WYDZIAŁ MATEMATYKI, INFORMATYKI I MECHANIKI
UNIWERSYTET WARSZAWKI

Zarządzanie Pamięcią Wirtualną w Windows NT

Referat na zajęcia:
„Systemy Operacyjne”
Przygotował:
Edwin Vaina **171 278**

WARSZAWA 2002

1. Historyczny wstęp.

Pierwsze komputery, były to maszyny raczej jedno procesorowe obsługujące co najwyżej jeden proces. System operacyjny stanowił sam użytkownik, gdyż to on był odpowiedzialny za ładowanie programów do pamięci (np. za pośrednictwem przełączników czy taśm perforowanych). Po załadowaniu programu należało podać adres startowy, skąd komputer miał zacząć czytać kod programu. Załadowanie i wykonanie więcej niż jednego programu jednocześnie nie było wtedy możliwe.

Rozwiązanie to powodowało, że procesory przez większość czasu działania pozostawały bezczynne. Postęp w technologii systemów operacyjnych oznaczał znalezienie sposobów na „zajęcie" procesorów pracą i wykonanie w ten sposób większej ilości pracy w określonym czasie. Powstały dzięki temu systemy wielozadaniowe i zrodził się od razu problem zarządzania pamięcią operacyjną systemu.

We wczesnym okresie rozwoju komputerów niemożliwe było uruchomienie programu, który w całości nie mieściłby się w pamięci operacyjnej. Później zaczęto tworzyć tzw. *overlays,* czyli programy, które potrafiły zapisać nieużywaną część kodu na dysku i wczytać do pamięci podręcznej potrzebny kawałek kodu z dysku. Programy te miały jednak wiele wad. Oprócz tego, że były trudne w implementacji, obsłudze i uaktualnianiu, wymagały od każdej aplikacji tego, aby jej kod uwzględniał partycjonowanie jej na części, które byłyby wczytywanie do pamięci, a to oznaczało przebudowę już istniejących programów.

Pamięć Wirtualna PW *Virtual Memory,* pierwszy raz zaimplementowana w roku 1959, zdjęła z barków programisty konieczność zarządzania pamięcią i obarczyła nią system operacyjny. PW jest to scentralizowany systemem odpowiedzialnym za zrzucanie zawartości pamięci operacyjnej na dysk, gdy jest ona pełna. Umożliwia tworzenie i uruchamianie programów, które wymagają więcej pamięci niż posiada używany do tego celu komputer. Obecnie stanowi już standard w technice zarządzania pamięcią we wszystkim współczesnych systemach operacyjnych.

Za zarządzanie pamięcią w systemach klasy Windows NT odpowiedzialny jest tzw. *VM manager* dalej zwany **VMM.** *VM Manager* został zaprojektowany i zaimplementowany przez Lou Perazzoli, głównego inżyniera i szefa projektu NT. Pod czas prac nad nim Perazzoli razem z pozostałymi członkami zespołu ustalili, że *VM Manager* powinien spełnić następujące warunki:

- musi być przenaszalny z jednego systemu na inny
- musi pracować stabilnie i efektywnie z aplikacjami różnej wielkości bez dokonywania zmian w systemie prze użytkownika czy administratora.
- musi udostępniać procesom możliwość alokowania i korzystania prywatnej części pamięci operacyjnej
- musi udostępniać mechanizm współpracy z pewnym *Environment subsystem,* umożliwiając mu między innymi udostępnianie procesowi klienckiemu korzystanie z pamięci wirtualnej (na ustalonych prawach).
- musi „równoważyć" stopień wieloprocesorowości z szybkością dostępu do pamięci. (Dla przykładu: ochrona struktur danych przez zastosowanie różnego rodzaju mechanizmów synchronizujących dostęp „podnosi stopień współbieżność", ale z drugiej strony powoduje większe opóźnienia w dostępie do pamięci)

2 Pamięć Wirtualna

Pamięć posiada pewną strukturę fizyczną, jednak system operacyjny widzi jakąś strukturę logiczną. Aby mógł poprawnie działać, musi potrafić tłumaczyć adresy fizyczne określające pewien obszar pamięci na logiczne. **Pamięć fizyczna** jest widziana jako ciąg jednobajtowych jednostek numerowanych od zera do rozmiaru pamięci (minus 1).

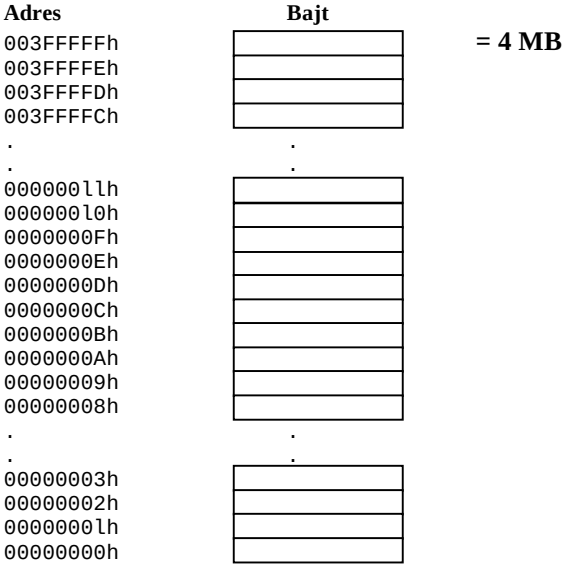


Figure 6-1. Fizyczna przestrzeń adresowa.

Pamięć logiczna określana jako **pamięć wirtualna**, jest to sposób, w jaki pamięć postrzega program komputerowy (w nowoczesnych systemach operacyjnych w małym stopniu koresponduje z fizyczną strukturą pamięci). Systemy pamięci wirtualnej postrzegają pamięć w postaci segmentowej jak i liniowej. Wczesne procesory oparte na technologii Intel, od rodziny 8086 do 80286 bazowały na segmentowym modelu pamięci. Model ten polega na podzieleniu fizycznej przestrzeni adresowej na pewne segmenty. Adres komórki w tym przypadku zawiera numer segmentu oraz tzw. offset w ramach segmentu. W procesorach typu RISC (w najnowszych CISC Intela również) zastosowano koncepcje adresowania liniowego. Adresowanie liniowe polega na operowaniu rzeczywistymi adresami komórek w pamięci.

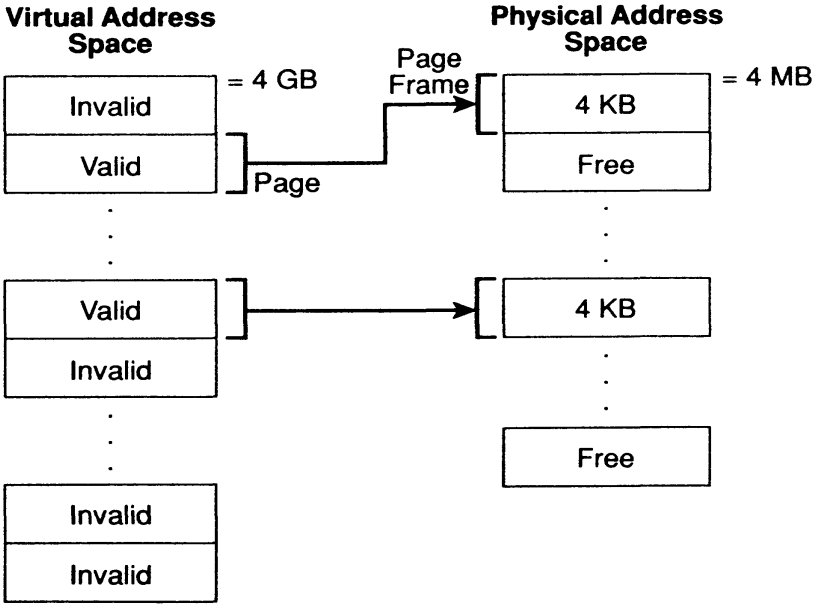
Wirtualna przestrzeń adresowa jest to zbiór adresów pamięci dostępny dla procesów. Każdy proces uzyskuje unikatową wirtualną przestrzeń adresową często o wiele większą niż przydzielona mu pamięć fizyczna. Mimo, że ilość adresów fizycznych jest ograniczona przez wielkość pamięci, to ilość adresów wirtualnych jest ograniczona co najwyżej przez ilość bitów je opisujących. Tak więc 32-bitowa przestrzeń adresowa potrafi opisać czterogigabajtową pamięć wirtualną.

Z rozbieżności między fizyczną a wirtualną przestrzenią adresową wynikają dwa podstawowe zadania dla systemu zarządzania pamięcią wirtualną:

- tłumaczenie adresów wirtualnych na fizyczne. Gdy proces próbuje zapisać lub odczytać coś z pamięci system używa adresu wirtualnego, aby znaleźć odpowiednie miejsce w pamięci fizycznej (mechanizm ten może być zrealizowany sprzętowo).
- zrzucenie części pamięci na dysk gdy zapotrzebowanie na pamięć zgłaszane przez procesy i system jest większe niż jest dostępne.

Dzięki możliwości mapowania adresów wirtualnych na fizyczne programy w łatwy sposób mogą być realokowane w pamięci podczas wykonywania. Pamięć wirtualna bardzo

ułatwia przenoszenie części programów z pamięci na dysk i odwrotnie np. w inne już miejsce. W przypadku takim następuje uaktualnienie informacji o tym gdzie dany program się znajduje. Oczywiście niemożliwe jest, aby dany proces mógł zaadresować np. aż 4 GB pamięci, podczas gdy fizycznie dostępne jest tylko 4 MB. Umożliwić to ma system pamięci wirtualnej, który używa dysku twardego jako pamięci podręcznej. System ten wybiera zgodnie z pewnymi zasadami te dane, które mają być zrzucone na dysk i zapisuje je postaci pliku. Kiedy jakiś proces odwołuje się do tych danych, system z powrotem wczytuje je z dysku do pamięci. Przemieszczanie danych z dysku do pamięci i odwrotnie bajt po bajcie byłoby bardzo nieefektywne, dlatego wirtualna przestrzeń adresowa jest podzielona na równe bloki zwane stronami *pages*. W taki sam sposób pamięć fizyczna jest dzielona na bloki tej samej wielkości zwane ramkami *page frames,* do których są zapisywane poszczególne strony. Każdy proces posiada pewną liczbę stron w wirtualnej przestrzeni adresowej. Strony, które aktualnie znajdują się w pamięci fizycznej i są natychmiast dostępne zwane są *valid pages.* Strony składowane na dysku nazywane są *invalid pages.*



Kiedy wykonujący się program odwoła się do strony oznaczonej „invalid" procesor generuje przerwanie zwane przerwaniem braku strony (*page fault trap).* System pamięci wirtualnej lokalizuje wymaganą stronę i wczytuje ją z dysku do wolnej ramki. Gdy ilość wolnych ramek w pamięci jest „mała" system wybiera odpowiednie strony i zrzuca je na dysk. Proces ten jest dla programisty niezauważalny.

Rozmiar strony jest najczęściej potęgą dwójki i często zależy od sprzętu. Windows NT uznaje wielkość strony, jaka jest „wszyta" w strukturę procesora Intel 386 a jest to 2^{12}=4KB. Procesor MIPS R4000 pozwala na programowe ustalanie wielkości strony.

Oprócz tłumaczenia adresów wirtualnych na logiczne oraz transferu danych między pamięcią i dyskiem system zarządzania pamięcią wirtualną zajmuje się też:

- umożliwianiem współdzielenia pamięci fizycznej przez różne procesy.
- ochroną prywatnych i współdzielonych obszarów pamięci przed nieuprawnionym odwołaniem.
- jeśli jest uruchomiony w systemie wieloprocesorowym musi też radzić sobie z sytuacjami, gdy dwa procesy odwołują się do pewnego obszaru pamięci w tym samym czasie.

Opisowi jak z tymi problemami radzi sobie system pamięci wirtualnej NT poświęcona jest dalsza część tego referatu.

3 Tryb użytkownika.

NT VMM dostępnia wiele funkcji procesom działającym w trybie użytkownika. Procesy systemowe *Environment subsystems* korzystają z usług udostępnianych przez VMM do zarządzania procesami klienckimi. Procesy uruchomione w trybie użytkownika mają możliwość współdzielenia pewnego obszaru pamięci. Mogą ustalić typ ochrony prywatnego obszaru pamięci, mogą również zablokować do nich dostęp. Mają także możliwość korzystania z otwartych plików podmapowanych pod ich wirtualną przestrzeń adresową.

3.1 Zarządzanie pamięcią.

VM Manager udostępnia procesom następujące usługi pomagające bezpośrednio zarządzać ich wirtualną przestrzenią adresową:

- alokowanie pamięci dla procesu,
- odczyt i zapis do pamięci wirtualnej
- blokada strony w pamięci fizycznej
- uzyskanie informacji o stronach
- ochrona stron
- zrzucenie określonych stron na dysk

Alokacja pamięci przez VM Manager odbywa się w dwóch fazach: rezerwowanie *reserving* i zapełnianie *committing*. Pamięc zarezerwowana *Reserved memory* jest to pewien zbiór adresów wirtualnych zarezerwowanych dla procesu, który będzie chciał ich użyć w przyszłości. Rezerwowanie jest operacją szybką i tanią w Windows NT. Pamięć zapełniona *Committed memory* jest to pamięć fizycznie zajęta przez dane procesu. VMM może rezerwować pamięć od razu przy zapełnianiu, lub zarezerwować i zapełnić ją, kiedy wystąpi taka potrzebna. Rezerwowanie jest użyteczne, kiedy proces korzysta z dynamicznych struktury danych. Proces rezerwuje pewne obszary pamięci i korzysta z nich tylko wtedy, kiedy są potrzebne. Kiedy struktura danych zaczyna rosnąć proces może ja zapisać w już zarezerwowanym obszarze pamięci. Strategia ta gwarantuje, że żaden inny proces (np. Win32 subsystem thread) nie zajmie pamięci, która będzie wykorzystana do zapamiętania dynamicznie rozrastającej się struktury danych.

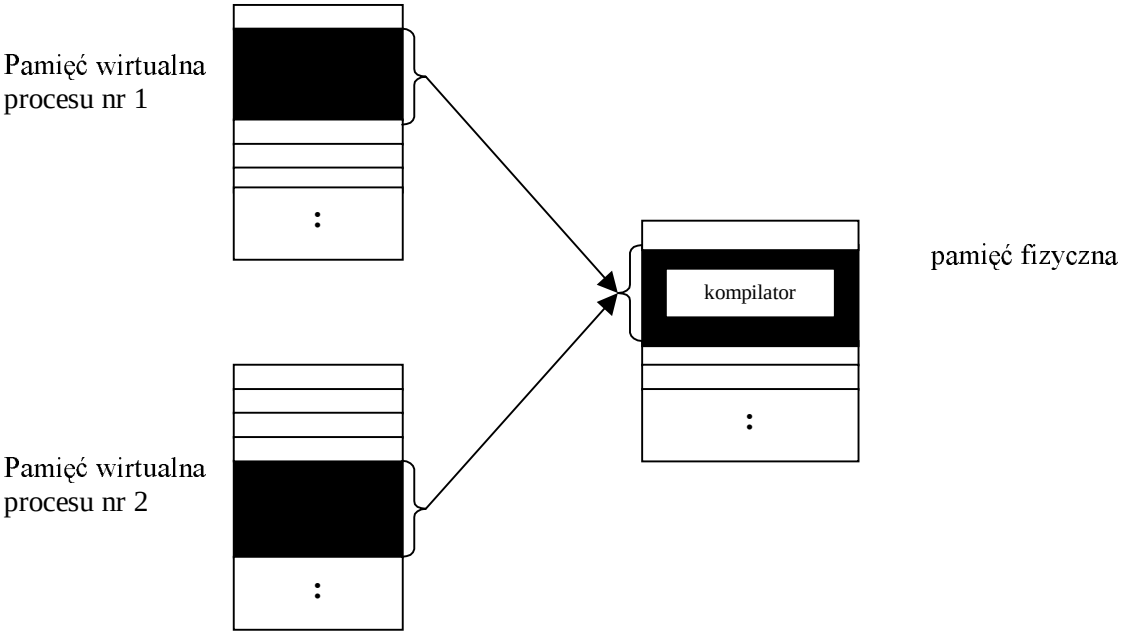
VM Manager pozwala aplikacjom lub innym procesom posiadającym specjalne prawa blokować wyszczególnione strony w pamięci. To gwarantuje, że dana strona nie zostanie usunięta z pamięci, podczas gdy w określonym procesie działa choć jeden wątek. Dla przykładu, system zarządzania bazą danych, który używa drzewa jako struktury do indeksowania danych, może chcieć zablokować obszar pamięci, gdzie jest przechowywany korzeń tego drzewa po to, aby częste odwoływanie się do bazy nie generowało niepotrzebnych przerwań braku strony.

VMM umożliwia powiadamianie danego procesu o tym, że obszar pamięci należący do niego ma być zmodyfikowany przez inny proces. Dzięki temu jeden proces działający w trybie użytkownika może stworzyć inny proces, przydzielając sobie prawo do manipulowania obszarem pamięci wirtualnej należącej do nowego procesu. Daje to możliwość *subsystems* zarządzania obszarami pamięci należącymi do ich procesów klienckich. Proces *Win32* może ona alokować wolną pamięć wirtualną, czytać i zapisywać do pamięci wirtualnej, rzucać strony na dysk, uzyskać informację o rozmieszczeniu stron, blokować określone strony i chronić wyszczególnione strony. Żadna z tych funkcji nie jest dostępna dla procesu Win32, gdy obszary pamięci należą do innych procesów, oprócz przypadków, gdy uruchomiony jest tryb *ReadProcessMemoryO* lub *WriteProcessMemoryQ*. Tryby te zostały stworzone z myślą o wykorzystaniu ich w przypadku ustawiania pułapek *breakpoints* podczas „debugowania" danego procesu.

3.2 Współdzielenie Pamięci

Ważnym aspektem jest możliwość współdzielenia pewnych obszarów pamięci przez procesy. Oczywiście każdy z tych procesów posiada własną pamięć prywatną, gdzie składuje własne dane. Dla przykładu, jeśli kilka procesów kompiluje programy napisane w C, zużycie pamięci może być zminimalizowane, jeśli tylko jedna kopia kompilatora będzie rezydować w pamięci..

VMM udostępnia odpowiedni mechanizm współdzielenia pamięci. Ponieważ każdy proces ma przestrzeń adresów wirtualnych należących tylko do niego, kompilator może być załadowany tylko raz i gdy inny proces chce wykorzystać go, VM podmapowuje do przestrzeni adresowej procesu obszar pamięci fizycznej zwierający kompilator.



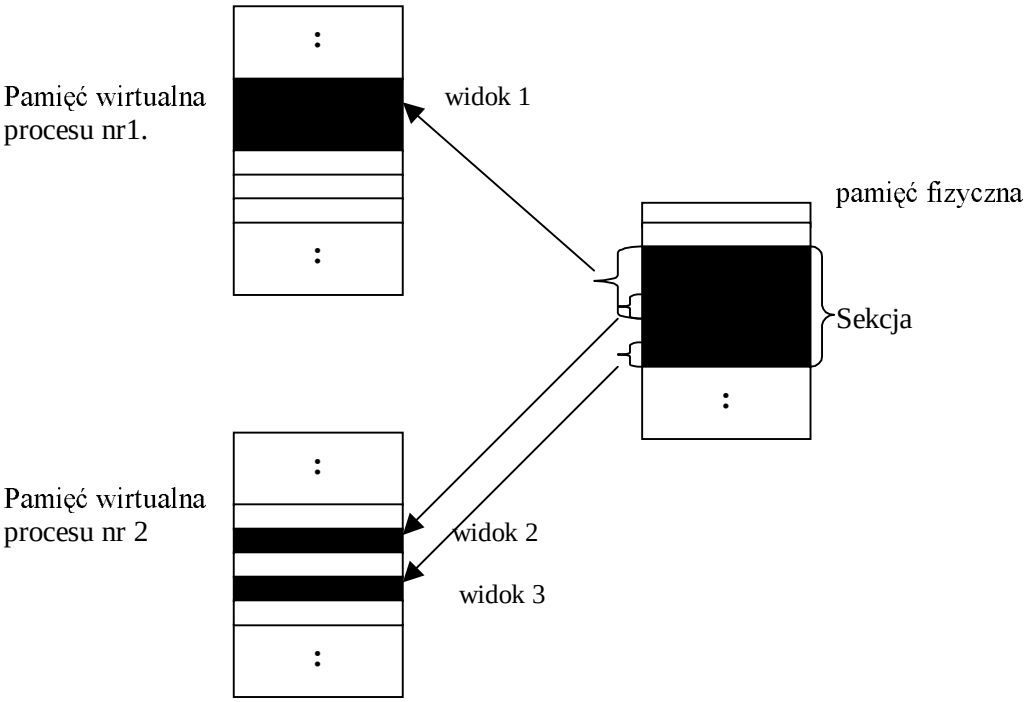
Podobnie w przypadku, gdy dwa procesy tworzą pewien bufor pamięci współdzielonej, do obszarów pamięci wirtualnej należących do nich zostają podmapowane strony pamięci zawierające tenże bufor. W przypadku z kompilatorem żaden z procesów nie ma prawa do modyfikacji obszaru pamięci zajętego przez tenże, gdyż dla nich jest ona ustawiona w trybie tylko-do-odczytu *read-only*. W przykładzie z buforem procesy mogą jednak chcieć mieć możliwość pisania do bufora. Wtedy używany jest tryb odczyt/zapis *read/write*. Oczywiście VMM musi dbać o to, aby operacje równoczesnego dostępu do danego obszaru pamięci były odpowiednio synchronizowane tak, aby nie doszło do rozspójnienia lub zniszczenia struktur danych znajdujących się we współdzielonym obszarze.

3.2.1 Sekcje, widoki i mapowanie plików *(Sections, Views, and Mapped Files)*

Podobnie jak inne komponenty systemu Windows NT, VMM jest całkowicie współbieżny. Może działać na wielu procesorach równocześnie i musi współdzielić swoje struktury danych z różnymi wątkami na nich uruchomionymi.

Pamięć współdzielona może być definiowana jako obszar pamięci widoczny dla wielu procesów. Jest ona obecna w kilku przestrzeniach adresów wirtualnych. W Windows NT każdy współdzielony zasób jest zaimplementowany jako chroniony obiekt i tak też jest w przypadku pamięci.

Obiekt Sekcji (section object) ma postać bloku pamięci, który może współdzielić kilka procesów. Wątek tworzy obiekt sekcja i zwraca jego nazwę tak, że inny wątek w innym procesie może go sobie otworzyć. Po otwarciu sekcji wątek, może zamapować całość lub część sekcji do swojej przestrzeni adresów wirtualnych. Obiekt sekcja w systemach NT może być bardzo duży rozmieszczony nawet na kilku tysiącach stron pamięci. Proces najczęściej potrzebuje tylko części Sekcji, którą nazywa się widokiem (część Sekcji widoczna dla procesu).



Różne procesy na swoje potrzeby mogą otwierać różne widoki, w szczególności jeden proces może otworzyć kilka widoków. Mapowanie widoków przez proces umożliwia im dostęp do bardzo dużych bloków pamięci, co byłoby nie możliwe z powodu zbyt małej wirtualnej przestrzeni adresowej. Dla przykładu, pewna firma posiada ogromną bazę danych o swoich pracownikach. Proces zarządzający bazą danych tworzy obiekt sekcja zwierający całą bazę danych. Kiedy dany program odwołuje się do bazy tworzony jest widok na sekcji z bazą w jego wirtualnej przestrzeni adresowej po odczytaniu potrzebnych danych jest odmapowany i mapowany następny w celu przeczytania kolejnych informacji. Ostatecznie program przegląda w ten sposób całą bazę danych bez wyczerpania całej wirtualnej przestrzeni adresowej.

Podobnie jak prywatne obszary pamięci także pamięć współdzielona może być zrzucana na dysk do tzw. pliku wymiany. Istnieje jednak możliwość zapisania sekcji do pliku tzw. *mapped file.* Takim plikiem jest na przykład wspomniana wcześniej baza pracowników. Program zarządzający bazą danych używa obiektu sekcja, aby zapisać zawartość bazy danych w pamięci wirtualnej. Program, który z niej korzysta ma dostęp do tego pliku przez mechanizm mapowania widoków na tymże obiekcie sekcji, więc czytania i pisania (są to operacje raczej na pamięci nie na fizycznym pliku). Jeśli program próbuje pisać do strony niedostępnej (znajdującej się na dysku) jest ona wczytywana, a po modyfikacji zmiany trafiają do pliku wtedy, gdy jest strona jest zrzucana na dysk zgodnie z mechanizmem VMM.

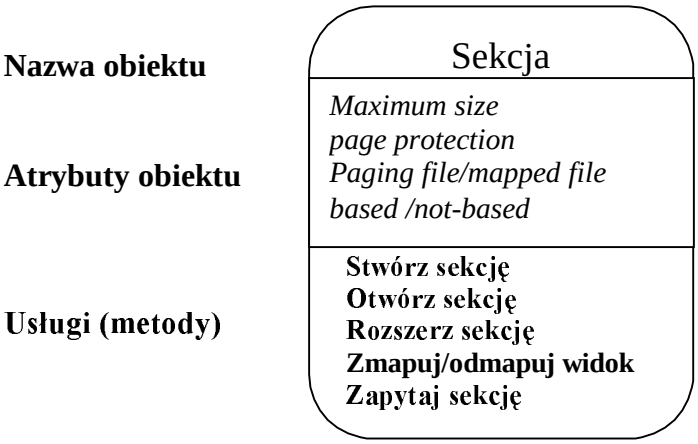
System NT używa mapowania plików podczas ładowania do pamięci plików wykonywalnych, tego samego mechanizmu używa VMM manager pamięci podręcznej systemu do zapisu i odczytu „zkeszowanych stron”. System I/O używa zmapowanego w

pamięci pliku do obsługi żądań wejścia/wyjścia, pozwalając VMM, aby zarządzał procesem zapisu zmian w pliku na dysk stosują normalną zasadę stronicowania.

Proces Win32 może używać techniki mapowania plików, aby mieć możliwość nieograniczonego dostępu do bardzo dużych plików. Inne aplikacje tworzą obiekt *file-mapping* (podobnie jak obiekt sekcja) odwołujący się do danego pliku i w ten sposób uzyskując możliwość pisania i czytania z tego pliku. VMM automatyczne wczytuje potrzebne strony i zapisuje zmiany w pliku na dysk.

3.2.2 Obiekt Sekcja.

Obiekt sekcja podobnie jak obiekty jest alokowany i dealokowany przez managera obiektów *object manager*. Object manager tworzy i inicjalizuje nagłówek obiektu, przez który można się do niego odwoływać. VMM tworzy ciało obiektu sekcji oraz udostępnia usługi pozwalające na manipulowanie wartościami atrybutów tych obiektów przez procesty działające w trybie użytkownika.



Poszczególne atrybuty obiektu oznaczają:
Tabela A.

Atrybut	*Znaczenie*
Maximum size	Maksymalny rozmiar sekcji w bajtach
Page protection	Czy strony pod którymi jest zmapowy dany plik są chronione
Paging file/mapped file	Określa czy sekcja jest utworzona jako pusta (zwierająca plik stronicowania) czy jest to załadowany plik.
Based/not based	sekcja jest *based* jeśli musi być mapowana pod tymi samymi adresami wirtualnymi dla wszytkich procesów korzystających z niej lub *not based* jeli nie musi.

Zmapowanie sekcji w postaci widoku powoduje, że jej część jest widoczna dla procesu w jego wirtualnej przestrzeni adresowej. Odmapowanie widoku usuwa go z wirtualnej przestrzeni adresowej procesu. Współdzielenie polega na tym, że dwa procesy mają zmapowany ten sam obszar sekcji. W takim przypadku konieczna jest synchronizacja dostępu. Używa się takich mechanizmów jak wyjątki czy semafory (czasami blokady realizowane sprzętowo). Synchronizacją dostępu do sekcji zajmuje się aplikacja Win32. Wątek, aby uzyskać dostęp do sekcji musi ją najpierw otworzyć. Proces tworzący sekcję automatycznie ją otwiera. Inne procesy mogą otworzyć sekcję, jeśli znają jej nazwę. Uzyskują ją przez dziedziczenie lub inny proces może im przekazać. Jeśli obiekt sekcja jest tworzony jako tymczasowy jest usuwany przez *object manager* z pamięci, gdy ostatnie odwołanie do niego jest zwalniane.

3.3. Ochrona pamięci.

Ochrona pamięci w Windows NT jest realizowana na cztery sposoby. Następujące trzy są standardem w większości nowoczesnych systemów operacyjnych:

- oddzielna przestrzeń adresowa dla każdego procesu. Sprzęt nie pozwala żadnemu procesowi na dostęp do przestrzeni adresowej innego procesu.
- dwa tryby dostępu do pamięci: tryb jądra, który umożliwia procesom dostęp do kodu czy danych systemowych i tryb użytkownika, który nie udostępnia takiej możliwości.
- mechanizm ochrony stron. Każda strona posiada zestaw flag określających dostęp do nich w trybie jądra oraz użytkownika.

Następujący mechanizm jest zrealizowany tylko w Windows NT.

- ochrona pamięci jako obiektu. Kiedy jakiś proces próbuje otworzyć obiekt lub zmapować widok na nim, Windows *NT security reference monitor* sprawdza czy proces próbujący wykonać tą operacje ma do tego prawo.

3.3.1 Pamięć prywatna.

VMM dba o to, aby dany proces nie odwołał się do strony pamięci należącej do innego procesu.

W niektórych systemach ochrona pamięci realizowana jest sprzętowo. Dla przykładu w procesorach MIPS R4000 strona jest tworzona w trybie użytkownika (jest umieszczana pod adresem poniżej 2G) lub trybie jądra (umieszczana pod adresem powyżej 2G). Dodatkowo dla każdej strony ustalane są prawa tylko-do-odczytu lub odczytu/zapisu. Proces działający w trybie użytkownika może czytać tylko strony dostępne w trybie użytkownika. MIPS R4000 generuje przerwanie naruszenia praw dostępu *(access violation)*, jeśli wątek próbuje wykonać jakąś operację na stronie, do której nie odpowiednich praw.

Bardzo podobnie ochronę stron realizuje VMM. Oprócz trybów tylko-do-odczytu, odczyt/zapis (jak w MIPS R4000) VMM otwiera strony z następującymi flagami:

- Execute-only, (jeśli umożliwia to sprzęt)
- Guard-page
- No-access
- Copy-on-write

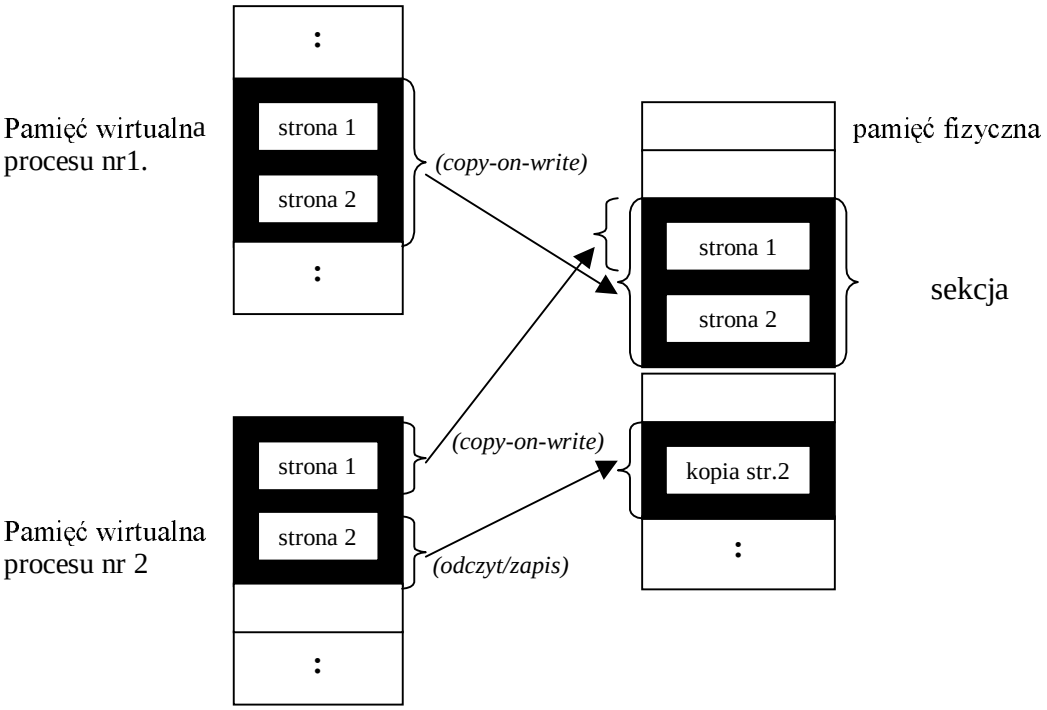
VMM dba o to, aby żaden wątek nie pisał do strony, która jest zaznaczona jako **tylko-do-odczytu**. VMM zabrania pisać i czytać wątkowi, jeśli strona jest oznaczona jako *execute-only,* zezwala jednak na skok z jednego adresu na inny w ramach tej strony. Tryb ten jest przydatny w przypadku, gdy kilka procesów chce współdzielić kod wykonywalny jakiegoś programu (np. kompilatora). (MIPS R4000 oraz Intel 386 i 486 nie udostępniają trybu ochrony pamięci *execute-only*, w procesorach tych jest on utożsamiony z trybem tylko-do-odczytu).

Tryb *guard-page* został zaimplementowany do ochrony stosów w pamięci, ale nadaje się z powodzeniem do ochrony też innych struktur danych. Jeśli dany wątek próbuje czytać coś ze strony oznaczonej ta flagą generowany jest wyjątek *guard-page* i VMM pozwala mu kontynuować operację. Jeśli jakiś proces systemowy odwołuje się do takiej strony to nie będzie mógł rozszerzyć zapisanej tam struktury danych, jeśli znajduje się ona na końcu takiej strony.

Tryb *No-access* nie pozwala czytać ani pisać na stronie żadnemu wątkowi. VMM w przypadku takiej próby generuje odpowiedni wyjątek. Tą flagą są oznaczone strony nie zaalokowane przez żaden proces lub zarezerwowane.

3.3.2 Pamięć współdzielona *(Shared Memory)*

Wspomniany tryb *copy-on-write* jest pewną optymalizacją służącą do zaoszczędzenia pamięci. Jeśli dwa procesy chcą czytać ten sam obszar pamięci, jest on oznaczany prze system flagą *copy-on-write* i od tej pory jest przez nie współdzielony. Jeśli jeden z wątków chce coś zapisać to taki kawałek pamięci jest kopiowany w inne miejsce, tam modyfikowany, uaktualniona zostaje przestrzeń adresów wirtualnych tego procesu, skopiowane strony otrzymują flagę odczyt/zapis.



Jak pokazano na rysunku skopiowane strony nie są już widoczne dla innego procesu, wątek jednak może pisać do swojej kopii nie wpływając w żaden sposób na działanie innych programów. Ochrona typu *Copy-on-write* jest bardzo użytecznym mechanizmem w przypadku kodu wykonywalnego. Jeśli jakiś proces zmieni kod (np. debugger wstawiając pułapki w kodzie) tworzy swoją kopie programu nie wpływając na działanie innych procesów również korzystających z niego. *Copy-on-write* jest stosowana przez subsystem Win32 np. w przypadku dynamicznie dołączanych bibliotek (DLL).

Ochrona pamięci *Copy-on-write* prezentuje klasyczne tzw. leniwe podejście w programowaniu. Leniwe algorytmy unikają wykonywania kosztownych operacji, aż do momentu, kiedy jest to niezbędne. Dla Przykładu, gdy jest wołana funkcja fork() w systemach klasy POSIX system operacyjny kopiuje przestrzeń adresową wołającego funkcję do przestrzeni adresowej potomka. Inaczej działa VMM, który tylko oznacza strony flagą *copy-on-write* i oba procesy rodzic i potomek współdzielą je. Oba korzystają z tego samego obszaru pamięci aż do momentu, gdy któryś chce coś w nim zapisać. Jeśli coś jest modyfikowane kopiowane są tylko te strony, które się zmieniły dla jednego procesu, a nie cała przestrzeń adresowa.

Ochrona pamięci może być implementowana sprzętowo lub programowo (programy bardzo niskiego poziomu). Obiektowa architektura systemu Windows NT pozwala na jeszcze jeden typ ochrony pamięci. System chroni obiekty sekcji tak samo jak inne obiekty, używając tzw. list kontroli dostępu *access control lists* (ACL). Wątek może utworzyć sekcję, którego

ACL określa, jaki użytkownik lub grupy użytkowników mogą czytać, pisać, uzyskać informację o niej lub rozszerzyć jej rozmiar.

Security reference monitor za każdym razem, gdy jakiś wątek próbuje otworzyć sekcję lub zamapować na niej widok sprawdza jego uprawnienia. Jeśli ACL nie pozwala wykonać wołanej operacji *object manager* odrzuca żądanie. Wątek może zmienić typ ochrony strony na inny, jeśli ta operacja nie jest w sprzeczności z ACL. Dla przykładu VMM pozwala na zmianę trybu *read-only* na *copy-on-write,* ale nie na *read/write,* gdyż ta operacja nie wpłynie na działanie innych procesów współkorzystających z tych samych danych.

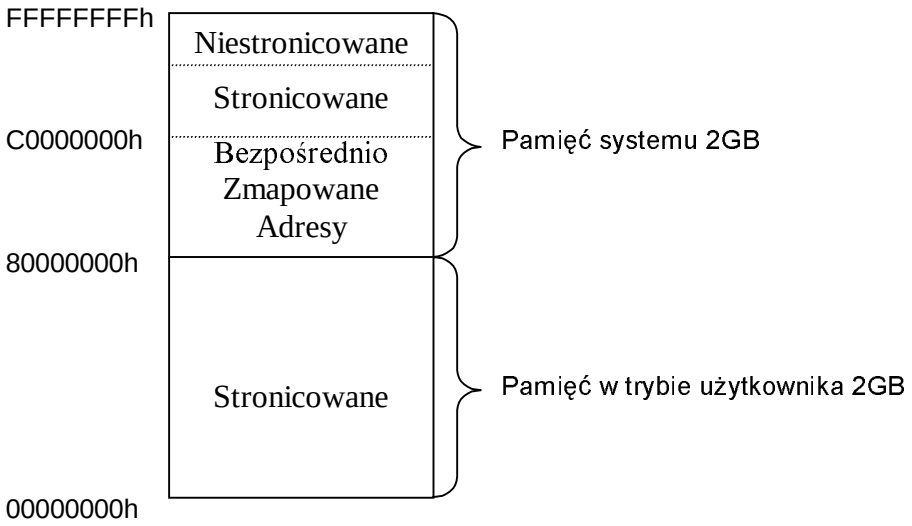
Podobnie jest w przypadku, gdy jakiś wątek tworzy sekcję zwierającą zmapowany plik. Aby było to możliwe, wątek musi mieć dostęp do tego pliku (przynajmniej prawo do odczytu). Kiedy plik jest już otwarty w sekcji, wątek może zmienić listę ACL obiektu sekcji, ale tylko w ramach określonych przez ACL zmapowanego pliku.

4 Implementacja Pamięci Wirtualnej.

Do tej pory zajmowaliśmy się kwestiami związanymi z ogólnym działaniem pamięci wirtualnej i trybem użytkownika realizowanym przez VMM. W tym rozdziale opisana zostanie metoda realizacji PW – a więc użyte algorytmy, struktury danych niezauważalne dla użytkownika systemu, ale kluczowe dla jego poprawnego działania.

4.1 Przestrzeń adresowa.

System NT potrafi zaadresować bardzo dużą wirtualną przestrzeń adresową do 4GB. Adresy od 2 do 4GB są zarezerwowane dla systemu i są dostępne tylko dla procesów działających w trybie jądra. Niższe adresy są dostępne dla wątków działających w trybie jądra lub użytkownika.



Na pokazanym rysunku kod jądra i potrzebne dane systemowe rezydują pod adresami od 80000000 do BFFFFFFFh i nigdy nie są z pamięci rzucane na dysk. W procesorach MIPS R4000 ten obszar pamięci jest mapowany sprzętowo. Procesor zeruje trzy najbardziej znaczące bity adresu wirtualnego i tak powstaje adres fizyczny. Ponieważ tłumaczenie adresów odbywa się sprzętowo dostęp do danych zapisanych w tym obszarze jest bardzo szybki. Dostęp do danych znajdujących się pod drugą częścią wirtualnej przestrzeni adresowej organizuje już VMM, gdyż tam znajdują się dane lub kod, które mogą być zrzucone na dysk.

4.2 Stronicowanie

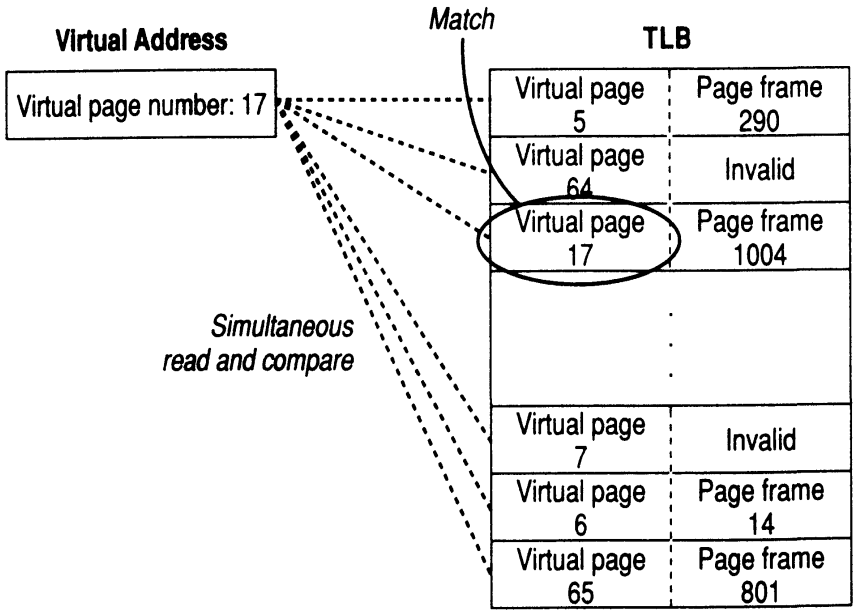
Stronicowaniem w systemach Windows NT zajmuje się oczywiście VMM. To on określa, kiedy dana strona może zostać zrzucona na dysk, lub kiedy powinna zostać wczytana do pamięci.

4.2.1 Mechanizm stronicowania

Pamięć wirtualna realizowana sprzętowo na każdym innym procesorze działa inaczej. Tak, więc kod programowej realizacji pamięci wirtualnej nie jest łatwo przenaszalny na inne platformy. W najlepszym wypadku kod taki może być mały i dobrze „wyizolowany" z systemu (jak to jest w Windows NT). Podrozdział ten będzie opisywał głownie procesor MIPS R4000 i ukaże jak współpracuje on z systemem NT. Wiele informacji tutaj przytoczonych będzie również pasowało do procesorów CISC Intela.

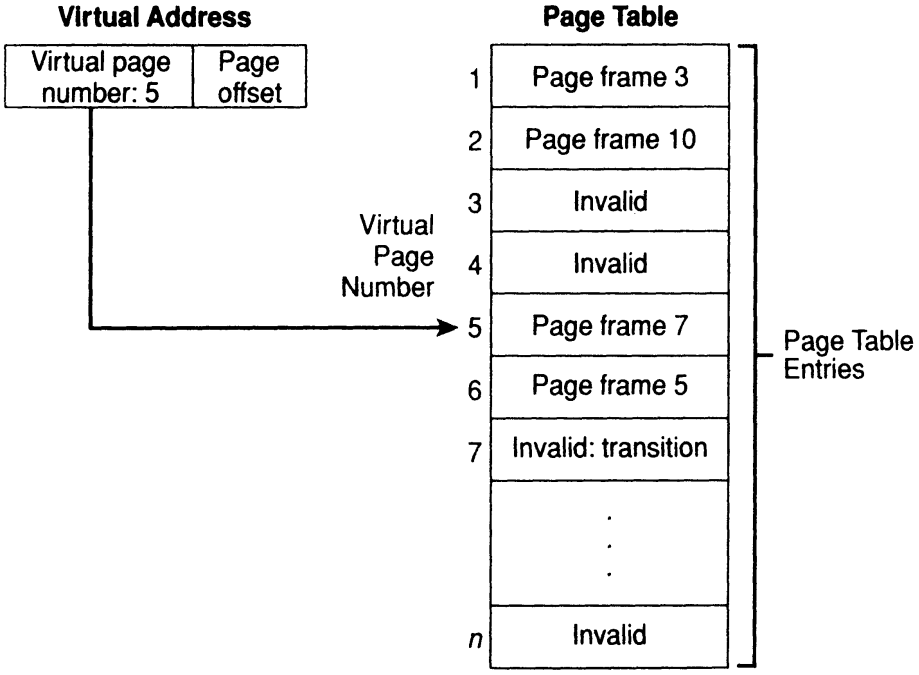
Procesor MIPS R4000 zawiera dwa moduły: 32-bitową jednostkę liczącą typu RISC (zwaną CP1) oraz oddzielny moduł zajmujący się tłumaczeniem adresów i obsługą wyjątków (zwany CP0). CP0 automatycznie przechwytuje każdy adres wygenerowany przez program i tłumaczy na fizyczny. Jeśli strona zawierająca podany adres jest w pamięci to CP0 odczytuje z niej potrzebną informację. Jeśli strona jest na dysku to CP0 generuje wyjątek *page-fault* resztą zajmuje się VMM.

Aby dostęp do pamięci był szybki, w MIPS R4000 (podobnie w procesorach Intela) zaimplementowano specjalną jednostkę zwaną *translation lookaside buffer* (TLB). TLB jest to tablica zawierająca informacje z adresami wirtualnymi i fizycznymi najczęściej używanych stron oraz typem ochrony, jaki ich dotyczy.

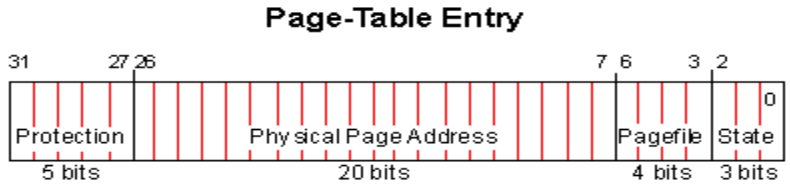


Często używane adresy wirtualne najprawdopodobniej mają swoje wpisy w tablicy TLB, dzięki czemu tłumaczenie ich na adresy fizyczne jest bardzo szybkie a tym samym dostęp do pamięci jest bardzo przyspieszany. Jeśli adres wirtualny nie znajduje się w tablicy TLB, ale informacja pod nim ciągle jest dostępna w pamięci dostęp do niej jest trochę wolniejszy, gdyż VMM musi przetłumaczyć adres wirtualny na fizyczny. Jeśli jakaś strona została zrzucona na dysk to przy wpisie w TLB dotyczącej tej tablicy znajdzie się informacja, że jest ona niedostępna. Gdy jakiś proces znów się do niej odwoła zostaje ona sprowadzona do pamięci i zmienia się wpis w TLB. Jądro i VMM używają **tablic stron** *page tables* do zbierania informacji, które są obecne w tablicy TLB, a które nie.

W zależności od systemu **tablice stron** mogą być realizowane sprzętowo lub programowo. Koncepcja tablicy jest przedstawiona na poniższym rysunku.



Pozycja w tablicy stron *page table entry* (PTE) zawiera całą informację potrzebną do zlokalizowania strony w pamięci fizycznej. W prostych realizacjach pamięci wirtualnej informacja niedostępny *invalid* oznacza, że strona znajduje się na dysku i trzeba ją sprowadzić i zaktualizować tablicę stron. Struktura PTE jest następująca:



Pierwsze 5 bitów jest przeznaczone na określenie typu ochrony założonej na stronę. Następne 20 bitów określa adres strony w pamięci fizycznej (strona ma 4 kB, więc poprzez 20 bitów można zaadresować 2^{20}x4096B=4GB przestrzeń). Jeśli strona jest na dysku to te 20 bitów określa *pagefile, d*zięki której zlokalizujemy potrzebną stronę na dysku. Następne 4 bity pozwala określić *pagefile,* która wskazuje na poszukiwaną stronę. Następne trzy bity określają stan strony (tabela B). Pierwszy z tych bitów (T) stanowi tzw. flagę przeniesienia, drugi (D) określa czy jest ona zmodyfikowana ostatni (P) mówi czy strona jest dostępna.

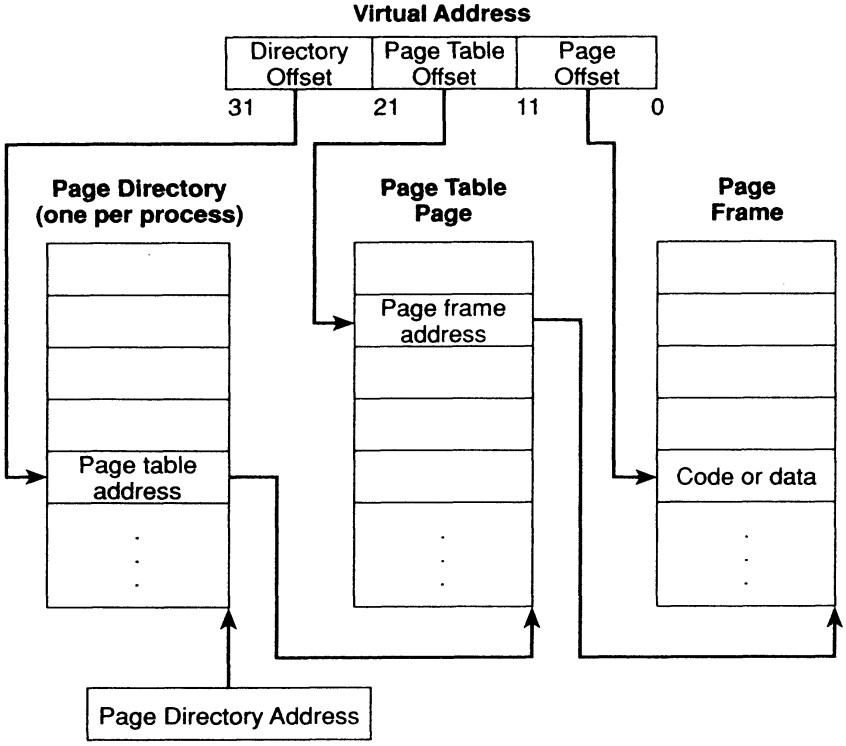
Tabela B

T	D	P	Stan strony.
0	-	0	niedostępna
-	0	1	dostępna
-	1	1	zmodyfikowana (dostępna)
1	0	0	niedostępna (w przeniesieniu)
1	1	0	niedostępna (zmodyfikowana, w przeniesieniu)

Bit modyfikacji jest ignorowany, gdy strona jest niedostępna i bez przeniesienia lub dostępna w pamięci.

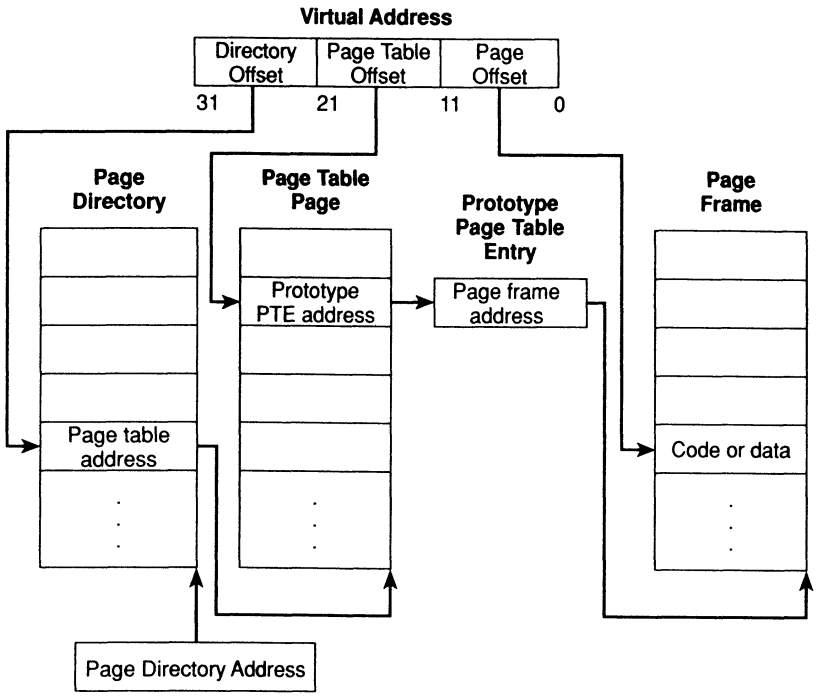
Procesor MIPS R4000 posiada 32-bitowe adresowanie, tak więc każdy proces może widzieć 2^{32} adresów wirtualnych. MIPS R4000 organizuje adresy wirtualne w strony wielkości 2^{12} bajtów (4 kB). Jeśli pozycja w tablicy stron zajmuje 4 bajty to na 1024 stronach można zmapować całą pamięć wirtualną. Każdy proces posiada własną przestrzeń adresów wirtualnych i aby uniknąć zapełnienia całej pamięci wyłącznie tablicami stron, VMM zrzuca je na dysk i ściąga do pamięci, gdy są potrzebne.

Procesor MIPS R4000 pozwala systemowi operacyjnemu określić strukturę tablicy stron. W procesorach 386 Intela format tablicy stron jest z góry określony. Aby zapewnić maksimum przenaszalności systemu NT zaimplementowano tam tablicę w dwu-poziomowym formacie wymaganym prze Intel. Tablica pierwszego poziomu zwana *page directory* zawiera pola wskazujące na strony w tablicy stron drugiego poziomu (jedna taka tablica należy do jednego procesu).



Tablica *page directory* oraz tablica stron może być dostępna lub nie (zrzucona na dysk). Jeśli *page directory* jest nie dostępna to jest sprowadzana z dysku podobnie dzieje się w przypadku tablicy stron oraz strony w pamięci, której szukamy. Każda pozycja w tablicy stron (również w tablicy *page directory*) w systemach NT zawiera dodatkowo tzw. flagę przeniesienia *transition flag*. Jeśli strona jest zaznaczona jako niedostępna, ale ma ustawioną flagę przeniesienia, oznacza, że jej zawartość jest ciągle w pamięci. Oprócz flagi przeniesienia wpis a tablicy stron zawiera też informację o tym, jaki typ ochrony dotyczy tej strony.

Jeśli ramka jest dzielona pomiędzy dwa procesy, VMM musi zmodyfikować dwie tablice stron. Aby pozbyć się konieczności modyfikowania kilku tablic stron tworzona jest tzw. *prototype page table entry* (prototype PTE).



Prototypowa PTE jest to 32-bitowa struktura wyglądająca podobnie jak normalna PTE, która umożliwia zarządzanie współdzielonymi stronami bez konieczności aktualizacji kilku należących do procesów tablic stron. Dla przykładu, jeśli jakaś strona jest zrzucana na dysk i następnie wczytywana do innej ramki niż była poprzednio, trzeba zmodyfikować tylko jeden wpis w prototypowej PTE wskazujący na nową lokalizację strony. W takim przypadku żadna tablica stron należąca do jakiegoś procesu nie jest modyfikowana.

4.2.2 Metody Stronicowania.

Stronicowanie obejmuje trzy kwestie::

- kolejność wczytywania stron
- umieszczenie strony w pamięci
- usunięcie strony z pamięci

Są dwie podstawowe metody wczytywania stron. Pierwsza stara się, aby w pamięci były już strony, których dany proces będzie potrzebował, zanim on ich zażąda. Druga zwana *demand paging policy* ładuje strony do pamięci dopiero w momencie odwołania się do nich. W tym przypadku zanim zostanie uruchomiony jakiś proces, wystąpi duża liczba wyjątków braku strony. Po załadowaniu programu do pamięci częstość wołania wyjątków *page fault* spadnie.

VMM używa metody *demand paging algorithm with "clustering"* czyli drugiej metody z poprzedniego akapitu trochę ulepszonej. Gdy wystąpi wyjątek *page fault* VMM ładuje do pamięci potrzebną stronę oraz pewną liczbę innych sąsiadujących z nią. Ponieważ programy wykonując się w jakimś niezbyt dużym rejonie swojej wirtualnej przestrzeni adresowej, taktyka *klastrowania* zmniejsza częstość wystąpienia wyjątków braku strony.

Kiedy wystąpi wyjątek *page fault* VMM musi także zdecydować gdzie w pamięci fizycznej umieścić daną stronę. W systemach NT, jeśli pamięć nie jest cała zajęta VMM wybiera z listy wolnych ramek pierwszą ramkę i tam umieszcza wczytaną stronę. Jeśli pamięć fizyczna jest pełna należy w jakiś sposób wybrać strony do zrzucenia na dysk. Najbardziej znane metody wyboru takich stron to:

- ostatnio nieużywane *least recently used* (LRU)
- kolejka fifo *first in, first out* (FIFO)

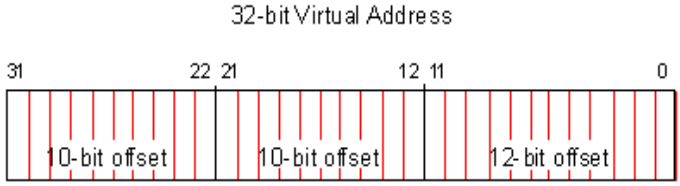
Algorytm LRU wymaga śledzenia, kiedy dana strona była używana. Gdy potrzebna jest wolna ramka, wybierana jest taka, do której najdłużej nikt się nie odwoływał. Algorytm FIFO jest prostszy, zwalniana jest ramka, która była używana najdłużej, bez patrzenia jak często ktoś się do niej odwoływał.

Metody zastępowania zawartości stron mogą być dwie. Pierwsza *local replacement policy* alokuje dla każdego procesu ustaloną liczbę ramek. Gdy brakuje dla niego wolnych ramek zwalniane są te, które należą do jego puli. Druga *Global replacement policy* wybiera wolną ramkę ze wszystkich dostępnych w systemie. Z drugą metodą związane są pewne problemy. Powoduje ona, że dany proces może wpływać na działanie innych. Dla przykładu, „pamięciożerny" program spowolni działanie innych obecnych w systemie procesów spowolniając pracę całego systemu. W systemie Windows NT dany proces nie współzawodniczy z innymi procesami o pamięć, uzyskując w pamięci określoną liczbę ramek, tak aby mógł prawidłowo działać. Zbiór takich ramek nazywany jest *process's working set.*

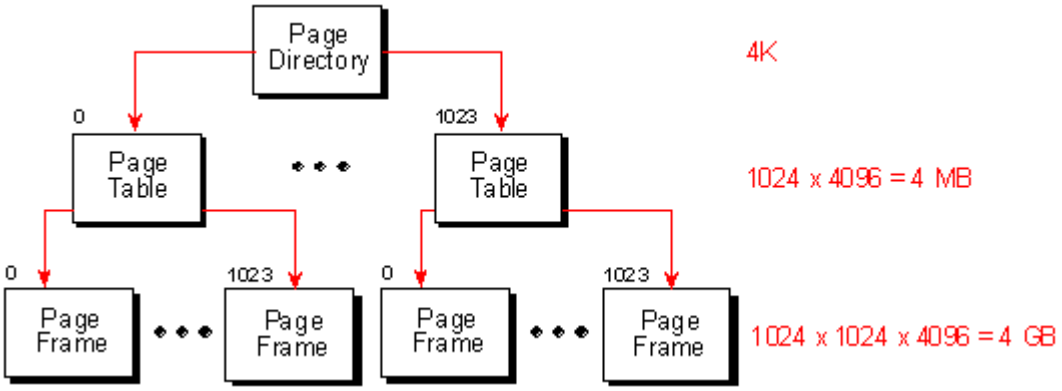
Podczas tworzenia procesu dostaje on minimalna liczbę takich stron *minimum working-set value*. Jeśli pamięć się nie zapełnia system pozwala zwiększyć tą liczbę. Jeśli wolnej pamięci fizycznej zrobi się niebezpiecznie mało VMM działa zgodnie z zasadą *automatic working-set trimming*. Przegląda wszystkie procesy i jeśli znajdzie taki, który zajmuje więcej stron w pamięci niż wskazuje na to jego *minimum working-set value,* zwalnia niektóre strony zajęte przez ten proces. Jeśli cały czas brakuje pamięci to VMM zwalnia zajęte ramki, aż do momentu, gdy wszystkie procesy osiągną swój *working-set minimum.* Gdy tak się stanie VMM śledzi ile wyjątków *page fault* zostanie wygenerowanych w wyniku działania danego programu. Jeśli jest jeszcze trochę wolnej pamięci to VMM zezwala procesowi na rozszerzenie jego *working set.* Proces może zmienić wartości atrybutó *working-set minimum* i *working-set maximum,* jednak możliwość ta jest rzadko wykorzystywana w praktyce.

4.3 Tłumaczenie adresów

W Windows NT 32-bitowy adres wirtualny AW jest podzielony na trzy części. Poszczególne grupy bitów są używane jako offset dla zlokalizowania różnych stron w pamięci.

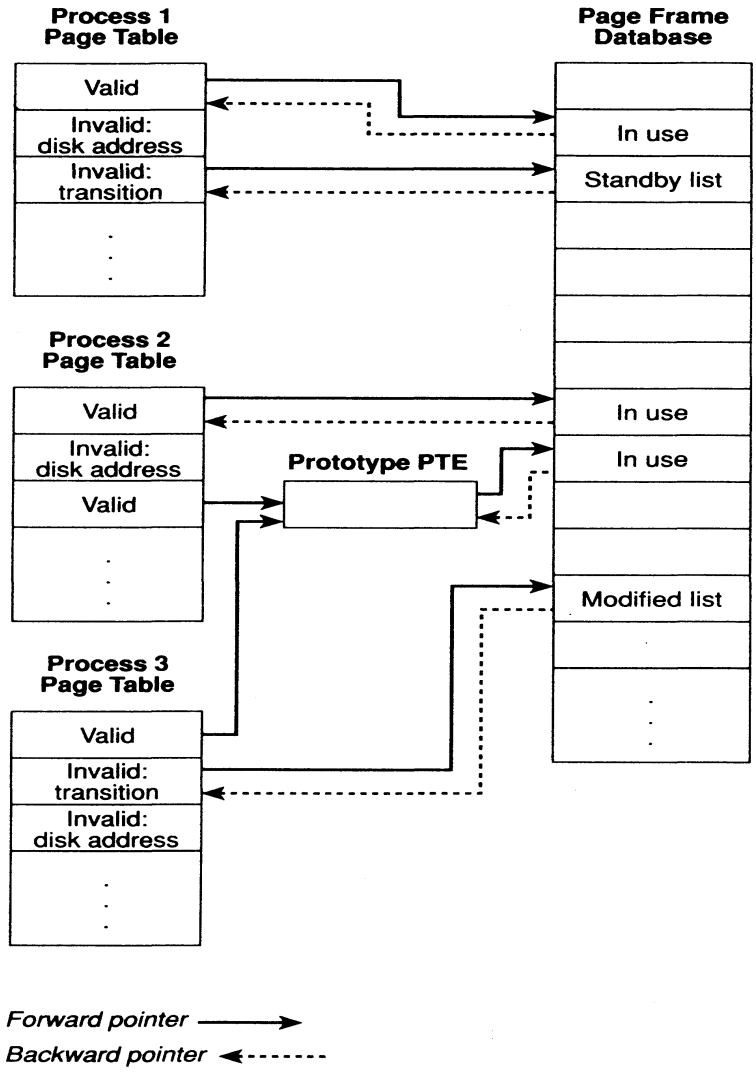


W pierwszym kroku tłumaczenia adresu wirtualnego na fizyczny, pobierane jest pierwsze 10 najbardziej znaczących bitów i na ich podstawie jest określana pozycja w tablicy *Page Directory* ***PD***, którą indywidualnie posiada każdy proces. Tablica *Page Directory* to 4 kB strona w pamięci podzielona na 1024 4-bajtowych segmentów zwanych *page-directory entries* (PDE). 10 bitów wystarcza, aby zindeksować każdą pozycję w PD (2^{10}=1024 pozycji). PDE jest używany do znalezienia strony w pamięci zwanej tablicą stron *page table* Druga 10-bitowa część AW określa pozycję w *page-table* zwaną *page-table entry* (PTE) w ten sam sposób jak w przypadku PD. W PTE jest zapamiętany numer ramki w pamięci fizycznej. Ostanie 12 bitów określa konkretny bajt w tejże 4kB ramce.



Dzięki takiej metodzie adresowania Windows NT potrafi zmapować inną dla każdego procesu wirtualną przestrzeń adresową niezależnie od fizycznie dostępnych zasobów. Dzięki temu, że jedna strona składa się z 4 kB może w danej chwili być tablicą *page directory* lub tablicą stron.

4.4 Baza danych ramek

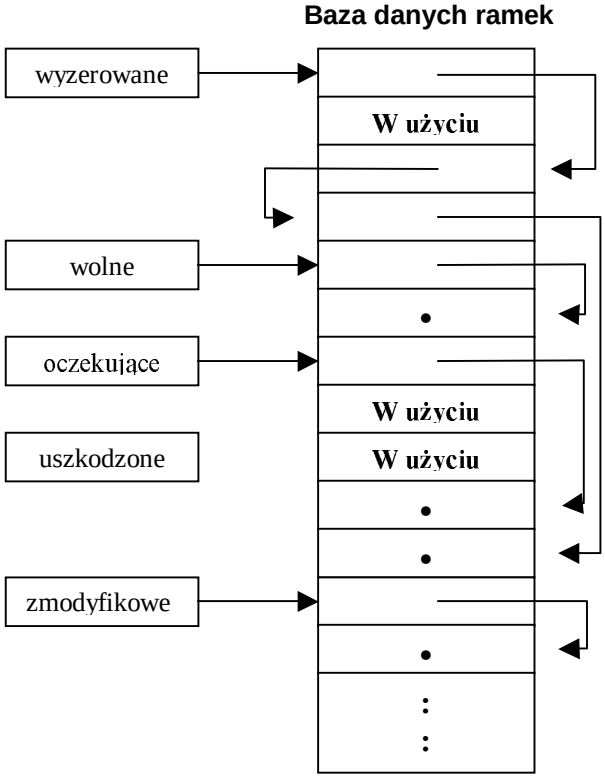


Tablica stron procesu pomaga mu zlokalizować, gdzie wirtualna strona jest umieszczona w pamięci fizycznej. VMM potrzebuje także informacji o stanie pamięci

fizycznej (czy dana ramka jest zajęta i kto jej używa). Do zbierania takich informacji służy baza danych ramek *page frame database PFD*. Jest to tablica ponumerowana od 0 do liczby ramek w systemie (minus 1), każde pole dotyczy jakieś ramki. Pozycja w tablicy stron dostępnych wskazuje na jakieś pole w bazie danych ramek i odwrotnie jeden wpis w PFD wskazuje na odpowiednią pozycję w tablicy stron. Wpisy w tablicy stron niedostępnych zawiera odnośnik do adresu dyskowego, gdzie przechowywana jest dana strona.
Ramka może być w jednym z sześciu stanów:

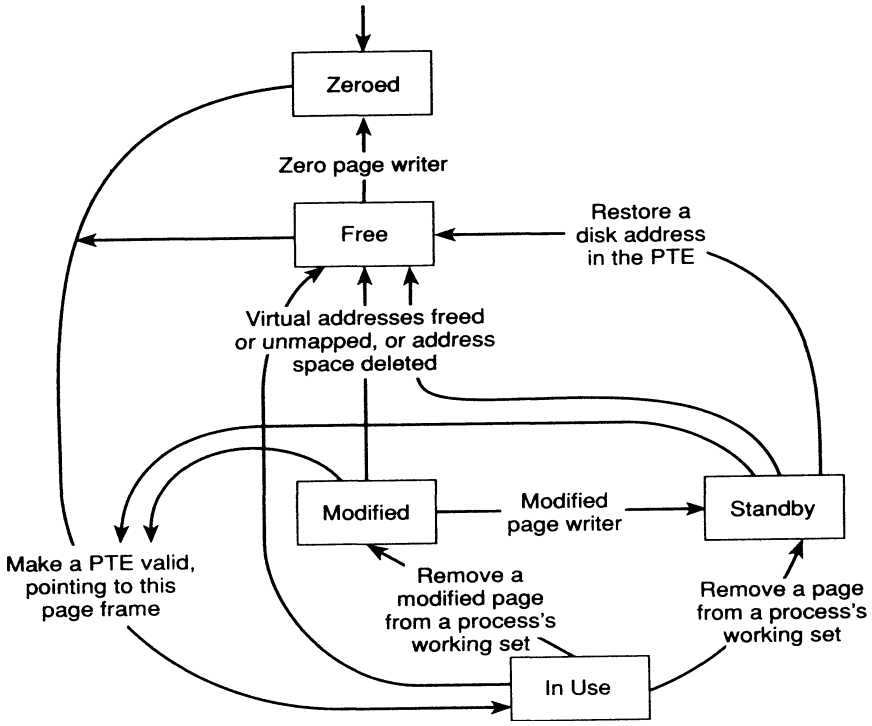
- **Dostępna.** Zawiera jakąś stronę, z której korzysta jakiś proces. Istnieje wpis w tablicy dostępnych stron, który na nią wskazuje.
- **Wyzerowana.** Ramka jest pusta i została wypełniona zerami.
- **Wolna.** Ramka jest wolna I nie zainicjalizowana.
- **Oczekująca.** Proces używał tej ramki, ale została ona usunięta z jego *working set*. Wpis w tablicy dostępnych stron wskazuje jest nie dostępna, ale zaznaczona jest flaga przeniesienia.
- **Zmodyfikowana.** Stan ten jest podobny do stanu *Oczekująca* różnica polega na tym, że proces zmodyfikował jej zawartość, ale nie została ona zapisana na dysk.
- **Uszkodzona**. Wygenerowała błąd parzystości lub inny błąd sprzętowy i nie może być używana.

Baza danych ramek grupuje w listy ramki, które nie są używana przez żaden proces. PFD zawiera: listy ramek wyzerowanych, wolnych, oczekujących, zmodyfikowanych i uszkodzonych.



Ramki, które są zajęte i używane są wskazywane przez jakieś pole w tablicy stron. Jeśli są zwalniane trafiają na odpowiednią listę nieużywanych. Jeśli VMM potrzebuje ramki zainicjalizowanej na zera szuka jej na liście wyzerowanych, jeśli jest ona pusta, to wybiera pierwszą wolną i ją zeruje. Jeśli VMM potrzebuje wolnej ramki a lista wolnych ramek jest pusta bierze jakąś z listy wyzerowanych. Jeśli obie te listy są puste wybierana jest jakaś ramka oczekująca. Jeśli liczba ramek wolnych, wyzerowanych i oczekujących spada poniżej określonego minimum, uruchamiany jest wątek *modified page writer*, który zapisuje

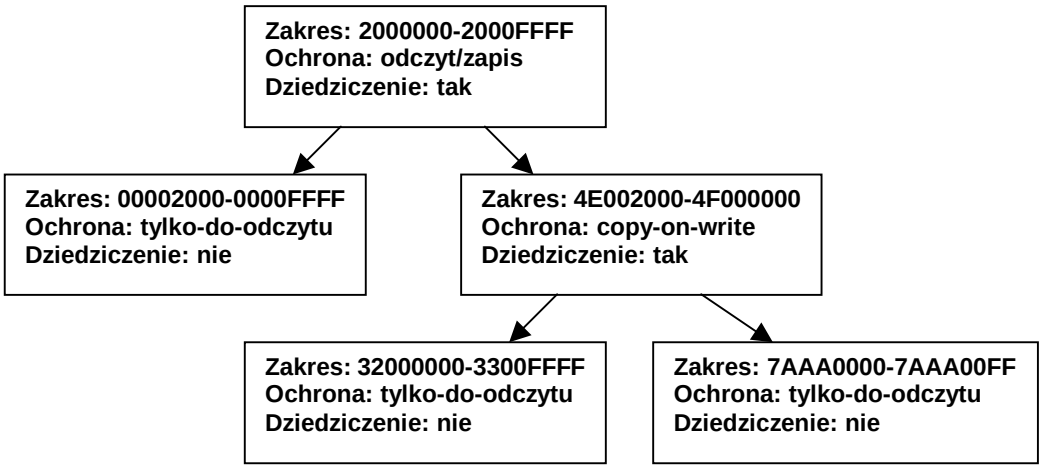
zawartość zmodyfikowanych ramek na dysk i przenosi je na listę oczekujących. Jeśli wszystkie te listy skracają się do niebezpiecznego poziomu, uruchamiana jest procedura *automatic working-set trimming.* Świeżo zwolnione ramki trafiają na listy zmodyfikowanych lub oczekujących. Cała ścieżka postępowania wygląda następująco:



Zanim VMM będzie mógł użyć danej ramki znajdujące się teraz na liście zmodyfikowanych lub oczekujących, musi uaktualnić odpowiedni wpis w tablicy stron, który ciągle wskazuje na tą ramkę. Jest to łatwe do wykonania, gdyż baza danych ramek ma odnośniki do odpowiednich pól w tablicy stron.

4.5 Deskryptory adresów wirtualnych

Alokowanie nawet dużych bloków pamięci wykorzystując algorytmy leniwe jest bardzo szybkie. Problem może jednak stanowić znalezienie w rozsądnym czasie odpowiednio dużego wolnego obszaru pamięci. Istnieje, zatem potrzeba zbierania przez proces informacji o tym, które adresy już są zajęte, które wolne. Temu służą tzw. deskryptory adresów wirtualnych *Virtual address descriptor.*



Gdy proces alokuje pamięć (lub mapuje widok na pamięci współdzielonej), VMM tworzy deskryptor adresów wirtualnych, który zawiera informację o zakresie alokowanej pamięci, czy jest obszar prywatny czy współdzielony, czy mają prawo wglądu do niej potomkowie procesu. Kiedy wątek pierwszy raz chce się dostać do pewnego adresu, VMM musi utworzyć wpis w tablicy stron dla strony, która zwiera ten adres. Aby to uczynić znajduje deskryptor adresów wirtualnych, którego zakres adresów zwiera ten adres i stąd uzyskuje informację potrzebne do wypełnienia tablicy stron. Jeśli adres jest poza przestrzenią opisaną w deskryptorach adresów wirtualnych oznacza to, że VMM nie zaalokował tego obszaru, w którym znajduje się adres i generowany jest błąd.

5 Wieloprocesorowość

Program, który jest uruchamiany na kilku procesorach, musi spełniać szereg wymagań: nie może dopuścić, aby dwa wątki miały dostęp jednocześnie do tej samej struktury danych, musi dbać o to, aby kilka wątków wykonujących ten sam program nie chciało skorzystać z zasobów komputera w ten sposób, że nastąpi blokada.

VMM potrafi zadbać o to, aby dwa procesy nie uzyskały dostępu do tej samej ważnej dla systemu struktury danych w tym samym czasie (do synchronizacji dostępu do tablicy ramek używa tzw. spinlocków). Gdy zostaje wygenerowany przez jakiś proces wyjątek braku strony, VMM pozwala procesowi zaktualizować odpowiednie struktury danych (baza danych ramek) i z powrotem blokuje do nich dostęp. Gdy dwa wątki jednocześnie chcą wczytać potrzebne strony do pamięci jeden z nich jest wstrzymywany aż do momentu, gdy pierwszy skończy pracę z bazą danych ramek.

Rozwiązanie te jest przykładem kompromisu między szybkością działania a potrzebami systemów wieloprocesorowych. VMM może udostępniać oddzielne części newralgicznych struktur różnym procesom jednocześnie, dzieląc w jakiś sposób je na części i na nie zakładając oddzielne *spinlocki*. Problem polega na tym, że założenie i uwolnienie takiej blokady jest operacją raczej kosztowną, więc obsłużenie wyjątku braku strony jeszcze by się wydłużyło, gdyby proces, aby uzyskać dostęp do bazy danych ramek musiał oczekiwać na zdjęcie np. czterech blokad.

Lou Perazzoli, twórca VMM, postawił raczej na szybkość działania sytemu niż jego „współbieżność”, zastosował jedną blokadę dla całej bazy danych ramek. Oznacza to, że w przypadku intensywnego stronicowania, obsługa PFD może stać się wąskim gardłem systemu. Aby rozwiązać ten problem, system Windows NT stara się utrzymać ilość wyjątków braku strony była jak najniższa. Robi to na dwa sposoby:

- każdemu procesowi przydziela tyle ramek do jego *working set* aby do minimum ograniczyć ilość wyjątków *page fault,* wołanych przez ten proces.
- automatycznie stara się nieużywane przez procesy strony zrzucać na dysk, aby zwolnić ramki dla innych procesów.

6 Przenaszalność

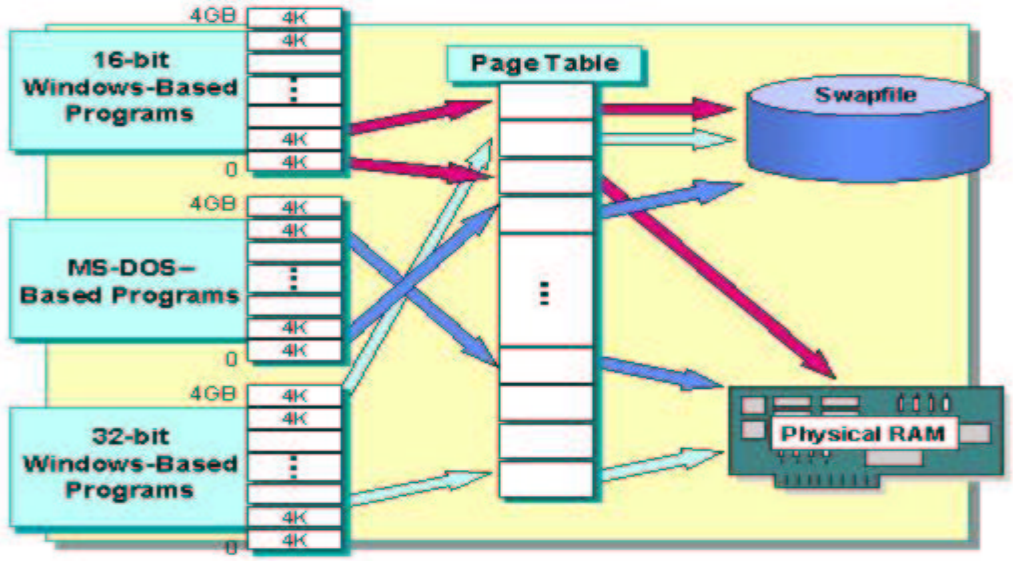
Działanie VMM jest uzależnione od sprzętu na którym jest uruchomiony. VMM stawia przed procesorem następujące wymagania:

- 32-bitowe adresowanie (64-bitowe adresowanie jest obsługiwane, jednak wymaga to kilku zmian w VMM)
- obsługa pamięci wirtualnej i mechanizmu stronicowania.
- przejrzyste, spójne pamięci cache w systemach wieloprocesorowych. Jeśli jakiś procesor zmienia dane w swojej pamięci podręcznej muszą wiedzieć o tym inne procesory po to, aby odpowiednie dane od razu skorygować u siebie.
- *aliasing* wirtualnych adresów. Procesor musi zezwalać odwoływanie się dwóch różnych pozycji w dwóch różnych tablicach stron do tej samej ramki.

VMM, aby być przeniesiony na inną platformę musi mieć zmienione następujące rzeczy:

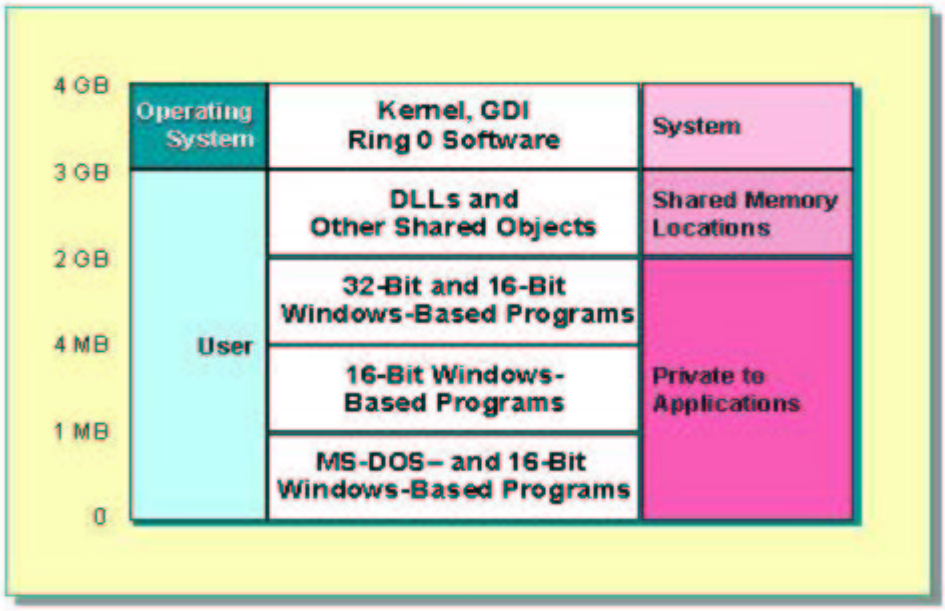
- **Format wpisów w tablicy stron.** Jeśli dany PTE określa stronę dostępną (obecną w pamięci), procesor dzieli 32bitowy adres na pewne części i na nich operuje w jakiś sposób, zależny od danej architektury.
- **Rozmiar strony**. VMM potrafi obsłużyć strony rozmiaru od 4 kB do 64 kB.
- **Mechanizmy ochrony stron**. Sposób, w jaki VMM korzysta ze sprzętowo realizowanej ochrony stron oraz sposób wykorzystania sprzętowych mechanizmów do implementacji dodatkowych trybów ochrony.
- **Sposób tłumaczenia adresów wirtualnych.**

7 Pamięć wirtualna w Windows 9*X*



W Windows 9x pamięć jest „wirtualizowana" w podobny jak sterowniki i inne zasoby sprzętowe. Podejście to powoduje, że każdy program zachowuje się jakby miał dostępną własny fizyczny RAM. Każdy program czy jest to 32-bitowa aplikacja Windows czy program systemu MS-DOS jest lokowany w swoim kawałku wirtualnej przestrzeni adresowej. Przestrzeń ta jest mapowana na fizyczna pamięć lub na dysk twardy. Windows 9x potrafi korzystać z mechanizmów stronicowania udostępnianych przez procesory Intel 80386 lub wyższe. Podczas zrzucania jakichś obszarów pamięci na dysk są one zapisywane do tzw. pliku wymiany *swapfile*. Rozmiar pliku wymiany zmienia się w zależności od potrzeb, jakie zgłasza system. Jeśli zaczyna brakować miejsca na dysku rozmiar ten jest ograniczany, jeśli na dysku jest sporo miejsca rozmiar ten jest zwiększany.

Podobnie jak systemach NT przestrzeń adresowa Windows 9x jest podzielona na 1,048,576 (2^{20}) stron rozmiaru 4 kB. Jedna strona może być fizycznie obecna w pamięci, zapisana na dysku w pliku wymiany lub zaznaczona jako nieużywana. Informacja o tym jest przechowywana w tablicy stron.



Na powyższym rysunku pokazano, w jaki sposób Windows 9x mapuje wirtualną przestrzeń adresową. Jest ona podzielona na sekcje:

- 0-1MB. Tutaj rezydują programy typu MS-DOS.
- 1-4MB. Normalnie nie używana. Żadna 32-bitowa aplikacja nie używa tych adresów mogą ich jednak używać aplikacje 16-bitowe.
- 4MB-2GB. Używana przez 32-bitowe (niektóre 16-bitowe również) programy systemu Windows
- 2-3GB. Rozmieszczone tu są biblioteki DLL oraz inne współdzielone obiekty np. standardowe okienka dialogowe opisane w plikach COMMDLG.DLL czy COMDLG32.DLL
- 3-4GB. Zarezerwowana przez system operacyjny np. na sterowniki.

Mechanizm stronicowania w Windows 9x jest trochę mniej wydajny niż w systemach klasy NT. System ten potrafi obsłużyć określoną liczbę wyjątków w danym odcinku czasu. Jeśli ta wartość zostanie przekroczona, następuje tzw. **migotanie stron** *disk thrashing.* Migotanie objawia się tym, że całe zasoby systemowe są przeznaczane na realizacje stronicowania i nie wykonuje on żadnej użytecznej z punktu widzenia użytkownika pracy i może skutkować nawet uszkodzeniem dysku twardego. Jedynym rozwiązaniem w takim wypadku jest dokupienie więcej RAM-u lub wyłączenie kilku programów.

ŹRÓDŁA:

Helen Custer: *Inside Windows NT* (rozdz. 6 ***The Virtual Memory Manager***)
http://cwdixon.com/support/win98_support/index.htm
http://msdn.microsoft.com/library/en-us/dngenlib/html/msdn_ntvmm.asp

Przygotował: Edwin Vaina **171278**