### ТЕХНИЧЕСКОЕ ЗАДАНИЕ

### Разработать загружаемый драйвер жесткого диска

Содержание

|  |  |
| --- | --- |
|  | #Cтр. |
| Техническое задание  | 1 |
| Содержание | 2 |
| Введение | 3 |
| *Основная часть* |  |
| Структура загружаемого драйвера | 3 |
| Связь драйвера с операционной системой | 6 |
| Инициализация драйвера | 8 |
|  |  |
| Разметка диска  | 9 |
| Контроллер жестких дисков для АТ-подобных ПЭВМ | 11 |
| Описание портов ввода-вывода | 11 |
| Система команд | 12 |
| Получение блока параметров BIOS | 13 |
|  |  |
| Подключение драйвера | 15 |
|  |  |
| Заключение | 16 |
| Список литературы | 17 |
| *Приложения* |  |
| Листинг программы | 18 |

### Введение

Фирмы-разработчики аппаратного обеспечения постоянно совершенствуют внешние устройства и другие узлы персонального компьютера. Постоянно появляются новая периферийная аппаратура и новые модификации уже существующих устройств. Старые устройства наделяются новыми возможностями, новые делают такое, о чем раньше не приходилось и мечтать.

Интуитивно ясно, что должна существовать какая-то программная прослойка между аппаратным и программным обеспечением, выполняющая "согласующие" и "унифицирующие" действия. Эта прослойка работает напрямую с аппаратурой, а прикладное (да и системное) программное обеспечение имеет дело только с этой интерфейсной прослойкой.

Операционная система MS-DOS, работающая на компьютерах фирмы IBM или совместимых с ними, тоже использует механизм драйверов.

Однако драйверы MS-DOS не всегда обращаются напрямую к аппаратуре. Обычно они вызывают функции BIOS, и уже BIOS выполняет все действия по вводу/выводу. Конечно, BIOS содержит программы обслуживания только стандартных устройств ввода/вывода, нестандартные устройства обслуживаются драйверами напрямую.

Использование BIOS как дополнительного интерфейса между драйверами стандартных устройств и аппаратурой резко повышает "живучесть" MS-DOS на не вполне совместимых с IBM персональных компьютерах. И это действительно так - самая распространенная на сегодняшний день операционная система MS-DOS версии 3.30 работает на всех компьютерах, хоть сколько-нибудь совместимых с IBM PC.

Это возможно благодаря тому, что производители совместимых компьютеров учитывают в программах BIOS все аппаратные особенности, и DOS "не видит" отличий. А прикладная программа - тем более.

Почему же этот способ не используется в операционных системах UNIX или OS/2? Дело в том, что к сожалению, программы BIOS не являются реентерабельными. Это не имеет значения для однозадачной MS-DOS, а мультизадачные операционные системы вынуждены сами организовывать обслуживание аппаратуры реентерабельным способом. (Существуют еще проблемы разделения ресурсов между параллельно выполняющимися процессами, которые тоже не решаются в рамках BIOS).

Таким образом, независимость аппаратного и программного обеспечения в DOS обеспечивается, с одной стороны, BIOS для стандартных устройств, с другой стороны - драйверами.

Пользователи могут легко дополнять операционную систему своими драйверами, составленными для нестандартных устройств. Возможна также замена стандартных драйверов, замена или расширение функций BIOS.

## Структура загружаемого драйвера

Иногда говорят, что драйверы - это разновидность COM-программ, но это не так. Скорее способ получения загрузочного модуля драйвера похож на способ получения программы в формате COM. Есть еще одно сходство драйверов и программ в формате COM (которое как раз и появляется из-за одинакового способа их получения) - загрузочные модули этих программ являются точным отображением исходного текста на языке ассемблера без добавления каких-либо управляющих блоков в начало файла, как это происходит в программах формата EXE

Но, оказывается, управляющий блок в самом начале модуля драйвера имеется. Это так называемый заголовок драйвера. Только в отличие от программ формата EXE, этот заголовок создается не редактором связи, а самим программистом и должен быть помещен в самое начало исходного текста программы-драйвера.

При загрузке драйвера в память заголовок драйвера тоже помещается в оперативную память, и в нем операционная система производит некоторые изменения, о которых мы еще будем говорить.

Таким образом, можно говорить и о сходстве драйвера с программами в формате EXE, так как в начале загрузочного модуля драйвера имеется управляющий блок. Только этот управляющий блок в отличие от заголовка EXE-файла является принадлежностью самой программы и загружается вместе с ней в память. Заголовок EXE-программы используется при загрузке EXE-программы, но после загрузки операционная система убирает его из памяти.

Не стоит пытаться запускать драйвер как программу в формате COM, так как управление будет передано в область памяти, содержащую заголовок драйвера, а там нет правильных машинных команд. Поэтому обычно файлы драйверов имеют расширения имени, отличные от COM или EXE. Чаще всего используются расширения SYS, DRV, иногда BIN. На самом деле расширение имени можно задавать любое, так как при описании драйвера в файле CONFIG.SYS указывается его полное имя.

Для драйвера никогда не создается префикс программного сегмента PSP. В начале исходного текста программы-драйвера не ставится директива ORG 100H, как это делается для COM-программы, так как не надо резервировать место для PSP.

Что же представляет из себя загрузочный модуль драйвера?

Как уже было сказано, в начале модуля находится заголовок драйвера. Мы уже немного говорили о нем при описании векторной таблицы связи операционной системы. Приведем формат заголовка:

|  |  |  |
| --- | --- | --- |
| (0) 4 | next  | указатель на заголовок следующего драйвера. Если смещение адреса следующего драйвера равно FFFF, это последний драйвер в цепочке  |
| (+4) 2 | attrib  | атрибуты драйвера |
| (+6) 2 | strateg  | смещение программы стратегии драйвера  |
| (+8) 2 | interrupt  | смещение программы обработки прерывания для драйвера  |
| (+10) 8 | dev\_name  | имя устройства для символьных устройств или количество обслуживаемых устройств для блочных устройств.  |

Как уже было сказано, все драйверы связаны в цепочку. Самый первый драйвер находится сразу за векторной таблицей связи. Поле next заголовка драйвера указывает на следующий драйвер (на его заголовок). Это поле имеет формат DWORD-указателя и состоит из компоненты адреса сегмента и смещения. Признаком того, что данный драйвер последний в цепочке, служит значение 0FFFFh в компоненте смещения поля next.

Программист, когда он составляет программу-драйвер, заносит в это поле либо 0FFFFh:0FFFFh, если исходный текст содержит только один драйвер, либо адрес следующего драйвера (в виде дальней ссылки на метку заголовка следующего драйвера). Если исходный текст содержит несколько драйверов, то в заголовке последнего в поле next должно находиться значение 0FFFFh:0FFFFh.

При загрузке драйверов в память операционная система изменит содержимое поля next в заголовках драйверов для того, чтобы это поле указывало на заголовок следующего драйвера в цепочке. (Изменит в памяти, а не в файле драйвера!)

Обычно исходный текст программы содержит один драйвер, и поле next задается следующим образом:

next DD 0FFFFFFFFh

Следующее поле в заголовке драйвера - поле атрибутов драйвера atrib.

Это поле описывает устройство, обслуживаемое данным драйвером. Каждый бит слова отвечает за ту или иную особенность устройства. Прежде чем мы детально рассмотрим назначение всех битов этого слова, заметим, что бит 15 (самый старший бит) указывает, является ли это устройство символьным или блочным.

Для драйверов блочных устройств формат слова атрибутов:

|  |  |
| --- | --- |
| Бит | Назначение |
| 0 | Зарезервировано, бит должен быть равен 0  |
| 1 | 1 - драйвер поддерживает 32-битовую адресацию сектора (для версий DOS, начиная с 4.00 и более поздних); если установлен этот бит, поле номера сектора всех запросов является двойным словом, добавляемым в конец заголовка запроса, старое поле номера сектора должно содержать -1); 0 - используется 16-битовая адресация сектора  |
| 2-5 | Эти биты зарезервированы и должны быть равны 0  |
| 6 | 1 - поддерживаются логические устройства (используется блочными драйверами для управления "виртуальными" флоппи-дисками, создаваемые драйвером DRIVER.SYS в DOS версии 3.2 и более поздних версиях); 0 - логические устройства для блочных драйверов не поддерживаются;  |
| 7-10 | Эти биты зарезервированы и должны быть равны 0  |
| 11 | 1 - единица в этом бите означает, что драйвер поддерживает функцию проверки замены носителя данных в устройстве (например, замены дискеты); используется для DOS версий 3.00 и более поздних; 0 - для блочных устройств функция проверки замены носителя данных не поддерживается  |
| 12 | Зарезервировано, бит должен быть равен 0  |
| 13 | 1 - драйвер не использует стандартное IBM-устройство, и необходимо выдать запрос на построение блока параметров BIOSBIOS BPB; 0 - используется IBM-устройство  |
| 14 | 1 - поддерживаются функции IOCTL; 0 - функции IOCTL не поддерживаются  |
| 15 | 1 - символьное устройство; 0 - блочное устройство  |

После слова атрибутов драйвера находятся два очень важных поля: смещение программы стратегии драйвера strateg и смещение программы обработки прерывания interrupt.

Эти две программы используются DOS для организации обращения к драйверу. Для обращения к драйверу DOS формирует в своей области данных запрос, состоящий из заголовка стандартного формата и переменной части запроса, длина и формат которой зависят от типа запроса. После этого DOS считывает из заголовка драйвера значение смещения программы стратегии и передает ей управление, записав в регистры ES:BX адрес заголовка запроса.

Задача программы стратегии - запомнить этот адрес внутри тела драйвера для дальнейшего использования или организовать очередь запросов обслуживания.

Сразу после вызова программы стратегии DOS вызывает программу обработки прерываний, определив ее адрес из поля interrupt заголовка драйвера.

Программа обработки прерывания извлекает только что записанный программой стратегии адрес заголовка запроса и выполняет ту функцию, номер которой записан в запросе. Номер функции находится в заголовке запроса.

Результаты выполнения функции программа прерывания записывает в специально отведенные поля заголовка запроса, и на этом процедура обращения DOS к драйверу завершается.

Формат заголовка запроса будет приведен ниже, а сейчас покажем, как в заголовке драйвера задаются смещения программ стратегии и прерывания:

strateg DW strateg\_proc

interrupt DW interrupt\_proc

Последнее поле заголовка драйвера dev\_name имеет различную интерпретацию для символьных и блочных устройств.

Для символьных устройств в этом поле должно располагаться выровненное по левому краю и дополненное до восьми символов пробелами имя устройства. Это имя будет использоваться для обращения к драйверу. Если Вы собираетесь заменить драйвер стандартного символьного устройства DOS на свой, Вы должны записать имя устройства заглавными буквами:

dev\_name DB 'AUX '

Для блочных устройств первый байт поля dev\_name содержит количество устройств, обслуживаемых данным драйвером, остальные семь байтов не используются:

dev\_name DB 1

 DB 7 dup(?)

Таким образом, мы выяснили, что драйвер содержит в самом начале заголовок, и где-то дальше должны располагаться программы стратегии и прерывания. (Не следует путать программу прерывания драйвера с программой обслуживания аппаратных или программных прерываний. Хотя программа прерывания драйвера немного похожа на обработчик программных прерываний, назначение этой программы и механизм ее использования совершенно другой).

Что еще может находиться в программе-драйвере?

Это могут быть области данных, используемые драйвером, и подпрограммы, вызываемые программами стратегии и прерывания. Иногда стандартные драйверы переназначают на себя некоторые вектора прерываний, и тогда они содержат в себе обработчики этих прерываний. В области памяти, отведенной операционной системой драйверу, может располагаться стек драйвера, если размер системного стека недостаточен.

На длину драйвера накладывается такое же ограничение, как и на длину COM-программ - 64 килобайта, то есть один сегмент.

## Связь драйвера с операционной системой

Рассмотрим теперь более подробно механизм взаимодействия драйвера и операционной системы.

После загрузки драйвер становится как бы частью операционной системы. Все обращения к драйверу DOS выполняет с использованием заголовка драйвера. Для примера приведем вид заголовка символьного драйвера, выполняющего только простейшие функции:

next DD 0FFFFFFFFh

attrib DW 8000h

strateg DW strateg\_proc

interrupt DW interrupt\_proc

dev\_name DB 'TESTDRV '

Это символьный драйвер (старший бит поля attrib равен 1), исходный текст содержит только один драйвер (поле next содержит значение 0FFFFFFFFh), имя устройства, которое нужно будет использовать при обращении к драйверу - TESTDRV. Имя устройства не должно совпадать с именем файла, содержащего символьный драйвер, иначе Вы не сможете обратиться к файлу, например, для его переименования - DOS будет работать не с файлом, а с устройством.

Как уже было сказано, перед обращением к драйверу DOS подготавливает заголовок запроса в своей области данных и вызывает программу стратегии, извлекая ее смещение из заголовка драйвера. Программа стратегии обычно очень проста, так как ее задача - запомнить адрес заголовка запроса в области памяти драйвера. Область для хранения адреса заголовка запроса может быть определена следующим образом:

req\_off DW ?

req\_seg DW ?

Тогда программа стратегии должна записать содержимое регистра ES в поле req\_seg, а регистра BX - в поле req\_off:

strateg\_proc: mov cs:req\_off,bx

 mov cs:req\_seg,es

 ret

Драйвер состоит из одного сегмента кодов, поэтому для адресации данных используется сегментный регистр CS.

Запрос операционной системы к драйверу соcтоит из заголовка, имеющего фиксированный формат и длину 13 байт, и переменной части, размер и формат которой зависит от выполняемой функции.

Приведем формат заголовка запроса:

|  |  |  |
| --- | --- | --- |
| (0) 1 | size | Длина запроса в байтах (длина заголовка запроса плюс длина переменной части запроса)  |
| (+1) 1 | unit | Номер устройства (используется для блочных устройств, указывает, с каким именно устройством, обслуживаемым драйвером, будет работать операционная система)  |
| (+2) 1 | cmd | Код команды, которую требуется выполнить (может иметь значение от 0 до 18h)  |
| (+3) 2 | status | Слово состояния устройства, заполняется драйвером перед возвратом управления операционной системе  |
| (+5) 8 | reserved | Зарезервировано  |

После вызова программы стратегии DOS передает управление программе прерывания (без параметров). Задача программы прерывания - выполнить команду, код которой находится в поле cmd заголовка запроса. Если драйвер блочного устройства обслуживает несколько логических устройств, то в поле unit находится номер устройства, для которого необходимо выполнить команду.

В зависимости от выполняемой команды запрос может содержать другую информацию, необходимую для выполнения команды.

**Как результаты выполнения команды возвращаются DOS?**

Данные (или адреса данных), полученные драйвером от физического устройства ввода/вывода, помещаются в область переменной части запроса. Кроме того, драйвер должен установить слово соcтояния устройства status в заголовке запроса в соответствии с результатами выполнения команды.

Приведем формат слова состояния устройства:

|  |  |
| --- | --- |
| Бит | Назначение |
| 0-7 | Код ошибки устройства (если команда выполнена с ошибкой и драйвер установил признак ошибки (бит 15) в единицу, в это поле он должен записать код ошибки).  |
| 8 | Команда выполнена. Этот бит всегда устанавливается драйвером перед тем, как он возвращает управление операционной системе.  |
| 9 | Занято. Этот бит устанавливается обработчиком команды, когда физическое устройство занято выполнением предыдущей операции и поэтому не может выполнить требуемую команду. Этот бит используется также для передачи такой информации, как "буфер клавиатуры не пуст", "среда носителя данных заменяемая" (в команде проверки возможности замены среды носителя данных).  |
| 10-14 | Зарезервировано. |
| 15 | Признак ошибки. Устанавливается драйвером, когда он не может обработать запрос или произошла физическая либо логическая ошибка при обработке правильного запроса. Биты 0-7 при этом должны содержать код ошибки.  |

Приведем таблицу возможных кодов ошибок:

|  |  |
| --- | --- |
| Код | Описание |
| 0 | Нарушение защиты от записи. Была предпринята попытка записи информации на защищенное от записи устройство.  |
| 1 | Неизвестное устройство. |
| 2 | Устройство не готово. |
| 3 | Неизвестная команда. Затребованная команда не поддерживается драйвером.  |
| 4 | Ошибка CRC. При выполнении команды обнаружена ошибка циклического кода проверки.  |
| 5 | Неправильная длина запроса. Поле длины в заголовке запроса содержит неверное значение.  |
| 6 | Ошибка при поиске дорожки (дорожка не найдена).  |
| 7 | Неизвестный носитель данных.  |
| 8 | Сектор не найден. |
| 9 | Нет бумаги в принтере. |
| 0Ah | Ошибка записи. |
| 0Bh | Ошибка чтения. |
| 0Ch | Общая ошибка. |
| 0Dh | Зарезервировано. |
| 0Eh | Зарезервировано. |
| 0Fh | Неразрешенная замена диска (только для DOS версии 3.0 и более поздних версий).  |

Общая схема действий программы прерывания драйвера такова:

* получив управление от операционной системы, программа прерывания сохраняет содержимое всех регистров процессора и считывает номер команды из заголовка запроса;
* при необходимости программа считывает дополнительную информацию из области запроса;
* затребованная команда выполняется (если она поддерживается драйвером);
* если драйвер считывает какие-либо данные от обслуживаемого физического устройства для передачи их DOS, то сами данные или их адреса программа прерывания записывает в область запроса;
* программа прерывания устанавливает слово состояния устройства в соответствии с результатами выполнения команды (если драйвер не поддерживает затребованную команду, в слове состояния устройства устанавливаются биты 15 и в биты 0-7 записывается код ошибки 3 - неизвестная команда);
* восстанавливается содержимое регистров процессора, и управление возвращается операционной системе с помощью команды возврата из дальней процедуры.

### Инициализация драйвера

Эта функция выполняется только один раз при загрузке драйвера и подключении его к операционной системе.

Функция инициализации должна поддерживаться любым драйвером, так как она сообщает операционной системе сведения, необходимые DOS для правильного подключения и использования драйвера.

Приведем формат запроса для команды инициализации:

|  |  |  |
| --- | --- | --- |
| (0) 13 | header | Заголовок запроса.  |
| (+13) 1 | n\_units | Количество устройств, обслуживаемых драйвером. Это поле заполняется только блочным драйвером.  |
| (+14) 4 | end\_addr | Конечный FAR-адрес резидентной части кода драйвера. В это поле драйвер записывает адрес байта памяти, следующего за той частью кода драйвера, которая должна стать резидентной.  |
| (+18) 4 | parm | FAR-адрес строки параметров инициализации драйвера из файла CONFIG.SYS. Эта строка содержит все, что находится в строке файла после команды 'DEVICE=', она заканчивается символами перевода строки и возврата каретки 0Ah, 0Dh. При возврате драйвер блочного устройства должен записать в это поле адрес массива указателей на блоки параметров BIOSBIOS (BPB), по одному указателю на каждое устройство, обслуживаемое драйвером.  |
| (+22) 1 | drive | Номер устройства. Для версии DOS 3.0 и более поздних версий в это поле при загрузке драйвера операционная система заносит номер, назначенный устройству, обслуживаемому драйвером. Например, для устройства А: это 0, для B: - 1 и т.д.  |

При инициализации драйвер символьного устройства сохраняет в своей внутренней области данных параметры инициализации, используя адрес parm. Если параметры содержат числовые величины, программа инициализации может произвести их перекодировку и сохранить значения в двоичном формате.

Затем драйвер может выполнить инициализацию обслуживаемого физического устройства ввода/вывода, инициализацию своих внутренних переменных, вывести на экран какие-либо сообщения либо даже запросить у оператора дополнительные данные - функция инициализации может пользоваться для организации диалога с оператором и других действий функциями прерывания 21h с номерами от 01h до 0Ch, 25h, 30h, 35h и функциями BIOS.

Кроме этого, драйвер должен заполнить поле end\_addr адресом конца резидентной части драйвера. Так как программа инициализации выполняется только один раз, обычно ее располагают в конце драйвера и для экономии памяти не оставляют резидентной.

Драйверы блочных устройств дополнительно должны возвратить DOS количество обслуживаемых устройств (в поле n\_units) и указатель на массив указателей на блоки BPB (в поле parm).

Количество устройств используется DOS для определения логических имен устройств. Например, если Ваш драйвер обслуживает три логических устройства, и на момент его загрузки в системе имеются устройства A:, B: и C:, то устройства, обслуживаемые Вашим драйвером, получат имена D:, E: и F:. Количество устройств необходимо указывать также и в заголовке драйвера, в первом байте поля имени устройства dev\_name.

Для каждого логического устройства драйвер должен содержать так называемый блок параметров BIOS (BIOS Parameter Block) BPB.

Блок BPB содержится в загрузочном секторе диска и содержит информацию, необходимую BIOS для работы с диском. Приведем формат BPB:

|  |  |  |
| --- | --- | --- |
| (0) 2 | sect\_siz | Количество байтов в одном секторе диска.  |
| (+2) 1 | clustsiz | Количество секторов в одном кластере.  |
| (+3) 2 | res\_sect | Количество зарезервированных секторов.  |
| (+5) 1 | fat\_cnt | Количество таблиц FAT.  |
| (+6) 2 | root\_siz | Максимальное количество дескрипторов файлов, содержащихся в корневом каталоге диска.  |
| (+8) 2 | tot\_sect | Общее количество секторов на носителе данных (в разделе DOS).  |
| (+10) 1 | media | Байт-описатель среды носителя данных.  |
| (+11) 2 | fat\_size | Количество секторов, занимаемых одной копией FAT.  |

Приведем фрагмент исходного текста драйвера, возвращающего при инициализации указатель на массив BPB:

 lea dx,bpb\_ptr

 mov es:[bx+18],dx

 mov es:[bx+20],cs

 . . . . . . . . . .

В этом примере предполагается, что ES:BX содержит адрес заголовка запроса.

**Разметка диска**

 Данные на жесткий диск записываются в секторах. Сектора располагаются на дорожках. Нумерация дорожек начинается с внешней стороны пластины (там расположена нулевая дорожка). Количество пластин (дисков) и головок, так же как и максимальное число дорожек, могут колебаться в довольно широких пределах и зависят от типа конкретного накопителя. Дорожка обычно содержит от 8 до 26 секторов и для данного конкретного накопителя число секторов на дорожке постоянно. Начало дорожки определяется сигналом "индекс", который генерируется накопителем при каждом обороте диска. Далее следует первый сектор дорожки. Второй сектор будет отстоять от первого на число секторов, равное значению фактора чередования минус 1, третий еще на столько же и т. д. Таким образом при факторе

 чередования равном 3, сектора на 17-секторной дорожке будут располагаться следующим образом:

 ЪДДВДДВДДВДДВДДВДДВДДВДДВДДВДДВДДВДДВДДВДДВДДВДДВДДї

 і 1і 7і13і2 і8 і14і3 і9 і15і4 і10і16і 5і11і17і 6і12і

 ГДДБДДБДДБВДБДДБДДБВДБДДБДДБДДБДДБДДБДДБДДБДДБДДБДДґ

 і і фактор і і

 і ічер-ния і і

 і<--индексі<------>і индекс --> і

 **Формат сектора приведен в таблице.**

 ЪДДДДДДДДДДДВДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДВДДДДДДї

 іОбозначениеі Назначение іДлина і

 і і і(байт)і

 ГДДДДДДДДДДДЕДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДЕДДДДДДґ

 і АМ і Адресный маркер і 4 і

 і ИНТ1 і Интервал і 9-12 і

 ГДДДДДДДДДДДЕДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДЕДДДДДДґ

 і і Поле идентификации: і і

 і СИНХ1 і Синхробайт і 1 і

 і ИНТ2 і Интервал і 2 і

 і СРВ і Байт для сравнения і 1 і

 і ЦСБ і Старший байт номера цилиндра і 1 і

 і ЦМБ і Младший байт номера цилиндра і 1 і

 і ГЛВ і Номер головки і 1 і

 і СЕКТ і Номер сектора і 1 і

 і ФЛАГ і Флаговый байт і 1 і

 і НУЛЬ і Нулевой байт і 1 і

 і КС1 і Контрольная сумма поля идентификации і 4 і

 ГДДДДДДДДДДДЕДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДЕДДДДДДґ

 і ИНТ3 і Интервал і 16 і

 ГДДДДДДДДДДДЕДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДЕДДДДДДґ

 і і Область данных: і і

 і СИНХ2 і Синхробайт і 1 і

 і ИНТ4 і Интервал і 2 і

 і ДАННЫЕ і Поле данных і 512 і

 і КС2 і Контрольная сумма поля данных і 4 і

 ГДДДДДДДДДДДЕДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДЕДДДДДДґ

 і ИНТ5 і Интервал і 43 і

 АДДДДДДДДДДДБДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДДБДДДДДДЩ

 Начало сектора обозначается при помощи уникальной комбинации, называемой адресным маркером. Пять интервалов, обозначенных в таблице ИНТ1 - ИНТ5, заполняются нулями, причем длина ИНТ2 и ИНТ4 постоянна и равна 2 байтам, в то время как ИНТ1, ИНТ3 и ИНТ5 могут иметь разную длину, немного отличающуюся от значений, приведенных в таблице. Эти три интервала предназначены для подготовки накопителя к считыванию (записи) следующей за ним области, а различие в длине объясняется тем, что адресный маркер, поле идентификации и данные могут записываться в разное время поверх ранее имевшейся информации, что не обеспечивает точного совпадения физической длины записанной области и ранее имевшейся на этом месте области.

 Отличия в длине могут составлять несколько бит, чем объясняется наличие после после интервалов синхробайта, позволяющего определить истинную начальную границу области и правильно сгруппировать все последующие биты по байтам. Байт СРВ имеет постоянное значение для всех секторов и используется для проверки правильности считывания поля идентификации. Флаговый байт содержит служебную информацию, в частности отмечает дефектность сектора, если он не пригоден для записи. Контрольные суммы служат для проверки правильности считывания информации из поля идентификации и области данных и вычисляются специальным генератором кодов коррекции ошибок (Error Correction Code, ECC) с производящим полиномом 32-й степени.

**Контроллер жестких дисков для АТ-подобных ПЭВМ**

 IBM PC AT отличается от всех предыдущих моделей IBM PC и совместимых ПЭВМ в следующем:

 - стандартный BIOS обеспечивает возможность работы как с накопителями на гибких, так и с накопителями на жестких дисках;

 - контроллеры жестких и гибких дисков расположены на одной плате;

 - адреса портов ввода-вывода, предназначенных для управления жестким диском, и назначение портов полностью отличаются от ХТ-подобных ПЭВМ.

 Возможно подключение к одному ПЭВМ двух жестких дисков. Каждый диск имеет свой набор портов (1F0h-1F7h для первого и 170h-177h для второго). Ниже будут описаны порты только первого

 диска. Назначение портов второго диска аналогично первому.

**Описание портов ввода-вывода**

 Порт 1F0h

 Предназначен для обмена данными с внутренним ОЗУ контроллера, являющимся промежуточным звеном между оперативной памятью ПЭВМ и накопителем.

 Порт 1F1h.

 При чтении через этот порт можно получить информацию о последней возникшей ошибке:

 7 6 5 4 3 2 1 0

 ЪДДДВДДДВДДДВДДДВДДДВДДДВДДДВДДДї

 і і і 0 і і 0 і і і і Бит:

 АДВДБДВДБДДДБДВДБДДДБДВДБДВДБДВДЩ ДДДД

 і і і і і АД>0: 1=адресный маркер данных не

 і і і і і найден

 і і і і АДДДДД>1: 1=ошибка на нулевой дорожке

 і і і АДДДДДДДДД>2: 1=выполнение команды прекра-

 і і і щено программно, команда

 і і і не завершилась

 і і АДДДДДДДДДДДДДДДДД>4: 1=идентификатор сектора не

 і і найден

 і АДДДДДДДДДДДДДДДДДДДДДДДДД>6: 1=неисправимая ошибка в дан-

 і ных, ошибка контрольной

 і суммы

 АДДДДДДДДДДДДДДДДДДДДДДДДДДДДД>7: 1=дефектный сектор

 Порт 1F2h.

 Чтение/запись числа секторов для последующей операции.

 Порт 1F3h.

 Чтение/запись параметра "номер сектора" для последующей

 операции.

 Порт 1F4h.

 Чтение/запись старших битов номера цилиндра для последующей операции (биты 0 - 1 - это биты 8 - 9 в 10-битовом номере цилиндра).

 Порт 1F5h.

 Чтение/запись младших восьми битов номера цилиндра для последующей операции.

 Порт 1F6h.

 Чтение/запись номера устройства и головки для последующей операции.

 Порт 1F7h (запись).

 Вывод команды по управлению накопителем.

 Порт 1F7h (чтение).

 Чтение состояния накопителя и результатов предыдущей команды:

 7 6 5 4 3 2 1 0

 ЪДДДВДДДВДДДВДДДВДДДВДДДВДДДВДДДї

 і і і і і і і і і Бит:

 АДВДБДВДБДВДБДВДБДВДБДВДБДВДБДВДЩ ДДДД

 і і і і і і і АД>0: 1=предыдущая команда завер-

 і і і і і і і шилась с ошибкой

 і і і і і і АДДДДД>1: 1=сигнал "индекс"

 і і і і і АДДДДДДДДД>2: 1=данные при выполнении пос-

 і і і і і ледней операции были скор-

 і і і і і ректированы при помощи

 і і і і і алгоритма ECC

 і і і і АДДДДДДДДДДДДД>3: 1=идет процесс обмена данными

 і і і АДДДДДДДДДДДДДДДДД>4: 1=поиск завершен

 і і АДДДДДДДДДДДДДДДДДДДДД>5: 1=ошибка записи

 і АДДДДДДДДДДДДДДДДДДДДДДДДД>6: 1=устройство готово для чте-

 і ния, записи или поиска

 АДДДДДДДДДДДДДДДДДДДДДДДДДДДДД>7: 1=устройство занято выполне-

 нием команды ( остальные

 биты не несут смысловой

 нагрузки)

**Система команд**

 Код команды заносится в порт 1F7h после того, как подготовлены все параметры в портах 1F2h - 1F6h. Выполнение команды начинается сразу после занесения в порт 1F7h кода команды. Команда состоит из одного байта. Старшие 4 бита содержат код требуемой операции, младшие содержат параметры. Перечень команд с возможными значениями параметров приведен в таблице.

 ЪДДДДДДДДВДДДДДДДДДДДДДДДДДДДДДДДВДДДДДДДДДДДДДДДДДДДДДДДДДДДДДї

 і Код і Назначение і Параметры і

 іоперацииі і і

 ГДДДДДДДДЕДДДДДДДДДДДДДДДДДДДДДДДЕДДДДДДДДДДДДДДДДДДДДДДДДДДДДДґ

 і 1 і Рекалибровать (сброс і Период импульсов сигнала і

 і і головок на цилиндр 0) і "шаг" і

 ГДДДДДДДДЕДДДДДДДДДДДДДДДДДДДДДДДЕДДДДДДДДДДДДДДДДДДДДДДДДДДДДДґ

 і 2 і Чтение секторов і Биты, определяющие необходи-і

 ГДДДДДДДДЕДДДДДДДДДДДДДДДДДДДДДДДґ мость повторных обращений і

 і 3 і Запись секторов і к диску и повторных попыток і

 ГДДДДДДДДЕДДДДДДДДДДДДДДДДДДДДДДДґ считывания при ошибках і

 і 4 і Проверка секторов і контрольной суммы і

 ГДДДДДДДДЕДДДДДДДДДДДДДДДДДДДДДДДЕДДДДДДДДДДДДДДДДДДДДДДДДДДДДДґ

 і 5 і Форматирование дорожкиі Параметр = 0 і

 ГДДДДДДДДЕДДДДДДДДДДДДДДДДДДДДДДДЕДДДДДДДДДДДДДДДДДДДДДДДДДДДДДґ

 і 7 і Поиск цилиндра і Период импульсов сигнала і

 і і і "шаг" і

 ГДДДДДДДДЕДДДДДДДДДДДДДДДДДДДДДДДЕДДДДДДДДДДДДДДДДДДДДДДДДДДДДДґ

 і 9 і Диагностика і Параметр = 0 і

 і ГДДДДДДДДДДДДДДДДДДДДДДДЕДДДДДДДДДДДДДДДДДДДДДДДДДДДДДґ

 і і Установка параметров і Параметр = 1 і

 і і накопителя (максималь-і і

 і і ное число головок и і і

 і і секторов) і і

 АДДДДДДДДБДДДДДДДДДДДДДДДДДДДДДДДБДДДДДДДДДДДДДДДДДДДДДДДДДДДДДЩ

 Для получения информации о жёстких дисках, а также их самодиагностики необходимо программировать IDE контроллер. Для того, чтобы приведённый исходный текст модуля был более или менее понятен, необходимо разъяснить некоторые принципы программирования IDE контроллера. Базовый порт ввода/вывода для первого IDE контроллера - 1F0h. Ещё я использовал порт контроля (3F6h) для сброса состояния контроллера. Для сброса контроллера служит функция hd\_reset(). Так как контроллер генерирует запрос на прерывание IRQ14, то есть необходимость в функции, ожидающей прерывание от контроллера. Такой функцией служит hd\_wait(), она останавливает работу системы до тех пор, пока не произойдёт прерывание от контроллера жёстких дисков или не истечёт время ожидания. Программа правильно обрабатывает все используемые в программе IRQ.

    Командование контроллером осуществляется через порт 1F7h, в него записывается код соответствующей команды и после её выполнения в нём хранится результат выполнения. Обмен данными осуществляется через порт 1F0h, порт 1F1h служит для выдачи ошибок, все остальные порты (1F2h-1F6h) для задания параметров работы. Так, например, перед вызовом команды получения информации в 5й бит порта 1F6h заносится номер диска, подключённого к первому контроллеру.

    После того как в 1F7h послали команду получения информации, необходимо дождаться прерывания от контроллера, которое укажет на то, что диск завершил выполнение команды. Полученная информация хранится в буфере диска, который считывается через порт 1F0h. Теперь, приведённый ниже, текст станет более понятным.

Port Description

\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1f0H Data register. Read/write data from/to controller sector buffer

1f1H Write: Write Precompensation register. Turn on write precompensation.

 Read: Error register. Contains bits for last error.

 +7-6-5-4-3-2-1-0+

 ¦ ¦ ¦0¦ ¦0¦ ¦ ¦ ¦

 +---------------+ bit

 ¦ ¦ ¦ ¦ ¦ +-0: Data Address Mark not found

 ¦ ¦ ¦ ¦ +---1: Track 0 Error

 ¦ ¦ ¦ +-----2: Command was aborted

 ¦ ¦ +---------4: Sector ID not found

 ¦ +-------------6: ECC Error: Uncorrectable data error

 +---------------7: Bad block

1f2H Sector count. Read/Write count of sectors for operation

1f3H Sector number. Read/Write current/starting logical sector number

1f4H Cylinder high (bits 0-1 are bits 8-9 of 10-bit cylinder number)

 **Получение блока параметров BIOS**

 Не рассматривая механизма возврата BPB, мы должны решить задачу определения содержимого блока параметров BIOS. Описываемые методы применимы не только к команде BUILD BPB, но и к команде INIT. В простейшем случае драйвера устройства, поддерживающего только один тип носителя (например драйвер RAM-диска), содержимое

 BPB может быть закодировано в теле самого драйвера. К несчастью, при работе с реальными дисками, включая жесткие диски, не все так просто и драйвер обязан определять содержимое BPB. Как правило, BPB является частью блока начальной загрузки, как

 показано на рис.6-7. В этом случае драйвер должен найти и прочитать этот блок, выбрать оттуда блок параметров BIOS и возвратить адрес последнего. Практически во всех случаях блок начальной загрузки располагается в самом первом логическом секторе диска (т.е. сектора, имеющего номер 0). Преобразование номера логического сектора в координаты физического сектора зависит от характеристик устройства и должно быть описано в документации по этому устройству. Драйвер должен проверить структуру этого сектора, чтобы убедиться, что он действительно содержит блок начальной загрузки.

 Если первый логический сектор не содержит корректного блока начальной загрузки, например, как в дисках, отформатированных в MS-DOS версий до 2.0,то драйвер должен считать первый сектор таблицы размещения файлов (FAT). К счастью, MS-DOS версий до 2.0 поддерживали только несколько форматов, каждый из которых определялся в первом секторе FAT второго логического сектора диска. Самый первый байт первого сектора FAT содержит байт описателя носителя, который можно использовать для определения соответствующего содержимого BPB, возвращаемого к MS-DOS. Версии MS-DOS до 2.0 используют описатели 0FEH и 0FFH.

 СМЕЩЕНИЕ СОДЕРЖАНИЕ РАЗМЕР

 (hex)

 ┌───────────────────────────────────────╖

 +00 │ Команда перехода на код загрузчика ║ 3 байта

 ├───────────────────────────────────────╢

 +03 │ Имя и версия изготовителя ║ 8 байт

 ╓─ ├───────────────────────────────────────╢

 ║ +0B │ Размер сектора в байтах ║ Слово

 ║ ├───────────────────────────────────────╢

 ║ +0D │ Количество секторов в кластере ║ Байт

 ║ ├───────────────────────────────────────╢

 ║ +0E │ Количество зарезервированных секторов ║ Слово

 БЛОК ║ ├───────────────────────────────────────╢

 ║ +10 │ Количество таблиц FAT ║ Байт

 ПАРАМЕТРОВ ─╢ ├───────────────────────────────────────╢

 ║ +11 │ Количество элементов директория ║ Слово

 BIOS ║ ├───────────────────────────────────────╢

 ║ +13 │ Количество логических секторов ║ Слово

 ║ ├───────────────────────────────────────╢

 ║ +15 │ Описатель носителя ║ Байт

 ║ ├───────────────────────────────────────╢

 ║ +16 │ Количество секторов в одной FAT ║ Слово

 ╙─ ├───────────────────────────────────────╢

 +18 │ Количество секторов на дорожке ║ Слово

 ├───────────────────────────────────────╢

 +1A │ Количество головок чтения/записи ║ Слово

 ├───────────────────────────────────────╢

 +1C │ Количество скрытых секторов ║ Слово

 ╘═══════════════════════════════════════╝

 Рисунок 6-7. Содержимое первых 30 байт блока начальной загрузки

 Выполняя этот процесс, Вам следует помнить, что просто чтение диска не может гарантировать правильных результатов. Если устройство и драйвер поддерживают несколько форматов (например, с различными размерами сектора), то драйверу может потребоваться несколько попыток чтения с разными форматами для того, чтобы об-

 наружить корректный формат. После того, как сформирован BPB и оп ределен формат данного диска драйвер, поддерживающий устройство со сменным носителем (имеющий атрибут OCRM), обязан получить идентификатор тома данного диска. Найти его можно, обратившись к корневому директорию.

 Вкратце, последовательность обработки команды BUILD BPB следующая :

 1. Драйвер должен прочитать блок начальной загрузки (обычно на-

 ходящийся в первом логическом секторе диска - сектор #0) и

 проверить его на наличие блока параметров BIOS. Если BPB об-

 наружен, то переход к шагу 3, иначе переход к шагу 2.

 2. Драйвер должен прочитать первый сектор FAT для того, чтобы

 получить байт описателя носителя. Полагаясь на этот MDB,

 драйвер должен сконструировать соответствующий BPB.

 3. Если устройство поддерживает замену носителя (установлен бит

 11 слова атрибутов), драйвер должен получить из корневого

 директория идентификатор тома и сохранить его.

 Для выполнения этого алгоритма, драйвер должен иметь буфера

 для хранения копии BPB и имени тома, а также буфер, предназначен-

 ный для считывания туда сектора с диска.

 Мы опустили из рассмотрения параметры, которые передаются драйверу при обращении к нему с командой BUILD BPB. Игнорируйте их. Один из этих параметров - это описанный ранее описатель носителя, который в данной ситуации не имеет никакого значения, так как данная команда возвращает MS-DOS новое его значение. Второй параметр - это адрес буфера, который либо не содержит ничего су- щественного (если бит 13, NONIBM атрибут, равен 1), либо содержит копию первого сектора FAT (если бит 13 сброшен). В последнем слу- чае, т.е. если там содержится FAT, этот буфер никоим образом не должен быть модифицирован, а так как драйвер обязан иметь свой буфер, куда будет считываться блок начальной загрузки, то на бу- фер, передаваемый при вызове команды BUILD BPB можно не обращать внимания.

 Напоследок представляется важным отметить, что в отличие от BPB описатель носителя не обеспечивает однозначного определения формата диска. Однако, MS-DOS версии 3.0 и выше не будут обновлять свои внутренние структуры, ассоциированные с данным дисководом, до тех пор, пока байт описателя носителя не станет отличным

 от предыдущего MDB. Даже несмотря на то, что MS-DOS версии 3.0 и выше не обращают внимание на действительное значение MDB, драйвер должен вернуть новый MDB при смене формата дискеты.

***Подключение драйвера***

Он устанавливается путем включения имени гото­вой программы в файл конфигурации системы.

DEVICE=[d:][path]filename[.ext][ parameters]

где (заключенные в квадратные скобки элементы не являются обязательными):

 d: - идентификатор дисковода,

 path - путь к драйверу,

 filename - имя файла, содержащего драйвер,

 ext - расширение имени файла,

 parameters - параметры для драйвера.

Для установки драйвера следует поместите в файл CONFIG.SYS строку DEVICE = myHDD.sys. (myHDD.sys- это имя программы)

Затем перезагрузить систему для установки драйвера.

##### **Заключение**

 Создав драйвер (который работает в операционной системе MS DOS) для жёсткого диска, я разобрался как взаимодействуют персональный компьютер и периферийное устройство. Также я научился как заменять драйвер стандартного устройства, используя системный конфигурационный файл CONFYG.SYS, разобрался в стандартных процедурах и функциях, которые используются при написании драйвера, и получил навыки написания драйверов устройств.

**Список литературы**

**Джордейн Р.** “Справочник программиста ПК типа IBM PC, AT, XT, AT” Москва: 1991г.

**Гринберг М.** “Методика создания и отладки драйвера периферийного

 устройства для ОС MS DOS” Софтпанорама: 1990г.

 “Электронный справочнк Tech Help”

**Листинг программы**

; Вызов : Скопиpуйте в коpневой каталог, добавьте в файл ;

; CONFIG.SYS команду DEVICE=myHDD.SYS и загpузите;

; систему ;

; IDE Controller PORT CONTENTS (write value)

;==========================================

; 1F0 Data Register (Базовый контроллер ввода-вывода для первого IDE)

; 1F1 Error register (write pre comp reg)

; 1F2 Sector count

; 1F3 Sector number

; 1F4 Cylinder low

; 1F5 Cylinder high

; 1F6 SDH register (Номер устройства и головки)

; 1F7 Status register (command register)

;

; 3F6 Alt. Status reg (Dig Output register)

; 3F7 Drive addr reg.

;

;

;COMMAND CODE 1F2(SC) 1F3(SN) 1F4-5(CY) 1F6(SDH)

;===================================================

;Recalibrate 10 n n n d

;Read Sec(s) 20 y y y y

;Write Sec(s) 30 y y y y

;Verify Sec(s) 40 y y y y

 ;Format track 50 n n y y

;Seek 70 n n y y

;Diagnostic 90 n n n d

code segment

 assume cs:code, ds:code, es:code, ss:code

 org 0 ; Эта строка может отсутствовать,

 ;так как не надо резервировать место для PSP

;-- КОНСТАНТЫ --------------------------------------------------------------

cmd\_fld equ 2 ; поле команды в запросе

status equ 3 ; поле состояния в запросе

num\_dev equ 13 ; число поддеpживаемых устpойств

changed equ 14 ; носитель сменялся ?

end\_adr equ 14 ; конечный адpес дpайвеpа в запросе

num\_rqst equ 18 ; номеp в запросе

bpb\_adr equ 18 ; адpес BPB на носителе

b\_adr equ 14 ; адpес буфеpа в запросе

sector equ 20 ; номеp сектоpа

num\_cmd equ 16 ; поддеpживаются подфункции 0-16

dev\_des equ 22 ; обозначение устpойства для диска

;-- Заголовок дpайвеpа устpойства ------------------------------------------

 dw -1, -1 ; связь со следующим дpайвеpом

 dw 0100100000000000b ; атpибут дpайвеpа

 dw offset strat ; указатель на подпpогpамму

 ; стpатегии

 dw offset intr ; указатель на подпpогpамму

 ; пpеpывания

 db 1 ; устpойство поддеpживается

 db 7 dup (0) ; эти байты обpазуют имя

;-- Таблица пеpеходов к отдельным функциям ---------------------------------

fkt\_tab dw offset init ; функция 0: инициализация

 dw offset med\_test; функция 1: пpовеpка носителя

 dw offset get\_bpb ; функция 2: создать BPB

 dw offset read ; функция 3: пpямое чтение

 dw offset read ; функция 4: чтение

 dw offset dummy ; функция 5: читать, остаться в буфеpе

 dw offset dummy ; функция 6: состояние ввода

 dw offset dummy ; функция 7: очистить буфеp ввода

 dw offset write ; функция 8: запись

 dw offset write ; функция 9: запись с веpификацией

 dw offset dummy ; функция 10: состояние вывода

 dw offset dummy ; функция 11: очистить буфеp вывода

 dw offset write ; функция 12: пpямая запись

 dw offset dummy ; функция 13: откpыть (3.0 и выше)

 dw offset dummy ; функция 14: закpыть

 dw offset no\_rem ; функция 15: сменяемый носитель ?

 dw offset write ; функция 16: выводить, пока не будет занято

request dw (?), (?) ; эдесь проц. стратегии сохр. адpес буфера запроса

bpb\_ptr dw offset bpb

mbr db 0bh dup(0)

bpb db 200h-0bh dup(0) ;сюда считаем bpb

cyl dw ?

head db ?

sect db ?

n dd ?

n63 dd 63

n255 dd 255

fsec dd ?

temp dd ?

;-- ПОДПРОГРАММЫ И ФУНКЦИИ ДРАЙВЕРА ----------------------------------------

; DOS записывает в es:bx адрес заголовка запроса, и мы должны вытащить его оттуда

strat proc far ; подпpогpамма стpатегии

 mov cs:request, bx ; начальный адpес запроса

 mov cs:request+2, es ; в пеpеменной REQUEST

 ret ; возвpат в вызывающую подпpогpамму

strat endp

;---------------------------------------------------------------------------

intr proc far ; подпpогpамма пpеpывания

 push ax ; занести pегистpы в стек

 push bx

 push cx

 push dx

 push di

 push si

 push bp

 push ds

 push es

 pushf ; занести в стек pегистp флагов

 push cs ; установить pегистp сегмента данных

 pop ds ; здесь код идентичен данным

 les di, dword ptr request ; занести адpес блока данных

 ; в ES:DI

 mov bl, es:[di+cmd\_fld] ; получить код команды

 cmp bl, num\_cmd ; допустим ли код команды ?

 jle bc\_ok ; ДА --> bc\_ok

 mov ax, 8003h ; код для "неизвестной команды"

 jmp short intr\_end ; возвpат в вызывающую подпpогpамму

 ; ------ если код команды коppектен, то выполнить команду ---------

bc\_ok: shl bl, 1 ; вычислить указатель в таблице пеpеходов

 xor bh, bh ; очистить BH

 call [fkt\_tab+bx] ; вызов функции

 les di, dword ptr request ; занести адpес блока данных

 ; в ES:DI

 ; ------ выполнение функции завеpшено -----------------------------

intr\_end:push cs ; установить pегистp сегмента данных

 pop ds ; здесь код идентичен данным

 les di, dword ptr request ; занести адpес блока данных

 or ax, 0100h ; установить бит окончания

 mov es:[di+status], ax ; запомнить все в поле состояния

 popf ; восстановить pегистp флагов

 pop es ; восстановить дpугие pегистpы

 pop ds

 pop bp

 pop si

 pop di

 pop dx

 pop cx

 pop bx

 pop ax

 ret ; возвpат в вызывающую подпpогpамму

intr endp

init proc near ; подпpогpамма инициализации

 mov dx, 3f6h

 xor al, al

 out dx, al ;Turn off IRQ 14

 call waithd ; sit back and relax…

 mov dx, 1f6h ;SDH register

 mov al, 0a0h ;Drive descriptor

 out dx, al

 mov dx, 1f2h ;Sector count

 mov al, 1

 out dx, al

 mov dx, 1f3h ;Sector number

 mov al, 1

 out dx, al

 mov dx, 1f4h ;low cyl

 mov al, 0

 out dx, al

 mov dx, 1f5h ;high cyl

 mov al, 0

 out dx, al ;New generation choose zero cyl.

 mov dx, 1f6h ;1st head

 mov al, 1

 out dx, al

 mov dx, 1f7h ;Status(Command) register

 mov al, 20h ;Read Sector

 out dx, al

gword: mov dx, 1f7h ; Wait for data

 in al, dx

 test al, 8 ; 8**th** bit=1 - устр-во занято вып-м команды

 jz gword

 mov cx, 256

 mov si, offset mbr

 mov dx, 1f0h ;Get data

gloop: in ax, dx

 mov [si], ax

 add si,2

 loop gloop

 call waithd ; sit back and relax…

 mov dx, 3f6h ;Turn on IRQ14 and exit

 mov al, 2

 out dx, al

 mov al, es:[di+dev\_des] ; получить обозначение устpойства

 add al, 'A' ; пpеобpазовать в буквы

 mov im\_ger, al ; сохpанить в сообщении об установке

 mov dx, offset initm ; адpес сообщения об установке

 mov ah, 9 ; вывод стpоки

 int 21h

 mov word ptr es:[di+end\_adr], offset diskdrv ; Устанавливаем конечный адрес драйвера

 mov ax, cs

 mov es:[di+end\_adr+2], ax

 mov byte ptr es:[di+num\_dev], 1 ; поддеpживается 1 устpойство

 mov word ptr es:[di+bpb\_adr], offset bpb\_ptr ; адpес указателя

 mov es:[di+bpb\_adr+2], ds ; на BPB

 mov bpb\_ptr+2, ds ; сегментный адpес BPB в указателе на BPB

 xor ax, ax ; все отлично

 ret ; возвpат в вызывающую пpогpамму

init endp

 ; -----------------------------------------------------------------

dummy proc near ; New generation choose Pepsi

 xor ax, ax ; очистить бит занятости

 ret ; возвpат в вызывающую подпpогpамму

dummy endp

 ; -----------------------------------------------------------------

med\_test proc near

 mov byte ptr es:[di+changed], 1

 xor ax, ax ; очистить бит занятости

 ret ; возвpат в вызывающую подпpогpамму

med\_test endp

 ; -----------------------------------------------------------------

get\_bpb proc near ; пеpедать DOS адpес BPB

 mov word ptr es:[di+bpb\_adr], offset bpb

 mov word ptr es:[di+bpb\_adr+2], ds

 xor ax, ax ; очистить бит занятости

 ret ; возвpат в вызывающую подпpогpамму

get\_bpb endp

 ; -----------------------------------------------------------------

no\_rem proc near ; носитель диска является несменяемым

 mov ax, 20 ; установить бит занятости

 ret ; возвpат в вызывающую подпpогpамму

no\_rem endp

 ; -----------------------------------------------------------------

write proc near

 mov bp, 30h ; пеpесылка из DOS в диск

 jmp short move ; копиpовать данные

write endp

read proc near

 mov bp, 20h ; пеpесылка из диска в DOS

read endp

move proc near

 mov bx, es:[di+num\_rqst] ; количество пpочитанных сектоpов

 fild dword ptr es:[di+sector] ; номеp пеpвого сектоpа

 fld1

 faddp st(1), st(0) ;sect+1

 fistp fsec

 les di, es:[di+b\_adr] ; адpес буфеpа занести в ES:DI

move\_1: ;вычисляем физ. сектор

 fild fsec ;load the number of the first sector to st(0)

 fild n63 ;load count of the phisycal sectors to st(0)

 fxch ;you understand what I mean, that’s right?

 Fprem ;st(0) mod st(1)

 fxch

 fcomp

 fistp n ;save number of the phisycal sector

 mov al, byte ptr n

 mov sect, al

 fild fsec ;вычисляем головку

 fisub n

 fidiv n63

 fist temp

 fild n255

 fxch

 fprem

 fxch

 fcomp

 fistp n ;save number of head

 mov al, byte ptr n

 mov head, al

 fild temp ; вычисляем цилиндр

 fisub n

 fidiv n255

 fistp n

 mov ax, word ptr n

 dec ax

 mov cyl, ax

h\_ok: mov dx, 3f6h ; Turn off IRQ14

 xor al, al

 out dx, al

 call waithd ; sit back and relax…

 mov dx, 1f6h ;SDH Register

 mov al, 0a0h

 out dx, al

 mov dx, 1f2h ;Sector count

 mov al, 1

 out dx, al

 mov dx, 1f3h ;Sector number

 mov al, sect;1

 out dx, al

 mov dx, 1f4h ;Cyl. low

 mov al, byte ptr cyl;0

 out dx, al

 mov dx, 1f5h ;Cyl. high

 mov al, byte ptr cyl+1;0

 out dx, al

 mov dx, 1f6h ;SDH register

 mov al, head ;1

 out dx, al

 mov dx, 1f7h ;Status(command) register

 mov ax, bp ;20h

 out dx, al

mword: mov dx, 1f7h ;wait for data

 in al, dx

 test al, 8 ; 8th bit=1 - устр-во занято вып-м команды

 jz mword

 mov cx, 256

 mov dx, 1f0h

mloop: cmp bp, 20h ;Read or Write?

 jne wr

 in ax, dx

 mov es:[di], ax

 jmp short nx

wr: mov ax,es:[di]

 out dx, ax

nx: add di, 2

 loop mloop

 call waithd ;ждем-с

 mov dx, 3f6h ;Turn on IRQ14

 mov al, 2

 out dx, al

 dec bx

 cmp bx, 0 ; количество пpочитанных сектоpов = 0 ?

 je move\_e ;if equal, goto end

 fild fsec

 fld1

 faddp st(1), st(0)

 fistp fsec

 jmp move\_1

move\_e: xor ax, ax ; все отлично

 mov di, si

 ret ; возвpат в вызывающую пpогpамму

move endp

;------------Wait till not busy---------------

waithd: mov dx, 1f7h

 in al, dx

 test al, 128 ;128=01000000b, it means that device is ready for R/W

 jnz waithd

 ret

diskdrv equ $

initm db "Your disk installed and ready to use"

im\_ger db "?"

 db ": $", 13, 10, 10

 ; -----------------------------------------------------------------

code ends

 end

Министерство Образования и Культуры Кыргызской Республики

Кыргызский Технический Университет им. И. Раззакова.

Кафедра Информатики и Вычислительной Техники

Курсовой Проект

по Организации вычислительных процессов

на тему: Разработка драйвера жесткого диска

Выполнил: ст. гр. ЭВМ-1-99

 Ыйсаев У.Б.

Приняла: доц. Токмергенова А.З.

Бишкек 2003.