Informatyka 2 (ES1E3017)

Politechnika Białostocka - Wydział Elektryczny

Elektrotechnika, semestr III, studia stacjonarne I stopnia

Rok akademicki 2021/2022

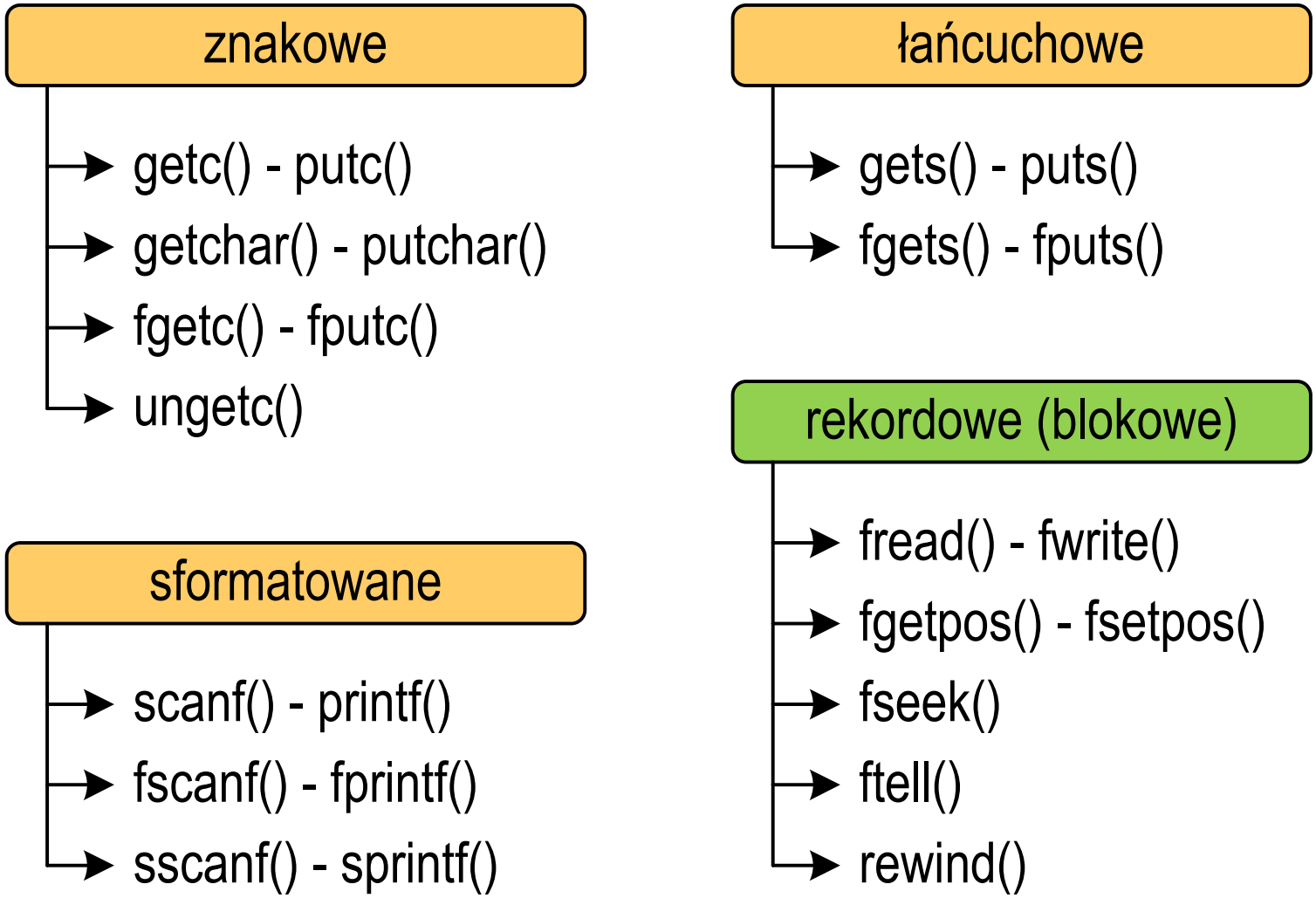
Wykład nr 6 (10.01.2022)

dr inż. Jarosław Forenc

Plan wykładu nr 6

- Operacje wejścia-wyjścia
  - rekordowe (blokowe)
- Definicje systemu operacyjnego
- Zarządzanie procesami
  - definicja procesu, blok kontrolny procesu
  - dwu- i pięciostanowy model procesu
- Zarządzanie dyskowymi operacjami we-wy
  - metody przydziału pamięci dyskowej (alokacja ciągła, alokacja listowa, alokacja indeksowa)
  - struktura dysku twardego (MBR, GPT)
  - systemy plików: FAT (FAT12)

Rekordowe (blokowe) operacje wejścia-wyjścia



Rekordowe (blokowe) operacje wejścia-wyjścia

FWRITE stdio.h

```
size_t fwrite(const void *p, size_t s, size_t n,
              FILE *fp);
```

- Zapisuje n elementów o rozmiarze s bajtów każdy, do pliku wskazywanego przez fp, biorąc dane z obszaru pamięci wskazywanego przez p
- Zwraca liczbę zapisanych elementów - jeśli jest ona różna od n, to wystąpił błąd zapisu (brak miejsca na dysku lub dysk zabezpieczony przed zapisem)

Przykład: zapisanie danych do pliku binarnego

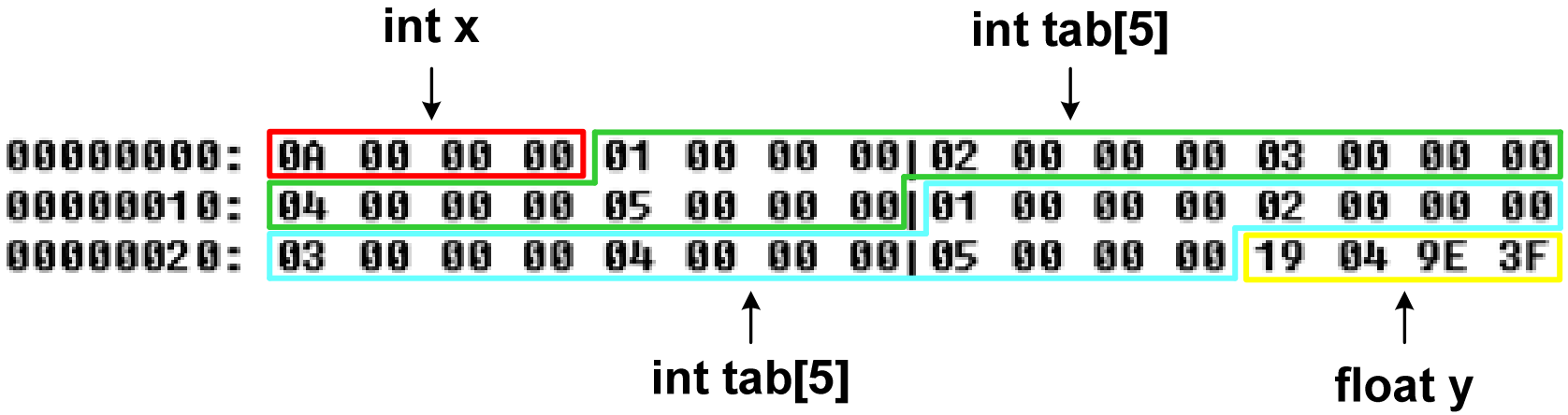
```
#include <stdio.h>

int main(void)
{
    FILE *fp;
    int   x = 10, tab[5] = {1,2,3,4,5};
    float y = 1.2345f;

    fp = fopen("dane.dat","wb");
    fwrite(&x,sizeof(int),1,fp);
    fwrite(tab,sizeof(int),5,fp);
    fwrite(tab,sizeof(tab),1,fp);
    fwrite(&y,sizeof(float),1,fp);
    fclose(fp);

    return 0;
}
```

Przykład: zapisanie danych do pliku binarnego

- Czterokrotne wywołanie funkcji fwrite()

```
fwrite(&x,sizeof(int),1,fp);          // int x = 10;
fwrite(tab,sizeof(int),5,fp);         // int tab[5] = {1,2,3,4,5};
fwrite(tab,sizeof(tab),1,fp);         // int tab[5] = {1,2,3,4,5};
fwrite(&y,sizeof(float),1,fp);        // float y = 1.2345;
```

spowoduje zapisanie do pliku 48 bajtów:



Rekordowe (blokowe) operacje wejścia-wyjścia

FREAD stdio.h

```
size_t fread(void *p, size_t s, size_t n,
             FILE *fp);
```

- Pobiera n elementów o rozmiarze s bajtów każdy, z pliku wskazywanego przez fp i umieszcza odczytane dane w obszarze pamięci wskazywanym przez p
- Zwraca liczbę odczytanych elementów - w przypadku gdy liczba ta jest różna od n, to wystąpił błąd końca strumienia (w pliku było mniej elementów niż podana wartość argumentu n)

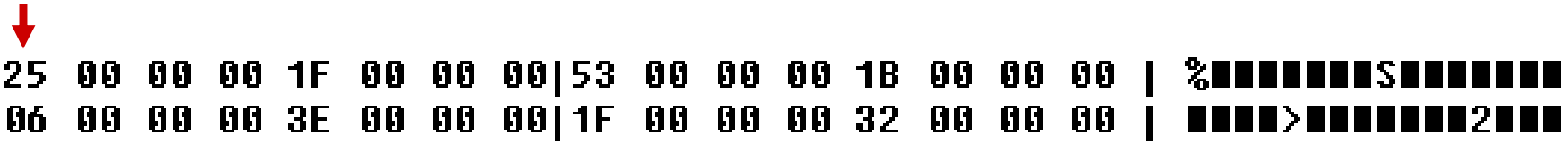
Przykład: odczytanie liczb z pliku binarnego

```c
#include <stdio.h>

int main(void)
{
    FILE *fp;
    int x, ile = 0;

    fp = fopen("liczby.dat","rb");
    fread(&x,sizeof(int),1,fp);
    while (!feof(fp))
    {
        ile++; printf("%d\n",x);
        fread(&x,sizeof(int),1,fp);
    }
    fclose(fp);
    printf("Odczytano: %d liczb\n",ile);
    return 0;
}
```

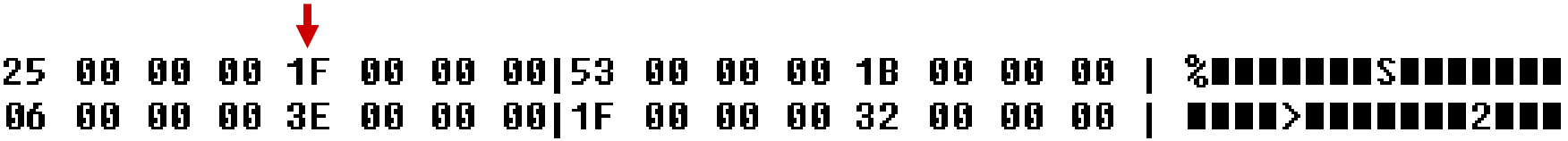
```
37
31
83
27
6
62
31
50
Odczytano: 8 liczb
```

Przykład: odczytanie liczb z pliku binarnego

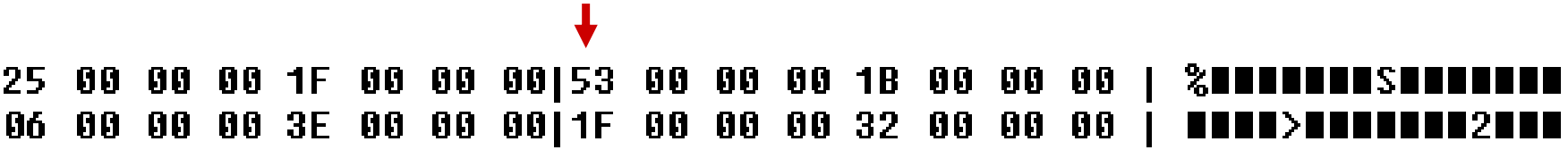
- Po otwarciu pliku wskaźnik pozycji pliku pokazuje na jego początek



- Po odczytaniu jednej liczby: fread(&x,sizeof(int),1,plik);
  wskaźnik jest automatycznie przesuwany o sizeof(int) bajtów



- Po odczytaniu kolejnej liczby: fread(&x,sizeof(int),1,plik);
  wskaźnik jest ponownie przesuwany o sizeof(int) bajtów



- Plik binarny zawiera liczby: 37 31 83 27 6 62 31 50

Rekordowe (blokowe) operacje wejścia-wyjścia

REWIND stdio.h

```
void rewind(FILE *fp);
```

- Ustawia wskaźnik pozycji w pliku wskazywanym przez fp na początek pliku

FTELL stdio.h

```
long int ftell(FILE *fp);
```

- Zwraca bieżące położeniu w pliku wskazywanym przez fp (liczbę bajtów od początku pliku)

Przykład: ile razy występuje w pliku wartość max

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

```
#include <stdio.h>

int main(void)
{
    FILE  *fp;
    int   x, max, ile = 0;

    fp = fopen("dane.dat","rb");

    fread(&x,sizeof(int),1,fp);
    max = x;
    while(!feof(fp))
    {
        if (x > max) max = x;
        fread(&x,sizeof(int),1,fp);
    }
    printf("Wartosc max: %d\n",max);
```

Przykład: ile razy występuje w pliku wartość max

```
    rewind(fp);

    fread(&x,sizeof(int),1,fp);
    while(!feof(fp))
    {
        if (x == max) ile++;
        fread(&x,sizeof(int),1,fp);
    }
    printf("Wystapienia max: %d\n",ile);

    fclose(fp);

    return 0;
}
```

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

```
Wartosc max: 9
Wystapienia max: 4
```

Rekordowe (blokowe) operacje wejścia-wyjścia

FSEEK stdio.h

```
int fseek(FILE *fp, long int offset, int mode);
```

- Pozwala przejść bezpośrednio do dowolnego bajtu w pliku wskazywanym przez fp
- offset określa wielkość przejścia w bajtach, zaś mode - punkt początkowy, względem którego określane jest przejście (SEEK_SET - początek pliku, SEEK_CUR - bieżąca pozycja, SEEK_END - koniec pliku)
- Gdy wywołanie jest poprawne, to funkcja zwraca wartość 0 gdy wystąpił błąd (np. próba przekroczenia granic pliku), to funkcja zwraca wartość -1

Przykład: odczytanie liczby o podanym numerze

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

```c
#include <stdio.h>

int main(void)
{
    FILE  *fp;
    int   x, nr;

    fp = fopen("dane.dat","rb");
    printf("Nr: "); scanf("%d",&nr);
    while (fseek(fp,(nr-1)*sizeof(int),SEEK_SET)==0)
    {
        fread(&x,sizeof(int),1,fp);
        printf("Liczba: %d\n",x);
        printf("Nr: "); scanf("%d",&nr);
    }
    printf("Koniec!\n");
    fclose(fp);
    return 0;
}
```

```
Nr: 6
Liczba: 9
Nr: 14
Liczba: 5
Nr: 29
Liczba: 1
Nr: -1
Koniec!
```

Rekordowe (blokowe) operacje wejścia-wyjścia

FGETPOS stdio.h

```
int fgetpos(FILE *fp, fpos_t *pos);
```

- Zapamiętuję pod zminną pos bieżące położenie w pliku wskazywanym przez fp; zwraca 0, gdy wywołania jest poprawne i wartość niezerową, gdy wystąpił błąd

FSETPOS stdio.h

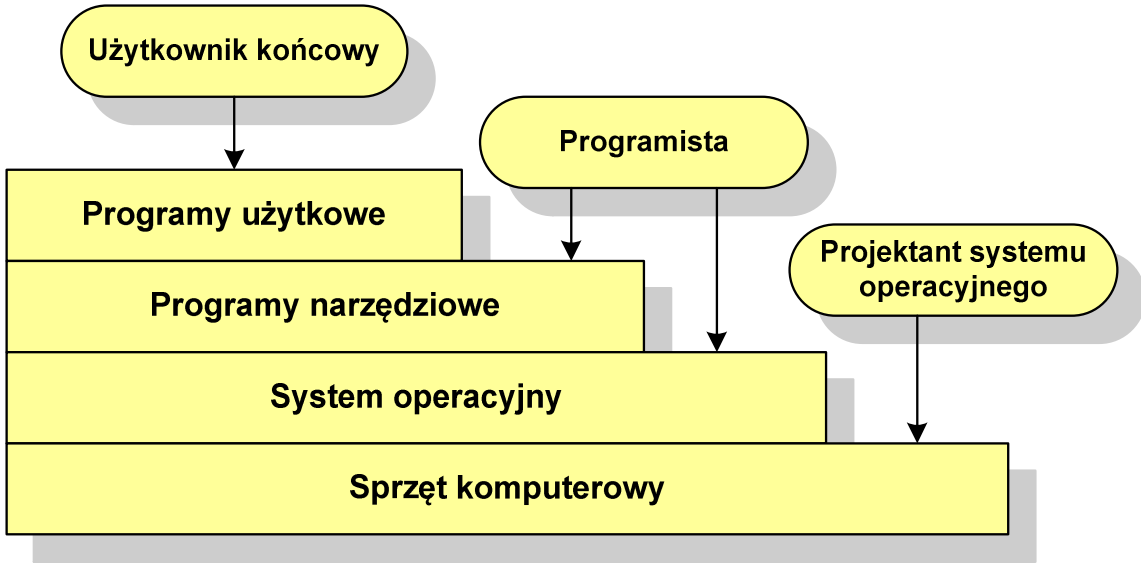
```
int fsetpos(FILE *fp, const fpos_t *pos);
```

- Przechodzi do położenia pos w pliku wskazywanym przez fp; zwraca 0, gdy wywołania jest poprawne i wartość niezerową, gdy wystąpił błąd

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

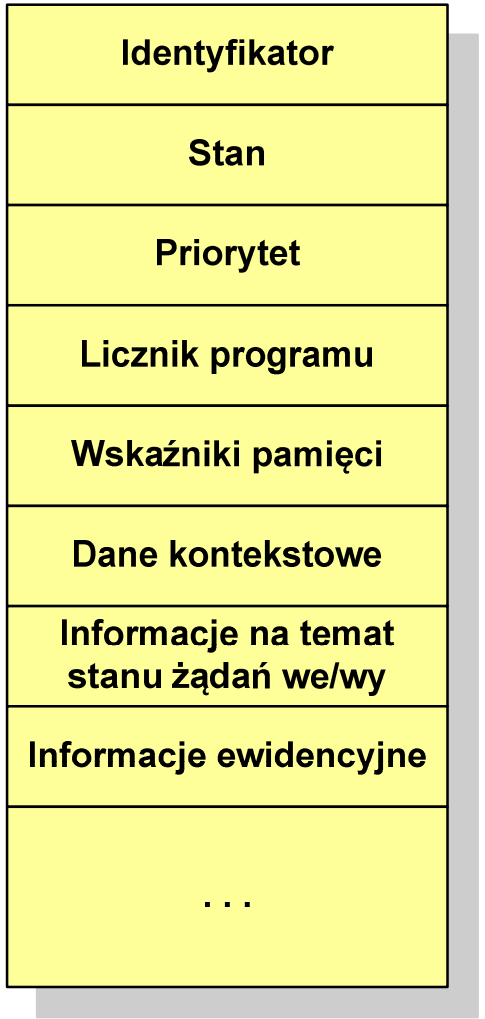
- System operacyjny - administrator zasobów - zarządza i przydziela zasoby systemu komputerowego oraz steruje wykonaniem programu
- zasób systemu - każdy element systemu, który może być przydzielony innej części systemu lub oprogramowaniu aplikacyjnemu
- do zasobów systemu zalicza się:
  - czas procesora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

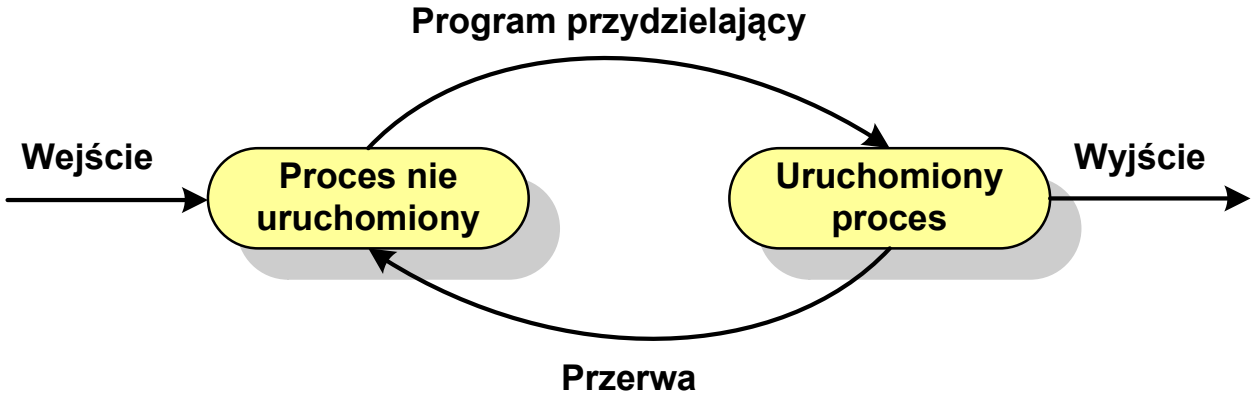
Blok kontrolny procesu (deskryptor procesu)

- 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

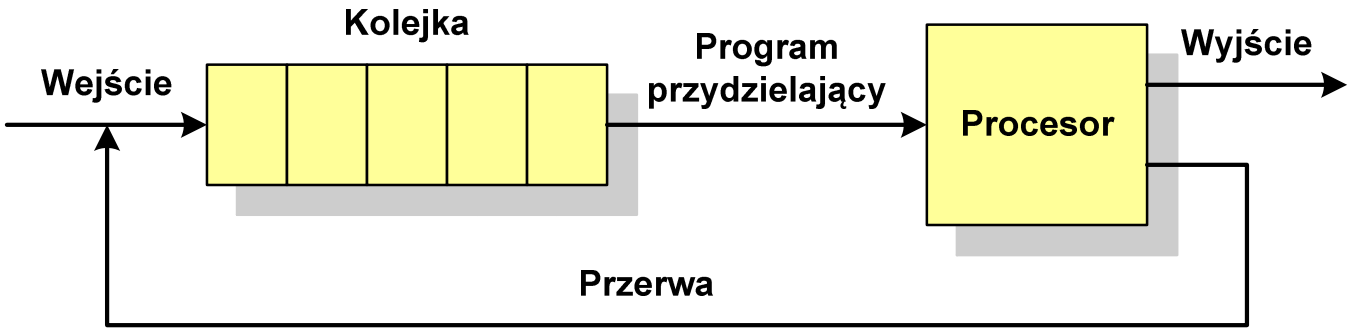
- najprostszy model polega na tym, że w dowolnej chwili proces jest wykonywany przez procesor (uruchomiony) lub nie (nie uruchomiony)



- system operacyjny tworząc nowy proces, tworzy blok kontrolny procesu po czym wprowadza proces do systemu jako nie uruchomiony
- w pewnym momencie aktualnie wykonywany proces zostaje przerwany i program przydzielający wybiera inny proces do wykonania
- stan poprzednio uruchomionego procesu jest zmieniany z uruchomionego na nie uruchomiony

Dwustanowy model procesu

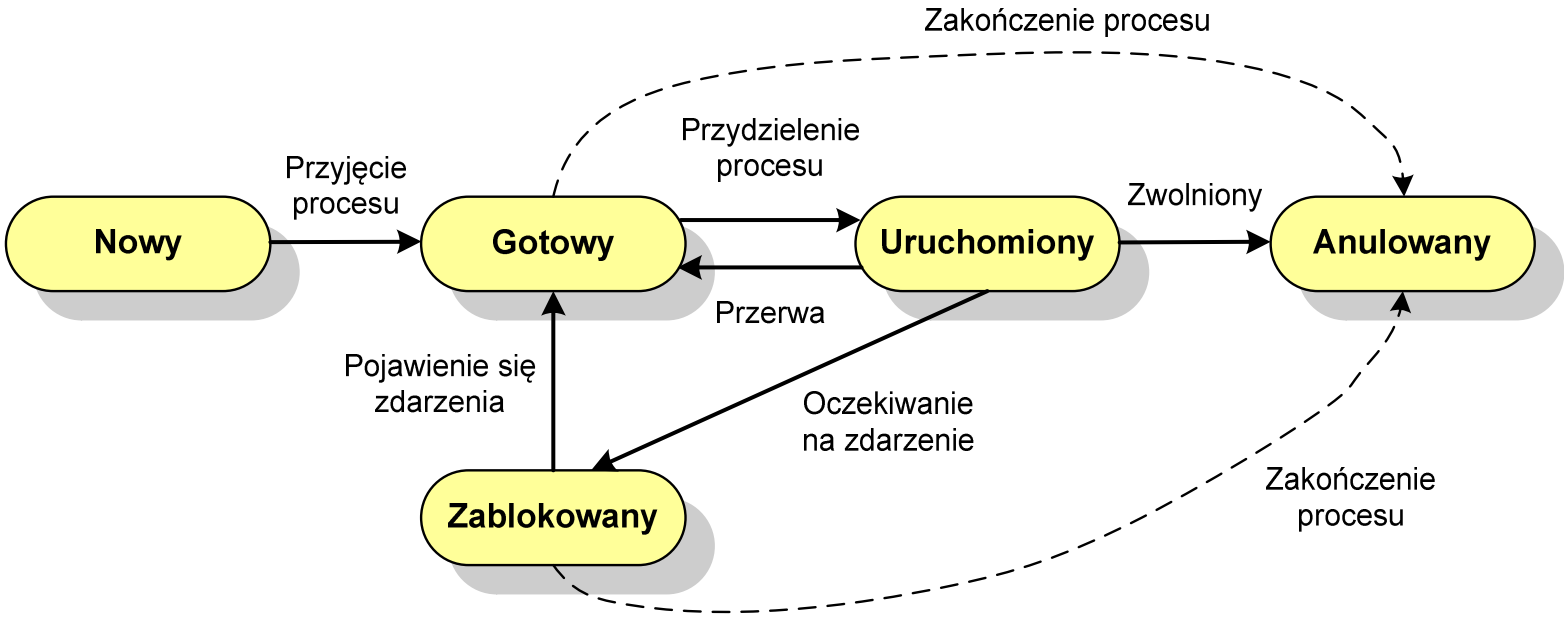
- procesy, które nie są uruchomione czekają w kolejce na wykonanie



- jeśli wykonywanie procesu zostało anulowane lub zakończone, to opuszcza on system, a program przydzielający wybiera kolejny proces z kolejki, który zostanie wykonany
- w dwustanowym modelu procesu kolejka działa na zasadzie FIFO, a procesor wykonuje procesy cyklicznie z kolejki
- problem pojawia się w przypadku, gdy kolejny proces pobierany do wykonania z kolejki jest zablokowany, gdyż oczekuje na zakończenie operacji we-wy

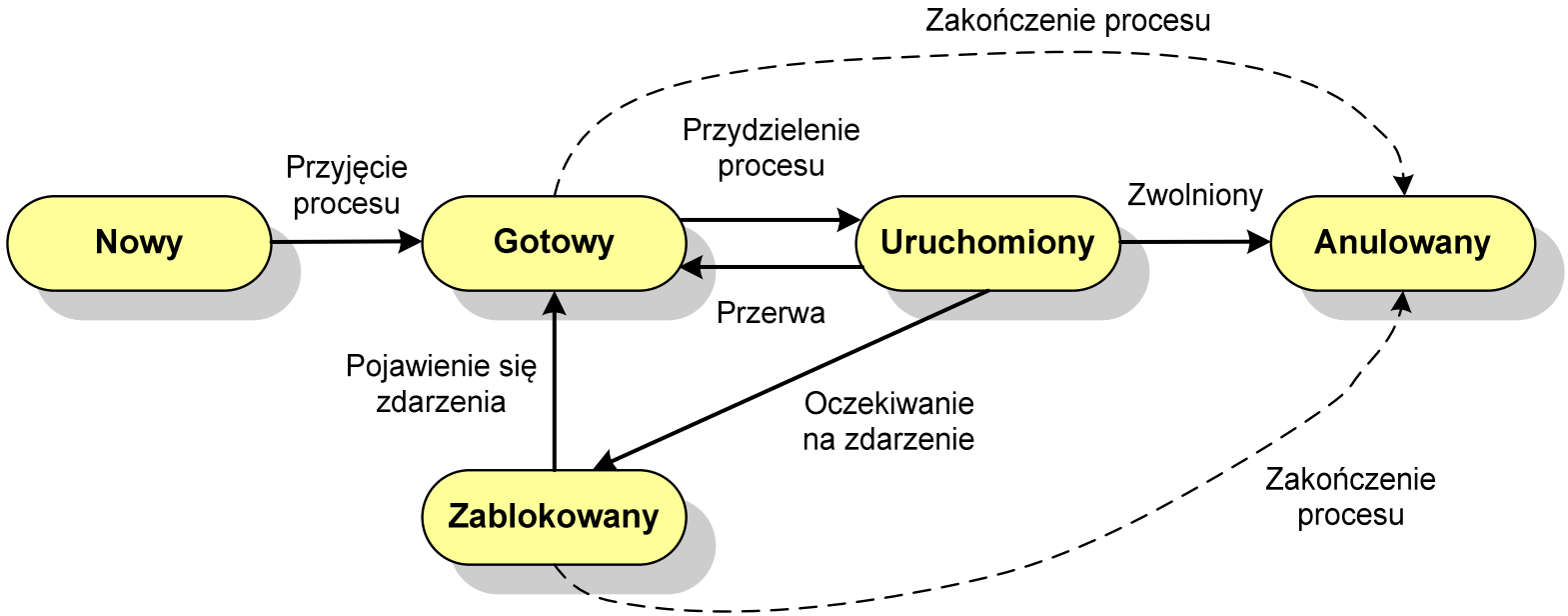
Pięciostanowy model procesu

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



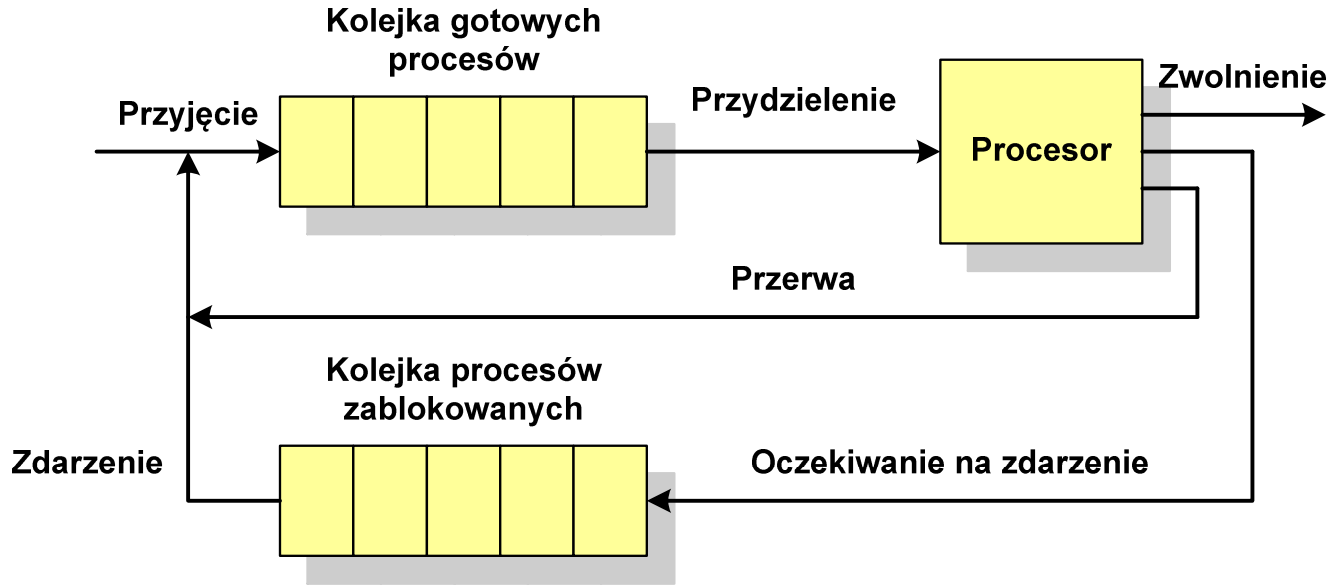
Pięciostanowy model procesu

- 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



Pięciostanowy model procesu

- podział procesów nieuruchomionych na gotowe do wykonania i zablokowane wymaga zastosowania minimum dwóch kolejek



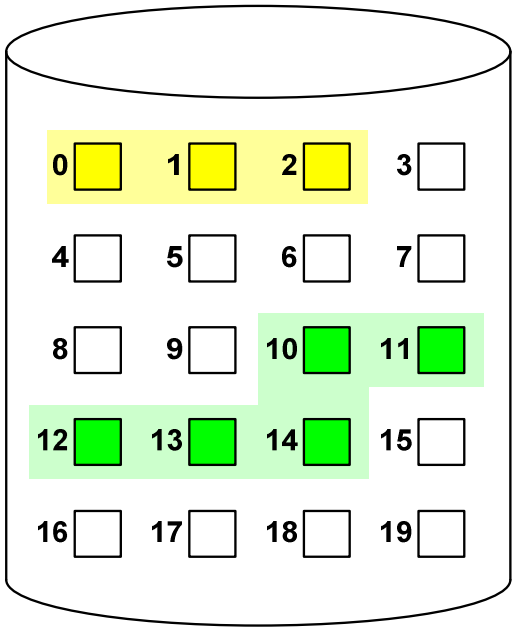
- gdy pojawia się zdarzenie system operacyjny musi przejrzeć kolejkę szukając procesów, który związane są z danym zdarzeniem
- w celu zapewnienia większej wydajności lepiej jest gdy dla każdego zdarzenia istnieje oddzielna kolejka

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

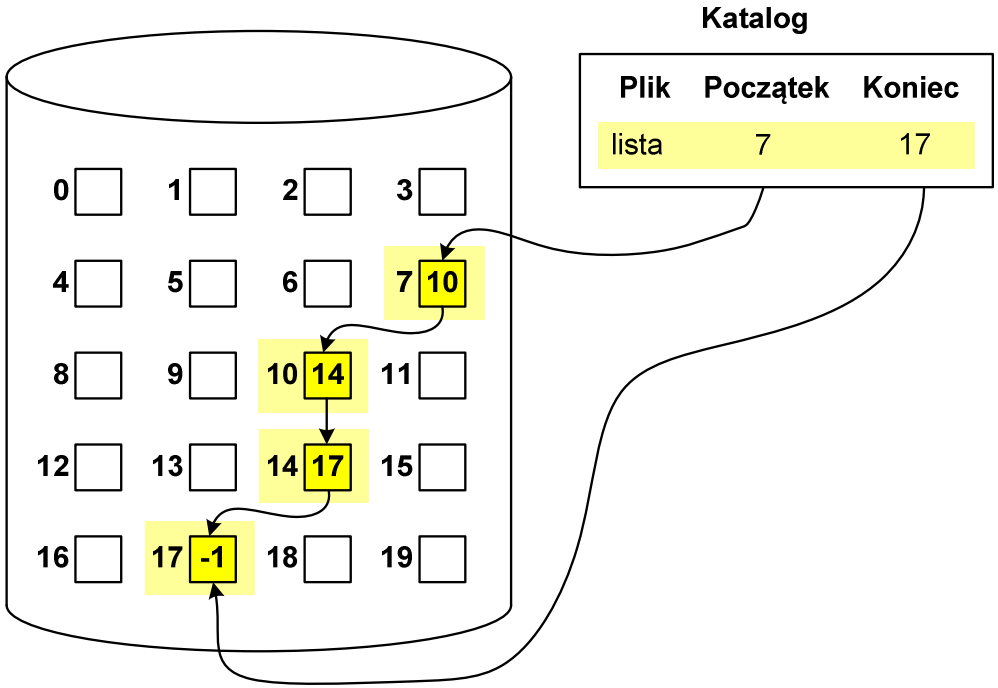


Katalog

Plik	Początek	Długość
lista	0	3
poczta	10	5

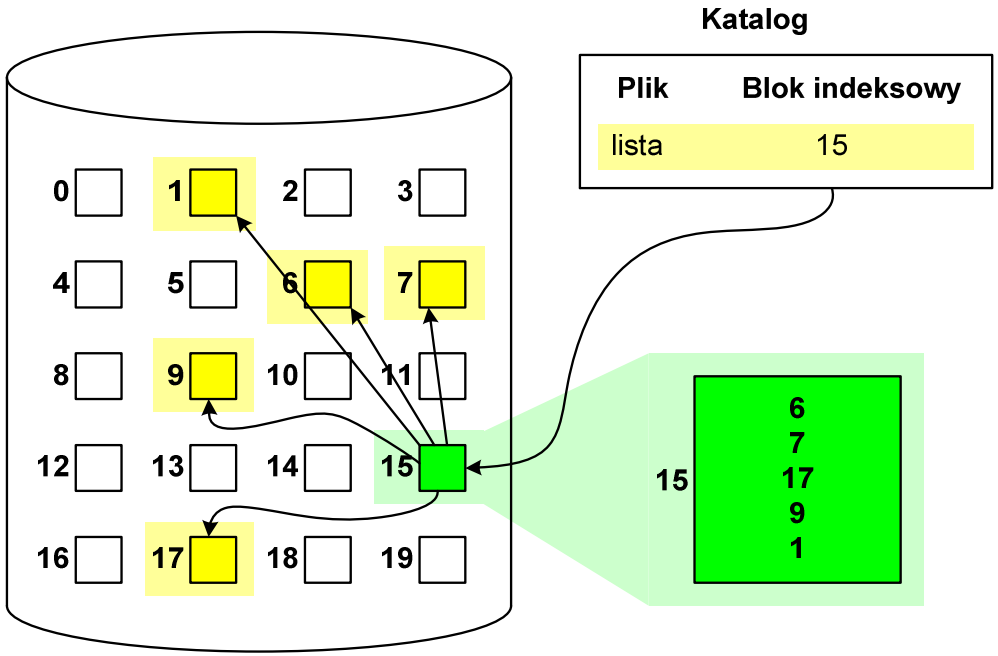
Przydział pamięci dyskowej - alokacja listowa

- 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



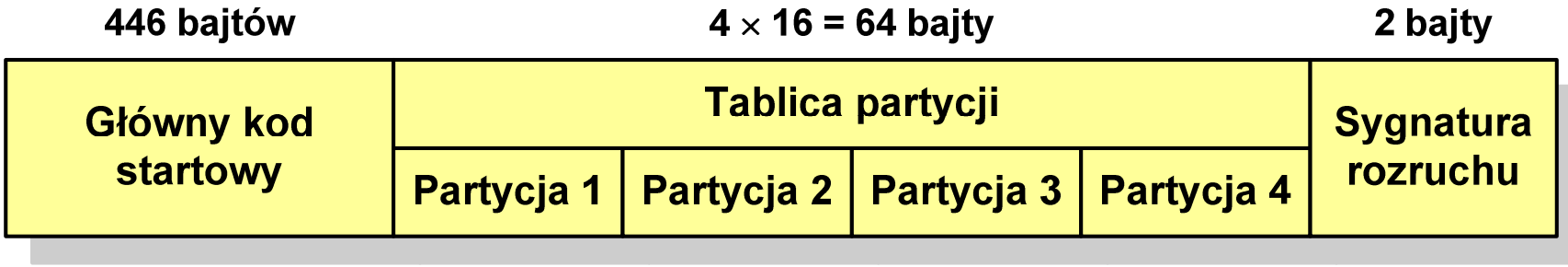
Przydział pamięci dyskowej - alokacja indeksowa

- 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
- sygnatura rozruchu (boot signature) - znacznik końca MBR (0x55AA)

Struktura dysku twardego - MBR (tablica partycji)

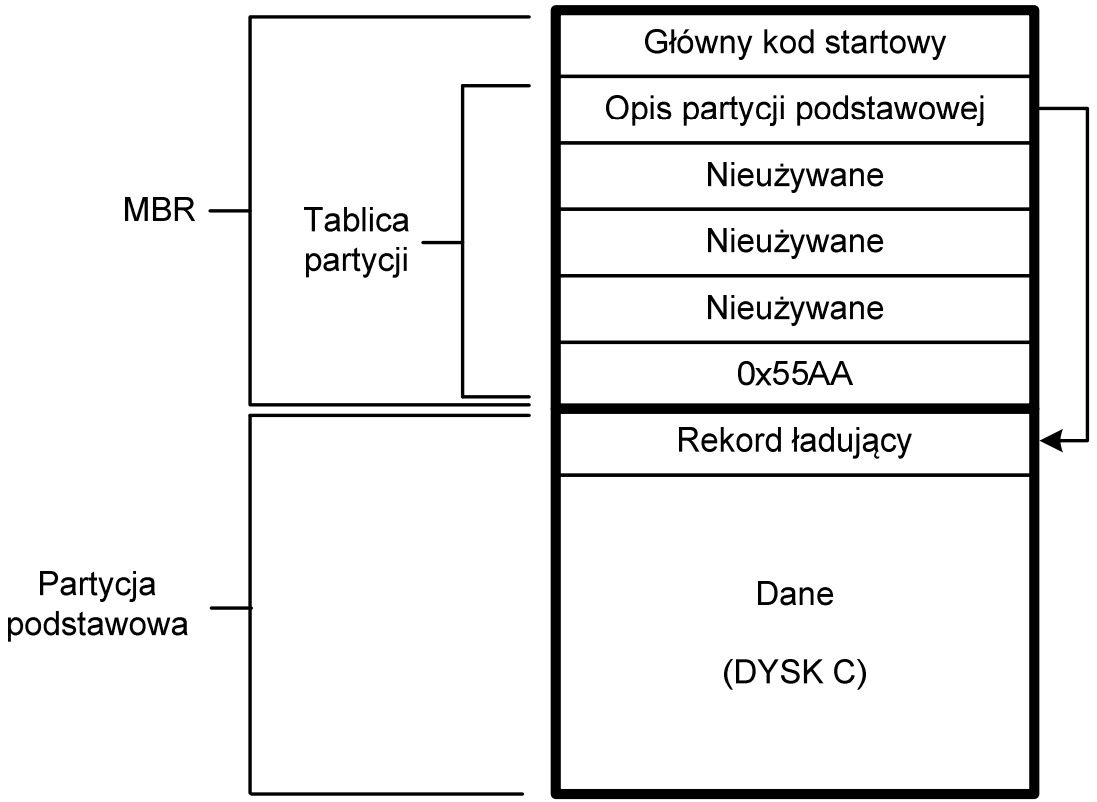
- zawartość rekordu w tablicy partycji

Bajty	Rozmiar	Zawartość
00H	1	Znacznik aktywności: 00H - nieaktywna, 80H - aktywna
01H	1	Początek partycji: numer głowicy
02H-03H	2	Początek partycji: numer cylindra i sektora
04H	1	Typ partycji (system plików)
05H	1	Koniec partycji: numer głowicy
06H-07H	2	Koniec partycji: numer cylindra i sektora
08H-0BH	4	Liczba sektorów: początek dysku → pierwszy sektor partycji
0CH-0FH	4	Rozmiar partycji: liczba sektorów w partycji

- zawartość i organizacja tablicy jest niezależna od systemu operacyjnego
- niewykorzystywany rekord zawiera same zera
- maksymalny rozmiar partycji to 2 TB ($2^{32} \times 512$ bajtów)

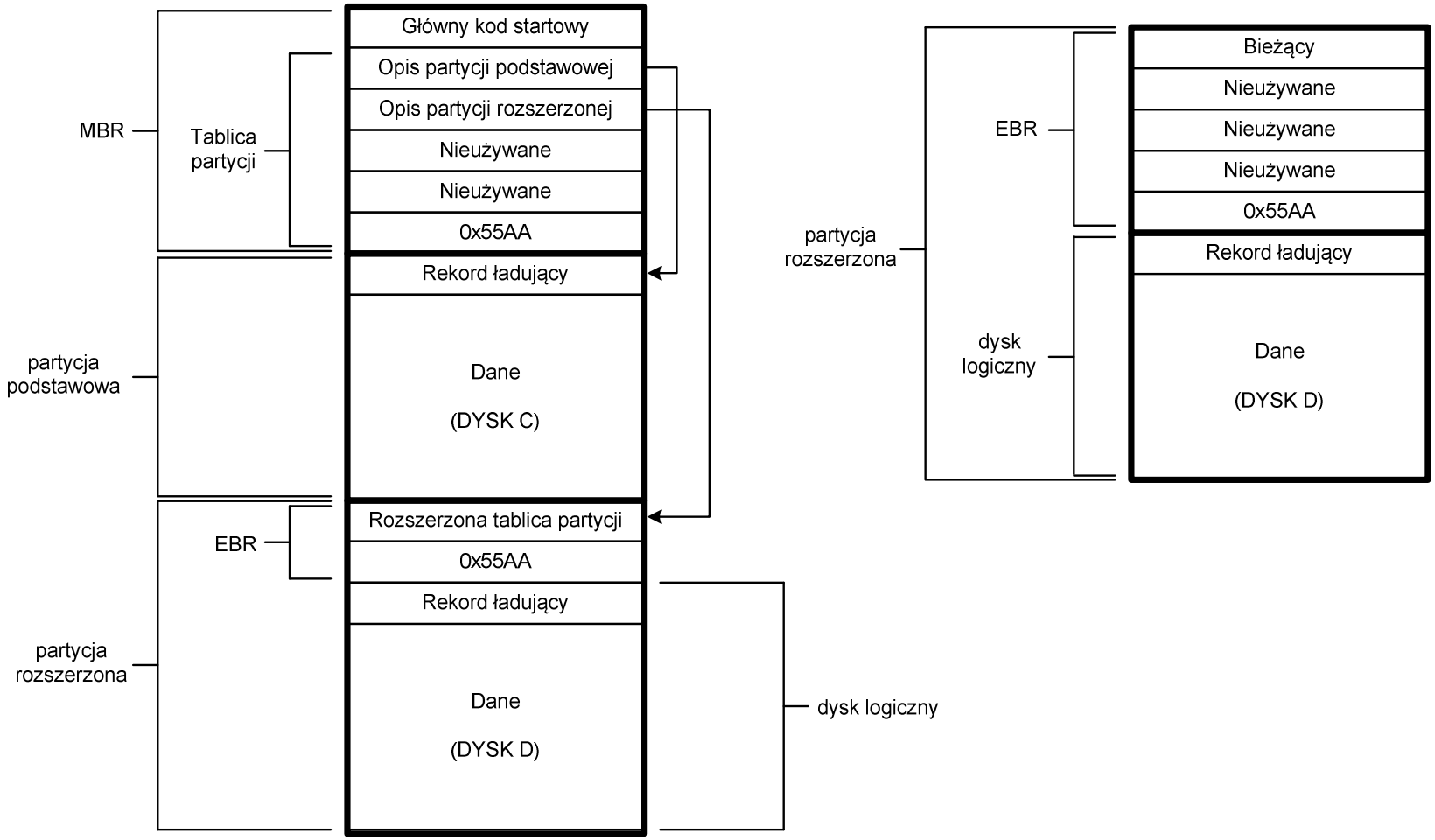
Struktura dysku twardego - MBR (tablica partycji)

- na dysku mogą znajdować się maksymalnie 4 partycje podstawowe (primary partition)
- każda partycja podstawowa może zawierać jeden dysk logiczny



Struktura dysku twardego - MBR (tablica partycji)

- w tablicy partycji można utworzyć jedną partycję rozszerzoną (extended partition), która może zawierać wiele dysków logicznych

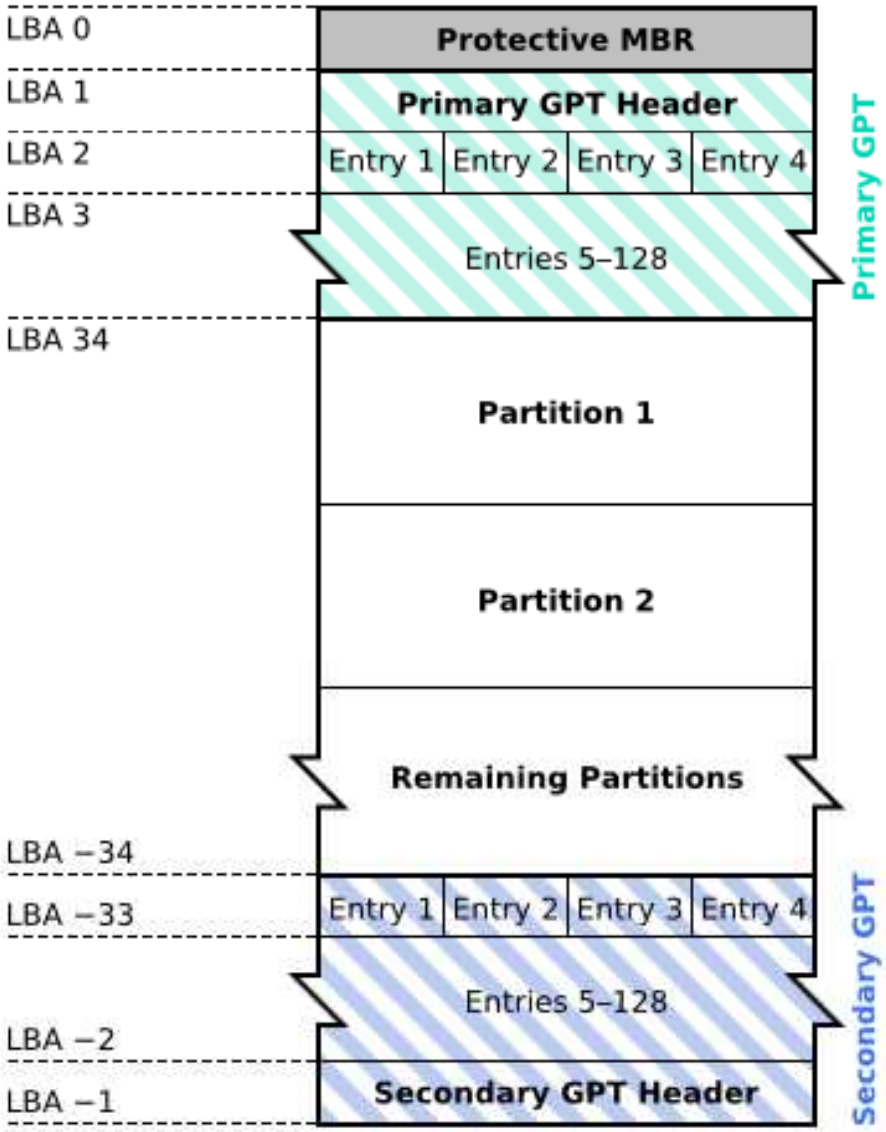


Struktura dysku twardego - GPT

- 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

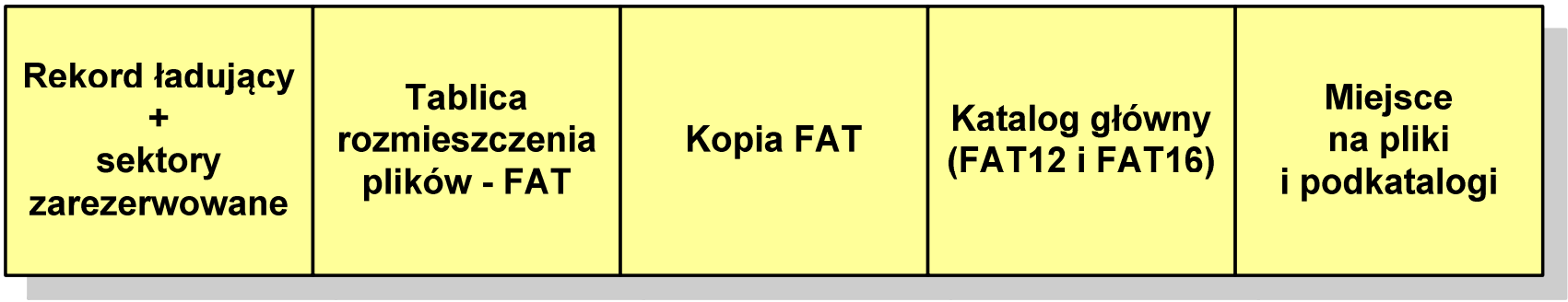
Struktura dysku twardego - GPT (struktura)

- 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



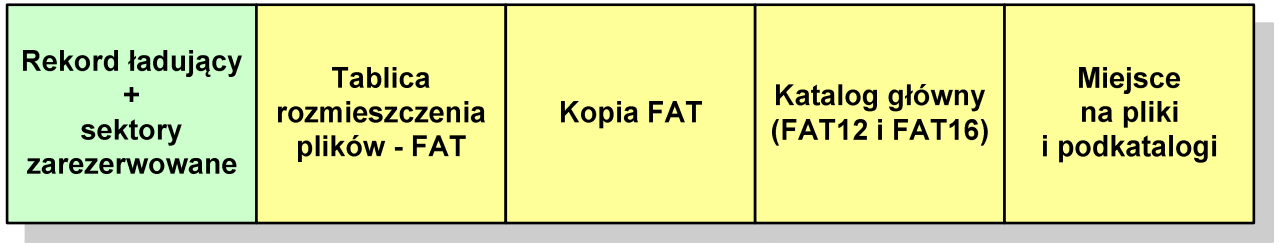
System plików FAT (File Allocation Table)

- 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

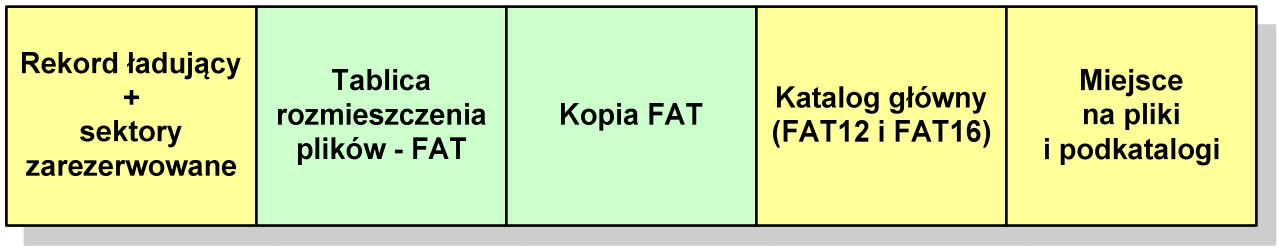
- system plików FAT12 przeznaczony jest dla nośników o małej pojemności
- obsługuje $2^{12} = 4096$ jednostek alokacji, max. rozmiar partycji to 16 MB
- rekord ładujący zajmuje pierwszy sektor dyskietki lub dysku logicznego



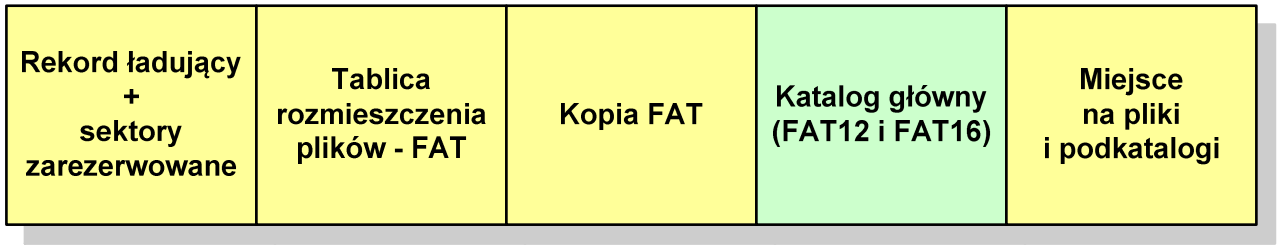
- rekord ładujący zawiera następujące dane:
  - instrukcja skoku do początku programu ładującego (3 bajty)
  - nazwa wersji systemu operacyjnego (8 bajtów)
  - struktura BPB (ang. BIOS Parametr Block) - blok parametrów BIOS (25 bajtów)
  - rozszerzony BPB (ang. Extended BPB, 26 bajtów)
  - wykonywalny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

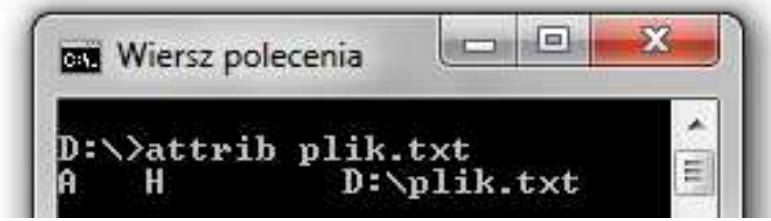
- katalog główny zawiera 32-bajtowe pola mogące opisywać pliki, podkatalogi lub etykietę dysku

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:

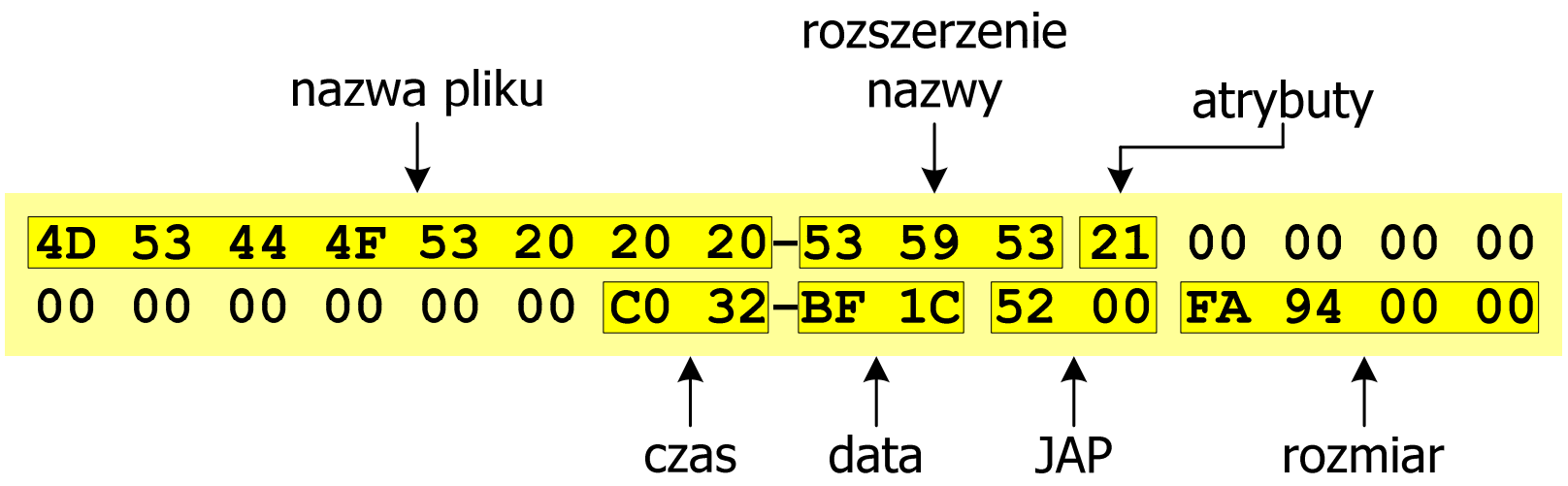
Bit	Znaczenie
0	Plik tylko do odczytu (read only)
1	Plik ukryty (hidden)
2	Plik systemowy (system)
3	Etykieta dysku (volume label)
4	Podkatalog
5	Plik archiwalny (archive)
6,7	Nie wykorzystywane



FAT12

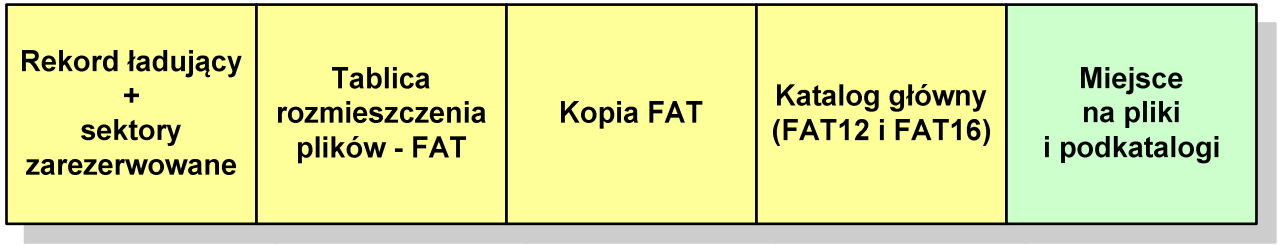
- przykładowa zawartość katalogu głównego:

```
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

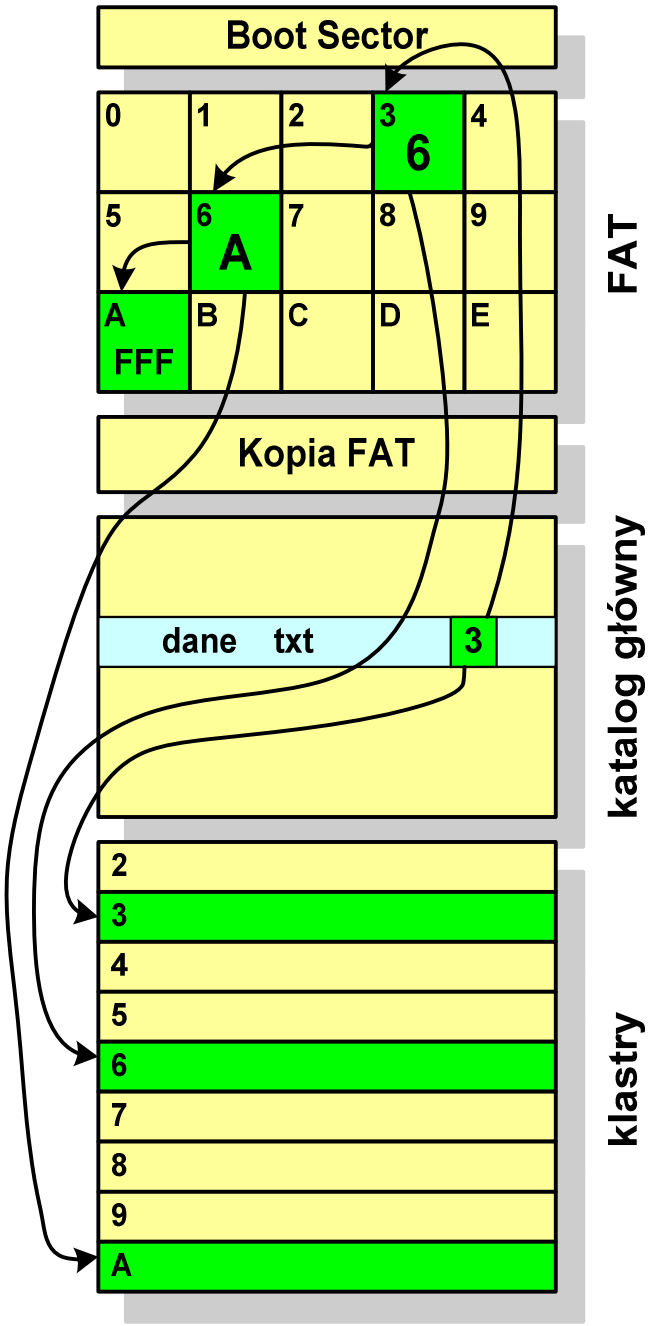
- pozostałą część dysku zajmuje miejsce na pliki i podkatalogi



- 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



Koniec wykładu nr 6

Dziękuję za uwagę!