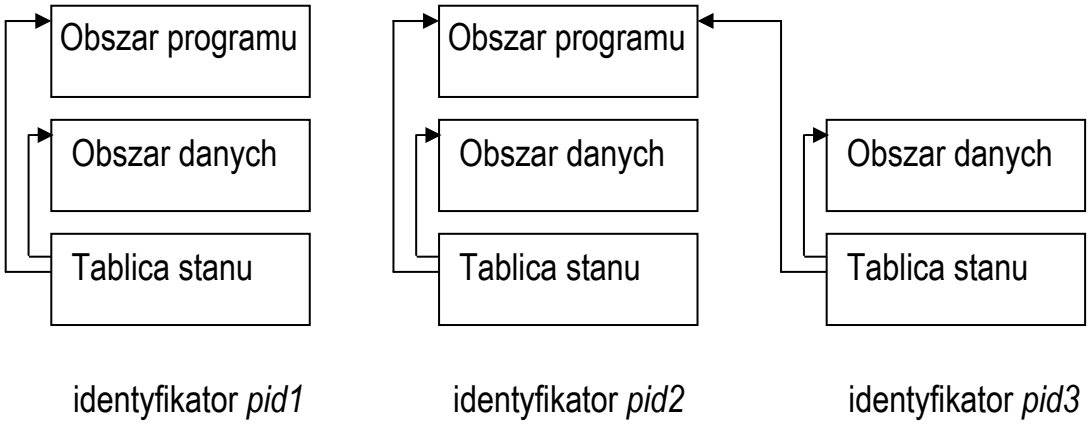
System operacyjny czasu rzeczywistego

Wymagania

- Znany czas operacji systemowych
- Kontrola czasu, w tym operacji potencjalnie nieskończonych
- Kontrola programisty nad szeregowaniem zadań
- Wspomaganie współpracy zadań
- Ograniczanie niedeterminizmu synchronizacji
- Dostęp do przerwań i układu we/wy
- Konfigurowalność

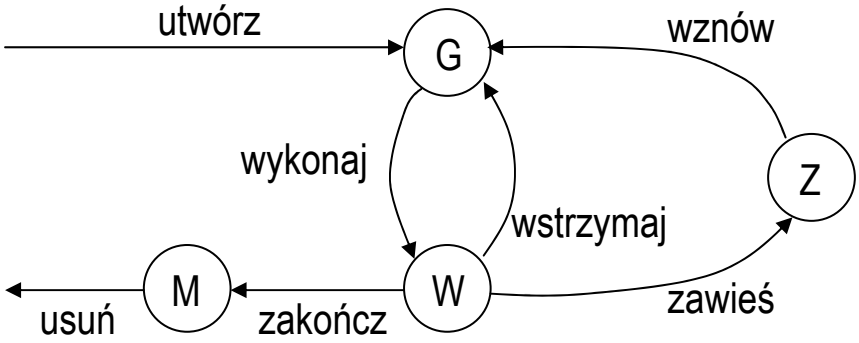
Model budowy i działania systemu

- Program
- Proces
- Reprezentacja procesu w systemie
- Wykonanie procesów



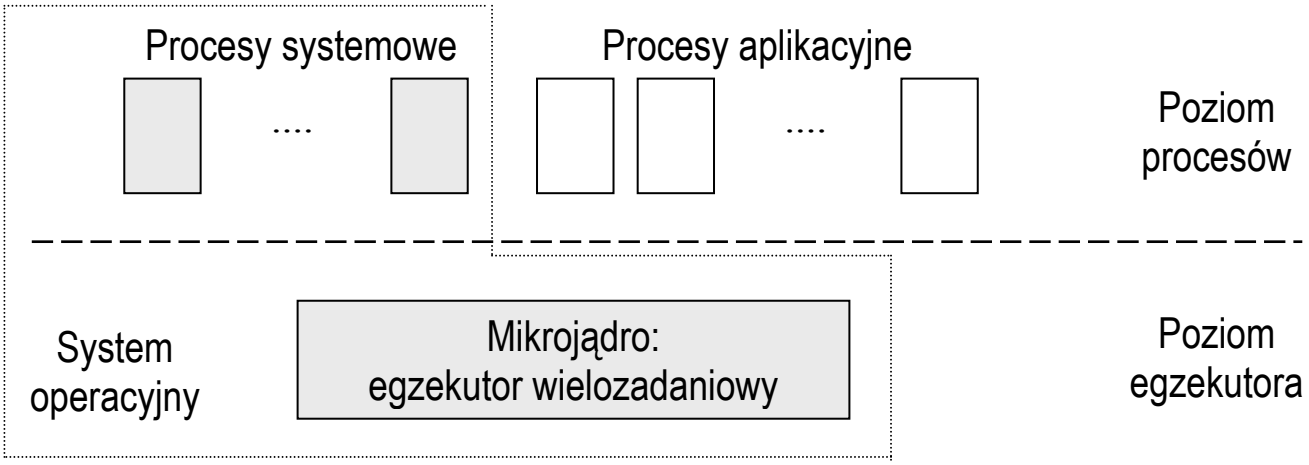
- **Stan procesu**
  - Gotowy
  - Wykonywany
  - Zawieszony
  - Martwy

- **Graf stanów**



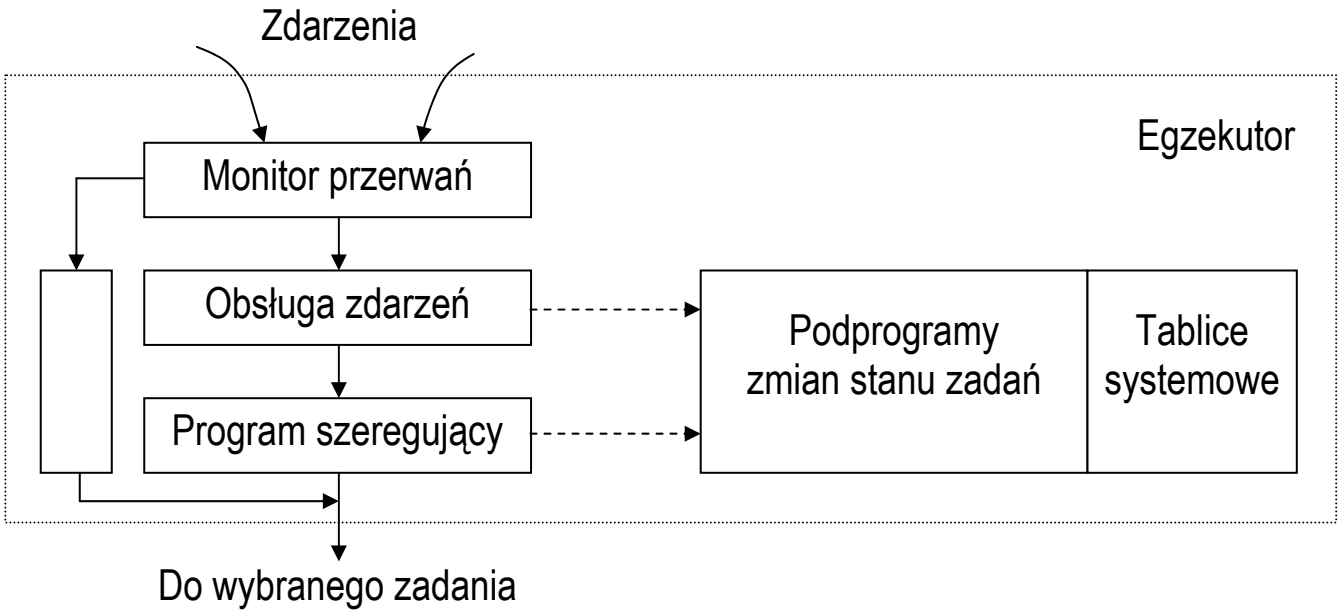
- **Zdarzenia**
  - Zewnętrzne
  - Czasowe
  - Wewnętrzne
  - Programowe

Model systemu operacyjnego z mikrojądrem



- Podział funkcji
- Architektura klient-serwer
- Mechanizm działanie

Model budowy egzekutora (mikrojądra)



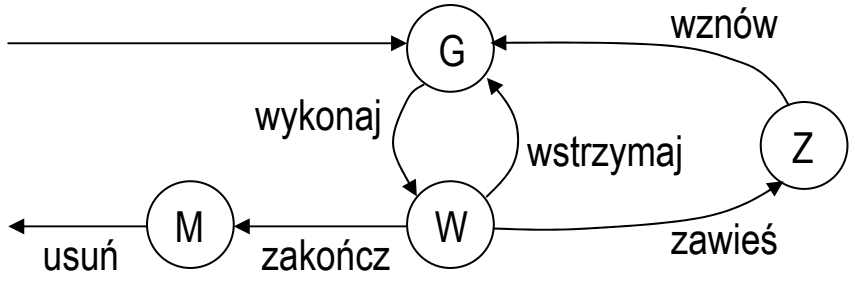
- Przejęcie sterowania
- Obsługa zdarzeń
- Szeregowanie i przełączenie kontekstu

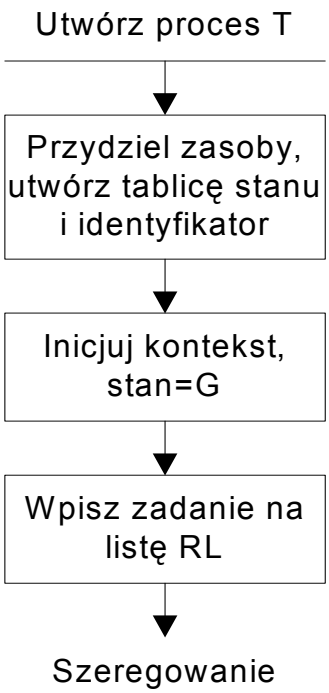
Tablice systemowe

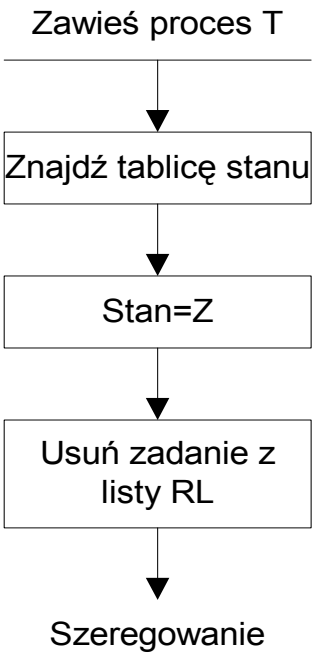
- Tablica stanu procesu
  - priorytet
  - stan bieżący
  - kontekst
  - identyfikatory
  - przydzielone obszary pamięci
  - inne przydzielone zasoby
- Lista procesów gotowych
- Listy zadań zawieszonych
- Tablice stanu zasobów

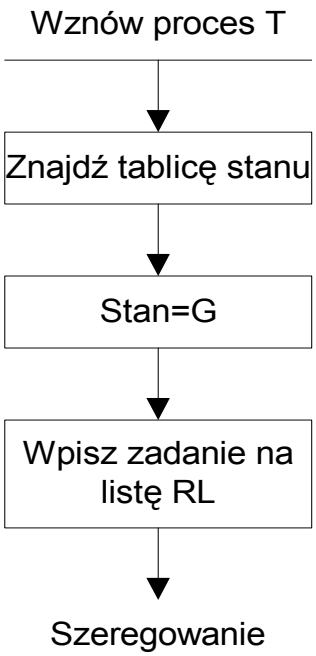
- Identyfikator procesu *pid*

Model działania egzekutora (1)

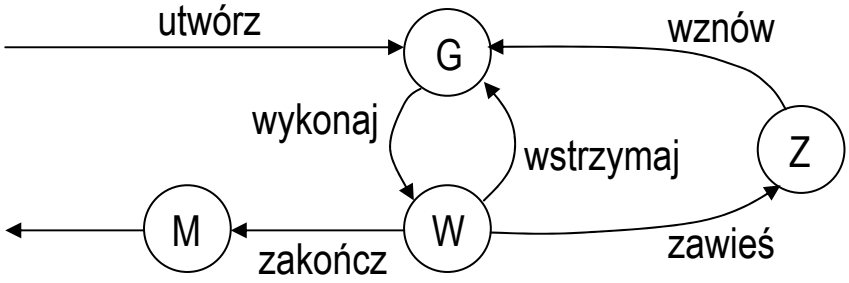


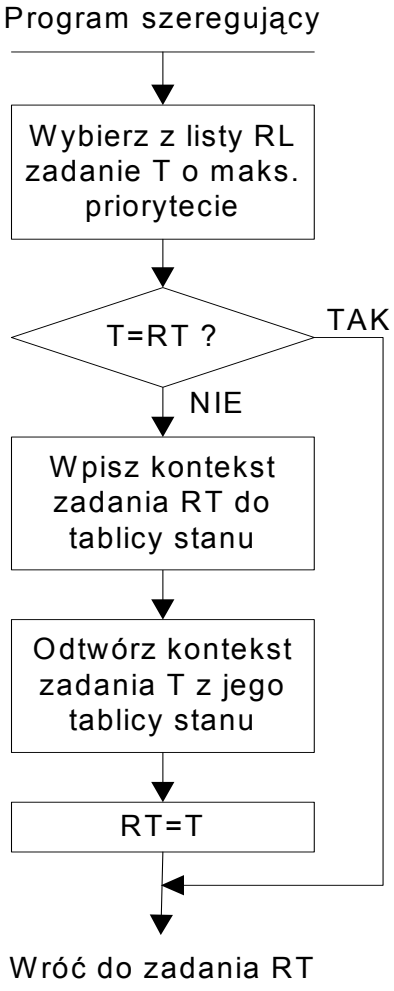


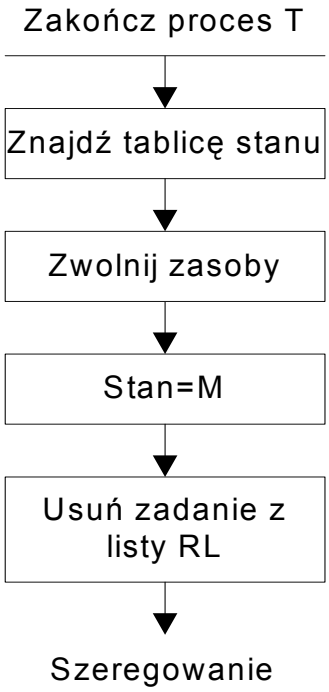


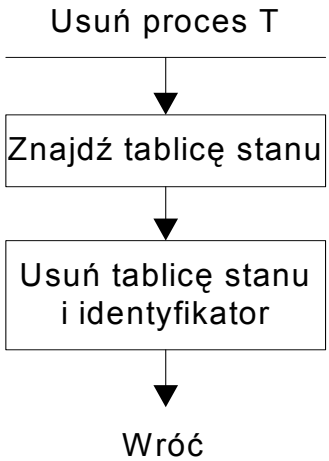


Model działania egzekutora (2)

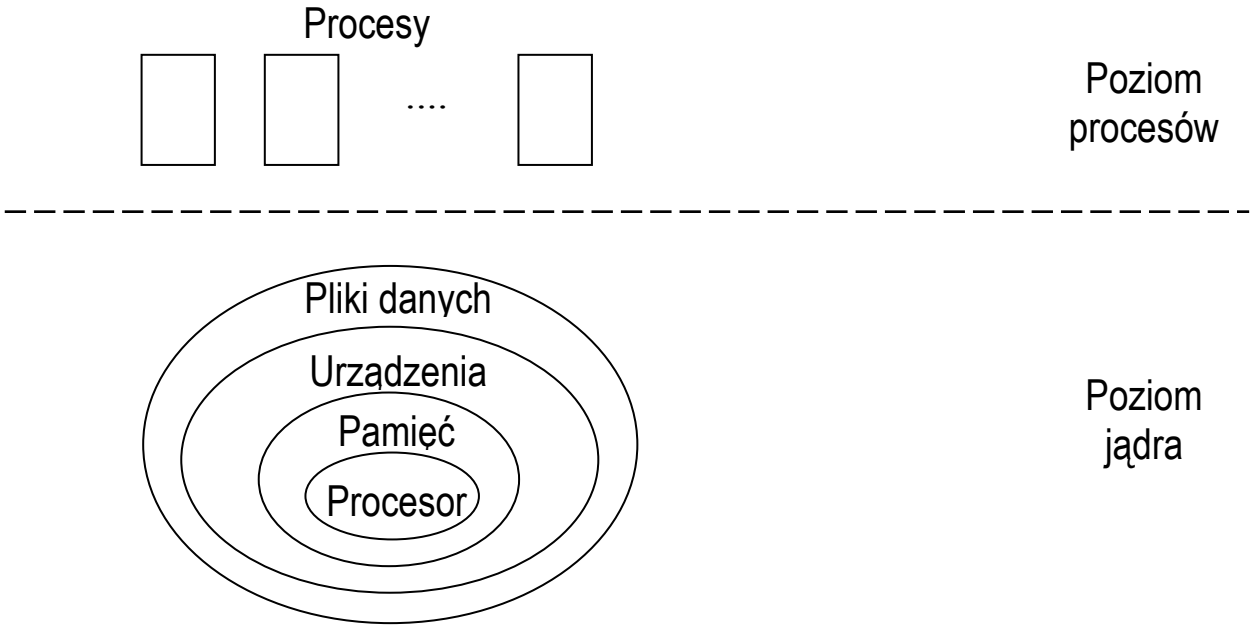




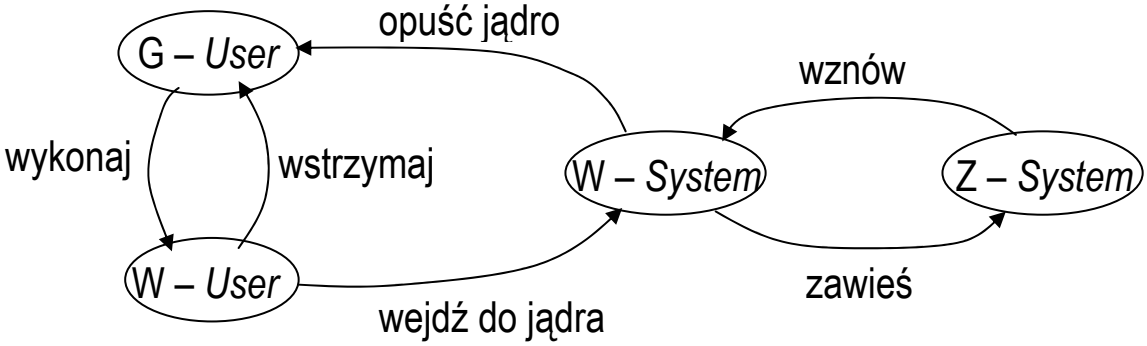




Model systemu z jądrem monolitycznym



- **Graf stanów**

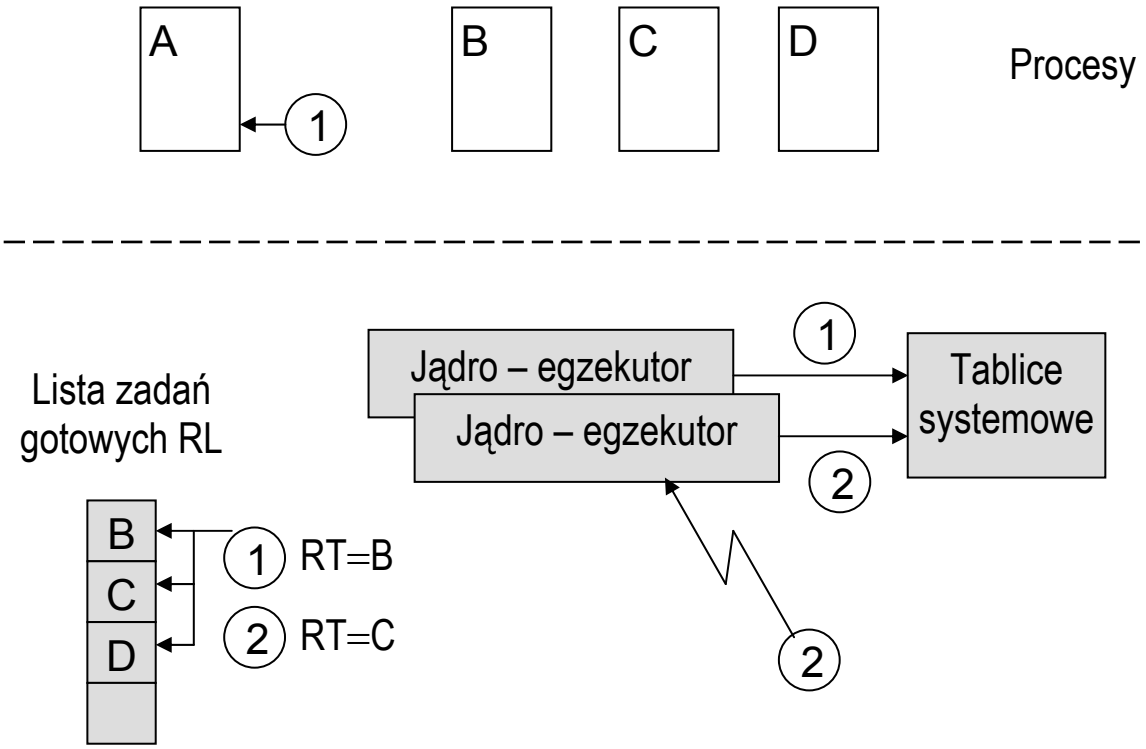


Porównanie architektur

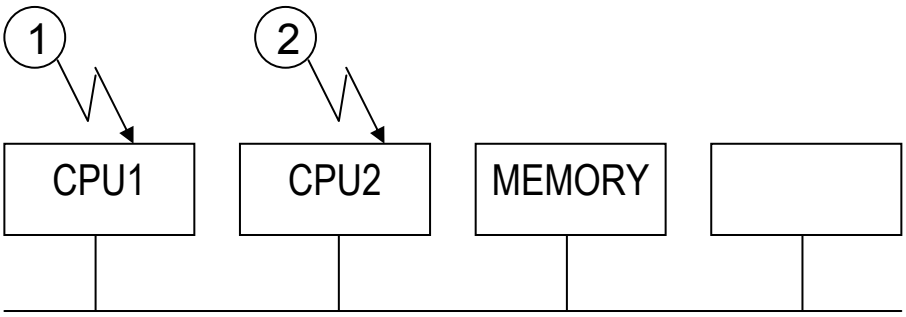
- Wydajność (przepustowość)
- Czas reakcji
  - *interrupt latency*
  - *context switch*
  - *scheduling latency*
- Modularyzacja i bezpieczeństwo
- Łatwość rekonfiguracji

Niepodzielność operacji jądra

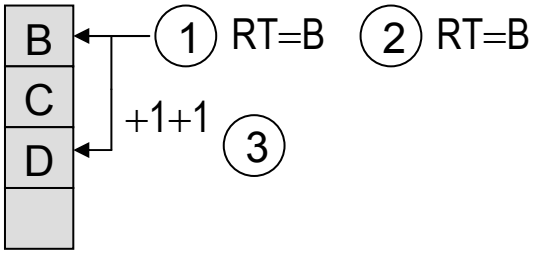
- **system jednoprocesorowy**



- **system wieloprocesorowy**



Lista zadań gotowych RL



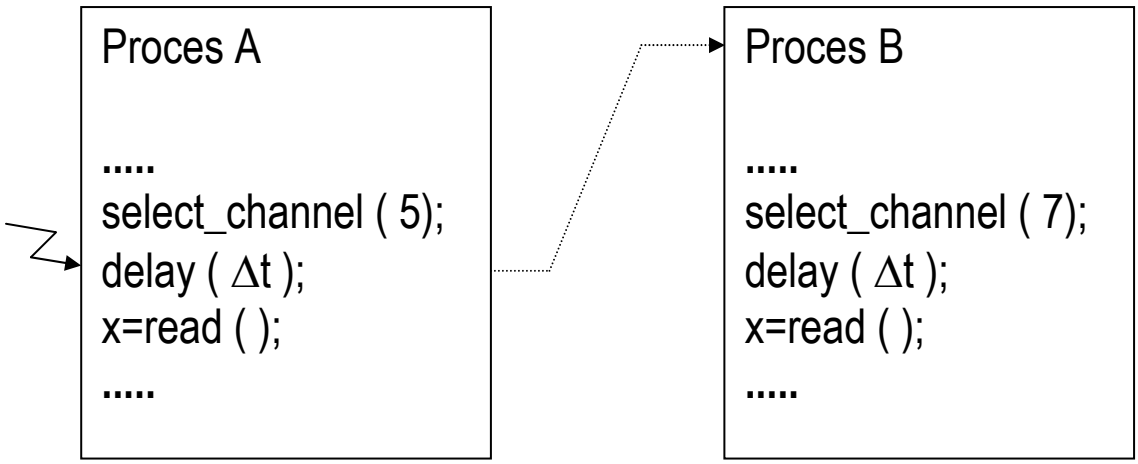
$$S = \begin{cases} 1 - \text{zasób wolny} \\ 0 - \text{zasób zajęty} \end{cases}$$

```
do { } while ( S==0 );
S=0;
.....
.....  // dostęp do tablic
.....
S=1;
```

```
R=0;
do {
    exchange ( R , S )
} while ( R==0 );
.....
.....  // dostęp do tablic
.....
S=1;
```

Synchronizacja procesów

- **Konkurencja o wspólne zasoby**
  - obszar pamięci
  - urządzenie (np. port szeregowy, przetwornik a/c)
  - oprogramowanie



- **Semafor**

Zmienna systemowa z niepodzielnymi operacjami:

Wait(S):

if (S>0) **then** S=S–1 **else** *zawieś wykonywany proces*

Post(S):

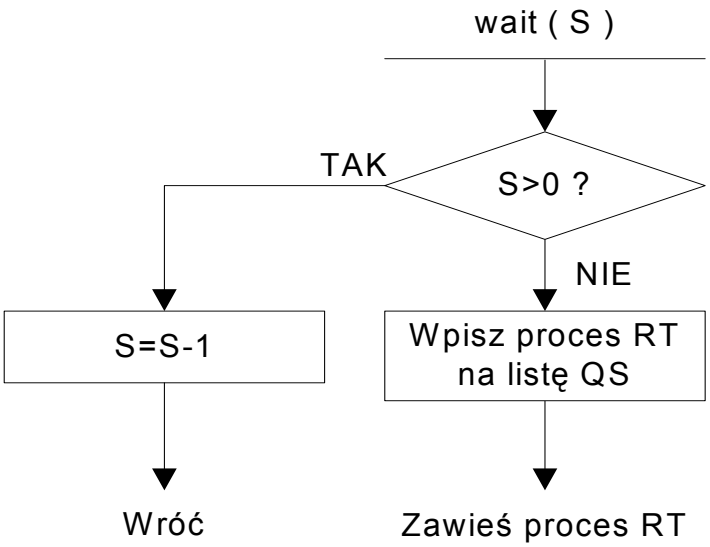
if (brak procesów zawieszonych) **then** S=S+1 **else** *wznów proces*

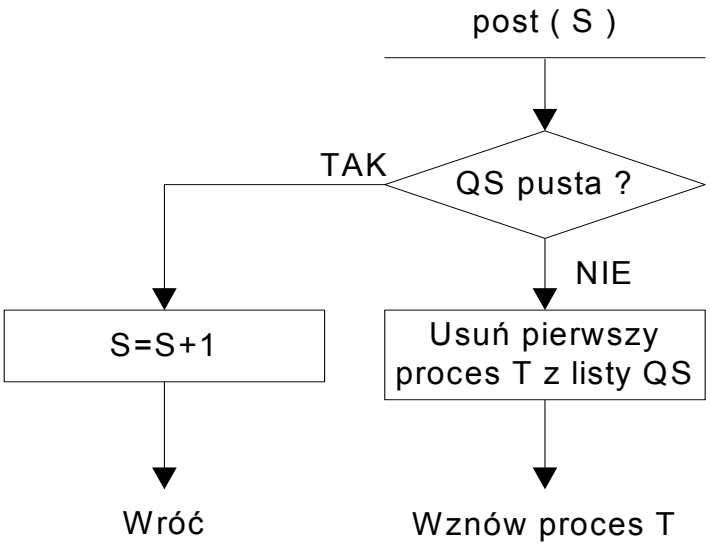
- **Wzajemne wykluczanie**

```
Proces A

.....
wait(S);
select_channel ( 5);
delay ( Δt );
x=read ( );
post(S);
.....
```

```
Proces B

.....
wait(S);
select_channel ( 7);
delay (Δt );
x=read ( );
post(S);
.....
```

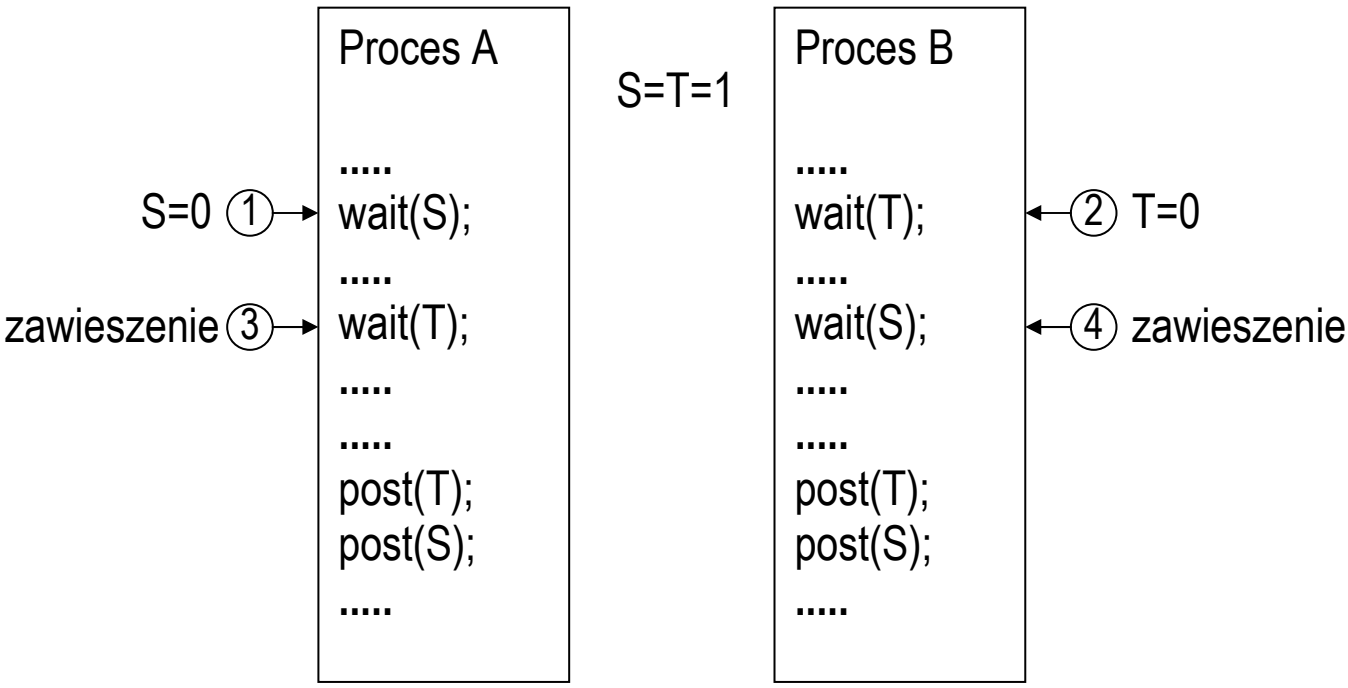
- **Implementacja semafora**
  - zmienna całkowita S
  - kolejka procesów QS





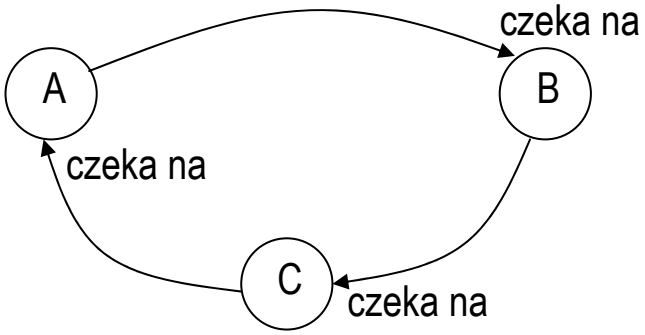
Anomalia synchronizacji

- **Zakleszczenie** (*deadlock*)



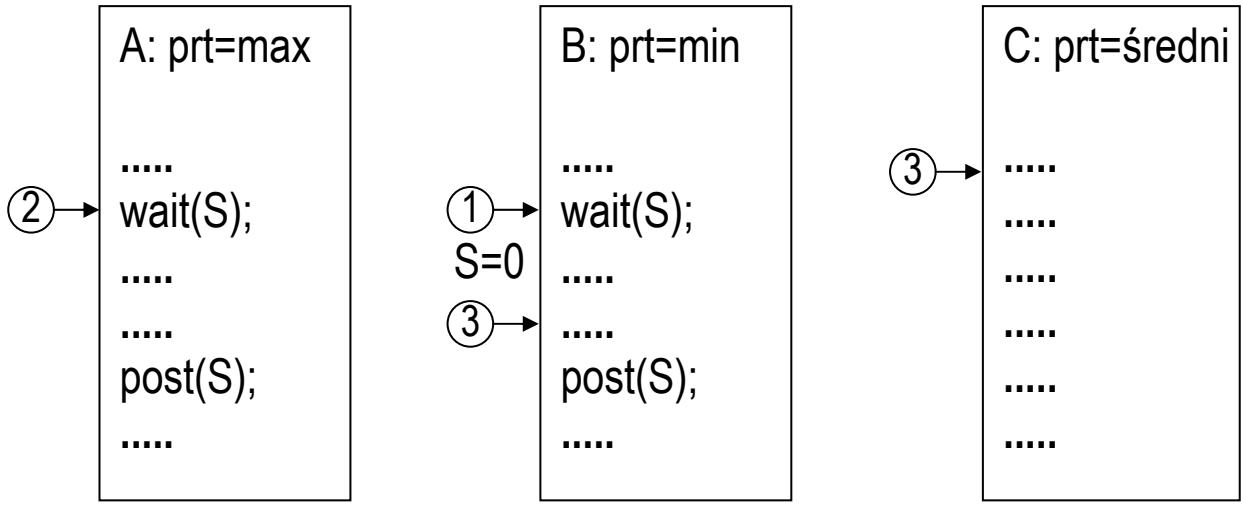
Graf zależności

Proces A	Proces B	Proces C
.....	.....	.....
wait(S);	wait(T);	wait(R);
.....	.....	.....
.....	.....	.....
post(T);	post(R);	post(S);
.....	.....	.....



warunek konieczny

- **Inwersja priorytetu** (*priority inversion*)

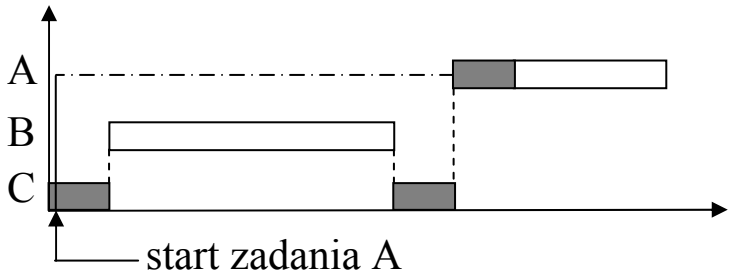


Twierdzenie (*Sha, Rajkunur, Lehoczky, 1990*)

Jeżeli: $\sum_{i=1}^{n} \frac{t_i}{c_i} + \max_{i=1..n} \left\lceil \frac{b_i}{c_i} \right\rceil \leq n\,(2^{\frac{1}{n}} - 1)$

oraz szeregowanie jest zgodne z pilnością (*RMS*)

to wszystkie zadania zostaną wykonane w terminie



Procesy i wątki (*thread*)

- Procesy jednowątkowe
- Procesy wielowątkowe
  - proces
  - wątek
- Zgodność modeli
- Implementacja
  - wątki systemu
  - wątki użytkownika
- Problemy

Przegląd systemów RT

- **Platforma sprzętowa**

	Producent	Platforma sprzętowa	POSIX
VxWorks	Wind River Systems	ARM, MIPS, PowerPC, x86, SH4, 68K, Sparc	TAK
QNX Neutrino	QNX Software Systems	ARM, MIPS, PowerPC, x86, SH4, xScale	TAK
Windows CE	Microsoft	ARM, MIPS, PowerPC, x86, SH4	NIE
LynxOS	Lynux Works	ARM, MIPS, PowerPC, x86, 68K, MPC8xx, Sparc	TAK
RT Linux	Open Source	ARM, MIPS, PowerPC, x86, SH4	TAK
eCOS	Open Source	ARM, MIPS, PowerPC, x86, SH4, 68K, MPC8xx, Sparc	TAK

- **Porównanie**

	QNX	VxWorks	Windows CE	RT Linux
Instalacja i konfiguracja	7	4	5	3
Architektura systemu	9	7	7	3
Bogactwo API programisty	7	8	7	5
Wsparcie Internetu	8	9	9	8
Narzędzia programistyczne	7	8	8	8
Dokumentacja i wsparcie	5	4	5	2
Testy wydajnościowe	9	5	7	2
okres przerwań	*10*	*25*	*11*	*60*
Średnia	7,4	6,4	6,9	4,4

Dedicated Systems Expert, RTOS Evaluation Project, 2002r (QNX v. 6.1)