БИЛЕТ № 3

1) Состав ОС. Назначение отдельных частей.

Если программа меняет информацию, с которой она работает (помимо своей собственной), то это *обрабатывающая* программа (компиляторы, загрузчики, редакторы связей), если не меняет, – то *управляющая*.

Управляющие программы выполняют следующие действия:

1. **Управление** – на уровне ядра, в режиме супервизора осуществляется:
   - отслеживание входа и выхода задания в системе;
   - отслеживание выполнения собственной задачи и управление ее выполнением;
   - обеспечение работы систем управления файлами, памятью, внешними устройствами.
2. **Управление задачами.** Следит за выполнением задачи на ресурсах системы
3. **Система управления файлами.**
4. **Управление памятью.**
   - программное (обеспечивает эффективное функционирование памяти в соответствии с ее организацией; программа решает свои задачи на уровне процесса, то есть память выделяется процессу);
   - микропрограммное (программа запрашивает у админа, какие разделы необходимо выполнять; у каждого раздела – своя очередь);
5. **Управление внешними устройствами.**
6. Управление заданиями отслеживает прохождение задания от входа до выхода

Современные ОС имеют сложную структуру, каждый элемент которой выполняет определенные функции по управлению компьютером.

- Управление файловой системой. Процесс работы компьютера сводится к обмену файлами между устройствами. В ОС имеются программные модули, управляющие файловой системой.
- Командный процессор. В состав ОС входит специальная программа – командный процессор, – которая запрашивает у пользователя команды (запуск программы, копирование, удаление и т.д.) и выполняет их.
- Драйверы устройств. В состав ОС входят драйверы устройств, специальные программы, которые обеспечивают управление работой устройств и согласование информационного обмена с другими устройствами, позволяют производить настройку параметров устройств. Каждому устройству соответствует свой драйвер. Программы (драйверы) для поддержки наиболее распространенных устройств входят в Windows, а для остальных устройств поставляются вместе с этими устройствами или контроллерами. Можно самому установить или переустановить драйвер.
- Графический интерфейс. Для упрощения работы пользователя в состав ОС входят программные модули, создающие графический пользовательский интерфейс. Пользователь может вводить команды с помощью мыши, а не с клавиатуры.
- Сервисные программы. (утилиты). Позволяют обслуживать диски (проверять, сжимать, дефрагментировать), выполнять операции с файлами (архивировать и т.д.), работать в сетях и т.д.
- Справочная система. Позволяет оперативно получить необходимую информацию о функционировании ОС.

1 Виды модулей. Виды: 1) исходный (текст программы); 2) объектный; 3) загрузочный; 4) исполняемый (абсолютный). Системные программы (компилятор, редактор связей, загрузчик) используются для преобразования от 1) к 4).

2 Отличие загрузочного модуля от объектного.Загрузочный модуль имеет все для своего выполнения . Написан на машинно-ориентированном языке, но быть выполненным не может. При использовании компилятора из исходного модуля получается объектный модуль, при использовании редактора связи из объектного получается загрузочный, далее из него при использовании загрузчика получается исполняемый модуль.

3 Отличие загрузочного модуля от абсолютного. Загрузочный модуль имеет все для своего выполнения . Написан на машинно-ориентированном языке, но быть выполненным не может. Нужно настроить адресные константы. Абсолютный – все адресные константы уже настроены.

4 Дать определение – резидентный модуль. модуль, который резидентно находится в памяти. (постоянно)Резидентные проги. – находятся в ОП, поддерживают быстрый отклик системы, Проги обработки прерываний, Процессы 4/ работы с ВУ, управление памятью, управление процессами, начальная прога. Управления заданиями.

5 Дать определение – транзитный модуль. Программы, связанные с выполнением функций ОС, но не находящиеся постоянно в ОП называются транзитными. Эти программы вызываются в ОП по мере необходимости. – проги., которые могут понадобится 4/ выполнения ф-ций ОС (но не резидентные) – могут вызываться по оверлейной || динамически-последовательной схеме в транзитную зону.

6 Связь модулей по управлению. Какие операции, какими программами выполняются. Связь по управлению предусматривает сохранение состояния модуля на момент перехода на вызываемый модуль, причем действия по этому сохранению выполняет вызываемый модуль. Операции: возврат из i-1 модуля только в i+1; возврат из i+1 в i.

Вызывающий модуль после передачи управления стирается. Виды программ могут быть: повторно не исполняемые, повторно исполняемые, чистые процедуры. (via common regs) Связь по управлению – связь для организации возврата. Возврат модуль В должен иметь команду, делающую так, что в области сохранения модуля А сохраняется состояние регистров. В модуле В есть адрес области сохранения в модуле А. Сохраняется адрес возврата, для возврата адреса следующей команды.

7 Связь модулей по данным. Виды связей Виды: по данным и по управлению. По данным может быть через общие регистры или через адрес списка параметром. Связь по данным – данные через общий регистр || через системный регистр передается адрес списка параметров.

Свойства модуля. Стандартность внутренней структуры, функциональная завершенность, параметрическая универсальность, взаимная независимость.

8 Виды загрузчиков Отличаются выполнением основных 4-х функций: распределения памяти, настройки, редактирования и загрузки. Загрузчики делятся: Абсолютный загрузчик, настраивающий загрузчик, непосредственно - связывающий загрузчик.

9 Функции абсолютного загрузчика Распределение памяти, настройка, редактирование;

10 Функции настраиваемого загрузчика Распределение памяти, настройка, редактирование, загрузка. Инфу для него готовит компилятор.

11 Какие топы загрузчика работают с модулями COM и EXE. С COM - абсолютный; с EXE - настраивающий.

12 Какую информацию и как компилятор передает настраивающему загрузчику Настраивающий – работает в загр. Модуле которому больше ничего не надо. Постепенно его ф-ции взял на себя редактор связей. Непосредственно в настраивающем загрузчике каждый модуль может транслироваться отдельно. Чтобы передать сообщение редактору связей надо ему непосредственно указать, что надо транслировать. В каждом модуле в начале трансляции выделяются вектора перехода, внешние и внутренние.(Экспорт и Импорт процедур и ф-ций). Кроме того для выполнения настройки каждая команда отмечается битом переместимости. ОС выделяет и пользуетс глобально выделенной памятью, а загрузчик с локальной.

13 Функции редактора связей Редактирование связей.(Выполнение связывания подпрограмм являющихся внешними по отношению к загружаемому модулю). (С помощью редактора связи мы получаем из объектного модуля загрузочный модуль имеющий всё для своего исполнения).

14 Выходная и входная информация редактора связей Входная - объектный модуль, выходная - загрузочный модуль.

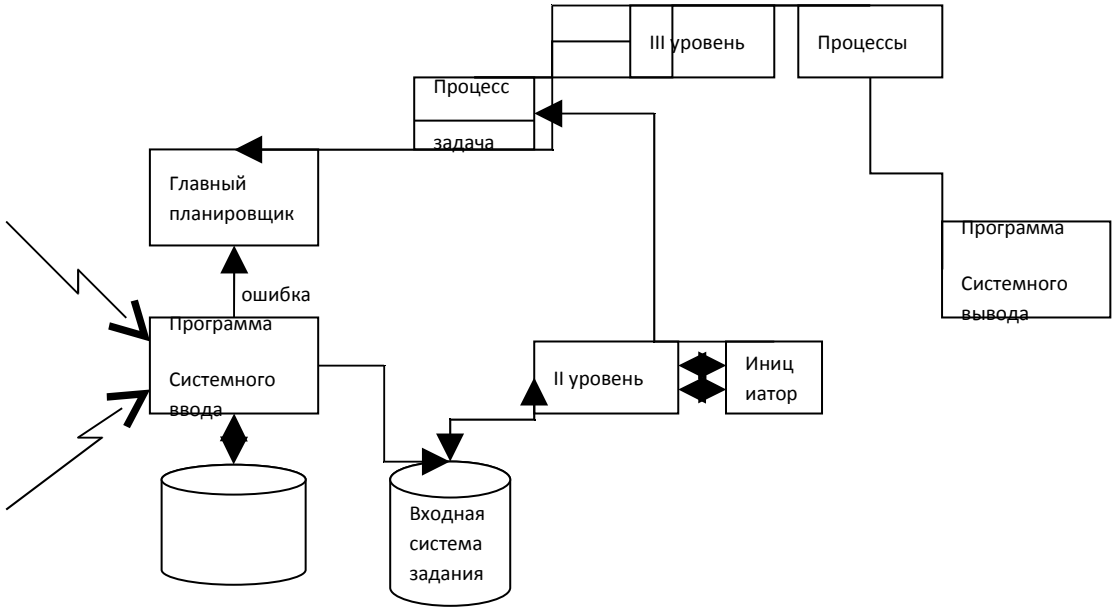
15 Дать определение – обрабатывающие программы ОС Программы выполнения стандартных (в рамках ОС) функций, обработки исключительных ситуаций. Обработка – изменение ин-фы с которой работает прога в составе ОС (дрова, загрузчик)

16 Дать определение – управляющие программы ОС Программы постоянно находящиеся в памяти (резидентные) организующие корректное выполнение процессов и функционирование всех устройств системы при решении задач. Составляют ядро ОС. Управление : Заданиями – слежение за прохождением заданий от входа до выхода на всех этапах его выполнения. Задачами – (Процессами) – слежение за всеми задачами активизированными в системе и процессами их выполнения на ресурсах. Памятью – решение задач эффективного u/ mem (internal) в соответствии с ее организацией.(Защита – согласование ин-фы в кешах) Данными – эффективное размещение и u/ данных на внешних носителях (проблема эффективности u/ процессора) Внешними ус-вами …

17 Основное отличие компилятора от интерпретатора Интерпретатор выполняет перевод части проги в машинные команды и тут же ее выполняет, выполняя все необходимые настройки адресных констант. Компилятор создает объектный модуль, который затем обрабатывается редактором связей. Во время работы интерпретатора исполняемый код программы записывается в фиксированное место и управление передается на стартовый адрес программы.

2) Этапы прохождения заданий через вычислительную систему.

Задание – это внешняя еденица работы системы, для которой система ресурсов не выделяет.



Программа системного ввода – планировщик 1 уровня
Командный интерпритатор берет входное задание. Схватывает задание, фотмирует командную строку, проверяет на правильность.
Задания накапливаются в буфере клавиатуры.
В том случае, если система считает, что есть ресурс для активизации задания, то активизируется планировщик 2-го уровня, который обрабатываетвходную очередь задания и обрабатывает задания с наивысшим приритетом.Передает управление инициатору. Инициатор проверяет наличие ресурсов для выполнения задания. Если ресурсов нет, то задание сбрасывается. Если всё в порядке, то образуется задача или процесс. Как только сформирован TCB система должна быть обязательно выполнена. При формировании PCB определяется программа подчиненная задаче, и данные, которые должна выполнить программа. Как только освобождается процессор, всплывает планировщик 3-го уровня, который обрабатывает TCB или PSB, ищет наиболее приоритетный процесс, который можно запустить

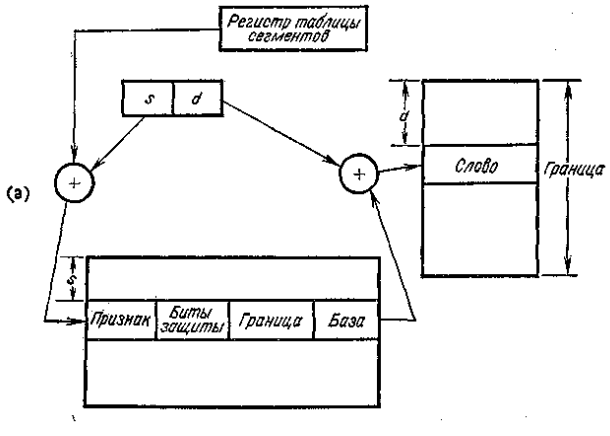
3) Особенности формирования исполнительного адреса в различных организациях памяти. Роль GDT и LDT.

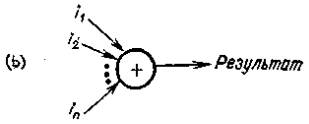
Доп. Инфа. Схема формирования исполнительного адреса при сегментной организации памяти.

В системе с сегментацией всякий адрес представляет собой пару *Is, d],* где *s* — имя сегмента, *ad* — смещение. Каждому заданию или процессу соответствует всегда присутствующая в памяти *таблица сегментов,* в которой каждому сегменту данного задания соответствует одна запись. С помощью этой таблицы система отображает программные адреса в истинные адреса оперативной памяти. Адрес таблицы хранится в аппаратном регистре, называемом *регистром таблицы сегментов.* В s-й *) строке таблицы находятся

1) *признак* (или *бит присутствия),* показывающий, присутствует ли s-й сегмент в данный момент в памяти;
2) *базовый адрес* s-го сегмента;
3) *граница,* указывающая количество ячеек, занимаемых данным сегментом;
4) биты защиты (не обязательные), используемые для контроля способа доступа.

Чтобы добраться до слова [s, *d*], надо *с* помощью регистра таблицы сегментов обратиться к таблице сегментов (рис. 4.4). В s-й строке таблицы указан адрес сегмента s в памяти. Его *d-я* ячейка содержит искомое слово [s, *d].* Следовательно, чтобы получить слово [s, *d],* потребуется два обращения к памяти: одно к таблице сегментов и одно к слову внутри сегмента. Поскольку сегменты бывают различной длины, не существует фиксированного верхнего предела для *d.* Поэтому для того, чтобы помешать заданию обращаться за пределы сегмента, необходимо знать значение границы.





Прежде чем система сможет вычислить адрес, аппаратным путем проверяется признак присутствия сегмента в оперативной памяти.. Если сегмент присутствует, то адрес в памяти можно вычислить автоматически, как описано в предыдущем абзаце. Если сегмент отсутствует в оперативной памяти, то происходит так называемое «прерывание из-за отсутствия сегмента», т. е. вырабатывается аппаратное прерывание, которое включает супервизорную программу обработки таких ситуаций. Эта программа отыскивает нужный сегмент во вспомогательной памяти и вводит его в оперативную. Если в оперативной памяти нет места для нового сегмента, то система освобождает его, выгнав один из находящихся в оперативной памяти сегментов на периферийное устройство.

В чем отличительная особенность формирования исполнительного адреса при страничной организации памяти.

В самом простом и наиболее распространенном случае страничной организации памяти (или paging) как логическое адресное пространство, так и физическое представляются состоящими из наборов блоков или страниц одинакового размера. При этом образуются логические страницы (page), а соответствующие единицы в физической памяти называют страничными кадрами (page frames). Страницы (и страничные кадры) имеют фиксированную длину, обычно являющуюся степенью числа 2, и не могут перекрываться. Каждый кадр содержит одну страницу данных. При такой организации внешняя фрагментация отсутствует, а потери из-за внутренней фрагментации, поскольку процесс занимает целое число страниц, ограничены частью последней страницы процесса.

Логический адрес в страничной системе – упорядоченная пара (p,d), где p – номер страницы в виртуальной памяти, а d – смещение в рамках страницы p, на которой размещается адресуемый элемент. Заметим, что разбиение адресного пространства на страницы осуществляется вычислительной системой незаметно для программиста. Поэтому адрес является двумерным лишь с точки зрения операционной системы, а с точки зрения программиста адресное пространство процесса остается линейным.

Описываемая схема позволяет загрузить процесс, даже если нет непрерывной области кадров, достаточной для размещения процесса целиком. Но одного базового регистра для осуществления трансляции адреса в данной схеме недостаточно. Система отображения логических адресов в физические сводится к системе отображения логических страниц в физические и представляет собой таблицу страниц, которая хранится в оперативной памяти. Иногда говорят, что таблица страниц – это кусочно-линейная функция отображения, заданная в табличном виде.

Интерпретация логического адреса показана на рис. 8.7. Если выполняемый процесс обращается к логическому адресу v = (p,d), механизм отображения ищет номер страницы p в таблице страниц и определяет, что эта страница находится в страничном кадре p', формируя реальный адрес из p' и d.

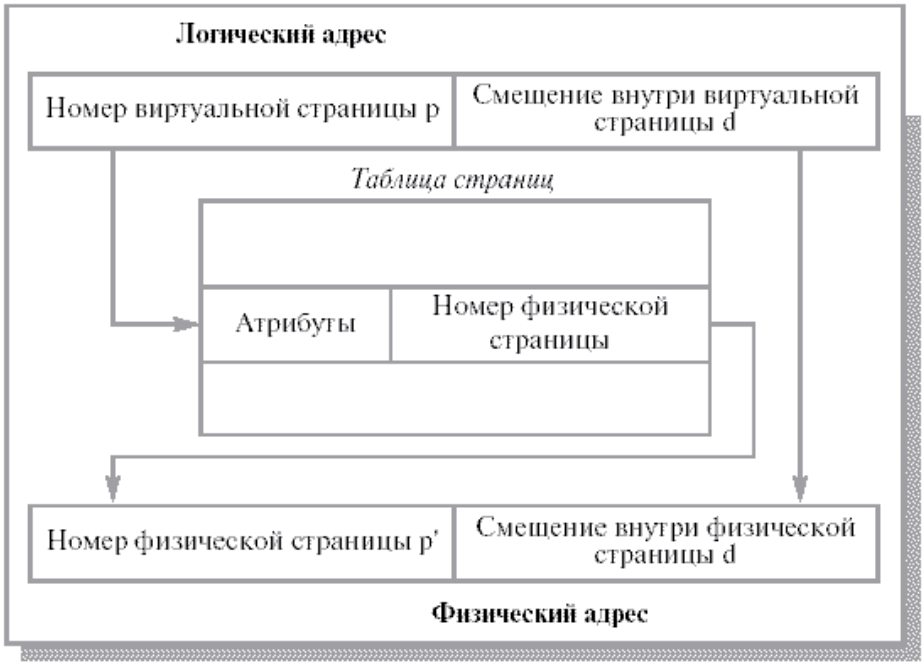


Рис. 8.7. Связь логического и физического адресов при страничной организации памяти

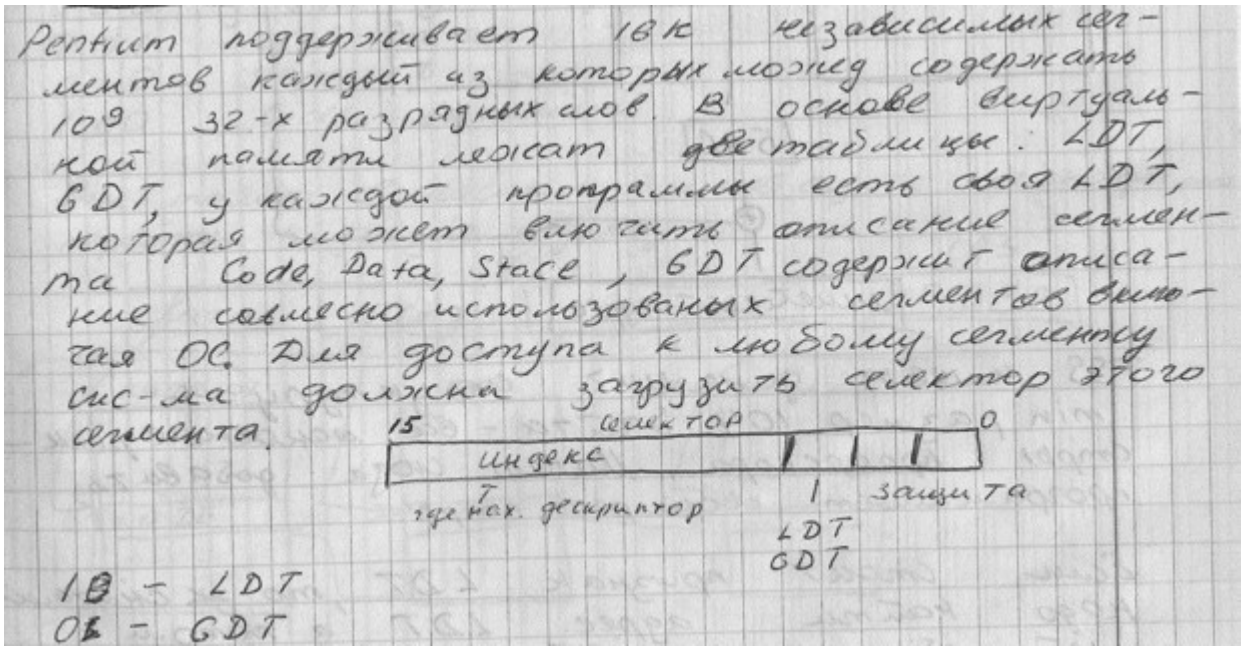
Таблица страниц (page table) адресуется при помощи специального регистра процессора и позволяет определить номер кадра по логическому адресу. Помимо этой основной задачи, при помощи атрибутов, записанных в строке таблицы страниц, можно организовать контроль доступа к конкретной странице и ее защиту.

Отметим еще раз различие точек зрения пользователя и системы на используемую память. С точки зрения пользователя, его память – единое непрерывное пространство, содержащее только одну программу. Реальное отображение скрыто от пользователя и контролируется ОС. Заметим, что процессу пользователя чужая память недоступна. Он не имеет возможности адресовать память за пределами своей таблицы страниц, которая включает только его собственные страницы.

Для управления физической памятью ОС поддерживает структуру таблицы кадров. Она имеет одну запись на каждый физический кадр, показывающий его состояние.

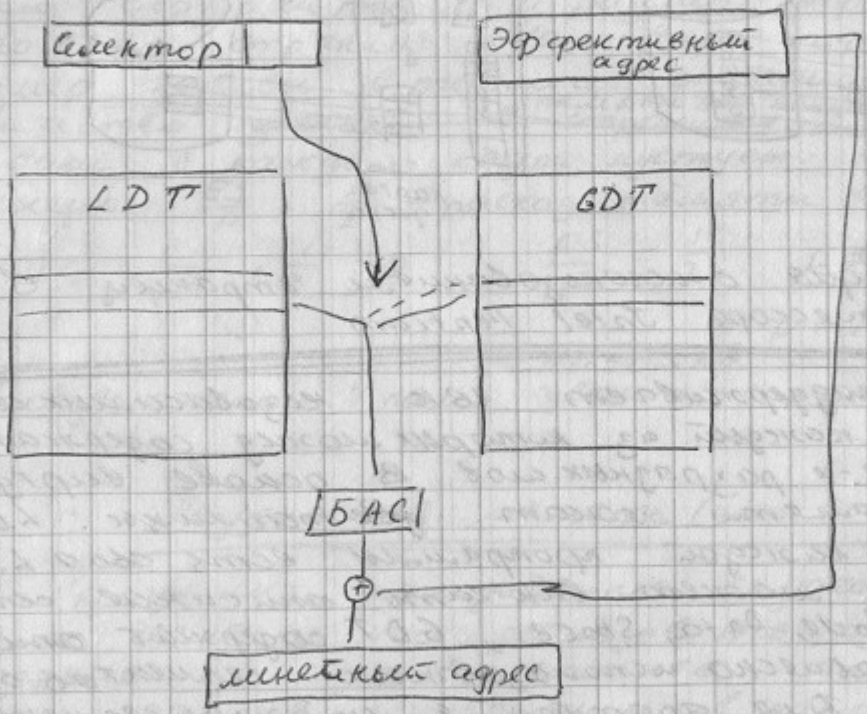
Отображение адресов должно быть осуществлено корректно даже в сложных случаях и обычно реализуется аппаратно. Для ссылки на таблицу процессов используется специальный регистр. При переключении процессов необходимо найти таблицу страниц нового процесса, указатель на которую входит в контекст процесса.

LRT,GDT - прошлогодний конспект



Для поиска GDT и LDT используется регистры GDTR, LDTR, IDTR, TR - содержит указатели на TSS текущей задачи. В TSS сохраняется контекст задачи при смене задач.

Дискриптор находится в GDT или LDT. В соответствии с селектором, а именно индексом требуется вычислить линейный адрес сегмента, который использ для вычисления физич адреса.



TSS имеет динамич структуру и min размер 104 байта - все основные регистры процессора. Может сюда добавить программист свои регистры.

Если стоит признак LDT, то ж сначала надо найти адрес LDT а потом к ней обращаться. LDT можем найти только через GDT.

4) Особенности файловой системы CP\M. В каких ОС есть ее наследие и какое именно.

Ответ:

Прямое указание на кластер, где записан файл – с точки зрения надежности – самая приличная система (впервые реализовано в CP/M)

Стоит взглянуть на операционную систему CP/M по нескольким причинам. Во-первых, исторически это была очень важная система, ставшая прямым предшественником системы MS-DOS. Во-вторых, разработчики сегодняшних операционных систем и систем будущего, полагающие, что компьютеру требуется 32 Мбайт памяти, только чтобы загрузить в нее операционную систему, могут поучиться простоте системы, которой вполне хватало 16 Кбайт ОЗУ. В-третьих, в ближайшие десятилетия широкое применение получат встроенные системы. В связи с ограничениями в цене, размерах, весе и потребляемой мощности операционные системы, используемые, например, в часах, видео- и фотокамерах, радиоприемниках и сотовых телефонах, обязательно должны быть компактными, подобно операционной системе CP/M. Конечно, у этих систем не будет 8-дюймовых гибких дисков, но они вполне могут пользоваться электронными дисками во флэш-ОЗУ, на которых несложно организовать файловую систему, подобную CP/M.

Распределение памяти в системе CP/M показано на рис. 6.27. Наверху оперативной памяти (в ОЗУ) располагается базовая система ввода-вывода BIOS, содержащая базовую библиотеку — 17 вызовов ввода-вывода, используемых системой CP/M (в данном разделе мы рассмотрим CP/M 2.2, являвшуюся стандартной версией, когда CP/M была на вершине популярности). Эти системные вызовы предназначены для чтения и записи с гибкого диска, ввода с клавиатуры и вывода на экран.

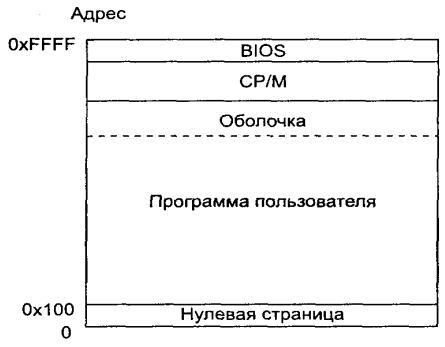


Рис. 6.27. Распределение памяти в системе CP/M

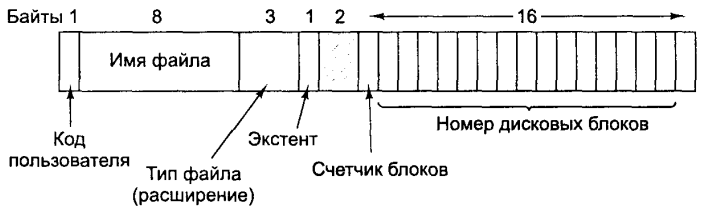


Рис. 6.28. Формат каталоговой записи в системе CP/M

Сразу под BIOS располагается сама операционная система. Для версии CP/M 2.2 ее размер составляет 3584 байт. Невероятно, но факт: вся операционная система занимала менее 4 Кбайт. Под операционной системой размещается оболочка (обработчик командных строк), съедающая еще 2 Кбайт. Остальная память используется для программ пользователя, кроме нижних 256 байт, зарезервированных для векторов аппаратных прерываний, некоторых переменных и буфера для текущей командной строки, в котором к ней могут получить доступ программы пользователя.

Причина, по которой система BIOS отделена от самой операционной системы CP/M (хотя обе системы располагаются в ОЗУ), заключается в переносимости. Операционная система CP/M взаимодействует с аппаратурой только с помощью обращений к BIOS. Для переноса системы CP/M на новую машину нужно всего лишь перенести туда BIOS. Когда это сделано, сама CP/M может быть установлена без изменений.

В файловой системе CP/M всего один каталог, содержащий записи фиксированного размера (32 байт). Размер каталога, фиксированный для данной реализации, может отличаться в других реализациях системы CP/M. В этом каталоге перечисляются все файлы системы. После загрузки система считывает каталог и рассчитывает битовый массив занятых и свободных блоков. Этот битовый массив, размер которого для 180-килобайтного диска составляет всего 23 байта, постоянно хранится в оперативной памяти. После завершения работы операционной системы он не сохраняется на диске. Благодаря такому подходу исчезает необходимость в проверке[1] непротиворечивости файловой системы на диске (вроде той, что выполняет программа *fsck* в UNIX) и сохраняется на диске один блок (в процентном отношении это эквивалентно сохранению 90 Мбайт на современном 16-гигабайтном диске).

Когда пользователь набирает команду, оболочка сначала копирует ее в буфер в нижние 256 байт памяти. Затем она ищет вызываемую программу и загружает ее в память по адресу 256 (над вектором прерываний), после чего передает управление по этому адресу. Программа начинает работу. Она обнаруживает свои параметры в буфере командной строки. Программе разрешается использовать память, занимаемую оболочкой, если ей нужно много памяти. Закончив работу, программа выполняет системный вызов CP/M, сообщая операционной системе, что следует перезагрузить оболочку (если занимаемая ею память использовалась программой) и запустить ее. В двух словах, вот, собственно, и весь рассказ об операционной системе CP/M.

Доп.инфа:

---

5) Значение приоритетных уровней прерываний.

Для реализации механизма прерываний необходима аппаратная схема фиксации сигнала запроса на прерывание. Такая схема обычно содержит регистр, на котором фиксируется наличие сигналов во входных линиях **запросов на прерывания**. Объединенные схемой "ИЛИ", сигналы с разрядов регистра формируют общий сигнал о наличии запроса на прерывание. Затем все разряды регистра опрашиваются в порядке приоритетов входных линий. При этом используются 2 варианта реализации опроса:

1. Полноупорядоченная схема**.** Регистры опрашиваются по порядку приоритетов, от высшего к низшему. Это простая схема, однако при возникновении прерываний с низким приоритетом необходимо проверить *все* регистры запросов на прерывания с более высоким приоритетом.
2. Частично упорядоченная схема**.** Все прерывания делятся на **классы** и вводится 2-уровневая система приоритетов: первый уровень — среди различных классов, второй — внутри каждого класса. При этом, сначала по полноупорядоченной схеме определяется класс прерывания, на которое поступил запрос, а затем, также по полноупорядоченной схеме внутри установленного класса, определяется само прерывание. Это ускоряет поиск прерывания с низким приоритетом, однако усложняет процедуру поиска.

Во время выполнения обработчика одного из прерываний возможно поступление другого сигнала прерывания, поэтому система должна использовать определенную *дисциплину обслуживания заявок* на прерывания. Обычно различным классам либо отдельным прерываниям присваиваются различные *абсолютные приоритеты*, т.о. при поступлении запроса с высшим приоритетом текущий обработчик снимается, а его место занимает обработчик вновь поступившего прерывания.
Количество уровней вложенности прерываний называют **глубиной системы прерываний**. Обработчики прерываний либо дисциплина обслуживания заявок на прерывание должны иметь также специальные средства для случая, когда при обработке прерывания возникает запрос на обработку прерывания от того же источника.