Zadanie 3
Badanie algorytmów zastępowania stron.
Należy samodzielnie sformułować założenia symulacji:
- rozmiar pamięci wirtualnej (ilość stron).
- rozmiar pamięci fizycznej (ilość ramek).
- długość (powinna być znaczna - min. 1000) i sposób generowania ciągu odwołań do stron (koniecznie uwzględnić zasadę lokalności odwołań).
Działanie programu:
- wygenerować losowy ciąg n odwołań do stron
- dla wygenerowanego ciągu podać liczbę błędów strony dla różnych algorytmów zastępowania stron:
1. FIFO (usuwamy stronę najdłużej przebywającą w pamięci fizycznej)
2. OPT (optymalny - usuwamy stronę, która nie będzie najdłużej używana)
3. LRU (usuwamy stronę, do której najdłużej nie nastąpiło odwołanie)
4. aproksymowany
5. RAND (usuwamy losowo wybraną stronę)
- symulacje przeprowadzić (na tym samym ciągu testowym) dla różnej liczby ramek (np. kilku (3, 5, 10?) wartości podanych przez użytkownika)
Zakres materiału: wszystko o pamięci wirtualnej (z wykładu).

Systemy operacyjne – zadanie 3

Uwagi ogólne:

Zadanie 3 – badania algorytmów zastępowania stron

Analogicznie do wcześniejszych zadań przypominam, że równie istotne co implementacja samych algorytmów jest przygotowanie środowiska symulacyjnego, które umożliwi nam wiarygodne weryfikację działania opracowanych algorytmów w ramach różnych, generowanych losowo, przypadków testowych. Tak jak wcześniej, istotna jest możliwość sterowania parametrami symulacji.

W ramach przygotowania środowiska testowego należy przygotować strukturę danych reprezentującą pamięć fizyczną (najprawdopodobniej tablicę), w której będziemy przechowywać aktualnie znajdujące się strony. Dodatkowo należy zdefiniować całkowitą liczbę stron w pamięci wirtualnej (najprawdopodobniej pojedynczą liczbę całkowitą), która będzie wykorzystywana do wygenerowania ciągu odwołań do strony z pamięci fizycznej. Istotną kwestią dotyczącą wygenerowanego ciągu odwołań jest uwzględnienie zasady lokalności odwołań. Zgodnie z tą zasadą, proces w każdej fazie realizacji korzysta jedynie z ograniczonego podzbioru wszystkich stron, w związku z tym generując ciąg odwołań należy zadbać, aby wygenerowany ciąg był podzielony na pewne sekcje (odpowiadające fazom realizacji procesów), w ramach których losowane będą jedynie strony z ograniczonego podzbioru wszystkich stron.

Parametrem badanym w ramach prowadzonej symulacji będzie liczba błędów stron. Jednocześnie przypomnę krótko, że **błąd strony** występuje w momencie, gdy proces odwołuje się do strony, która **nie znajduje się w pamięci fizycznej** i konieczne jest zastąpienie jednej ze stron znajdującej się w pamięci fizycznej stroną z pamięci wirtualnej.

W ramach prowadzonych badań algorytmów proszę pamiętać o przyjmowaniu założeń dotyczących parametrów symulacji, które będą rozsądne. Jako przykład mogę podać sytuację, w której pamięć fizyczna będzie większa niż całkowita liczba stron w pamięci wirtualnej – poświęcając chwilę na

przeanalizowanie takiego przypadku bez wątpienia każdy z Państwa stwierdzi, że jego badanie nie jest konieczne dla określenia wyników, które zostaną uzyskane.

Prowadząc badania algorytmów sugeruję testowanie różnej liczby ramek (rozmiarów pamięci fizycznej) rozpoczynając od bardzo małych liczb i stopniowo je zwiększając, w taki sposób aby stopniowo pojemność pamięci fizycznej stawała się coraz większą częścią całkowitej liczby stron zdefiniowanych w pamięci wirtualnej.

Podobnie jak w przypadku wcześniejszych zajęć przypominam, że zależy nam na wiarygodnej symulacji i istotne jest, **aby każdy z algorytmów operował na tym samym ciągu odwołań**, aby możliwe było ich porównanie.

W ramach prowadzonych badań ma zostać przeprowadzone porównanie 5 algorytmów. Krótkie wprowadzenie do każdego z nich zostało przedstawione poniżej, natomiast w celu poznania szczegółów ich działania odsyłam Państwa do literatury oraz do materiałów z wykładu.

1) FIFO (First In First Out)

Prosty w implementacji algorytm zastępowania stron, który wykorzystuję założenie, iż w przypadku wystąpienia błędu strony usunięta powinna zostać strona, która najdłużej przebywa w pamięci fizycznej. Zwracam uwagę, że algorytm ten nie uwzględnia tego, czy pojawiły się odwołania do poszczególnych stron, po prostu usuwa tę stronę, która jest najdłużej w pamięci.

W związku z powyższym w ramach praktycznej implementacji dla każdej strony przechowywanej w pamięci fizycznej powinien zostać zdefiniowany licznik, który będzie zliczał jak długo znajduje się ona w pamięci (dla uproszczenia można przyjąć, że jedno obsługiwane odwołanie to jeden kwant czasu).

Poniżej zamieszczony został przykład działania algorytmu w systemie, w którym pamięć fizyczna składa się z **4** ramek, dla ciągu zapytań: **[1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5]**

Moment w czasie	**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
Błąd strony	x	x	x	x			x	x	x	x	x	x

Liczba błędów strony: **10**

2) OPT (optymalny)

Jest to algorytm referencyjny, który zgodnie z założeniami powinien dawać najlepsze wyniki. Zasada jego działania opiera się na usuwaniu stron, które najdłużej nie będą używane. Ze względu na fakt, iż do tej pory nie opracowano niezawodnej metody przewidywania przyszłości, możliwości jego implementacji są ograniczone, niemniej stanowi on dobry wyznacznik dla porównania efektywności działania innych algorytmów.

Poniżej zamieszczony został przykład działania algorytmu w systemie, w którym pamięć fizyczna składa się z **4** ramek, dla ciągu zapytań: **[1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5]**

Moment w czasie	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
Błąd strony	x	x	x	x			x				x	

Liczba błędów strony: **6**

3) LRU (Least Recently Used)

Algorytm LRU w pewnym sensie jest odwrotnością algorytmu optymalnego, jeżeli chodzi o koncepcję jego działania. W przypadku wystąpienia błędu strony algorytm ten usuwa z pamięci fizycznej stronę, która nie była używana przez najdłuższy okres. W tym przypadku przy okazji implementacji algorytmu należy zwrócić uwagę na konieczność zliczania jak długo każda ze stron znajdujących się z pamięci fizycznej nie była używana. Podobnie jak w przypadku algorytmu FIFO można przyjąć, że jedno obsługiwane odwołanie to jeden kwant czasu.

Poniżej zamieszczony został przykład działania algorytmu w systemie, w którym pamięć fizyczna składa się z **4** ramek, dla ciągu zapytań: **[1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5]**

Moment w czasie	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
Błąd strony	x	x	x	x			x			x	x	x

Liczba błędów strony: **8**

4) aproksymowany LRU

Ze względu na konieczność śledzenia odwołań do poszczególnych stron praktyczna implementacja algorytmu LRU w systemach informatycznych nie jest zadaniem prostym i dla efektywnego działania wymaga dedykowanych rozwiązań sprzętowych umożliwiających jego efektywną realizację. Z tego powodu klasyczny algorytm LRU jest najczęściej zastępowany algorytmami przybliżającymi (aproksymującymi) jego działanie. Przykładem takiego przybliżenia może być algorytm, który każdą ze stron kojarzy z odpowiadającym jej bitem odwołania. Wartość tego bitu jest ustawiana na 1 przy każdym odwołaniu do strony, natomiast algorytm wybiera do zastąpienia jedną ze stron dla których bit ten ma wartość 0.

Przykładem algorytmu aproksymującego działanie algorytmu LRU jest algorytm drugiej szansy. Algorytm w swoim działaniu wykorzystuje kolejkę FIFO. Każda kolejna strona, która jest wczytywana do pamięci jest dodawana na koniec kolejki FIFO, a jej bit odwołania jest ustawiany na 1. W

momencie gdy występuje błąd strony pierwszym kandydatem do usunięcia jest strona znajdująca się na początku kolejki.

W przypadku, gdy strona znajdująca się na początku kolejki ma bit odwołania ustawiony na 0 zostaje ona od razu zastąpiona przez nową stronę, do której pojawiło się odwołanie, i jest ona usuwana z kolejki, natomiast nowa strona, zgodnie z wcześniejszym opisem, jest wstawiana na koniec kolejki z bitem odwołania ustawionym na 1.

Alternatywnie, jeżeli okaże się, że strona na początku kolejki ma bit odwołania ustawiony na 1, to, zgodnie z nazwą algorytmu, dostaje ona drugą szansę. Nie jest ona usuwana z pamięci, jednak jej bit odwołania jest ustawiany na 0, a strona ta zostaje przeniesiona na koniec kolejki. Następnie algorytm analizuje kolejną stronę w kolejce (która teraz jest na jej początku) i analogicznie strona ta dostaje drugą szansę, jeżeli jej bit odwołania jest równy 1, natomiast jeżeli jest on równy 0, to zostaje ona od razu usunięta z pamięci i jest zastępowana przez nową stronę.

Poniżej zamieszczony został przykład działania algorytmu w systemie, w którym pamięć fizyczna składa się z **4** ramek, dla ciągu zapytań: **[1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5].** Dodatkowo, w ostatnim wierszu przedstawiona została zawartość kolejki FIFO (pierwszy element od lewej, to początek kolejki) i wartość bitu odwołania (w dolnym indeksie) dla każdej ze stron po zakończeniu przetwarzania nadchodzącego odwołania do strony w pamięci fizycznej.

Moment w czasie	**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
Błąd strony	x	x	x	x			x	x	x	x	x	x
Kolejka FIFO	1_1	$1_1,2_1$	$1_1,2_1,3_1$	$1_1,2_1,3_1,4_1$	$1_1,2_1,3_1,4_1$	$1_1,2_1,3_1,4_1$	$2_0,3_0,4_0,5_1$	$3_0,4_0,5_1,1_1$	$4_0,5_1,1_1,2_1$	$5_1,1_1,2_1,3_1$	$1_0,2_0,3_0,4_1$	$2_0,3_0,4_1,5_1$

Liczba błędów strony: **10**

Moment w czasie	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Odwołanie	1	2	3	4	1	2	5	3	2	1	4	5
Ramka 1	**1**	1	1	1	1	1	**5**	5	5	5	5	5
Ramka 2		**2**	2	2	2	2	2	2	2	2	**4**	4
Ramka 3			**3**	3	3	3	3	3	3	3	3	3
Ramka 4				**4**	4	4	4	4	4	**1**	1	1
Błąd strony	x	x	x	x			x			x	x	
Kolejka FIFO	1_1	$1_1,2_1$	$1_1,2_1,3_1$	$1_1,2_1,3_1,4_1$	$1_1,2_1,3_1,4_1$	$1_1,2_1,3_1,4_1$	$2_0,3_0,4_0,5_1$	$2_0,3_1,4_0,5_1$	$2_1,3_1,4_0,5_1$	$5_1,2_0,3_0,1_1$	$3_0,1_1,5_0,4_1$	$3_0,1_1,5_1,4_1$

Liczba błędów strony: **8**

5) RAND (losowy)

Jak można się spodziewać jest to najprostszy algorytm do implementacji. W momencie pojawienia się błędu strony usuwamy losową stronę z pamięci fizycznej i zastępujemy ją tą, do której wystąpiło odwołanie. Ze względu na niedeterministyczny charakter tej metody zastępowania stron przykład jej działania został pominięty.