ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ
1972 год Выпуск 51

УДК 681.31:681.3.06

МОДЕЛИРОВАНИЕ СИСТЕМЫ ПАРАЛЛЕЛЬНЫХ ПРОЦЕССОВ НА Р-ЛЯПАСе

В.А. Воробьев

Предлагается набор операторов II-го уровня Р-ЛЯПАСа, образующих в совокупности полный язык квазипараллельного программирования. Возможности языка могут наращиваться применительно к классу систем, моделируемых пользователем.

Предварительные замечания

0.1. При проектировании вычислительных систем (ВС) рассматриваются относительно независимые компоненты проекта:

а) класс задач, для решения которых проектируется ВС;

б) математическое обеспечение системы;

в) структурная схема ВС (граф связей, характеристики каналов, способы коммутации и т.д.);

г) характеристики отдельных блоков ВС (быстродействие, объем памяти и т.п.).

Цель проектирования состоит в том, чтобы при заданных ограничениях на некоторые компоненты найти остальные так, чтобы удовлетворить требованиям технического задания. При этом возникает задача оценки качества проекта (определения праксеоло-

гических характеристик: эффективности использования оборудования, надежности, живучести, стоимости и т.п.).

Достаточно универсальным методом, позволяющим оценить эффективность ВС, в настоящее время является моделирование на ЦВМ.

0.2. В данной работе рассматриваются средства моделирования ВС, позволяющие автоматически реализовать параллельные процедуры на одной ЦВМ, на макроструктурном уровне. У проектировщика сохраняется представление об одновременном функционировании отдельных устройств.

Подобная задача построения языков квазипараллельного программирования неоднократно решалась другими авторами как у нас [1,2], так и за рубежом [3,4]. Особенностью этих работ является фиксация изобразительных средств языка, в то время как заранее не ясно, какие именно структуры придется моделировать. Из перечисления компонент проекта в п.0.1 видно, что при моделировании может потребоваться описание и программ математического обеспечения, и отдельных функциональных или логических схем.

В этих условиях желательно иметь систему моделирования, располагающую удобными средствами обогащения и накопления новых изобразительных средств без вмешательства в программирующую систему. Программа построения такой растущей системы была изложена в [5]. В качестве базового языка моделирования был предложен двухуровневый язык Р-ЛЯПАС, первый уровень которого располагает средствами описания алгоритмов функционирования дискретных устройств.

0.3. Данная работа является развитием [5] и использует второй уровень языка Р-ЛЯПАС. Основное внимание уделяется построению набора операторов второго уровня, обеспечивающего квазипараллельное исполнение системы алгоритмов функционирования. Естественно, что здесь не преследуется цель исчерпать все ситуации, возникающие при структурном моделировании. Однако набор операторов должен быть достаточно полным, чтобы с его помощью можно было моделировать любые взаимодействия параллельных процессов.

Полнота не может быть обеспечена анализом конкретных моделей и в данном случае основана на рассмотрении структуры алгоритма функционирования устройства в терминах параллельного программирования [6,7]. В результате этого простого анализа

обнаруживается новый вид взаимодействий алгоритмов системы через внешние метки и предлагаются адекватные ему средства описания. Кроме того, принятая точка зрения дает возможность построить стройную систему понятий квазипараллельного программирования. Естественным образом получаются строгие определения таких объектов, как системы процессов с общей информацией и с общими метками.

Введенная терминология может быть легко перенесена на вычислительные системы. При этом следует иметь в виду, что ВС с общей памятью и общим управлением являются скорее исключением, чем правилом. Применение различных терминов зависит здесь от того, какой вид взаимодействия (через информацию или через управление) будет в системе основным и наиболее эффективным с точки зрения конструктора или пользователя ВС. Примерами ВС с общей памятью и общим управлением (в указанном смысле) могут служить проекты "НИКА-ИКС" [8] и "ИЛЛИАК-4" [9], соответственно.

Однородные вычислительные системы (ОВС) [6] относятся к классу ВС, эффективно реализующих оба вида взаимодействия между устройствами. Основные усилия конструкторов ОВС направлены, таким образом, на использование обеих возможностей. Это обстоятельство позволяет построить универсальную систему моделирования ВС, которая в то же время будет моделью ОВС с переменной структурой. Предлагаемые ниже средства моделирования включают в себя такую модель. Читатель без труда обнаружит здесь понятия и операторы, адекватные введенным в [6].

Предполагается, что читатель знаком с языком Р-ЛЯПАС [5].

§ I. Система параллельных процессов

I.I. Процессу соответствует поведение одного устройства в реальном времени. В дальнейшем предполагается, что каждый процесс P описан своим алгоритмом функционирования A, представляющим вполне упорядоченное множество операторов. Каждый оператор a_k алгоритма A требует для своего исполнения время t_k.

Операторы определены на множестве операндов X и по области значений распадаются на два класса:

а) о п е р а т о р ы п е р е с ы л к и (множество F), вырабатывающие новые значения операндов из X;

б) о п е р а т о р ы п е р е х о д о в (множество Φ), вырабатывающие логические условия (метки) из множества булевских переменных $\mathcal{M}$ и управляющие порядком реализации операторов пересылки.

Таким образом,

$$P \sim A \sim \langle F, \Phi, X, \mathcal{M}, \beta \rangle \qquad (I)$$

(где β – отношение полного порядка) или

$$P = P(X, \mathcal{M}, t).$$

Алгоритм A удовлетворяет всем известным требованиям, предъявляемым к программам. Поскольку состояние процесса явно зависит от текущего времени, то в X содержится соответствующий операнд. Описание (I) задает бесконечное множество п о т е н ц и а л ь н ы х процессов. Для того, чтобы один из них стал р е а л ь н ы м, то есть начал функционировать, необходимо задать начальные значения входных операндов множества $X^0 \subseteq X$, указать время запуска t_0 и входной оператор, помеченный меткой $\mathcal{M}_0 \in \mathcal{M}$.

В момент достижения последнего оператора алгоритма A процесс становится з а в е р ш е н н ы м (не изменяющим значений операндов из X).

1.2. Процессы называются п а р а л л е л ь н ы м и, если они реальны в одно и то же время.

Рассмотрим множество параллельных процессов:

$$P = \{P_i / i \in \mathcal{I}\}.$$

Два процесса $P_i(X_i, \mathcal{M}_i, t) \in P$ и $P_j(X_j, \mathcal{M}_j, t) \in P$ назовем и н ф о р м а ц и о н н о с в я з а н н ы м и, если $X_i \cap X_j \neq \emptyset$. Отношение связности ξ рефлексивно и симметрично:

$$(P_i, P_i) \in \xi,$$
$$(P_i, P_j) \in \xi \leftrightarrow (P_j, P_i) \in \xi.$$

Транзитивное замыкание Γ_ξ есть эквивалентность.

Множество P параллельных процессов назовем с и с т е м о й п а р а л л е л ь н ы х п р о ц е с с о в с о б щ е й и н ф о р м а ц и е й, если и только если определенная выше эквивалентность порождает единственный класс эквивалентности, то есть $\Gamma\xi = P^2$. Другими словами, граф $G \sim \langle P, \xi \rangle$

должен быть связным, а граф $G^* \sim \langle P, \Gamma_\xi \rangle$ – полным. Если указанное условие не выполняется, то каждому классу эквивалентности (или компоненте связности графа G) соответствует система, не зависимая от остальных по информации.

Аналогичным образом отношение логической связности π связывает такие два процесса P_i и P_j, множества M_i и M_j которых пересекаются. Компонентам связности графа $\langle P, \pi \rangle$ соответствуют системы параллельных процессов с общими метками. Логические условия (метки), которые могут быть выработаны вне данного алгоритма функционирования, назовем внешними; процесс, вырабатывающий внешнюю метку, – ведущим, а процесс, совершающий переход по внешним меткам, – ведомым.

Множество параллельных процессов, отвечающее хотя бы одному из указанных выше условий, назовем системой параллельных процессов.

1.3. Множество всех операндов системы можно представить в следующем виде:

$$\bigcup_{i \in J} X_i \equiv S \cup L, \quad S \cap L \equiv \phi,$$

где S – множество системных (глобальных) операндов, на которых определено несколько процессов системы;

L – множество локальных операндов, на каждом из которых определено не более одного процесса.

Рассмотрим локальные операнды i-го процесса L_i. Для организации взаимодействия процессов выделим в L_i два подмножества: сообщение и рабочее поле (РП). В сообщение войдут те локальные операнды процесса, значения которых непосредственно влияют на его поведение в системе. В них может храниться имя (или номер) процесса, его приоритеты в различных ситуациях, имя (номер, адрес) оператора алгоритма функционирования, выполняемого процессом в данный момент времени, и прочее. Остальные локальные операнды входят в РП процесса. Среди глобальных операндов системы имеет смысл выделить текущее время.

1.4. Более подробный анализ отношений информационной и логической связности приводит к понятиям информационной и логической зависимости операторов в алгоритмах функционирования системы.

Вследствие этого процесс, достигший оператора, зависимого от некоторого другого процесса, не может продолжаться далее. Он запускается после того, как будут вычислены соответствующие системные операнды или внешние метки.

Таким образом, реальный процесс может быть действующим и ждущим.

Условимся, что если внешняя метка вырабатывается для завершенного процесса, то происходит его запуск с указанного места и процесс снова становится действующим.

§ 2. Квазипараллельное исполнение системы параллельных процессов

2.1. При моделировании системы параллельных процессов на последовательной машине необходима организация их квазипараллельного исполнения. Метод такого исполнения известен для систем с дискретными событиями (систем, состояние которых можно описать значениями операндов множества S и сообщений, а изменение состояния – событие – происходит за время, пренебрежимо малое по сравнению с временем пребывания в нем).

В этом случае можно разделить моделирование функций, исполняемых процессом, и моделирование времени, которое на это необходимо. Вводятся два состояния (фазы) действующего процесса: активное и пассивное. В активной фазе вычисляется очередное состояние системы, в пассивной – моделируется задержка. В свою очередь, процесс, находящийся в активной фазе может быть текущим (исполняемым в данный момент времени в программе) или приостановленным (ожидающим своей очереди на исполнение). Приостановленный процесс уже совершил всю процедуру своей активной фазы, однако результаты ее выясняются только тогда, когда он становится текущим.

2.2. Введенную систему понятий для процесса легко продемонстрировать иерархической диаграммой (рис. 1).

2.3. Очевидны следующие условия правильного функционирования квазипараллельной модели.

УСЛОВИЕ I. Всякой активной фазе процесса предшествует его пассивная фаза.

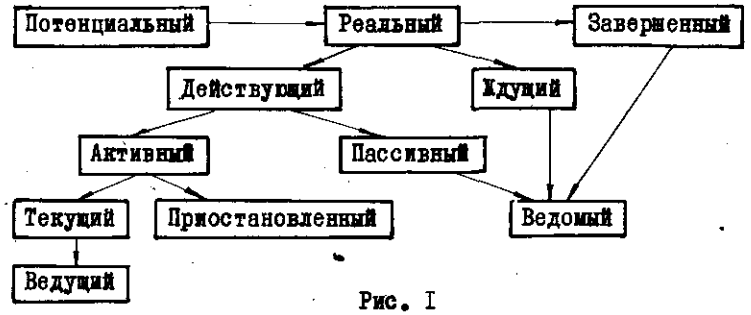

Рис. I

Выбор очередного текущего процесса (а к т и в и з а ц и я) происходит среди ждущих и приостановленных согласно их п р и о р и т е т а м (в дальнейшем этот термин имеет только такой смысл).

УСЛОВИЕ 2. Приостановленный процесс может стать текущим, если нет ни одного ждущего процесса, который может быть активизирован.

УСЛОВИЕ 3. Ведущий процесс может стать текущим только после активизации всех приостановленных процессов.

УСЛОВИЕ 4. Текущее время (см. п. I.3) увеличивается, когда не остается ни одного процесса, который может быть активизирован. Приращение времени Δt равно минимальной длительности пассивных фаз. Время ожидания всех пассивных процессов уменьшается на Δt, и те из процессов, у которых оно стало равным нулю, переходят в состояние приостановленных.

§ 3. Техника квазипараллельного программирования на Р-ЛЯПАСе

3.1. Предлагаемые ниже средства квазипараллельного программирования построены при следующих предположениях:

I) глобальные переменные доступны всем процессам;

2) сообщения всех процессов одинаковы по количеству операндов;

3) РП всех процессов имеют одинаковую мощность;

4) в каждый данный момент времени большая часть процессов завершена или пассивна.

3.2. Поставим в соответствие каждому состоянию процесса определенное положение ссылки на него в различных массивах.

С п и с о к с о о б щ е н и й (СС) содержит все сообщения реальных и завершенных процессов. В сообщение обязательно входят адрес (или команда) возврата в очередную активную фазу и приоритет процесса.

М а с с и в Р П (МРП) содержит рабочие поля тех же процессов и в том же порядке, что и сообщения в СС.

Т а б л и ц а з а д е р ж е к (ТЗ) содержит время нахождения каждого реального или завершенного процесса в очередной пассивной фазе. Эта величина находится в левой половине 32-разрядного элемента таблицы. В правых I6-ти разрядах хранится приоритет. Порядок ссылок в таблице задержек тот же, что и в списке сообщений. Время ждущих, приостановленных и завершенных процессов равно нулю.

О ч е р е д ь н а а к т и в и з а ц и ю содержит в первой половине упорядоченные по приоритету ссылки на ждущие процессы, во второй – на приостановленные. В левой половине ссылки находится приоритет процесса, в правой – его номер в таблице задержек. Ведущие процессы располагаются в конце очереди на активизацию с нулевым приоритетом. Потенциальные процессы ссылок не имеют, завершенные – имеют нулевое время ожидания, но не имеют ссылок в очереди на активизацию.

При большом числе процессов некоторые массивы (в первую очередь, МРП) могут не уместиться в оперативной памяти. Потребуем, чтобы в ней целиком поместилась очередь на активизацию (см. п. 3.I).

3.3. Работа с описанными выше массивами и организация квазипараллельного исполнения возложены на Λ – оператор П-го уровня Р-ЛЯПАСа:

м о д е л ь $\alpha\beta\gamma\delta\varepsilon\zeta\eta\vartheta\varkappa\lambda\mu\nu\xi //$,

где α – текущее время (переменная или индекс);

β – приоритет (переменная или индекс) – обязательный элемент сообщения;

γ – список переменных и индексов –произвольная часть сообщения;

δ – РП-комплекс, в котором размещаются локальные комплексы процесса, помеченные в векторе μ;

ε – максимальное число процессов (реальных и завершенных);

ς – длина очереди на активизацию;

η – длина страницы таблицы задержек;

ϑ – длина страницы списка сообщений (на входе) или длина сообщения (на выходе);

$\varkappa$ – рабочий комплекс;

λ – начало рабочего комплекса модели на внешних запоминающих устройствах;

μ – вектор локальных комплексов системы;

ν – выход на авост;

ξ – признак отладки ($\xi \neq 0$ – отладка).

Λ – оператор **модель** помещается в начале Λ-программы II-го уровня Р-ЛЯПАСа (ЛЯПАСа) и исполняет следующие функции:

I) служит признаком квазипараллельной программы;

2) распределяет память в рабочем комплексе $\varkappa$ согласно значениям δ, ε, ς, η, ϑ;

3) формирует первый текущий процесс системы;

4) исполняет все операторы квазипараллельного программирования (см.ниже);

5) отсчитывает текущее время;

6) производит выбор очередного текущего процесса согласно условиям I,II,III,IУ;

7) при $\xi \neq 0$ выдает на печать время,содержимое сообщения и рабочего поля после каждого события в системе;

8) обнаруживает аварийные ситуации и печатает соответствующие признаки.

Модель обнаруживает следующие аварийные ситуации:

I) переполнение рабочего комплекса;

2) число процессов превышает ε;

3) переполнение очереди на активизацию;

4) время задержки одного из процессов больше 2^{16};

5) зависание, то есть такое состояние системы, в котором все реальные процессы являются ждущими или завершенными;

6) нет процессов.

3.4.Управление системой процессов на верхнем уровне иерархии состояний (см.рис.I) происходит с помощью Λ – операторов: **процесс** $\alpha\beta\gamma//$, **финиш** //, **стоп** //.

Текущий процесс, встретивший в ходе реализации оператор **процесс** $\alpha\beta\gamma//$, порождает новый реальный процесс системы.Произвольная часть сообщения нового процесса получает значения операндов, перечисленных в списке α,приоритет– β, а место запуска определено меткой γ. Новый процесс получает статус приостановленного, ему отводится РП,содержимое которого не определено.

Текущий процесс выводится из модели и становится потенциальным при встрече оператора **финиш** //. При этом на те места, где находились ссылки о нем и его РП,переносятся ссылки процесса, находившегося в конце ТЗ, СС и МРП.Таким образом,эти массивы сокращаются.

Оператор **стоп** // превращает текущий реальный процесс в завершенный.При этом все ссылки на него и порядок перенумерации процессов в ТЗ, СС, МРП сохраняются.

3.5. Текущий процесс может стать ждущим,если его активная фаза заканчивается оператором **ждать** $\alpha//$.Здесь α – **условие блокировки**, то есть некоторая программа первого уровня, заканчивающаяся оператором условного перехода, например:

$$\alpha \equiv (\underline{\alpha}[i] \wedge c[0] \rightarrowtail).$$

В условие блокировки могут входить лишь глобальные операнды и элементы сообщения. Оператор,в котором встречается условие блокировки, выполняется для данного процесса только в том случае, если не осуществляется переход в конце условия. Это положение справедливо и для всех условных операторов, описанных ниже.

Процесс, остановленный оператором **ждать** $\alpha//$,находится в ждущем состоянии до тех пор, пока условие блокировки не будет удовлетворено (не произойдет переход).

3.6. Пассивная фаза процесса моделируется Λ-оператором **задержка** $\alpha//$, где α-время пребывания его в этом состоянии (время,необходимое для совершения всех действий следующей затем активной фазы).

3.7. Тот факт, что процесс является ведущим, выясняется только в текущем состоянии при реализации операторов **пуск** $\alpha\beta//$ или **вызов** $\alpha\beta//$.

Оператор п у с к $\alpha\beta//$ запускает все процессы, для которых не выполняется условие блокировки α. Место запуска указывается внешней меткой, стоящей на месте β. Условие проверяется на всех сообщениях реальных и завершенных процессов.

Если номер ведомого процесса в ТЗ заранее известен, то запуск его извне делается оператором в ы з о в $\alpha\beta//$. Здесь α – номер процесса, β – внешняя метка.

Номер процесса может быть фиксирован в глобальной переменной α в любой его активной фазе оператором н о м е р //. Собственная переменная τ на выходе последнего оператора равна номеру текущего процесса в ТЗ, СС и МРП. При использовании операторов н о м е р // и в ы з о в $\alpha\beta//$ следует иметь в виду, что фиксированная нумерация процессов может быть нарушена оператором ф и н и ш //.

3.8. Следующие два оператора облегчают моделирование распространенных режимов функционирования системы с общим управлением.

Оператор п р е р ы в а н и е $\alpha\beta\gamma//$ прерывает деятельность всех процессов, для которых не выполняется условие блокировки α. Прерванный процесс выполняет процедуру β и задерживается на время γ. После этого прерванный процесс продолжает собственные операции.

Оператор о б м е н $\alpha\beta\gamma\delta\varepsilon//$ производит пересылку информации из РП ведущего процесса непосредственно в РП ведомых. Множество ведомых задается условием блокировки α. Начальный адрес передаваемого массива в РП ведущего процесса – β, мощность – γ, начало в РП ведомого – δ, время задержки на обмен информацией – ε.

Если передаваемый или принимающий массивы являются локальными комплексами, то их положение в РП можно восстановить с помощью вспомогательного Λ-оператора н а ч а л о $\alpha\beta//$, где α – локальный комплекс; β – рабочее поле (соответствует δ оператора м о д е л ь). Собственная переменная τ на выходе оператора н а ч а л о $\alpha\beta//$ принимает запрашиваемое значение.

3.9. Оператор п р е р ы в а н и е $\alpha\beta\gamma//$ предназначен для описания простейших ситуаций такого рода. Во-первых, процедура прерывания β не может содержать операторов моделирования. Во-вторых, при использовании этого оператора задержка на время γ (в нарушение условия I) происходит после активной фазы β, и программист должен иметь это в виду. В-третьих, прерванный процесс не может быть прерван при исполнении процедуры β.

В общем случае прерывание произвольной глубины с последующим возвратом процесса в исходное состояние описывается набором операторов: п у с к $\alpha\beta//$, в ы з о в $\alpha\beta//$, в м а г а з и н $\alpha\beta\gamma//$, и з м а г а з и н а $\alpha\beta\gamma//$, в о з в р а т //.

Операторы п у с к $\alpha\beta//$ и в ы з о в $\alpha\beta//$, кроме уже описанных действий, формируют два обязательных элемента сообщения: прерванный адрес возврата ($\underline{\mathcal{U}_0}$) и ссылку в ТЗ в момент прерывания ($\underline{\mathcal{U}_1}$). Если после перехода по внешней метке ведомый процесс встретит оператор в о з в р а т //, то происходит возврат процесса в прерванное состояние.

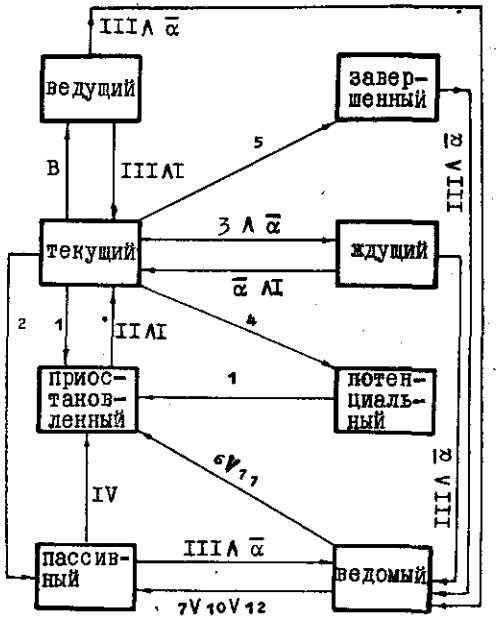

Рис. 2

При необходимости сохранить более полную информацию о прерванном процессе, строить произвольные иерархии ведомых и делать прерывание произвольной глубины требуются следующие дополнительные средства.

Оператор в м а г а з и н $\alpha\beta\gamma//$ запоминает в магазине α значения обязательных элементов $\mathcal{U}_0$ и $\mathcal{U}_1$ и значения операндов, перечисленных в списке β. В случае переполнения магазина происходит выход по метке γ.

Магазин α представляет собой локальный комплекс, размещенный в РП прерванного процесса. В начальном состоянии α_0 – – максимальная мощность магазина. В любой данный момент B_α – – текущее заполнение магазина.

Оператор и з м а г а з и н а $\alpha\beta\gamma//$ восстанавливает значения $\underline{ш}_0$, $\underline{ш}_1$ и операндов из списка β, то есть посылает в них значения последних элементов магазина α. Если магазин пуст, происходит выход по метке γ.

3.10. Наглядную картину поведения квазипараллельной программы можно получить, если поставить в соответствие каждому процессу системный автомат. Диаграмма состояний такого автомата дана на рис. 2.

На вход автомата поступают признаки условий I,II,III,IУ (см. п. 2.3), признак выполнения (α) или невыполнения ($\bar{\alpha}$) условия блокировки и признаки системных операторов, пронумерованные арабскими цифрами. Порядок нумерации следующий:

п р о ц е с с	–	1	п у с к	–	6
з а д е р ж к а	–	2	п р е р ы в а н и е	–	7
ж д а т ь	–	3	о б м е н	–	10
ф и н и ш	–	4	в ы з о в	–	11
с т о п	–	5	в о з в р а т	–	12

Буквой B обозначен признак ведущего процесса:

$$B = 6 \vee 7 \vee 10 \vee 11.$$

На рис. 2 изображены только те входные наборы, которые меняют состояние процесса. Все прочие комбинации являются запрещенными.

При интерпретации диаграммы следует иметь в виду, что во всех состояниях, кроме текущего, находится некоторое множество процессов. Условия II,III,IУ отражают ситуацию во всей системе, а условие I служит для выбора единственного текущего процесса.

§ 4. Дальнейшее наращивание возможностей моделирования

Дальнейшее развитие методики моделирования системы параллельных процессов достигается средствами второго уровня языка Р–ЛЯПАС и зависит от потребностей пользователя. Приведем простой пример.

Пусть необходимо моделировать процесс обслуживания фиксированного множества пользователей без приоритетов и отказов. В этом случае могут оказаться полезными операторы в о ч е р е д ь $\alpha//$ и и з о ч е р е д и $\alpha\beta\gamma//$, где α – комплекс, моделирующий очередь; β – адрес запуска процесса после выхода из очереди; γ – выход в случае пустой очереди. Так как при фиксированном числе процессов оператор ф и н и ш не употребляется, то соответствующие подпрограммы могут иметь следующий вид:

в о ч е р е д ь $\alpha//$,
§ 0 н о м е р $//\Rightarrow\alpha\ \alpha\Rightarrow(\alpha)$ с т о п $//$.
и з о ч е р е д и $\alpha\beta\gamma//$,
§ 0 $B_\alpha \to \gamma\ \alpha_0 \Rightarrow \alpha\ \Delta B_\alpha \to 2\ \bar{o}\,в\,о\,с$,
§ I $\Delta в \oplus B_\alpha \to 2\ \Delta с\ \alpha_с \Rightarrow \alpha_в \to 1$,
§ 2 в ы з о в $\alpha\beta//$.

Накопление подобных средств в библиотеке системы Р– ЛЯПАС практически не ограничено.

Л и т е р а т у р а

1. КАЛИНИЧЕНКО А.А. СЛЭНГ – экспериментальный язык программирования, ориентированный на описание и моделирование вычислительных машин и систем. – "Теория автоматов", Киев, 1967, вып. I.

2. ПОЛЯКОВ А.К. Система автоматизации моделирования ЦВМ.– – "Цифровая вычислительная техника и программирование".М.,"Сов. радио", 1968.

3. ДАЛ У., МЮРХАУГ Б., НЮГАРД К. СИМУЛА-67 –универсальный язык программирования. М., "Мир", 1969.

4. ШЕРР А.Анализ вычислительных систем с разделением времени. М., "Мир", 1970.

5. ВОРОБЬЕВ В.А. Р–ЛЯПАС –базовый язык моделирования цифровых устройств. – "Вычислительные системы", Новосибирск,1970, вып. 39.

6. ЕВРЕИНОВ Э.В., КОСАРЕВ Ю.Г. Однородные универсальные вычислительные системы высокой производительности. Новосибирск, "Наука"СО, 1966.

7. КОСАРЕВ Ю.Г. Распараллеливание по циклам.–"Вычислительные системы", Новосибирск, "Наука" СО, 1967, вып. 24.

8. Вычислительная система для антиракеты "НИКА-ИКС".-"Радиоэлектроника за рубежом", 1965, № 1.

9. РИЛЛЕЙ В.Б. Проект "ИЛЛИАК-4". -"Электроника", 1967 , т.40, № 10.

Поступила в ред.-изд.отд.
7. У. 1972 г.