Федеральное государственное автономное образовательное
учреждение высшего образования
"Национальный исследовательский университет
"Высшая школа экономики"

Московский институт электроники и математики

Департамент компьютерной инженерии

СТРАНИЧНОЕ РАСПРЕДЕЛЕНИЕ ПАМЯТИ

Методические указания к лабораторной работе

Составитель: к.т.н., доцент Е.М.Иванова

Москва 2020

1. ЦЕЛЬ И ПРАКТИЧЕСКОЕ СОДЕРЖАНИЕ МЕТОДИЧЕСКИХ УКАЗАНИЙ

1.1. Цель работы

Целью работы является закрепление теоретических знаний по разделам «Память».

1.2. Краткое содержание

В настоящих указаниях приводится описание особенностей управления виртуальной памятью: оперативной или буферной кэш-памятью. Суть механизма управления – перемещение блоков данных между устройствами памяти соседних уровней иерархии.

Методические указания содержат достаточное число вариантов заданий на курсовую работу для студентов группы.

2. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

При работе вычислительной системы (ВС) информация (коды программ и данных) перемещаются между различными устройствами памяти. Подсистема памяти представляет собой иерархическую структуру (см. рис. 1).

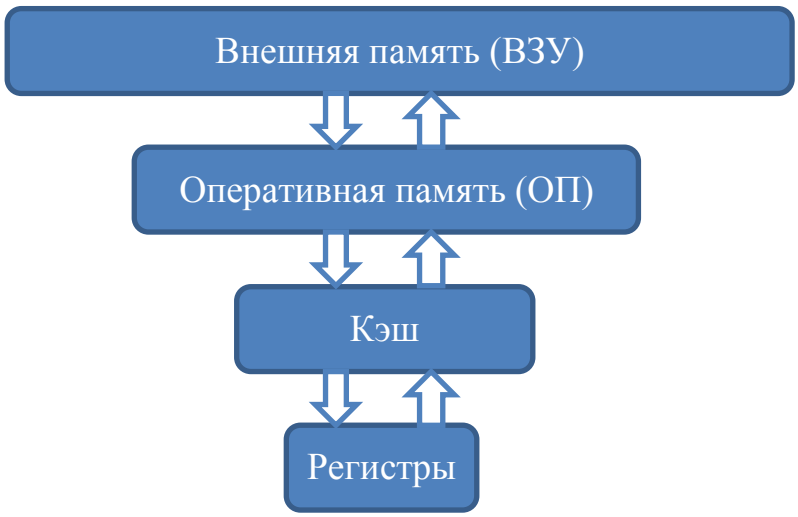


Рис.1. Иерархия памяти

В ходе вычислений данные перемещаются ближе к вычислителю (процессору), т.е. с более удалённого уровня иерархии в более приближенный к процессору, где запоминающие устройства (ЗУ) характеризуются меньшей ёмкостью. Всегда встает вопрос о возможности размещения данных и наличии свободной памяти на более низком уровне иерархии. Как правило такая задача решается при перемещении данных 1) между ВЗУ и ОП, и 2) между ОП и Кэш. Существует несколько управляющих стратегий в каждом случае.

Всегда задача разбивается на фрагменты и память разбивается на блоки. Вместо размещения всей задачи в памяти целиком при необходимости в память копируется лишь нужный фрагмент задачи в свободный блок памяти. Т.о. в нужный момент времени (при обращении по определенному адресу) нужный

блок оказывается в памяти, но только на время работы с ним (пока процессор генерирует адрес обращения в память, который находится в пределах адресного пространства внутри блока), см. рис.2.

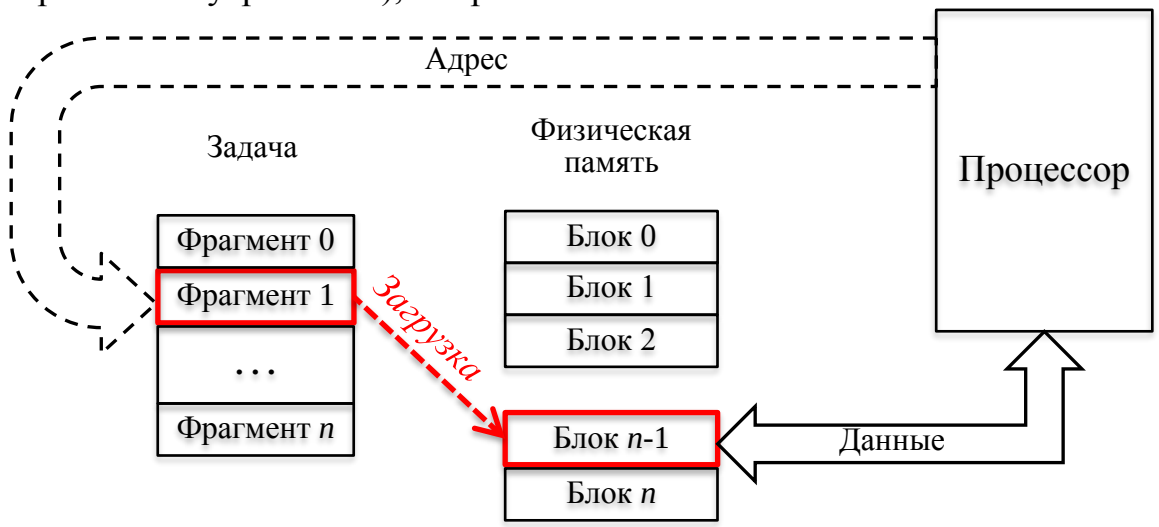


Рис. 2. Принцип организации работы виртуальной памяти

Как только адрес обращения выходит за границы блока, блок перестает быть актуальным и существует возможность его замещения на более востребованную копию другого блока данных из памяти. Данная работа посвящена изучению алгоритмов определения наименее востребованного блока – кандидата на замещения. Алгоритмы замещения похожи как для стратегии обмена страницами между ВЗУ и ОП, так и для стратегии обмена строками Кэш и ОП. В первом случае блок представляет собой страницу/сегмент с ориентацией на организацию ОП (размер страницы примерно от 2 КБ до 4МБ), во втором случае – на строку Кэш (размером примерно 16-32 Б).

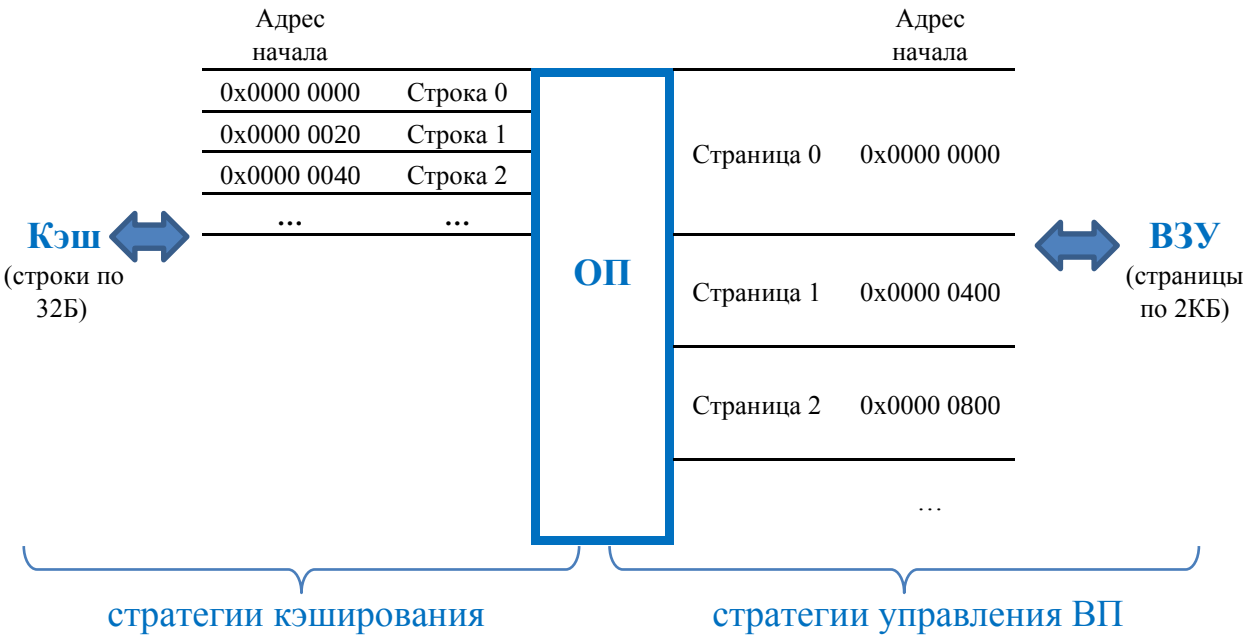


Рис.3. Интерпретация размера блоков ОП

Физически никакая память на блоки/строки/страницы/сегменты не разбивается, просто при реализации механизма управления область памяти с

определенными адресами м.б. интерпретирована как страница или как строка.

2.1. Понятие Виртуальной памяти

При перемещении данных между ВЗУ и ОП используется понятие виртуальной памяти (ВП). Стратегия возникла как средство решения проблемы размещения программ и данных, размер которых значительно превышает имеющуюся в наличии свободную или выделенную задаче оперативную память.

Виртуальным называют такой ресурс, который для пользователя представляется обладающим теми свойствами, которыми он в действительности не обладает. Касательно памяти это свойство – ёмкость. Физически ёмкость оперативной памяти увеличить нельзя, но можно создать видимость большого доступного объёма. Пользователь пишет программы так, как будто в его распоряжении имеется однородная ОП большого объема, но в действительности все данные, используемые программой, хранятся на нескольких разнородных ЗУ, обычно в ОП и внешней памяти (на диске), и при необходимости частями перемещаются между ними (см. рис.4).

Рис. 4. Область виртуальной памяти

Т.о., **виртуальная память** (ВП) – это совокупность программно-аппаратных средств, включающих кроме ОП другие внешние запоминающие устройства (чаще HDD) и позволяющих пользователям писать программы, размер которых превосходит выделенную задаче ОП.

Для организации ВП нужно:

- разместить данные в ЗУ разного типа, например, часть программы в ОП, а часть на диске;
- перемещать данные по мере необходимости между ЗУ разного типа, например, подгружает нужную часть программы с диска в ОП;

- преобразовывать виртуальные адреса в физические при каждом обращении к памяти.

Все эти действия выполняются без участия программиста, т.е. можно сказать, что механизм ВП является «прозрачным» по отношению к пользователю. С его помощью организуется управление ОП компьютера.

2.2. *Способы управления ОП с использованием внешнего ЗУ*

Существуют различные алгоритмы отгрузки задач на диск, а также различные способы выделения оперативной и дисковой памяти загружаемой задаче.

Наиболее распространенными способами реализации ВП являются страничное, сегментное и странично-сегментное распределение памяти, а также свопинг.

Свопингом называется метод организации вычислительного процесса, при котором задачи, находящиеся в состоянии ожидания, целиком могут отгружаться (откачиваться) на диск, а на их место подгружаться другие готовые к исполнению задачи.

При страничной организации памяти виртуальное адресное пространство каждой задачи и ОП делятся на механически равные части – страницы, что существенно упрощает процедуру замены страниц ОП↔Диск, но не позволяет дифференцировать доступ к разным типам данных на странице и зачастую приводит к наличию не использующейся памяти в конце последней страницы задачи.

При сегментном распределении виртуальное адресное пространство задачи делится на сегменты, размер которых определяется программистом с учетом смыслового значения и фактического количества содержащейся в них информации. Отдельным сегментом может быть программа, массивы данных или стек. Иногда сегментация программы может выполняться по умолчанию компилятором. Это позволяет дифференцировать способы доступа к разным сегментам, в отличие от страниц, но усложняет организацию работы с памятью. Например, можно запретить обращаться с операцией записи в сегмент кода программы, а для сегмента данных разрешить как чтение, так и запись. Памяти для хранения сегмента выделяется ровно столько, сколько нужно. Кроме того, разбиение программы на сегменты, а не на страницы делает возможным разделение одного сегмента несколькими процессами. Тогда этот сегмент загружается в ОП в единственном экземпляре.

Странично-сегментное распределение сочетает в себе достоинства обоих подходов, представляя собой их комбинацию. Всё виртуальное адресное пространство задачи составляется из сегментов, что позволяет при обращении к памяти автоматически выполнять проверки разрешенных адресов. Совокупность сегментов специальным образом преобразуется в совокупность виртуальных страниц. ОП при этом делится на физические страницы. Загрузка процесса в ОП осуществляется постранично. Очевидно, что такая организация памяти будет позволять дифференциацию способов доступа к разным сегментам и имеет простую подкачки/откачки страниц из/в ОП – все страницы

одинаковые. Но при реализации потребуется организовать управление как сегментами так и страницами и все-таки не удается избежать наличия в конце последней страницы неиспользуемой памяти.

2.3. *Страничное распределение*

Виртуальное адресное пространство каждой задачи (задачи1 и задачи2, см. рис.5) делится на части фиксированного размера, называемые виртуальными страницами.

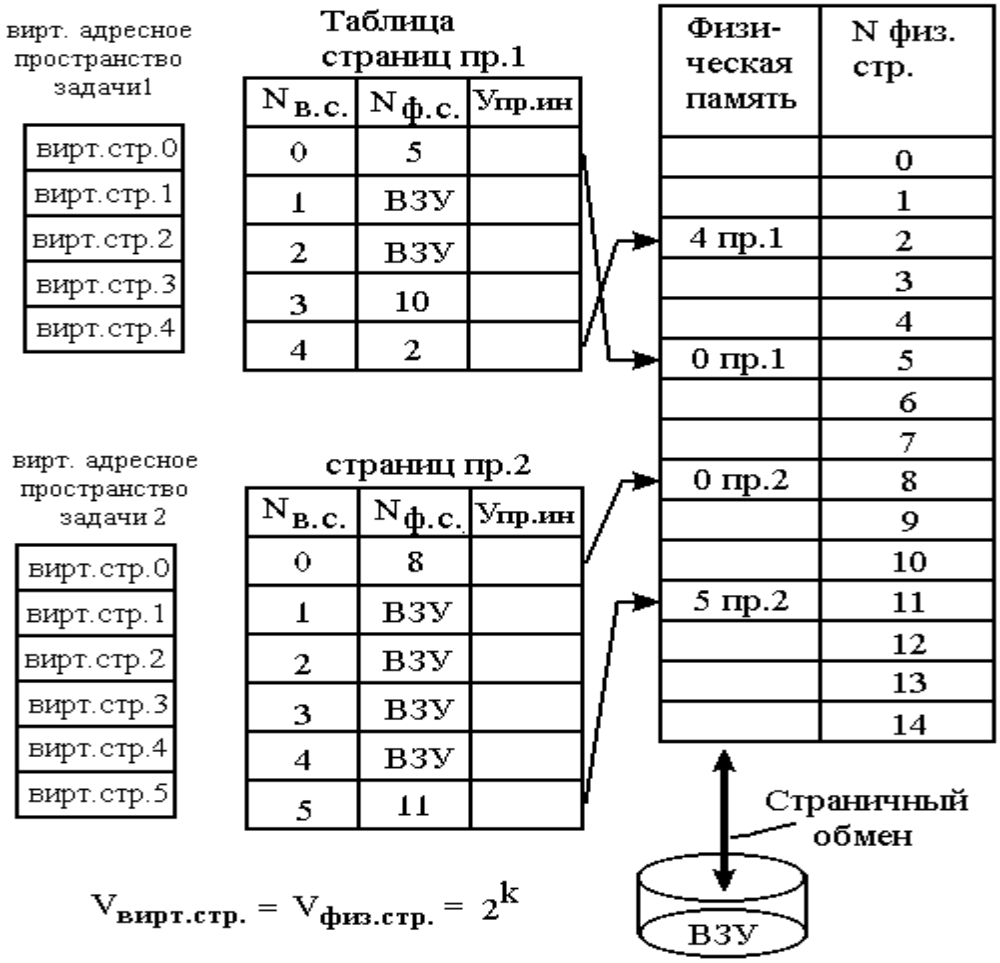


Рис. 5. Страничное распределение памяти

Размер виртуального адресного пространства в общем случае не является кратным размеру страницы, поэтому последняя страница каждого процесса дополняется фиктивной областью. Вся ОП ЭВМ делится на части такого же размера, называемые физическими страницами. Размер страницы обычно выбирается кратным степени двойки, т.к. это позволяет упростить механизм преобразования адресов.

Часть виртуальных страниц процесса при его загрузке помещается в ОП (в свободные физические страницы), а часть на жесткий диск. ОС при загрузке процессора формирует для него таблицу страниц, в которой устанавливается соответствие адреса виртуальной страницы (№в.с.) адресу физической страницы (№ф.с.).

Также в таблице страниц содержится управляющая информация (Упр.ин.[1]): признак присутствия страницы в ОП, признак модификации страницы, признак невыгружаемости (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 создаваемые и используемые механизмом ВП.

При каждом обращении к памяти происходит чтение из таблицы страниц информации о виртуальной странице, к которой произошло обращение. Если данная виртуальная страница находится в ОП, выполняется преобразование виртуального адреса в физический. Если же нужная ВС отгружена на диск, то генерируется страничное прерывание и организуется процедура подкачки. Процесс переводится в состояние ожидания, и активизируется другой процесс, из очереди готовых. Параллельно с этим программа обработки страничного прерывания находит на диске требуемую страницу и пытается загрузить её в ОП.

Если в ОП имеется свободная ФС, то соответствующая ВС подгружается в ОП, если же свободных страниц нет, то реализуется процедура выталкивания из ОП какой-нибудь страницы и перемещение её во внешнее ЗУ. Критерии выбора выталкиваемой ВС могут быть различными (LRU, RND, LFU, FIFO…).

Рассмотрим механизм преобразования виртуального адреса в физический при страничной организации памяти (см. рис. 6).

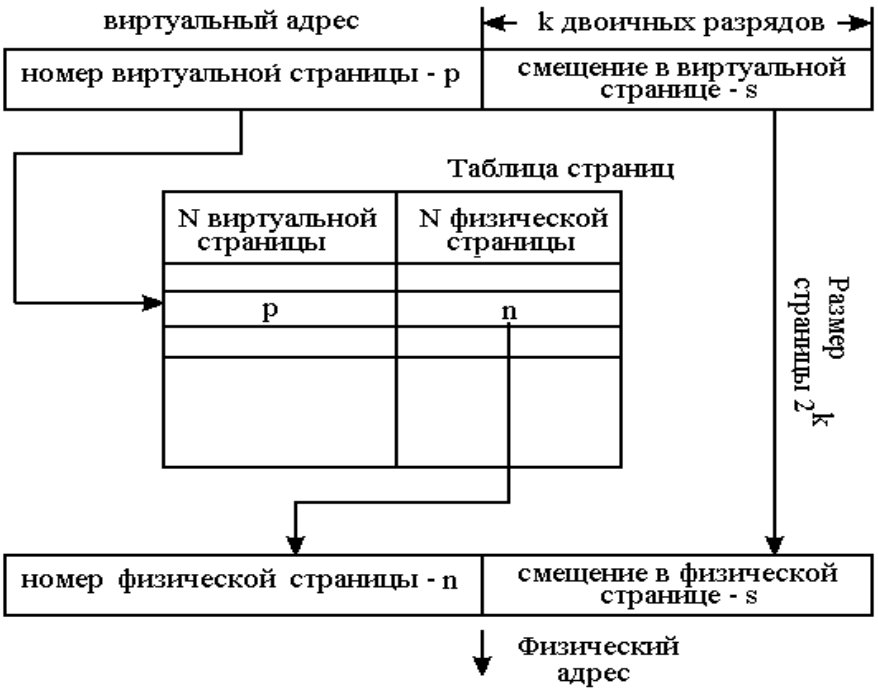


Рис. 6. Механизм преобразования виртуального адреса

При каждом обращении к ОП аппаратными средствами выполняются следующие действия.

[1] – для простоты в ЛР рассматривается «Упр.ин.» состоящая из одного поля «Mem/Disk» – местоположение страницы: в ОП (Memory) или на внешнем ЗУ (Disk).

- На основании начального адреса таблицы страниц, номера виртуальной страницы и длины записи в таблице страниц определяется нужный адрес записи в таблице страниц.
- Из этой записи извлекается номер физической страницы.
- К номеру физической страницы присоединяется смещение, т.е. младшие разряды ВА (путем конкатенации).

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

На производительность системы со страничной организацией памяти влияют временные затраты, связанные с обработкой страничных прерываний и преобразованием виртуального адреса в физический. Чем чаще возникает страничное прерывание, тем больше времени тратится на перемещение страниц. Чтобы уменьшить частоту страничных прерываний, нужно увеличить размер страницы. Увеличение размера страницы уменьшает размер таблицы страниц, а значит, и уменьшает затраты памяти. Но, с другой стороны, чем больше размер виртуальной страницы, тем больше памяти занимает фиктивная область в конце последней виртуальной страницы каждой программы.

Время преобразования ВА в ФА в значительной степени определяется временем доступа к таблице страниц. Поэтому таблицу страниц следовало бы размещать в «быстрых» ЗУ. Это может быть набор специальных регистров или буферная память, использующая ассоциативный поиск и кэширование данных. На самом деле размер ОП/ВП современных ВС достаточно велик, а значит и таблицы страниц будут очень большими, что не дает возможности разместить их в быстрых ЗУ целиком. Однако предусмотрена возможность хранения быстрых ссылок (пар №в.с. – №ф.с.). в специальных КЭШ буферах – TLB.

2.4. *Способы подкачки страниц*

Предварительное размещение. Все страницы, требующиеся данному процессу, предварительно размещаются в оперативной памяти, если размер свободной области ОП это позволяет.

Преимущества: Свободную память можно использовать без страничных прерываний.

Недостатки: снижается коэффициент мультипрограммирования (количество одновременно выполняемых процессов), т.к. каждая задача занимает много места, и в ОП может быть помещено меньшее количество задач. Стратегия не подходит для больших по объему задач.

Опережающая подкачка. При опережающей подкачке операционная система пытается заблаговременно предсказать, какие страницы потребуются процессу, а затем, когда в основной памяти появляется свободное место, загружает в нее эти страницы. Пока задача работает со своими текущими страницами, система запрашивает новые страницы, которые будут уже готовы к использованию, когда процесс к ним обратится. Если решения о выборе страниц для подкачки принимаются правильно, то удается значительно

сократить общее время выполнения данного процесса.

Преимущества: Если в большинстве случаев удается принимать правильное решение о выборе страниц для подкачки, то время выполнения процесса значительно уменьшается.

Недостатки: Этот метод эффективен для программ с небольшим количеством переходов, поскольку в этом случае с большой вероятностью, следующей понадобится страница с порядковым виртуальным номером на 1 больше, чем у текущей. Такое предсказание сделать легко.

Подкачка по требованию. Считается, что наиболее рационально загружать в ОП страницы, необходимые для работы процесса, по его запросу. Не следует переписывать из внешней памяти в основную ни одной страницы до тех пор, пока к ней явно не обратится выполняющийся процесс.

Преимущества: Так как путь, который выберет программа при своем выполнении, точно предсказать невозможно, то любая попытка заранее загрузить страницы в память в предвидении того, что они потребуются в работе, может оказаться неудачной. Будут загружены не те страницы, и выполнена лишняя работа. А подкачка страниц по запросу гарантирует, что в основную память будут переписываться только те страницы, которые фактически необходимы для работы процессов.

Недостатки: *Процесс должен накапливать в памяти, требуемые ему страницы по одной. При появлении ссылки на каждую новую страницу процессу приходится ждать, когда эта страница будет передана в основную память.*

2.5. *Алгоритмы замещения блоков (строк/страниц)*

При замещении какого-либо блока из памяти нижнего уровня иерархии блоком из верхнего уровня для освобождения места должен использоваться принцип оптимальности. Он говорит о том, что для обеспечения оптимальных скоростных характеристик и эффективного использования ресурсов следует заменять тот блок, к которому в дальнейшем не будет новых обращений в течение наиболее длительного времени. Такой алгоритм невозможно реализовать, т.к. нельзя предсказать будущее. Однако были разработаны множество алгоритмов замещения, отличающиеся назначением, принципом работы, скоростными характеристиками и сложностью реализации. В настоящее время существует большое многообразие алгоритмов замещения и их модификаций, направленных на повышение производительности ВС: RND, FIFO, LRR, LRU, MRU, LFU, Second-Chance, Clock (GClock), LARC, Two Clock Hand, Clock-Pro, NRU, комбинации LRU и LFU (LRD, FRC, LIRS, LRU-K, 2Q, MQ, FBR, LRFU, SF-LRU, ARC, CAR, FR-Cache. Такое многообразие связано с желанием подобрать алгоритм, наиболее приближенный к оптимальному. Но эффективность алгоритмов зависит от сложности алгоритма, от параметров системы (размер кэш-памяти, размер оперативной памяти, ширину шины, число и тип контроллеров памяти) и параметров программ (объём, тип программного кода). На разных задачах разные алгоритмы демонстрируют наилучшую эффективность.

2.6. *Базовые алгоритмы замещения*

RND выталкивание случайного блока/страницы/строки. Если нужно иметь стратегию выталкивания блока, которая характеризовалась бы малыми издержками и не являлась бы дискриминационной по отношению к каким-либо конкретным пользователям/программам, то можно пойти по очень простому пути, т.е. выбирать случайный блок. В этом случае все блоки, находящиеся в памяти, могут быть выбраны для выталкивания с равной вероятностью, в том числе даже следующий блок, к которому будет производиться обращение (и который, естественно, удалять из памяти наиболее нецелесообразно). Поскольку такая стратегия, по сути, рассчитана на "слепое" везение, то в реальных системах она применяется редко, только когда специальные требования по времени/ресурсам не выставляются и надо сделать что-то быстро и просто.

Достоинство: *простота, малые издержи, равноправие задач.*

Недостатки: *не учитывает приоритетную важность некоторых блоков и особенности задач (последовательности обращения к памяти).*

FIFO выталкивание наиболее раннего записанного в память блока/строки/страницы. Есть два варианта реализации данного алгоритма (см. рис. ниже). В одном из них с каждым блоком/строкой/страницей ассоциирован счетчик. Когда блок заносится в память, его счетчик обнуляется. Через определенные интервалы времени (например, по таймеру или при каждом обращении в память) содержимое всех счетчиков увеличивается на единицу. Следовательно, большее число будет у счетчика блока, который дольше всего находится в памяти, поэтому он является первым кандидатом на замещение.
Еще один способ можно реализовать при помощи очереди (путём ведения связанных списков). В такой список в порядке заполнения блоков памяти (страниц ОП или строк Кэш) заносятся ссылки на эти блоки. С каждым обращением к блоку ссылка переносится в конец списка. Соответственно первая ссылка в очереди каждый раз оказывается та, к которой дольше всего не обращались. Данный блок заменяется прежде всего.

Рис. 7. Стратегия FIFO

При выталкивании страниц по принципу FIFO (First In First Out, т.е. "первый пришел - первый ушел") каждой странице в момент поступления в основную память присваивается временная метка. Когда надо удалить из

памяти какую-нибудь страницу, то выбирать надо ту, которая находится в памяти дольше других, т.е. с самой маленькой временной меткой. К сожалению, стратегия FIFO с достаточно большой вероятностью будет приводить к замещению активно используемых страниц, поскольку тот факт, что страница находится в основной памяти в течение длительного времени, вполне может означать, что она постоянно в работе.

Достоинство: *простота, малые издержи, равноправие задач.*

Недостатки: *рассчитана на последовательную обработку, не подходит для реальных сложных программ, где множество переходов и циклов.*

LRU - выталкивание дольше всего не использовавшейся страницы (Least Recently Used). Один из наиболее распространенных алгоритмов замещения, предложенный одним из первых алгоритмов и до сих пор успешно применяемый на большинстве реальных задач. Он основан на замещение того блока, к которому дольше всего не было обращений.

Рис. 8. Стратегия LRU

Есть два варианта реализации данного алгоритма. В одном из них с каждым блоком ассоциирован счетчик. Через определенные интервалы времени к содержимому всех счетчиков прибавляется единица. Когда блок заноситься в память или когда происходит повторное обращение к блоку, счетчик обнуляется. Следовательно, большее число будет у счетчика того блока, к которому дольше всего не обращались, поэтому он является первым кандидатом на замещение. Еще один способ можно реализовать при помощи очереди (путём ведения связанных списков). В такой список в порядке заполнения блоков памяти (ОП или Кэш) заносятся ссылки на эти блоки. С каждым обращением к блоку ссылка переносится в конец списка. Соответственно первая ссылка в очереди каждый раз оказывается на тот блок, к которому дольше всего не обращались. Он заменяется прежде всего.

Достоинство: *относительная простота, малые издержи, наилучшая эффективность при повторяющихся действиях.*

Недостатки: *рассчитан на короткие циклы (сопоставимые с размером памяти), отсутствие устойчивости к сканированию (при многократном просмотре большого массива данных – большие циклы).*

MRU (Most Recently Used) или Fetch-and-Discard алгоритм похож на LRU, но удаляет блок/строку/страницу, наиболее недавно использованную. Этот алгоритм, в отличии от LRU, считается устойчивым к сканированию из-за стремления сохранять старые данные. В случае, когда происходит периодическое сканирование файла по циклической схеме, данный алгоритм является наилучшим. Как и в предыдущем случае используется единственный критерий выбора строки для замещения – давность обращения к строке.

LFU выталкивания реже всего используемого блока/страницы/строки**.**

Рис. 9. Стратегия LFU

Согласно алгоритму LFU (Least Frequently Used) выталкивается наименее часто (наименее интенсивно) использовавшийся в вычислениях блок/строка/страница. Алгоритм эффективен для программ, в которых относительная частота обращений к блокам не меняется со временем, что нечасто на практике. В противном случае «популярные» в прошлом блоки образуют в памяти балласт, очень негативно влияющий на эффективность процесса буферизации.

Достоинство: *относительная простота, малые издержи.*

Недостатки: *специфичность применения.*

3. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

Для выполнения работы требуется запустить моделирующую программу и задать исходные данные согласно вашему варианту. Таблица вариантов с указанием исходных параметров моделирования приводится ниже (см.табл.1).

Табл.1. Варианты задания

№ по списку	Параметры файла			Параметры памяти	
	Количество виртуальных страниц	Кол-во операций переходов в сегменте команд	Количество доступов к файлам	Кол-во выделенных физических страниц	Размер страницы в килобайтах
1.	8	28	20	5	1
2.	6	20	16	4	2
3.	10	30	20	7	1
4.	9	27	15	6	1
5.	11	35	25	8	1
6.	11	33	21	7	1
7.	6	26	18	5	2

№ по списку	Параметры файла			Параметры памяти	
	Количество виртуальных страниц	Кол-во операций переходов в сегменте команд	Количество доступов к файлам	Кол-во выделенных физических страниц	Размер страницы в килобайтах
8.	6	25	20	4	2
9.	8	15	19	7	1
10.	10	30	16	8	1
11.	5	24	17	4	2
12.	6	26	18	3	2
13.	7	20	19	4	1
14.	8	17	25	7	1
15.	7	26	21	5	2
16.	5	15	15	3	2
17.	7	17	21	6	1
18.	8	16	25	7	1
19.	9	21	21	7	1
20.	7	22	30	6	2
21.	9	16	19	8	1
22.	10	23	17	7	1
23.	10	18	26	6	1
24.	11	16	29	8	1
25.	5	14	12	3	1
26.	6	20	11	4	2
27.	7	16	19	6	1
28.	8	17	22	6	1

Для моделирования выбрана страничная организация памяти и рассмотрены несколько возможных стратегий откачки страниц. Моделируется последовательное выполнение команд задачи, которые условно подразделяются на команды (см. рис. 10):

- простые,
- передачи управления,
- доступа к данным.

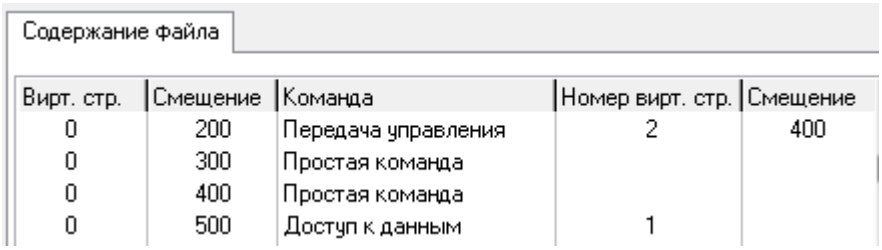


Рис. 10. Содержимое файла задачи: разные типы команд

Их основное отличие заключается в следующем.

При выполнении простой команды происходит однократное обращение к странице ОП, на которой расположена команда, для выборки команды (на рис.10 обращение к странице 1 согласно команде «0 300 Простая команда»).

При выполнении команды передачи управления происходит однократное обращение к странице ОП, на которой расположена команда (на рис.10 обращение к странице 0 согласно команде «0 200 Передача управления»), второе обращение к странице, на которую команда передала управление (к странице 2) будет выполнено при выполнении команды, на которую

управление передано (команда с адресом 400 на рис.10).

При выполнении команды доступа к данным происходит два доступа к ОП, первый – к странице, на которой расположена команда (на рис.10 к странице 0), второй – к странице, на которой расположены данные (на рис. к странице 1). После обращения к странице с данными управление передается опять на страницу с командами (на стр.0) и она будет считаться текущей.

Для простоты моделирования считаем, что каждая команда программы (задачи) расположена относительно предшествующей со смещением в 100 байт.

Для выполнения лабораторной работы требуется:

1. Согласно Вашему № варианта выберите параметры моделирования из табл.1.

2. Запустите на исполнение файл «Progect1».

Перед Вами откроется окно программы. В нём есть меню из пяти опций:

- Инициализация – с выбора этой опции следует начать работу.
- Загрузка – начальная загрузка первых виртуальных страниц задачи в физическую (оперативную) память.
- Выполнение – при выборе этой опции происходит выполнение одной команды задачи.
- Авторы – сведения о создателях программного продукта.
- Выход – окончание работы с программой.

3. Выберите опцию «инициализация», щёлкнув её левой кнопкой мыши. После этого появятся две опции подменю:

- параметры симулятора,
- сброс параметров.

4. Выберите опцию «параметры симулятора». На экране появится ещё одно окно, в котором Вы можете задать свои параметры из таблицы вариантов. Затем нажмите в этом дополнительном окне кнопку «принять». Если Вы хотите вернуться к предыдущему циклу моделирования, то нажмите кнопку «отмена».

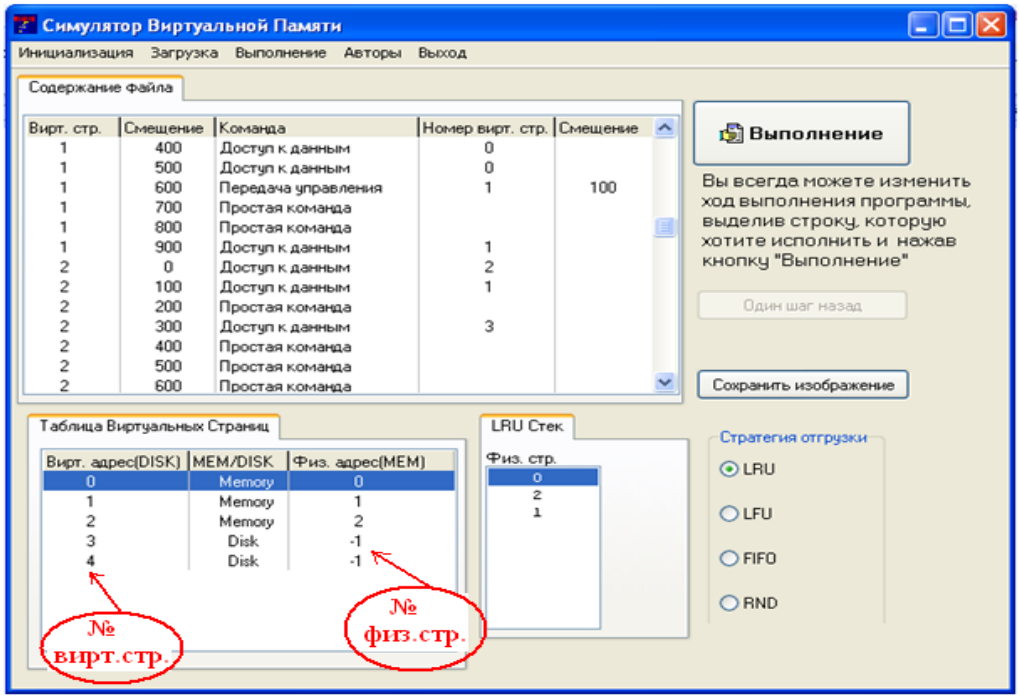
5. После задания параметров моделирования на экране появляется ещё три окна:

- содержание файла,

в нём отражается последовательность команд задачи с указанием № страницы, на которой расположена команда, её адреса на этой странице, типа команды, а также возможно № страницы, к которой обращается команда (для команд доступа к данным) или № страницы и смещения (для команд передачи управления).

- таблица страниц задачи (пока пусто),
- LRU (LFU, FIFO)-стек (пока пусто).

6. Выберите опцию меню «загрузка». После этого в трёх окнах информация изменится следующим образом:



- содержание файла, (не изменится)
- таблица страниц задачи,

в нём отражается текущее состояние задачи: каждая виртуальная страница (первый столбец строки таблицы - № виртуальной страницы) в настоящий момент может присутствовать в ОП (состояние MEMORY во втором столбце) либо быть выгруженной во внешнюю память (состояние DISK во втором столбце). Если страница присутствует в ОП, то в таблицу страниц (в третий столбец) заносится № соответствующей физической страницы.

- LRU (LFU, FIFO)-стек (пока его содержимое не имеет значения).

7.Выберите одну из стратегий откачки (LRU, RND, LFU, FIFO) страниц из ОП в окне «стратегия отгрузки», пометив её мышью.

8.Теперь можно начинать моделирование: для последовательного выполнения каждой команды нажимайте либо кнопку, либо опцию меню «выполнение». После каждого нажатия состояние трёх окон будет изменяться.

- **содержание файла,**

синим цветом выделена команда, которая должна будет выполниться при следующем нажатии кнопки «выполнение».

- **таблица страниц задачи,**

при переходе к новой странице или при операции доступа к данным возможно изменение: новая страница подгрузится в ОП, если её там до этого не было. Это состояние отразится соответствующими изменениями во втором и третьем столбцах.

- **LRU (LFU, FIFO)-стек**

в нём отражается текущее состояние оперативной памяти - последовательность страниц, с которыми работал процессор. Оно будет различным для различных стратегий откачки. При подгрузке новой страницы из ОП будет удалена страница, номер которой расположен внизу стека. На её место заносится подгружаемая виртуальная страница (из внешней памяти).

Рассмотрим подробнее содержимое разных видов стеков.

LRU-стек. На вершине стека указана страница, с которой процессор только,

что работал. По мере удаления от вершины стека указаны страницы, к которым процессор давно не обращался (с момента последнего обращения).

LFU-стек. На вершине стека указана страница, к которой процессор чаще всего обращался. По мере удаления от вершины стека указаны страницы, к которым было меньше обращений в последнее время (с момента загрузки). Кроме № страницы указано количество обращений к ней.

FIFO-стек. На вершине стека указана страница, которая подгружена в память последней. По мере удаления от вершины стека указаны страницы, которые появились в ОП раньше.

Для стратегии **RND** стек не создаётся, поскольку при произвольном выборе удаляемой страницы никакую информацию запоминать и анализировать не нужно.

9. По мере выполнения моделирования необходимо обратить внимание на изменения во всех окнах и объяснить их. Для этого перед и после выполнения команды, изменяющей состояние ОП, сохраните и распечатайте изображение на экране компьютера с помощью кнопки «сохранить изображение». Изменения могут вызвать следующие команды

- любая команда при переходе на новую страницу программы,
- команда доступа к данным,
- команда передачи управления.

Если Вы не успели проследить эти изменения, то можно вернуть состояние моделирующей программы на один шаг назад с помощью кнопки «один шаг назад» и потом продолжить моделирование. Кроме того, можно изменить последовательность выполнения задачи, выделив с помощью мышки любую команду в окне «содержание файла».

4. СОДЕРЖАНИЕ ОТЧЁТА

В отчёт по лабораторной работе следует включить скриншоты состояния системы моделирования для выбранных вами команд (выполнение которых вызовет страничное прерывание и запустит процедуру откачки/подкачки) с видом окон до выполнения команды и после для всех стратегий откачки (LRU, RND, LFU, FIFO) с объяснениями, почему произошло каждое изменение.

5. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое виртуальная память?
2. В чем причина появления понятия вртуальной памяти?
3. Назовите способы реализации ВП?
4. В чём заключается страничная организация памяти?
5. В чём заключается сегментная организация памяти?
6. В чём заключается сегментно-страничная организация памяти?
7. В чём заключается процедура свопинга?
8. Что такое виртуальное адресное пространство задачи?
9. Что такое таблица страниц, её назначение?
10. Какую информацию содержит таблица страниц задачи?

11. Как происходит преобразование виртуального адреса в физический?
12. Какие факторы влияют на производительность системы со страничной организацией памяти?
13. Сравните достоинства и недостатки различных способов управления ВП?
14. Что такое подкачка страниц в ОП, когда она происходит?
15. Что такое откачка страниц из ОП, когда она происходит?
16. Какие существуют методы подкачки страниц?
17. Какие существуют методы откачки страниц?
18. Для чего нужно постоянно производить страничный обмен между оперативной и внешней памятью?
19. Что такое LRU (LFU, FIFO)-стек? Для чего он нужен? В чём их отличия?
20. Исполнение каких типов команд моделируется в лабораторной работе? В чём их различия с точки зрения страничного управления памятью?