УАЖАЕМЫЕ СТУДЕНТЫ! Изучите приведенную лекцию, составьте краткий конспект, дайте ответы на контрольные вопросы.

Результаты работы, фотоотчет, предоставить преподавателю на e-mail: v.vika2014@mail.ru

При возникновении вопросов по приведенному материалу обращаться по следующему номеру телефона: (072)1744922

***ВНИМАНИЕ!!!* При отправке работы, не забывайте указывать ФИО студента, наименование дисциплины, дата проведения занятия (по расписанию).**

Лекция продолдение

Тема: Иерархическая структура памяти. Основная память ЭВМ. Оперативное и постоянное запоминающее устройства: назначение и основные характеристики.

Цель: Изучить организацию оперативной памяти. Виды адресации памяти. Кэш-память: назначение, структура, основные характеристики.

План:

1. Память
2. Организация внутренней памяти процессора.
3. Оперативная память и методы управления ОП
4. Методы управления памятью без использования дискового пространства (без использования внешней памяти).
5. Организации виртуальной памяти.
6. Методы повышения пропускной способности виртуальной памяти.
7. Методы организации кэш-памяти
8. Методы обновления строк в динамической памяти

1. Память

Память - один из блоков ЭВМ, которая состоит из запам'ятовуючих пристроил ЗП и предназначен для запоминания, хранения и выдачи информации (алгоритму обработки данных и самих данных).

Основными характеристиками отдельных устройств памяти (запам'ятовуючих пристроил) являются емкость памяти, быстродействие и стоимость хранения единицы информации (бита).

Быстродействие (задержка) памяти определяется порою доступу и длительностью цикла памяти. Время доступа являет собой промежуток времени между выдачей запроса на чтение и моментом поступления спрошенного 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 является противоречивыми. Чем больше быстродействие, тем технически тяжелее достигается и дороже обходится увеличение емкости памяти. Стоимость памяти составляет значительную часть общей стоимости ЭВМ.

Как и большинство устройств ЭВМ, память имеет иерархическую структуру. Обобщенная модель такой структуры, которая отображает многообразие ЗП и их взаимодействие, приведенное на

рис.8.1. Все запам'ятовуючі устройства владеют разным быстродействием и емкостью. Чем выше уровень иерархии, тем выше быстродействие соответствующей памяти, но меньше ее емкость.

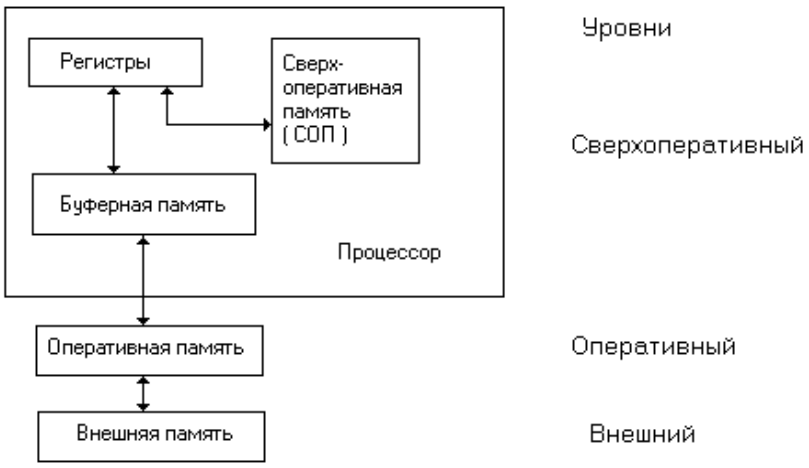

Рисунок 8.1 - Иерархическая структура памяти

До наивысшего уровня - найоперативного - относятся регистры управляющих и операционных блоков процессора, найоперативна память, управляющая память, буферная память (кэш-память).

На втором оперативном уровне, более низком, находится оперативная память (ОП), которая служит для хранения активных программ и данных, то есть тех программ и данных, с которыми работает ЭВМ.

На следующем низшем внешнем уровне размещается внешняя память.

Местная память или регистровая память процессора. Входит в состав ЦП (регистры управляющих и операционных блоков процессора) и предназначена для временного хранения информации. Она имеет малую емкость и наибольшее быстродействие. Построена на базе регистров общего назначения (РЗП). РЗП конструктивно совмещенные с процессором ЭВМ. Этот тип ЗП используется для хранения управляющих и служебных кодов, а также информации, к которой наиболее часто обращается процессор при выполнении программы.

Найоперативна память. Иногда в архитектуре ЭВМ регистровая память организуется в виде найоперативного ЗП с прямой адресацией. Такая память имеет то же назначение как и РЗН, служит для хранения операндов, данных и служебной информации, необходимой процессору.

Управляющая память предназначена для хранения управляющих микропрограмм процессора. Сделана в виде постоянного ЗП (ПЗП) или программируемого постоянного ЗП (ППЗП). В системах с микропрограммным способом обработки информации управляющая память применяется для хранения однажды записанных микропрограмм, управляющих программ, констант и тому подобное.

Буферная память. В функциональном отношении кэш-память рассматривается как буферный ЗП, размещенный между основной (оперативной) памятью и процессором. Основное назначение кэш-памяти - кратковременное хранение и выдача активной информации процессору, который сокращает число обращений к основной памяти, скорость работы которой меньше, чем кэш-памяти. Кэш-память от французького «cashe» - тайник. Она не является программно доступной. Поэтому она влияет на производительность ЭВМ, но не влияет на программирование прикладных задач. В современных ЭВМ различают кэш первого, второго и третьего уровней. Кэш первого уровня интегрирована с блоком предыдущей выборки команд и данных ЦП и служит, по обыкновению, для хранения наиболее часто используемых команд. Кэш второго уровня служит буфером между ОП и процессором. В некоторых ЭВМ существует кэш память отдельно для команд и отдельно для данных.

ОП (ОЗП) служит для хранения информации, непосредственно участвует в вычислительном процессе (что происходит в операционном устройстве - АЛП). Из ОЗП в процессор поступают коды и операнды, над которыми проводятся предвиденные программой операции, из процессора в ОЗП направляются для хранения промежуточные и конечные результаты обработки информации. ОЗП имеет сравнительно большую емкость и высокое быстродействие, однако меньше, чем ЗП найоперативного уровня.

Внешняя память (ЗовнП) используется для хранения больших массивов информации в течении длительного времени. Обычно ЗовнП не имеет непосредственной связи с процессором. Обмен информацией носит групповой характер, который значительно сокращает время обмена. ЗовнП имеет сравнительно низкое быстродействие (поиск информации). В качестве носителя используются магнитные диски (гибкие и жесткие), лазерные диски (CD - R) и др.

Сравнительно небольшая емкость оперативной памяти (на настоящее время до 16 ГБ в домашних ПК) компенсируется практически неограниченной емкостью внешних запам'ятовуючих пристроил. Однако эти устройства сравнительно медленны - время обращения по данным для магнитных дисков представляет десятки микросекунд. Для сравнения: цикл обращения к оперативной памяти (ОП) представляет где-то 50 нс. Исходя из этого, вычислительный процесс должен протекать из возможно меньшим числом обращений к внешней памяти.

Рост производительности ЭВМ оказывается в первую очередь в увеличении скорости работы процессора. Быстродействие ОП также растет, но все время отстает от быстродействия аппаратных средств процессора потому, что одновременно происходит опережающий рост ее емкости, которая делает тяжелее уменьшение времени цикла работы памяти. Вследствие этого быстродействие ОП оказывается недостаточным для обеспечения необходимой производительности ЭВМ. Проявляется это в несоответствии пропускных способностей процессора и памяти. Для выравнивания этих пропускных способностей и назначена найоперативна буферная память небольшой емкости (как правило, не больше 6 Мбайт) и повышенного быстродействия.

При обращении к блоку данных, который находится на оперативном уровне, его копия пересылается в найоперативну буферную память. Дальнейшие обращения к этому блоку данных проводится к буферной памяти. Поскольку время выборки из НайОЗП ТНайОЗП много менее времени выборки из оперативной памяти Топ, введение в структуру ЭВМ НайОЗП приводит к уменьшению эквивалентного времени обращения Те по сравнению со временем обращения до оперативной памяти ТОП :

$$Te = TНайОЗП + \alpha TОП$$

где $\alpha = 1 - q$,

q - достоверность попадания, то есть вероятность того, что блок данных, к которому осуществляется обращение, находится в НайОЗП._

2. Организация внутренней памяти процессора.

В архитектуре современных ЭВМ стал стандартным прием организации регистров общего назначения в виде НайОЗП с прямой адресацией (адреса регистров размещаются в команде). В машинах с коротким словом, которое вынуждает удаваться к одноадресным командам, один из общих регистров выделяется в качестве аккумулятора, регистра, в котором находится один из операндов и в который помещается результат операции. Регистр аккумулятора в явном виде в команде не адресуется, используется адресация за замовченням. В этом случае НайОЗП с прямой адресацией состоит из совокупности регистров, связанных со входной X и исходной Z шинами (см. рис. 8.2).

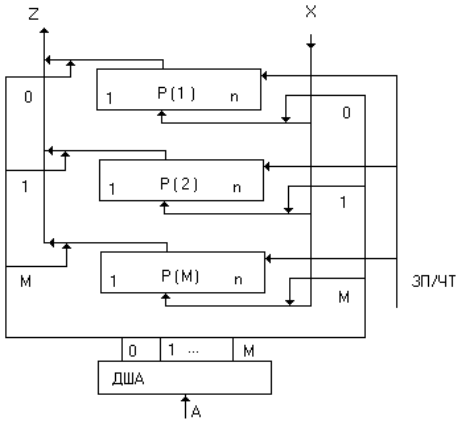

Рисунок 8.2 - НайОЗП с прямой адресацией с одним адресным входом

Дешифратор адреса формирует управляющие сигналы 0,1, ..., M, подключают регистр с заданным адресом к шинам НайОЗП.

Адрес регистра, к которому проводится обращение с целью записи или чтения (управляющий сигнал ЗП / ЧТ) информации, поступает по шине А.

При использовании двухадресных команд типа «регистр - регистр» подобная организация НайОЗП становится неэффективной, потому что за один такт может быть выбрано содержимое только одного регистра.

Для реализации таких команд за один такт НайОЗП строится в виде совокупности регистров, соединенных из одной входной и двумя исходными шинами (см. рис. 8.3). Адреса регистров, к которым проводится обращение с целью чтения информации, поступают по шинам Но и В. Адрес регистра для записи информации поступают по входу В.

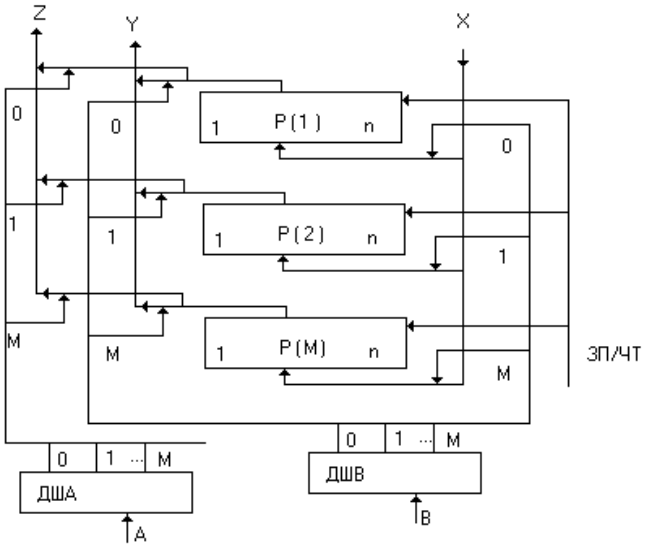

Рисунок 8.3 - НайОЗП с прямой адресацией с двумя адресными входами

Двохадресна команда, в которой адресуются два операнда, расположенные в регистрах, и результат операции размещается за одним из этих адресов [0 ... M].

Дешифратори адреса формируют управляющие сигналы, которые подключают два регистра к исходным шинам при чтении и один регистр при записи.

Стековая память реализует безадресное задание операндов, является эффективным элементом архитектуры ЭВМ. Стек являет собой группу последовательно пронумерованных регистров (*аппаратный стек*) или ячеек памяти (*программный стек*), обеспеченных указателем стека (обычно регистром), в котором автоматически при записи и считывании устанавливается номер (адрес) первой свободной ячейки стека (вершина стека).

При операции записи заносятся в стек слово помещается в свободную ячейку стека, а при считывании из стека вытягивается последнее слово, которое поступило у него. Таким образом, в стеці реализуется принцип LIFO «последний пришел - первый пошел».

Механизм стековой адресации объясняется на рис.8.4.

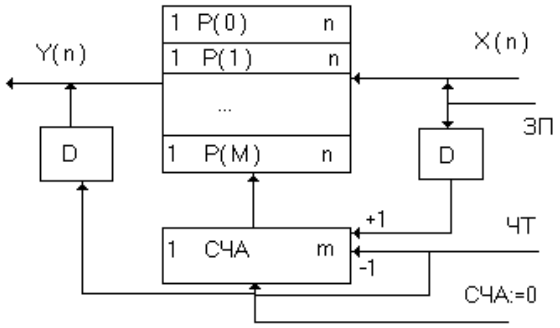

Рисунок 8.4 - Стековая память.

Предусматривается, что отрасль памяти для стека находится в сегменте стека, база которого определяется регистром SS - сегментным регистром стека. При добавлении записи в стек сначала проверяется, или содержит указатель стека (ESP) значения, не меньше длины помещается в стек записи (2 байта для 16-разрядного и 4 байта для 32-разрядного процессора). Если это условие не довольно, то генерируется особенный случай нарушения стека. Если же ESP содержит значение не меньше необходимого, проводится декремент указателя стека на 2 (4) и операнд хранится по адресу SS : SP (SS: ESP) в текущем сегменте стека, на который указывает указатель стека (Число 4 - число байт в 32-х разрядном процессоре). При получении данных из стека содержимое ESP сравнивается с пределом SS. Если обращение оказывается за пределы, формируется особенный случай нарушения стека. Когда обращение оказывается разрешенным, считываются данные по адресу SS : [ESP] и осуществляется инкремент ESP на 4. Вытянуть данные из стека можно в регистр или в память.

У современных архітектурах процессоров стек и стековая адресация широко используется при организации переходов к подпрограммам и возвращению из них, а также в системах прерывания.

Прежде чем приступать к изучению принципов организации внешней памяти (ЗовнП), следует отметить следующее.

В последнее время емкость микросхем динамической памяти учетверяється каждые три года. Но скорость этих микросхем за тот же период росла намного меньшими темпами (приблизительно 7% в год). В то время, как производительность процессоров, начиная с 1987р, увеличивалась на 50% в год. Таким образом, согласование производительности современных процессоров со скоростью ОП вычислительных машин и систем остается одной из важнейших проблем. Методы повышения производительности за счет увеличения размеров Кэш-памяти и введения многоуровневой организации КЭШ могут оказаться недостаточно эффективными с точки зрения стоимости системы. Поэтому важным направлением современных разработок являются методы повышения пропускной способности памяти за счет ее организации, включая специальные методы организации динамических ЗП.

3. Оперативная память и методы управления ОП

Оперативная память (system memory) - имеет относительно небольшую емкость - до 32 Гбайт (в некоторых машинах - больше). Количество и быстродействие оперативной памяти предоставляет чрезвычайно серьезное влияние на быстродействие современных компьютеров. Работает на частоте системной шины. Время доступа до оперативной памяти представляет порядку 20-70 нс (для сравнения - время доступа до жестких магнитным дискам представляет десятки микросекунд).

Доступ процессора к оперативной памяти происходит через кэш 2-го уровня. Некоторые подсистемы компьютера способны обращаться к оперативной памяти непосредственно, проходя процессор. Строится ОП как правило на DRAM Dynamic Random Access Memory) - динамических запам'ятовуючих устройствах случайного доступа.

DRAM (Dynamic RAM) - динамическая память - разновидность памяти, единичная ячейка которой являет собой конденсатор с диодным конструкцией. Наличие или отсутствие заряда конденсатора отвечает единице или нулю. Основной вид, применяемый для оперативной памяти, видеопамяти, а также разных буферов и кешей более медленных устройств. В сравнении из SRAM заметно более дешевая, хотя и более медленная по двум причинам - емкость заряжается не мгновенно, и, кроме того, имеет ток истока, который делает необходимой периодическую подзарядку.

SRAM (Static RAM) - статичная память - разновидность памяти, единицей хранения информации в которой является состояние «открыто-закрытый» у транзисторной сборнику. Используется преимущественно в качестве кэш-памяти 2-го уровня. Ячейка SRAM более сложная в сравнении с ячейкой DRAM, потому высшее быстродействие SRAM компенсируется высокой ценой. Невзирая на низкое энергопотребление, является энергозависимой, то есть при отключении питания информация теряется.

ОП является наиболее дефицитным и наиболее важным ресурсом в вычислительных машинах и системах.

Проблема управления ОП усложняется при переходе к мультипрограммным системам, потому что в них ОП используют одновременно несколько вычислительных процессов.

Для идентификации переменных и команд используют символьные имена (метки), виртуальные адреса и физические адреса.

4. Методы управления памятью без использования дискового пространства (без использования внешней памяти).

Все методы управления памятью могут быть разделены на два класса (рис. 8.5) :

- методы распределения ВП без использования внешней памяти (дискового пространства);
- методы распределения памяти с использованием дискового пространства.

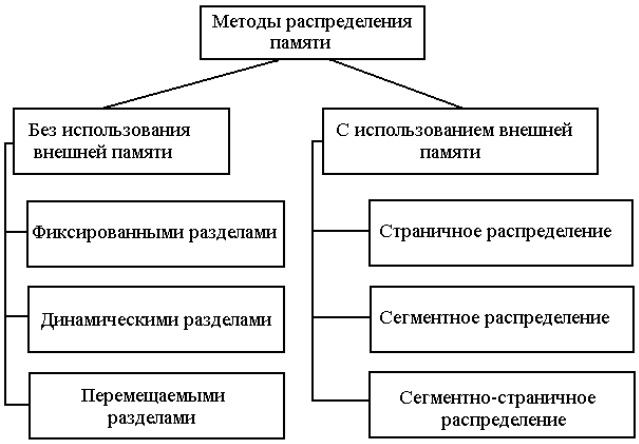

Рисунок 8.5 - Методы управления памятью

РАСПРЕДЕЛЕНИЕ ПАМЯТИ ФИКСИРОВАННЫМИ РАЗДЕЛАМИ.

Это наиболее простой способ распределения памяти. Вся ОП разделяется на определенное число разделов фиксированной величины. Дежурный процесс (или задание) поступила на выполнение, становится в общую очередь или в очередь к несвойственному по размеру разделу памяти (Когда раздел освобождается, дежурный процесс (программа) подгружается в ОП).

В этом случае подсистема управления памятью выполняет задание:

- сравнение размера поступила на выполнение программы с размерами свободных разделов памяти;
- выбору подходящего раздела;
- загрузка программы и настройка адресов.

При очевидном преимуществе - простоте реализации - этот способ распределения имеет очевидный минус: жесткость. Во-первых, одна программа занимает весь раздел полностью. Это ведет к тому что, во-первых, неэкономно тратится память, во-вторых, коэффициент мультипрограммирования ограничен числом разделов. С другой стороны, даже если суммарный объем свободной ОП машины позволяет выполнять некоторую программу, разбития памяти на разделы не позволяет сделать этого.

Другой способ - распределение памяти разделами переменной величины. При таком способе распределения в начале работы ЭВМ вся ОП свободная. Каждой поступает на выполнение задачи выделяется необходимый ей объем ОЗП. Если достаточный объем памяти отсутствующей, задание не принимается на выполнение и стоит в очереди. По завершению задачи память освобождается, и на это место может быть загружена другая задача.

Т.о., в произвольный момент времени ОП являет собой случайную последовательность свободных и занятых областей памяти приблизительно такого вида.

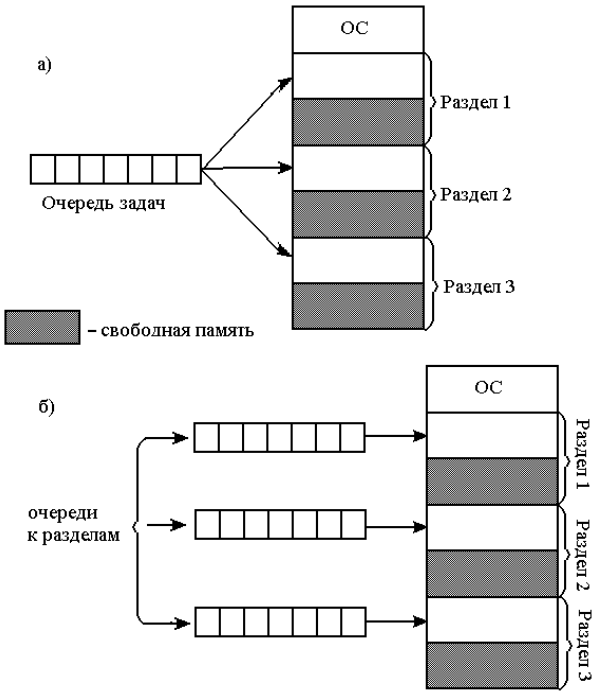

Рисунок 8.6 - Распределение памяти фиксированными разделами.

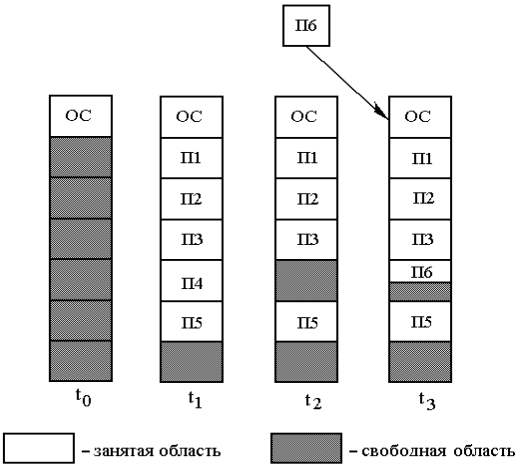

Рисунок 8.7 - Распределению памяти динамическими разделами

В начальный момент времени Т0 в ОП загружена только ОС. К моменту времени Т1 ОП разделена между ОС и 5 программами (задачами), есть также свободная область. К моменту времени Т2 задания П2 уже завершена и покидает ОП, а на ее место может быть догружая задание На освобожденное место загружается задача П6, которая поступила в момент времени Т3. Выбором раздела для задачи, которая опять поступила, занимается ОС. Осуществляется выбор раздела по правилам: «первый-попавшийся раздел достаточного размера», «раздел, который имеет наименьший достаточный размер», «раздел, который имеет наибольший достаточный размер».

Кроме выбора раздела для для задачи, которая опять поступила, ОС также выполняет задание:

- ведение таблиц свободных и занятых областей, в которых указывается начальные адреса и размеры участков памяти;
- анализ запроса (при поступлении новой задачи);
- пересмотр таблицы свободных областей (с целью выбора раздела для размещения для задачи, которая опять поступила);
- загрузка задачи в выделенный ей раздел;
- корректировка таблиц свободных и занятых областей (как после загрузки дежурной задачи в ОП, так и по завершению задания).

В сравнении с методом распределения памяти фиксированными разделами данный метод гибче, но ему присущий серьезный недостаток - *фрагментация памяти*. Фрагментация - это наличие многих несмежных областей памяти, настолько маленьких по размеру, что ни в одну из них нельзя поместить ни одну из тех, которые пришли на выполнение программ, хотя суммарный объем таких фрагментов может сложить значительную величину.

РАЗМЕЩЕНИЕ ПАМЯТИ С ПЕРЕМЕЩАЕМЫМИ РАЗДЕЛАМИ.

Одним из методов борьбы с фрагментацией есть перемещение всех занятых участков в сторону старших или в сторону младших адресов так, чтобы все свободные участки памяти складывали единственную область (см. рис.8.8). В добавление к функциям, которые выполняет ОС при распределении памяти переменными разделами, в данном случае она должна еще время от времени копировать содержимое разделов с одного места памяти в другое, корректируя таблицы свободных и занятых областей.

Такая процедура называется сжатием и выполняется ОС в добавление к функциям, которые ОС выполняет при распределении ОП переменными разделами.

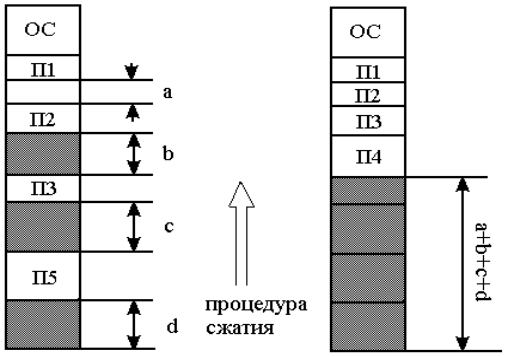

Рисунок 8.8 - Распределение памяти перемещаемыми разделами.

Сжатие может выполняться или при каждом завершении задачи, или только тогда, когда для опять поступила задаче нет свободного раздела достаточного размера. В первом случае нужно меньше вычислительной работы при корректировке таблиц, а во втором - реже выполняется процедура сжатия. Так как программы перемещаются по оперативной памяти в ходе своего выполнения, то превращение адресов из виртуальной формы в физическую должно выполняться динамическим способом.

Хотя процедура сжимания и приводит к более эффективному использованию памяти, она может требовать значительного времени, которое может свести на нет преимущества данного метода.

5. Организации виртуальной памяти.

Виртуальная память возникла как средство решения проблемы размещения в ОП программ, размер которых значительно превышает имеющуюся свободную память.

Виртуальным называют такой ресурс, который для пользователя представляется владеет теми свойствами, которыми он в действительности не владеет.

Пользователь пишет программы так, как будто в его распоряжении однородна ОП большого объема, но в действительности все данные, используемые программой, хранятся на нескольких разнородных ЗП, обычно в ОП и на дисках, и при необходимости частями перемещаются между ними.

Т.о., виртуальная память (ВП) - это совокупность программно-аппаратных средств, которые позволяют пользователям писать программы, размер которых превосходит имеется ОП.

Для этого ВП решает след. задание:

- размещает данные в ЗП разного типа, например, часть программы в ОП, а часть на диске;
- перемещает данные по мере необходимости между ЗУ разного типа, например, догружает нужную часть программы из диска в ОП;
- превратит виртуальные адреса в физические.

Все эти действия производятся без участия программиста. Можно еще сказать, что механизм ВП является «прозрачным» по отношению к пользователю. Наиболее распространенными реализациями ВП является страничное, сегментное и странично-сегментное распределение памяти, а также свопинг.

СТРАНИЧНОЕ РАСПРЕДЕЛЕНИЕ.

Виртуальное адресное пространство каждого процесса (пр.1 и пр.2, см. рис. 8.9) делящееся на части фиксированного размера, называемые виртуальными страницами. Размер виртуального адресного пространства в общем случае не является кратным размеру страницы, потому последняя страница каждого процесса дополняется фиктивной областью. Вся ОП ЭВМ делящаяся на части такого же размера, називанi физическими страницами (блоками ли).

Размер страницы обычно выбирается ровным степени двойки, потому это позволяет упростить механизм превращения адресов.

Часть виртуальных страниц процесса при его загрузке помещается в ОП, а часть на жесткий диск. ОС при загрузке процессора формирует для него отдельную информационную структуру - *таблицу страниц*, в которой устанавливается соответствие *виртуальных страниц* с определенными номерами (N в.с.) *физическим страницам* с определенными номерами (N ф.с.). Также в таблице страниц содержится *управляющая информация* (КІ) : признак модификации страницы, признак невивантажності (выгрузка некоторых страниц может быть запрещена), признак обращения к странице (используется для подсчета обращений к странице за определенный период времени), а также некоторые другие данные создаются и используемые механизмом ВП.

При активизации дежурного процесса в регистр адреса таблицы страниц считывается адрес таблицы страниц этого процесса.

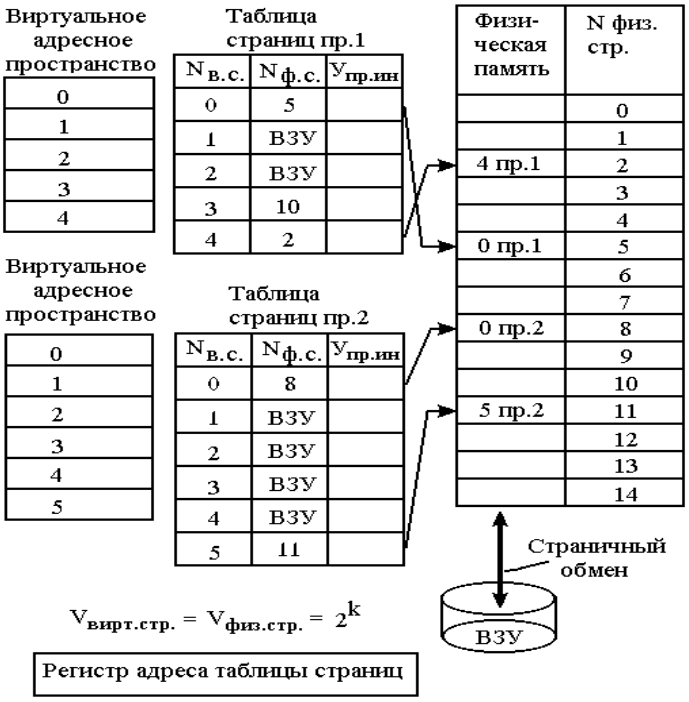

Рисунок 8.9 - Страничное распределение памяти

При каждом обращении к памяти происходит чтение из таблицы страниц информации о виртуальной странице, к которой состоялось обращение. Если данная виртуальная страница находится в ОП, выполняется превращение виртуального адреса в физический. Если же нужна НД выгружена на диск, то генерируется т.н. *случай страничного прерывания.* Процесс переводится в состояние ожидания, и активизируется другой процесс, из очереди готовых. Параллельно с этим программа обработки страничного прерывания находит на диске необходимую страницу и попробует загрузить ее в ОП. Если в памяти есть свободная ФС, то соответствующая НД загружается в ОП, если же свободных страниц нет, то реализуется процедура выталкивания из ОП страницы. Критерии выбора выталкивания НД могут быть следующими:

- дольше всего не использовалась страница;
- первая-попавшаяся (случайная) страница;
- страница, к которой было менее всего обращений.

После того, как избранная страница, которая должна быть удалена из ОП, анализируется ее признак модификации. Если страница, которая выталкивается, с момента загрузки процесса была модифицирована, то ее модификация должна быть переписана на диск, если же нет, то соответствующая физическая страница делается свободной, а виртуальная страница просто уничтожается.

В некоторых системах используется механизм рабочего множества страниц для каждого процесса. Это *рабочее множество* - перечень наиболее часто используемых страниц, которые постоянно находятся в ОП и выгрузка которых запрещена.

Рассмотрим механизм превращения виртуального адреса в физический при страничной организации памяти (см. рис. 8.10).

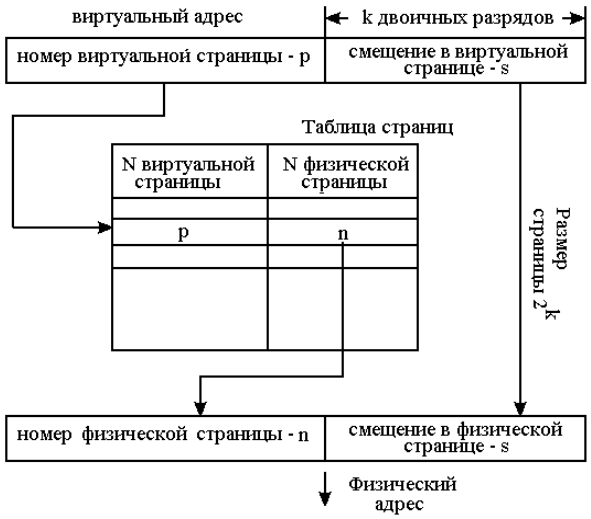

Рисунок 8.10 - Механизм превращения виртуального адреса

При каждом обращении к ОП аппаратными средствами производятся следующие действия.

1. На основании начального адреса таблицы страниц (содержимому регистра адреса таблицы страниц), номера виртуальной страницы и длины записи в таблице страниц определяется нужный адрес записи в таблице страниц.
2. Из этой записи вытягивается номер физической страницы;
3. К номеру физической страницы присоединяется сдвиг, то есть младшие разряды ВА (путем конкатенации).

Применении операции конкатенации вместо более длительных операций добавления уменьшает время получения физического адреса, а значит и повышает производительность компьютера.

На производительность системы из страничной организацией памяти влияют временные расходы, связанные с обработкой страничных прерываний и превращением ВА в физическую. Чем чаще возникает страничное прерывание, тем более времени тратится на перемещение страниц. Чтобы уменьшить частоту страничных прерываний, нужно увеличить размер страницы. Увеличение размера страницы уменьшает размер таблице страниц, а значит, и уменьшает расходы памяти. Но, с другой стороны, чем больше размер виртуальной страницы, тем более памяти занимает фиктивная область в конце последней виртуальной страницы каждой программы.

Время превращения ВА в ФА в значительной степени определяется порою доступу к таблице страниц. Поэтому таблицу страниц, как правило, размещают в «скорых» ЗП. Это может быть набор специальных регистров или буферная память, которая использует ассоциативный поиск и кэширование данных.

СЕГМЕНТНОЕ РАСПРЕДЕЛЕНИЕ

При страничной организации памяти ВА - пространство процесса делящееся на механически равные части. Это не позволяет дифференцировать способы доступа к разным частям программы (сегментам), а это свойство часто бывает полезным. Например, можно запретить обращаться с операциями записи и чтения в сегмент кода программы, а для сегмента данных позволить только чтение. Кроме того, разбивка программы на сегменты, а не на страницы делает возможным разделение одного сегмента несколькими процессами.

Рассмотрим, каким образом сегментное распределение памяти реализует эти возможности (рис. 8.11.). Виртуальный адрес памяти ВАП процесса разделяется на сегменты, размер которых определяется программистом с учетом значения содержится в них информации. Отдельным сегментом м.б. подпрограммы, массив данных и др. Иногда сегментация программы может выполняться за замовченням компилятором.

При загрузке процесса одни сегменты записываются в ОП, другие остаются на диске. Сегменты одной программы могут занимать несмежные участки ОП. Во время загрузки ОС создает для процесса таблицу сегментов (аналогичную таблице страниц), в которой для любого сегмента указывается начальный физический адрес сегмента в ОП, размер, правила доступа, признак модификации, признак обращения к данному сегменту за последний интервал времени и некоторая другая информация. Если ВАП нескольких процессов включают тот же сегмент, то в таблицах сегментов этих процессов делаются ссылки на один и тот же участок ОП, в который этот сегмент загружается в единственном экземпляре.

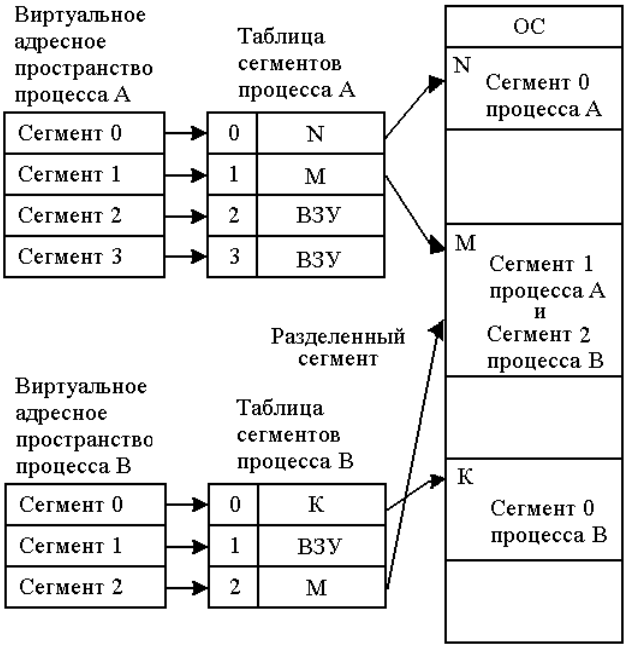

Рисунок 8.11 - Распределение памяти сегментами

Система с сегментной организацией функционирует аналогично системе из страничной организацией.

При каждом обращении к ОП из таблицы сегментов считывается информация о сегменте, к которому проводится обращение.

Если сегмент находится в ОП, то осуществляется превращение виртуального адреса в физический. Если же сегмент находится на диске, то возникает случай страничного прерывания; процесс переводится в состояние ожидания, и активизируется другой процесс. Аналогичным способом при необходимости освобождения ОП некоторые сегменты выгружаются. Вместе с тем, при обращении к памяти проверяется, или разрешенный доступ необходимого типа к данному сегменту.

Виртуальный при сегментной организации представлен парой чисел (g, s), где первое - номер сегмента, второе - смещение внутри сегмента.

ФА выходит путем добавления начальной ФА сегмента, найденного в таблице сегментов по номере g, и сдвига s. Операция складывания является медленнее, что в целом имеет негативное влияние на быстродействие машины. Еще одним недостатком сегментной организации также является фрагментация на уровне сегментов.

СТРАНИЧНЫЙ - СЕГМЕНТНОЕ РАСПРЕДЕЛЕНИЕ.

Данный метод сочетает в себе преимущества обоих подходов, являя собой их комбинацию. Все ВАП процесса разделяется на сегменты, а каждый сегмент - на страницы. ОП при этом разделяется на физические страницы. Загрузка процесса в ОП осуществляется постраничный. ОС при загрузке процесса создает *таблицу сегментов* (TS) процесса. Для каждого сегмента создается своя *таблица страниц* (TP), структура которой полностью аналогична структуре таблицы страниц при страничном распределении. Адрес таблицы сегментов загружается в специальный регистр процессора, когда активизируется соответствующий процесс. В таблице страниц находятся адреса таблиц страниц для всех сегментов данного процесса. По схеме, приведенной на рисунке 8.12, виртуальный адрес превратится в физический.

Процесс подобного превращения является достаточно длительным. Чтобы его ускорить и тем же повысить производительность ЭВМ, применяют быстродействующие ассоциативные ЗП, то есть устройства, в которых применяется принцип ассоциативности.

Если в системе реализовано такое устройство, то превращение ВА в ФА осуществляется по такому принципу, называемому динамическим превращением адресов (см. рис. 8.13) : ВА, представленный парой чисел (g, p) передается в качестве поискового признака в ассоциативном запам'ятовуючому устройстве АЗП, т.а . VAi первой появляется в ячейке А буфера. Вторым полем ячейки является ФА страницы в ОП. При выявлении совпадения VAi с содержимым памяти из соответствующей ячейки Асс. Буфера, вытягивается соответствующий ФА страницы, что позволяет сформировать полный ФА элемента данных в ОП. Если же VAi не совпадает ни с одной ячейкой АЗП, то превращение осуществляется обычным способом через таблицы сегментов и страниц. Эффективность применения динамического превращения адресов определяется тем, насколько редко используются табличные превращения. Как правило, при первом обращении к странице, расположенной в ОП, ФА определяется с помощью таблиц и загружается в соответствующую ячейку АЗП, с тем, чтобы следующие обращения к странице могли выполняться с использованием АЗП.

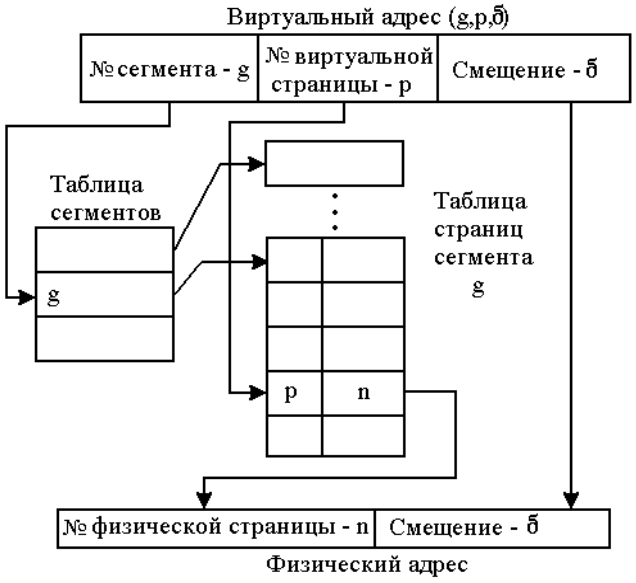

Рисунок 8.12 - Схема превращения виртуального адреса в физическую

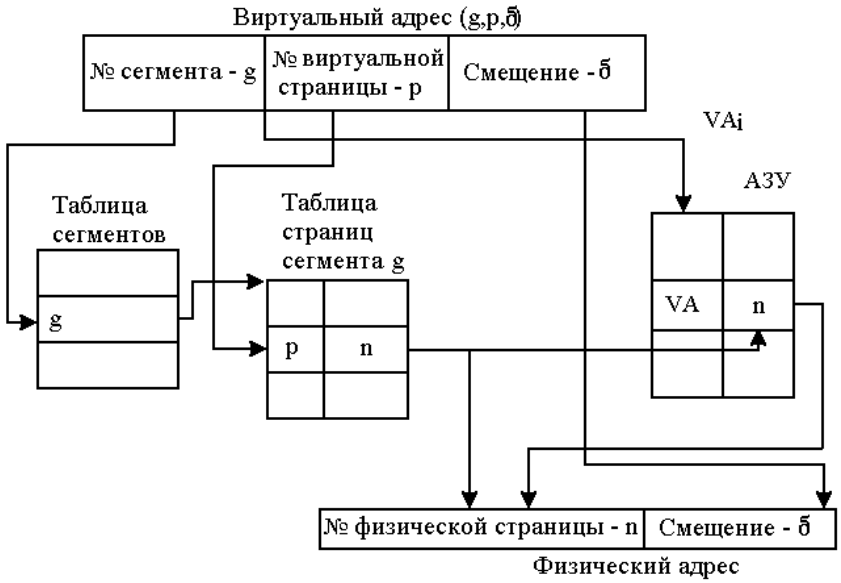

Рисунок 8.13 - Механизм динамического превращения адресов

Разновидностью виртуальной памяти является *свопинг*. Для того, чтобы задача могла начать выполняться, она должна быть загружена в ОП, объем которой ограничен.

На рис. 8.14 показанный график зависимости коэффициента загрузки процессору в зависимости от числа одновременно выполняемых процессов и часть времени, проведенного этими процессами в состоянии ожидания введения-выведения.

Экспериментально выведенная зависимость загрузки процессора от числа одновременно выполняемых заданий и от интенсивности введения / выведения. Из черт. 8.14 видно, что для того, чтобы загрузить процессор на 90%, достаточно 3-х заданий с небольшой интенсивностью введения-выведения, а, чтобы обеспечить такую же загрузка интерактивными заданиями с интенсивным введением-выводом, нужно 10 таких заданий. Объем же ВП ограничен. Чтобы увеличить уровень мультипрограммирования, и был предложенный метод организации вычислительного процесса, называемый *свопинг.* В соответствии с этим методом некоторые процессы (задание), которые обычно находятся в состоянии ожидания, полностью могут выгружаться (выкачиваться) на диск, а на их место догружаются другие. При этом программа-планировщик ОС не исключает их из своего рассмотрения и при воссозданные условий, в которых возможно выполнять некоторую задачу, которая находится в области свопинга на диске, эта задача перемещается в ОП. Существуют разные алгоритмы выгрузки процессов на диск и закачки других процессов, а также разные способы выделения оперативной и дисковой памяти процессу, который загружается.

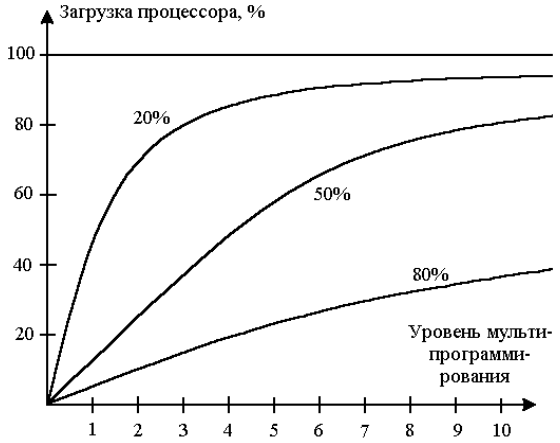

Рисунок 8.14 - Зависимость загрузки процессора от числа задач и интенсивности введения / выведения

6. Методы повышения пропускной способности ВП.

Для чего нужно повышать пропускную способность ОП? Прежде всего, для того, чтобы за одно обращение к памяти можно было считать большее количество информации и тем же сократить число обращений к ней. Основными методами увеличения полосы пропускания памяти являются: увеличение разрядности или «ширины» памяти, использования расслоения памяти, использования независимых банков памяти, обеспечения режима бесконфликтного поведения к банкам памяти, использования специальных режимов работы динамических микросхем памяти.

ВЫБОРКА ШИРОКИМ СЛОВОМ

Прямой способ сокращения числа обращений к ОП складывается в организации. выборки широким словом. При выборке широким словом за одно обращение к ОП выполняется одновременное считывание (или запись) нескольких команд или слов данных с широкой ячейке. Широкое слово заносится в буферную память (кэш-память или регистр), где оно расформировывается на отдельные команды (или слова данных), которые могут (последовательно) использоваться процессором уже без дополнительных обращений к ОП.

В системах с кэш-памятью 1 уровня ширина шин данных ОП часто отвечает ширине шин данных кэш-памяти, которая во многих случаях имеет физическую ширину шин данных, соответствующую количеству разрядов в слове. Удваивание или учетверение ширины шин кэш-памяти и ОП удваивает или учетверяет соответствующую полосу пропускания системы памяти.

Реализация выборки широким словом вызывает необходимость мультиплексирования данных между кэш-памятью и процессором, поскольку основной единицей обработки данных в процессоре остается слово. Кэш-память второго уровня позволяет смягчить эту проблему, так как в этом случае мультиплексоры могут располагаться между двумя уровнями кэш-памяти, и задержка, которая вносится ими, не такая критическая. Другая проблема, связанная с увеличением разрядности памяти, заключается в необходимости определения минимального инкремента, то есть минимального объема памяти для поэтапного ее расширения, которое часто выполняется самими пользователями во время эксплуатации вычислительной системы.

Удваивание или учетверение ширины памяти приводит к удваиванию или учетверению этого минимального инкремента. Кроме того, есть проблемы и с организацией коррекции ошибок в системах с широкой памятью.

Примером системы с организацией широкой ОП есть система Alpha AXP 21064, в которой кэш 2 уровня, шина памяти и сама ОП имеют разрядность 256 бит.

РАССЛОЕНИЕ СООБЩЕНИЙ.

Другой способ повышения пропускной способности ВП связан с построением памяти, которое находится на физическом уровне из нескольких модулей (банков) с автономными схемами адресации, записи и чтения. При этом на логическом уровне управления памятью организуются последовательные обращения к разным физическим модулям. Обращение к разным модулям могут перекрываться, и т.о. образуется своеобразный конвейер. Эта процедура носит название *расслоения памяти*. Целью данного метода является увеличение скорости доступа к памяти с помощью сочетания фаз обращений до многих модулям памяти. Существуют несколько вариантов организации расслоения. Наиболее часто используемый способ - расслоение обращений за счет расслоения адресов. Этот способ основывается на свойстве локальности программ и данных, что допускает, что адрес следующей команды на 1 больше адреса предыдущей (другими словами, линейность программы нарушается только командами перехода). Аналогичная последовательность адресов генерируется процессором при чтении слов данных.

Т.о., типичный случай распределения адресов - последовательность вида a, a+1, a+2, a+3 и так далее (Для слов данных - увеличения на 1 - условно, в действительности 1 - число байт в машинном

слове). Из этого выплывает, что расслоение обращений возможно, если ячейке с адресами a, a+1, a+2, a+3 и т.д будут размещаться в блоках 0,1,2 ... Такое распределение ячеек по модулям (банкам) обеспечивается за счет использования адресов вида (см. рис. 8.15).

Здесь B - k -розрядний адрес модуля (младшая часть m -розрядної адреса), C - n -розрядний адрес ячейки в модуле B (старшая часть адреса)

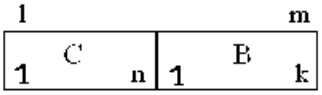

Рисунок 8.15 - Формат адреса при организации расслоении обращений к памяти за счет расслоения адресов.

Принцип расслоения адресов иллюстрирован на рис 8.15 (а).

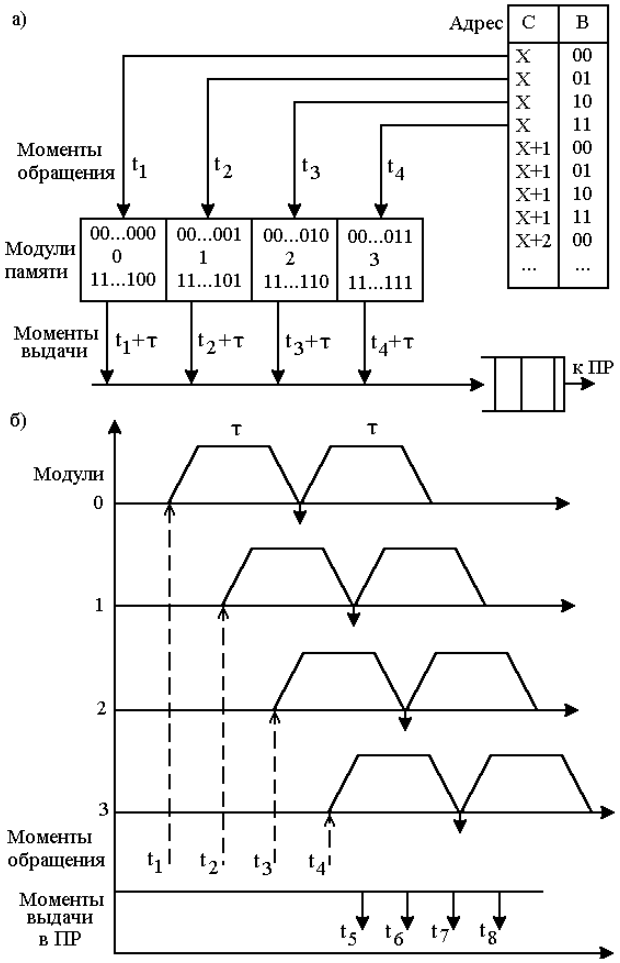

Рисунок 8.16 - Организация адресного пространства при расслоении памяти (а), временная диаграмма работы модулей (б)

Все команды и данные размещены в адресном пространстве последовательно. Однако ячейки памяти, которые имеют смежные адреса, находятся в разных модулях памяти. Если ОП состоит из 4-х модулей, то номер модуля кодируется двумя младшими разрядами адреса. При этом полные m -разрядные адреса 0,4,8, ... Относятся к блоку 0, адреса 1,5,9,13 - к блоку 1, адреса 2,6,10 - к блоку 2 и адреса 3,7,11,15 - к блоку 3. Т.о., последовательность обращений к адресам 0,1,2,3,4,5,6 .... будет расслоена между 4 модулями: 0,1,2,3,0,1,2 ... (чудес черт 8.16)

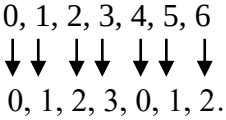

Рисунок 8.17 - Расслоение последовательности обращений к адресам между модулями памяти.

Поскольку каждый модуль памяти имеет собственные схемы управления выборкой, можно обращение к следующему модулю производить, не ожидая ответа от предыдущего.

На временной диаграмме видно, что время доступа до каждого модуля одно: $\tau = 4T$, где $T = t_{i+1} - t_i$ - длительность такта. В каждом такте следуют обращения к модулям памяти в моменты времени t1, t2, t3, ... При наличии 4-х модулей темп выдачи квантов информации из памяти в процессор будет отвечать 1 такту Т, при этом скорость выдачи информации из каждого модуля в 4 раза ниже, то есть сложит 4Т.

Задержка в выдаче кванта информации относительно момента обращения к модулю также составит 4 такта, однако задержка в выдаче каждого следующего кванта относительно момента выдачи предыдущего сложит Т.

При реализации расслоения за адресами число модулей памяти может быть произвольным и необязательно кратным 2. В некоторых компьютерах предвидено произвольное отключение модулей памяти, что позволяет исключить из конфигурации неисправные модули.

В современных высокопроизводительных компьютерах число модулей представляет 4-16, но иногда превышает 64.

Для повышения производительности мультипроцессорных систем, которые работают в многозадачных режимах, применяют другие методы расслоения, при которых разные процессоры обращаются к разным модулям памяти.

Необходимо помнить, что процессоры введения-выведения также занимают циклы памяти и вследствие этого могут сильно влиять на производительность системы. Для уменьшения этого влияния обращению ЦП и процессоров вводу-виводу организуют к разным модулям памяти.

Обобщением идеи расслоения памяти является возможность реализации нескольких независимых обращений, когда несколько контролеров памяти позволяют модулям памяти (или группам расслоенных модулей памяти) работать независимо.

Прямое уменьшение числа конфликтов чередуются при обращении к памяти может быть достигнуто путем размещения программ данных в разных модулях.

Поскольку обращение к командам и элементам данных чередуются, то разделение памяти на память команд и память данных повышает быстродействие машины подобно рассмотренному выше механизму расслоения. *Подол памяти на память команд и память данных* широко используется в системах управления или обработки сигналов. У подобного рода системах в качестве данных используются ПЗП, цикл которых меньше цикла устройств, которые допускают запись. Такое решение делает разделение данных и команд достаточно эффективным.

Выбор той или другой схемы расслоения для компьютерной и другой вычислительной системы определяется целями (достижение высокой производительности при решении огромного количества заданий или высокого быстродействия при решении одной задачи), а также архитектурными и структурными особенностями системы и элементной базой (соотношением длительности циклов памяти и узлов обработки).

7. Методы организации кэш-памяти

В функциональном отношении кэш-память рассматривается как буферный ЗП, размещенный между основной (оперативной) памятью и процессором. Основное назначение кэш-памяти - кратковременное хранение и выдача активной информации процессору, который сокращает число обращений к основной памяти, скорость работы которой меньше, чем кэш-памяти.

За единицу информации при обмене между основной памятью и кэш-памятью принята *строка*, причем под строкой понимает набор слов, избираемый из оперативной памяти при одном к ней обращении. Сохраненная в оперативной памяти информация есть, таким образом,

совокупностью строк с последовательными адресами. В любой момент времени строки в кэш-памяти являют собой копии строк из некоторого их набора в ОП, однако расположенные они необязательно в такой же последовательности, как в ОП.

ТИПИЧНАЯ СТРУКТУРА КЭШ-ПАМЯТИ

Рассмотрим типичную структуру кэш-памяти (рис. 8.18), которая включает основные блоки, которые обеспечивают ее взаимодействие из ОП и центральным процессором.

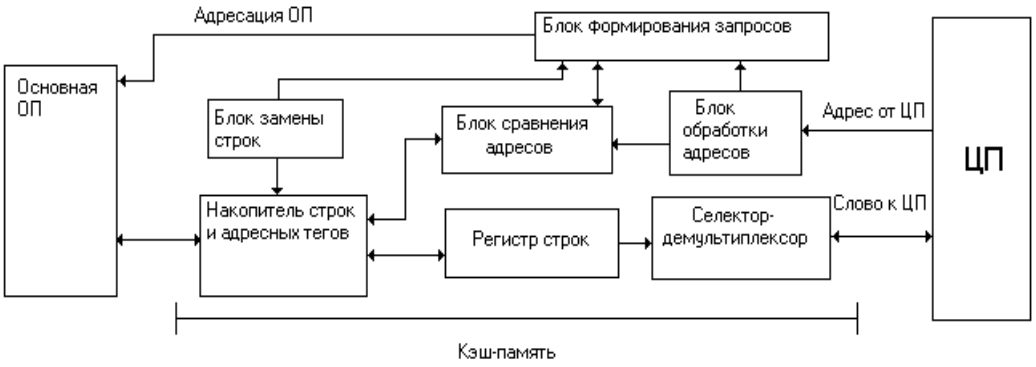

Рисунок 8.18 - Типичная структура кэш-памяти

Строки, составленные с информационных слов, и связанные с ними адресные теги хранятся в накопителе, который является основой кэш-памяти. Адрес необходимого слова, которое поступает от центрального процессора (ЦП), вводится в блок обработки адресов, в котором реализуются принятые в данной кэш-памяти принципы использования адресов при организации их сравнения с адресными тегами. Само сравнение производится в блоке сравнения адресов (БПА), который конструктивно сочетает с накопителем, если кэш-память строится по схеме ассоциативной памяти. Назначение БПА заключается в выявлении попадания или промахе при обработке запросов от центрального процессора. Если имеет место *кэш-попадание* (то есть слово, которое ищется, хранится в кэш-памяти, о чем свидетельствует совпадение кодов адреса, которые поступают от центрального процессора, и одного с адресом некоторого адресного тега), то соответствующая строка из кэш-памяти переписывается в регистр строк. С помощью селектора-демультиплексора из нее выделяется необходимое слово, которое и направляется в центральный процессор. В случае *промаха* с помощью блока формирования запросов осуществляется инициализация выборки из ОЗП необходимые строки. Адресация ОП при этом проводится в соответствии с информацией, которая поступила от центрального процессора. Избранная из памяти строка вместе со своим адресным тегом размещается в накопителе и регистре строк, а потом слово, которое искалось, передается в центральный процессор.

Для высвобождения места в кэш-памяти с целью записи избранной из ОП строки, одна из строк удаляется. Определение строк для удаления проводится с помощью блока замены строк, в котором хранится информация, необходимая для реализации принятой стратегии возобновления находятся в накопителе строк.

СПОСОБЫ РАЗМЕЩЕНИЯ ДАННЫХ В КЭШ-ПАМЯТИ.

Существует четыре способа размещения данных в кэш-памяти:

- прямое распределение
- полностью ассоциативное
- частично ассоциативное
- распределение секторов.

Рассмотрим обстоятельно каждый способ размещения и механизмы превращения адресов. Допустимо, что КЭШ содержит 128 строк, размер строки 16 слов, а основная память может

содержать 16384 строки. Для адресации основной памяти используется 18 бит. Из них 14 старших показывают адрес строки, а младшие 4 - адрес слова внутри этой строки. Строки Кэш-памяти указываются 7-разрядными адресами.

Прямое распределение. При прямом распределении место хранения строк в кэш-памяти однозначно определяется по адресу строки. Структура кэш-памяти с прямым распределением показана на рис. 8.19.

Адрес основной памяти состоит из 14-ти разрядного адреса строки и 4-х разрядного адреса слова внутри этой строки. Адрес строки подразделяется на старшие 7 бит (тег) и младшие 7 бит (индекс). Для того, чтобы поместить в кэш-память строку из основной памяти с адресом ABC, выбирается область внутри кэш-памяти с адресом B, который равняется 7 младшим битам адреса строки AB. Превращение из ABC в B возводится только к выборке младшие 7 бит адреса строки AB. По адресу B в кэш-памяти может быть помещена будь из 128 строк основной памяти, которые имеют адрес, 7 младших бит которого равняются адресом B. Для того, чтобы определить, которая именно строка хранится в памяти данных в данный момент времени, используется запам'ятовуючий устройство емкостью 7 * 128 слов, в котором помещается по соответствующему адресу в качестве тега 7 старших бит адреса строки, которая хранится в настоящее время по адресу B кэш-памяти. Это запам'ятовуючий устройство называется тегов памятью. Память, в которой хранятся строки, называется памятью данных. Тег из тегов памяти считывается по адресу B, который образует 7 младших бит адреса строки AB. Параллельно считыванию тега осуществляется доступ к памяти данных с помощью 11 младших бит (НД) адреса основной памяти ABC. Если тег и старшие 7 бит адреса основной памяти сбегаются, значит что данная строка существует в памяти данных (строка-V), то есть осуществляется кэш-попадание.

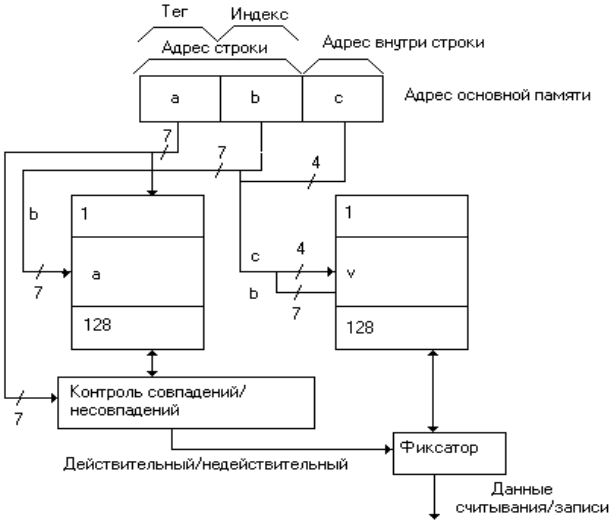

Рисунок 8.19 - Структура кэш -пам'яті с прямым распределением

Если же происходит кэш-промах, то есть тег отличается от старших 7 бит, то из основной памяти считывается соответствующая строка, а из кэш-памяти удаляется строка-V, обусловленная 7 младшими разрядами адреса строки, а на ее место помещается строка, считанная из основной памяти. Осуществляется также обновление соответствующего тега в тегов памяти. Способ прямого распределения реализуется достаточно просто, однако из-за того, что место хранения строк в кэш-памяти однозначно определяется по адресу строки, вероятность сосредоточения областей хранения строк в некоторой части кэш-памяти высока, то есть замены строк будут происходить достаточно часто. В этой ситуации эффективность кэш-памяти заметно снижается.

Полностью ассоциативное распределение. При таком способе размещения данных каждая строка основной памяти может быть розміщен на месте любой строки кэш-памяти. Структура кэш-памяти с полностью ассоциативным распределением выглядит как показано на рис 8.20.

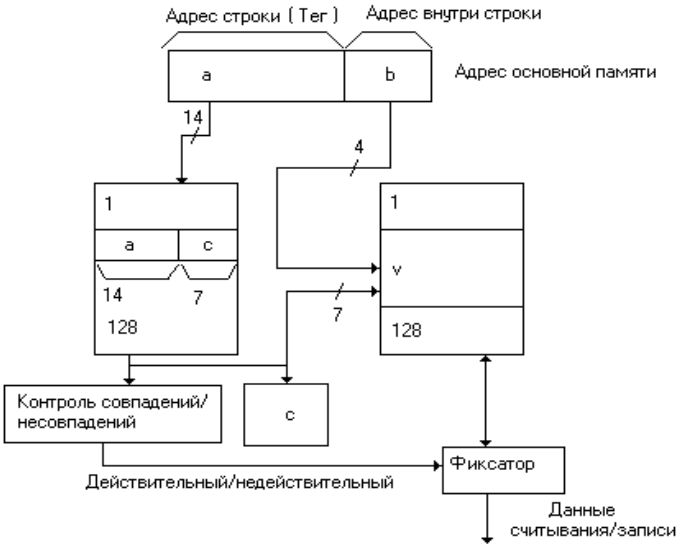

Рисунок 8.20 - Структура кэш-памяти с полностью ассоциативным распределением

При полностью ассоциативном распределении механизм превращения адресов должен давать ответ на вопрос, или существует копия строки с произвольным адресом в кэш-памяти, и, если существует, то по какому адресу. Для этого необходимо, чтобы тегова память была ассоциативной памятью. Входной информацией для ассоциативной памяти тегов является тег А, а исходной информацией - адрес строки внутри кэш-памяти (С). Каждое слово тегов памяти состоит из 14-разрядного тега и

7-разрядного адреса Со строки внутри кэш-памяти. Ключом для поиска адреса строки внутри кэш-памяти является тег А (старшие 14 разрядов адреса основной памяти). При совпадении ключа А с одним из тегов Т тегової памяти (случай попадания) происходит выборка соответствующих данному тегу адреса С и обращение к памяти данных. Входной информацией для памяти данных есть 11-ти разрядное слово НД (7 бит адреса строки У + 4 бита адреса слова в этой строке С). В случае неспівпадання ключа ни с одним из тегов тегової памяти (случай промаха) формируется запрос к основной памяти на оборыше строки с соответствующим адресом и считывание этой строки. По этому способу при замене строк кандидатом на удаление могут быть все строки в кэш-памяти.

Частично ассоциативное распределение. При данном способе размещения, несколько соседних строк (фиксированное число, не менее двух) из 128 строк кэш-памяти образуют структуру называемую группой. Структура кэш-памяти, основанная на использовании частично ассоциативного распределения, показана на рис. 8.21. В данном случае в одну группу Е входят 4 строки А, В, С, D.

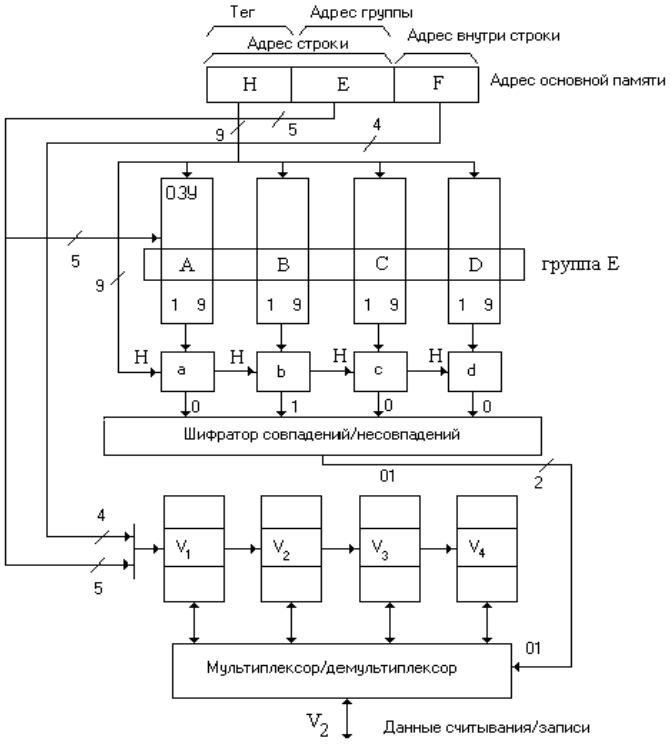

Рисунок 8.21 - Структура кэш-памяти, которая основана на использовании частично ассоциативного распределения.

Адрес строки HE основной памяти (14 бит) разделяется на две части: H-тег (старшие 9 бит) и E - адрес группы (младшие 5 бит). Адрес строки внутри кэш-памяти, которая состоит из 7 бит, разделяется в адрес группы E (5 бит) и адрес строки внутри группы (2 бит: 00,01,10,11).

Для расположения в кэш-память строки, сохраненной в ОП по адресу HEF, необходимо выбрать группу с адресом E. При этом не суть важно, которая из четырех строк в группе может быть избранным. Для выбора группы используется метод прямого распределения, а для выбора строки в группе используется метод полностью ассоциативного распределения.

Когда центральный процессор спрашивает доступ по адресу HEF, то осуществляется обращение к массиву тегов по адресу E, выбирается группа из четырех тегов (A, B, C, D), каждый из которых сравнивается со старшими 9 битами (H) адреса строки. На выходе четырех схем сравнение формируется унитарный код совпадения (H = A - код: 1000, H = B - код: 0100, H = C - код: 0010, H = D - код: 0001), который на шифраторы превратится в двухразрядный позиционный код, служащий адресом для выбора банка данных (00,01,10,11) - адрес строки внутри группы.

Одновременно осуществляется обращение к массиву данных (банкам V1, V2, V3, V4,) по адресу EF (9 бит) и считывание из банка V2 необходимой строки или слова.

При пересылке новой строки в кэш-память удаляется из нее строка выбирается из четырех строк соответствующего набора (группы).

Распределение секторов. По этому способе основная память разбивается на секторы, которые состоят из фиксированного числа строк, кэш-память также разбивается на секторы, которые состоят из такого же числа строк. Допустимо, в секторе 16 строк, а в строке - 16 слов. Структура кэш-памяти с распределением секторов представлена на рис. 8.22.

В адресе основной памяти 10 старших бит задают адрес сектора A, следующие 4 бита - адрес строки B в секторе и младшие 4 бита - адрес слова C в строке.

При данной организации кэш-памяти, распределение секторов у кэш-памяти и основной памяти осуществлено полностью ассоциативно, то есть, каждый сектор A основной памяти может отвечать любому сектору D в кэш-памяти. К каждой строке V, которая хранится в кэш-памяти, добавляется один бит достоверности (действительности); он показывает, сбегается или нет содержимое этой строки с содержанием строк в основной памяти, которая в данный момент

анализируется на соответствие строка кэш-памяти. Если слова, что спрашивают центральным процессором при доступе, не существует в кэш-памяти (бит достоверности, избранный по адресу BD равняется 0), то сначала центральный процессор проверяет, или был сектор A, что содержит это слово, размещенный раньше в кэш-память. Если он отсутствует, то один из секторов кэш-памяти заменяется на этот сектор.

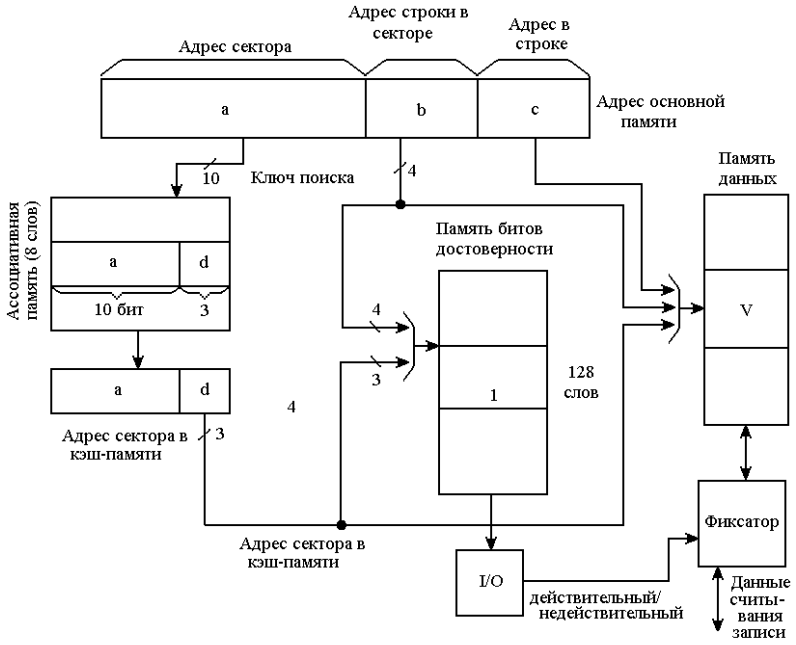

Рисунок 8.22 - Структура кэш-памяти с распределением секторов.

Если все секторы кэш-памяти используются, то выбирается один какой-либо сектор, и при необходимости только некоторые строки этого сектора возвращаются в основную память, а этот сектор можно использовать дальше.

Когда осуществляется доступ к сектору A в кэш-памяти и строку B, что содержит нужное слово C, пересылается из основной памяти, то бит достоверности устанавливается к строке, которая пересылалась. Все биты достоверности других строк этого сектора смахивают. Если сектор A, что содержит слово B доступ к которому спрашивают, уже находится в кэш-памяти, то, в том случае когда бит достоверности строки, которая содержит это слово, равняется 0, этот бит устанавливается и строка пересылается из основной памяти в данную область кэш- памяти. В том случае, когда бит достоверности уже равняется 1, нужное слово можно считывать из кэш-памяти.

8. Методы обновления строк в основной памяти

В таблице 8.1. приведенные условия сохранения и обновления информации в ячейках кэш-памяти и основной памяти.

Если процессору требуется информация из некоторой ячейки основной памяти, а копия этой ячейки уже есть в кэш-памяти, то вместо оригинала считывается копия. В этом случае информация ни в кэш-памяти, ни в основной памяти не изменяется.

При записи строк существует несколько методов обновления старой информации. Эти методы называются стратегией возобновления строк основной памяти. Если результат возобновления строк кэш-памяти не возвращается в основную память, то содержимое основной памяти становится неадекватным вычислительному процессу. Во избежание таких ошибок, предвидены разные методы обновления основной памяти.

Таблица 8.1

Условия сохранения и обновления информации

Режим работы	Наличие копии ячейки ОП в кэш-памяти	Информация в ячейке кэш-памяти	Информация в ячейке ОП
Чтение	Копия есть	Не изменяется	Не изменяется
	Копии нет	Обновляется (создается копия)	Не изменяется
Сквозная запись	Копия есть	Обновляется	Обновляется
	Копии нет	Не изменяется	Обновляется
Обратная запись	Копия есть	Обновляется	Не изменяется
	Копии нет	Обновляется	Не изменяется

Контрольные вопросы:

1. Дайте определение термину «память».
2. Сделайте перечень основных характеристик запам'ятовуючих пристроил.
3. Чем определяется быстродействие памяти?
4. Проанализируйте иерархическую структуру памяти.
5. Какая память имеет название сверх оперативной?
6. Опишите назначение буферной памяти.
7. Для чего используется внешняя память?
8. Опишите организацию внутренней памяти процессора.
9. В чем уникальность стековой памяти?
10. Опишите механизм стековой адресации.
11. Дайте определение термину «оперативная память».
12. Сделайте сравнительный анализ DRAM и SRAM.
13. Какие Вы знаете методы управления памятью без использования дискового пространства (без использования внешней памяти)?
14. Каким образом происходит распределение памяти фиксированными разделами?
15. Опишите принцип размещения памяти с перемещаемыми разделами.
16. Дайте определение термину «виртуальная память».
17. Сделайте перечень наиболее распространенных реализаций виртуальной памяти.
18. Чем отличается страничное распределение памяти от сегментного?
19. Что такое свопинг?
20. Какие Вы знаете методы повышения пропускной способности виртуальной памяти?
21. Для чего назначенная кэш-память?
22. Наведите типичную структуру кэш-памяти.
23. Какие существуют способы размещения данных в кэш-памяти?
24. Опишите методы организации кэш-памяти.
25. Для чего осуществляется обновление строк в основной памяти?
26. Какие системы внешней памяти Вы знаете?

Рекомендованная литература

1. Агуров П. В. Последовательные интерфейсы ПК. Практика программирования. - СПб.: БХВ-Петербург, 2004. - 496с.

2. Балашов Е.П., Пузанков Д.В. Микропроцессоры и микропроцессор-ные системы: Учеб. пособие для вузов /Под редакторши, В.Б. Смоляная. - М.: Радио и связь, 1981. - 328 с.
3. Лихтциндер Б.Я., Кузнецов В.Н. Микропроцессоры и вычисли - тельные устройства в радиотехнике: Учеб, пособие. - Киев: Вища.шк., 1988. - 272 с.
4. Самофалов К.Г., Викторов О. В. Микропроцессоры. - Киев: Техника 1989, - 312 с.