



Fundusze Europejskie
Wiedza Edukacja Rozwój



Rzeczpospolita
Polska

Unia Europejska
Europejski Fundusz Społeczny



Wydział Elektryczny
Katedra Elektrotechniki, Energoelektroniki i Elektroenergetyki

Materiały do wykładu z przedmiotu:
Informatyka
Kod: EDS1B1007

WYKŁAD NR 7

Opracował: dr inż. Jarosław Forenc

Białystok 2020

Materiały zostały opracowane w ramach projektu „PB2020 - Zintegrowany Program Rozwoju Politechniki Białostockiej” realizowanego w ramach Działania 3.5 Programu Operacyjnego Wiedza, Edukacja, Rozwój 2014-2020 współfinansowanego ze środków Europejskiego Funduszu Społecznego.

Plan wykładu nr 7

- Algorytmy komputerowe
 - rekurencja, złożoność obliczeniowa
 - algorytmy sortowania
- Klasyfikacja systemów komputerowych (Flynna)
- Architektura von Neumanna i architektura harwardzka
- Struktura i funkcjonowanie komputera
 - procesor, rozkazy, przerwania, magistrala
 - pamięć komputerowa, pamięć podręczna
- Systemy operacyjne
 - definicje systemu operacyjnego
 - zarządzanie: procesami, dyskowymi operacjami we-wy, pamięcią operacyjną

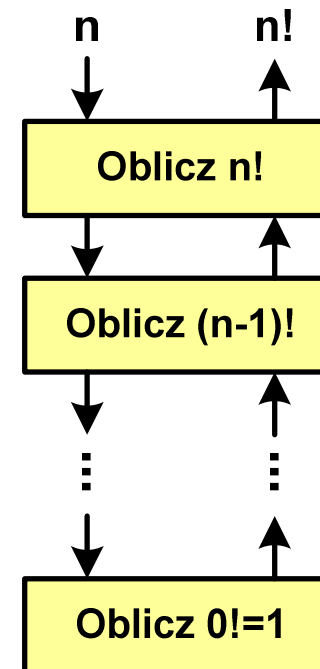
Rekurencja

- **Rekurencja** lub **rekursja** - jest to odwoływanie się funkcji lub definicji do samej siebie
- Rozwiązanie danego problemu wyraża się za pomocą rozwiązań tego samego problemu, ale dla danych o mniejszych rozmiarach
- W matematyce mechanizm rekurencji stosowany jest do definiowania lub opisywania algorytmów

- Silnia:

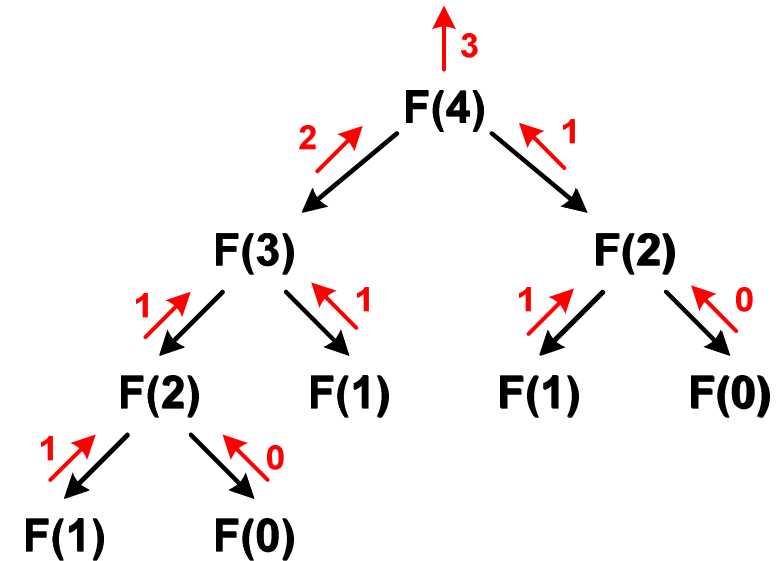
$$n! = \begin{cases} 1 & \text{dla } n = 0 \\ n(n-1)! & \text{dla } n \geq 1 \end{cases}$$

```
int silnia(int n)
{
    return n==0 ? 1 : n*silnia(n-1);
}
```



Rekurencja - ciąg Fibonacciego

$$F_n = \begin{cases} 0 & \text{dla } n = 0 \\ 1 & \text{dla } n = 1 \\ F_{n-1} + F_{n-2} & \text{dla } n > 1 \end{cases}$$



```
int F(int n)
{
    if (n==0) return 0;
    if (n==1) return 1;
    return F(n-1) + F(n-2);
}
```

Złożoność obliczeniowa

- W celu rozwiązania danego problemu obliczeniowego szukamy algorytmu najbardziej **efektywnego** czyli:
 - najszybszego (najkrótszy czas otrzymania wyniku)
 - o możliwie małym zapotrzebowaniu na pamięć
- **Problem:** Jak ocenić, który z dwóch różnych algorytmów rozwiązujących to samo zadanie jest efektywniejszy?
- Do oceny efektywności służy **złożoność obliczeniowa algorytmu** (**koszt algorytmu**) czyli ilość zasobów potrzebnych do jego działania (czas, pamięć)
- Miarą złożoności **czasowej** jest liczba podstawowych (dominujących) operacji (porównanie, podstawienie, operacja arytmetyczna) - pozostałe operacje są pomijane
- Miarą złożoności **pamięciowej** jest liczba wykorzystanych komórek pamięci (bajty lub liczba zmiennych określonego typu)

Złożoność obliczeniowa

- Złożoność obliczeniowa algorytmu jest **funkcją** opisującą zależność między **liczbą danych** a **liczbą operacji** wykonywanych przez ten algorytm
- W praktyce stosuje się oszacowanie powyższej funkcji - są to tzw. notacje (klasy złożoności):
 - O (duże O) - najbardziej popularna
 - Ω (omega)
 - Θ (theta)

Notacja O („duże O ”)

- Wyraża złożoność matematyczną algorytmu
- Do wyznaczenia złożoności bierze się pod uwagę liczbę dominujących operacji wykonywanych w algorytmie
- Przykład zapisu: $O(n^2)$
 - po literze O występuje wyrażenie w nawiasach zawierające literę n , która oznacza liczbę elementów, na których działa algorytm
- W funkcji opisującej złożoność bierze się pod uwagę tylko najistotniejszy składnik, np.

$$f(n) = n^2 + 2n \rightarrow O(n^2) \quad f(n) = n^2 + n - 5 \rightarrow O(n^2)$$

- W powyższych przykładach dla dużego n wpływ składnika liniowego i stałego na wartość funkcji jest nieistotny w porównaniu ze składnikiem głównym n^2

Notacja O („duże O”)

- Porównanie najczęściej występujących złożoności:

Elementy (n)	$O(\log n)$	$O(n)$	$O(n \log n)$	$O(n^2)$	$O(n^3)$	$O(2^n)$
10	3	10	33	100	1 000	1024
100	7	100	664	10 000	1 000 000	$1,27 \cdot 10^{30}$
1 000	10	1 000	9 966	1 000 000	10^9	$1,07 \cdot 10^{301}$
10 000	13	10 000	132 877	10^8	10^{12}	$1,99 \cdot 10^{3010}$

- $O(\log n)$ - logarytmiczna (np. przeszukiwanie binarne)
- $O(n)$ - liniowa (np. porównywanie łańcuchów znaków)
- $O(n \log n)$ - liniowo-logarytmiczna (np. sortowanie szybkie)
- $O(n^2)$ - kwadratowa (np. proste algorytmy sortowania)
- $O(n^3)$ - sześcienna (np. mnożenie macierzy)
- $O(2^n)$ - wykładnicza (np. problem komiwojażera)

Sortowanie

- **Sortowanie** polega na **uporządkowaniu** zbioru danych względem pewnych cech charakterystycznych każdego elementu tego zbioru (wartości każdego elementu)
- W przypadku liczb, sortowanie polega na znalezieniu kolejności liczb zgodnej z relacją \leq lub \geq

Przykład:

- Tablica nieposortowana:

6	4	5	2	3	1
---	---	---	---	---	---

- Tablica posortowana zgodnie z relacją \leq (od najmniejszej do największej liczby):

1	2	3	4	5	6
---	---	---	---	---	---

- Tablica posortowana zgodnie z relacją \geq (od największej do najmniejszej liczby):

6	5	4	3	2	1
---	---	---	---	---	---

Sortowanie

- W przypadku słów sortowanie polega na ustawieniu ich w porządku **alfabetycznym** (**leksykograficznym**)

Przykład:

- Tablica nieposortowana:

Paweł	Piotr	Adrian	Ela	Ola	Henryk
-------	-------	--------	-----	-----	--------

- Tablice posortowane:

Adrian	Ela	Henryk	Ola	Paweł	Piotr
--------	-----	--------	-----	-------	-------

Piotr	Paweł	Ola	Henryk	Ela	Adrian
-------	-------	-----	--------	-----	--------

Sortowanie

- W praktyce sortowanie sprowadza się do porządkowanie danych na podstawie porównania - porównywany element to **klucz**

Przykład:

- Tablica nieposortowana (imię, nazwisko, wiek):

Piotr	Ola	Paweł	Jan	Ela	Magda
Kowalski	Nowak	Wójcik	Kamiński	Król	Mazur
25	18	23	20	22	15

- Tablica posortowana (klucz - nazwisko):

Jan	Piotr	Ela	Magda	Ola	Paweł
Kamiński	Kowalski	Król	Mazur	Nowak	Wójcik
20	25	22	15	18	23

- Tablica posortowana (klucz - wiek):

Magda	Ola	Jan	Ela	Paweł	Piotr
Mazur	Nowak	Kamiński	Król	Wójcik	Kowalski
15	18	20	22	23	25

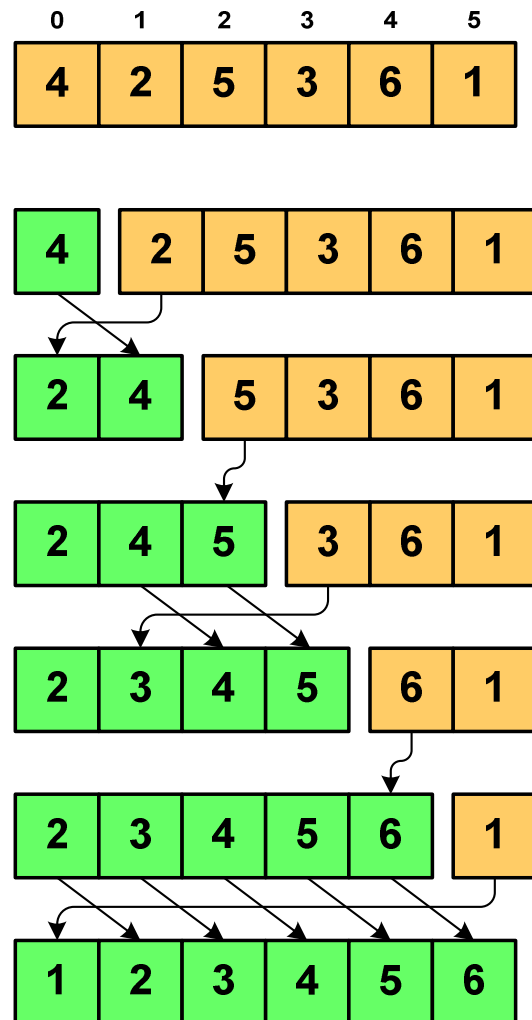
Sortowanie

- Po co stosować sortowanie?
 - posortowane elementy można szybciej zlokalizować
 - posortowane elementy można przedstawić w czytelniejszy sposób

- Przykładowe algorytmy sortowania
 - proste wstawianie (insertion sort)
 - proste wybieranie (selection sort)
 - bąbelkowe (bubble sort)
 - szybkie (quick sort)
 - przez scalanie (merge sort)
 - kubełkowe / przez zliczanie (bucket sort)

Proste wstawianie (insertion sort)

Przykład:

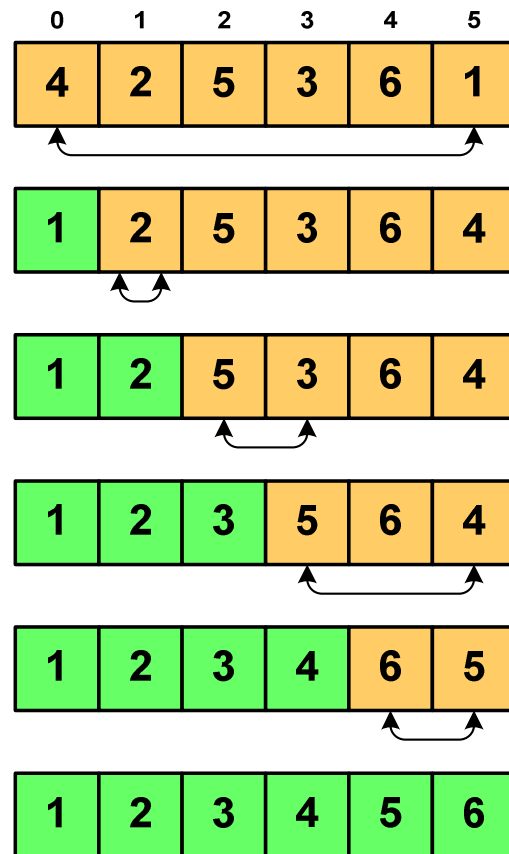


Program w języku C:

```
int main(void)
{
    int tab[N], i, j, tmp;
    // ...
    for (i=1; i<N; i++)
    {
        j=i;
        tmp=tab[i];
        while (tab[j-1]>tmp && j>0)
        {
            tab[j]=tab[j-1];
            j--;
        }
        tab[j]=tmp;
    }
}
```

Proste wybieranie (selection sort)

Przykład:



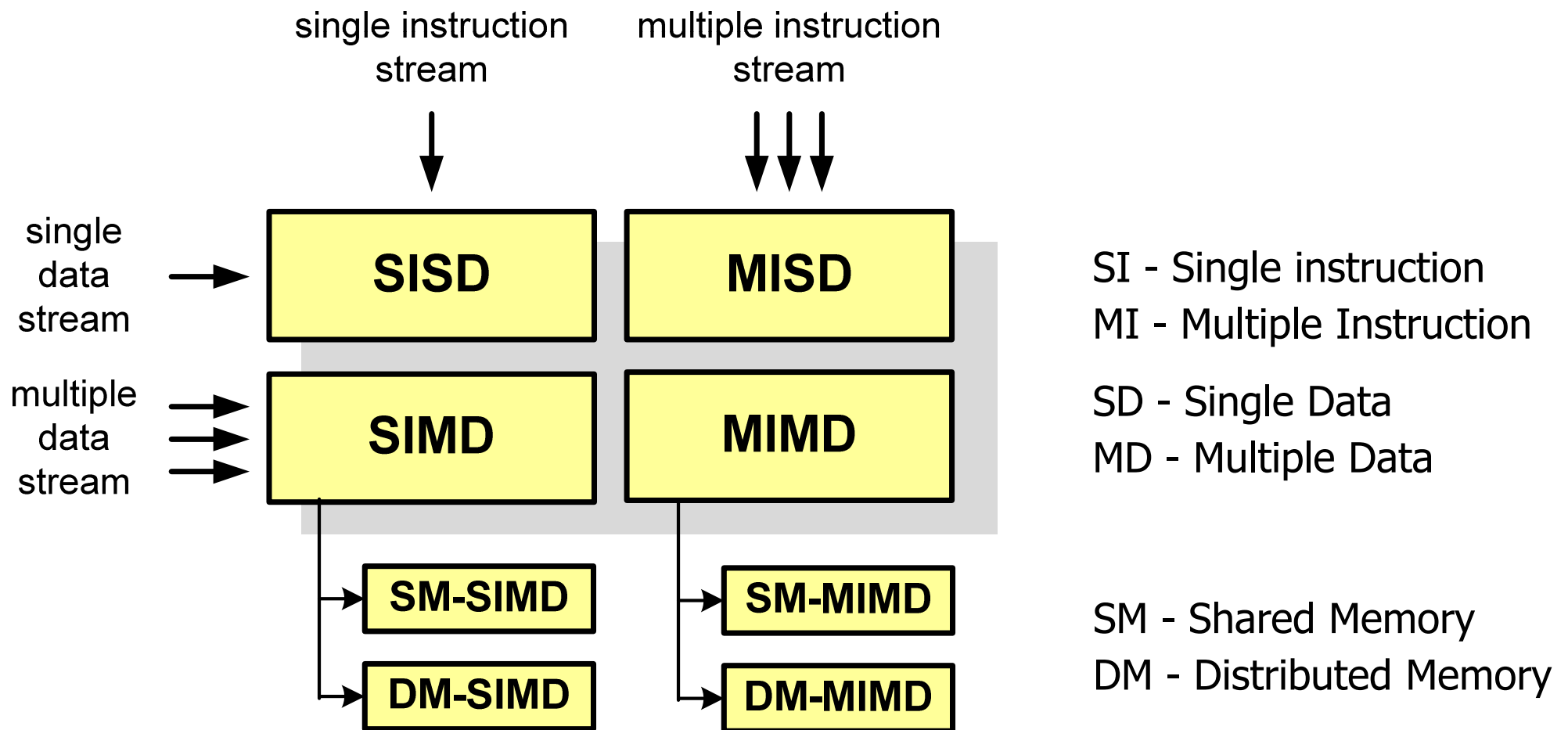
Program w języku C:

```
int main(void)
{
    int tab[N], i, j, k, tmp;
    // ...
    for (i=0; i<N-1; i++)
    {
        k=i;
        for (j=i+1; j<N; j++)
            if (tab[k]>=tab[j])
                k = j;
        tmp = tab[i];
        tab[i] = tab[k];
        tab[k] = tmp;
    }
}
```


Klasyfikacja systemów komputerowych

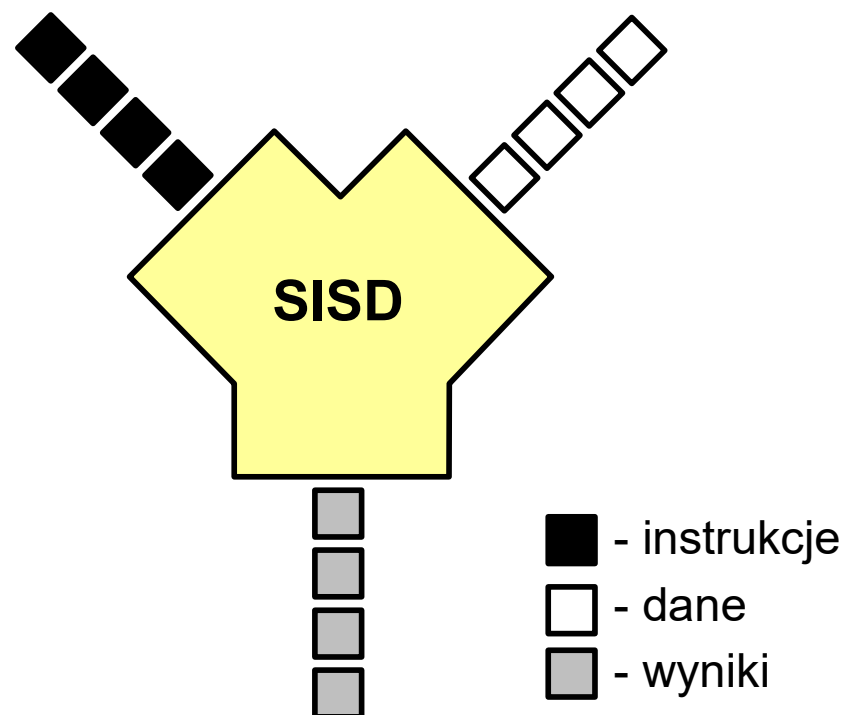
- **Taksonomia Flynna** - pierwsza, najbardziej ogólna klasyfikacja architektur komputerowych (1972):
 - Flynn M.J.: „Some Computer Organizations and Their Effectiveness”, IEEE Transactions on Computers, Vol. C-21, No 9, 1972.
- Opiera się na liczbie przetwarzanych strumieni rozkazów i strumieni danych:
 - **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



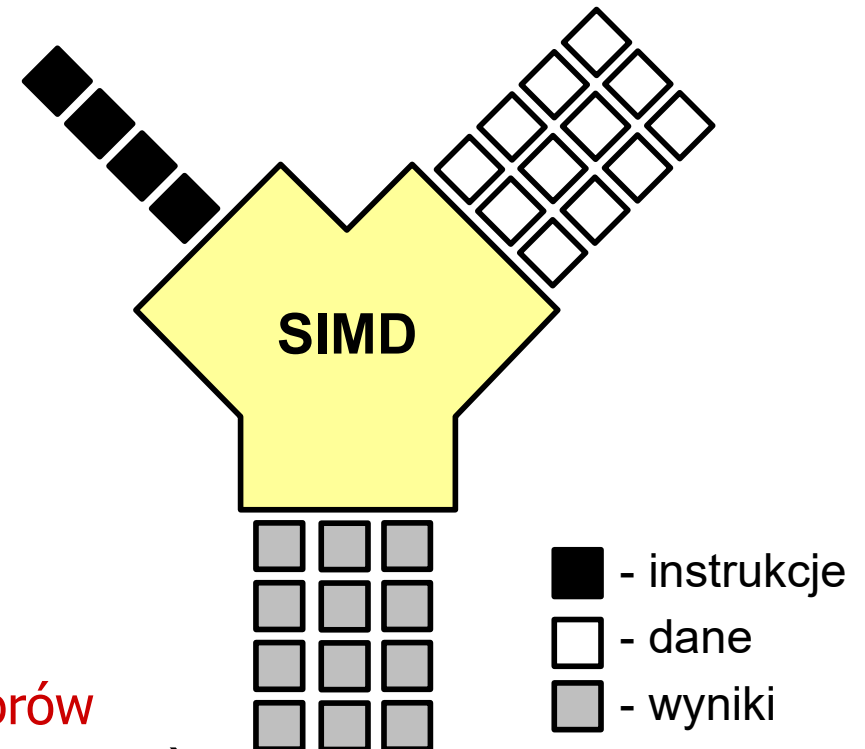
SISD (Single Instruction, Single Data)

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



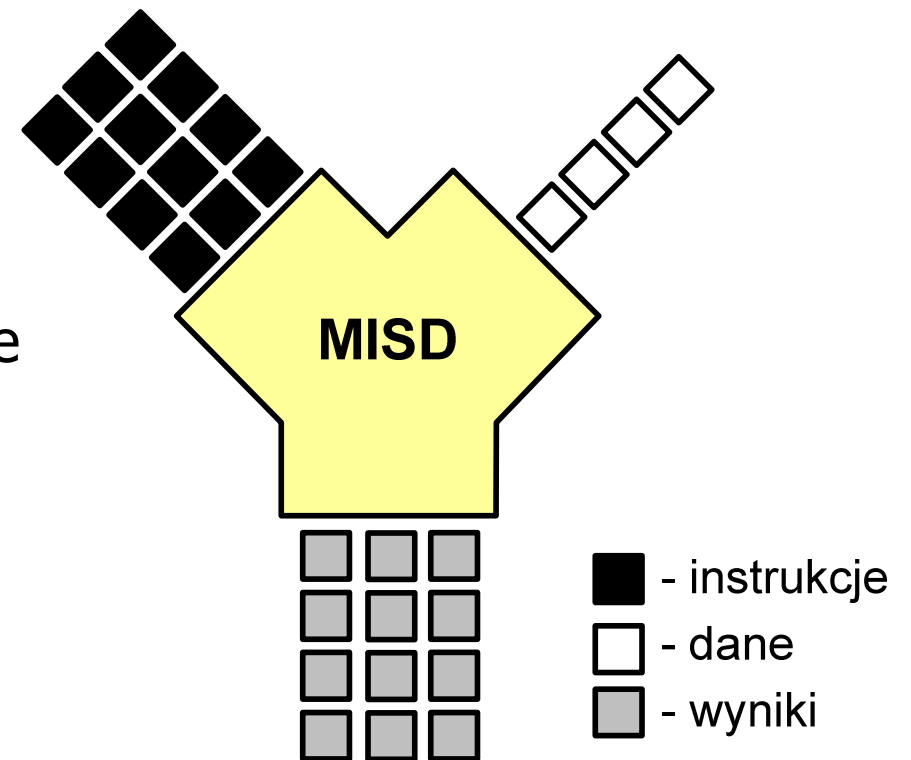
SIMD (Single Instruction, Multiple Data)

- 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)



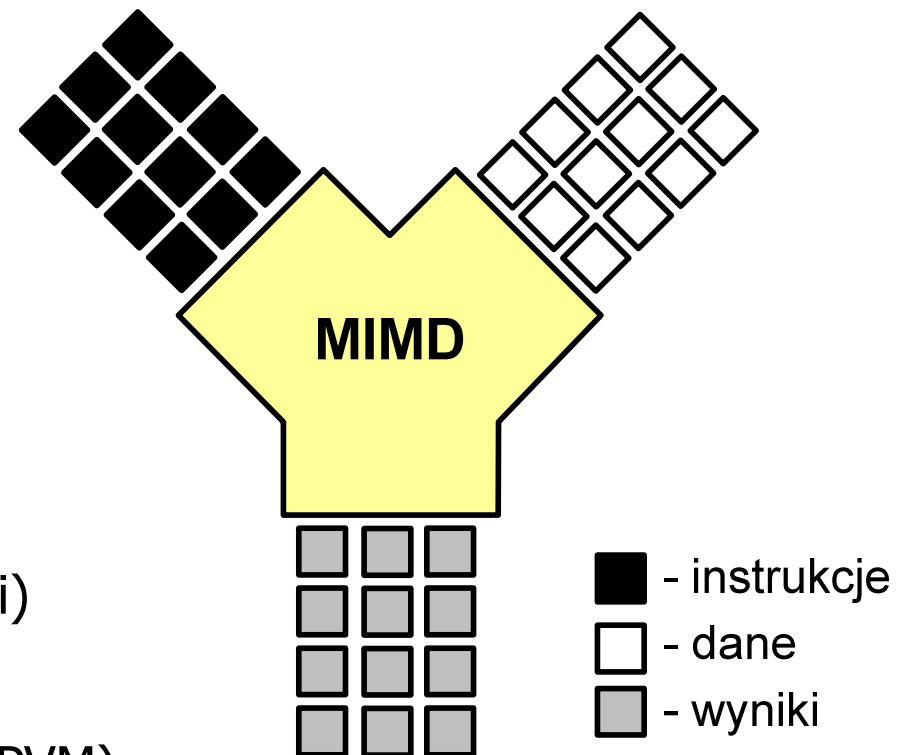
MISD (Multiple Instruction, Single Data)

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



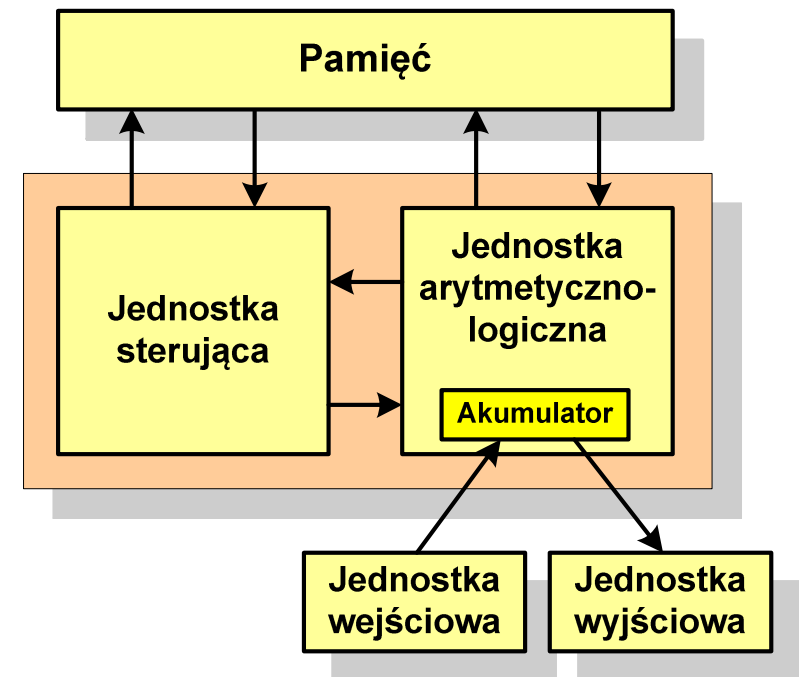
MIMD (Multiple Instruction, Multiple Data)

- Równoległe wykonywanych jest wiele programów, z których każdy przetwarza własne strumienie danych
- Podział:
 - SM-MIMD (Shared Memory):
 - **wieloprocesory** (np. komputery z procesorami wielordzeniowymi)
 - DM-MIMD (Distributed Memory):
 - **wielokomputery** (biblioteki MPI, PVM)
 - **klastry**
 - **gridy**



Architektura von Neumanna

- Rodzaj architektury komputera, opisanej w 1945 roku przez matematyka Johna von Neumanna
- Inne 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

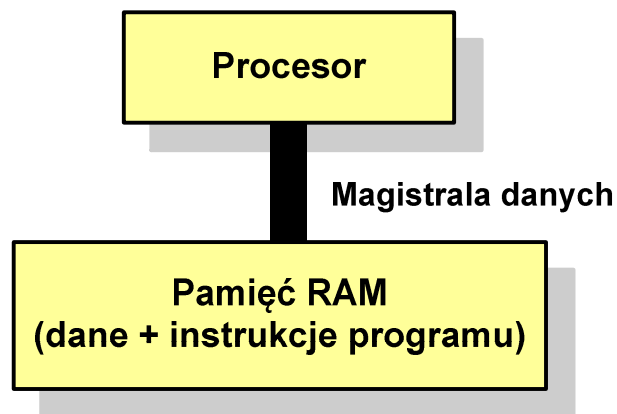
- Informacje przechowywane są w komórkach pamięci (**cell**) o jednakowym rozmiarze, każda komórka ma numer - **adres**
- **Dane oraz instrukcje programu (rozказы) zakodowane są za pomocą liczb i przechowywane w tej samej pamięci**
- Dane i instrukcje czytane są przy wykorzystaniu **tej samej magistrali**
- 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

Architektura harwardzka

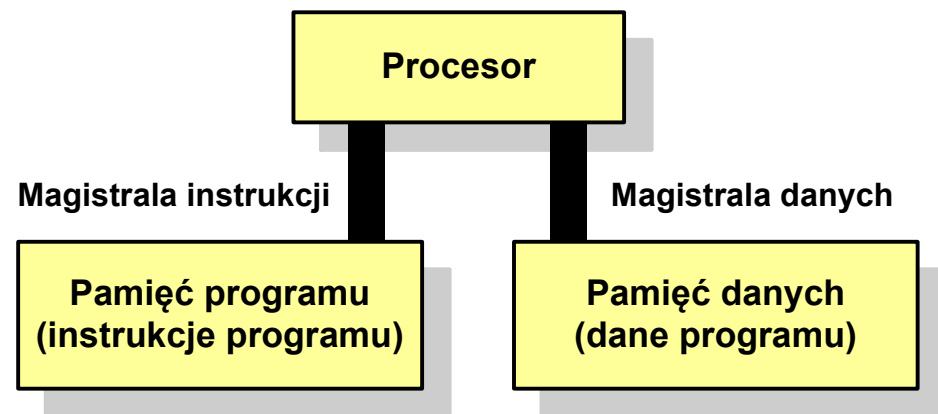
- Nazwa architektury pochodzi od komputera **Harward Mark I:**
 - zaprojektowany przez Howarda Aikena
 - pamięć instrukcji - taśma dziurkowana,
pamięć danych - elektromechaniczne liczniki
- Architektura komputera, w której **pamięć danych jest oddzielona od pamięci instrukcji**
- Pamięci danych i instrukcji mogą różnić się:
 - technologią wykonania
 - strukturą adresowania
 - długością słowa
- **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

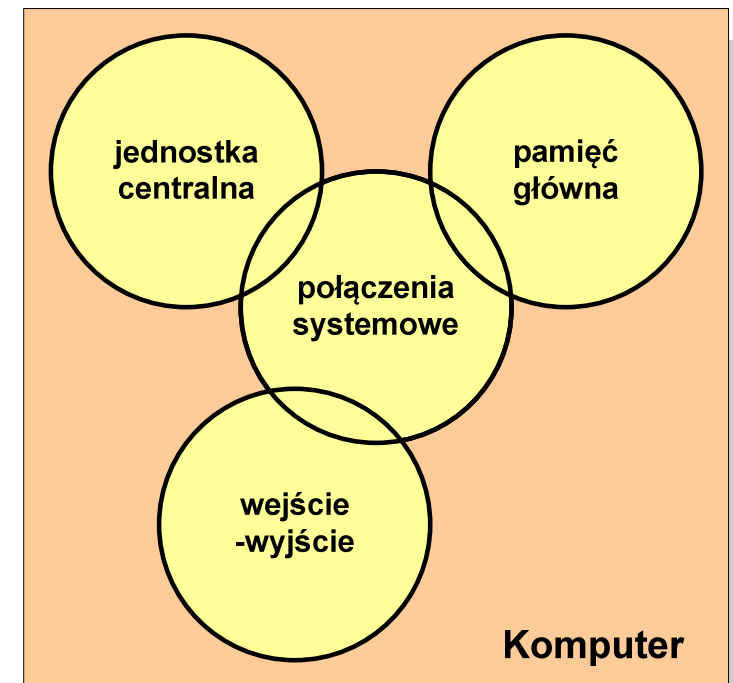


Architektura harwardzka

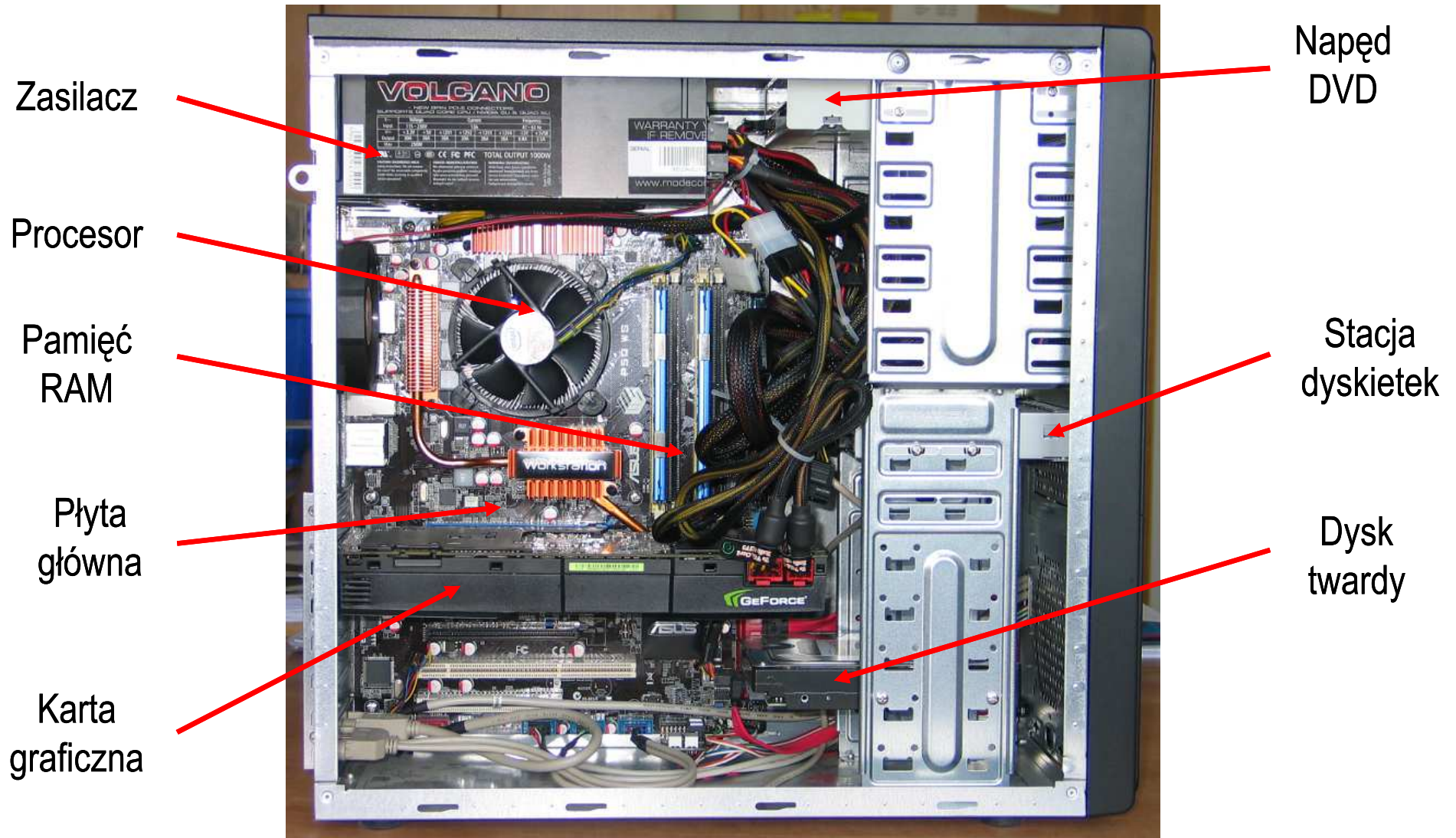
- Zmodyfikowana architektura harwardzka:
 - oddzielone pamięci danych i rozkazów, lecz wykorzystujące wspólną magistralę

Ogólna struktura systemu komputerowego

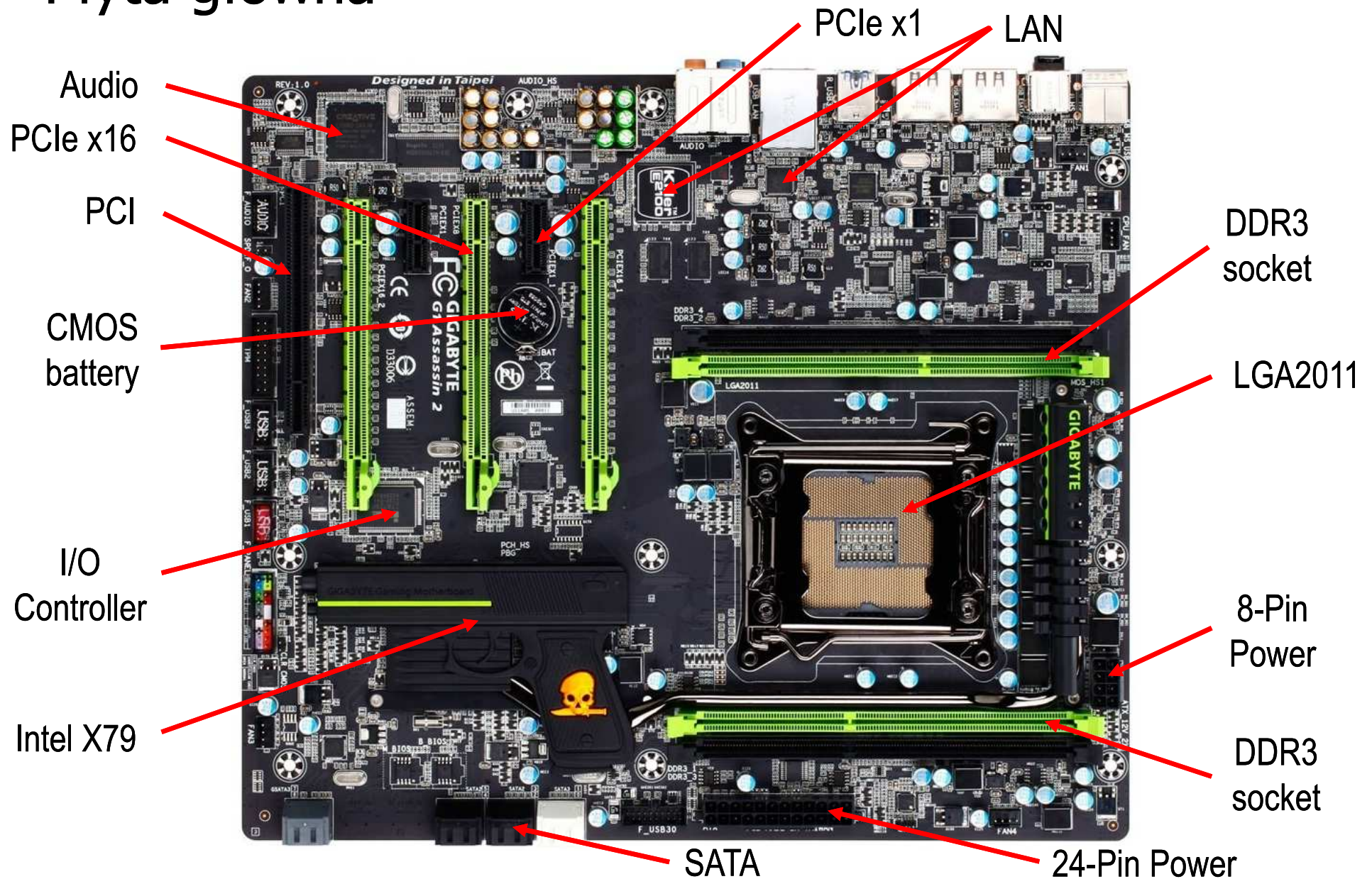
- Komputer tworzą cztery główne składniki:
 - **procesor** (jednostka centralna, CPU)
- steruje działaniem komputera
i realizuje przetwarzanie danych
 - **pamięć główna** - przechowuje dane
 - **wejście-wyjście** - przenosi dane
między komputerem a jego
otoczeniem zewnętrznym
 - **połączenia systemu** - mechanizmy
zapewniające komunikację między
składnikami systemu



Jednostka centralna

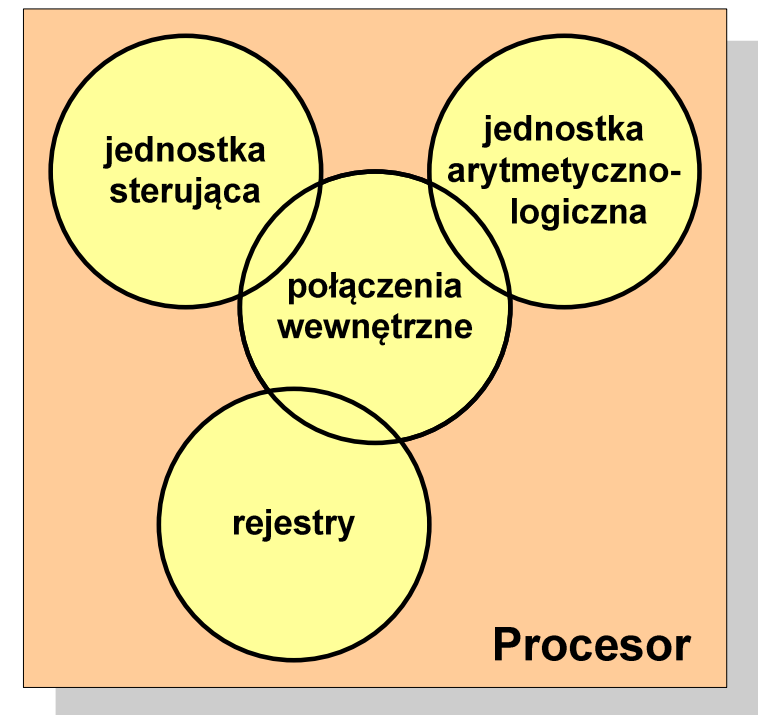


Płyta główna



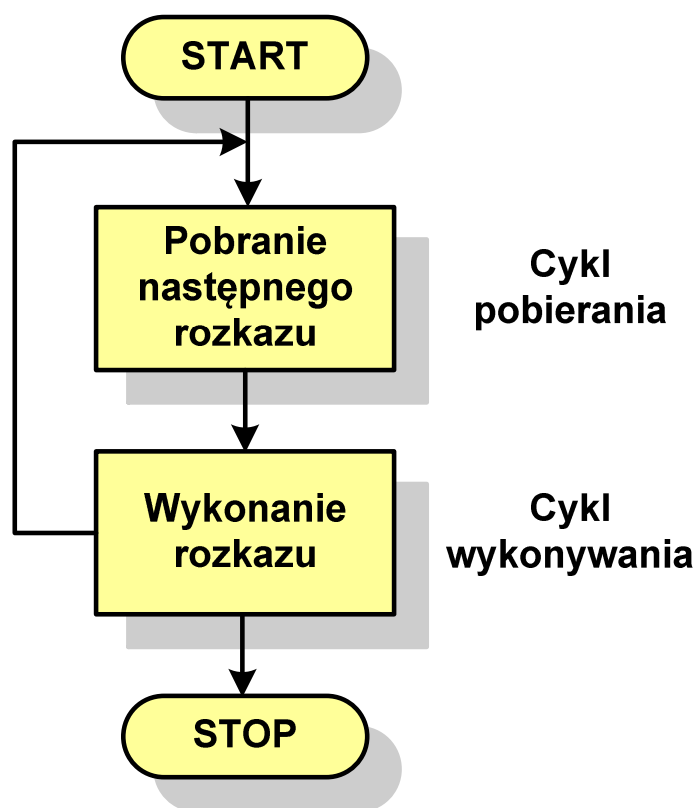
Ogólna struktura procesora

- Główne składniki strukturalne procesora to:
 - **jednostka sterująca** - steruje działaniem procesora i pośrednio całego komputera
 - **jednostka arytmetyczno-logiczna (ALU)** - realizuje przetwarzanie danych przez komputer
 - **rejstry** - realizują wewnętrzne przechowywanie danych w procesorze
 - **połączenia procesora** - wszystkie mechanizmy zapewniające komunikację między jednostką sterującą, ALU i rejestrami.



Działanie komputera

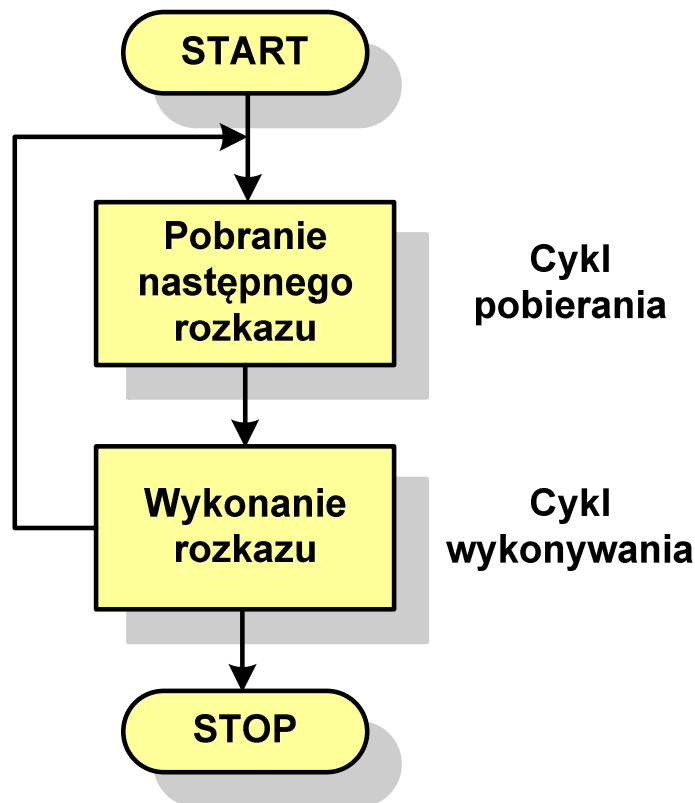
- Podstawowe zadanie komputera to wykonywanie programu
- Program składa się z rozkazów przechowywanych w pamięci
- Rozkazy są przetwarzane w dwu krokach:



- Cykl pobierania (ang. fetch):
 - odczytanie rozkazu z pamięci
 - licznik rozkazów (PC) lub wskaźnik instrukcji (IP) określa, który rozkaz ma być pobrany
 - jeśli procesor nie otrzyma innego polecenia, to inkrementuje licznik PC po każdym pobraniu rozkazu.

Działanie komputera

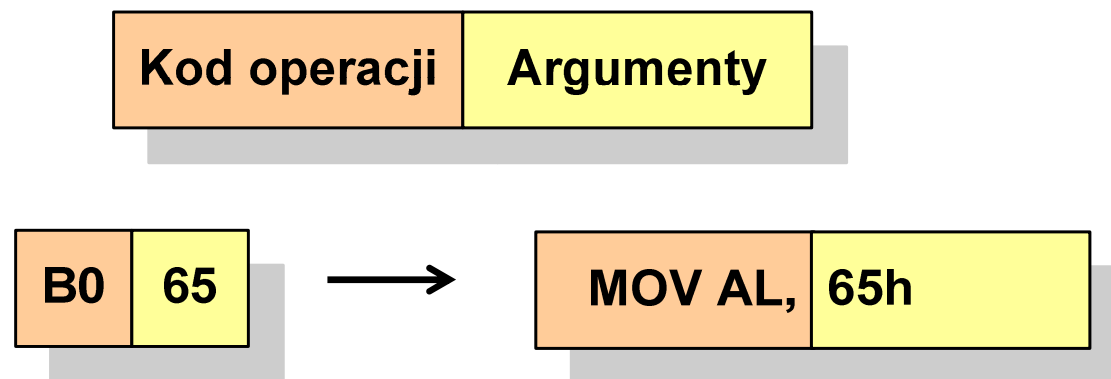
- Podstawowe zadanie komputera to wykonywanie programu
- Program składa się z rozkazów przechowywanych w pamięci
- Rozkazy są przetwarzane w dwu krokach:



- Cykl wykonywania (ang. execution):
 - pobrany rozkaz jest umieszczany w rejestrze rozkazu (IR)
 - rozkaz określa działania, które ma podjąć procesor
 - procesor interpretuje rozkaz i przeprowadza wymagane operacje.

Działanie komputera

- Rozkaz:
 - przechowywany jest w postaci **binarnej**
 - ma określony **format**
 - używa określonego **trybu adresowania**
- **Format** - sposób rozmieszczenia informacji w kodzie rozkazu
- Rozkaz zawiera:
 - **kod operacji** (rodzaj wykonywanej operacji)
 - **argumenty** (lub adresy argumentów) wykonywanych operacji



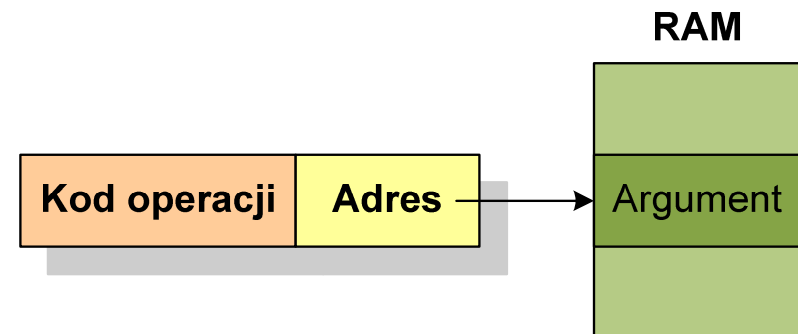
Działanie komputera

- **Tryb adresowania** - sposób określania miejsca przechowywania argumentów rozkazu (operandów)
- Przykładowe rodzaje adresowania:

- **natychmiastowe** - argument znajduje się w kodzie rozkazu



- **bezpośrednie** - kod rozkazu zawiera adres komórki pamięci, w której znajduje się argument



- **rejestrowe** - kod rozkazu zawiera oznaczenie rejestru, w którym znajduje się argument

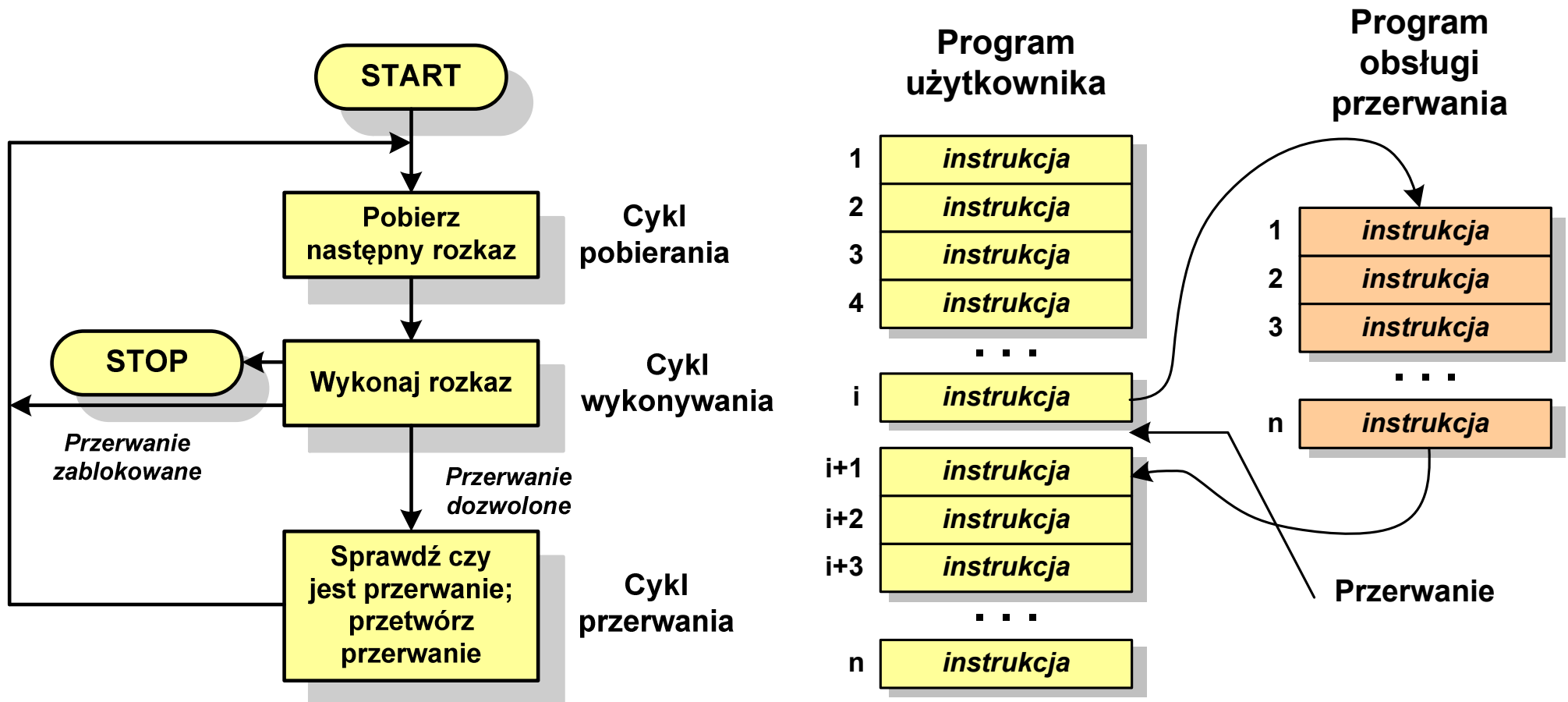


Działanie komputera - przerwania

- Wykonywanie kolejnych rozkazów przez procesor może zostać przerwane poprzez wystąpienie tzw. **przerwania** (**interrupt**)
- Przerwanie jest to **sygnał** pochodzący od sprzętu lub oprogramowania informujący procesor o wystąpieniu jakiegoś zdarzenia (np. wciśnięcie klawisza na klawiaturze)
- Bez przerwania procesor musiałby ciągle kontrolować wszystkie urządzenia zewnętrzne, np. klawiatura, port szeregowy
- Każde przerwanie posiada procedurę obsługi przerwania, która jest wykonywana w momencie jego wystąpienia
- Adresy procedur obsługi przerwania zapisane są w tablicy wektorów przerwania

Działanie komputera - przerwania

- Implementacja przerwania wymaga dodania cyklu przerwania do cyklu rozkazu



Rodzaje przerwań

■ Sprzętowe

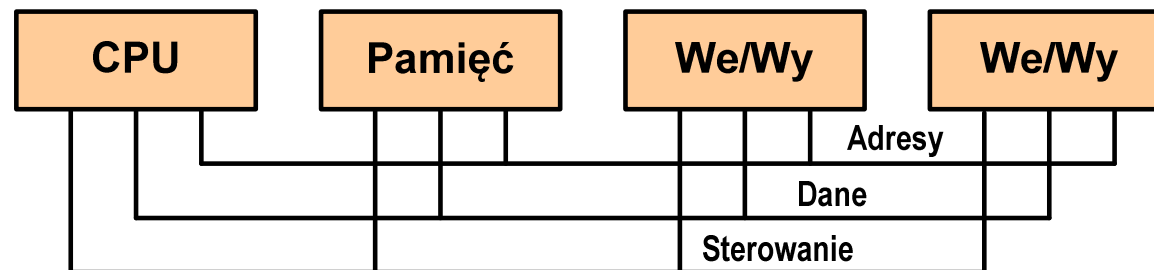
- **zewnętrzne** - sygnały pochodzące z urządzeń zewnętrznych i służące do komunikacji z nimi, np. 08H - zegar, 09h - klawiatura
- **wewnętrzne** - wywoływane przez procesor w celu zasygnalizowania sytuacji wyjątkowych (faults, traps, aborts)

■ Programowe

- instrukcje programu wywołują przerwanie - tym samym wykonywana jest procedura obsługi przerwania
- służą głównie do komunikacji z systemem operacyjnym (DOS - 21h, Windows - 2h, Linux - 80h)

Magistrala

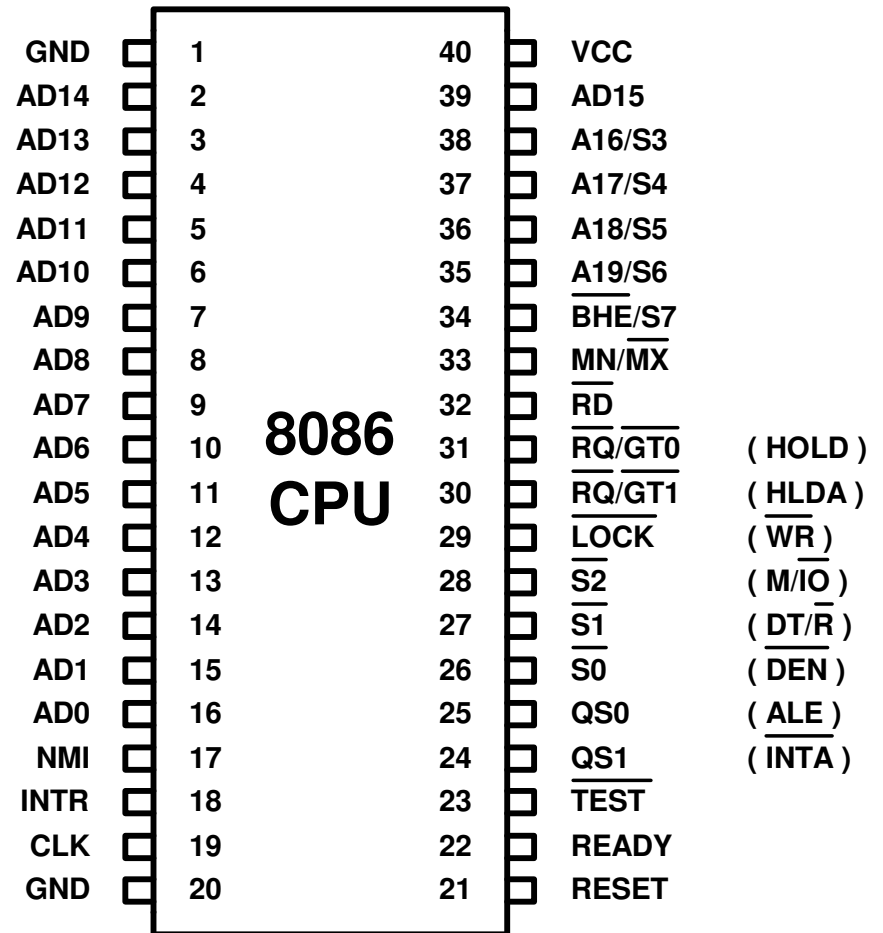
- Najczęściej stosowana struktura połączeń to **magistrala**, składająca się z wielu linii komunikacyjnych, którym przypisane jest określone znaczenie i określona funkcja



- **linie danych (szyna danych)** - przenoszą dane między modułami systemu, liczba linii określa szerokość szyny danych (8, 16, 32, 64 bity)
- **linie adresowe** - służą do określania źródła i miejsca przeznaczenia danych przesyłanych magistralą; liczba linii adresowych określa maksymalną możliwą pojemność pamięci systemu
- **linie sterowania** - służą do sterowania dostępem do linii danych i linii adresowych

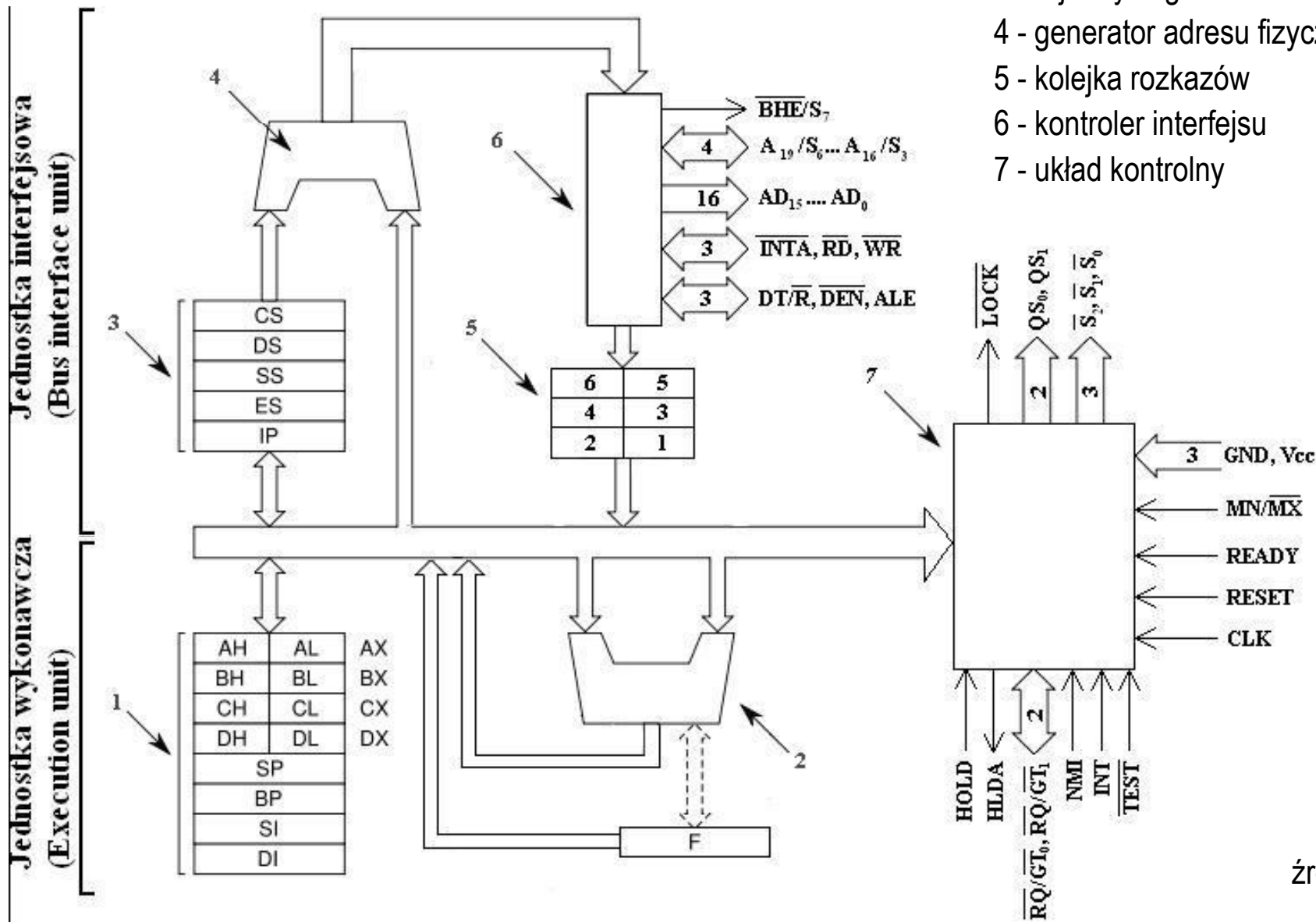
Intel 8086

- 1978 rok
- Procesor 16-bitowy
- 16-bitowa magistrala danych
- 20-bitowa magistrala adresowa
- Adresowanie do 1 MB pamięci
- Częstotliwość: 10 MHz
- Multipleksowane magistrale:
danych i adresowa
- Litografia: 3 μm



Intel 8086

- 1 - rejestry ogólnego przeznaczenia
- 2 - ALU + rejestr znaczników (flag)
- 3 - rejestry segmentowe + licznik rozkazów
- 4 - generator adresu fizycznego
- 5 - kolejka rozkazów
- 6 - kontroler interfejsu
- 7 - układ kontrolny



Półprzewodnikowa pamięć główna

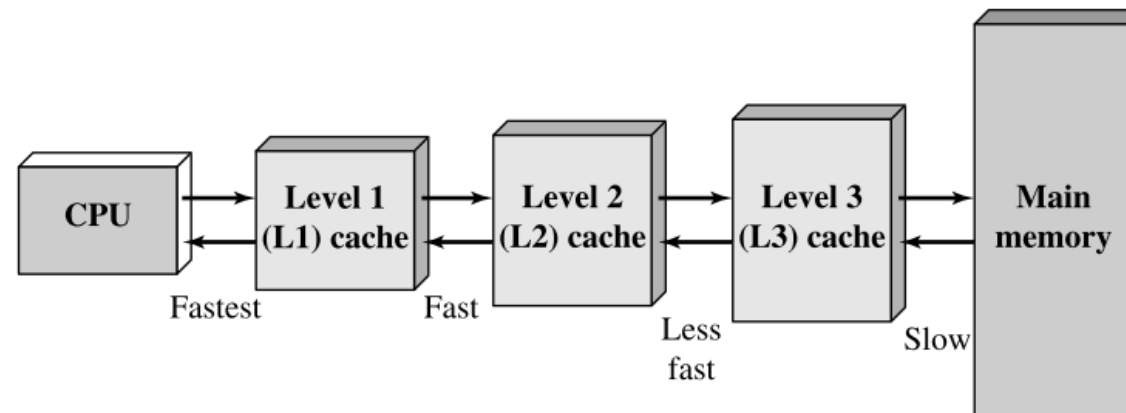
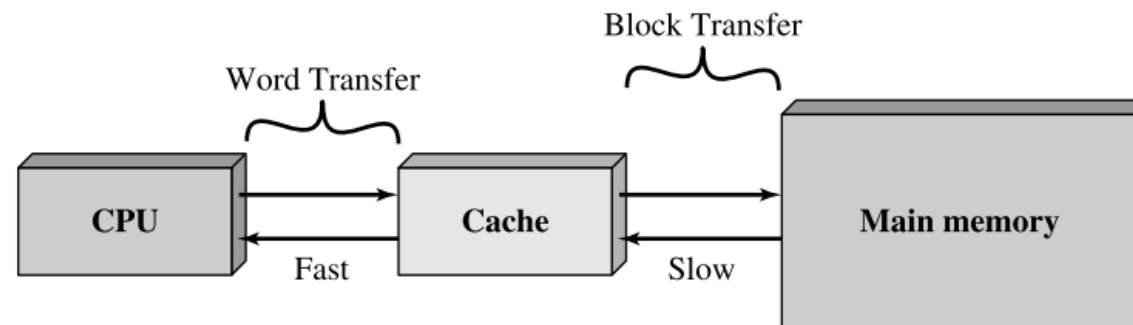
- **RAM** (Random Access Memory) - pamięć o dostępie swobodnym
 - odczyt i zapis następuje za pomocą sygnałów elektrycznych
 - pamięć ulotna - po odłączeniu zasilania dane są tracone
 - **DRAM** - pamięć dynamiczna:
 - przechowuje dane podobnie jak kondensator ładunek elektryczny
 - wymaga operacji odświeżania
 - jest mniejsza, gęściej upakowana i tańsza niż pamięć statyczna
 - stosowana jest do budowy głównej pamięci operacyjnej komputera
 - **SRAM** - pamięć statyczna:
 - przechowuje dane za pomocą przerzutnikowych konfiguracji bramek logicznych
 - nie wymaga operacji odświeżania
 - jest szybsza i droższa od pamięci dynamicznej
 - stosowana jest do budowy pamięci podręcznej

Półprzewodnikowa pamięć główna

- **ROM** (ang. Read-Only Memory) - pamięć stała
 - pamięć o dostępie swobodnym przeznaczona tylko do odczytu
 - dane są zapisywane podczas procesu wytwarzania, pamięć nieulotna
- **PROM** (ang. Programmable ROM) - programowalna pamięć ROM
 - pamięć nieulotna, może być zapisywana tylko jeden raz
 - zapis jest realizowany elektrycznie po wyprodukowaniu
- **EPROM** - pamięć wielokrotnie programowalna, kasowanie następuje przez naświetlanie promieniami UV
- **EEPROM** - pamięć kasowana i programowana na drodze elektrycznej
- **Flash** - rozwinięcie koncepcji pamięci EEPROM, możliwe kasowanie i programowanie bez wymontowywania pamięci z urządzenia

Pamięć podręczna (cache)

- Dodatkowa, szybka pamięć (SRAM) umieszczana pomiędzy procesorem a pamięcią główną
- Zastosowanie pamięci podręcznej ma na celu przyspieszenie dostępu procesora do pamięci głównej



System operacyjny - definicja

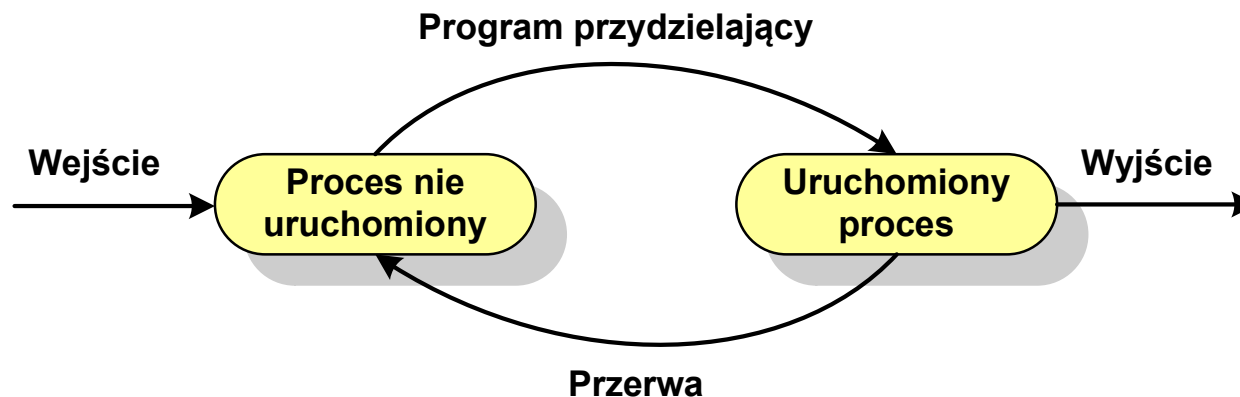
- **System operacyjny** - jest to program sterujący wykonywaniem aplikacji i działający jako interfejs pomiędzy aplikacjami (użytkownikiem) a sprzętem komputerowym
- System operacyjny - **administrator zasobów** - zarządza i przydziela zasoby systemu komputerowego oraz steruje wykonaniem programu
- **zasób systemu** - każdy element systemu, który może być przydzielony innej części systemu lub oprogramowaniu aplikacyjnemu
- do zasobów systemu zalicza się:
 - czas procesora
 - pamięć operacyjną
 - urządzenia zewnętrzne

Zarządzanie procesami

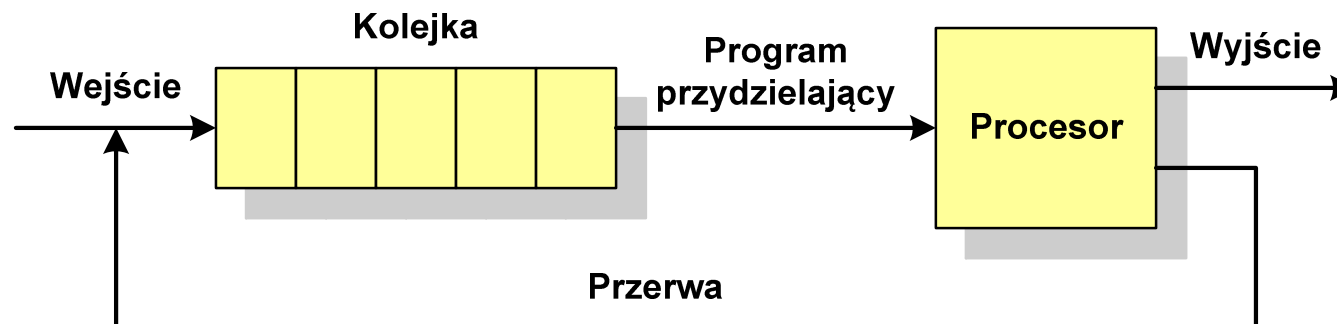
- Głównym zadaniem systemu operacyjnego jest **zarządzanie procesami**
- Definicja procesu:
 - **proces** - program w trakcie wykonania
 - **proces** - ciąg wykonań instrukcji wyznaczanych kolejnymi wartościami licznika rozkazów wynikających z wykonywanej procedury (programu)
 - **proces** - jednostka, którą można przypisać procesorowi i wykonać
- Proces składa się z kilku elementów:
 - **kod programu**
 - **dane potrzebne programowi** (zmienne, przestrzeń robocza, bufory)
 - **kontekst wykonywanego programu** (stan procesu) - dane wewnętrzne, dzięki którym system operacyjny może nadzorować proces i nim sterować

Dwustanowy model procesu

- najprostszy model polega na tym, że w dowolnej chwili proces jest wykonywany przez procesor (**uruchomiony**) lub nie (**nie uruchomiony**)

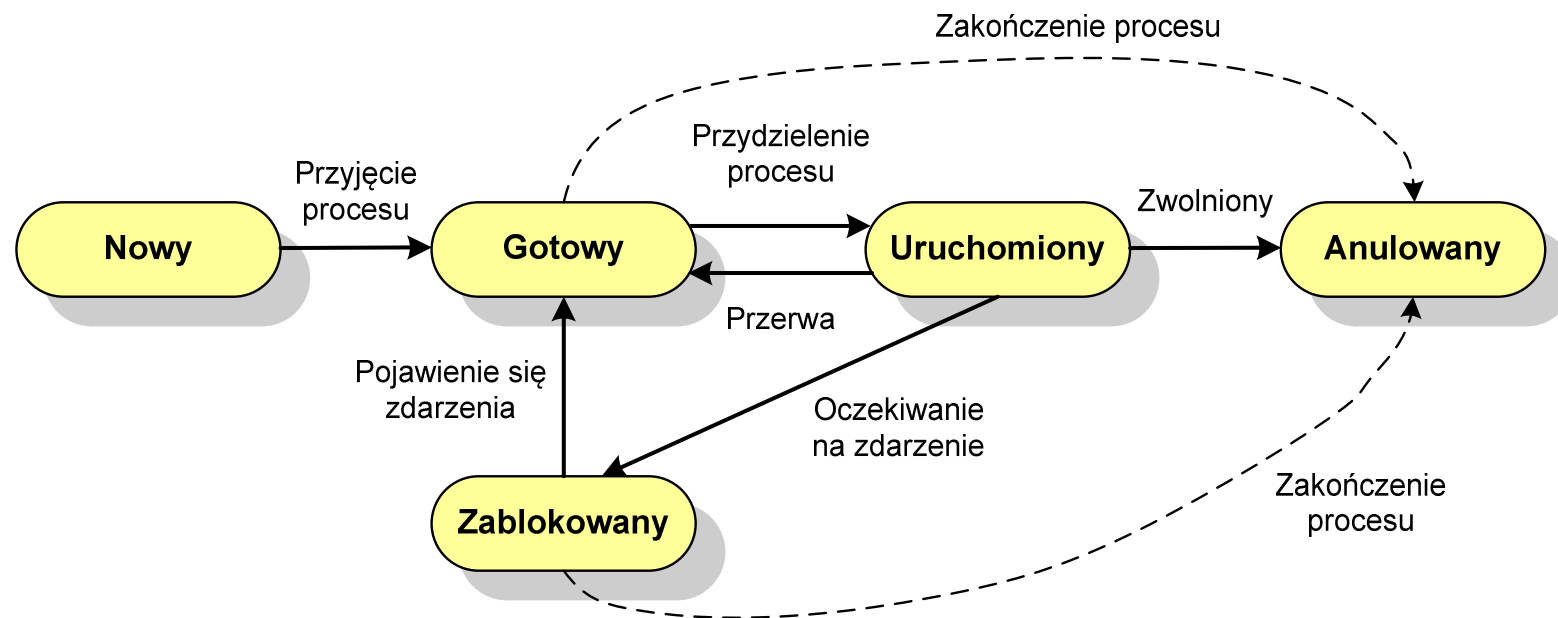


- procesy, które nie są uruchomione czekają w kolejce na wykonanie



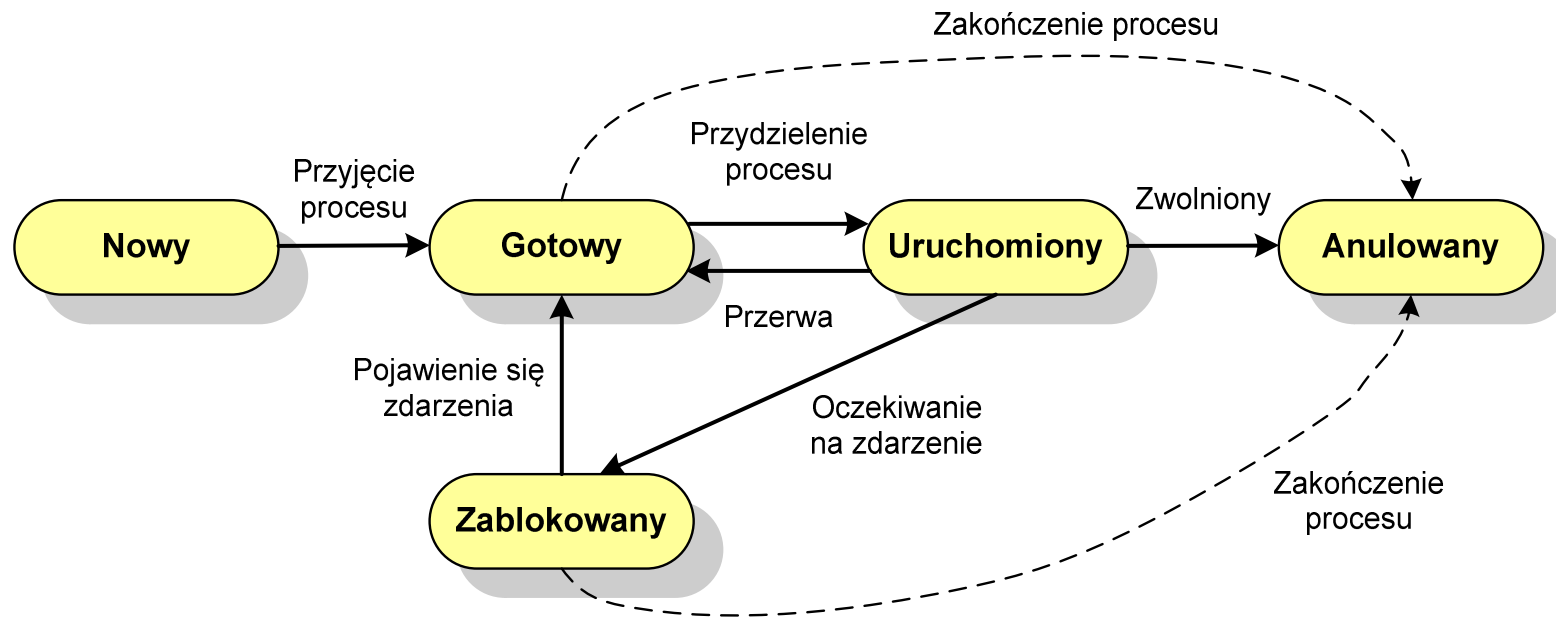
Pięciostanowy model procesu

- ❑ wadą dwustanowego modelu procesu jest sytuacja, gdy kolejny proces pobierany do wykonania z kolejki jest zablokowany, gdyż oczekuje na zakończenie operacji we-wy
- ❑ rozwiązaniem powyższego problemu jest podział procesów nieuruchomionych na **gotowe do wykonania** i **zablokowane**



- ❑ pięciostanowy model procesu wymaga zastosowania dwóch kolejek

Pięciostanowy model procesu



- **uruchomiony** - proces aktualnie wykonywany
- **gotowy** - proces gotowy do wykonania przy najbliższej możliwej okazji
- **zablokowany** - proces oczekujący na zakończenie operacji we-wy
- **nowy** - proces, który właśnie został utworzony (ma utworzony blok kontrolny procesu, nie został jeszcze załadowany do pamięci), ale nie został jeszcze przyjęty do grupy procesów oczekujących na wykonanie
- **anulowany** - proces, który został wstrzymany lub anulowany z jakiegoś powodu

Zarządzanie dyskowymi operacjami we-wy

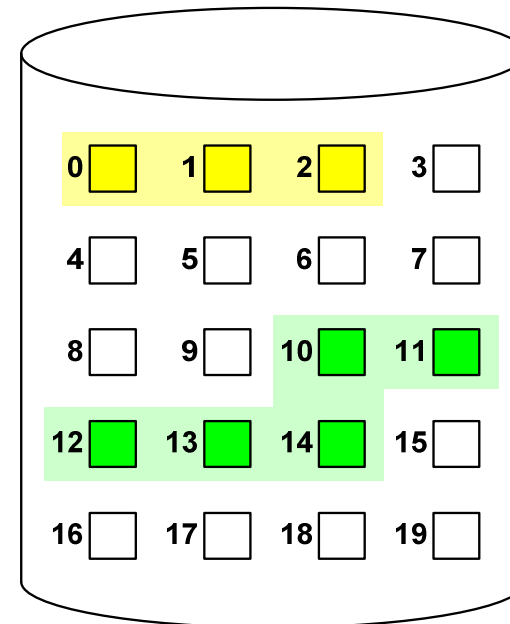
- Metody przydziału pamięci dyskowej (teoria)
 - alokacja ciągła
 - alokacja listowa
 - alokacja indeksowa

- Struktura dysku twardego
 - MBR (BIOS)
 - GPT (UEFI)

- Systemy plików (praktyka)
 - FAT (FAT12, FAT16, FAT32, exFAT)
 - NTFS
 - ext2

Przydział pamięci dyskowej - alokacja ciągła

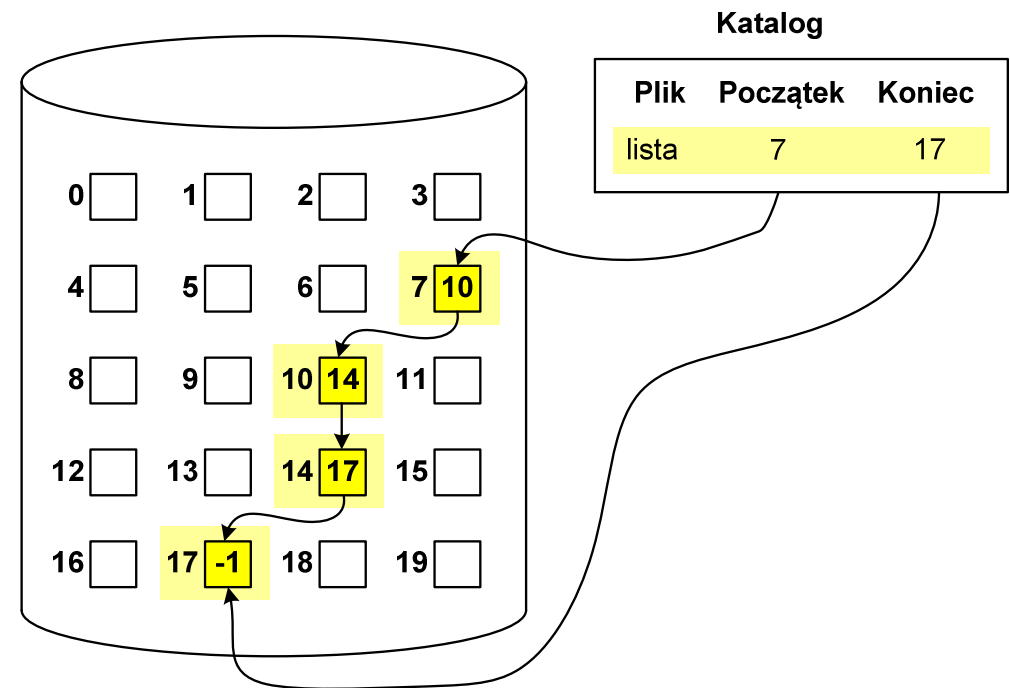
- każdy plik zajmuje ciąg kolejnych bloków na dysku
- plik zdefiniowany jest przez adres pierwszego bloku i ilość kolejnych zajmowanych bloków
- zalety: małe opóźnienia w transmisji danych, łatwy dostęp do dysku
- wady: trudność w znalezieniu miejsca na nowy plik



Plik	Początek	Długość
lista	0	3
poczta	10	5

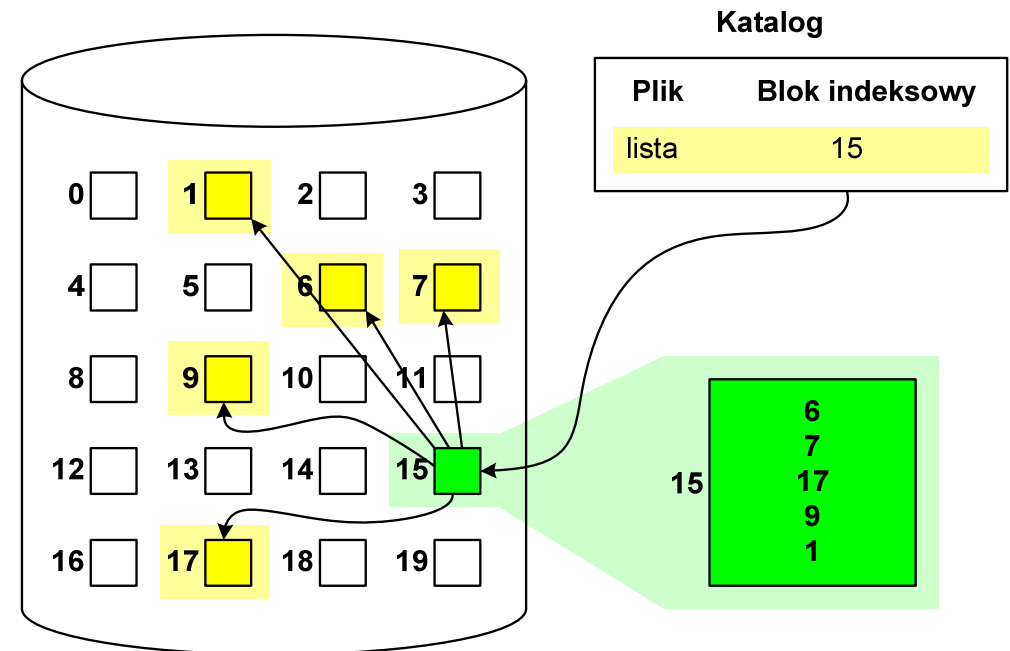
Przydział pamięci dyskowej - alokacja listowa

- każdy plik jest listą powiązanych ze sobą bloków dyskowych, które mogą znajdować się w dowolnym miejscu na dysku
- w katalogu dla każdego pliku zapisany jest wskaźnik do pierwszego i ostatniego bloku pliku
- każdy blok zawiera wskaźnik do następnego bloku



Przydział pamięci dyskowej - alokacja indeksowa

- każdy plik ma własny blok indeksowy, będący tablicą adresów bloków dyskowych
- w katalogu zapisany jest dla każdego pliku adres bloku indeksowego



Struktura dysku twardego - MBR

- **MBR (Master Boot Record)** - główny rekord ładujący (1983, PC DOS 2.0)
- struktura danych opisująca podział dysku na partycje
- pierwszy sektor logiczny dysku (CHS → 0,0,1), zajmuje 512 bajtów

446 bajtów	4 × 16 = 64 bajty				2 bajty
Główny kod startowy	Tablica partycji				Sygnatura rozruchu
	Partycja 1	Partycja 2	Partycja 3	Partycja 4	

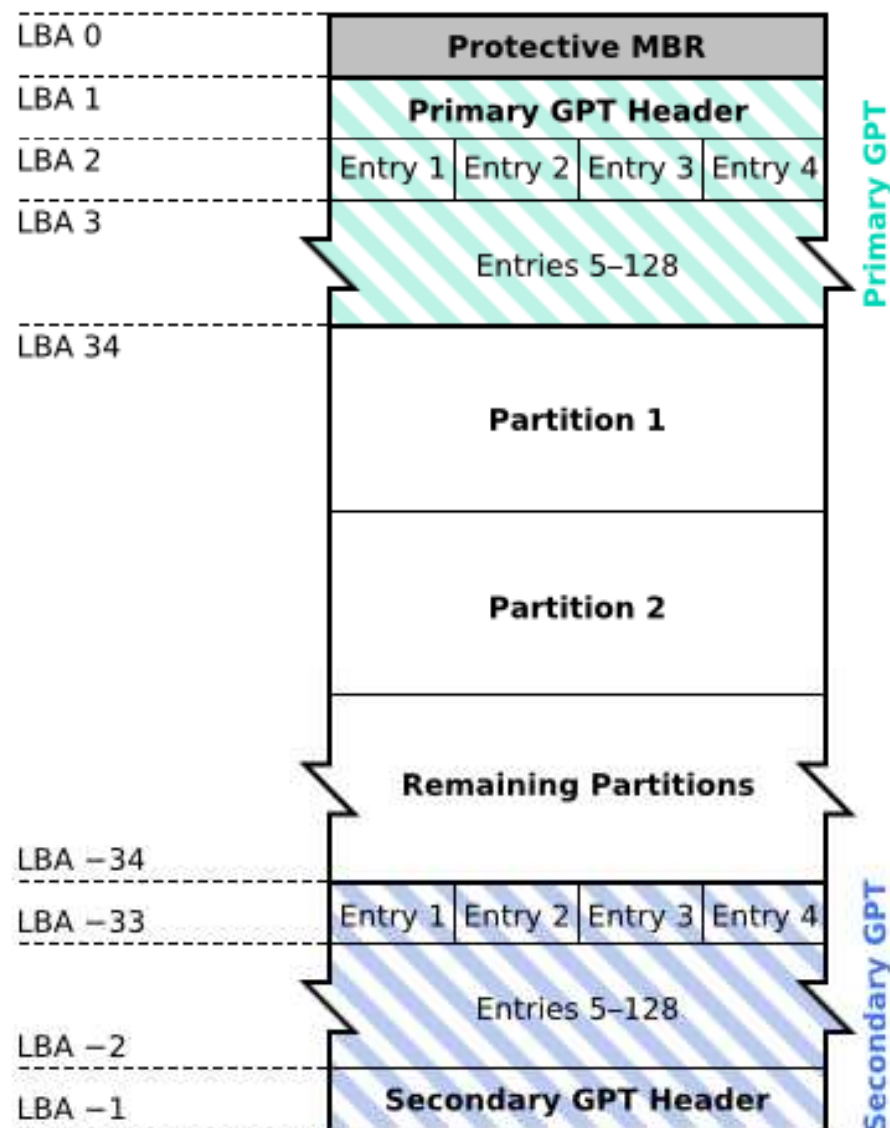
- **główny kod startowy (Master Boot Code, bootloader)** - program odszukujący i ładujący do pamięci zawartość pierwszego sektora aktywnej partycji
- **tablica partycji** - cztery 16-bajtowe rekordy opisujące partycje na dysku
- **sygnatura rozruchu (boot signature)** - znacznik końca MBR (**0x55AA**)
- maksymalny rozmiar partycji to **2 TB** ($2^{32} \times 512$ bajtów)

Struktura dysku twardego - GPT

- **GPT (GUID Partition Table)** - standard zapisu informacji o partycjach na dysku twardym
- **GUID (Globally Unique Identifier)** - 128-bitowa liczba stosowana do identyfikowania informacji w systemach komputerowych
- GPT to część standardu **UEFI (Unified Extensible Firmware Interface)**, który zastąpił BIOS w komputerach PC (interfejs graficzny, obsługa myszki)
- opracowanie: IBM/Microsoft, 2010 rok
- maksymalny rozmiar dysku to **9,4 ZB** (2^{64} sektorów \times 512 bajtów)
- możliwość utworzenia do 128 partycji podstawowych

Struktura dysku twardego - GPT (struktura)

- **Protective MBR** - pozostawiony dla bezpieczeństwa
- **GPT Header** (512 bajtów):
 - liczba pozycji w tablicy partycji
 - rozmiar pozycji w tablicy partycji
 - położenie zapasowej kopii GPT
 - unikatowy identyfikator dysku
 - sumy kontrolne
- **Entry x** (128 bajtów):
 - typ partycji
 - unikatowy identyfikator
 - początkowy i końcowy numer LBA
 - atrybuty
 - nazwa



System plików FAT (File Allocation Table)

- opracowany na przełomie lat 70. i 80. dla systemu MS-DOS
- występuje w czterech wersjach: FAT12, FAT16, FAT32 i exFAT (FAT64)
- numer występujący po słowie FAT oznacza liczbę bitów przeznaczonych do kodowania (numeracji) **jednostek alokacji pliku** (JAP), tzw. **klastrów** (ang. cluster) w tablicy alokacji plików
 - 12 bitów w systemie FAT12
 - 16 bitów w systemie FAT16
 - 32 bity w systemie FAT32
 - 64 bity w systemie exFAT (FAT64)
- ogólna struktura dysku logicznego / dyskietki w systemie FAT:

Rekord ładujący + sektory zarezerwowane	Tablica rozmieszczenia plików - FAT	Kopia FAT	Katalog główny (FAT12 i FAT16)	Miejsce na pliki i podkatalogi
--	--	------------------	---	---

FAT12

- system plików FAT12 przeznaczony jest dla nośników o małej pojemności
- **rekord ładujący** zajmuje pierwszy sektor dyskietki lub dysku logicznego



- rekord ładujący zawiera następujące dane:
 - instrukcja skoku do początku programu ładującego (3 bajty)
 - nazwa wersji systemu operacyjnego (8 bajtów)
 - struktura BPB (ang. BIOS Parametr Block) - blok parametrów BIOS (25 bajtów)
 - rozszerzony BPB (ang. Extended BPB, 26 bajtów)
 - wykonywalny kod startowy uruchamiający system operacyjny (448 bajtów)
 - znacznik końca sektora - 55AAH (2 bajty)

FAT12

- **tablica rozmieszczenia plików FAT** tworzy swego rodzaju „mapę” plików zapisanych na dysku
- za tablicą FAT znajduje się jej kopia, która nie jest wykorzystywana



- za kopią tablicy FAT znajduje się **katalog główny** zajmujący określoną dla danego typu dysku liczbę sektorów

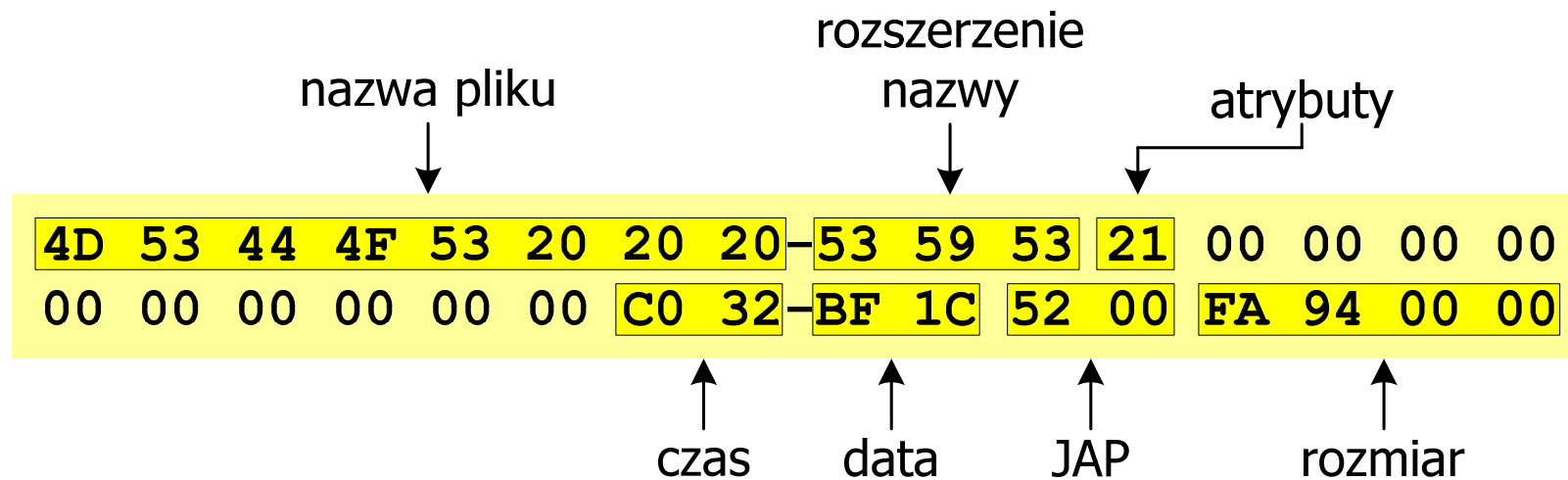


- katalog główny zawiera 32-bajtowe pola mogące opisywać pliki, podkatalogi lub etykietę dysku

FAT12

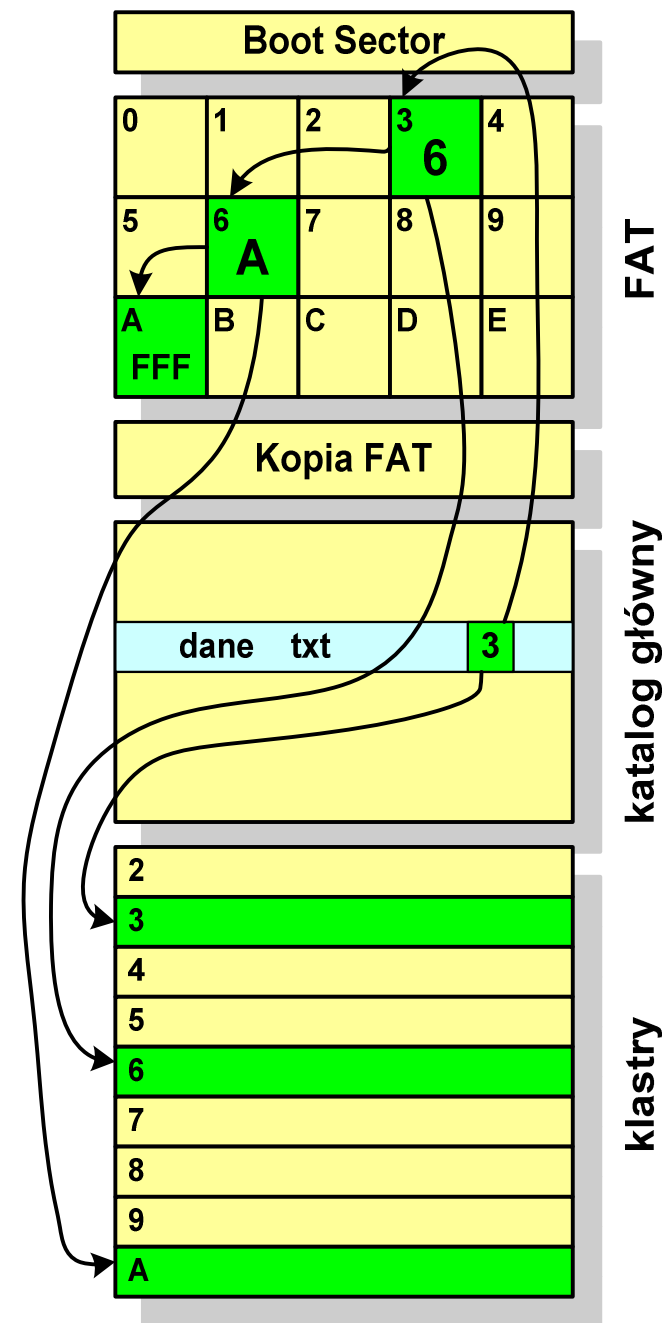
- przykładowa zawartość katalogu głównego:

0000	49 4F 20 20 20 20 20 20	20-53 59 53 21 00 00 00 00	IO	SYS!....
0010	00 00 00 00 00 00 00 C0	32-BF 1C 02 00 46 9F 00 002....F...	
0020	4D 53 44 4F 53 20 20 20	20-53 59 53 21 00 00 00 00	MSDOS	SYS!....
0030	00 00 00 00 00 00 00 C0	32-BF 1C 52 00 FA 94 00 002..R.....	
0040	43 4F 4D 4D 41 4E 44 20-43	4F 4D 20 00 00 00 00 00	COMMAND	COM
0050	00 00 00 00 00 00 00 C0	32-BF 1C 9D 00 75 D5 00 002....u...	
0060	41 54 54 52 49 42 20 20-45	58 45 20 00 00 00 00 00	ATTRIB	EXE
0070	00 00 00 00 00 00 00 C0	32-BF 1C 08 01 C8 2B 00 002.....+..	



FAT12 - położenie pliku na dysku

- w katalogu, w 32-bajtowym polu każdego pliku wpisany jest początkowy numer JAP
- numer ten określa logiczny numer sektora, w którym znajduje się początek pliku
- ten sam numer JAP jest jednocześnie indeksem do miejsca w tablicy FAT, w którym wpisany jest numer kolejnej JAP
- numer wpisany we wskazanym miejscu tablicy rozmieszczenia plików wskazuje pierwszy sektor następnej części pliku i równocześnie położenie w tablicy FAT numeru następnej JAP
- w ten sposób tworzy się łańcuch, określający położenie całego pliku
- jeśli numer JAP składa się z samych FFF, to oznacza to koniec pliku



FAT32

- po raz pierwszy wprowadzony w systemie Windows 95 OSR2
- ogólna struktura systemu FAT32 jest taka sama jak w FAT12/FAT16 - nie ma tylko miejsca przeznaczonego na katalog główny
- w systemie FAT32 katalog główny może znajdować się w dowolnym miejscu na dysku i może zawierać maksymalnie 65 532 pliki i katalogi

Rekord ładujący + sektory zarezerwowane	Tablica rozmieszczenia plików - FAT	Kopia FAT	Miejsce na pliki i katalogi
--	--	------------------	--

- do adresowania JAP stosuje się, obcięty o 4 najstarsze bity, adres 32-bitowy i dlatego dysk z FAT32 może zawierać maksymalnie 2^{28} JAP
- w systemie FAT32 można formatować tylko dyski, nie można natomiast zainstalować go na dyskietkach

FAT32 - długie nazwy plików

- wprowadzone w systemie Windows 95
- informacje o nazwie pliku zapamiętywane są jako:
 - długa nazwa (13 znaków w każdej 32-bajtowej strukturze, Unicode)
 - skrócona nazwa (32-bajtowa struktura, stary format 8+3)

długa nazwa pliku



0000	43	20	00	64	00	6F	00	6D	00	6F	00	0F	00	CF	77	00	C	.	d	.	o	.	m	.	o	.	.	.	w	.				
0010	61	00	2E	00	74	00	78	00	74	00	00	00	00	00	00	FF	FF	a	.	.	.	t	.	x	.	t			
0020	02	63	00	79	00	6A	00	6E	00	65	00	0F	00	CF	20	00	.	c	.	y	.	j	.	n	.	e			
0030	2D	00	20	00	70	00	72	00	61	00	00	00	63	00	61	00	-	.	.	p	.	r	.	a	.	.	.	c	.	a	.			
0040	01	53	00	79	00	73	00	74	00	65	00	0F	00	CF	6D	00	.	S	.	y	.	s	.	t	.	e	m	.		
0050	79	00	20	00	4F	00	70	00	65	00	00	00	72	00	61	00	y	.	O	.	p	.	e	r	a	.	
0060	53	59	53	54	45	4D	7E	31	54	58	54	20	00	4B	03	80	S	Y	S	T	E
0070	67	32	67	32	00	00	08	80	67	32	02	00	06	00	00	00	g	2	g	2

skrócona nazwa pliku



exFAT (FAT64)

- po raz pierwszy pojawił się w listopadzie 2006 roku w Windows Embedded CE 6.0 i Windows Vista SP1
- obsługiwany także przez Windows 7/8/10, Windows Server 2003/2008, Windows XP SP2/SP3, Linux
- stworzony przez Microsoft na potrzeby pamięci Flash
- podstawowe cechy:
 - maksymalna wielkość pliku to $2^{64} = 16 \text{ EB}$
 - maksymalna wielkość klastra - do 32 MB
 - nieograniczona liczba plików w pojedynczym katalogu
 - prawa dostępu do plików i katalogów

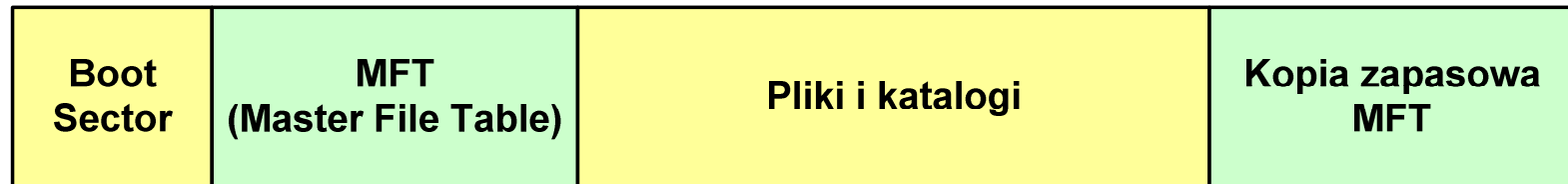
NTFS (New Technology File System)

- **wersja 1.0** (połowa 1993 r.) - Windows NT 3.1
- **wersja 3.1** (NTFS 5.1) - Windows XP/Server 2003/Vista/7/8/10
- struktura wolumenu (dysku) NTFS:



- **Boot Sector** rozpoczyna się od zerowego sektora partycji, może zajmować 16 kolejnych sektorów, zawiera podobne dane jak w systemie FAT

NTFS



- **MFT (Master File Table)** - specjalny plik, niewidoczny dla użytkownika, zawiera wszystkie dane niezbędne do odczytania pliku z dysku, składa się z rekordów o stałej długości (1 kB - 4 kB)
- pierwsze 16 (NTFS 4) lub 26 (NTFS 5) rekordów jest zarezerwowane dla tzw. metaplików, np.
 - rekord nr: 0 plik: **\$Mft** (główna tablica plików)
 - rekord nr: 1 plik: **\$MftMirr** (główna tablica plików 2)
 - rekord nr: 5 plik: **\$** (indeks katalogu głównego)
- pozostała część pliku MFT przeznaczona jest na rekordy wszystkich plików i katalogów umieszczonych na dysku

NTFS

- struktura wolumenu (dysku) NTFS:



- plik w NTFS to **zbiór atrybutów**
- wszystkie atrybuty mają dwie części składowe: **nagłówek** i **blok danych**
- **nagłówek** opisuje atrybut, np. liczbę bajtów zajmowanych przez atrybut, rozmiar bloku danych, położenie bloku danych, znacznik czasu
- **bloku danych** zawiera informacje zgodne z przeznaczeniem atrybutu

NTFS - Pliki

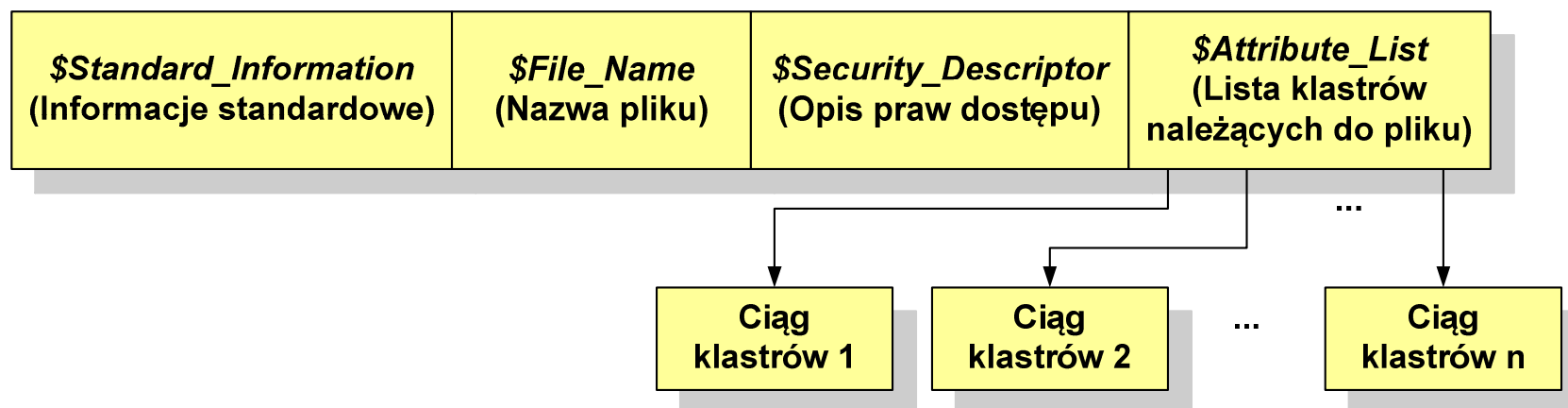
- pliki w systemie NTFS są reprezentowane w MFT przez rekord zawierający atrybuty:
 - **\$Standard_Information**
 - **\$File_Name**
 - **\$Security_Descriptor**
 - **\$Data**

<i>\$Standard_Information</i> (Informacje standardowe)	<i>\$File_Name</i> (Nazwa pliku)	<i>\$Security_Descriptor</i> (Opis praw dostępu)	<i>\$Data</i> (Dane)
--	--	--	--------------------------------

- w przypadku małych plików wszystkie jego atrybuty zapisywane są bezpośrednio w MFT (atrybuty **rezydentne**)

NTFS - Pliki

- jeśli atrybuty pliku są duże (najczęściej dotyczy to atrybutu **\$Data**), to w rekordzie w MFT umieszczany jest tylko nagłówek atrybutu oraz wskaźnik do jego bloku danych, a sam blok danych przenoszony jest na dysk poza MFT (atrybuty **nierezydentne**)
- blok danych atrybutu nierezydentnego zapisywany jest w przyległych klastrach
- jeśli nie jest to możliwe, to dane zapisywane są w kilku ciągach jednostek alokacji i wtedy każdemu ciągowi odpowiada wskaźnik w rekordzie MFT



NTFS - Katalogi

- katalogi reprezentowane są przez rekordy zawierające trzy takie same atrybuty jak pliki:
 - **\$Standard_Information**
 - **\$File_Name**
 - **\$Security_Descriptor**

<i>\$Standard_Information</i> (Informacje standardowe)	<i>\$File_Name</i> (Nazwa pliku)	<i>\$Security_Descriptor</i> (Opis praw dostępu)	<i>\$Index_Root</i>	<i>\$Index_Allocation</i>	<i>\$Bitmap</i>
--	--	--	----------------------------	----------------------------------	------------------------

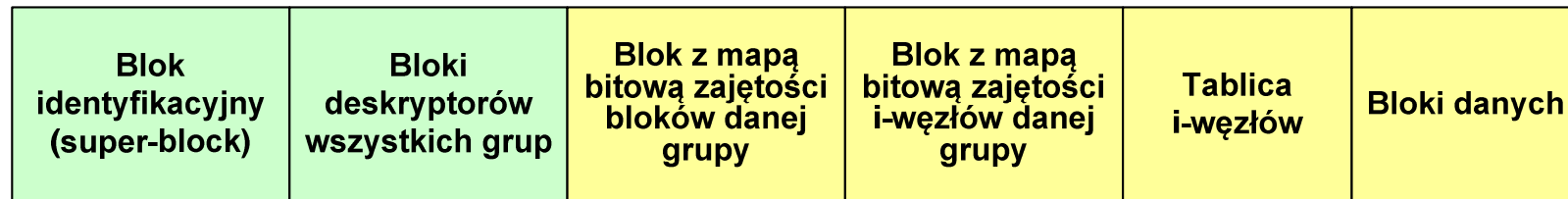
- zamiast atrybutu **\$Data** umieszczone są trzy atrybuty przeznaczone do tworzenia list, sortowania oraz lokalizowania plików i podkatalogów
 - **\$Index_Root**
 - **\$Index_Allocation**
 - **\$Bitmap**

ext2

- pierwszy system plików w Linuxie: **Minix** (14-znakowe nazwy plików i maksymalny rozmiar wynoszący 64 MB)
- system Minix zastąpiono nowym systemem nazwanym rozszerzonym systemem plików - **ext** (ang. **extended file system**), a ten, w styczniu 1993 r., systemem **ext2** (ang. **second extended file system**)
- w systemie ext2 podstawowym elementem podziału dysku jest **blok**
- wielkość bloku jest stała w ramach całego systemu plików, określana na etapie jego tworzenia i może wynosić 1024, 2048 lub 4096 bajtów
- w celu zwiększenia bezpieczeństwa i optymalizacji zapisu na dysku posługujemy się nie pojedynczymi blokami, a **grupami bloków**

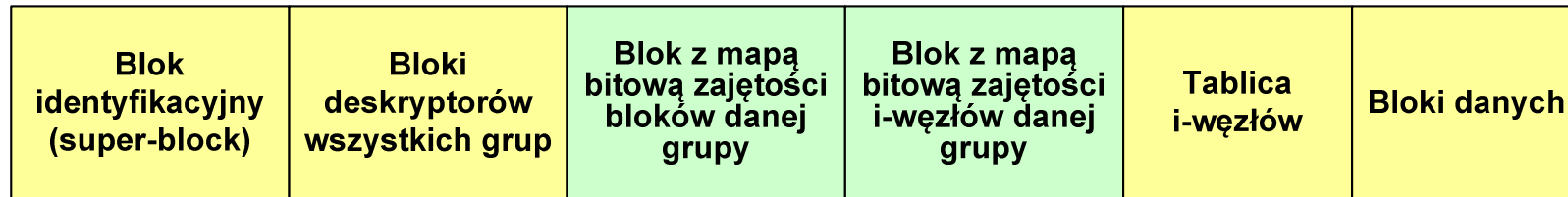


ext2



- w każdej grupie bloków znajduje się kopia tego samego bloku identyfikacyjnego oraz kopia bloków z deskryptorami wszystkich grup
- **blok identyfikacyjny** zawiera informacje na temat systemu plików (rodzaj systemu plików, rozmiar bloku, czas dokonanej ostatnio zmiany , ...)
- w **deskryptorach grupy** znajdują się informacje na temat grupy bloków (numer bloku z bitmapą zajętości bloków grupy, numer bloku z bitmapą zajętości i-węzłów, numer pierwszego bloku z tablicą i-węzłów, liczba wolnych bloków, liczba katalogów w grupie)

ext2



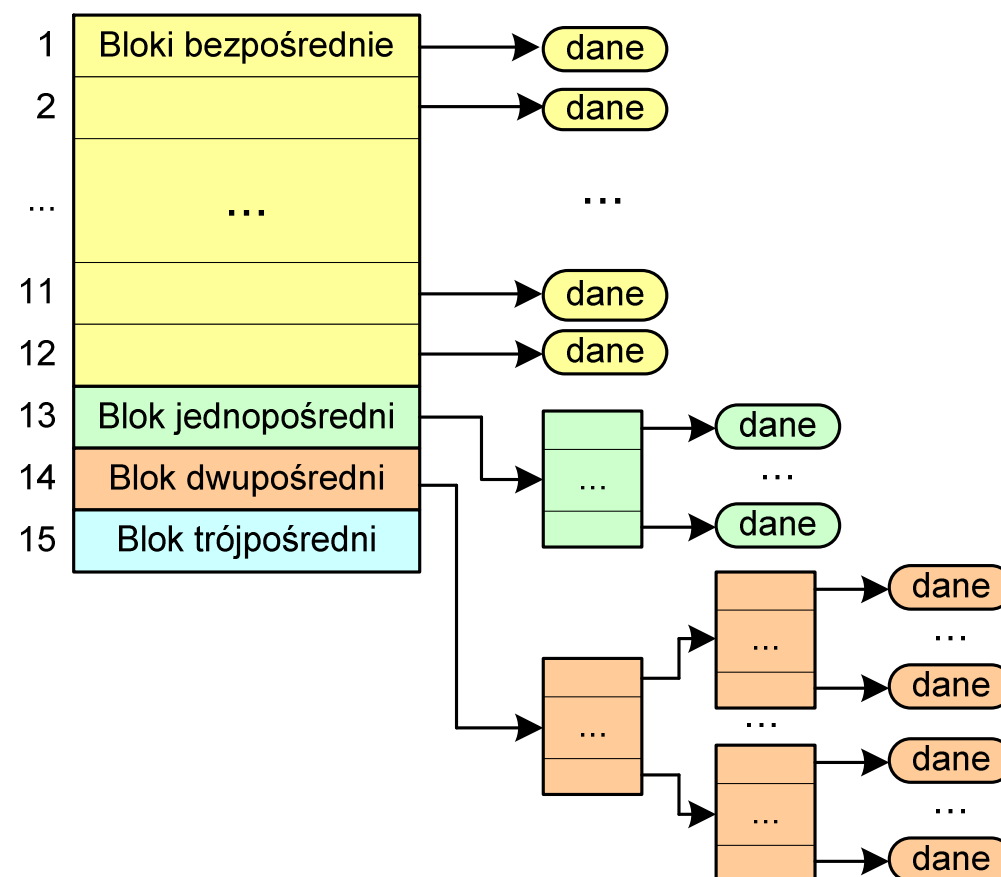
- **blok z mapą bitową zajętości bloków danej grupy** jest tablicą bitów o rozmiarze jednego bloku
 - jeśli blok ma rozmiar 1 kB to pojedynczą mapą można opisać fizyczna grupę 8096 bloków czyli 8 MB danych
 - jeśli natomiast blok ma rozmiar 4 kB, to fizyczna grupa bloków zajmuje 128 MB danych
- przed tablicą i-węzłów znajduje się **blok z mapą bitową zajętości i-węzłów danej grupy** - jest to tablica bitów, z których każdy zawiera informację czy dany i-węzeł jest wolny czy zajęty

ext2 - i-węzeł

- pliki na dysku reprezentowane są przez **i-węzły** (ang. **i-node**)
- każdemu plikowi odpowiada dokładnie jeden i-węzeł, który jest strukturą zawierającą m.in. następujące pola:
 - numer i-węzła w dyskowej tablicy i-węzłów
 - typ pliku: zwykły, katalog, łącze nazwane, specjalny, znakowy
 - prawa dostępu do pliku: dla wszystkich, grupy, użytkownika
 - liczba dowiązań do pliku
 - identyfikator właściciela pliku
 - identyfikator grupy właściciela pliku
 - rozmiar pliku w bajtach (max. 4 GB)
 - czas utworzenia pliku
 - czas ostatniego dostępu do pliku
 - czas ostatniej modyfikacji pliku
 - liczba bloków dyskowych zajmowanych przez plik

ext2 - i-węzeł

- położenie pliku na dysku określają w i-węźle pola:
 - 12 adresów bloków zawierających dane (w systemie Unix jest ich 10)
 - **bloki bezpośrednie**
 - 1 adres bloku zawierającego adresy bloków zawierających dane - **blok jednopięśredni** (ang. single indirect block)
 - 1 adres bloku zawierającego adresy bloków jednopięśrednich - **blok dwupięśredni** (ang. double indirect block)
 - 1 adres bloku zawierającego adresy bloków dwupięśrednich - **blok trójpięśredni** (ang. triple indirect block)



ext2

- **nazwy plików** przechowywane są w **katalogach**, które w systemie Linux są plikami, ale o specjalnej strukturze
- katalogi składają się z ciągu tzw. **pozycji katalogowych** o nieustalonej z góry długości
- każda pozycja opisuje dowiązanie do jednego pliku i zawiera:
 - numer i-węzła (4 bajty)
 - rozmiar pozycji katalogowej (2 bajty)
 - długość nazwy (2 bajty)
 - nazwa (od 1 do 255 znaków)

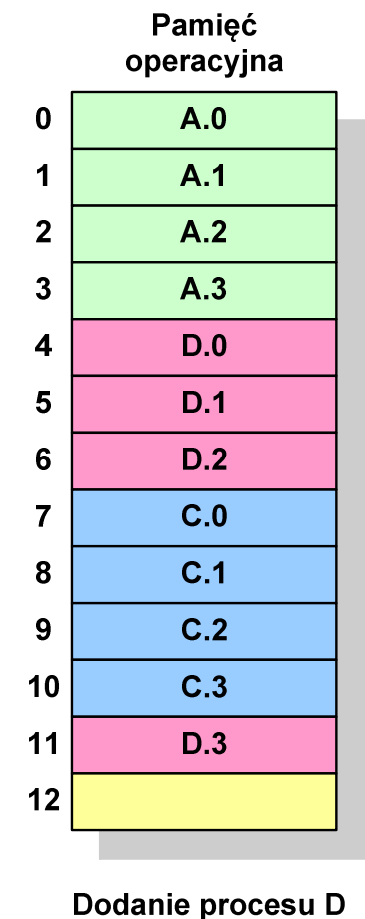
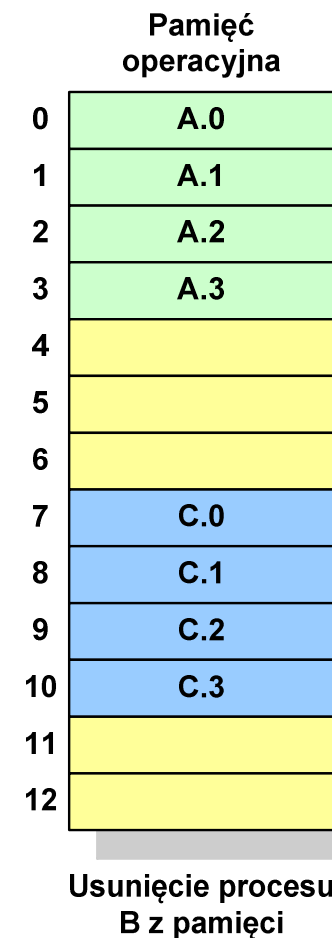
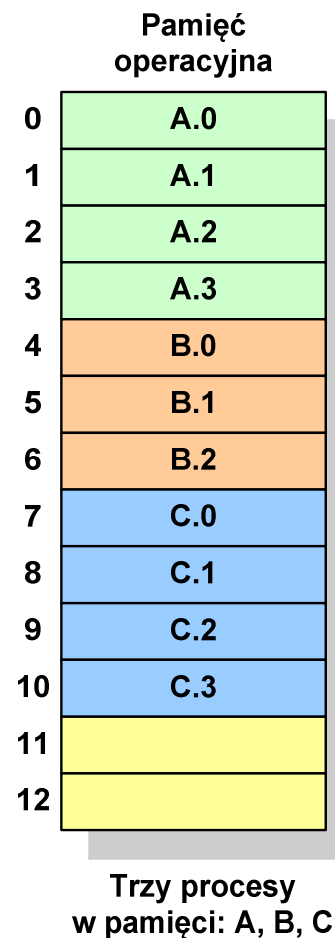
```
struct ext2_dir_entry
{
    _u32  inode           /* numer i-wezla          */
    _u16  rec_len        /* dlugosc pozycji katalogowej */
    _u16  name_len       /* dlugosc nazwy          */
    char  name[EXT2_NAME_LEN] /* nazwa                  */
}
```

Zarządzanie pamięcią

- zarządzanie pamięcią polega na wydajnym przenoszeniu programów i danych do i z pamięci operacyjnej
- w nowoczesnych wieloprogramowych systemach operacyjnych zarządzanie pamięcią opiera się na **pamięci wirtualnej**
- pamięć wirtualna bazuje na wykorzystaniu **segmentacji** i **stronicowania**
- z historycznego punktu widzenia w systemach komputerowych stosowane były/są następujące metody zarządzania pamięcią:
 - proste stronicowanie, prosta segmentacja
 - stronicowanie pamięci wirtualnej, segmentacja pamięci wirtualnej
 - **stronicowanie i segmentacja pamięci wirtualnej**

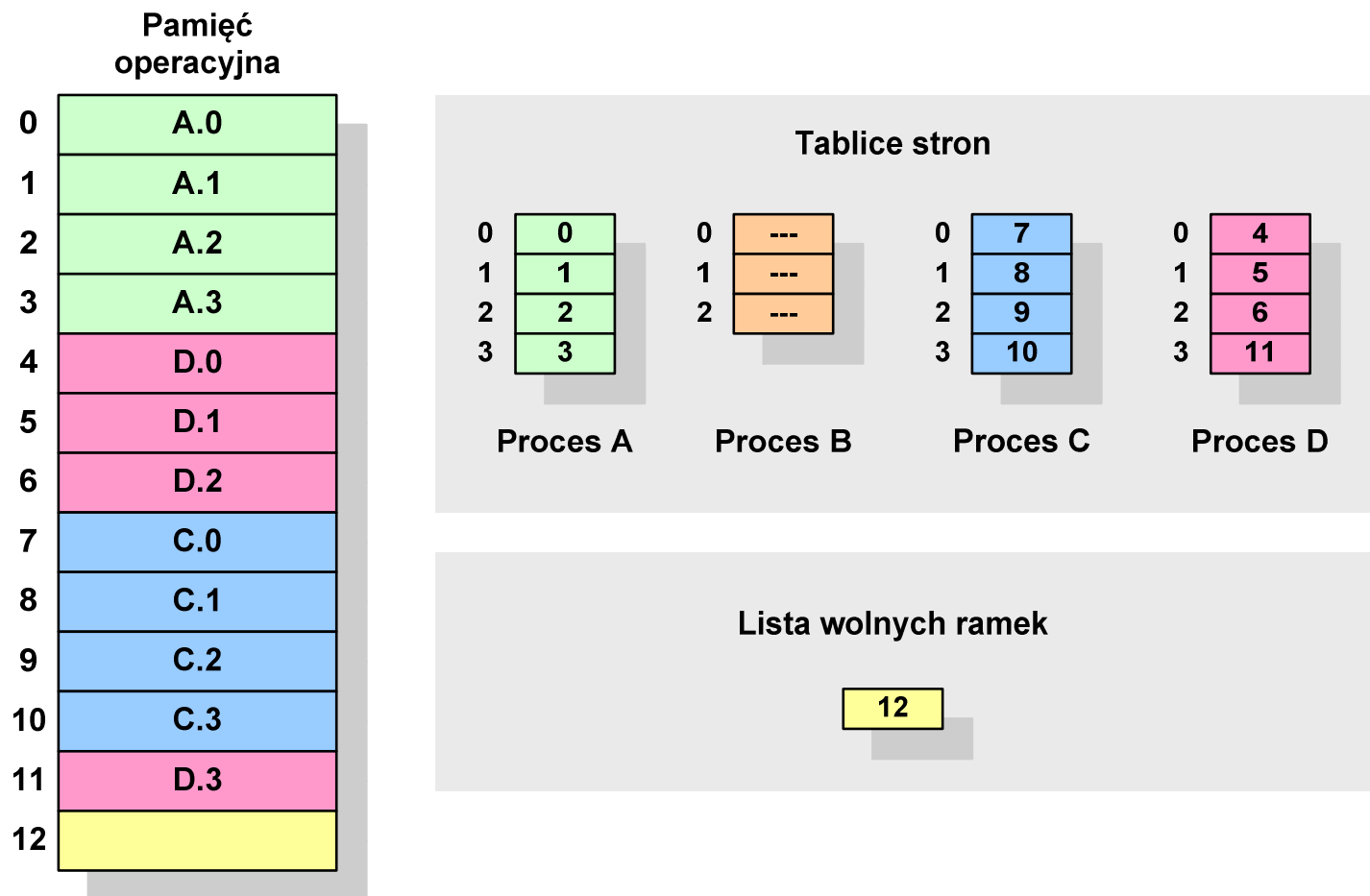
Proste stronicowanie

- pamięć operacyjna podzielona jest na jednakowe bloki o stałym niewielkim rozmiarze nazywane **ramkami** lub **ramkami stron** (page frames)
- do tych ramek wstawiane są fragmenty procesu zwane **stronami** (pages)
- aby proces mógł zostać uruchomiony wszystkie jego strony muszą znajdować się w pamięci operacyjnej



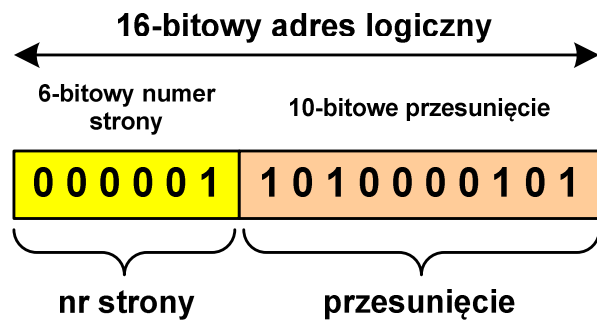
Proste stronicowanie

- dla każdego procesu przechowywana jest **tablica strony** (page table) zawierająca lokalizację ramki dla każdej strony procesu



Proste stronicowanie

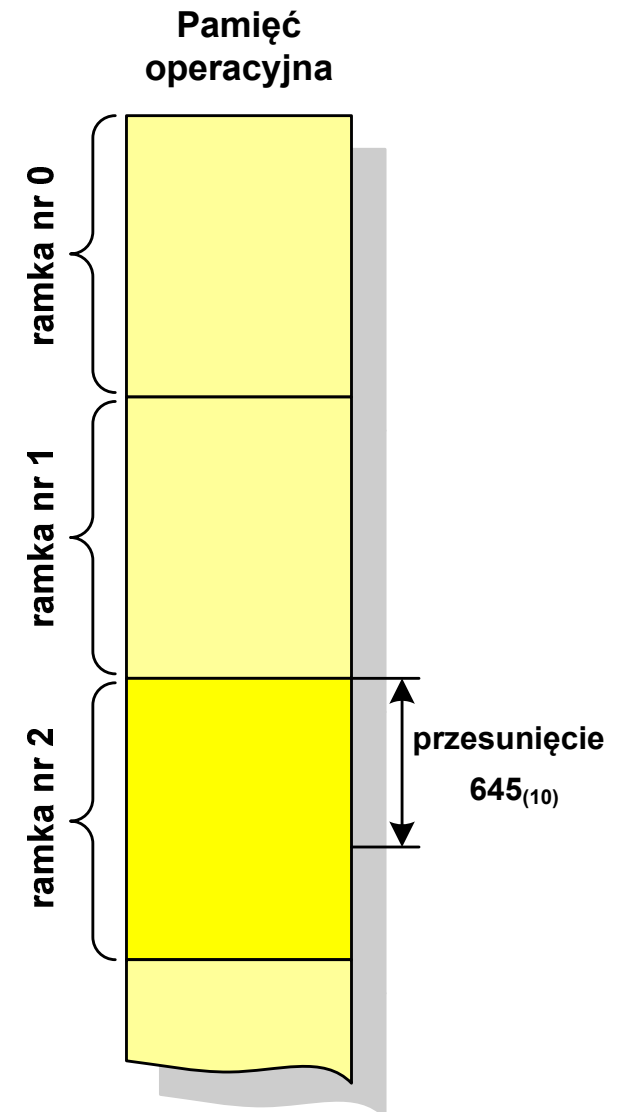
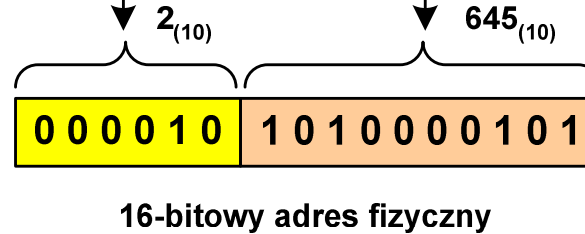
Przykład:



0	010111
1	000010
2	110011

tablica stron procesu

A table with three rows. The first row has page number 0 and physical address 010111. The second row has page number 1 and physical address 000010. The third row has page number 2 and physical address 110011. The table is labeled 'tablica stron procesu'.

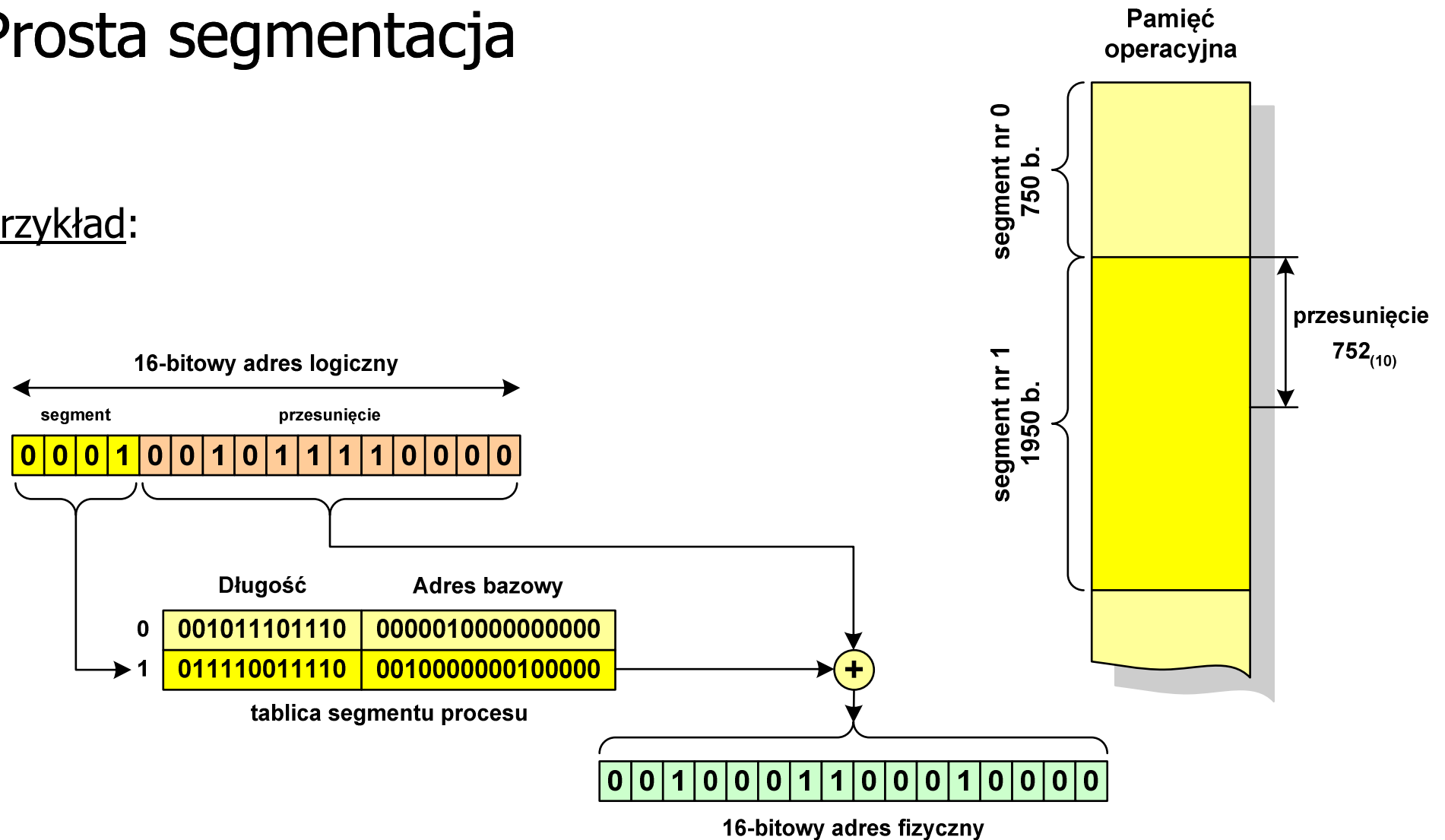


Prosta segmentacja

- polega na podzieleniu programu i skojarzonych z nim danych na odpowiednią liczbę **segmentów** o **różnej długości**
- ładowanie procesu do pamięci polega na wczytaniu wszystkich jego segmentów do partycji dynamicznych (nie muszą być ciągłe)
- segmentacja jest widoczna dla programisty i ma na celu wygodniejszą organizację programów i danych
- **adres logiczny** wykorzystujący segmentację składa się z dwóch części:
 - numeru segmentu
 - przesunięcia
- dla każdego procesu określana jest **tablica segmentu procesu** zawierająca:
 - długość danego segmentu
 - adres początkowy danego segmentu w pamięci operacyjnej

Prosta segmentacja

Przykład:



Pamięć wirtualna

- **pamięć wirtualna** umożliwia przechowywanie stron/segmentów wykonywanego procesu w pamięci dodatkowej (na dysku twardym)

Co się dzieje, gdy procesor chce odczytać stronę z pamięci dodatkowej?

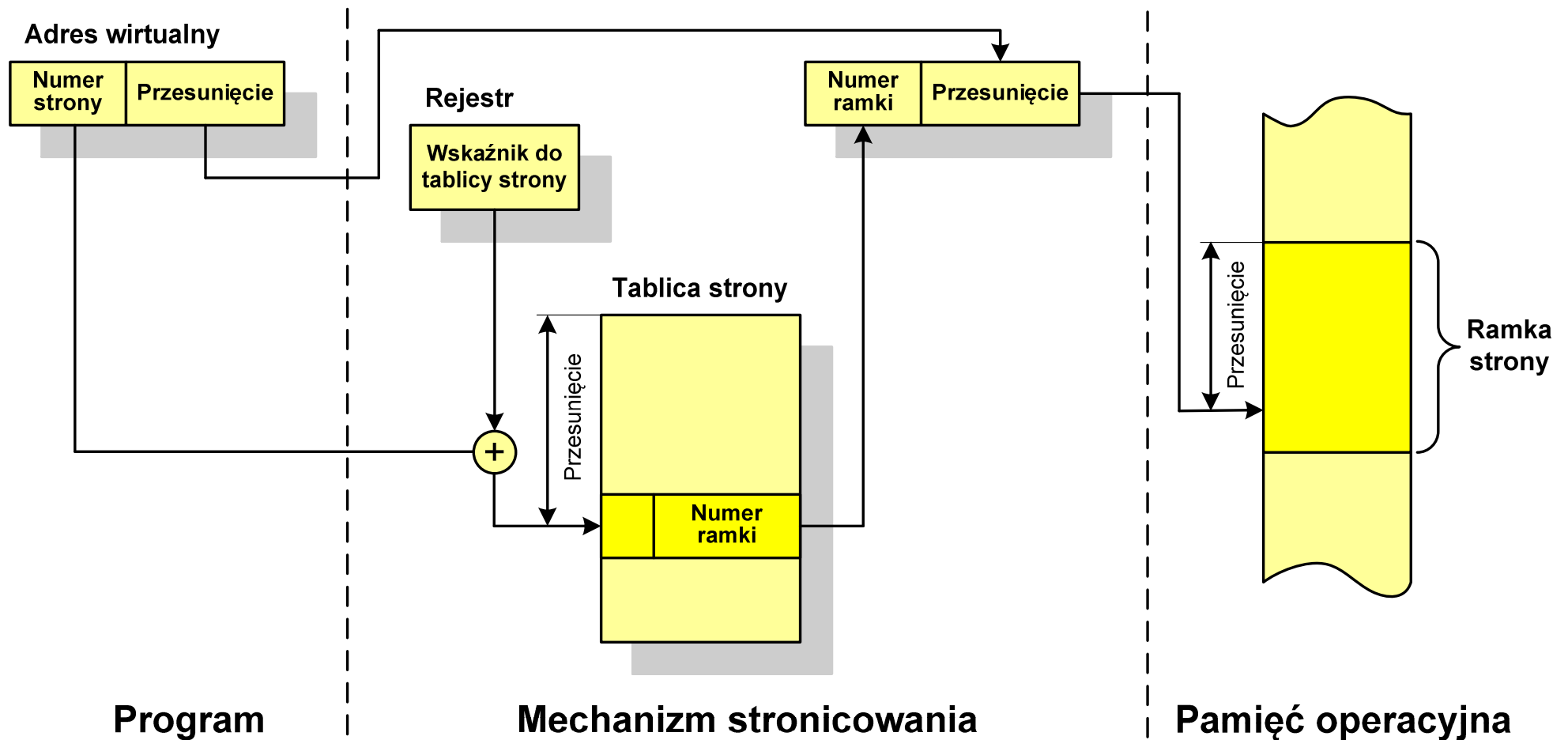
- generowanie przerwania sygnalizującego błąd w dostępie do pamięci
- zmiana stan procesu na zablokowany
- wstawienie do pamięci operacyjnej fragment procesu zawierający adres logiczny, który był przyczyną błędu
- zmiana stanu procesu na uruchomiony

Dzięki zastosowaniu pamięci wirtualnej:

- w pamięci operacyjnej może być przechowywanych więcej procesów
- proces może być większy od całej pamięci operacyjnej

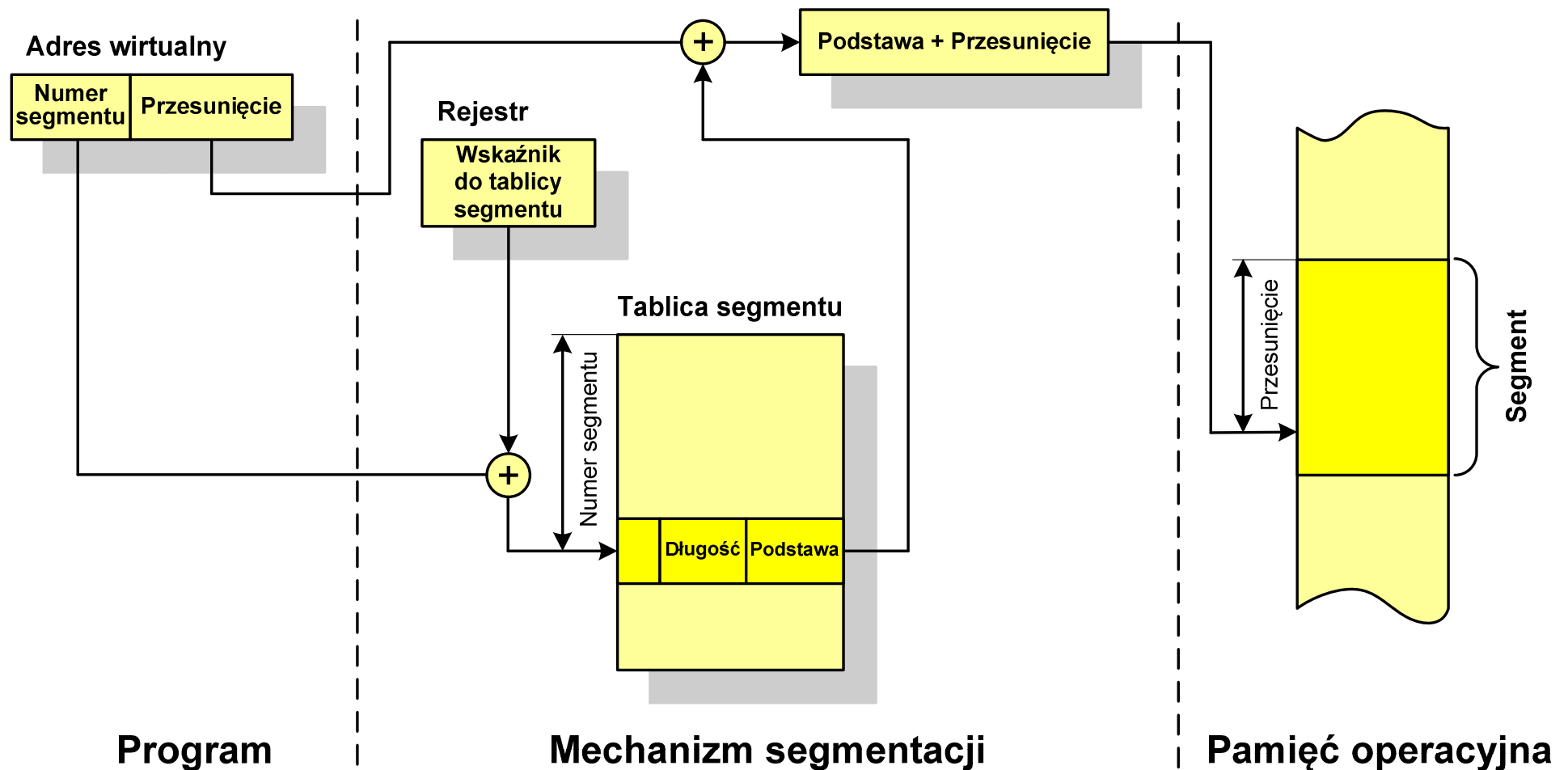
Stronicowanie pamięci wirtualnej

- odczytanie strony wymaga translacji adresu wirtualnego na fizyczny



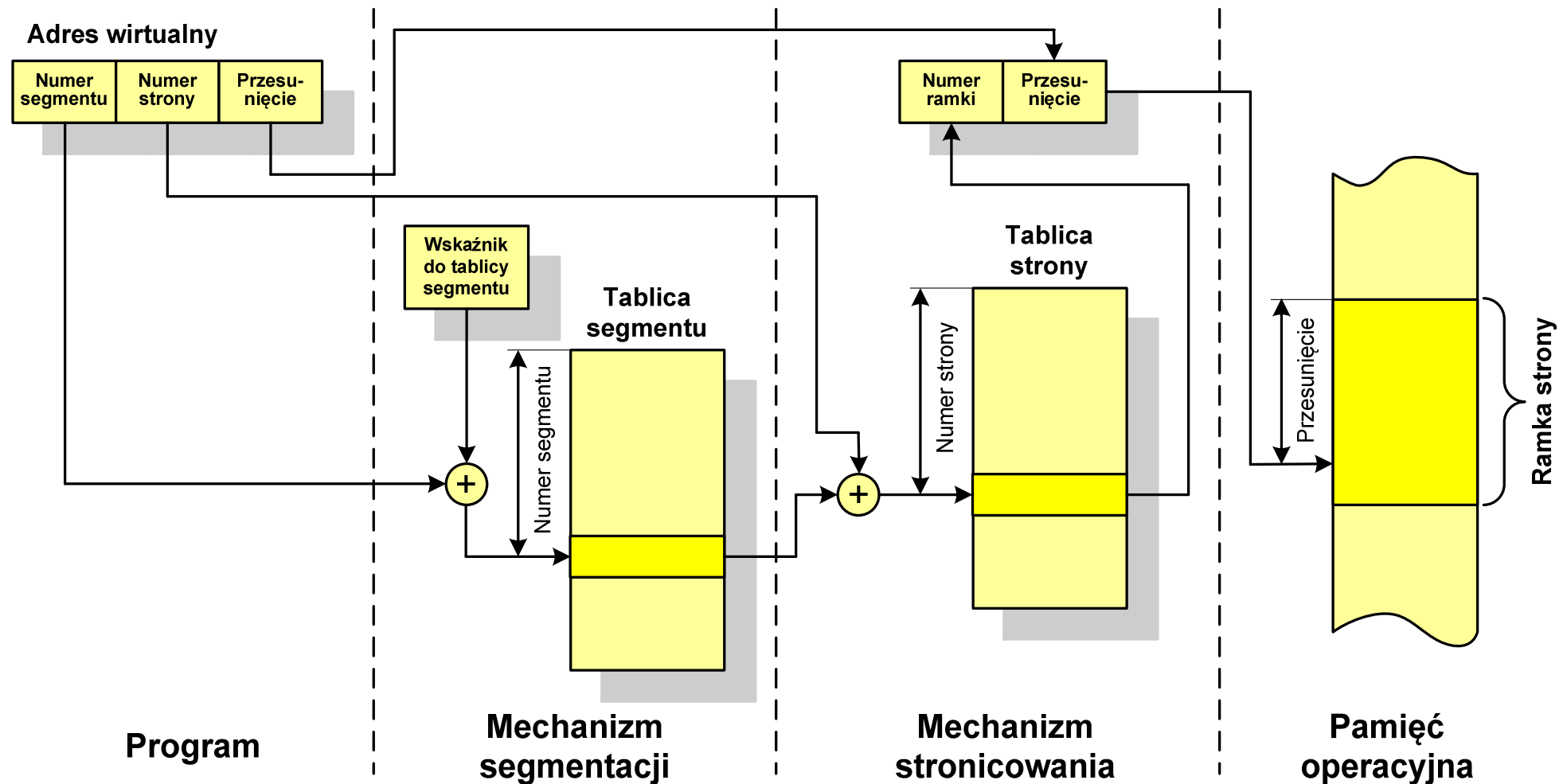
Segmentacja pamięci wirtualnej

- mechanizm odczytania słowa z pamięci obejmuje translację adresu wirtualnego na fizyczny za pomocą tablicy segmentu



Stronicowanie i segmentacja pamięci wirtualnej

- tłumaczenie adresu wirtualnego na adres fizyczny:



Koniec wykładu nr 7

Dziękuję za uwagę!