ВОПРОСЫ ТЕОРИИ И ПОСТРОЕНИЯ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ
(Вычислительные системы)
1977 год Выпуск 70

УДК 681.323:681.3.01

ОДНОРОДНЫЙ ПРОЦЕССОР ДЛЯ ОБРАБОТКИ СИГНАЛОВ

С.Н.Сергеев

Рассматриваются вопросы построения однородного специализированного процессора для цифровой обработки сигналов на основе алгоритмов обобщенных подстановок [1,2]. Особенностью таких алгоритмов является параллельный способ переработки информации.

Известно, что проблема повышения быстродействия особенно остро стоит для вычислительных систем, работающих в реальном масштабе времени. Для названного режима были предложены достаточно интересные проекты машин с параллельной организацией. Как правило, это процессоры с логикой в памяти. Машины с однородной структурой, реализующие алгоритмы обобщенных подстановок, также ориентированы на работу в реальном масштабе времени. Операция подстановки, выбранная в качестве элементарного преобразования информации, с одной стороны, не требует большой логической сложности элементарного модуля, на базе которого строится вычислительное устройство, и, с другой стороны, обладает достаточной алгоритмической содержательностью, не приводящей к очень громоздким микропрограммам.

Каждый алгоритм обобщенных подстановок задается конечной системой правил, называемых подстановками. Они имеют вид: $S_1 * S_2 \to S_3$, где S_j – конфигурация – функция от натуральных аргументов $\langle i_1,\dots,i_n\rangle$, $S_1 * S_2$ – левая часть, S_2 – контекст, S_3 – правая часть подстановки. В данной статье используются конфигурации, параметрическая запись которых имеет вид:

$$S=\{(\alpha_1,(i,j)),(\alpha_2,(i+1,j)),(\alpha_3(i,j+1)),(\alpha_4(i-1,j)),(\alpha_5(i,j-1))\},$$

где α_i – символы некоторого алфавита, $i,j=\{1,2,\dots\}$, т.е.

в качестве клеточного множества [1] рассматривается двумерный массив клеток. Соседями клетки (i,j) являются клетки с координатами $(i+1,j)$, $(i,j+1)$, $(i-1,j)$, $(i,j-1)$. Состояние клетки (i,j) и ее соседей образуют конфигурацию.

В [1] рассмотрены вопросы построения алгоритмов обобщенных подстановок и для достаточно широкого класса алгоритмов найден метод перехода от алгоритма к сети автоматов, адекватно выполняющей этот алгоритм. Показано, что любой алгоритм обобщенных подстановок может быть реализован на однородной машине, состоящей из двух однородных частей: устройства управления (память программ) и памяти с логикой (память данных). Выполняемый машиной алгоритм хранится в устройстве управления. В машине каждой подстановке сопоставляется команда. Например, для подстановки

$$\{(\alpha_1,(i,j)),(\alpha_2,(i+1,j))\} \to \{(b,(i-1,j))\}$$

левая и правая части команды имеют вид:

I	2	3	4	5	I	2	3	4	5	
α_1	α_2	$\otimes$	$\otimes$	$\otimes$	$\otimes$	$\otimes$	b	$\otimes$	$\otimes$	(I)
i,j	$i+1,j$	$i-1,j$	$i,j+1$	$i,j-1$	i,j	$i+1,j$	$i-1,j$	$i,j+1$	$i,j-1$	

Характерной чертой этих однородных машин является параллельное преобразование информации в памяти данных под воздействием команд подстановок из памяти программ. На каждом такте работы машины одна или несколько выдаваемых из памяти программ команд преобразует информацию во всех местах исходного массива, где это возможно. При этом основной операцией, выполняемой в памяти данных, является операция подстановки. Она состоит в том, что в исходной информации код, совпадающий с кодом левой части команды, заменяется кодом правой части, указанным в этой же команде. Основные функции, выполняемые элементарной ячейкой памяти данных, состоят в а) сравнении кодов по двум группам входов; б) изменении состояния собственной памяти и памяти соседних ячеек в соответствии с кодом, поступившим на третью группу входов. Подробно выполнение подстановки ячейкой рассмотрено в [1].

Структура машин инвариантна по отношению к классу задач, а построение вычислительного устройства, ориентированного на какой-либо класс задач, фактически сводится к обеспечению этой машины нужным набором микропрограмм, а также соединением необходимого чи-

сла элементарных блоков памяти данных и памяти программ в соответствии с требованиями задач.

Функциональная блок-схема однородного процессора для обработки сигналов. Большинство задач цифровой обработки сигналов обладают одним общим свойством, которое заключается в следующем. Задано некоторое множество точек $1, 2, \ldots, N$. Каждой из этих точек сопоставлен набор значений входных величин. Требуется для каждой точки провести некоторое вычисление над соответствующим ей множеством входных величин, причем для всех точек вычисление производится по одному и тому же алгоритму, кроме того, с некоторым интервалом T множество наборов значений входных величин для всех точек изменяется. Поэтому в каждой точке к началу следующего цикла счет для предыдущих входных значений должен быть закончен.

Необходимость счета в реальное время накладывает жесткие требования на производительность вычислительного устройства. Так, например, простой подсчет показывает, что для решений одной задачи идентификации объектов по последовательной схеме (на машине с одним процессором) требуется, чтобы рабочий такт процессора был не более 0,5 нсек [3], причем это время получено без учета времени на загрузку процессора (обращение к памяти и другие обслуживающие операции).

Из этого факта следует необходимость такой организации вычислительного процесса, которая позволяла бы максимально учитывать параллельность, допускаемую задачей. Отметим, что алгоритмы цифровой обработки сигналов наилучшим образом приспособлены для организации параллельных вычислений, так как вычисления всегда проводятся для многих точек по одной и той же схеме, и ход вычислительного процесса в каждой точке не зависит от состояния вычислительного процесса в других точках.

С учетом сказанного структура однородного процессора имеет вид, представленный на рис.I. Блок I представляет собой однородный процессор, состоящий из одинаковых вычислительных модулей M. Каждый модуль M может выполнять функции хранения чисел, а также выполнять любую программу, которая поступает из блоков II и III. Эти блоки набраны из управляющих модулей U. Блоки II и III осуществляют раздачу исходных данных и команд модулям M. Управление блоками II и III осуществляется устройством управления, для которого они

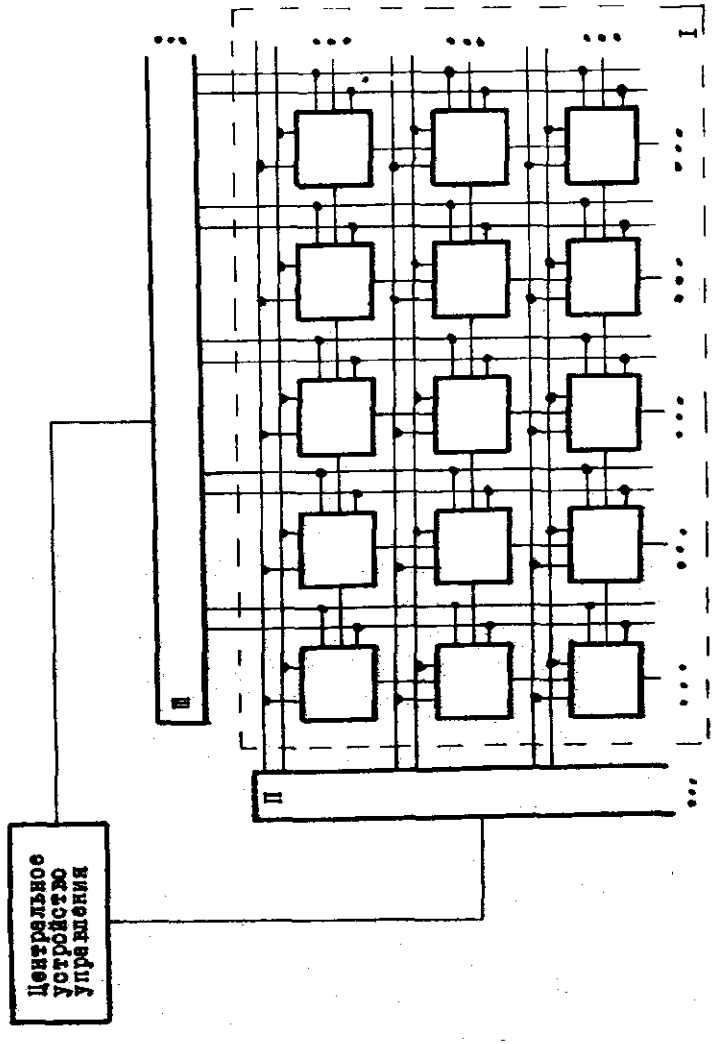


Рис. I

в свою очередь являются памятью данных. С помощью центрального устройства можно настроить тот или иной участок блоков II и III на выдачу какой-либо конкретной программы и числе модулям M, связанным с этим участком. Обмен информацией между модулями может осуществляться как непосредственно, так и через управляющие линейки. Размер блока I определяется числом точек, для которых проводится вычисление.

Блок-схема вычислительного модуля M. Вычислительный модуль M имеет структуру, изображенную на рис.2, и представляет собою совокупность одинаковых ячеек. У ячейки есть шесть входов, четыре из них от соседних ячеек. Эти соседние ячейки в соответствии с номерами входов называются первым соседом, вторым и т.д. Для удобства дальнейшего описания введем систему координат, сопоставив каждой ячейке тройку чисел: x_1, x_2, x_3 $(1 \le x_1 \le 2n_1,\ 1 \le x_2 \le n_2,\ 1 \le x_3 \le n_3)$.

Группу ячеек с фиксированными второй и третьей координатами $(x_2 = c_2,\ x_3 = c_3)$: $\{(1, c_2, c_3), (2, c_2, c_3), \ldots, (n_1, c_2, c_3)\}$ будем называть строкой (c_2, c_3). Группу ячеек с фиксированной третьей координатой $(x_3 = c_3)$ будем называть слоем c_3 модуля M.

Каждый модуль M имеет четыре n_1-разрядные группы входов для записи числовой информации. Первую группу входов образуют входы I (рис.2) ячеек $\{(I,I,I), (2,I,I),\ldots,(n_1,I,I)\}$, вторую группу входов образуют входы I ячеек $\{(n_1+I,I,I), (n_1+2,I,I),\ldots,(2n_1,I,I)\}$, третью группу входов образуют входы 3 ячеек $\{(I,n_2,I),(2,n_2,I),\ldots,(n_1,n_2,I)\}$, четвертую группу входов образуют входы 3 ячеек $\{(n_1+1,n_2,I),\ldots,(2n_1,n_2,I)\}$. Кроме того, каждый модуль имеет две группы управляющих входов, о назначении которых будет сказано при рассмотрении примера.

Каждая группа содержит n_2 входов по n_3 разрядов, причем i-й разряд j-го входа первой группы входов соединен со входами 6 ячеек с координатами $\{(I,j,i),(2,j,i,),\ldots,(n_1,j,i)\}$; i-й разряд j-го входа второй группы входов соединен с входами 6 ячеек с координатами $\{(n_1+I,j,i),(n_1+2,j,i),\ldots,(2n_1,j,i)\}$.

Кроме перечисленных групп входов, каждый модуль имеет две группы входов для подачи левой и правой частей команды подстановки, которые соединены со всеми ячейками модуля. На рис.2, во избежание его усложнения, эти связи не показаны.

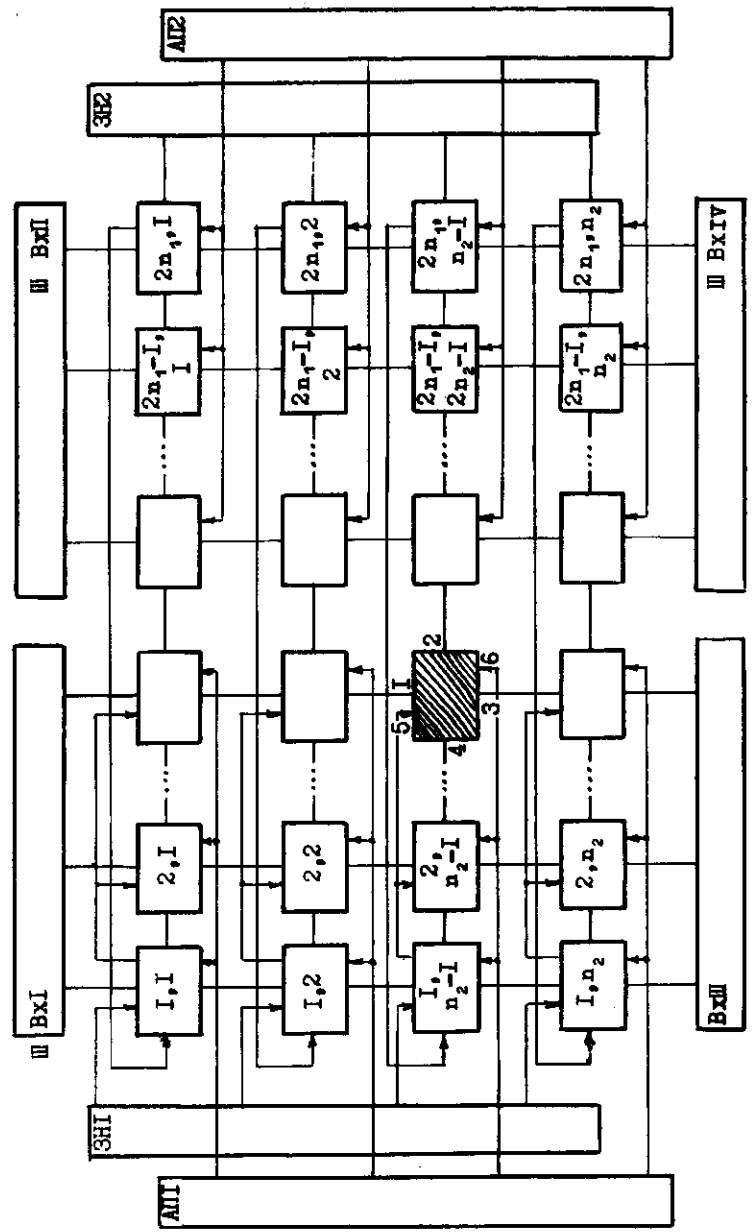


Рис. 2

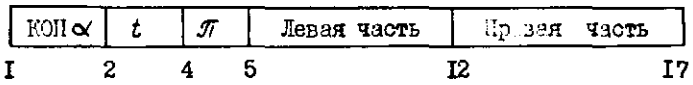
Величины n_1, n_2, n_3 выбираются исходя из требований задачи.

Функции модуля U. Перечислим основные действия, выполняемые модулем.

А) Выдача числовой информации в вычислительные модули. Числа, поступающие извне, записываются в ячейки памяти тех модулей, ассоциативные признаки которых совпадают с ассоциативными признаками, указанными в числах. Затем эти числа подаются на одну из четырех групп числовых входов модулей M, связанных с этим управляющим модулем. Аналогичным образом по ассоциативному признаку этот модуль может принять число из какого-либо модуля M, с которым он связан.

Б) Управление процессом вычислений в модулях M. Любая программа, реализуемая на машинах рассматриваемого класса, всегда представляется как совокупность микропрограмм. Все микропрограммы хранятся в памяти центрального устройства управления. Команды всех микропрограмм циклически выдаются одна за другой каждая на свою группу шин. Каждый управляющий модуль в зависимости от хода вычислительного процесса подает ту или иную микропрограмму в связанную с ним группу модулей.

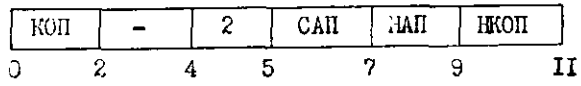
Форматы и типы команд. Микропрограммы строятся из команд трех типов. Основным типом команд является команда подстановки, выдаваемая на исполнение в модули M. Формат команды имеет вид:



где КОП – код операции, в котором первый символ является именем микропрограммы, к которой относится данная команда; второй – служит для записи рабочего признака; команда, имеющая $\alpha = 1$, является начальной командой микропрограммы, все остальные команды данной микропрограммы имеют признак $\alpha = 0$. Если $\pi = 0$, то команда подстановки называется командой подстановки без обратной связи; если $\pi = 1$, то команда подстановки называется командой с заданным числом повторений t.

Левая часть подстановки записывается с 6-го по 12-й символ формата команды. Причем первым указывается символ, который должен сравниваться по первому входу ячеек модулей M, вторым – символ, который должен сравниваться с символом по второму входу ячеек модулей M и т.д. Шестым является символ, который является ассоциативным признаком, включающим в работу ту или иную линию ячеек. Он сравнивается с символом по шестому входу ячеек модулей M. Седьмым указывается символ, который должен сравниваться с собственным состоянием. Правая часть записывается в таком же порядке с 13-го по 17-й разряды.

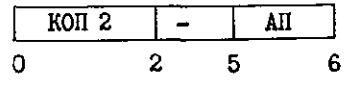
Второй тип команды – команда модификации ассоциативных признаков и кодов операции. Формат этой команды имеет вид:



где КОП – код операции, в котором первый символ является именем микропрограммы, к которой относится данная команда; САП – старый ассоциативный признак, который заменяется на НАП – новый ассоциативный признак; НКОП – имя микропрограммы, вызываемой данной командой.

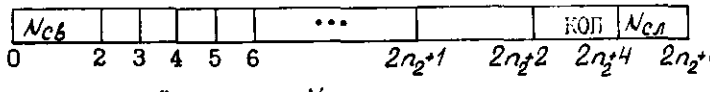
Назначение этой команды, так же как и команды третьего типа, состоит в обеспечении взаимодействия микропрограмм друг с другом, а также с основной программой.

Третий тип команды – команда безусловного вызова очередной команды основной программы. Формат команды:



где КОП – код операции, в которой первый символ является именем микропрограммы, к которой относится данная команда; АП – ассоциативный признак.

Как уже отмечалось, все описанные команды предназначены для построения микропрограмм. Наряду с этим есть команда, которая служит для записи собственно алгоритма решения задачи, оперирует именами микропрограмм как операндами и указывает те ячейки модулей M, над которыми должна быть произведена операция. Формат этой команды:



где $N_{сб}$ – номер данной команды, $N_{сл}$ – номер команды, которая должна выполняться после завершения данной команды.

Разряды (3,4), (5,6),...,$(2n_2+1, 2n_2+2)$ служат для записи ассоциативных признаков, выделяющих соответственно первую строку той группы модулей M, которая соединена с данным управляющим устройством U, вторую строку группы модулей и т.д., n_2-ю строку группы модулей.

Реализация команд микропрограмм и взаимодействие микропрограмм в управляющем модуле происходит следующим образом. Отмечалось, что микропрограммы и основная программа находятся в памяти программ центрального устройства управления и поступают по соответствующим шинам на все управляющие модули. Каждый цикл работы управляющего модуля начинается с некоторой начальной команды основной программы, которая непосредственно записывается в регистр команд модуля U. По имени микропрограммы, записанному в КОП этой команды, дешифратор осуществляет подключение к данному управляющему модулю того блока памяти программ центрального управляющего устройства, в котором хранится указанная микропрограмма. Первой из списка команд микропрограммы дешифратором выбирается команда, имеющая признак $\alpha = 1$. Следующей выполняется команда, записанная в блоке сразу же за этой командой, и т.д.

Если очередная команда имеет признак $\pi = 0$, то она выдается в модули M на исполнение, не вызывая никаких переключений в управляющем модуле.

Если же очередная команда, выбранная из блока, имеет признак $\pi = 1$, то она выдается в модули M и, кроме того, она взводит триггер ожидания в модуле U. Сигнал обратной связи вырабатывается ячейками модулей M в том случае, если ни в одной ячейке модулей M выдаваемая команда неприменима. Это служит сигналом, что данная микропрограмма (либо ее часть) выполнена и дешифратор выбирает из блока микропрограмм очередную команду только с признаком $\pi = 2$.

Если очередная выполняемая команда имеет признак $\pi = 2$, то эта команда не выдается на модули M, а производит замену содержимого разрядов $2-(2n_2+2)$ в соответствии с информацией, записанной в разрядах 5–11 этой команды. Следующей за ней командой может быть либо команда с признаком $\pi = 2$, либо команда с $\alpha = 1$.

Если очередная команда, выбранная дешифратором, имеет признак $\alpha = 2$ и ее ассоциативный признак совпадает с признаком, записанным в одном из разрядов $2-(2n_2+2)$, то на регистр команд вызывается команда основной программы, имеющая номер $N_{сл}$, и т.д.

Пример организации вычислительного процесса. Для иллюстрации основных способов организации вычислений приведем пример программной реализации цифрового фильтра по следующему алгоритму [4]. Для каждой из $N = N_1 \times N_2$ точек нужно производить вычисления итеративных сумм $u_k^{i,j}$ и $v_k^{i,j}$ по формулам:

$$\left.\begin{aligned} u_k^{i,j} &= u_{k-1}^{i,j} + a_{jk} X_i - b_{jk} Y_i, \\ v_k^{i,j} &= v_{k-1}^{i,j} + b_{jk} X_i + a_{jk} Y_i, \end{aligned}\right\} \qquad (2)$$

$$i \in N_1, \quad j \in N_2,$$

до тех пор, пока выполняется неравенство $(u_k^{i,j})^2 + (v_k^{i,j})^2 - h^2 < 0$. Величины x_i, y_i, a_{jk}, b_{jk}, h для данного алгоритма являются входными. Каждой точке (i,j) сопоставлен свой модуль M, т.е. каждый модуль M выполняет цифровую фильтрацию сигнала, соответствующего точке (i,j).

В приложении приведен текст основной программы вычислений по формулам (2). В примечаниях указано назначение команд.

Набор микропрограмм позволяет производить на одном модуле либо на их группе все основные арифметические операции с фиксированной запятой с разрядностью, обусловливаемой требованиями задачи. С помощью данного набора микропрограмм можно организовать любой алгоритм обработки сигналов (вычисление свертки, преобразование Фурье и др.) на одном и том же оборудовании; причем изменение параметров задачи приводит только к изменению числа модулей M и U.

В заключение остановимся на некоторых вопросах реализации процессора. Выше было показано, как, используя алгоритмы подстановок в качестве средства программирования на микро- и макроуровнях, удается перейти к структуре машины, адекватной решаемой задаче. Таким образом, алгоритмы обобщенных подстановок могут послужить алгоритмической основой для разработки ряда унифицированных процессоров с микропрограммным управлением, ориентированных на тот или иной класс задач. Основной особенностью, затрудняющей интегральное исполнение отдельной ячейки памяти данных или модуля M, является большое количество внешних связей. Однако можно заметить, что в микропрограммах "коэффициент использования окрестности" в среднем составляет 50%. Поэтому если перейти к последовательной

передаче отдельной микрокоманды, то рабочий такт ячейки удлинится в среднем в 4 раза, а количество внешних связей сократится в 7 раз. В результате такой модификации мы получаем модуль, имеющий около 40 внешних выводов, что уже приемлемо для технологии.

Итак, рассмотрены вопросы реализации однородного вычислителя на основе алгоритмов обобщенных подстановок. Использование алгоритмов обобщенных подстановок как универсального средства параллельного микропрограммирования однородных структур типа с логикой в памяти дает возможность свести задачу проектирования вычислительного устройства под заданный алгоритм к построению некоторого набора параллельных микропрограмм в терминах локальных преобразований исходной информации.

Литература

1. КОРНЕВ Ю.Н., ПИСКУНОВ С.В., СЕРГЕЕВ С.Н. Алгоритмы обобщенных подстановок и вопросы их интерпретации сетями автоматов и однородными машинами.-"Изв.АН СССР. Техническая кибернетика",1971, № 6, с. 131-142.

2. КОРНЕВ Ю.Н., ПИСКУНОВ С.В., СЕРГЕЕВ С.Н. Микропрограммный однородный процессор,ориентированный на задачи линейной алгебры. -В кн.: Оптимизация. № 6(23), Новосибирск, 1972.

3. ХВОСТАНЦЕВ М.А. Микропроцессоры и системы обработки данных.-"Зарубежная радиоэлектроника", 1975, № 9, с. 31-56.

4. ГОЛД Б., РЭЙДЕР Ч. Цифровая обработка сигналов. М., "Сов. радио", 1973, с.14-15.

Поступила в ред.-изд.отд.
2 февраля 1977 года

ОСНОВНАЯ ПРОГРАММА

$N_{св}$	Ассоциативные признаки								КОП	$N_{сл}$	
1 2	3	4	5	6	7	8	9	10	11 12	13 14	
0	0 0 1	1 0 0	0 1 0	0 1 1	0 1 0	1 0 0	0 0 1	0 1 1	$П_1$	1	Начало цикла вычислений ввод a_j, b_j, x_i, y_i
1	0 0 1		0 0 0		1 1 0		0 0 0		C_1	2	Перевод в дополнительный код и другие вспомогательные операции
2	1 0 0		0 0 0		1 0 0		0 0 0		C_2	3	$Z_1 =: X_i a_{jk} - b_{jk} Y_i$ $Z_2 =: X_i b_{jk} + a_{jk} Y_i$
3	1 0 0		0 0 0		1 0 0		0 0 0		C	4	Вычисление $U_{k-1} + Z_1$ $V_{k-1} + Z_2$
4	0 0 1	0 0 1	0 0 1	0 0 1	0 0 0	0 0 0	0 0 0	0 0 0	$П_2$	5	Вычисление: U_k^2, V_k^2
5	1 1 0	1 1 0	1 1 0	1 1 0					S	6	$Z := U_k^2 + V_k^2$
6	0 0 0	1 1 1	0 0 0	1 0 1	0 0 0	0 0 0	0 0 0	0 0 0	C	7	$Z - h^2$
7	0 0 0	1 0 0	0 0 0	0 0 1	0 0 0	1 0 0	0 0 0	1 0 0	$П_5$		Вывод результата сравнения