Pamięć wirtualna w systemie AS/400

Tomasz Kokoszka

3 grudnia 2002

Spis treści

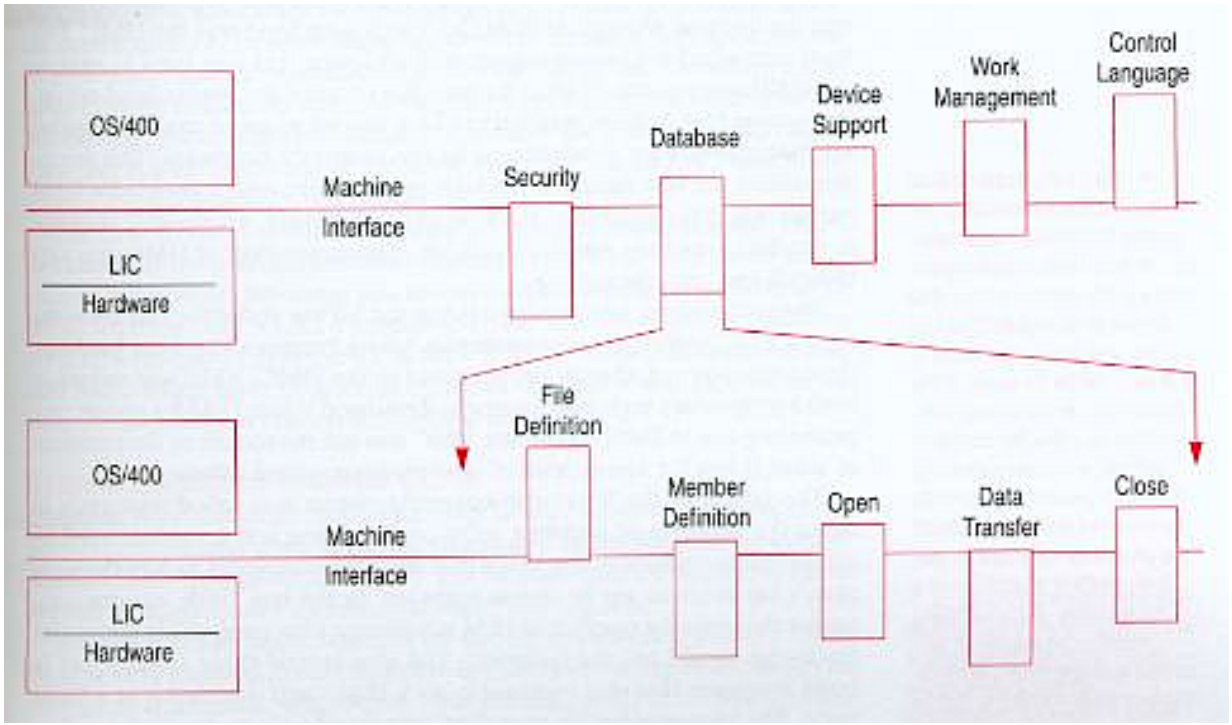
1 Wstęp

Plany firmy IBM koncentrowały się zawsze wokół zintegrowanych systemów dla biznesu (jak w tym żarcie „Co by było gdyby ... produkował tostery”). Oczywiście wraz z rozwojem techniki systemy musiały ewaluować. Tak powstała linia System/3 (System/32 z roku 1975, System/34 z 1977) które nie były ze sobą kompatybilne.

W pewnym momencie „mądrzy” ludzie z IBM stwierdzili, że dobrze by było zastąpić to wszystko jednym systemem zgodnym ze wszystkimi poprzednikami (system Fort Knox). Oczywiście ten projekt szybko upadł.

1.1 System/38

W latach siedemdziesiątych rozpoczęto pracę nad nowym rozwiązaniem dla biznesu – System/38. Miał on zastąpić całą linie System/3. Był to system bazujący na obiektach, ale nie obiektowy – nie ma dziedziczenia. Obiekty są pogrupowane w biblioteki. Wszystkie operacje (metody) wykonywane są na wskaźnikach do obiektów. Obiekty są jedynym pomostem pomiędzy aplikacją a systemem (zobacz rysunek numer 1).



Rysunek 1: System/38 – budowa

Objaśnienia do rysunku numer 1:

(TI)MI Technology-Independent Machine Interface to interfejs na potrzeby aplikacji i (częściowo) oprogramowania systemowego. Całkowicie zasłania sprzęt przed oprogramowaniem

położonym wyżej (a więc uniezależnia je od zmian sprzętowych). Ułatwia integrację różnych części systemu.

LIC Licensed Internal Code to część systemu operacyjnego leżąca poniżej MI.

OS/400 Obiekty i programy leżące powyżej MI. System operacyjny dla AS/400 jest połączeniem OS/400 i LIC.

SLIC System Licensed Internal Code: LIC dla systemów opartych na procesorach RISC (jądro systemu operacyjnego AS/400).

Taki schemat organizacji zapewnia bardzo dużą przenośność. Wystarczy napisać nowy SLIC by system działał na nowej architekturze.

Kolejną ciekawostką była tzw. jednopoziomowa pamięć wirtualna – single-level store. W nomenklaturze IBM pamięci w komputerze nie określa się jako *memory* – to by nadawało komputerowi cechy ludzkie – zamiast tego używa się terminu *store*.

Gotowy System/38 był dostępny dla klientów w roku 1980 (w 1978 roku odbyła się pierwsza prezentacja, jednak system zawierał kilka błędów). Było to z pewnością nowoczesne rozwiązanie jednak klienci uznali, że jest zbyt duży i zbyt drogi. Dlatego w 1983 roku wprowadzono mniejszy System/36 (później okazało się, że aplikacje z tego systemu mogą działać jako środowisko w System/38 ale to już inna historia).

1.2 AS/400

System AS/400 wywodzi się z Systemu/38, tak naprawdę jest to „przepakowany” System/38 (stary produkt w nowym pudełku). W latach osiemdziesiątych jeden z menedżerów z ośrodka Rochester przekonał szefów IBM, że AS/400 to zupełnie nowy produkt. Zrobił to tak skutecznie, że AS/400 szybko stał się projektem priorytetowym dla IBM i tak zostało do dziś.

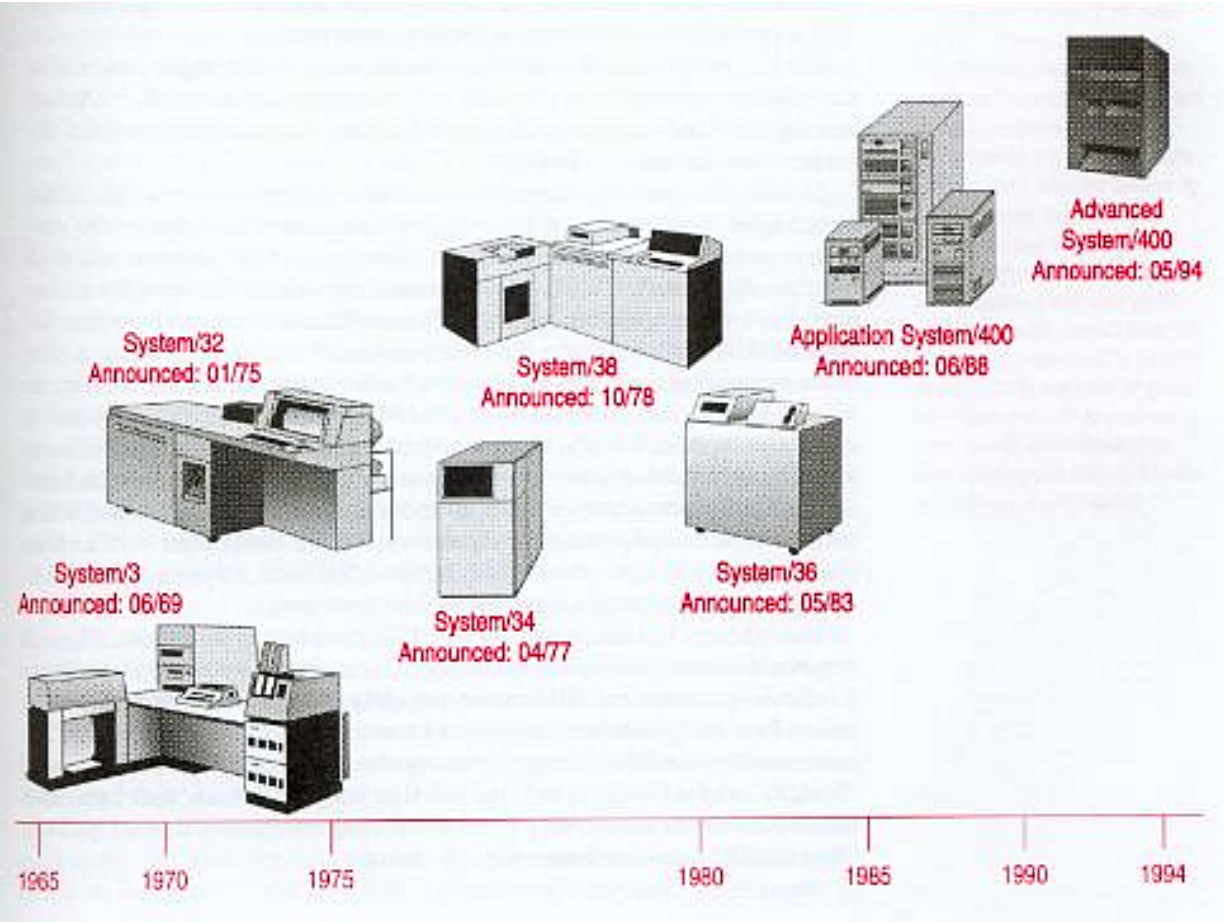
1.3 Architektura PowerPC

W roku 1991 Apple Computer, Motorola oraz IBM podpisały porozumienie w sprawie budowy nowej rodziny procesorów. Ta nowa architektura (PowerPC, POWER = Performance Optimization Witch Enhanced RISC) miała być wielką rewolucją. Zakładano, że może być wykorzystywana do wielu zastosowań od urządzeń przenośnych po superkomputery.

W tym samym czasie ludzie z Rochester poszukiwali nowego procesora dla systemu AS/400. Powstał 64 bitowy procesor bazujący na architekturze PowerPC. Przejście na nową architekturę było stosunkowo bezbolesne. Napisano nowego SLIC’a, aplikacje działające na poprzedniej architekturze 48 bitowej działały „od razu” na nowej platformie.

Przejście z architektury 48 bitowej do 64 bitowej trwało około dwa lata. W tej chwili praktycznie wszystkie stacje działają w nowej architekturze. Porównajmy to z sytuacją na rynku PC, od jak dawna mówi się o 64 bitowych procesorach Intela?

1.4 Historia ewolucji systemu



Rysunek 2: Historia ewolucji od System/3 do AS/400

2 Organizacja pamięci

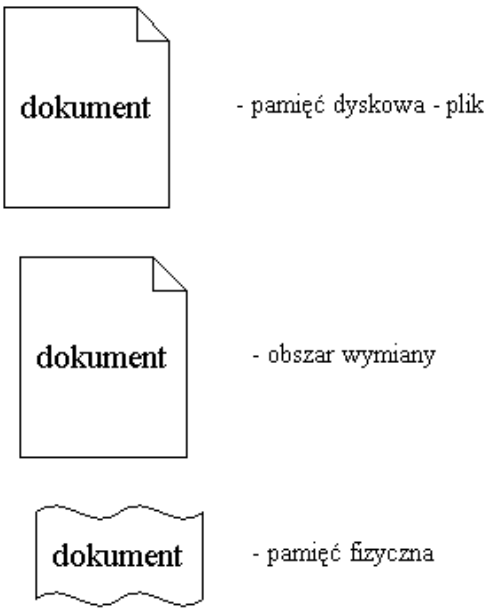
2.1 Two-level store

W konwencjonalnych systemach jest kilka poziomów pamięci:

- fizyczna – tylko w niej procesor może wykonywać operacje
- obszar wymiany – „przedłużenie” pamięci fizycznej
- pamięć dyskowa

Przy czym z punktu widzenia procesu dane mogą występować albo w jego pamięci wirtualnej albo w pliku na dysku.

Rozważmy przypadek gdy w edytorze tekstów zmieniasz plik. Po pierwsze otwierany jest plik dyskowy zawierający dokument. Jego zawartość jest wczytywana do pamięci wirtualnej i z reguły tylko część jest w pamięci fizycznej. Kiedy przejdziesz do innej części dokumentu inny fragment jest wczytywany z obszaru wymiany do pamięci fizycznej. Czyli są dwie kopie naszego dokumentu: w obszarze wymiany i w systemie plików. Część dokumentu jest też w pamięci fizycznej (zobacz rysunek 3).



Rysunek 3: Dwa poziomy pamięci – trzy kopie tych samych danych

Kiedy zakończysz zmieniać dokument zostaniesz zapytany/a czy zapisać zmiany. Innymi słowy czy chcesz aby dane z pamięci wirtualnej zostały trwale zapisane do pliku dyskowego.

2.2 Single-level store

Dwupoziomowa pamięć powoduje pewne dodatkowe obciążenie systemu. Otwarcie dokumentu powoduje zapis do obszaru wymiany. Zamykanie wymaga zapisu na dysk do określonego obszaru.

Nie można tego zrobić tak by mieć tylko jedną kopię pliku?

Bez dwóch poziomów pamięci nie będziemy musieli rezerwować miejsca na plik wymiany. W tym podejściu cały system plików staje się częścią pamięci wirtualnej. System plików nadal utrzymuje skorowidz plików, ale teraz łączy nazwy z odpowiednimi miejscami pamięci wirtualnej. Otwarcie i zamknięcie pliku nie wymaga już fizycznego kopiowania całej jego zawartości z jednego miejsca w drugie. Tylko używany fragment (rekord) jest kopiowany do bufora w pamięci. Można powiedzieć, że plik jest używany „w miejscu".

Tak jak w modelu dwupoziomowym, pamięć fizyczna jest nadal używana jako bufor. Procesor może operować tylko na danych w pamięci fizycznej. Różnica polega na tym, że dla modelu jednopoziomowego pamięć fizyczna jest buforem dla całego obszaru dyskowego. Gdy jeden użytkownik wprowadzi zmiany do pliku to są one od razu widoczne dla każdego innego użytkownika który również używa tego pliku.

Z drugiej strony w single-level store przestrzeń adresowa musi pokryć cały dostępny obszar dyskowy. 32 bity wykorzystywane w większości współczesnych systemów to zdecydowanie za mało.

System/38 oraz pierwsze wersje AS/400 używały 48 bitów do adresowania. Obecnie AS/400 wykorzystuje 64 bitowe procesory z rodziny PowerPC. Tak duży obszar adresowy pozwala zarządzać ogromną ilością danych ($2^{64} = 18'446'744'073'709'551'616$) i wydaje się, że nie szybko ulegnie to zmianie.

2.3 Obiekty w pamięci

Pojęcie pamięci istnieje jedynie poniżej MI (w SLIC). Program powyżej MI widzi i używa tylko obiektów (zobacz rozdział 1.1 i rysunek 1).

Inne obiekty pojawiają się na poziomie OS/400 i inne na poziomie MI. Przykładowe obiekty OS/400:

- dokument
- plik
- folder
- biblioteka
- program

Przykłady systemowych obiektów MI:

- kontekst

- kursor
- obszar danych
- profil użytkownika

Dostęp do obiektu wymaga użycia wskaźnika systemowego, który zajmuje 16 bajtów i zawiera adres obiektu oraz informację o typie wskazywanego obiektu. Wskaźniki są zabezpieczone przed modyfikowaniem (zobacz rozdział 2.5).

Cała pamięć tworzy jedną przestrzeń adresową. Przestrzeń jest podzielona logicznie na segmenty (rozmiaru 16MB, w najnowszej wersji 64MB). Segmenty są rozłączne. Obiekty zawsze zajmują jeden lub więcej segmentów i zawsze rozpoczynają się na granicy segmentów.

Jednopoziomowa pamięć jest podzielona na segmenty, jednak długi adres umożliwia programowi **poniżej** MI na zaadresowanie dowolnego segmentu w całej wirtualnej przestrzeni adresowej, a nie jedynie podzbioru segmentów.

Pamięć jednopoziomowa jest widoczna jedynie przez SLIC. **Powyżej** MI widać obiekty, do których odwołujemy się za pomocą nazw. Obiekty są pogrupowane w biblioteki. Nazwa jest odwzorowywana w adres wirtualny. Jest on przechowywany w bibliotece zawierającej obiekt (biblioteka jest w pewnym sensie skorowidzem – łączy nazwy z odpowiednimi obiektami – coś jak katalog na dysku).

2.4 Strony pamięci

Pamięć składa się z ramek (w architekturze PowerPC mają one rozmiar 4KB). Obiekty są podzielone na strony. Duży obiekt może zajmować wiele stron, pojedyncza strona nigdy nie zawiera więcej niż jednego obiektu. Strona sprowadzona do pamięci przez jeden proces jest dostępna dla każdego innego, np. dowolna liczba zadań może współdzielić instrukcje programu (takie nieograniczone współdzielenie ogranicza konieczność transmisji dyskowych).

Sprowadzanie odbywa się na żądanie. Tablica stron wykorzystywana podczas tłumaczenia adresu wirtualnego na rzeczywisty znajduje się w pamięci operacyjnej. Jeśli strony nie ma w pamięci, to jest generowane przerwanie braku strony: SLIC odszukuje adres dyskowy strony, wysyła odpowiedni komunikat do procesora we-wy z żądaniem sprowadzenia strony. Podczas wymiany stosuje się algorytm **LRU** (ang. *least recently used* – zastępowanie najdawniej używanych stron).

Programy mogą również jawnie żądać:

- sprowadzenia stron do pamięci,
- stworzenia wyzerowanej strony (przydzielenia wyzerowanej ramki pamięci),
- wymiany stron, tzn. program może wskazać strony, które mają być usunięte zamiast tych usuwanych przez standardowy algorytm wymiany.

Ponadto można wskazać strony do **przypięcia** w pamięci (pewne strony muszą być rezydentne).

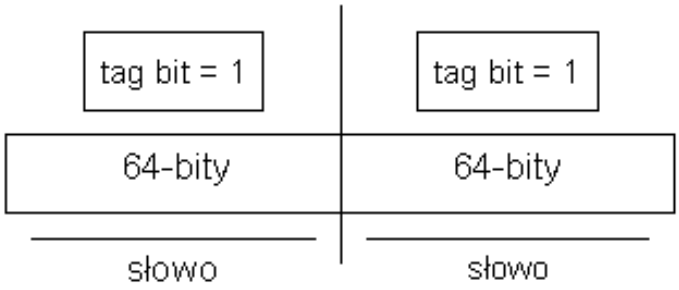
Stronicowaniu podlegają wszystkie obiekty. Pewne struktury SLIC (jak np. tablica stron) i procedury SLIC nie są stronicowane. Ich adres wirtualny jest równoważny adresowi rzeczywistemu.

2.5 Zabezpieczanie wskaźników do obiektów

Obiekt jest jedynym pomostem pomiędzy programem a systemem. Poszczególne obiekty są identyfikowane na podstawie 16-bajtowego wskaźnika, który zawiera m.in. adres wirtualny.

Program powyżej MI nie może sięgać do dowolnego obiektu. W trakcie uruchamiania taki program dostaje pewne wskaźniki do obiektów zapisane przez SLIC. Jeśli program zmieniłby taki wskaźnik, to mógłby korzystać z dowolnych obiektów – jest przecież jedna przestrzeń adresowa.

Aby zapobiec takiej sytuacji w AS/400 stosuje się sprzętową ochronę wskaźników systemowych przed modyfikacją. W tym celu używa się dodatkowego bitu, tzw. bitu znacznikowego (ang. *tag bit*), związanego z każdym słowem pamięci. Słowo ma 64 bity, wskaźnik zajmuje dwa słowa. Za każdym razem, gdy wskaźnik jest wstawiany przez SLIC do dwóch kolejnych słów pamięci, sprzęt ustawia odpowiednie cztery bity znacznikowe na 1 co oznacza, że wskaźnik zawiera poprawny adres (zobacz rysunek 4).



Rysunek 4: Poprawny wskaźnik do obiektu

Jeśli użytkownik zmienił dowolną część wskaźnika, to sprzęt wyłączył bit znacznikowy. Przy odwołaniu do obiektu (np. przez wykonanie metody na obiekcie) SLIC sprawdza czy wszystkie bity znacznikowe wskaźnika są ustawione Jeśli jakiś bitów był zgaszony, to wskaźnik jest traktowany jako błędny i nie może być użyty do zaadresowania pamięci.

Do obsługi bitów znacznikowych służą specjalne instrukcje maszynowe, dostępne jedynie w specjalnym trybie pracy procesora (ang. *tag-active mode*). Instrukcje te mogą być używane jedynie przez SLIC.

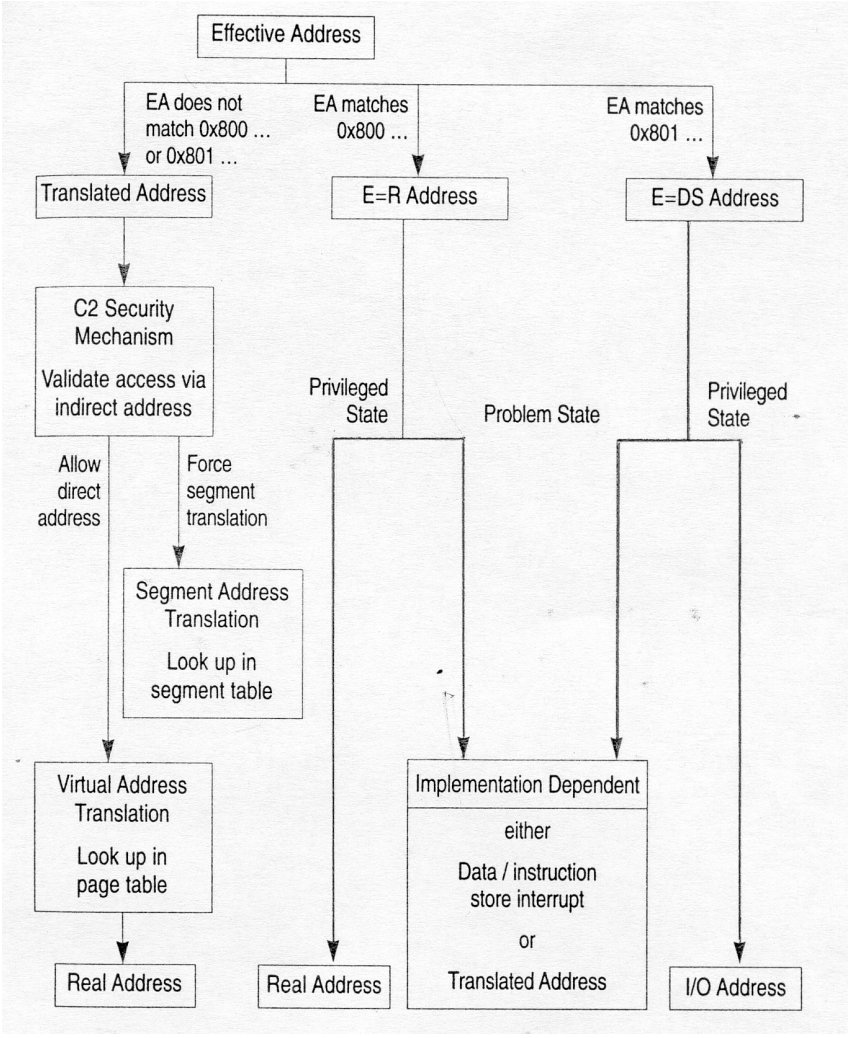
Warto zauważyć, że taki sposób zabezpieczenia działa inaczej niż np. zabezpieczanie segmentów kodu w Linuxie przed zapisem. Tu program może zmienić wskaźnik, może przypisać mu nową wartość. Błąd pojawia dopiero gdy spróbuje skorzystać z nowego wskaźnika.

Bity znacznikowe wymagają dodatkowej pamięci (z reguły wykorzystuje się wolny bit z korekcji ECC) i muszą być pamiętane przy zapisywaniu ramki na dysk fizyczny. Pamięć dyskowa jest podzielona na bloki 520 bajtowe. Jedna strona (ramka) pamięci fizycznej to 4KB. Ustalono, że blok dyskowy nie może zawierać danych z różnych ramek – czyli jedna ramka zawsze zajmuje 8 bloków (4160B). Każdy taki fragment dyskowy ma 64 bajty wolne. Jedna ramka może zawierać co najwyżej 256 wskaźników. To daje dodatkowe 32 bajty na tag-bit które można upchnąć w wolnym miejscu bloków.

3 Tłumaczenie adresów

3.1 Sposób tłumaczenia adresów

Rysunek 5 pokazuje jak przebiega tłumaczenie adresów w architekturze PowerPC w trybie *tag-active*.



Rysunek 5: Tłumaczenie adresów w architekturze PowerPC

Sprzęt sprawdza czy 64 bitowy adres efektywny wygenerowany przez program jest: adresem do tłumaczenia, typu E=R, czy typu E=DS. Decyduje o tym 12 najstarszych bitów (trzy znaki w zapisie hex).

E=R ($0x800$**)**

Jeśli 12 najstarszych bitów adresu efektywnego wynosi $0x800$, adres jest typu E=R (**ef-**

fective=real), tzn. pozostała część adresu (52 bity) jest bezpośrednio odwzorowana do pamięci fizycznej (rzeczywistej).

Następnie sprzęt testuje czy proces może wykonywać uprzywilejowane instrukcje – czy ustawiono odpowiedni tryb pracy. Jeśli tak pozostałe 52 bity stają się adresem fizycznym. W przeciwnym przypadku procesor może podnieść wyjątek lub potraktować adres jako zwykły adres do tłumaczenia (translated address). To zależy od zastosowanego procesora.

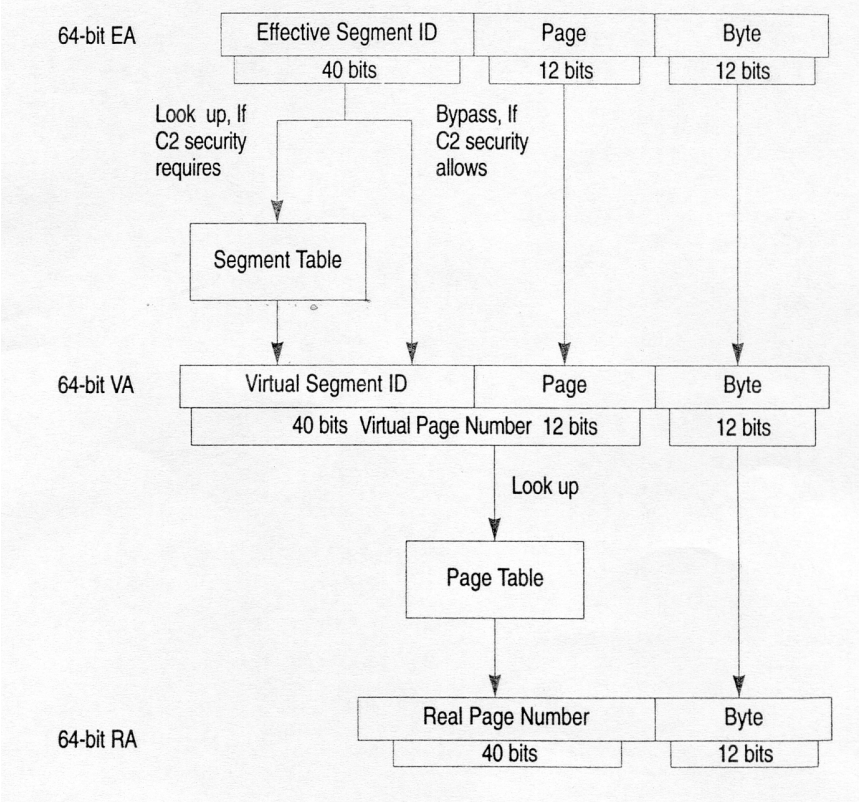
Jest 2^{52} takich adresów ($64 - 12 = 52$).

E=DS, ($0x801$**)**

Jeśli 12 najstarszych bitów adresu efektywnego wynosi $0x801$, adres jest typu E=DS (**effective=direct-store access**). Adres jest przekazywany do podsystemu I/O i z reguły jest wykorzystywany do identyfikacji urządzenia. Procesor stosuje taką samą metodę sprawdzania trybu pracy jak w przypadku E=R.

Translated Address

Gdy adres efektywny nie pasuje do żadego z powyższych typów sprzęt próbuje go przetłumaczyć na adres wirtualny (zobacz rysunek 6).



Rysunek 6: Tłumaczenie adresów typu Translated Address

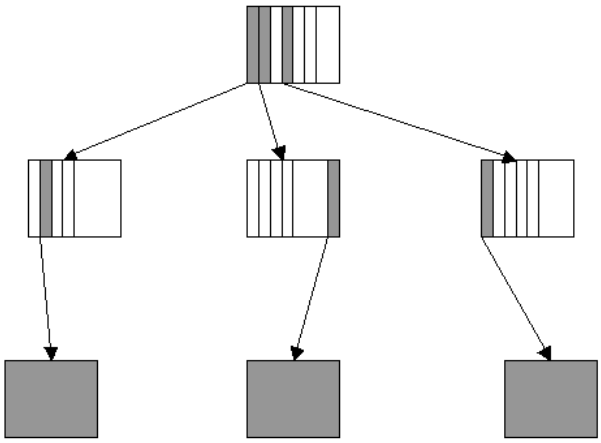
Jeśli włączono poziom zabezpieczeń C2 sprzęt zagląda do tablicy segmentów by określić numer strony z przestrzeni wirtualnej.

Poziom C2 jest rzadko używany. W większości przypadków tablica segmentów nie jest przeglądana i adres wirtualny (**VA**)jest adresem efektywnym (**EA**).

Do tłumaczenia VA na 52 bitowy adres fizyczny wykorzystuje się tablicę stron.

3.2 Wady wielopoziomowej tablicy stron

Systemy Linux i Windows korzystają z tablic wielopoziomowych. Takie rozwiązanie nie daje optymalnego wykorzystania pamięci. Rozważmy przykład z rysunku 7.



Rysunek 7: Nieefektywne wykorzystanie tradycyjnej tablicy stron

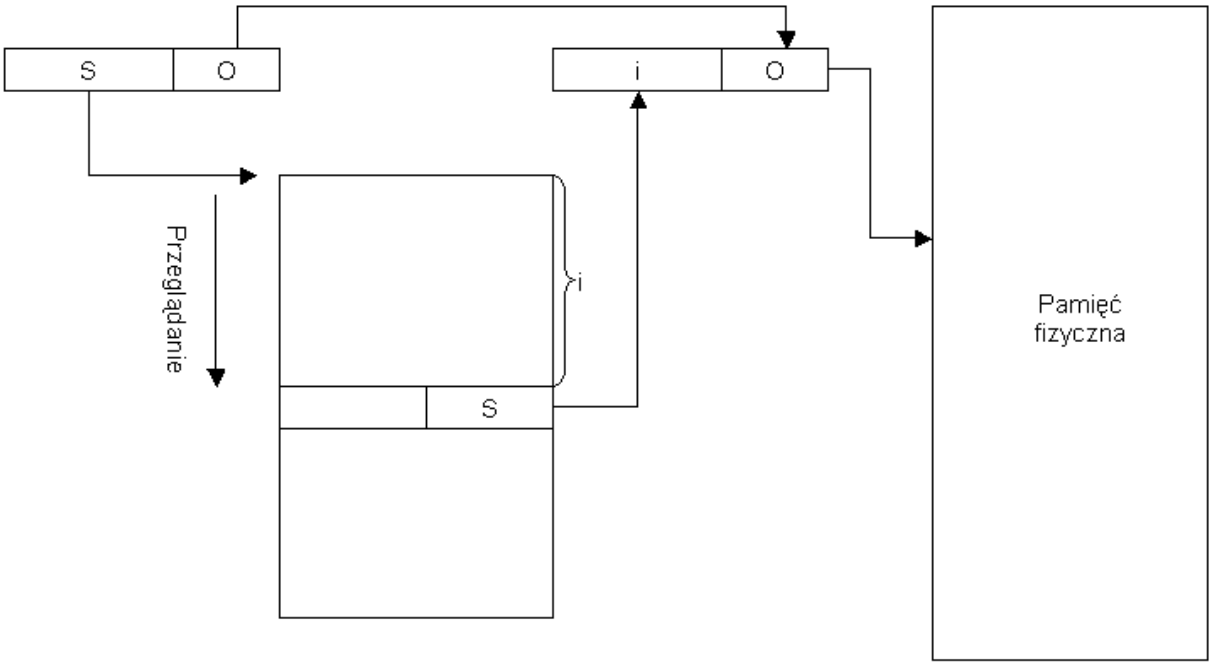
Proces używa tylko trzech stron ale tablica stron musi zajmować aż cztery ramki pamięci fizycznej.

Wspomniane systemy dysponują 32 bitową przestrzenią wirtualną. Tablica stron zajmuje wtedy 4MB. Przy adresowaniu 64 bitowym tablica stron byłaby zbyt duża – sama struktura zajęłaby całą pamięć fizyczną.

3.3 Odwrócona tablica stron

W systemie AS/400 zastosowano tzw. odwróconą tablicę stron. Zawiera ona po jednej pozycji dla każdej rzeczywistej strony pamięci. Każda pozycja zawiera adres wirtualny strony przechowywanej w ramce rzeczywistej pamięci (zobacz rysunek 8).

Opisany schemat zmniejsza rozmiar pamięci potrzebnej do pamiętania wszystkich tablic stron, jednak zwiększa czas potrzebny do przeszukania tablicy przy odwołaniu do strony. Ponieważ odwrócona tablica stron jest uporządkowana według adresów fizycznych, a przeglądanie dotyczy adresów wirtualnych, zatem w celu znalezienia dopasowania należy przeszukać ją w całości. Szukanie takie może trwać o wiele za długo.



Rysunek 8: Odwrócona tablica stron zawiera po jednym wpisie dla każdej ramki pamięci fizycznej.

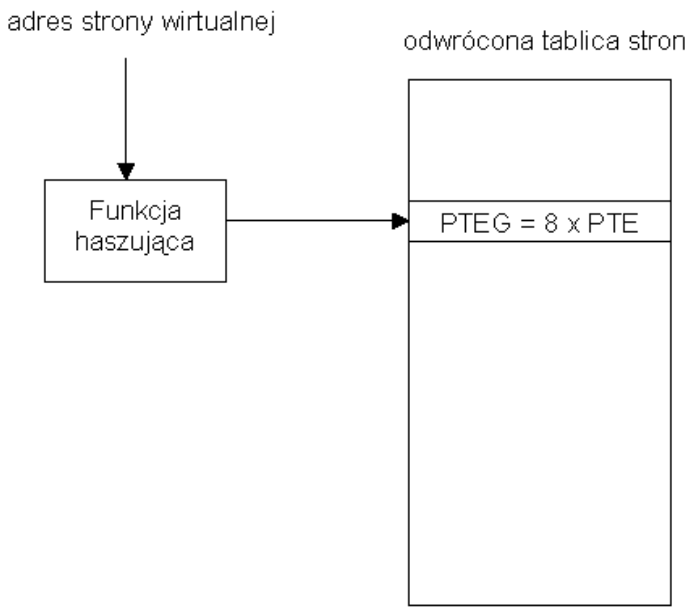
3.4 Wyszukiwanie w odwróconej tablicy stron

Aby przyspieszyć wyszukiwanie pozycji w odwróconej tablicy stron wykorzystuje się haszowanie. Funkcja haszująca wykonuje XOR pewnej liczby bitów z prawej strony adresu wirtualnego z pewną liczbą bitów z lewej strony adresu wirtualnego, wykonuje AND z odpowiednią maską pobraną z rejestru, w którym przechowuje się rozmiar tablicy stron. Wreszcie na wyniku wykonuje OR z rzeczywistym adresem tablicy stron.

Każda pozycja odwróconej tablicy stron zawiera jeden **PTEG** (ang. *page table entry group*. Każdy PTEG zawiera osiem wpisów **PTE** (ang. *page table entry*). Korzystając z funkcji haszującej procesor odnajduje PTEG, który przegląda w poszukiwaniu odpowiedniego PTE (zobacz rysunek 9).

Tablica w AS/400 ma dwa razy mniej pozycji niż jest ramek pamięci fizycznej. Oznacza to, że przy optymalnym rozkładzie każdy PTEG zawiera tylko dwa PTE. Oczywiście w rzeczywistości może się zdarzyć, że osiem pozycji to za mało. W takim przypadku stosuje się drugą tablicę stron, która może przechowywać dowolnie dużo wpisów. Druga tablica stron ma zmienny rozmiar i używa innej funkcji haszującej.

Procesor PowerPC dysponuje rejestrami TLB (ang. *translation lookaside buffer*) które przechowują ostatnio używane połączenia **adres wirtualny – adres rzeczywisty**. Okazuje się, że w średnio 95% przypadków wystarczy skorzystać z TLB by przeliczyć adres. Ponieważ jest tylko jedna tablica stron nie ma potrzeby czyszczenia TLB przy przełączaniu kontekstu co daje

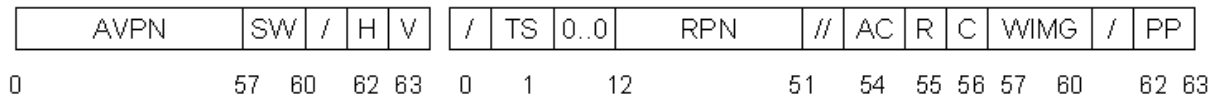


Rysunek 9: Przeszukiwanie odwróconej tablicy stron.

wyjątkowo dużą skuteczność trafień.

3.5 Zawartość PTE

Każdy wpis z odwróconej tablicy stron (PTEG) zawiera informację o ośmiu ramkach pamięci (PTE). PTE zajmuje 16 bajtów (dwa słowa) i ma budowę jak na rysunku 10.



Rysunek 10: Zawartość PTE.

Znaczenia pól:

Bity	**Nazwa**	**Opis**
0–56	AVPN	*„Skrócony”* numer strony wirtualnej. Skrócony w stosunku do architektury PowerPC która obsługuje 80 bitowe adresy wirtualne.
57–60	SW	Zarezerwowane dla zarządcy pamięci w SLIC.
62	H	Czy PTE jest w podstawowej czy drugiej tablicy stron.

Bity	Nazwa	Opis
63	V	Poprawność wpisu (0=„pusty” wpis).
1	TS	Czy strona zawiera jakieś wskaźniki do obiektów – czy istnieje bit znacznikowy ustawiony na 1.
2–11		Zawsze równe 0.
12–51	RPN	Numer strony fizycznej.
54	AC	Uruchamia mechanizm porównywania danych (wykrywa dostęp do bloków pamięci).
55	R	Ustawiany przy każdym odwołaniu do strony (ang. *reference*).
56	C	Ustawiany przy każdej zmianie zawartości strony (ang. *change*).
57–60	WIMG	Tryb dostępu do pamięci: **W** (write throught) każda zmiana danych w pamięci podręcznej (cache) musi być zapisana do pamięci głównej. **I** (caching inhibited) dostęp do ramki tylko z pominięciem pamięci podręcznej (cache). **M** (memory coherence) dane zapisywane przez procesor do jednej komórki pamięci mogą być przez kontroler pamięci zapisywane w innej kolejności (w celu zwiększenia wydajności). Ustawienie bitu wyłącza ten mechanizm – dane są zapisywane w kolejności przyjścia. **G** (guarded storage) instrukcje mogą być wykonywane z wyprzedzeniem w potokach. Jednak nie zawsze procesor dobrze przewidzi kolejną instrukcję. Czasami rozpoczęta instrukcja musi być cofnięta. W takim przypadku pamięć nie zawsze reaguje natychmiastowo. Ustawienie bitu zapobiega takiej sytuacji.
62–63	PP	Tryb dostępu – zobacz rozdział 3.6.

3.6 Tryb dostępu do ramki

Strony mają określone uprawnienia dostępu które definiują bity PP z PTE (zobacz rozdział 3.5).

Klucz	PP	Tryb dostępu
0	00	odczyt i zapis

Klucz	PP	Tryb dostępu
0	01	odczyt i zapis
0	10	odczyt i zapis
0	11	tylko odczyt
1	00	brak dostępu
1	01	tylko odczyt
1	10	odczyt i zapis
1	11	odczyt i zapis

Jeśli system nie jest w trybie C2 to klucz jest równy bitowi trybu pracy:

0 tryb systemowy – uprzywilejowany

1 tryb użytkownika

Dla trybu C2 klucz ma wartość $(K_P and MSR_{PR}) or (K_S and not MSR_{PR})$ gdzie:

K_P, K_S stan krytyczny pamięci, stan krytyczny systemu – zapisywane w tablicy segmentów

MSR_{PR} 0 jeśli procesor może wykonywać dowolne instrukcje, 1 w przeciwnym przypadku

4 Zalety single-level store

4.1 Szybkie przełączanie kontekstu

Ponieważ jest tylko jedna tablica stron przełączenie kontekstu jest bardzo szybkie. Eksperymentalnie zbadano, że średnio wykonuje się je raz na 1200 instrukcji, podczas gdy w niektórych systemach sama operacja przełączenia kontekstu wymaga około 1000 instrukcji. Dzięki temu AS/400 zapewnia dobrą wydajność w środowisku interakcyjnym, zorientowanym na aplikacje działające w trybie transakcji. Można dołączać wiele terminali. Pojedynczy duży AS/400 może obsługiwać ponad 2000 działających współbieżnie użytkowników.

4.2 Lepsze wykorzystanie TLB

W AS/400 jest tylko jedna tablica stron dla wszystkich użytkowników. Nie ma więc potrzeby czyszczenia TLB podczas przełączania kontekstu. To daje dużą skuteczność trafień. Okazuje się, że w średnio 95% przypadków wystarczy skorzystać z TLB by przeliczyć adres.

5 Literatura

1. F.Soltis – „Inside the AS/400”
2. http://rainbow.mimuw.edu.pl/SO-MSUI/7as400/as400.html – strona wykładu „Współczesne systemy operacyjne” I roku MSUI.
3. http://www.as400.ibm.com
4. http://www-5.ibm.com/pl/as400
5. http://www-912.ibm.com – „iSeries and AS/400 Technical Support”
6. http://publib.boulder.ibm.com/pubs/html/as400/v4r4/ic2978/info/index.html – „Centrum informacyjne AS/400”