УДК 681.142.001.2.

ПОСТРОЕНИЕ БЛОК-СХЕМЫ ЦВМ ПО АЛГОРИТМУ ЕЕ ФУНКЦИОНИРОВАНИЯ

И.В. Иловайский, Э.Е. Сергеева

При проектировании ЦВМ либо другого дискретного устройства инженер вначале составляет блок-схему (структурную схему) устройства [1] и описывает, как должны взаимодействовать элементы блок-схемы.

Далее блок-схема детализируется, составляется структурная схема каждого узла блок-схемы. Объединив схемы узлов, инженер получает подробную блок-схему устройства, на которой показаны элементы памяти (регистры) и преобразователи информации с необходимой подробностью. Блок-схеме такого вида соответствует алгоритм функционирования устройства, показывающий, в каком порядке должны взаимодействовать элементы схемы.

Дальнейшая детализация блок-схемы (указание каждого разряда элемента схемы, введение цепей управления) дает функциональную схему устройства, которой соответствует микропрограммное описание устройства (или временная диаграмма). Затем выполняются операции технического синтеза.

В настоящее время известны системы машинного проектирования, входом в которые является либо функциональная схема, либо функционально-логическая (функциональная, детализированная до логических узлов). В частности, в работе [2] описан транслятор, исходной информацией для которого служит блок-схема устройства и временная диаграмма, а выходом – система логических уравнений. В [3] приведены алгоритмы, с помощью которых можно от описания работы ЦВМ на адресном языке перейти к системе логических уравнений. В [4] по алгоритму работы ЦВМ строится функциональная схема (без указания разрядности ее объектов).

Однако во всех этих работах не указано явно, какой абстрактный объект является исходным, какой результирующим и каков общий (не зависящий от условий описываемого авторами примера) алгоритм. В этих работах изложение ведется на примерах без указания возможной области действия описываемых методов.

Мы попытаемся дать в этой и в последующей работах несколько более общий подход, позволяющий по линейной записи алгоритма на каком-либо формальном языке получить блок-схему и функциональную схему устройства.

I.

Пусть мы имеем линейную запись некоторого алгоритма на формальном языке. Сопоставим каждому вхождению в запись алгоритма каждого оператора Q вершину графа q, а дуги графа определим так. Если в записи оператор Q_2 непосредственно следует за Q_1 и при этом после Q_1 нет передачи управления (по метке на другие операторы), то вводится дуга $q_1 \rightarrow q_2$. Если от оператора Q_1 имеются передачи управления к операторам $Q_i, \ldots, Q_n$, то вводятся дуги $q_1 \rightarrow q_i, \ldots, q_1 \rightarrow q_n$. Операторы Q делятся на две группы - операторы переработки информации A и операторы перехода по условиям P. Всякому A - оператору соответствует вершина графа с полустепенью исхода единица. P - оператору соответствует вершина с полустепенью исхода больше единицы. В этом случае полустепень исхода равна количеству возможных переходов по условиям. Полученный нами объект есть граф-схема алгоритма [5]. Роль операторов играют A - операторы, роль распознавателей-P -операторы. От определенной в [5] граф-схемы наша отличается непринципиальной особенностью - после P -оператора возможно более двух переходов по условию.

По построению каждой записи алгоритма можно сопоставить только одну граф-схему. Все записи, дающие одну и ту же граф-схему, сильно эквивалентны [5].

Любой оператор переработки информации имеет аргументы, над которыми определена его работа и результаты этой работы. Аргументы суть значения входных переменных оператора, результаты суть значения выходных переменных оператора. Разные вхождения одного и того оператора могут иметь разный набор переменных.

Оператор перехода по условию имеет аргументы, над которыми определена его работа (это его входные переменные), результаты

его работы - осуществление перехода по одному из условий.

Сопоставим каждой вершине графа символ - имя того оператора, который она представляет. Некоторые вершины могут получить одно имя, так как один и тот же оператор может иметь несколько вхождений в запись алгоритма.

Занумеруем все входные переменные (α) во всех вхождениях операторов так, что каждая переменная получит свой номер. Аналогично занумеруем все выходные переменные (β). При этом одна и та же переменная может получить один номер как входная и другой - как выходная, т.е. некоторые α_i и β_j попарно совпадают, или, иначе, одна и та же переменная имеет два обозначения - одно как входная, другое как выходная. Свяжем с каждой вершиной граф-схемы два вектора - вектор входных переменных и вектор выходных переменных. i -я компонента вектора соответствует i -й переменной. Если какая-либо входная (выходная) переменная не связана с данным вхождением оператора в запись, во входном (выходном) векторе соответствующей вершины граф-схемы компонента этой переменной пуста. В противном случае эта компонента есть имя самой переменной.

Для P -операторов вектор входных переменных определяется вышеописанным образом, а вектор выходных переменных сводится к вектору имен меток переходов по условиям. В тексте - записи алгоритма этими метками помечены вхождения операторов, на которые передается управление. Результат работы P -оператора понимается как единичное или нулевое значение соответствующей выходной переменной (метки).

Граф-схему алгоритма с подобным образом отмеченными вершинами назовем отмеченной граф-схемой алгоритма. Далее мы везде будем говорить об отмеченной граф-схеме и для краткости изложения слово "отмеченная" будем опускать.

2.

Сопоставим каждой переменной, упомянутой в записи алгоритма, регистр, а каждому оператору, упомянутому в записи алгоритма, - схему, выполняющую требуемое этим оператором преобразование информации.

Будем строить блок-схему машины, реализующую данный алгоритм. Блок-схема (структурная схема) понимается как графическое изображение устройства (машины) в виде набора функциональ-

ных устройств (схем) и информационных связей между ними. Будем изображать блок-схему графом, вершины которого соответствуют либо регистрам - переменным, либо схемам преобразования информации - операторам, а дуги определяются следующим образом.

Будем обозревать i -ю компоненту векторов входных переменных. Отметим все те вершины граф-схемы, у которых i -я компонента этого вектора не пуста. Выделим в блок-схеме вершину, соответствующую переменной a_i, и вершины, соответствующие операторам $Q_1, \ldots, Q_m$ - именам помеченных вершин граф-схемы. В блок-схеме из вершины a_i проведем дуги в вершины $Q_1, \ldots, Q_m$ (мы отождествляем вершину q и соответствующий ей оператор Q). Так поступим для всех компонент векторов входных переменных. В результате мы получим связи выходов регистров со входами схем преобразования информации.

Будем обозревать j -й - оператор. Отметим в граф-схеме все вершины, названные им. У всех отмеченных вершин в векторах выходных переменных отметим те компоненты, которые не пусты хотя бы один раз. Выпишем соответствующие этим компонентам переменные $b_1, \ldots, b_n$.

В блок-схеме выделим вершину Q_j и вершины, соответствующие переменным $b_1, \ldots, b_n$. Из вершины Q_j в блок-схеме проведем дуги к вершинам $b_1, \ldots, b_n$. Так поступим для всех операторов. В результате мы получим связи выходов схем преобразования информации со входами регистров.

Из вершин, соответствующих в блок-схеме именам меток, проведем дуги к специальной вершине τ. Эта вершина графа соответствует управляющему устройству машины. В реальной машине всякое устройство, являющееся элементом блок-схемы, может управляться от генератора τ, но мы эти связи не строим, здесь мы ограничиваемся построением схемы передачи информационных потоков, игнорируя цепи управления. Это неизбежно, так как управление связано с определением моментов времени, в которые выполняются шаги алгоритма, а этой информации в принятой нами исходной записи алгоритма нет.

Таким образом, мы дали процедуру построения блок-схемы устройства по граф-схеме алгоритма. При практическом использовании этой процедуры придется пользоваться не граф-схемой, а линейной записью алгоритма.

Приведем соображения, позволяющие делать подобную замену. Заметим, что при построении блок-схемы по граф-схеме информация о дугах последней не использовалась. С другой стороны, записи, дающие одну и ту же граф-схему, могут отличаться только порядком расстановки вхождений операторов в запись [5], то есть записи могут быть получены друг из друга всевозможными перестановками вхождений операторов (разумеется, чтобы не нарушить связность записи, необходимо в нужных местах вводить метки).

Из этого следует, что безразлично, строить ли блок-схему по граф-схеме или по записи алгоритма. Отсюда же следует, что если граф-схеме соответствует единственная блок-схема, то все записи, сильно эквивалентные между собой [5], дадут одну и ту же единственную блок-схему.

3

Построение блок-схемы по граф-схеме (а значит, и по записи алгоритма) суть процесс однозначный. Действительно, пусть по данной граф-схеме построены две различные блок-схемы $Б_1$ и $Б_2$.

Множества вершин у них будут совпадать, так как они взаимно однозначно соответствуют множеству переменных и операторов. Остается предположить, что блок-схемы различаются дугами. Заметим, что по построению дуги "переменная-переменная" и "оператор-оператор" невозможны, возможны лишь дуги типа "оператор-переменная" и "переменная-оператор".

Рассмотрим некоторую пару вершин (одну и ту же в обеих блок-схемах): вершину - переменную π и вершину - оператор q.

Возможны следующие случаи:

1. Вершины в обеих блок-схемах не соединены.

2. Вершины в обеих блок-схемах соединены идентично.

Тогда надо рассмотреть другую пару вершин.

3. В блок-схеме $Б_1$ есть дуга $\pi \to q$, а в $Б_2$ - $q \to \pi$ (или наоборот). Но это означает, что у одной и той же вершины граф-схемы переменная π входит во входной вектор, но не входит в выходной и одновременно не входит во входной, но входит в выходной (или наоборот), что невозможно.

4. В блок-схеме $Б_1$ есть дуга, соединяющая π и q, а в $Б_2$ - нет (или наоборот). Но это означает, что одна и та же переменная у одной и той же вершины граф-схемы одновременно входит в вектор входных (либо выходных) переменных и не входит, что невозможно.

5. В блок-схеме $Б_1$ есть дуги $\pi \to q$ и $q \to \pi$, а в $Б_2$ только $\pi \to q$ (либо $q \to \pi$). Исключив из рассмотрения идентичные дуги, $\pi \to q$ (либо $q \to \pi$), получим случай 4.

6. Случай, когда в $Б_1$ есть дуги $\pi \to q$ и $q \to \pi$, а в $Б_2$ нет ни одной из них (или наоборот), сводится к повторному применению случая 4.

Таким образом, возможны только случаи I либо 2, следовательно обе блок-схемы совпадают.

Обратное построение невозможно, так как из блок-схемы не виден порядок следования операторов. Поэтому данной блок-схеме соответствует множество алгоритмов такое, что множество вхождений операторов (или вершин отмеченной граф-схемы) у них одно и то же (т.е. одной блок-схеме соответствуют граф-схемы, отличающиеся лишь множествами дуг).

Выбор конкретного алгоритма из этого множества производит схема τ (через управляющие связи). Именно на этом факте основана универсальность и гибкость ЦВМ, а схемная реализация этого процесса с помощью перестройки схемы осуществляется в машинах с микропрограммным управлением (в смысле Уилкса).

4.

Уточним, что мы понимаем под регистром и схемой преобразования информации. Регистр представляет собой либо шины передачи информации, либо схему, реализующую оператор $'\pi(t) = '\pi(t-1)$, если в момент t на эту схему не поступает новая информация. Преобразования информации регистр не осуществляет.

Схема преобразования информации реализует преобразование, задаваемое оператором. Поэтому уточним понятие оператора. Алгоритм имеет входные переменные, значения которых определены при входе в алгоритм, и выходные - значения которых определены при выходе из алгоритма, и реализует некоторое преообразование входных данных в выходные. То же делает A -оператор. Поэтому A -оператор просто есть алгоритм.

Если существует запись алгоритма некоторого A - оператора, входящего в исходный алгоритм, то блок-схему можно пополнить, изъяв из нее вершину, соответствующую этому оператору, и дуги, ей инцидентные, и построив по описанному ранее правилу блок-схему для алгоритма A - оператора. Эта блок-схема, естественно, включается в общую через общие вершины - входные и выходные переменные этого оператора (алгоритма). Так поступаем, пока не дойдем до элементарного алгоритма (оператора).

Элементарным, неразложимым алгоритмом (оператором) может быть, например, оператор $\wedge, \vee, \neg$, штрих Шеффера, суммирование и т.д. Соответственно, считается элементарной и неразложимой схема, реализующая такой оператор.

P - оператор имеет входные переменные, как и A - оператор, осуществляет над их значениями некоторое преобразование, но результат этого преобразования специфичен, все выходные переменные могут иметь лишь два значения - 0,I, и лишь одна из них имеет значение I после выполнения оператора. С учетом этих ограничений P - оператор может быть разложен так же, как и A -оператор.

(Заметим, что мы всюду игнорировали структуру самой переменной и не определяли ее разрядность. Наша задача этого не требовала, тем более, что в некоторых языках в записи разрядность явно не присутствует).

Из построения видно, что предлагаемый способ позволяет получить именно блок-схему ЦВМ в ее классическом виде:

процессор $\rightleftharpoons$ управление (генератор)

Схема строится иерархически. Если какой-либо оператор раскрывается как алгоритм, то ему сопоставляется блок-схема со своим управлением τ, встроенная в общую. Можно, конечно, объединить все такие устройства управления в одно общее.

5.

Изложим алгоритм, реализующий описанное выше построение. Поскольку ЦВМ воспринимает линейную запись, входом в алгоритм должна служить линейная запись алгоритма функционирования или описание граф-схемы в виде линейно записанных матрицы смежности и матрицы, состоящей из векторов переменных. Выходом алгоритма должна быть линейная запись матрицы смежности блок-схемы или другое линейное описание.

Примем, что исходная информация задана в виде линейной записи алгоритма функционирования и списков операторов и переменных, упомянутых в этой записи. Примем, что результирующая информация - это матрица M смежности блок-схемы, определенная так, что если есть дуга $a \to b$, то элемент M_{ab} матрицы M не пуст. Матрица записана в виде списка бинарных отношений, т.е. строка начинается именем этой строки и содержит имена тех столбцов, которым соответствуют в строке матрицы M непустые элементы.

I. Для каждой переменной из списка переменных строится строка в записи матрицы M следующим образом:

Первый элемент строки есть эта переменная.

Последующие элементы строки находятся так: обозревается исходная запись алгоритма и отыскиваются те вхождения операторов в запись, где данная переменная употреблена в качестве аргумента. Оператор выписывается в строку матрицы M (естественно, повторно он не выписывается).

2. Для каждого оператора из списка операторов строится строка в записи матрицы M следующим образом:

Первый элемент строки есть этот оператор.

Последующие элементы строки строятся так: обозревается исходная запись алгоритма, и для каждого вхождения данного оператора в запись все переменные - результаты выписываются в строку матрицы M (естественно, повторно переменная не выписывается).

Выходная переменная P -оператора - это метка. Строки, указывающие связь меток со схемой управления $\mathcal{C}$, не строятся, они очевидны.

6.

Опишем программу построения блок-схемы. Исходный объект есть описание алгоритма, выполненное средствами фрагмента Φ- языка [6]. Программа состоит из двух блоков - блока реализации алгоритма и блока обмена с внешней памятью.

Блок реализации алгоритма есть программа, работающая с оперативной памятью машины и переводящая Φ- описание (помещающееся в МОЗУ) в матрицу бинарных отношений, задающую блок-схему. Предполагается, что матрица целиком помещается в МОЗУ.

Если Φ- описание велико, то оно разбивается на части и каждая часть обрабатывается независимо от других.

Получающаяся совокупность матриц не дает нужного нам описания блок-схемы, так как строки с одним и тем же именем могут встречаться более одного раза, поэтому требуется программа обмена с внешней памятью.

Блок обмена с внешней памятью объединяет в одну те строки из матриц, которые имеют одно имя, и внутри каждой строки ликвидирует повторяющиеся имена. Предполагается, что в МОЗУ одновременно находится не более двух обрабатываемых матриц и одна

результирующая подматрица. Программирование ведется на языке ЛЯПАС [7] с использованием машины М-20 и ПС-ЛЯПАС на М-20 [8].

I. Исходной информацией для работы блока реализации алгоритма служат следующие комплексы:

A - запись алгоритма работы проектируемого устройства, выполненная в Φ- языке, представляющая собой последовательность списков. Каждый список есть запись оператора Φ- языка. Каждый элемент записи оператора (см. Приложение 2) занимает полную ЛЯПАС - ячейку.

Эта запись сопровождается комплексом адресов начал списков (начал операторов) - B. Результирующий комплекс R суть матрица бинарных отношений блок-схемы и построен, как последовательность списков, начинающихся каждый именем того элемента, с выхода которого идут связи, и содержащего имена всех тех элементов блок-схемы, к которым идут эти связи. Адреса начал списков в R хранятся в комплексе U. В окончательном представлении результирующего массива адреса начал списков не используются, а списки разделяются ляпасным символом f_{10}. Это сделано для облегчения работы с внешней памятью в последующем блоке. Этот массив носит имя P.

2. Блок реализации алгоритма *реализация* состоит из программы *имена* получения списка D имен операторов и списка C имен переменных, собственно программы получения матрицы бинарных отношений *матрица*, служебной программы *ликвид*, ликвидирующей в каждом списке в R повторяющиеся имена, и служебной программы *упак*, перестраивающей R в P.

очистка - оператор очистки комплекса. Он описан в [7].

```
                реализация 114 001
очистка C ‖ очистка D ‖ очистка E ‖ очистка F ‖
0 ⇒ B2  0 ⇒ B3  0 ⇒ B4  0 ⇒ B5  имена A B C D E F M Q ‖
очистка G ‖ очистка H ‖ матрица A C D E F G H M Q ‖
очистка R ‖ очистка U ‖ ликвид G H R U ‖ очистка P ‖
упак P R ‖ § 1 P * .
```

3. Для программы *имена* исходной информацией служат комплексы A и B и служебный комплекс Q, в который занесены все разделители используемого фрагмента Φ - языка.

Результирующая информация:

C - комплекс имен переменных из A.

$\underline{D}$ – комплекс имен A – операторов (имен функциональных преобразователей) из $\underline{A}$.

$\underline{E}$ – комплекс имен P – операторов (имен схем формирования сигналов условных переходов) из $\underline{A}$.

$\underline{F}$ – комплекс адресов имен A – операторов в комплексе $\underline{A}$.

$\underline{M}$ – комплекс адресов начал записей P – операторов в $\underline{A}$.

Работа программы

– Для каждого списка в $\underline{A}$, представляющего собой тело оператора, определяется, какой оператор записан в данном списке (A – оператор, P – оператор, заголовок, окончание, безусловный переход).

– Программа членится на независимые блоки, каждый из которых обрабатывает свой тип оператора Φ – языка.

– Из оператора конца (заголовка) выписываются переменные одна за другой и заносятся в очередную ячейку комплекса $\underline{C}$, если они еще в $\underline{C}$ не записывались.

– Из операторов пересылки (A – операторов) и условных операторов выписываются переменные из списка аргументов по аналогичному правилу.

– Из списков (тел A – операторов) выписываются (неповторяющиеся) имена операторов в комплекс $\underline{D}$. Адреса всех имен операторов заносятся в комплекс $\underline{E}$.

– Аналогично строится комплекс $\underline{E}$. Адреса первых ячеек записей P – операторов заносятся в комплекс $\underline{M}$.

4. Для программы *матрица* исходной информацией служат комплексы $\underline{A}, \underline{C}, \underline{D}, \underline{E}, \underline{Q}, \underline{M}$, описанные выше.

Результирующая информация – это комплекс $\underline{G}$, запись матрицы бинарных отношений в виде:

$$M = \{a_i, \{a^j\}_i\},$$

где a^j – имена столбцов таких, что $a^j_i = 1$ в матрице смежности блок-схемы; и комплекс $\underline{H}$ – адреса начал строк матрицы M в $\underline{G}$ (адреса элементов a_i).

Работа программы

– Для каждого имени переменной (элемента комплекса $\underline{C}$) строится строка матрицы M (начинающаяся с этого имени) следующим образом (см.§§ I – II).

– Для каждого A – оператора проверяется, входит ли эта переменная в его вектор входных переменных и, если входит, то в строку матрицы M , начинающуюся этой переменной , вписывается имя этого оператора (см. §§ 2 – 5).

– Аналогично поступаем с P – операторами.

– Для каждого имени A – оператора (элемента комплекса $\underline{D}$) строится строка матрицы M следующим образом. Находятся все вхождения A – оператора с этим именем в Φ – записи, и все выходные переменные этих операторов выписываются в строку матрицы M (в очередной список в комплексе $\underline{G}$) (см. §§ I2–I5).

Адрес каждого элемента, начинающего список в $\underline{G}$, заносится в $\underline{H}$.

матрица 4I5 0I6

$\bar{o}i\ \bar{o}h\ \bar{o}j\ \bar{o}\ell$

§ 1 $\Delta j \oplus B_2 \circ\!\!-\!\!- 12\ \Delta i\ \Delta h\ \underline{C}_j \Rightarrow \underline{G}_i\ i \Rightarrow \underline{H}_h$

§ 2 $\bar{o}a$

§ 3 $\Delta a \oplus B_5 \circ\!\!-\!\!- 6\ \underline{F}a \Rightarrow t\ \Delta t\ \underline{F}a \Rightarrow p$

§ 4 $\underline{A}_t \oplus \underline{C}_j \circ\!\!-\!\!- 5\ \Delta t\ \underline{A}_t \oplus \underline{Q}_4 \circ\!\!-\!\!- 3 - 4$

§ 5 $\Delta i\ \underline{A}p \Rightarrow \underline{G}_i - 3$

§ 6 $\bar{o}a$

§ 7 $\Delta a \oplus B_{14} \circ\!\!-\!\!- 1\ \underline{M}a \Rightarrow t\ \Delta t \Rightarrow p\ \Delta t$

§10 $\underline{A}_t \oplus \underline{C}_j \circ\!\!-\!\!- 11\ \Delta t\ \underline{A}_t \oplus \underline{Q}_5 \circ\!\!-\!\!- 7 - 10$

§11 $\Delta i\ \underline{A}p \Rightarrow \underline{G}_i - 7$

§12 $\Delta h\ \Delta i\ \Delta\ell \oplus B_3 \circ\!\!-\!\!- 16\ \underline{D}_\ell \Rightarrow \underline{G}_i\ i \Rightarrow \underline{H}_h\ \bar{o}a$

§13 $\Delta a \oplus B_5 \circ\!\!-\!\!- 12\ \underline{F}a \Rightarrow t\ \underline{A}_t \oplus \underline{Q}_4 \vdash\!\!- 14$

§14 $\Delta t\ \underline{A}_t \wedge \underline{Q}_{11} \Rightarrow \underline{A}_t\ \underline{A}_t \oplus \underline{Q}_4 \vdash\!\!- 14$

§15 $\Delta t\ \underline{A}_t \wedge \underline{Q}_{11} \Rightarrow \underline{A}_t\ \underline{A}_t \oplus \underline{Q}_3 \circ\!\!-\!\!- 13\ \Delta i\ \underline{A}_t \Rightarrow \underline{G}_i - 15$

§16 $i \Rightarrow \underline{H}_h\ \Delta i \Rightarrow B_6\ \Delta h \Rightarrow B_7$.

5. Для программы *ликвид* исходной информацией служат комплексы $\underline{G}$ и $\underline{H}$. Программа строит комплексы $\underline{R}$ и $\underline{U}$, имеющие то же назначение, что $\underline{G}$ и $\underline{H}$, с той лишь разницей, что в $\underline{R}$ внутри каждого списка (строки матрицы M) ликвидированы повторные упоминания одних и тех же элементов. Для этого внутри каждого списка (имя строки не трогается) организуется два цикла по всей его длине – внешний и внутренний, завершающийся полностью на каждом шаге внешнего. Тем самым каждый элемент списка сравнивается со всеми, которые правей его, и если эти элементы не совпадают с проверяемым, то идет запись проверяемого в комплекс $\underline{R}$.

6. Программа *реализация* работает только с МОЗУ. Второй блок работает с внешней памятью. Поэтому программа *реализация* дополняется, чтобы обеспечить работу второго блока с внешней памятью.

Предполагается, что исходный массив $\tilde{A}$ разбит на зоны, не превышающие по мощности B_0. Массив $\tilde{B}$ разбит на зоны так, что в i-й зоне массива $\tilde{B}$ собраны адреса из i-й зоны массива $\tilde{A}$. Каждая пара зон $\tilde{A}_i$ и $\tilde{B}_i$ обрабатывается независимо программой *реализация* и дает зону $\tilde{P}_i$ массива $\tilde{P}$ на ленте (барабане).

7. Логическая схема алгоритма обмена с внешней памятью.

Блок обмена с внешней памятью работает следующим образом. С ленты вызываются массивы $\tilde{P}_i$ и $\tilde{P}_{i+n}$, где n меняется от $i+1$ до N (число зон), записываются в МОЗУ на места комплексов $\underline{P}$ и $\underline{R}$ соответственно. Мощность комплекса $\underline{P}$ задается вдвое больше мощности комплекса $\underline{R}$.

В комплексе $\underline{P}$ происходит объединение одноименных строк подматрицы M. Это выполняет подпрограмма A_4.

Далее, подпрограммой A_8 объединяются матрицы $\underline{P}$ и $\underline{R}$ так, что если в $\underline{P}$ и $\underline{R}$ встретятся одноименные строки, то тело строки из $\underline{R}$ (без первого элемента – имени строки) приписывается к строке в $\underline{P}$ (при этом массив в $\underline{P}$ раздвигается, чтобы вписать строку). В противном случае вся строка из $\underline{R}$ просто записывается в комплекс $\underline{C}$.

Если получаемая таким образом матрица не помещается в $\underline{P}$, излишек записывается в комплекс $\underline{C}$. В $\underline{P}$ и в $\underline{C}$ всегда помещается целое число строк. Работа продолжается до заполнения комплекса $\underline{C}$.

После заполнения комплекса $\underline{C}$ комплексы $\underline{C}$ и $\underline{P}$ записываются во внешнюю память – $\underline{P}$ как очередная зона результирующего массива $\tilde{P}$, а $\underline{C}$ как i-я зона исходного массива $\tilde{P}$.

Перед отправкой комплекса $\underline{P}$ во внешнюю память в нем внутри каждой строки ликвидируются повторения имен элементов. Это делает подпрограмма A_{13}.

Работа всей программы завершается, когда очередной комплекс $\underline{C}$ оказывается пустым.

$$U_0 Q_1(i,k,\ell-1)\uparrow^{11}_{1} F_2(i)\uparrow^{2}_{1} A^{i}_{3} A^{i}_{4} Q_5(j=i+1)\uparrow^{1}_{1}$$
$$\uparrow^{1}_{1} F_6(j)\uparrow^{2}_{1} A^{j}_{7} A^{ij}_{8} P_9(\underline{C}\ \text{полно})\uparrow^{5}_{1} L^{\kappa}_{10} F_{11}(K)\uparrow^{2}_{1} P_{12}(j=N)\uparrow^{1}_{1}$$
$$A^{\ell}_{13} L^{\ell}_{14} F_{15}(\ell) P_{16}(\underline{C}\ \text{пусто})\uparrow^{6}_{1} A_{17}\uparrow^{6}_{1} L^{\kappa}_{18} F_{19}(K)\uparrow^{1}_{1}$$

Здесь операторы Λ осуществляют вызов зон массива $\tilde{P}$ с ленты, операторы L – запись на ленту $\underline{C}$ в массив $\tilde{P}$ и запись на ленту зон результирующего массива $\tilde{P}$.

7.

Для проведения эксперимента с программой было составлено Φ-описание примера, использованного в [3]. Пример был несколько усложнен. Это описание работы процессора, выполняющего арифметические команды с фиксированной запятой (сложение и вычитание), логические команды (дизъюнкция, конъюнкция, сравнение и отрицание), команду засылки в МОЗУ, команды условной передачи управления по $\omega=1$ и $\omega=0$, команду выхода из цикла и команду останова.

Система команд примера – одноадресная, с модификацией адреса. Цикл выполнения команды состоит из вызова команды на регистр команд ($P3$), модификации адреса (на сумматоре $C1$) и исполнения (на сумматорах $C1, C2$ и схемах $C3, C4$). Совмещение алгоритмов команд в один общий делалось содержательно при составлении примера, т.е. регистры (и другие блоки) с одинаковым назначением получали в алгоритмах всех команд одни и те же имена.

Для моделирования работы с лентой все Φ – описание было разбито на четыре зоны по 127_8 кодов каждая и записаны на барабан. Для каждой зоны Φ – описания составлялась зона массива адресов начал операторов. Зоны массива адресов имели по 25_8 кодов.

Дать полное описание примера на всех этапах работы программы не представляется возможным, ввиду его громоздкости, поэтому мы ограничимся демонстрацией фрагментов.

В приложении 2 приведены фрагменты первой зоны Φ – описания и соответствующей части первой зоны массива адресов. Там же дан фрагмент первой зоны массива $\tilde{P}$, задающего матрицу связей блок-схемы. В соответствии с записью фрагмента начерчен участок общей блок-схемы.

По результирующему описанию матрицы бинарных отношений была начерчена блок-схема нашей машины – примера. Сопоставляя эту схему (приложение 3) с описанием работы примера (приложение 1), можно убедиться в их согласии. Полное время обработки примера составило около 5 минут. Число команд блока реализации алгоритма 1467_8, блока обмена 761_8.

Литература

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

2. R.Proctor. A logic design translator experiment demonstrating relationship of languages to systems and logic design.IEEE Trans.,EC-13,1964,N4.

3. И.В. Иловайский, В.С. Лозовский, Я.И. Фет. Применение адресного языка для автоматизации синтеза цифровых вычислительных устройств.- Вычислительные системы, Новосибирск,I965,Изд-во "Наука",Сиб.отд.,вып.I8.

4. Ф. Реймон. Автоматика переработки информации.М., Физматгиз,I96I.

5. Л.А. Калужнин. Об алгоритмизации математических задач.- Проблемы кибернетики, М.,Физматгиз,I959,вып.2.

6. И.В. Иловайский, Б.А. Сидристый. Алгоритмический язык для описания функционирования дискретных вычислительных устройств.-Тезисы докладов к Всесоюзному коллоквиуму по автоматизации синтеза дискретных вычислительных устройств,Новосибирск,I966.

7. А.Д. Закревский. Описание языка ЛЯПАС. В кн."Логический язык для представления алгоритмов синтеза релейных устройств." М.,Наука,I966.

8. М.Я. Товштейн. Инструкция к ПС ЛЯПАС.- Вычислительные системы,Новосибирск, Изд-во "Наука",Сиб.отд.,I967,вып.25.

Поступила в редакцию
I0.IУ.I967 г.

Φ - описание примера

```
 0: ПУСК ' P0/I,I2/= I, ' PI / I,I2/, ' PI6/I,I2/= 0,
    'M/I,24/.
 I: ПI ' PI/I,I2/ ≡  H / I,I2/.
 2: ПI0 ' M/I,24/ ≡ P3 / I,24/.
 3: П2 ' P3/I,I2/ ≡ P4 / I,I2/.
 4: П3 ' P3/I8,20/≡ P5 / I, 3/.
 5: ДI ' P3/2I,24/≡ P6 / I,I4/.
 6: Х ЛI ('P6/I,6/, 'P6/7,7/, 'P6 /I0,II/,
    'P6/I2, I2/, ' P6/I3,I3/) (7,35, 52, 60, 74).
 7: CI ('PI/I,I2/, 'P0 / I,I2/) ≡ PI7/I,I2/.
I0: П4 ' PI7/I,I2/ ≡  PI/I,I2/.
II: Х Л2 ('P5/I,3/)(2I,I2).
I2: П5 ' PI6/I,I2/ ≡  P2/I,I2/.
I3: П6 'P5 / I, 3/ ≡  P2/I, 3/.
I4: П7 ' P2 / I,I2/ ≡  H / I,I2/.
I5: ПI0 ' M/I,24/ ≡ P3 /I,24/.
I6: CI ('P3/I3,24/, ' P4/I,I2/) ≡ P20/I,I2/.
I7: ПII ' P20/I,I2/ ≡ P4/I.I2/.
20: ПI2  P4/I,I2/ = P2 /I,I2/.
2I: П7  P2/I.I2/ = H /I.I2/.
22: ПI0 'M/I,24/ ≡  P3/I,24/.
23: ПI3 'P3/I,24/ ≡ P7/I,24/.
24: CI ('P22/I,24/, 'P7/I,24/) ≡ P2I/I,24/.
25: C2 ('P22/I,24/, 'P7/I,24/) ≡ P2I/I,24/.
26: C3 ('P22/I,24/, 'P7/I,24/, 'P6/3,5/) ≡
      P2I/I,24/.
27: C3 ('P22/I,24/, 'P7/I,24/, 'P6/3,5/) ≡
      P2I/I,24/.
30: C4 'P22/I,24/ ≡ P2I/I,24/.
3I: C3  ('P22/I,24/, 'P7/I,24/, 'P6/3,5I/) ≡
      P2I/I,24/.
32: ПI4 ' P2I/I,24/ ≡ P22/I, 24/.
33: Д2 ' P22/I.24/ ≡ P23 / I, I/.
34: I : .
35: CI ('PI/I,I2/, 'P0 / I,I2/) ≡ PI7/I,I2/.
36: П4 ' PI7/I,I2/ ≡ PI/I,I2 /.
37: Х Л2 ('P5/I,3I/)(40,46).
40: П5 ' PI6/I,I2/≡ P2/I,I2/.
4I: П6 ' P5 /I, 3/≡ P2/I, 3/.
```

```
42: П7 ' P2/I,I2/ ≡ H/I,I2/.
43: ПIO ' M/I,24/ ≡ P3/I,24/.
44: CI ('P3/I3,24/, 'P4/I,I2/) ≡ P20/I,I2/.
45: ПII ' P20/I,I2/ ≡ P4/I,I2/
46: ПI2 ' P4/I, I2/≡ P2/I,I2/.
47: П7 ' P2/I,I2/ ≡ H/I,I2/.
50: ПI5 ' P22/I,24/ ≡  M/I,24/.
5I: Г : .
52: Д3 ' P23/I,I/ ≡  PI3/I,2/.
53: Д4 ' P23/I,I/ ≡  PI4/I,2/.
54: ПI6 ' P4/I,I2/≡  PI /I,I2/.
55: CI ('PI/I,I2/, 'P0/I,I2/) ≡ PI7/I,I2/.
56: П4 'PI7/I,I2/≡  PI/I,I2/.
57: I : .
60: П5 ' PI6/I,I2/ ≡ P2/I,I2/.
6I: П6 ' P5/I,3/ ≡ P2/I,3/.
62: П7 'P2/I,I2/≡ H/I,I2/.
63: ПIO ' M/I,24/≡ P3/I,24/.
64: C2 ('P3/I,I2/, 'P0/I,I2/) ≡ PI0/I,I2/.
65: П7 ' P2/I,I2/≡ H/I,I2/.
66: ПI7 ' PI0/I,I2/ ≡ M/I,I2/.
67: Д5 ' PI0/I,I2/ ≡ PI5/I,I2/.
70: CI  ('PI/I,I2/, 'P0/I,I2/) ≡ PI7/I,I2/.
7I: П4 'PI7/I,I2/ ≡ PI/I,I2/.
72: ПI6 'P4/I,I2/ ≡ PI/I,I2/.
73:  I: .
74:  П20'PI6/I,I2/ ≡ PI/I,I2/.
75:  ? M/I,24/, PI/I,I2/.
```

I. Оператор заголовка "ПУСК" - это список входных переменных алгоритма. Оператор конца /метка 75/ - список выходных переменных.

2. Операторы I - 6 - вызов команды, запись ее на регистр команд и расшифровка кода операции.

3. Операторы 7 - 34 - алгоритм исполнения арифметических команд.

Операторы 7 - IO - увеличение счетчика команд на единицу, II - 20 - модификация адреса в команде, и 2I - 34 - исполнение команды.

4. Операторы 35 - 5I описывают алгорити исполнения команды засылки в МОЗУ.

Операторы 35 - 36 описывают увеличение счетчика команд, 4[illegible] - 46 - модификацию, 47 - 5I - исполнение.

5. Операторы 52 - 57 описывают алгоритм исполнения команд перехода по условиям. Модификации адреса нет.

6. Операторы 60 - 73 описывают алгоритм исполнения команды выхода из цикла.

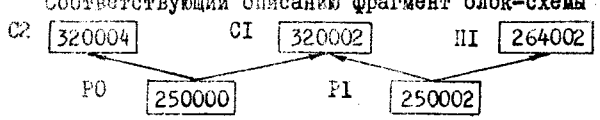
7. Операторы 74 - 75 - команда останова.

Приложение 2

Фрагмент Φ -описания и соответствующий ему фрагмент массива адресов начал операторов (машинное представление)

30	000	0034	0000	0000			
1	000	0264	0060	0000	П3		2
2	000	0250	006I	I024	Р3/I8,20/		3
3	000	0040	0000	0000	≡	4-й оператор	4
4	000	0250	0I20	0403	Р5/I,3/		5
5	000	0034	0000	0000	.		6
6	000	03I0	0020	0000	ДI		7
7	000	0250	006I	2430	Р3/2I,24/	5-й оператор	8
40	000	0040	0000	0000	≡		9
1	000	0250	0I40	04I6	Р6/I,I4/		10
2	000	0034	0000	0000	.		11
3							12
4	000	0000	0000	003I			
5	000	0000	0000	0036			2
6							3
7	**Фрагмент результирующего описания блок-схемы.**						4
	000	0250	0000	0000			5
1	000	0320	0040	0000			6
2	000	0320	0020	0000			7
3	000	0377	7777	7777			8
4	000	0250	0020	0000			9
5	000	0264	0020	0000			10
6	000	0320	0020	0000			11
7	000	0377	7777	7777			2

Соответствующий описанию фрагмент блок-схемы



Приложение 3. Блок-схема процессора.