ОБЪЕДИНЕННЫЙ
ИНСТИТУТ
ЯДЕРНЫХ
ИССЛЕДОВАНИЙ
ДУБНА

Р10 - 9954
6/XII-76

Л.Дади, А.Матеева, Ю.Намсрай, И.М.Саламатин

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ИЗМЕРИТЕЛЬНОГО МОДУЛЯ НА БАЗЕ ЭВМ **ТРА-1001-i**

I. Программа динамического распределения памяти

1976

Р10 - 9954

Л.Дади, А.Матеева, Ю.Намсрай, И.М.Саламатин

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ИЗМЕРИТЕЛЬНОГО МОДУЛЯ НА БАЗЕ ЭВМ **ТРА-1001-i**

I. Программа динамического распределения памяти

Направлено в журнал "Программирование"

Введение

При разработке /либо в процессе эксплуатации/ архитектура универсальной ЭВМ приводится в соответствие с ожидаемыми средними характеристиками потока программ. При этом учитываются потребности в памяти, коэффициент использования процессора, необходимая скорость вычислений и другие факторы. На вычислительных комплексах с фиксированной конфигурацией выполнение программ, объем которых превышает наличные ресурсы оперативной памяти, возможно благодаря программно или аппаратно реализованным механизмам свопинга, оверлея, динамического распределения памяти /ДРП/ и др. При свопинге необходимый для инициирования очередной программы объем оперативной памяти изымается у временно приостановленной программы. Текущее состояние этой программы запоминается на внешнем запоминающем устройстве. Для оверлея характерно разделение программы на сегменты на этапе ее написания. Сегменты поочередно загружаются на одно и то же заранее выделенное место в оперативной памяти, затирая при этом предыдущий работавший сегмент. Механизм ДРП располагает некоторым конечным резервом оперативной памяти. Часть этой памяти выделяется программе /или ее сегменту/ не заранее, а на этапе выполнения. По мере заполнения распределяемой области памяти ненужные сегменты заменяются новыми. Средства такого типа применялись на машинах первого поколения[1] и первых моделях мини-ЭВМ для компенсации ограничения длины программы, обусловленного малым объемом оперативной памяти. По мере развития архитектуры и технологии производства ЭВМ эти же средства были использованы

для повышения коэффициента загрузки оборудования, в первую очередь - процессора, в больших и средних вычислительных комплексах. При этом увеличился вес аппаратных средств в механизме, реализующем распределение памяти.

Появившиеся в начале 60-х годов мини-ЭВМ получили широкое распространение в различных областях науки и техники. На их базе создано значительное число специализированных систем реального времени /СРВ/. Обычными компонентами СРВ являются подсистемы регистрации, накопления и обработки данных. Характерные особенности СРВ - резервирование времени центрального процессора и небольшие потребности в вычислительной мощности. Потребители мини-ЭВМ заинтересованы в снижении реальной стоимости систем, создаваемых на базе этих машин. Наблюдаемая в последние годы тенденция к программной организации ДРП на мини-ЭВМ /2/ обеспечивает /помимо других достоинств/ дальнейшее снижение стоимости систем накопления и обработки данных.

Помимо известных отличительных особенностей механизма ДРП, отметим особо простоту согласования его с интерпретирующими программами. Интерес к этому свойству ДРП достаточно пояснен в работе /3/, описывающей общую структуру программного обеспечения измерительного модуля.

Данная работа посвящена программам, реализующим ДРП на ЭВМ ТРА-1001-i. Эти программы входят в состав монитора, структура которого описана в работе /3/.

Организация динамического распределения памяти

Программа ДРП разработана для использования библиотеки стандартных программ /СП/, описанной в работе /4/. Библиотека имеет модульную структуру. СП скомпонованы в загрузочные модули длиной 1, 2, 3 или 4 страницы. Монитор или программа пользователя при

обращении к программе ДРП передает информацию об СП в виде ее номера. Способ кодирования номера показан на *рис. 1.* Номер СП является виртуальным адресом строки входа в загрузочном модуле /ЗМ/. В старших 9 разрядах номера СП кодируется адрес первой страницы ЗМ на системном внешнем запоминающем устройстве /ВЗУ/. Адрес кодируется относительно начала библиотеки на ВЗУ.

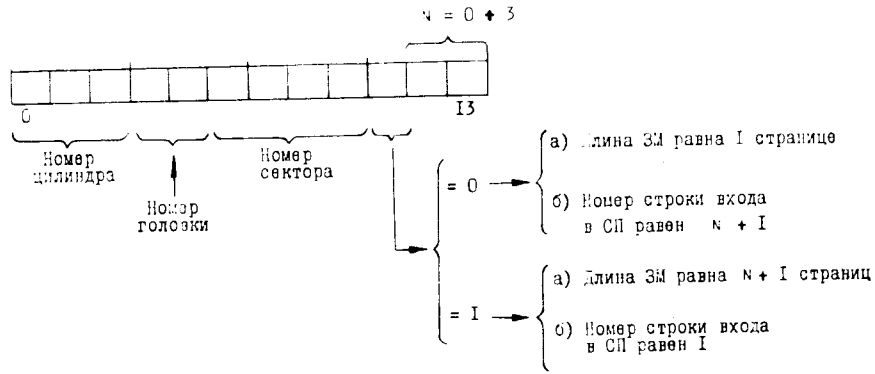

Рис. 1. Способ кодирования номера стандартной программы.

Программа ДРП управляет состоянием выделенного участка оперативной памяти - рабочего поля /РП/. РП предназначено для загрузки программ, необходимых на данном этапе работы монитора или программы пользователя /программы эксперимента/. Информацию о текущем состоянии РП программа ДРП хранит в ассоциативной таблице ТХ. Каждой странице РП соответствует одна строка в ТХ /*см. рис. 2*/. Строка ТХ для каждой из занятых страниц РП содержит закодированный адрес данной страницы на ВЗУ. Для первой из принадлежащих ЗМ страниц содержание этой строки составляют 9 старших разрядов номера СП.

ТХ заполняется сверху вниз. Первой строкой с нулевым содержанием начинается свободная часть ТХ. Соответствующая ей часть РП также считается свободной. Ненулевое содержание строк ТХ, стоящих после пустой, отражает предысторию работы и игнорируется.

Программа ДРП по номеру СП выясняет, находится ли она уже в оперативном запоминающем устройстве /ОЗУ/. Для этого из номера СП выделяется образ нужной строки и просматривается ТХ. Вывод о наличии нужной СП в РП делается при обнаружении строки, совпадающей со строкой, выделенной из номера. Порядковый номер найденной в ТХ строки, адрес начала РП в ОЗУ и номер строки входа в СП совместно определяют абсолютный адрес входа в нужную СП.

При отсутствии СП в ОЗУ программа ДРП загружает нужный ЗМ на свободное место в РП. Если свободного места недостаточно для загрузки очередного ЗМ, то зачеркивается ТХ и заполнение РП начинается заново.

Для уменьшения системных затрат введен механизм фиксации программ на РП. С этой целью РП разделено на две части /РП1 и РП2/ плавающей границей. РП1 расположено с начала РП, РП2 - непосредственно за ним. Начальное состояние РП характеризуется нулевой длиной РП1. Двум частям РП соответствуют две части ассоциативной таблицы - ТХ1 и ТХ2.

Монитор перед обращением к программам, требующимся чаще других, выставляет совокупность признаков, по которым программа ДРП определяет, нужно фиксировать СП на РП или нет. Фиксируемые СП загружаются только на РП1. Плавающая граница фиксации по мере заполнения РП1 смещается вниз. Одновременно с этим смещается начало РП2 и уменьшается выделенная ему часть РП.

Введенный механизм фиксации является "мягким". Программа ДРП может зачеркнуть ТХ1, но лишь в том случае, когда для загрузки следующего ЗМ недостаточно длины РП2, т.е. если иначе дальнейшая работа невозможна.

Значения идентификаторов, упоминаемых в тексте, и адреса ячеек, которые могут быть нужны программисту, даны в *табл. 1,2* данной работы и в *табл. 2* работы /4/.

На *рис. 2* показан пример РП, на которое загружены 5 ЗМ. Номера ЗМ, количество содержащихся в них СП и номера этих СП приведены на этом же рисунке. Справа

Таблица 1

Значения идентификаторов, используемых в тексте данной работы

Идентификатор и адрес ячейки памяти		Содержание ячейки
SWLEV	0105	Номер таблицы очередей
SAVEDL	0110	Номер фазы обработки
TRAKT1	0116	Начало группы из 6 рабочих ячеек монитора
ATABL	0114	Адрес таблицы распределения памяти
DISLIB	1403	Адрес начала используемой версии библиотеки на ВЗУ
RABPOL	1647	Адрес начала рабочего поля РП
CTOVRL	1711	Константа, управляющая глубиной ссылок вверх из оверлейной области
ATTX	2000	Адрес начала ассоциативной таблицы
ATTX2	2001	Текущий адрес начала строк ТХ для нефиксируемых программ
ATTX	2002	Текущий адрес первой свободной строки в ТХ
AGRNTX	2003	Адрес конца ассоциативной таблицы
DLINTX	2004	Обратная длина РП
STRTX	2164	Номер страницы ЗМ
NOMERP	2165	Номер СП — Для последней из программ,
AENTRY	2166	Адрес входа в СП — к которым были обращения
SFAZA	2167	Признак фиксации

Таблица 2

Адреса входов в резидентные подпрограммы

Идентификатор программы	и адрес	Выполняемая функция
LOADER	1416	Вычисление адреса ЗМ на ВЗУ
PDISK	1456	Работа с накопителем на магнитном диске
ISTADR	1600	Вычисление адреса загрузки ЗМ и адреса входа в СП
ZATIRK	1630	Аннулирует прямые ссылки на фиксированные СП
CORREC	1652	Перерабатывает ЗМ в соответствии с адресом его загрузки
SHORTC	1715	Осуществляет аварийный алгоритм
ABONEN	2134	Обеспечивает прямой доступ к зафиксированной СП при повторных обращениях к ней (заполняет ссылки на фиксируемые СП)

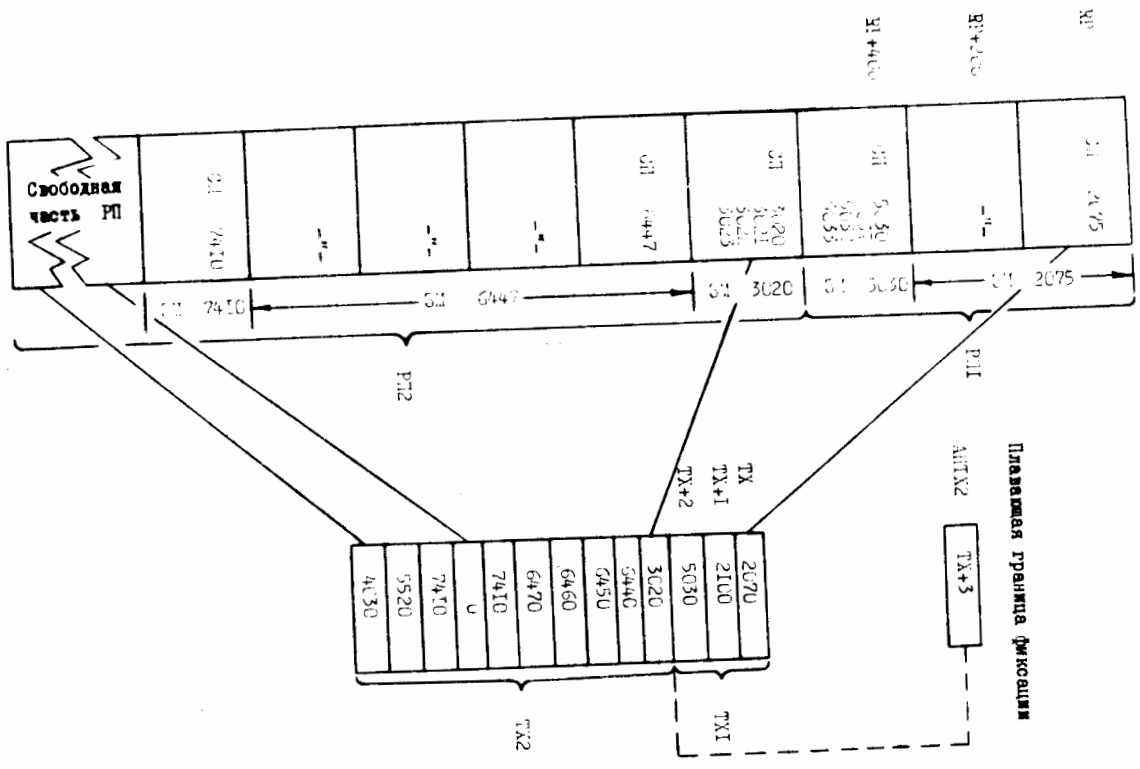

Рис. 2. Соответствие страниц рабочего поля и строк ассоциативной таблицы.

показано, как данное состояние РП отражается в ТХ и в значении плавающей границы фиксации. Первые три строки ТХ составляют ТХ1 и отражают фиксированную часть РП. Об этом свидетельствует содержание ячейки АНТХ2, равное ТХ+3. Занято 9 страниц РП, остальная часть свободна.

Описание блок-схемы

Блок-схема программы ДРП и связывающего загрузчика /СЗ/ приведена на *рис. 3.* Введены два типа обращений к программе ДРП: с ненулевым и с нулевым значением аккумулятора /АС/. В первом случае номер нужной СП содержится в АС. Во втором случае номер разыскивается по адресу (ТРАКТ1)+2. Скобки означают "содержимое". Предполагается, что ячейка ТРАКТ1 была заполнена монитором. Номер СП для дальнейшей работы запоминается в ячейке NOMERP. Из номера СП выделяются 9 старших разрядов и заносятся в ячейку STRTX . Вычисляется сумма (SFAZA)=(SAVEDL)+(SWLEV). Содержание SAVEDL и SWLEV было задано монитором. В дальнейшем нулевое содержание ячейки SFAZA интерпретируется, как требование фиксировать СП. Затем выполняется цикл просмотра ТХ. Выход из этого цикла происходит по одной из трех причин:

1. Найдена строка в ТХ, совпадающая с образом, выделенным из номера СП.
2. Просмотрена вся ТХ, нужная строка не найдена.
3. Встретилась нулевая строка.

В первом случае /правая ветвь блок-схемы на *рис.3/* СП находится на РП. Вычисляется адрес входа в СП /программа ISTADR /. Если программа фиксируется на РП, то заполняются ссылки на эту СП /программой ABONEN /. В дальнейшем обращение к фиксированной СП будет выполняться, минуя программу ДРП. После этого управление передается СП.

Во втором и третьем случаях требуется загрузить программу. Проверяется, разрешена ли загрузка в ОЗУ нового ЗМ. Такой запрет мог быть установлен монитором,

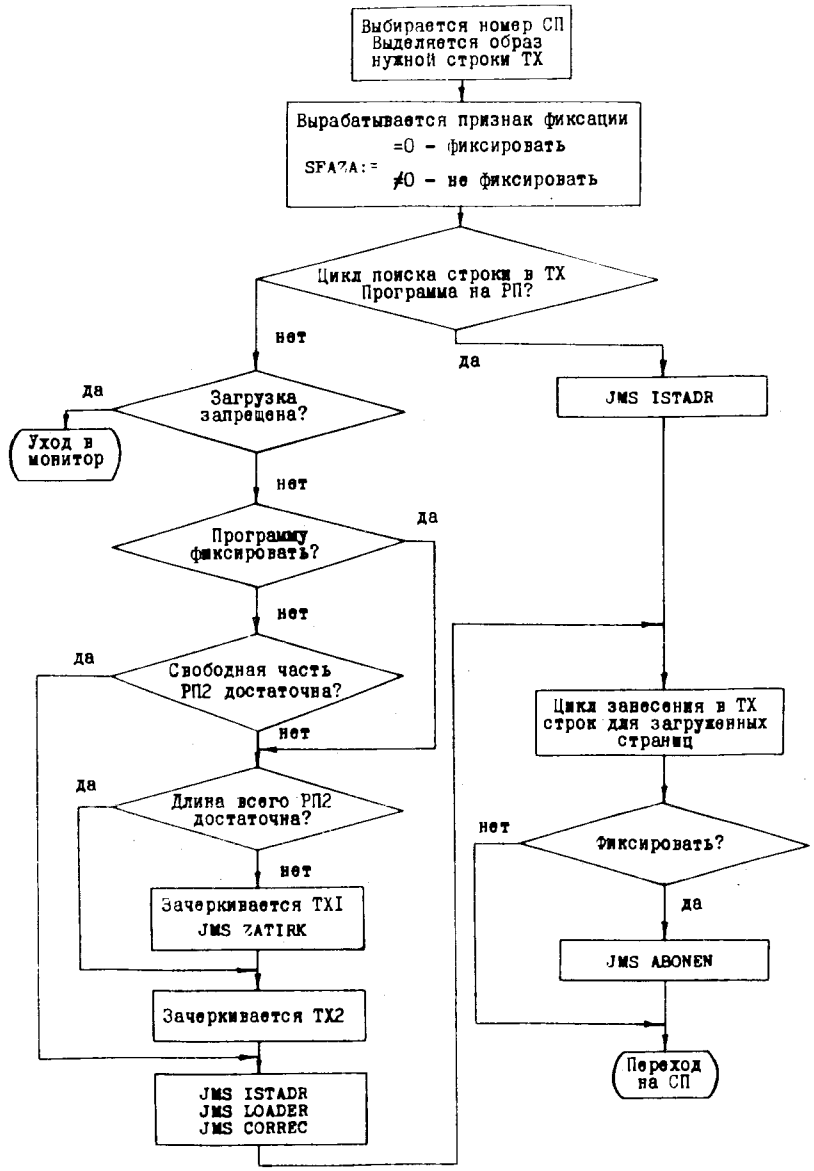

Рис. 3. Блок-схема программы динамического распределения памяти.

если в этот момент еще не была закончена предшествовавшая операция с системным ВЗУ или на РП находилась программа, ожидавшая готовности ВЗУ. При наличии запрета загрузки управление передается монитору. Если загрузка разрешена, то с учетом признака фиксации определяется место размещения нужного ЗМ. Место размещения ЗМ с нефиксируемыми СП выделяется на первом же участке достаточной длины из числа следующих:

1. На свободном участке РП2.
2. С начала РП2.
3. С начала РП1.

Анализ состояния РП при поиске места загрузки ЗМ выполняется по содержимому рабочих ячеек АНТХ, АНТХ2, АТТХ, AGRNTX, DLINTX, описывающих состояние ассоциативной таблицы ТХ.

В случае загрузки ЗМ с фиксируемыми СП опускается проверка длины свободной части РП2.

Занесение ЗМ любого типа на начало РП2 сопровождается затиранием старого содержания ТХ2. Загрузка ЗМ на начало РП1 приведет к затиранию всего ТХ, т.е. ТХ1 и ТХ2. Помимо затирания ТХ1, ссылки на СП, ранее зафиксированные на РП1, заменяются ссылками на программу ДРП. Эта операция выполняется программой ZATIRK. После принятия решения о месте загрузки ЗМ в рабочей ячейке АТТХ оставляется вычисленный текущий адрес части ТХ, соответствующей занимаемой части РП. Программа ISTADR вычисляет абсолютные адреса. программа LOADER выполняет считывание ЗМ, после чего в ТХ заносятся строки, описывающие данный ЗМ. Дальнейшие действия аналогичны перечисленным выше для случая, когда СП найдена на РП.

Поиск места для ЗМ с фиксируемыми СП начинается с проверки длины всего РП2. Факт наличия свободной части РП2 игнорируется. Если места достаточно, то ЗМ заносится на начало РП2, старая ТХ2 затирается, а в ячейке АНТХ2 запоминается новая нижняя граница ТХ1. Помимо этого, в программе пользователя заполняются ссылки на фиксируемую программу /программой ABONEN /. Остальные действия аналогичны описанным выше для нефиксируемых СП.

Вспомогательные программы

Группа программ ISTADR, LOADER, PDISK, CORREC, ABONEN, ZATIRK составляет связывающий загрузчик.

Программа ISTADR использует значения из ячеек АНТХ и АТТХ для вычисления порядкового номера используемой страницы РП. Для определения абсолютного адреса этой страницы из ячейки RABPOL выбирается адрес RP начала РП. Эта же программа вычисляет адрес входа в СП по формуле:

$$(AENTRY) = RP + [(ATTX) - (AHTX)] * 200_8 + 2 + N,$$

где N - порядковый номер строки входа в СП.

Программы LOADER *и* PDISK обеспечивают привязку к конкретному системному устройству. LOADER переводит номер ЗМ, содержащийся в ячейке STRTX, в адрес ЗМ относительно начала библиотеки. Абсолютный адрес ЗМ получается, как сумма относительного адреса ЗМ и абсолютного адреса используемой копии библиотеки из ячейки DISLIB. Считывание ЗМ выполняется программой PDISK. Контроль правильности чтения выполняется по номеру ЗМ и контрольной сумме. Помимо этого, в программе PDISK контролируется статусное слово устройства. В случае неудачной операции чтения работает аварийный алгоритм. Аварийные программы последовательно вводят дополнительные операции из числа следующих:

1. Повторение операции /три попытки/.
2. Повторное позиционирование головок /три попытки/.
3. Вывод головок на нужную позицию с нулевой дорожки /три попытки/.
4. Замена библиотеки ее дубликатом /имеется две копии/.

Если будет исчерпан аварийный алгоритм, произойдет останов по адресу 1727. Повторение его возможно с адреса SHORTC.

Программа CORREC перерабатывает ЗМ в соответствии с адресом его загрузки. Для построения достаточно простого алгоритма настройки модуля по месту загрузки

разработана методика построения ЗМ[4]. Все ЗМ транслируются для адреса загрузки АТ. Абсолютные адресные константы в ЗМ размещаются в конце страницы в поле ссылок. Программный текст отделяется от поля ссылок терминатором - нулевым кодом.

Программа CORREC перерабатывает поля ссылок всех страниц ЗМ. Каждый адрес A_i в поле ссылок, для которого выполняется условие $A_i \geq AT - 1000_8$, заменяется кодом A_i^*:

$$A_i^* = A_i + AZ - AT,$$

где AZ - адрес загрузки модуля.

Граничное значение AT-1000 хранится в ячейке CTOVRL. Принятое отличие его от AT позволяет использовать в ЗМ ссылки на 4 страницы, размещенные выше него. Это необходимо при работе в режиме оверлей либо для единых программ, компонуемых на РП из нескольких ЗМ.

Программа ABONEN обеспечивает программе пользователя прямой /минуя программу ДРП/ доступ к зафиксированной СП при повторных обращениях к ней. Для этого адрес входа в СП заносится в ячейку, адрес которой определяется монитором и равен (TPATT) +1. Заметим, что фиксируется на РП весь ЗМ. Если он содержит несколько СП, то прямые ссылки на них заполняются для каждой в отдельности при первом обращении.

Программа ZATIRK аннулирует прямые ссылки на фиксированные СП при зачеркивании TX1. Таких ссылок может быть 8 /по количеству независимых входов системы регистрации/. Данная операция сводится к занесению значения INTERP в 8 ячеек с шагом 10_8, начиная с адреса из ячейки ATABL.

Программа INTJMS обеспечивает передачу управления по виртуальному адресу с возвратом. Обращение имеет вид:

```
к       JMS  INTJMS
к + 1   NOMER          /номер СП.
```

Программа INTJMS перехватывает управление в ячейке EXIT при попытке вернуться из СП в монитор[4], после чего восстанавливает начальное содержание этой ячейки.

Программа LOADM введена для загрузки на РП данных /текстов, таблиц, каталогов и др/. Обращение имеет вид:

```
к       TAD  NOMER
к + 1   JMS  LOADM
```

Данная программа аннулирует обращение загрузчика к программе CORREC, после чего обращается к программе INTJMS.

Управление программой ДРП

Описываемый механизм ДРП работает автоматически на любом указанном участке ОЗУ. Удобства его использования существенно расширяются благодаря возможности управлять положением и длиной РП из СП. Ниже описываются операции управления рабочим полем. Мы будем говорить об операциях над РП, хотя на самом деле изменяется состояние ассоциативной таблицы.

1. Затирание РП2:

```
TAD  AHTX2
DCA  R
DCA  I  R,
```

где R - любая рабочая ячейка.

2. Затирание всего РП:

```
TAD  AHTX
DCA  AHTX2
TAD  AHTX
DCA  R
DCA  I  R
JMS  ZATIRK.
```

3. Смещение РП вверх /или вниз/ без изменения длины. Для этого в ячейку RABPOL следует занести новый адрес РП:

```
TAD  C1
DCA  RABPOL,
```

где (C1) - новый адрес РП. Данная операция управления должна сопровождаться затиранием старого содержания ТХ.

4. Расширение поля за счет верхней границы:

```
TAD C1
DCA RABPOL
TAD C2
DCA DLINTX.
```

где (C2) - новая длина ТХ. Длина задается в виде дополнения до единицы. Данная операция должна сопровождаться затиранием старого содержания ТХ.

5. Расширение поля за счет нижней границы:

```
TAD C2
DCA DLINTX
TAD C3
DCA AGRNTX,
```

где (C3) - новая нижняя граница ТХ, определенная по формуле:

(C3) = (AHTX) - (C2) + 1.

6. Компоновка на РП программ длиной более 4 страниц. В случаях, когда требуемая программа состоит из двух /и более/ ЗМ и по каким-либо причинам необходимо их одновременное присутствие в ОЗУ, вызывающая СП в качестве первого исполняемого кода должна содержать обращение к специальной программе. Функция этой служебной программы - обеспечить такое размещение ЗМ с вызывающей СП на РП, чтобы следом за ним поместился дополнительный текст нужной длины. На *рис. 4* приведен пример такой программы. В первой строке должна быть указана нужная длина дополнительного текста.

7. Любой ЗМ может содержать программу, работающую в режиме оверлей. Нет ограничений на последовательность, в которой должны вызываться оверлейные сегменты. Перед вызовом следующего сегмента работающий сегмент /или огранизующая секция, если она существует/ должен затереть в ТХ строки, соответствующие занятой им памяти. Для этого используется содер-


```
        SIGMA=7          /Суммарная длина
                         /дополнительных модулей
RESERV, HLT
        TAD SECTOR       /Учтена длина ЗМ
        TAD AHTX
        JMS TEST         /Длина всего РП доста-
                         /точна?
        SKP                    /Да
        JMP ERROR              /Нет. Уход на
                         /диагностику.
        TAD ATTX
        JMS TEST         /Длина свободной части
                         /РП2 достаточна?
        JMP I RESERV           /Да, можно вызывать
                               /дополнительные ЗМ
        TAD AHTX2              /Нет
        JMS TEST         /Длина всего РП2 доста-
                         /точна?
        JMP CALL               /Да
        JMS ZATIRK             /Нет. Затирается
                               /вся ТХ
        TAD AHTX
        DCA AHTX2
 CALL,  TAD AHTX2        /Затирается ТХ2
        DCA RESERV
        DCA I RESERV
        TAD NOMER1       /Вызывающая СП
        JMP INTERP       /загружается заново
 TEST,  HLT
        TAD DLINA
        CIA
        TAD AGRNTX
        SPA CLA
        ISZ TEST
        JMP I TEST
DLINA,  SIGMA
NOMER1, ...              /Номер вызывающей СП
```

Рис. 4. Пример программы, настраивающей положение ЗМ на РП так, чтобы ниже него оставалось заданное число свободных страниц.

жимое ячейки ATTX. Так, перед вызовом оверлейного сегмента длиной в N страниц должны быть выполнены команды:

```
    ...
    TAD  DLINA
    TAD  ATTX
    DCA  R
    DCA I  R
    ...
DLINA ,-N
```

Обсуждение

Описанные алгоритмы использованы в программном обеспечении эксперимента по исследованию $(n,\gamma f)$-реакции [5,6]. Программы ДРП и СЗ заняли 3 страницы в нулевом кубе памяти. Длина РП на разных этапах эксперимента /измерения и обработка/ попеременно устанавливалась равной 5 и 10_{10} страниц. Объем части библиотеки, необходимой в эксперименте [7], составил 64_{10} страницы /60 наименований СП/. В состав измерительного модуля входила ЭВМ с объемом ОЗУ 16К. Некоторые операции обработки, например, сортировка матрицы 512 К слов, требовали 13К слов для данных. Т.О., необходимая программа измерений и обработки при такой конфигурации не могла поместиться в ОЗУ. Существенных потерь времени в эксперименте из-за хранения программ на ВЗУ не происходило, т.к. даже при длине РП в 5 страниц система автоматически компоновала в ОЗУ набор программ, работавший затем несколько часов без изменений.

Сочетание ДРП с режимом интерпретации программы измерений [3] приводит к достаточно общим следствиям.

1. Объем памяти для конкретной программы измерений и системы обработки экспериментальных данных может быть ограничен малой величиной порядка 10 страниц практически безотносительно к сложности алгоритмов. Это упрощает расчет необходимой емкости ОЗУ на

этапе проектирования измерительного модуля, ибо на этом этапе программы, как правило, отсутствуют.

2. При фиксированном объеме ОЗУ действующая программа перестает быть консервативной, т.к. ее развитие более не сдерживается наличным объемом ОЗУ.

3. Данный подход может обеспечить существенно более высокий коэффициент использования оперативной памяти η.

В исполняемой программе можно выделить n частей, реализующих каждая некоторую частную функцию в общем алгоритме. Такая часть может быть представлена в виде СП и далее считаться неделимой. Представление о коэффициенте η для случая, когда нужные n частей программы находятся в ОЗУ, может дать величина

$$\eta = \frac{\sum_{i=1}^{n} M_i t_i}{\sum_{i=1}^{n} M_i \sum_{i=1}^{n} t_i},$$

где M_i - объем части программы, t_i - время, в течение которого работает данная часть.
В случае использования ДРП эта величина примет вид:

$$\eta_{\text{дрп}} = \frac{\sum_{i=1}^{n} M_i t_i}{(P_1 + P_2) \sum_{i=1}^{n} t_i},$$

где P_i - длина программы ДРП, P_2 - длина РП.

По мере усложнения методики эксперимента /а именно, такая тенденция наблюдается/ растет отношение $\eta_{\text{дрп}}/\eta$. Например, для системы [6] коэффициент использования памяти с введением ДРП увеличился в 7÷12 раз.

Мы полагаем, что эти замечания будут справедливы и для другого типа систем с низким коэффициентом использования процессора /например, АСУ/, существенным элементом которых является система обработки данных.

При использовании подобных алгоритмов в многомашинной сети [3] существенный выигрыш может быть

получен от применения иерархической структуры ВЗУ для программ. При управлении размещением программ в такой памяти может быть использован ассоциативный механизм, накапливающий информацию для управления в процессе работы.

Программы написаны на языке SLANG-3 /7/.

Авторы благодарны Г.П.Жукову, В.П.Ширикову, В.В.Галактионову за полезные обсуждения, Л.Б.Пикельнеру за поддержку данной работы, М.А.Фурман, участвовавшей в разработке блок-схемы программы ДРП, Н.П.Копыловой за помощь в оформлении рисунков.

Литература

1. *В.Ф.Ляшенко. Программирование для ЦВМ с системой команд типа М-20. М., Сов. радио, 1974.*
2. *Дж.Шеффлер, Л.Броннер. В сб. Мини-ЭВМ, М., Мир, 1971, стр. 51.*
3. *К.Дади и др. ОИЯИ, 10-9060, Дубна, 1975.*
4. *К.Дади и др. ОИЯИ, Р10-9484, Дубна, 1976.*
5. *З.Длоугы и др. ОИЯИ, Р3-9613, Дубна, 1976.*
6. *A.A.Bogdzel, J.Brankowcki, K.Dady et al. Second Intrn. Symp. on CAMAC in Computer Applications. Brussel, 14–16 Oct., 1975, IU, 2–12.*
7. *В.В.Галактионов. ОИЯИ, 10-5911, Дубна, 1971.*

Рукопись поступила в издательский отдел
8 июля 1976 года.