Informatyka 1 (EZ1E2008)

Politechnika Białostocka - Wydział Elektryczny

Elektrotechnika, semestr II, studia niestacjonarne I stopnia

Rok akademicki 2019/2020

Wykład nr 8 (15.05.2020)

dr inż. Jarosław Forenc

Plan wykładu nr 8

- Język C - tablice jednowymiarowe (wektory)
  - deklaracja, odwołania do elementów, inicjalizacja tablicy
  - generator liczb pseudolosowych, operacje na wektorze
- System operacyjny
  - definicje systemu operacyjnego
- Zarządzanie procesami
  - definicja procesu, dwu- i pięciostanowy model procesu
- Zarządzanie dyskowymi operacjami we-wy
  - metody przydziału pamięci dyskowej
  - struktura dysku twardego (MBR, GPT)
  - systemy plików (FAT12, FAT16, FAT32, exFAT, NTFS, ext2)
- Zarządzanie pamięcią operacyjną
  - partycjonowanie, stronicowanie, segmentacja, pamięć wirtualna

Język C - tablica elementów

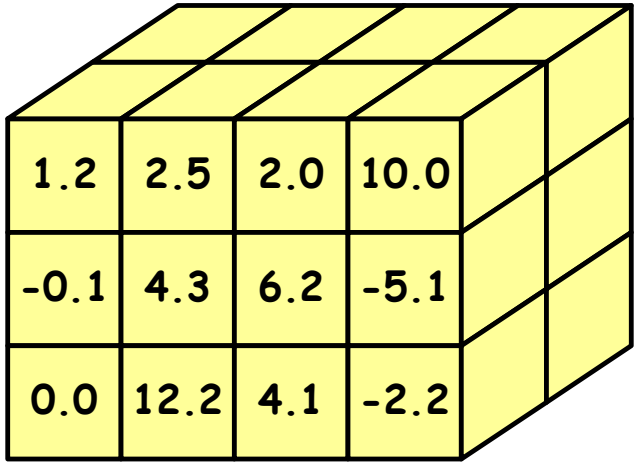
- Tablica - ciągły obszar pamięci, w którym umieszczone są elementy tego samego typu

wektor

5	3	-2	1	-4

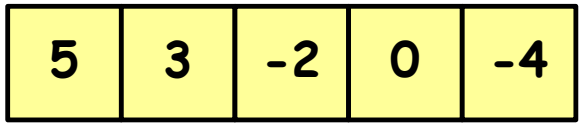
macierz

a	c	d	m
p	d	q	l
a	t	x	v

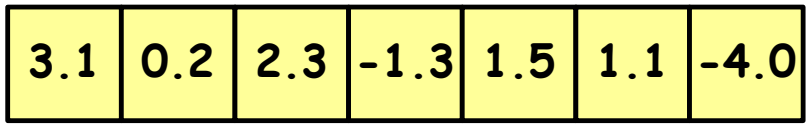


Język C - tablica jednowymiarowa

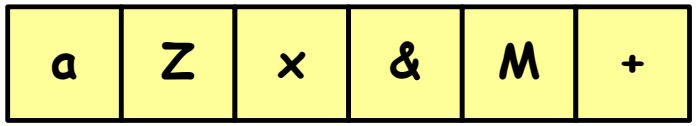
- Tablica - ciągły obszar pamięci, w którym umieszczone są elementy tego samego typu
- Wektor - tablica jednowymiarowa



- liczby całkowite



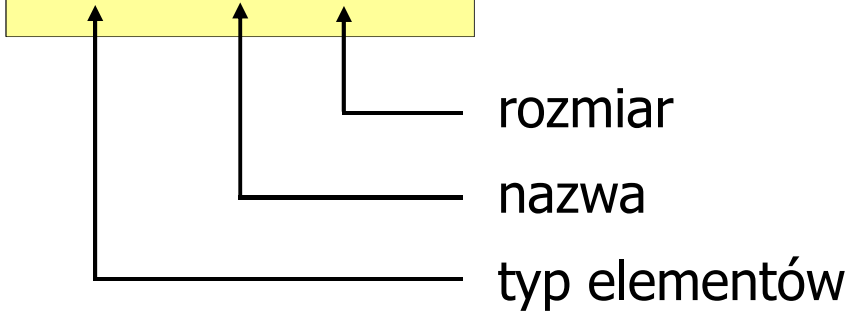
- liczby rzeczywiste

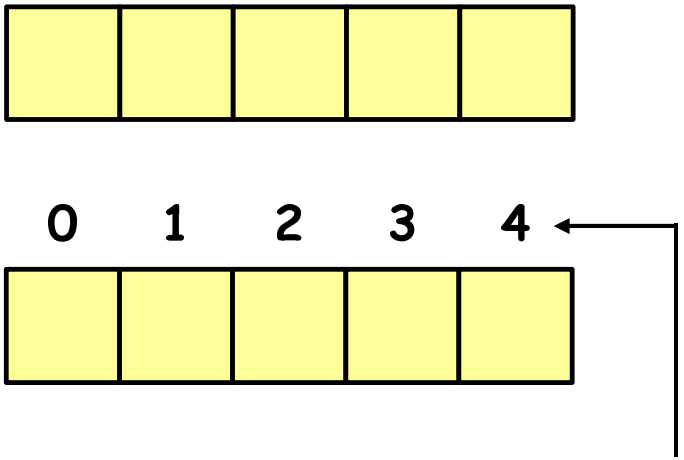


- znaki

Język C - deklaracja tablicy jednowymiarowej

```
int tab[5];
```





- rozmiar tablicy to wartość:
  - całkowita, dodatnia
  - znana na etapie kompilacji programu
    (stała liczbowa: 5, #define N 5, const int n = 5;)

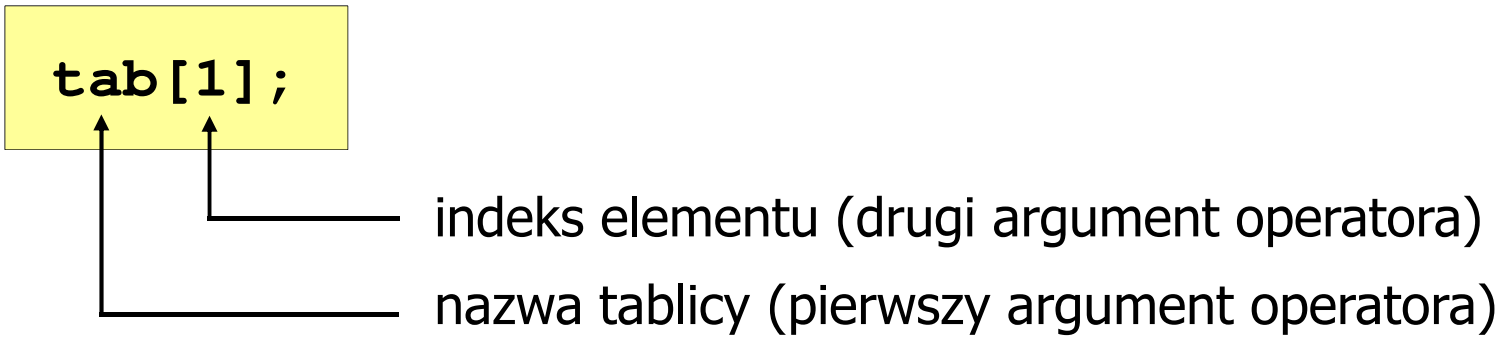
```
int tab[5];
```

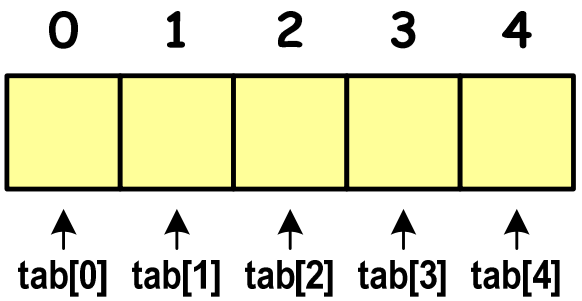
```
int tab[N];
```

```
int tab[n];
```

Język C - odwołania do elementów tablicy

[ ] - dwuargumentowy operator indeksowania

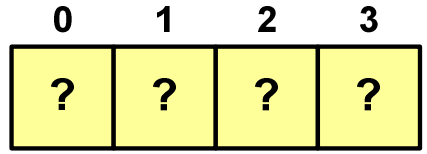




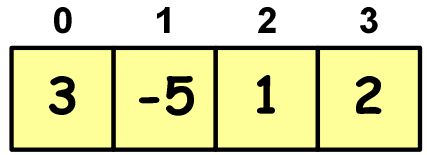
- indeks:
  - stała liczbowa, np. 0, 1, 10
  - nazwa zmiennej, np. i, idx
  - wyrażenie, np. i*j+5

Język C - odwołania do elementów tablicy

```
int tab[4];
```




```
tab[0] = 3;
tab[1] = -5;
tab[2] = 1;
tab[3] = 2;
```



- Każdy element tablicy traktowany jest tak samo jak zmienna typu int

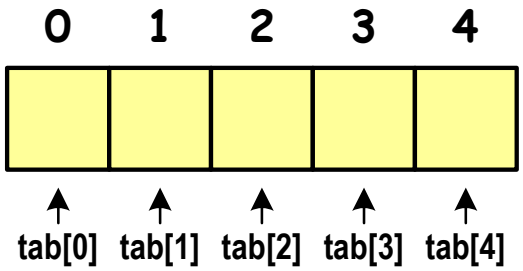
```
printf("%d",tab[0]);
```

```
scanf("%d",&tab[1]);
```

Język C - odwołania do elementów tablicy

- Przy odwołaniach do elementów tablicy kompilator nie sprawdza poprawności indeksów

```
int tab[5];

tab[5] = 10;
```



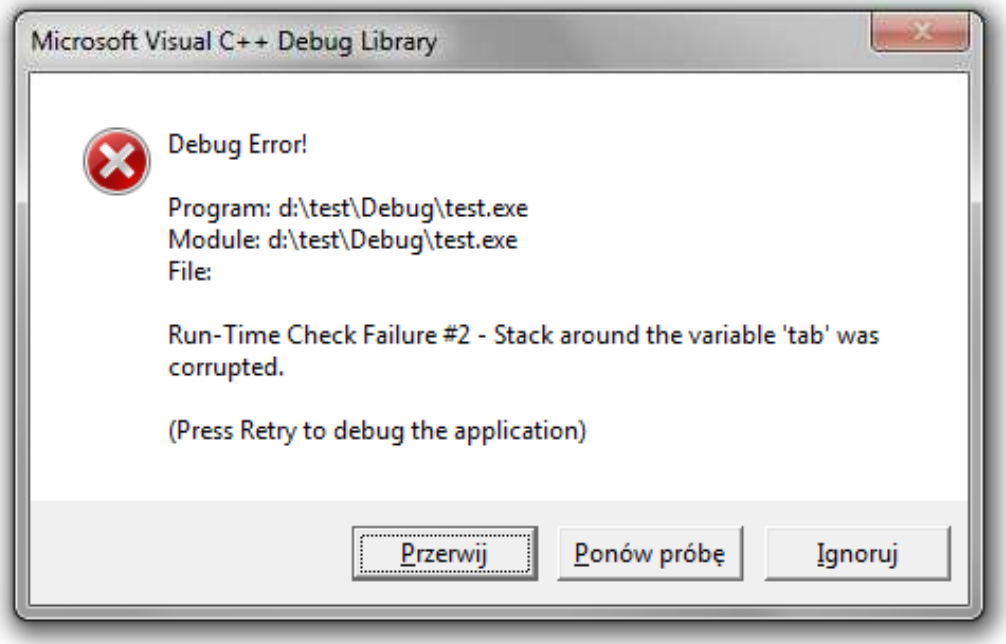
- błąd!!! - nie istnieje element tab[5]

- Kompilator nie zasygnalizuje błędu
- Program wykona operację
- Środowisko programistyczne może zasygnalizować problem

Język C - odwołania do elementów tablicy

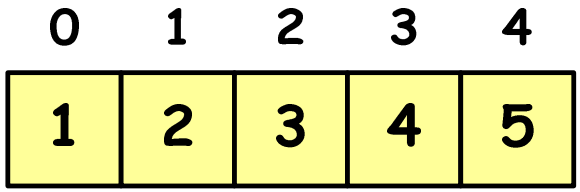
- Przy odwołaniach do elementów tablicy kompilator nie sprawdza poprawności indeksów

```
int tab[5];

tab[5] = 10;
```

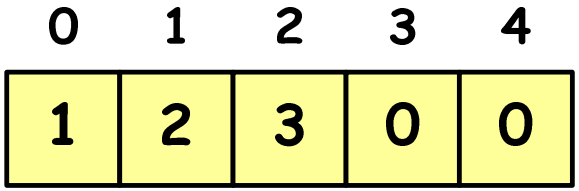


Język C - inicjalizacja tablicy jednowymiarowej

```
int tab[5] = {1,2,3,4,5};
```



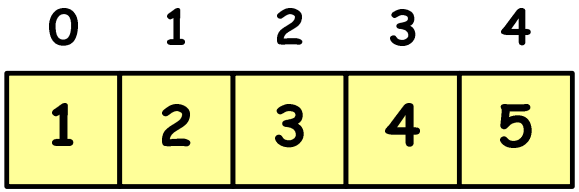

```
int tab[5] = {1,2,3};
```




```
int tab[5] = {1,2,3,4,5,6};
```

- błąd kompilacji

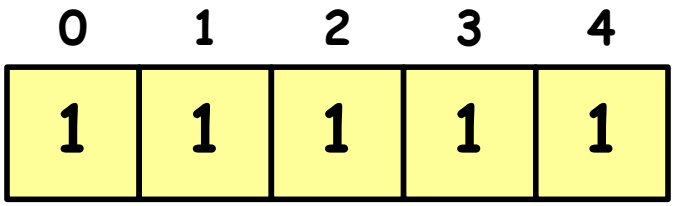
```
int tab[] = {1,2,3,4,5};
```



Język C - odwołania do elementów tablicy

- Zapisanie wartości 1 do wszystkich elementów tablicy

```
int tab[5];

tab[0] = 1;
tab[1] = 1;
tab[2] = 1;
tab[3] = 1;
tab[4] = 1;
```

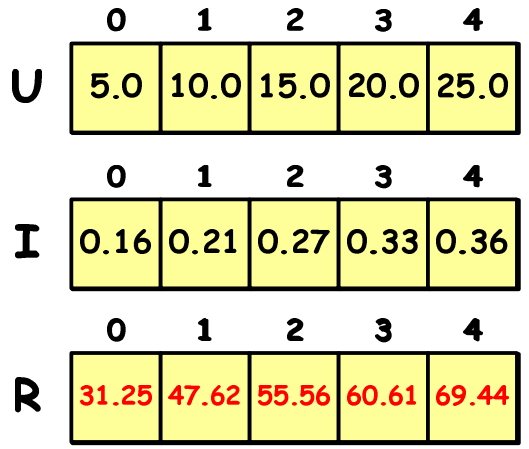



```
int tab[5], i;

for (i=0; i<5; i++)
   tab[i] = 1;
```

Język C - operacje na dużej ilości danych (tablica)

```
R1 = 31.250000
R2 = 47.619048
R3 = 55.555556
R4 = 60.606061
R5 = 69.444444
```

```c
#include <stdio.h>

int main(void)
{
    double U[5] = { 5.0, 10.0, 15.0, 20.0, 25.0 };
    double I[5] = { 0.16, 0.21, 0.27, 0.33, 0.36 };
    double R[5];
    int i;

    for (i=0; i<5; i++)
        R[i] = U[i]/I[i];

    for (i=0; i<5; i++)
        printf("R%d = %f\n",i+1,R[i]);

    return 0;
}
```

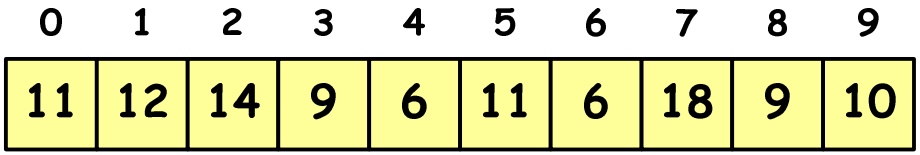


Język C - generator liczb pseudolosowych

- rand() - zwraca liczbę pseudolosową - zakres: 0 ... 32767
- srand() - inicjalizuje generator liczb pseudolosowych
- Plik nagłówkowy: stdlib.h (time.h)

```
int x, y;
srand((unsigned int) time(NULL));
x = rand();                 // zakres <0,32767>
y = rand() % 100;           // zakres <0,99>
```

Język C - operacje na wektorze

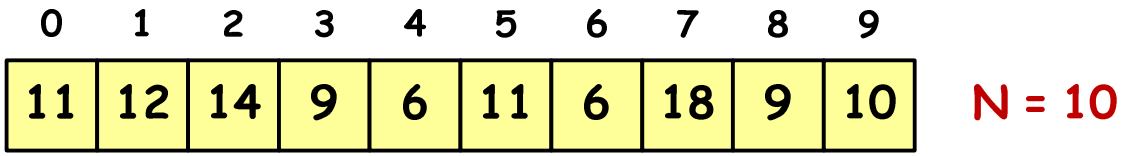



```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define N 10

int main(void)
{
      int tab[N], i;

      /* generowanie elementów tablicy */

      srand((unsigned int) time(NULL));

      for (i=0; i<N; i++)
            tab[i] = rand() % 20;
```

Język C - operacje na wektorze

```
/* wyświetlenie elementów tablicy */

printf("Elementy tablicy:\n");
for (i=0; i<N; i++)
    printf("%d  ",tab[i]);
printf("\n");
```

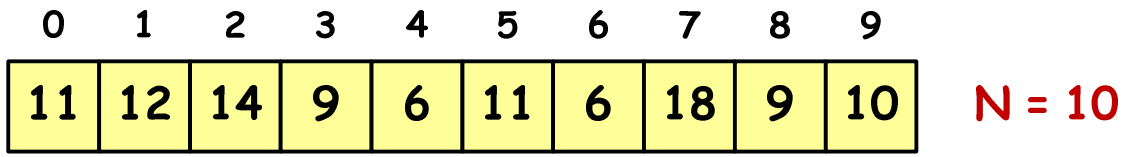
```
Elementy tablicy:
11  12  14  9  6  11  6  18  9  10
```



Język C - operacje na wektorze

```
/* wyświetlenie elementów w odwrotnej kolejności */

printf("Elementy w odwrotnej kolejnosci:\n");
for (i=N-1; i>=0; i--)
    printf("%d  ",tab[i]);
printf("\n");
```

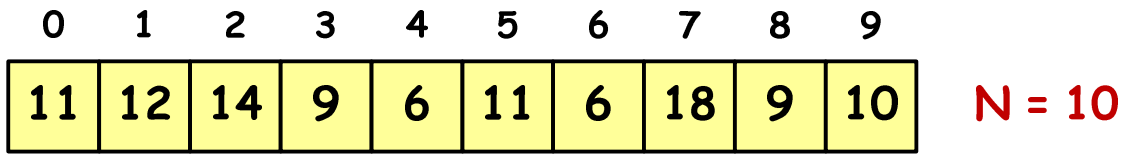
```
Elementy w odwrotnej kolejnosci:
10  9  18  6  11  6  9  14  12  11
```



Język C - operacje na wektorze

```
/* wyszukanie elementu o najmniejszej wartości */

int min;

min = tab[0];
for (i=1; i<N; i++)
    if (tab[i]<min)
        min = tab[i];
printf("Wartosc elementu najmniejszego: %d\n",min);
```

```
Wartosc elementu najmniejszego: 6
```



Język C - operacje na wektorze

```
/* indeksy elementów o najmniejszej wartości */

printf("Indeksy elementu najmniejszego: ");
for (i=0; i<N; i++)
    if (tab[i]==min)
        printf("%d  ",i);
printf("\n");
```

```
Indeksy elementu najmniejszego: 4  6
```

0	1	2	3	4	5	6	7	8	9
11	12	14	9	6	11	6	18	9	10

N = 10

Język C - operacje na wektorze

```
/* suma i średnia arytmetyczna elementów tablicy */

int suma = 0;
float srednia;

for (i=0; i<N; i++)
    suma = suma + tab[i];
srednia = (float) suma/N;
printf("Suma: %d, srednia: %g\n",suma,srednia);
```

```
Suma: 106, srednia: 10.6
```

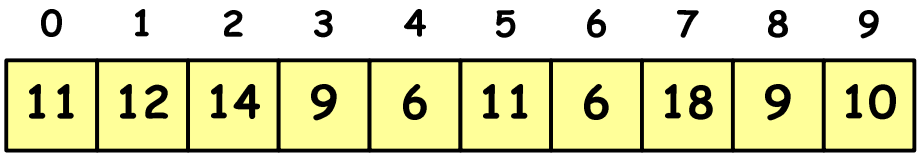
0	1	2	3	4	5	6	7	8	9
11	12	14	9	6	11	6	18	9	10

N = 10

Język C - operacje na wektorze

```
/* liczba parzystych elementów tablicy  */

int ile = 0;

for (i=0; i<N; i++)
    if (tab[i]%2==0)
        ile++;
printf("Liczba parzystych elementow: %d\n",ile);
```

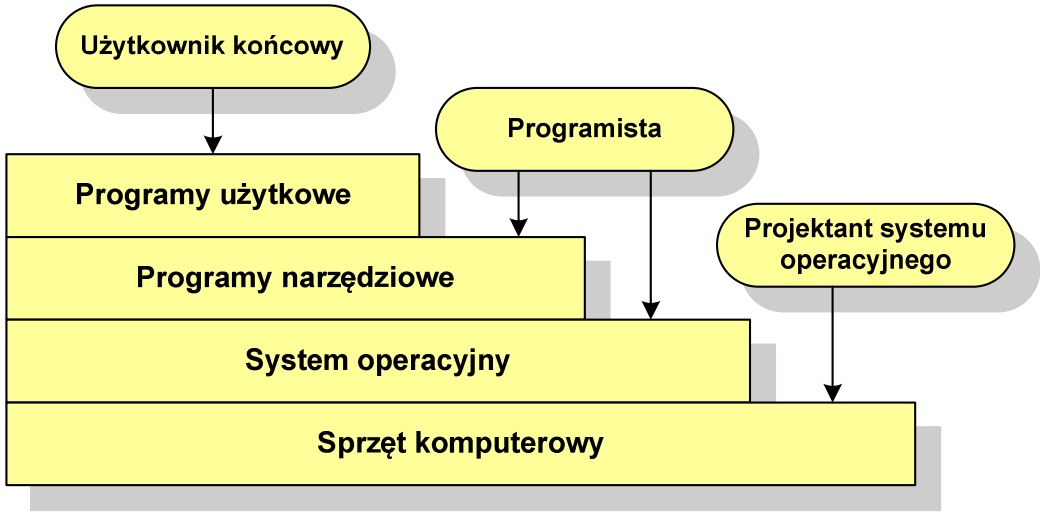
```
Liczba parzystych elementow: 6
```



System operacyjny - definicja

- System operacyjny - jest to program sterujący wykonywaniem aplikacji i działający jako interfejs pomiędzy aplikacjami (użytkownikiem) a sprzętem komputerowym

- użytkownik końcowy nie jest zainteresowany sprzętem, interesują go tylko aplikacje (programy użytkowe)
- aplikacje są tworzone przez programistów za pomocą języków programowania



System operacyjny - definicja

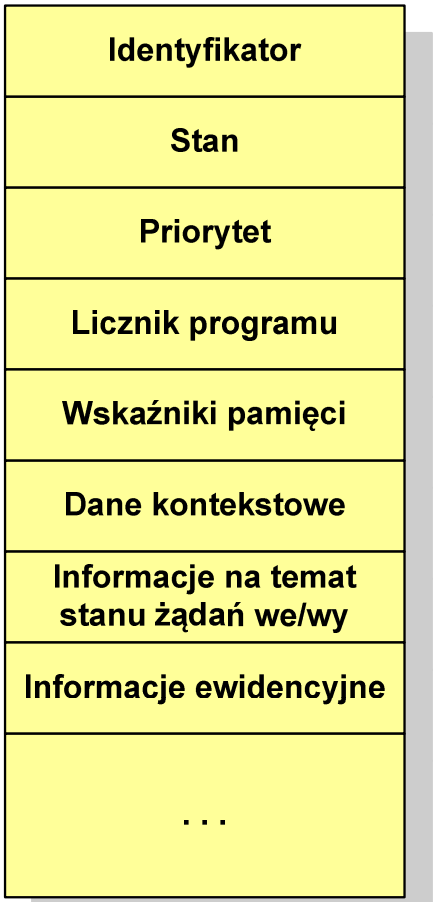
- System operacyjny - administrator zasobów - zarządza i przydziela zasoby systemu komputerowego oraz steruje wykonaniem programu
- zasób systemu - każdy element systemu, który może być przydzielony innej części systemu lub oprogramowaniu aplikacyjnemu
- do zasobów systemu zalicza się:
  - czas procesora
  - 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

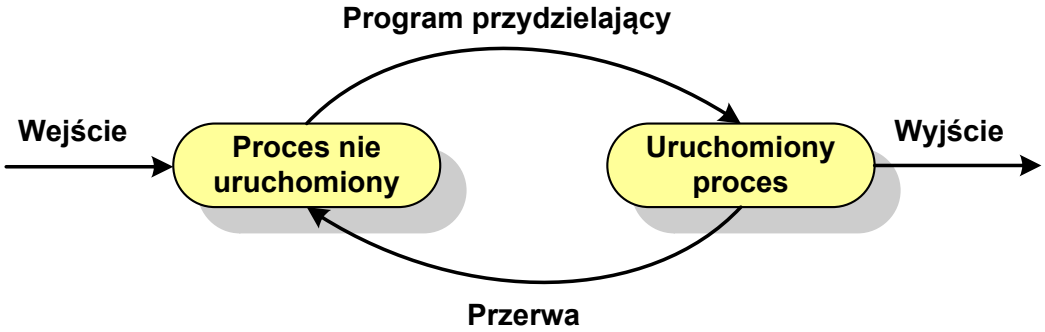
Blok kontrolny procesu

- struktura danych tworzona i zarządzana przez system operacyjny, a opisująca właściwości procesu
- identyfikator - unikatowy numer skojarzony z procesem, dzięki któremu można odróżnić go od innych procesów
- stan procesu: nowy, gotowy, uruchomiony, zablokowany, anulowany
- priorytet - niski, normalny, wysoki, czasu rzeczywistego
- licznik programu - adres kolejnego rozkazu w programie, który ma zostać wykonany
- wskaźniki pamięci - wskaźniki do kodu programu, danych skojarzonych z procesem, dodatkowych bloków pamięci
- dane kontekstowe - dane znajdujące się w rejestrach procesora, gdy proces jest wykonywany
- informacje na temat stanu żądań we-wy - informacje na temat urządzeń we-wy przypisanych do tego procesu

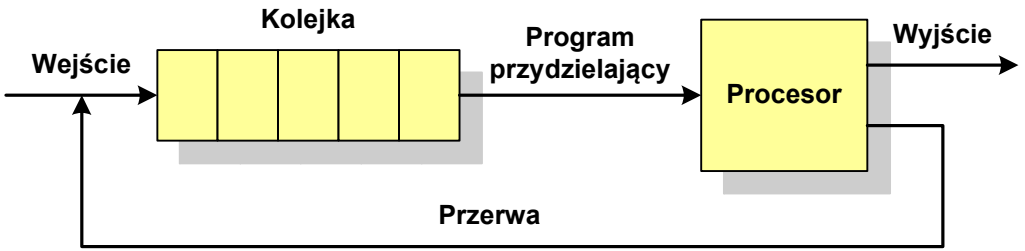


Dwustanowy model procesu

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



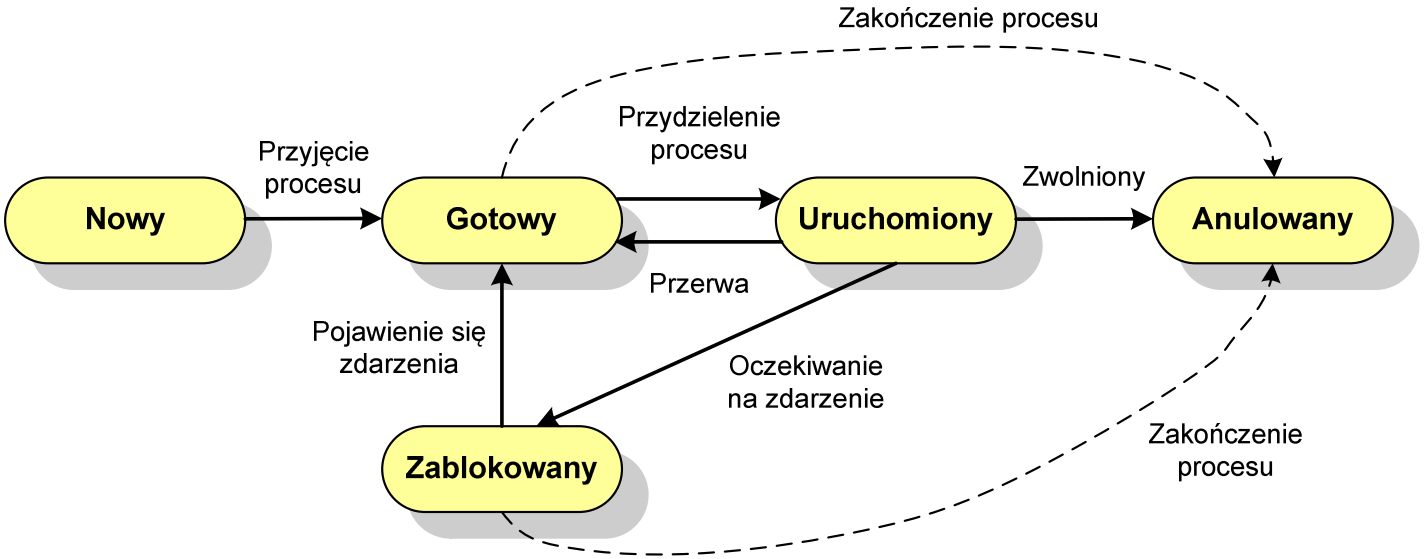
- procesy, które nie są uruchomione czekają w kolejce na wykonanie



- wadą tego modelu jest sytuacja, gdy kolejny proces pobierany do wykonania z kolejki jest zablokowany, gdyż oczekuje na zakończenie operacji we-wy

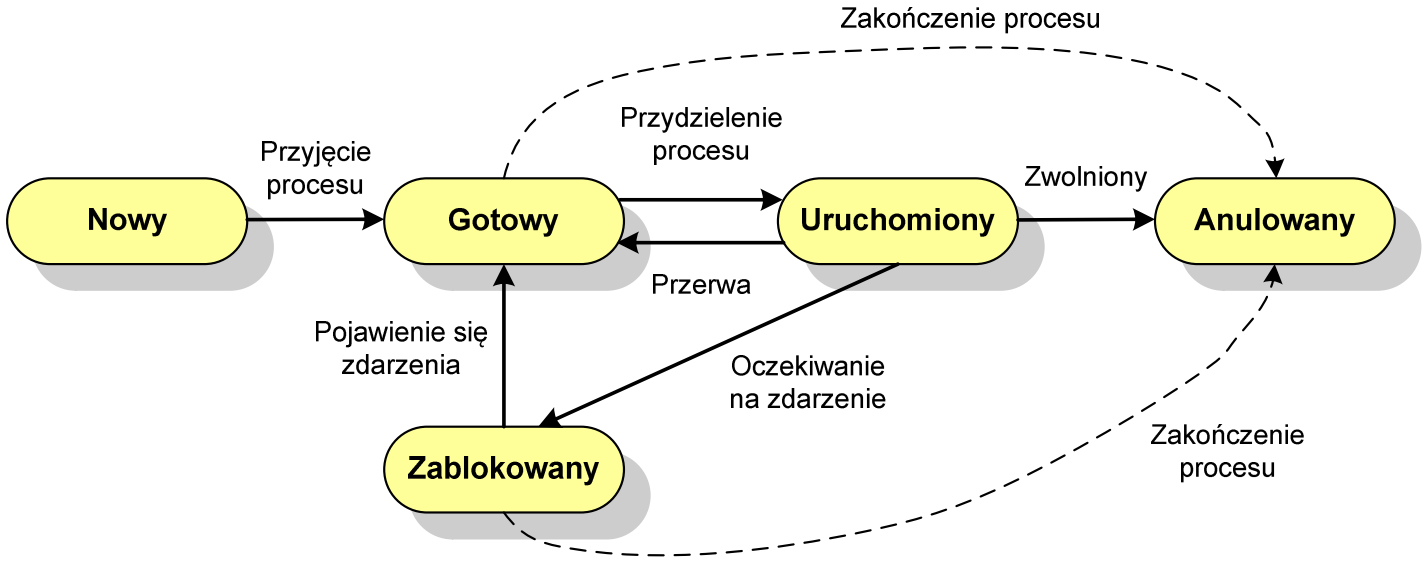
Pięciostanowy model procesu

- rozwiązaniem powyższego problemu jest podział procesów nieuruchomionych na gotowe do wykonania i zablokowane



- pięciostanowy model procesu wymaga zastosowania minimum dwóch kolejek: dla procesów gotowych do wykonania i zablokowanych

Pięciostanowy model procesu



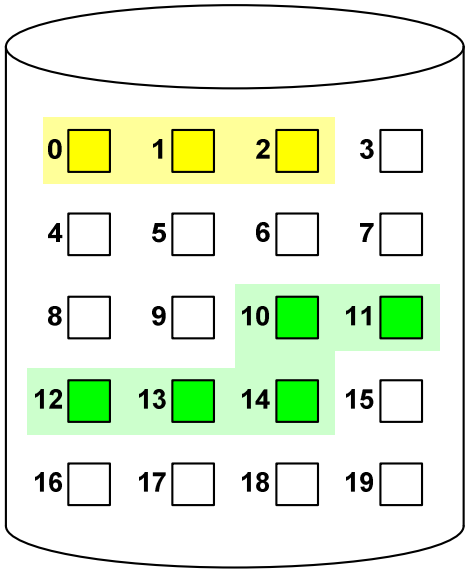
- uruchomiony - proces aktualnie wykonywany
- gotowy - proces gotowy do wykonania przy najbliższej możliwej okazji
- zablokowany - proces oczekujący na zakończenie operacji we-wy
- nowy - proces, który właśnie został utworzony (ma utworzony blok kontrolny procesu, nie został jeszcze załadowany do pamięci), 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

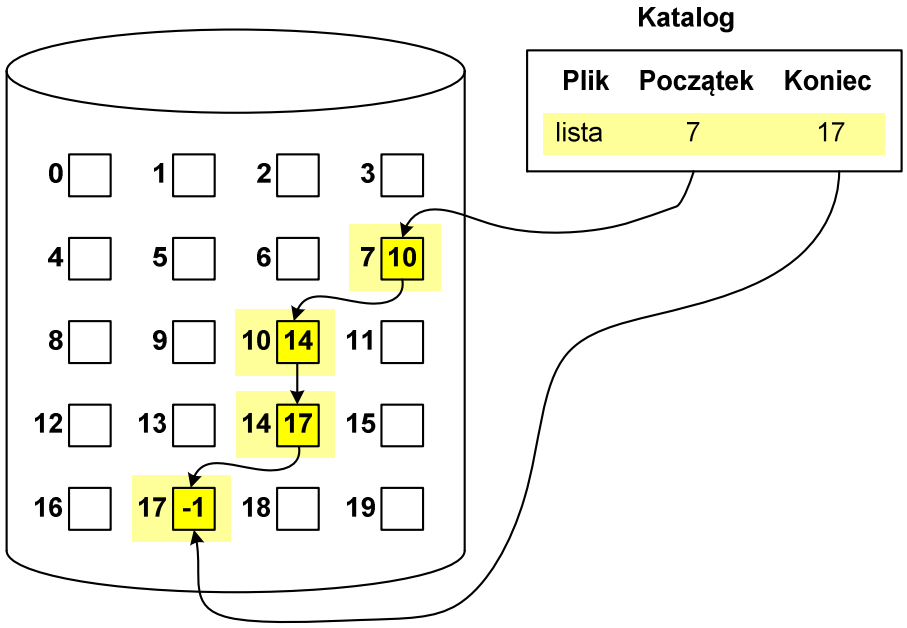


Katalog

Plik	Początek	Długość
lista	0	3
poczta	10	5

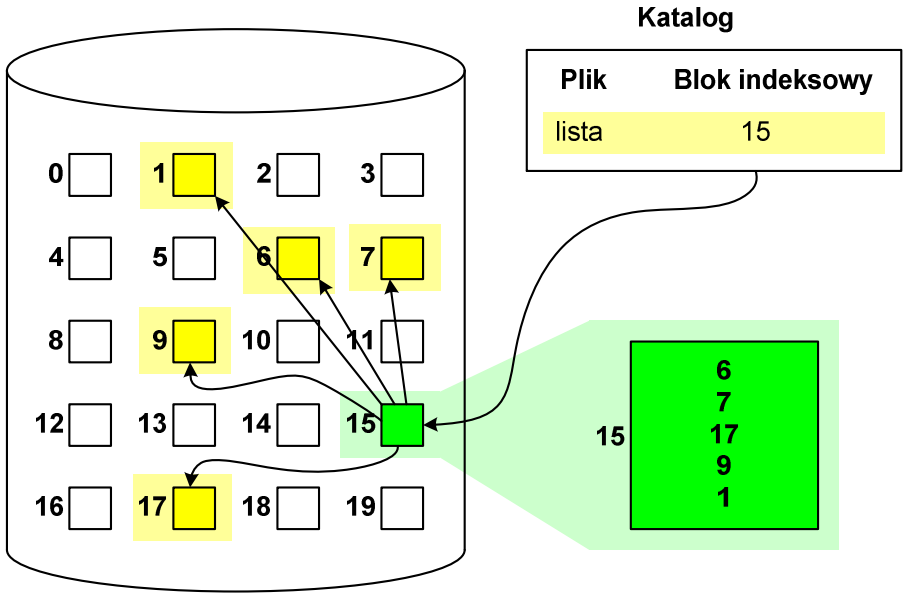
Przydział pamięci dyskowej - alokacja listowa

- 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



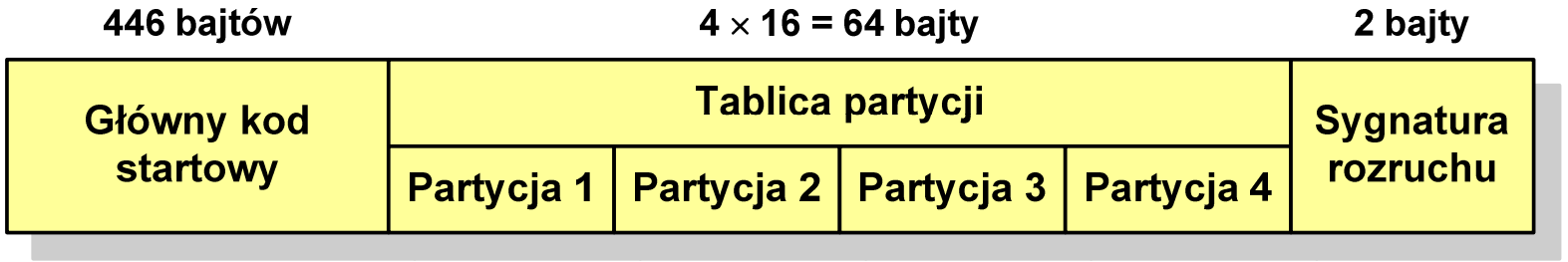
Przydział pamięci dyskowej - alokacja indeksowa

- 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



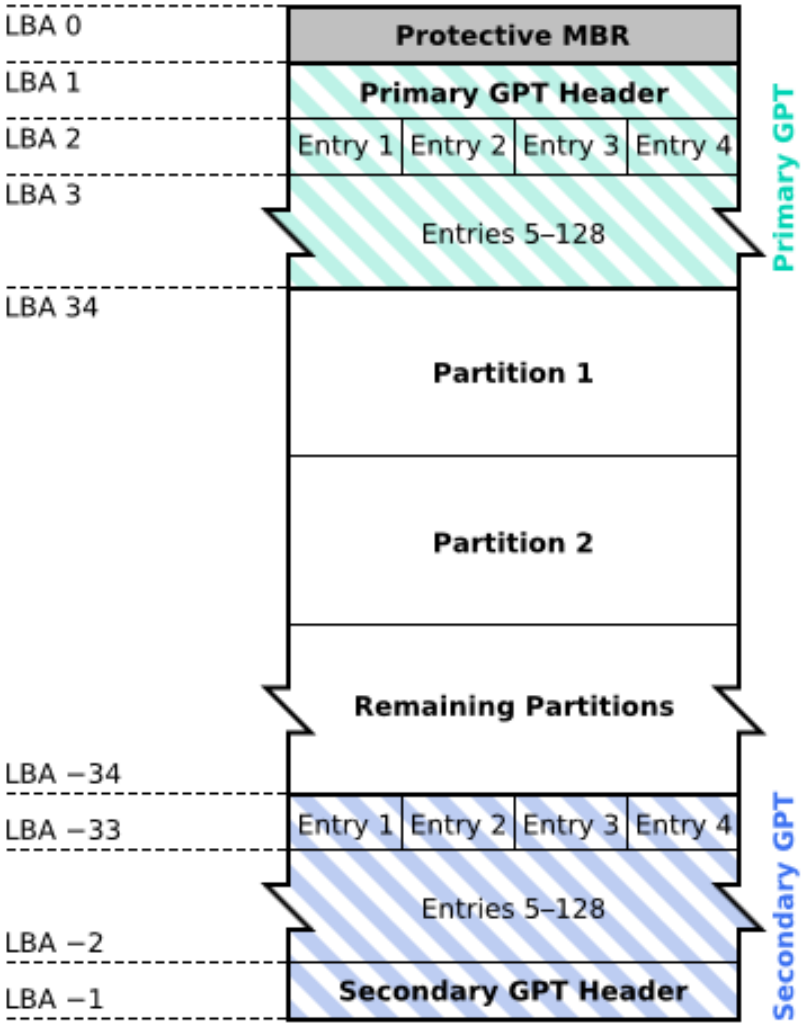
- główny kod startowy (Master Boot Code, bootloader) - program odszukujący i ładujący do pamięci zawartość pierwszego sektora aktywnej partycji
- tablica partycji - cztery 16-bajtowe rekordy opisujące partycje na dysku
  - zawartość i organizacja tablicy jest niezależna od systemu operacyjnego
  - maksymalny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

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

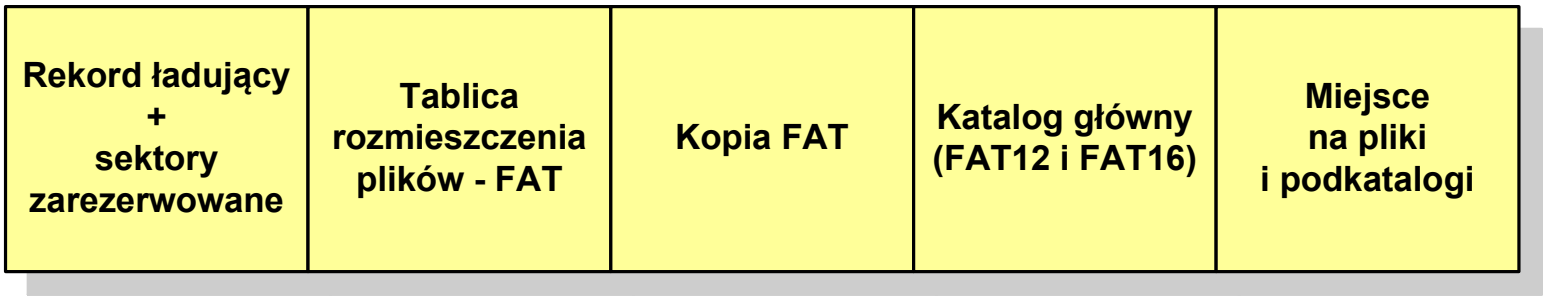
Struktura dysku twardego - GPT (struktura)

- 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



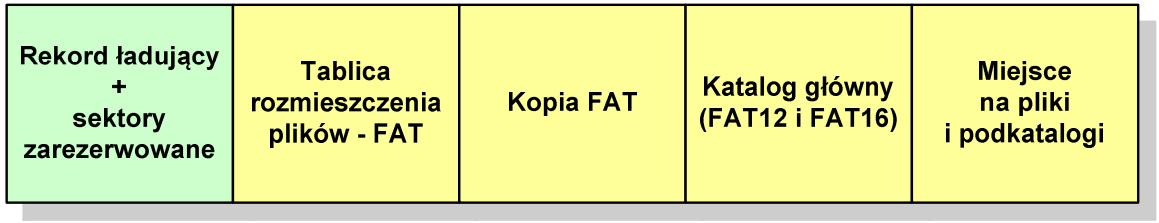
System plików FAT (File Allocation Table)

- 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 (praktycznie 28)
  - 64 bity w systemie exFAT (FAT64)
- ogólna struktura dysku logicznego / dyskietki w systemie FAT:



FAT12

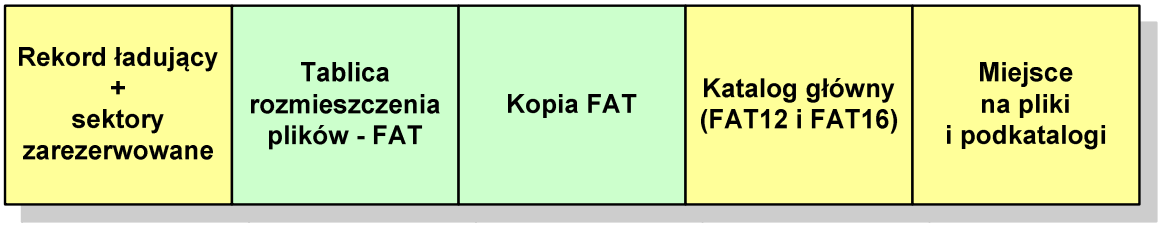
- system plików FAT12 przeznaczony jest dla nośników o małej pojemności
- obsługuje $2^{12} = 4096$ jednostek alokacji, max. rozmiar partycji to 16 MB
- rekord ładujący zajmuje pierwszy sektor dyskietki lub dysku logicznego



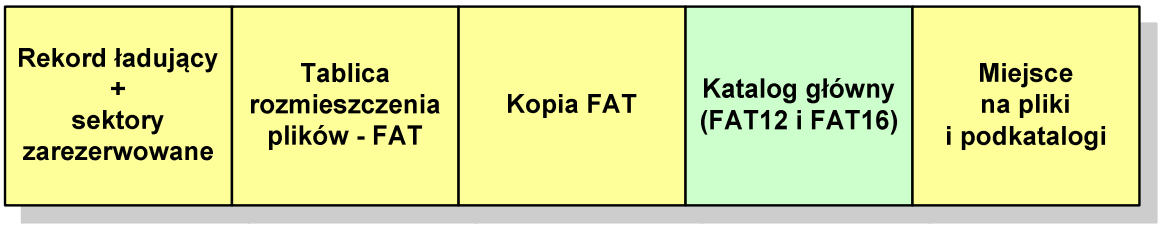
- rekord ładujący zawiera następujące dane:
  - instrukcja skoku do początku programu ładującego (3 bajty)
  - nazwa wersji systemu operacyjnego (8 bajtów)
  - struktura BPB (ang. BIOS Parametr Block) - blok parametrów BIOS (25 bajtów)
  - rozszerzony BPB (ang. Extended BPB, 26 bajtów)
  - wykonywalny 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



FAT12

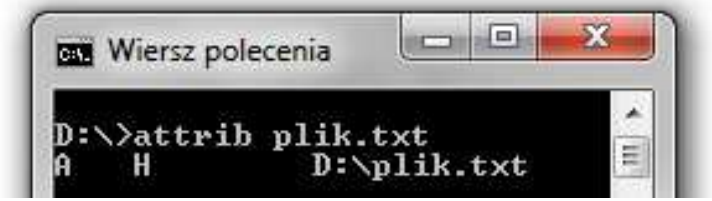
- katalog główny zawiera 32-bajtowe pola mogące opisywać pliki, podkatalogi lub etykietę dysku

Zawartość pola:

Bajty	Rozmiar	Zawartość
00H-07H	8	Nazwa pliku w kodach ASCII
08H-0AH	3	Rozszerzenie nazwy pliku
0BH	1	Atrybuty pliku
0CH-15H	10	Zarezerwowane
16H-17H	2	Czas utworzenia lub aktualizacji pliku
18H-19H	2	Data utworzenia lub aktualizacji pliku
1AH-1BH	2	Numer pierwszej JAP
1CH-1DH	2	Mniej znaczące słowo rozmiaru pliku
1EH-1FH	2	Bardziej znaczące słowo rozmiaru pliku

Atrybuty pliku:

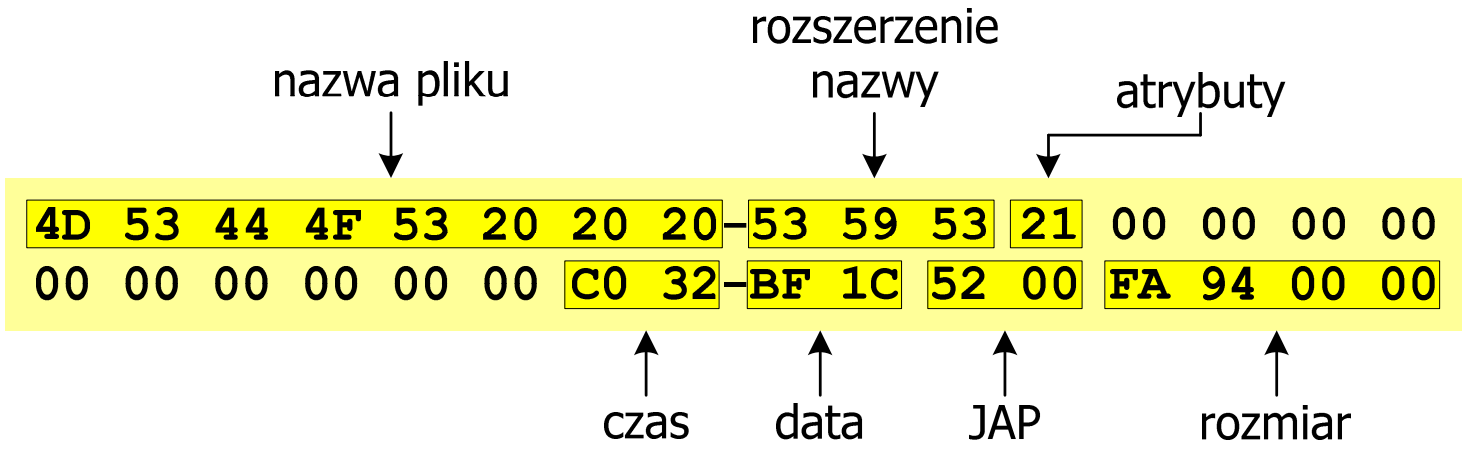
Bit	Znaczenie
0	Plik tylko do odczytu (read only)
1	Plik ukryty (hidden)
2	Plik systemowy (system)
3	Etykieta dysku (volume label)
4	Podkatalog
5	Plik archiwalny (archive)
6,7	Nie wykorzystywane



FAT12

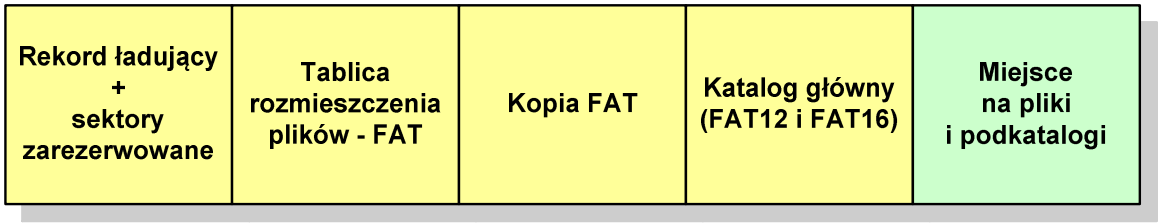
- przykładowa zawartość katalogu głównego:

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



FAT12

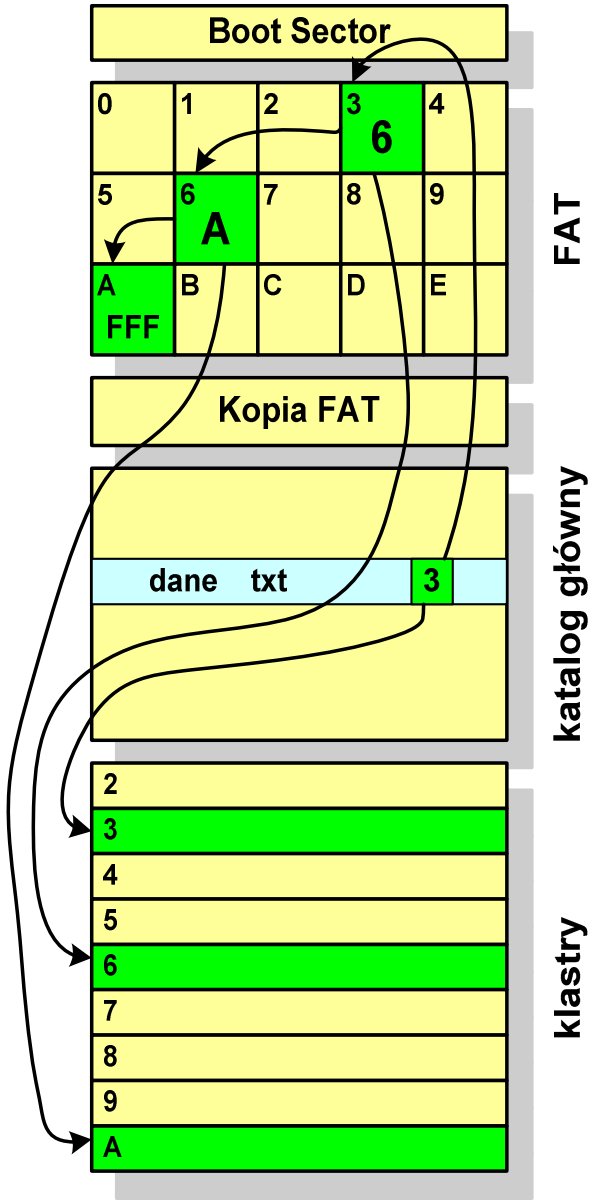
- pozostałą część dysku zajmuje miejsce na pliki i podkatalogi



- podkatalogi nie są ograniczone co do wielkości, zapisywane są na dysku w sposób identyczny jak pliki użytkowe i także zawierają 32-bajtowe pola

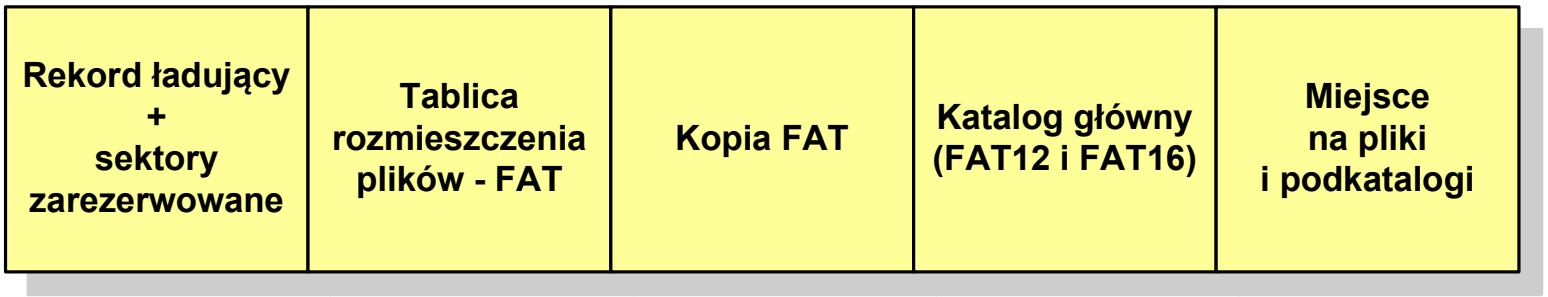
FAT12 - położenie pliku na dysku

- 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



FAT16

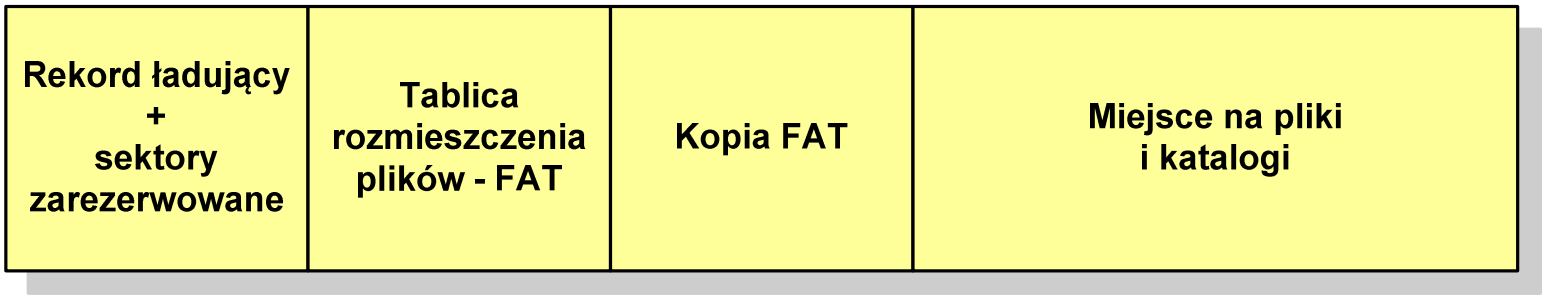
- po raz pierwszy pojawił się w systemie MS-DOS 3.3
- ogólna struktura dyskietki / dysku logicznego w systemie FAT16 jest taka sama jak w przypadku FAT12



- maksymalna liczba JAP ograniczona jest do 2^{16} czyli 65536
- maksymalny rozmiar dysku logicznego:
  - DOS, Windows 95 - ok. 2 GB (gdyż maksymalny rozmiar JAP to 2^{15} bajtów)
  - Windows 2000 - ok. 4 GB (gdyż maksymalny rozmiar JAP to 2^{16} bajtów)

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



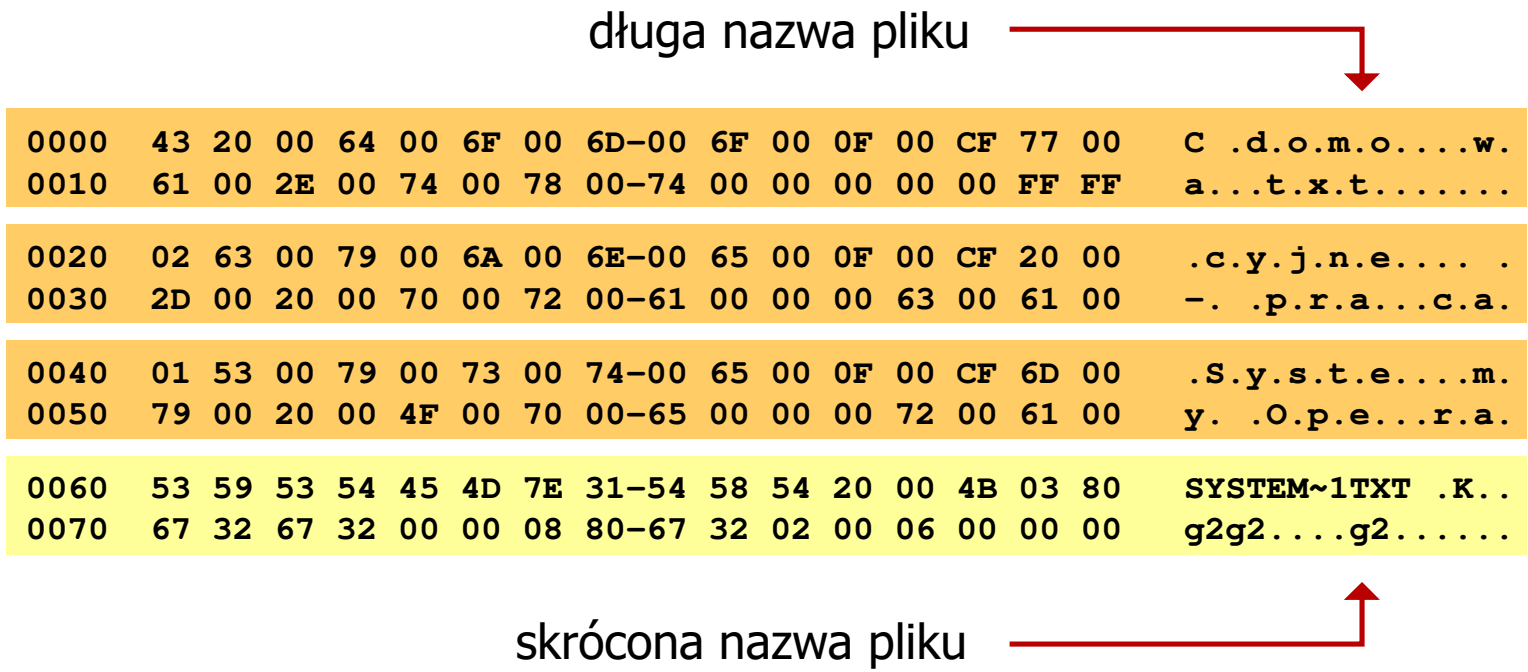
- do adresowania JAP stosuje się, obcięty o 4 najstarsze bity, adres 32-bitowy i dlatego dysk z FAT32 może zawierać maksymalnie 2^{28} JAP
- w systemie FAT32 można formatować tylko dyski, nie można natomiast zainstalować go na dyskietkach

FAT32 - długie nazwy plików

- wprowadzone w systemie Windows 95
- informacje o nazwie pliku zapamiętywane są jako:
  - długa nazwa
  - skrócona nazwa (tzw. alias długiej nazwy)
- długie nazwy plików zapisywane są także w 32-bajtowych strukturach, przy czym jedna nazwa zajmuje kilka struktur (w jednej strukturze umieszczonych jest 13 kolejnych znaków w formacie Unicode)
- skrócona nazwa pliku przechowywana jest w identycznej, 32-bajtowej, strukturze jak w przypadku plików w starym formacie 8+3
  - rozszerzenie długiej nazwy staje się rozszerzeniem skróconej nazwy
  - pierwsze 6 znaków długiej nazwy staje się pierwszymi sześcioma znakami skróconej nazwy (niedozwolone znaki zamieniane są na znak podkreślenia, małe litery zamieniane są na wielkie litery)
  - pozostałe dwa znaki nazwy skróconej to ~1 lub jeśli plik o takiej nazwie istnieje ~2, itd.

FAT32 - długie nazwy plików

- Nazwa pliku: Systemy Operacyjne - praca domowa.txt



FAT - wady systemu plików FAT

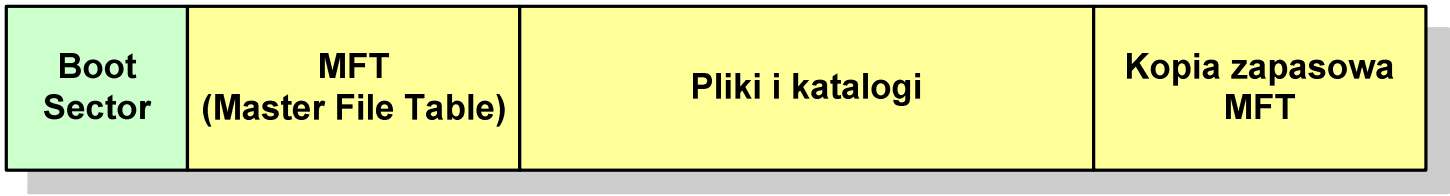
- fragmentacja wewnętrzna - nawet najmniejszy plik zajmuje całą JAP - gdy rozmiar klastra jest duży, a na dysku znajduje się dużo małych plików - pewna część miejsca jest tracona
- fragmentacja zewnętrzna - silna fragmentacja plików pomiędzy wiele klastrów o bardzo różnym fizycznym położeniu na dysku (konieczność okresowej defragmentacji przy użyciu specjalnych narzędzi programowych)
- duże prawdopodobieństwo powstawania błędów zapisu, polegających na przypisaniu jednego klastra dwóm plikom (tzw. crosslinks), co kończy się utratą danych z jednego lub obu „skrzyżowanych" plików
- typowym błędem, pojawiającym się w systemie FAT, jest również pozostawianie tzw. zagubionych klastrów (lost chains), tj. jednostek alokacji nie zawierających informacji, ale opisanych jako zajęte
- brak mechanizmów ochrony - praw dostępu

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 EB
  - maksymalna wielkość klastra - do 32 MB
  - nieograniczona liczba plików w pojedynczym katalogu
  - prawa dostępu do plików i katalogów

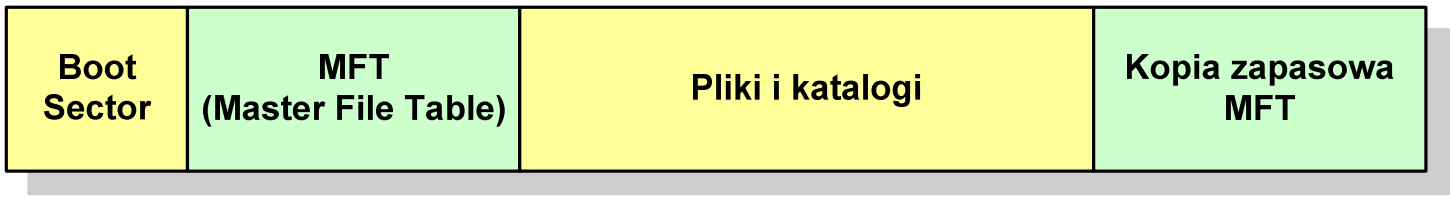
NTFS (New Technology File System)

- 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

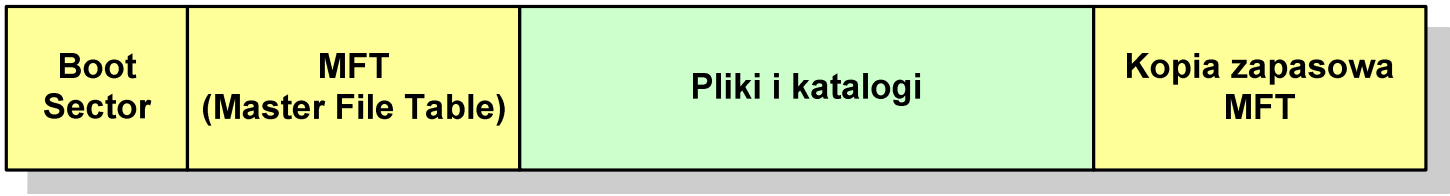
NTFS



- 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

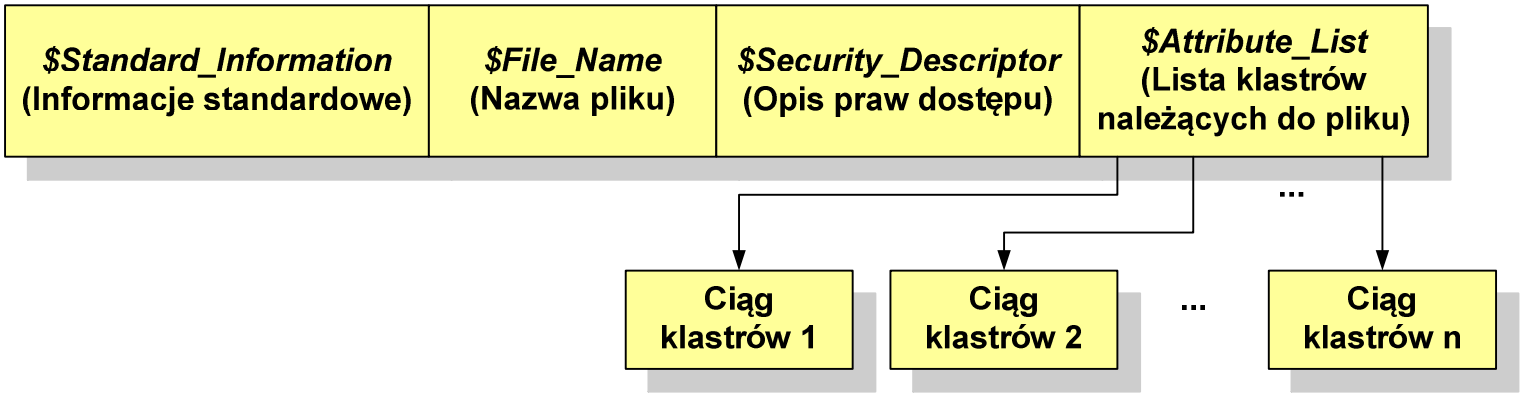
- pliki w systemie NTFS są reprezentowane w MFT przez rekord zawierający atrybuty:
  - $Standard_Information
  - $File_Name
  - $Security_Descriptor
  - $Data

$Standard_Information **(Informacje standardowe)**	***$File_Name*** **(Nazwa pliku)**	***$Security_Descriptor*** **(Opis praw dostępu)**	***$Data*** **(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

$Standard_Information (Informacje standardowe)	*$File_Name* (Nazwa pliku)	*$Security_Descriptor* (Opis praw dostępu)	*$Index_Root*	*$Index_Allocation*	*$Bitmap*

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

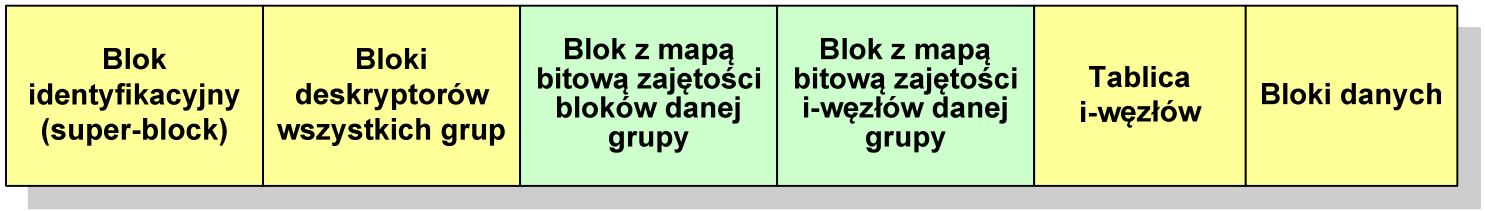
ext2

- 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



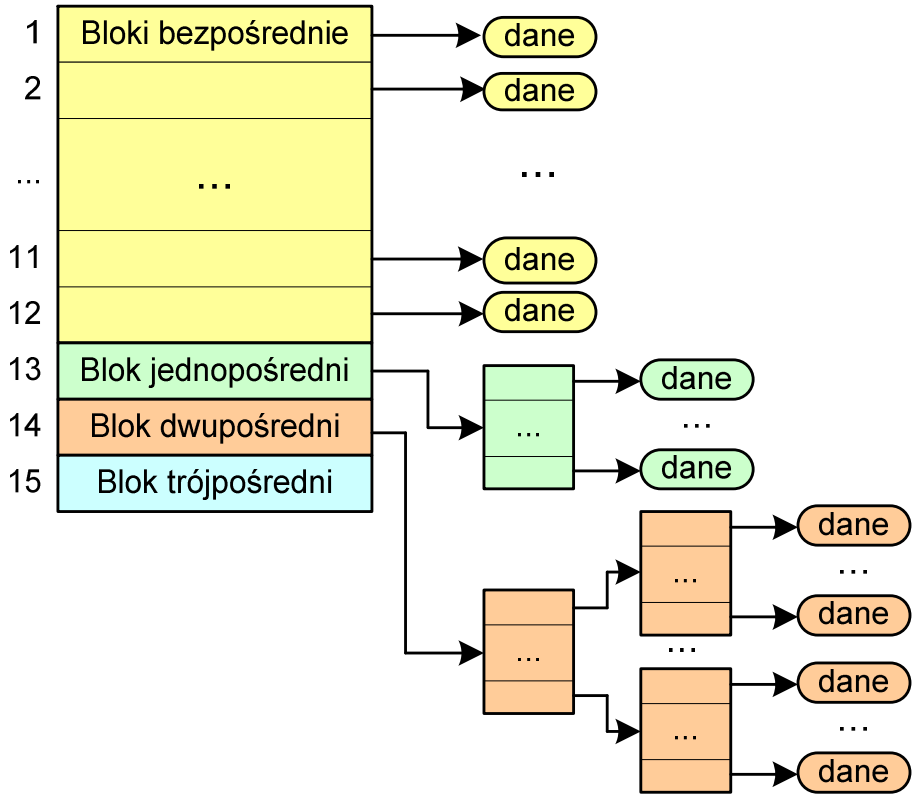
- blok z mapą bitową zajętości bloków danej grupy jest tablicą bitów o rozmiarze jednego bloku
  - jeśli blok ma rozmiar 1 kB to pojedynczą mapą można opisać fizyczna grupę 8096 bloków czyli 8 MB danych
  - jeśli natomiast blok ma rozmiar 4 kB, to fizyczna grupa bloków zajmuje 128 MB danych
- przed tablicą i-węzłów znajduje się blok z mapa 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 jednopośredni (ang. single indirect block)
  - 1 adres bloku zawierającego adresy bloków jednopośrednich - blok dwupośredni (ang. double indirect block)
  - 1 adres bloku zawierającego adresy bloków dwupośrednich - blok trójpośredni (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)

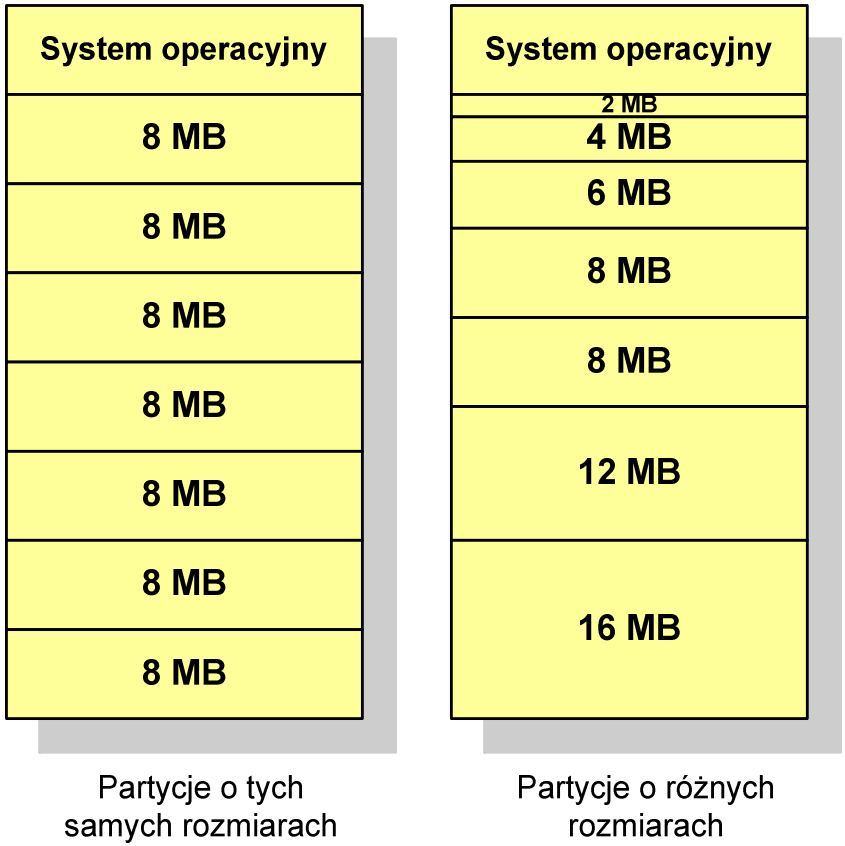
```
struct ext2_dir_entry
{
   _u32  inode                  /*  numer i-wezla                */
   _u16  rec_len                /*  dlugosc pozycji katalogowej  */
   _u16  name_len               /*  dlugosc nazwy                */
   char  name[EXT2_NAME_LEN]    /*  nazwa                        */
}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

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:
  - partycjonowanie statyczne, partycjonowanie dynamiczne
  - proste stronicowanie, prosta segmentacja
  - stronicowanie pamięci wirtualnej, segmentacja pamięci wirtualnej
  - stronicowanie i segmentacja pamięci wirtualnej

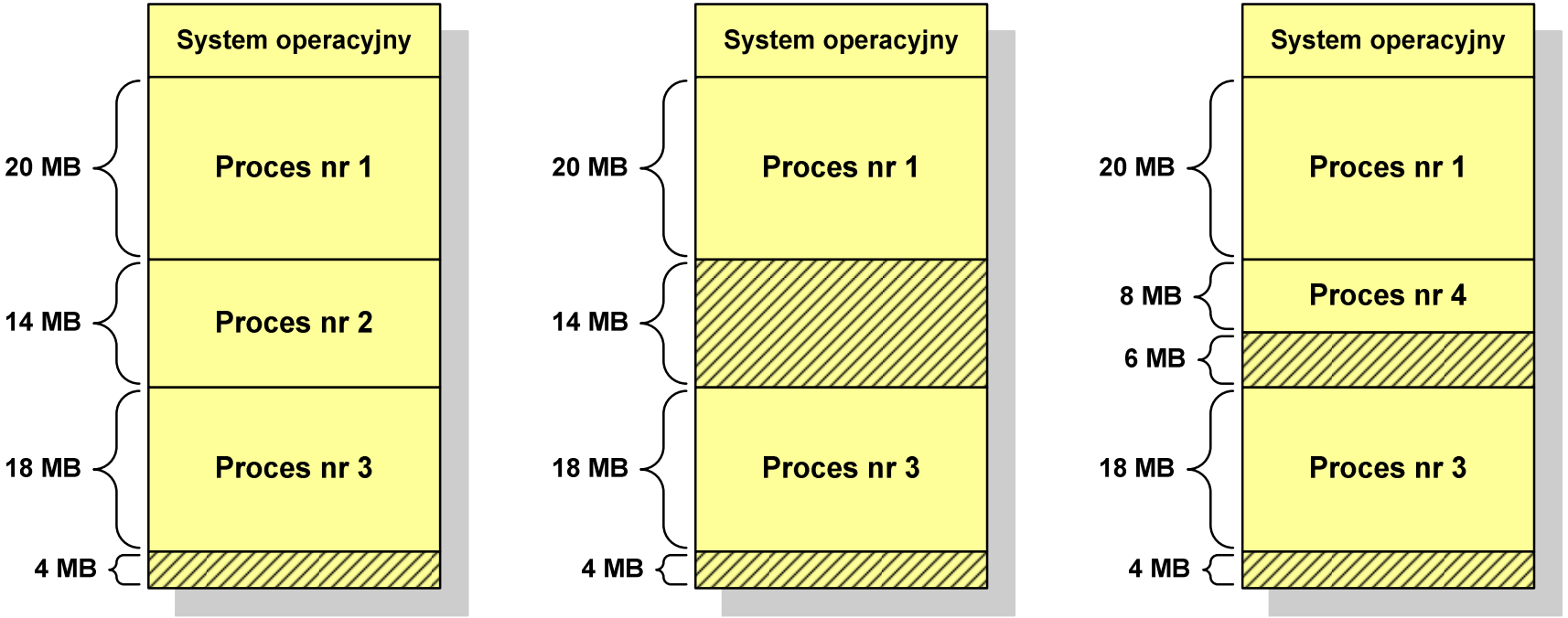
Partycjonowanie statyczne

- podział pamięci operacyjnej na obszary o takim samym lub różnym rozmiarze, ustalanym podczas generowania systemu



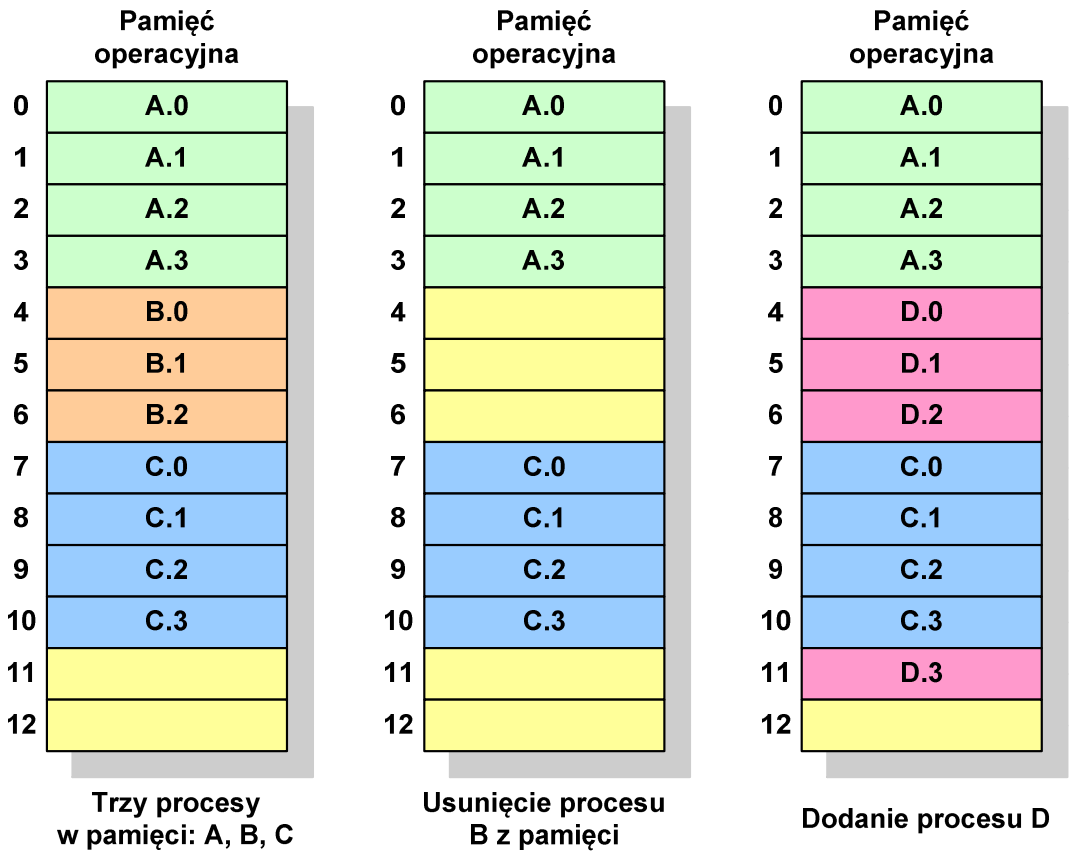
Partycjonowanie dynamiczne

- partycje są tworzone dynamicznie w ten sposób, że każdy proces jest ładowany do partycji o rozmiarze równym rozmiarowi procesu
- partycje mają różną długość, może zmieniać się także ich liczba
- przykład - w systemie działa 5 procesów: 20 MB, 14 MB, 18 MB, 8 MB, 8 MB



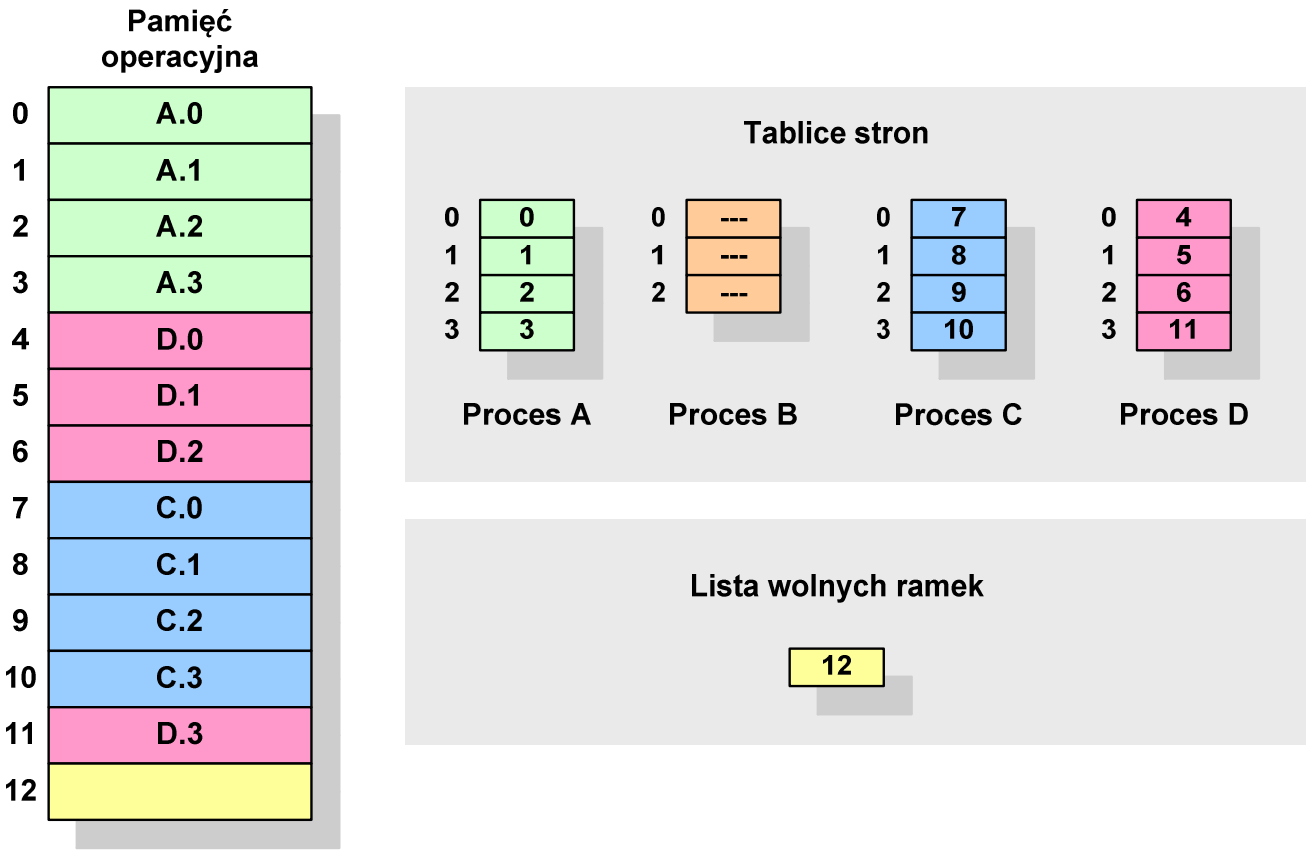
Proste stronicowanie

- 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



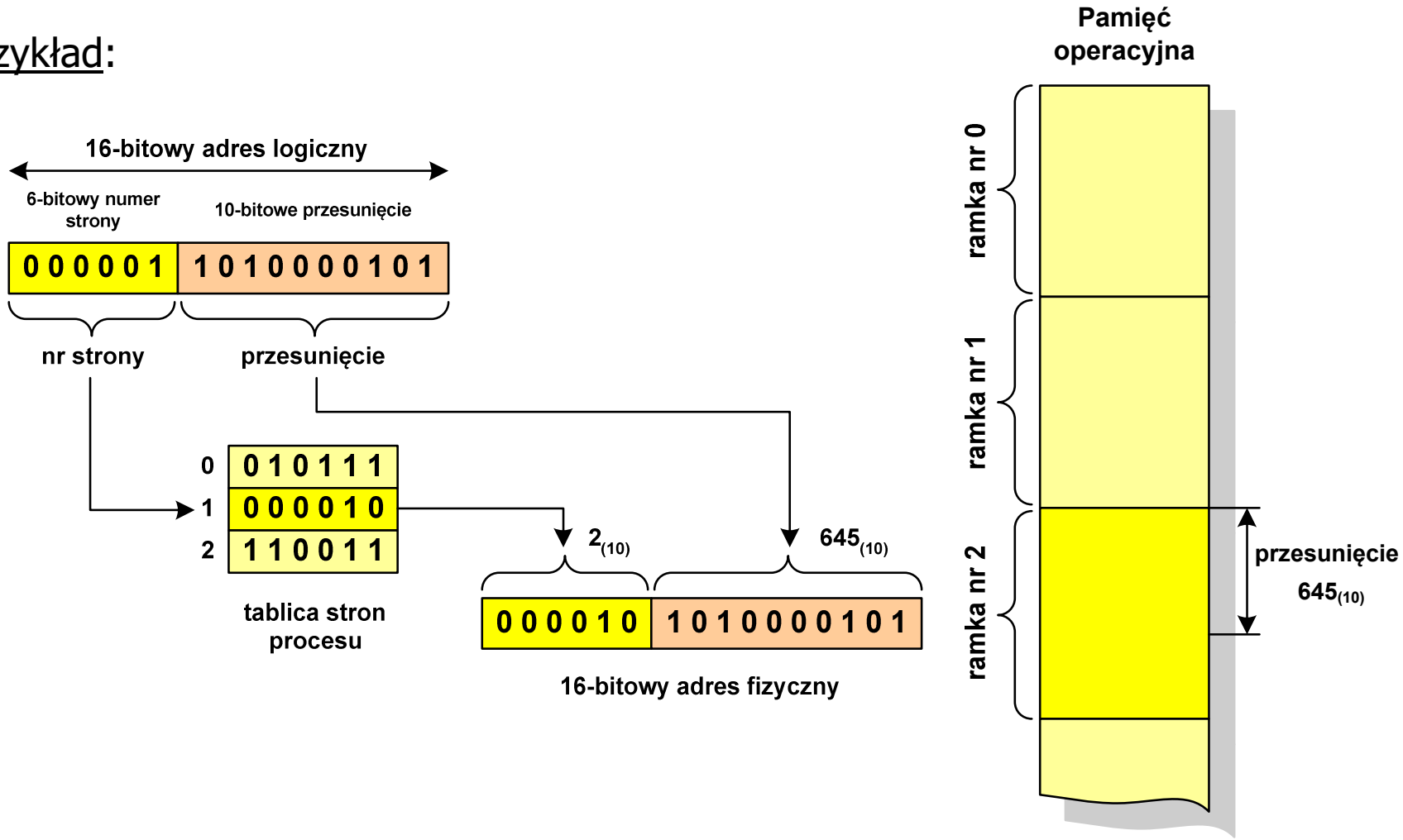
Proste stronicowanie

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



Proste stronicowanie

Przykład:

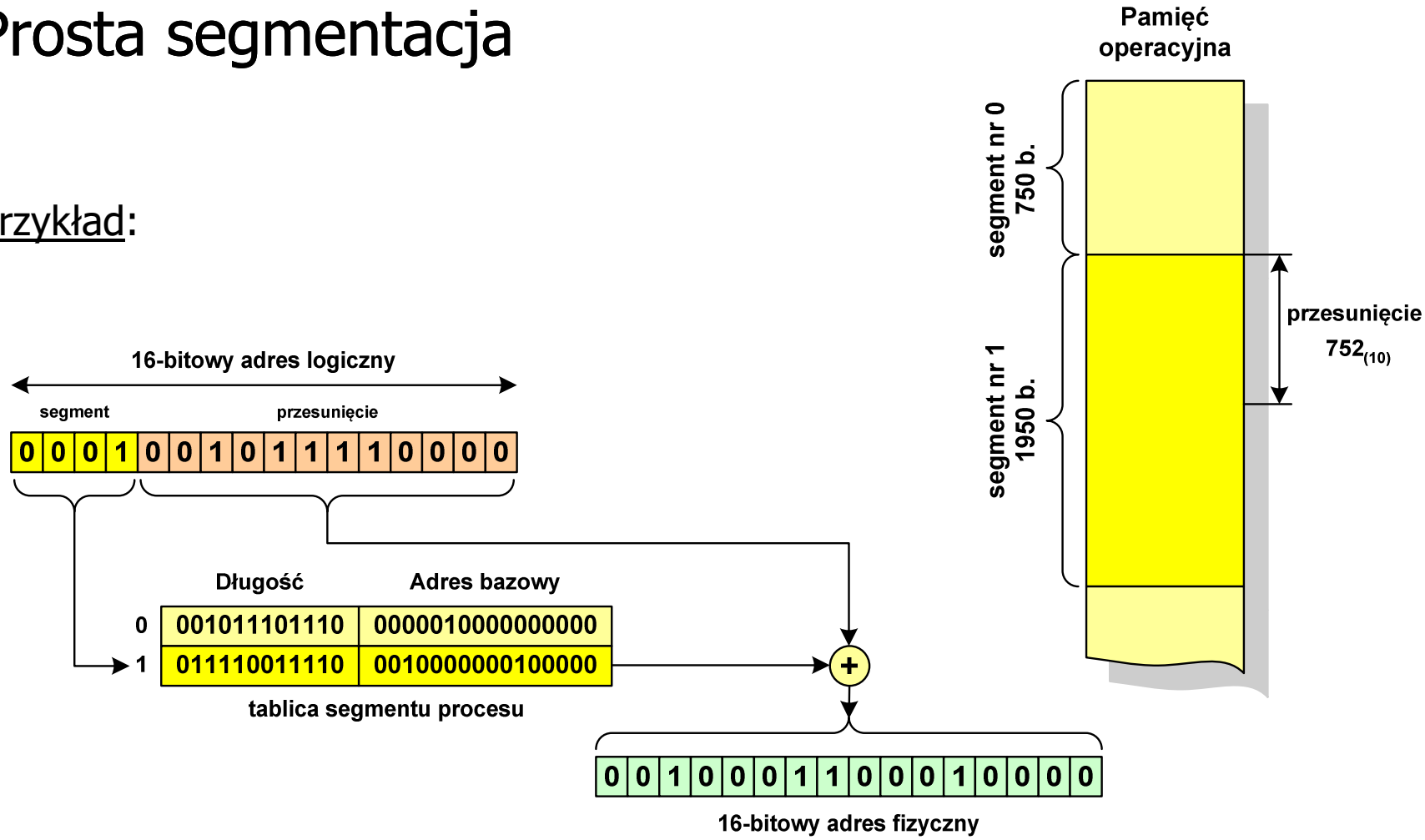


Prosta segmentacja

- 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?

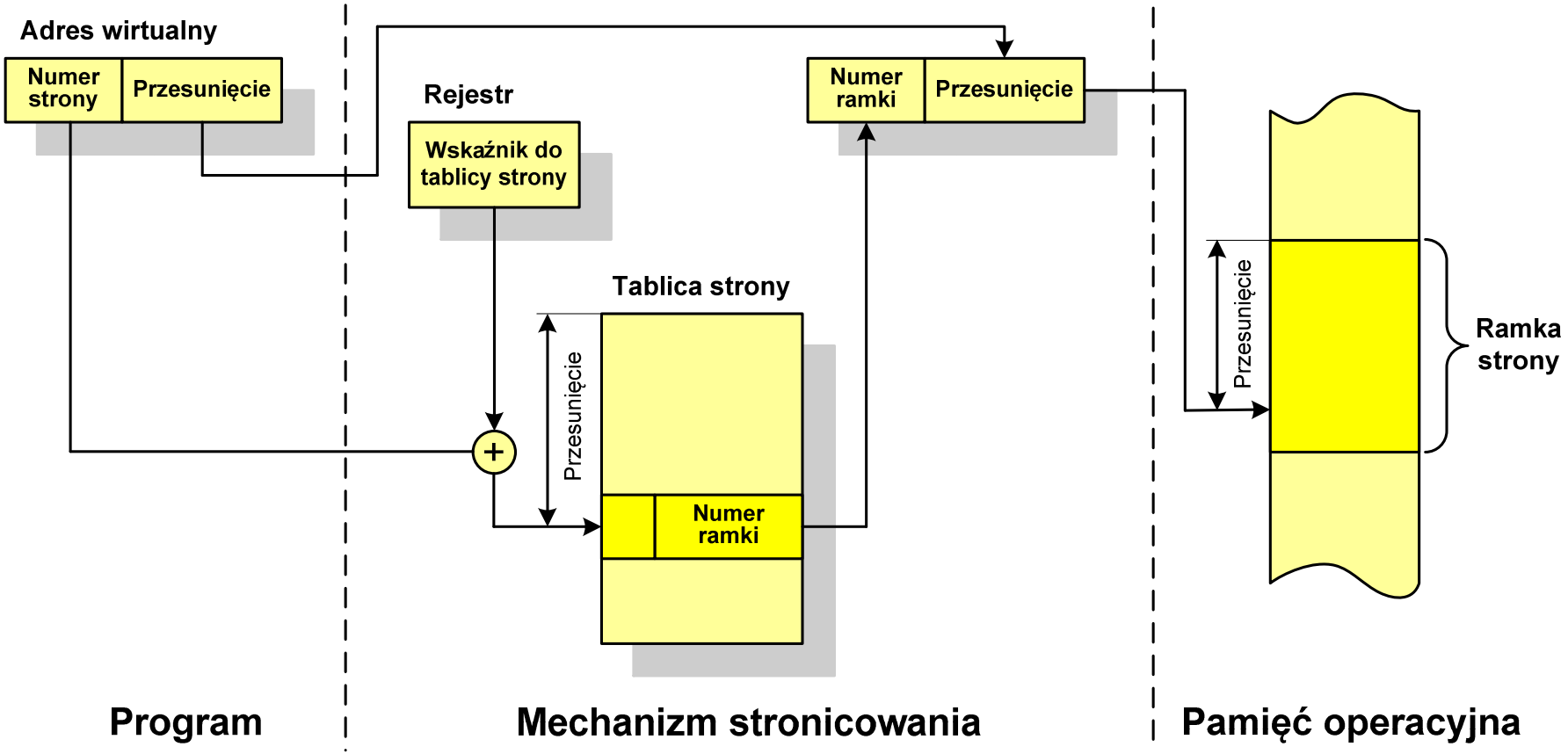
- generowanie przerwania sygnalizującego błąd w dostępie do pamięci
- zmiana stan procesu na zablokowany
- wstawienie do pamięci operacyjnej fragment procesu zawierający adres logiczny, który był przyczyną błędu
- zmiana stanu procesu na uruchomiony

Dzięki zastosowaniu pamięci wirtualnej:

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

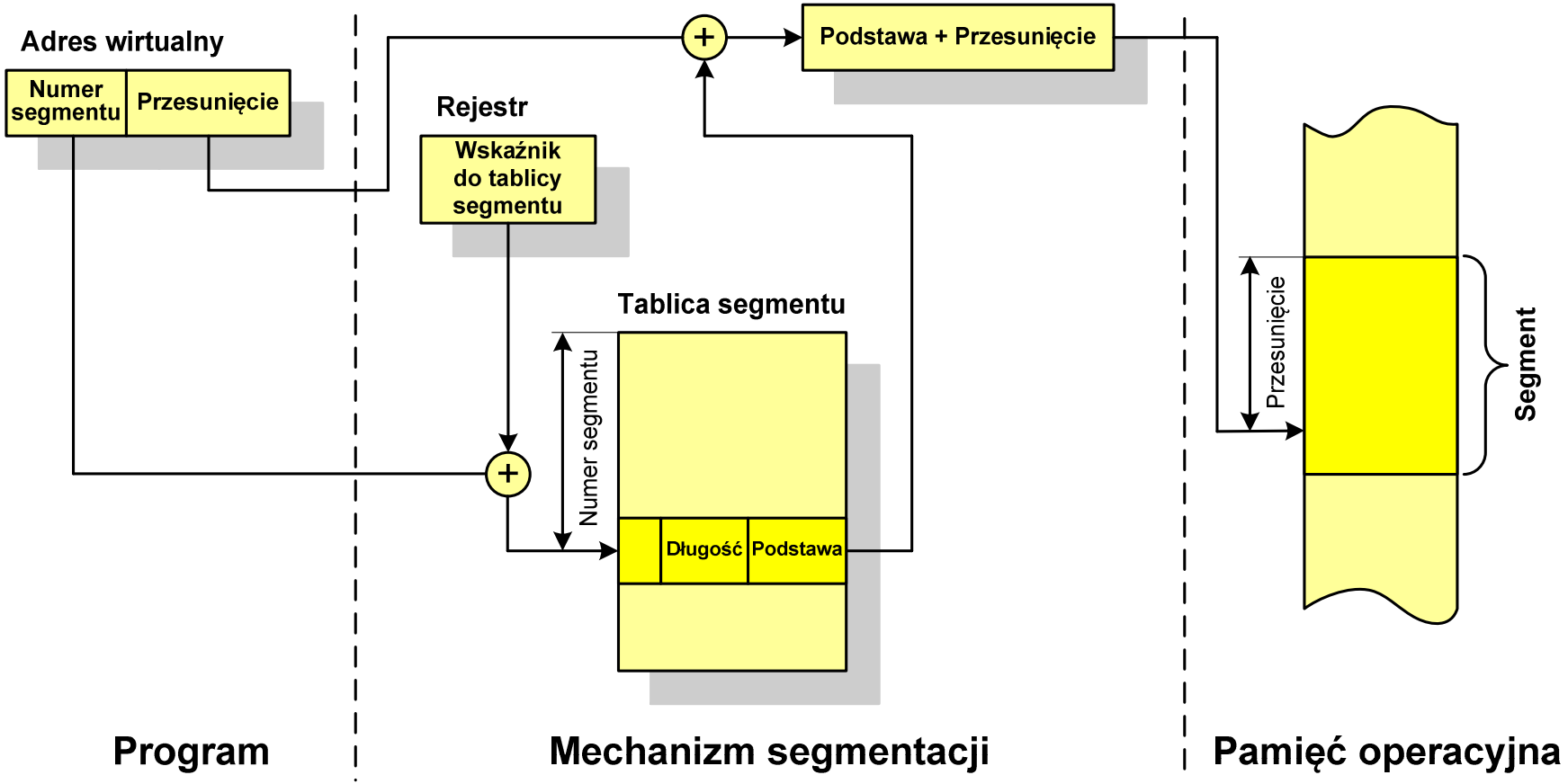
Stronicowanie pamięci wirtualnej

- odczytanie strony wymaga translacji adresu wirtualnego na fizyczny



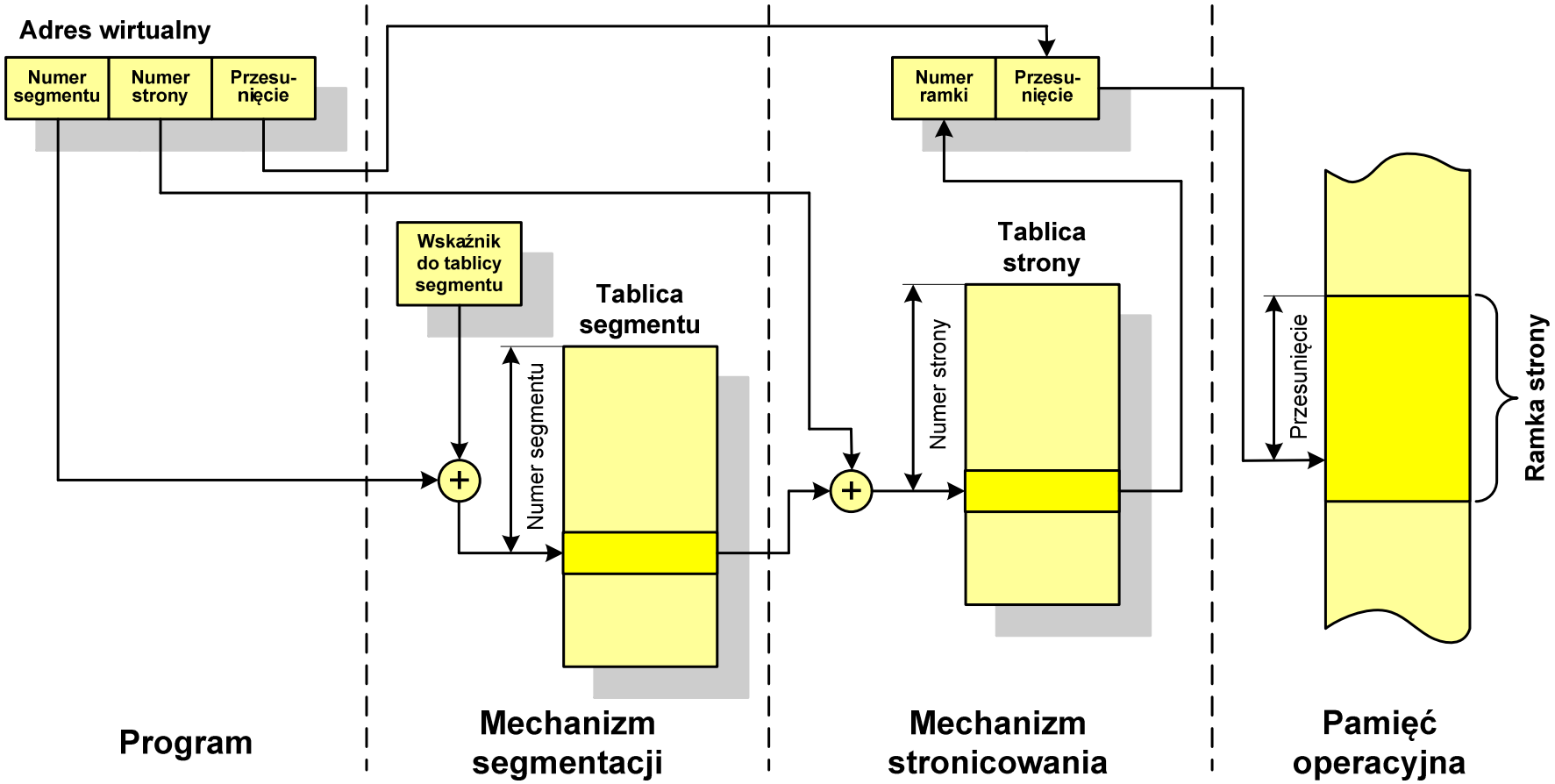
Segmentacja pamięci wirtualnej

- 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 8

Dziękuję za uwagę!

(następny wykład: 05.06.2020)