МИНИСТЕРСТВО ВЫСШЕГО И СРЕДНЕГО
СПЕЦИАЛЬНОГО ОБРАЗОВАНИЯ РСФСР

КУЙБЫШЕВСКИЙ ордена ТРУДОВОГО КРАСНОГО ЗНАМЕНИ
АВИАЦИОННЫЙ ИНСТИТУТ имени АКАДЕМИКА С. П. КОРОЛЕВА

ИССЛЕДОВАНИЕ АЛГОРИТМОВ ЗАМЕЩЕНИЯ

КУЙБЫШЕВ 1989

Министерство высшего и среднего специального образования
Р С Ф С Р

Куйбышевский ордена Трудового Красного Знамени авиационный
институт имени акад.С.П.Королева

ИССЛЕДОВАНИЕ АЛГОРИТМОВ ЗАМЕЩЕНИЯ

У т в е р ж д е н о
редакционно-издательским
советом института
в качестве
методических указаний
к лабораторной работе

Куйбышев 1989

Составитель С.В.С м и р н о в

УДК 68I.I42.2

Исследование алгоритмов замещения:Метод.указ./С.В.Смирнов; Куйб.авиац.ин-т. Куйбышев, I989. 24 с.

Приводятся краткие теоретические сведения о страничной организации виртуальной памяти вычислительных систем и алгоритмах замещения страниц в оперативной памяти. Предлагаются формализмы, достаточные для описания и исследования широкого класса алгоритмов замещения. Даются рекомендации по разработке имитационных моделей для исследования алгоритмов замещения. Устанавливается порядок выполнения лабораторной работы и содержание отчета.

Методические указания к лабораторной работе "Исследование алгоритмов замещения" курса "Теория вычислительных систем" предназначены для студентов факультета системотехники Куйбышевского авиационного института.

Рецензенты: И.П. Т о к а р е в, С.В. С у х а н о в

Ц е л ь р а б о т ы – ознакомиться с алгоритмами замещения страниц в виртуальной памяти ЭВМ и получить практический опыт исследования вычислительных систем методом имитационного моделирования.

ТЕОРЕТИЧЕСКИЕ ОСНОВЫ РАБОТЫ

Основная, или оперативная память (ОП),т.е.память,к которой может быть организован прямой доступ центрального процессора за данными и командами,часто является критическим и ограничивающим ресурсом в вычислительной системе.Поэтому операционные системы ЭВМ зачастую реализуют дисциплины управления памятью,позволяющие использовать внешнюю память как расширение оперативной,создавая тем самым видимость того, что объем оперативной памяти больше фактически существующего.Наличие такого виртуального ресурса памяти существенно облегчает программирование,избавляя пользователя от учета действительно доступной ОП.Кроме того,при этом обычно достигается более эффективное использование фактических ресурсов памяти,но,как правило,за счет снижения скорости выполнения программ [1,2].

Страничная организация – это один из методов реализации виртуальной памяти. Упорядоченное множество адресов ячеек воображаемой памяти, или виртуальное адресное пространство, разбивается на непрерывные части равного размера – виртуальные страницы (размер страниц на современных ЭВМ колеблется от 1 до 4 К байт). Когда программа "загружается" в таким образом организованное виртуальное пространство, ее собственное адресное пространство (т.е. совокупность адресов ячеек, занимаемых командами и данными программы) автоматически делится на части, соответствующие виртуальным страницам – страницы программы. Оперативная память аналогично разделяется на физические страницы, или блоки, имеющие тот же размер, что и их виртуальные двойники.

Программно-аппаратные средства, реализующие страничную организацию, позволяют помещать любую страницу программы в любой блок ОП, причем так, что логически страницы остаются смежными, в то время как блоки не обязательно расположены подряд [1,2].

В системах с виртуальной памятью в ОП обычно загружена лишь часть адресного пространства программы. Применительно к страничной организации это означает, что ряд страниц программы размещается во внешней памяти, которая в данном случае служит расширением ОП. (Для этой цели наиболее подходящими устройствами памяти являются те, которые не требуют времени для подвода головки: например, магнитные бара-

баны или логически эквивалентные им диски с фиксированными головками. Менее эффективно использование в качестве страничных устройств – СУ, дисков с подвижными головками). Обращение программы к отсутствующей в ОП странице (точнее к ячейке адресного пространства программы, находящейся в отсутствующей в ОП странице) вызывает страничный сбой и перенос в ОП со страничного устройства целой страницы, содержащей ячейку, на которую была ссылка. При этом, если количество выделенных программе блоков ОП фиксированно, одна из находившихся в ОП страниц программы должна быть вытолкнута на СУ. Выбор такой страницы проблематичен и производится операционной системой в соответствии с алгоритмом замещения.

Частота страничных сбоев во время выполнения программы зависит как от поведения самой программы, так и от используемого алгоритма замещения, формирующего состав страниц в ОП. В свою очередь частота страничных сбоев определяет быстродействие виртуальной памяти в целом, поскольку время необходимое для переноса страницы со вращающегося СУ в ОП на несколько порядков больше, чем время доступа к ячейке ОП. Таким образом, частота страничных сбоев является важнейшим критерием качества страничной виртуальной памяти в целом и критерием качества алгоритма замещения в частности.

Известно [2], что минимальную частоту страничных сбоев давал бы алгоритм, замещающий те страницы в ОП, обращение к которым в будущем произойдет через максимально долгое время. Однако последовательность обращений программы заранее неизвестна. Поэтому практические алгоритмы замещения стремятся предсказать будущие обращения программы в предположении, что прошлая история обращений будет повторяться в ближайшем будущем. Задача заключается в нахождении возможно простых алгоритмов, которые дают приемлемую частоту страничных сбоев для самых различных программ, поведение которых заранее неизвестно.

Первая схема, которую можно рассмотреть – случайное замещение: замещаемая страница определяется случайным образом, с помощью датчика случайных чисел. Эта схема может быть полезна, если вышеупомянутое предположение не имеет силы (по крайней мере статистически) или как средство сравнения с другими методами.

Алгоритм, реализующий принцип "раньше пришла – раньше ушла" (РПРУ) замещает страницу, которая дольше всех находилась в ОП. Этот метод предполагает, что страницы, дольше всего находящиеся в памяти, будут с наименьшей вероятностью работать в будущем.

Третья основная схема выполняет замещение в ОП наиболее давно использованной страницы (НДИ). Доводом в пользу такого решения является то, что страницы, к которым не обращались относительно долго, вероятно не потребуются в ближайшем будущем.

Для исследования проблемы замещения формализуем некоторые из введенных выше понятий [3].

Пусть $\{1,2,\dots,n\}$ совокупность номеров страниц данной программы, которой в ОП отводится фиксированное число блоков, равное $m<n$.

Поведение программы опишем последовательностью страниц $x_1,x_2,\dots$, к которым происходят обращения в процессе ее выполнения. Если $x_t=i$, то это значит, что обращение к странице i происходит в момент t.

Реализация как РПРУ, так и НДИ требует, чтобы номера страниц, находящихся в ОП, в каждый момент времени t составляли упорядоченный списокок B_t. Такой список, отражающий историю и текущее состояние ОП, будем называть списком состояния ОП.

Например, если $B_{t-1}=(i_1,i_2,\dots,i_m)$ есть состояние ОП в момент времени $t-1$, то для РПРУ страница i_m поступила в ОП раньше, чем i_{m-1}; i_k – раньше, чем i_e при $k>e$. При алгоритме РПРУ состояние ОП изменяется только в моменты страничных сбоев. Если запрашиваемая в момент t страница $x_t\notin B_{t-1}$, то страница i_m замещается, и новое состояние ОП есть $B_t=(x_t,i_1,i_2,\dots,i_{m-1})$. Для алгоритма НДИ состояние ОП изменяется и при обращениях к страницам, находящимся в ОП. Если $B_{t-1}=(i_1,i_2,\dots,i_\ell,\dots,i_m)$ и $x_t=i_\ell$, то $B_t=(i_\ell,i_1,\dots,i_{\ell-1},i_{\ell+1},\dots,i_m)$. При страничном сбое состояние ОП для НДИ изменяется по тому же правилу, что и для РПРУ.

Экспериментальные исследования показывают, что, как правило, программе для продвижения вперед в течение некоторого интервала времени не требуется все ее адресное пространство целиком. Для нормальной работы ей достаточно ограниченного набора страниц, который можно назвать рабочим телом программы. Состав страниц, входящих в рабочее тело программы, может меняться в процессе ее выполнения. Поэтому желательно, чтобы алгоритм замещения обладал двумя отчасти противоречивыми свойствами: с одной стороны, алгоритм должен сохранять в ОП страницы рабочего тела, а с другой – быстро обновлять содержимое ОП при смене рабочего тела.

Наиболее простой в реализации алгоритм РПРУ эффективен только в части быстрого обновления ОП, он не выделяет в списке B_t страницы рабочего тела, обращение к которым происходит чаще, чем к остальным

страницам. Алгоритм НДИ также позволяет быстро обновлять содержимое ОП, но в отличие от РПРУ он изменяет список B_t при всех обращениях к памяти. Такая регистрация использования страниц позволяет при определенных условиях сохранить в ОП страницы рабочего тела. Однако для схемы НДИ последовательность одиночных обращений достаточной длины к страницам, находящимся на СУ, вытеснит из ОП все страницы рабочего тела. Поскольку номер только что введенной в ОП страницы помещается в первую позицию или "хвост" списка B_t, то даже несколько однократных обращений к страницам из СУ может привести к вытеснению такого же числа страниц рабочего тела. Еще одним недостатком НДИ является относительно высокая стоимость реализации (о практической реализации алгоритмов замещения см. в работе [2]).

Компромисса между издержками реализации и эффективностью можно попытаться достигнуть, объединяя схемы РПРУ и НДИ (рис.I). В списке ОП

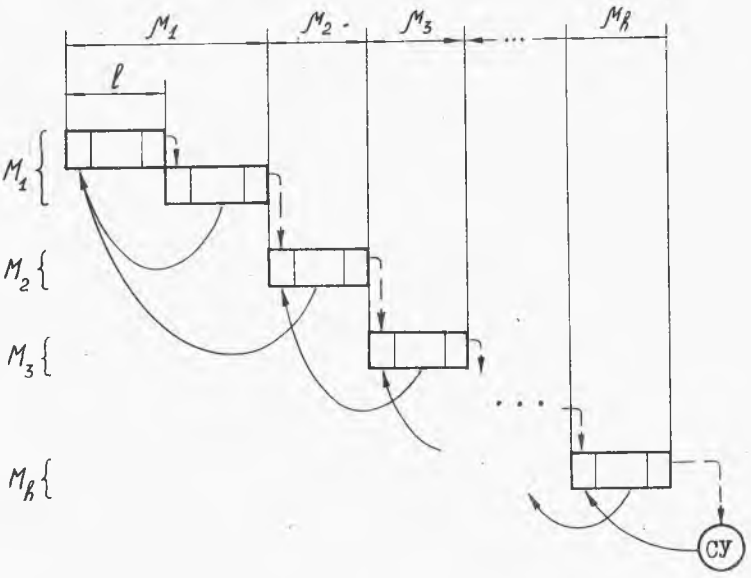
Р и с. I. Логическая схема смешанного алгоритма РПРУ-НДИ (в скобках - число позиций списка)

выделяется уровень ядра (позиции с I по ℓ, $\ell < m$) и переходный уровень. Если запрашиваемая страница находится в ОП и в списке B_t ей соответствует позиция $k \leq \ell$, работа смешанного алгоритма происходит так же, как при РПРУ, а при $k > \ell$ - так же, как при НДИ, причем номер страницы i_k перемещается в хвост, т.е. в первую позицию списка состояния ОП (подобную модификацию алгоритма НДИ, когда осуществляется перемещение номера страницы на другой уровень списка B_t, будем обозначать далее НДИ - М).

РПРУ, НДИ, смешанная схема РПРУ-НДИ являются частными алгоритма-

ми из большого класса многоуровневых алгоритмов замещения A_ℓ^h, определяемых $h+1$ параметром: $\mu_1, \mu_2, \ldots, \mu_h$ и ℓ, т.е. $A_\ell^h(\mu_1, \mu_2, \ldots, \mu_h)$, где $\sum_{i=1}^{h} \mu_i = m$ и $\ell \le \mu_1$ [3]. Многоуровневый алгоритм предполагает разделение списка состояния ОП на h непересекающихся приоритетных уровней – подсписков: главный уровень M_1 размером μ_1, включающий подсписок ядра и переходной подсписок, границу между которыми определяет величина параметра ℓ (как это установлено в рассмотренной смешанной схеме РНРУ-НДИ), и $h-1$ переходной подсписок $M_2 M_3, \ldots, M_h$ размер каждого из которых определяет соответствующий параметр $\mu_2, \mu_3, \ldots, \mu_h$.

Общая логическая схема алгоритма замещения A_ℓ^h представлена на рис.2. Номер только что введенной в ОП страницы помещается в хвост

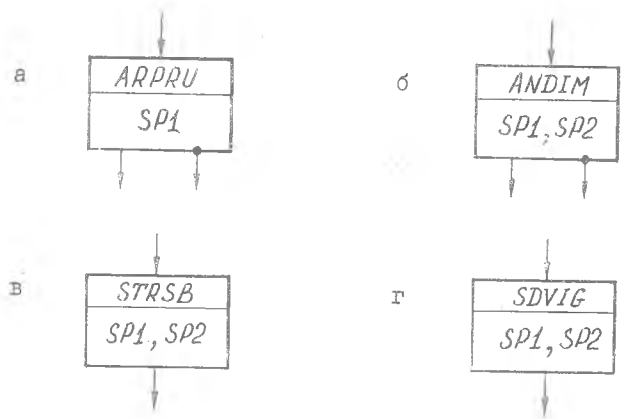


Р и с. 2. Логическая схема алгоритма замещения

младшего переходного подсписка M_h. При каждом новом обращении к этой странице ее номер в списке состояния ОП будет последовательно перемещаться в хвост подсписка все более высокого приоритета. Схема перемещения номера страницы в пределах главного уровня M_1 совпада-

ет с рассмотренной ранее смешанной схемой РПРУ-НДИ. Так как определяемые константами $\ell, \mu_1, \mu_2, \ldots, \mu_h$ объемы подсписков должны оставаться неизменными, то в общем случае помещение нового номера страницы в подсписок сопровождается вытеснением номера, оказавшегося в голове (последней позиции) данного подсписка. Вытеснение номеров страниц происходит по цепочке от высших уровней к низшим вплоть до вывода из списка при замещении данной страницы. Нетрудно видеть, что такая общая схема алгоритма A_ℓ^h при $h = I$ и $\ell = m$ дает алгоритм РПРУ, а при $h = \ell = 1$ - алгоритм НДИ. Смешанные схемы РПРУ-НДИ составляют множество алгоритмов вида A_ℓ^1, где $1 < \ell < m$.

Для операционного описания алгоритма $A_\ell^h(\mu_1, \mu_2, \ldots, \mu_h)$ достаточен конструктор, состоящий всего из 4-х операторов, графическое представление которых приведено на рис.3. Под операционным описа-



Р и с. 3. Конструктор операционного описания алгоритмов замещения вида $A_\ell^h(\mu_1, \mu_2, \ldots, \mu_h)$. SP_i - входной параметр, определяющий обрабатываемый список (подсписок) состояния ОП

нием алгоритма замещения здесь понимается описание действий и их последовательности по преобразованию списка состояния ОП, выполняемых при каждом очередном обращении программы к какой-либо ячейке ее адресного пространства, т.е. при каждом обращении к какой-либо странице программы .

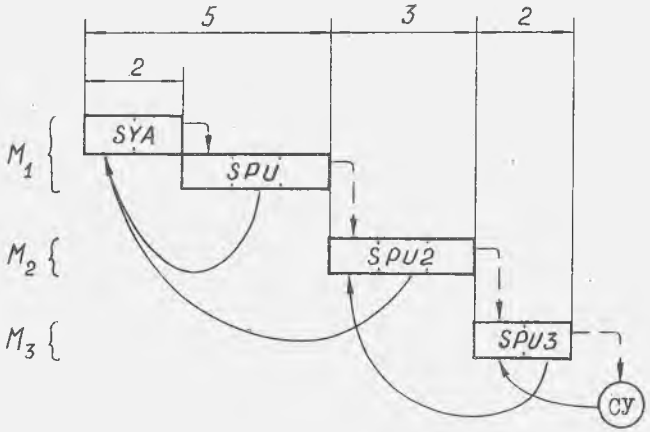
Оператор *ARPRU(SP1)* обрабатывает список (или подсписок) состояния ОП, идентифицируемый в описании как *SP1*, в соответствии с алгоритмом РПРУ. Оператор завершает свои действия либо обнаружив в *SP1* номер запрошенной страницы ($x_t \in B_{t-1}$), либо установив, что в *SP1* такой номер отсутствует. Последнему случаю на рис.3,а соответствует помеченный кружком выход.

Оператор *ANDIM(SP1, SP2)* обрабатывает *SP1* в соответствии с алгоритмом НДИ-М. Если номер запрашиваемой страницы в *SP1* отсутствует, то оператор завершает свои действия, передавая управление на "выход с кружком" (рис.3,б). В противоположном случае обнаруженный номер страницы извлекается из *SP1* и помещается в хвост *SP2*, и оператор НДИ-М завершает свои действия передачей управления на "нормальный выход" (см.рис.3,б).

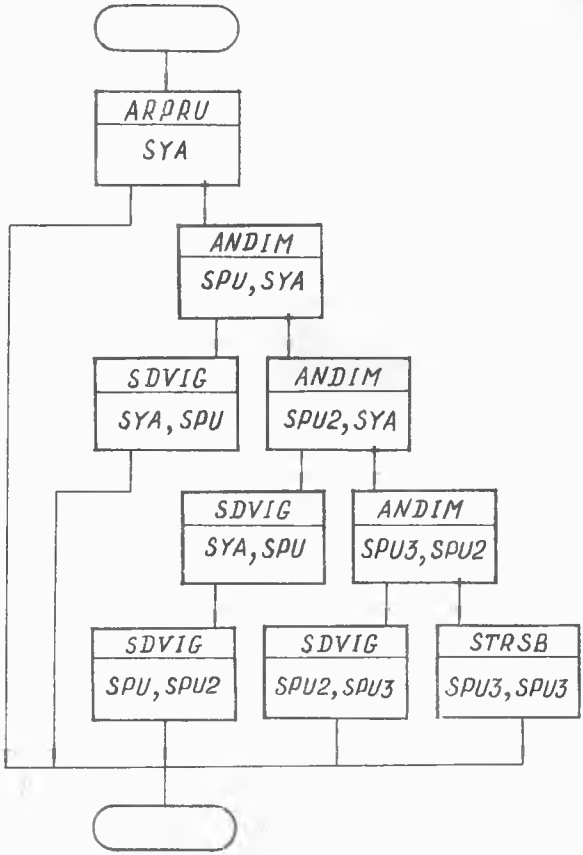
Оператор *STRSB(SP1, SP2)* выводит из *SP1* головной номер, а номер запрашиваемой страницы помещает в хвост *SP2* (рис.3,в).

Оператор *SDVIG(SP1,SP2)* вытесняет номер, стоящий в голове *SP1*, в хвост *SP2* (рис.3,г).

Например, приведенной на рис.4 логической схеме алгоритма A_2^3 (5,3,2) соответствует операционное описание в форме блок-схемы на рис.5. Отметим, что роль операторов *SDVIG* заключается в поддержа-



Р и с. 4. Логическая схема алгоритма замещения A_2^3 (5,3,2): *SYA* – подсписок ядра; *SPU* – подсписок переходного уровня; *SPU2* – подсписок уровня 2; *SPU3* – подсписок уровня 3



Р и с. 5. Операционное описание алгоритма замещения A_2^3 (5,3,2)

нии установленных объемов подсписков списка состояния ОП. Следует также подчеркнуть, что для описания некоторых частных алгоритмов замещения из всего рассмотренного множества алгоритмов не требуется использовать все четыре введенные разновидности операторов.

В целом введенные многоуровневые алгоритмы при адаптивном подборе их параметров соединяют свойство быстрого обновления части ОП

со свойством сохранения в ОП тех страниц, которые наиболее часто запрашиваются (т.е. содержатся в рабочем теле). Отсюда возникают дополнительные возможности синтеза алгоритмов замещения по критерию минимума частоты страничных сбоев.

Для оценки и сравнения различных алгоритмов замещения необходимо прежде всего указать способ задания последовательности обращений $x_1, x_2, \ldots$, т.е. определить модель поведения программы. Например, одной из простейших моделей поведения программ в указанном смысле является модель восстановления [3].

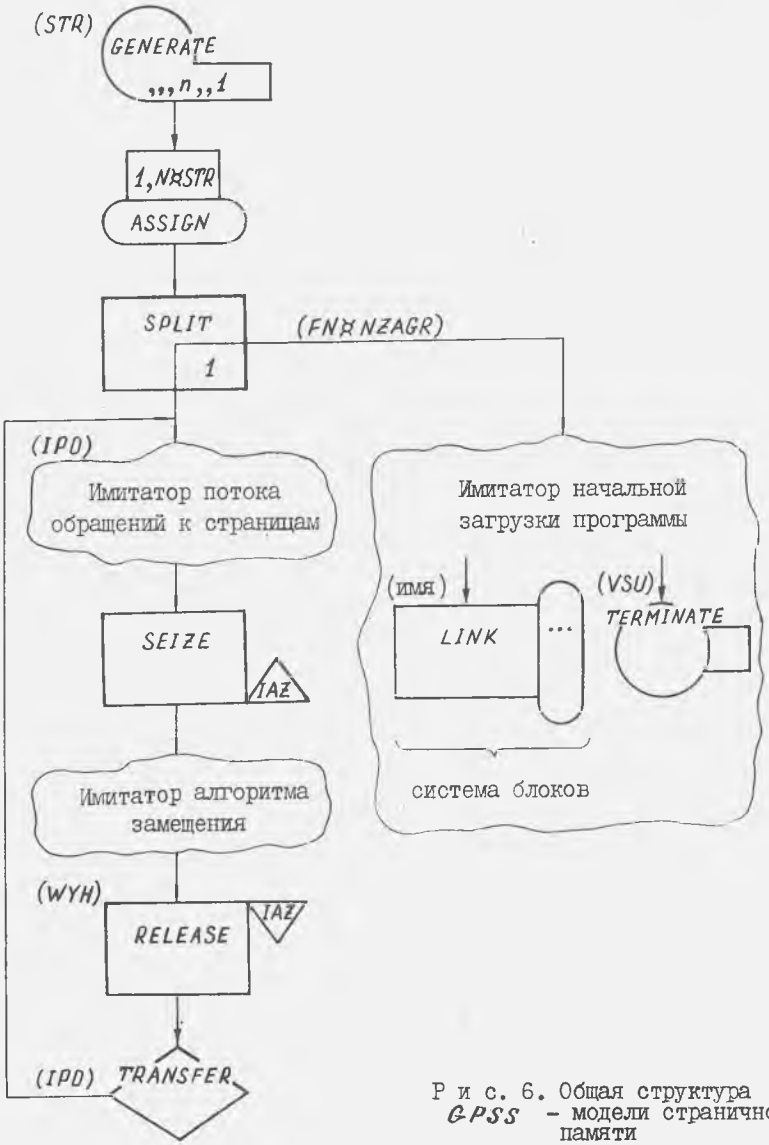
Для некоторых моделей поведения программ можно получить явные выражения критерия качества различных алгоритмов замещения; как правило, получаемые формулы громоздки и мало пригодны для вычислений [3]. В общем случае исследование алгоритмов замещения связано с использованием метода имитационного моделирования.

РЕКОМЕНДАЦИИ ПО РАЗРАБОТКЕ *GPSS* -ПРОГРАММ ИССЛЕДОВАНИЯ АЛГОРИТМОВ ЗАМЕЩЕНИЯ

Изучая возможность моделирования работы алгоритмов замещения на языке *GPSS* [4,5], необходимо прежде всего установить *GPSS* -объект, пригодный для имитации списка состояния ОП. Анализ показывает, что в качестве такого объекта можно выбрать список пользователя (а для многоуровневого списка состояния ОП – совокупность списков пользователя). Этот выбор определяет роль транзактов в модели: одни из них ("транзакт-номера"), находясь в списке пользователя, показывают состав страниц в ОП и имитируют своим взаимным расположением структуру списка состояния ОП; перемещение других ("транзактов-инициаторов") отражает происходящие в исследуемой системе обращения к страницам и возникающие при этом преобразования списка состояния ОП. Разумеется, транзакты той и другой разновидности должны обладать атрибутом "номер страницы".

Если поведение программы описывается моделью восстановления, то в качестве "заготовки" для построения *GPSS* -модели страничной памяти может быть использована блок-схема на рис.6. Приведем необходимые комментарии к рис.6 и рекомендации по разработке *GPSS* -программ исследования алгоритмов замещения класса A^k_e. В прил. дан пример такой программы с результатами счета.

Блок *GENERATE* последовательно вводит в модель совокуп-



Р и с. 6. Общая структура *GPSS* - модели страничной памяти

ность транзактов-инициаторов; их число должно быть равно количеству страниц программы n . Вводимые транзакты снабжаются одним параметром, значение которого для каждого транзакта устанавливает блок *ASSIGN*. Приведенная на рис.6 конструкция обеспечивает присвоение первому параметру входящего в блок *ASSIGN* транзакта порядкового номера этого транзакта во вводимой последовательности транзактов-инициаторов (т.е. I,2,..., n). Следовательно, этот параметр в дальнейшем может использоваться как атрибут "номер страницы".

Блок *SPLIT* вводит в модель транзакты-номера и определяет с помощью функции *NZAGR* , зависящей от первого параметра транзакта, начальную загрузку программы в виртуальную память. Функция *NZAGR* адресует в общем случае систему наименованных блоков *LINK*, обеспечивающих начальное заполнение списков пользователя, представляющих в модели различные уровни списка состояния ОП. Напомним, что список состояния ОП должен отражать как содержимое ОП, так и историю обращений к страницам. Следовательно, до начала работы может быть известно лишь содержимое списка состояния ОП, но взаимное расположение номеров страниц, вообще говоря, неопределено. В этих условиях имитацию начальной загрузки программы рекомендуется выполнить, исходя из принципа экономии усилий на программирование модели: первые m транзактов-номеров разместить в списках пользователя, например, в порядке "раньше пришел - раньше ушел", а $n-m$ оставшихся уничтожить в блоке *TERMINATE* , имитируя начальное помещение соответствующих страниц на СУ (см.прил.).

Имитатор потока обращений к страницам задерживает транзакт-инициатор на время до очередного обращения к соответствующей странице адресного пространства программы. Имитатор состоит всего лишь из одного блока

IPO ADVANCE FN¤VOSST

Функция *VOSST*, определяющая величину задержки, зависит от величины первого параметра транзакта-инициатора (т.е. от номера запрашиваемой страницы) и является выходным полюсом каскада функций и переменных, который реализует описывающую поведение программы модель восстановления. При этом входным полюсом каскада кроме *P1*-величины первого параметра транзакта, является *RN1* -реализация равномерно распределенной на интервале [0,I) случайной величины. Предполагается, что для аппроксимации реальных потоков обращений к каждой странице (с точностью до совпадения среднего и дисперсии величины промежутка времени

между событиями в потоке) использованы обобщенные потоки Эрланга и потоки с гиперэкспоненциальным распределением промежутка времени между событиями (см.лабораторно-расчетную работу I).

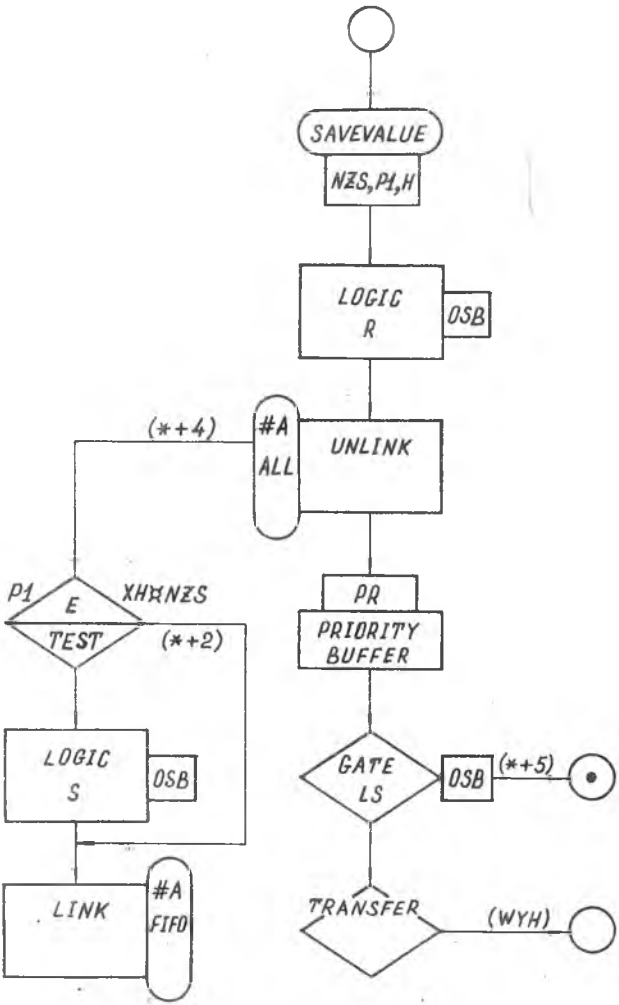
GPSS - устройство *IAZ* и оперирующие им блоки используются в модели (см.рис.6) для исключения появления в имитаторе алгоритма замещения одновременно двух и более транзактов-инициаторов. Возможность же возникновения подобной ситуации объясняется несовершенством машинной реализации модели восстановления для имитации поведения программы (имеется отличная от нуля вероятность того, что величина интервала условного времени между последовательными обращениями к памяти будет равна нулю).

Имитатор алгоритма замещения строится на основе операционного описания этого алгоритма. При этом операторы *ARPRU*, *ANDIM*, *STRSB* и *SDVIG* в *GPSS* могут быть реализованы как макро с некоторыми дополнительными параметрами. Блок-схемы предлагаемых макро приведены на рисунках, ниже даются необходимые комментарии.

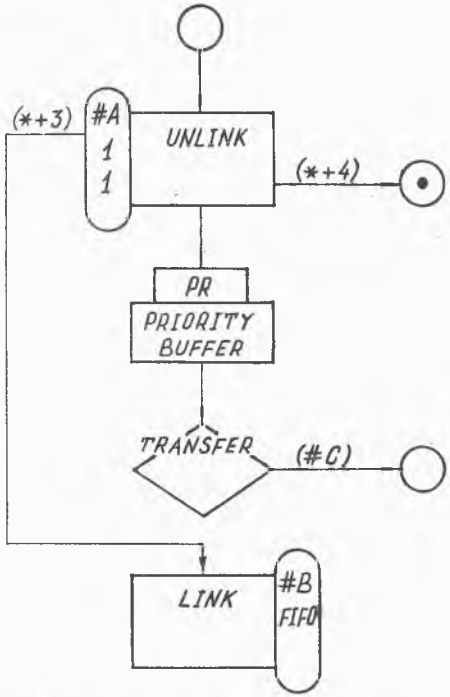
Действия макро *ARPRU* (рис.7) связаны с последовательным просмотром всех членов списка пользователя с именем *#A* с целью формирования состояния логического переключателя *OSB*. Если в результате просмотра он оказывается установленным (*LS*), то запрошенная страница находится в ОП, и транзакт-инициатор может покинуть имитатор алгоритма замещения (*TRANSFER , WVH*). Альтернативой служит направление транзакта-инициатора в следующий за макро *ARPRU* блок ("выход с кружком" на рис.3,а).

GPSS - реализация оператора *ANDIM* (рис.8) предусматривает извлечение из списка пользователя с именем *#A* транзакта-номера, величина первого параметра которого равна величине первого параметра транзакта-инициатора (см.смысл и задание параметра *D* блока *UNLINK* [5]). Извлеченный транзакт-номер помещается в хвост списка пользователя с именем *#B* (напомним, в общем случае *#A* и *#B* -разные списки), а транзакт-инициатор направляется в блок с именем *#C* ("нормальный выход" на рис.3,б). Если искомый транзакт-номер в списке *#A* отсутствует, то транзакт-инициатор будет направлен в следующий за макро *ANDIM* блок (этот выход на рис.3,б отмечен кружком).

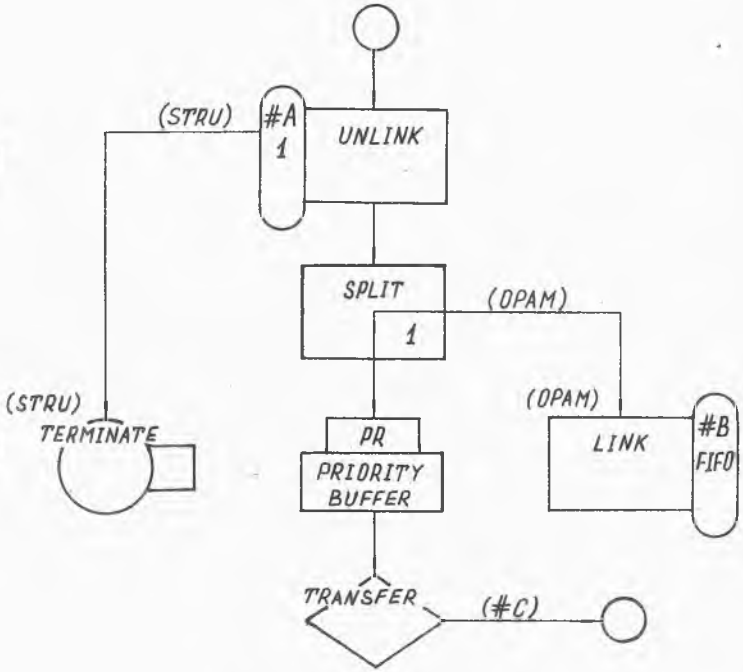
В макро *STRSB*(рис.9) из модели выводится головной транзакт-номер списка пользователя с именем *#A* , что имитирует выталкивание соответствующей страницы на СУ. В хвост списка пользователя с именем *#B*(в общем случае *#A* и *#B* разные) помещается новый транзакт-



Р и с. 7. *GPSS* -реализация оператора *ARPRU*



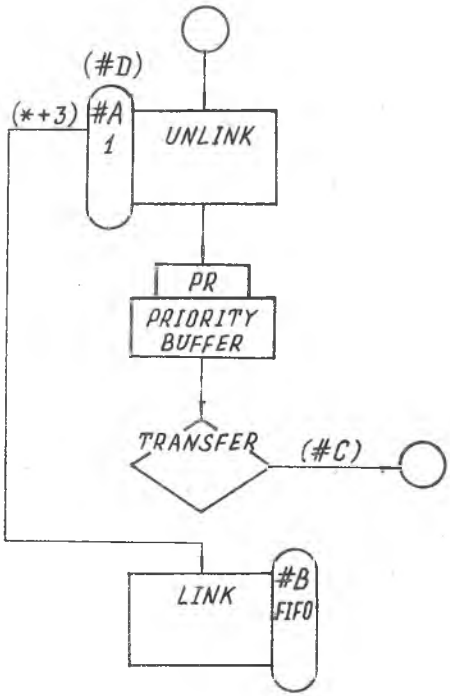
Р и с. 8. *GPSS* -реализация оператора *ANDIM*



Р и с. 9. *GPSS* -реализация оператора *STRSB*

номер, являющийся копией транзакта-инициатора, т.е. производится необходимое изменение содержимого списка состояния ОП в случае подкачки запрошенной страницы с СУ. Транзакт-инициатор направляется в блок с именем $\#C$.

Использование макро *SDVIG* (рис.10) направлено на поддержание установленных объемов списков пользователя, имитирующих различные уровни списка состояния ОП. Параметр $\#A$ задает имя списка пользователя, из которого извлекается головной транзакт-номер, а $\#B$ - имя списка пользователя, в хвост которого этот транзакт помещается. Транзакт-инициатор направляется в блок с именем $\#C$. Параметр $\#D$ определяет имя первого блока макрорасширения *SDVIG*.



Р и с. 10. *GPSS* реализация оператора *SDVIG*

Во всех четырех рассмотренных макро важную роль играет блок *PRIORITY* . Формальное переназначение транзакту-инициатору уровня приоритета приводит к помещению этого транзакта на последнее место в цепи текущих событий системы моделирования, а параметр *BUFFER* требует от системы возобновить просмотр цепи текущих событий с начала [5] . Тем самым в *GPSS*-моделях на рис.7-10 после попадания транзактов-инициаторов в блок *PRIORITY* обеспечивается приоритет перемещениям транзактов-номеров и, следовательно, завершение действий каждого макро прежде, чем его покинет транзакт-инициатор.

Сборка имитаторов алгоритмов замещения A_e^h $(\mu_1, \mu_2, \ldots, \mu_h)$ с использованием рассмотренных макро осуществляется по единой методике в полном соответствии с операционным описанием алгоритма замещения.

Макровызовы *ARPRU* , последующие *ANDIM* и *STRSB* должны располагаться строго последовательно; макровызовы *SDVIG* рекомендуется помещать после вызова *STRSB* до блока, помеченного именем *WYH*. Например, *GPSS* – определение имитатора алгоритма замещения A_2^3 (5,3,2) (см.рис.4,5), имеет вид

```
ARPRU    MACRO    SYA
ANDIM    MACRO    SPU, SYA, SDV1
ANDIM    MACRO    SPU2, SYA, SDV2A
ANDIM    MACRO    SPU3, SPU2, SDV3
STRSB    MACRO    SPU3, SPU3, WYH
SDVIG    MACRO    SYA, SPU, WYH, SDV1
SDVIG    MACRO    SYA, SPU, SDV2B, SDV2A
SDVIG    MACRO    SPU, SPU2, WYH, SDV2B
SDVIG    MACRO    SPU2, SPU3, WYH, SDV3
WYH      . . .
```

Для организации эксперимента с *GPSS* –моделью на рис.6 следует воспользоваться приемом, использующим транзакт (процесс) –таймер [4,5] . Например, таймер

```
GENERATE      100000
ADVANCE       0
GATE NU       IAZ
TERMINATE     1
```

при использовании в *GPSS* –программе карты *START 1* обеспечивает имитацию работы алгоритма замещения в течение 100000 единиц модельного времени. Наличие в таймере блоков *ADVANCE* и *GATE* обеспечивает правильную увязку движений транзакта-таймера и транзактов-инициаторов.

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

Порядок выполнения этой работы не отличается от установленного для других лабораторно-расчетных работ и является таким:

получить задание на исследование и изучить необходимые теоретические материалы;

разработать программу моделирования и эксперимента;

просчитать программу на ЭВМ;

оформить отчет.

СОДЕРЖАНИЕ ОТЧЕТА

Отчетом служит распечатка АЦПУ, содержащая программу и результаты выполненного имитационного эксперимента. В программе должны быть подчеркнуты операторы *ARPRU*, *ANDIM*, *STRSB*, *SDVIG*.

На обратной стороне распечатки помещаются:

формулировка задания на лабораторную работу;

логическая схема исследуемого алгоритма замещения;

операционное описание алгоритма замещения в форме блок-схемы;

выводы о качестве исследованных алгоритмов замещения (сравнение результатов).

ЗАДАНИЕ НА ИССЛЕДОВАНИЕ

Требуется оценить качество алгоритма замещения $A_\ell^h(\mu_1, \mu_2, \ldots, \mu_h)$ по критерию частоты обращений к страничному устройству. Исследование предлагается выполнить в форме вычислительного эксперимента с имитационной моделью, где воспроизводится работа собственно алгоритма замещения и поведение некоторой программы. Источником сведений о поведении программы и соответствующей модели восстановления служит индивидуальное задание и результаты выполнения лабораторно-расчетной работы I.

По усмотрению преподавателя устанавливается число страниц m, отводимых программе в ОП, и для индивидуального исследования указывается алгоритм замещения A_ℓ^h ($\mu_1, \mu_2, \ldots, \mu_h$). Кроме указанного алгоритма необходимо исследовать (без построения граф-схем) качество еще не менее двух алгоритмов замещения, варьируя в исходном алгоритме параметры $\mu_1, \mu_2, \ldots, \mu_h$ при условии $\sum_{i=1}^{h} \mu_i = m$, $h = const$, $\ell = const$.

Поведение программы и работу алгоритма замещения имитировать на отрезке в I00000 единиц модельного времени.

КОНТРОЛЬНЫЕ ВОПРОСЫ

I. Сформулируйте цели управления памятью в вычислительных системах.

2. Раскройте понятие виртуальной памяти вычислительной системы.

3. Что представляет собой страничная организация памяти?

4. Как следует понимать термин "обращение к странице"?

5. В чем состоит сущность проблемы замещения страниц и технической задачи по осуществлению решения проблемы замещения?

6. Как будет влиять на эффективность страничной памяти количество отводимых программе блоков ОП? размер страницы?

7. Какие предположения о поведении программы делаются при конструировании алгоритмов замещения? Охарактеризуйте с этой точки зрения алгоритмы случайного замещения, РПРУ, НДИ.

8. Какова роль понятия рабочего тела программы при анализе и синтезе алгоритмов замещения?

9. Перечислите известные Вам модели поведения программ, отражающие их взаимодействие с ОП.

10. Поясните смысл введения многоуровневых алгоритмов замещения. Каково количество различных алгоритмов замещения вида $A^k_{\mu_1}(\mu_1,\mu_2,\dots,\mu_k)$, где $\sum_{i=1}^{k}\mu_i = m$?

11. Покажите эквивалентность логической схемы и блок-схемы операционного описания алгоритма замещения $A^k_{\ell}(\mu_1,\mu_2,\dots,\mu_k)$.

12. Выделите основные функциональные фрагменты Вашей программы исследования алгоритмов замещения. Сколько использовано Вами разновидностей введенных операторов описания алгоритмов замещения?

Библиографический список

1. Кейлингерт П. Элементы операционных систем. Введение для пользователей.М.:Мир, 1985. С.295.

2. Шоу А. Логическое проектирование операционных систем. М.:Мир, 1981. С.360.

3. Авен О.И., Коган Я.А. Управление вычислительным процессом в ЭВМ (Алгоритмы и модели). М.:Энергия, 1978. С.240.

4. Шрайбер Т.Дж. Моделирование на *GPSS* . М.:Машиностроение, 1980. С.592.

5. Кораблин М.А. Программирование моделей дискретных систем на языке *GPSS* : Учебное пособие. Куйбышев:КуАИ, 1981. С.72.

GPSS -ПРОГРАММА И РЕЗУЛЬТАТЫ ИССЛЕДОВАНИЯ АЛГОРИТМА ЗАМЕЩЕНИЯ A_2^3 (5,3,2)

```
SIMULATE
*
```

Каскад функций и переменных, который реализует описывающую поведение программы модель восстановления.
Входные полюса каскада - RN1, P1.
Выходной полюс - функция VOSST.

```
*
      NZAGR  FUNCTION  P1, D5
2,    VSYA/5, VSPU/8, VSPU 2/1Ф, VSPU3/11, VSU
*     DSS  FVARIABLE  1ФФФ*NX STRU/NX WYH
*     ARPRU  STARTMACRO
            SAVEVALUE       NZS, P1, H
            LOGIC R         OSB
            UNLINK          #A, *+4, ALL
            PRIORITY        PR, BUFFER
            GATE LS         OSB, *+5

            TRANSFER        , WYH
            TEST E          P1, XHX NZS, *+2
            LOGIC S         OSB
            LINK            #A, FIFO
            ENDMACRO

*   ANDIM   STARTMACRO
            UNLINK          #A, *+3,1,1,, *+4
            PRIORITY        PR, BUFFER
            TRANSFER        , #C
            LINK            #B, FIFO
            ENDMACRO
```

```
*
STRSB   STARTMACRO
        UNLINK      #A, STRU, 1
        SPLIT       1, OPAM
        PRIORITY    PR, BUFFER
        TRANSFER    , #C
STRU    TERMINATE
OPAM    LINK        #B, FIFO
        ENDMACRO

*
SDVIG   STARTMACRO
#D      UNLINK      #A, *+3, 1
        PRIORITY    PR, BUFFER
        TRANSFER    , #C
        LINK        #B, FIFO
        ENDMACRO

*
*
STR     GENERATE    ,,, 12,, 1
        ASSIGN      1, NXSTR
        SPLIT       1, FNX NZAGR

*
IPO     ADVANCE     FNX VOSST
        SEIZE       IAZ

*
ARPRU   MACRO       SVA
ANDIM   MACRO       SPU, SVA, SDV1
ANDIM   MACRO       SPU2, SVA, SDV2A
ANDIM   MACRO       SPU3, SPU2, SDV3
STRSB   MACRO       SPU3, SPU3, WYH
SDVIG   MACRO       SVA, SPU, WYH, SDV1
SDVIG   MACRO       SVA, SPU, SDV2B, SDV2A
SDVIG   MACRO       SPU, SPU2, WYH, SDV2B
SDVIG   MACRO       SPU2, SPU3, WYH, SDV3
```

```
WYH     RELEASE     IAZ
        TRANSFER    ,IPO

*
VSYA    LINK        SYA,FIFO
VSPU    LINK        SPU,FIFO
VSPU2   LINK        SPU2,FIFO
VSPU3   LINK        SPU3,FIFO
VSU     TERMINATE
*
*
        GENERATE    1ФФФФФ
        ADVANCE     Ф
        GATE NU     IAZ
        SAVEVALUE   KOBRA,NX$WYH,H
        SAVEVALUE   KSTRS,NX$STR,H
        SAVEVALUE   REZ ,VX$DSS,H
        TERMINATE   1
*
*
        START       1
        REPORT
        OUTPUT
2Ф      TEXT        ОБРАЩЕНИЙ К ПАМЯТИ #XH$KOBR/2/XXXXX#
2Ф      TEXT        СТРАНИЧНЫХ СБОЕВ #XH$KSTRS,2/XXXXX#
2Ф      TEXT        ПРОЦЕНТ СТР.СБОЕВ #XH$REZ,2/1LXXXX.X#
        END
```

Стандартный протокол моделирования

```
ОБРАЩЕНИЙ К ПАМЯТИ      3278
СТРАНИЧНЫХ СБОЕВ         354
ПРОЦЕНТ СТР. СБОЕВ       10.7
END
```

Составитель Сергей Викторович С м и р н о в

ИССЛЕДОВАНИЕ АЛГОРИТМОВ ЗАМЕЩЕНИЯ

Редактор Е.Д.А н т о н о в а
Техн.редактор Н.М.К а л е н ю к
Корректор Л.Я.Ч е р н о д а е в а

Подписано в печать 21.04.89 г. Формат 60x84 1/16.
Бумага оберточная белая. Печать оперативная.
Усл.п.л. 1,4. Уч.-изд.л. 1,3. Т. 500 экз.
Заказ № 5873. Бесплатно.

Куйбышевский ордена Трудового Красного Знамени авиационный институт имени академика С.П.Королева. 443001, Куйбышев, ул.Молодогвардейская, 151.

Типография имени В.П.Мяги Куйбышевского полиграфического объединения. 443099, Куйбышев, ул.Венцека, 60.