ОДНОРОДНЫЕ ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ
Вычислительные системы
1981 год Выпуск 90

УДК 681.322.06+681.3.323

ЯДРО ОПЕРАЦИОННОЙ СИСТЕМЫ ЭМ ВЫЧИСЛИТЕЛЬНОЙ СИСТЕМЫ С ПРОГРАММИРУЕМОЙ СТРУКТУРОЙ

В.В.Корнеев, О.Г.Монахов, М.С.Тарков

Элементарная машина (ЭМ) вычислительной системы с программируемой структурой реализует заранее неизвестное число программ, использующих общие программно-аппаратные ресурсы. Эти программы обеспечивают решение задач пользователей, обслуживание внешних устройств, функционирование ЭМ как узла коммутации сети межмашинных обменов и как элемента системы самодиагностики и самовосстановления при отказах [1-3].

Согласование работы программ в ЭМ осуществляют программно-аппаратные средства ядра операционной системы ЭМ, обеспечивающие моделирование на одном процессоре ЭМ множества процессоров $\{P_i \mid i = 1, \ldots, \tilde{n}\}$. Процессоры P_i, $i = 1, \ldots, \tilde{n}$, называются виртуальными, разделяющими процессор элементарной машины. Каждый виртуальный процессор P_i выполняет одну породившую его программу G_i, $i \in \{1, \ldots, \tilde{n}\}$, $\tilde{n}$ – число программ, реализуемых ЭМ в мультипрограммном режиме.

Исполнение программы G_i виртуальным процессором P_i образует вычислительный процесс p_i, $i \in \{1, \ldots, \tilde{n}\}$. Совокупность всех процессов, реализуемых виртуальными процессорами P_i, $i = 1, \ldots, \tilde{n}$, образует систему совместно протекающих процессов.

Взаимодействие программ пользователей с ядром операционной системы ЭМ осуществляется посредством синхронизирующего примитива when $\Phi(X)$. В известном смысле синхронизирующие примитивы подобны условным операторам if $\Phi(X)$ then...else. Разница между ними состоит в том, что первые определяют, продолжать ли исполнение программы или переключать процессор на исполнение другой программы, в то время как вторые – выделяют оператор программы, исполнение которого предписано условием $\Phi(X)$. Переключение процессора с од-

ной программы на другую определяется значением предиката $\Phi(X)$. Этот предикат определен на множестве X общих (глобальных) для всех исполняемых программ объектов, называемых семафорами в отличие от индивидуальных (локальных) для каждой программы G_i, $i = 1,\ldots,\tilde{n}$, переменных.

Семафор S представляет собой тройку $\langle sem(S), \tilde{m}_S, i_S \rangle$. Здесь sem(S) - значение семафора S ; $\tilde{m}_S$ - строго упорядоченное множество, состоящее из sem(S) элементов; i_S - множество допустимых операций над семафором S. Семафоры делятся на три класса: неструктурированные, структурированные и семафоры-устройства.

Если не требуется различать и использовать элементы множества $\tilde{m}_S$, а важно лишь знать значение мощности этого множества, то само множество как таковое может отсутствовать. Семафоры этого класса называются неструктурированными. Операции над неструктурированными семафорами изменяют их значения.

В случае, когда элементы множества $\tilde{m}_S$ рассматриваются как буфера, посредством которых осуществляются коммуникации между процессами, семафор S называется структурированным. Операции над структурированным семафором S исключают/включают буфера в множество $\tilde{m}_S$ и изменяют значение семафора S .

Семафор-устройство содержит множество, элементами которого являются массивы программно доступных регистров устройства. Операции над семафорами-устройствами трактуются как определение состояния, захват, освобождение устройства.

В настоящей работе предлагается проект [4] ядра операционной системы ЭМ, который может служить базой для выбора конкретной программно-аппаратной реализации ЭМ, изготовляемой специально для компоновки мощных вычислительных систем.

I. Ядро операционной системы

I.I. Объектами входного языка ядра операционной системы ЭМ являются переменные, массивы переменных и семафоры. Переменные определяются как объекты $\alpha \in VAR$, имеющие имя name(α) и значение val(name(α)), где VAR - множество переменных ядра операционной системы ЭМ.

С целью конкретизации изложения будем далее считать, что в качестве переменных используются слова оперативной памяти. При этом именем переменной является адрес слова, а значением переменной - содержимое слова. Указанное отображение множества имен переменных

на множество переменных фиксировано аппаратурой ЭМ и не допускает изменения имени переменной и наличия у нее нескольких различных имен.

Массивы определяются как упорядоченные последовательности $\mathcal{M}$ из k переменных, $\mathcal{M} \in VAR^*(k \in \{1, 2, \ldots, n, \ldots\}$, $VAR^* = VAR^1 \cup \ldots \ldots \cup VAR^n \cup \ldots, VAR^n = \underbrace{VAR \times VAR \times \ldots \times VAR}_{n})$. Массив $\mathcal{M}$ обязательно состоит из переменных с именами a, a+1, ..., a+k-1 , где k - число переменных в массиве. Имя массива $\mathcal{M}$ совпадает с именем a первой переменной массива. Будем обозначать массив a через [a,b] , где b - длина массива. В дальнейшем нам понадобятся массивы $\gamma_i (i = 1, 2, \ldots, n, n \in \{0, 1, \ldots\})$ фиксированной длины, состоящие из четырех переменных γ_i, γ_i+1, γ_i+2, γ_i+3 . Для этих массивов будет использоваться специальное обозначение $[\gamma_i]$.

Семафоры $\beta \in SEM$, где SEM - множество семафоров ядра операционной системы ЭМ, определяются как объекты, имеющие имя name(β) значение sem(name(β)) , множество допустимых операций interpret (name(β)).

Набор операций interpret(name(β)) для неструктурированных семафоров включает, по крайней мере, следующие операции: S ← n , S ← S+ n, S ← S -n, задающие соответственно начальное значение семафора S (sem(S):= n) , увеличение на n значения семафора S (sem(S):=sem(S) + n) и уменьшение на n значения семафора S (sem(S):= sem(S) - n), $n \in \{0, 1, \ldots\}$.

I.2. Элементами множества $\tilde{m}_S$ структурированного семафора с именем S (в дальнейшем будем говорить "семафора S ") могут быть массивы и имена семафоров.

Для задания множеств выделяется общая для всех семафоров область коммуникационных буферов (см. рис.I). Каждый коммуникационный буфер α состоит из пяти расположенных подряд в памяти ЭМ слов с адресами α, α+1 , α+2, α+3, α+4 . Первое слово коммуникационного буфера α служит указателем link(α) на следующий коммуникационный буфер, принадлежащий той же цепочке, что и рассматриваемый (если указатель равен нулю, рассматриваемый буфер является последним в цепочке буферов). Следующие три слова colour(α), address(α) , buf(α) задают подсистему, адрес машины в этой подсистеме и адрес слова оперативной памяти ЭМ [3-6]. Содержимое buf(α) интерпретируется при нулевом содержимом пятого слова, length(α) = 0 , как имя семафора. При ненулевом содержимом, length(α) ≠ 0, buf(α) со-

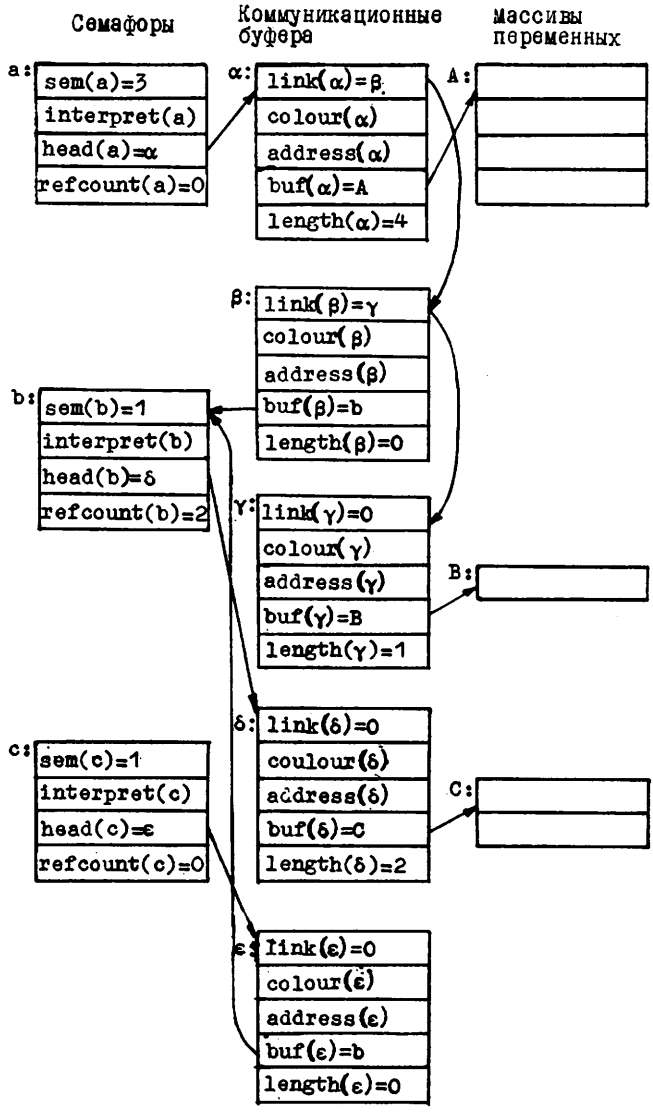

Рис.I. Организация структурированных семафоров.

держит начальный адрес массива, состоящего из length(a) слов, расположенных подряд в оперативной памяти ЭМ.

Семафор требует для своего размещения от двух до четырех слов памяти ЭМ с адресами a, a+1, a+2, a+3. Именем семафора, по определению, полагаем адрес a. Содержимое слова с адресом a задает значение семафора sem(a). Содержимое слова с адресом a+1 является указателем на программу (аппаратуру) interpret(a), интерпретирующую операции над семафором a. Содержимое слова с адресом a+2 является указателем head(a) на первый коммуникационный буфер в цепочке буферов, выделяющих массивы слов оперативной памяти ЭМ, составляющие множества $\tilde{m}_a$ (см. рис.I). Содержимое четвертого слова задает значение refcount(a). Счетчик ссылок refcount(a) семафора a равен количеству использований имени семафора a в качестве элемента множеств структурированных семафоров.

Неструктурированные семафоры используют слова с адресами a, a+1. Семафоры-устройства используют слова с адресами a, a+1, a+2. Все четыре слова используют только структурированные семафоры.

Вышеприведенная организация семафоров задает фиксированное отображение множества имен семафоров на множество семафоров аналогично случаю с переменными.

Набор допустимых операций interpret(S) над структурированными семафорами включает следующие операции: $S \leftarrow n([\gamma_1],\ldots,[\gamma_n])$; $S \leftarrow S+m,n([\gamma_1],\ldots,[\gamma_n])$; $S \leftarrow S-m,n([\gamma_1],\ldots,[\gamma_n])$; $S \leftarrow S-\langle\Gamma\rangle$, $n([\gamma_1],\ldots,[\gamma_n])$; $S \leftarrow S \backslash m,n([\gamma_1],\ldots,[\gamma_n])$; $S \leftarrow S \backslash \langle\Gamma\rangle,n([\gamma_1],\ldots$ $\ldots,[\gamma_n])$.

Операция $S \leftarrow n([\gamma_1],\ldots,[\gamma_n])$ делает sem(S):= n и создает множество $\tilde{m}_S$, элементами которого становятся массивы и имена семафоров, задаваемые массивами $[\gamma_i]$, i = 1,...,n.

При внесении в множество $\tilde{m}_S$ массива или имени семафора, задаваемых массивом $[\gamma_i]$, содержимое слов с адресами γ_i, γ_i+1, γ_i+2, γ_i+3 переносится в colour(a_i), address(a_i), buf(a_i), length(a_i) соответственно, где a_i - свободный буфер, включаемый в цепочку коммуникационных буферов, соответствующих элементам множества $\tilde{m}_S$ (см.рис.I). Отметим, что если содержимое слова с адресом γ_i+3 равно нулю, то в $\tilde{m}_S$ заносится имя семафора.

Операция $S \leftarrow S+m,n([\gamma_1],\ldots,[\gamma_n])$ увеличивает на n значение семафора S (sem(S):=sem(S)+n) и включает в множество $\tilde{m}_S$ массивы переменных или имена семафоров. При этом задаваемый $[\gamma_i]$, i= 1,...,n, массив (или имя семафора) становится (t+i)-м элементом $\tilde{m}_S$,

$$t = \begin{cases} sem(S), \text{если } sem(S) < m, \\ m, \text{ если } sem(S) \geq m. \end{cases}$$

При включении семафора $S_i = buf(a_i)$ в множество $\tilde{m}_S$ счетчик ссылок $refcount(S_i)$ увеличивается на единицу ($refcount(S_i) := refcount(S_i) + 1$). Операция $S \leftarrow S - m, n([\gamma_1], \ldots, [\gamma_n])$ уменьшает на t значение семафора S,

$$t = \begin{cases} m, \text{ если } sem(S) - m \geq n; \\ sem(S) - m, \text{ если } n > sem(S) - m > 0; \\ 0 - \text{во всех остальных случаях,} \end{cases}$$

и извлекает t элементов из множества $\tilde{m}_S$. При извлечении $(m+i)$-го элемента $(i = 1,2,\ldots,t)$ содержимое соответствующего ему коммуникационного буфера a_i, $colour(a_i)$, $address(a_i)$, $buf(a_i)$, $length(a_i)$ заносится в слова с адресами γ_i, γ_i+1, γ_i+2, γ_i+3. По выполнении указанных занесений, если $refcount(S) = 0$, коммуникационный буфер a_i освобождается и переводится в цепочку пустых коммуникационных буферов [I-5]. При $refcount(S) > 0$ буфер a_i сохраняется неизменным. Операции $S \leftarrow S - m, n([\gamma_1], \ldots, [\gamma_n])$ и $S \leftarrow S + m, n([\gamma_1], \ldots, [\gamma_n])$ обладают следующим свойством. Если $sem(S) \geq m+n$, то последовательное применение к семафору S этих операций не меняет семафора. Операция $S \leftarrow S - \langle \Gamma \rangle, n([\gamma_1], \ldots, [\gamma_n])$ извлекает $t \leq n$ элементов из числа элементов множества $\tilde{m}_S$, удовлетворяющих свойству Γ.

Исполнение операций $S \leftarrow S \sqcap m, n([\gamma_1], \ldots, [\gamma_n])$ и $S \leftarrow S \sqcap \langle \Gamma \rangle, n([\gamma_1], \ldots, [\gamma_n])$ приводит к появлению в $[\gamma_i]$, $i = 1, \ldots, n$, того же содержимого, что и при исполнении операций $S \leftarrow S - m, n([\gamma_1], \ldots, [\gamma_n])$ и $S \leftarrow S - \langle \Gamma \rangle, n([\gamma_1], \ldots, [\gamma_n])$, но, в отличие от последних, указанные операции не меняют значений семафоров и не извлекают элементов из множеств $\tilde{m}_S$.

Структурированные семафоры и операции над ними обеспечивают задание и преобразование произвольных списковых структур. Каждый семафор S задает список, элементами которого являются элементы множества $\tilde{m}_S$. Семафоры S_j, $j = 1, \ldots, e$, имена которых указаны в $\tilde{m}_S$, задают подсписки S_j списка S. Так, списки (A, (C), B) и ((C)) представлены на рис. I семафорами с именами a и c соответственно. Элементы списков A, B, C являются массивами, состоящими из четырех, одного и двух слов оперативной памяти ЭМ.

I.3. Элементом одноэлементного множества семафора-устройства processor (процессора ЭМ) является совокупность регистров, храня-

щих информационный вектор текущего процесса (счетчик команд и другие "спасаемые" при прерываниях регистры). Над семафором processor определены следующие операции. Операция processor ← 1([γ]) создает из содержимого слов с адресами a, a+1, ... , где a - содержимое слова с адресом γ+2, информационный вектор процесса и затем загружает созданный вектор в процессор ЭМ. Операция processor ← ← processor + 1([γ]) загружает в процессор информационный вектор процесса, размещенный в массиве с начальным адресом, равным содержимому γ+2. Соответственно операция processor ← processor-1([γ]) создает в массиве с начальным адресом, равным содержимому γ + 2, копию информационного вектора, загруженного в процессор в момент ее исполнения.

К семафорам-устройствам относится также семафор memory , посредством которого осуществляется управление памятью ЭМ. Операция memory ← n([γ_1,δ_1],...,[γ_n,δ_n]) создает список из n свободных массивов слов оперативной памяти ЭМ. Каждый такой массив $\mathcal{M}_i$ начинается с адреса (γ_i +2) и имеет длину (δ_i); (γ_i + 2),(δ_i) - содержимое слов с адресами γ_i + 2, δ_i соответственно.

Операция memory ← memory + n([γ_1,δ_1],..., [γ_n, δ_n]) добавляет в список свободных массивов слов оперативной памяти ЭМ n массивов $\mathcal{M}_i$, заданных посредством [γ_i,δ_i], i = 1,2,...,n .Если при включении в список массива $\mathcal{M}_i$, i∈{1,...,n } , из него и уже имеющихся в списке массивов образуется непрерывный массив $\mathcal{M}_0$ слов памяти, то все массивы, вошедшие в $\mathcal{M}_0$, удаляются из списка свободных массивов, а вместо них заносится один массив $\mathcal{M}_0$. Операция memory ← memory-n([γ_1,δ_1],...,[γ_n,δ_n]) формирует из списка свободных массивов слов n массивов $\mathcal{M}_i$, i = 1,...,n, длины которых равны (δ_i) , а начальные адреса равны (γ_i+ 2), i = 1,...,n .

На базе операций над семафором memory вводятся операции создания семафора create semaphore([γ,δ]) и уничтожения семафора destroy semaphore([γ,δ]) . Первая из указанных операций извлекает из списка свободных массивов слов оперативной памяти ЭМ с адресом (γ + 1) подсистемы (γ) массив длиной (δ) = 2 либо (δ) = 4 слов. При этом в слово с адресом γ+2 записывается начальный адрес a извлеченного массива. В слово с адресом a+1 записывается значение указателя интерпретирующих программ interpret в соответствии с типом создаваемого семафора. Остальные слова извлеченного массива обнуляются. На этом формирование семафора a = (γ+2) заканчивается.

Операция destroy semaphore([γ,δ]) уничтожает семафор a = = (γ+2) в машине подсистемы (γ), имеющей адрес (γ + 1), если refcount(a) = 0. Уничтожение семафора a состоит из следующих операций:

1. Из множества $\tilde{m}_a$ извлекаются все элементы.

2. Соответствующие коммуникационные буфера освобождаются и переводятся в цепочку пустых коммуникационных буферов.

3. Если извлеченный элемент b сам является семафором, то счетчик ссылок refcount(b) семафора b уменьшается на единицу. Если после этого refcount(b) = 0, то семафор b уничтожается посредством операции destroy semaphore([ε,ζ]), где(ζ)- длина массива, отведенного под семафор b = (ε+2).

4. Массив a, содержащий переменные sem(a), interpret(a), head(a), refcount(a), помещается в список свободных массивов memory.

Если счетчик ссылок refcount(a) ≥ 1, то операция destroy semaphore([γ,δ]) не изменяет семафора.

1.4. Процесс p_i, $i = 1,\ldots,\tilde{\mathcal{R}}$, $\tilde{\mathcal{R}} \in \{1,2,\ldots\}$, системы совместно протекающих процессов задается совокупностью действий:

$$M_{ij}:\ \underline{\text{when}}\ L_i = M_{ij} \wedge \bigwedge_{k\in A_{ij}} S_k > 0 \wedge \bigwedge_{k\in B_{ij}} S_k = 0 \wedge \bigwedge_{k\in C_{ij}} S_k < 0\ \underline{\text{do}}\ \tau_1 \leftarrow$$

$$\leftarrow \varphi_{ij1}(\nu_i);\ldots;\ \tau_h \leftarrow \varphi_{ijh}(\nu_i);\ \text{state} \leftarrow 0;$$

$$\rho_1 \leftarrow \psi_{ij1}(\mathcal{R}_i);\ldots;\ \rho_g \leftarrow \psi_{ijg}(\mathcal{R}_i)\ \underline{\text{od}},$$

где $j \in \{1,\ldots,\mathcal{L}_i\}$, $\mathcal{L}_i$ - количество действий, задающих процесс p_i, $\nu_i = \{L_i \cup \text{SEM} \cup \text{VAR}_i\}$, $\mathcal{R}_i = \{L_i \cup \text{VAR}_i\}$, L_i - счетчик команд процесса p_i, VAR_i - множество переменных процесса p_i, SEM - множество семафоров ядра операционной системы ЭМ. Операторы $\tau_f \leftarrow \varphi_{ijf}(\nu_i)$, $f = 1,\ldots,h$, выполняемые в j-м критическом интервале процесса ρ_i [7], вычисляют новые значения $\tau_f \in \nu_i$ семафоров, переменных и счетчика команд L_i. Операторы $\rho_r \leftarrow \psi_{ijr}(\mathcal{R}_i)$, $r = 1,\ldots,g$, вычисляющие новые значения $\rho_r \in \mathcal{R}_i$ переменных и счетчика команд L_i, выполняются вне критического интервала.

Каждый процесс p_i, $i = 1,\ldots,\tilde{\mathcal{R}}$, представлен в ядре операционной системы ЭМ информационным вектором PSW_i, что позволяет говорить о виртуальной ЭМ, реализующей процесс p_i. Информационный вектор процесса PSW_i, $i \in \{1,\ldots,\tilde{\mathcal{R}}\}$, имеет следующие состав-

ляющие: PSW_i [имя процесса] - код, задающий имя процесса; PSW_i [счетчик команд] - значение счетчика команд L_i, равное метке предназначенного к исполнению оператора процесса p_i; PSW_i [указатель на область определения операторов] - содержимое "спасаемых" при прерываниях регистров процессора ЭМ; PSW_i[positive], PSW_i[zero], PSW_i [negative] - множества A,B,C имен семафоров соответственно, $A \cap B = \emptyset$, $A \cap C = \emptyset$, $B \cap C = \emptyset$. Информационный вектор процесса p_i, $i \in \{1,2,\dots,\mathcal{K}\}$, либо загружен в процессор, либо находится в очереди готовых процессов (структурированный семафор ready), либо помещен в очередь ждущих процессов (структурированный семафор wait). Соответственно процесс p_i при этом находится в текущем, готовом и ждущем состояниях. Порождение процесса сводится либо к загрузке его информационного вектора в процессор ЭМ, либо к вне - сению его информационного вектора в одну из очередей ready или wait. Уничтожение процесса p_i, $i \in \{1,\dots,\mathcal{K}\}$, выполняется как удаление информационного вектора PSW_i из процессора (если про - цесс p_i текущий), из множества $\tilde{m}_{ready}$ (если процесс p_i готовый) или из множества $\tilde{m}_{wait}$ (если p_i ждущий).

Действие с меткой M_{ij} интерпретируется процессором ЭМ как совокупность g+h+2 операторов: M_{ij}: M_{ij0}: when... do; M_{ij1}: $\tau_1 \leftarrow \varphi_{ij1}(\nu_1)$;...; M_{ijh}: $\tau_h \leftarrow \varphi_{ijh}(\nu_i)$; $M_{ij(h+1)}$: state $\leftarrow$ 0; $M_{ij(h+2)}$: $\rho_1 \leftarrow \phi_{ij1}(\mathcal{R}_i)$; ... ; $M_{ij(h+g+1)}$: $\rho_g \leftarrow \psi_{ijg}(\mathcal{R}_i)$, где M_{ijq} - метка, сопоставляемая оператору при размещении его в памяти ЭМ, q = 0,..., h+g+1. Процессор ЭМ в ходе исполнения текущего опера - тора определяет новое значение счетчика команд, что обеспечивает переход процессора на интерпретацию следующего оператора с меткой, равной содержимому счетчика команд. Получение счетчиком команд процессора значения M_{ij}, $i \in \{1,\dots,\mathcal{K}\}$, $j \in \{1,\dots,\mathcal{L}_i\}$, вызывает исполнение оператора processor $\leftarrow$ 1([γ]), где (γ+2) - начальный адрес массива, содержащего PSW_i, посредством которого в процес - сор загружается новый информационный вектор, формируемый из ста - рого путем следующих преобразований: PSW_i [positive]:= A_{ij}; PSW_i[zero]:= B_{ij}; PSW_i[negative]:= C_{ij}; PSW_i[счетчик команд] := M_{ij1}, остальные составляющие PSW_i остаются без изменения. Загрузка в процессор информационного вектора PSW_i, $i \in \{1,\dots,\mathcal{K}\}$, про-

цесса p_i , вызывает вычисление значения предиката*)

$$\bigwedge_{k\in A_{ij}} sem(S_k)>0 \wedge \bigwedge_{k\in B_{ij}} sem(S_k)=0 \wedge \bigwedge_{k\in C_{ij}} sem(S_k)<0.$$

Если значение предиката равно false , загруженный в процессор информационный вектор посылается в очередь ждущих процессов, из очереди готовых выбирается информационный вектор, который загружается в процессор. Указанная последовательность действий реализуется посредством следующих операций над семафорами processor, ready и wait: processor←processor-1([γ]); wait←wait+sem(wait), 1([γ]); ready←ready-0,1([γ]); processor←processor+1([γ]) .

Если значение предиката равно true, составляющие информационного вектора PSW_i[positive], PSW_i[zero], PSW_i[negative] , получают значение ∅ , что соответствует заданию тождественно истинного предиката. По выполнении указанных преобразований информационного вектора PSW_i процесс p_i входит в свой критический интервал, что отмечается установкой значения семафора state , равного единице (sem(state):= 1) . Первым оператором, исполняемым в критическом интервале, является оператор с меткой M_{ij1} . Последовательность операторов процесса p_i , выполняемых в критическом интервале, не допускает прерывания. Иными словами, при sem(state)=1 запрещается переключение процессора на исполнение прерывающего процесса, обслуживающего поступивший от внешнего устройства или межмашинной связи запрос на прерывание. При этом указанный запрос вызывает присчет единицы к счетчику необслуженных прерываний interruptcount и внесение в начало очереди готовых процессов информационного вектора прерывающего процесса.

Запрос на прерывание, поступивший вне критического интервала (при sem(state) = 0), убирает из процессора информационный вектор текущего процесса, помещает указанный вектор в начало очереди готовых процессов и загружает в процессор информационный вектор прерывающего процесса.

Процесс выходит из критического интервала после исполнения операции state ← 0. Подпрограмма, интерпретирующая операцию state ← 0 , последовательно с начала очереди вычисляет предикаты

*) Указанный предикат получает значение true , если значения всех семафоров, имена которых принадлежат множеству A , больше нуля, значения всех семафоров, имена которых принадлежат множеству B , равны нулю и значения всех семафоров, имена которых принадлежат множеству C , меньше нуля.

всех информационных векторов, находящихся в очереди ждущих процессов (множестве $\tilde{m}_{wait}$). Если предикат информационного вектора PSW_j, $j \in \{1,\dots,sem(wait)\}$ равен true, то PSW_j[positive]:= $\emptyset$, PSW_j[zero]:= $\emptyset$, PSW_j[negative]:= $\emptyset$ и информационный вектор PSW_j помещается в конец очереди готовых процессов. Если предикат информационного вектора равен false, указанный информационный вектор остается в очереди ждущих процессов. После просмотра всей очереди ждущих процессов анализируется значение счетчика необслуженных прерываний interruptcount. Если значение interruptcount=0, продолжается текущий процесс. При interruptcount $\neq$ 0 информационный вектор текущего процесса помещается в качестве (interruptcount) + + I-го элемента в очередь готовых процессов (семафор ready), а из начала очереди готовых процессов выбирается информационный вектор, соответствующий прерывающему процессу. Указанный вектор загружается в процессор, и значение счетчика необслуженных прерываний уменьшается на единицу.

Таким образом, ядро операционной системы ЭМ обязательно содержит семафоры processor, state, ready, wait. Два последних являются очередями готовых и ждущих процессов [I-5]. Представление указанных очередей семафорами сводит порождение и уничтожение процессов к операциям над семафорами.

2. Организация линий межмашинного обмена

2.I. Линия межмашинного обмена (i_p, j_q) обеспечивает передачу информации с выходного полюса p $ЭМ_i$ на входной полюс q $ЭМ_j$, $p,q \in \{1,\dots, v\}$, где v - степени вершин графа межмашинных связей [I-3]. В передающей $ЭМ_i$ имеются структурированные семафоры $EMPTY^i$ и $ВЫХ^i_p$, играющие роль цепочек пустых и полных буферов, традиционно используемых при работе с внешними устройствами. Принимающая $ЭМ_j$ содержит структурированные семафоры $EMPTY^j$ и $ВХ^j_q$, являющиеся цепочками пустых и полных буферов на приемном конце линии. Передача данных по линии (i_p, j_q) инициируется передающей $ЭМ_i$, в которой исполняется действие (путем передачи управления на метку M_1)

M_1: when L=M_1 $\wedge$ $EMPTY^i$>0 do $EMPTY^i$ $\leftarrow$ $EMPTY^i$-0,1([a]); L$\leftarrow$ M_2 od.

Указанное действие обеспечивает изъятие первого пустого буфера из $\tilde{m}_{EMPTY^i}$. Пусть этим буфером будет γ_0 (см.рис.2). В буфер данных γ_0, γ_0 равно содержимому a+2, заносится блок данных, который необходимо передать в принимающую $ЭМ_j$ (действие с меткой M_2)

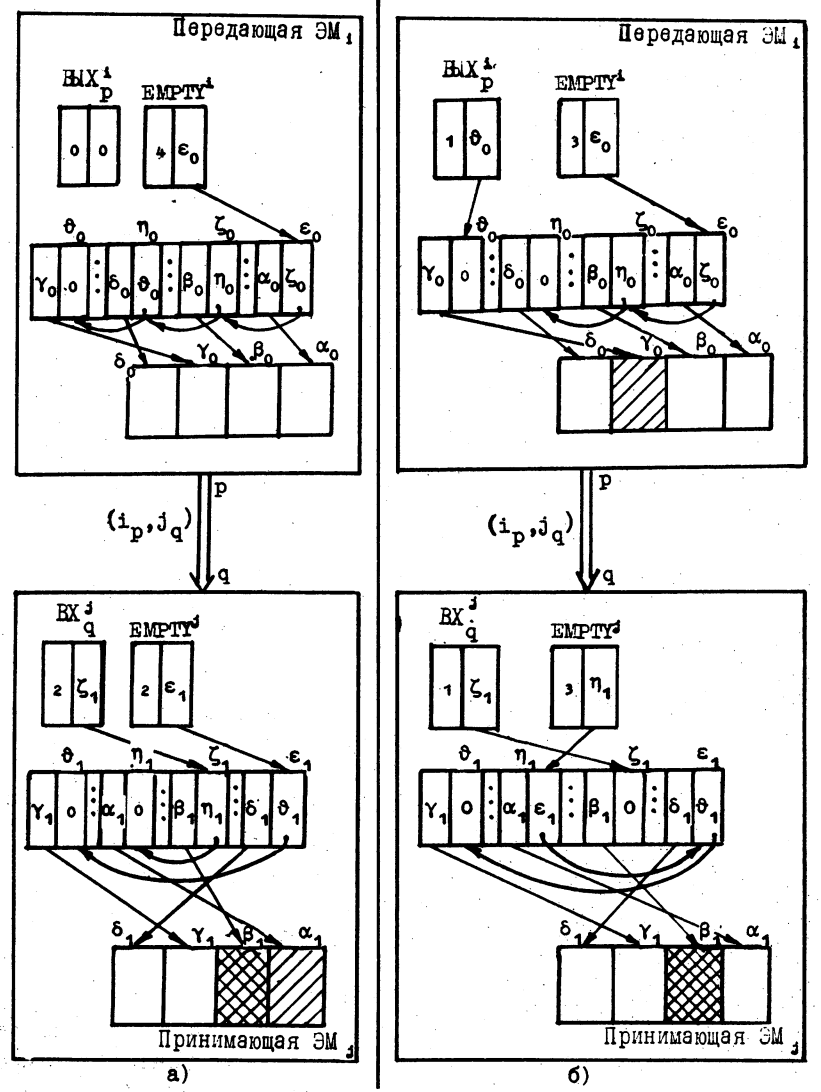

Рис.2. Состояние коммуникационных буферов, связанных линией (i_p, i_q): а) до начала передачи по линии; б) по окончании передачи.

M_2: when L = M_2 do заполнение γ_0; L ← M_3 od.

По заполнении буфера γ_0 выполняется действие

M_3: when L=M_3 do $ВЫХ_p^i \leftarrow ВЫХ_p^i + sem(ВЫХ_p^i), 1([a])$; L ← M_1 od,

включающее γ_0 в $\widetilde{m}_{ВЫХ_p^i}$ в качестве последнего элемента (на рис.2,а изображена ситуация в передающей и принимающих ЭМ по выполнении указанного включения). С целью упрощения изображения структурированных семафоров $EMPTY^i$, $ВЫХ_p^i$, $EMPTY^j$, $ВХ_q^j$ на рис.2 показаны не все атрибуты, а лишь head и sem. Это же касается изображения коммуникационных буферов, для которых показаны только link и buf.

Механизм линии межмашинного обмена обеспечивает реализацию действия when $ВЫХ_p^i > 0 \wedge EMPTY^j > 0$ do $ВЫХ_p^i \leftarrow ВЫХ_p^i - 0,1([a])$; $EMPTY^j \leftarrow EMPTY^j - 0,1([c])$; [c] ⇐[a]; comment пересылка данных по линии из области памяти buf(a) передающей $ЭМ_i$ в область памяти buf(c) принимающей $ЭМ_j$ (в случае рис.2 из области памяти γ_0 в область памяти α_1); $EMPTY^i \leftarrow EMPTY^i + sem(EMPTY^i), 1([a])$: $ВХ_q^j \leftarrow ВХ_p^j + sem(ВХ_q^j), 1([c])$ od.

Таким образом, механизм линии запускается при наличии заполненного буфера в передающей ЭМ и пустого в принимающей. По окончании обмена опустошенный буфер включается в цепочку пустых буферов передающей ЭМ, а заполненный буфер - в цепочку полных буферов принимающей ЭМ. Следует отметить, что конкретная реализация линии межмашинного обмена определяет метод организации обмена: блоками фиксированной или блоками переменной длины. В последнем случае длины буферов α_0, β_0, γ_0, δ_0, α_1, β_1, γ_1, δ_1 устанавливаются динамически.

2.2. Программно-аппаратные средства линии межмашинного обмена делают ее либо специализированной (принадлежащей к одному из трех типов линий: обмена данными, обмена командами, расширения регистров и передачи их значений [I-4]), либо разделяемой во времени. В последнем случае оборудование одной и той же физической линии используется несколькими виртуальными линиями. Введение виртуальных линий (i_{pe}, j_{qe}) основано на том, что интерпретация данных, пришедших в ЭМ по линии (i_p, j_q), целиком определяется процессом acceptdata q, изымающим данные из семафора $ВХ_q^j$ и помещающим их в семафоры V_{qe} на приемных концах виртуальных линий, e = = 0,...,E, E+1 - количество виртуальных линий, разделяющих физическую линию.

Процесс acceptdata q задается следующей программой:

process acceptdata q: MA_q: when $L_q = MA_q \wedge BX_q^j > 0$ do

$BX_q^j \leftarrow BX_q^j - 0{,}1([\alpha])$; go to $P([\alpha])$;

comment значением $P([\alpha])$ является метка MC_e, e = 0,...,E+1, вычисляемая на основании служебной информации, содержащейся в блоке данных buf(α), $EMPTY^j$ - общая для всех входных полюсов цепочка пустых буферов. Если служебная информация, содержащаяся в блоке данных buf(α), не позволяет идентифицировать последний, значение $P([\alpha]) = MC_{E+1}$:

MC_0: $V_{q0} \leftarrow V_{q0} + sem(V_{q0}), 1([\alpha]); L_q \leftarrow MA_q;$

. . . . . .

MC_E: $V_{qE} \leftarrow V_{qE} + sem(V_{qE}), 1([\alpha]); L_q \leftarrow MA_q;$

MC_{E+1}: $EMPTY^j \leftarrow EMPTY^j + sem(EMPTY^j), 1([\alpha]); L_q \leftarrow MA_q$ od .

Дальнейшее использование блока данных, занесенного в семафор V_{qe}, $q \in \{1,...,v\}$, $e \in \{0,..., E\}$, определяется процессом, манипулирующим с этим семафором. Так, например, указанный процесс может рассматривать блок данных как информационный вектор и тело процесса. В этом случае имеет место виртуальная линия обмена командами.

Выдача из $ЭМ_i$ блока данных в виртуальную линию (i_{pe}, j_{qe}), $e \in \{0,..., E\}$, предваряется приформировыванием к этому блоку служебной информации. Получившийся в результате блок данных [γ] заносится в семафор $ВЫХ_p^i$ посредством передачи управления на действие с меткой MT_p

MT_p: when $L_T = MT_p \wedge EMPTY^i > 0$ do $EMPTY^i \leftarrow EMPTY^i - 0{,}1([\delta])$;

$[\delta] \Leftarrow [\gamma]$;

comment пересылка блока данных из области памяти buf(γ) в область памяти buf(δ); $ВЫХ_p^i \leftarrow ВЫХ_p^i + sem(ВЫХ_p^i), 1([\delta])$; $L_T \leftarrow MT_p + 1$; comment переход на следующее действие; od.

3. Организация параллельных вычислений

3.1. При исполнении $\tilde{n}$ совместно протекающих процессов (каждый процесс отнесен к счетчику команд L_i, $i=1,\ldots,\tilde{n}$) в вычислительной системе с программируемой структурой создаются и функционируют $\tilde{n}$ виртуальных ЭМ. Совокупность совместно протекающих процессов исполняется параллельно, если соответствующие процессам виртуальные ЭМ порождены в разных физических ЭМ. Будем также говорить о конкурентном исполнении процессов, если хотя бы две соответствующие им виртуальные ЭМ разделяют одну физическую ЭМ. Таким образом, организация параллельных вычислений (параллельного исполнения процессов системы совместно протекающих процессов) предполагает обязательное использование линий межмашинного обмена*).

Обеспечение возможности решения задачи, требующей $\tilde{n}$ машин, на любых $\tilde{n}$ связанных между собой машинах системы базируется на специальном способе программирования задач. При программировании задачи будем полагать, что $\tilde{n}$ необходимых для ее параллельного решения машин образуют адресную подсистему [6,8], машины которой имеют адреса A_j, $j=0,\ldots,\tilde{n}-1$. Каждая ЭМ$_j$ (ЭМ с адресом A_j) содержит семафоры $ВХ_0^j$, $ВЫХ_0^j$, $EMPTY^j$, $j\in\{0,1,\ldots,\tilde{n}-1\}$. Передача блока данных T из ЭМ$_j$ в ЭМ$_{j_1},\ldots,$ЭМ$_{j_r}$, $r\in\{1,\ldots,\tilde{n}\}$, $j\in\{0,1,\ldots,\tilde{n}-1\}$, $i=1,\ldots,r$, $ji\in\{0,1,\ldots,\tilde{n}-1\}$, осуществляется посредством засылки коммутационного сообщения [1,3-6] $H\{A_j,B_1,\ldots,B_t;T\}$ в семафор $ВЫХ_0^j$. Указанная засылка инициируется передачей управления на метку M_k

M_k: when $L = M_k \wedge EMPTY^j > 0$ do

$EMPTY^j \leftarrow EMPTY^j - 0{,}1([\alpha])$; $[\alpha] \Leftarrow H\{A_j,B_1,\ldots,B_t;T\}$;

comment засылка в массив buf(α) коммутационного сообщения; $ВЫХ_0^j \leftarrow ВЫХ_0^j + sem(ВЫХ_0^j), 1([\alpha])$; $L \leftarrow M_k+1$; comment переход на следующее действие, отнесенное к счетчику L; od.

Коммутационное сообщение $H\{A_j,B_1,\ldots,B_t;T\}$ имеет структуру: H - признак коммутационного сообщения; A_j - адрес передающей ЭМ; B_i, $i=1,\ldots,t$, $t\le r$, являются адресами принимающих ЭМ$_{j_1},\ldots,$ЭМ$_{j_r}$ или именами подмножеств адресов принимающих ЭМ. В частности, для

*) В дальнейшем мы не будем различать виртуальные и физические линии, полагая, что если линия является виртуальной, то вместо семафоров ВХ и ВЫХ должны стоять семафоры, соответствующие виртуальной линии.

обозначения всего подмножества машин адресной подсистемы будем использовать символ ВСЕ. Передача блока данных T из машины $ЭМ_j$, $j \in \{0,1,\ldots,\tilde{n}-1\}$, в машины подмножества $ЭМ_{j_1},\ldots,ЭМ_{j_r}$ подсистемы Q происходит путем трансляции коммутационного сообщения $H\{A_j, B_1,\ldots,B_t; T\}$ через последовательность машин подсистемы Q, расположенных между передающей $ЭМ_j$ и принимающими $ЭМ_{j_1},\ldots,ЭМ_{j_r}$. Реализацию передачи данных посредством путевой процедуры с позиции программиста можно мыслить как исполнение следующего действия, определенного над множеством семафоров передающей и принимающих машин (МПП):

$$\underline{\text{when}}\ L_{ПП} = МПП \wedge ВЫХ_0^j > 0\ \ \underline{\text{do}}\ \ ВЫХ_0^j \leftarrow ВЫХ_0^j - 0{,}1([\gamma]);$$

comment содержимым массива buf(γ) становится коммутационное сообщение $H\{A_j, B_1,\ldots,B_t; T\}$, в котором $B_1,\ldots,B_t$ задают адреса принимающих $ЭМ_{j_1},\ldots,ЭМ_{j_r}$; $[\gamma] \Leftarrow \mathcal{P}[\gamma]$; comment содержимым массива buf(γ) становится пара A_j, T, состоящая из адреса передающей $ЭМ_j$ и передаваемых данных T; $EMPTY^{j_1} \leftarrow EMPTY^{j_1} - 0{,}1([\varepsilon]);\ [\varepsilon] \Leftarrow [\gamma];$

$BX_0^{j_1} + sem(BX_0^{j_1}), 1([\varepsilon]); EMPTY^{j_2} \leftarrow EMPTY^{j_2} - 0{,}1([\varepsilon]);\ [\varepsilon] \Leftarrow [\gamma]\ ;$

$BX_0^{j_2} \leftarrow BX_0^{j_2} + sem(BX_0^{j_2}), 1([\varepsilon]);\ldots\ ; EMPTY^{j_r} \leftarrow EMPTY^{j_r} - 0{,}1([\varepsilon]);$

$[\varepsilon] \Leftarrow [\gamma]; BX_0^{j_r} \leftarrow BX_0^{j_r} + sem(BX_0^{j_r}),\ 1([\varepsilon]);\ EMPTY^j \leftarrow EMPTY^j +$

$+ sem(EMPTY^j), 1([\gamma]);\ L_{ПП} \leftarrow МПП$ od.

Таким образом, в результате исполнения путевой процедуры в множествах структурированных семафоров $ВХ_0^{j_i}$ $(i = 1,\ldots,r)$ каждой из принимающих машин $ЭМ_{j_1}, ЭМ_{j_2},\ldots,ЭМ_{j_r}$ появляется элемент A_j, T, состоящий из адреса A_j передающей $ЭМ_j$ и передаваемых данных T.

Выбор в каждой машине $ЭМ_j$ блоков данных, пришедших в нее из других $ЭМ_k$, $j = 0,\ldots,\tilde{n}-1$, $k \in \{0,1,\ldots,\tilde{n}-1\}$, производится действием

$$M_1:\ \underline{\text{when}}\ L_{rec\ j} = M_1 \wedge\ BX_0^j > 0\ \underline{\text{do}}\ BX_0^j \leftarrow BX_0^j - 0{,}1([\alpha]);$$

comment содержимым массива buf(α) становится $A_i, T; R([\alpha])$; comment использование поступившего блока данных процедурой R ; $EMPTY^j \leftarrow$ $\leftarrow EMPTY^j + sem(EMPTY^j), 1([\alpha]);\ L_{rec\ j} \leftarrow M$ od.

3.2. Рассмотрим программирование для вычислительных систем с программируемой структурой на примере решения методом итерации системы линейных уравнений n-го порядка. Систему уравнений представляем в виде:

$$x_1^{(k)} = d_1 + c_{11}x_1^{(k-1)} + \ldots + c_{1n}x_n^{(k-1)},$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$x_n^{(k)} = d_n + c_{n1}x_1^{(k-1)} + \ldots + c_{nn}x_n^{(k-1)},$$

где $x_i^{(k-1)}$, $x_i^{(k)}$ $(i = 1,\ldots,n)$ соответственно значения x_i при (k-1)-й и k-й итерациях. Задача считается решенной, если выполнено условие $\max\limits_{1\le i\le n} |x_i^{(k)} - x_i^{(k-1)}| < \delta$.

Пусть для решения задачи в системе выделена [8] адресная подсистема из n+1-й элементарной машины, каждой из которых поставлен в соответствие адрес A_i, $i = 0,\ldots,n$. Подсчет $x_i^{(k)}$, $i = 1,\ldots,n$, производится в $ЭМ_i$ с адресом A_i. Проверка условия осуществляется в $ЭМ_0$ с адресом A_0.

В $ЭМ_i$, $i = 1,\ldots,n$, помещаем процессы $result_{i0}$, $result_{i1}$ и $iteration_i$.

```
process result_i0;
M1: when L = M1 do P_i0 ← 0; x_i0 := x_i^(0) ; L ← M2 od
M2: when L = M2 ∧ P_i0 = 0 ∧ EMPTY^i > 0 do
    EMPTY^i ← EMPTY^i - 0,1([α]); [α] ∈ H{A_i, ВСЕ; 1,i,x_i0};
    x_i0 := d_i; P_i0 ← n; ВЫХ_0^i ← ВЫХ_0^i + sem( ВЫХ_0^i), 1([α]);
    L ← M2 od
process result_i1;
M1: when L = M1 do P_i1 ← n; x_i1 := d_i; L ← M2 od
M2: when L = M2 ∧ P_i1 = 0 ∧ EMPTY^i > 0 do
    EMPTY^i ← EMPTY^i - 0,1([α]); [α] ∈ H{A_i, ВСЕ; 0,i,x_i1};
    x_i1 := d_i; P_i1 ← n; ВЫХ_0^i ← ВЫХ_0^i + sem( ВЫХ_0^i), 1([α]);
    L ← M2 od
process iteration_i;
M1: when L = M1 ∧ ВХ_0^i > 0 do ВХ_0^i ← ВХ_0^i - 0,1([γ]); a := γ[номер]-1;
    K_i ← K_i ⊏ a,1([b]); if γ[индикатор ]=1
    then begin x_i1 := x_i1 + [b] × γ[значение ];
```

$P_{i1} \leftarrow P_{i1} - 1$ end else begin $x_{i0} := x_{i0} +$

$+ [b] \times \gamma$[значение]; $P_{i0} \leftarrow P_{i0} - 1$ end

$EMPTY^i \leftarrow EMPTY^i + sem(EMPTY^i), 1([\gamma])$;

$L \leftarrow M_1$ od

comment принимаемое в массив buf(γ) сообщение имеет вид A_j, f, j, x_j, $j \in \{1,\ldots,n\}$. Обозначим через γ[индикатор], γ[номер], γ [значение] соответственно составляющие f, j, x_j.

Начальные установки семафоров в $ЭМ_i$: $sem(P_{i0})=0$; $sem(P_{i1})=n$; $sem(EMPTY^i) = m$, $\tilde{m}_{EMPTY^i} = \{\nu_1,\ldots,\nu_m\}$, ν_g - буфера под размещение коммутационных сообщений, $g=1,\ldots,m$; $sem(ВЫХ_0^i) = 0$, $\tilde{m}_{ВЫХ_0^i} = \emptyset$, $sem(ВХ_0^i) = 0$, $\tilde{m}_{ВХ_0^i} = \emptyset$; $sem(K_i) = n$, $\tilde{m}_{K_i} = \{c_{i1}, c_{i2}, \ldots, c_{in}\}$ - строка коэффициентов. Кроме того, $x_{i1} = d_i$, $x_{i0} = x_i^0$, т.е. начальному приближению, $i \in \{1,\ldots, n\}$.

Функционирование $ЭМ_i$, $i = 1,\ldots, n$, начинается с процесса $result_{i0}$, который выдает всем ЭМ адресной подсистемы сформированное в $ЭМ_i$ коммутационное сообщение, содержащее x_i^0. После того как $ЭМ_i$ выдала сообщение, она готова к приему значений x_j^0, $j = 1, 2,\ldots, n$. Обработка принятых значений осуществляется процессом $iteration_i$. Семафор $ВХ_0^i$ в качестве элементов получает $\{A_j, f, j, x_j\}$, $f \in \{0,1\}$, $j = 1,\ldots,n$. Из полученной четверки выделяется индекс j, используемый для выбора из семафора K_i (хранящего строку коэффициентов $c_{i1}, c_{i2},\ldots, c_{in}$) коэффициента c_{ij}. Далее выполняется $x_{if} := x_{if} + c_{ij} x_j$. По окончании указанного присваивания обработанный элемент семафора $ВХ_0^i$ помещается в семафор $EMPTY^i$. Тем самым обеспечивается прием следующих значений x_j, $j = 1, 2,\ldots, n$.

В $ЭМ_i$, проверяющую выполнение условия окончания итераций, помещается процесс delta.

process delta;

M_1: when $L=M_1 \wedge BX_0^0 > 0 \wedge R=0$ do $BX_0^0 \leftarrow BX_0^0 - 0,1([\gamma])$;

if γ[индикатор]=1 then begin $j_1 := j_1 + 1$;

h:= γ[индекс]; if $|Y[h] - \gamma$[значение]$| < \delta$

then $r_1 := r_1 + 1$; $Y[h] := \gamma$[значение]; if $j_1 = n$ then begin

if $r_1 = n$ then R ← 1([Y]); else begin $r_1 := 0$; $j_1 := 0$

end end end

else begin $j_0 := j_0 + 1$; h:= γ[индекс];

if |Y[h] - γ[значение]|< δ then $r_0 := r_0 + 1$;

Y[h]:= γ[значение]; if $j_0 = n$ then begin if $r_0 = n$

then R ← 1([Y]) else begin $r_0 := r_0 + 1$; $j_0 := j_0 + 1$ end

end end $EMPTY^0$ ← $EMPTY^0$ + sem($EMPTY^0$),1([γ]); L ← M_1 od

Начальные установки семафоров в $ЭМ_0$: sem(BX_0^0)= 0, $\tilde{m}_{BX_0^0} = \emptyset$; sem(R)=0, $\tilde{m}_R = \emptyset$; sem($EMPTY^0$) = m, $\tilde{m}_{EMPTY^0} = \{\nu_1, \ldots, \nu_m\}$.

Начальные значения переменных: j = r = 0. Элементы Y[i] массива Y[1:n] содержат в начальный момент значения $x_i^{(-1)}$, удовлетворяющие условию: $|x_i^{(-1)} - x_i^{(0)}| > \delta$, i = 1,2,...,n.

Процесс delta проверяет условие окончания итераций. Если оно выполнено, вектор X , хранящийся в массиве Y ,засылается в семафор R, предназначенный для результата. Занесение вектора (x_1, x_2,..., x_n) в R инициирует процесс, прекращающий вычисления и возвращающий ресурсы подсистемы в систему.

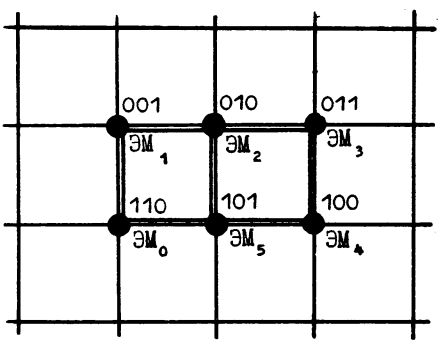

Рис.3

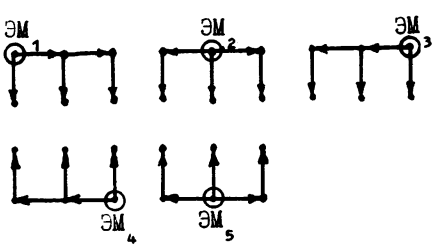

Рис.4

Передача коммутационного сообщения H{A_i, ВСЕ; f, i, x_{if}}, i ∈ {1,2,...,r}, f ∈ {0,1} из $ЭМ_i$ во все остальные машины адресной подсистемы Q, |Q| = n + 1, происходит под управлением основанной на D(z,m)-адресации [6] путевой процедуры с одновременным поиском пути.В этом случае для передачи коммутационного сообщения из $ЭМ_i$ использу-

ются линии, образующие корневое покрывающее дерево подсистемы Q с корнем ЭМ$_i$, $i \in \{0,1,\ldots,n\}$. Тем самым фиксируются пути передачи вычисленного на k-й итерации значения $x_i^{(k)}$, $i \in \{1,\ldots,n\}$, $k \in \{0, 1,\ldots\}$. Так, например, для подсистемы Q, состоящей из 6 ЭМ (см. рис.3), адреса которых A_0= 110, A_1= 001, A_2= 010, A_3= 011, A_4= 100, A_5= = 101, корневые покрывающие деревья при D(1,3)- адресации показаны на рис.4.

Предотвращение смешения в одной ЭМ$_j$, $j \in \{1,2,\ldots,n\}$, значений $x_i^{(k)}$ и $x_q^{(k+1)}$, $i,q \in \{1,\ldots, n\}$, достигается тем, что в процесс iteration$_i$ вводятся x_{i0} и x_{i1}, хранящие текущие значения x_i при k-й и (k+1)-й итерациях. При этом процессы result$_{i0}$ и result$_{i1}$ обеспечивают накопление во всех ЭМ подсистемы Q значений $x_i^{(k)}$ для фиксированного k в x_{if}, $f \equiv k(\mathrm{mod}2)$, $i = 1,2,\ldots,n$. В силу того, что между машинами подсистемы Q возможна передача значений x_i, $i = 1,2,\ldots,n$, только двух последовательных итераций k-й и (k+1)-й, $k \in \{0,1,\ldots\}$, смешения значений не произойдет.

Заключение

Ядро операционной системы ЭМ вводит средства для порождения/уничтожения виртуальных ЭМ, связанных виртуальными линиями межмашинных обменов. При этом количество виртуальных ЭМ и виртуальных линий не ограничено (в пределах ресурсов памяти ЭМ) каким-либо наперед заданным числом. Последнее выражает свойство наращиваемости вычислительной системы без изменения ее архитектуры: каждая виртуальная ЭМ и каждая виртуальная линия могут быть заменены физическими, и наоборот. Указанные реконфигурации изменяют только показатели производительности, надежности, живучести и т.д. вычислительных систем с программируемой структурой.

Литература

1. КОРНЕЕВ В.В., ХОРОШЕВСКИЙ В.Г. Вычислительные системы с программируемой структурой. -Электронное моделирование, 1979, №1, с.42-52.

2. КОРНЕЕВ В.В., ХОРОШЕВСКИЙ В.Г. Архитектура вычислительных систем с программируемой структурой. - Новосибирск, 1979. - 48 с. (Препринт/ИМ СО АН СССР, ОВС-10.)

3. КОРНЕЕВ В.В., ХОРОШЕВСКИЙ В.Г. Структура и функциональная организация вычислительных систем с программируемой структурой. - Новосибирск, 1979. - 48 с. (Препринт/ИМ СО АН СССР, ОВС-11.)

4. КОРНЕЕВ В.В., МОНАХОВ О.Г., ТАРКОВ М.С. Вычислительная система с программируемой структурой на базе ЭВМ "Электроника 60". Отчет ИМ СО АН СССР, Новосибирск, 1979, 129 с.

5. КОРНЕЕВ В.В. Элементарная машина однородной вычислительной системы с программируемой структурой. -Кибернетика, 1980, №1, с.75-81.

6. КОРНЕЕВ В.В., МОНАХОВ О.Г. Организация межмашинных взаимодействий в вычислительных системах с программируемой структурой. -Электронное моделирование, 1980, №6, с. 16-22.

7. DIJKSTRA E.W. The structure of the "The"-multiprogramming system. - Comm.of ACM,1968,v.11, N 5,p.341-347.

8. КОРНЕЕВ В.В., МОНАХОВ О.Г. О децентрализованном распределении заданий в однородных вычислительных системах. -В кн.: Архитектура вычислительных систем с программируемой структурой (Вычислительные системы, вып.82). Новосибирск, 1980, с.3-17.

Поступила в ред.-изд.отд.
4 марта 1981 года