ВОПРОСЫ ТЕОРИИ ПОСТРОЕНИЯ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ
1978 год (Вычислительные системы) Выпуск 73

УДК 681.325.5

ГРАФ-ПРОГРАММНОЕ УПРАВЛЕНИЕ НА ОСНОВЕ АССОЦИАТИВНОЙ ВЫБОРКИ ИНСТРУКЦИЙ И ДАННЫХ

Т.Ц.Кънчев, К.Л.Боянов

Одной из основных тенденций, определяющих структуру современных систем для обработки данных, является стремление к максимальному совмещению выполнения отдельных частей процесса обработки. Несмотря на большие достижения в этом направлении, управление вычислительным процессом в обрабатывающем устройстве современных ЭВМ принципиально не отличается от управления первых машин, в которых команды выполнялись последовательно одна за другой. Конвейерная обработка (PIPELINING), предварительный анализ потока команд (LOOK-AHEAD) (см.[2]) и др.режимы, позволяющие реализацию параллелизма в вычислительном процессе, являются только развитием классической модели Неймана, при котором сохраняются принципы концептуальной последовательности выполнения команд [1].

Существенным недостатком быстродействующих систем указанных типов является разница между концептуальной и действительной последовательностями выполнения команд. Это значительно усложняет устройство управления, так как оно должно иметь дополнительные средства для обнаружения и реализации локального параллелизма в потоке команд. Кроме того, степень параллельности относительно низка и в целом еще не соответствует возможностям выбранного математического метода решения задачи. Она сильно зависит от способа программирования из-за наличия "скрытого параллелизма" при хранении промежуточных результатов [3].

Применение разных вариантов мультипроцессорной обработки [3-5] позволяет избежать только часть из указанных недостатков, поскольку распараллеливание проводится на уровне участков программы (макрораспараллеливание). Так как число процессоров обыкновенно на

много ниже возможной степени параллельности задач, такой подход ставит большое количество операционных проблем (например, проблему диспетчеризации), которые в принципе не решены [7]. Кроме того, непроизводительные затраты времени на управление процессом растут при сокращении длины и увеличении числа параллельных программных участков, что является определенным препятствием для достижения большой степени параллельности. Мультипроцессорная обработка решает все эти вопросы с макроструктурной точки зрения, не затрагивая вопросов организации вычислительного процесса в управляющем устройстве отдельных процессоров. Таким образом, выполнение отдельных частей программы остается обычно в пределах традиционной концептуальной последовательности.

П о с т а н о в к а п р о б л е м ы. Перспективной, с точки зрения современной теории параллельной обработки [7-8], является такая организация центрального процессора, которая позволяет:

1. сохранять естественную асинхронность задачи при ее описании (программировании), что дает системе возможность самостоятельно решать вопросы распараллеливания в зависимости от наличной вычислительной мощности. Это приводит к отказу от алгоритмичности машинного языка;

2. плавно наращивать вычислительную мощность простым прибавлением по желанию потребителя дополнительных модулей (модульность структуры);

3. организовать обработку данных в условиях неограниченной вычислительной мощности, тем самым достичь максимального параллелизма, заданного в программе (глобальный параллелизм на уровне инструкций).

Удовлетворить первое требование можно при помощи представления объектной программы в виде графа, задающего явно или косвенно допустимую (а не обязательную) последовательность выполнения команд. Такой принцип описывается в [9].

В настоящей статье для реализации граф-программного управления предлагается использование ассоциативной памяти.

В первой части работы рассматриваются принципы граф-программного управления. Более подробно вопросы асинхронного программирования описаны в [6-8]. Аналогом спусковых функции, введенных в [6], здесь являются предикаты выполнимости операторов. Выполнимость оператора связана с окончанием вычислений всех промежуточ-

ных результатов, входящих в набор операндов. Преимуществом этого подхода является упрощение алгоритма работы управляющего устройства и связанной с ним технической реализации.

Во второй части предлагается общая структурная схема процессора, имеющая вышеуказанные свойства. Для реализации граф-программного управления используется ассоциативная выборка инструкций и промежуточных результатов.

Принципы граф-программной параллельной обработки. Рассмотрим с абстрактной точки зрения принцип граф-программного управления.

Пусть дан ациклический ориентированный граф $G = (W, U)$, где $W=\{1,2,\dots n\}$ интерпретируется как множество операторов для обработки данных, а $U=\{u_1,u_2,\dots u_m\}$ - как связи, определяющие непосредственную информационную зависимость между ними. Вершины в G, не имеющие входно- и выходно-инцидентных дуг, образуют соответственно подмножества W^I и W^O начальных и конечных операторов:

$$W^I = \{k \mid k \in W \text{ и } \delta^+(k) = 0\},$$

$$W^O = \{k \mid k \in W \text{ и } \delta^-(k) = 0\} .$$

Через U_k^I и U_k^O будем обозначать соответственно множества входных и выходных дуг каждого оператора $k \in W$.

Каждому оператору $k \in W$ поставлены в соответствие:

1. Операция f_k для обработки данных, являющаяся отображением $f_k: Q^l \to Q$ (Q^l - l-я декартова степень); множество всех f_k - $F = \{ f_1, f_2, \dots f_n \}$.

2. Операция анализа p_k, являющаяся отображением $p_k: Q^l \to \{0,1\}$; множество всех p_k, $P = \{p_1, p_2, \dots p_n\}$.

3. Переменная v_k, принимающая значения из области Q. Будем считать, что Q есть множество всех двоичных векторов определенной разрядности; $V = \{v_1, v_2, \dots v_n\}$ является множеством всех переменных. Через $V^I \subseteq V$ и $V^O \subseteq V$ будем обозначать соответственно образы $W^I \to V$ и $W^O \to V$, являющиеся подмножествами входных данных и конечных результатов. Переменную $v_k \notin V^I \cup V^O$ будем называть промежуточным результатом.

4. Набор операндов x_k оператора k, являющийся упорядоченной последовательностью всех элементов V_k (V_k - множество входных переменных оператора k), т.е. $x_k = (v_{s1}, v_{s2}, \dots v_{sl})$, такой что $\{v_{s1}, v_{s2}, \dots v_{sl}\} = V_k = \{v_i \mid v_i \in V;\ i,k \in W$ и $(i,k) \sim u_j \in U\}$, причем $X = \{x_1, x_2, \dots, x_n\}$, является множеством всех наборов операндов.

Каждой дуге $u_i \in U$ ставится в соответствие двоичное состояние q_{u_i}, принимающее значения (1,0] (активное, неактивное). Определим предикат E(k) для выполнимости оператора k и процедуры OUTON(k) и INOFF (k)*) соответственно для смены состояний его выходных и входных дуг следующим образом:

$$E(k) = \forall u_i((u_i \in U_k^I) \rightarrow (q_{u_i}=1)) \wedge U_k^I \neq \emptyset$$

procedure OUTON(k); begin для всех $u_i \in U_k^O$ выполнить $q_{u_i}:=1$ end

procedure INOFF(k); begin для всех $u_i \in U_k^I$ выполнить $q_{u_i}:=0$ end

Предполагая, что в начале выполнения все выходно-инцидентные дуги входных операторов находятся в активном состоянии, для каждого оператора k определим следующую процедуру управления:

```
procedure C(k)
begin  A: if ¬E(k) go to A ;
          if p_k(x_k) then begin v_k := f_k(x_k); OUTON(k) end
          INOFF(k)
end
```

Через C обозначим процесс параллельного асинхронного выполнения [10] всех процедур C(k) (k = 1,2,...,n).

Пусть посредством отображений $\phi_V: V^I \rightarrow Q$, $\phi_F: F \rightarrow \Phi$ и $\phi_P: P \rightarrow \Pi$ переменным $v_i \in V^I$ присваиваются начальные значения из области Q, а f_i и p_i определяются соответственно из множества операций обработки $\Phi = \{\varphi_1, \varphi_2, \ldots, \varphi_r\}$ и множества операций анализа $\Pi = \{\pi_1, \pi_2, \ldots, \pi_s\}$.

Тогда пятерку $H=(G, \phi_V, \phi_F, \phi_P, X)$ будем называть граф-программой параллельной обработки, тройку $M = (\Phi,\Pi,C)$ – элементарным граф-управляемым процессором, а пару Z = (H,M) – выполнением программы H на процессоре M. Программу H будем считать законченной, если в результате работы M множество выполнимых операторов станет пустым. Программу H будем считать корректной, если после конечного числа шагов выполнение C, Z является законченным, причем определено значение хотя бы одной переменной $v_k \in V^O$.

---

*) Для краткости описания процедур не следуют формальным правилам синтаксиса эталонного языка ALGOL – 60. Семантика нестандартных выражений ясна из контекста.

Граф-управляемый процессор для параллельной обработки. Реализация абстрактной модели в виде действующей системы технических средств ставит в основном следующие проблемы:

а) реализацию процедур OUTON и INOFF , обеспечивающих выполнение операторов в допустимой последовательности, заданной программным графом;

б) временное хранение вычисленных промежуточных результатов до тех пор, пока есть невыполненные операторы, использующие эти результаты в наборе операндов;

в) удаление из запоминающей среды неактуальных переменных. Переменная v_k считается неактуальной, если все выходные дуги оператора k находятся в состоянии "неактивно".

Для разрешения этих проблем предлагается использование ассоциативных памятей.

На рис.1 показана общая структура элементарного процессора. Он состоит из программной памяти (ПП), памяти данных (ПД), обрабатывающего устройства (ОУ), собирательного буфера (СБ), распределительного буфера (РБ) и управляющего устройства (УУ).

Программная память. В программной памяти хранится граф-программа параллельной обработки. В данном случае выбрана двухоперандная система инструкций, формат которых показан на рис.2,а. Каждому оператору графа соответствует одна инструкция. Отдельные поля инструкций имеют следующее назначение:

N - номер инструкции (каждая инструкция идентифицируется своим кодом (номером); нумерация инструкций программного графа произвольная, но однозначная);

I_1 - флаг первого операнда (этот двоичный индикатор находится в состоянии "I", когда инструкция, номер которой записан в поле N_1, выполнена и соответствующий промежуточный результат сформирован);

I_2 - флаг второго операнда имеет назначение, аналогичное I_1 (I_1 - начальное состояние флангов и I_2 - нулевое);

N_1 - номер первого операнда или номер инструкции, результат выполнения которой является первым операндом в настоящей инструкции;

N_2 - номер второго операнда (интерпретация та же самая, как у поля N_1);

F - код операции (в этом поле записывается код двухоперандной арифметико-логической операции или операции анализа).

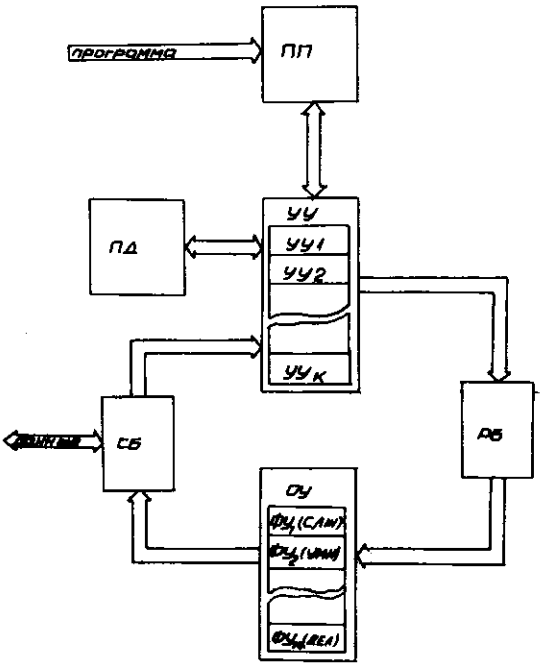

Рис.I. Структурная схема граф-управляемого процессора.

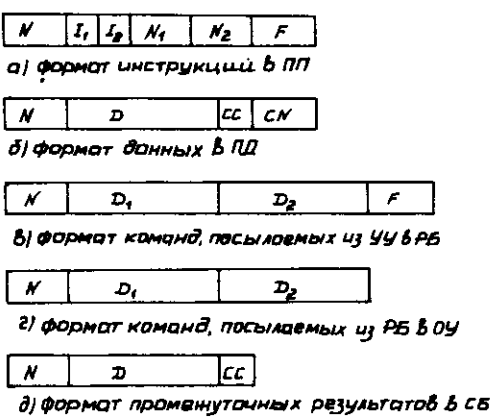

Рис.2. Формат информационного потока в процессоре.

Будем предполагать, что в программной памяти имеются средства для реализации ассоциативного поиска по содержанию полей N_1 и N_2 и для обновления содержания полей I_1 и I_2. Эту обобщенную операцию обозначим через UPFETCH(A) и будем считать, что выполняется следующим образом:

1. Устанавливаются в единицу флаги I_1 и/или I_2 всех инструкций, у которых совпадают $N_1 = A$ и/или $N_2 = A$.

2. Из области совпадения выбираются все инструкции, для которых $I_1 \wedge I_2 = 1$ (выполнимые инструкции).

3. Формируется число CNT инструкций из области совпадения, для которых $I_1 \wedge I_2 = 0$ (невыполнимые инструкции).

В памяти данных хранятся промежуточные результаты в процессе обработки. Будем считать, что каждый промежуточный результат занимает одну ячейку в памяти данных и имеет формат, показанный на рис.2,б. Отдельные поля имеют следующее назначение:

N - номер инструкции, формировавшей результат,

D - код самого операнда,

CC - код условия, являющийся предикатом D ,

CN - счетчик актуальности, указывающий число невыполнимых инструкций в программной памяти, использующих этот результат в качестве операнда.

Будем предполагать, что в памяти данных имеются средства для реализации ассоциативного поиска по содержанию поля N , для обновления поля CN и для записи новых данных в память данных. Эти действия осуществляются при помощи операций GET(A) и PUT(X) .

Операция GET(A) выполняется следующим образом:

1. Выбирается из памяти данных содержимое ячейки, для которой $N = A$.

2. Вычитается единица из содержания ее поля CN .

3. Если $CN \neq 1$, промежуточный результат с новым содержанием поля CN записывается обратно в память данных. Если $CN = 0$, выбранная ячейка освобождается.

Операция PUT(X) записывает в память данных промежуточный результат X (в указанном выше формате). Запись осуществляется в произвольную свободную ячейку.

Обрабатывающее устройство имеет организацию, подобную известным системам, составленным по принципу концептуальной последовательности выполнения команд с предварительным анализом [2]. Оно состоит из нескольких функциональных блоков, каждое из которых ориентировано на выполнение некото-

рой операции (или класса операций): сложение, вычитание, умножение, деление, логические операции и т.д. Каждый функциональный блок работает параллельно с остальными и независимо от них. Он получает команды из распределительного буфера, выполняет ее и формирует результат, который посылает в собирательный буфер. Формат команд, посылаемых из распределительного буфера в обрабатывающее устройство, показан на рис.2,г, где N - номер инструкции, из которой формирована команда, а D_1 и D_2 являются соответственно первым и вторым операндами. На рис.2,д показан формат промежуточных результатов. Содержание поля N берется из одноименного поля команды, содержания полей D и CC являются соответственно результатом и кодом условия.

С о б и р а т е л ь н ы й б у ф е р является стэком для временного хранения промежуточных результатов. В случае однопрограммного выполнения он работает в режиме "первый вошел-первый вышел". При многопрограммном выполнении порядок обслуживания зависит от приоритета программ.

Р а с п р е д е л и т е л ь н ы й б у ф е р принимает подготовленные в управляющем устройстве команды в формате, который показан на рис.2,в. Он дешифрирует поле F команд и распределяет их по функциональным устройствам. Порядок, в котором посылаются команды, зависит от приоритета программ, от занятости функциональных устройств и, наконец, от очередности их передачи в распределительный буфер.

У п р а в л я ю щ е е у с т р о й с т в о формирует параллельно выполнимые команды, которые посылает в распределительный буфер. На каждом цикле работы устройства выбираются один промежуточный результат из собирательного буфера и все выполнимые инструкции из программной памяти, для которых этот результат является операндом. На основе каждой из этих инструкций из памяти данных выбирается соответствующий приоритетный операнд и формируется команда в указанном выше формате. Блок-схема алгоритма работы устройства показана на рис.3.

Перед стартом программы в программную память заносится множество инструкций программного графа, а в собирательный буфер - начальные данные в виде сформированных в обрабатывающем устройстве промежуточных результатов. Содержание программного счетчика PCT равно числу начальных данных в собирательном буфере.

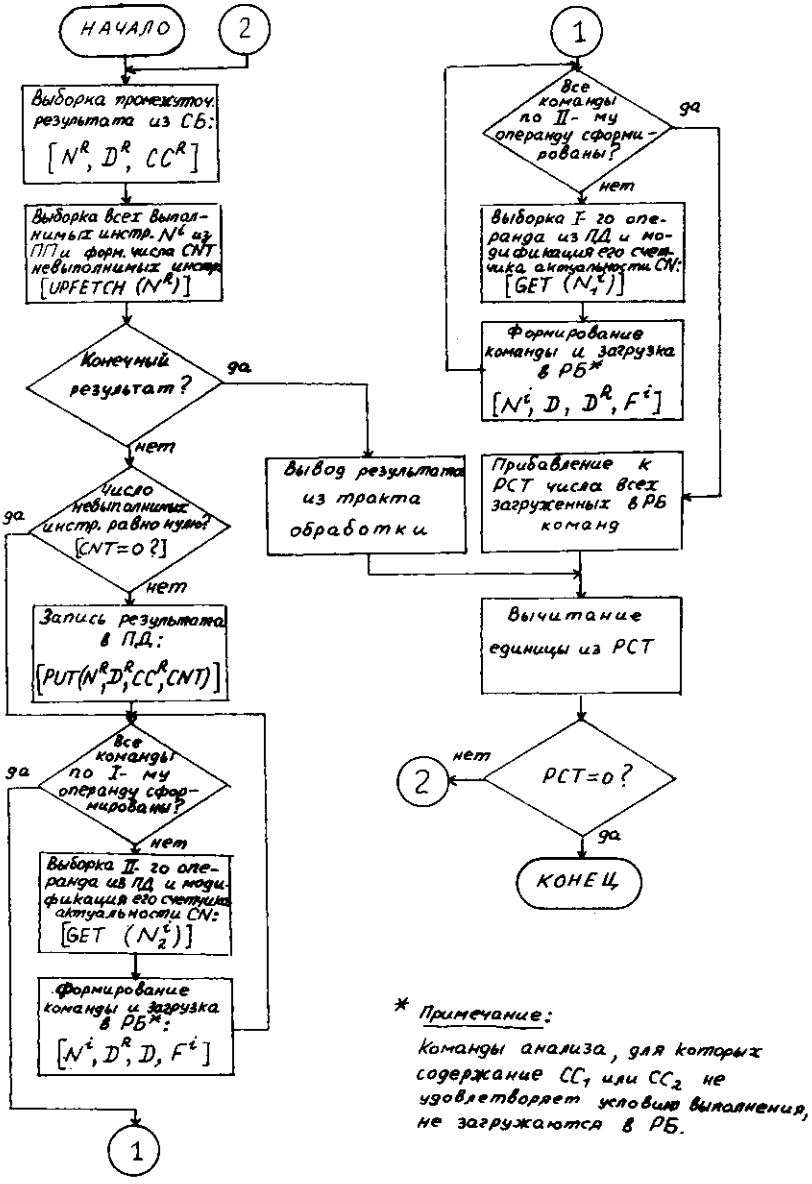

Рис.3. Алгоритм работы УУ.

В процессе обработки выбранный из собирательного буфера результат считается конечным, если область совпадения при операции UPFETCH пуста. Конечный результат выводится из тракта обработки и посылается в основную память (внешнюю для процессора).

Состояние программного счетчика поддерживается всегда равным числу команд (промежуточных результатов), находящихся в тракте "распределительный буфер–обрабатывающее устройство–собирательный буфер". Программа считается выполненной, если после обработки следующего результата программный счетчик пуст. При многопрограммной работе для каждой программы предусмотрен отдельный программный счетчик. Для ускорения процесса обработки управляющее устройство может быть реализовано в виде нескольких параллельно и асинхронно работающих устройств. Для их синхронизации может быть использован аппарат семафоров [10]. Рис.4 иллюстрирует работу процессора при вычислении выражения $y = (a,b)/(c,d) + d/(a,b)$. На рис.4,а показан граф программы. Принято, что входные данные находятся в собирательном буфере в порядке нарастания номеров начальных операторов – 1,2,3,4, так что первой выбирается из собирательного буфера входная данная с номером 1. На временной диаграмме на рис.4,в показаны последовательные стадии обработки отдельных инструкций и интервалы времени, в которых работают соответствующие функциональные блоки обрабатывающего устройства. Считается, что в набор входят два устройства умножения (УМН 1 и УМН 2), два устройства деления (ДЕЛ 1 и ДЕЛ 2) и устройство сложения (СЛЖ). Прохождение информации через распределительный буфер не рассмотрено.

П р е и м у щ е с т в а п р е д л о ж е н н о й о р г а н и з а ц и и. Несмотря на ограниченные возможности элементарного процессора с точки зрения структурной обработки данных, реализации циклов и подпрограммных обращений, можем отметить следующие существенные преимущества предложенной организации.

1. Высокий параллелизм обработки, приближающийся к возможностям выбранного математического метода решения задачи.

2. Экономичный расход памяти при хранении промежуточных результатов. Интервал времени, в котором переменная v_k занимает ячейку в ПД, равен периоду ее актуальности. Число актуальных переменных меняется динамически и зависит, главным образом, от степени параллельности программного графа. Таким образом, в отличие от систем типа КПК необходимые ресурсы памяти в каждый момент будут соответствовать действительным потребностям процесса.

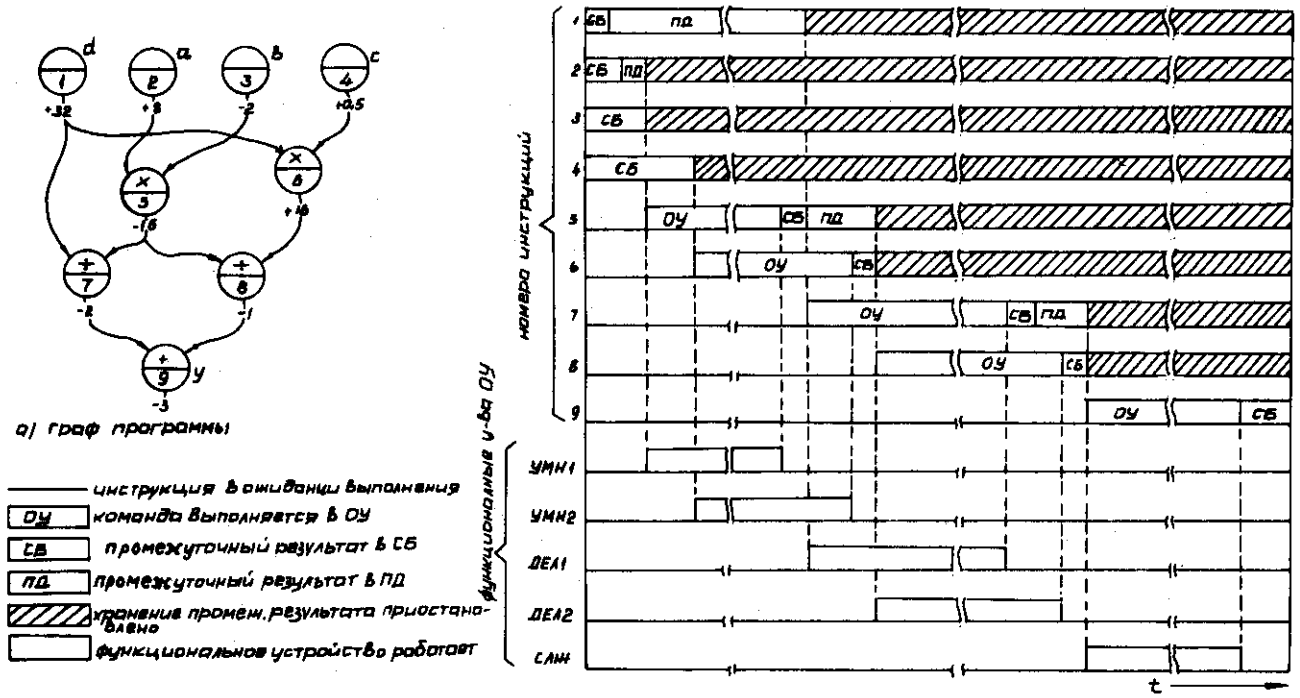

Рис.4. Выполнение программы для вычисления $y=(a.b)/(c.d)+d/(a.b)$.

3. Простая организация многопрограммной работы. Отпадает необходимость прерываний, так как при снижении графика данной программы (из-за низкой степени параллельности или других причин) система автоматически нагружается потоком более низкоприоритетных программ. Посредством незначительных аппаратурных затрат в собирательном и распределительном буферах можно организовать динамическое изменение приоритета отдельных программ в процессе обработки.

4. Более простой и естественный способ программирования для некоторых вычислительных задач.

5. Специализация отдельных функциональных блоков обрабатывающего устройства, позволяющая сделать их максимально быстродействующими. Универсальные обрабатывающие устройства, характерные для большинства систем указанного выше типа, трудно приспособить к специфике каждой операции из стандарного набора команд.

6. Возможность модульной реализации памяти данных, программной памяти, обрабатывающего устройства, собирательного и распределительного буферов, позволяющая легко изменять вычислительную мощность по желанию потребителя. Число модулей и их эффективное использование на практике ограничено только техническими проблемами.

7. Высокая надежность и доступность, поскольку процессор состоит из параллельно работающих однотипных компонент.

8. Возможность широкого применения элементов высокой степени интеграции при реализации отдельных унифицированных модулей обрабатывающего и управляющего устройств, памяти данных и программной памяти.

З а к л ю ч е н и е. Очевидно, что наиболее существенные трудности практической реализации на настоящем уровне развития технических средств связаны с конструированием ассоциативных памятей с указанными функциональными характеристиками и достаточным быстродействием. Возможно частичное решение этой проблемы с помощью структурных методов, например, расслоения памяти данных и программной памяти на секциях и использования параллельного поиска.

Предварительное общее исследование при помощи модели [II], составленной на базе языка APL , подтвердило функциональную работоспособность элементарного процессора. Дальнейшее развитие архитектуры систем, основанных на подобных принципах, очевидно, потребует проведения обширных исследований в различных направлениях: совершенствования абстрактной модели и исследования её эффективно-

сти, исследования зависимости между параметрами составных компонент и производительностью системы, разработки базового машинного языка и языка программирования высокого уровня, исследования тупиковых состояний, разработки системного программного обеспечения и т.д.

Опубликованная недавно статья Румбаха [I2] и цитированные в ней работы говорят о проведении активных исследований в области граф-программного управления, которые, вероятно, окажут в будущем влияние на разработку новых систем для обработки данных.

Авторы выражают глубокую благодарность Хр.А.Турлакову за ряд ценных замечаний и за большую помощь в написании статьи.

Литература

I. Принципы работы системы IBM/370 . Под ред. Л.Д.Райкова,М., "Мир", I975, с.28-49.

2. KELLER R.M. Look-Ahead Processors.- "ACM Computing Surveys",1975,v.7,N 4,Dec.

3. BEAR J.L. A survey of some theoretical aspects of multiprocessing.-"ACM Computing Surveys",1973,v.5,N 4,Mart.

4. ЕВРЕИНОВ Э.В., КОСАРЕВ Ю.Г. Однородные универсальные вычислительные системы высокой производительности.Новосибирск,"Наука", I966.

5. Мультипроцессорные системы и параллельные вычисления. Под ред. Ф.Г.Энслоу. М., "Мир", I976.

6. КОТОВ В.Е., НАРИНЬЯНИ А.С.Асинхронные вычислительные процессы над памятью. -"Кибернетика" (Киев), I966, № 3.

7. КОТОВ В.Е. Теория параллельного программирования. Прикладные аспекты. -"Кибернетика" (Киев), I974, № I,2.

8. НАРИНЬЯНИ А.С. Теория параллельного программирования.Формальные модели.-"Кибернетика", (Киев), I974, № 3,5.

9. DENNIS J.B. A preliminary architecture for a basic dataflow processor. MIT Project MAC, Computations Structure Group Memo 102, Aug.,1974.

I0. ДИЙКСТРА Э. Взаимодействие последовательных процессов. -В кн.: Языки программирования. М., "Мир", I972.

II. КЪНЧЕВ Т., ЯНЕВ К., БОЯНОВ К. Модель высокопроизводительной системы для обработки данных с ассоциативным доступом к программной информации и данным. - УI-й Балканский математический конгресс. Резюме докладов, Варна, I977.

I2. RUMBAUGH J. A data-flow multiprocessor.- "IEEE Trans.Computers",1977,v.C-26,N 2,Feb.

Поступила в ред.-изд.отд.
I4 сентября I977 года