МАТЕМАТИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ
(Вычислительные системы)
1981 год Выпуск 89

УДК 681.323

ВОПРОСЫ РАЗРАБОТКИ МАТРИЧНЫХ КОМПИЛЯТОРОВ

В.А.Райхлин, А.С.Медведев, В.Г.Мотягин

I. В в е д е н и е. Известны следующие пути снижения временных затрат на выполнение функций компиляции: микропрограммная реализация компиляторов, повышение уровня внутреннего языка системы, аппаратная поддержка компиляторов. Микропрограммная реализация позволяет ускорить процесс компиляции благодаря полному использованию всех возможностей, предоставляемых аппаратурой машины [I]. Но само написание компиляторов на языке микрокоманд достаточно трудоемко. С повышением уровня внутреннего языка связаны многочисленные разработки, начиная с аппаратной (микропрограммной) реализации отдельных конструкций языка высокого уровня до практически полного подобия входного и внутреннего языков [2]. Это облегчает задачу компиляции и убыстряет процесс ее выполнения. Однако ориентация систем на конкретный язык высокого уровня приводит к тому, что трансляция с других языков выполняется недостаточно эффективно.

Аппаратная поддержка компиляции подразумевает использование ряда средств. Это - аппаратные стеки, аппаратно реализованные списковые структуры, ассоциативная память, быстрая буферная память. Применение аппаратных стеков ускоряет этап синтаксического анализа [3-5]. Аппаратная реализация списков - удобное средство обработки текстовой информации, имеющей списковую структуру [6]. Наличие быстрого буфера характерно для любого метода. Наибольшую перспективу связывают с использованием ассоциативной памяти - для ускорения синтаксического анализа [7], организации сложного информационного поиска [8-I0], сортировки [II] и символьной обработки [I2-I3]. При этом, как правило, размеры АЗУ полагаются достаточно большими.

В настоящей работе процесс компиляции организуется на базе сравнительно небольшой по объему, но довольно сложной по структуре ассоциативной памяти (логико-запоминающей среды [I4]). За основу при разработке аппаратного компилятора принята следующая блок-схема специализированного матричного процессора (рис. I,а). Здесь М - операционная матрица, БФП - буферная память, УУ - устройство управления, УС - устройство сопряжения с оперативной памятью (ОП) системы; I-3 - шины управления, 4 - шина обмена информацией с ОП, 5 - шина результата анализа, 6 - шина настройки, 7,8 - шины обмена управляющими сигналами с системой, 9 - шина организации внутренних прерываний.

Эффективность аппаратной компиляции характеризуется дополнительными издержками на оборудование при заданном повышении быстродействия. Как минимум, буфер (рис. I,б) содержит два слоя - загрузки БФI и выгрузки БФ2. Обмен информацией между каждым слоем буфера и матрицей происходит за один такт. Высокая эффективность матричной компиляции достигается при условии выравнивания времени обработки в матрице и суммарного времени загрузки и выгрузки. При этом время компиляции определяется только обменом с ОП и размеры матрицы неизбыточны. Процесс можно ускорить, если часть промежуточной информации не отсылать в ОП, а хранить в дополнительных буферах БФ3,...,БФк,... .

В обрамление операционной матрицы входит ряд регистров (см. рис. I,в). Они предназначены для маскирования строк (МУ) и столбцов (МХ) матрицы, указания обрабатываемого байта (УБ), индикации результатов операции (РИ), хранения кодов сравнения или записи (СХ) и содержимого отдельных строк (РЧ). Имеются дополнительные регистры для организации сдвигов (СДI, СД2) и хранения условных масок (УМI, УМ2), получающихся в процессе матричной обработки. Необходимые анализы реализуются в регистре РИ (СД2).

С разработкой матричных компиляторов связано решение ряда задач. В их числе: разработка языка написания компиляторов, ориентированных на матричную аппаратуру; разработка алгоритма матричной компиляции в целом; определение неизбыточных размеров буфера и матрицы; оценка эффективности матричных компиляторов; синтез операционной матрицы по заданному составу процедур. В статье намечаются возможные подходы к решению этих задач на примере разработки аппаратного транслятора с языка Ассемблера ЕС ЭВМ.

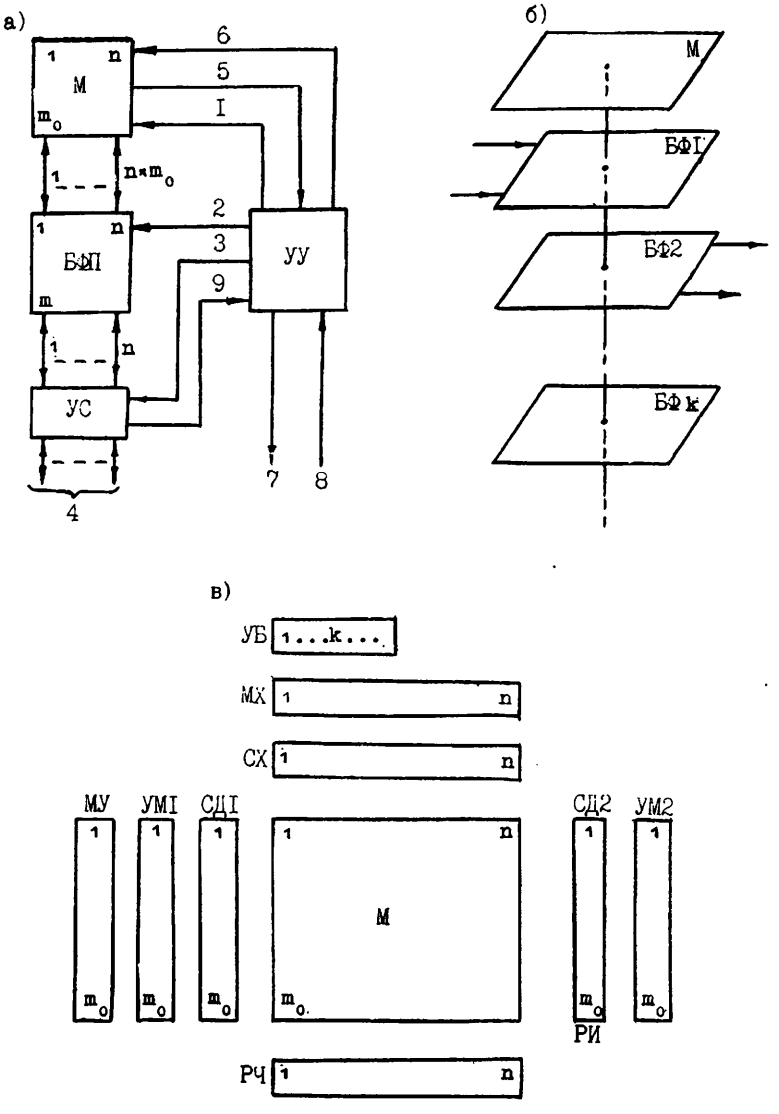


Рис. I. Блок-схема специализированного матричного процессора.

2. Процедуры и алгоритм компиляции. Компиляцию в целом можно рассматривать как процесс преобразования таблиц. Особенно это относится к программам, написанным на языках низкого уровня. Компилятор преобразует заданную таблицу (исходную программу на языке программирования) при помощи некоторых постоянных таблиц (таблицы мнемонических кодов операций и др.) и таблиц, составляемых в процессе компиляции (таблицы символов и др.), в результирующую таблицу (машинную программу). Анализ показывает, что на матричной аппаратуре, способной выполнять функции ассоциативного поиска, чтения, записи, сдвига, простейшие логические операции над масками, хорошо реализуются основные операции над таблицами. Однако удобства написания компиляторов и стремление к повышению их эффективности требуют разработки более специального состава процедур.

При этом необходимо учесть следующее.

Несколько одновременно обрабатываемых предложений исходной программы составляют блок. В силу ограниченных размеров матрицы блоки обрабатываются фрагментами.

Большинство языков программирования (в том числе язык Ассемблера) не фиксируют положение полей. Поэтому части одного поля могут оказаться в разных обрабатываемых фрагментах.

Исходная программа имеет символьную структуру. Поэтому удобно проводить по байтную обработку. Однако она не должна исключать обработки матрицы как единого целого.

Маски в специализированном матричном процессоре (СМП) служат для выделения обрабатываемой информации. Логическое преобразование масок определяет всю логику обработки. Поэтому надо предусмотреть удобное описание действий над масками.

Полный состав процедур СМП-компилятора с языка Ассемблера ЕС ЭВМ приведен в таблице. Основными среди них являются процедуры условного сдвига и стирания (процедуры 5-I3). Макрокоманды обработки масок (процедуры I7, I8) позволяют записывать действия над масками в виде логических формул. Для составления программы компиляции с использованием введенных процедур разработан простой язык описания.

В целом, матричный компилятор - это комплекс из программной части, выполняемой на процессоре основной ЭВМ, и микропрограммной части, реализованной на матричном процессоре. Необходимость такого разделения вызвана следующим.

Таблица

№	Процедура	Мнемонический код	
I	Сдвиг влево на I бит	SHBIL	
2	вправо		SHBIR
3	Сдвиг влево на I байт	SHBYL	
4	вправо		SHBYR
5	Сдвиг влево до символа	SHNEL	
6	вправо		SHNER
7	Сдвиг влево символов	SHEQL	
8	вправо		SHEQR
9	Стирание-лево до символа	CLNEL	
I0	право		CLNER
II	Стирание-лево после символа	CLEQL	
I2	право		CLEQR
I3	Стыковка	DOCK	
I4	"+A"	ADD	
I5	"Итого"	SUM	
I6	"="	FIND	
I7	Логика горизонтальная	LOGH	
I8	вертикальная		LOGV
I9	Расширение единиц вниз	PROLB	
20	вверх		PROLT
2I	Запись горизонтальная	WRHOR	
22	вертикальная		WRVER
23	Чтение горизонтальное	RDHOR	
24	вертикальное		RDVER
25	Чтение БФk:= М	BFOUT	
26	$\underline{\text{БФk}}$ & М	BOUTA	
27	БФk v М	BOUTO	
28	Запись М:= М + БФk		BFINX
29	М & БФk		BFINA
30	$\underline{\text{М v БФk}}$		BFINO
3I	БФk		BFIN
32	Переход	BRAN	
33	на подпрограмму		SUBPR
34	Управление обменом с ОП	CONTR	

Противоречие между быстрой обработкой и сравнительно медленной загрузкой ограничивает круг задач, которые можно эффективно решать на матричном процессоре.

Для хранения всей промежуточной информации, помимо буфера, необходим определенный объем оперативной и внешней памяти.

Организация доступа к устройствам ввода-вывода требует соответствующего программного обеспечения.

Спецпроцессор является одним из внешних устройств машины, работающих под контролем программной части, которая входит в состав операционной системы как одна из системных обрабатывающих программ.

Введение укрупненных операций над масками позволило сформулировать блочный алгоритм работы матричного процессора (рис. 2). Для улучшения согласования времени обработки и обменов вся обработка разделена на предварительную, обязательную, ждущую и заключительную.

Программная часть выполняет диспетчерские функции и функции, связанные с приемом, передачей, сортировкой, макрогенерацией и формированием массивов информации. В дальнейшем планируются основные функции макрогенерации и сортировки реализовать на матричном процессоре. За счет выполнения основных функций компиляции на процессоре объем программной части существенно уменьшается по сравнению с полностью программным вариантом. Это позволяет снизить количество проходов трансляции и число обращений к внешней памяти. Организацию процесса компиляции в целом иллюстрирует рис.3.

3. Размеры СМП и оценка эффективности. Время матричной компиляции, по условию, определяется числом обращений V к оперативной памяти. Величина V зависит от объема N исходной программы и числа строк m буфера. В общем случае

$$V(N,m) = A\frac{N^2}{m} + B\frac{N}{m} + CN + D.$$

Здесь A,B,C и D - некоторые статистические постоянные, характеризующие компилируемую программу. Анализ показывает, что использование введенных процедур при условии их принятой реализации даже в случае $m=1$ приводит к существенному снижению V по сравнению с полностью программной компиляцией. При этом $AN^2 + BN \gg CN + D$, $N \geq 500$. Производительность матричной компиляции во многом определена величиной m. Есть основания предположить, что выбор $m = 10\text{-}20$ обеспечит повышение скорости компиляции, в среднем, мини-

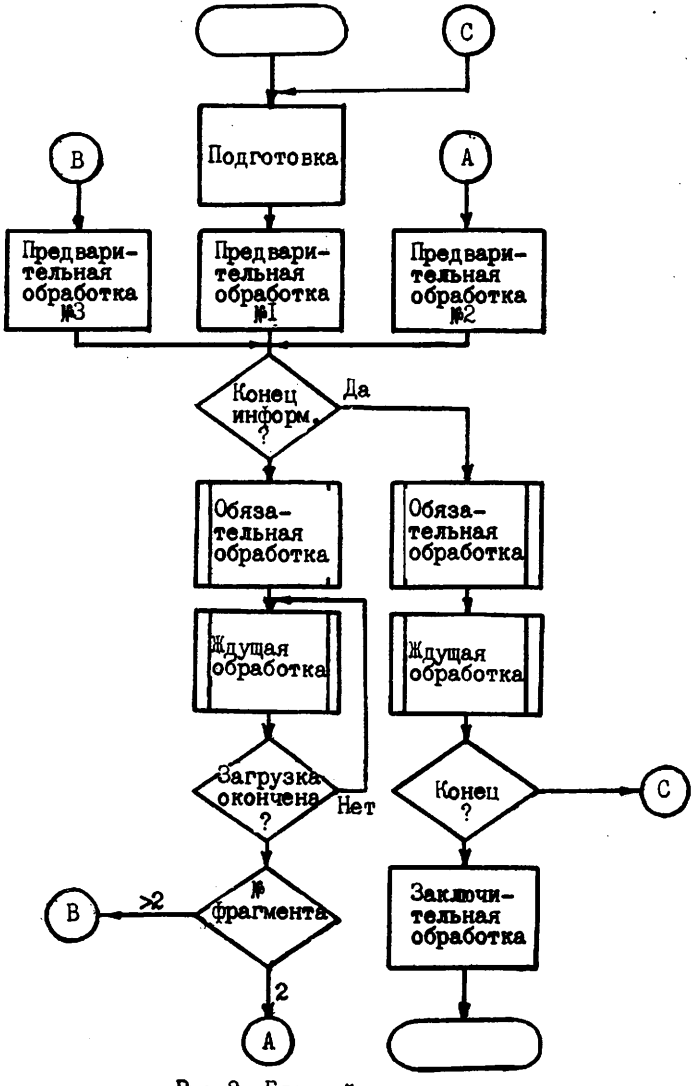


Рис.2. Блочный алгоритм.

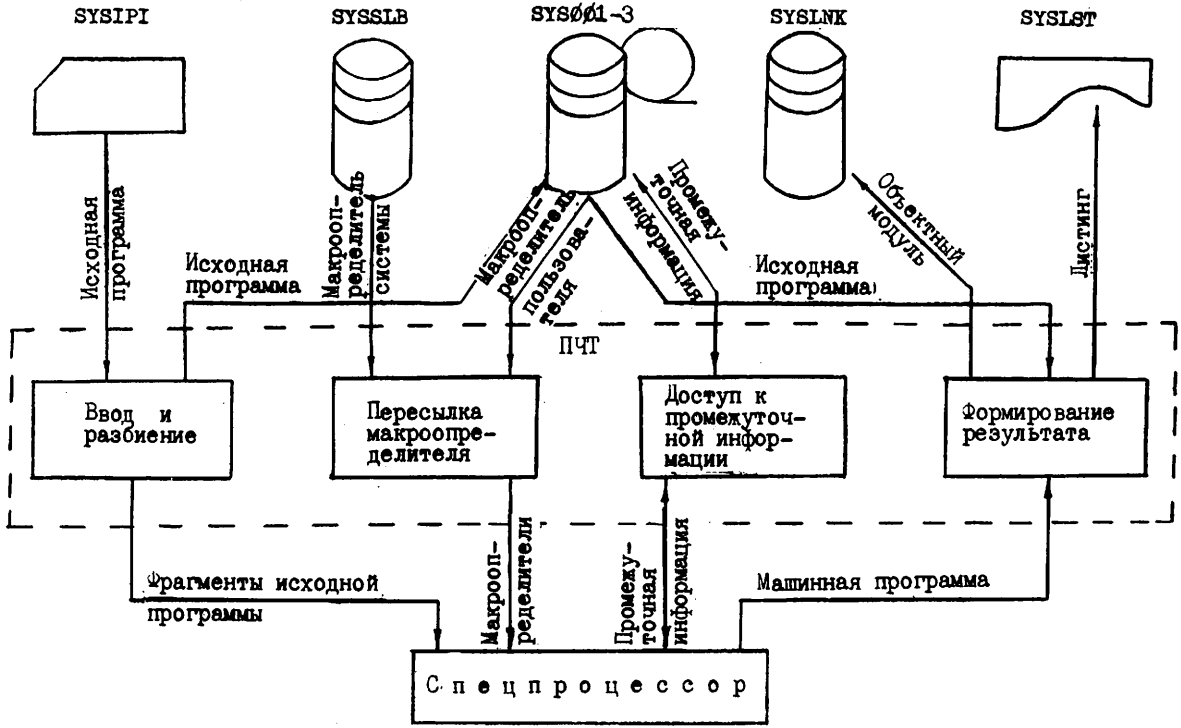


Рис.3. Организация процесса компиляции.

мум на порядок по сравнению с программным вариантом. Выбор значений $m > 20$ не должен дать заметного эффекта в этом смысле при $N \le 1000$ и может быть продиктован только требованием совмещения времени обработки и обменов.

Обработка отдельных фрагментов информации в матричном процессоре идет порциями по m_0 строк, $m_0 \le m$. Программы обработки различных фрагментов в общем случае отличаются и могут идти по разным путям. Неизбыточное число строк матрицы m_0 можно найти из условия согласования времени обработки и обменов по множеству фрагментов $\{j\}$ и путей $\{l\}$. Пусть $V_l^j(m)$ - объем окончательного результата обработки j-го фрагмента по l-му пути; $\beta_i^{j,l,r}(m/m_0)$ - число повторений i-й процедуры при обработке j-го фрагмента (порциями по m_0) по l-у пути на r-м линейном участке программы; τ_i - относительное (ко времени обращения к ОП) время выполнения i-й процедуры. Тогда относительное время обработки j-й порции по l-у пути

$$T_l^j(m/m_0) = \sum_i \beta_i^{j,l}(m/m_0)\tau_i ,$$

$$\beta_i^{j,l}(m/m_0) = \sum_r \beta_i^{j,l,r}(m/m_0),$$

должно удовлетворять условию

$$T_l^j(m/m_0) \le [V^{j-1}(m) + mn]/32,$$

$$V^{j-1}(m) = \min_l [V_l^{j-1}(m)] .$$

Здесь учтено, что обращение к ОП происходит словами по 32 бита.

Теперь можно записать систему равенств $(j,l = 1,2,\dots,)$

$$\{\sum_i \sum_r \beta_i^{j,l,r}(m/m_0^{j,l})\tau_i = [V^{j-1}(m) + mn]/32, \quad r\in\{1,2,\dots,k^j\},$$

где k^j - суммарное число линейных участков программы обработки j-го фрагмента, $V^0(m) = 0$. Необходимое значение m_0 определено соотношением $m/k = m_0 \ge \max_{j,l}(m_0^{j,l})$, $k=1,2,\dots$. Если для некоторых j и l соответствующее равенство нельзя удовлетворить никаким $m_0^{j,l} \le m$, то следует увеличить m и повторить процесс вычислений.

За основу при оценке эффективности матричной компиляции в целом могут быть приняты, например, такие соображения. Пусть пол-

ное исключение процедуры компиляции из цикла работы системы дает относительный выигрыш во времени обработки пакета заданий, равный в среднем γ. При использовании матричного процесса этот выигрыш снижается до величины $(1-\eta)\gamma$, где η – относительное время матричной компиляции (по отношению к программному варианту). Обозначим: C_M – первоначальная стоимость ЭВМ (включая стоимость питания, обслуживания и т.д.), C'_M – стоимость ЭВМ при наличии матричного процесса, δC_M – относительное приращение стоимости, T – срок окупаемости, который полагаем одинаковым в обоих случаях. Стоимость I часа машинного времени $C_1 = C_M/T$, $C'_1 = C'_M/T > C_1$. Себестоимость решения одного и того же пакета заданий $C = C_1\tau$, $C' = C'_1\tau[1-(1-\eta)\gamma]$. Здесь τ – время решения на исходной ЭВМ. За показатель эффективности использования матричного процессора примем величину $Э = C/C' = \{(1+C_M)[1-(1-\eta)\gamma]\}^{-1}$. Из условия $Э>1$ получаем $\delta C_M < (1-\eta)\gamma/[1-(1-\eta)\gamma]$. Отсюда следует, что для задач вычислительного характера, когда $\gamma \approx 0{,}2$, выбор слишком малых значений η (меньше 0,I) не оправдан.

Аналогичные оценки можно ввести и для случая использования в качестве аппаратного компилятора отдельной ЭВМ (компилятор-приставка), обслуживающей одновременно M машин, $1 \le M \le M_{max}$, $M_{max} = (1-\gamma)/(\gamma\eta_П)$. Поскольку при этом достигается максимальный выигрыш γ в повышении производительности системы, то показатель $Э_П = C/C'_П = \{(1+\delta C^П_M)(1-\gamma)\}^{-1}$. Здесь $\delta C^П_M = \delta C_П/M$, $\delta C_П$ – отнесенная к C_M стоимость приставки. Введем параметр $k = M/M_{max}$ – коэффициент загрузки приставки. Тогда

$$Э_П = \{(1-\gamma) + \delta C_П \frac{\eta_П \gamma}{k}\}^{-1}.$$

Из условия $Э_П > 1$ имеем $\delta C_П < k/\eta_П$. Отношение $k/\eta_П = M\gamma/(1-\gamma)$ при данном M есть величина постоянная. Поэтому с ростом $\eta_П$ пока $M \le M_{max}$ из-за снижения $\delta C_П$ показатель $Э_П$ несколько растет. В отношении приставки поэтому имеется определенная свобода выбора аппаратных решений. Однако использование отдельных ЭВМ-компиляторов может быть оправдано только при сравнительно высоких коэффициентах загрузки. Действительно, условие $Э_П > Э$ эквивалентно такому

$$\delta C_П < \eta\left[1 + \delta C_M\left(1 + \frac{1-\gamma}{\eta\gamma}\right)\right]\frac{k}{\eta_П}.$$

При $\gamma = 0{,}2$, $\eta = 0{,}I$ приближенно получаем $\delta C_П < M\delta C_M$.

4. Синтез операционной матрицы. Предложенный состав процедур можно реализовать и на простейшей ассоциативной матрице, но при этом каждая процедура развернется в циклическую микропрограмму. Для получения требуемой производительности придется пойти на чрезмерное увеличение числа строк матрицы и буфера, что снизит эффективность матричного процессора и увеличит его размеры в целом. Вот почему применение логико-запоминающей среды в данном случае связано с их сравнительно узкой специализацией.

Логико-запоминающая среда - это итеративная двумерная структура, каждый элемент которой содержит триггерную память, для хранения обрабатываемой информации, и автоматную часть. При заданном алгоритме задача синтеза такой среды сводится к определению структуры связей, схемы элемента и значений сигналов на границах среды. Достаточным условием реализуемости произвольного алгоритма на ее основе является возможность его декомпозиции на множестве процедур поиска, распознавания или преобразования кодов [I5-I7]. Если задан состав процедур, то их интерпретация применительно к использованию логико-запоминающей среды совместно с методом модифицированных таблиц переходов [I7] позволяет сравнительно просто синтезировать искомую структуру.

В связи с синтезом операционной матрицы ограничимся рассмотрением процедур 5-I3 (см. таблицу). Они выполняются по следующим алгоритмам.

Процедуры 5-8.

I. $q = 0$.

2. Сравнение с символом признака в указанном (k-м) байтном столбце. Если в j-й строке произошло совпадение (несовпадение) идти к 3, иначе переход к 4.

3. Блокировка сдвига j-й строки.

4. Сдвиг влево (вправо) на I байт с занесением в крайний правый (левый) байт символа пробела.

5. $q = q + 1$.

6. Если $q < k$, переход к 2.

7. Конец.

Процедуры 9-I2.

I. Одновременное сравнение с символом признака во всех байтных столбцах. Если в j-й строке имеет место совпадение при значениях $k_{min} \le k \le k_{max}$, идти к 2, иначе переход к 3.

2. Запись новой информации в байты: $(k_{max}+1),\ldots$ - для процедуры 9; $1,\ldots,(k_{min}-1)$ - для 10; $1,\ldots,(k_{max}-1)$ - для 11; $(k_{min}+1),\ldots$ - для 12.

3. Конец.

Процедура 13.

1. Одновременное сравнение с символом признака во всех байтных столбцах. Если в j-й строке имеет место совпадение при $k_{min} \le k \le k_{max}$, идти к 2, иначе переход к 4.

2. Сдвиг правых $(k_{max}+1),\ldots$ байтов на 1 байт влево с заполнением крайнего правого байта символом пробела.

3. Переход к 1.

4. Конец.

Обработка в данном случае имеет символьный характер, ведется побайтно. Поэтому для построения матрицы удобно использовать байтные строчные модули (рис.4,а). Основу модуля составляет одномерный ассоциативный каскад из 8-ми ячеек, в котором реализуется поиск на равенство (сигнал настройки $\gamma_1 = 1$) или неравенство ($\gamma_1 = 0$). Боковой выход ячейки ($\{\alpha_i^1\}$ и $\{m_i^1\}$ - символ и маска признака, $a_{j,i}$ - содержимое элемента памяти ячейки)

$$\varphi_i = \varphi_{i-1}[\gamma_1(\bar{\alpha}_i^1 \bar{a}_{j,i} \vee \alpha_j^1 a_{j,i}) \vee \bar{\gamma}_1(\alpha_i^1 \bar{a}_{j,i} \vee \bar{\alpha}_i^1 a_{j,i}) \vee \bar{m}_i^1],$$

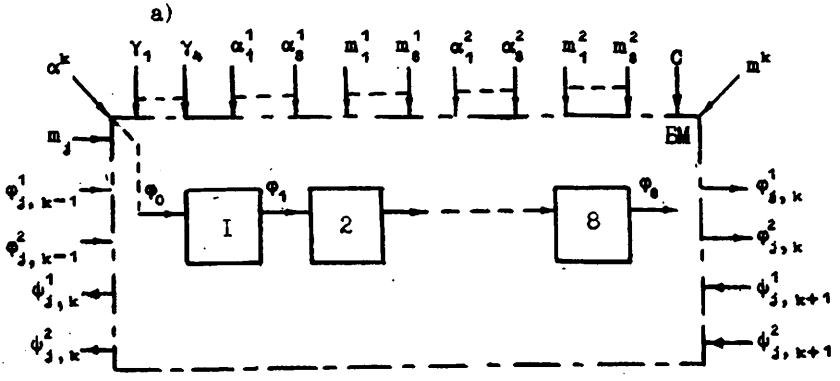
$$i = 1,\ldots,8.$$

Граничный сигнал $\varphi_0 = \alpha^k$. Выполнение операции сравнения в k-м байте инициируется подачей сигнала $\alpha^k = 1$. Для реализации блокировки сдвига или записи в байтном модуле организуются левые (правые) входы и выходы анализа $\varphi_{j,k}^1 = \varphi_8 \vee \varphi_{j,k-1}^1$, $\varphi_{j,0}^1 = 0$; $\phi_{j,k}^1 = \varphi_8 \vee \phi_{j,k+1}^1$, $\phi_{j,9}^1 = 0$.

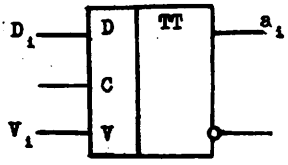
По условию, в качестве элемента памяти ячейки используется синхронный 2-ступенчатый DV-триггер (рис.4,б). Функция его управляющего входа (входа блокировки)

$$V_i = V^k m_i^2, \quad i = 1,\ldots,8;$$

$$V^k = \overline{[\bar{\gamma}_2\bar{\gamma}_3(\varphi_{j,k}^1 \vee \phi_{j,k}^1) \vee \gamma_2\bar{\gamma}_3\gamma_4\phi_{j,k+1}^1 \vee \bar{\gamma}_2\gamma_3\gamma_4\phi_{j,k}^1 \vee \gamma_2\gamma_3\bar{\gamma}_4\varphi_{j,k}^1 \vee \bar{\gamma}_2\gamma_3\bar{\gamma}_4\bar{\phi}_{j,k+1}^1 \vee \gamma_2\gamma_3\gamma_4\bar{\varphi}_{j,k-1}^1]m_j m^k}.$$



б)



в)

№ процедур	Y_1	Y_2	Y_3	Y_4
5	1	0	0	0
6	1	0	0	1
7	0	0	0	0
8	0	0	0	1
9	1	0	1	1
10	1	1	1	0
11	1	0	1	0
12	1	1	1	1
13	1	1	0	1

г)

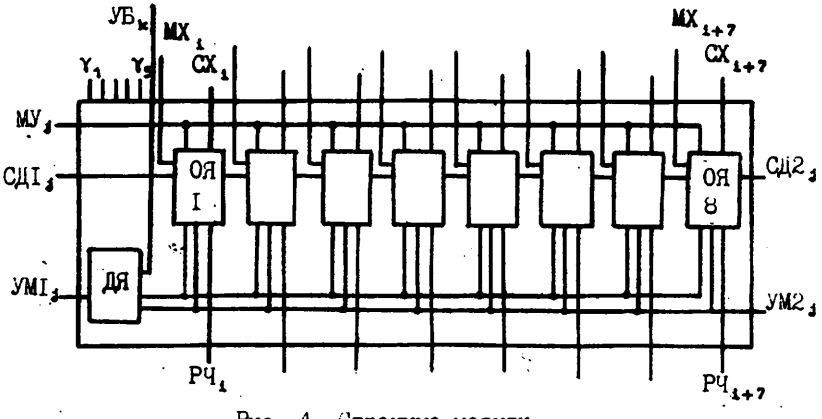


Рис. 4. Строчные модули.

Здесь m_j и m^k - маски j-й строки и k-го байта, $\{m_i^2\}$ - маска записи. Значения сигналов настройки для различных процедур указаны в таблице (рис.4,в). При поступлении синхроимпульса С указатель байта блокируется ($\varphi_0 = 0$). Поэтому в модуле используются дополнительно 2 триггера для сохранения на это время результатов анализа $\varphi^1_{j,k}$, $\phi^1_{j,k}$. Выходы сдвига байтного модуля

$$\varphi^2_{j,k} = a_8 m^k \vee \varphi^2_{j,k-1}\bar{m}^k; \quad \phi^2_{j,k} = a_1 m^k \vee \phi^2_{j,k+1}\bar{m}^k.$$

Функция возбуждения элемента памяти ячейки

$$D_i = \bar{Y}_2\bar{Y}_3\bar{Y}_4\, a_{i+1} \vee \bar{Y}_2\bar{Y}_3 Y_4 a_{i-1} \vee Y_3\alpha_i^2, \quad i=1,\ldots,8.$$

Здесь $a_0 = \varphi^2_{j,k-1}$, $a_9 = \phi^2_{j,k+1}$.

5. Результаты разработки. Алгоритм работы СМП-транслятора с языка Ассемблера ЕС ЭВМ реализован на языке предложенных процедур. Объем полученной микропрограммы в несколько раз меньше, чем в случае минимального состава процедур. При использовании спецпроцессора количество просмотров исходного текста программы, если объем доступной памяти не менее 16К, сокращается с 11 до 4. Объем программной части составляет не более 30% объема программного транслятора. При этом на долю матричного процессора приходится не менее 90% общего объема работ, что обеспечивает требуемое повышение скорости трансляции в целом.

Структура специализированного матричного процессора разработана до уровня функциональных схем. Синтезирована полная схема байтного строчного модуля БИС (рис.4,г), размещаемого в корпусе на 40 выводов. Модуль состоит из 8 одинаковых основных ячеек (ОЯ) и одной дополнительной (ДЯ), которая служит для логической обработки и временного запоминания результатов сравнения в байте. Использование элементов "с открытым коллектором" и мультиплексирования по входам модуля позволило сократить число его внешних связей. Степень интеграции модуля - 700 э.в. При $m = m_0 = 20$, $n=64$ и выборе 16 слоев буфера для построения матрицы, ее обрамления и буферной памяти требуется 160 корпусов БИС, 732 корпуса 155ХЛ1, 320 корпусов 155РУ2, 456 корпусов 1ЛБ553. Эта часть процессора требует для своего размещения не более 100 ТЭЗ (150 х 150 мм, 135 выводов). Дополнительно необходимо учесть оборудование устройства микропрограммного управления (не более 10 ТЭЗ) и специального

(4-байтного) селекторного канала связи с системой (не более 50 ТЭЗ). Таким образом, матричный процессор в целом можно разместить на 160 ТЭЗ, что, не считая питания, составляет 2/3 стандартной рамы ЕС.

Литература

1. ПЕЛИПЕНКО В.И., ШТРАНИХ И.В. ЭВМ "Мир-1" в качестве аппаратного транслятора для больших машин. Гл. I. -М., 1973. - 60 с. (Препринт/ИК АН СССР: № 119).

2. КОРОЛЕВ Л.Н. Структура ЭВМ и их математического обеспечения. -М.: Наука, 1974. - 255 с.

3. ХОПГУД Ф. Методы компиляции. -М.: Мир, 1972. - 160 с.

4. КАЛИТИН С.С. Некоторые вопросы реализации транслятора на основе стекового автомата. - Труды МЭИ, 1972, вып. 118, с.9-19.

5. **GAJDOSIK M. On the use of the stack in translators. - Inform.Proces.Machines,1971, N 15, p.45-55.**

6. АСРАТЯН Р.Э. и др. Командное обеспечение обработки символьной информации. -В сб.: Многопроцессорные вычислительные системы. М., 1975, с. 84-90.

7. ПРОНИНА В.А., ЧУДИН А.А. Реализация синтаксического анализа в ассоциативном параллельном процессоре. -Автоматика и телемеханика, 1975, № 8, с. 106-112.

8. **RUDBERG D.A., HANNA W.A. The threedimentional computer : a multiplane array processor. - Comput. and Elec. Eng.,1975,v. 2, N 2-3, p.141-148.**

9. ЧОГОВАДЗЕ Р.А. Трехмерный ассоциативный параллельный процессор. -В кн.: Вопросы кибернетики. Однородные микроэлектронные структуры. М., 1973, с. 69-73.

10. АСРАТЯН Р.Э., ВОЛКОВ А.Ф., ЛЫСИКОВ В.Т. Об одном подходе к аппаратной реализации отдельных этапов трансляции. - УСиМ, 1976, № 3, с. 27-31.

11. ВИЛЕНКИН С.Я., ЧАРНАЯ Н.С. Алгоритм сортировки на однородных ассоциативных структурах. - Программирование, 1975, № 2, с. 30-37.

12. **BEAVEN P.A., LEWIN D.V. An associative parallel processing system for non-numerical computation.-Comput.J., 1972,v.15, N 4,p.343-349.**

13. АКСЕНОВ В.П. Ассоциативные процессоры и области их применения. Обзор. - Зарубежная радиоэлектроника, 1977, №1, с.52-80.

14. КАУТЦ У.Х. Однородные логико-запоминающие среды и большие интегральные схемы. -В сб.: Синтез автоматов и управление на сетях связи. М., 1973, с. 73-83.

15. Однородные микроэлектронные ассоциативные процессоры /Под ред. И.В.Прангишвили. -М.: Сов. радио, 1973. - 280 с.

16. ФЕТ Я.И. Массовая обработка информации в специализированных однородных процессорах. - Новосибирск: Наука, 1976. - 200 с.

17. ВАРШАВСКИЙ и др. Однородные структуры. - М.: Энергия, 1973. - 152 с.

Поступила в ред.-изд.отд.
2 января 1980 года