В Ы Ч И С Л И Т Е Л Ь Н Ы Е  С И С Т Е М Ы

Сборник трудов
1965 г. Института математики СО АН СССР Выпуск I8

# К ВОПРОСУ ЭКОНОМИИ ОПЕРАТИВНОЙ ПАМЯТИ ЦВМ

Н.В. Бухгольц, В.Ф. Дьяченко, В.Г. Лазарев,
К.К. Черныиев, В.А. Шаров

## В в е д е н и е

Одной из важных сфер применения цифровых автоматов с программным управлением является использование их в системах автоматического контроля и управления.

Специфика работы ЦВМ в этих системах, когда перечень решаемых задач определен, а время работы является достаточно большим, позволяет использовать для записи программ постоянную память - ПЗУ. Как правило, ПЗУ отличается высокой надежностью, небольшими габаритами и малым потреблением энергии. В таких случаях особое значение приобретает минимизация объема оперативной памяти - МОЗУ, так как "цена" этого вида памяти выше, чем "цена" ПЗУ.

В данной работе рассматривается метод минимизации рабочего поля программы, позволяющий с учетом определенных ограничений, накладываемых на программу, определить минимально необходимое для решения задачи количество ячеек МОЗУ.

## I. Особенности машинной программы как объекта оптимизации

Структура цифровых автоматов с программным управлением базируется на принципах адресности и программного управления. В соответствии с первым из них для исходных данных, констант, промежуточных и конечных результатов, а также команд (операций) программы отводятся определенные ячейки памяти машины, каждая из которых сопоставляется с фиксированным числом, называемым адресом. Таким образом, память машины представляет собой множество ячеек (адресов) $Q=\{a_i\}$, которые подразделяются на два подмножества $Q_n$ и $Q_c$ для хранения переменных и постоянных величин $x$ и $c$, соответственно.

Множество переменных величин $G_n=\{x_i\}$ определенным образом сопоставляется с подмножеством ячеек $Q_n$, которые называются рабочими ячейками программы. Рабочие ячейки программы используются для засылки хранения и выборки исходных данных, промежуточных и конечных результатов, а также вспомогательных величин (включая в некоторых случаях машинные команды), используемых для реализации заданного алгоритма. Некоторым переменным перед "входом" в программу могут быть присвоены начальные значения.

Множество постоянных величин $G_c=\{c_i\}$ включает в себя неизменяемые команды программ и константы и определенным образом сопоставляется с подмножеством ячеек (адресов). Для цифровых автоматов с программным управлением, используемых в системах автоматического контроля и управления с фиксированным перечнем решаемых системой задач, подмножество $Q_c$ может представлять собой ПЗУ.

Принцип программного управления состоит в том, что управление работой машины по переработке информации осуществляется в соответствии с алгоритмом решения задачи, записанным в виде последовательности машинных команд $K_i$, называемой программой.

Формируя из нескольких машинных команд укрупненные операторы, которые характеризуются законченностью действий по преобразованию информации либо структурными особенностями выделяемого участка программы ([1]-[3]), можно получить операторную схему программы. На операторной схеме программы можно выделить три типа операторов [4]: действующие $D(x)$, логические $P(x)$ и варьирующие $V(x)$.

Участки программы, состоящие только из действующих операторов, меняют либо используют содержимое рабочих ячеек программы, но не меняют заданной последовательности выполнения машинных операций. Такие участки программы будем называть линейными и обозначать меткой $L:[x_i]$, где в скобках перечисляются все переменные $x_i \in G_n$, а знаком "тильда" ($\sim$) над переменной будем обозначать отсутствие любых воздействий на переменные (выборка, присваивание новых значений) со стороны действующих операторов "помеченного" участка программы. Отсутствие знака указывает на то, что на помеченном участке программы переменная участвует в операциях.

Наличие в операторной схеме программы логических операторов $P(x)$ приводит к появлению в программе разветвлений. Логические операторы осуществляют условную либо безусловную передачи управления на участки программы, которые могут содержать операторы типа $D(x)$, $P(x)$ и $V(x)$.

В логических выражениях условных операторов могут использоваться как постоянные, так и переменные величины.

Помимо разветвлений и линейных участков программа может содержать такие циклические участки, обязательным элементом которых является варьирующий оператор $V(x)$. Циклический участок программы, управляемый оператором $V(x)$, также может содержать операторы типа $D(x)$, $P(x)$ и $V(x)$.

Тривиальным распределением памяти машины является такое распределение, при котором с каждым объектом программы сопоставляется отдельная ячейка памяти.

При составлении программы обычно учитывается возможность хранения нескольких переменных в одной ячейке МОЗУ, а также используется ряд других приемов, которые в той или иной степени связаны с изменением самой программы. Формализация этих приемов связана с проблемой оптимизации программы с точки зрения ее длины и затрат машинного времени и является темой специального рассмотрения.

Ниже решается следующая задача: без внесения изменений в программу найти такое отображение множества переменных программы на ее рабочее поле, чтобы число рабочих ячеек было минимальным.

## 2. Построение пространственно-временной диаграммы следов переменных

Сформулируем условия совместимости адресов нескольких переменных. Будем исходить из того, что существует тривиальное распределение ячеек МОЗУ для множества $G_n$ . Упорядоченное подмножество адресов команд программы (инструкций) $M_{инстр.} = \{K_j\} \in Q_c$, $j = 1, \ldots, N$, представляющее собой натуральный ряд чисел, можно отобразить на ось абцисс (см. рис. I), где $K_1$ и $K_N$ - начальный и конечный адреса программы. На оси ординат условно наносятся идентификаторы переменных $x_i \in G_n$ индексы которых также могут быть упорядочены. В такой системе координат для каждой переменной можно задать функцию $\Phi_i = f(K_j)$; причем $\Phi_i = 1$, если на данном участке программы переменная $x_i$ участвует в операциях либо сохраняется возможность использовать присвоенное ей ранее значение; $\Phi_i = 0$ - в противном случае.

Единичное значение функции $\Phi_i$ назовем следом переменной $x_i$ на подмножестве $M_{инстр.}$, или просто следом переменной. Условную линию, параллельную оси абцисс и совмещенную со следом переменной (то-есть область определения функции $\Phi_i$ ), назоем трассой следа переменной $x_i$ . Семейство следов всех переменных программы назовем пространственно-временной диаграммой следов переменных. Будем считать, что знак " x " на следе переменной $x_i$ означает, что в данном месте программы осуществляется выборка значения $x_i$ из памяти машины, а знак " / " означает, что переменной $x_i$ присвоено новое значение.

Рассмотрим правила построения пространственно-временной диаграммы следов переменных с учетом особенностей структуры программы, описанных выше.

I) Программа, состоящая из линейного участка. На трассах следов переменных программы расставим значки " / " и " x ", смысл которых определен выше. Тогда можно сказать, что след каждой переменной начинается с первого от начала программы значка " / " и обрывается на последнем значке " x ". Это относится к следам всех переменных, за исключением тех, значения которых заданы перед "входом" в программу (исходные данные), а также тех, значения которых должны сохраняться в памяти до конца вычислений по машинной программе (конечные результаты). В первом случае след переменной прочеркивается от начала машинной программы, а во втором случае до ее конца.



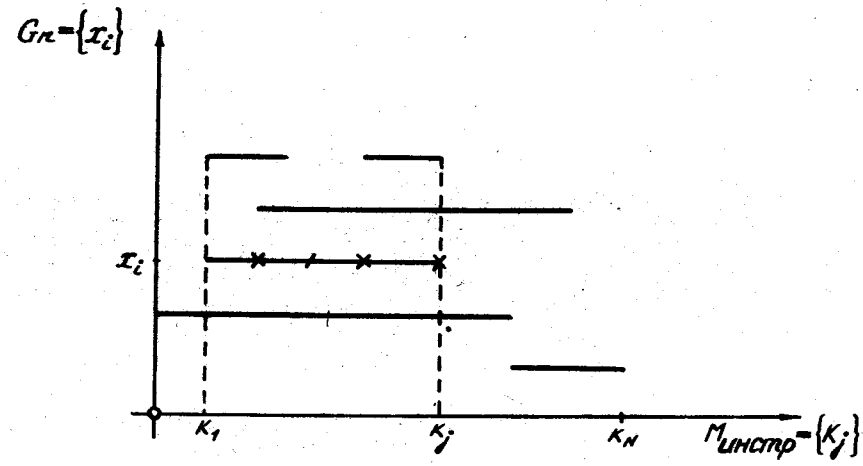
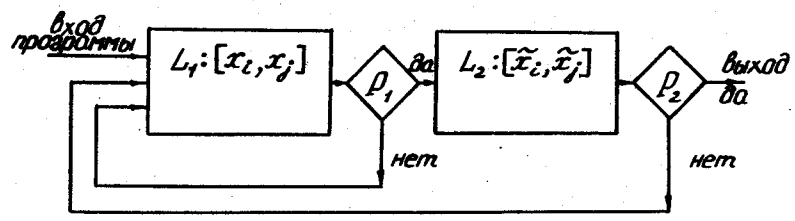
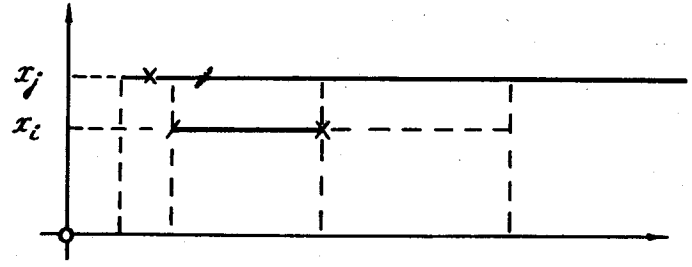



Рис.I. Пространственно-временная диаграмма следов переменных.




а)




б)

Рис.2. Программа с разветвлениями.

2) П р о г р а м м а  с  р а з в е т в л е н и я м и. В качестве примера такой программы возьмем программу, блок-схема которой показана на рис. 2,а. Рассмотрим эту программу относительно двух переменных $x_i$ и $x_j$. В точках трассы следа, соответствующих командам программы, производящим выборку или засылку в память значений переменных, поставим соответствующие значки (рис. 2,б). Анализ расположения этих значков показывает, что блок I можно пометить меткой $L_1$: $[x_i, x_j]$, а блок 2 – меткой $L_2$: $[\widetilde{x_i}, \widetilde{x_j}]$. Будем считать, что перед входом в программу значение переменной $x_j$ задано. Исходя из данного выше определения, след переменной $x_j$ должен быть прочерчен на всем множестве адресов программы, так как в каждой точке блока с меткой $L_2$: $[\widetilde{x_i}, \widetilde{x_j}]$ сохраняется возможность использования значения переменной $x_j$, потому что условный оператор $P_2$ передает управление в начало блока $L_1$: $[x_i, x_j]$, причем первым на трассе следа стоит знак " $x$ ". По тем же соображениям след переменной $x_j$ будет выглядеть так, как показано на рис. 2,б.

3) Ц и к л и ч е с к и е  п р о г р а м м ы. Среди циклических программ можно выделить два типа:

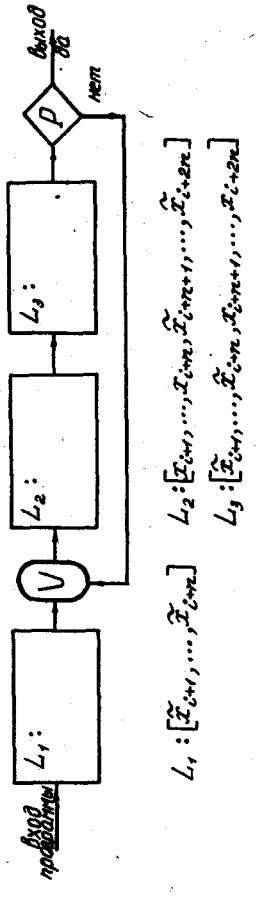
а) циклические программы с фиксированным числом повторений;

б) циклические программы с переменным числом повторений.

Как тот, так и другой типы циклов могут выполняться, как с переадресацией, так и без неё.

Диаграммы для программ без переадресации строятся так же, как описано выше. Чтобы определить правила построения диаграммы следов переменных для циклических программ с переадресацией рассмотрим условную программу, блок-схема которой представлена на рис.3,а. Будем считать, что: I) число повторений линейных участков программы с метками $L_2$: и $L_3$:, управляемых варьирующим оператором V, фиксировано и равно $n$ (то-есть параметр цикла равен $n$); 2) значения переменных $x_{i+1}, \ldots, x_{i+n}$ заданы перед входом в программу (исходные данные), а значения переменных $x_{i+n+1}, \ldots, x_{i+2n}$ представляют собой конечные результаты, причем соответствующее значение результата засылается в МОЗУ по команде с адресом $j$ после переработки соответствующих исходных данных (например, после переработки $x_{i+1}$ засылается значение $x_{i+n+1}$).

Будем исходить из того, что ячейки МОЗУ для переменных программы распределены тривиально. Группу следов переменных



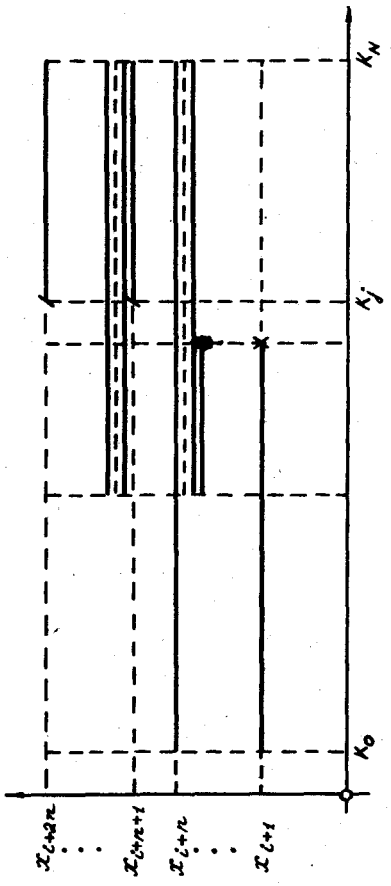




Рис.3. Циклическая програм...

$x_{i+1}, \ldots, x_{i+n}$ будем называть блоком следов переменных с выборкой их значений или просто блоком с выборкой, а группу следов переменных $x_{i+n+1}, \ldots, x_{i+2n}$ – блоком следов переменных с засылкой их значений или просто блоком с засылкой.

Легко понять, что на различных участках программы один и тот же блок следов переменных может рассматриваться как блок с выборкой либо как блок с засылкой.

Заметим, что для простоты рассматриваются только циклы с единичным шагом переадресации, с одной командой выборки исходных данных и с одной командой засылки в МОЗУ результатов. Однако эти ограничения являются несущественными.

Расставим на трассах следов переменных значки "/" и "х". После первого выполнения участков программы, помеченных метками $L_2$: и $L_3$: , освобождается ячейка МОЗУ, отведенная под переменную $x_{i+1}$ , а в точке $K_j$ трассы следа занимается ячейка, отведенная под переменную $x_{i+n+1}$ . Однако следы остальных переменных $x_{i+2}, \ldots, x_{i+n}$ должны быть прочерчены до конца программы в соответствии с определением следа переменной, данным выше.

При повторении участков программы $L_2$: и $L_3$: в конце блока $L_2$ освобождается ячейка МОЗУ, отведенная под переменную $x_{i+2}$ , а в точке $K_j$ занимается ячейка, отведенная под переменную $x_{i+n+2}$ , и т.д. Таким образом, для циклических программ с переадресацией возможность присвоения различным переменным одинаковых адресов определяется не только длиной следа переменной, но и временной картиной работы циклической программы по отдельным переменным.

Введем понятие толщины следа переменной. Для циклической программы с переадресацией след переменной должен прочерчиваться на участках программы, для которых сохраняется возможность использования значения переменной столько раз, сколько повторяются участки программы, управляемые варьирующим оператором. Следовательно, толщина следов переменных определяется параметром цикла и последовательностью переработки их значений.

При составлении циклических программ с переменным числом повторений тоже можно пользоваться понятием толщины следа переменной. Последняя будет определяться максимально возможным числом повторений участка программы в цикле и последовательностью переработки значений переменных. Очевидно, толщина следа переменной на участках программы, не управляемых варьирующим оператором (т.е. следа, не входящего в блок следов), всегда равна единице.



Так как трасса следа любой переменной $x_i$ параллельна оси абсцисс, то длина проекции следа переменной $x_i$ на ось абсцисс (или просто длина проекции следа переменной $x_i$ ) равна длине следа этой переменной. Толщиной проекции следа переменной $x_i$ на отрезке $K_i$ , $K_j$ будем считать толщину следа на этом же отрезке. Будем говорить, что проекции следов переменных $x_i$ и $x_j$ не пересекаются, если эти проекции не имеют общих точек.

Построение пространственно-временной диаграммы следов переменной и их проекций позволяет выявить все возможности совмещения адресов различных переменных.

## 3. Выбор минимального числа адресов

Введем следующие определения.

ОПРЕДЕЛЕНИЕ 1. Следы переменных $x_i$ и $x_j$ , хотя бы один из которых не входит ни в один блок следов, называются совместимыми, если их проекции не пересекаются.

ОПРЕДЕЛЕНИЕ 2. Следы переменных $x_{i+l}$ и $x_{j+m}$ , входящие в блоки двух различных циклов с параметрами $n_i$ и $n_j$ , соответственно, называются совместимыми, если выполняются два условия: 1) их проекции не пересекаются и 2) при $1 \leqslant l < n_i$ и $1 \leqslant m < n_j$ совместимы также следы переменных $x_{i+l+1}$ и $x_{j+m+1}$ .

ОПРЕДЕЛЕНИЕ 3. Следы переменных $x_{i+l}$ и $x_{i+n+m}$ , входящие соответственно в блок с выборкой и в блок с засылкой одного и того же цикла с параметром $n$ , называются совместимыми, если выполняются два условия: 1) суммарная толщина их проекций не превышает параметра цикла и 2) при $1 \leqslant l < n$ и $1 \leqslant m < n$ совместимы также следы переменных $x_{i+l+1}$ и $x_{j+n+m+1}$ .

ОПРЕДЕЛЕНИЕ 4. Группой совместимых следов называется такая группа, в которой любая пара следов совместима.

ОПРЕДЕЛЕНИЕ 5. Максимальной группой совместимых следов называется такая группа, добавление в которую хотя бы одного из оставшихся следов делает ее группой несовместимых следов.

ОПРЕДЕЛЕНИЕ 6. Множество $\Pi$ групп совместимых следов называется полным, если каждый след пространственно-временной

диаграммы следов переменных программы входит хотя бы в одну группу совместимых следов этого множества.

**ОПРЕДЕЛЕНИЕ 7.** Пусть следы переменных $x_{i+\ell}, \dots, x_{j+m}$ входят в одну группу множества $\Sigma$ и содержатся в блоках следов. Тогда множество $\Sigma$ называется замкнутым, если при $1 \le \ell < n_i, \dots, 1 \le m < n_j$ следы переменных $x_{i+\ell+1}, \dots, x_{j+m+1}$ также входят в одну группу этого множества.

Очевидны следующие леммы:

**ЛЕММА 1.** Переменным $x_1, \dots, x_t$ программы можно присвоить один и тот же адрес, если их следы входят в одну группу совместимых следов полного и замкнутого множества $\Sigma$.

**ЛЕММА 2.** Пусть полное и замкнутое множество содержит $\rho$ групп, тогда для переменных программы достаточно иметь $\rho$ адресов.

**ОПРЕДЕЛЕНИЕ 8.** Минимальным множеством $\Sigma_{мин}$ называется такое полное и замкнутое множество групп совместимых следов, которое содержит наименьшее число групп среди всех возможных полных и замкнутых множеств $\Sigma$.

Таким образом, если удастся найти $\Sigma_{мин}$, то в соответствии с леммой 2, можно будет выбрать минимальное число адресов.

Нетрудно заметить, что задача нахождения минимального числа адресов аналогична задаче минимизации числа внутренних состояний частичного конечного автомата([5]-[7]). Поэтому для получения $\Sigma_{мин}$ по парам совместимых следов можно воспользоваться процедурой, описанной в работе [8] для минимизации частичного конечного автомата. Заметим, однако, что эта процедура не всегда позволяет получать минимальный частичный автомат. Вместе с тем, как будет показано ниже, учитывая свойства отношения совместимости следов переменных, данная процедура всегда приводит к $\Sigma_{мин}$.

Так как здесь рассматриваются программы, в которых один цикл имеет не более одной команды выборки и одной команды засылки, то нетрудно доказать следующие леммы:

**ЛЕММА 3.** Пусть следы переменных $x_{i+\ell}, x_{j+m}, \dots, x_{k+\rho}$, содержащиеся в блоках различных циклов с параметрами $n_i, n_j, \dots, n_k$, соответственно, где



$1 \le \ell < n_i, 1 \le m < n_j, \dots, 1 \le \rho < n_k$ входят в одну и ту же группу множества $\Sigma$ максимальных групп совместимых следов. Тогда, если множество $\Sigma$ полное, то в нем любая группа, содержащая след переменной $x_{i+\ell+1}$, содержит также и следы переменных $x_{j+m+1}, \dots, x_{k+\rho+1}$.

**ЛЕММА 4.** Пусть следы переменных $x_{i+\ell}$ и $x_{i+n+m}$, содержащиеся соответственно в блоке выборки и в блоке засылки одного и того же цикла с параметром $n (\ell, m < n)$, входят в одну и ту же группу множества $\Sigma$ максимальных групп совместимых следов. Тогда, если множество $\Sigma$ полное, то в нем любая группа, содержащая след переменной $x_{i+\ell+1}$, содержит также и след переменной $x_{i+n+m+1}$.

Из определения 7 и лемм 3 и 4 вытекает справедливость следующей теоремы:

**ТЕОРЕМА 1.** Любое полное множество $\Sigma$ максимальных групп совместимых следов переменных замкнуто.

**ДОКАЗАТЕЛЬСТВО.** Пусть $(x_{i+\ell}, x_{j+m}, \dots, x_{k+\tau})$ – одна из максимальных групп множества $\Sigma$. Тогда, если $x_{i+\ell}$, $x_{j+m}, \dots, x_{\rho+s}$ ($\rho \le k$, а $\ell = \tau$ при $\rho = k$) входят в различные блоки циклов с параметрами $n_i, n_j, \dots, n_\rho$, соответственно, а множество $\Sigma$ замкнуто, то следы всех переменных $x_{i+\ell+1}$, $x_{j+m+1}, \dots, x_{\rho+s+1}$ также должны быть включены в одну группу множества $\Sigma$. В соответствии с определениями 2 и 3 все следы всех переменных $x_{i+\ell+1}, x_{j+m+1}, \dots, x_{\rho+s+1}$ должны быть между собой попарно совместимыми, так как совместимы следы переменных $x_{i+\ell}, x_{j+m}, \dots, x_{\rho+s}$.

По леммам 3 и 4 любая максимальная группа совместимых следов, включающая след переменной $x_{i+\ell+1}$, включает также следы переменных $x_{j+m+1}, \dots, x_{\rho+s+1}$. По условию теоремы I в множестве $\Sigma$ всегда имеется хотя бы одна максимальная группа, включающая след переменной $x_{i+\ell+1}$. Следовательно, в множестве $\Sigma$ должна быть по крайней мере одна группа, включающая следы всех переменных $x_{i+\ell+1}, x_{j+m+1}, \dots, x_{\rho+s+1}$. Из этого следует, что $\Sigma$ замкнутое.

Теорема доказана.

ТЕОРЕМА 2. Если задана программа с тривиальным распределением памяти, то минимальное число адресов переменных программы равно числу групп минимального полного множества $\Sigma_{мин}$ максимальных групп совместимых следов переменных.

. ДОКАЗАТЕЛЬСТВО. Допустим, что число адресов переменных минимально. Если при этом с каждым адресом сопоставлена группа минимального полного множества максимальных групп совместимых следов переменных, то теорема доказана. Поэтому предположим, что среди адресов найдется хотя бы один, который сопоставлен с находящейся в минимальном полном множестве $\Sigma'$ немаксимальной группой $\alpha$ . Тогда образуем новое множество $\Sigma$ причем в множестве $\Sigma$ каждая немаксимальная группа $\alpha$ множества $\Sigma'$ заменена на максимальную $\beta$ , включающую группу $\alpha$ . Множество $\Sigma$ , во-первых, содержит такое же число групп, что и множество $\Sigma'$ . Значит, множество $\Sigma$ минимальное. Во-вторых, множество $\Sigma$ полное, так как множество $\Sigma'$ полное, а $\Sigma \supset \Sigma'$ . В-третьих, вследствие теоремы I множество $\Sigma$ замкнутое. Поэтому в соответствии с леммой I со следами переменных, вошедших в одну группу множества $\Sigma$ , может быть сопоставлен один адрес. Таким образом, сопоставляя с группами множества $\Sigma$ адреса переменных, в соответствии с леммой 2, получаем программу с минимальным числом адресов.

Теорема доказана.

ОПРЕДЕЛЕНИЕ 9. Две группы множества $\Sigma$ называются пересекающимися, если в каждую из них включен хотя бы один след одной и той же переменной, и непересекающимися - в противном случае.

ОПРЕДЕЛЕНИЕ IO. Отображение множества адресов переменных при тривиальном распределении памяти на минимальное множество $\Sigma_{мин}$ называется правильным, если в $\Sigma_{мин}$ не содержатся пересекающиеся группы. В остальных случаях отображения считаются неправильными и они не представляют практического интереса. Поэтому необходимо выбирать такие $\Sigma_{мин}$ , в которых нет пересекающихся групп. Если получено $\Sigma_{мин}$ с пересекающимися группами, то, исключая каким-либо образом из отдельных групп те или иные следы, добиваются того, чтобы $\Sigma_{мин}$ с пересекающимися группами перешло в $\Sigma_{мин}$ с непересекающимися



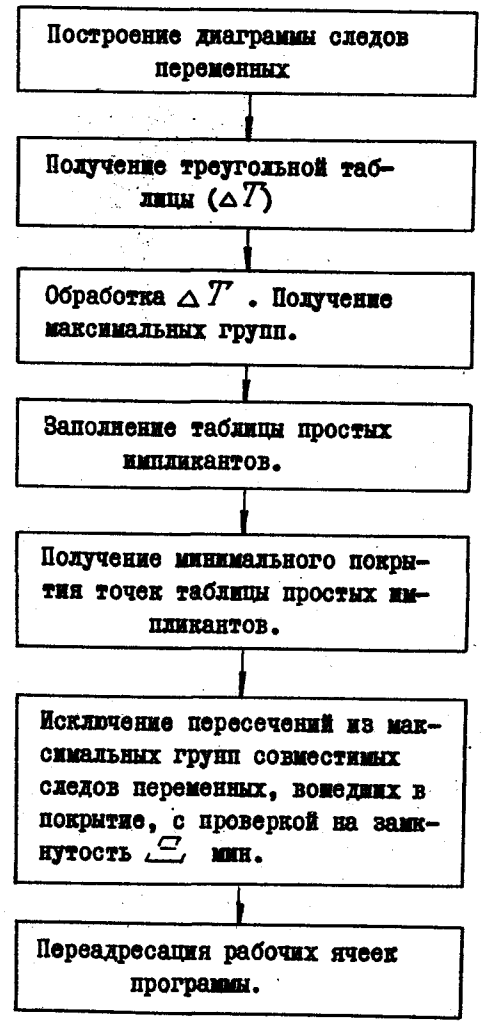



Рис. 4. Блок-схема программы минимизации числа ячеек памяти.

группами. Важно только, чтобы адреса переменных, входящих в один и тот же блок, следовали в таком же порядке, как и в пространственно-временной диаграмме следов переменных.

Учитывая справедливость теоремы 2, для выбора $\Sigma_{мин}$ можно использовать упрощенную процедуру, применяемую при минимизации определенного класса частичных автоматов [9]. Однако в таблице простых импликантов (см.[9]) вместо импликанта указывается не максимальная группа, а нормальное эквивалентно связанное множество максимальных групп (см.[8]). Вместо же конъюнкций булевой функции в таблице простых импликантов указываются следы переменных.

Упрощенная блок-схема программы минимизации числа ячеек памяти МОЗУ приведена на рис. 4, где $\Delta\nabla$ – треугольная таблица (см.[8]).

## 4. Пример минимизации числа рабочих ячеек программы

В качестве примера рассмотрим программу вычисления функции $y = \sqrt{x}$ $(x>0)$ методом итераций по формуле:

$$y_{i+1} = \frac{1}{2}\left(y_i + \frac{q_x}{y_i}\right), \qquad (1)$$

где $q_x$ – мантисса нормализованного аргумента.

Начальное значение $y_0$ определяется по формуле:

$$y_0 = \alpha q_x + \beta, \qquad (2)$$

где $\alpha$ и $\beta$ – константы, значения которых выбраны так, что достаточная точность вычислений (младший разряд аргумента) получается при двух приближениях по формуле (2). При отрицательном и нулевом значениях аргумента $y$ присваивается значение 0. Значения аргумента перед входом в программу и значения результата находятся на одном из регистров АУ ЦВМ ($C_M$) и представляются в виде двоичных кодов с запятой, фиксированной перед старшим разрядом мантиссы.

При построении пространственно-временной диаграммы следов переменных необходимо определять начало и конец следа на множестве адресов $M_{инстр.}$ с точностью до адреса отдельной операции по переменным величинам. Следовательно, перед решением задачи минимизации числа рабочих ячеек программа должна быть представлена на языке, позволяющем учитывать особенности кон-

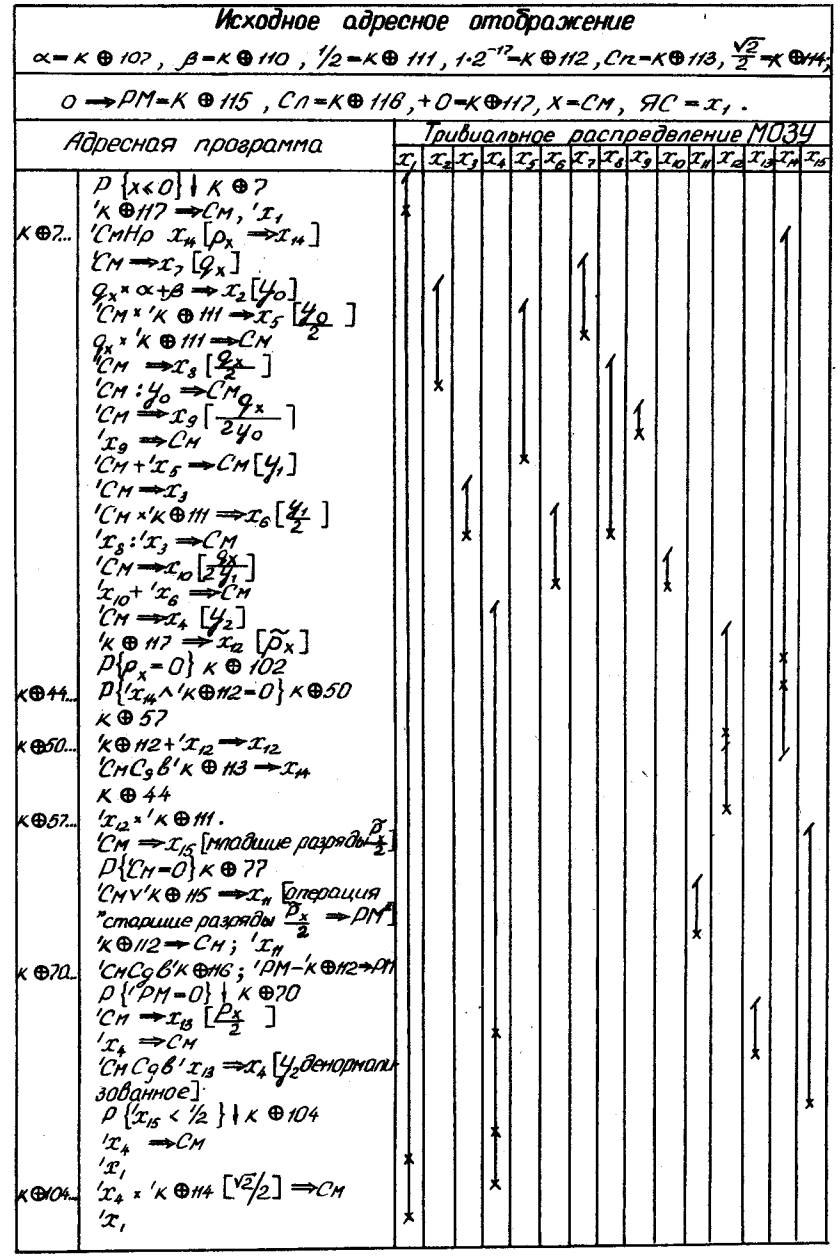



Рис.5. Программа вычисления функции $y=\sqrt{x}$ в адресном языке и пространственно-временная диаграмма следов переменных.

кретной ЦВМ. В качестве такого языка может быть использован адресный язык [10].

Помимо общепринятых символов, используемых в адресном языке, в программе, приведенной на рис. 5, используются символы следующих операций:

1) $C_{\mathcal{A}\mathcal{B}}$ - двухместная операция арифметического сдвига. Формула вида $AC_{\mathcal{A}\mathcal{B}}B \Rightarrow C$ означает, что результат вычисления адресной функции $A$ сдвигается в соответствии с результатом вычисления адресной функции $B$; сдвинутый код засылается по адресу $C$. Результат вычисления адресной функции $B$ представляет собой информацию о числе и направлении сдвигов (константа сдвига).

2) $Hp$ - двухместная операция нормализации кода. Формула вида $AHpB$ означает, что результат вычисления адресной функции $A$ нормализуется и остается на регистре $C_H$, а константа денормализации засылается по адресу $B$.

В качестве адресов используются также символы некоторых регистров машины.

Для рассматриваемой программы была построена пространственно-временная диаграмма следов переменных (рис. 5), по которой построена треугольная таблица (рис. 6). В результате применения указанной выше процедуры минимизации числа рабочих ячеек программы получены следующие полные и замкнутые множества $\Xi_{МИН}$.

$\Xi_1 = \{(x_1);(x_2,x_4,x_6,x_9);(x_3,x_4,x_5,x_{10});(x_7,x_8,x_{10},x_{12},x_{15});$
$(x_{11},x_{13},x_{14})\}$.

$\Xi_2 = \{(x_1);(x_2,x_4,x_6,x_9);(x_3,x_5,x_{10},x_{12},x_{15});(x_7,x_8,x_{10},x_{12},x_{15});$
$(x_{11},x_{13},x_{14})\}$.

$\Xi_3 = \{(x_1);(x_2,x_3,x_4,x_5,x_{10});(x_5,x_6,x_{12},x_{15});(x_7,x_8,x_{10},x_{12},x_{15});$
$(x_{11},x_{13},x_{14})\}$.

$\Xi_4 = \{(x_1);(x_2,x_6,x_9,x_{12},x_{15});(x_3,x_4,x_5,x_{10});(x_4,x_7,x_8,x_{10});$
$(x_{11},x_{13},x_{14})\}$.

$\Xi_5 = \{(x_1);(x_2,x_3,x_9,x_{10},x_{12},x_{15});(x_4,x_5,x_6);(x_4,x_7,x_8,x_{10});(x_{11},x_{13},x_{14})\}$.

$\Xi_6 = \{(x_1);(x_2,x_4,x_6,x_9);(x_3,x_4,x_5,x_{10});(x_7,x_8,x_{10},x_{12},x_{15});(x_{11},x_{13},x_{14})\}$.



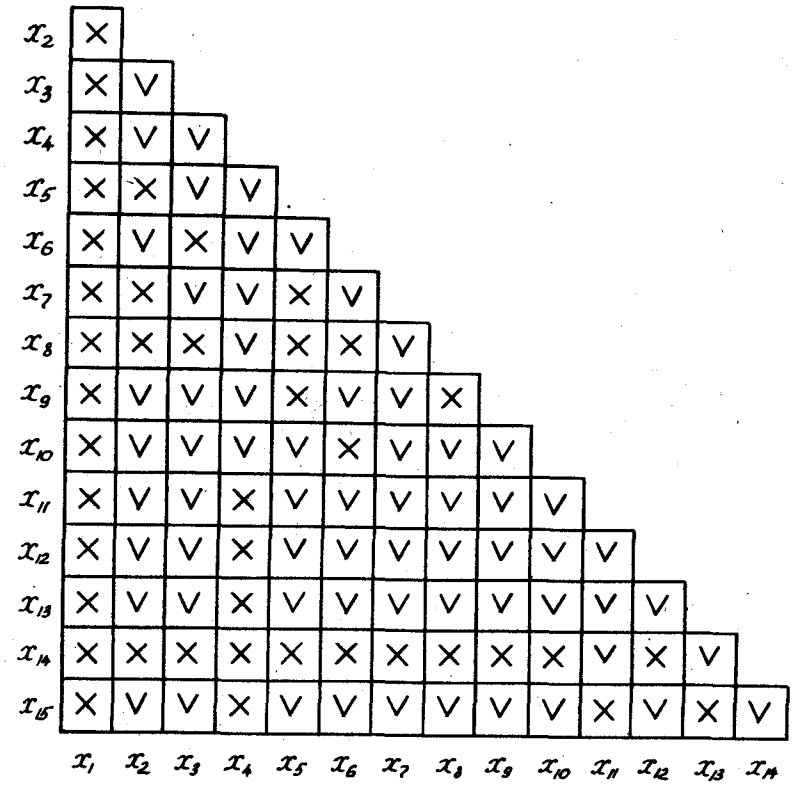



Рис.6. Треугольная таблица для диаграммы рис.5.

$$\Xi_7 = \{(x_1);(x_2,x_6,x_9,x_{12},x_{15});(x_3,x_5,x_{10},x_{12},x_{15});(x_4,x_7,x_8,x_{10});(x_{11},x_{13},x_{14})\}.$$
$$\Xi_8 = \{(x_1);(x_2,x_3,x_4,x_9,x_{10});(x_4,x_7,x_8,x_{10});(x_5,x_6,x_{12},x_{15});(x_{11},x_{13},x_{14})\}.$$
$$\Xi_9 = \{(x_1);(x_2,x_4,x_6,x_9);(x_3,x_5,x_{10},x_{12},x_{15});(x_4,x_7,x_8,x_{10});(x_{11},x_{13},x_{14})\}.$$
$$\Xi_{10} = \{(x_1);(x_2,x_4,x_8,x_9);(x_3,x_4,x_5,x_{10});(x_7,x_8,x_{10},x_{12},x_{15});(x_{11},x_{13},x_{14})\}.$$
$$\Xi_{11} = \{(x_1);(x_2,x_3,x_4,x_9,x_{10});(x_4,x_5,x_6);(x_7,x_8,x_{10},x_{12},x_{15});(x_{11},x_{13},x_{14})\}.$$

Оказалось, что для данной программы все $\Xi_1 \div \Xi_{11}$. включают по 5 максимальных групп совместимых следов. Таким образом, для данной программы достаточно 5 рабочих ячеек. Отметим, что программист средней квалификации использовал 8 рабочих ячеек.

## Выводы

I. Изложенная выше методика может быть применена к готовым программам.

2. Описанный алгоритм минимизации рабочего поля программы может быть использован в системах автоматического программирования.

3. Применение данной методики наиболее целесообразно при проектировании специализированных вычислительных устройств с фиксированным перечнем решаемых задач либо при создании библиотеки стандартных подпрограмм.

## Литература


I. Ляпунов А.А. О логических схемах программ. В сб.:"Проблемы кибернетики", вып. I, М., Физматгиз, I958, 46-74.

2. Лавров С.С. Об экономии памяти в замкнутых операторных схемах. Журнал вычисл. матем. и матем. физ., I96I, I, № 4, 687-70I.

3. Ершов А.П. Сведение задачи распределения памяти при составлении программы к задаче раскраски вершин графов. Докл. АН СССР, I962, I42, № 4, 785-787.





4. Ириницкий Н.А., Миронов Г.А., Фролов Г.Д. Программирование. М., Физматгиз, I963.

5. Глушков В.М. Синтез цифровых автоматов. М., Физматгиз, I962.

6. Кобринский Н.Е., Трахтенброт Б.А. Введение в теорию конечных автоматов. М., Физматгиз, I962.

7. Айзерман М.А., Гусев Л.А., Розоноэр Л.И., Смирнова И.М., Таль А.А. Логика. Автоматы. Алгоритмы. М., Физматгиз, I963.

8. Лазарев В.Г., Пийль Е.И. Синтез асинхронных конечных автоматов. М., Изд. "Наука", М., I964.

9. McCluskey E.J.,Jr. Minimum-state sequential circuits for a restricted class of incompletely specified flow-tables. B.S.T.J.,1962,41,N 6,1759-1768.

I0. Ющенко Е.Л. Адресное программирование. Киев, Гостехиздат, УССР, I963.



Ин-т проблем передачи
информации АН СССР



Поступила в редакцию
25.IУ.1965 г.