АРХИТЕКТУРА ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ С ПРОГРАММИРУЮЩЕЙ СТРУКТУРОЙ
(Вычислительные системы)
1980 год Выпуск 82

УДК 681.142.2:681.31

ПРОГРАММИРУЮЩАЯ СИСТЕМА Р-ЛЯПАС-БЭСМ-6. РЕАЛИЗАЦИЯ ПЕРВОЙ ОЧЕРЕДИ

В.А.Воробьев, В.М.Горелик, Э.А.Монахова

§1. О целях и принципах разработки

1.1. Первая очередь программирующей системы (ПС) Р-ЛЯПАС-БЭСМ-6 предназначена для

- реализации программ, написанных на базовом уровне, макроуровне и системном уровне инструментального языка Р-ЛЯПАС [1];
- накопления в архиве и автоматического использования модулей макроуровня и системного уровня;
- обеспечения дальнейшего наращивания программирующей системы, т.е. разработки второй очереди.

Короче говоря, в ПС-Р-ЛЯПАС-БЭСМ-6 реализована та программа, которая была намечена в [1] при описании языка Р-ЛЯПАС.

1.2. В соответствии с вышесказанным система состоит из трех основных частей:

- о п е р а ц и о н н о й части, преобразующей з а к а з программиста в машинную программу или систему машинных программ;
- а р х и в а, в котором накапливаются модули, написанные на языке Р-ЛЯПАС;
- с е р в и с н о й части, обеспечивающей наращивание и коррекцию архива.

1.3. В основу разработки системы были заложены следующие принципы.

П р е е м с т в е н н о с т ь. ПС-Р-ЛЯПАС-БЭСМ-6 строится из блоков ПС-ЛЯПАС-70 для ЭВМ БЭСМ-6 [2] и ПС-Р-ЛЯПАС для ЭВМ "Минск-22" [3], взятых с минимальными дополнениями. В свою очередь, готовая система воспринимает программы, написанные для систем предшественников. Таким образом, созданные ПС-Р-ЛЯПАС-БЭСМ-6 сводится к

наращиванию готовых систем и загрузке ЭВМ методом динамической раскрутки.

С т а т и ч н о с т ь. При разработке системы авторы стремились максимально использовать возможности статического планирования, которое избавляет готовый продукт (т.е. систему машинных программ) от динамического управления ресурсами памяти. Статическими в данной системе являются:

- планирование памяти, отводимой под машинные программы;
- распределение базовых имен по модулям;
- планирование памяти, отводимой под системные и собственные комплексы.

Остальная оперативная память предоставляется программисту как единый комплекс, начало и мощность которого выдаются по завершении трансляции. Эта память может распределяться пользователем как статически (вводом специальных комплексов А и В), так и динамически (вычислением значений элементов комплексов А и В). Аналогично обстоит дело и с внешней памятью ЭВМ.

Принятый способ планирования памяти приводит к некоторым издержкам, которые, на наш взгляд, вполне компенсируются отсутствием потерь времени и памяти на динамическое управление.

З а в е р ш е н н о с т ь п р о д у к т а. Продукт системы представляет собой СИСТЕМУ готовых машинных программ, помещаемую на магнитную ленту (МЛ) вместе с планом ее размещения в ОЗУ и на магнитных барабанах (МБ). Для реализации СИСТЕМЫ продукт вызывается с МЛ специальной программой ЗАПУСК, которая размещает ее в ОЗУ и МБ согласно плану и передает управление головному сегменту.

Н а р а щ и в а е м о с т ь. Автоматизация системного уровня программирования позволяет без особого труда ввести в систему новые блоки или переделать ее целиком. С этой целью все модули системы включены в архив, и на последнем этапе загрузки система строит сама себя из этих модулей.

§2. Статическая модель системы программ

В основу реализации ПС-Р-ЛЯПАС-БЭСМ-6 положена следующая единая статическая модель системы программ, записанной на инструментальном языке Р-ЛЯПАС.

2.1. Система представляет собой упорядоченное множество сопрограмм [1]. Головная программа, с которой система начинает работу, имеет номер 0 и вводится в ЭВМ первой. Порядок остальных сопрограмм произволен.

В свою очередь, любой сопрограмме может быть сопоставлено д е р е в о, каждой вершине которого соответствует модуль. Модуль j глубины d подчинен модулю i глубины d-1, если и только если макрос j употребляется в прямой форме в теле модуля i. Подчинение модулей изображается ветвями дерева. Если в теле модуля i глубины d-1 упомянуто несколько макросов, то вершины соответствующих модулей располагаются на ярусе глубины d слева направо в порядке упоминания этих макросов в последовательности операций модуля i.

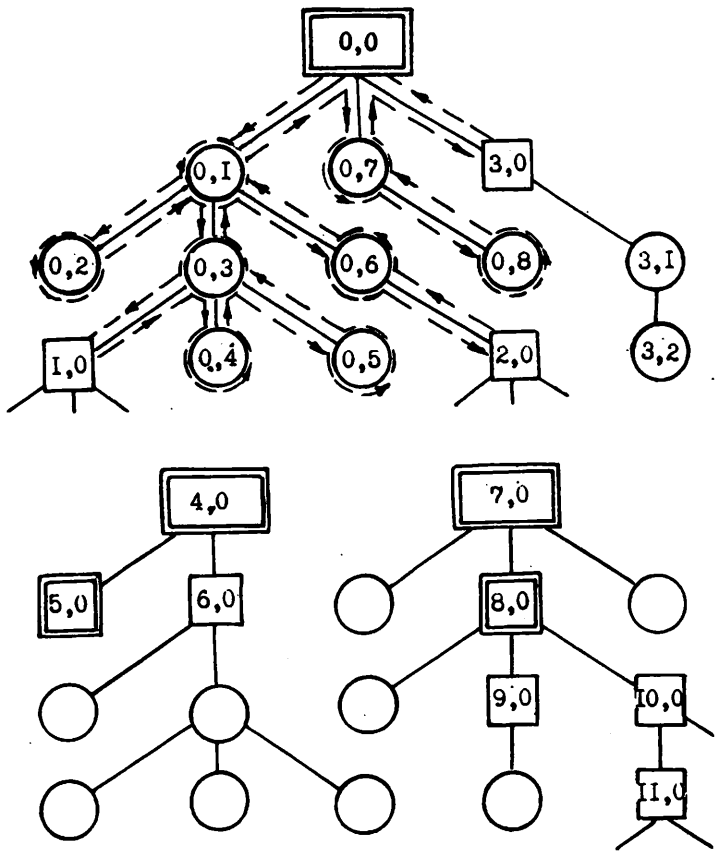

Рис. I

На рис.I изображены поддеревья трех сопрограмм некоторой системы. Кружками изображены модули макроуровня, прямоугольниками - сегменты, двойной линией обведены прямоугольники, соответствующие головным и жестким сегментам [I].

2.2. Нумерация вершин дерева должна быть такой, чтобы фиксировать определенный порядок подстановки модулей вместо соответствующих макросов. Р-КОМПИЛЯТОР, реализующий эту процедуру, совершает обход дерева, как это показано на рис.I, добираясь только до сегментов. (Подробнее о процедуре компиляции см. [4].) Обход дерева сегмента откладывается,так как вместо подстановки тела сегмента в программу Р-КОМПИЛЯТОР (точнее, его жесткий сегмент - СЕГМЕНТАТОР) вписывает обращение к ИСПОЛНИТЕЛЮ, сообщающее этой сопрограмме номер требуемого модуля. Таким образом, вершины дерева нумеруются парой (i,j) , где i - номер сегмента, j - номер макроса в дереве данного сегмента. Соответствующая нумерация показана на рис.I. Нумерация сегментов, если все операции макроуровня убрать из дерева, показана на рис.2. Именно в этом порядке они и будут обрабатываться Р-КОМПИЛЯТОРОМ,и, следовательно, в этом же порядке их логичнее всего расположить в памяти ЭВМ.

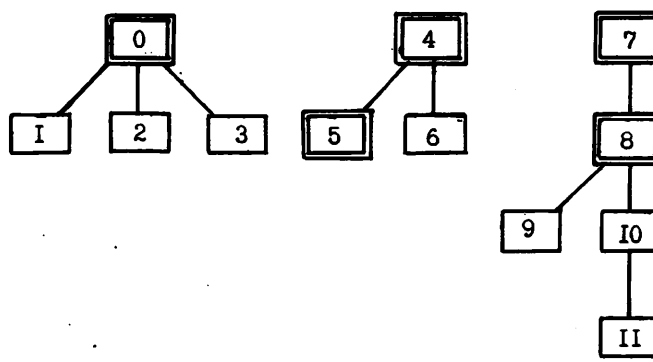

Рис. 2

Расположение модулей в порядке компиляции и с указанием глубины называется ниже К-п р о г р а м м о й. Последняя формируется во внешней памяти, а в ОЗУ присутствует только тот сегмент,который обрабатывается в данный момент.

2.3. Сопоставим каждому i-у сегменту i-й разряд булева вектора и поместим единицу в те разряды, которые соответствуют головным и жестким сегментам. Полученная логическая константа называется в е к т о р о м головных и жестких сегментов. Вектор головных и жестких сегментов системы, изображенный в нашем примере, имеет значение I000II0II000. В соответствии со значением вектора головных и жестких сегментов ЗАГРУЗЧИК программирующей системы должен установить порядок следования машинных программ сегментов при расположении их в оперативной памяти.

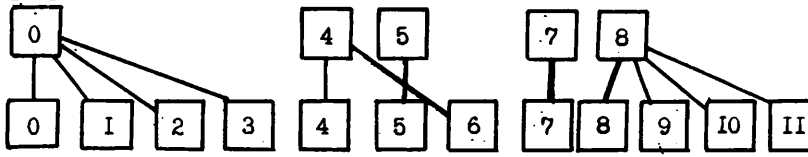

Рис. 3

На рис. 3 показано исходное рабочее положение машинных программ сегментов разбираемого примера. Верхний ряд - сегменты, помещенные в ОЗУ, нижний - на МБ. Дугами соединены зависимые сегменты, которые будут вытеснять друг друга в процессе работы. Если не считать дублей головных и жестких сегментов, хранящихся на МБ, рис.3 дает представление о графе зависимостей сегментов по месту в ОЗУ, так как каждая компонента связности этого графа размещается на одном и том же программном поле памяти. На рис.3 такие компоненты суть: (0,I,2,3), (4,6), (5), (7), (8,9,I0,II). Если L - множество номеров сегментов, L_k - множество номеров из L, принадлежащих k-й компоненте, K - множество номеров компонент, такое что $L = \bigcup_{k \in K} L_k$, то объем памяти ОЗУ, занимаемый k-й компонентой системы машинных программ, равен $\max_{i \in L_k} \{l_i\}$, где l_i - мощность машинной программы i-го сегмента. Объем системы в ОЗУ - $\sum_{k \in K} \max_{i \in L_k} \{l_i\}$. Объем системы на МБ равен $\sum_{i \in L} l_i$.

Теперь не составляет труда разместить в ОЗУ системные и собственные комплексы, мощность которых программист задает в паспортах модулей. Адрес и размеры оставшегося свободного поля памяти ЗАГРУЗЧИК сообщает программисту.

2.4. Как следует из описания системного уровня и макроуровня языка Р-ЛЯПАС, при компиляции сегментов возникает задача раз-

вязки модулей по именам их внутренних объектов. Причем количество базовых имен языка точно известно и составляет 32 имени комплексов, 128 имен типа Л (логических) и 32 имени типа D (действительных). Специальные комплексы являются заведомо системными, и их имена не подлежат изменениям. Используя предлагаемую статическую модель, рассмотрим действия Р-КОМПИЛЯТОРА при решении этой задачи.

Каждому i-у модулю дерева сегмента сопоставляется четыре множества внутренних объектов: системные - S_i, собственные - M_i, рабочие - R_i и служебные - V_i с мощностями s_i, m_i, r_i и v_i, соответственно. Изобразим весь наличный р е з е р в имен логическим комплексом Р мощности 6, в котором нулевой элемент соответствует комплексам, четыре следующих - логическим переменным, а пятый - числам. Единицами в комплексе Р отмечены те имена, которые может использовать Р-КОМПИЛЯТОР при создании программы базового уровня из модулей данного сегмента. Ясно, что при этом тип объекта не должен меняться, т.е. резерв каждого типа должен быть выделен отдельно.

Очевидно, что элементы множества $S = \bigcup_{i \in I} S_i$, где I - множество номеров модулей, т. е. все имена системных объектов не должны быть отмечены в Р, поскольку их изменение запрещено. Остается выделить в Р резервы для имен собственных, рабочих и служебных объектов. Общее число собственных, очевидно, есть $m = \sum_{i \in I} m_i$ и может быть сразу выделено из общего резерва в резерв собственных имен, который расходуется по мере включения модулей в программу. Аналогично выделяется резерв служебных имен общим объемом $v = \max_{i \in I} \{v_i\}$, но, в отличие от собственных, эти имена предоставляются каждому модулю. И наконец, мощность резерва рабочих имен есть $r = \max_k \{ \sum_{i \in I_k} r_i \}$, где $I_k \subseteq I$ - номера модулей, лежащих на пути от корня дерева сегмента до k-го листа дерева ($I = \bigcup_k I_k$, $I_k \cap I_l \neq \emptyset$). При обходе дерева сегмента Р-КОМПИЛЯТОР расходует резерв рабочих имен в момент увеличения глубины компиляции, а в момент уменьшения глубины возвращает в резерв глубины $d-1$, те имена, которые были заняты на глубине d.

Нетрудно видеть, что все эти рассуждения справедливы и для модели системы программ в целом. Таким образом, выделение резервов собственных, рабочих и служебных имен каждому сегменту систе-

мы может быть сделано еще до компиляции. Эту задачу решает специальный модуль системы – ПЛАНИРОВЩИК.

2.5. Только что описанная процедура переименования объектов с целью сохранения их значений в соответствующих областях действия приводит к задаче сохранения структуры этих объектов. Как следует из [1], структура внутренних объектов модуля задается в основном статическими операциями согласования размерностей, причем для стандартных объектов соответствующие операции вообще отсутствуют. При компиляции имена разных рабочих и служебных объектов могут совпасть, а размерности перепутаться.

Чтобы избежать этого, в системе вводится специальный модуль – РЕДАКТОР, который расставляет в исходном тексте дополнительные операции согласования размерностей рабочих и служебных объектов. Размерность служебных объектов стандартной размерности поддерживается операцией ⊛ 40 → $\bar{n}$ ⊛, где $\bar{n}$ – список имен стандартных служебных объектов. Эта операция вписывается сразу за § 0 и после каждой операции согласования размерностей, имеющейся в теле модуля. Перед концом модуля восстанавливается стандартная размерность всех служебных и рабочих объектов модуля.

2.6. Сопоставим теперь каждой вершине дерева модели т и п, а д р е с и м о щ н о с т ь соответствующего модуля в архиве или в другом месте памяти, если модуль введен с заказом программиста. Эта информация позволит нам заранее вычислить место каждого модуля в К-программе и так упорядочить вызов модулей из архива, что он произойдет без реверсов магнитной ленты. Процедуру формирования К-программы в соответствии с этой идеей производит модуль программирующей системы – СБОРЩИК.

2.7. Итак, мы уже можем точно сказать, какая информация нам потребуется в каждой вершине дерева, чтобы решать основные задачи системы. Искомое описание вершины дерева называется с и с т е м н ы м п а с п о р т о м модуля и содержит следующие данные:

идентификатор модуля (# N_1 N_1), мощность модуля, мощность паспорта, список системных операндов;

число собственных комплексов, векторов, чисел;

число рабочих комплексов, векторов, чисел;

число служебных комплексов, векторов, чисел;

список макросов и сегментов модуля в порядке их упоминания в теле;

список идентификаторов сопрограмм модуля;

адрес модуля.

Вся эта информация, за исключением адреса, проставляется соответствующими кодами и упаковывается в ячейки ЭВМ так, что данные, разделенные "запятой", следуют друг за другом, а данные, разделенные "точкой с запятой", начинаются с новой ячейки. Последнее правило касается и списка идентификаторов макросов, сегментов и сопрограмм.

2.8. Системные паспорта модулей заказа получаются в процессе редактирования, а остальные извлекаются из архива и упорядочиваются в соответствии с принятой ранее (п.2.2) нумерацией БИБЛИОГРАФОМ программирующей системы. Для облегчения работы БИБЛИОГРАФА архив системы состоит из двух частей: к а т а л о г а - списка системных паспортов и ф о н д а - списка отредактированных модулей. Термины каталог и фонд будут в дальнейшем употребляться для любых неупорядоченных списков системных паспортов и модулей, прошедших редакцию. Каталог предварительно переписывается на барабан, и БИБЛИОГРАФ, пробегая систему ссылок, извлекает из него все паспорта, необходимые для построения модели системы. Отсутствие тел модулей позволяет поместить в ОЗУ всю эту информацию.

2.9. Очевидно теперь и устройство сервисной части системы - АРХИВАРИУСА. Основу его составляет тот же РЕДАКТОР, продукция которого помещается на МЛ в порядке поступления отлаженных программ. Коррекция архива, т.е. удаление из него ненужных модулей или модификация некоторых из них, достигается простым затиранием ненужной части каталога с помощью модуля КОРРЕКТОР, включенного в АРХИВАРИУС.

2.10. Таким образом, рассматривая и постепенно уточняя статическую модель системы программ, мы логически вывели основные задачи, решаемые в системе, и те способы, которыми эти задачи решались. Осталось упомянуть только, что получение машинных программ возлагается на Л-ТРАНСЛЯТОР системы ЛЯПАС, встроенной в ПС-Р-ЛЯПАС, а перевод с языка Р-ЛЯПАС на ЛЯПАС делает Р-ТРАНСЛЯТОР. Оба этих блока - трансляторы компилирующего типа, но термин компиляция используется в ЛЯПАСе для обозначения процедуры перехода с макроуровня на базовый уровень.

Следует, кроме того, заметить, что возможности статической модели далеко не исчерпаны. Учитывая логические связи между модулями и используя тот факт, что машинные программы сегментов перемещаемы, можно было бы экономичнее распорядиться памятью, отводимой в ОЗУ под машинные программы. Слегка усложнив ИСПОЛНИТЕЛЬ, так

чтобы он сохранял не только машинные программы, но и данные сегментов, можно было бы сделать резерв базовых имен неограниченным. Присовокупив сюда же граф-схемы модулей, можно минимизировать число внутренних объектов. При необходимости эти задачи могут быть решены во второй очереди программирующей системы вместе с развитием входного языка.

§3. Структура и функционирование программирующей системы

3.1. ПС-Р-ЛЯПАС представляет собой систему из трех сопрограмм, подобную той, которая была рассмотрена в предыдущем пункте. Сюда входят:

- головная программа, или собственно программирующая система;
- сопрограмма ИСПОЛНИТЕЛЬ, организующая смену сегментов программирующей системы в ОЗУ ЭВМ БЭСМ-6;
- сопрограмма АВОСТ, выдающая информацию в случае обращений к несуществующим системным полюсам [1].

Две последние сопрограммы появляются в процессе развертывания программирующей системы автоматически: ИСПОЛНИТЕЛЬ - в случае наличия в ней мягких сегментов [1], АВОСТ - в случае наличия в ней переходов к системным полюсам. Такое автоматическое появление этих сопрограмм отражает тот факт, что разрабатываемая система является своим собственным продуктом в процессе раскрутки.

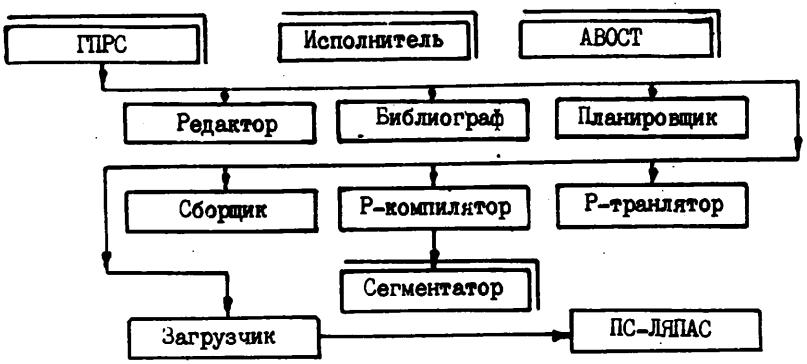

Рис. 4

3.2. Статическая модель программирующей системы содержит 66 модулей, из которых 64 принадлежат ее головной программе, а ИСПОЛНИТЕЛЬ и АВОСТ модулей не содержат. На рис.4 представлено дерево сегментов головной программы системы, подобное тому, как показано на рис.2. Единственный модуль макроуровня, который отображен на рис.4, это ПС-ЛЯПАС, используемый ЗАГРУЗЧИКОМ и организующий исполнение Л-ТРАНСЛЯТОРА независимо от ИСПОЛНИТЕЛЯ. Таким образом, только сегменты ПС-ЛЯПАС не входят в стандартную схему и не имеют отображения в модели (им не присвоены номера). При реализации сегментов ПС-ЛЯПАС последние рассматриваются и как данные, и как программы. В связи с этим Л-ТРАНСЛЯТОР вызывается в рабочее поле памяти ЗАГРУЗЧИКА как обычный комплекс, а затем реализуется и использует под свои внутренние комплексы рабочее поле того же ЗАГРУЗЧИКА.

Развертывание системы в рабочее состояние осуществляется точно так же, как и любого ее продукта, фиксированного на МЛ (см.§4).

3.3. Функциональную схему операционной части программирующей системы удобно представлять, следуя [5], в виде двудольного графа. Вершины графа разбиты на два класса: данные и программы. В табличном представлении графа (см. табл.I) строки соответствуют данным, столбцы - программам. На пересечениях строк и столбцов ставятся знаки отношения соответствующих данных и программ: знак "-" указывает, что данные используются программой, "+" - продуцируются программой, "-/+" - перерабатываются (изменяются) программой, "." - остаются без изменения. Пустое пересечение означает, что соответствующие данные не имеют отношения к программе. Строго говоря, знаки "-/+" и "." являются избыточными и введены здесь лишь для отображения загрузки памяти ЭВМ. Знак "-/+" соответствует различным данным, занимающим одно и то же место в памяти, так что входной массив затирается выходным. Знак ".", как нетрудно видеть, заполняет все позиции данной строки от первого "+" до последнего "-", фиксируя тем самым с л е д данных в системе. Длина следа зависит от порядка выполнения программ, который в общем случае неоднозначен. В рассматриваемом случае порядок выполнения сегментов (за исключением сопрограмм АВОСТ и ИСПОЛНИТЕЛЬ) совпадает с порядком столбцов табл. I. Очевидно, что ПЛАНИРОВЩИК и СБОРЩИК можно переставить местами, что существенно изменит следы данных, находящихся в оперативной памяти.

Таблица I

Функциональная схема операционной части ПС-Р-ЯПАС

№ строки	Данные	Сегменты ПС-Р-ЯПАС									Память	
											Тип ЗУ	
		АВОСТ	ИСПОЛНИТЕЛЬ	РЕДАКТОР	БИБЛИОГРАФ	ПЛАНИРОВЩИК	СБОРЩИК	Р-КОМПИЛЯТОР	Р-ТРАНСЛЯТОР	ЗАГРУЗЧИК	основной промежуточный	ОЗУ
I	Общая часть	-	-/+	-/+	-/+	-/+	-/+	-/+	-/+	-/+	НМЛ ОЗУ	5000
2	Управляющее слово			-	-	-	-	-	-	-	ЛФК ОЗУ	/I
3	Заказ			-							ЛФК ОЗУ	2000
4	Каталог архива				-						НМЛ НМБ	2000
5	Фонд архива						-			-	НМЛ ОЗУ	2000
6	ПС-ЯПАС-70									-	НМЛ НМБ	/I,20
7	Каталог продукта			+	-/+	-					ОЗУ	20000
8	Фонд заказа			+	.	.	-				НМБ	2000
9	Вектор ГС, ГЖС*)				+	-	.	-/+	.	-	ОЗУ	/I
I0	Типы модулей				+	.	-				ОЗУ	2000
II	Адреса модулей				+	.	-				ОЗУ	2000
I2	Мощности модулей				+	.	-				ОЗУ	2000
I3	Модель продукта				+	-					ОЗУ	30000
I4	Резервы сегментов					+	.	-			ОЗУ	I300
I5	Адреса сегментов						+	-/+	-/+	-/+	НМЛ ОЗУ	4I
I6	К-программы						+	-			НМБ	2000
I7	Р-программы							+	-		НМБ	2000
I8	Л-программы								+	-	НМБ	2000
I9	М-программы									+	НМЛ НМБ	60000
20	Системные полюса**)									+	НМЛ ОЗУ	/I

*) ГС, ГЖС - головных сегментов, головных и жестких сегментов.

**) см. [I].

Поясним теперь содержательно смысл данных, отображенных в табл. I.

Общая область занимает первые 1000_8 и последние 4000_8 ячеек ОЗУ. Сюда входят простые операнды, специальные комплексы, некоторые машинные программы общего назначения, тексты, комплекс параметров ПС-Р-ЛЯПАС, исходный резерв имен, комплекс параметров продукта системы, т.е. план его размещения на МЛ, МБ и в ОЗУ, и, наконец, буфер ввода-вывода (2000_8 ячеек). После срабатывания ПС-Р-ЛЯПАС первые 1000_8 и последние 2000_8 ячеек ОЗУ содержат информацию, необходимую для запуска и реализации продукта, т.е. являются его общей областью и записываются на МЛ, отведенную продукту. Буфер ввода-вывода продукта сохраняется на том же месте, что и в ПС-Р-ЛЯПАС.

Управляющее слово задает режим работы программирующей системы.

Заказ - программный материал, вводимый пользователем. Сюда входит головная программа заказываемой системы и неупорядоченный набор модулей, написанных пользователем впервые. Заказ разбит на блоки, включающие целое число модулей и не превышающие объема буфера ввода.

Понятия каталог, фонд, вектор головных сегментов и головных и жестких сегментов, резерв, тип, адрес и мощность модуля, резерв сегмента были введены в §2 настоящей статьи при описании статической модели.

Модель продукта - введенная ранее статическая модель системы программ. В памяти ЭВМ модель продукта представляет собой список системных паспортов модулей продукта. Каждому паспорту предшествует ячейка с указанием глубины соответствующего модуля. Порядок расположения паспортов в модели соответствует порядку нумерации вершин статической модели.

К-программы для каждого сегмента получаются из моделей сегментов заменой системного паспорта полным описанием модуля. Р-программы - программы на базовом уровне языка Р-ЛЯПАС, Л-программы - программы на первом уровне ЛЯПАСа, М-программы - программы на машинном языке.

Все эти программы получаются отдельно для каждого сегмента и хранятся на МБ.

3.4. Из предыдущего должно быть достаточно ясно, что ПС-Р-ЛЯПАС функционирует, как любая система, описанная на языке Р-ЛЯПАС. Рассмотрение статической модели программирующей системы и табл. I дает достаточно ясное представление об основных функ -

циях каждого сегмента. Ниже мы остановимся на распределении памяти под данные и программы системы.

3.5. Для учета фактора загрузки памяти в табл. I введены три дополнительных столбца, характеризующих расположение данных в памяти ЭВМ: основной носитель данных, промежуточный носитель, на котором данные хранятся в промежутках между использованиями, и объем ОЗУ, необходимый для операции над данными. Очевидно, что не все данные, используемые в программе, обязательно будут вызваны одновременно и займут отдельную память. В случае, когда данные i и j занимают одно и то же место в памяти, указывается максимальный объем (например, для i), а в строке j данных меньшего объема дается ссылка вида /i на строку с номером i. Данные с такой ссылкой не занимают отдельной памяти и в дальнейших расчетах не учитываются. Очевидно также, что ссылка может указывать на некоторую совокупность данных, на месте которых можно расположить информацию соответствующей строки (например, /k, l, m).

Рассматривая теперь столбец табл.I, мы можем подсчитать объем ОЗУ, необходимый системе в момент исполнения соответствующей программы. Искомая величина есть сумма объемов ОЗУ всех данных, помеченных в столбце знаками "+", "-", "-/+", и объемов ОЗУ данных, помеченных знаком "." и имеющих ОЗУ в качестве промежуточного носителя. Для полноты сюда необходимо добавить еще объем рабочего комплекса программы, необходимый для разрешения ее внутренних промежуточных результатов. Полная картина загрузки ОЗУ данными и программами системы дана в табл. 2.

Учитывая теперь статическое планирование загрузки программ в ОЗУ, можно подсчитать максимальную оперативную память, необходимую для работы системы. Она равна сумме максимального объема данных (строка 7 табл.2) и максимального объема программ, одновременно размещаемых в ОЗУ (строки, отмеченные звездочкой). Общий объем программ ПС-Р-ЛЯПАС составляет более 110000_8 команд, если учесть, что часть данных ЗАГРУЗЧИКА составляют примерно 10000_8 команд ПС-ЛЯПАС и в одной ячейке БЭСМ-6 расположено 2 команды.

§4. Развитие ПС-Р-ЛЯПАС

Программирующая система Р-ЛЯПАС на БЭСМ-6 была развита частично из готовых блоков ПС-Р-ЛЯПАС "Минск-22" методом динамической раскрутки. В данном случае этот метод можно представить последовательностью следующих этапов развития системы, начиная с момента принятия решения о ее создании.

№ строки	Сегменты ПС-Р-ЛЯПАС	Данные		Машинные программы	Загрузка ОЗУ
		основные	рабочие		
0	ГПС[1]-Р-ЛЯПАС	5000	-	I565	6565
I	Редактор	3I000	50	2240	333I0
2	Библиограф	67000	I00	I357	70457
3	Планировщик	66300	652	I332	70504
4	Сборщик	2234I	4000	662	27223
5	Р-компилятор	I234I	I0000	4067	26430
6	Р-транслятор	II04I	700	42I6*	I5257
7	Загрузчик	6704I	I000	2I72	72232
8	Сегментатор	-	-	I443*	I443
9	Исполнитель	-	-	336*	336
I0	Авост	-	-	2I5*	2I5
II	Архивариус	3I000	50	3000	34050
I2	Запуск	6434	3000	2000	II434
	max загрузка ОЗУ	6704I	I000	6434	76475
	Общий объем	5I3773		4005I	554044

4.I. Проектирование системы на уровне основных блоков и связей между ними. Освоение базовой системы ПС-ЛЯПАС. Принятие основных решений, установление ограничений и принципов развития системы.

4.2. Разработка основных модулей системы. Часть модулей при этом просто переносилась с носителей ЭВМ "Минск-22" на носители БЭСМ-6 (с перфолент на перфокарты). Часть из них подвергалась существенным дополнениям. Многие модули писались и отлаживались заново. На этом этапе основные модули системы (такие как РЕДАКТОР, БИБЛИОГРАФ и т.д.) отлаживались с помощью ПС-ЛЯПАС, как обычные программы второго уровня языка ЛЯПАС. При этом каждый модуль отлаживался на специфическом для него тесте и был рассчитан на обработку одной программы (сегмента).

4.3. Разработка сегментов ПС-Р-ЛЯПАС. Основные модули системы использовались для построения сегментов программирующей системы. Последние выполняют те же функции, но уже не на отдельной программе, а на некотором множестве сегментов, хранимых на МБ или МЛ.

I) ГПС - головная программа системы.

Таким образом, на этом этапе решались основные проблемы взаимодействия системы с внешними накопителями.

4.4. Загрузка первой версии системы. В принципе для этой цели можно использовать ЗАГРУЗЧИК, разработанный как один из модулей системы. Однако, во-первых, ЗАГРУЗЧИК системы использует только Л-ТРАНСЛЯТОР ПС-ЛЯПАС, а все загружаемые сегменты написаны на втором уровне ЛЯПАСа и требуют работы всей ПС-ЛЯПАС в целом; во-вторых, на вход ЗАГРУЗЧИКА программирующей системы должны подаваться продукты работы всех предыдущих ее сегментов, которые как раз и надо еще загрузить. Первое из этих препятствий легко преодолевается простой заменой модуля ЗАГРУЗЧИКА, который обращался к ТРАНСЛЯТОРУ, на модуль, обращающийся к ПС-ЛЯПАС в целом. При принятом в ЛЯПАСе модульном способе построения любой программы эта замена не составляет особого труда. Таким образом, был получен СИСТЕМНЫЙ ЗАГРУЗЧИК.

Второе препятствие потребовало вручную организовать требуемую входную информацию. При этом ситуация в ЭВМ в момент запуска СИСТЕМНОГО ЗАГРУЗЧИКА выглядела точно такой же, какой ее должны были создать отсутствующие блоки. Это обстоятельство позволило заодно проверить и правильность принятых решений по организации систем программ. Этот и все дальнейшие этапы представлены на рис. 5.

4.5. Развертывание АРХИВАРИУСА. Загруженная в ЭВМ и зафиксированная в МЛ ПС-Р-ЛЯПАС используется для получения сервисной системы АРХИВАРИУС. АРХИВАРИУС фиксируется на системной ленте наряду с программирующей системой. Таким образом, ПС-Р-ЛЯПАС-I развертывает свою первую систему - продукт.

4.6. Формирование архива ПС-Р-ЛЯПАС. Первая порция модулей, запоминаемых в архиве системы - модули самой системы, из которых удалены все ручные (внесенные на этапе 4) добавки. В результате первая версия ПС-Р-ЛЯПАС может воспроизвести себя саму уже автоматически.

4.7. Получение второй версии ПС-Р-ЛЯПАС. Головная программа системы пишется уже как обычная программа системного уровня на языке Р-ЛЯПАС и вводится как исходная информация к ПС-Р-ЛЯПАС-I. В результате получится вторая версия, несколько отличающаяся от первой, так как она получена автоматически.

4.8. Окончательная проверка системы состоит в том, что предыдущий этап повторяется уже для ПС-Р-ЛЯПАС-II. Если в системе нет

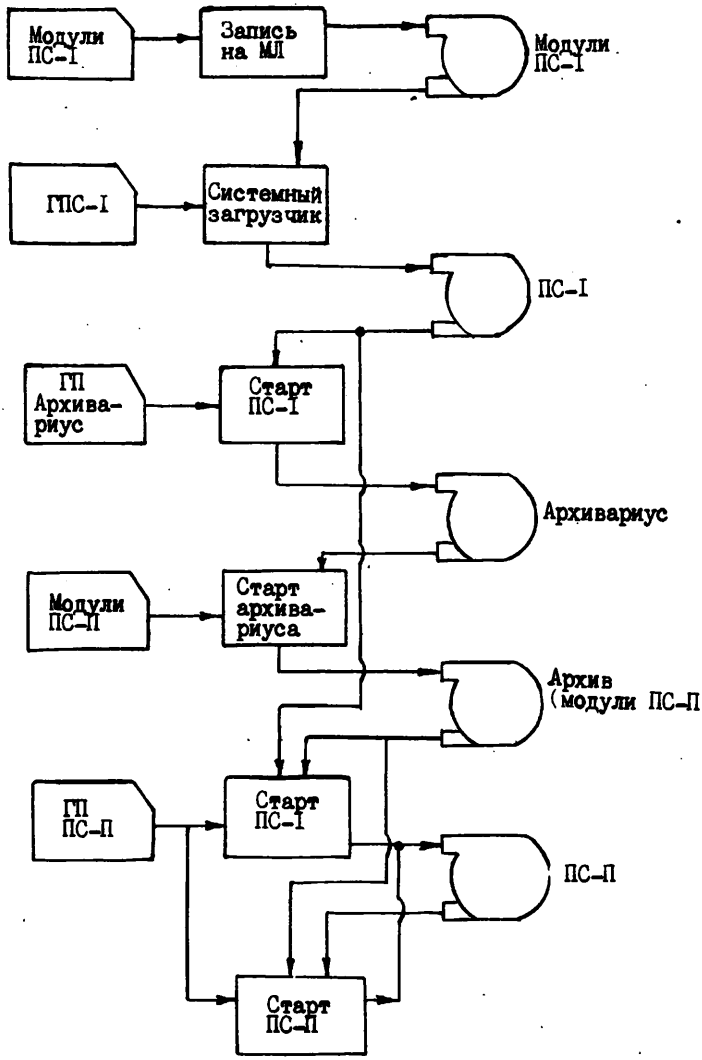

Рис. 5. Динамическая раскрутка ПС-Р-ЛЯПАС.

ошибок, то ее продуктом будет в точности такая же система. Дальнейшая прогонка системы через самое себя уже ничего не изменит, т.е. динамическая раскрутка достигла неподвижной точки.

4.9. Если изменить головную программу ПС-Р-ЛЯПАС-П или какие-то ее модули, находящиеся в архиве, то этапы 7 и 8 можно повторить еще раз и получить еще одну версию программирующей системы. Таким образом, кроме накопления архива модулей, возможен еще один путь совершенствования системы - дальнейшая динамическая раскрутка. Это, однако, относится уже к разработке второй очереди ПС-Р-ЛЯПАС на БЭСМ-6.

Литература

1. ВОРОБЬЕВ В.А. Программирующая система Р-ЛЯПАС. Инструментальный язык. -В кн.: Вопросы теории и построения вычислительных систем (Вычислительные системы, вып. 80). Новосибирск, 1979, с.97-121.

2. ЛЕВАШНИКОВ А.А. Система ЛЯПАС-70 для БЭСМ-6. Инструкции пользователю и программисту. -Томск, 1975 (Сиб. физ.-тех. ин-т).

3. ВОРОБЬЕВ В.А., ПЕТРОВА Э.А. Программирующая система ЛЯПАС-Р-ЛЯПАС для "Минск-22" (инструкция по эксплуатации). -В кн.: Вычислительные системы. Вып. 59. Стандартные программы обработки информации, Новосибирск, 1974, с.162-179.

4. ЗАКРЕВСКИЙ А.Д. Компилятор ЛЯПАСа. -В кн.: Логический язык для представления алгоритмов синтеза релейных устройств. -М., 1966, с. 70-74.

5. ЗАКРЕВСКИЙ А.Д., ТОРОПОВ Н.Р. Программирующая система ЛЯПАС-М. -Минск: Наука и техника, 1978. -240 с.

Поступила в ред.-изд.отд.
3 октября 1980 года