**загрузчик**

**Текстпрограммы**

2017

#### АННОТАЦИЯ

Данный документ является руководством системного программиста (системного администратора) «Загрузчик».

В документе приводится необходимая информация по установке и настройке загрузчика, а также сведения по его администрированию.

1. Текст программы

Листинг 1: Файл boot.asm

|  |  |
| --- | --- |
| ;Пример реализации вывода сообщения из MBR  use16  org 0x7C00  ; Clean registr  xor ax, ax  mov es, ax  mov ds, ax  mov ss, ax  mov sp, 0x1000  ; Create video rejim for clean monitor  mov ax, 3  int 10h  ; Hello on the monitor  mov si, mHello  call print  ; Write just  die: jmp short die  mHello db 'Hello, world - i was booted!',10,13,0  ; subprogram present  print:  cld  pusha  .PrintChar:  lodsb  test al, al  jz short .Exit  mov bl, 7  int 10h  jmp short .PrintChar  .Exit:  popa  ret  BITS 16  org0x700  ; Обнулим регистры, установим стек  cli  mov ax, 0  mov ds, ax  mov es, ax  mov ss, ax  mov sp, 0x700  sti  ;Сообщение о приветствии  mov si, msg\_start  call kputs  ; Сообщение о переходе в защищенный режим  mov si, msg\_entering\_pmode  call ps  ; Отключение курсора (просто так)  mov ah, 1  mov ch, 0x20  int 0x10  ; Установим базовый вектор контроллера прерываний в 0x20  mov al,00010001b  out 0x20,al  mov al,0x20  out 0x21,al  mov al,00000100b  out 0x21,al  mov al,00000001b  out 0x21,al  ; Отключим прерывания  cli  ; Загрузка регистра GDTR:  lgdt [gd\_reg]  ; Включение A20:  in al, 0x92  or al, 2  out 0x92, al  ; Установка бита PE регистра CR0  mov eax, cr0  or al, 1  mov cr0, eax  ; С помощью длинного прыжка мы загружаем селектор нужного сегмента в регистр CS  jmp 0x8: \_protect  ps:  pusha  .loop:  lodsb  test al, al  jz .quit  mov ah, 0x0e  int 0x10  jmp short .loop  .quit:  popa  ret  ; Следующий код — 32-битный  [BITS 32]  ;При переходе в защищенный режим, сюда будет отдано управление  \_protect:  ; Загрузим регистры DS и SS селектором сегмента данных  mov ax, 0x10  mov ds, ax  mov es, ax  mov ss, ax  ; Наше ядро слинковано по адресу 2мб, переносим его туда. ker\_bin — метка, после которой вставлено ядро  mov esi, ker\_bin  ; Адрес, по которому копируем  mov edi, 0x200000  ; Размер ядра в двойных словах (65536 байт)  mov ecx, 0x4000  rep movsd  ; Ядро скопировано, передаем управление ему  jmp 0x200000  gdt:  dw 0, 0, 0, 0  ; Нулевой дескриптор  db 0xFF  ; Сегмент кода с DPL=0 Базой=0 и Лимитом=4 Гб  db 0xFF  db 0x00  db 0x00  db 0x00  db 10011010b  db 0xCF  db 0x00  db 0xFF  ; Сегмент данных с DPL=0 Базой=0 и Лимитом=4Гб  db 0xFF  db 0x00  db 0x00  db 0x00  db 10010010b  db 0xCF  db 0x00  ; Значение, которое мы загрузим в GDTR:  gd\_reg:  dw 8192  dd gdt  msg\_start: db «Get fun! New loader is on», 0x0A, 0x0D, 0  msg\_epm: db «Protected mode is greeting you», 0x0A, 0x0D, 0  xor eax, eax  ret  org 7C00h ;адреса нашей программы расчитываются с учетом данной директивы  use16 ;генерируется шестнадцатиричный код  cli ;запрещаем прерывания для смены адресов в сегментных регистрах  mov ax, 0  mov es, ax  mov ds, ax  mov ss, ax  mov sp, 7C00h  sti ;разрешаем прерывания (после изменения адресов)  mov ax, 0003h ;установка видео режима для вывода строки на экран  int 10h  mov ax, 1301h ;собственно вывод строки функция 13h int 10h  mov bp, stroka ;адрес выводимой строки  mov dx, 0000h ;строка и колонка в которой выводится текст  mov cx, 21 ;количество символов выводимой строки  mov bx, 000eh ;00-номер видео страницы (лучше не трогать) 0eh-атрибуты cимвола(цвет,фон)  int 10h  ;------------наша супер функция ah 42h int 13h-----------  mov ah, 42h ;функция копирования - главное действующее лицо этого урока  mov si, lba ;смещение структуры LBA  mov dl, 80h ;для флешки это значение должно быть 80h, впрочем, как и для жесткого диска  int 13h  jc error ;если ошибка чтения сектора, то выдаем соответствующее сообщение  jmp 0000:7E00h ;т. к. ядро мы грузили по адресу 0000:7E00h, то и управление ядру мы  ;передаем прыгая на этот адрес  error:  mov ax, 1301h ;уже знакомая нам функция вывода строки  mov bp, err0 ;  mov dx, 0100h ;  mov cx, 6 ;  mov bx, 000eh ;  int 10h ;  jmp $ ;топтаться на месте  err0 db 'Error!'  stroka db 'Ok, MBR loaded again!'  lba:  ; +----------------------------------------------------------------------------------  ; | Смещение Тип Назначение  ; |  ; | 00h byte размер структуры  ; | 01h byte зарезервировано  ; | 02h word сколько секторов читать  ; | 04h dword адрес буфера в формате сегмент(word):смещение(word)  ; | 08h qword стартовый номер сектора для чтения  ; | 10h qword хвост 64 битного адреса (используется при 32х битном FFFF:FFFF)  ; +----------------------------------------------------------------------------------  db 10h  db 00h  dw 0001h ;копируем 1 сектор - больше нам пока не надо  dd 00007e00h ;куда грузим первый сектор грузим как у нас на рисунках сразу ;после нулевого если 7С00h + 200h(512 байт) = 7E00h  dq 0000000000000001h ;начинаем копировать с первого сектора у нас там ядро  dq 0000000000000000h  times 510 - ($ - $$) db 0 ;заполнение нулями промежутка между предыдущим байтом и пос-  db 0x55,0xAA ;ледними двумя байтами |  |