WT/NP/17.05

Jakub Pomykała 209897

Ocena:

Oddano:

Proste jądro systemu operacyjnego

Architektura Komputerów 2 – projekt
INF 2014/15

Prowadzący:
dr inż. Tadeusz Tomczak

Spis treści

1 Wprowadzenie

1.1 Plan projektu i osiągnięcia

Projekt polegał na napisaniu prostego jądra systemu operacyjnego, przejścia w tryb chroniony i przełączaniu zadań za pomocą przerwań wywoływanych po przez zegar systemowy. Kod źródłowy jądra został napisany w Turbo Assemblerze i uruchamiany jest w DOSBoxie 0.74. Początkowy plan zakładł napisanie jądra, bootloadera i uruchamianie jądra na komputerze PC z procesorem Intel Pentium z dyskietki. Niestety nie udało mi się skończyć pisać bootloadera, dlatego jądro uruchamiane jest w emulatorze DOSBox. Plan prac wyglądał następująco:

- przygotowanie środowiska pracy oraz narzędzi
- przełączenie procesora w tryb chroniony
- obsługa pamięci rozszerzonej
- obsługa przerwań i wyjątków
- przełączanie zadań przez przerwania czasomierza systemowego

1.2 Podstawowe pojęcia

1. **tryb rzeczywisty** - jest to tryb pracy mikroprocesorów z rodziny procesorów x86, w którym procesor pracuje jak Intel 8086. Tryb ten nie zapewnia ochorny pamięci prze użyciem jej przez inny proces oraz obsługi wielozadaniowości. Dostępna jest jedynie 1 megabajtowa przestrzeń adresowa

2. **tryb chroniony procesora** - tryb pracy procesora który umożliwia adresowanie pamieci przekraczającej 1 megabajt pamięci, sprzętowa ochrona pamięci, wsparcie w przełączeniu kontekstu procesora, stronnicowanie pamięci (32 bitowe procesory)

3. **deskryptor** - 64 bitowa struktura danych w której przechowywane są informacje na temat miejsca w pamięci danego segmentu, typu, rozmiaru, zasady dostępu do segmentu oraz pozostałe informacje przydatne przy dostępie do segmentu w trybie chronionym procesora.

4. **tablice deskryptorów** - w trybie chronionym posługujemy się tablicami deskryptorów, wyróżniamy trzy podstawowe struktury:

    - Global Descriptor Table (GDT) - globalna tablica, zawiera deskryptory, które mogą być wykorzystane przez dowolne zadanie w systemie. Przechowują pamięć ekranu oraz ogólnie dostępne segmenty kodu i danych
    - Local Descriptor Table (LDT) - lokalna tablica, zawiera deskryptory dostępne tylko dla konkretnego zadania
    - Interrupt Descriptor Table (IDT) - tablica deskryptorów przerwań, używana do poprawnego reagowania na przerwania oraz wyjątki

5. **rejestry segmentowe** - zawierają adresy bazowe tablic systemowych, służą do organizacji segmentacji w trybie chronionym

- Global Descriptor Table Registers (GDTR) - liniowy adres bazowy i rozmiar globalnej tablicy deskryptorów
- Interrupt Descriptor Table Registers (IDTR) - liniowy adres bazowy i rozmiar tablicy deskryptorów przerwań
- Local Descriptor Table Registers (LDTR) - selektor segmentu tablicy deskryptorów lokalnych
- Task Registers (TR) - rejestr stanu zadania, selektor stanu zadania

6. **selektor** - w trybie chronionym procesora selektory są umieszczone w rejestrach segmentowych. Format selektora prezentuje się następująco:

   - INDEX - indeksu w tablicy deskryptorów, bity numer 15 - 3
   - TI - wyróżnika tablicy, czy tablica jest globalna (0) czy lokalna (1), bit numer 2
   - RPL - poziomu uprzywilejowania, bity numer 1 - 0

7. **segmentacja pamięci w trybie chronionym** - każdy segment danych bądź stosu jest opisany parametrami:

   - lokalizacja w przestrzeni adresowej pamięci
   - zasady dostępu
   - 8 bajtowa struktura danych nazywana deskryptorem

   Tablice mogą zawierać od 8 bajtów do 64kB (8192 deskryptory)

   Odwołanie do odpowiedniego deskryptora wykonuje się za pomocą selektora zapisanego w jednym z 16 bitowych rejestrów segmentowych:

   - rejestr DS, ES, FS, GS - segement musi mieć zezwolenie tylko do odczytu
   - rejestr SS - musi mieć ustawione prawa zapisu oraz odczytu
   - rejestr CS - wymaga prawa kodu wykonywalnego

   FS oraz GS są dostępne tylko w trybie chronionym. W przypadku wpisania błędnego selektora do rejestru segmentowego otrzymamy błąd Ógólnego naruszenia ochrony".

8. **przerwanie** - jest to sygnał, który powoduje zmianę przepływu sterowania, niezależnie od aktualnie wykonywanego programu. W przypadku pojawienie się przerwania wstrzymywany jest aktualne wykonywane zadanie i następuje skok do procedury obsługi przerwania. Procedura ta wykonuje czynności związane z obsługą przerwania i na końcu wydaje instrukcję powrotu z przerwania, która powoduje powrót do programu realizowanego przed przerwaniem. Rozróżniamy kilka typów przerwań:

   - programowe - wywoływane przez programistę, instrukcją INT + kod przerwania, lub w przypadku operacji niedozwolonych, np. dzielenie przez zero
   - sprzętowe - generowane przez urządzenia zewnętrzne, np. obsługa klawiatury, czyli wciśniecie jakiegoś klawisza, może to też być drukarka, myszka, dysk twardy itp.
   - wyjątki - generowane przez zewnętrzne układy procesora

9. **kontroler przerwań** - układ obsługi przerwań w komputerach PC jest zbudowany z dwóch połączonych kaskadowo układów 8259A, dzięki temu możliwa jest obsługa 15 przerwań sprzętowych - wejście IRQ2 układu master jest połączone z wyjściem układu slave. Kontroler klawiatury znajduje się na linii IRQ1, a czasomierz systemowy na linii IRQ0

10. **czasomierz systemowy (lub zegar systemowy)** - jest to fizyczne urządzenie znajdujące się na płycie głównej komputera, odpowiedzialne za dostarczanie aktualnego czasu i daty do komputera. Odpowiada również za dostarczanie sygnałów synchronizujących działanie podzespołów komputera z dokładnością do tysięcznych części sekundy.

2 Praca jądra systemu w trybie chronionym

2.1 Środowisko pracy i narzędzia

Jądro systemu było testowane za pomocą programu DOSBox 0.74 na komputerze z systemem Windows 8.1 x64. Program DOSBox 0.74 jest pełnym emulatrem procesora Intel 80386 udostępnianym na licencji GNU GPL. Kod jądra był asemblowany za pomocą TASM.exe (Turbo Assembler) oraz linkowany za pomocą TLINK.exe (Turbo Linker).

2.2 Przełączanie procesora w tryb chroniony

Procesor na początku swojego działania znajduje się w trybie rzeczywistym, żeby przełączyć go w tryb chroniony musimy zdefiniować strukturę globalnej tablicy deskryptorów (GDT), następnie w rejestrze CR0 ustawić pierwszy bit (tzw. bit PE - Protection Enable) na 1. Można to zrobić za pomocą instrukcji SMSW lub MOV. Od tej pory nasz procesor pracuje w trybie chronionym. Żeby powrócić do trybu rzeczywistego wystarczy, że wyzerujemy bit PE w rejestrze CR0.

2.3 Obsługa przerwań i wyjątków za pomocą kontrolera przerwań

W wprowadzenia obsługi przerwań musimy:

- utworzyć tablicę deskryptorów przerwań IDTR
- umieścić w niej adresy procedur obsługi wyjątków, które będa w programie
- załadować adres tablicy IDT do rejestru IDTR za pomocą instrukcji LIDTR
- odpowiednie do potrzeb skonfigurowanie kontrolera przerwań

2.4 Oprogramowanie kontrolera przerwań

Zaprogramowanie pracy kontrolera przerwań polega na zamaskowaniu nieobsługiwanych programowo przerwań sprzętowych (np. myszka czy dysk twardy), zależy nam jedynie na obsłudze zegara systemowego. W pliku PODST.TXT makropolecenie które przyjmuje jako parametr maskę przerwań układu. Wartość 1 na danej pozycji oznacza zablokowanie przerwań na tej linii. Czasomierz systemowy, który posłuży nam do wywoływania przerwań systemowych znajduje się na linii IRQ0. W takim razie użyjemy maski FEh, która binarnie wynosi 1111 1110. Co oznacza że jedynymi przerwaniami sprzętowymi jakie będziemy otrzymywać będą przerwania z czasomierza systemowego.

2.5 Obsługa przerwania pochodzącego z czasomierza systemowego

W momencie poprawnej konfiguracji kontrolera przerwań ostatnim krokiem do obsługi przerwań jest odblokowanie ich otrzymywania za pomocą instrukcji STI. W tym momencie z każdym przerwaniem czasomierza program będzie przenosić się do linii 55 w pliku MAIN.ASM, gdzie następuje obsługa przerwania. Obsługa przerwania to przełączenie zadania na jedno z dwóch za pomocą intrukcji JMP, czyli skoku odległego. W momencie skoku do odpowiedniego zadania, wywoływane jest przerwanie programowe, (instrukcja INT) którego obsługa polega na wyświetleniu informacji o aktywnym zadaniu. Następnie wykonywane jest makro OPOZNIENIE przy pomocy dwóch zagnieżdzonych pętli i wykorzystniu makra z pliku PODST.TXT (linie 312 - 326). Sygnał zakończenia przerwania, czyli informacja dla kontrolera przerwań o zakończeniu obsługi przerwania po przez zapis wartości 20H na port 20H. Skok na początek aktualnie wykonywanego zadania. W momencie przyjscia kolejnego przerwania jądro znów znajdzie się na linii 56 i całą procedura rozpocznie się od nowa.

2.6 Przełączenie zadań z wykorzystaniem przerwań czasomierza systemowego

Podczas przełączania zadania procesor zapamiętuje kontekst bieżącego zadania w jego segmencie stanu zadania, a następnie odczytuje z TSS kontekst nowego zadania, zawierający selektor segmentu i offset, od którego należy rozpocząć jego realizację. Następnie wykonywany jest rozkaz skoku odległego

3 Zakończenie

4 Bibiliografia

Literatura

[1] W. Stanisławski, D.Raczyński *Programowanie systemowe mikroprocesorów rodziny x86*, PWN, Warszawa, 2010.

[2] G.Syck, *Turbo Assembler - Biblia użytkownika*, LT&P, Warszawa, 1994.