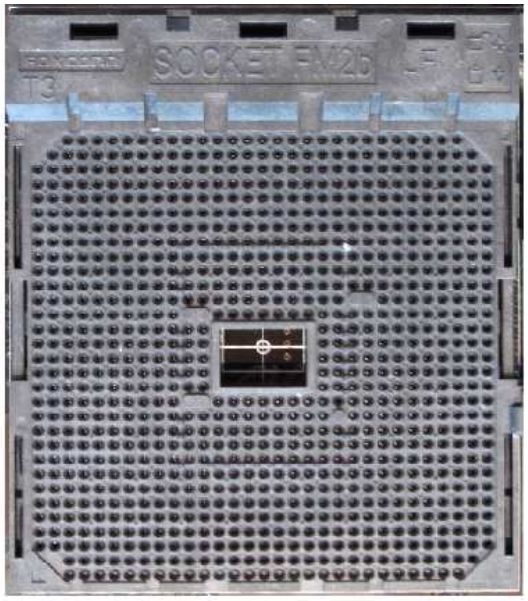
Socket AM4

Procesory AMD - Socket FM2/FM2+

- FM2: wrzesień 2012, liczba kontaktów: 904, AMD Trinity
- FM2+: 2013, liczba kontaktów: 906, AMD Kaveri
- przeznaczenie: APU (Accelerated Processing Unit) drugiej generacji
- APU - połączenie tradycyjnego procesora x86 z proc. graficznym



Socket FM2



Socket FM2+

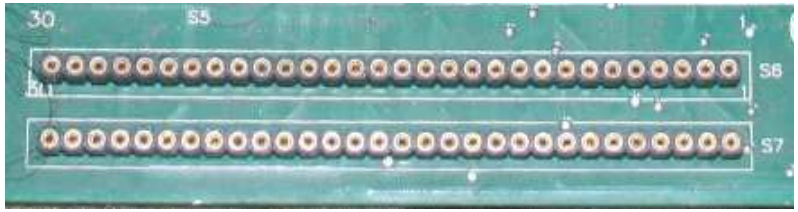
Moduły pamięci

DIP

- Dual In-line Package
- zastosowanie: XT, AT
- rok: 1981

SIPP

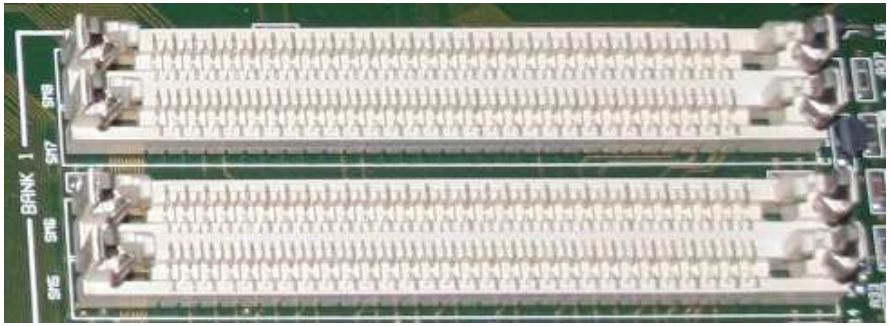
- Single In-line Pin Package
- liczba pinów: 30
- zastosowanie: AT, 286, 386
- rok: 1983



Moduły pamięci

SIMM (30-pins)

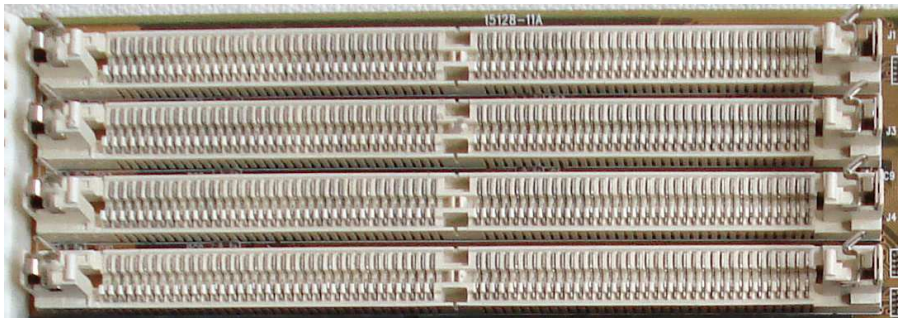
- Single Inline Memory Module
- liczba styków: 30 (te same styki po obu stronach modułu)
- pojemność: 256 KB, 1 MB, 4 MB, 16 MB
- zastosowanie: 286, 386, 486
- rok: 1994

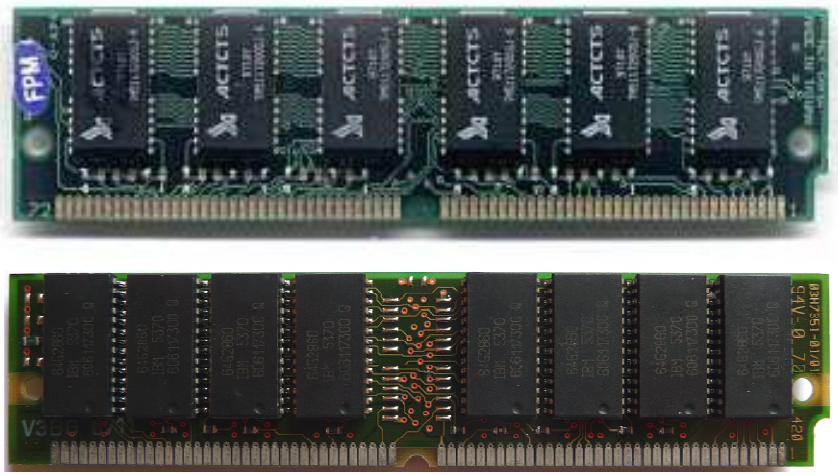


Moduły pamięci

SIMM (72-pins)

- Single Inline Memory Module
- liczba styków: 72 (te same styki po obu stronach modułu)
- pojemność [MB]: 1, 2, 4, 8, 16, 32, 64, 128
- zastosowanie: 486, Pentium, AMD K5, AMD K6
- rok: 1996





Moduły pamięci

DIMM

- Dual In-Line Memory Module
- styki po przeciwnych stronach modułu mają inne znaczenie
- najczęściej stosowane moduły DIMM:
  - 72-pinowe, stosowane w SO-DIMM (32-bitowe)
  - 144-pinowe, stosowane w SO-DIMM (64-bitowe)
  - 168-pinowe, stosowane w SDR SDRAM
  - 184-pinowe, stosowane w DDR SDRAM
  - 240-pinowe, stosowane w DDR2 SDRAM
  - 240-pinowe, stosowane w DDR3 SDRAM
  - 288-pinowe, stosowane w DDR4 SDRAM

Moduły pamięci

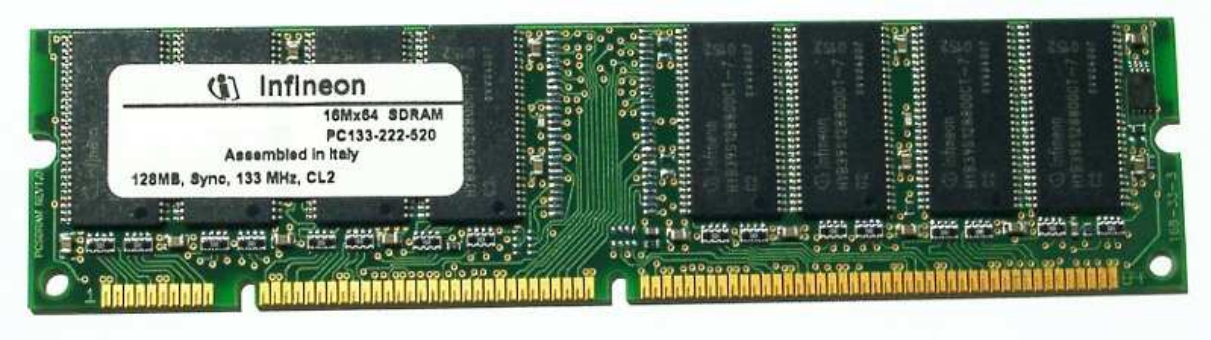
SDR SDRAM

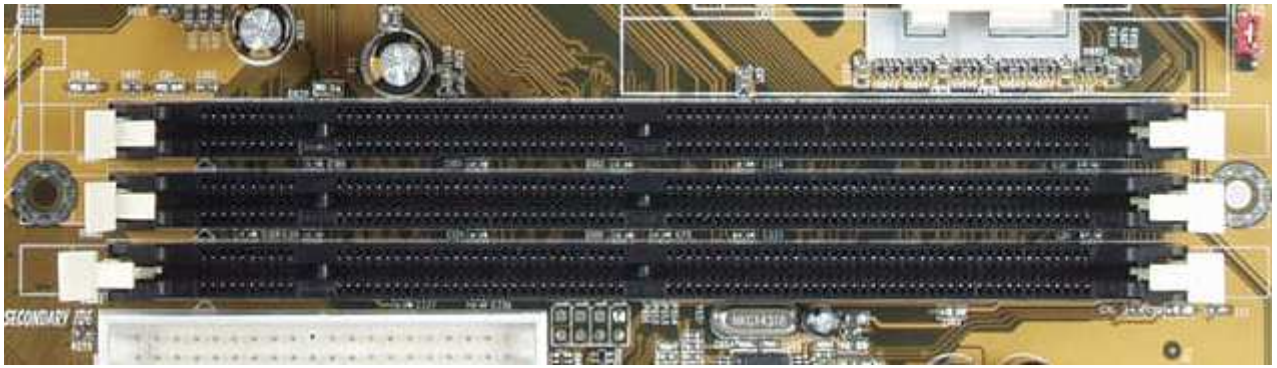
- Single Data Rate Synchronous Dynamic Random Access Memory
- liczba styków: 168
- pojemność [MB]: 16, 32, 64, 128, 256, 512
- zasilanie: 3,3 V
- zastosowanie: Pentium, Pentium II, Pentium III, Pentium IV Celeron, AMD K6

Oznaczenie	Częstotliwość	Przepustowość	Czas dostępu	Rok
PC66	66 MHz	533 MB/s	12-15 ns	1997
PC100	100 MHz	800 MB/s	8-10 ns	1998
PC133	133 MHz	1067 MB/s	7,5 ns	1999

Moduły pamięci

SDR SDRAM





Moduły pamięci

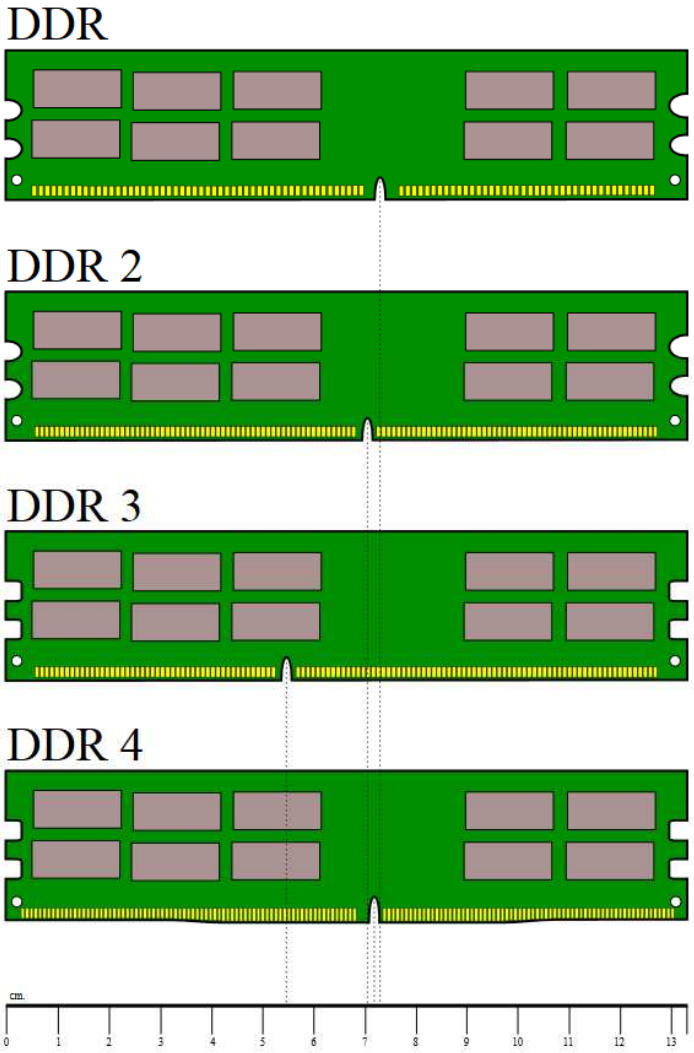
DDR SDRAM

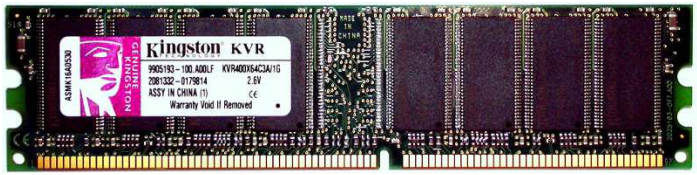
- Double Data Rate Synchronous Dynamic Random Access Memory

Typ	Piny	Zasilanie	Rok
DDR	184	2,5 V	1999
DDR2	240	1,8 V	2003
DDR3 DDR3L DDR3U	240	1,5 V 1,35 V 1,2 V	2007/2009
DDR4	288	1,2 V	2014

- DDR przesyła 2 bity w ciągu jednego taktu zegara
- DDR2 przesyła 4 bity w ciągu jednego taktu zegara

Moduły pamięci DDR - porównanie





źródło: http://en.wikipedia.org

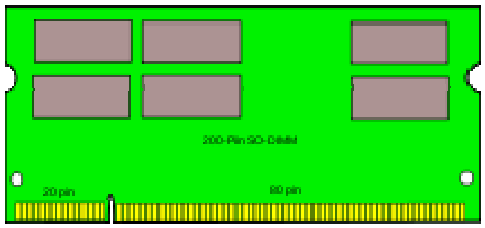
Moduły pamięci

SO-DIMM

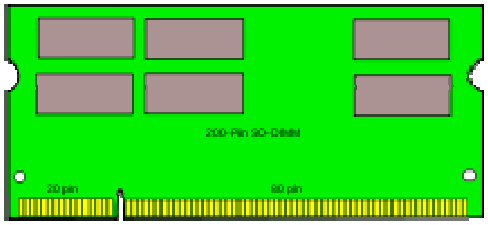
- Small Outline Dual In-line Memory Module
- stosowane głównie w laptopach, drukarkach, ruterach
- najczęściej stosowane moduły:
  - 72-pinowe (32-bitowe)
  - 100-pinowe
  - 144-pinowe (64-bitowe)
  - 200-pinowe pamięci DDR SDRAM i DDR-II SDRAM
  - 204-pinowe DDR3
  - 260-pinowe DDR4

Moduły pamięci SO-DIMM - porównanie

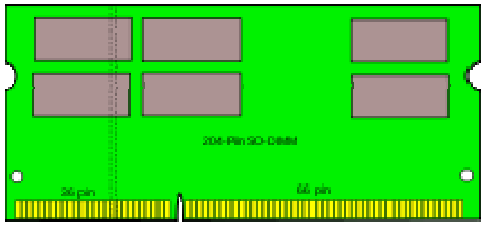
SO-DIMM DDR



SO-DIMM DDR 2



SO-DIMM DDR 3



Obudowa komputera - podział (wymiary, kształt)

Desktop

Mini-ITX

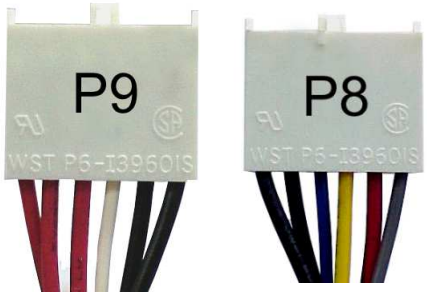
Mini tower

Midi tower

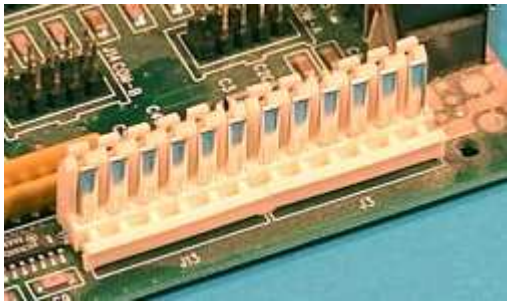
Big tower

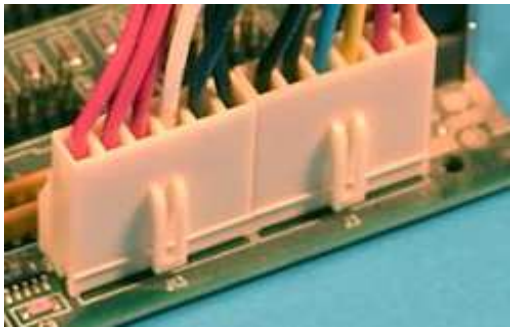
Obudowa komputera - architektura AT

Zasilacz AT



P9/P8 connectors



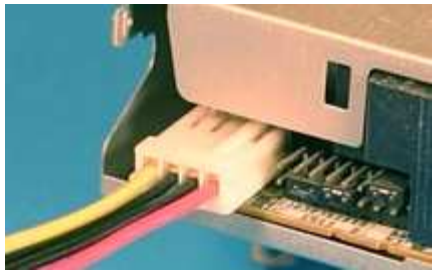


źródło:

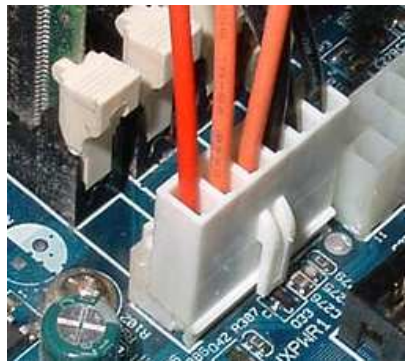
http://www.playtool.com/pages/psuconnectors/connectors.html

Obudowa komputera - architektura AT

4-pin Molex connector



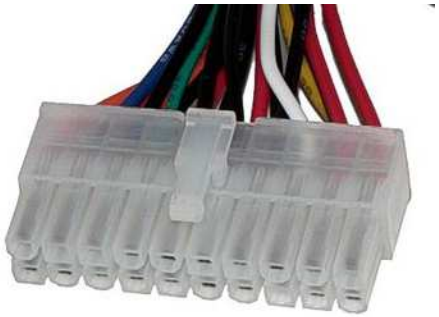
4-pin Berg connectors



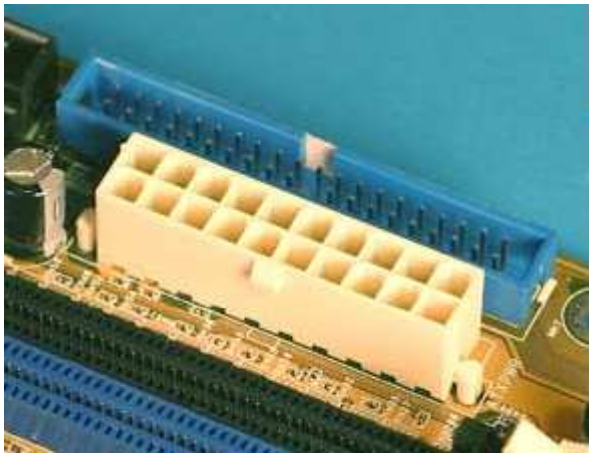
6-pin Auxiliary
Power Connector

Obudowa komputera - architektura ATX

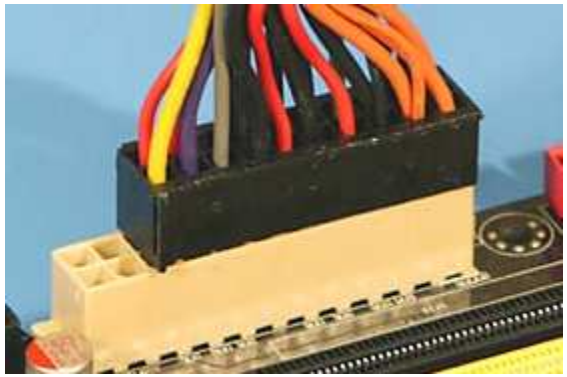
Zasilacz ATX

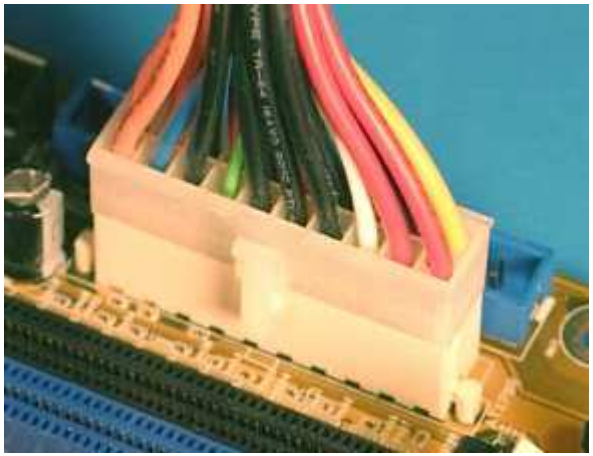


20-pin ATX power connector



Złącze 20-pinowe można włożyć do gniazda 24-pinowego





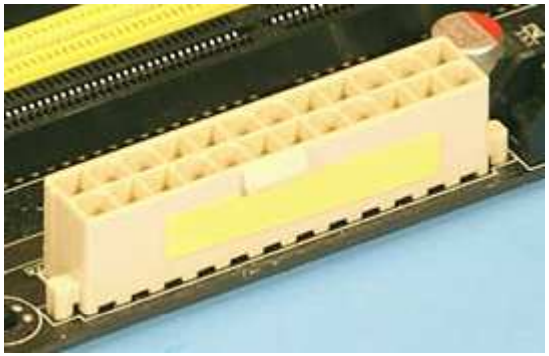
źródło:

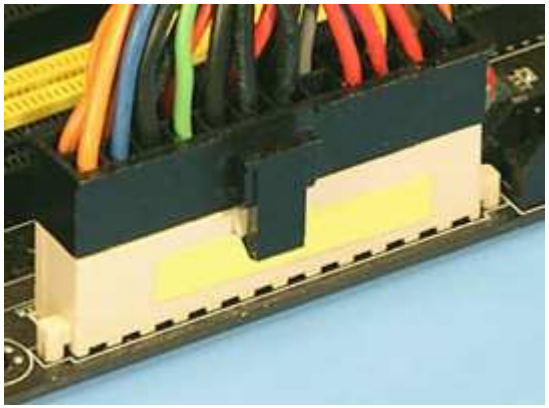
http://www.playtool.com/pages/psuconnectors/connectors.html

Obudowa komputera - architektura ATX

Zasilacz ATX

24-pin ATX power connector





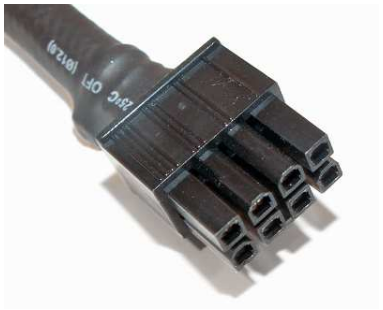
Złącze 24-pinowe można włożyć do gniazda 20-pinowego

źródło:

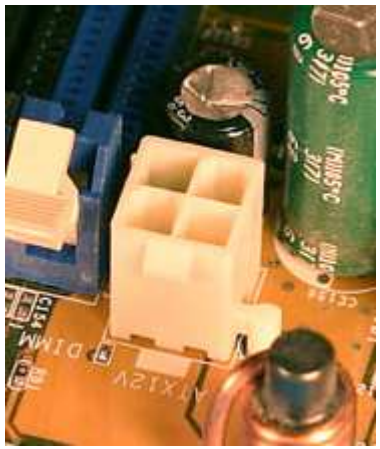
http://www.playtool.com/pages/psuconnectors/connectors.html

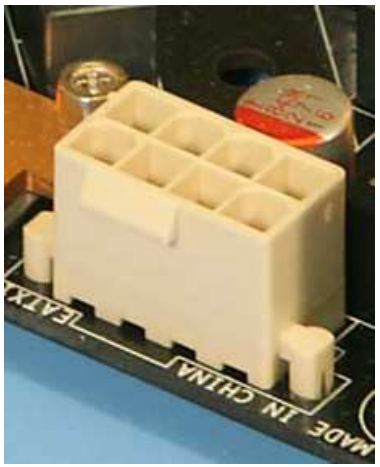
Obudowa komputera - architektura ATX

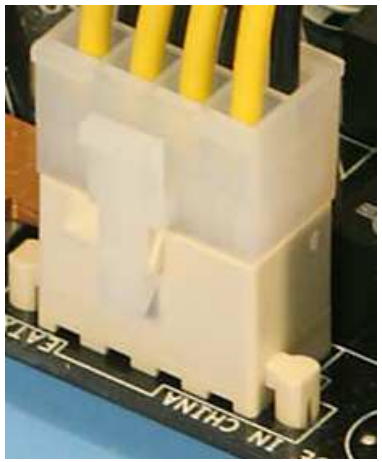
4-pin ATX 12 V



8-pin ATX 12 V

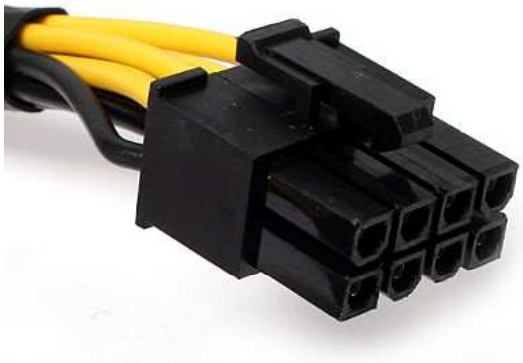






Obudowa komputera - architektura ATX

6-pin PCI Express



8-pin PCI Express

Serial ATA power connector

4-pin Berg connector

4-pin Molex connector

Koniec wykładu nr 4/5

Dziękuję za uwagę!

(następny wykład: 20.04.2018)