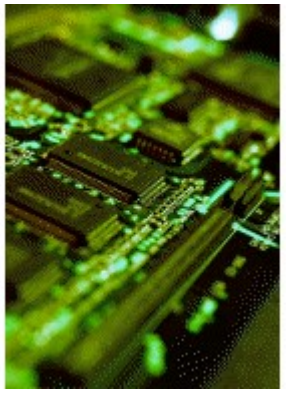
# Systemy Operacyjne i Sieci Komputerowe



Sprzęt komputerowy

System Operacyjny
+
Programy

Łatwe użytkowanie

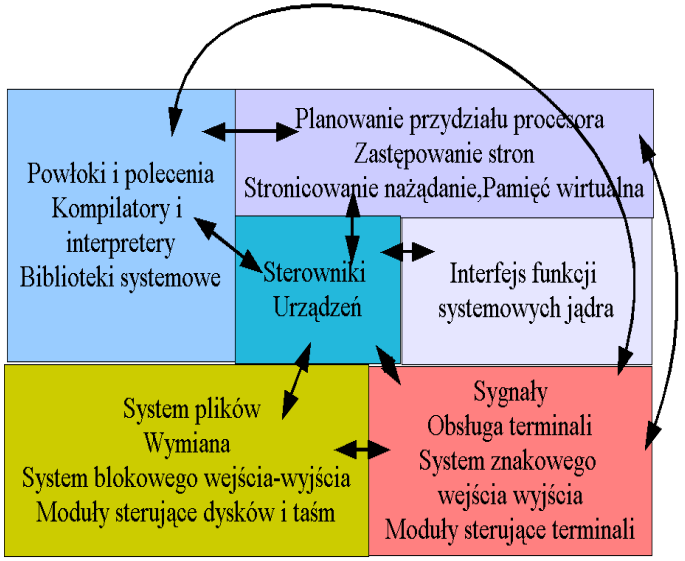
*Prowadzący:* *Robert Szmurło*
*szmurlor@iem.pw.edu.pl*
*GE 229*

# Systemy operacyjne od strony architekta

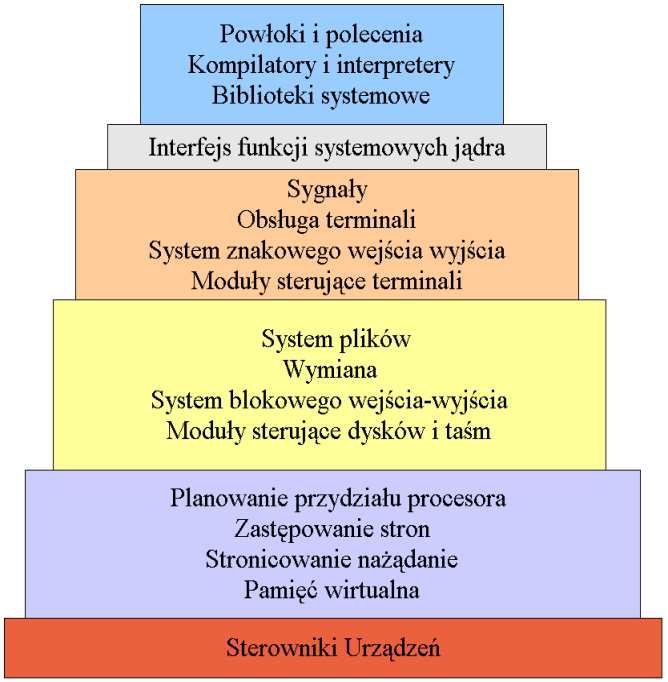
- Architektury systemów operacyjnych
- Urządzenia wejścia-wyjścia
- Zarządzanie pamięcią
- Zarządzanie procesami
- Wątki

# Architektury SO

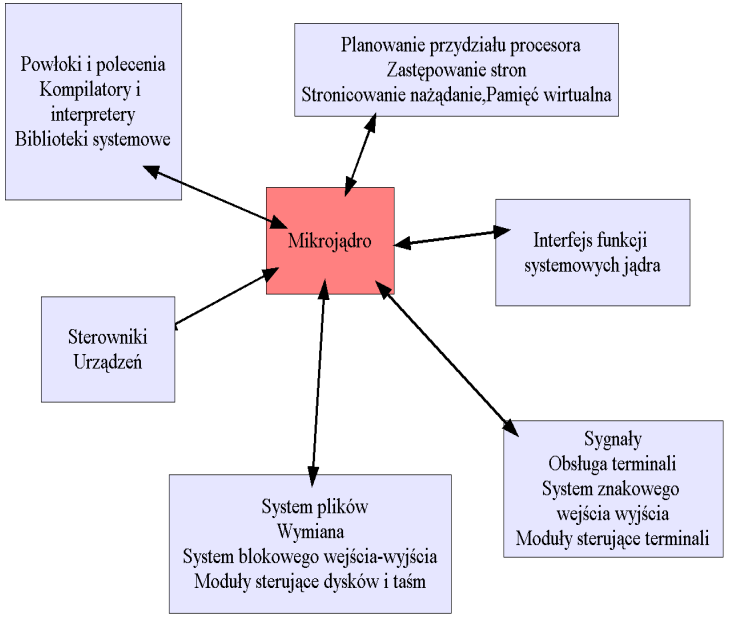
- Jednolita



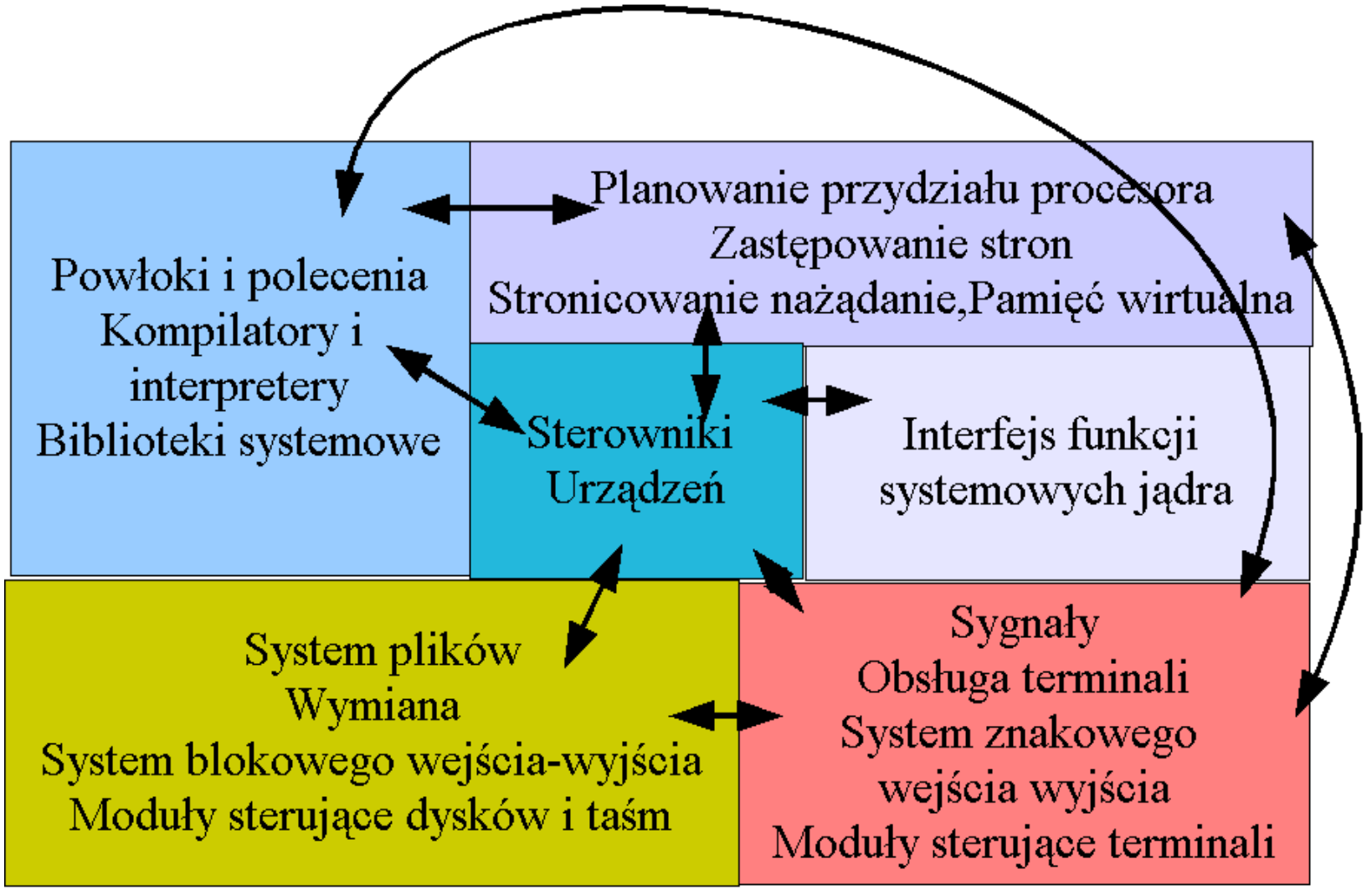

- Warstwowa



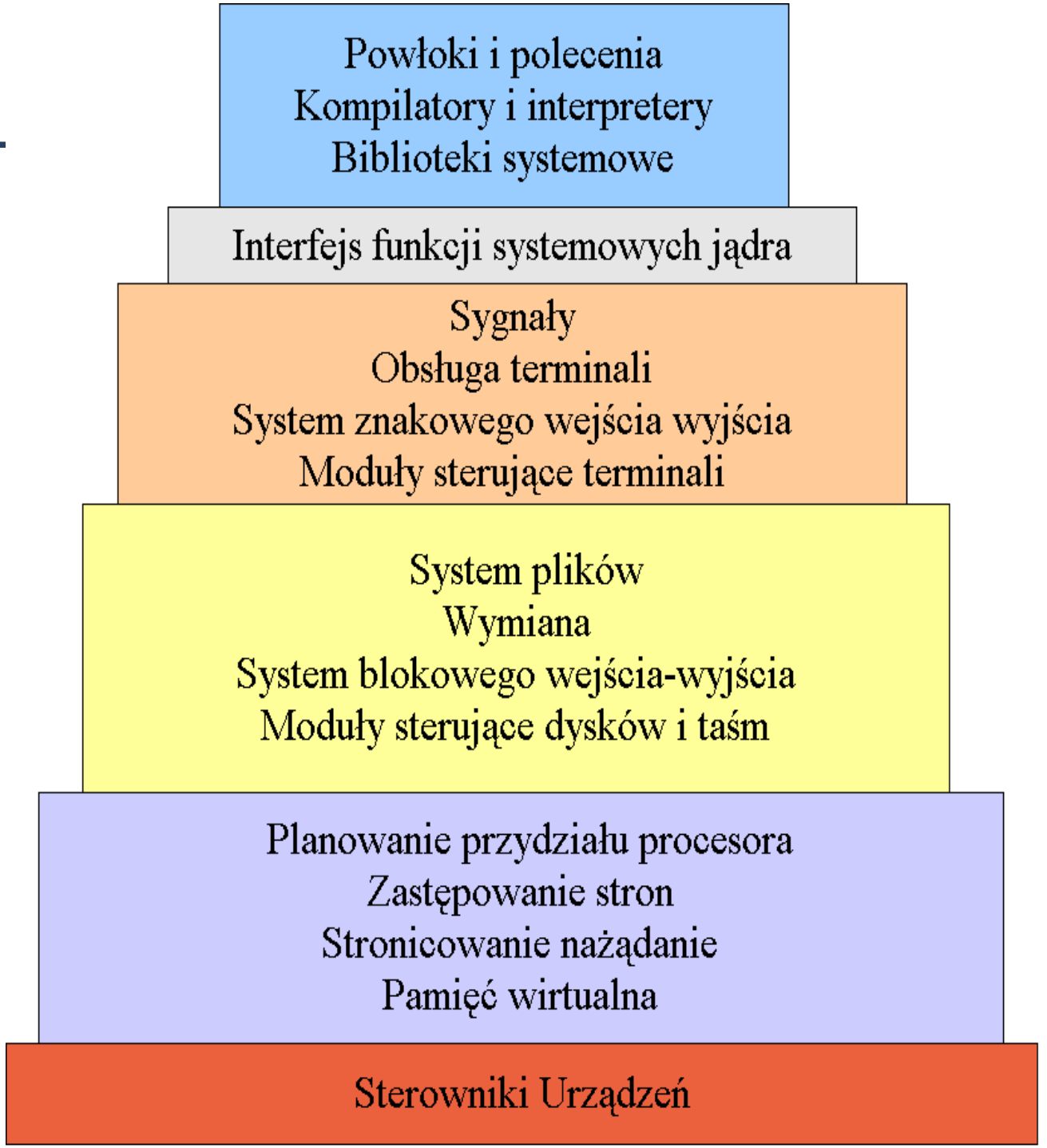

- Klient-Serwer



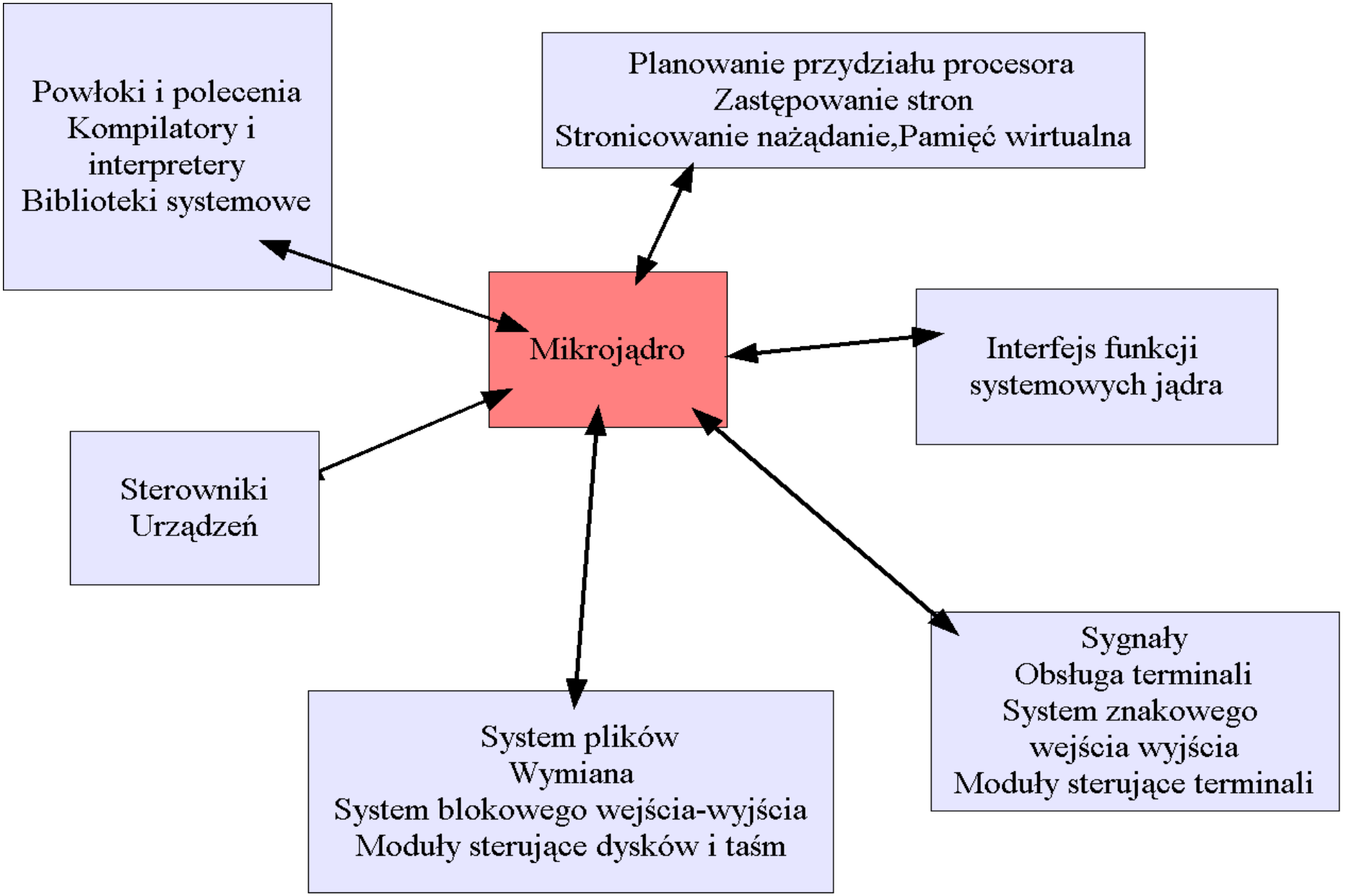
# Jednolita


Powłoki i polecenia
Kompilatory i
interpretery
Biblioteki systemowe
Planowanie przydziału procesora
Zastępowanie stron
Stronicowanie nażądanie,Pamięć wirtualna
Sterowniki
Urządzeń
Interfejs funkcji
systemowych jądra
System plików
Wymiana
System blokowego wejścia-wyjścia
Moduły sterujące dysków i taśm
Sygnały
Obsługa terminali
System znakowego
wejścia wyjścia
Moduły sterujące terminali

# Warstwowa


Powłoki i polecenia
Kompilatory i interpretery
Biblioteki systemowe
Interfejs funkcji systemowych jądra
Sygnały
Obsługa terminali
System znakowego wejścia wyjścia
Moduły sterujące terminali
System plików
Wymiana
System blokowego wejścia-wyjścia
Moduły sterujące dysków i taśm
Planowanie przydziału procesora
Zastępowanie stron
Stronicowanie nażądanie
Pamięć wirtualna
Sterowniki Urządzeń

# Klient-serwer



# Urządzenia wejścia-wyjścia

- urządzenia pamięci (dyski, taśmy, pamięci trwałe)
- urządzenia przesyłania danych (karty sieciowe, modemy)
- urządzenia interfejsu człowieka (monitory, klawiatury, myszki, drukarki)
- urządzenia specjalne (dane pomiarowe, sterowanie robotami)
- urządzenia multimedialne (karty dźwiękowe, mikrofony, karty TV, kamery)

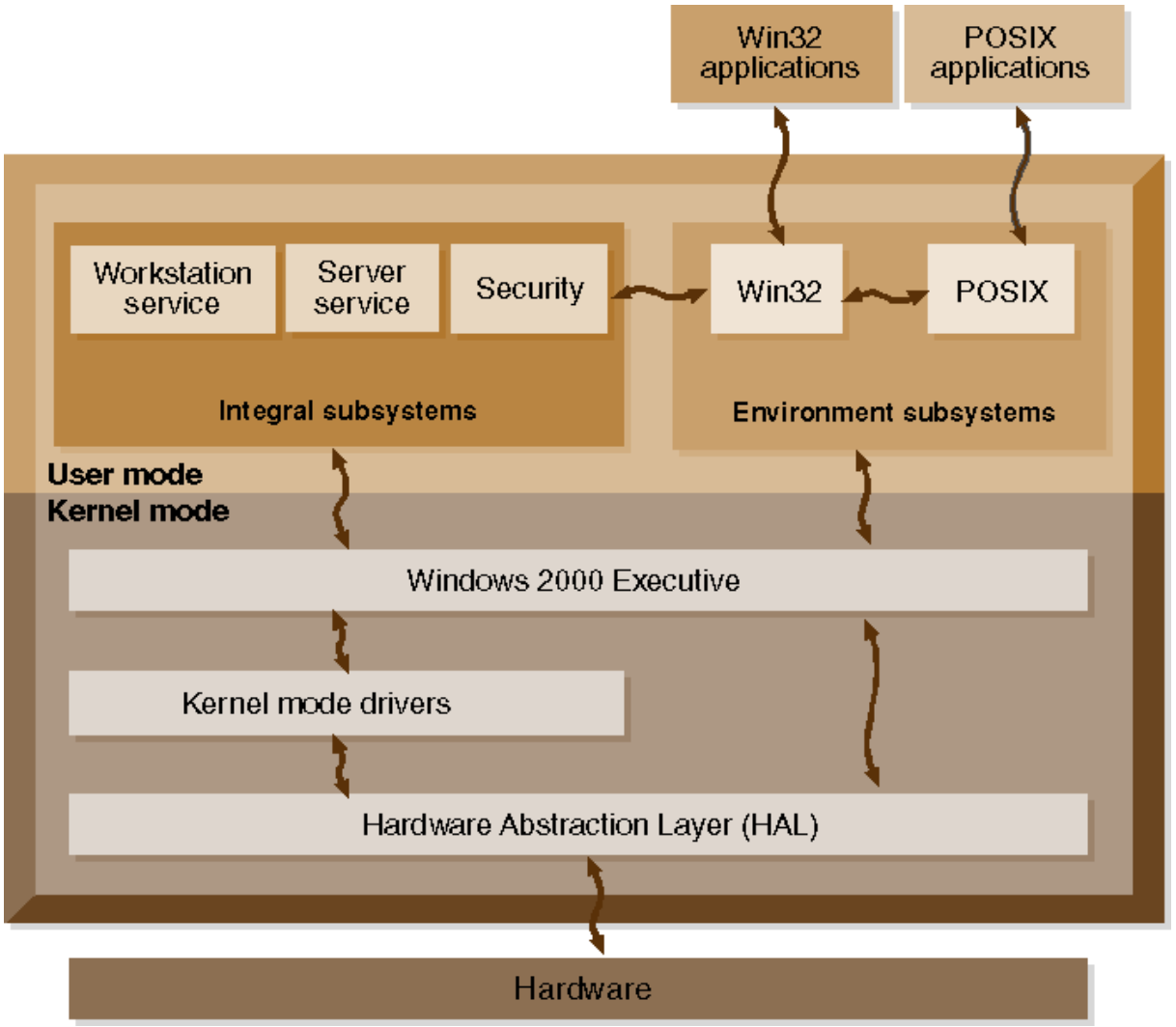
# Właściwości urządzeń We/Wy

- Tryb przesyłania danych:
  - znakowy – przesyłane jest po kolei bajt po bajcie (przykład klawiatura, często występuje bufor)
  - blokowy – przesyłane całe bloki danych (przykład sektor dysku twardego, ramka komuniatu w karcie sieciowej)
- Sposób dostępu do danych:
  - sekwencyjny – dostęp do danych w określonej kolejności, aby odczytać ostatni bajt musimy wczytać cały strumień danych (przykład taśma z backupem)
  - swobodny – mamy dostęp do wolnej informacji poprzez pozycję w strumieniu danych (przykład: sektor dysku twardego)
- Organizacja przesyłania:
  - asynchroniczna – dane przekazywane są z góry znanym czasie
  - synchroniczna – dane moga pojawić się w dowolnym momencie.

# Właściwości urządzeń We/Wy

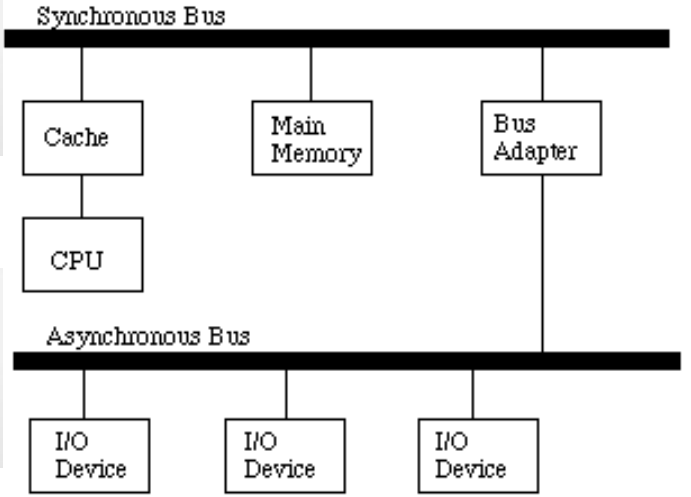
- Tryb użytkowania:
  - współdzielony — dopuszczalne jest współbieżne używanie urządzenia przez wiele procesów, np.: dysk
  - wyłączny — niemożliwe jest współbieżne używanie urządzenia przez wiele procesów, przykład: drukarka
- Kierunek przesyłania danych:
  - Tylko do odczytu
  - Tylko do zapisu
  - Do zapisu i odczytu

# Schemat komunikacji z urzadzeniami

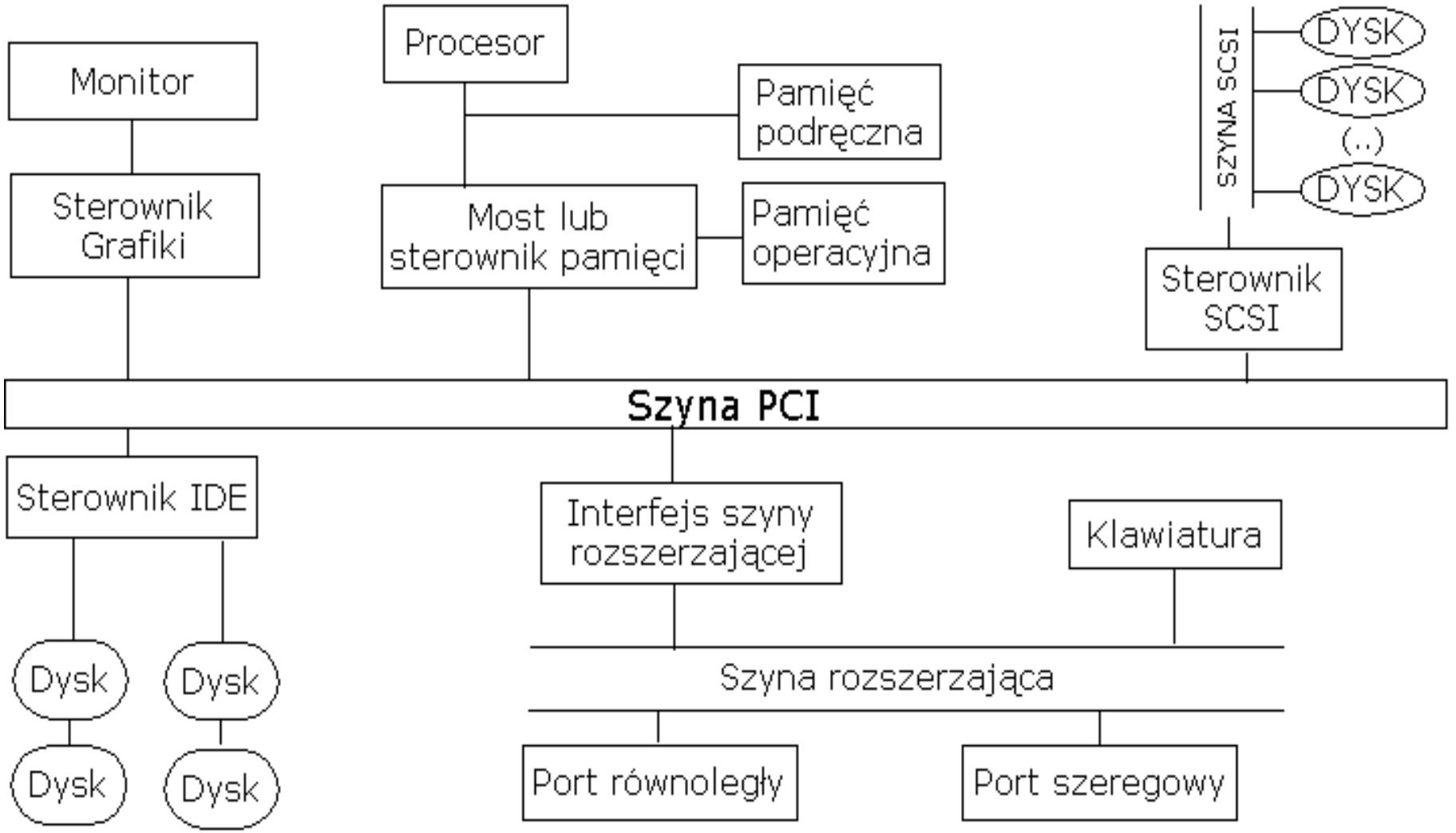


# Punkty komunikacyjne / pośredniczące

- Czyli to co łączy system operacyjny z urządzeniami.
- Port - jedno urządzenie wykorzystuje fizyczną wiązkę przewodów
  - (FireWire, USB, RS 232 – port szeregowy, LPT – port równoległy, BlueTooth)
- Magistrala (szyna komunikacyjna)
  - do jednej wiązki przewodów dołączonych jest kilka urzadzeń.
  - skomplikowana praca
  - wymaga specjalnego układu zarządzającego komunikacją



# Szyna komunikacyjna



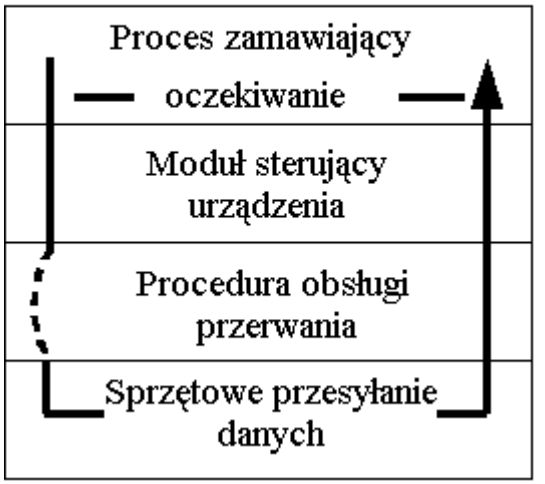
# Typy szyn komunikacyjnych

- Szyna komunikacyjna ISA lub AT BUS - (ang. Industry Standard Architecture) Prędkość: 1.5-1.8 MB/s (MB - megabity), Słowo: 16bitów, Zegar: 16MHz
- Szyna PCI (ang. Peripheral Component Interconnect - Wprowadzona w 1993r), Prędkość: 133 MB/s, Słowo: 32bity, Zegar 33MHz, Zasilanie,
- AGP (ang. Accelerated Graphics Port) - Interfejs umożliwia karcie graficznej używać wyodrębnionego obszaru pamięci operacyjnej w takiej postaci jakby karta graficzna korzystała z niego jak z pamięci podręcznej.
  - AGP - przepustowość: 266 MB/s
  - AGP 2 - przepustowość: 533 MB/s
  - AGP 4 - przepustowość: 1066 MB/s

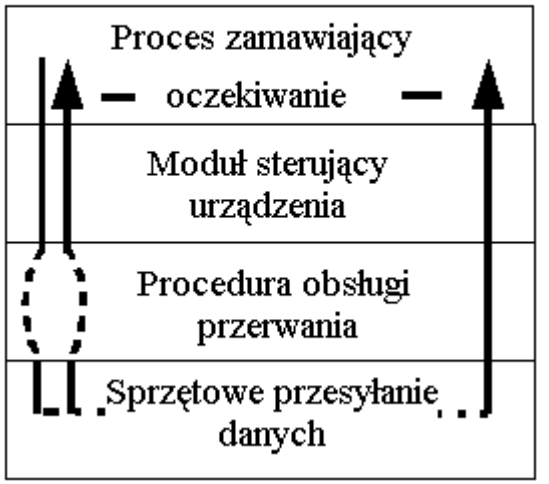
# Struktura We/Wy

- Synchroniczny
  - Po rozpoczęciu procedury We/Wy program użytkownika odzyskuje kontrolę po zakończeniu wymiany informacji We/Wy.
- Asynchroniczny
  - Po rozpoczęciu procedury We/Wy program uzytkownika odzyskuje natychmiast kontrolę.
  - System komputerowy musi posiadać tablicę urządzeń, która będzie przechowywac aktualny stan (idle-wolny lub busy-zajęty).

Synchroniczny




Asynchroniczny

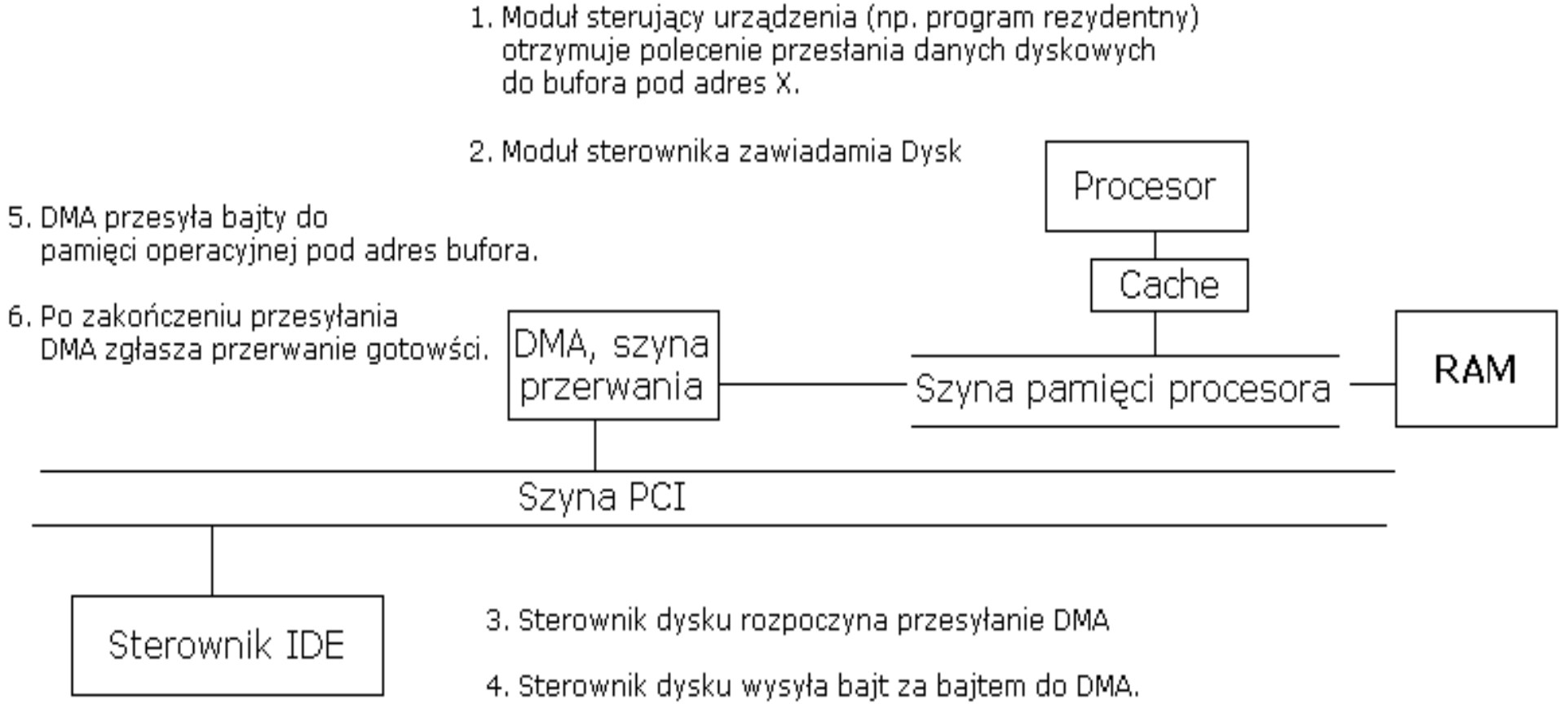


# Jak przebiega komunikacja systemu operacyjnego z urządzeniami?

- Komunikacja za pomocą portów
  - rejestry sterownika (porty) widoczne są w przestrzeni adresowej wejścia-wyjścia systemu komputerowego i dostępne są przez specjalne rozkazy (np. in i out w procesorach firmy Intel). (Dla magistral ISA 16 bitów) Komunikacja tylko za pośrednictwem procesora!
  - odwzorowanie w przestrzeni adresowej pamięci — rejestry sterownika widoczne są w przestrzeni adresowej pamięci fizycznej i dostępne są pod odpowiednimi adresami tak samo, jak inne komórki pamięci.
- Bezpośredni dostęp do pamięci (DMA – Direct Memory Access)
  - Używane dla urządzeń o dużej prędkości transferu danych (prędkość zbliżona do prędkości dostępu do pamięci)
  - Kontroler urządzenia po inicjalizacji wykonanej przez procesor , samodzielnie wykonuje transfer danych z bufora bezpośrednio do pamięci.
  - Przerwanie jest generowane co przesłany blok danych a nie co każdy bajt.

# Bezpośredni dostęp do pamięci: DMA

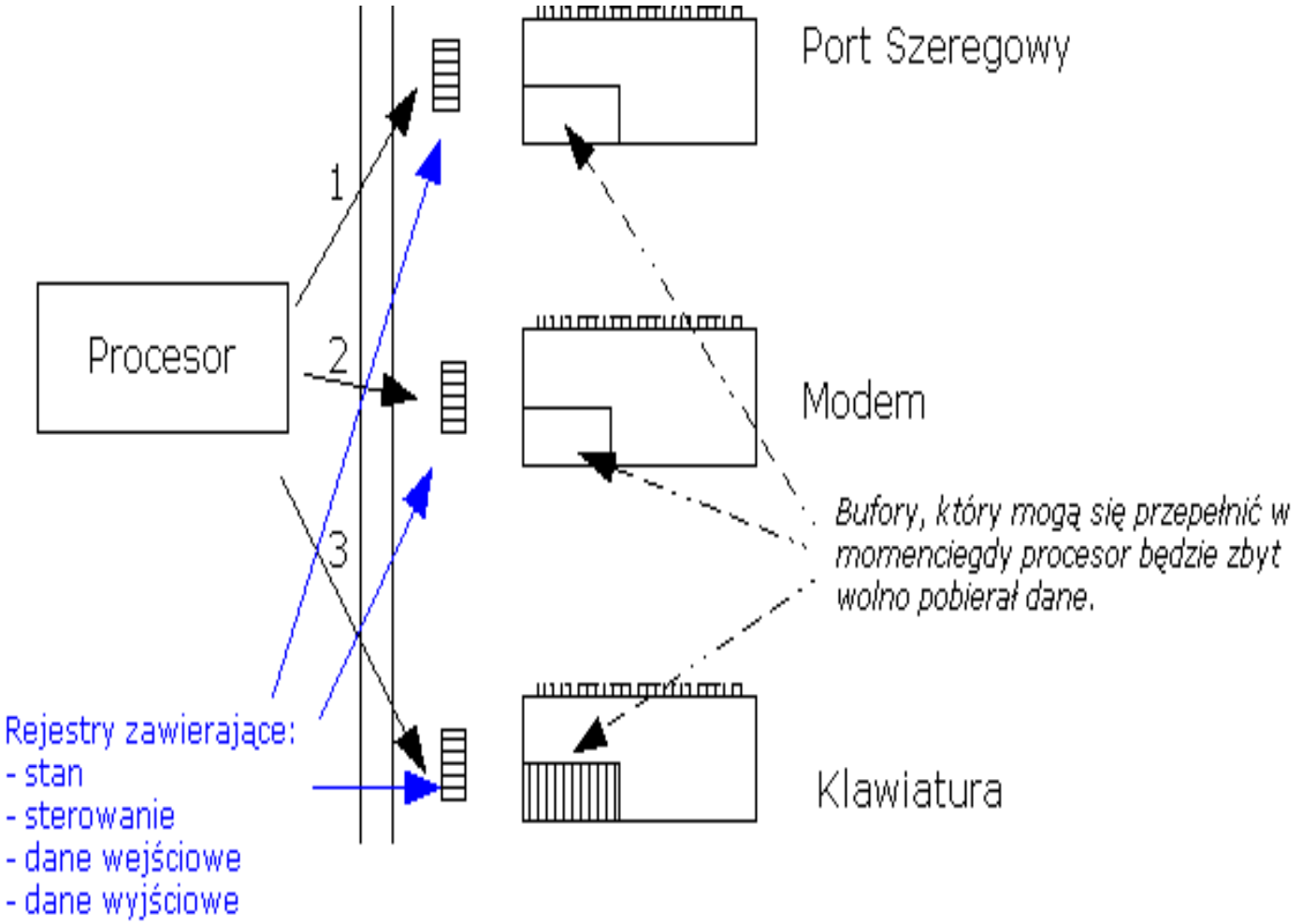
- źródło przesyłania (sterownik urządzenia czy bufor w pamięci)
- wskaźnik do miejsca w pamięci (bufor)
- liczba bajtów do przesłania



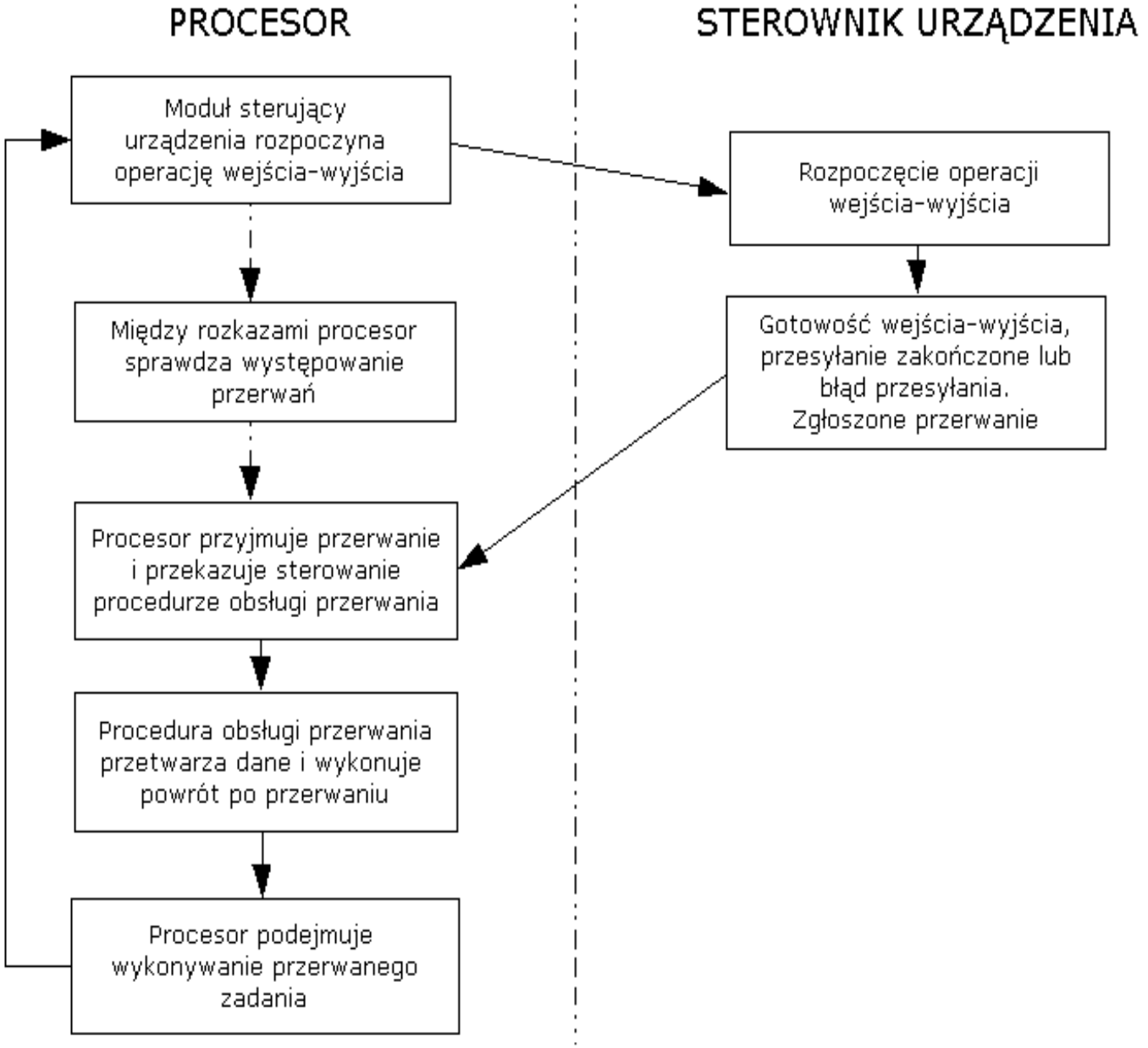
# Tryby sprawdzania gotowości danych do odczytu-zapisu

- Poprzez odpytywanie (polling) - które polega na tym że procesor w odstępach czasowych sprawdza odpowiednie bity w rejestrze każdego sterownika.
  - metoda mało wydajna, obciążenie procesora
  - bufor sterownika może się zapełnić,
  - mało wydajny algorytm gdy odpytywanie musi być realizowane zbyt często.
- Przerwania - w momencie urządzenie posiada dane, do odczytania przez procesor zgłasza do niego przerwanie.
  - przerwania niemaskowalne
  - przerwania maskowalne (używane przez urządzenia do zgłaszania żądań obsługi)

# Odpytywanie



# Przerwania


PROCESOR
STEROWNIK URZĄDZENIA
Moduł sterujący urządzenia rozpoczyna operację wejścia-wyjścia
Między rozkazami procesor sprawdza występowanie przerwań
Procesor przyjmuje przerwanie i przekazuje sterowanie procedurze obsługi przerwania
Procedura obsługi przerwania przetwarza dane i wykonuje powrót po przerwaniu
Procesor podejmuje wykonywanie przerwanego zadania
Rozpoczęcie operacji wejścia-wyjścia
Gotowość wejścia-wyjścia, przesyłanie zakończone lub błąd przesyłania. Zgłoszone przerwanie

# Przerwania

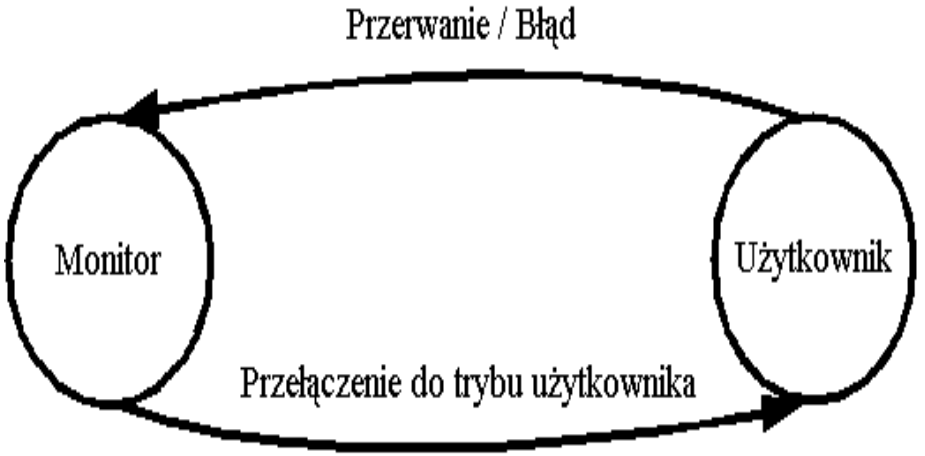
- Przerwanie przekazuje sterowanie do procedury obsługi przerwania, zazwyczaj poprzez tablicę wektorów przerwań.
- Architektura przerwań musi zachować adres procedury w którym nastąpiło przerwanie.
- Przerwania są zablokowane podczas wykonywania procedury obsługi innego przerwania (przerwania niemaskowalne).
- Wyjątek jest przerwaniem wygenerowanym przez aplikację na skutek błędu lub życzenia użytkownika.
- System operacyjny  jest sterowalny za pomocą przerwań.
- Schemat obsługi przerwań (zapamiętanie stanu CPU, typ przerwania: odpytywanie lub wektor, wykonanie procedury)

# Klasy przerwań

- Programowe – przerwania generowane przez nieprawidłową instrukcję, np. Przepełnienie wartości zmiennej, dzielenie przez zero, odwołanie się do zabronionego fragmentu pamięci operacyjnej.
- Timer (zegar) – generowane w określonych odstępach czasu (zazwyczaj co milisekundę) przez procesor. Pozwala systemowi oper. W określonych odstępach czasowych przejąć kontrolę nad procesorem.
- I/O (We/Wy) – Generowane przez kontroler wejścia/wyjścia, aby zasygnalizować zagończenie jakiejś operacjy przesyłania danych, lub zakomunikować błąd.
- Awaria sprzętowa – Generowane zazwyczaj przez 'chipset' płyty głównej aby zkomunikować niekorygowalny błąd sprzętowy (np. awaria zasilania, lub błąd parzystości pamięci).

# Ochrona sprzętowa i dualny tryb operacji

- Współdzielenie zasobów wymaga od systemu operacyjnego zapewnienia, że nieprawidłowo działający program nie spowoduje błędów w innym programie.
- Sprzęt (procesor) musi udostępniać co najmniej dwa podstawowe tryby pracy:
  - Tryb użytkownika (user mode)
  - Tryb nadzorcy, monitora (supervisor mode)
  - Sprzęt posiada przełącznik bitowy: monitor (1), użytkownik (0).
  - Instrukcje uprzywilejowane mogą być uruchamiane tylko w trybie monitora (nadzorcy).



# Struktura pamięci

- Rejestry – w procesorze.
- Cache – pamięć podręczna
- Pamięć główna – jedyny typ pamięci do której procesor ma bezpośredni dostęp.
- Dyski magnetyczne – stanowią pamięć pomocniczą.
- Taśmy magnetyczne, dyski optyczne, itp.

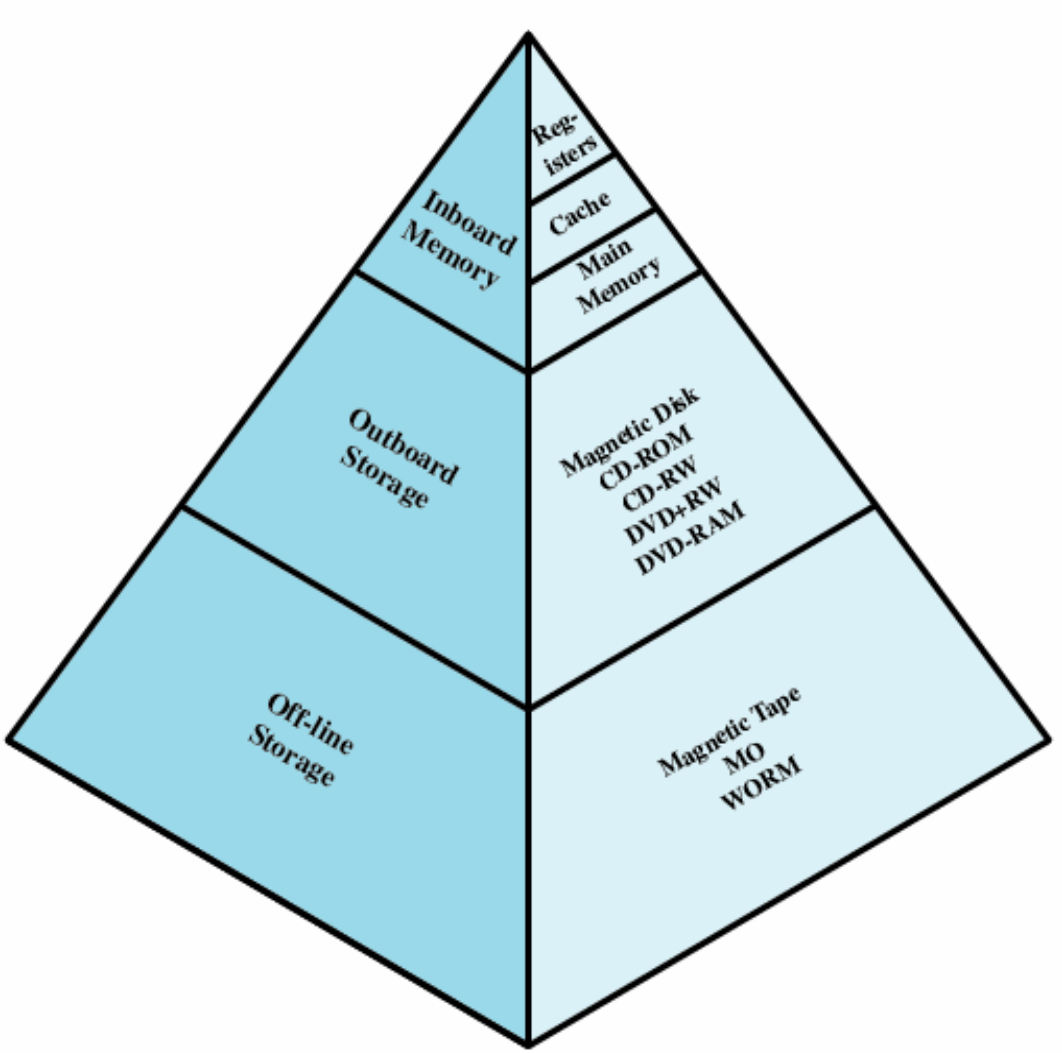
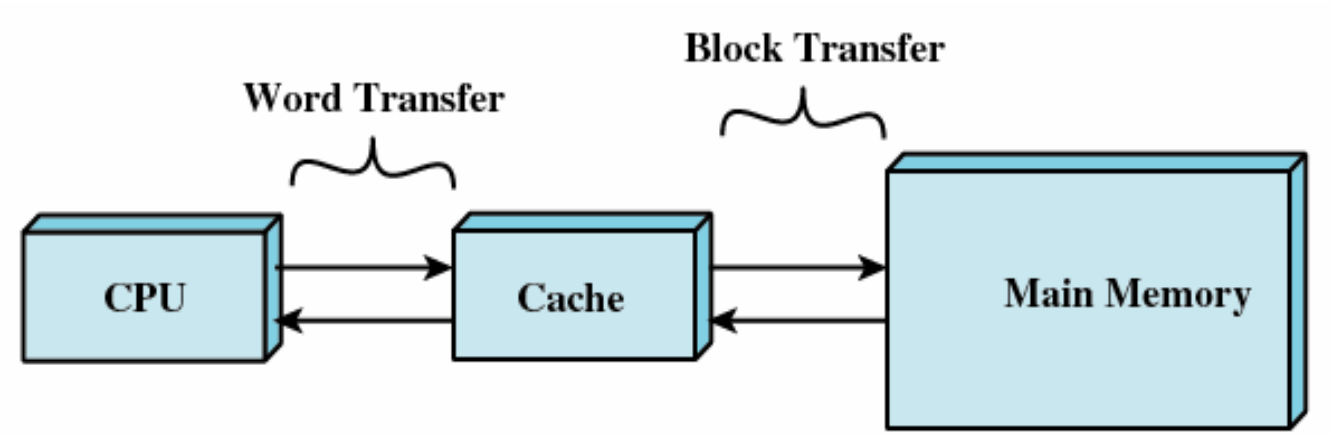



**Figure 1.14 The Memory Hierarchy**

# Pamięć podręczna (cache)

- Pamięć o dużej szybkości.
- Zawiera kopię fragmentu pamięci okryginalnej.
- Wymaga zarządzania pamięcią podręczną (staranne zarządzanie pamięcią przez system może pozwolić na to 80-90% odwołań do pamięci będzie w pamięci podręcznej)



# Przestrzeń adresowa

- Przestrzeń adresowa jest to zbiór wszystkich dopuszczalnych adresów w pamięci.
- W zależności od charakteru adresu definiuje się:
  - przestrzeń fizyczną — zbiór adresów przekazywanych do układów pamięci głównej (fizycznej).
  - przestrzeń logiczną — zbiór adresów generowanych przez procesor w kontekście aktualnie wykonywanego procesu.
- Wirtualna przetrzeń adresowa procesu – Z punktu widzenia procesu ma on do dyspozycji pewien wirtualny obszar pamięci, którego adresy nie mają nic wspólnego z fizycznymi adresami pamięci fizycznej. Jest to właśnie przestrzeń logiczna.

# Wsparcie sprzętowe

- Segmentacja: Obszar pamięci jest oczywiście odwzorowany w pamięci fizycznej, ale w zupełnie innej postaci niż widzi go proces użytkownika. Ciągły obszar pamięci procesu użytkownika może być w rzeczywistości podzielony pomiędzy różne strony pamięci fizycznej. Każdy proces posiada dwa zasadnicze segmenty wirtualnej pamięci:
  - segment jądra (KERNEL_DS) - dostęp do tego obszaru mają jedynie procedury systemowe
  - segment użytkownika (USER_DS) - dostęp do tego obszaru w sposób swobodny posiada proces użytkownika
- Adres wirtualny procesu użytkownika jest przekształcany na adres fizyczny przez procesor lub specjalny układ elektroniczny: MMU - Memory Management Unit.
- Do zapamiętania struktury oraz adresów fizycznych wirtualnych pamięci procesów służy tablica stron, która przechowuje: adresy stron, użytkownika w pamięci fizycznej, flagi określające tryby dostępu do danej strony pamięci, stany strony pamięci, atrybuty "obecności" - wskazuje czy dana strona znajduje się pamięci fizycznej czy jest aktualnie zapamiętana w pliku wymiany lub na partycji wymiany.

# Wsparcie sprzętowe

- Stronicowanie - pamięć wirtualna jest podzielona na równe kawałki zwane stronami (zazwyczaj 4Kb). Stronicowanie polega na zapamiętaniu pewnych rzadko używanych stron pamięci operacyjnej w specjalnym pliku na dysku twardym (dysk wymiany) lub na specjalnej partycji wymiany.
- Wymiatanie polega na usuwaniu pamięci fizycznej procesów, które aktualnie są zawieszone lub przez długi czas oczekują na jakieś zdarzenia.
- Z obydwu technologii, stronicowanie jest korzystniejsze jeśli chodzi o zarządzanie zasobami pamięci operacyjej, natomiast wymaga ono większych nakładów od systemu operacyjnego do zarządzania tym procesem. Wymiatanie w wielu przypadkach może nie zadziałać, natomiast wymaga zdecydowanie mniejszych nakładów procesora. Współczesne systemy operacyjne uzywają jedynie technologii stronicowania.

# Procesy

- **Proces jest elementarną jednostką pracy (aktywności) zarządzaną przez system operacyjny, która ubiega się o zasoby systemu komputerowego w celu wykonania programu.**
- Proces - program w trakcie wykonywania, który do wykonania określonego zadania potrzebuje pewnych zasobów: procesor, pamięć, pliki, urządzenia wejścia-wyjścia (klawiatura,ekran,skaner,karta sieciowa,port szeregowy lub równoległy itp.)
- Synonimami procesu, które są stosowane w literaturze są: praca (job) lub zadanie (task).
  - Zadanie – odnosi sie żazwyczaj do systemów wsadowych, w danej chwili może być wykonywane tylko jedno,
  - Praca - systemy z podziałem czasu (czas wykorzystania zasobów w tym procesora) jest dzielony na wiele prac (multitasking).

# Elementy składowe procesu

- program — definiuje zachowanie procesu, („*Proces jest czymś więcej niż samym kodem programu (sekcją tekstu - text section)*“).
- dane — zbiór wartości przetwarzanych oraz wyniki,
  - Stos procesu (przechowuje dane tymczasowe).
  - Sterta - Sekcja danych (zawiera zmienne globalne).
- zbiór zasobów tworzących środowisko wykonawcze, (np. zawartość rejestrów procesora.)
- blok kontrolny procesu (PCB, deskryptor) — opis
- bieżąca czynność reprezentowana przez wartość licznika rozkazów. (rejestr PC – program counter).

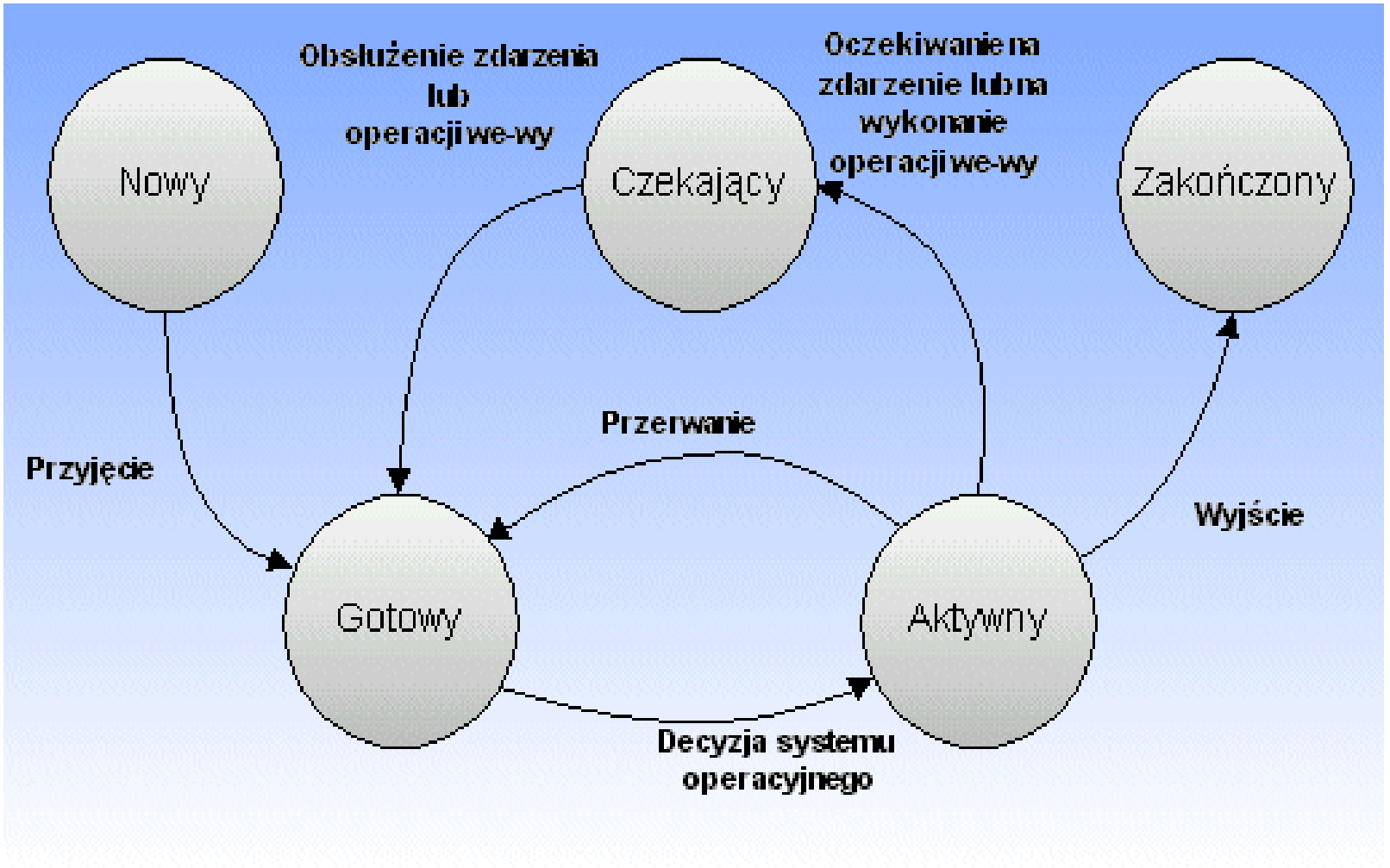
- ***Program jest obiektem pasywnym, natomiast proces jest obiektem akywnym.***

# Blok kontrolny procesu

- Struktura przechowująca informację o procesie.
- Blok kontrolny procesu przechowuje następujące informacje:
  - Stan procesu
  - Licznik rozkazów (pozycja aktualnie wykonywanej instrukcji)
  - Rejestry procesora (akumulatory,rejestry indeksowe,wskaźniki stosu)
  - Informacje o planowaniu przydziału procesora (np. priorytet procesu)
  - Informacje o zarządzaniu pamięcią (rejestry graniczne, tablice stron, lub tablice segmentów)
  - Informacje do rozliczeń (ilość zużytego procesora i czasu rzeczywistego, ograniczenia czasowe,numery kont,numery zadań,numery procesów)
  - Informacje o stanie wejścia-wyjścia

| Wskaźnik | Stan procesu |
|---|---|
| Numer procesu (PID) | |
| Licznik rozkazów (PC) | |
| Rejestry | |
| Ograniczenia pamięci | |
| Wykaz otwartych plików | |
| … … … | |

# Stan procesu



# Operacje na procesach

- załadowanie,wykonanie
- zakończenie,zaniechanie
- utworzenie procesu,zakończenie
- pobranie atrybutów, określenie atrybutów,
- czekanie czasowe, (np. CTRL+D – czyli zawieszenie)
- oczekiwanie na zdarzenie,
- przydział i zwolnienie pamięci

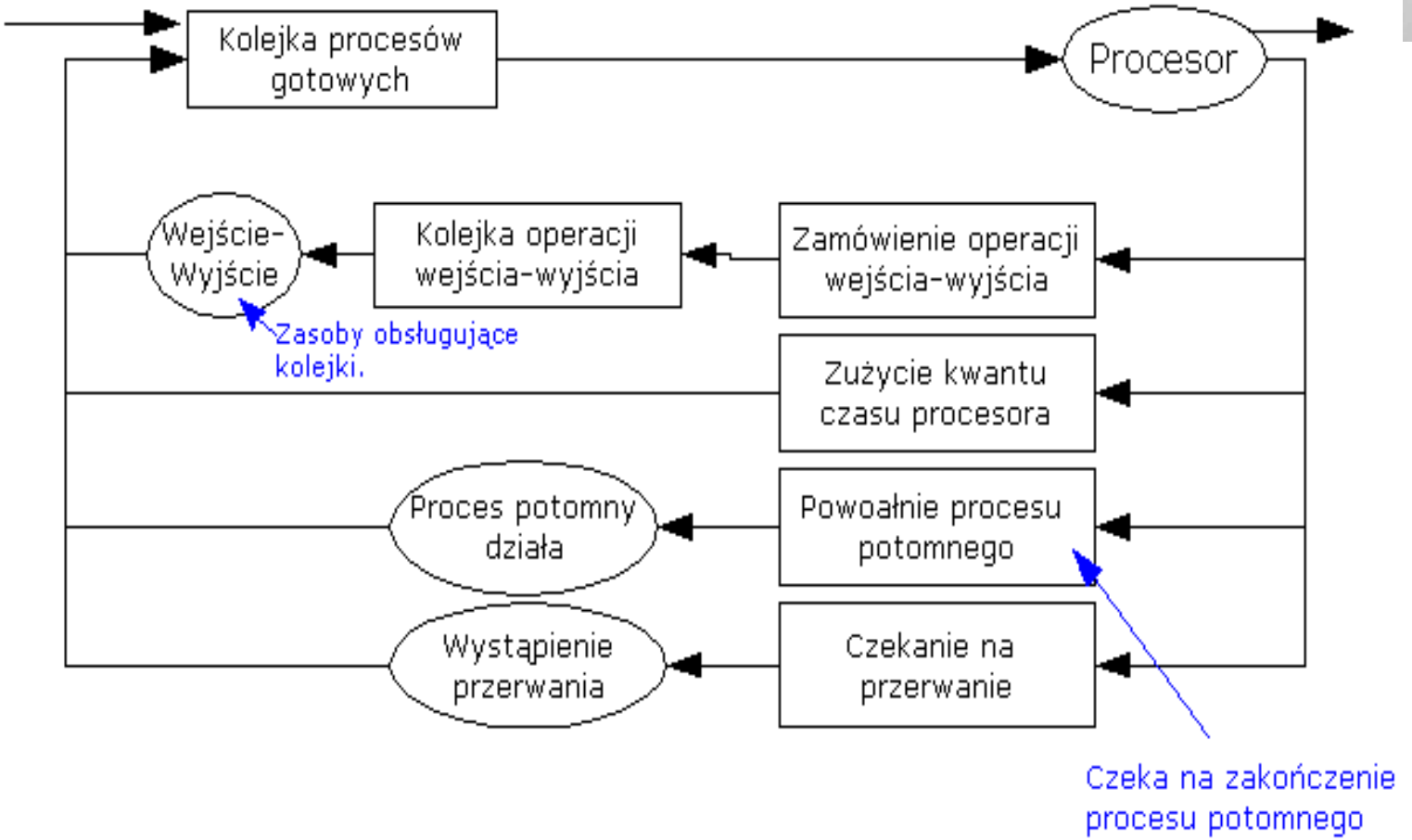
# Przykład tworzenia procesu w Unix

```
#include <stdio.h>
void main()
{
   int fork_return;
   int count = 0;
   fork_return = fork();
   system("ps");
   if( fork_return > 0)
   {
      printf("Utworzylem proces potomny od pid=%d.\n",
              fork_return);
   }
   else
   {
   sleep(200);
   printf("Jestem procesem potomnym.\n");
    }
}
```

# Tworzenie procesu w Windows

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

void main( VOID )
{
    STARTUPINFO si;
    PROCESS_INFORMATION pi;

    ZeroMemory( &si, sizeof(si) );
    si.cb = sizeof(si);
    ZeroMemory( &pi, sizeof(pi) );

    // Start the child process.
    if( !CreateProcess( NULL,   // No module name (use command line).
        TEXT("MyChildProcess"), // Command line.
        NULL,             // Process handle not inheritable.
        NULL,             // Thread handle not inheritable.
        FALSE,            // Set handle inheritance to FALSE.
        0,                // No creation flags.
        NULL,             // Use parent's environment block.
        NULL,             // Use parent's starting directory.
        &si,              // Pointer to STARTUPINFO structure.
        &pi )             // Pointer to PROCESS_INFORMATION structure.
    )
    {
        printf( "CreateProcess failed (%d).\n", GetLastError() );
        return;
    }
    // Wait until child process exits.
    WaitForSingleObject( pi.hProcess, INFINITE );

    // Close process and thread handles.
    CloseHandle( pi.hProcess );
    CloseHandle( pi.hThread );
}
```

- Kolejka procesów - Dynamika

# Planowanie przydziału procesora - Planiści

- Planista - Algorytm szeregowania (ang. scheduler - planista) to algorytm rozwiązujący jedno z najważniejszych zagadnień informatyki - jak rozdzielić czas procesora i dostęp do innych zasobów pomiędzy zadania, które w praktyce zwykle o te zasoby konkurują.
- Planiści:
  - Planista długoterminowy (ang. long term scheduler)
  - Planista krótkoterminowy (ang. short term scheduler)
  - Kryteria planowania: wykorzystanie procesora, przepustowość, czas cyklu przetwarzania, czas oczekiwania, czas odpowiedzi
- Algorytmy planowania:
  - Pierwszy zgłoszony pierwszy obsłużony (FCFS – first come first serve)
  - Najpierw najkrótsze zadanie (SJF - Shortest Job First)
  - Planowanie priorytetowe
  - Planowanie rotacyjne (RR – round-robin)

# Planiści w szczegółach

- Planista krótkoterminowy, planista przydziału procesora (ang. CPU scheduler) — zajmuje się przydziałem procesora do procesów gotowych.
- Planista średnioterminowy (ang. medium-term scheduler) — zajmuje się wymianą procesów pomiędzy pamięcią główną a pamięcią zewnętrzną (np. Dyskiem).
- Planista długoterminowy, planista zadań (ang. long-term scheduler, job scheduler) — zajmuje się ładowaniem nowych programów do pamięci i kontrolą liczby zadań w systemie oraz ich odpowiednim doborem w celu zrównoważenia wykorzystania zasobów.

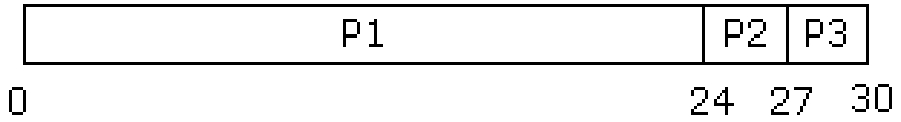
# Definicje

- Kwant czasu - minimalny przedział czasu, który jest wyznaczany przerwaniem zegara, ang. timer)
- 
- Opóźnienie ekspedycji (dispatch latency) – czas od momentu zgłoszenia do momentu rozpoczęcia wykonywania zadania.
- 
- Czas oczekiwania – całkowity czas od momentu zgłoszenia, do momentu zakończenia procesu NIE użytkowany przez dany proces.
- 
- Czas wykonania zadania – całkowity czas od momentu zgłoszenia zadania do jego zakończenia.

# Pierwszy zgłoszony pierwszy obsłużony

- Załóżmy że procesy P1, P2, P3, nadejdą (zostanie zgłoszone żądanie wykonania) w tym samym momencie (czyli w tej samej chwili) w kolejności P1,P2,P3.
- Załóżmy niezbędny czas użycia procesora dla procesów:
- P1 - 24 [kwanty czasu], P2 – 3 [kwanty czasu], P3 – 3 [kwanty czasu]

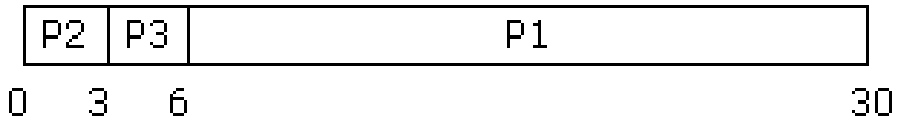
Przypadek gdy procesy nadchodzą w kolejności: P1,P2,P3:




Średni czas oczekiwania: $(c_{P1} + c_{P2} + c_{P3})/3 = (0 + 24 + 27)/3 = 17$

---

Przypadek gdy procesy nadchodzą w kolejności: P2,P3,P1:



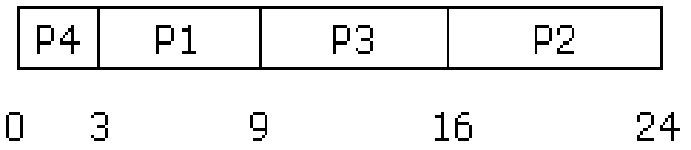

Średni czas oczekiwania: $(c_{P1} + c_{P2} + c_{P3})/3 = (6 + 0 + 3)/3 = 3$

# Najpierw najkrótsze zadanie

- Załóżmy cztery procesy o czasach użycia procesora w kwantach czasu:

P1 – 6,
P2 – 8,
P3 – 7,
P4 - 3

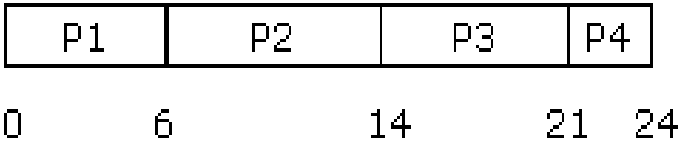
Przypadek gdy procesy nadchodzą w kolejności: P1,P2,P3,P4:




Średni czas oczekiwania: $(c_{p1} + c_{p2} + c_{p3} + c_{p4})/3 = (3 + 16 + 9 + 0)/4 = 7$

---

Natomiast przy zastosowaniu metody FCFS:

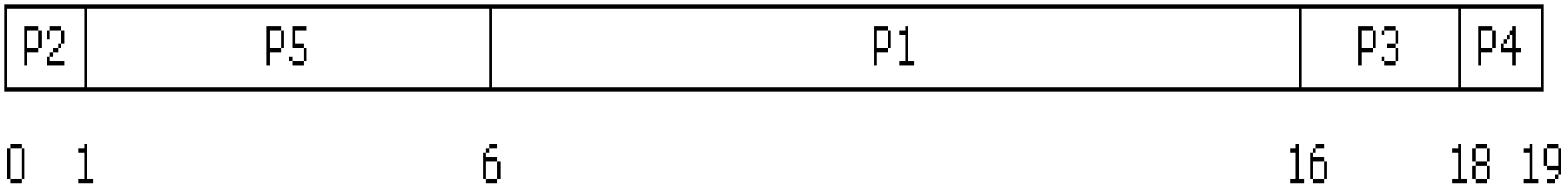



Średni czas oczekiwania: $(c_{p1} + c_{p2} + c_{p3} + c_{p4})/4 = (0+6+14+21)/4 = 10,25$

# Planowanie priorytetowe

- Załóżmy następujący przypadek procesów:

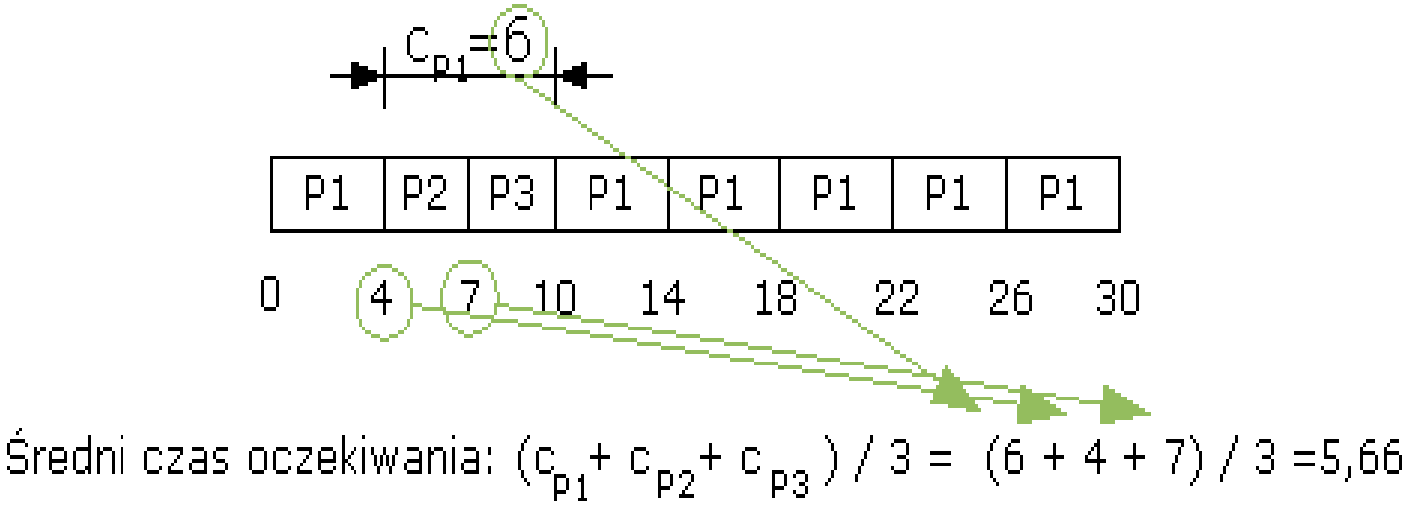
| Proces | Czas trwania | Priorytet |
|---|---|---|
| P1 | 10 | 3 |
| P2 | 1 | 1 |
| P3 | 2 | 3 |
| P4 | 1 | 4 |
| P5 | 5 | 2 |




$$(c_{P1} + c_{P2} + c_{P3} + c_{P4} + c_{P5}) / 5 = (6 + 0 + 16 + 18 + 1) / 5 = 8,2$$

# Planowanie rotacyjne (RR - round-robin)

- Dla ustalonego kwantu czasu: 4ms, oraz dla założonych procesów: P1 – 24, P2 - 3, P3 - 3




- Planowanie tylko dla systemów z podziałem czasu! (Wymagane jest wywłaszczanie.)

# Wątek Definicja

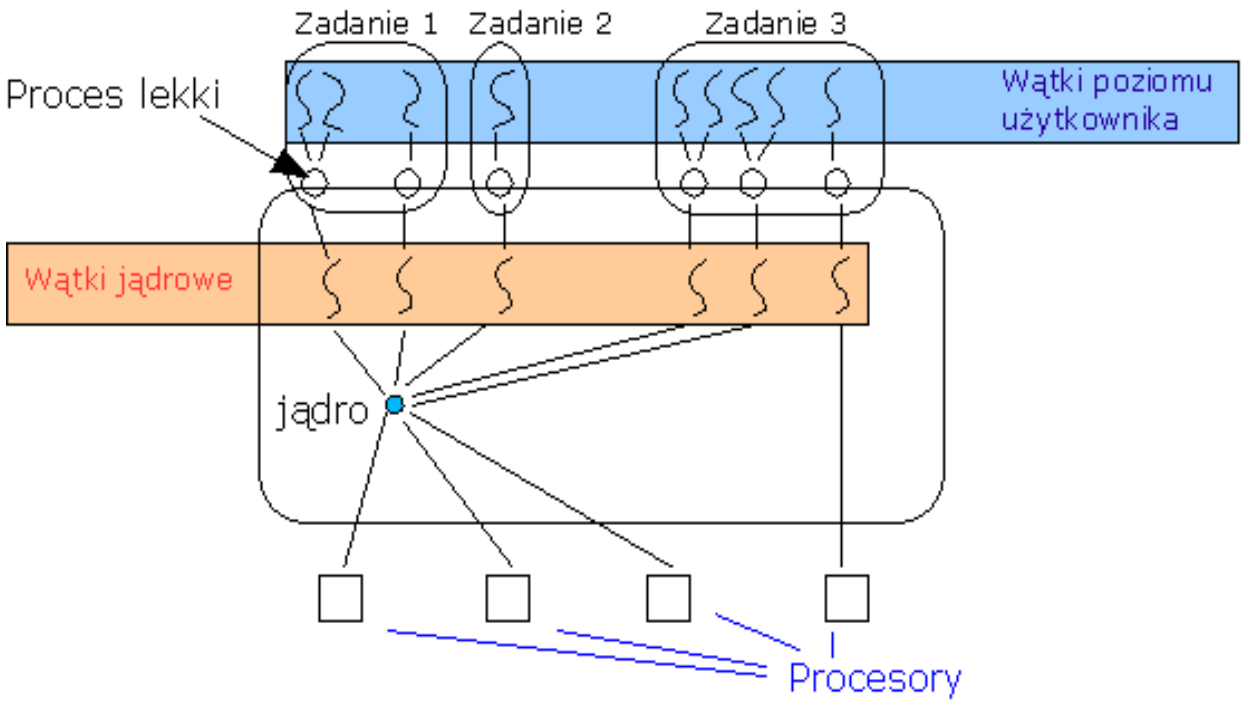
- Wątek (ang. thread) - to jednostka wykonawcza w obrębie jednego procesu, będąca kolejnym ciągiem instrukcji wykonywanym w obrębie tych samych danych (w tej samej przestrzeni adresowej).
- **Wątki tego samego procesu korzystają ze wspólnego kodu i danych, mają jednak oddzielne stosy.**
- W systemach wieloprocesorowych, a także w systemach z wywłaszczaniem, wątki mogą być wykonywane równocześnie (współbieżnie). Równoczesny dostęp do wspólnych danych grozi jednak utratą spójności danych i w konsekwencji błędem działania programu.
- Do zapobiegania takim sytuacjom wykorzystuje się mechanizmy synchronizacji wątków: semafory, muteksy, sekcje krytyczne.

# Wątki

- Wątek (inaczej zwany procesem lekkim) jest to podstawowa jednostka wykorzystania procesora w skład której wchodzą:
  - własny licznik rozkazów,
  - własny zbiór rejestrów,
  - własny obszar stosu.
- Wątki uruchamiane są w ramach jednego procesu oraz współużytkują:
  - sekcję kodu,
  - sekcję danych,
  - zasoby systemu
- Poziom na którym następuje zarządzanie wątkami:
  - Wątki ma poziomie jądra systemu  operacyjnego
  - Wątki zewnętrzne, (w zewnętrznych bibliotekach)

# Wątki w Solaris

- Trzywarstwowa struktura:
  - wątki użytkownika
  - procesy lekkie
  - wątki poziomu jądra
  - Celem



# Interakcja

- Jeżeli coś cię zainteresowało i chciałbyś aby na następnym wykładzie zostało rozszerzone, powtórzone, omówione dokładniej, to nie krępuj się i napisz maila:

  szmurlor@iem.pw.edu.pl

- Jeżeli coś było nie jasne, napisz maila:

  szmurlor@iem.pw.edu.pl

- Jeżeli coś cię znudziło, napisz maila:

  szmurlor@iem.pw.edu.pl