ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1995

Seria: INFORMATYKA z. 29 Nr kol. 1306

Urszula STAŃCZYK

PRZEŁĄCZANIE TRYBÓW PRACY PROCESORÓW 80386 I i486

Streszczenie. Procesory takie jak 80386 i i486 mogą pracować w dwóch podstawowych trybach: trybie adresowania rzeczywistego albo w trybie adresowania wirtualnego z ochroną. W trybie wirtualnym dostępnych jest bardzo dużo możliwości – między innymi mechanizmy ochrony, wielozadaniowości i pamięci wirtualnej. Aby przełączenie pomiędzy trybami było możliwe, konieczne jest wykonanie pewnych określonych działań programowych, nazywanych algorytmem przełączania trybów pracy procesora.

SWITCHING PROCESSORS 80386 AND i486 MODES OF WORK

Summary. Microprocessors like 80386 and i486 can work in two basic modes: real address mode or virtual protected mode. In protected mode there are available a lot of features – like protection, multitasking, virtual memory and so on. For succesfull switching between modes there is necessery to complete some special programming actions, called an algorithm of switching between mikroprocessor's modes.

ПЕРКЛЮЧЕНИЕ РЕЖИМОВ РАБОТЫ ПРОЦЕССОРОВ 80386 И и486.

Резюме. Микропроцессоры такие как 80386 и и486 могут работать в двох основных режимах: режиме действительной адресации или режиме виртуальной адресации со зщитой. В виртуальном режиме есть доступ к очень много возможностей – как механизм защиы, многозадачность и виртуальная память. Для правилного переключения между режимами нужно исполнить некоторое определенные програмические деятельности, которые называем алгоритмом переключения режимов работы процессора.

1. Możliwości procesorów serii Intel 80386DX i i486

Procesory 80386 i i486 są może nie najnowszymi z serii firmy Intel 80x86 ale obecnie chyba najbardziej popularnymi procesorami. Mimo że upłynęło już trochę lat, odkąd pojawiły się na rynku, wiele z ich możliwości nie doczekało się pełnego wykorzystania w codziennej pracy programistów (po części ze względu na brak odpowiedniej dokumentacji). Poniższe punkty w sposób skrótowy i uproszczony opisują podstawowe cechy obu procesorów.

1.1. Tryb adresowania rzeczywistego

Tryb adresowania rzeczywistego (*ang. real address mode*) jest automatycznie włączany podczas inicjalizacji procesora. Architektura w tym trybie jest prawie identyczna z architekturą procesorów 8086, 8088, 80186 i 80188. Z punktu widzenia programisty procesory 80386 i i486 w trybie adresowania rzeczywistego wydają się szybkim procesorem 8086 z rozszerzoną listą rozkazów i rejestrów. Najbardziej charakterystyczne cechy tego trybu, to:

– sposób formowania adresu,
– obsługa przerwań i wyjątków[1].

1.1.1. Formowanie adresu

W trybie adresowania rzeczywistego procesor tworzy adres w taki sam sposób jak procesor 8086: mnoży zawartość odpowiedniego rejestru segmentowego przez 16 (przesuwa o 4 bity w lewo) dla uzyskania 20–bitowego adresu początku segmentu i do niego dodaje wartość przesunięcia (rozszerzoną o 4 wyzerowane bity na najstarszych pozycjach) wewnątrz segmentu.

Ponieważ w wyniku dodawania może wystąpić przeniesienie, więc wynikowy adres liniowy może mieć 21 bitów (w procesorze 8086 z powodu istnienia tylko dwudziestu linii adresowych taki wypadek powoduje tzw. zawijanie się adresu wokół granicy 1MB), więc istnieje możliwość zaadresowania maksymalnie 1MB+64KB–16B (maksymalny adres to

[1] Pod pojęciem przerwania (*ang. interrupt*) rozumie się tutaj zewnętrzne przerwanie sprzętowe, czyli ustawienie na wejściu INTR lub NMI procesora sygnału aktywnego. Wyjątek (*ang. exception*), generowany wewnętrznie przez procesor, jest związany z aktualnie wykonywaną instrukcją. Tzw. przerwanie programowe (rezultat wykonania instrukcji INT) również zalicza się do kategorii wyjątków.

10FFEFH) liniowej przestrzeni adresowej (*ang. linear address space*). Adres liniowy jest w tym trybie równy adresowi fizycznemu.

Procesory 80386 i i486 mają możliwość generowania 32–bitowego adresu efektywnego (czyli przesunięcia w adresie logicznym), jeżeli wykorzystywany jest specjalny przedrostek rozmiaru adresu. Jednakże nawet wówczas rozmiar segmentu nie może przekroczyć granicy 64KB.

1.1.2. Obsługa przerwań i wyjątków

Obsługa przerwań i wyjątków w trybie adresowania rzeczywistego procesora przebiega w taki sam sposób jak w procesorze 8086. Adresy procedur obsługi zapisane są w tablicy przerwań IDT (chociaż nosi ona swoją angielską nazwę – *Interrupt Descriptor Table* – Tablica Deskryptorów Przerwań, to w tym trybie nie przechowuje deskryptorów, lecz dalekie wskaźniki – adres początku segmentu i przesunięcie – do punktów wejścia do procedur obsługi). W procesie obsługi procesor mnoży identyfikator przerwania (tzw. wektor) przez 4 – długość elementu tablicy i dodając do adresu bazowego całej tablicy otrzymuje adres komórki, zawierającej wskaźnik do właściwej procedury obsługi. W momencie wystąpienia przerwania procesor odkłada na stosie aktualną zawartość rejestrów CS (*ang. code segment*) i IP (*ang. instruction pointer*), blokuje przerwania maskowalne (zeruje znacznik przerwań IF – *ang. interrupt flag*), zeruje znacznik TF (*ang. trap flag*) i przenosi sterowanie pod adres określony przez pozycję tablicy przerwań. Instrukcja IRET, kończąca procedurę obsługi przerwania, przeprowadza operacje odwrotne przed powrotem do przerwanej procedury.

Podstawową różnicą pomiędzy obsługą przerwań i wyjątków w procesorach 80386 i i486 w porównaniu do procesora 8086 jest to, że położenie tablicy przerwań jest zależne od zawartości rejestru IDTR (*ang. Interrupt Descriptor Table Register* – Rejestr Tablicy Deskryptorów Przerwań). Po inicjalizacji rejestr IDTR zawiera adres bazowy 0 i limit równy 3FFH (co jest zgodne z procesorem 8086) i zwykle programy nie zmieniają tych wartości.

Tablica przerwań może zawierać maksymalnie 256 wskaźników do procedur obsługi. Ponieważ każdy wskaźnik ma długość 4 bajtów, więc rozmiar tablicy wynosi 1KB.

1.2. Tryb adresowania wirtualnego z ochroną

Tryb adresowania wirtualnego z ochroną (*ang. virtual address protected mode*) umożliwia wykorzystywanie wszystkich mechanizmów zaimplementowanych w architekturze procesorów 80386 i i486. O przełączeniu do trybu z ochroną informuje ustawienie bitu PE (*ang. protection enable*) rejestru sterującego CR0 (jego młodsza połowa to rejestr MSW w procesorze 80286). Format rejestru CR0 pokazuje rysunek rys. 1. Wyzerowanie bitu PE lub ustawienie sygnału RESET procesora umożliwia powrót do trybu adresowania rzeczywistego.

Najbardziej charakterystyczne elementy trybu z ochroną to:

- sposób formowania adresu,
- obsługa przerwań i wyjątków,
- mechanizmy ochrony,
- wielozadaniowość,
- tryb wirtualny 8086.

31	30 … 5	4				0	
PG	0	R	TS	EM	MP	PE	CR0

Rys. 1. Rejestr sterujący CR0
Fig. 1. Control Register CR0

1.2.1. Formowanie adresu

W trybie wirtualnym z ochroną w formowaniu adresu można wyróżnić dwa etapy: segmentację i stronicowanie. Segmentacja jest wykorzystywana dla zapewnienia każdemu uruchamianemu programowi własnej, niezależnej i chronionej przestrzeni adresowej. Stronicowanie jest używane w celu sprawnej gospodarki pamięcią operacyjną i organizacji środowiska, w którym duża przestrzeń adresowa jest symulowana za pomocą małej ilości pamięci operacyjnej RAM i przechowywania danych na dysku. Projektanci systemu mogą wybrać jeden lub oba mechanizmy. Jeśli w tym samym czasie uruchomionych jest kilka programów, zarówno segmentacja jak i stronicowanie mogą być stosowane do ochrony programów przed ingerencją innych programów.

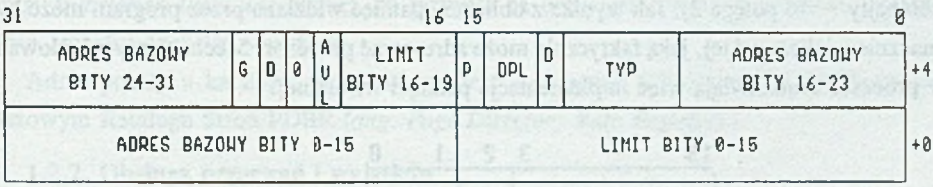
Każdy segment jest niezależną przestrzenią adresową, do której dostęp jest kontrolowany przez dane, które opisują jej rozmiar, poziom ochrony wymagany dla dostępu, rodzaj odwołań, które mogą być wykonane i oczywiście obecność w pamięci operacyjnej. Translacja adresu na poziomie segmentacji tłumaczy adres logiczny, o postaci selektor (*ang. selector*) i przesunięcie (*ang. offset*), na adres w ciągłej przestrzeni adresów liniowych (*ang. linear address space*).

Stronicowanie jest włączane przez ustawienie bitu PG (*ang. paging enable*) rejestru CR0. Jeżeli nie jest wykorzystywane, adres liniowy jest równy fizycznemu, tak jak w trybie adresowania rzeczywistego.

1.2.1.1. Segmentacja

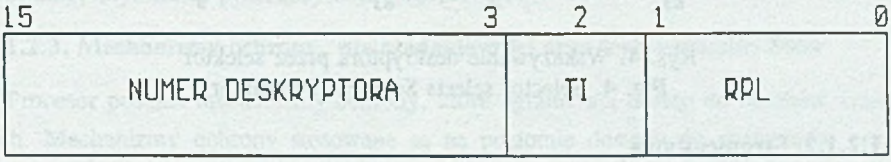
W trybie z ochroną adres logiczny składa się z 16-bitowego selektora segmentu i 32-bitowego przesunięcia wewnątrz segmentu. Adres logiczny tłumaczony jest na liniowy przez dodanie offsetu do adresu początku segmentu. Adres początku pochodzi z deskryptora segmentu (*ang. segment descriptor*), struktury danych przechowującej rozmiar i położenie segmentu, jak również informacje kontrolne. Format deskryptora pokazuje rysunek rys. 2.

Deskryptory segmentów zgrupowane są w tablicach deskryptorów, znajdujących się w pamięci. W systemie istnieją dwa rodzaje tablic deskryptorów: Globalna Tablica Deskryptorów GDT (*ang. Global Descriptor Table*) i Lokalne Tablice Deskryptorów LDT (*ang. Local Descriptor Table*). Dla wszystkich programów w systemie dostępna jest jedna tablica GDT i jedna tablica LDT dla każdego oddzielnego programu, który jest uruchamiany. Tablica LDT należy do tzw. segmentów systemowych procesora i sama posiada deskryptor, który musi być umieszczony w tablicy globalnej.



Rys. 2. Format deskryptora
Fig. 2. Format of Segment Descriptor

Każdy adres logiczny jest związany z segmentem i chociaż program może mieć tysiące segmentów, tylko sześć jest gotowych do natychmiastowego użycia. Są to segmenty, których selektory znajdują się w rejestrach segmentowych procesora. Podczas ładowania rejestru selektorem automatycznie do ukrytej części rejestru (jest to ukryty rejestr deskryptora) ładowany jest adres początku segmentu, limit i informacje kontrolne. Procesor nie musi się odwoływać do tablic deskryptorów aż do momentu kolejnego ładowania selektora segmentu.

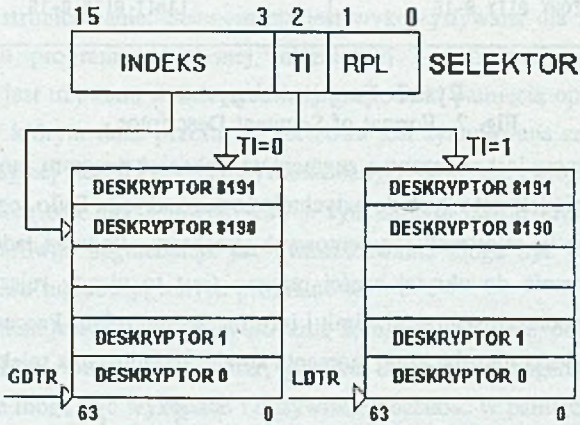


Rys. 3. Format selektora segmentu
Fig. 3. Format of Segment Selector

Selektor segmentu, którego format pokazuje rysunek rys. 3, zawiera 13-bitowy numer, będący indeksem do tablicy deskryptorów, który mnożony jest przez 8 (rozmiar deskryptora)

i dodawany do adresu bazowego tablicy deskryptorów. Adresy początków tablic przechowywane są w rejestrach GDTR (*ang. Global Descriptor Table Register* – Rejestr Globalnej Tablicy Deskryptorów) i LDTR (*ang. Local Descriptor Table Register* – Rejestr Lokalnej Tablicy Deskryptorów). Bit TI selektora (*ang. Table Indicator* – Wskaźnik Tablicy) wskazuje, która tablica ma być wykorzystywana do translacji adresu. Wartość bitu równa 0 oznacza używanie tablicy globalnej, wartość bitu równa 1 oznacza używanie LDT. Wskazywanie deskryptora przez selektor pokazuje rysunek rys. 4.

Ponieważ na indeks do tablic deskryptorów przeznaczono 13 bitów, więc każda z tablic może zawierać 8192 deskryptory. Rozmiar logicznej przestrzeni adresowej to maksymalna wielkość pamięci widziana przez pojedyncze zadanie. Ponieważ program ma dostęp do tablicy GDT i jednej LDT, potencjalnie może więc korzystać z 16384 segmentów (8192 deskryptory w GDT i 8192 deskryptory w LDT). Maksymalna długość każdego segmentu to 4GB (32 bity przesunięcia), więc rozmiar pamięci logicznej to 16KB*4GB=64TB (terabajty – 46 potęga 2). Jak wynika z obliczeń, pamięć widziana przez program może być znacznie większa od tej, jaką faktycznie może adresować procesor. Mechanizmy wbudowane w procesor umożliwiają więc implementację pamięci wirtualnej.



Rys. 4. Wskazywanie deskryptora przez selektor
Fig. 4. Selector selects Segment Descriptor

1.2.1.2. Stronicowanie

Adres liniowy (*ang. linear address*) jest 32–bitowym adresem w ciągłej przestrzeni adresowej. Ta przestrzeń adresowa może być tak duża, jak maksymalna wartość adresu (4GB) lub może być symulowana z wykorzystaniem małej ilości pamięci operacyjnej RAM i przechowywaniem danych na dysku ze stosowaniem systemu stronicowania. Jeżeli stronicowanie jest używane, to tłumaczy adres liniowy na odpowiadający mu adres fizyczny

i jest całkowicie niezależne od procesu segmentacji. Stronicowanie jest również wykorzystywane dla efektywniejszej gospodarki pamięcią operacyjną. Operowanie na fragmentach pamięci o stałej wielkości 4KB (są to tzw. strony – *ang. pages*) zabezpiecza przed możliwością powstawania fragmentacji pamięci, która może wystąpić przy działaniach na segmentach o różnej długości. Cała przestrzeń adresowa 4GB dzielona jest na strony, co daje 1M stron, rozmieszczonych sekwencyjnie, począwszy od adresu 0. Położenie strony jest jednoznacznie określane za pomocą 20–bitowego numeru.

Informacje, pozwalające na translację adresu liniowego na adres fizyczny, zapisane są w specjalnych strukturach danych w pamięci, zwanych tablicami stron (*ang. page tables*). Dla lepszej organizacji zastosowano dwa poziomy takich tablic, tzw. katalogi stron (*ang. page directories*) i tablice stron (*ang. page tables*). Podobnie jak w przypadku ukrytych rejestrów deskryptorów, dla przyspieszenia procesu translacji stronicowania procesor przechowuje opisy najczęściej używanych stron w specjalnym pomocniczym buforze stronicowania TLB (*ang. translation lookaside buffer*), co pozwala na uniknięcie częstych odwołań do tablic stron, znajdujących się w pamięci.

Adres początku katalogu stron zawarty jest w rejestrze CR3, zwanym też Rejestrem Bazowym Katalogu Stron PDBR (*ang. Page Directory Base Register*).

1.2.2. Obsługa przerwań i wyjątków

W trybie z ochroną tablica deskryptorów przerwań zawiera 8–bajtowe deskryptory specjalnego typu. Są to tzw. deskryptory furtek (*ang. gate descriptors*). Tablica IDT może zawierać trzy rodzaje deskryptorów furtek:

– furtki zadania (*ang. task gates*),
– furtki przerwania (*ang. interrupt gates*),
– furtki wyjątku (*ang. trap gates*).

W chwili wystąpienia przerwania lub wyjątku wektor wskazuje pozycję w tablicy IDT, z której pobierane są informacje co do adresu, pod jaki ma być następnie przeniesione sterowanie, czyli adres procedury obsługi przerwania.

1.2.3. Mechanizmy ochrony, wielozadaniowości oraz tryb wirtualny 8086

Procesor posiada mechanizmy ochrony, które ograniczają dostęp do zasobów systemowych. Mechanizmy ochrony stosowane są na poziomie dostępu do segmentów oraz na poziomie dostępu do stron, o ile stronicowanie jest wykorzystywane.

Ochrona segmentu polega na sprawdzaniu, czy wykonywany program ma prawo do skorzystania z zasobów systemowych. Jest to uzależnione od tzw. poziomu ochrony (*ang. protection level*) czy też uprzywilejowania. Wyróżniane są cztery poziomy ochrony. Poziom zerowy rezerwowany jest dla jądra systemu operacyjnego i zapewnia mu największe możliwe

uprzywilejowanie i największą ochronę. Poziomy pierwszy i drugi wykorzystywane są przez różne części oprogramowania systemowego, w tym różne części systemu operacyjnego, które mają już mniejsze uprzywilejowanie i mniej są chronione. Najmniej chroniony jest poziom trzeci, przeznaczony dla programów użytkowych, które są najmniej uprzywilejowane.

Informacje o poziomach uprzywilejowania zapisane są w deskryptorach i selektorach segmentów.

Na poziomie stronicowania również wyróżnia się poziomy uprzywilejowania, przy czym istnieją tylko dwa: poziom systemowy (dla segmentów z poziomem ochrony 0, 1 lub 2) i poziom użytkowy (dla segmentów z poziomem ochrony 3). Informacje o poziomach ochrony stron zapisane są w elementach katalogów i tablic stron.

Procesory 80386 i i486 umożliwiają wykorzystywanie mechanizmów wielozadaniowości (*ang. multitasking*), czyli jednoczesnego uruchomienia kilku programów na zasadzie oddzielnych zadań, z których każde ma zapewnioną chronioną niezależną przestrzeń adresową, własny stos i dostęp do innych zasobów procesora. W danej chwili może wykonywać się tylko jedno zadanie, a stany zadań zawieszonych zapisywane są w specjalnych segmentach stanu zadania TSS (*ang. Task State Segment*). Przy stosowaniu wielozadaniowości często wykorzystywany jest tryb wirtualny 8086.

Tryb wirtualny 8086 (*ang. virtual 8086 mode*) jest w zasadzie podtrybem trybu wirtualnego z ochroną i umożliwia wykonywanie programów przeznaczonych dla mikroprocesora 8086 (a także 8088, 80186 i 80188) w środowisku trybu z ochroną. Każdy program 8086[2] wykonywany w trybie wirtualnym 8086 wykonuje się faktycznie na pewnym procesorze wirtualnym. Liniowa przestrzeń adresowa tego procesora obejmuje dla każdego zadania obszar 1MB pamięci adresowany od 0, tak jak w procesorze 8086. W taki sam sposób realizowana jest też translacja adresu logicznego na liniowy (mimo że faktycznie procesor pracuje w trybie z ochroną). Możliwe jest jednak stosowanie stronicowania. W tym trybie procesory 80386 i i486 wykonują poprawnie instrukcje procesora 8086 (kilka różnic wynika z konieczności wykonywania programu w środowisku wielozadaniowym). Obsługa przerwań i wyjątków jest realizowana tak jak w trybie z ochroną (tablica IDT rzeczywiście zawiera deskryptory a nie wskaźniki tak jak w trybie rzeczywistym).

Dla prawidłowego wykonywania programów w trybie wirtualnym 8086 potrzebne są pewne elementy programowe, tworzące tzw. monitor V86[3] (*ang. virtual 8086 monitor*).

---

[2] Przez pojęcie program 8086 rozumie się tutaj kod programu nie wykorzystującego żadnych specjalnych mechanizmów, charakterystycznych dla danego procesora a jedynie te instrukcje, które są wspólne dla wszystkich procesorów serii 80x86.

[3] Patrz [4]

Celem monitora jest zapewnienie bezkonfliktowego współistnienia zadania 8086 z innymi zadaniami w systemie.

Pracując w trybie wirtualnym 8086 procesor zawsze wykonuje programy z poziomem uprzywilejowania równym 3.

Tryb wirtualny 8086 sygnalizowany jest przez ustawienie bitu VM (*ang. virtual 8086 mode*) rejestru znaczników EFLAGS. Ze względu na ograniczenia w możliwości modyfikowania tego bitu przełączenie może nastąpić wskutek przełączenia zadań lub po wykonaniu instrukcji IRET bez zmiany zadania. Opuszczenie trybu następuje w wyniku wystąpienia przerwania lub wyjątku zerującego ten bit.

2. Algorytmy przełączania trybów pracy

Przedstawione algorytmy przełączania obejmują etapy dla najbardziej ogólnego przypadku, czyli dopuszczają możliwość stosowania stronicowania, wielozadaniowości, innych poziomów ochrony poza zerowym i podają warunki minimalne, które muszą być spełnione, aby była możliwa prawidłowa praca procesorów.

2.1. Przełączenie do trybu wirtualnego z ochroną

Podczas procesu inicjalizacji (zarówno po włączeniu zasilania jak i po podaniu sygnału zerującego RESET) w procesorach 80386 i i486 automatycznie ustawiany jest tryb adresowania rzeczywistego. Jedynym sposobem na opuszczenie go jest programowe przełączenie się do trybu z ochroną. Jednak przed wykonaniem przełączenia należy wykonać pewne operacje przygotowawcze.

(1) W celu umożliwienia dostępu do segmentów programów i danych musi być utworzona przynajmniej globalna tablica deskryptorów i dwa deskryptory: dla segmentu kodu i dla segmentu danych. Ponieważ segment stosu może być umieszczony w zwykłym, zapisywalnym segmencie danych, więc nie ma konieczności tworzenia dla niego oddzielnego deskryptora. Przed użyciem tablicy należy załadować rejestr GDTR, tak aby prawidłowo wskazywał jej początek i limit.

(2) Jeżeli rozwiązania sprzętowe pozwalają na generowanie przerwania NMI (*ang. non–maskeable interrupt* – przerwania niemaskowalnego), musi zostać stworzona tablica deskryptorów przerwań z furtką dla procedury obsługi przerwania NMI. Przed użyciem tablicy IDT rejestr IDTR, opisujący ją, musi zostać zainicjowany odpowiednimi wartościami.

(3) Ustawienie bitu PE (*ang. protection enable*) rejestru sterującego CR0, oznaczające przełączenie do trybu z ochroną.

(4) Ponieważ podczas realizacji rozkazu ustawiania bitu PE w kolejce czekają już kolejne zdekodowane instrukcje trybu adresowania rzeczywistego (ze względu na stosowanie przez procesory mechanizmu potokowania – *ang. pipelining*), należy opróżnić kolejkę przez wykonanie rozkazu skoku JMP. Skok ten musi być skierowany pod adres, pod którym znajduje się dalsza część oprogramowania inicjującego.

Po wykonaniu przełączenia rejestry segmentowe zawierają zawartości odpowiednie do trybu rzeczywistego, więc oprogramowanie musi zadbać o odpowiednią inicjalizację w trybie z ochroną.

(5) Stworzenie struktur danych odpowiednich do typu zarządzania pamięcią, czyli tablica LDT, jeżeli jest wykorzystywana, czy deskryptory, odpowiednio do liczby wykorzystywanych segmentów.

(6) Stosowanie stronicowania jest opcjonalne i sygnalizowane ustawieniem bitu PG (*ang. paging*) rejestru CR0. Po inicjalizacji bit ten jest wyzerowany i przed jego ustawieniem muszą być stworzone przynajmniej dwie tablice: katalog stron i jedna tablica stron a rejestr PDBR załadowany odpowiednią wartością.

(7) Ustawianie bitu PG.

(8) Następnie należy wykonać instrukcję skoku lub program musi znajdować się na stronie, mającej identyczne adresy liniowe z fizycznymi.

(9) Jeżeli wielozadaniowość jest stosowana, należy zadbać o stworzenie segmentu stanu zadania TSS i deskryptora dla tego segmentu oraz załadowanie rejestru TR.

2.2. Powrót do trybu adresowania rzeczywistego

Procesor powraca do trybu adresowania rzeczywistego przez wyzerowanie bitu PE (*ang. protection enable*) rejestru sterującego CR0. Procedura dokonująca tej akcji musi przeprowadzić następujące operacje:

(1) Jeżeli stronicowanie jest stosowane, należy:
- przenieść sterowanie pod adres liniowy, który jest równy fizycznemu,
- wyzerować bit PG (*ang. paging*) rejestru CR0,
- przesłać zero do rejestru CR3, co spowoduje opróżnienie bufora stronicowania TLB.

(2) Przenieść sterowanie do segmentu, którego limit wynosi 64K, co spowoduje załadowanie rejestru CS wartościami odpowiednimi dla trybu adresowania rzeczywistego.

(3) Załadować rejestry segmentowe SS, DS, ES, FS i GS selektorami dla deskryptorów zawierających wartości odpowiednie dla trybu adresowania rzeczywistego:
- limit = 64K,
- ziarnistość bajtową (bit G=0),
- rozszerzalne w górę (bit E=0),
- zapisywalne (bit W=1),
- obecne (bit P=1),
- adres bazowy = jakakolwiek wartość.

Jeżeli takie załadowanie nie nastąpi, to wykonywanie będzie kontynuowane z deskryptorami załadowanymi w trybie z ochroną.

(4) Zablokować przerwania.

(5) Wyzerować bit PE rejestru CR0.

(6) Wykonać skok do programu wykonywanego w trybie adresowania rzeczywistego, za pomocą instrukcji JMP. Spowoduje to opróżnienie kolejki rozkazów i załadowanie odpowiedniej wartości do rejestru CS.

(7) Załadować rejestr IDTR wartościami odpowiednimi dla tablicy przerwań w trybie adresowania rzeczywistego.

(8) Odblokować przerwania.

(9) Załadować rejestry segmentów w sposób wymagany przez program trybu adresowania rzeczywistego.

LITERATURA

[1] i486 Mikroprocessor Programmer's Reference Manual. Intel Corporation Literature Sales, Santa Clara 1990.

[2] 386DX Mikroprocessor Programmer's Reference Manual. Intel Corporation Literature Sales, Mt. Prospect 1990.

[3] Turley J. L.: Advanced 80386 Programming Techniques. Osborne McGraw–Hill, Berkeley 1988.

[4] Goczyński R., Tuszyński M.: Mikroprocesory 80286, 80386 i i486. Komputerowa Oficyna Wydawnicza "HELP", Warszawa 1991.

[5] Bułhak L., Goczyński R., Tuszyński M.: DOS od środka. Komputerowa Oficyna Wydawnicza "HELP", Warszawa 1990.

Recenzent: Dr inż. Eugeniusz Wróbel

Wpłynęło do Redakcji 30 czerwca 1995

Abstract

After initialization processor begins execution in real–address mode. If protected mode is to be used the software changes the setting of a mode bit in the CR0 register. Format of this register shows figure Fig. 1. Protected mode differs from real–address mode mainly in address translation. In real–address mode memory management can consist only of segmentation and address is formed like in processor 8086. In protected mode microprocessor interprets entire of segment register as selector which referrs to segment descriptor. Formats of selectors and descriptors show figures Fig. 2 and Fig. 3. Fig. 4 shows in which way selector selects segment descriptor. Memory menagement can also support paging (completily independent of segmentation). This type of address translation makes mechanism of virtual memory available.

The only way to leave real–address mode is to switch to protected mode. The procesor enters protected mode when the PE (protection enable) bit in the CR0 register is set. But before that to allow protected mode software to access programmes and data, at least one descriptor table (GDT) and two descriptors must be created. Before GDT can be used, the base address and limit for the GDT must be loaded into GDTR register. If hardware allows NMI interrupts to be generated, the IDT and a gate for NMI interrupt handler need to be created. Before the IDT can be used a proper value must be loaded into the IDTR register. After that the PE bit can be set. Because the processor overlaps the interpretation of several instruction, it is necessary to discard the instruction which already have been read into the processor. A jump instruction immediatly after setting protected mode changes the flow of execution and has the effect of emptying queue. Software should also reload segment register and other system structures.

The processor re–enters real–address mode if software clears the PE bit in the CR0 register. But before that procedure which re–enters real–address mode should proceed through a few points. If paging is enabled there is necessary to transfer control to linear addresses which are equal physical addresses, clear the PG bit in the CR0 register and move a 0 into CR3 register (to flush the TLB). Next transfer control to a segment which has a limit of 64K loads the CS register with the segment limit it needs to have in real mode. Then,

after loading other segment registers with selectors containing the values which are appropriate for real mode, and disabling interrupts it is passible to clear the PE bit in the CR0 register. A jump instruction immediatily after switching back to the real—address mode flushes the instruction queue. Before enabling interrupts there is necessary to load the base address and limit for real mode interrupt table. The last action is loading segment registers as needed by the real mode software.