

Bootloader

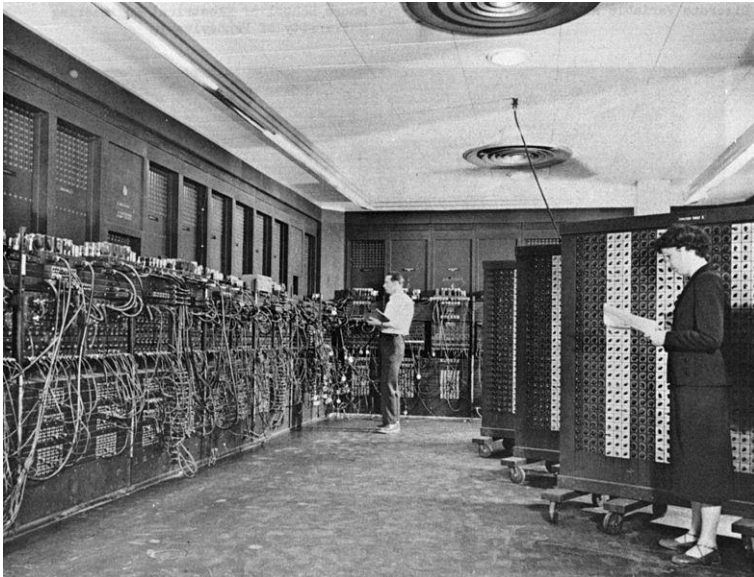
część pierwsza
Dominik Kobyra

Plan prezentacji

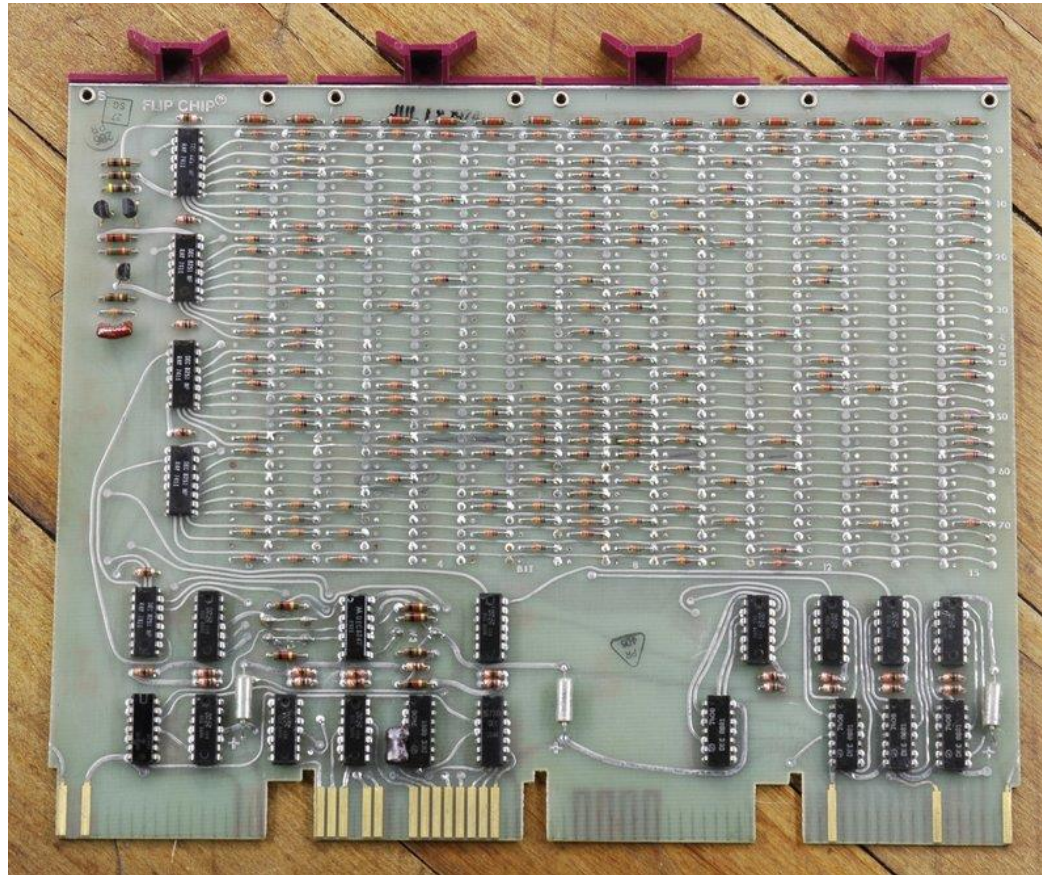
- wstęp teoretyczny i historyczny
- sposoby projektowania bootloaderów
- sekwencja rozruchu komputera
- tryby pracy procesora
 - rzeczywisty
 - chroniony
- przerwania BIOSu

Wstęp teoretyczny i historyczny

Wstęp teoretyczny i historyczny



Wstęp teoretyczny i historyczny



Wstęp teoretyczny i historyczny

bootloader – mały program napisany w celu załadowania bardziej skomplikowanego programu

Wstęp teoretyczny i historyczny

bootloader – mały program napisany w celu załadowania bardziej skomplikowanego programu

Główne zadania bootloadera:

- załadowanie jądra systemu
- dostarczenie jądra niezbędnych mu informacji
- konfiguracja środowiska
- przekazanie kontroli do systemu operacyjnego

Wstęp teoretyczny i historyczny

1. Ładowanie jądra

Wstęp teoretyczny i historyczny

1. Ładowanie jądra
2. Dostarczenie kernelowi niezbędnych informacji

Wstęp teoretyczny i historyczny

1. Ładowanie jądra
2. Dostarczenie kernelowi niezbędnych informacji
3. Konfiguracja środowiska

Wstęp teoretyczny i historyczny

1. Ładowanie jądra
2. Dostarczenie kernelowi niezbędnych informacji
3. Konfiguracja środowiska:
 - odblokowanie A20

Wstęp teoretyczny i historyczny

1. Ładowanie jądra
2. Dostarczenie kernelowi niezbędnych informacji
3. Konfiguracja środowiska:
 - odblokowanie A20

segment:offset

lokacja w pamięci = (segment • 16) + offset

Wstęp teoretyczny i historyczny

1. Ładowanie jądra
2. Dostarczenie kernelowi niezbędnych informacji
3. Konfiguracja środowiska:
 - odblokowanie A20

0007:7B90	0008:7B80	0009:7B70	000A:7B60	000B:7B50	000C:7B40
0047:7790	0048:7780	0049:7770	004A:7760	004B:7750	004C:7740
0077:7490	0078:7480	0079:7470	007A:7460	007B:7450	007C:7440
01FF:5C10	0200:5C00	0201:5BF0	0202:5BE0	0203:5BD0	0204:5BC0
07BB:0050	07BC:0040	07BD:0030	07BE:0020	07BF:0010	07C0:0000



0x7C00

Wstęp teoretyczny i historyczny

1. Ładowanie jądra
2. Dostarczenie kernelowi niezbędnych informacji
3. Konfiguracja środowiska:
 - odblokowanie A20

Ostatni adres: 0xFFFF

Wstęp teoretyczny i historyczny

1. Ładowanie jądra
2. Dostarczenie kernelowi niezbędnych informacji
3. Konfiguracja środowiska:
 - odblokowanie A20

Ostatni adres: 0xFFFFF

- 0xFFFF:0x000F
- 0xFFFF0:0x00FF
- 0xFF00:0x0FFF

Wstęp teoretyczny i historyczny

1. Ładowanie jądra
2. Dostarczenie kernelowi niezbędnych informacji
3. Konfiguracja środowiska:
 - odblokowanie A20
 - załadowanie GDT

Wstęp teoretyczny i historyczny

1. Ładowanie jądra
2. Dostarczenie kernelowi niezbędnych informacji
3. Konfiguracja środowiska:
 - odblokowanie A20
 - załadowanie GDT
 - przełączenie z trybu rzeczywistego na chroniony

Wstęp teoretyczny i historyczny

1. Ładowanie jądra
2. Dostarczenie kernelowi niezbędnych informacji
3. Konfiguracja środowiska:
 - odblokowanie A20
 - załadowanie GDT
 - przełączenie z trybu rzeczywistego na chroniony
4. Przekazanie kontroli do systemu operacyjnego

Sposoby projektowania bootloaderów

Sposoby projektowania bootloaderów

- single stage

Sposoby projektowania bootloaderów

- single stage
- two-stage

Sposoby projektowania bootloaderów

- single stage
- two-stage
- mixed

Sekwencja rozruchu komputera

Sekwencja rozruchu komputera

1. Wciśnięcie przycisku zasilania
2. Uruchomienie procedury POST
3. Wyszukanie nośnika rozruchowego
4. Załadowanie systemu operacyjnego
5. Przekazanie kontroli do systemu

Sekwencja rozruchu komputera

1. Wciśnięcie przycisku zasilania
2. Uruchomienie procedury POST
3. Wyszukanie nośnika rozruchowego
4. Załadowanie systemu operacyjnego
5. Przekazanie kontroli do systemu



- weryfikacja rejestrów procesora
- weryfikacja poprawności kodu BIOSu
- inicjalizacja BIOSu
- wykrycie, inicjalizacja i skatalogowanie wszystkich magistrali systemowych i urządzeń
- dostarczenie użytkownikowi interfejsu w celu konfiguracji systemu

Sekwencja rozruchu komputera

1. Wciśnięcie przycisku zasilania
2. Uruchomienie procedury POST
3. Wyszukanie nośnika rozruchowego
4. Załadowanie systemu operacyjnego
5. Przekazanie kontroli do systemu

- weryfikacja rejestrów procesora
- weryfikacja poprawności kodu BIOSu
- inicjalizacja BIOSu
- wykrycie, inicjalizacja i skatalogowanie wszystkich magistrali systemowych i urządzeń
- dostarczenie użytkownikowi interfejsu w celu konfiguracji systemu

0x55, 0xAA

Sekwencja rozruchu komputera

1. Wciśnięcie przycisku zasilania
2. Uruchomienie procedury POST
3. Wyszukanie nośnika rozruchowego
4. Załadowanie systemu operacyjnego
5. Przekazanie kontroli do systemu

- weryfikacja rejestrów procesora
- weryfikacja poprawności kodu BIOSu
- inicjalizacja BIOSu
- wykrycie, inicjalizacja i skatalogowanie wszystkich magistrali systemowych i urządzeń
- dostarczenie użytkownikowi interfejsu w celu konfiguracji systemu

0x55, 0xAA

0x0000:0x7C00

Tryb rzeczywisty procesora

Tryb rzeczywisty procesora

Tryb rzeczywistego adresowania – adresy odwołują się do rzeczywistych lokacji w pamięci

20 bitowa przestrzeń adresowa

Tryb rzeczywisty procesora

Zalety i ograniczenia

Tryb rzeczywisty procesora

Rejestry 32-bitowe są dostępne

0x66 - Operand Size Override Prefix - zmiana rozmiaru danych oczekiwanych przez tryb domyślny instrukcji, 16 bit na 32 bit i vice versa

Tryb rzeczywisty procesora

Dostępne 16 bitowe rejestry segmentowe: CS, DS, ES, FS, GS, SS

Tryb rzeczywisty procesora

Dostępne 16 bitowe rejestry segmentowe: CS, DS, ES, FS, GS, SS

Stos – przechowuje słowa 16 bitowe; rejestry SS i SP; wykorzystywany zawsze, gdy program używa instrukcji PUSH, POP, CALL, INT lub RET, a także gdy BIOS obsługuje przerwanie sprzętowe

Tryb rzeczywisty procesora

Dostępne 16 bitowe rejestry segmentowe: CS, DS, ES, FS, GS, SS

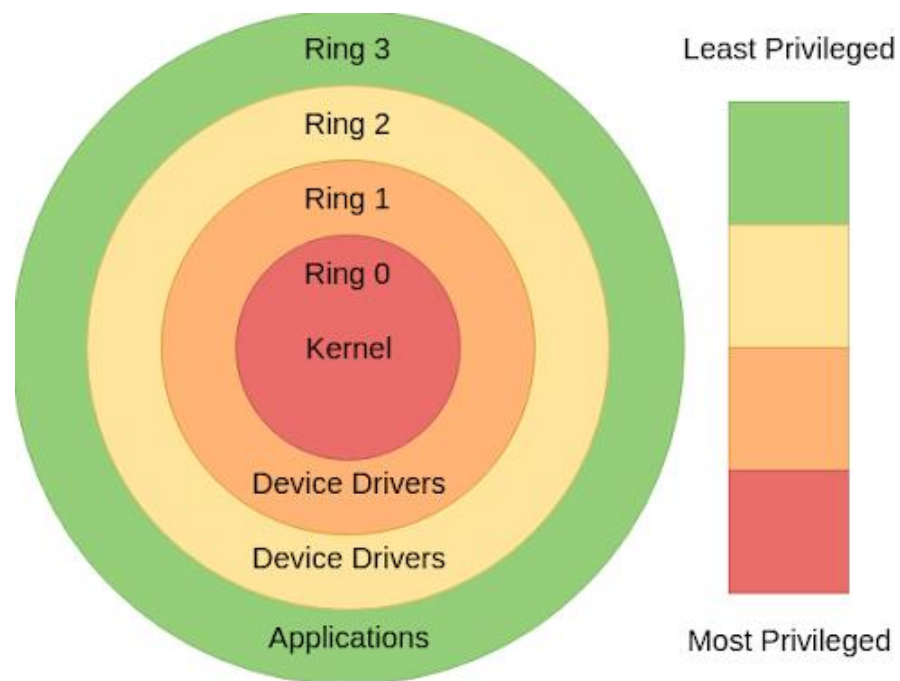
Stos – przechowuje słowa 16 bitowe; rejestry SS i SP; wykorzystywany zawsze, gdy program używa instrukcji PUSH, POP, CALL, INT lub RET, a także gdy BIOS obsługuje przerwanie sprzętowe

High Memory Area – obszar zajmujący prawie 64 kB, lokacja w pamięci od 0x100000 do 0x10FFEF; dostęp poprzez:

- ustawienie rejestru segmentowego na 0xFFFF
- użycie tego rejestru jako bazy przy obliczaniu adresu z offsetem z zakresu 0x10 - 0xFFFF

Tryb chroniony procesora

Tryb chroniony procesora



Tryb chroniony procesora

1982 – pojawienie się w architekturze x86 trybu chronionego, procesor 80286

Tryb chroniony procesora

1982 – pojawienie się w architekturze x86 trybu chronionego, procesor 80286

1985 – wydanie procesora 80386, udoskonalenie trybu chronionego

Tryb chroniony procesora

- stronicowanie

Tryb chroniony procesora

- stronicowanie
- 32 bitowa przestrzeń adresowa

Tryb chroniony procesora

- stronicowanie
- 32 bitowa przestrzeń adresowa
- 32 bitowe segmenty przesunięć

Tryb chroniony procesora

- stronicowanie
- 32 bitowa przestrzeń adresowa
- 32 bitowe segmenty przesunięć
- przełączanie w tryb rzeczywisty

Tryb chroniony procesora

- stronicowanie
- 32 bitowa przestrzeń adresowa
- 32 bitowe segmenty przesunięć
- przełączanie w tryb rzeczywisty
- wirtualny tryb 8086

Tryb chroniony procesora

Aktywacja trybu chronionego

Tryb chroniony procesora

Aktywacja trybu chronionego

```
cli
lgdt [gdtr]
mov eax, cr0
or al, 1
mov cr0, eax

jmp 08h:PModeMain

PModeMain:
; jesteśmy w trybie chronionym
```

Przerwania BIOSu

Przerwania BIOSu

INT 0x13, AH = 0 - reset dysku twardego lub dyskietki

Przerwania BIOSu

RBIL – Ralf Brown's Interrupt List

Przerwania BIOSu

INT 0x10, AH = 0x1 - ustawienie kursora

INT 0x10, AH = 0xE - wyświetlenie znaku

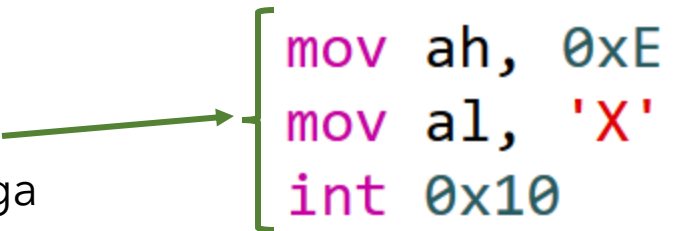
INT 0x10, AH = 0x13 - wyświetlenie stringa

Przerwania BIOSu

INT 0x10, AH = 0x1 - ustawienie kursora

INT 0x10, AH = 0xE - wyświetlenie znaku

INT 0x10, AH = 0x13 - wyświetlenie stringa



```
mov ah, 0xE  
mov al, 'X'  
int 0x10
```

Dziękuję za uwagę