Systemy Operacyjne

Dr inż. Sławomir Samolej
email: ssamolej@prz-rzeszow.pl
WWW: ssamolej.prz-rzeszow.pl

Slajdy zostały przygotowane
na podstawie materiałów opublikowanych na
(http://wazniak.mimuw.edu.pl/)

Literatura

- A. Silberschatz, J.L. Peterson, G. Gagne, *Podstawy systemów operacyjnych.* WNT, Warszawa 2005

Plan wykładu

- Wprowadzenie
- Procesy, zasoby, wątki
- Planowanie przydziału procesora
- Zarządzanie pamięcią operacyjną
- Urządzenia wejścia-wyjścia
- System plików
- Współbieżność i synchronizacja procesów

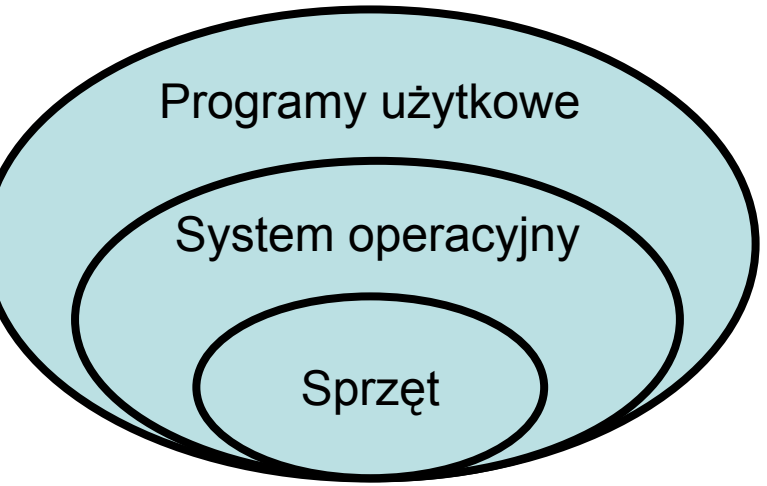
Wprowadzenie

Definicja systemu operacyjnego

System operacyjny (nadzorczy, nadrzędny, sterujący) jest to zorganizowany zespół programów, które **pośredniczą między sprzętem a użytkownikami**, dostarczając użytkownikom zestawu środków **ułatwiających projektowanie, kodowanie, uruchamianie i eksploatację programów** oraz w tym samym czasie sterują przydziałem zasobów dla zapewnienia efektywnego działania.

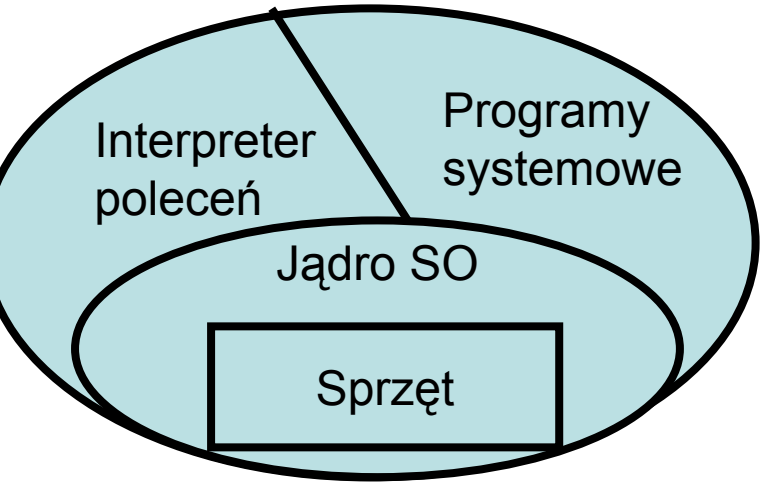
Alan Shaw

SO w architekturze komputera



System operacyjny pośredniczy pomiędzy użytkownikiem a sprzętem, dostarczając wygodnego środowiska do wykonywania programów.
Użytkownik końcowy korzysta z programów (aplikacji), na potrzeby których przydzielane są zasoby systemu komputerowego.
Przydziałem tym zarządza system operacyjny, dzięki czemu można uzyskać stosunkowo duży stopień niezależności programów od konkretnego sprzętu oraz odpowiedni poziom bezpieczeństwa i sprawności działania.

Ogólna struktura systemu operacyjnego

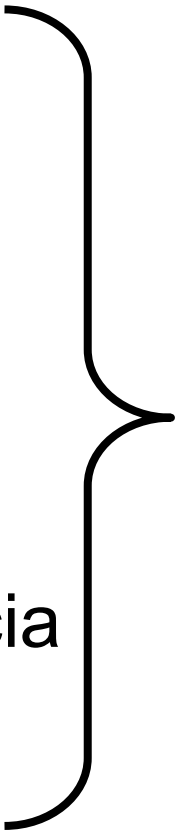


- W ogólnym przypadku w strukturze systemu operacyjnego wyróżnia się **jądro** oraz **programy systemowe**, które dostarczane są razem z systemem operacyjnym, ale nie stanowią integralnej części jądra.

- **Jądro** jest zbiorem modułów, które ukrywają szczegóły sprzętowej realizacji systemu komputerowego, udostępniając pewien zestaw usług, wykorzystywanych między innymi do implementacji programów systemowych.
- **Interpreter** wykonuje pewne polecenia wewnętrznie, tzn. moduł lub program interpretera dostarcza implementacji tych poleceń. Jeśli interpreter nie może wykonać wewnętrznie jakiegoś polecenia, uruchamia odpowiedni program (tzw. polecenie zewnętrzne), jako odrębny proces.

Zadania SO

- Definicja interfejsu użytkownika
- Udostępnianie systemu plików
- Udostępnianie środowiska do wykonywania programów użytkownika
  - mechanizm ładowania i uruchamiania programów
  - mechanizmy synchronizacji i komunikacji procesów
- Sterowanie urządzeniami wejścia-wyjścia
- Obsługa podstawowej klasy błędów



Zarządzanie zasobami systemu komputerowego

- Przydział zasobów
- Planowanie dostępu do zasobów
- Ochrona i autoryzacja dostępu do zasobów
- Odzyskiwanie zasobów
- Rozliczanie — gromadzenie danych o wykorzystaniu zasobów

Zasoby zarządzane przez SO (1)

- Procesor
  - przydział czasu procesora
- Pamięć
  - alokacja przestrzeni adresowej dla procesów
  - ochrona i transformacja adresów

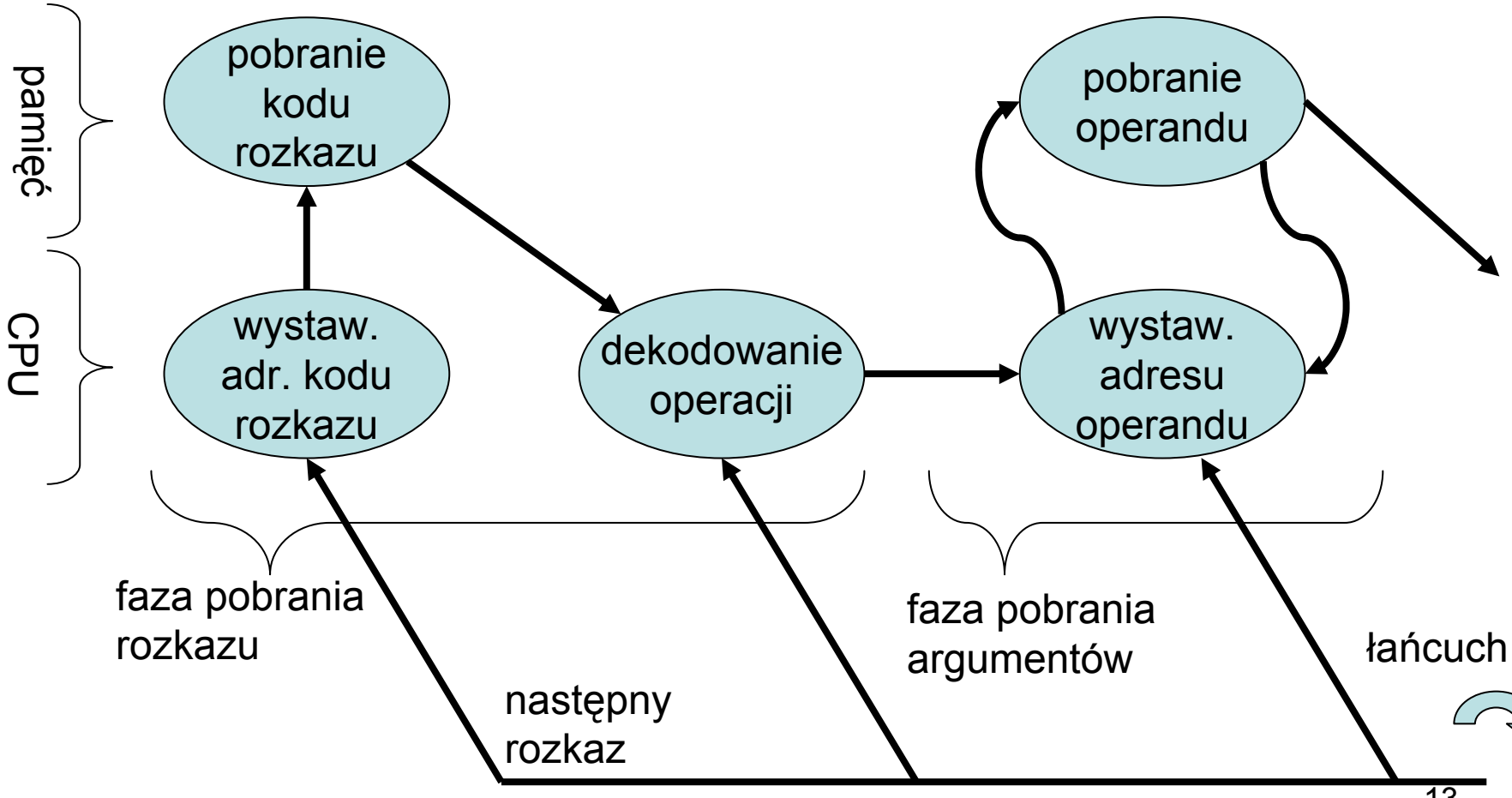
Zasoby zarządzane przez SO (2)

- Urządzenia wejścia-wyjścia
  - udostępnianie i sterowanie urządzeniami pamięci masowej
  - alokacja przestrzeni dyskowej
  - udostępnianie i sterownie drukarkami, skanerami itp.
- Informacja (system plików)
  - organizacja i udostępnianie informacji
  - ochrona i autoryzacja dostępu do informacji

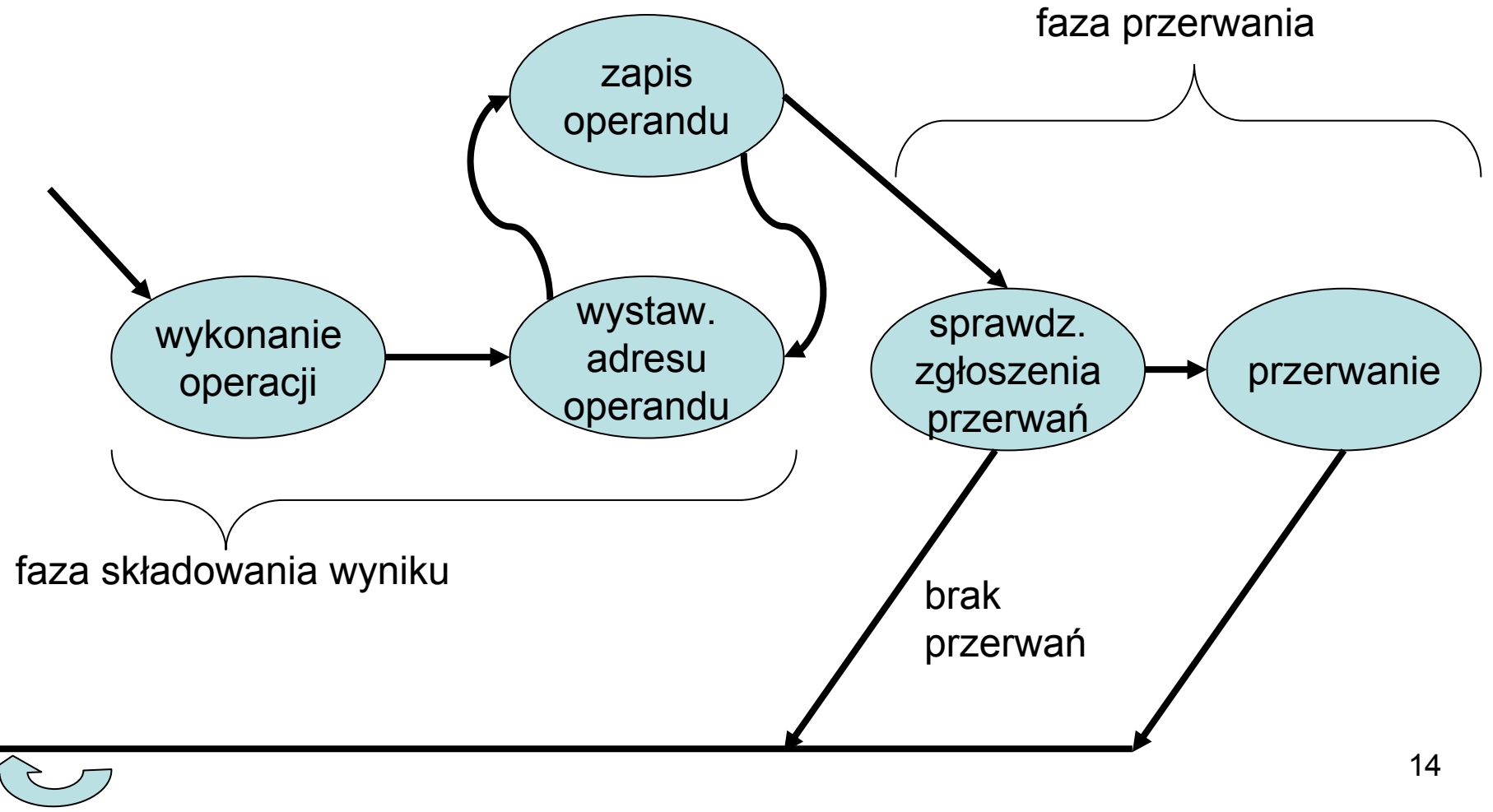
Inne rodzaje systemów operacyjnych

- **Systemy czasu rzeczywistego** (ang. real-time systems) — zorientowane na przetwarzanie z uwzględnieniem czasu zakończenie zadania, tzw. linii krytycznej (ang. deadline), np. VxWorks, QNX, Windows CE, RT Linux.
- **Systemy sieciowe i rozproszone** (ang. network and distributed systems) — umożliwiają zarządzanie zbiorem rozproszonych jednostek przetwarzających, czyli zbiorem jednostek (komputerów), które są zintegrowane siecią komputerową i nie współdzielą fizycznie zasobów, np. Windows, Novel Netware, Linux, Unix .
- **Systemy operacyjne komputerów naręcznych** — tworzone dla rozwiązań typu PDA, czy telefonów komórkowych, podlegają istotnym ograniczeniom zasobowym, np. Symbian, Windows Mobile, Brew, OS X iPhone.

Cykl rozkazowy (1)



Cykl rozkazowy (2)



Podstawy działania systemu operacyjnego

- Odwołania do jądra systemu przez system przerwań lub specjalne instrukcje (przerwanie programowe)
- Sprzętowa ochrona pamięci
- Dualny tryb pracy — tryb użytkownika (ang. user mode) i tryb systemowy (tryb jądra, ang. system mode)
- Wyróżnienie instrukcji uprzywilejowanych, wykonywanych tylko w trybie systemowym
- Uprzywilejowanie instrukcji wejścia-wyjścia
- Przerwanie zegarowe

Procesy, zasoby, wątki

Koncepcja procesu

- **Proces** jest elementarną jednostką pracy (aktywności) zarządzaną przez system operacyjny, która ubiega się o zasoby systemu komputerowego w celu wykonania programu.
- **Proces** = wykonujący się program.
- **Elementy składowe** procesu:
  - program — definiuje zachowanie procesu,
  - dane — zbiór wartości przetwarzanych oraz wyniki,
  - zbiór zasobów tworzących środowisko wykonawcze,
  - blok kontrolny procesu (PCB, deskryptor) — opis bieżącego stanu procesu.

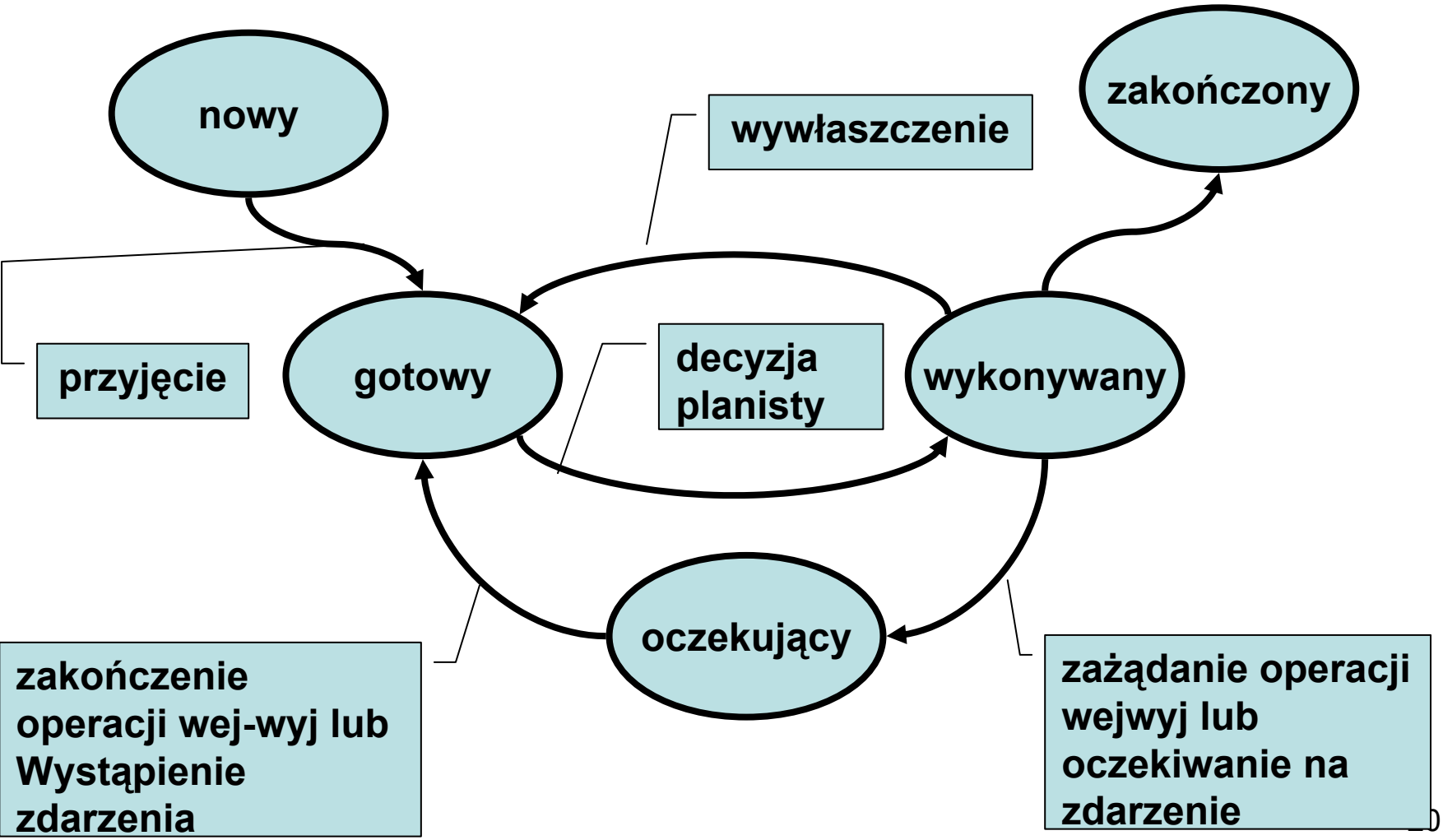
Koncepcja zasobu

- Zasobem jest element sprzętowy lub programowy systemu komputerowego, którego brak może potencjalnie zablokować wykonywanie programu (przetwarzanie)
- Przykłady zasobów: procesor, pamięć, plik (dane) itp.

Stany procesu

- **Nowy** (ang. new) — proces jest tworzony.
- **Wykonywany** (ang. running) — wykonywane są instrukcje programu.
- **Oczekujący** (ang. waiting) — proces oczekuje na jakieś zdarzenie, np. na zakończenie operacji wejścia-wyjścia, na przydział dodatkowego zasobu, synchronizuje się z innymi procesami.
- **Gotowy** (ang. ready) — proces czeka na przydział procesora.
- **Zakończony** (ang. terminated) — proces zakończył działanie i zwalnia zasoby.

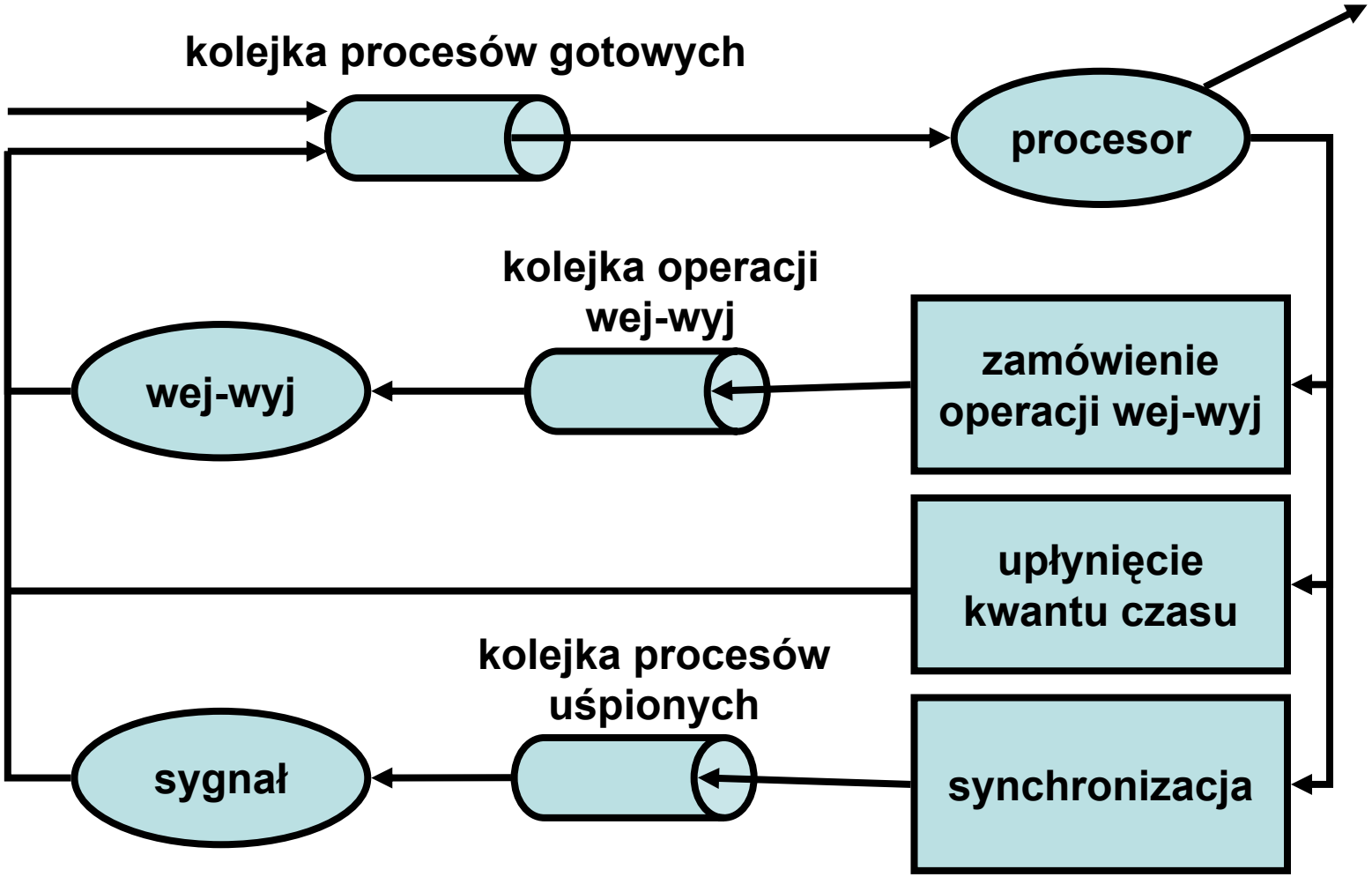
Cykl zmian stanów procesu



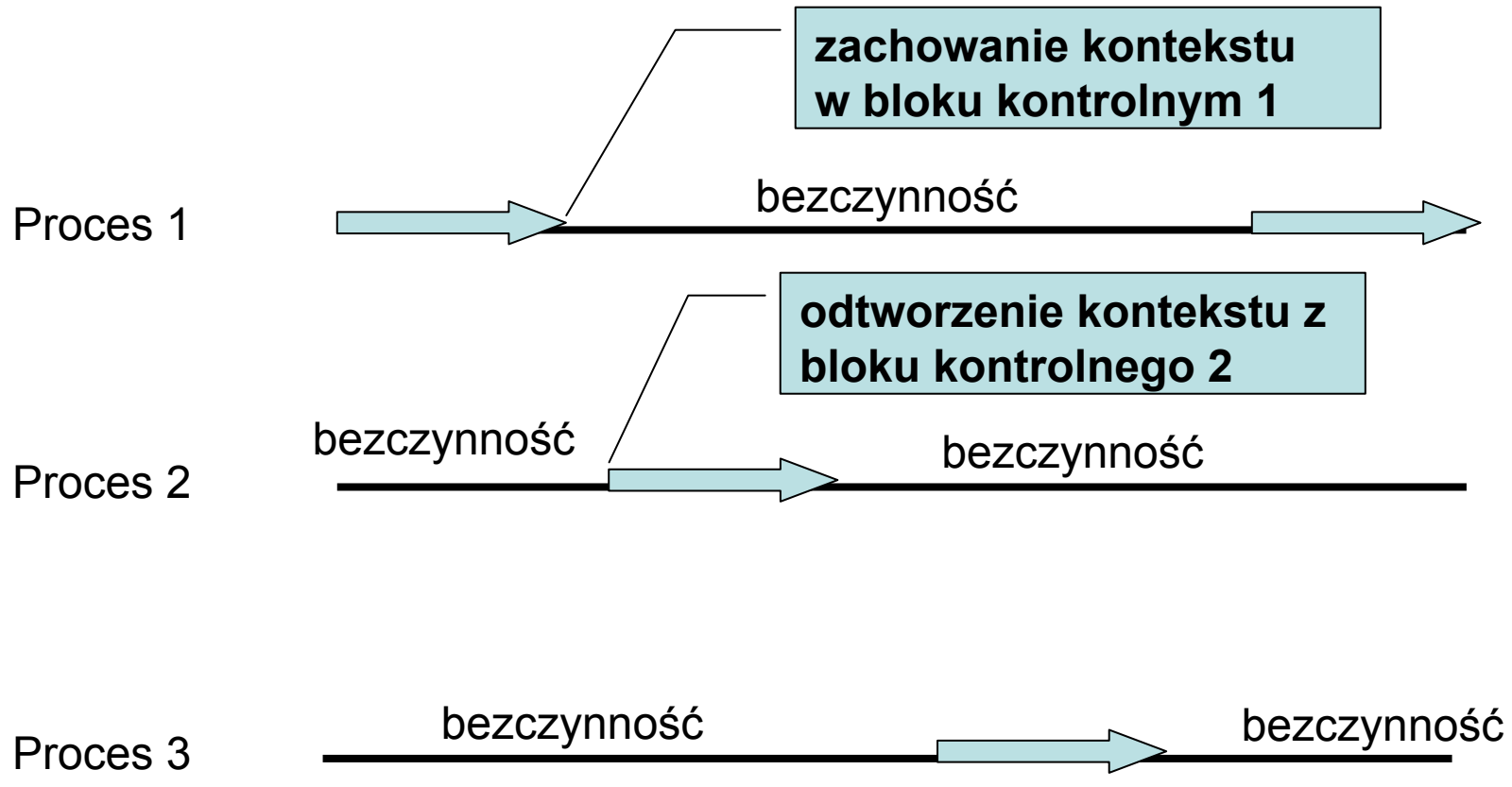
Kolejki procesów

- **Kolejka zadań** (ang. job queue) — wszystkie procesy systemu.
- **Kolejka procesów** gotowych (ang. ready queue) — procesy gotowe do działania, przebywające w pamięci głównej.
- **Kolejka do urządzenia** (ang. device queue) — procesy czekające na zakończenie operacji wejścia-wyjścia.
- **Kolejka procesów oczekujących na sygnał** synchronizacji od innych procesami (np. kolejka procesów na semaforze).

Diagram kolejek w planowaniu przydziału procesora



Przełączanie kontekstu



Wątki

- **Wątek** (lekki proces, ang. lightweight process — LWP) jest obiektem w obrębie procesu ciężkiego (heavyweight), posiadającym własne sterowanie i współdzielącym z innymi wątkami tego procesu przydzielone (procesowi) zasoby:
  - segment kodu i segment danych w pamięci
  - tablicę otwartych plików
  - tablicę sygnałów

Planowanie przydziału procesora

Komponenty jądra w planowaniu

- **Planista krótkoterminowy** (ang. CPU scheduler) — wyznacza wartość priorytetu procesów gotowych i wybiera proces (o najwyższym priorytecie) do wykonania.
- **Ekspedytor** (zwany również dyspozytorem) ang. dispatcher) — realizuje przekazanie sterowanie do procesu wybranego przez planistę (dokonuje przełączenia kontekstu).

Ogólna koncepcja planowania

- **Tryb decyzji** — określa okoliczności, w których oceniane i porównywane są priorytety procesów oraz dokonywany jest wybór procesu do wykonania.
- **Funkcja priorytetu** — funkcja wyznaczająca aktualny priorytet procesu na podstawie parametrów procesu i stanu systemu.
- **Reguła arbitrażu** — reguła rozstrzygania konfliktów w dostępie do procesora w przypadku procesów o tym samym priorytecie

Tryb decyzji

- **Schemat niewywłaszczeniowy** (ang. nonpreemptive) —proces po uzyskaniu dostępu do procesora wykonywany jest do momentu zakończenie lub zgłoszenia żądania obsługi do systemu.
- **Schemat wywłaszczeniowy** (ang. preemptive) — proces może zostać zatrzymany i umieszczony w kolejce procesów gotowych, a procesor zostaje przydzielony procesowi o wyższym (lub równym) priorytecie.

Podejmowanie decyzji o wywłaszczeniu

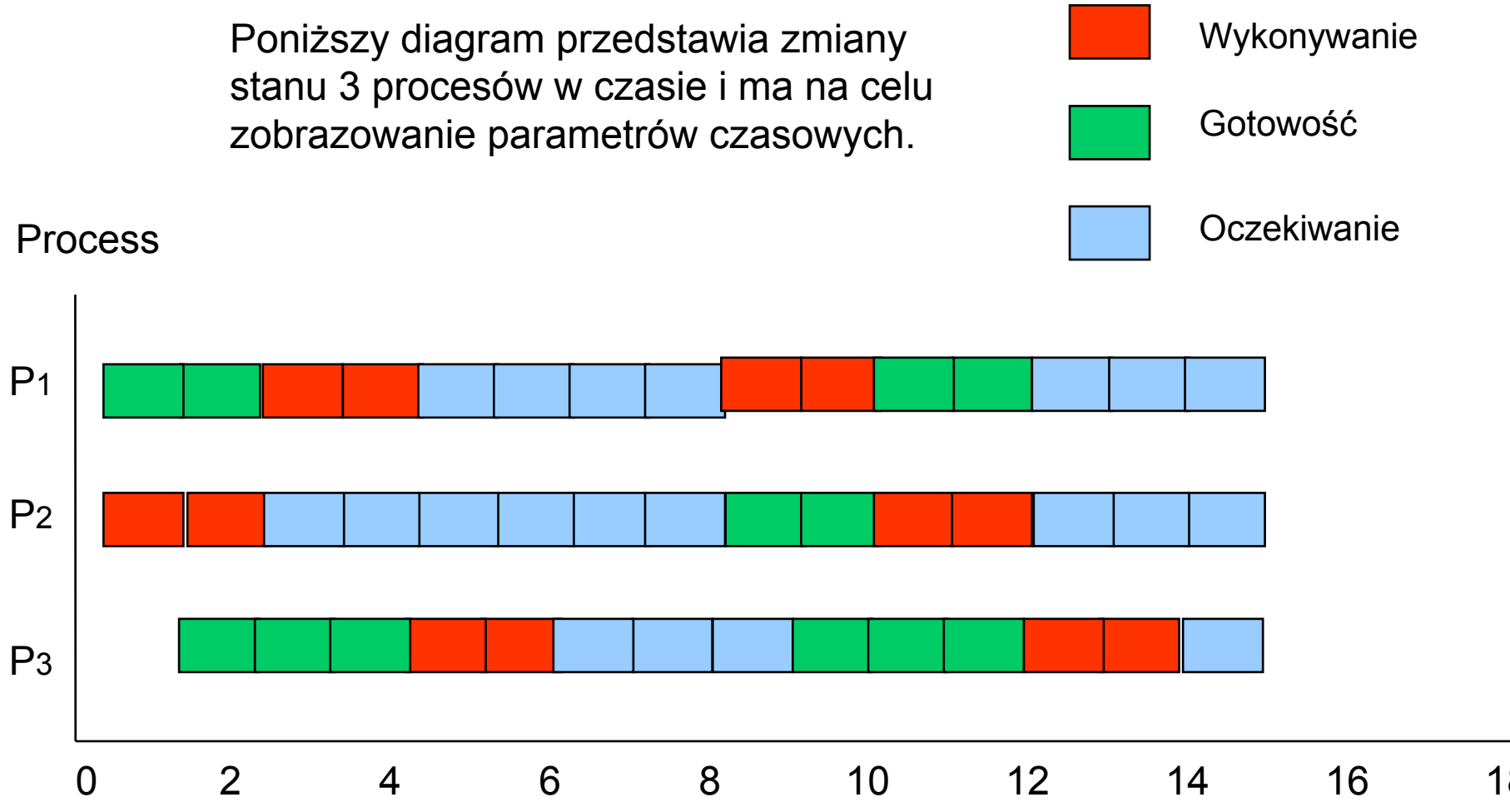
- Utworzenie i przyjęcie nowego procesu
- Obudzenie procesu w wyniku otrzymania komunikatu, sygnału gotowości urządzenia (przerwanie) lub sygnału wynikającego z synchronizacji
- Upłynięcie kwantu czasu odmierzanego przez czasomierz
- Wzrost priorytetu innego procesu w stanie gotowy powyżej priorytetu procesu wykonywanego — możliwe w systemie ze zmiennymi priorytetami

Funkcja priorytetu

- Argumentami funkcji priorytetu są wybrane składowe stanu procesu oraz stanu systemu.
- Priorytet procesu w danej chwili jest wartością wynikową funkcji priorytetu dla bieżących wartości parametrów stanu danego procesu i aktualnego stanu systemu.

Przykład realizacji przetwarzania

Poniższy diagram przedstawia zmiany stanu 3 procesów w czasie i ma na celu zobrazowanie parametrów czasowych.



Reguła arbitrażu

- **Losowo** — możliwe w przypadku, gdy liczba procesów o tym samym priorytecie jest niewielka
- **Cyklicznie** — cykliczny przydział procesora kolejnym procesom
- **Chronologicznie** — w kolejności przyjmowania procesów do systemu (w kolejności FIFO)

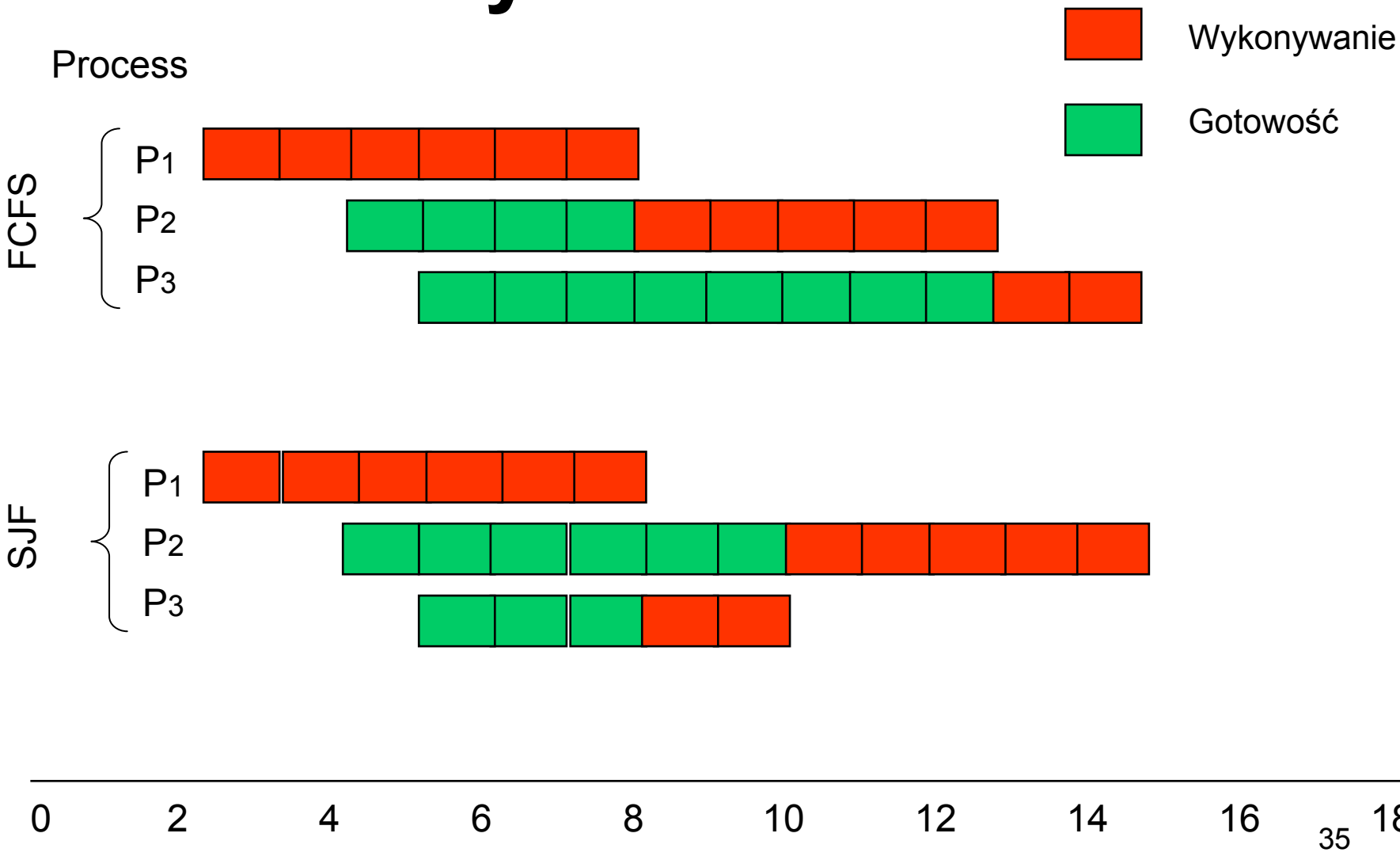
Algorytmy planowania niewywłaszczającego

- FCFS (First Come First Served) — pierwszy zgłoszony, pierwszy obsłużony
- LCFS (Last Come First Served) — ostatni zgłoszony, pierwszy obsłużony
- SJF (SJN, SPF, SPN, Shortest Job/Process First/Next) — najpierw najkrótsze zadanie

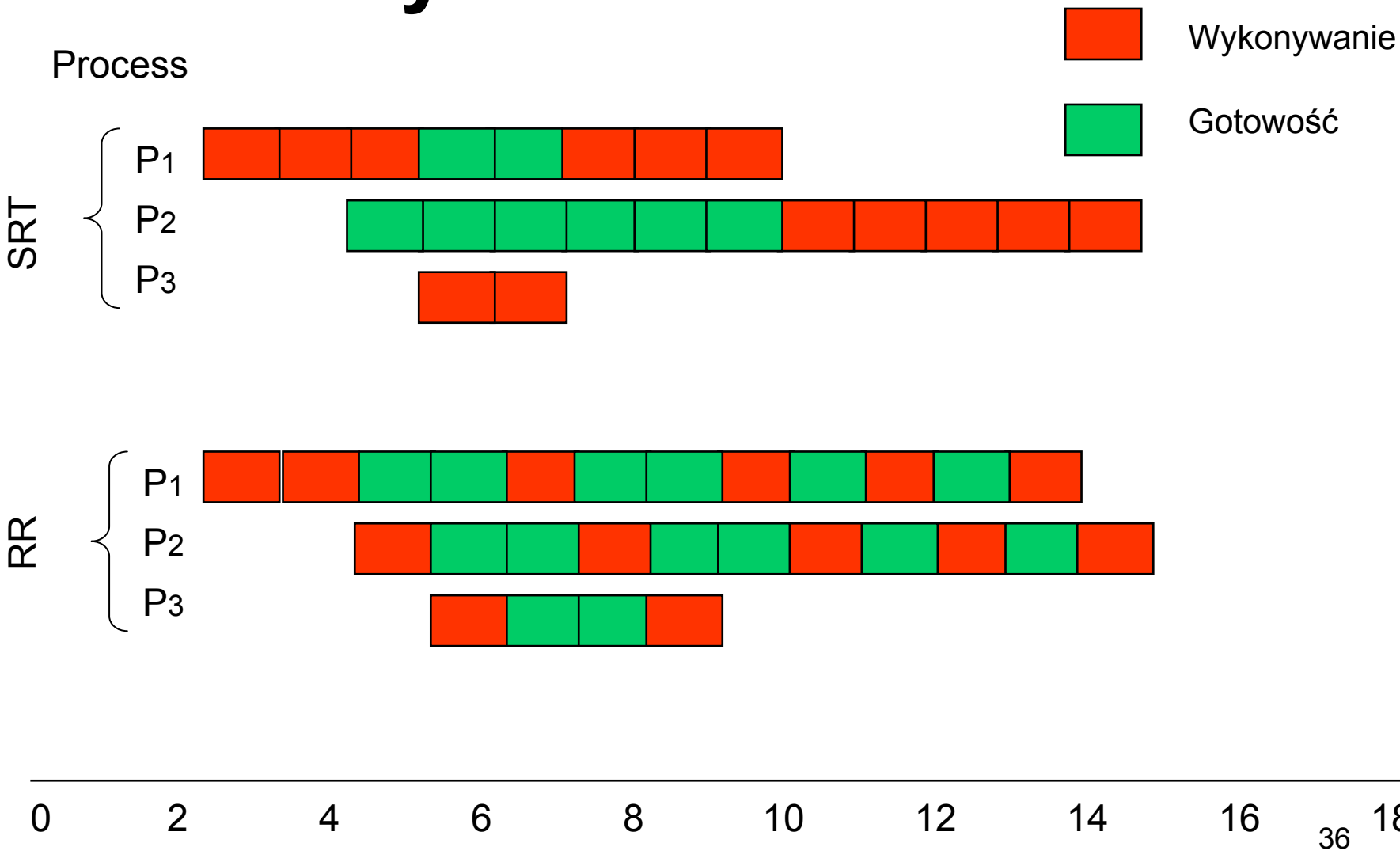
Algorytmy planowania wywłaszczającego

- Planowanie rotacyjne (ang. Round Robin, RR) — po ustalonym kwancie czasu proces wykonywany jest przerywany i trafia do kolejki procesów gotowych.
- SRT (Shortest Remaining Time) — najpierw zadanie, które ma najkrótszy czas do zakończenia

Przykłady uszeregowania bez wywłaszczeń



Przykłady uszeregowania z wywłaszczeniami



Algorytm RR — dobór kwantu czasu

- Krótki kwant czasu oznacza zmniejszenie czasu cyklu przetwarzania procesów krótkich, ale zwiększa narzut czasowy związany z przełączaniem kontekstu.
- Z punktu widzenia interakcji z użytkownikiem kwant czasu powinien być trochę większy, niż czas odpowiedzi (reakcji).

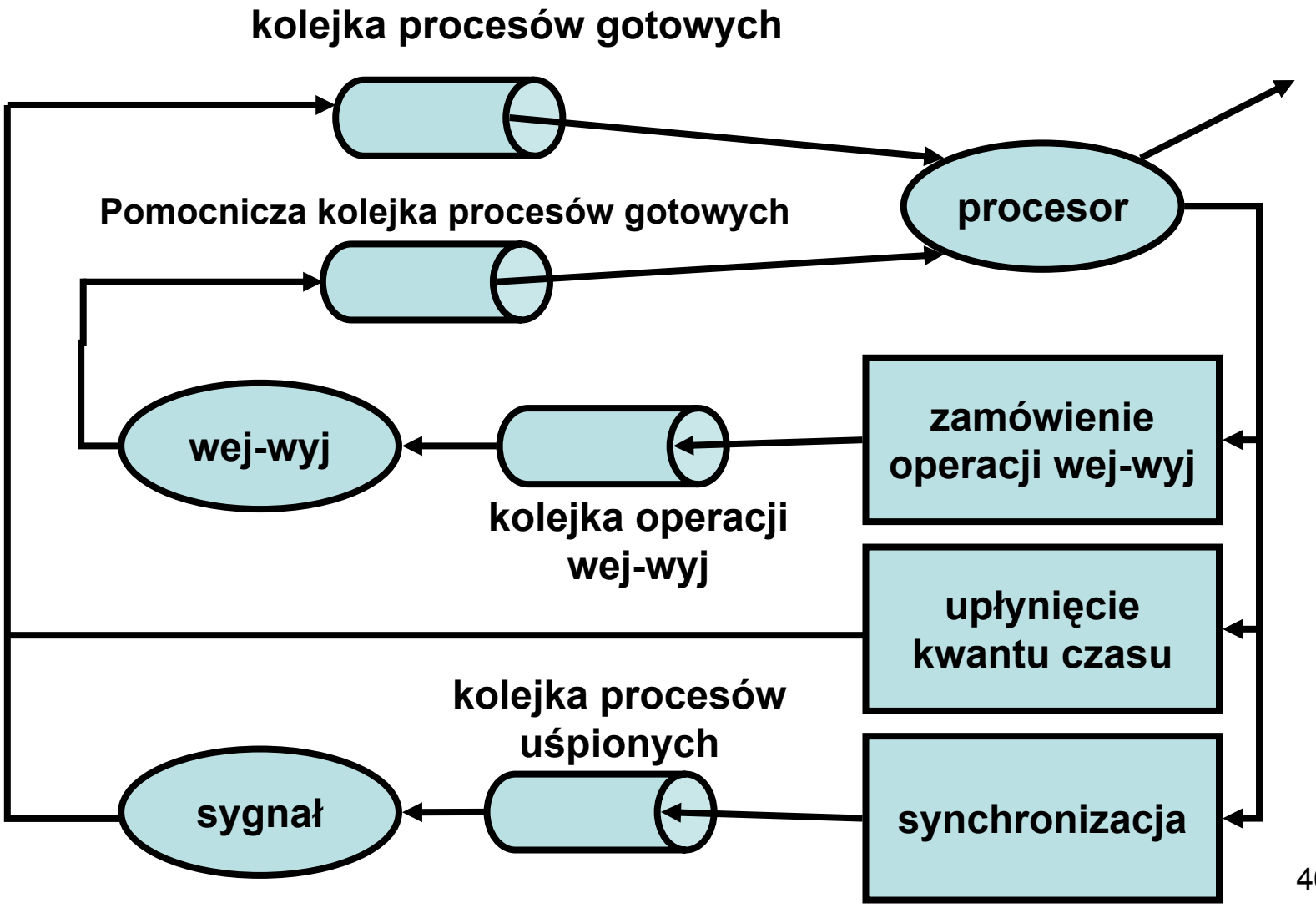
Inne algorytmy planowania

- Planowanie priorytetowe — oparte na priorytecie zewnętrznym
- Planowanie wielokolejkowe — w systemie jest wiele kolejek procesów gotowych i każda z kolejek może być inaczej obsługiwana.
- Planowanie przed liniami krytycznymi — zakończenie zadania przed czasową linią krytyczną lub możliwie krótko po tej linii

Szeregowanie procesów, ograniczonych wejściem-wyjściem

- Procesy ograniczone wejściem-wyjściem potrzebują niewiele czasu procesora, większość czasu w systemie spędzając na oczekiwaniu na urządzenia zewnętrzne.
- Opóźnianie przydziału procesora dla tego typu procesów powoduje zmniejszenie wykorzystania urządzeń zewnętrznych, a przydział — ze względu na nie długą fazę procesora — nie powoduje istotnego zwiększenia czasu oczekiwania innych procesów.
- Właściwym algorytmem byłby SJF lub SRT.
- Bezwzględna preferencja dla procesów oczekujących na gotowość urządzeń może spowodować głodzenie procesów ograniczonych procesorem.

Wirtualne planowanie rotacyjne



Zarządzanie pamięcią operacyjną

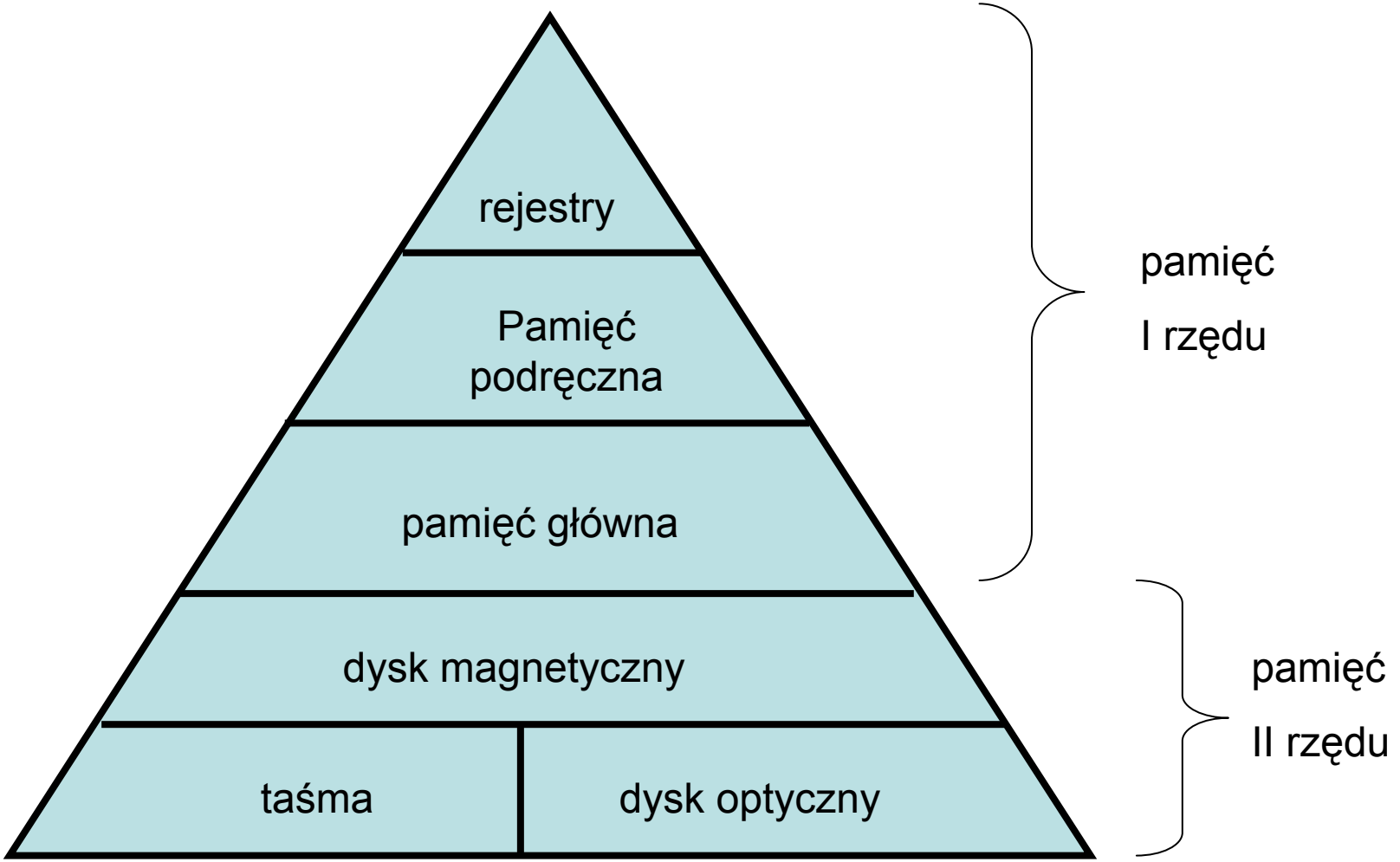
Podstawowe zagadnienia związane z zarządzaniem pamięcią operacyjną

- Przydział pamięci i jej odzyskiwanie
- Ochrona
- Udostępnienie w celu współdzielenia
- Transformacja adresów
- Transfer danych pomiędzy poszczególnymi poziomami w hierarchii pamięci.

Pamięć jako zasób systemu komputerowego

- Pamięć jest zasobem służący do przechowywania danych i programów.
- Z punktu widzenia systemu pamięć jest zasobem o strukturze hierarchicznej (począwszy od rejestrów procesora, przez pamięć podręczną, pamięć główną, skończywszy na pamięci masowej), w której na wyższym poziomie przechowywane są dane, stanowiące fragment zawartości poziomu niższego.
- Z punktu widzenia procesu (również procesora) pamięć jest zbiorem bajtów identyfikowanych przez adresy, czyli tablicą bajtów, w której adresy są indeksami.

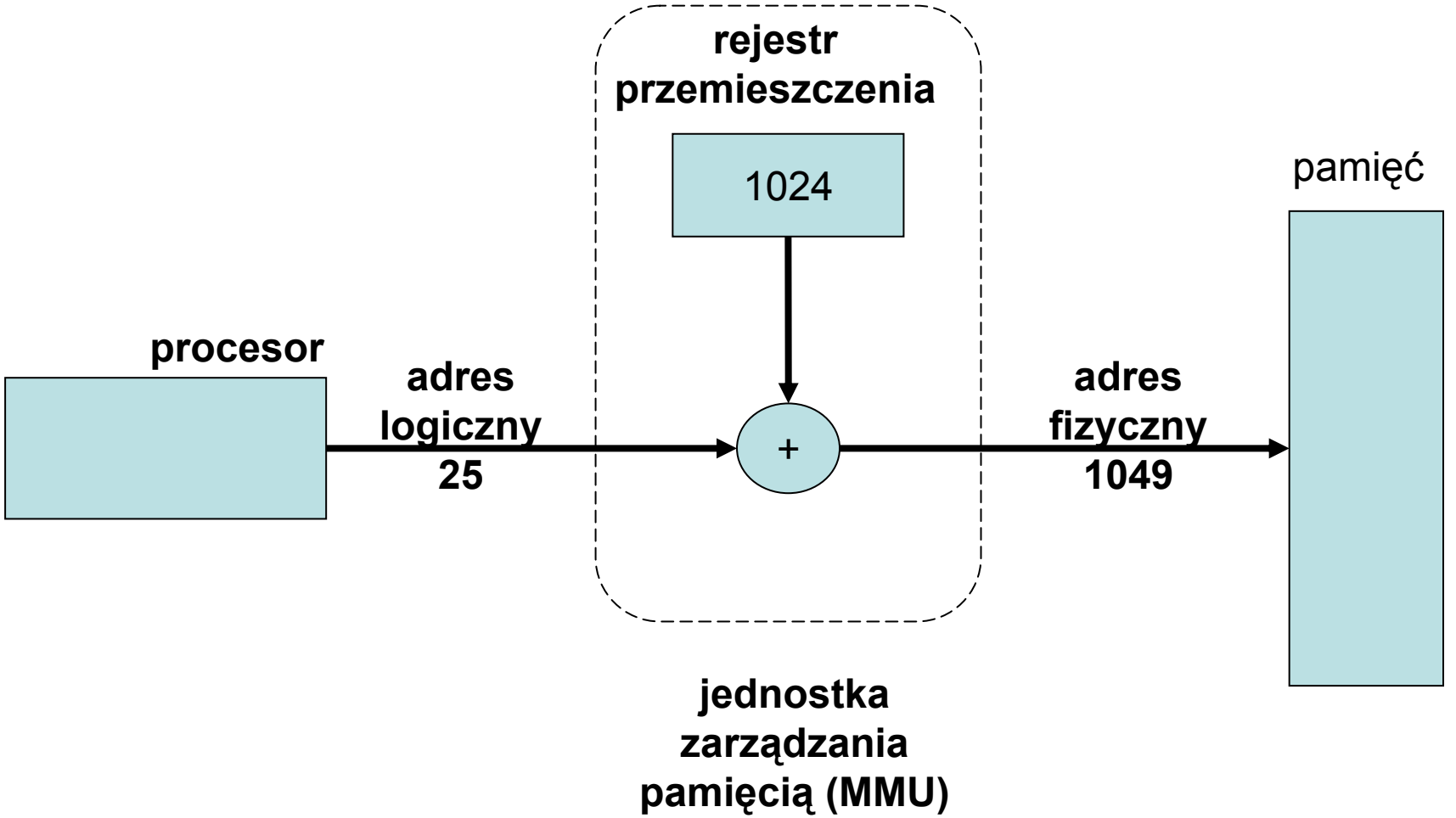
Hierarchia pamięci



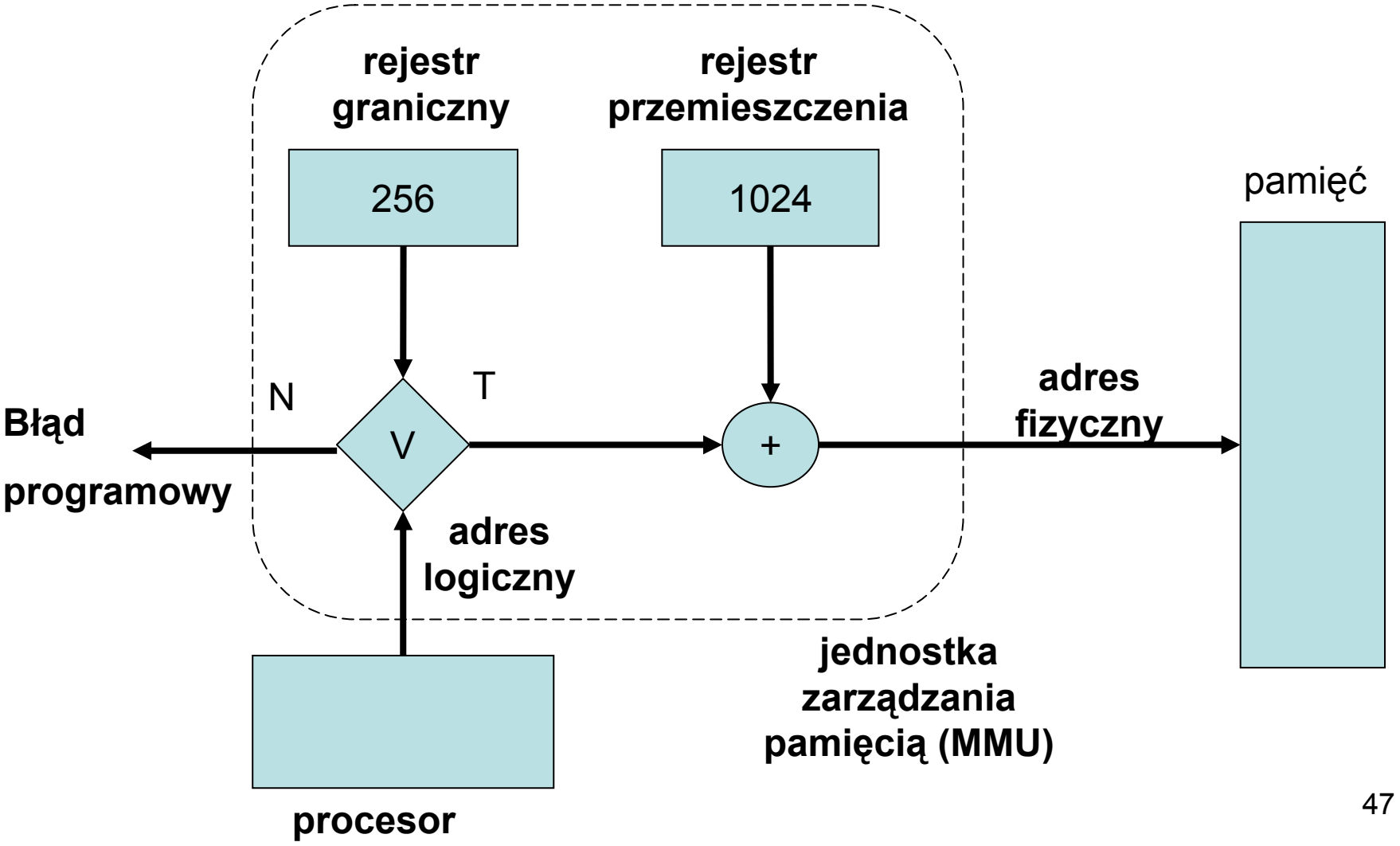
Przestrzeń adresowa

- Przestrzeń adresowa jest zbiór wszystkich dopuszczalnych adresów w pamięci.
- W zależności od charakteru adresu odróżnia się:
  - przestrzeń fizyczną — zbiór adresów przekazywanych do układów pamięci głównej (fizycznej).
  - przestrzeń logiczną — zbiór adresów generowanych przez procesor w kontekście aktualnie wykonywanego procesu.

Przykład odwzorowania adresu logicznego na fizyczny



Przykład weryfikacji poprawności adresu



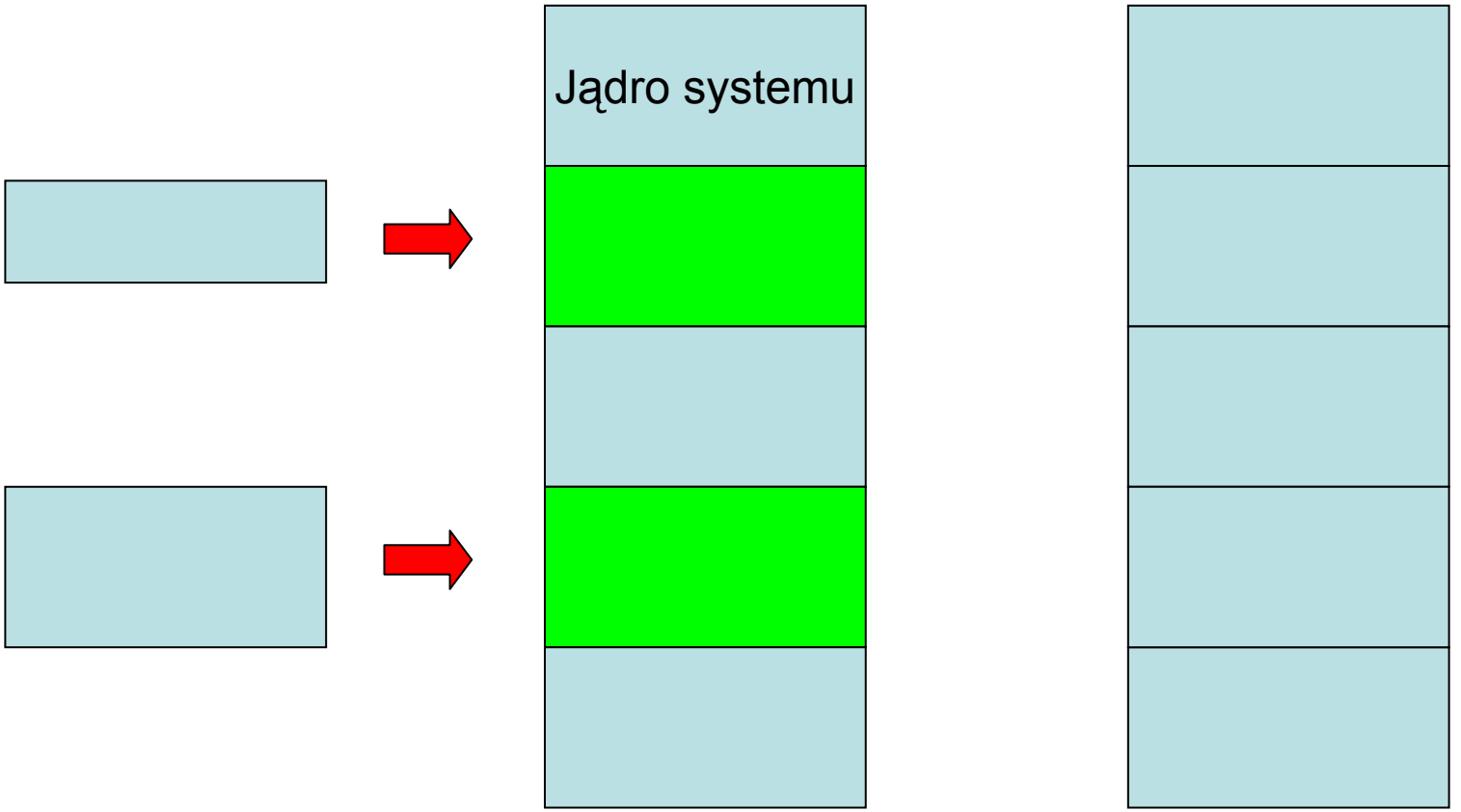
Podział pamięci

- Podział stały
  - partycje o równym rozmiarze
  - partycje o różnych rozmiarach
- Podział dynamiczny
- Podział na bloki bliźniacze (zwany też metodą sąsiedzkich stert)

Podział stały

- Podział pamięci na stałe obszary (strefy, partycje), których rozmiar i położenie ustalane są na etapie konfiguracji systemu.
- Przydział całego obszaru o rozmiarze większym lub równym zapotrzebowaniu, określonym w żądaniu.
- Zalety: łatwość implementacji i zarządzania
- Wady: słaba efektywność wykorzystania pamięci (fragmentacja, ograniczona odgórnie liczba jednocześnie przydzielonych partycji).

Podział stały — partycje o równym rozmiarze



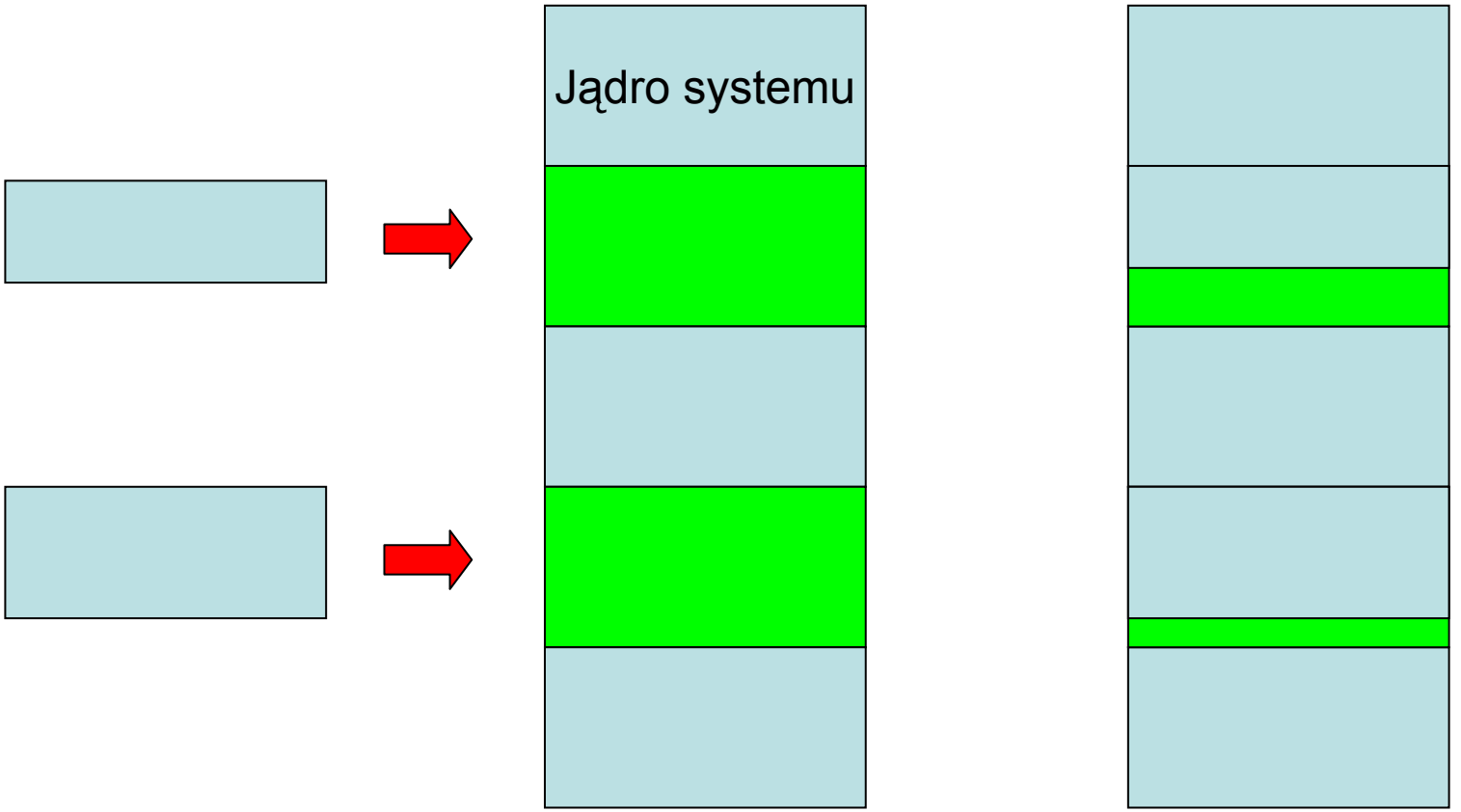
obraz przed przydziałem

obraz po przydziale

Podział dynamiczny

- Podział pamięci tworzony jest w czasie pracy systemu stosownie do żądań procesów.
- Proces ładowany jest w obszar o rozmiarze dosyć dokładnie odpowiadającym jego wymaganiom.
- Zalety: lepsze wykorzystanie pamięci (brak fragmentacji wewnętrznej)
- Wady: skomplikowane zarządzanie, wynikające z konieczności utrzymywania odpowiednich struktur danych w celu identyfikacji obszarów zajętych oraz wolnych.

Obraz pamięci przy podziale dynamicznym



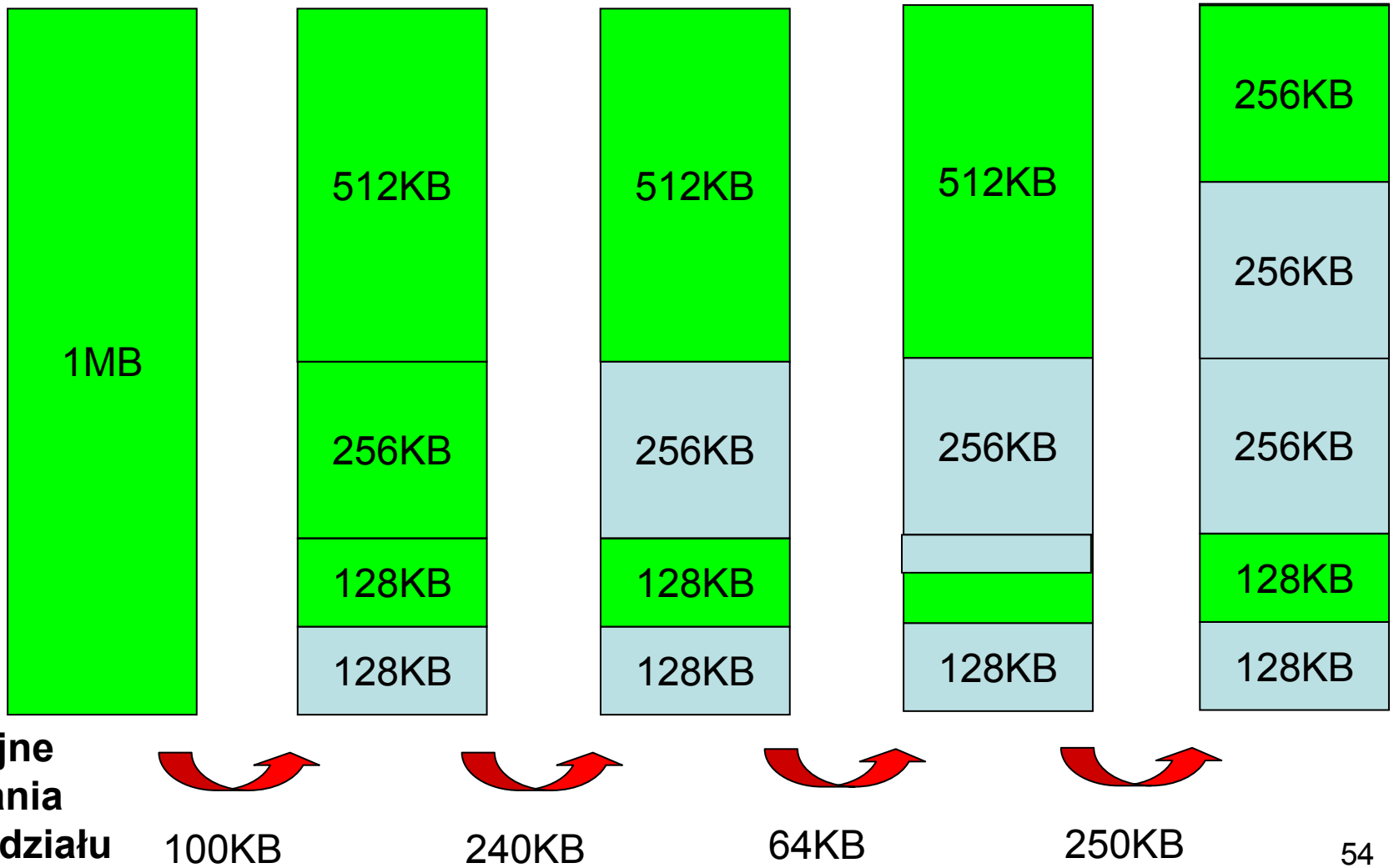
obraz przed przydziałem

obraz po przydziale

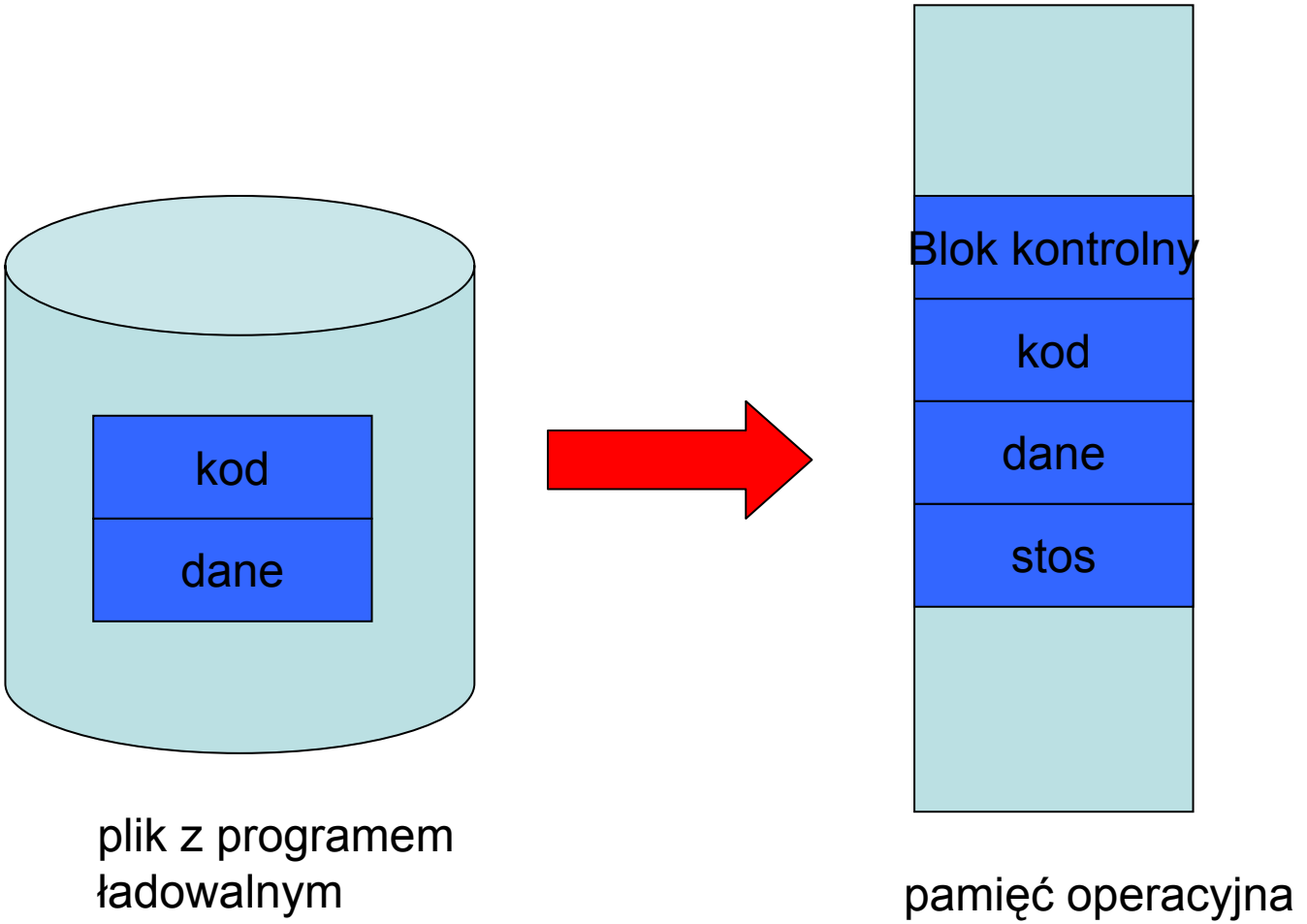
System bloków bliźniaczych

- Pamięć dostępna dla procesów użytkownika ma rozmiar 2^U.
- Przydzielany blok ma rozmiar 2^K, gdzie $L \leq K \leq U$.
- Początkowo dostępny jest jeden blok o rozmiarze 2^U.
- Realizacja przydziału obszaru o rozmiarze s polega na znalezieniu lub utworzeniu (przez połowienie) bloku o rozmiarze 2^i takim, że $2^{i-1} < s \leq 2^i$.

System bloków bliźniaczych — przykład



Obraz procesu w pamięci



Pamięć wirtualna

- Wykonywane programy przechowywane są nie tylko w pamięci fizycznej ale i w masowej.
- Jeśli zachodzi konieczność wykonania fragmentu programu przeniesionego do pamięci masowej, następuje przeniesienie tego fragmentu do pamięci fizycznej.
- Zalety:
  - „rozszerzenie” pamięci fizycznej
  - racjonalizacja wykorzystania pamięci fizycznej
  - Możliwość zmniejszenia czasu odpowiedzi (np. kod jest sprowadzany w niewielkich porcjach na żądanie)
- Wady:
  - Bardziej złożone zarządzanie pamięcią.
  - Przy zbyt dużym obciążeniu – redukcja wydajności systemu.

Urządzenia wejścia-wyjścia

Rodzaje urządzeń wejścia-wyjścia

- Urządzenia składowania danych (dyski, dyskietki, taśmy, CD ROM, DVD itp.)
- Urządzenia transmisji danych na odległość (karty sieciowe, modemy)
- Urządzenia do komunikacji z człowiekiem (monitory, projektory, klawiatury, myszy, drukarki, skanery itp.)
- Urządzenia specjalizowane
  - układy sterowania (np. elektrownią, samolotem, systemem obrony antyrakietowej itd.)
  - kasy i drukarki fiskalne itp.
  - urządzenia medyczne

Właściwości urządzeń wejścia wyjścia (1)

- Tryb transmisji danych:
  - znakowy — przykazywanie danych odbywa się bajt po bajcie, przykład: port szeregowy
  - blokowy — przykazywanie danych odbywa się w blokach (np. po 512 bajtów), przykład: dysk
- Sposób dostępu do danych:
  - sekwencyjny — dane przekazywane są w określonym porządku, narzuconym przez urządzenie, przykład: karta sieciowa
  - bezpośredni (swobodny) — możliwe jest określenie lokalizacji danych na urządzeniu, przykład: dysk

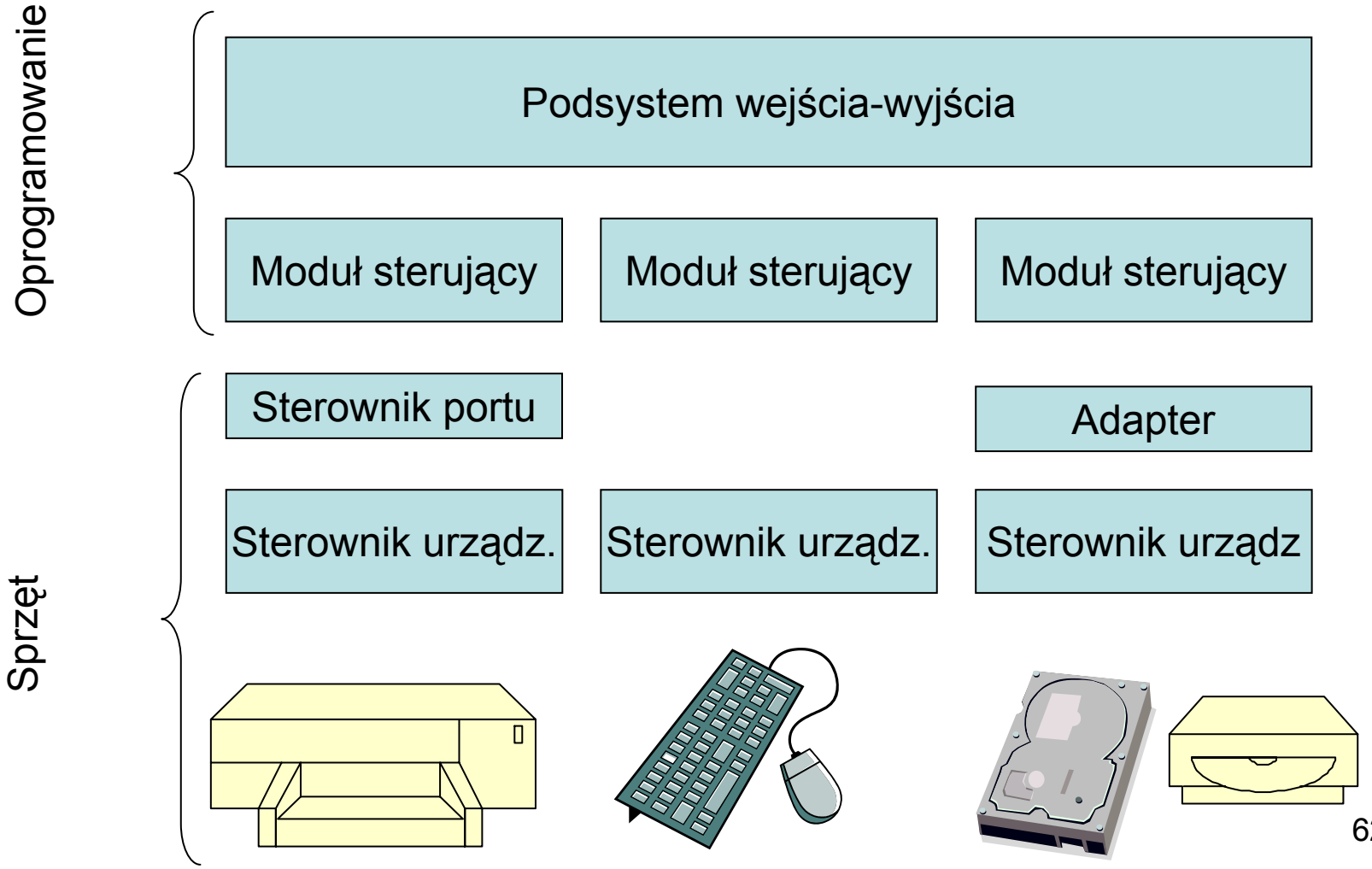
Właściwości urządzeń wejścia wyjścia (2)

- Tryb pracy urządzenia:
  - synchroniczny — dane zostaną przekazane w znanym z góry (przewidywalnym) czasie, przykład: dysk, karta graficzna
  - asynchroniczny — dane mogą zostać przesłane w dowolnym, trudnym do przewidzenia, momencie, przykład: klawiatura, karta sieciowa
- Tryb użytkowania:
  - współdzielony — dopuszczalne jest współbieżne używanie urządzenia przez wiele procesów, np.: dysk
  - wyłączny — niemożliwe jest współbieżne używanieurządzenia przez wiele procesów, przykład: drukarka

Właściwości urządzeń wejścia wyjścia (3)

- Szybkość działania (transmisji)
  - od bardzo wolnych, przykład: drukarka
  - do stosunkowo szybkich, przykład: dysk
- Kierunek przekazywania danych
  - urządzenia wejścia i wyjścia — możliwość zarówno zapisu jak i odczytu, przykład dysk, karta sieciowa
  - urządzenia wejścia — tylko możliwość odczytu z urządzenia, przykład: klawiatura
  - urządzenia wyjścia — tylko możliwość zapisu,przykład: drukarka

Struktura mechanizmu wejścia-wyjścia



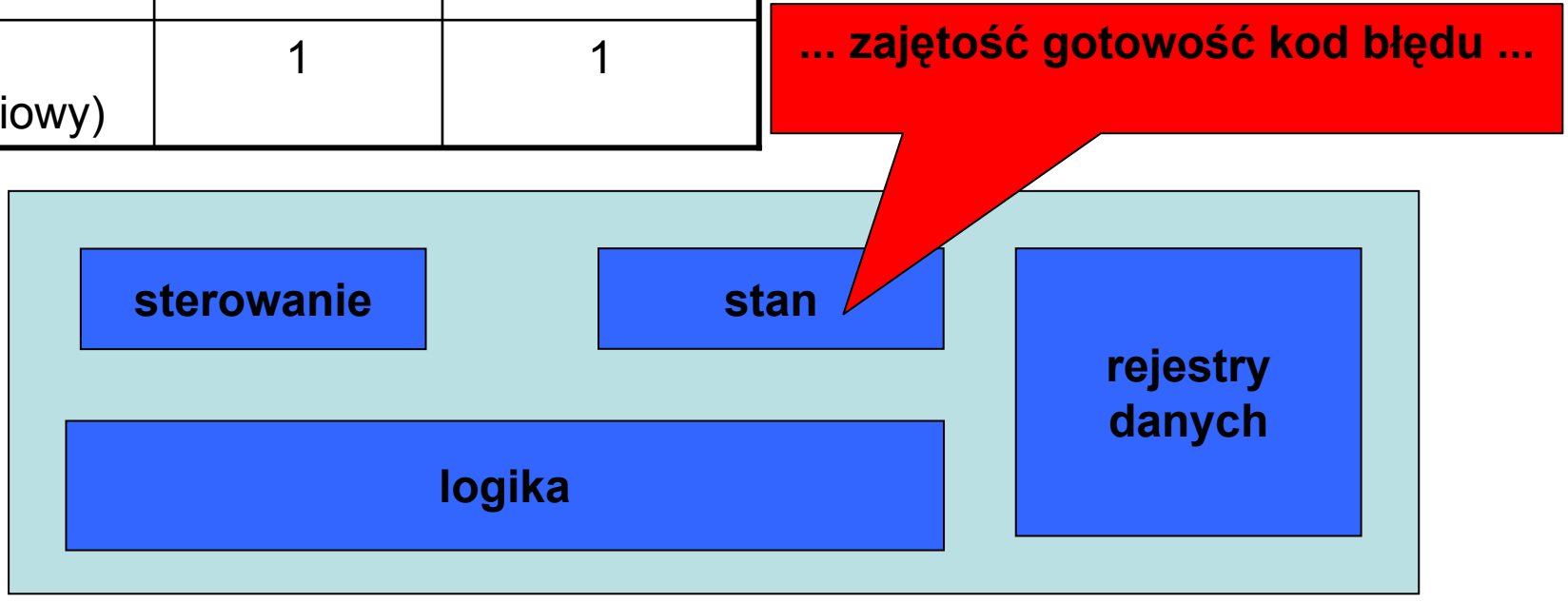
Oprogramowanie obsługi wejścia-wyjścia

- Podsystem wejścia-wyjścia:
  - interfejs wejścia-wyjścia — specyfikacja operacji (API), umożliwiających jednolity sposób dostępu do urządzeń wejścia-wyjścia na poziomie aplikacji
  - buforowanie
- Moduł sterujący — ukrywanie sprzętowych szczegółów realizacji danego urządzenia przed interfejsem wejściawyjścia:
  - dostarczanie implementacji dla operacji z interfejsu wejścia-wyjścia w odniesieniu do danego urządzenia
  - kontrola realizacji operacji na urządzeniu (sprawdzanie stanu, poprawności, transfer danych)

Sterownik urządzenia

Rejestr stanu:

	zajętość	gotowość
Bezczynność	0	0
Zakończenie	0	1
Praca	1	0
(stan przejściowy)	1	1



Miejsce urządzeń wejścia-wyjścia w architekturze systemu komputerowego

- Odwzorowanie w przestrzeni adresowej wejścia-wyjścia (tzw. izolowane wejście-wyjście) — rejestry sterownika widoczne są w przestrzeni adresowej wejścia-wyjścia systemu komputerowego i dostępne są przez specjalne rozkazy (np. in i out w procesorach firmy Intel).
- Odwzorowanie w przestrzeni adresowej pamięci — rejestry sterownika widoczne są w przestrzeni adresowej pamięci fizycznej i dostępne są pod odpowiednimiadresami tak samo, jak inne komórki pamięci.

Interakcja jednostki centralnej ze sterownikiem urządzenia wejścia-wyjścia

- **Odpytywanie** (ang. polling) — ciągłe lub okresowe sprawdzanie stanu sterownika
- **Sterowanie przerwaniami** (ang. interrupt-driven I/O) — inicjalizacja pracy sterownika przez procesor i obsługa urządzenia po zakończeniu działania w ramach reakcji na przerwanie
- **Bezpośredni dostęp do pamięci** (ang. direct memory access) — inicjalizacja pracy sterownika przez procesor i uruchomienie układu bezpośredniego dostępu do pamięci w celu realizacji transferu danych pomiędzy sterownikiem a pamięcią

Buforowanie

- **Dopasowanie urządzeń różniących się szybkością przekazywania danych** — dopasowanie chwilowo szybszego producenta danych do możliwości konsumenta.
- **Dopasowanie urządzeń różniących się podstawową jednostką transmisji danych** — dopasowanie w celu efektywnego przekazywania danych urządzeń przesyłających mniejsze jednostki danych do urządzeń wymagających większych jednostek lub odwrotnie (fragmentowanie).
- **Semantyka kopii** — zagwarantowanie niezmienności danych w czasie wykonywania operacji wejścia-wyjścia.

System plików

Pojęcie pliku

- Intuicyjnie plik jest ciągiem danych (bitów, bajtów, rekordów itp.), których znaczenie (semantykę) określa jego twórca i jego użytkownik. Np. użytkownik, tworząc plik z programem w języku C, określa, że jest to plik, na podstawie którego kompilator potrafi wygenerować kod pośredni, a po dołączeniu odpowiednich bibliotek konsolidator (linker) potrafi wygenerować plik z programem binarnym.

Zadania systemu operacyjnego w odniesieniu do plików

- Zadaniem systemu operacyjnego w odniesieniu do plików jest zapewnienie odwzorowania pomiędzy abstrakcyjnym obrazem informacji a jego reprezentacjąna urządzeniu fizycznym.
- Wyszczególnienie zadań:
  - identyfikacja pliku (hierarchiczna struktura katalogów),
  - udostępnienie interfejsu operacji plikowych (API),
  - realizacja operacji dostępu do plików i katalogów z zapewnieniem bezpieczeństwa (synchronizacja i autoryzacja dostępu), spójności i efektywności.
- Uwaga:
  Istnieją systemy operacyjne (np. czasu rzeczywistego), w których świadomie rezygnuje się ze struktury plików dołączając kolejne aplikacje jako moduły jądra ładowane przy starcie komputera.

Atrybuty pliku

- Nazwa — ciąg znaków służących użytkownikowi do identyfikacji pliku
- Typ — informacja służąca do rozpoznania rodzaju zawartości pliku i tym samym sposobu interpretacji
- Lokalizacja — informacja służąca do odnalezienia pliku w systemie komputerowym (urządzenie i położenie pliku w tym urządzeniu)
- Rozmiar — bieżący rozmiar pliku w ustalonych jednostkach (bajtach, słowach, blokach itp.)
- Ochrona — informacje umożliwiające kontrolę dostępu
- Czasy dostępów — daty i czasy wykonywania pewnych operacji na pliku, typu odczyt, modyfikacja, utworzenie

Podstawowe operacje na plikach (1)

- Tworzenie pliku — konieczne jest określenie podstawowych atrybutów pliku, znalezienie miejsca na ten plik w systemie komputerowym oraz jego zaewidencjonowanie (utworzenie wpisu katalogowego)
- Zapis do pliku — konieczne jest określenie, co ma być zapisane i gdzie ma być zapisane (w którym pliku i w jakim miejscu tego plik, zależnie od sposobu dostępu)
- Odczyt z pliku — konieczne jest określenie, co ma być odczytane (z którego pliku i z jakiego miejsca tego plik, zależnie od sposobu dostępu) i gdzie mają być umieszczone odczytane dane

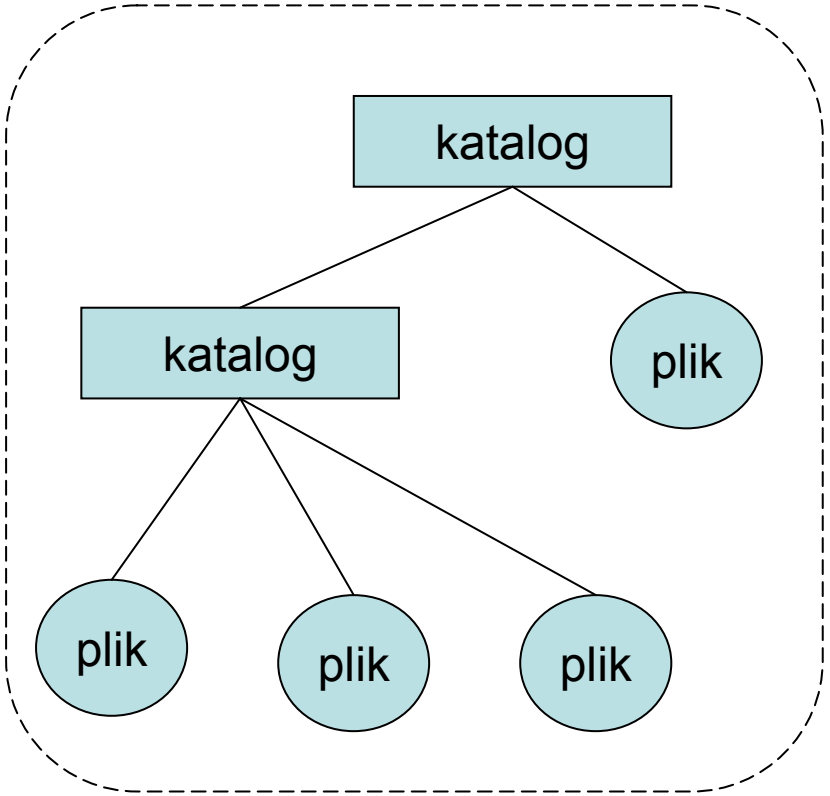
Podstawowe operacje na plikach (2)

- Usuwanie informacji z pliku — należy określić jaki fragment pliku (i którego pliku) ma być usunięty. Najczęściej możliwe jest tylko skracanie pliku, czyli usuwanie jego końcowej zawartości lub całej jego zawartości.
- Usuwanie pliku — należy określić plik do usunięcia. Usuwana jest zawartość oraz wpis ewidencyjny pliku.
- Dodatkowe operacje na plikach, wykonywane w celu uzyskania dostępu do zawartości pliku:
  - otwieranie,
  - zamykanie,
  - przesuwanie wskaźnika bieżącej pozycji.

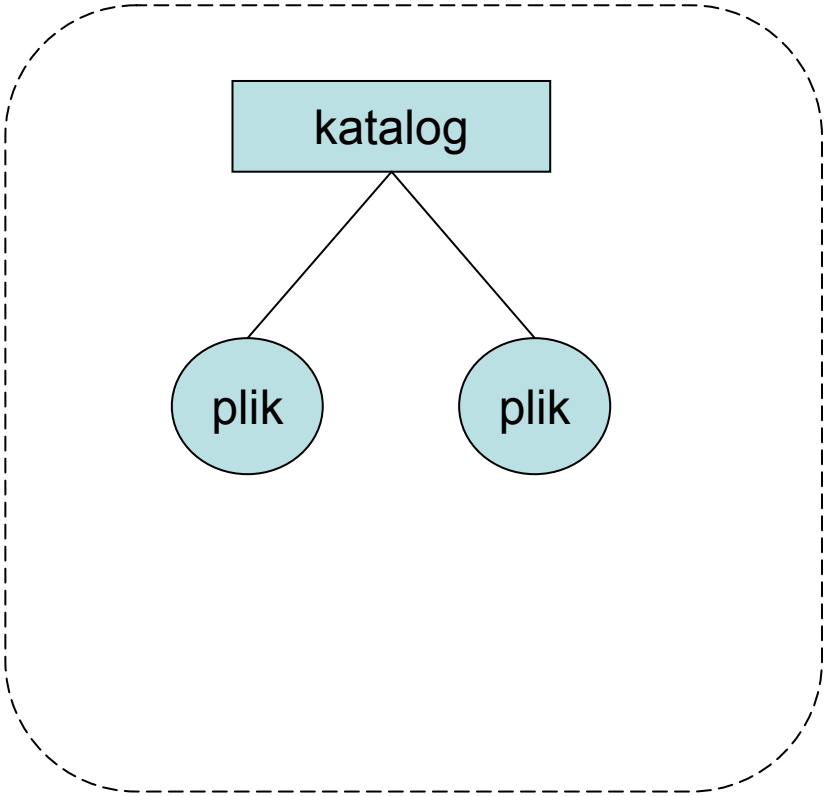
Organizacja logiczna systemu plików (1)

- Podział na strefy (wolumeny, woluminy, tomy, partycje)
  - strefa obejmuje część dysku, jeden lub kilka dysków,
  - strefa zawiera pliki i katalogi.
- Organizacja katalogów:
  - katalog jest tablicą kojarzącą nazwy plików z wpisami katalogowymi, obejmującymi inne atrybuty plików,
  - katalogi mogą być jedno- lub wielopoziomowe
  - katalogi wielopoziomowe zorganizowane mogą być w różne struktury logiczne (drzewo, graf acykliczny, dowolny graf).
- Pliki identyfikowane są przez nazwy, znajdujące się w katalogach.

Organizacja logiczna systemu plików (2)



strefa/partycja/wolumen



strefa/partycja/wolumen

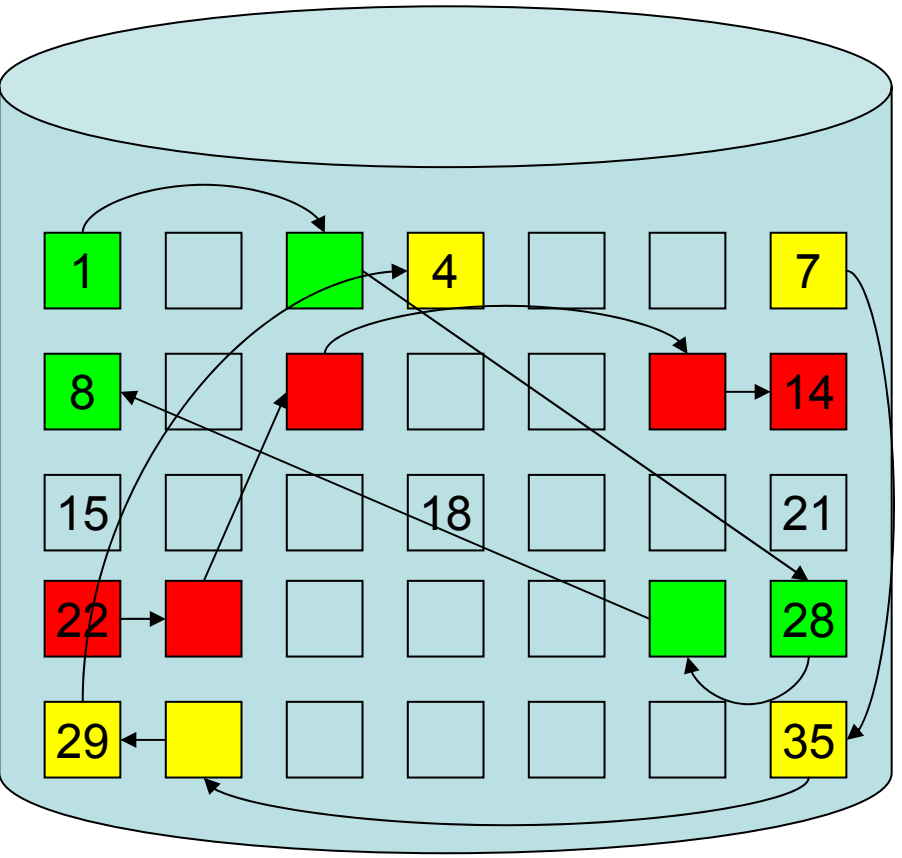
Operacje na katalogu

- Tworzenie katalogu
- Usuwanie katalogu
- Tworzenie wpisu katalogowego — gdy tworzony jest plik, jego nazwa alternatywna, podkatalog itp.
- Usuwanie wpisu katalogowego
- Przemianowanie pliku (zmiana nazwy)
- Odnajdowanie wpisu katalogowego
- Tworzenie wykazu wpisów katalogowych (listing zawartości)

Organizacja fizyczna systemu plików

- Przestrzeń dyskowa na potrzeby systemu plików zorganizowana jest w jednostki alokacji, zwane krótko blokami. Blok jest wielokrotnością sektora dysku.
- Reprezentatywnym przykładem organizacji fizycznej plików może być FAT (ang. file allocation table):
  - FAT jest dodatkową strukturą (tablicą) umieszczoną w odpowiednim obszarze na dysku
  - Każdy element tablicy FAT odpowiada dokładnie jednej jednostce alokacji (blokowi) z przestrzeni bloków plikowych i indeksowany jest numerem bloku
  - Element tablicy FAT zawiera indeks następnego bloku przydzielonego danemu plikowi lub pewną wartość specjalną oznaczającą wolną pozycję lub ostatnią pozycję danego pliku

Struktura tablicy alokacji plików



1	3
2	
3	28
4	#
5	
6	
7	35
8	#
…	
27	8
28	27

Katalog:

blok początkowy: 1
blok końcowy: 8

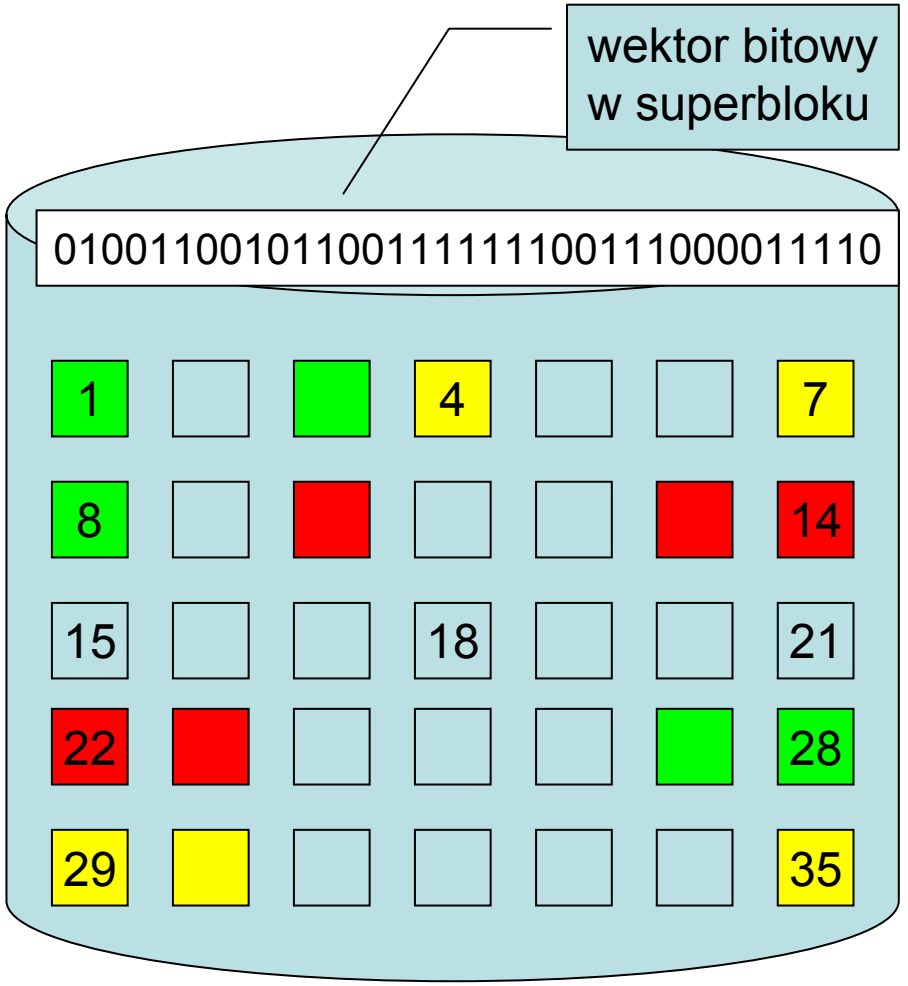
blok początkowy: 22
blok końcowy: 14

blok początkowy: 7
blok końcowy: 4

Zarządzanie wolną przestrzenią

- Wektor bitowy — każdy bit odpowiada jednemu blokowi, wartość 1 oznacza wolny blok.
- Lista powiązana — każdy wolny blok zawiera indeks następnego wolnego bloku.
- Grupowanie — niektóre wolne bloki zapełnione są w całości indeksami innych wolnych bloków, ostatni indeks wskazuje na kolejny blok zapełniony w całości indeksami.
- Zliczanie — wykaz wolnych bloków obejmuje indeks pierwszego wolnego bloku oraz liczbę wolnych bloków znajdujących się za nim, stanowiących ciągły obszar.

Zarządzanie wolną przestrzenią — wektor bitowy



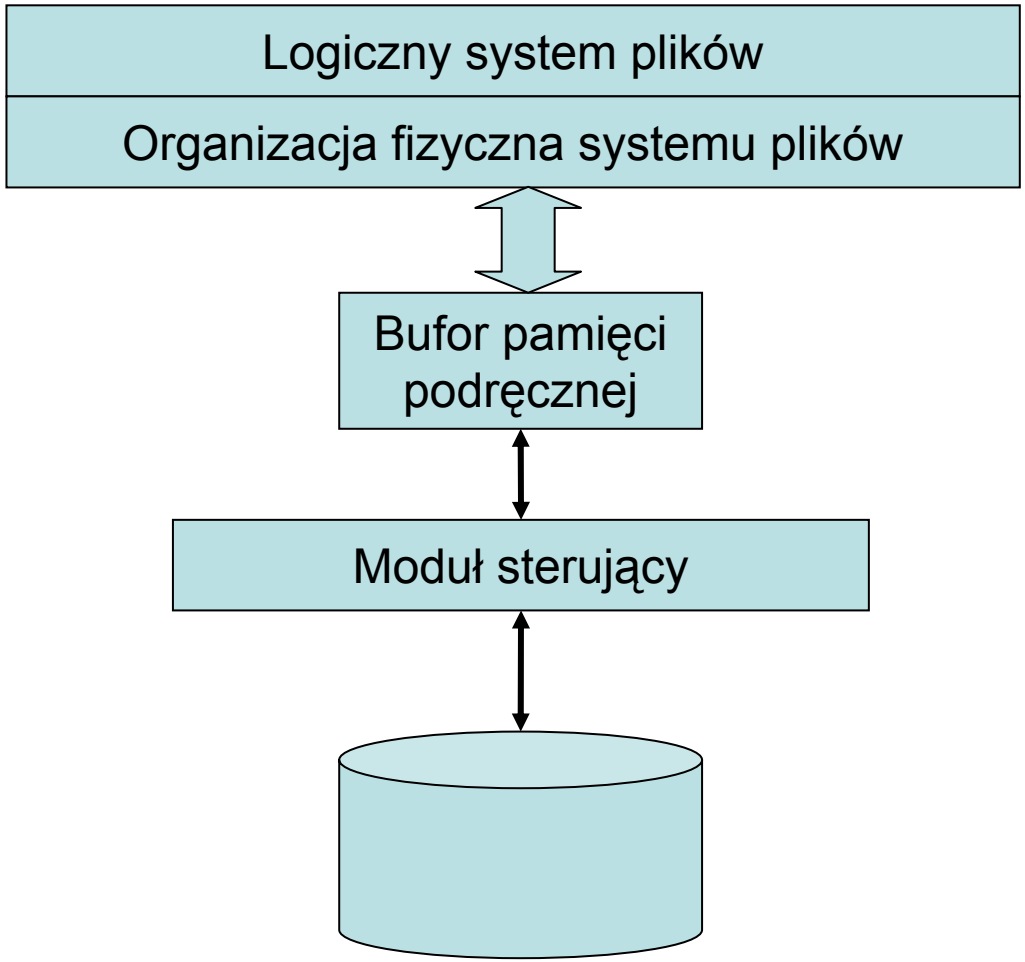
Przykład implementacji katalogu — lista liniowa

- Katalog składa się z ciągu wpisów katalogowych ogólnej postaci:

nazwa pliku	inne atrybuty

- Lokalizacja wpisu polega na przeszukiwaniu liniowym (sprawdzane są kolejne pozycje, począwszy od pierwszej)
- Lokalizacją wpisu można przyspieszyć poprzez posortowanie wg. nazwy, jednak utrzymanie takiej struktury jest kosztowne.

Operacja dostępu do danych w pliku



Zasady przechowywania podręcznego

- Zawartość aktualnie wykorzystywanych bloków dyskowych utrzymywana jest w podręcznej pamięci buforowej.
- Obsługa żądania odczytu bloku polega najpierw na sprawdzeniu czy dany blok znajduje się w podręcznej pamięci buforowej, a później ewentualnie sprowadzenia z dysku.
- Żądany fragment kopiowany jest z podręcznej pamięci buforowej w odpowiednie miejsce w przestrzeni adresowej procesu.
- Obsługa żądania zapisu oznacza transfer danych do podręcznej pamięci buforowej.

Współbieżność i synchronizacja procesów

Przetwarzanie współbieżne

- Dopóki system składa się ze zbioru niezależnych procesów, z których każdy odwołuje się do własnych zasobów system operacyjny tylko przydziela procesy do procesorów lub szereguje je do wykonania na danym procesorze.
- Gdy jednak procesy odwołują się do współdzielonych zasobów, lub w pewien sposób ze sobą kooperują wymagane jest, aby system operacyjny dostarczał odpowiednich mechanizmów na to pozwalających.
- Z problematyką komunikacji pomiędzy procesami związane są zagadnienia:
  - Synchronizacji procesów
  - Wzajemnego wykluczania dostępu do zasobów
  - Możliwości zakleszczenia procesów

Semafor

- **Semafor** jest nieujemną liczbą całkowitą, która poza inicjalizacją może być modyfikowana przez 2 procedury: P (lub WAIT) i V (lub SIGNAL).
- Procedura WAIT(S): Jeśli wartość S >0 to zmniejsz wartość S o jeden; w przeciwnym wypadku zatrzymaj proces do chwili, gdy S > 0 (i wtedy zmniejsz wartość S).
- Procedura SIGNAL(S): zwiększ wartość S o jeden.
- Procedury WAIT i SIGNAL są atomowe (traktowane jako pojedyncza nierozerwalna instrukcja). Dwa procesy wykonujące operację WAIT na tym samym semaforze nie mogą na siebie wpływać i nie mogą się załamać podczas wykonywania operacji na semaforze.

Synchronizacja procesów z zastosowaniem semafora

```
var consyn : semaphore (* init 0 *)
```

```
process P1;
   (* waiting process *)
   statement X;
   wait (consyn)
   statement Y;
end P1;
```

```
process P2;
   (* signalling proc *)
   statement A;
   signal (consyn)
   statement B;
end  P2;
```

Wzajemne wykluczanie z zastosowaniem semafora

```
(* mutual exclusion *)
var mutex : semaphore; (* initially 1 *)
```

```
process P1;
  statement X
  wait (mutex);
    statement Y
  signal (mutex);
  statement Z
end P1;
```

```
process P2;
  statement A;
  wait (mutex);
     statement B;
  signal (mutex);
  statement C;
end P2;
```

Zakleszczenie (1)

- Dwa procesy są zakleszczone, jeśli każdy z nich przechwycił zasób i oczekuje na drugi zasób przechwycony przez drugiego.

```
type Sem is ...;
X : Sem := 1; Y : Sem := 1;
```

```
task A;
task body A is
begin
      ...
      Wait(X);
      Wait(Y);
      ...
end A;
```

```
task B;
task body B is
begin
      ...
      Wait(Y);
      Wait(X);
      ...
end B;
```

Zakleszczenie (2)

a

-

-

Wait(s_Q)

Wait(s_V)

Signal(s_V)

Signal(s_Q)

b

-

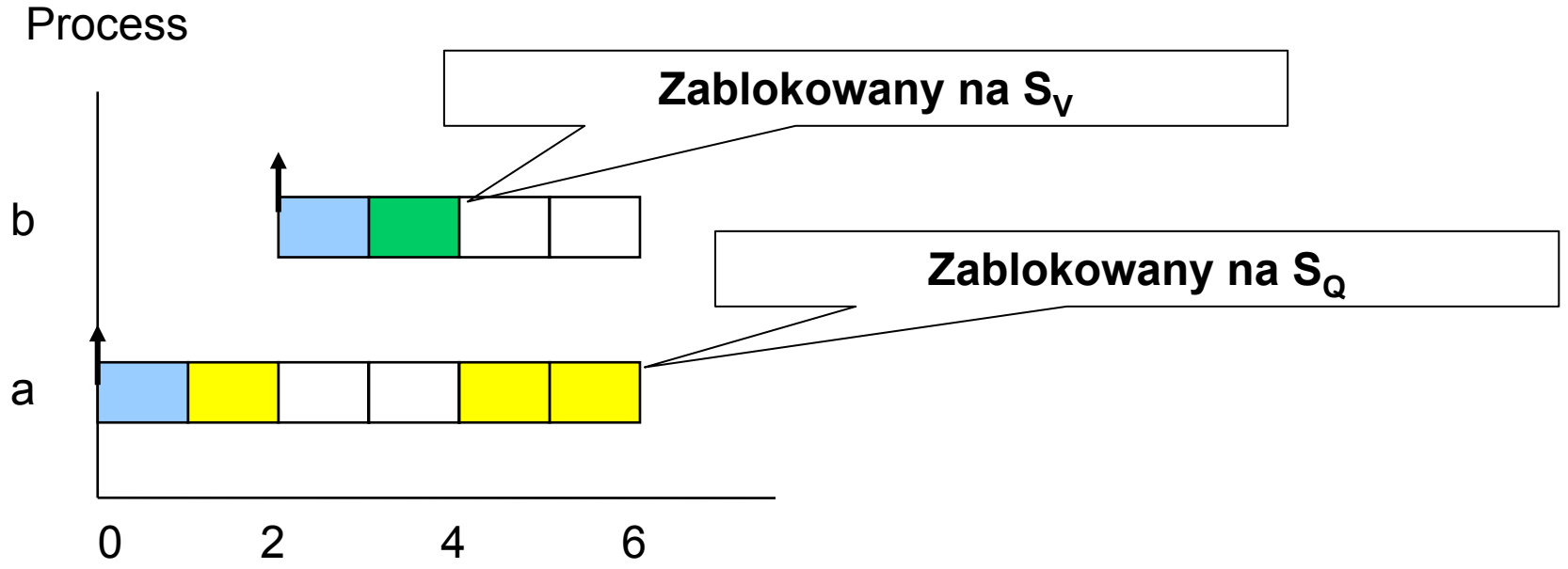
-

Wait(s_V)

Wait(s_Q)

Signal(s_Q)

Signal(s_V)



Ograniczone buforowanie

```
type Sem is ...;
Pusty : Sem := n; Pełny : Sem := 0; Mutex : Sem := 1;
```

```
{Producent}
task A;
task body A is
begin
repeat
      produkuj
      Wait(Pusty);
      Wait(Mutex);
      dodaj do bufora
      Signal(Mutex);
      Signal(Pełny);
until false;
end A;
```

```
{Konsument}
task B;
task body B is
Begin
repeat
      Wait(Pełny);
      Wait(Mutex);
      odbierz z bufora
      Signal(Mutex);
      Signal(Pusty);
      konsumuj
until false;
end B;
```

Czytelnicy i pisarze

```
type Sem is ...;
Pis : Sem := 1; Mutex : Sem := 1;
liczba_czyt : Integer := 0;
```

```
{Pisarz}
Wait(Pis);
Pisz
Signal(Pis);
```

```
{Czytelnik}
Wait(Mutex);
     liczba_czyt := liczba_czyt + 1;
     if liczba_czyt = 1
          then Wait(Pis);
Signal(Mutex);
     Czytaj
 Wait(Mutex);
     liczba_czyt := liczba_czyt - 1;
     if liczba_czyt = 0
          then Signal(Pis);
 Signal(Mutex);
```

Pięciu ucztujących filozofów