УДК 621.396.664

А. Е. ДЮСЕМБАЕВ, Е. Т. РАМАЗАНОВ

О НЕКОТОРЫХ АСПЕКТАХ МОДЕЛИРОВАНИЯ СТРАНИЧНОЙ СИСТЕМЫ НА ОСНОВЕ ИНВЕРТИРОВАННОЙ ТАБЛИЦЕЙ СТРАНИЦ

Работа посвящена задаче моделирования систем с виртуальной памятью со страничной и/или сегментно-страничной организацией на основе инвертированной таблицы страниц (IPT). Построен алгоритм, управления вычислительным процессом по преобразованию виртуального адреса в физический на основе IPT, который относится к алгоритмам нечисленного типа

Одной из интересных задач управления вычислительным процессом является задача повышения производительности и оптимизации систем с виртуальной памятью [1-4]. Дополнив структуру [1], некоторыми деталями, отметим, что в системах со страничной или сегментно-страничной организацией одним из ключевых инструментов управления является таблица страниц (ТС), отражающая обмен между основной (ОП) и вспомогательной памятью (ВП). Схематически взаимодействие компонентов, речь идет о TLB буфере, ТС, ЦП, ОП, ВП-вовлеченных вычислительный процесс, который кроме прочего осуществляет и преобразование виртуального адреса, можно представить себе так

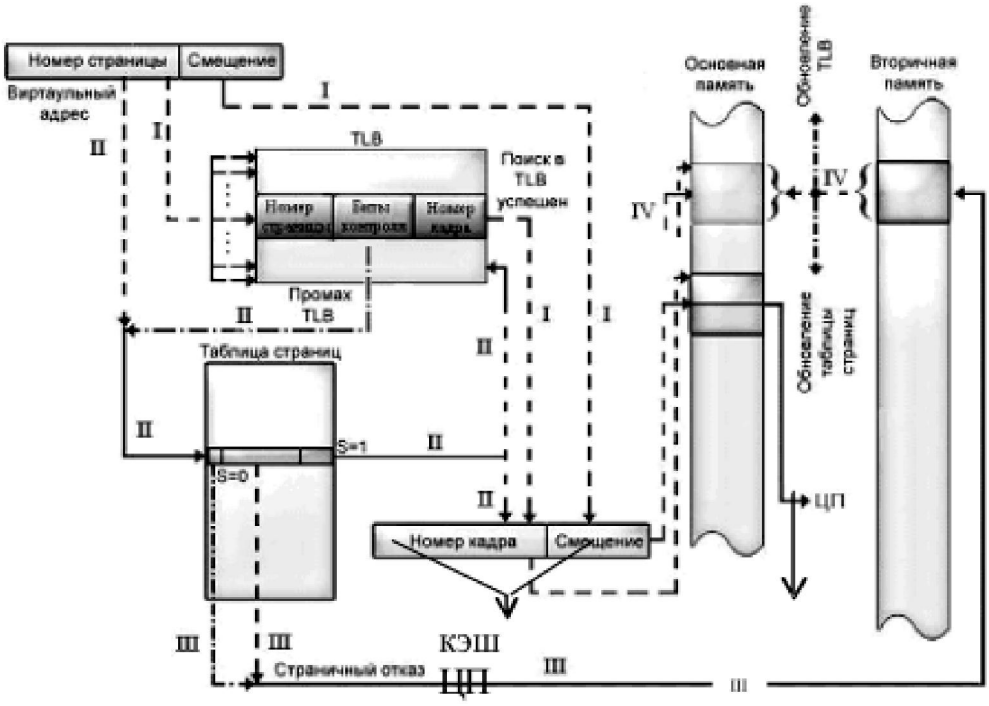

Рис. 1. Преобразование виртуального адреса

Здесь пунктирная линия означает информационную, а штрих-пунктирная линия обозначает управляющую связи между компонентами из рис. 1. Сплошная линия это конъюнкция управляющей и информационной связи. Римскими цифрами I, II, III, IV обозначены номера процессов. Заметим, что цель здесь – определить номер физического кадра, который соответствует заданной виртуальной странице, если таковой существует. Наиболее быстро этот кадр может определиться в результате выполнения процесса I (TLB hit). В случае промаха TLB (TLB-mis),

конъюнкция управляющего и информационного сигнала инициирует процесс II. И здесь в случае успеха пересылка номера страничного кадра из таблицы страниц на регистр адреса завершает процесс II. В случае страничного отказа наступает очередь процесса III, который, в свою очередь, инициирует процесс IV, в результате исполнения которого нужная страница из вспомогательной памяти загрузится в соответствующий кадр ОП и одновременно обновятся TLB-буфер и ТС. Напомним, что каждая строка TLB бу-фера содержит: *номер виртуальной страницы, биты контроля, номер кадра.* Строки *TLB*, как видим, по схеме на рис. 1, в отличии от ТС, проверяется в параллельном режиме.

Заметим, что обычная таблица страниц является весьма неэффективной структурой данных для поддержки вычислительного процесса, так как содержит информацию о неактивных страницах, что можно проследить уже по системе с 32-разрядным логическим адресом (Pentium или Power PC), где на номер виртуальной страницы отводится 20 или 40 бит соответственно (рис. 2).

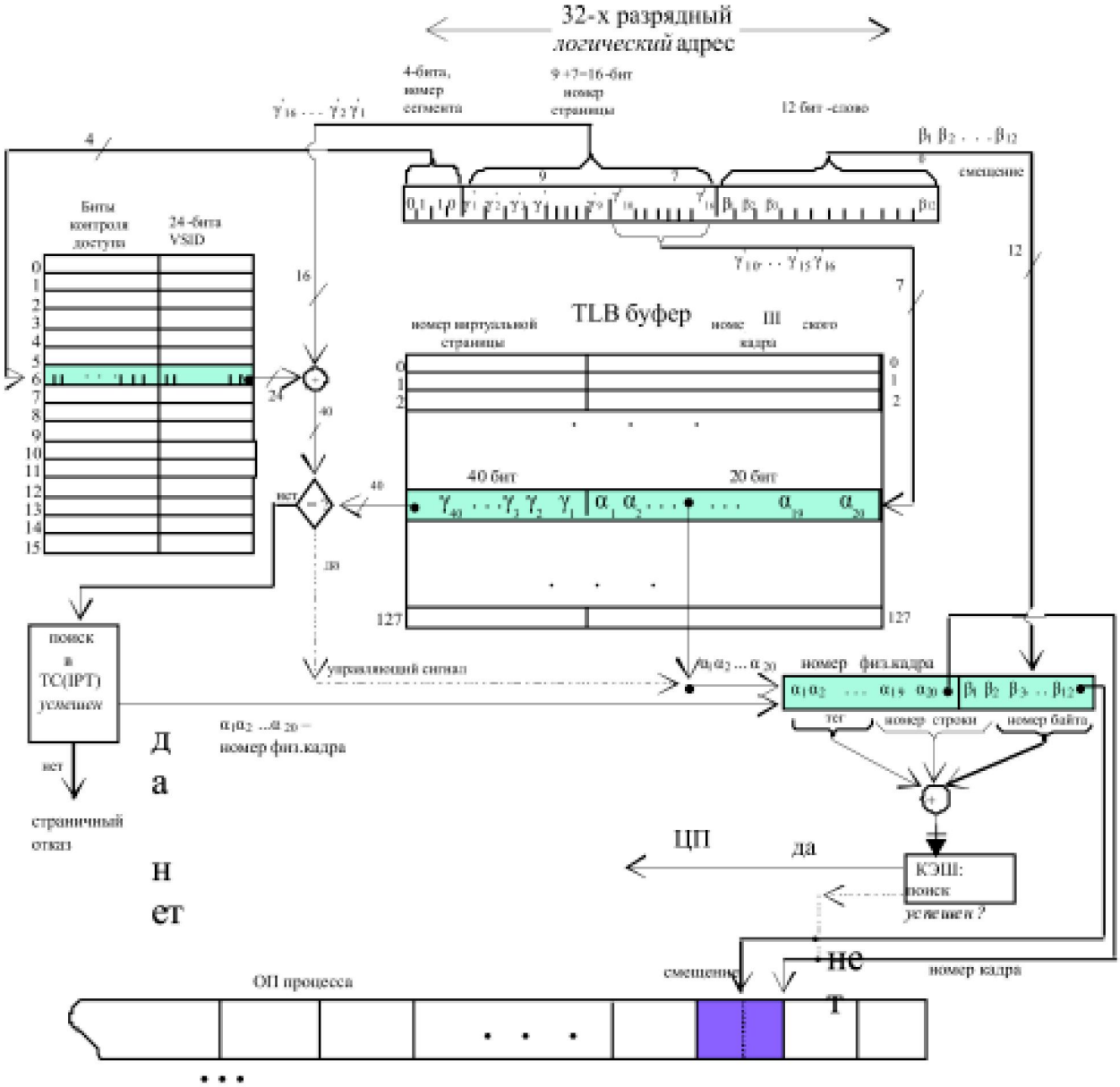

Рис. 2. Преобразование виртуального адреса в физический в системе с сегментно-страничной организацией с 32 разрядным логическим адресом

Для схемы на рис. 1 (без учета КЭШа) можно предложить модель, отражающую особенности взаимодействия ЦП, ОП, ВП, TLB буфера, ТС, где в качестве ТС может выступать и IPT, а именно граф-схему (рис. 3), которая, по-сути, является автоматной моделью рассматриваемого вычислительного процесса

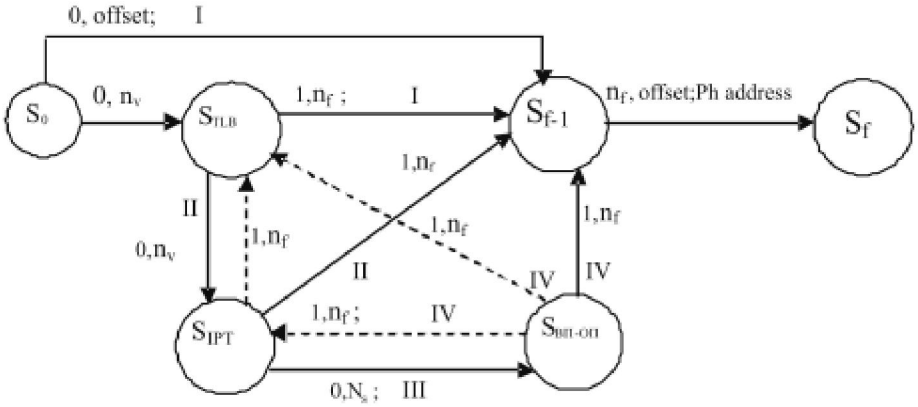

Рис. 3

где S_0 – это начальное состояние процесса, S_{TLB} – это состояние вычислительного процесса, определяемого TLB буфером, S_{IPT} – состояние вычислительного процесса определяемого цепочкой: *промах* TLB буфера → таблица страниц (IPT). Состояние $S_{ВП–ОП}$ это состояние, задает цепочку:

*промах*TLBбуфера → *отказ*Таб.стр.(IPT) →
→ (ВП→ОП, *обновление*(TLB, IPT)) → S_{f-1} .

Штрих-дуги ведущие от вершины $S_{ВП–ОП}$ к вершинам S_{TLB} и S_{IPT}, это информационные связи, по-существу, означают обновление, как битов контроля, так и соответствующих строк TLB буфера и таблицы страниц (процесс IV). Подобное обновление показывает и штрих-дуга, направленная от состояния S_{TL} к состоянию S_{TLB} (процесс II). Состояние S_{f-1} – это состояние предшествующее заключительному состоянию процесса, соответствует регистру адреса R на рис. 1, сопровождается выдачей физического адреса по исходному виртуальному адресу. Другие обозначения: n_v – это номер виртуальной страницы, n_f – номер соответствующей физической страницы (кадра), N_s – адрес физической памяти, (например, на диске) по которому располагается физическая страница с виртуальным номером n_v. Бит контроля, точнее *бит присутствия* со значениями 0(отсутствует) или 1(присутствует), также отражен на схеме, как 1-я компонента *веса дуги*. Таким образом, начиная с вершины S_0, «вес» дуги (*а; в*): *а* – бит присутствия, значение которого фактически или формально получается в результате прохождения данной вершины. Вторая компонента *в* – это информация, ставшая доступной процессу в результате прохождения процессом через вершину, из которой исходит данная дуга и одновременно, это аргумент вершины, в которую входит данная дуга. Заметим, что именно, путь, начинающийся в S_0 и завершающийся в состоянии S_f, показывает развитие вычислительного процесса на данной граф-схеме. Узким местом моделей основанных на рис. 1, является наличие обычной ТС. При решении задачи оптимизации вычислительного процесса, в рассматриваемой проекции среди разработанных подходов [2], наиболее перспективным, наряду с двухуровневой таблицей, является использование IPT. Поддерживая подобную структуру, а речь здесь идет об IPT, система, в любой момент вычислительного процесса в основной памяти сохраняет, в таблице IPT, лишь информацию об активных страницах программы, т.е. тех страницах программы, которые находятся в ОП. Кроме самой таблицы IPT в основной памяти хранится и хеш таблица, длина которой совпадает с длиной таблицы IPT. В качестве хеш функции мы возьмем хеш функцию классического типа: n**mod**m. И именно о таком варианте хеш-функции для построения IPT упоминается в фундаментальной книге [2]. Однако, в [2] и в других источниках, например, в хрестоматийной книге[1] ничего не говорится

о том, как реально можно организовать вычислительный процесс, используя IPT, с функцией n **mod** m.

Данная публикация посвящена моделированию элементов вычислительного процесса и определению особенностей построения алгоритма, включая определение основных структур данных, поддерживающего функционирование системы с таблицей IPT. Построение такого алгоритма, алгоритма не численного типа [3], нужно начать с определения структуры данных, с которой должен оперировать проектируемый, управляющий алгоритм. Следуя логике закрытой адресации, с хеш функцией, типа n mod m при построении такой таблицы не должны образовываться списки с избыточным количеством звеньев. Как правило, предпочтение отдают системам с небольшой длиной списка, 1-2 звена на один список, хотя имеются разработки со сравнительно большим количеством звеньев(Power PC601). Соблюдая, стратегию закрытой адресации, строка инвертированной таблицы страниц, на которую указывает ссылка из хеш-таблицы, является началом списка номеров виртуальных страниц, имеющих одно и то же значение хеш-функции. Предпочтение и одновременно наша цель, построить инвертированную таблицу, в которой одно-два звена на один список. При построении мы будем идти на то, чтобы на начальном этапе вычислительного процесса допускать списки длины большей, чем 2 (рис. 4).

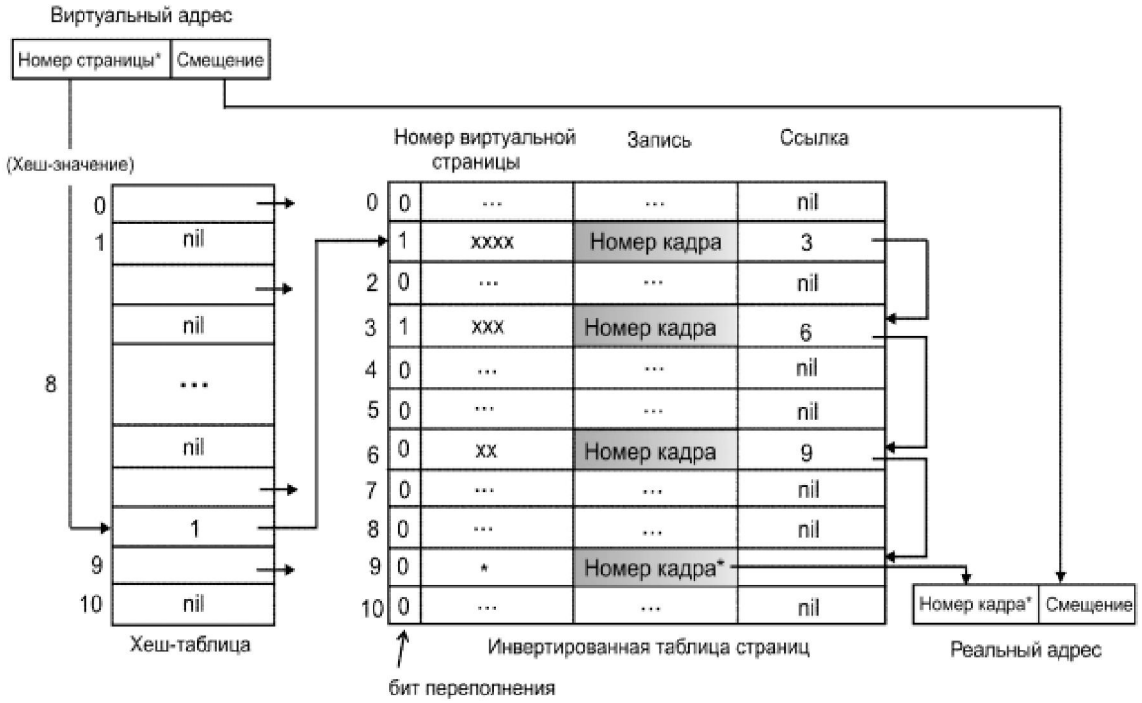

Рис. 4

Здесь по мере поступления альтернативных номеров виртуальных страниц потребуется незначительная перестройка IPT и списков длины больше 2, что несколько замедлит работу системы. Однако в случае, когда в IPT не будет списков длины больше 2 и сам алгоритм преобразования да и вся система будет работать более эффективно. Значение хеш-функции указывает на строку IPT, где, как мы видим, (рис. 4), возможно, хранится информация о виртуальной странице с данным номером и её расположении в ОП (номер кадра). На начальном этапе, до полного заполнения инвертированной таблицы, понадобится регистр R, а по существу процедура-функция с тем же названием. Содержимое этого регистра на начальном этапе указывает на первую по счету свободную строку таблицы IPT. В этом случае содержимое регистра R одновременно является и номером свободного физического кадра ОП, среди последовательности идущих подряд кадров, выделенных системой для исполнения программы-процесса. Кроме того, на начальном

этапе важную роль играет содержимое специального бита-бита переполнения, показывающего (равен 1) наличие переполнения списка, голова, а скорее остаток, которого начинается в данной строке. Этот бит, расположен, в каждой строке таблицы страниц. Таким образом, если для некоторой строки IPT значение бита переполнения равно 1, то длина списка, начинающегося в этой строке, строго больше 2, в противном случае бит переполнения равен 0. Очевидно, что последние два звена любого списка, если, конечно, они имеются, или единственное звено списка, имеют бит переполнения – 0. Кроме того, важно помнить, что, в реальности, каждая строка инвертированной таблицы, среди битов контроля, должна содержать и бит модификации, переустанавливаемый из 0 в 1, если в ходе вычислительного процесса какой либо фрагмент программы, расположенной на данной странице, изменился. В соответствии с выбранной стратегией, будем допускать, в особенности, на начальном этапе вычислительного процесса, списки длины больше 2, стремясь всё же уменьшить длины этих списков по ходу развития вычислительного процесса. Предположим, что IPT состоит из 11 строк (рис. 4, m = 11), т.е. для исполнения программы в ОП выделено 11 страниц. Рассмотрим ситуацию, когда все строки IPT уже заняты (R = nil) и имеется лишь один список, занимающий более двух звеньев. Биты переполнения равны 1 лишь у двух звеньев этого списка, сигнализируя о длине оставшейся части списка. Голова этого списка находится в строке IPT с номером 1. Пусть последнее обращение произошло к виртуальной странице с номером N: N mod 11=8. Ситуация не привела к страничному отказу, так как после по-следовательного просмотра звеньев списка, начиная с головы списка(строка с номером 1), номер искомой страницы оказался в конце этого списка, в строке IPT с номером 9. Тот же результат, в смысле страничного отказа, наблюдался бы, если рассматриваемый виртуальный номер был бы записан в поле номера любого звена данного списка, т.е. в строке с номером 1или 3 или 6. Посмотрим, к чему ведёт ситуация, если следующее обращение произойдет к странице с номером N′: N′mod 11=4 (рис. 5) и поле хеш-таблицы с номером 4 имеется запись nil, что сразу же сигнализирует о страничном отказе

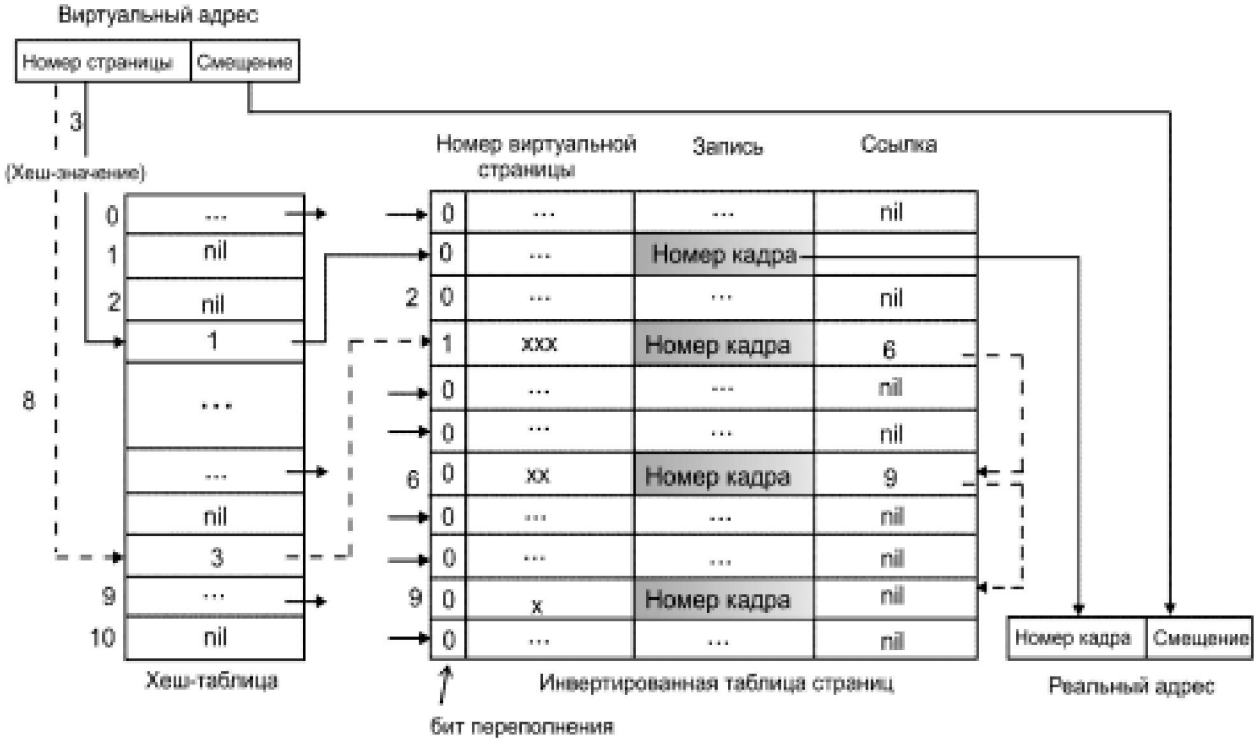

Рис. 5

и о том, что в настоящий момент, длина соответствующего списка равна 0. При этом, считаем, что список свободных страниц(физических кадров) уже пуст, т.е. R=0. Помним, что используемый нами локальный вариант неклассической стратегии замещения страниц, относится к стратегиям замещения с постоянным числом страниц ОП, которые образуют некоторый непрерывный участок последовательных страниц ОП. Далее, на текущем этапе, должен использоваться системный алгоритм(процедура Find), который ищет список, если таковой вообще найдется, бит переполнения которого равен 1 или хотя бы список, состоящий из 2-х звеньев. Если в IPT, на данный момент, нет ни одного списка с битом переполнения, равным 1 и более того, отсутствуют списки длины 2, то процедура Find, возвращает 0-ю строку IPT (помним, что R=0). Далее алгоритм поддерживая эту привлекательную ситуацию, должен действовать уже по-иному. В нашем же случае (рис. 4), список с битом переполнения равным 1, нашелся и модификации подверглось 1-ое звено этого списка, в результате, очевидно, на одно звено сократится и сам найденный список. При этом, возможно, что разброс значений номеров виртуальных страниц, к которым происходят последовательные обращения, приведёт в конце концов к желаемой ситуации, о которой говорилось выше. Более того, если однажды, процесс придёт в состояние, когда каждый список содержит только одно звено, алгоритм, поддерживая это состояние, уже не даст процессу выйти из него. С другой стороны, если же R 0, то длина, текущего списка, может возрастать за счёт свободных ячеек IPT, которые и будут образовывать новые звенья этого списка. Возьмем за основу структуры данных для таблицы IPT, структуру каждая строка которой:

IPT[i]:	bit_overfl	n_v	n_f	ref

где поле bit_ovfl – это поле бита переполнения; поле n_v – это поле номера виртуальной страницы; поле n_f - это поле номера страничного кадра виртуальной страницы с номером n_v; поле ref – это поле ссылки на следующее звено, для заключительного звена списка равно nil. Имеется также и хеш-таблица Hash_Tab, в строке которой лишь одно поле, в которое записывается ссылка на строку таблицы IPT, дающей начало списку страниц, виртуальные номера которых имеют одинаковый остаток от деления на длину таблицы m. Строка хеш-таблицы, соответствующая номеру n виртуальной страницы – это n **mod** m, а содержимое строки хеш-таблицы с таким номером это Hash_Tab[n**mod**m]. Итак, работу алгоритма по преобразованию виртуального адреса в физический адрес и поддерживающего IPT, можно представить себе в следующем виде:

Вход: номер виртуальной страницы n; i ← n **mod** m;

1. Hash_Tab[i]=nil; // страничный отказ, список пуст, открываем новый список //

1. a) R≠nil; // в IPT имеется свободная строка и кадр в ОП с тем же номером R, открываем новый список, голова которого будет в кадре с номером R, заполняем строку IPT //

1. b) R=nil; **Find**; // все строки IPT и кадры в ОП заняты, в IPT надо определить строку (процедура Find), точнее голову длинного списка, (если таковой найдётся), и ей соответствующий, занятый кадр в ОП, куда размещать новую страницу с номером n. Если длинного списка в IPT нет, то с помощью **Find** будем искать голову списка двумя звеньями. При перечисленных выше условиях-1), 1.b) такой список обязательно найдется. Требуется переключение ссылки(ок), чтобы образовать новую голову у старого списка, голова же старого списка была использована для нового списка.

2. Hash_Tab[i]≠nil; // к строке i хеш-таблицы были обращения, список не пуст, возможен страничный отказ; Надо проверить список(процедура **Search**), что бы определить будет ли отказ и если «*да*», то вставить новое звено в конец списка либо заменой хвоста(без добавления звена, R=0) либо добавлением еще одного звена (R≠0) к уже существующему списку, вдоль которого осуществлялся поиск //

Процедура функция **Search**; // проверяет есть ли номер n среди номеров виртуальных страниц, в каком либо из звеньев, исследуемого списка //

if Search=0 then exit; // да имеется номер n среди номеров списка, страничного отказа нет, выходим из основной процедуры //

else // страничный отказ, надо проверить, есть ли свободная строка и кадр в ОП //

2.1. R≠nil //есть свободная строка IPT и кадр ОП, цепляем новое звено к хвосту уже существующего списка, делаем необходимые записи в новой строке IPT //

2.2. R=nil //*свободных кадров в ОП нет. Выбираем между вариантами: если свой список содержит по крайней мере три звена, то обязательно заменяем его последнее звено не давая списку расти. Если свой список короткий, а в IPT найдётся список с числом звеньев больше (>) 2, то используем голову этого списка, цепляем новое звено к уже существующему короткому списку. Если в IPT нет длинных списков, все списки короткие, в том числе и просмотренный, то заменяем хвостовое звено своего списка, делаем записи в строке IPT //

Выход: номер физического кадра, куда только что размещена или где уже размещается виртуальная страница с номером n.

В минимальной конфигурации вспомогательными процедурами, используемыми в ходе работы алгоритма по преобразованию виртуального адреса в физический адрес, являются:

процедура *Find*, которая просматривает строки IPT и ищет среди них такую, в которой бит переполнения равен 1 или, если нет такой строки ищет список, у которого два звена, если нет и такого, то процедура Find возвращает 0-ю строку;

процедура *R* – эта процедура используется на начальном этапе и возвращает номер свободного кадра ОП, если все кадры ОП уже заняты процедура *R* возвращает nil; процедура *Search* эта процедура просматривает свой список строк IPT и если находит страницу с заданным виртуальным номером, то возвращает 0, в противном случае возвращает 1, что означает страничный отказ;

процедура *Load*, которая загружает виртуальную страницу с номером n в физический кадр с номером *R*. Здесь могут быть использованы также и другие системные процедуры, например, процедура *Copy(x),* которая учитывает возможность изменения содержимого страницы в ходе вычислительного процесса. Процедура *Copy(x)*, подразумевает создание копии текущего содержимого страницы с номером x на внешнем носителе. На особенностях и этой процедуры и других возможных процедур мы останавливаться не будем.

Следует сказать и о битах контроля. В нашем случае среди битов контроля рассматривался лишь один бит – бит переполнения, однако количество таких битов должно быть несколько больше и включать такие биты как бит модификации, бит защиты страницы(сегмента), а также и другие биты. Не обсуждая деталей скажем лишь, что в принципе бит переполнения может быть переустановлен так , что длинными списками могут считаться списки, содержащие, как минимум $k+1 \leq m$ звено, короткими же будут списки, содержащие k звеньев, тогда крайним случаем будет случай k=m-1 (длинных списков нет).

ЛИТЕРАТУРА

1. *William Stallings* Computer Organization and architecture. USA: Prentice hall six-th edition, 2003. 744 p.

2. *Таненбаум Э.* Архитектура компьютеров. М: Изд. ПИТЕР, 2003. 698 с.

3. *Кнут Д.* Искусство программирования. Получисленные алгоритмы. Т. 1. М.: Изд.Вильямс, 2002. 437 с.

4. *Дюсембаев А.Е.* Математические модели сегментации программ. М.: Физматлит (МАИК, Наука), 2001. 208с.

Резюмн

Виртуальді жадылы жүйелерде есептеу процесін модельдеу есебі қарастырылған. Виртуальді жадылы жүйелерде есептеу процесін басқару моделі инвертирленген беттер кестесі негізінде ұйымдастырылған. Инвертирленген беттер кестесі негізінде есептеу процесінің виртуальді адресті физикалық адреске алмастырудың сандық емес алгоритмі құрылды.

Summary

Work is devoted a problem of modeling of systems with virtual memory with page and/or segment - page organization on the basis of the inverted table of pages (IPT). The algorithm, managements of computing process on transformation of the virtual address in physical one on the basis of IPT is constructed. The constructed algorithm of management of computing process belong to algorithms not numerical type

КазНУ им. аль-Фараби,
г. Алматы

Поступила 19.09.09г.