Akademia Nauk Stosowanych
Wydział Nauk Inżynieryjnych
Kierunek: Informatyka
studia I stopnia, semestr 2

AKADEMIA NAUK STOSOWANYCH W NOWYM SĄCZU

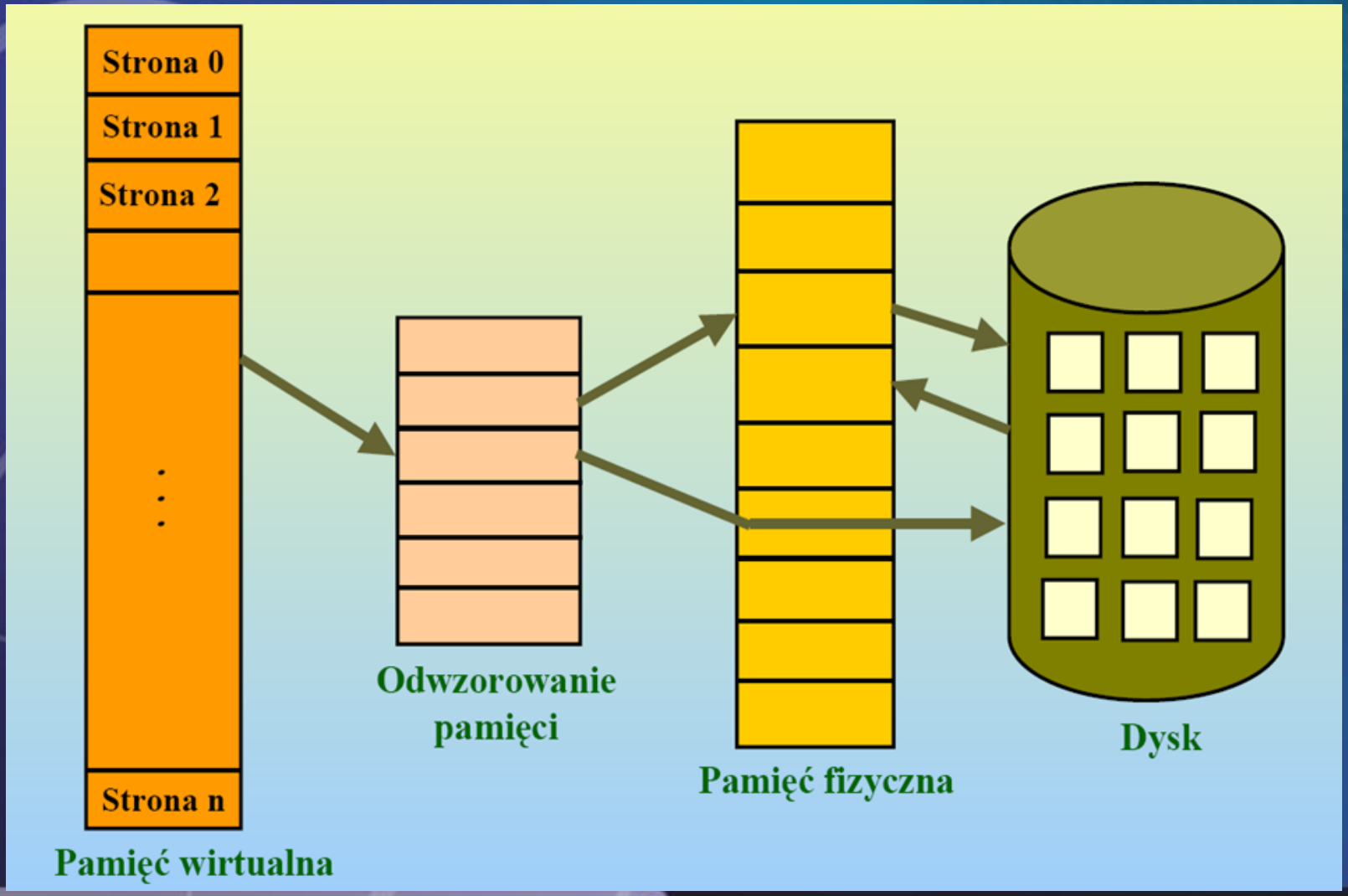
Systemy operacyjne

WYKŁAD 11

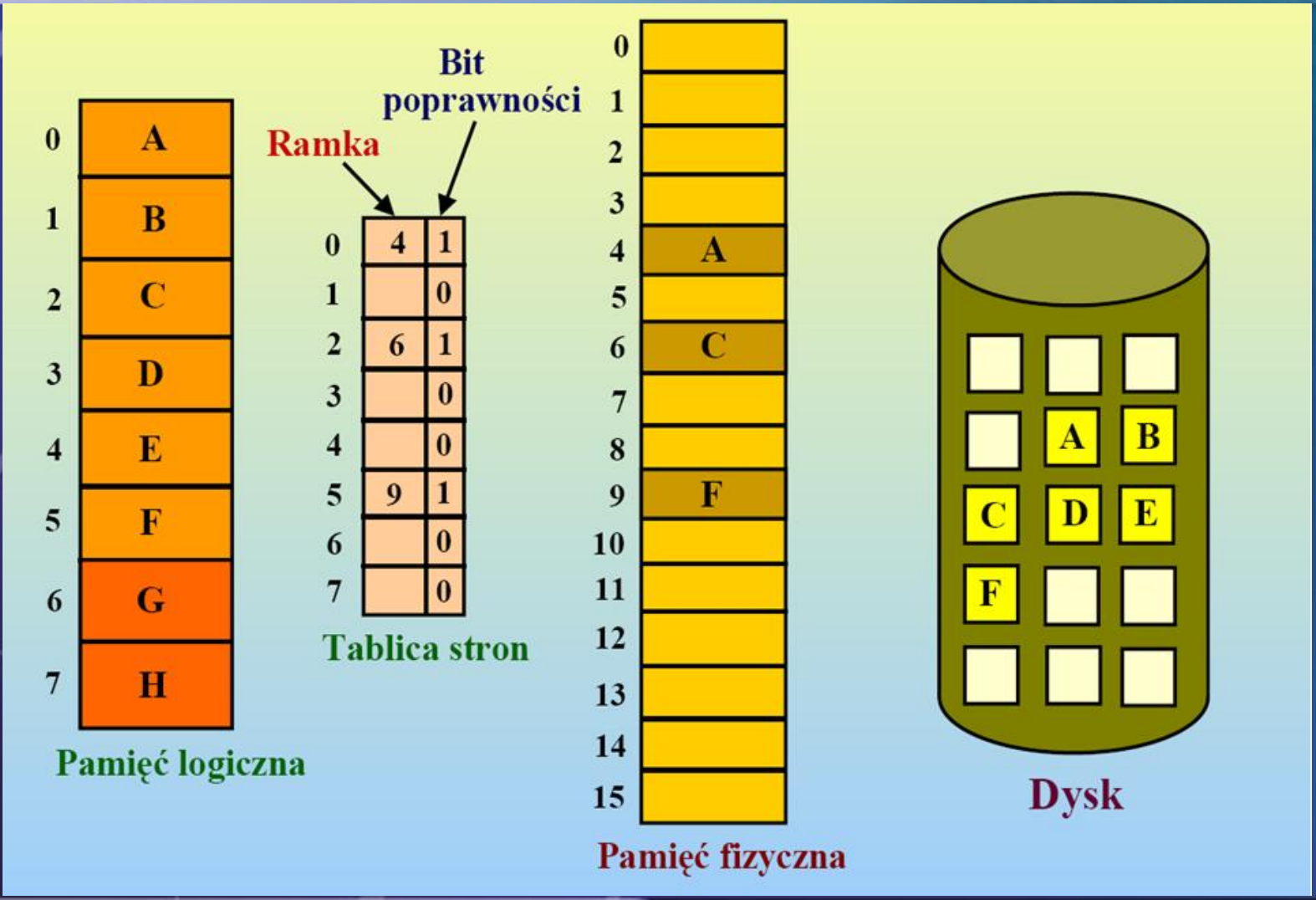
dr inż. Stanisława Plichta

splichta@ans-ns.edu.pl

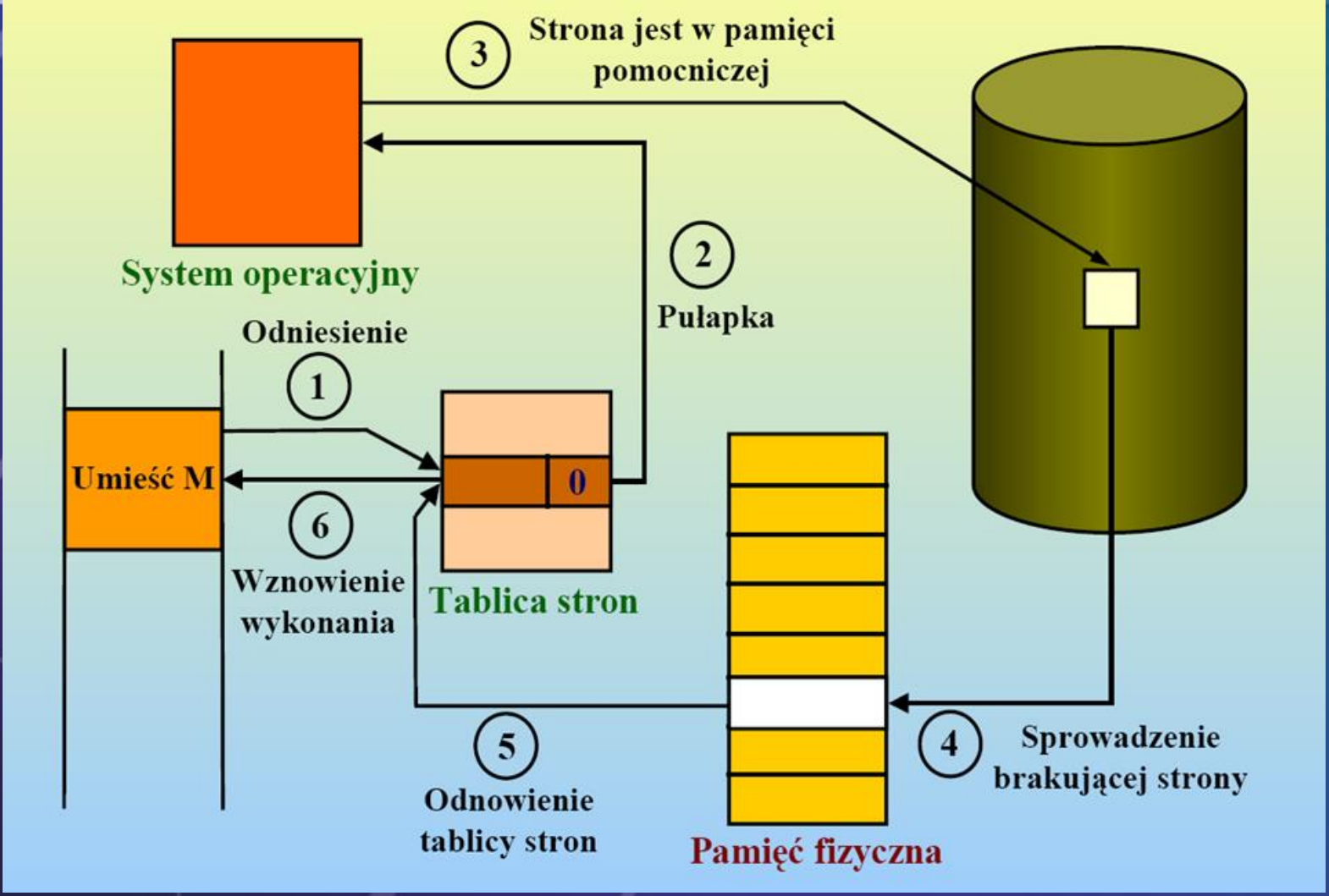
Schemat pamięci wirtualnej



Tablica stron z brakami stron w pamięci



Etapy obsługi braku strony



Efektywny czas dostępu

Oznaczmy przez *p* prawdopodobieństwo braku strony

- *Efektywny czas dostępu* (*Effective Access Time* (*EAT*) do pamięci stronicowanej – średni czas wymagany na obsługę odwołania do pamięci (pewna miara sprawności).

EAT=

(1 - p)	*	czas dostępu do pamięci
+ p	*	(narzut związany z przerwaniem braku strony + [zapisanie na dysk strony ze zwalnianej ramki] + wczytanie z dysku żądanej strony + narzut związany ze wznowieniem procesu)

Efektywny czas dostępu

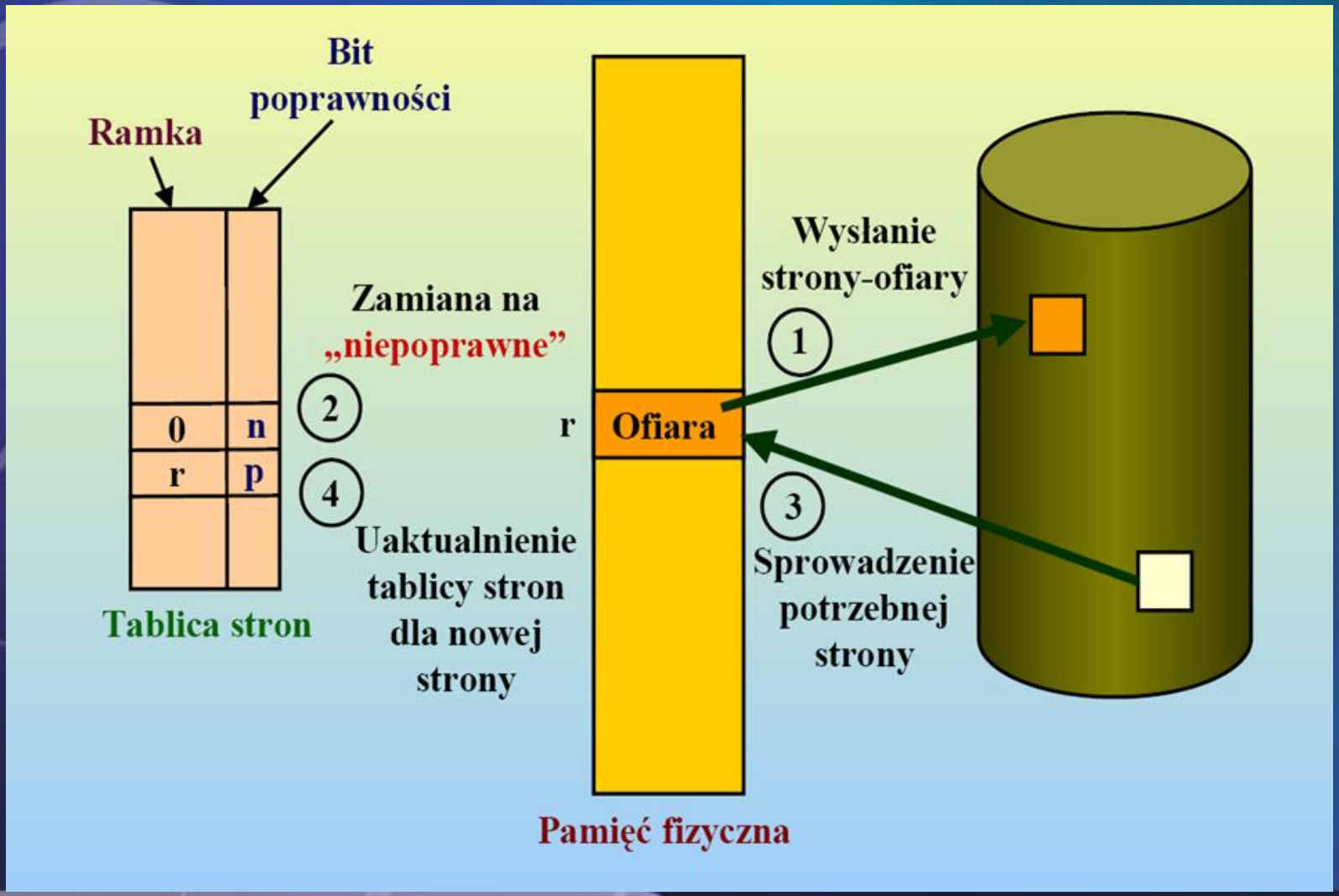
Przyjmijmy następujące założenia:

- Czas dostępu do konwencjonalnej pamięci to 1 mikrosekunda
- Połowa z zastępowanych stron wymaga zapisu na dysk
- Czas dyskowego zapisu i odczytu strony to 10 milisekund (10000 mikrosekund)
- Zaniedbujemy narzuty związane z obsługą przerwania i wznowieniem procesu (są one niewielkie w porównaniu z gigantycznym kosztem operacji dyskowej)

$$EAT = (1 - p) * 1 + p * (50\% * 10000 + 10000) = 1 + 14999 * p$$

EAT jest wprost proporcjonalny do prawdopodobieństwa wystąpienia przerwania braku strony

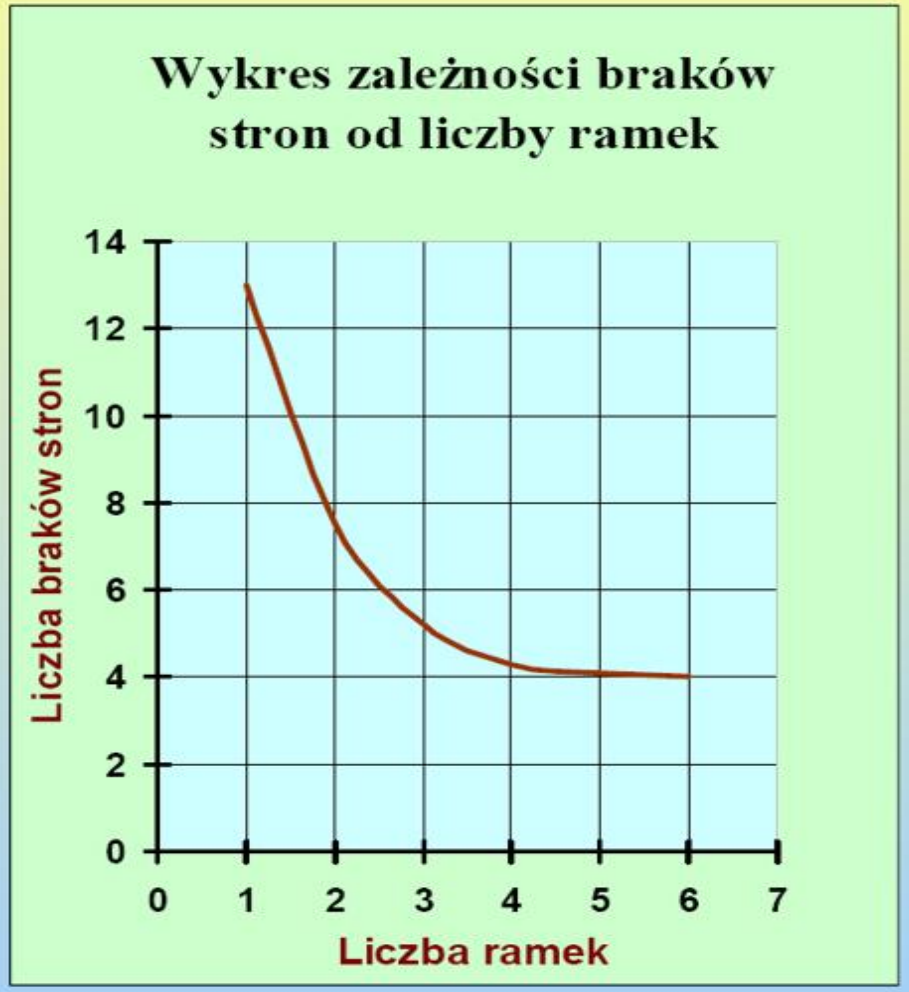
Schemat zastępowania stron



Algorytm zastępowania stron

- Algorytm zastępowania stron powinien minimalizować **częstość braków stron** *(page-fault rate)*.
- Algorytm ocenia się na podstawie wykonania go na pewnym **ciągu odniesień** *(reference string)* do pamięci i zsumowania braków stron.
  - Ciąg odniesień można tworzyć sztucznie lub na podstawie śledzenia danego systemu.
- W poniższych przykładach ciąg odniesień ma postać:

1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5.



Algorytm FIFO zastępowania stron

liczba ramek wynosi 3

Chwila	1	2	3	4	5	6	7	8	9	10	11	12
Odwołanie	1	2	3	4	1	2	5	1	2	3	4	5
Ramka 1	1	1	1	4	4	4	5	5	5	5	5	5
Ramka 2		2	2	2	1	1	1	1	1	3	3	3
Ramka 3			3	3	3	2	2	2	2	2	4	4

ofiarą staje się strona, która najdłużej przebywa w pamięci

autor: dr inż. Stanisława Plichta

Algorytm FIFO zastępowania stron

Przyjrzyjmy się teraz, co się stanie, gdy liczba ramek wyniesie 4

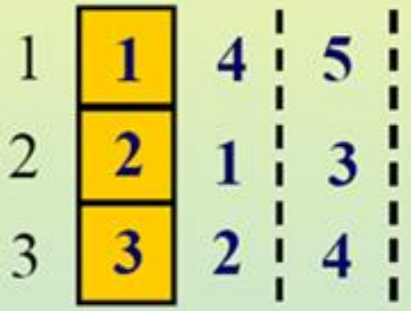
Chwila	1	2	3	4	5	6	7	8	9	10	11	12
Odwołanie	1	2	3	4	1	2	5	1	2	3	4	5
Ramka 1	1	1	1	1	1	1	5	5	5	5	4	4
Ramka 2		2	2	2	2	2	2	1	1	1	1	5
Ramka 3			3	3	3	3	3	3	2	2	2	2
Ramka 4				4	4	4	4	4	4	3	3	3

ofiarą staje się strona, która najdłużej przebywa w pamięci

autor: dr inż. Stanisława Plichta

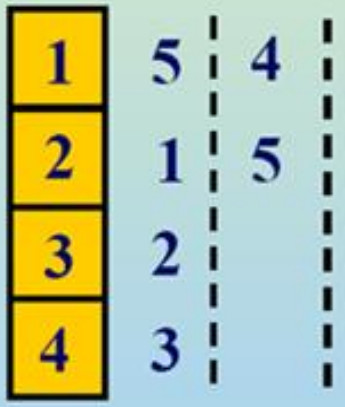
Algorytm FIFO zastępowania stron

- **Ciąg odniesień: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5.**
- **3 ramki** (3 strony mogą być w pamięci w tym samym czasie):



← **9 braków stron**

- **4 ramki:**



← **10 braków stron**

Anomalia Belady'ego

Anomalia Belady'ego polega na tym, że współczynnik braków stron wzrasta ze wzrostem liczby wolnych ramek

autor: dr inż. Anna Plichta

Algorytm optymalny

Chwila	1	2	3	4	5	6	7	8	9	10	11	12
Odwołanie	1	2	3	4	1	2	5	1	2	3	4	5
Ramka 1	1	1	1	1	1	1	1	1	1	1	4	4
Ramka 2		2	2	2	2	2	2	2	2	2	2	2
Ramka 3			3	3	3	3	3	3	3	3	3	3
Ramka 4				4	4	4	5	5	5	5	5	5

ofiarą staje się strona, która będzie nieużywana przez najdłuższy okres

Algorytm optymalny

- Przykład: **4 ramki, ciąg odniesień: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5;**

- Algorytm optymalny ma najniższy współczynnik braków stron ze wszystkich algorytmów, wolny od anomalii Belady'ego
- Jest trudny w realizacji, ponieważ wymaga wiedzy o przyszłej postaci ciągu odniesień (skąd ją wziąć?).
- Jest używany głównie w studiach porównawczych jako punkt odniesienia do innych algorytmów

Algorytm LRU

Chwila	1	2	3	4	5	6	7	8	9	10	11	12
Odwołanie	1	2	3	4	1	2	5	1	2	3	4	5
Ramka 1	1	1	1	1	1	1	1	1	1	1	1	5
Ramka 2		2	2	2	2	2	2	2	2	2	2	2
Ramka 3			3	3	3	3	5	5	5	5	4	4
Ramka 4				4	4	4	4	4	4	3	3	3

ofiarą strona, która nie była używana od najdłuższego czasu

Algorytm LRU

- Przykład: **4 ramki, ciąg odniesień: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5;**

- Lepszy od algorytmu FIFO (mniej braków stron).
- Wolny od anomalii Belady'ego - często stosowany.
- Trudność z zapamiętaniem historii użycia stron – może wymagać sporego zaplecza sprzętowego.

Implementacja algorytmu LRU

- **Za pomocą liczników** - każda strona ma wówczas specjalne pole licznika. Gdy jakiś proces odwołuje się do strony, do licznika kopiuje się stan zegara systemowego. Gdy trzeba wybrać ofiarę, szuka się strony z najmniejszą wartością licznika. Ustawianie licznika strony musi odbywać się sprzętowo.
- **Za pomocą stosu** - na stosie trzymamy numery stron, do których były odwołania. Odwołanie do strony powoduje przesunięcie jej numeru na wierzchołek tego stosu.

Algorytmy przybliżające LRU

Niewiele systemów posiada odpowiedni sprzęt umożliwiający realizację algorytmu LRU

Często stosowane są algorytmy przybliżające metodę LRU:

- algorytm bitów odniesień
- algorytm dodatkowych bitów odniesień
- algorytm drugiej szansy

Algorytm bitów odniesień

- Z każdą pozycją w tablicy stron związany jest bit odniesienia ustawiony początkowo na 0.
- Przy odwołaniu do strony jest on ustawiany na 1.
- Zastępowana jest ta strona w porządku FIFO, która ma bit odniesienia ustawiony na 0.

Algorytm dodatkowych bitów odniesień

- Z każdą stroną związany jest **8 bitowy** rejestr ustawiony na początek na **00000000**
- W regularnych odstępach czasu (np. co 100ms) SO wprowadza na najbardziej znaczącą pozycję rejestru bit odniesienia
- Wymieniana jest strona najdawniej używana - najmniejsza liczba w rejestrze **np. 0111010<1100010**
- Jeśli kilka stron ma taką samą wartość rejestru - wymiana wszystkich lub FIFO

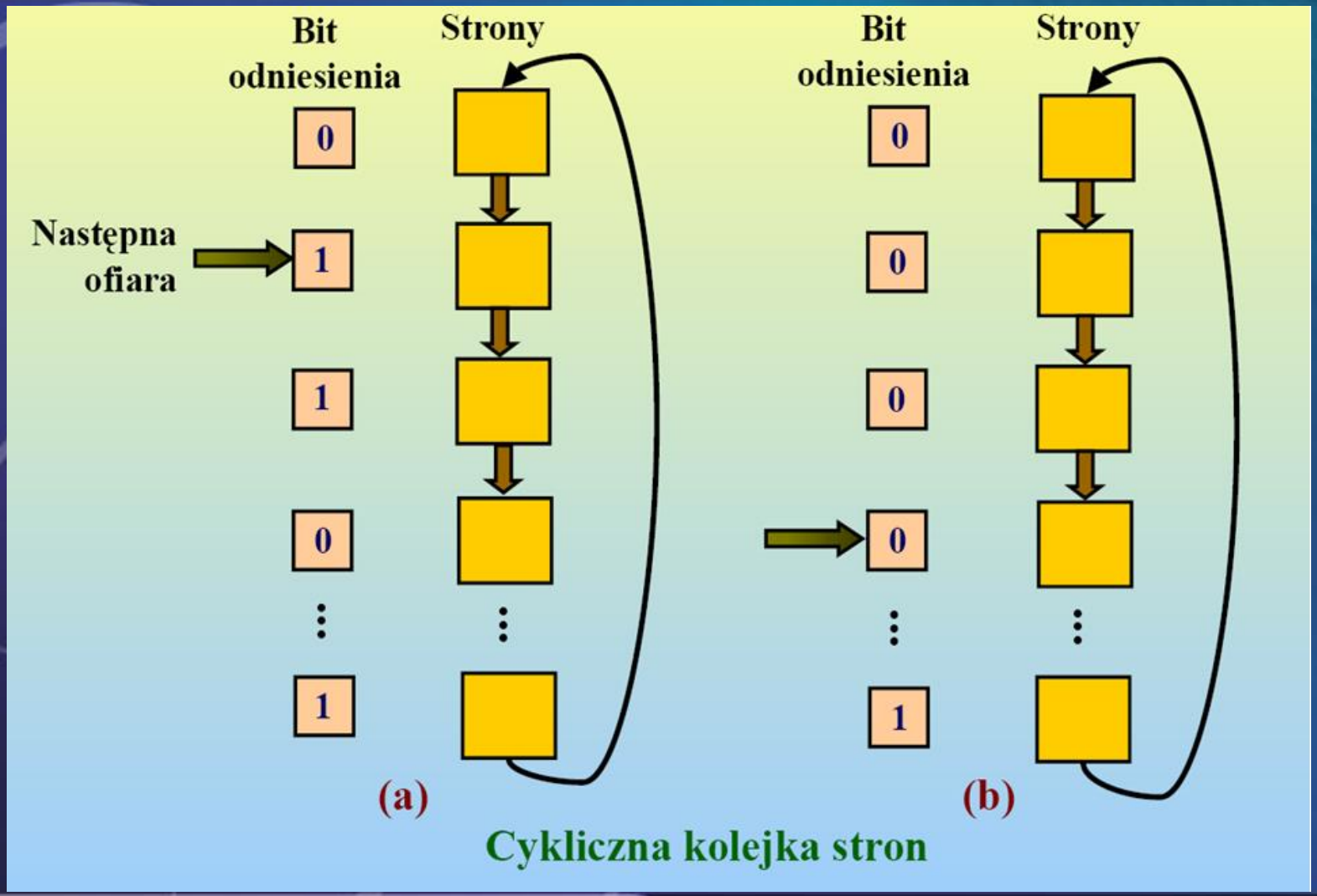
Algorytm drugiej szansy (zegarowy)

- Strony przeglądane są w porządku FIFO
- Sprawdzenie bitu odniesienia;

  jeśli 0 - strona zastąpiona;

  jeśli 1 - „druga szansa”
- **„druga szansa”** – zerowanie bitu odniesienia; ustawienie czasu bieżącego (koniec kolejki),
- Przewijanie stron dokonuje się cyklicznie
- Strona często eksploatowana - nigdy nie będzie zastąpiona

Algorytm drugiej szansy (zegarowy)



Ulepszony algorytm drugiej szansy

(0,0) – nieużywana ostatnio i niezmieniona – najlepsza do zastąpienia

(1,0) – używana ostatnio lecz niezmieniona – prawdopodobnie za chwilę będzie znów używana

(0,1) – nie używana ostatnio, ale zmieniona – nieco gorsza kandydatka, ponieważ stronę taką trzeba będzie zapisać na dysku przed zastąpieniem

(1,1) – używana ostatnio i zmieniona – prawdopodobnie znów będzie potrzebna, w przypadku zastępowania należy ją przekopiować z pamięci

Algorytmy zliczające

- Algorytmy zliczające korzystają ze związanego z każdą stroną licznika odwołań.
- Licznik zrealizowany sprzętowo - odwołanie do danej strony powoduje zwiększenie jej licznika o jeden.
- Istnieją dwa algorytmy oparte na przeciwstawnych założeniach:
  - Least Frequently Used (LFU) - ofiarą staje się strona, do której było najmniej odwołań (najrzadziej używana strona) – może być obarczony błędami wynikającymi z faktu, ze strona na początku była intensywnie używana a później wcale nie była potrzebna (realizacja – przesuwanie liczników w prawo o jeden bit w regularnych odstępach czasu)
  - Most Frequently Used (MFU) - ofiarą staje się strona, do której było najwięcej odwołań (najczęściej używana strona).

Implementacja tych algorytmów jest kosztowna i nie przybliżają one dobrze algorytmu optymalnego

Algorytmy buforowania stron

Procedury wspomagające:

- przechowywanie puli wolnych ramek

  zanim strona-ofiara zostanie usunięta potrzebna strona czytana do wolnej ramki z puli

- przechowywanie listy zmienionych stron; zapis zmienionych stron na dysk przez urządzenie stronicujące w wolnym czasie
- pula wolnych ramek + informacja o tym jaka strona rezydowała w każdej ramce (możliwość ponownego jej wykorzystania)

Algorytmy przydziału ramek

- W systemie wieloprogramowym ramki są współdzielone przez kilka procesów - procesy rywalizują o te ramki
- Istnieje pewna minimalna liczba ramek, które muszą być przydzielone procesowi – zdefiniowana przez architekturę logiczną komputera
- Liczba ramek musi być wystarczająca do zaalokowania wszystkich stron do których może odnosić się pojedynczy rozkaz

Schematy przydziału ramek:

- *przydział stały*
  - *równy* – każdy proces dostaje tyle samo ramek.
  - *proporcjonalny – liczba ramek proporcjonalna do jego rozmiaru*.
- *przydział priorytetowy*

Algorytmy przydziału ramek

- Zastępowanie globalne – proces może wybierać ramki do zastąpienia ze zbioru wszystkich ramek – *jeden proces może odebrać ramkę drugiemu*
- Zastępowanie lokalne – proces może wybierać ramki tylko z własnego zbioru przydzielonych ramek – *gorsza przepustowość, rzadziej stosowany*

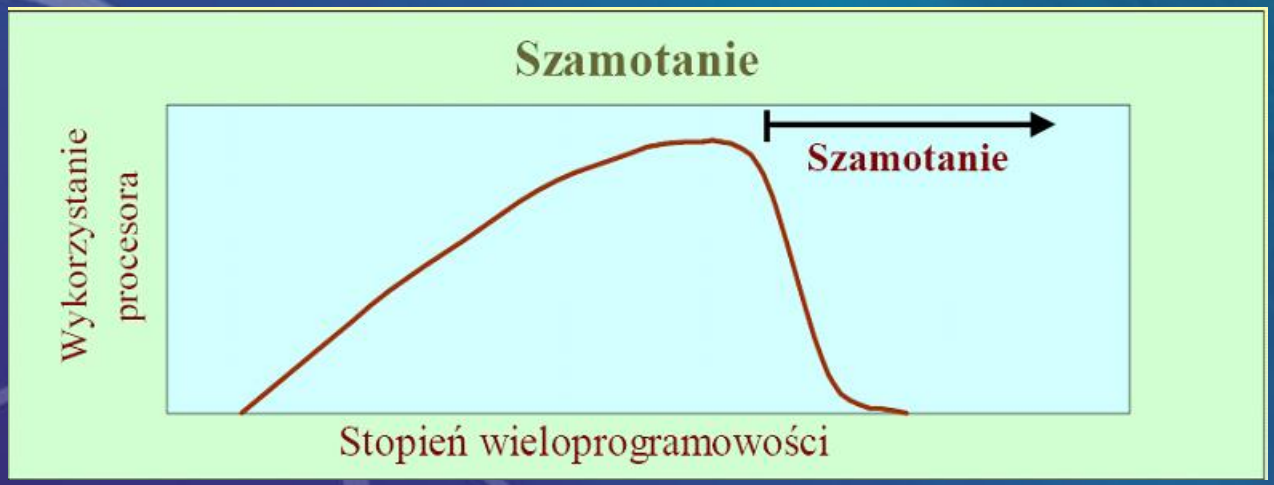
Szamotanie

Jeśli proces nie ma wystarczającej liczby stron w pamięci operacyjnej to częstość braków stron będzie wysoka

Prowadzi to do:

- zmniejszenia wykorzystania procesora
- mniejsze wykorzystanie procesora może być sygnałem dla planisty do zwiększenia wieloprogramowości
- kolejny proces zostaje załadowany do systemu pogarszając sytuację – zwiększa liczbę braków stron
- wykorzystanie procesora się zmniejsza co jest sygnałem dla planisty do dalszego zwiększania wieloprogramowości
- Powstaje szamotanie – *przepustowość systemu gwałtownie spada*

Szamotanie



szamotanie to zmniejszenie wykorzystania procesora przy zwiększeniu stopnia wieloprogramowości (wykonywanie przesyłania pomiędzy pamięcią operacyjną a dyskiem)

Szamotanie

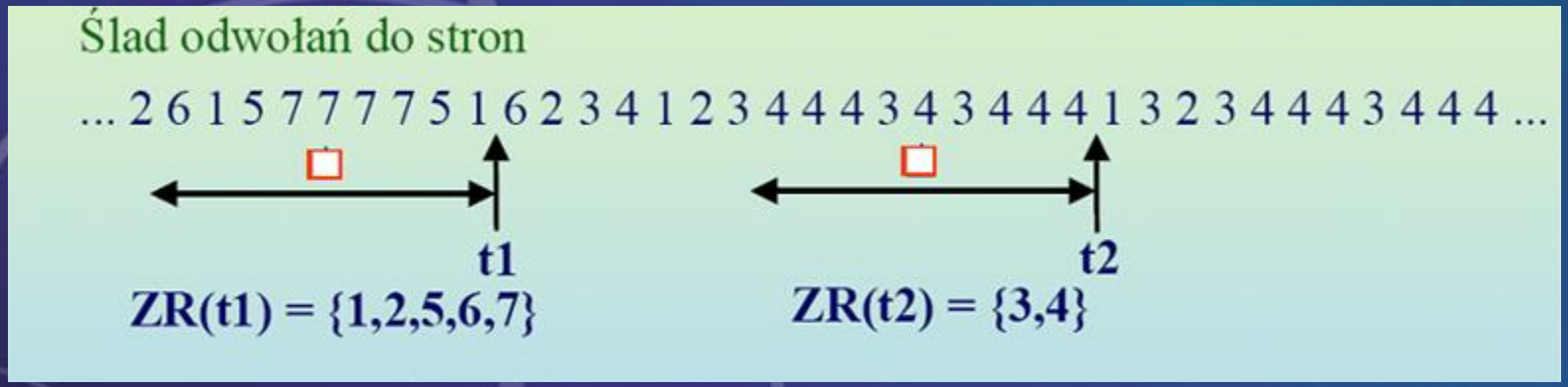
- Aby zapobiec szamotaniu należy dostarczyć procesowi tyle ramek ile potrzebuje – ***jak się o tym dowiedzieć?***
- Aby ograniczyć szamotanie należy:
  - zastosować lokalny lub priorytetowy algorytm zastępowania stron
  - dostarczyć procesowi właściwą liczbę ramek - strategia tworzenia zbioru roboczego **model strefowy wykonania procesu**

Model zbioru roboczego

- ***Strefa programu*** – zbiór stron pozostających we wspólnym użyciu.
- Program składa się z wielu stref, które mogą na siebie zachodzić.
- Program w trakcie wykonania przechodzi od strefy do strefy.
- Aby uniknąć szamotania należy przydzielić procesowi taka liczbę ramek, aby mógł w nim pomieścić swoją ***bieżącą strefę.***

Model zbioru roboczego

- ***Model zbioru roboczego*** opiera się na założeniu, że program ma charakterystykę strefową (lokalność odwołań).
- ***Okno zbioru roboczego*** □ to ustalona liczba odwołań do stron.
- ***Zbiór roboczy*** to zbiór stron do których nastąpiło □ ostatnich odwołań.



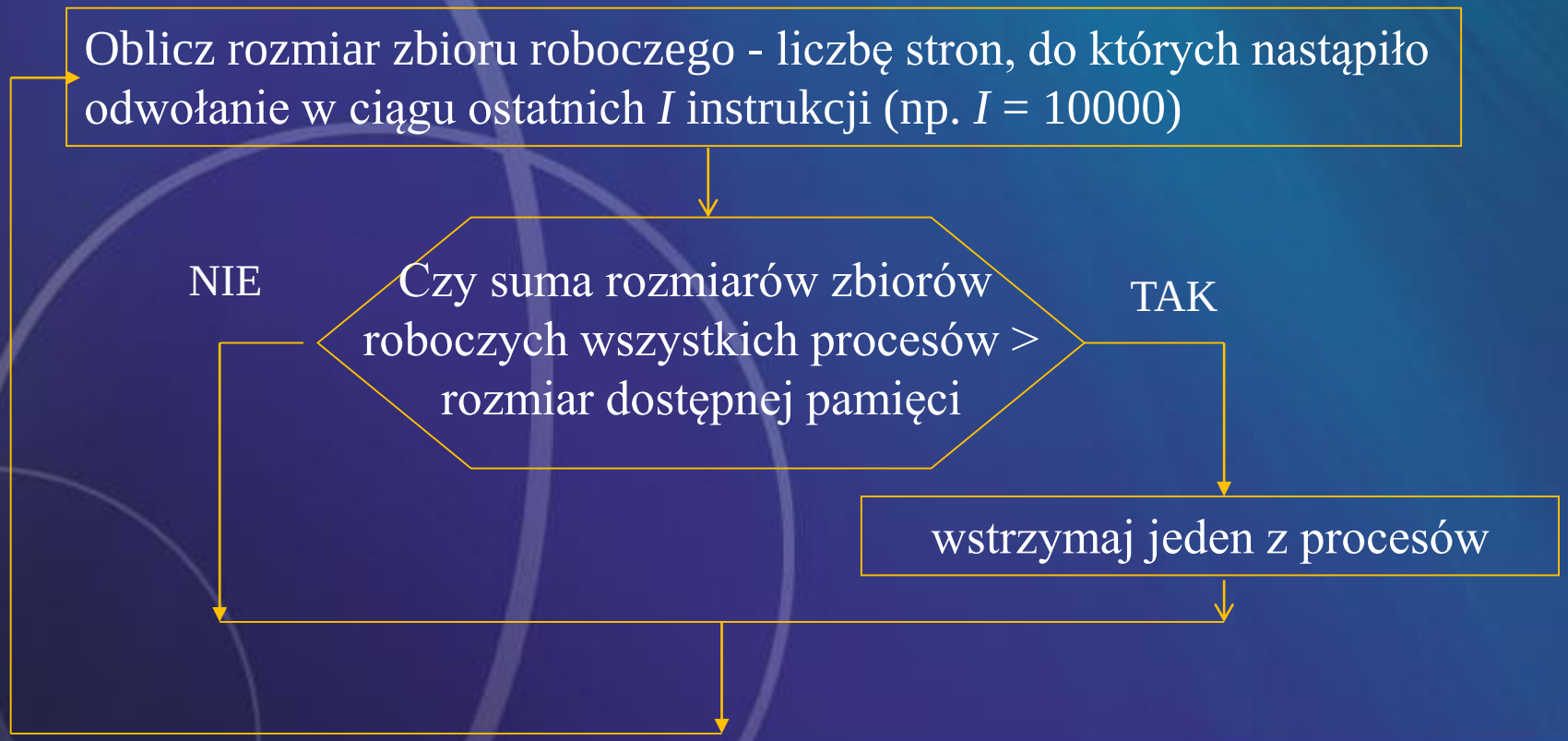
RZR_i – rozmiar zbioru roboczego i-tego procesu

Z – całkowite zapotrzebowanie na ramki $Z=\Sigma RZR_i$

Szamotanie powstaje gdy Z> liczba dostępnych ramek

Zbiory robocze procesów

Strategia zbioru roboczego zapobiega szamotaniu i utrzymuje stopień wieloprogramowości na wysokim poziomie – optymalizuje wykorzystanie procesora



Częstość braków stron

- Model zbioru roboczego daje dobre rezultaty, jednak nie jest wygodną metodą nadzorowania szamotania.
- Prostszym sposobem jest mierzenie częstości braków stron.
  - Ustala się dolną i górną granicę częstości braków stron.
  - Jeśli proces przekracza górną granicę, przydziela mu się dodatkową ramkę (w przypadku niedoboru ramek można wstrzymać jakiś proces).
  - Jeżeli częstość braku stron procesu spada poniżej dolnej granicy, odbiera mu się ramkę.