# Organizacja i architektura komputerów [1]

## Wykład 9

Piotr Patronik

24 kwietnia 2015

[1] (Prawie) dokładna kopia slajdów dr hab inż. J. Biernata

# Jeden procesor, jedno zadanie

Proces (zadanie) – każdy program w trakcie wykonania

Poprawność programu sekwencyjnego

- własność częściowej poprawności
  - jeśli program się zatrzyma to zwróci poprawny wynik
- własność stopu (własność poprawności całkowitej)
  - program na pewno się zatrzyma (i zwróci poprawny wynik)

Wymagania – warunki konieczne poprawności programu sekwencyjnego

- program tworzą instrukcje, które procesor może wykonać
- wszystkie zmienne są jednoznacznie identyfikowane
  - każda zmienna jest odwzorowana w pamięci (głównej)
  - używane są dozwolone tryby adresowania
- istnieje i jest obecna w programie instrukcja zatrzymania procesora

# Jeden procesor, wiele zadań (procesów)

Procesy współbieżne – jeden rozpoczyna się przed zakończeniem drugiego

Poprawność procesu (programu współbieżnego)

- własność żywotności
  - każde oczekiwane zdarzenie (działanie) nastąpi
  - każdy proces ma realną szansę wykonania (sprawiedliwość, uczciwość)
  - własność bezpieczeństwa
  - program jest zawsze w stanie pożądanym

Wymagania – warunki konieczne poprawności programu sekwencyjnego

- żywotność – każdy proces zostanie wykonany $\longrightarrow$ przydział procesora
  - na czas wykonania procesu – możliwe zagłodzenie innych procesów
  - okresowo na ustalony czas – podział czasu (time-sharing)
- bezpieczeństwo – ochrona procesu
  - prywatność
    – osobne przestrzenie adresowe $\longrightarrow$ przestrzeń wirtualna
    – ograniczenie dostępu: tryb dostępu: (r-w-x)

# Ochrona procesu

Spójność systemu (*system integrity*)

- bezpieczeństwo systemu (*security*)
- prywatność procesów (*privacy*)

Ochrona zasobów:

- zapobieganie (*prevention*) naruszeniu spójności systemu
- reagowanie na ingerencję w mechanizm ochrony
  - wykrywanie (*detection*) błędów i ataków
  - rozpoznawanie i neutralizacja skutków ingerencji
  - unieważnianie (*nullify*) działań ingerujących w mechanizm ochrony

Ochrona zasobów na poziomie architektury maszyny rzeczywistej:

- w przestrzeni kodów – uniemożliwienie wykonania instrukcji uprzywilejowanych (*privileged*) w procesie użytkownika
- w przestrzeni operandów – wykluczenie wykonania w trybie użytkownika operacji używających zastrzeżonych operandów
- w przestrzeni danych – przypisanie każdej danej znacznika (*tag*), sprzeczne z klasyczną koncepcją pamięci

# Ochrona pamięci procesu

Separacja obszarów pamięci – prawo dostępu (*access right*)
jednolite reguły dostępu: zakaz/zezwolenie + tryb (r-w-x)

- pamięć jednozakresowa (*single domain*) – proces otrzymuje wyłącznie przydział własnej pamięci, niedostępnej dla innych procesów, co wyklucza użycie wspólnych danych i komunikację przez pamięć

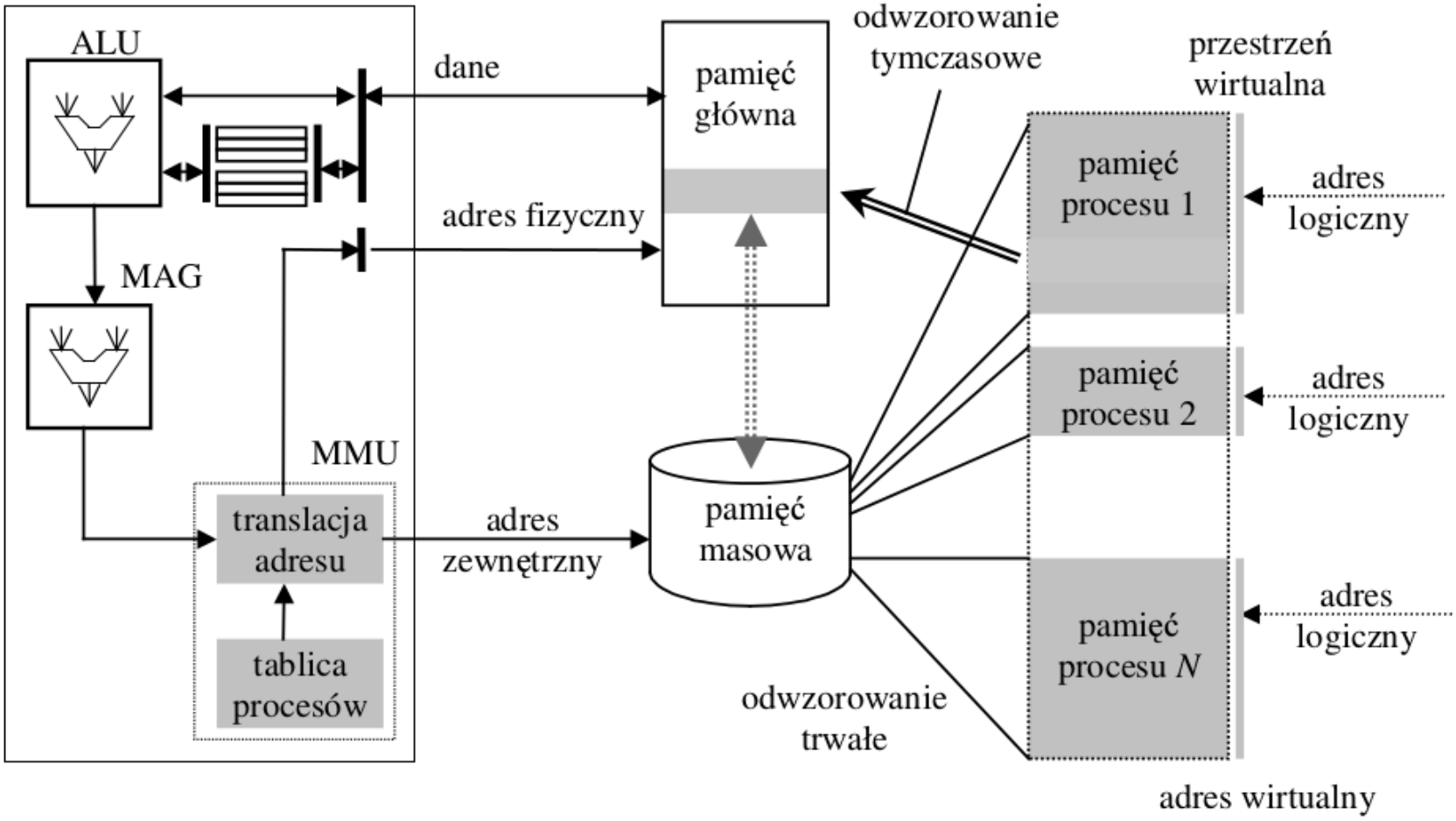
alternatywne prawa dostępu do zasobów pamięci

- pamięć dwuzakresowa (*two domain*) – jeden z obszarów jest współdzielony i dostępny dla wszystkich procesów, drugi jest obszarem własnym procesu, niedostępnym dla innych procesów, wszystkie obszary są separowane

selektywne prawa dostępu do zasobów pamięci

- pamięć wielozakresowa (*multi–domain*) – cała pamięć jest podzielona na $N$ rozłącznych obszarów (*domain*), każdy proces uzyskuje prawo dostępu do pewnego podzbioru tych obszarów
- furtki (*gate*) – deskryptory w tablicach dostępu (część kontekstu procesu)

# Pamięć wirtualna



# Odwzorowania pamięci wirtualnej

odwzorowanie spójnego bloku pamięci

- jedna reguła dostępu
- adres początku bloku
- rozmiar bloku – niezmienny podczas odwzorowania
- adres względny – niezmienny podczas odwzorowania

odwzorowanie bloku logicznego – segmentacja

- adres początku bloku
- rozmiar bloku – weryfikacja poprawności adresu

odwzorowanie bloku fizycznego (?) – stronicowanie

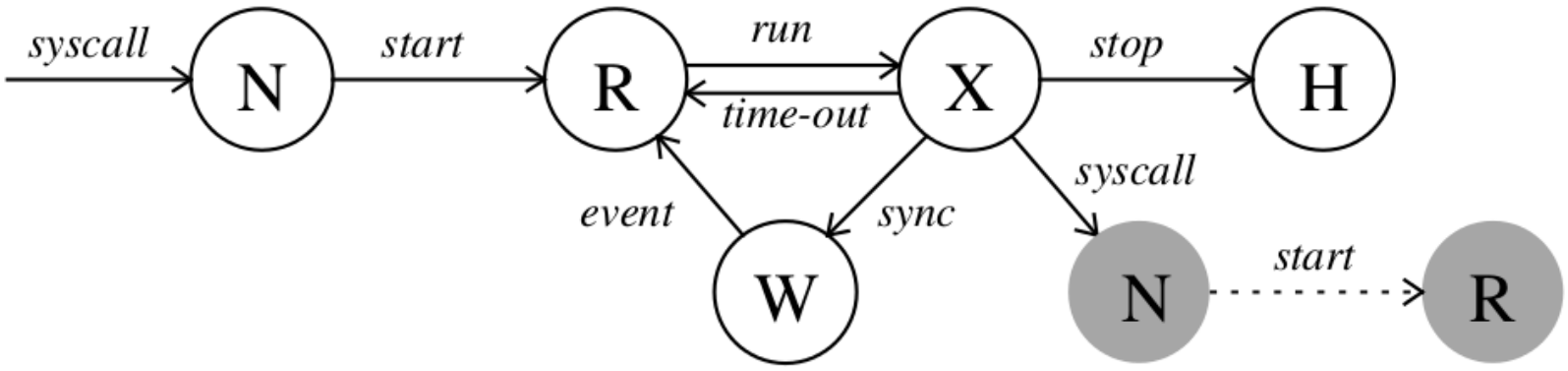
- dobór rozmiaru bloku – rozmiar $2^k$ upraszcza odwzorowanie
- adres początku bloku (numer bloku)

# Model realizacji procesu

Mechanizmy ochrony muszą być jednolite dla wszystkich procesów

- uaktywnienie – przydział czasu procesora (uwłaszczenie zadania)
- start – odtworzenie kontekstu
- wykonanie – przerwania, wyjątki, punkty komunikacji
- wstrzymanie – przechowanie kontekstu
  - zwolnienie procesora (wywłaszczenie zadania/procesu)
  - szeregowanie zadań – umieszczenie w kolejce
- kolejka krótkoterminowa – procesy aktywne, często wykonywane
- kolejka średnioterminowa – procesy aktywne, rzadko wykonywane
- kolejka długoterminowa – procesy uśpione

# Model procesowy systemu




| *syscall* | *start* | *run* |
|---|---|---|
| • nadanie identyfikatora | • wpis do kolejki | • odtworzenie kontekstu |
| • nadanie priorytetu | • | • przekazanie sterowania |
| • definicja środowiska | • | • usunięcie z kolejki |

| *time-out* | *sync* | *stop* |
|---|---|---|
| • wstrzymanie | • wstrzymanie | • likwidacja środowiska |
| • przechowanie kontekstu | • przechowanie kontekstu | • |
| • wpis do kolejki | • uśpienie | • |

# Kontekst procesu

Kontekst procesu – opis środowiska wykonania procesu
!!! (zagnieżdżenie) procesu $\longrightarrow$ utworzenie procesu potomnego !!!
Tablica procesów (*process table*, PT) – struktura danych

- obiekt w tablicy PT – blok sterujący procesu (*process control block*, PCB)
- identyfikator procesu – wskaźnik bloku PCB w tablicy procesów

Blok sterujący procesu

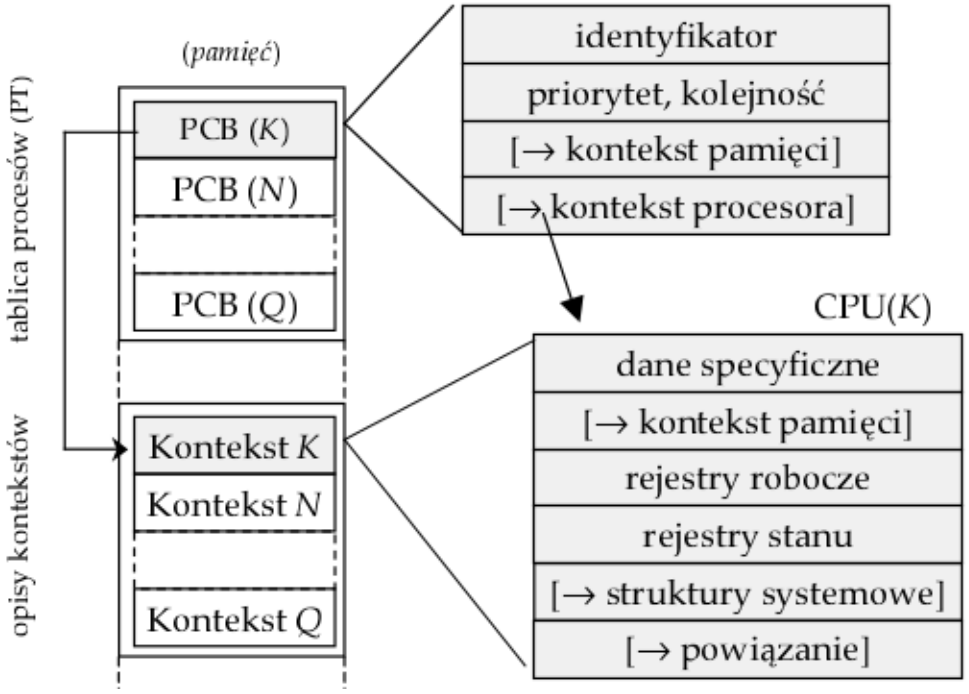
- kontekst minimalny procesu:
- (identyfikator procesu)
- minimalny kontekst procesora (rejestr stanu i licznik programu)
- minimalne zapotrzebowanie na pamięć (rozmiar zb. roboczego)
- wskaźnik pełnego kontekstu procesora
- wskaźnik pełnego kontekstu pamięci
- priorytet i parametry harmonogramowania.

Minimalny kontekst procesora – zmienne stanu, które są lub mogą być automatycznie zmienione podczas wykonania kolejnej instrukcji

# Kontekst procesora

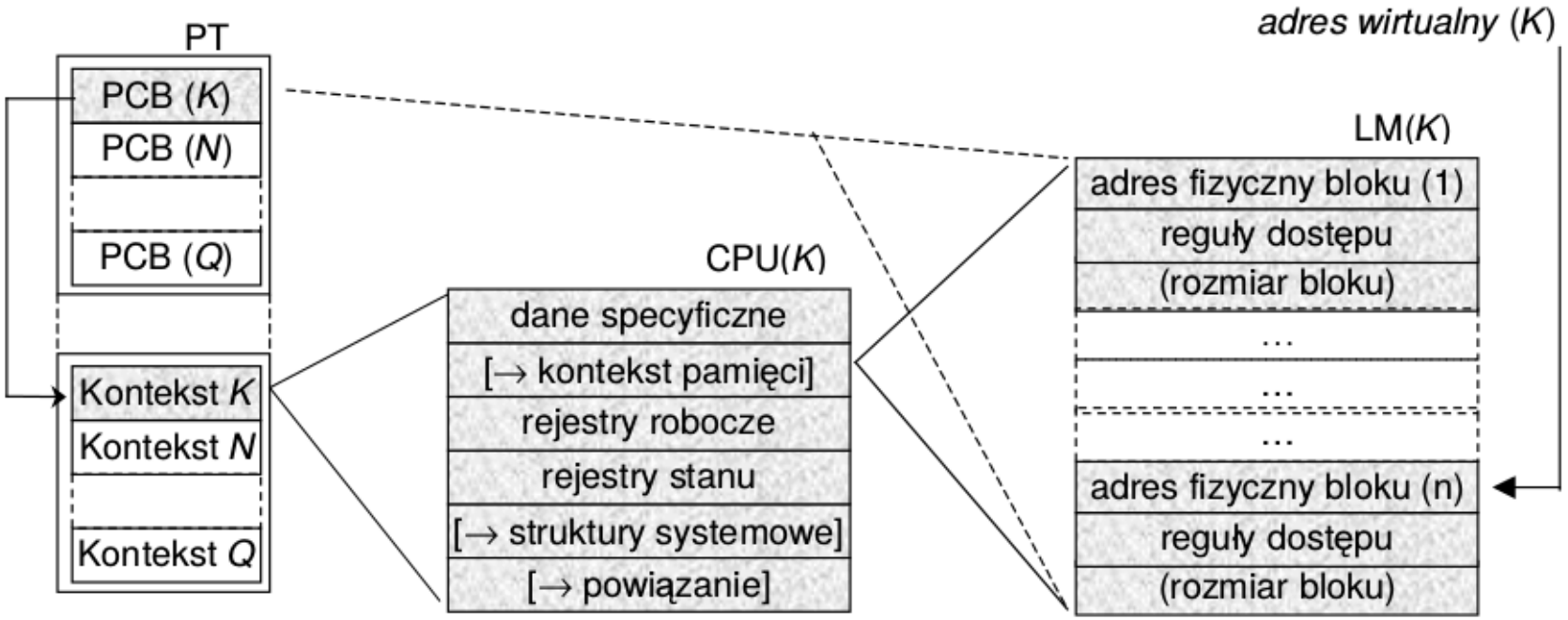
Kontekst procesora (CPU context) – opis bieżącego stanu procesu:

- zawartość rejestrów roboczych, rejestrów stanu i rejestrów wyjątków
- [wskaźnik kontekstu pamięci]
- wskaźniki dostępnych systemowych struktur danych (stosy)
- wskaźnik powiązania z procesem wywołującym (... RET)
- dane specyficzne

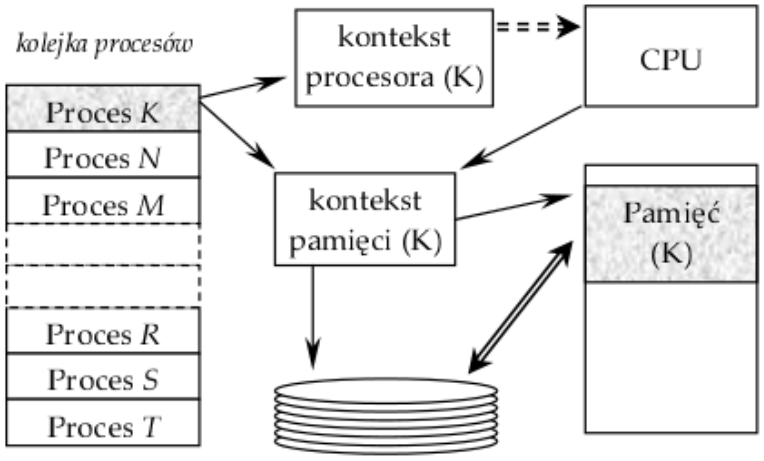


# Kontekst pamięci

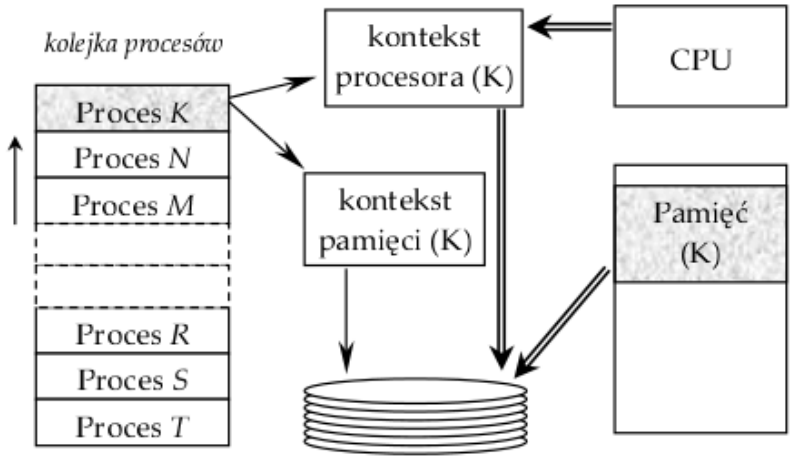
- Kontekst pamięci –
  lokalny (LM) i globalny (GM)
- Aktualizacja kontekstu pamięci tylko w trybie nadzoru
- Kontekst pamięci (memory context) –
  w tablicy opisów (deskryptorów) pamięci
  - adresy i rozmiary bloków pamięci procesu
  - obecność bloków pamięci procesu w pamięci głównej
  - reguły dostępu do bloków pamięci procesu



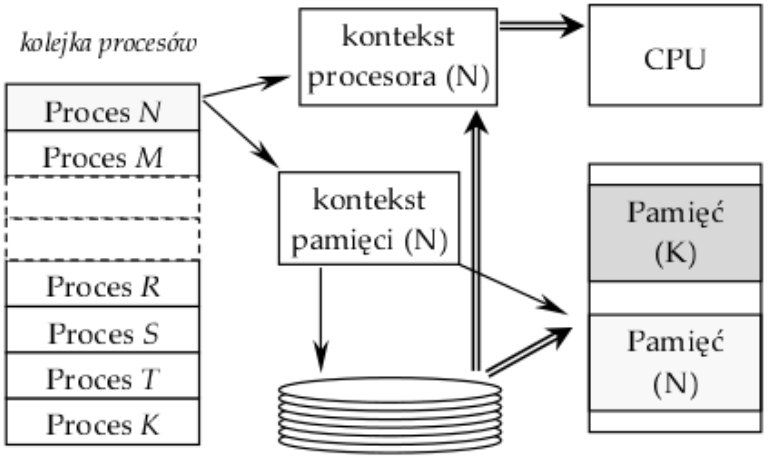
# Przełączanie kontekstów



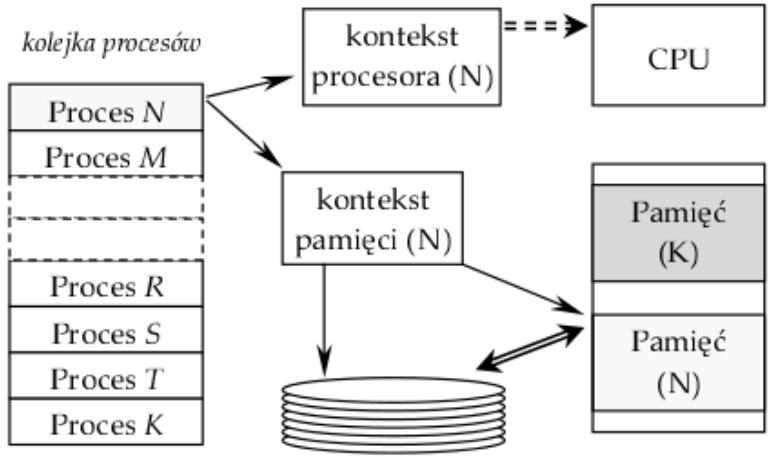

- wykonywany proces $K$




- wywłaszczenie procesu $K$




- wznowienie procesu $N$




- wykonywany proces $N$

# Synchronizacja procesów

Procesy współbieżne wymagają synchronizacji

Syndromy współbieżności procesów

- współpraca $\rightarrow$ komunikacja (wymiana danych)
  - synchronizacja wewnętrzna, określona w programie
- współzawodnictwo $\rightarrow$ konkurowanie o unikatowy zasób
  - synchronizacja zewnętrzna, realizowana w trybie nadzoru

Wzajemne wykluczanie – niezbędny warunek

- schemat dostępu do zasobu współdzielonego (shared resource)

```
...własne sprawy...
protokół wstępny - naleganie (czekanie na dostęp)
sekcja krytyczna - realizacja dostępu do zasobu
unikatowego
protokół końcowy - zwolnienie zasobu
...własne sprawy...
```

- warunek bezpieczeństwa – w każdej chwili w sekcji krytycznej może przebywać tylko jeden proces
- postulat żywotności – proces nalegający uzyska dostęp

# Problemy synchronizacji

Wzajemne wykluczanie (*mutual exclusion*) – uniemożliwia jednoczesny dostęp procesów do dzielonego zasobu, ale nie chroni przed:

- blokadą (*deadlock*) (brak bezpieczeństwa)
  - każdy proces z podzbioru procesów jest wstrzymywany w oczekiwaniu na zdarzenie, które może spowodować tylko inny proces z tego podzbioru
- zagłodzenie (*starvation*), wykluczenie (brak żywotności)
  - proces nie zostaje wznowiony, mimo że zdarzenie na które czeka występuje nieskończenie wiele razy

# Metody synchronizacji procesów

Metody synchronizacji dostępu do zasobu współdzielonego

- blokowanie przerwań
  - dominacja procesu aktywnego, wykluczenie innych procesów do chwili odblokowania przerwań (sekcja krytyczna = czas procesora)
- aktywne oczekiwanie na dostęp (*busy waiting, spin lock*)
  - naleganie wymagające aktywności procesora
- kolejka – zgłoszenie żądania i oczekiwanie na potwierdzenie
  - kolejka prosta – wyklucza zagłodzenie, uniemożliwia priorytety
  - kolejka z priorytetami – nie wyklucza ryzyka zagłodzenia
  - kilka kolejek – wymaga ustalenia zasad sprawiedliwości
- semafory, monitory (mechanizmy systemowe)
  - synchronizacja na poziomie procesu

Podstawowy mechanizm synchronizacji na poziomie architektury

- instrukcje niepodzielne:
  testuj i ustaw (*test and set*)
  porównaj i przestaw (*compare and swap*)
  - łatwa realizacja aktywnego oczekiwania
  - uproszczenie realizacji kolejek i semaforów

# Aktywne oczekiwanie (1)

Synchronizacja metodą aktywnego oczekiwania (busy waiting)
(Motorola 68020+)

**test-and-set**

```
 wait:   tas lock      ; test dostępności dzielonej pamięci
         bmi wait      ; testuj ponownie, gdy zablokowane
         ...           ; sekcja krytyczna
         clr.b lock    ; zwolnij dostęp dla innych procesów
         ...
```

**compare-and-swap**

```
         moveq #$80, d2      ; ustaw zmienną testującą (1000 0000_B)
 loop:   clr.w d1            ; przygotuj blokadę
         cas.w d1, d2, lock  ; zablokuj, gdy jest dostęp
         bne loop            ; powtórz testowanie gdy zablokowane
         ...                 ; sekcja krytyczna
         clr.w lock          ; zwolnij dostęp dla innych procesów
```

# Aktywne oczekiwanie (2)

Synchronizacja metodą aktywnego oczekiwania (Intel 80x86)

**test-and-set**

```
        mov eax, zakaz              ; przygotuj blokadę
 wait:  lock xchg eax, dostep       ; test dostępności dzielonej pamięci
        cmp eax, zakaz
        je wait                     ; testuj ponownie, gdy zablokowane
        ...                         ; sekcja krytyczna (blokada dostępu)
        mov eax, wolne              ; zwolnij dostęp dla innych procesów
        mov dostep, eax
        ...
```

**compare-and-swap**

```
        mov ebx, zakaz                  ; przygotuj blokadę
 wait:  mov eax, klucz                  ; przygotuj klucz dostępu
        lock cmpxchg dostep, ebx        ; zablokuj, gdy jest dostęp (eax=do
        jne wait                        ; powtórz testowanie gdy zablokowa
        ...                             ; sekcja krytyczna
        mov eax, klucz                  ; przygotuj klucz dostępu
        mov dostep, eax                 ; zwolnij dostęp dla innych procesó
```

# Obsługa przerwań

Priorytet – zależy przede wszystkim od jego pilności (*urgency*):

- (H) naruszenie bezpieczeństwa lub błąd sprzętowy
- (M) krytyczne ze względu na czas obsługi (np. transmisja danych)
- (L) spowodowane obniżeniem przepustowości systemu lub związane z obsługą zdarzeń o ważności określonej przez użytkownika

Przerwania nieprecyzyjne (zwykle niemaskowalne)

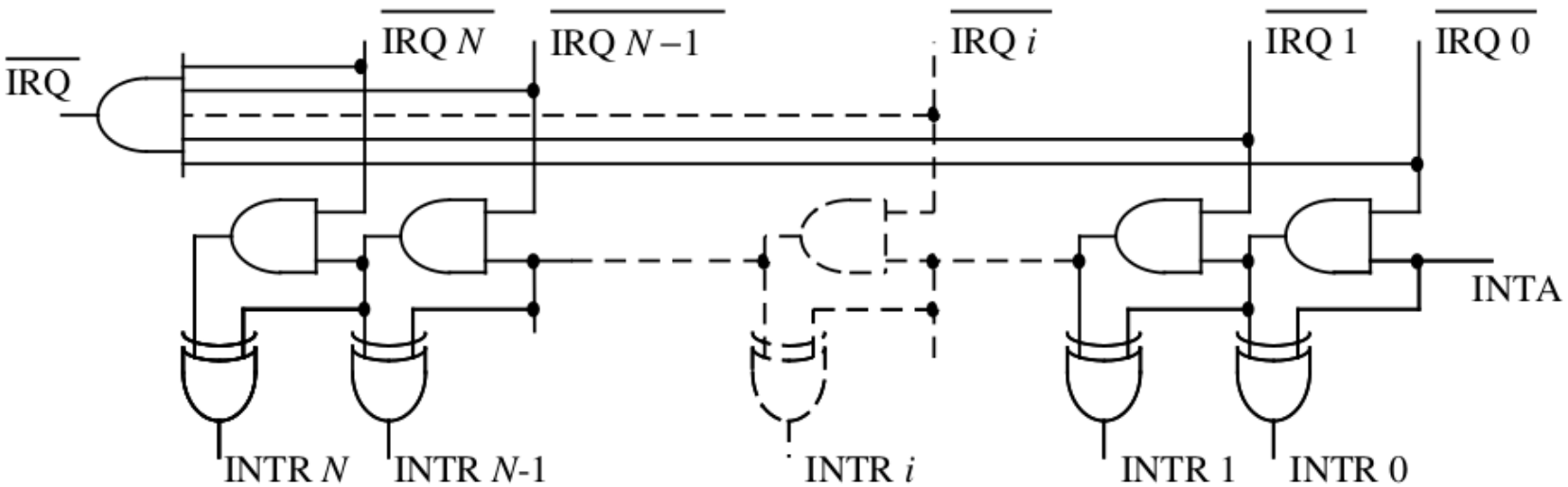
Przerwania precyzyjne – wewnętrzne (błędy wykonania)

- instrukcja została zakończona (*completed*), lecz wytworzony wynik jest błędny (na przykład wykryto nadmiar)
- instrukcja została pominięta (*suppressed*), bo naruszono reguły ochrony
- instrukcja została zignorowana (*nullified*) – wykonanie było niemożliwe, lecz po usunięciu przyczyny możliwe jest powtórzenie działania
- instrukcja została wstrzymana (*terminated*). – zewnętrzne (zdarzenia w otoczeniu) $\longrightarrow$ zwłoka obsługi (*interrupt latency*)

# Identyfikacja przerwania (1)

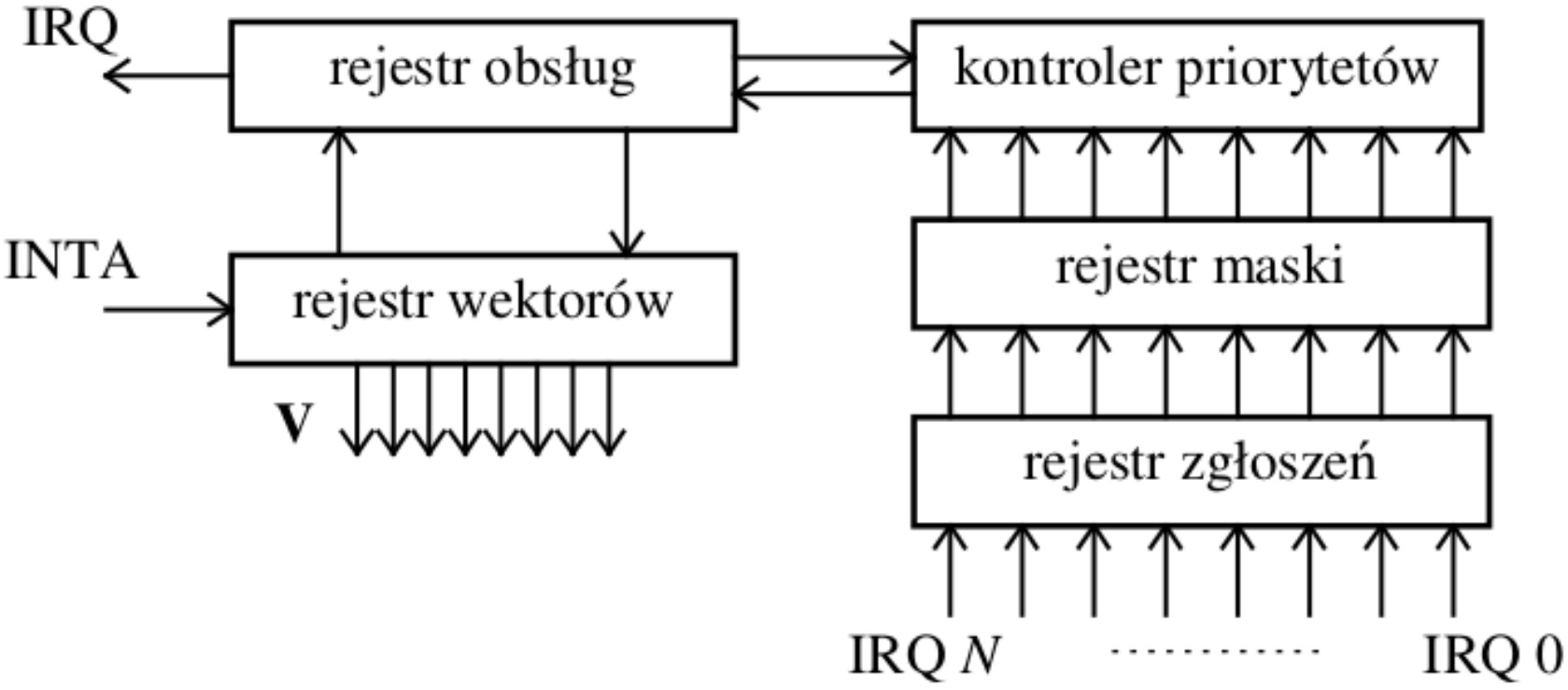
Identyfikacja źródła przerwania

- odpytywanie (*polling*)
- samoidentyfikacja
  - ustanowienie łańcucha priorytetów zgłoszeń (*daisy-chain*)
  - wektoryzacja przerwań



# Identyfikacja przerwania (2)

Sterownik przerwań wektoryzowanych




Dystrybucja obsługi (system wieloprocesorowy)

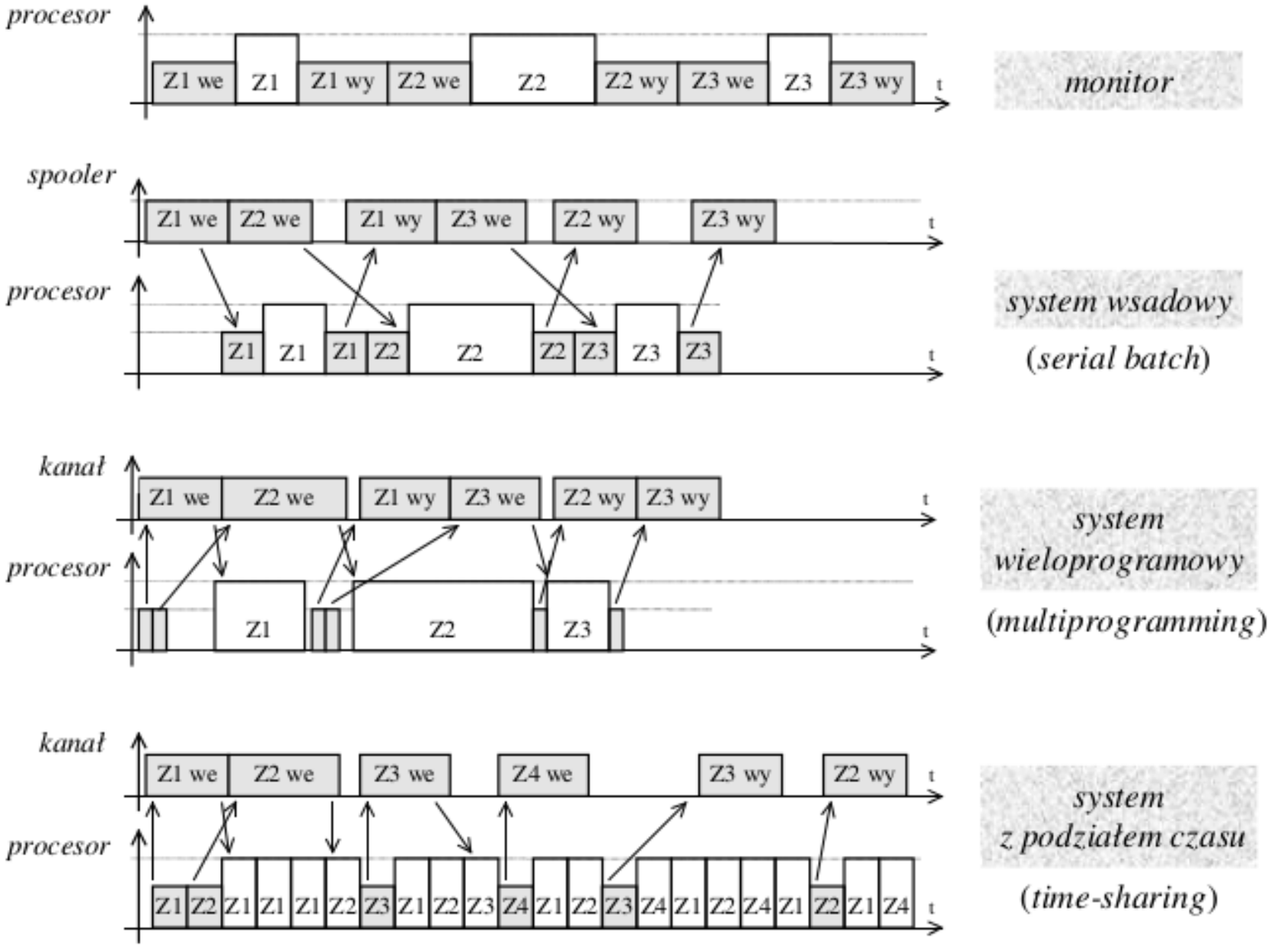
- adresowanie geograficzne
  – jawnie wskazany procesor obsługujący
- adresowanie logiczne i rozgłaszanie (broadcasting)
  – obsługę podejmuje jeden z mniej obciążonych procesorów

# Nadzór

Zapewnienie bezpieczeństwa i żywotności procesu wymaga nadzorowania

- Funkcje zarządzania procesem
  - funkcje ochrony procesu
    - tworzenie i aktualizacja struktur danych procesu
    - przełączanie procesu
  - funkcje nadzorowania procesu
    - synchronizacja procesu
    - harmonogramowanie

# System operacyjny

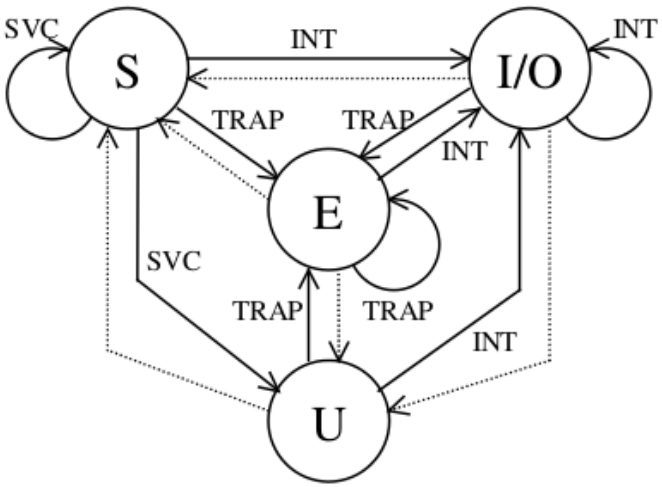


# Funkcje systemu operacyjnego

- wspomaganie użytkownika (*user functions*)
  - sterowanie i utrzymanie kontroli nad programem (*program control*)
  - obsługa wejścia/wyjścia (*I/O handling*)
  - obsługa plików (*file system manipulation*)
- funkcje systemu (*system functions*)
  - zarządzanie pamięcią (*memory management*)
  - ochrona zasobów (*resource protection*)
  - przydział zasobów (*resource allocation*)
  - obsługa wyjątków (*exception handling*)
  - harmonogramowanie (*scheduling*)
  - raportowanie (*accounting*)
- **System z podziałem czasu** (*time sharing*)
  – każde działanie jest procesem lub jego częścią

# Istotność procesu i schemat przełączania

- Poziomy uprzywilejowania (istotności) procesów
  - obsługa wyjątków (*exception handling*) – utrzymanie integralności systemu
  - obsługa we/wy (*I/O handling*) – funkcje krytyczne względem czasu
  - zadania nadzoru (*supervisor functions*) – zarządzanie procesami i pamięcią,
  - zadania użytkowników (*user jobs*)




  - zdarzenia asynchroniczne – przerwanie zewnętrzne (INT)
  - zdarzenia synchroniczne – wywołanie systemowe (SVC), pułapka (TRAP)