# 4. ПРИНЦИПЫ ОРГАНИЗАЦИИ ПОДСИСТЕМЫ ПАМЯТИ ЭВМ

## 4.1. Иерархическая структура памяти ЭВМ

Памятью ЭВМ называется совокупность устройств, служащих для запоминания, хранения и выдачи информации.

Основными характеристиками отдельных устройств памяти (запоминающих устройств) являются емкость памяти, быстродействие и стоимость хранения единицы информации (бита).

Емкость памяти определяется максимальным количеством данных, которые могут в ней храниться. Емкость измеряется в двоичных единицах (битах), машинных словах, но большей частью в байтах (1 байт = 8 бит). Часто емкость памяти выражают через число К = 1024, например, Кбит — килобит, Кбайт — килобайт, 1024 Кбайт = 1 Мбайт (Мегабайт), 1024 Мбайт = 1 Гбайт (гигабайт), 1024 Гбайт = 1 Тбайт (терабайт).

Быстродействие (задержка) памяти определяется временем доступа и длительностью цикла памяти. Время доступа представляет собой промежуток времени между выдачей запроса на чтение и моментом поступления запрошенного слова из памяти. Длительность цикла памяти определяется минимальным временем между двумя последовательными обращениями к памяти.

Требования к увеличению емкости и быстродействия памяти, а также к снижению ее стоимости являются **противоречивыми.** Чем больше быстродействие, тем технически труднее достигается и дороже обходится увеличение емкости памяти. Стоимость памяти составляет значительную часть общей стоимости ЭВМ. Исходя из этого, память ЭВМ организуется в виде иерархической структуры (рис. 4.1) запоминающих устройств, обладающих различным быстродействием и емкостью. Чем выше уровень, тем выше быстродействие соответствующей памяти, но меньше её емкость. К верхнему (сверхоперативному) уровню относятся регистры операционных и управляющих блоков процессора, сверхоперативная память, управляющая память, буферная память. На втором уровне находится основная или оперативная память. На последующих уровнях размещается внешняя и архивная память. Система управления памятью обеспечивает обмен информационными блоками между уровнями, причем обычно первое обращение к блоку информации вызывает его перемещение с низкого медленного уровня на более высокий. Это позволяет при последующих обращениях к данному блоку осуществлять его выборку с более быстродействующего уровня памяти.

Сравнительно небольшая емкость оперативной памяти (8 - 64 Мбайта) компенсируется практически неограниченной емкостью внешних запоминающих устройств. Однако эти устройства сравнительно медленные — время обращения за данными для магнитных дисков составляет десятки микросекунд. Для сравнения: цикл обращения к оперативной памяти (ОП) составляет 50 нс. Исходя из этого, вычислительный процесс должен протекать с возможно меньшим числом обращений к внешней памяти.

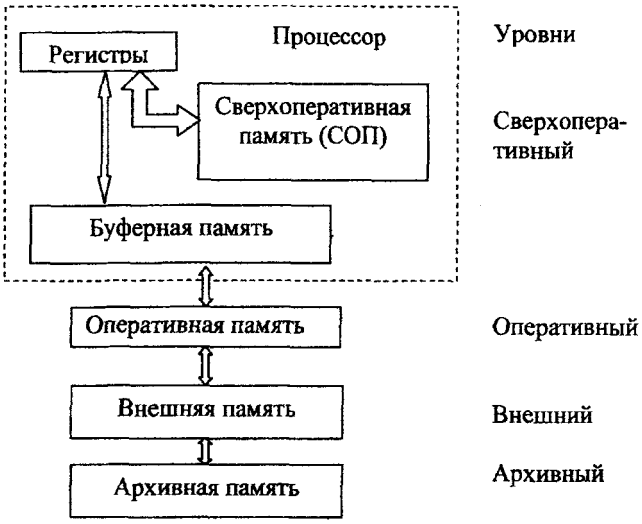



Рис.4.1. Иерархическая структура памяти

Непрерывный рост производительности ЭВМ проявляется, в первую очередь, в повышении скорости работы процессора. Быстродействие ОП также растет, но все время отстает от быстродействия аппаратных средств процессора в значительной степени потому, что одновременно происходит опережающий рост её емкости, что делает более трудным уменьшение времени цикла работы памяти. Вследствие этого быстродействие ОП часто оказывается недостаточным для обеспечения требуемой производительности ЭВМ. Это проявляется в несоответствии пропускных способностей процессора и ОП. Возникающая проблема выравнивания их пропускных способностей решается путем использования сверхоперативной буферной памяти небольшой емкости и повышенного быстродействия, хранящей команды и данные, относящиеся к обрабатываемому участку программы.

При обращении к блоку данных, находящемуся на оперативном уровне, его копия пересылается в сверхоперативную буферную память (СБП). Последующие обращения производятся к копии блока данных, находящейся в СБП. Поскольку время выборки из сверхоперативной буферной памяти (5 нс) намного меньше времени выборки из оперативной памяти ton, введение в структуру памяти СБП приводит к уменьшению эквивалентного времени обращения по сравнению с ton:

$$t_э = t_{сбп} + \alpha t_{оп},$$

где a = (1 — q) и q — вероятность нахождения блока в СВП в момент обращения к нему, т.е. вероятность «попадания». Очевидно, что при высокой вероятности попадания эквивалентное время обращения приближается к времени обращения к СБП.

В основе такой организации взаимодействия ОП и СБП лежит принцип локальности обращений, согласно которому при выполнении какой-либо программы (практически для всех классов задач) большая часть обращений в пределах некоторого интервала времени приходится на ограниченную область адресного пространства ОП, причем обращения к командам и элементам данных этой области производятся многократно. Это позволяет копии наиболее часто используемых участков программ и некоторых данных загрузить в СБП и таким образом обеспечить высокую вероятность попадания q.

Высокая эффективность применения СБП достигается при $q \geq 0{,}9$.

Буферная память не является программно доступной. Это значит, что она влияет только на производительность ЭВМ, но не должна оказывать влияния на программирование прикладных задач. Поэтому она получила название кэш-памяти (в переводе с английского - тайник). В структуре одних ЭВМ используется объединенная кэш-память команд и данных, в других ЭВМ — раздельные кэш-памяти для команд и для данных. Кэш-память, входящую в состав процессора, называют кэш-памятью первого уровня. В современных компьютерах применяют кэш-память второго уровня, которая находится между процессором и ОП и еще больше повышает производительность ЭВМ.

## 5.1.Основная память

### 5.1.1.Состав, устройство и принцип действия основной памяти

Основная память включает два типа устройств: оперативное запоминающее устройство (ОЗУ или RAM — Random Access Memory) и постоянное запоминающее устройство (ПЗУ или ROM — Read Only-Memory).

ОЗУ предназначено для хранения переменной информации. Оно допускает изменение своего содержимого в ходе выполнения процессором вычислительных операций с данными и может работать в режимах записи, чтения и хранения.

ПЗУ содержит информацию, которая не должна изменяться в ходе выполнения процессором вычислительных операций, например стандартные программы и константы. Эта информация заносится в ПЗУ перед установкой микросхемы в ЭВМ. Основными операциями, которые может выполнять ПЗУ, являются чтение и хранение.

Функциональные возможности ОЗУ шире, чем ПЗУ, но ПЗУ сохраняет информацию при отключении питания (т.е. является энергонезависимой памятью) и может иметь более высокое быстродействие, так как ограниченность функциональных возможностей ПЗУ и его специализация на чтении и хранении позволяют сократить время выполнения реализуемых им операций считывания.

В современных ЭВМ микросхемы памяти (ОП и СОЗУ) изготавливают из кремния по полупроводниковой технологии с высокой степенью интеграции элементов на кристалле (микросхемы памяти относятся к так называемым «регулярным» схемам, что позволяет сделать установку элементов памяти в кристалле (чипе) настолько плотной, что размеры элементов памяти становятся сопоставимыми с размерами отдельных атомов).

Основной составной частью микросхемы является массив элементов памяти (ЭП), объединенных в матрицу накопителя.

Каждый элемент памяти может хранить 1 бит информации и имеет свой адрес. ЗУ, позволяющие обращаться по адресу к любому ЭП в произвольном порядке, называются ***запоминающими устройствами с произвольным доступом.***

При матричной организации памяти реализуется координатный принцип адресации ЭП, в связи с чем адрес делится на две части (две координаты) — X и Y. На пересечении этих координат находится элемент памяти, чья информация должна быть прочитана или изменена.
ОЗУ связано с остальным микропроцессорным комплектом ЭВМ через системную магистраль (рис. 5.1).

По шине управления передается сигнал, определяющий, какую операцию необходимо выполнить.

По шине данных передается информация, записываемая в память или считываемая из нее.

По шине адреса передается адрес участвующих в обмене элементов памяти (поскольку данные передаются машинными словами, а один ЭП может воспринять только один бит информации, блок элементов памяти состоит из *n* матриц ЭП, где *n* — количество разрядов в машинном слове).

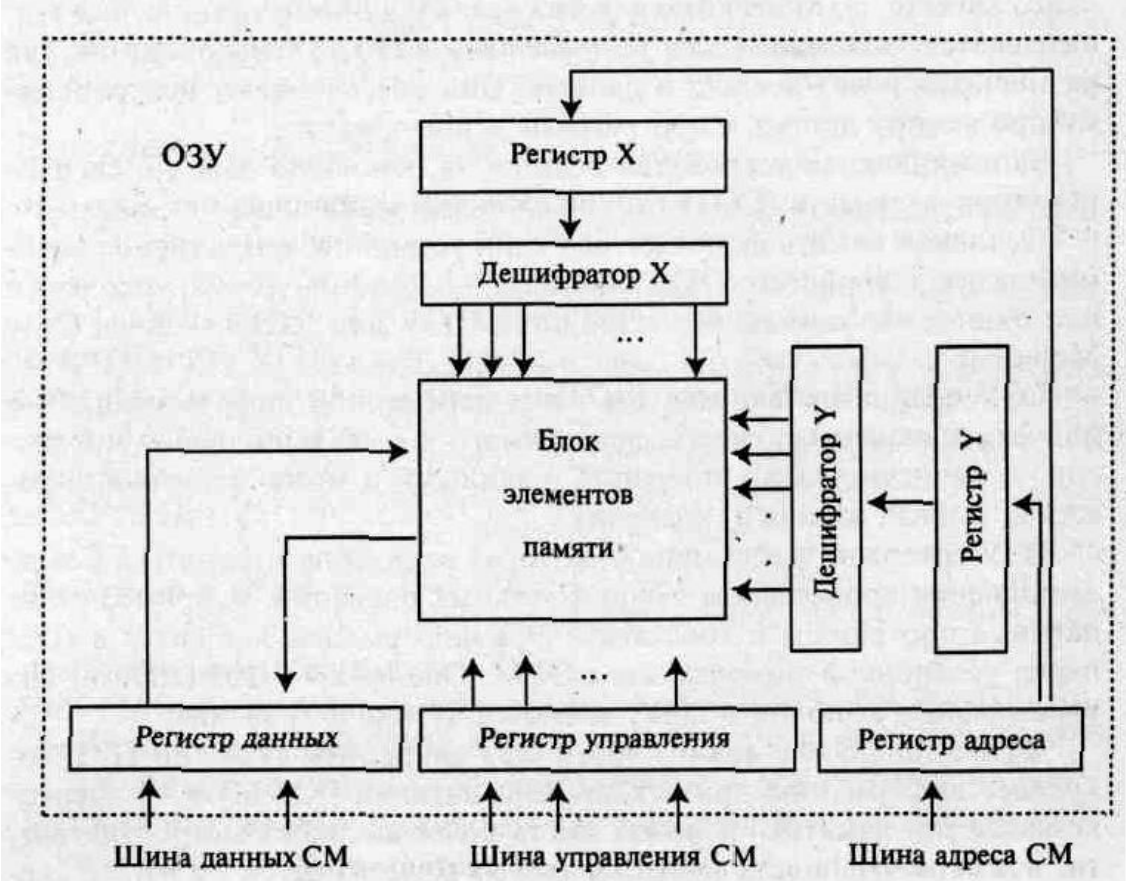



**Рис. 5.1.** Структурная схема ОЗУ

Максимальная емкость памяти определяется количеством линий в шине адреса системной магистрали: если количество линий обозначить через *m,* то емкость памяти (т.е. количество элементов памяти, имеющих уникальные адреса) определяется как $2^m$. Так, в IBM PC XT шина адреса СМ содержит 20 линий. Поэтому максимальный объем ОП в этих машинах равен $2^{20}$ = 1 Мбайт. В IBM PC AT (с микропроцессором i80286) СМ содержит 24 линии, поэтому объем ОП может быть увеличен до 16 Мбайт. Начиная с МП i80386, шина адреса содержит 32 линии. Максимальный объем ОП увеличился до $2^{32}$ = 4 Гбайта.

Микросхемы памяти могут строиться на статических (SRAM) и динамических (DRAM) ЭП. В качестве статического ЭП чаще всего выступает статический триггер. В качестве динамического ЭП может использоваться электрический конденсатор, сформированный внутри кремниевого кристалла.

Статические ЭП способны сохранять свое состояние (0 или 1) неограниченно долго (при включенном питании). Динамические ЭП с течением времени записанную в них информацию теряют (например, из-за саморазряда конденсатора), поэтому они нуждаются в периодическом восстановлении записанной в них информации — в регенерации.

Микросхемы элементов памяти динамических ОЗУ отличаются от аналогичных ЭП статических ОЗУ меньшим числом компонентов в одном элементе памяти, в связи с чем имеют меньшие размеры и могут быть более плотно упакованы в кристалле. Однако из-за необходимости регенерации информации динамические ОЗУ имеют более сложные схемы управления.

Основными характеристиками ОЗУ являются объем и быстродействие.

В современных ПЭВМ ОЗУ имеет модульную структуру. Сменные модули могут иметь различное конструктивное исполнение (SIP, ZIP, SIMM, DIMM). Увеличение объема ОЗУ обычно связано с установкой дополнительных модулей, которые выпускаются в 30-контактном (ЗО-pin) и 72-контактном исполнениях на 1, 4, 8, 16, 32 и 64 Мбайта. Время доступа к модулям DRAM составляет 60 — 70 нс.

На производительность ЭВМ влияют не только время доступа, но и такие параметры (связанные с ОЗУ), как тактовая частота и разрядность шины данных системной магистрали. Если тактовая частота недостаточно высока, то ОЗУ простаивает в ожидании обращения. При тактовой частоте, превышающей возможности ОЗУ, в ожидании будет находиться системная магистраль, через которую поступил запрос в ОЗУ.

Разрядность шины данных (8, 16, 32 или 64 бита) определяет длину информационной единицы, которой можно обменяться с ОЗУ за одно обращение. Интегральной характеристикой производительности ОЗУ с учетом частоты и разрядности является ***пропускная способность,*** которая измеряется в мегабайтах в секунду. Для ОП с временем доступа 60-70 нс и разрядностью шины данных 64 бита максимальная (теоретическая) пропускная способность при тактовой частоте СМ 50 МГц составляет 400 Мбайт/с, при частоте 60 МГц — 480 Мбайт/с, при 66 МГц — 528 Мбайт/с в режиме группового обмена, реализуемом, например, при прямом доступе к памяти. Для группового обмена характерно (и это является еще одной характеристикой ОЗУ), что при каждом обращении к памяти для считывания первого слова необходимо больше времени, чем для последующих. Так, при использовании стандартной динамической памяти FPM (Fust Page Mode) DRAM на 60 — 70 не каждое обращение к памяти в групповом режиме описывается формулой 7-3-3-3, т.е. для обработки первого слова необходимо 7 тактов (в течение 6 из которых СМ простаивает в ожидании), а для обработки следующих трех слов - по 3 такта, по 2 из которых СМ простаивает.

Память типа EDO (Extended Data Output) DRAM позволяет уменьшить количество циклов ожидания (x-2-2-2, где x — количество тактов, необходимое для обработки первого слова). Память типа BEDO (Burst EDO) DRAM обеспечивает обмен по формуле x-1-1-1 для первого обращения и 1-1-1-1 — для последующих. Приведенные формулы характерны для тактовых частот до 60 МГц. Синхронная динамическая память (SDRAM — Synchronous DRAM) способна обмениваться блоками данных на рабочей тактовой частоте (внешняя частота процессора) без циклов ожидания: при времени доступа 10 нс — до 100 МГц, 12 нс — до 83 МГц и 15 нс — до 66 МГц.

Микросхемы ПЗУ также построены по принципу матричной структуры накопителя. Функции элементов памяти в них выполняют перемычки в виде проводников, полупроводниковых диодов или транзисторов. В такой матрице наличие перемычки может означать «1», а ее отсутствие — «О». Занесение информации в микросхему ПЗУ называется ее ***программированием,*** а устройство, с помощью которого заносится информация, — ***программатором.*** Программирование ПЗУ заключается в устранении (прожигании) перемычек по тем адресам, где должен храниться «О». Обычно схемы ПЗУ допускают только одно программирование, но специальные микросхемы — репрограммируемые ПЗУ (РПЗУ) — допускают их многократное стирание и занесение новой информации. Этот вид микросхем также относится к энергонезависимым, т.е. может длительное время сохранять информацию при выключенном питании (стирание микросхемы происходит либо за счет подачи специального стирающего напряжения, либо за счет воздействия на кристалл ультрафиолетового излучения, для этого в корпусе микросхемы оставляется прозрачное окно).

Сверхоперативные ЗУ используются для хранения небольших объемов информации и имеют значительно меньшее время (в 2 — 10раз) считывания/записи, чем основная память. СОЗУ обычно строятся на регистрах и регистровых структурах.

Стековая память получила широкое распространение. Для ее реализации в ЭВМ разработаны специальные микросхемы. Но часто работа стековой памяти ***эмулируется*** в основной памяти ЭВМ: с помощью программ операционной системы выделяется часть памяти под стек (в IBM PC для этой цели выделяется 64 Кбайта). Специальный регистр микропроцессора (указатель стека) постоянно хранит адрес ячейки ОП, выполняющей функции вершины стека. Чтение числа всегда производится из вершины стека, после чего указатель стека изменяется и указывает на очередную ячейку стековой памяти (т.е. фактически стек остается неподвижным, а перемещается вершина стека). При записи числа в стек сначала номер ячейки в указателе стека модифицируется так, чтобы он указывал на очередную свободную ячейку, после чего производится запись числа по этому адресу. Такая работа указателя стека позволяет реализовать принцип «первым вошел — последним вышел».

Кэш-память может быть размещена в кристалле процессора (так называемая «кэш-память I уровня») или выполнена в виде отдельной микросхемы (внешняя кэш-память, или кэш-память II уровня). Встроенная кэш-память (I уровня) в процессорах Pentium имеет объем около 16 Кбайт, время доступа 5 - 10 не, работает с 32-битовыми словами и при частотах 75 - 166 МГц обеспечивает пропускную способность от 300 до 667 Мбайт/с. Внешняя кэш-память (II уровня) имеет объем 256 Кбайт - 1 Мбайт, время доступа 15 не, работает с 64-битовыми словами и при частоте 66 МГц обеспечивает максимальную пропускную способность 528 Мбайт/с. Конструктивно исполняется либо в виде 28-контактной микросхемы, либо в виде модуля расширения на 256 или 512 Кбайт.

### 5.1.2. Размещение информации в основной памяти IBM PC

Адресуемой единицей информации основной памяти IBM PC является байт. Это означает, что каждый байт, записанный в ОП, имеет уникальный номер (адрес). При использовании 20-битовой шины адреса абсолютный (физический) адрес каждого байта является пятиразрядным шестнадцатеричным числом, принимающим значения от 00000 до FFFFF. В младших адресах располагаются блоки операционной системы (векторы прерываний, зарезервированная область памяти BIOS), в этой же части могут размещаться драйверы устройств, дополнительные обработчики прерываний DOS и BIOS, командный процессор операционной системы. Затем располагается область памяти, отведенная пользователю. Область памяти пользователя заканчивается адресом 9FFFF. Этот адрес является физической границей оперативного ЗУ, последним адресом 640-Кбайтовой основной памяти. Остальное адресное пространство (128 Кбайт с адреса AOOOO по BFFFF) отведено под видеопамять, которая физически размещается не в ОП, а в адаптере дисплея. После видеопамяти расположено адресное пространство (256 Кбайт) постоянного запоминающего устройства (ПЗУ), хранящего программы базовой системы ввода-вывода (BIOS — Basic Input-Output System). Эта часть ОП еще называется ROM-BIOS. Из отведенных 256 Кбайт непосредственно ПЗУ занимает 64 Кбайта, а остальные 192 Кбайта оставлены для расширения ПЗУ. Поскольку большая часть оставленной'для расширения BIOS части адресного пространства не используется, в этих адресах часто располагается информация, необходимая для работы сетевых карт, графических расширителей и др. Запись в ОП (и чтение из нее) может осуществляться не только байтами, но и машинными словами. При этом машинное слово при размещении в памяти занимает несколько смежных байтов. Каждый байт ОП имеет свой адрес. Но машинное слово характеризуется не всеми адресами занятых байтов, а только одним - адресом младшего байта слова. Обычно графически машинное слово изображается так, что младший байт находится справа (рис. 5.3).


**Рис. 5.3.** Стереотипное представление машинного слова

При записи слова младший байт размещается по адресу, который является адресом машинного слова, старший байт машинного слова размещается в следующем по порядку байте ОП, имеющем номер, увеличенный на 1 (здесь действует мнемоническое правило «младший байт — по младшему адресу»).


**Рис. 5.4.** «Вращение» байтов при чтении машинного слова из ОП

При чтении из ОП двух следующих подряд байтов машинного слова их принято размещать слева направо: сначала первый из прочитанных байтов (с меньшим адресом), а затем — следующий. В результате происходит «вращение» байтов (рис. 5.4), которое психологически трудно воспринимается. Необходимо помнить, что при записи отдельных байтов каждый байт располагается в ОП по своему адресу, при чтении никакого вращения не происходит. При записи же в ОП единиц информации, имеющих в своем составе больше одного байта, адресом информационной единицы является адрес самого младшего байта, запись в ОП ведется побайтно, начиная с самого младшего байта, каждый последующий байт располагается в ячейке, адрес которой на 1 больше предыдущего. Иными словами, запись машинного или двойного слова производится справа налево, тогда как при чтении считанные байты обычно располагаются слева направо — происходят «вращение» байтов, перестановка их местами, что необходимо учитывать при работе с ОП на физическом уровне.

### 5.1.3. Расширение основной памяти IBM PC

Рабочая концепция фирмы IBM при создании IBM PC содержала гипотезу, что объем основной памяти ЭВМ, предназначенной для персонального использования в любой предметной области, не должен превышать 640 Кбайт. Поэтому в базовую модель IBM PC заложили 20-разрядную шину адреса системной магистрали. Наличие 20 линий в шине адреса позволяло адресовать память большего объема, чем было предусмотрено концепцией ($2^{20}$ = 1 Мбайт). «Излишек» адресного пространства в 384 Кбайта был поделен между видеопамятью (128 Кбайт) и ПЗУ (256 Кбайт).

Физически увеличить объем памяти несложно, для этого необходимо только подключить к системной магистрали дополнительные модули. Такая возможность в IBM PC была предусмотрена. Но каждый байт дополнительной памяти должен иметь уникальный адрес, а адресного пространства для дополнительной памяти нет.

Существует несколько способов разрешения таких конфликтов. Один из них — банкирование памяти: вся память делится на блоки (банки), емкость которых не выходит за пределы допустимого адресного пространства; во время работы специальными командами можно переключать банки, делая активным любой из них или осуществляя групповую перепись информации из одного банка в другой.

В IBM PC XT фирма IBM применила другой способ: 256 Кбайт было сначала оставлено для ПЗУ, в котором размещалась базовая система ввода-вывода (BIOS). Анализ программ BIOS показал, что в оставленном для ПЗУ адресном пространстве (UMB — Upper Memory Block) имеются «окна» — неиспользуемые участки. Четыре таких участка (paqe frames), no 16 Кбайт каждый, были выделены, и их адреса стали использоваться для адресации дополнительной памяти, подключенной к системной магистрали. Таким образом, общий объем ОП удалось увеличить на 64 Кбайта. Специальная программа (драйвер дополнительной памяти) «перехватывала» обращение к «окнам» ПЗУ и вместо них «подставляла» дополнительный модуль памяти (Expended Memory).

Дополнительная память не обязательно должна была иметь объем 64 Кбайта. Ее объем мог быть и большим (фирма IBM выпускала модули допблнительной памяти объемом 8 и 32 Мбайта). При этом драйвер дополнительной памяти делил ее на блоки по 16 Кбайт и «отображал» каждое окно UMB на один из блоков Expended Memory. Из-за этого память такого вида получила название ***отображаемой.***

Но развитие персональных ЭВМ привело к необходимости более, серьезной корректировки рабочей концепции. Поэтому в IBM AT с микропроцессором i80286 разрядность шины адреса увеличили до 24, что позволило увеличить ее объем до 16 Мбайт. В МП i80386 разрядность шины адреса и адресных регистров микропроцессора увеличена до 32, в результате чего допустимый объем ОП увеличился до 4 Гбайт.

Наряду с этим изменился принцип формирования абсолютного адреса ОП, в результате чего утрачена совместимость с программным обеспечением, разработанным для IBM PC XT.

Для того чтобы обеспечить совместимость AT с XT, было решено реализовать два режима работы микропроцессоров, имеющих номер, больший 80286: реальный и защищенный.

В реальном режиме дополнительные разряды шины адреса заблокированы, что обеспечивает совместимость с микропроцессором i8086 и позволяет использовать операционную систему MS DOS и программное обеспечение, разработанное для XT. Но при этом остается неиспользованной вся дополнительная память, находящаяся за пределами 1 Мбайта.

В защищенном режиме применяется другой принцип формирования абсолютного адреса ОП, благодаря чему возможно использование всей имеющейся в наличии дополнительной (расширенной) памяти, но возникают трудности с использованием программного обеспечения, разработанным для MS DOS.

В IBM PC XT 20-битовый адрес формировался из двух машинных слов: базового адреса сегмента (16 бит) и смещения (16 бит). Это было связано с тем, что вся ОП делилась на сегменты емкостью 64 Кбайта. Адресация байтов внутри сегмента начиналась с 0 и заканчивалась адресом FFFF.

Внутрисегментный адрес байта называется *смещением (т.е.* смещением относительно начала сегмента). Начало же сегмента (т.е. его базовый 20-битовый адрес) однозначно определялось 16-битовым адресом, который преобразовывался в 20-битовый адрес дописыванием справа четырех нулей. В машинных командах абсолютный (физический) адрес задавался либо прямым указанием базового адреса сегмента и смещения (которые разделялись двоеточием, например OA12:F4B2, где OA12 — 20-битовый адрес начала сегмента; F4B2 — 16-битовое смещение внутри сегмента), либо по умолчанию (базовые адреса сегментов программы, данных, стека запоминаются в специальных регистрах микропроцессора), либо указанием регистра, в котором содержится необходимый базовый адрес (например, если регистр называется CS, то абсолютный адрес в машинной команде может быть задан в виде CS:F4B2).

Начиная с МП i80386, благодаря увеличению длины всех регистров для смещений до 32 бит, реализована возможность работы «с плоской памятью», не разделяемой на сегменты. Это допускает адресацию $2^{32}$ байт или 4 Гбайта ОП. Кроме того, в защищенном режиме (начиная с МП i80286) можно использовать и сегментированную память, но сегментные регистры не суммируются со смещением, а предназначены в качестве указателя на управляющие таблицы, содержащие необходимую информацию о сегментах.

Желание использовать в реальном режиме всю фактически имеющуюся в наличии дополнительную память привело к созданию двух виртуальных режимов, один из которых — стандарт EMS (Expended Memory Specifications), реализующий принцип банкирования дополнительной памяти.

Вся дополнительная память делится на страницы (банки) емкостью по 16 Кбайт; выбираются четыре страницы и объявляются активными. Выбранные активные страницы отображаются на четыре окна UMB, теперь при обращении к одному из окон UMB вместо него подставляется отображенная на него страница дополнительной памяти.

Поскольку любое окно UMB можно отобразить на любую страницу дополнительной памяти (объявив ее активной), то, изменяя отображение в процессе работы, можно использовать всю дополнительную память любого объема.

Стандарт EMS реализуется программным путем — с помощью драйвера дополнительной памяти, который «перехватывает» каждое

обращение к окну, имеющемуся в адресном пространстве ПЗУ, и «подставляет» вместо ПЗУ соответствующий участок дополнительной памяти.

В соответствии с этим стандартом работают драйверы XMA2EMS.SYS, EMM386.SYS и др.

Стандарт EMS несколько снижает производительность системы, но не накладывает никаких ограничений на размещение в дополнительной памяти программ и данных.

Другой виртуальный режим основан на том, что за счет разблокирования на время дополнительных (по сравнению с XT) линий шины адреса системной магистрали удается увеличить доступное MS DOS адресное пространство еще почти на 64 Кбайта, начиная с адреса FFFFF (т.е. за пределами адресного пространства 1 Мбайт). Эта область адресного пространства (64 Кбайта, начиная с 1 Мбайта) получила название НМЛ (High Memory Area) — старшая область памяти. Ее также можно использовать, работая в MS DOS, для хранения и программ, и данных.

Блоки памяти, расположенные выше границы НМЛ, называются EMB (Extended Memory Blocks) — расширенные блоки памяти, хотя часто расширенной памятью (EM — Extended Memory) называют всю дополнительную память, расположенную в адресном пространстве выше 1 Мбайта, иногда выделяя в ней область НМЛ.

Кратковременное разблокирование дополнительных линий шины адреса системной магистрали позволяет реализовать стандарт XMS(extended Memory Specification), при котором разделенная на страницы EM отображается на НМЛ, но в этом стандарте программные модули могут располагаться только в НМА, а остальная память может использоваться лишь для хранения данных. Стандарт XMS реализуется драйвером HIMEM.SYS, который способен работать с шиной адреса, имеющей до 32 линий.

## 4.4. Особенности управления основной памятью ЭВМ

### 4.4.1. Отображение адресного пространства программы на основную память

Алгоритмы распределения, использования, освобождения ресурсов и представления к ним доступа предназначены для наиболее эффективной организации работы всего комплекса устройств ЭВМ. Рассмотрим их на примере управления основной памятью.

Для выполнения программы при ее загрузке в основную память ей выделяется часть ***машинных ресурсов*** — они необходимы для размещения команд, данных, управляющих таблиц и областей ввода-вывода, т.е. производится трансляция адресного пространства откомпилированной программы в местоположение в реальной памяти.

Выделение ресурсов может быть осуществлено самим программистом (особенно, если он работает на языке, близком машинному), но может производиться и операционной системой.

Если выделение ресурсов производится перед выполнением программы, такой процесс называется ***статическим перемещением,*** **в** результате которого программа «привязывается» к определенному месту в

памяти вычислительной машины. Если же ресурсы выделяются в процессе выполнения программы, это называется ***динамическим перемещением,*** и в этом случае программа не привязана к определенному месту в реальной памяти. Динамический режим можно реализовать только с помощью операционной системы.

При статическом перемещении могут встретиться два случая:

- реальная память больше требуемого адресного пространства программы. В этом случае загрузка программы в реальную память производится, начиная с 0-го адреса (рис. 4.2).

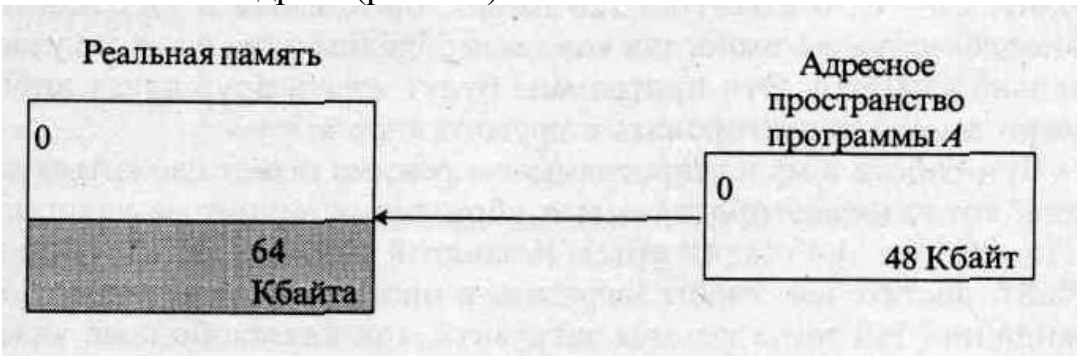



**Рис. 4.2.** Загрузка программы в избыточную реальную память

Загружаемая программа А является ***абсолютной программой,*** так как никакого изменения адресов в адресном пространстве, подготовленном компилятором, при загрузке в основную память не происходит — программа располагается с 0-го адреса реальной памяти;

- реальная память меньше требуемого адресного пространства программы (рис. 4.3). В этом случае программист (или операционная система) вынужден решать проблему, как организовать выполнение программы. Методов решения проблемы существует несколько: можно создать ***оверлейную структуру*** (т.е. разбить программу на части, вызываемые в ОП по мере необходимости), сделать модули программы ***реентерабельными*** (т.е. допускающими одновременную работу модуля по нескольким обращениям из разных частей программы или из различных программ) и т. д.

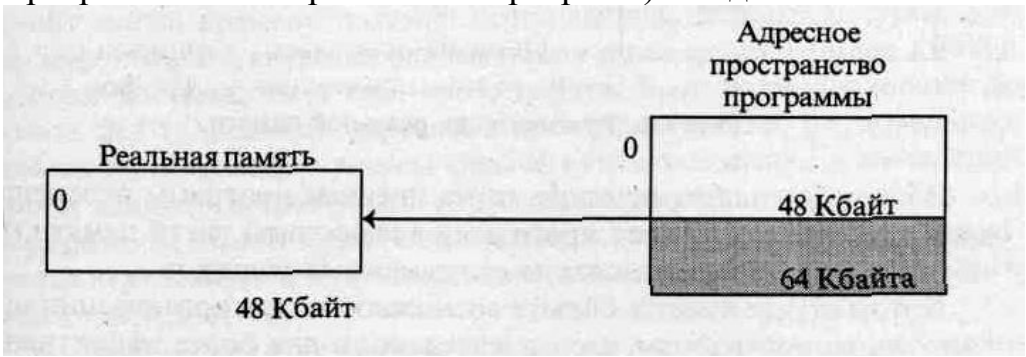



**Рис. 4.3.** Загрузка программы в реальную память при недостатке памяти

В некоторых операционных системах адреса откомпилированной (с 0-го адреса) программы могут быть преобразованы в адреса реальной памяти,

отличные от 0. При этом создается ***абсолютный модуль,*** который требует размещения его в памяти всегда с одного и того же адреса.

При мультипрограммном режиме, если имеем программы A, B и C, для которых известно, что программа A выполняется при размещении в памяти с адреса 60 Кбайт до 90 Кбайт, B — с 60 Кбайт до 90 Кбайт, C — с 50 Кбайт до 120 Кбайт, организовать их совместное выполнение невозможно, так как им необходим один и тот же участок реальной памяти. Эти программы будут ждать друг друга либо их нужно заново редактировать с другого адреса.

При работе в мультипрограммном режиме может сложиться ситуация, когда между программами образуются незанятые участки памяти. На рис. 4.4 общий объем незанятой памяти, составляющий 50 Кбайт, достаточен, чтобы загрузить и программу D, находящуюся в ожидании. Но ее не удается загрузить, так как свободные участки памяти не являются смежными. Такое состояние называется ***фрагментацией*** реальной памяти. Оно характерно для систем со статическим перемещением.

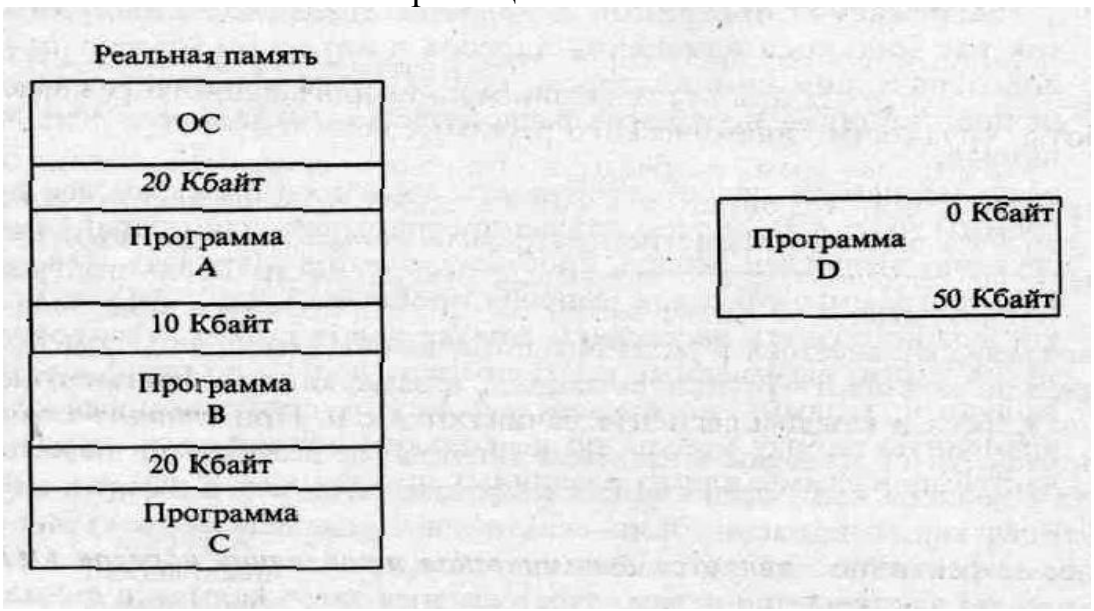



**Рис. 4.4.** Фрагментация реальной памяти

В системах с динамическим перемещением программ перемещающий загрузчик размещает программу в свободной части памяти (рис. 4.5) и допускает использование ее несмежных участков.

В этом случае имеется больше возможностей для организации мультипрограммной работы, а следовательно, и для более эффективного использования временных ресурсов ЭВМ.

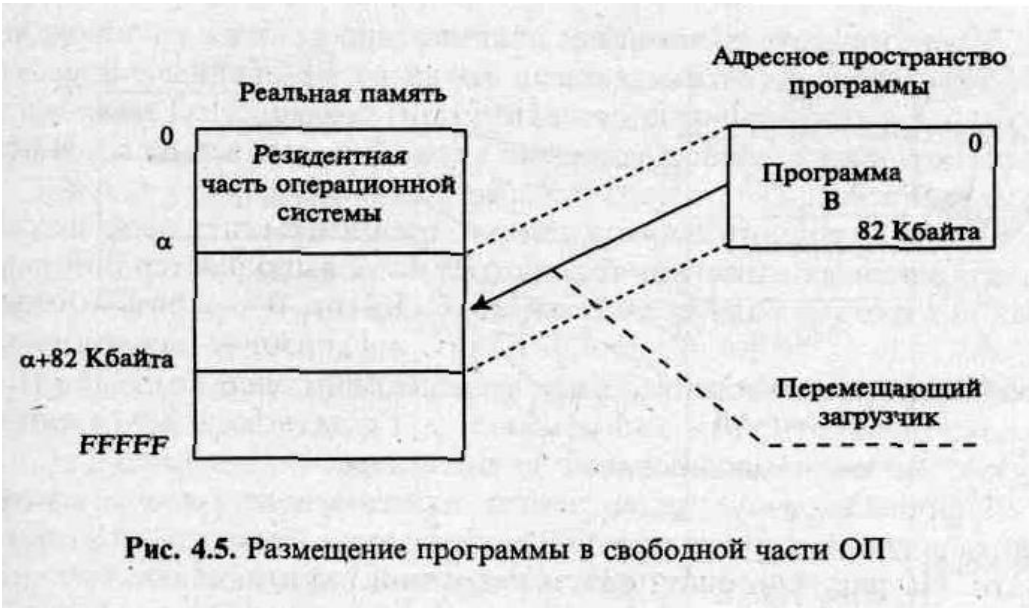



**Рис. 4.5.** Размещение программы в свободной части ОП

### 4.4.2. Адресная структура команд микропроцессора и планирование ресурсов

При больших размерах реализуемых программ возникают некоторые противоречия при организации мультипрограммного режима работы, трудности динамического распределения ресурсов.

В настоящее время разработано несколько способов решения этих противоречий. Например, для борьбы с фрагментацией основной памяти адресное пространство программы может быть разбито на отдельные ***сегменты,*** слабо связанные между собой. Тогда программа D общей длиной 50 Кбайт может быть представлена в виде ряда сегментов, загружаемых в различные области ОП (рис. 4.6). Это позволяет использовать реальную память, теряемую из-за фрагментации.

Адреса в каждом сегменте начинаются с 0. При ***статическом перемещении*** программы в процессе загрузки ее в основную память адреса должны быть привязаны к конкретному месту в памяти, на что уходит много времени и отвлекаются вычислительные ресурсы. Более эффективной является ***динамическая трансляция адресов (ДТА),*** которая заключается в том, что сегменты загружаются в основную память без трансляции адресного пространства (т.е. без изменения адресов в программе с учетом физического размещения в памяти команд и данных), а трансляция адресов каждой команды производится в процессе ее выполнения. Этот тип трансляции называется ***динамическим перемещением*** и осуществляется специальными аппаратными средствами ДТА.

Каждый сегмент программы должен иметь свое имя. Форма имени сегмента может быть любой, например номер (рис. 4.7, *а,б).*

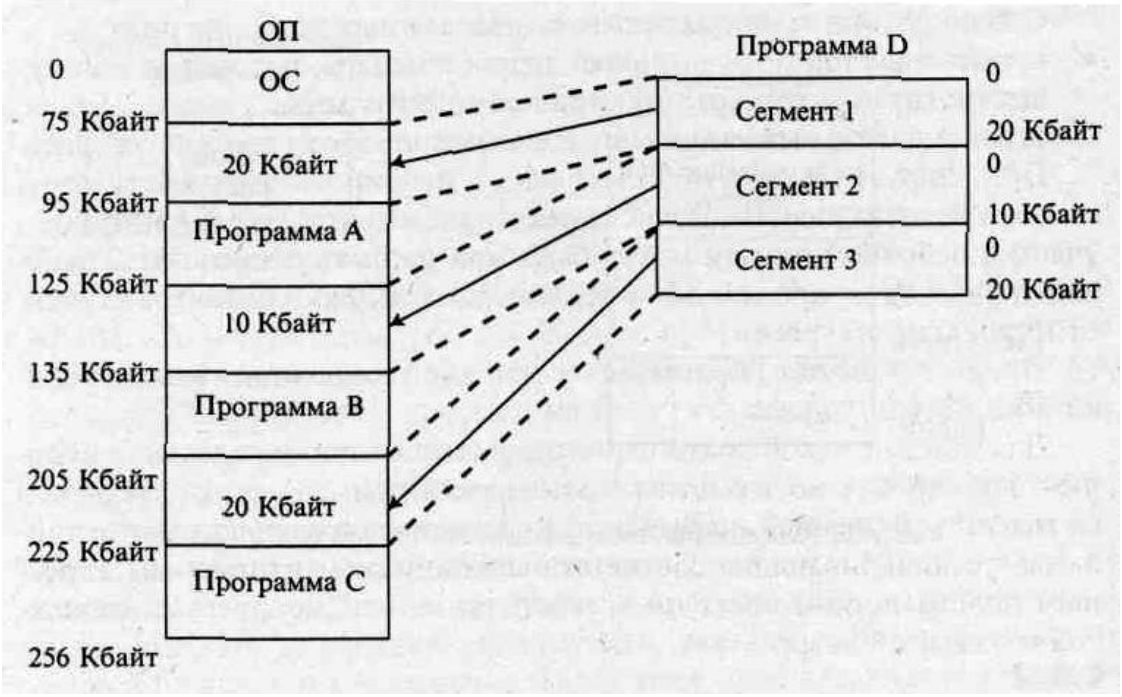



**Рис. 4.6.** Фрагментация ОП. Загрузка сегментированной программы

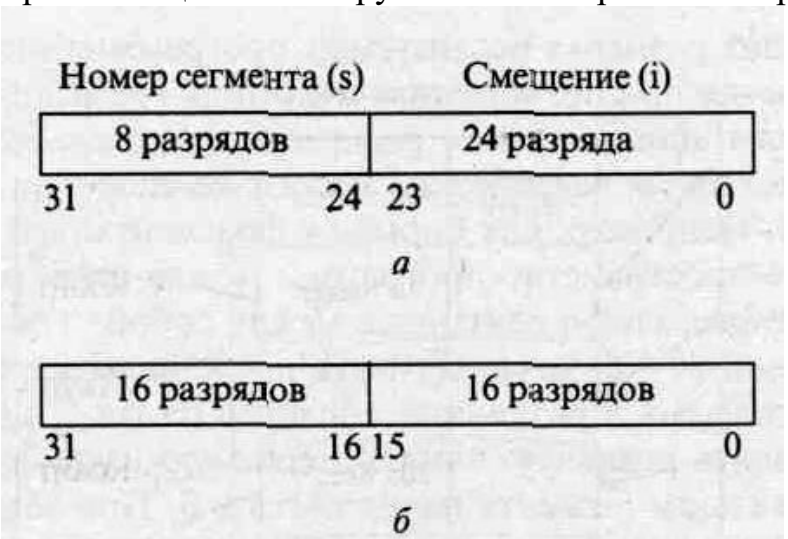



**Рис.** 4.7. Форма имени сегмента: *а* - при выделении номеру сегмента 8 разрядов; б - при выделении номеру сегмента 16 разрядов

При таком представлении адрес будет состоять из двух частей: *s, i,* где *s* — имя сегмента, / — адрес внутри сегмента.

Если ЭВМ имеет 32-битовую адресную структуру, максимальная длина адреса в единственном сегменте будет 32 разряда. Если 16 разрядов из 32 отвести под номер сегмента (а 16 — под смещение), то в этом случае все адресное пространство программы может состоять из $2^{16}$ = 64 К сегментов. Сегмент может содержать $2^{16}$ = 64 Кбайта (т.е. иметь адреса от 0 до 65535). При другой структуре адреса изменяется количество сегментов и их длина.

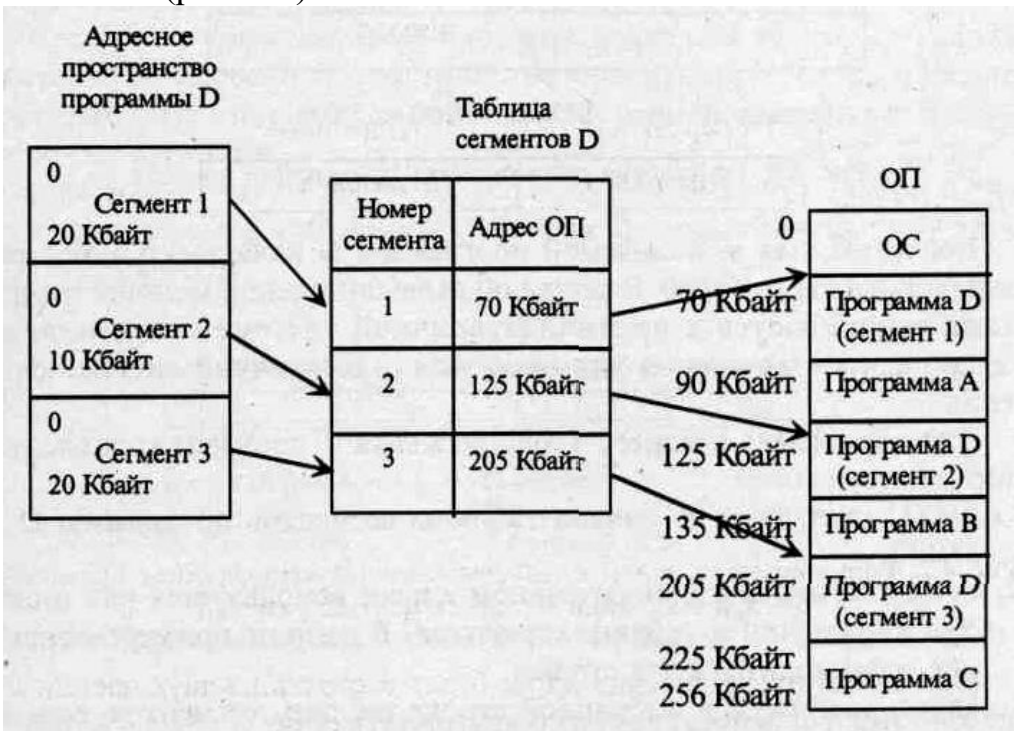
Структура адресов накладывает два важных ограничения:

- ограничивается максимальное число сегментов, которое может существовать в адресном пространстве программы;
- ограничивается максимальное смещение любого адреса в сегменте.

При загрузке в основную память сегментированной программы каждый сегмент перемещается в реальную память отдельно, причем участки основной памяти могут быть или не быть смежными. Трансляция адресов не происходит — сегменты по-прежнему содержат свои относительные адреса.

Процессор может обращаться к основной памяти, используя только абсолютные адреса.

Для динамической трансляции адресов (т.е. при определении абсолютных адресов по известным относительным, содержащим номер сегмента и смещение) операционная система строит специальные таблицы, устанавливающие соответствие между сегментируемым адресным пространством программы и действительными адресами сегментов в реальной памяти (рис. 4.8).




-Рис. **4.8.** Динамическая трансляция адресов при сегментной организации программы

Каждая строка таблицы сегментов содержит адрес начала сегмента в реальной памяти. Для каждого сегмента имеется одна строка таблицы.

Таблицу сегментов содержит каждая выполняемая программа.

В дополнение к таблице сегментов для динамической трансляции адреса используется специальный управляющий регистр, называемый ***регистром начала таблицы сегментов*** (РНТС или STOR — segment table origin register). В этот регистр занесен адрес таблицы сегментов выполняемой в данный момент программы.

На рис. 4.9 изображено выполнение программы D. В РНТС находится адрес таблицы сегментов этой программы. Если программа B прервет выполнение программы D, то в РНТС будет занесен начальный адрес таблицы сегментов программы B.

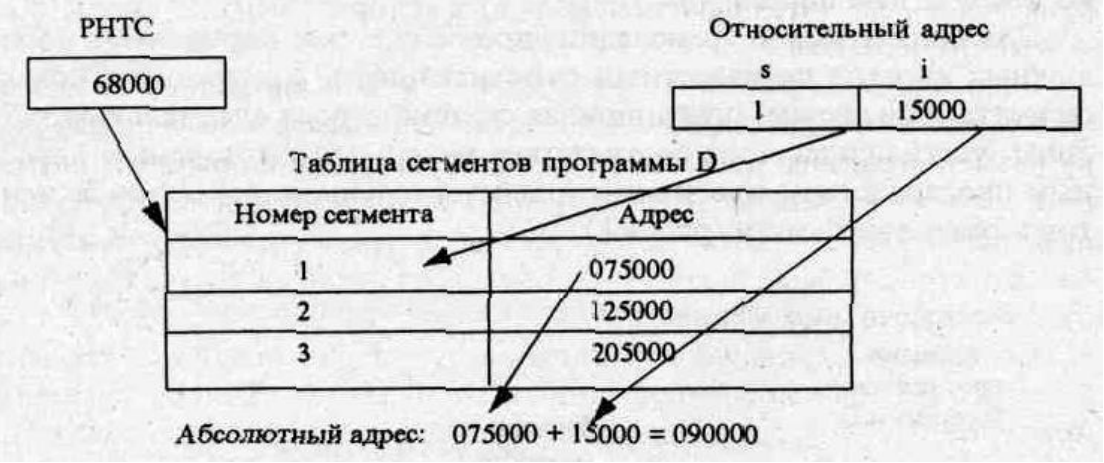



Рис.4.9 Технология динамической трансляции адресов

Допустим, для выполняемой программы D начальный адрес таблицы сегментов — 68000. В реальной вычислительной машине все действия выполняются в шестнадцатеричной системе счисления, мы же проведем вычисления для простоты в десятичной системе счисления.

Для обращения к адресу 15000 сегмента 1 производятся следующие действия:

- РНТС указывает на начало таблицы сегментов программы D — 68000;
- номер сегмента в относительном адресе используется как индекс при обращении к таблице сегментов. В данном примере обращение производится к 1-й строке;
- адрес, хранимый в выбранной строке таблицы сегментов, есть адрес начала сегмента в реальной памяти. Смещение в относительном адресе добавляется к начальному адресу, и результат является адресом в реальной памяти: 15000+75000=90000.

Для относительного адреса (сегмент 3, смещение 13000) будет получен абсолютный адрес 218000.

При ДТА такое определение адресов ведется в процессе выполнения каждой команды.

Если операционной системе понадобится переместить исполняемую программу в другую часть памяти (например, чтобы исключить фрагментацию), сначала надо будет переслать команды и данные сегмента. Затем строку таблицы сегментов для данного сегмента нужно изменить так, чтобы она содержала новый адрес, и выполнение программы может быть продолжено. Это дает возможность динамического управления реальной памятью в процессе выполнения программы.

Использованием сегментации программ достигается уменьшение фрагментации основной памяти, но полностью фрагментация не устраняется — остаются фрагменты, длина которых меньше длины сегментов программы.

Если сегменты разделить на одну или несколько единиц, называемых ***страницами,*** которые имеют фиксированный размер, то, поскольку размер страницы достаточно мал по сравнению с обычным размером сегментов, неиспользуемые фрагменты ОП значительно сокращаются в объеме — будет иметь место так называемая фрагментация внутри страниц. Следовательно, потери все-таки останутся, но они будут существенно меньше.

***Сегментно-страничная организация*** добавляет еще один уровень в структуре адресного пространства программы. Теперь адресное пространство программы дробится на сегменты, внутри сегментов — на страницы, а внутри страниц — на адреса байтов. Структура адреса: *(*S, P, i) — рис. 4.10, где *S* — имя сегмента внутри адресного пространства программы; *P* — имя внутри страницы; i — адрес внутри страницы.

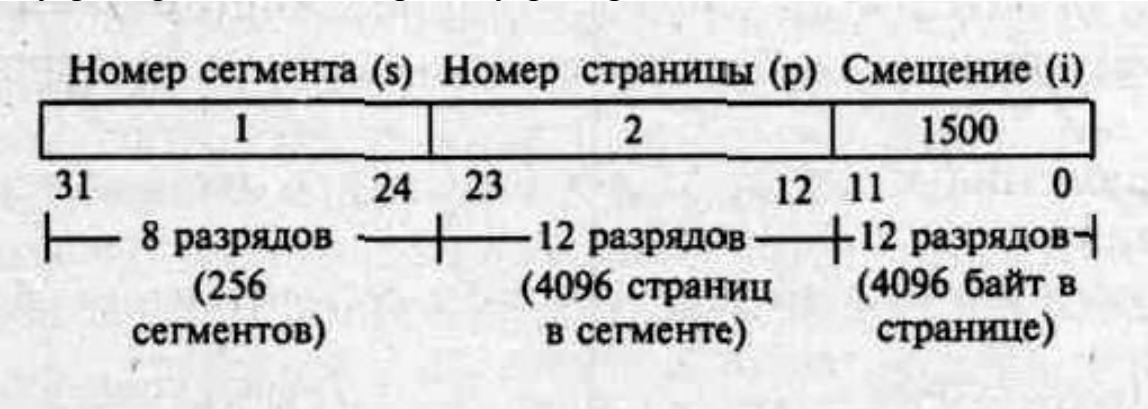



**Рис. 4.10.** Адресная структура при сегментно-страничной организации памяти

Формирование сегментно-страничной структуры выполняется автоматически с помощью операционной системы.

Для динамической трансляции адресов каждому сегменту необходима одна таблица сегментов и несколько таблиц страниц (рис. 4.11). Динамическая трансляция адресов будет выполняться следующим образом:

- регистр начала таблицы сегментов содержит начальный адрес таблицы сегментов выполняемой программы 28000;
- номер сегмента в относительном адресе используется как индекс для обращения к записи таблицы сегментов. Эта запись идентифицирует начало таблицы страницы (реальный адрес) 30000;
- номер страницы в относительном адресе используется как индекс для обращения к записи таблицы страниц. Эта запись идентифицирует начало страничного блока, содержащего эту страницу — 128000;
- смещение в относительном адресе и местоположение страничного блока объединяются вместе, формируя абсолютный адрес 129564. В реальной системе адрес страничного блока и смещение связываются, т.е. соединяются вместе для образования абсолютного адреса.

Все преимущества динамического перемещения с использованием сегментации и страничной организации достигаются благодаря аппаратуре и программному обеспечению, а не пользователям системы. Специальные программы во время загрузки разбивают адресное пространство программы

на сегменты и страницы, строят таблицы сегментов и страниц. Средства ДТА автоматически транслируют адрес в процессе выполнения программы.

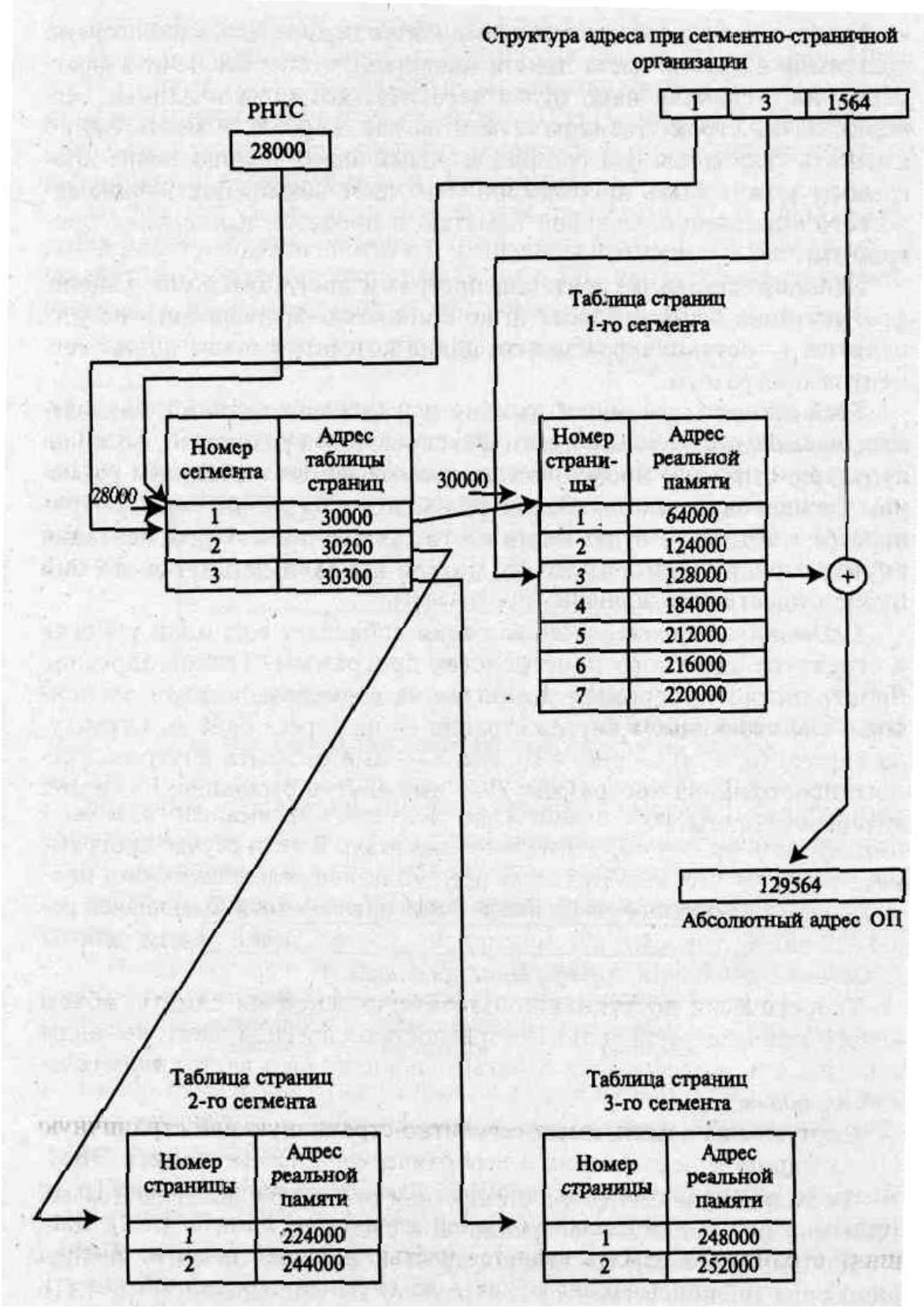



**Рис. 4.11.** Структурная схема формирования абсолютного адреса при сегментно-страничной организации ОП

### 4.4.3.Виртуальная память

Имея иерархическую структуру запоминающих устройств, на реальном объеме памяти, значительно меньшем максимального, можно имитировать работу с максимальной памятью. В этом случае программист работает так, как будто ему предоставлена реальная память максимально допустимого для данной ЭВМ объема, хотя имеющаяся реальная память значительно меньше по объему. Такой режим работы называется ***режимом виртуальной памяти.***

Теоретически доступная пользователю основная память, объем которой определяется только разрядностью адресной части команды и которая не существует в действительности, называется ***виртуальной памятью.***

Виртуальная память имеет сегментно-страничную или страничную организацию и реализована в иерархической системе памяти ЭВМ. Часть ее размещается в ***страничных блоках основной памяти*** (page frames), а часть — в ячейках ***внешней страничной памяти*** (slot). Внешняя страничная память является частью внешней памяти. Ячейка (слот) — это записываемая область во внешней страничной памяти (например, на жестком магнитном диске). Она того же размера, что и страница.

Все программные страницы физически располагаются в ячейках внешней страничной памяти. Виртуальная же память существует только как продукт деятельности операционной системы (функционирующей на основе совместного использования внешней и страничной памяти).

***Загрузить программу в виртуальную память*** — значит переписать несколько программных страниц из внешней страничной памяти в основную память. Если в процессе выполнения программы А система обнаружит, что требуемой страницы нет в реальной памяти, она должна переслать копию этой страницы из внешней страничной памяти в реальную память. Этот механизм называется ***принудительным страничным обменом.***

Максимальный размер виртуальной памяти определяется только длиной физического адреса (32 бита): $2^{32}$ = 4 Гбайта. Размер страницы в IBM PC фиксирован — 4 Кбайта. При таком объеме страниц для адресации байтов внутри страницы необходимо 12 бит. Поэтому адрес виртуальной памяти состоит из двух частей: номера страницы (20 бит) и смещения (12 бит).

Сегмент в виртуальной памяти не влияет на ее размер. Он имеет логический характер, позволяющий специализировать соответствующую часть памяти, определять допустимый характер ее использования. Например, с помощью сегментации можно выделить часть памяти для размещения в ней программ, часть — для размещения данных, часть — для размещения стека. Можно выделить часть памяти только для чтения или для полного доступа и т.д.

Учитывая, что при виртуальной организации в основной памяти хранится только часть страниц, а основным хранилищем информации

являются слоты на жестком диске, номера виртуальных страниц необходимо преобразовывать в номера физических страниц (слотов), в которых должны содержаться такие данные, как имя накопителя, номер цилиндра, номер головки, номер трека, номер сектора и т.д. Это преобразование осуществляется при помощи таблицы страниц. Если производить такое преобразование за один этап, потребуется линейная таблица, содержащая 1М элементов. При размере элемента таблицы 4 байта для хранения таблицы страниц необходим блок памяти 4 Мбайта, причем в мультизадачной среде такая таблица может потребоваться для каждой задачи. Содержать эти страницы в ОП практически невозможно. Поэтому в современных системах реализовано более гибкое двухуровневое преобразование, при котором линейный адрес делится не на две части (номер виртуальной страницы — 20 бит, смещение — 12 бит), а на три: каталог — 10 бит, таблица — 10 бит, смещение — 12 бит. В основной памяти при таком подходе постоянно должны храниться только каталог и активные таблицы страниц.

В каждой записи каталога страниц один из служебных битов (бит присутствия) указывает, является ли данная таблица активной (т.е. присутствует ли она в основной памяти). В записях каждой активной таблицы страниц аналогичный бит присутствия отмечает страницы, которые в настоящий момент находятся в основной памяти. Благодаря такой организации сокращается количество обращений к внешней памяти, что сказывается на производительности ЭВМ.