ОРДЕНА ЛЕНИНА
ИНСТИТУТ ПРИКЛАДНОЙ МАТЕМАТИКИ
АКАДЕМИИ НАУК СССР

И.Б. Задыхайло, Е.И. Котов,

А.Н. Мямлин, Л.А. Поздняков, В.К. Смирнов.

ВЫЧИСЛИТЕЛЬНАЯ СИСТЕМА С ВНУТРЕННИМ ЯЗЫКОМ ПОВЫШЕННОГО УРОВНЯ.

Препринт № 41 за 1975 г.

Москва.

ОРДЕНА ЛЕНИНА ИНСТИТУТ ПРИКЛАДНОЙ МАТЕМАТИКИ АН СССР

И.Б.Задыхайло, Е.И.Котов, А.Н.Мамлин,
Л.А.Поздняков, В.К.Смирнов

ВЫЧИСЛИТЕЛЬНАЯ СИСТЕМА
С ВНУТРЕННИМ ЯЗЫКОМ ПОВЫШЕННОГО УРОВНЯ

Москва 1975

УДК 681.3

Вычислительная система с внутренним языком повышенного уровня.

Задыхайло И.Б., Котов Е.И., Мямлин А.Н.,
Позднияков Л.А., Смирнов В.К.

ИПМ АН СССР, Препринт № 41, М., 1975,
42с., библиогр. 72 назв.

Излагаются принципы построения вычислительной системы с внутренним языком, близким к естественному языку описания процессов обработки информации. На основе анализа функций, выполняемых современной вычислительной системой, предлагается структура системы, состоящей из пяти специализированных процессов: арифметического, символьного, архивного, управляющего и процессора ввода-вывода. Проводится сопоставление вычислительной системы предложенной структуры с некоторыми известными вычислительными машинами.

Ключевые слова : вычислительная система, фон-неймановская машина, специализированный процессор, арифметический процессор, символьный процессор, архивный процессор, управляющий процессор, процессор ввода-вывода .

С О Д Е Р Ж А Н И Е

I. ВВЕДЕНИЕ

В настоящей работе сделана попытка обосновать выбор архитектуры вычислительной системы с внутренним языком, близким к естественному языку описания обработки информации, которая разрабатывается в ИПМ АН СССР в соответствии с постановлением Комитета по Науке и Технике при СМ СССР.

Постановка задачи приближения языка машины к языку описания процесса обработки информации вызвана стремлением повысить эффективность работы вычислительной системы. Дело в том, что быстрое расширение функций, выполняемых современной вычислительной системой привело к тому, что аппаратная часть системы слабо учитывает и отражает специфику некоторых функций, которые приходится выполнять современной вычислительной системе. Такое расширение функций осуществляется сейчас главным образом программным путем. Из-за этого, с одной стороны, сильно возрастает объем и сложность программного обеспечения, с другой стороны, некоторые функции выполняются недостаточно эффективно. Сближение внутреннего языка системы с естественным языком описания обработки информации приводит к тому, что уровень внутреннего языка поднимается и в нем появляются объекты и операции, адекватные отдельным областям и этапам обработки информации, типичным для современных задач, решаемых на вычислительных машинах. В дальнейшем будет показано, что при этом нельзя обойтись без функционального расширения состава вычислительной системы и без специализации составляющих ее компонентов.

Ясно, что стремление к существенному повышению уровня языка машины и его приближение к естественному связано со значительным смещением акцента применения вычислительных машин и дальнейшими тенденциями увеличения этого смещения. Задачи неарифметического плана уже сейчас занимают значительную долю машинного времени. Поэтому необходимо стремиться повысить эффективность работы вычислительных машин на задачах трансляции (в широком смысле), символьных преобразований, редактирования, создания справочных систем и т.д.

Мы будем называть рассматриваемую систему Базовой Системой Повышенной Квалификации (БСПК), поскольку она представляет собой, по нашему мнению, необходимую и достаточно полную основу для решения задач из различных областей науки, техники и производства.

Сбалансированный подъем уровня языка системы достигается путем выделения нескольких областей специализации, каждая из которых отражает определенную функцию, выполняемую машиной в вычислительном процессе. Эти функции закладываются и закрепляются в машине аппаратно, что и обеспечивает повышение ее "квалификации".

Изложение материала разбито на три части. Первая посвящена анализу причин ревизии фон-неймановской архитектуры вычислительных машин. Вторая - обоснованию принципов построения БСПК. В третьей части анализируются некоторые, наиболее интересные с нашей точки зрения, вычислительные системы и отмечаются их недостатки, помешавшие разработчикам осуществить сбалансированный подъем производительности. Эту работу можно рассматривать как введение к эскизному проекту системы БСПК.

2. РАЗВИТИЕ ФОН-НЕЙМАНОВСКИХ ПРИНЦИПОВ

Почти 30-летняя история развития вычислительных машин проходила под знаком эволюционного отхода от принципов, сформулированных фон-Нейманом. Этот отход проходил постепенно и явился следствием попыток приспособить фон-неймановскую структуру к требованиям, выдвигавшимся по мере расширения сферы применения вычислительных машин и совершенствования их функциональных возможностей. В настоящее время тенденция ревизии фон-неймановских принципов построения вычислительных машин становится неодолимой. Почти каждая попытка создания новой вычислительной системы связана с этим веянием (см., например, [I,I2,33,55,57]). В указанных работах явно подчеркивается необходимость ревизии фон-неймановской схемы в том или ином направлении. В этом разделе мы попытаемся разобраться в причинах этого явления.

2.I. Основные направления развития фон-неймановской схемы

Фон-Нейман, обосновывая логическую структуру вычислительной машины [3] , выделил в ней следующие основные блоки, которые он назвал органами: орган памяти, арифметический орган, орган управления и орган ввода-вывода. Чтобы проследить основные направления развития фон-неймановской схемы, рассмотрим кратко, в каких аспектах совершенствовались эти основные компоненты вычислительной машины, введенные фон-Нейманом.

2.I.I. Орган памяти.

Начнем с памяти. В фон-неймановской машине память имела следующие особенности:

I) Она была однородной - состояла из набора несвязанных между собой ячеек, в каждой из которых хранился один элемент информации.

2) Информация в памяти идентифицировалась с помощью адреса или номера ячейки, в которой она хранилась.

3) Для хранения программ и данных использовалась одна и та же память, и содержимое любой ячейки могло быть операндом машинной операции.

Последняя особенность считалась одной из важнейших черт фон-неймановской структуры в целом, и фон-неймановские машины даже получили название машин с хранимой в памяти программой.

С самого начала пользователи вычислительных машин испытывали недостаток в памяти. В результате в машинах наряду с оперативной памятью стала использоваться внешняя память, на которой хранилась информация, не умещающаяся в оперативной памяти и в данный момент не нужная для вычислений. Это привело к тому, что объекты, с которыми имела дело вычислительная машина, могли находиться как в оперативной, так и во внешней памяти, а следовательно, должны были по разному идентифицироваться. Программист должен был сам следить за перемещением объектов из одного вида памяти в другой и осуществлять переход от одного обозначения объекта к другому. Стремление избавиться от этого недостатка привело к концепции виртуальной памяти, в которой объекты идентифицируются с помощью виртуального адреса, не зависящего от физического положения объекта. Широкое использование алгоритмических языков привело в вычислительную систему давно выработанное человечеством понятие имени (идентификатора), как способа абсолютной идентификации объектов. Использование имени для идентификации объекта по существу означает, что должна адресоваться не ячейка, где хранится объект, а сам объект. Поэтому память должна быть структуризована так, чтобы легко можно было адресоваться к объекту любого типа, независимо от того, каков формат его внутреннего представления в машине. Слежение за тем, как и сколько единиц адресуемой информации помещается в физической ячейке памяти, необходимо переложить на аппаратуру.

Другой аспект структуризации памяти связан с отказом от представления о памяти как о совокупности несвязанных ячеек. Появились различные структуры памяти – списочные [55], древовидные [I], стековые [5I]. Вначале структуризация памяти осуществлялась чисто программными средствами. Затем были предложены различные аппаратные механизмы, реализующие ту или иную структуру памяти. В настоящее время трудно отдать предпочтение какой-то одной структуре для построения оперативной памяти машины. Скорее всего можно предполагать, что память будет специализироваться под отдельные функции, выполняемые вычислительной машиной, и вычислительная машина будет иметь не одну, а несколько структур памяти.

Третий принцип поначалу казался очень удобным, так как давал способ реализации циклических алгоритмов, поскольку позволял перед каждым новым исполнением цикла модифицировать адреса некоторых команд. Однако программная модификация адресов команд приводила

к большим потерям в эффективности. Это вызвало развитие методов модификации адресов. Команды в памяти практически перестали изменяться: адрес разделился на две части – изменяемую и неизменяемую: неизменяемая указывается в команде, а изменяемая (база, индекс) хранится на внутренних регистрах процессора. Аппаратная модификация адресов позволила получить ощутимый выигрыш во времени. Вскоре выяснилось и принципиальное неудобство модификации команд в памяти. Появление концепции реентерабельных программ, т.е. программ, которыми одновременно могут пользоваться различные процессы, сделало такое изменение по существу невозможным.

Таким образом, выкристаллизовалась разница между программной информацией и данными. Программа – это информация, подлежащая исполнению, данные – это информация, подлежащая переработке. Появилась необходимость в защите информации. Каждый элемент информации, хранящейся в памяти, должен иметь свой статус защиты: один можно только брать (читать) из памяти, другой допустимо изменять, третий можно исполнять и т.д. По мере расширения сферы применения машин сами данные перестали быть однородными – появился целый набор типов информации в машине. То, как машина воспринимает информацию, какие выполняет преобразования – определяется типом информации. Оказалось удобным считать, что каждый объект, находящийся в памяти машины, обладает определенным набором свойств, которые определяют не только, что можно и что нельзя делать с этим объектом, но в ряде случаев и конкретизируют, как именно осуществляется преобразование объекта, если оно вообще возможно. Таким образом, память машины перестала быть вместилищем однотипных объектов – кодов.

В результате простая линейная память, введенная фон-Нейманом, перестала отвечать требованиям многих задач, решаемых на вычислительной машине. Поэтому приходится моделировать программными методами структуры памяти, адекватные определенным классам задач. Это приводит к потере эффективности. Естественный путь в преодолении этого недостатка состоит в создании аппаратной поддержки для реализации этих структур.

На пути совершенствования рассматриваемого органа памяти мы видим два наиболее существенных направления: а) использование имен для адресации объектов и б) структуризация памяти. Повышение уровня машинного языка при введении в машину этих возможностей позволит, как уже указывалось, повысить эффективность их реализации за счет следующих факторов:

I) Увеличения быстродействия машины благодаря более решительному внедрению аппаратных решений (так как аппаратные решения обычно бывают более эффективными).

2) Повышения удобства работы с программами на языке машины как при их составлении (кодировании), так и при их развитии (отладке, изменении).

К этим и некоторым другим вопросам мы еще вернемся в дальнейшем. Отметим также, что вопросы повышения уровня машинного языка довольно широко и подробно рассматриваются в работе [8], а вопросы, относящиеся к органу памяти, наиболее интересно решены в вычислительных системах, описанных в работах [7,I8,45,5I,55, 6I].

2.I.2. Орган управления

Развитие этого органа определилось кроме стремления к повышению уровня языка и увеличению производительности, еще и появлением мультипрограммирования.

Мультипрограммирование привело к тому, что появились совершенно новые механизмы:

I. прерывания;

2. защита памяти;

3. привилегированные команды и режимы.

Дальнейшее совершенствование этого направления привело к появлению многопроцессорных систем и аппаратных средств, которые упрощают их использование и дают возможность извлечь из подобных систем максимальные выгоды (например, семафорные данные и автоматическое отключение неисправных процессоров). Появились особые режимы работы: разделение времени и работа в реальном времени.

Восприятие более развитых входных языков намного усложнило блок расшифровки команд, который теперь зачастую должен работать с кодами команд неодинаковой длины, шифрующих порой весьма тонкие и сложные алгоритмы.

Наконец, погоня за скоростью вызвала к жизни такие приемы как поточный метод обработки команд, применение сверхоперативных памятей (кэшей), параллельное выполнение операций и т.д.
Мы считаем, что эти механизмы и принципы должны быть учтены при создании новой системы.

Более подробное описание принципов построения органов управления можно найти, например, в работах [I,7,I8,29,45,47,5I,55,6I]

2.I.3. Арифметический орган.

В идейном отношении этот орган претерпел наименьшие изменения в процессе эволюции вычислительных машин. Но это и понятно. Математическая основа этого органа, сложившаяся в результате длительной эволюции носит весьма законченный характер. Основные нововведения, которые связаны с работой этого органа, больше касаются вопросов, разобранных в связи с органами памяти и управления. Несколько более разнообразными стали виды значений (например, появились логические) и формы их представлений. Кроме того, появилась возможность агрегатирования данных.

Консервативность этого органа подчеркивается тем, что языки программирования типа ФОРТРАНа и АЛГОЛа-60, которые выражают вычислительную направленность применения машин, с успехом используются для решения задач, несмотря на то, что "прожили" около I5 лет. Более того, как показывают эксперименты [46], чаще всего в программах используются простейшие конструкции таких языков. Однако и в арифметическом органе можно наблюдать тенденцию повышения уровня машинного языка с целью получения большего быстродействия. Эта тенденция наиболее ярко проявляется в стремлении ввести в машину более "крупные" типы данных, такие как вектора, матрицы, и определить над ними соответствующие операции [47,56]. Новейшие системы "выжимают" арифметическое быстродействие также и за счет массовых однородных вычислений, что возможно лишь при решении специализированных задач.

2.I.4. Орган ввода-вывода.

Этому последнему органу фон-Нейман уделил не так много внимания. В основном оценивались разумные скорости изображения, печати и перфорации, а также ввода информации в машину. Между тем, эволюция этого органа чрезвычайно интересна, как мы увидим позднее, с точки зрения нашей работы. В настоящее время он объединяет набор разнородных внешних устройств.

Пожалуй, наибольшей стройности и законченности в реализации этого органа удалось достичь фирме IBM.

В системе IBM/360 орган ввода/вывода выделен в самостоятельный процессор (канал), обладающий собственной системой команд. За счет этого удалось существенно повысить уровень языка общения центрального процессора с устройствами ввода/вывода. Машинная команда, задающая работу каналу, выглядит так: "осуществить обмен с таким-то устройством по такой-то программе". Это позволяет освободить программу центрального процессора от указаний специфи-

ческих для отдельных устройств режимов и способов работы. Вся эта специфика выносится в программу канала, т.е. специализированного процессора ввода/вывода.

Вторым важным достижением организации ввода/вывода, предложенным IBM, является стандартизация подключения к каналу разнотипных устройств ввода/вывода.

Каждое внешнее устройство имеет строгий уровень квалификации. Все устройства снабжаются дополнительными аппаратными средствами (контроллерами), это позволяет унифицировать управление разнородными по своим функциям устройствами. Строго оговаривается форма содержательной и управляющей информацией, которой обменивается канал с контроллерами внешних устройств, а также дисциплина их обслуживания. В результате получается возможность единообразно подключать к системе большое количество разнообразных периферийных устройств, которые могут работать параллельно и асинхронно. В большинстве вычислительных машин, даже самых новейших, используется этот принцип. В некоторых из них устройства подключены к вспомогательной специализированной машине, которая производит дополнительную обработку информации перед передачей (приемом) ее центральной части. Интересно отметить, что уже в работе [39] фон-Нейман рассматривал вариант с использованием для органа ввода/вывода специализированной вычислительной машины.

2.2. Необходимость новых органов

До сих пор рассматривался лишь характер изменения тех органов, которые были предложены фон-Нейманом. Теперь мы переходим к краткому обоснованию того, что в современных вычислительных машинах необходимо ввести еще два органа.

2.2.I. Орган символьных преобразований

В настоящее время вычислительные машины все чаще и настойчивее используются для решения неарифметических задач, к ним относятся прежде всего следующие:

а) редактирование данных при вводе/выводе;

б) трансляция;

в) аналитические выкладки.

Задачи подобного рода уже сейчас занимают заметную долю машинного времени (по некоторым данным [2] - до половины). Понятно, что удельный вес таких задач будет в дальнейшем возрастать. Поэтому этот орган требует к себе не меньшего внимания, чем арифметический. Предварительные расчеты показывают, что аппаратная реализация перечисленных выше функций с помощью специального органа, предназначенного для символьных преобразований, может дать весьма существенный (не меньше чем на порядок) выигрыш во времени, по сравнению с программной реализацией на современных машинах [I4]. Заметим, что указанные расчеты делались на базе языка РЕФАЛ [23,24], который кажется нам хорошей основой для создания органа символьных преобразований. Язык этот к настоящему времени хорошо проверен в рамках программной реализации [4,5,26]. Включение в систему такого органа фактически означает создание основ для полнокровного развития языков и письменности.

2.2.2. Орган накопления информации (архив)

Современные вычислительные системы предполагают наличие огромного программного обеспечения. Отдельные программы выступают в качестве индивидуумов, которые могут пользоваться знаниями, накопленными в виде программ и информационных массивов. Поэтому мы обязаны постулировать наличие в современной вычислительной машине развитого органа накопления информации, на базе которого мог бы быть организован архив. Работа этого органа должна руководствоваться теми алгоритмами и приемами, которые накоплены человечеством при работе с документами и книгами. Сюда относится: возможность создания различных библиотек, уничтожения их; запоминания в библиотеках объектов (с указанием их свойств) по названию, поиск объектов по названию, уничтожение их; защита библиотек от неаккуратного использования отдельными программами, обеспечение секретности информации, повышение уровня доступности (по времени) отдельных объектов и т.д.

Работы [I5,20,52] показывают, что аппаратные решения повышают производительность и в рамках создания этого органа. А обеспечение подлинной секретности, вероятно, вообще невозможно без создания специальной аппаратуры.

2.3. Причины изменения архитектуры вычислительных систем

В дальнейших разделах мы вскроем те причины, которые вызвали изменение логической структуры вычислительных систем.

2.3.I. Оценка эффективности вычислительной системы

С точки зрения пользователя вычислительной системы, мерилом ее качества является быстрота решения на ней его задачи.

Эффективность можно оценивать по времени, которое уходит на формулировку задачи τ_{φ}, времени, затрачиваемому пользователем на ее отладку и доводку $\tau_{от}$, и машинному времени $\tau_{сч}$, затраченному в процессе отладки и счета по готовой программе. Можно качественно определить эффективность вычислительной системы для решения некоторой задачи ($Э_з$) по формуле:

$$Э_з = 1/(C_1\tau_{\varphi} + C_2\tau_{от} + C_3\tau_{сч}).$$

Коэффициенты C_1, C_2 и C_3 – это весовые коэффициенты, учитывающие значимость соответствующих времен. Обычно все эти коэффициенты отличны от нуля и поэтому разумно рассмотреть методы, уменьшающие каждое время в отдельности. Усредним каждое из времен по набору задач на данной установке, тогда формулу эффективности системы можно записать в виде

$$Э = 1/(C_1 t_{\varphi} + C_2 t_{от} + C_3 t_{сч}),$$

где t – усредненное время.

Напомним, что мы рассматриваем вопрос лишь качественно и поэтому идем на грубые упрощения. Из этих же соображений, а также из-за некоторого различия в целях, мы отказываемся от определений и формул работы [8].

2.3.I.I. Уменьшение $t_{сч}$.

Уменьшение этой величины до последнего времени шло, в основном, за счет оптимизации следующих величин: параметров памяти, времени межрегистровых передач и времени срабатывания схемных элементов. Однако эти ресурсы сейчас исчерпываются (в предельных случаях), так как начинают сказываться принципиальные физические ограничения.

Новые резервы состоят здесь в укрупнении и обобщении опера-

ций и операндов. Такой подход дает возможность за счет усложнения аппаратных решений оптимизировать обращения к памяти (выбор команд, работа с промежуточными результатами), а также организовать оптимальное совмещение операций во времени (параллельность).

В настоящее время получила широкое распространение методика организации более развитых операций, основанная на использовании дескрипторов и тэгов [38]. Мы считаем этот аппарат хорошим вспомогательным средством для продвижения по этому пути.

Наши исследования на пути выбора и реализации операций показывают, что разные по типу операции должны требовать особого подхода и специальной реализации. Быстрота выполнения сложной операции зависит в основном от параметров памяти и метода ее использования. А резервы на этом пути выбираются лишь за счет раннего обособления отдельных операций, строгой дисциплины работы с памятью и совмещения действий. Интересно, что почти во всех направлениях использования более развитых операций можно получить выигрыш во времени $t_{сч}$. Однако, этот выигрыш зависит от типа операции. Например, известно, что введение специального процессора для операций над векторами фирмой IBM [56] дает ускорение на этих операциях порядка 3÷I0 раз в зависимости от модели. В работе [I6] показано, что выигрыш в классе таких операций (имеется в виду смесь GAMMA-MIX [58]) принципиально не может быть очень велик без появления качественно новых элементов. В работе дано не строгое доказательство того, что на памяти типа БЭСМ-6, даже с учетом расслоения памяти, т.е. такой ее организации, при которой возможна одновременная выборка нескольких операндов, нельзя в принципе получить выигрыш более чем в I0 раз на смеси GAMMA-MIX по сравнению с временем исполнения на теперешних машинах БЭСМ-6. Это объясняется тем, что машины издавна применялись для задач вычислительного плана, а оценка GAMMA-MIX как раз и предложена для определения производительности машин в классе таких проблем.

В области задач обработки символьной информации, к которой относятся, например, процессы трансляции и аналитических преобразований формул, применение аппаратных решений дает выигрыш, по крайней мере, в I0 раз даже в том случае, когда аппаратное решение поддерживает устоявшуюся программную схему [I4]. Использование же возможности параллельного выполнения различных операций [I4] сулит дальнейшее существенное увеличение производительности. Даже в области работы с информацией по наименованиям имеются определен-

ные резервы. В работах [I5,20] показано, что можно организовать быструю работу со справочной системой за счет аппаратных решений.

Таким образом, за счет структурных решений можно на порядок уменьшить $t_{сч}$. Конечно, под узкие классы задач можно в некоторых случаях так подогнать аппаратуру, что выигрыш окажется чрезвычайно большим. Однако, мы хотим остаться в рамках универсальной машины, поэтому оставляем в стороне вопросы узко специализированных машин.

2.3.I.2. Оптимизация $t_{от}$.

Перейдем теперь к рассмотрению $t_{от}$. В предположении, что программист всегда имеет возможность выйти на машину, время отладки зависит от точности информации о работе машины по его задаче и форме этой информации. Кроме того, большое значение имеет возможность удобного внесения в программу исправлений и задания тестовой информации и тестовых режимов исполнения.

Уже известно большое количество работ, в которых обосновываются и предлагаются конкретные аппаратные средства для оптимизации $t_{от}$ [I,8,55] . Приведем наиболее важные черты, которыми должна обладать современная машина в рассматриваемом плане.

I). Обеспечение общения с программой на уровне условных (мнемонических) наименований [I,8,55] .

2). Сохранение структурной целостности объектов, при которой обеспечивается точная идентификация отдельных частей структуры программы и данных, их защита, а также возможность свободного маневрирования формой и размерностями [I,8,55] .

3). Возможности гибкого управления отслеживанием отдельных элементов программы с печатью информации или автоматической ее обработкой [I,55,I4] .

4). Возможности быстрого и удобного аппарата исправления программ на уровне входного языка.

Следует отметить, что программное решение этих вопросов, обычно приводит к резкому увеличению $t_{сч}$. Приведем два примера.

а) Программная проверка выхода индекса за заданный диапазон приводила к такому расходу времени (каждая переменная с индексом под подозрением), что использовалась редко, Аппаратная проверка, которая имеется почти на всех современных вычислительных системах, делается крайне быстро и весьма облегчает отладку.

б) Пусть требуется проследить за несколькими идентификатора-

ми из данного списка. При программном методе мы должны сравнивать адрес каждого операнда с адресами из списка. При аппаратной реализации соответствующие значения могут сопровождаться признаком, вызывающим прерывание. Таким образом, машина "отвлекается" лишь на заданных операндах.

Из сказанного в этом разделе видно, что тенденция использования аппаратных решений для уменьшения $t_{ом}$ вполне оправдана и уже приносит ощутимые плоды.

2.3.I.3. Оценка $t_{ф}$.

Перейдем теперь к обсуждению времени формулировки задачи $t_{ф}$. На первый взгляд кажется, что это время мало зависит от архитектуры машины, а целиком определяется тем программным обеспечением, которое является надстройкой (причем огромной) над системой команд и архитектурными принципами машины (базисом). Действительно, рассмотрим вычислительные системы, поставляемые фирмой IBM. В настоящее время колоссальное математическое обеспечение этой фирмы определило ее господство на рынке, так как для большинства пользователей время $t_{ф}$ является решающим. Однако система команд и архитектурные принципы, положенные в основу 360 и 370 серий машин этой фирмы, устаревают и становятся тормозом: налицо противоречие между базисом и надстройкой.

Остановимся на рассмотрении этого явления несколько более подробно. Поскольку мы имеем в виду лишь универсальные вычислительные машины, то, в принципе, можно на любой машине при помощи программного обеспечения выйти на один уровень удобства программирования. Однако никому не приходит в голову реализовать для практических целей машину Тюринга. Все дело здесь в том, что в качестве первичных операций рассматриваются более сложные операции и операнды, такие как, например, сложение, умножение и деление чисел с достаточной разрядностью. Основное требование – выполнить их как можно быстрее, используя любые аппаратные ухищрения (в частности, совмещение во времени физических процессов).

Теперь чрезвычайно важно разобраться в отличии между аппаратной и программной реализацией элементов вычислительной системы. До сих пор просто предполагалось (и на примерах показывалось), что аппаратная реализация в пределе позволяет достичь большой эффективности. В следующем разделе мы вскроем источники потерь эффективности при программной реализации.

2.3.2. Сравнение аппаратных и программных реализаций

Предположим, что для некоторого класса задач существует множество K_0, минимизирующее t_Φ. Это множество включает набор операций, операндов и дисциплину их выполнения.

Чем уже класс задач, для которых строится K_0, тем крупнее будут его элементы. Появление принципиально новых задач делает, конечно, невозможным создание единого, раз и навсегда фиксированного K_0, однако, для уже устоявшихся классов задач построение таких множеств, вероятно, возможно.

Итак, пусть для некоторого класса задач выбрано K_0. Естественно пытаться реализовать на машине элементы этого множества. Существуют два пути такой реализации: аппаратный и программный. При аппаратной реализации сами элементы набора K_0 являются первичными; при программной реализации первичными являются более мелкие элементы $K_м$ – множества, характеризующего машину, на которой реализуется K_0, а каждый из элементов K_0 представляется в виде некоторой программы в $K_м$.

Прежде, чем перейти к источникам потери эффективности при программной реализации по сравнению с аппаратной, сделаем несколько замечаний.

Множество операций любой машины можно разбить на две группы: операции преобразования информации и операции управления. Операцию преобразования информации можно рассматривать как такое устройство, на вход которого поступают операнды данного множества, а на выходе получаются результаты их преобразования, т.е. операнды того же множества. Поступающие на вход операнды претерпевают в этом устройстве, вообще говоря, несколько последовательных преобразований, причем каждое из преобразований может быть достаточно сложным, включая параллельную обработку различных операндов и их частей. Таким образом, в работе каждого из устройств-операций можно выделить ряд последовательных шагов преобразования информации. Последовательность этих шагов в каждом устройстве строго фиксирована и ее исполнение не требует дополнительного управления. Значит, всякую операцию преобразования информации можно рассматривать как последовательность более простых операций с жестким управлением. Как правило, эти простые операции не требуют обращений к памяти, а преобразуют информацию уже на регистрах.

Аппаратная реализация каждой из операций преобразования информации допускает наилучшее решение за счет свободного выбора составляющих простых операций и возможности любой оптимизации связи между ними и управлением.

Укажем далее источники потери эффективности при программной реализации:

I) $К_м$ может не содержать всех простых операций, составляющих операции преобразования информации из $К_о$ и не содержать операций, из которых оптимальным образом можно было бы построить эти простые операции.

2) Даже в том случае, когда $К_м$ содержит все необходимые простые операции, последовательность их выполнения при программной реализации связана принятым методом выполнения программы и может потребовать дополнительных обращений к памяти как за командами, так и за операндами, а также потерять возможность совмещения.

3) До сих пор мы говорили об операциях из $К_о$ и $К_м$. Но источником потери эффективности может быть и низкий уровень (мелкость) операндов $К_м$. Ведь не только операции $К_о$, но и составляющие их простые операции применяются к операндам $К_о$ или производным от них промежуточным результатам. Низкий уровень операндов $К_м$ приводит к тому, что в том месте, где при аппаратной реализации $К_о$ требуется лишь одна простая операция, при программной реализации средствами $К_м$ может потребоваться несколько операций $К_м$, а это увеличивает количество операций $К_м$, необходимых для представления операций набора $К_о$, т.е. ведет, вообще говоря, к потере эффективности при исполнении.

4) Текст, написанный в терминах $К_о$, необходимо перевести в текст, написанный в терминах $К_м$. Этот перевод также снижает эффективность программной реализации. Фактически это означает, что $К_м$ должно учитывать не только $К_о$ для фиксированного класса, но и $К_{ом}$ для задачи трансляции $К_о$ в $К_м$, иначе потери на этапе трансляции могут оказаться большими.

5) Наконец, если элементы $К_м$ крупнее элементов $К_о$, то при выражении последних через первые могут возникнуть большие трудности и в результате значительная потеря эффективности при реализации $К_о$ на машине с $К_м$.

Большинство работ по ревизии фон-неймановских принципов архитектуры вычислительных машин как раз и базируются на том, что их

авторы предлагают перенести в аппаратуру различные операции и формы операндов (иногда крайне сложные) для того, чтобы приблизиться к гипотетическому оптимальному набору операций. Каждый раз предложения авторов являются фактически критикой машин серий 360 и 370 (за исключением, конечно, концепции каналов), которые воспринимаются многими как высшее достижение на пути плавного развития фон-неймановских принципов. Критика эта настолько многогранна, что мы лишь отсылаем читателя к ссылкам на литературу, данным в предыдущих разделах. Здесь же только заметим, что в связи с громадной надстройкой, которая создана над весьма прогрессивной в свое время, но уже устаревшей системой команд и архитектурой, изменение последних чрезвычайно затруднено, а в некоторых отношениях невозможно. Налицо классическая ситуация противоречия между базисом и надстройкой, которая чревата революционными изменениями.

2.4. Резюме

Из проведенного обсуждения ясно, что хотя для пользователей первостепенное значение имеет t_{φ} , им вовсе не безразлично, какими методами это достигается (ведь C_2 и $C_3 \neq 0$). Действительно, t_{om} и $t_{cч}$ существенным образом сказывается и на стоимости расчетов и на производительности вычислительной системы. Конечно, при усложнении операций растет и стоимость аппаратуры, но в гораздо меньшей степени. Удешевление элементной базы и стремительный рост возможностей настоятельно диктует внедрение более развитых аппаратных решений в вычислительные системы. Эта неудержимая тенденция также имеет первостепенное значение для объяснения ревизии фон-неймановских принципов архитектуры вычислительных машин.

Заметим, что мы не уделяем должного внимания объему памяти, стоимостной вес которой достаточно велик. Это связано с тем, что точные оценки изменения этого параметра для новейших систем привести трудно, а грубые прикидки лишь показывают, что не ожидается, при прочих равных условиях, резкого изменения этого параметра в худшую сторону. Размеры программы обычно уменьшаются из-за использования более емких операторов.

3. НЕОБХОДИМОСТЬ АППАРАТНОЙ РЕАЛИЗАЦИИ БАЗОВЫХ ФУНКЦИЙ ВЫЧИСЛИТЕЛЬНОЙ СИСТЕМЫ

3.I. О необходимости специализации

В соответствии с рассуждениями предыдущей главы при конструировании вычислительной системы следовало бы поступить следующим образом: на основе K_0 определить границу между программным и аппаратным обеспечением, а затем развивать то и другое с полной гарантией успеха. Заметим, что реализацию команд и операндов K_0 можно было бы осуществлять и программно, и аппаратно. Устойчивость всей системы в целом от этого бы не пострадала, а выбор пропорции определялся бы в основном экономическими соображениями. Трудно подсчитать выгоды от такого подхода, однако, он в общем случае неосуществим. В самом деле - K_0 определяется, например, набором решаемых задач, а последний стремительно меняется во времени как в качественном, так и в количественном отношении. Отсюда следует, что K_0 существует лишь при некоторых предположениях о наборе тех задач, для решения которых создается вычислительная система. Казалось бы, идеальным является создание специализированных систем под каждый класс задач. При этом мы должны рассматривать множество K_0^i, где i определяет данный набор задач. Такой путь сулит наибольшую выгоду для эффективности на классе задач $Э^i$ (естественно, теперь и у величины Э ввести индекс i). Поскольку понятие "класс" носит для нас формальный характер, будем считать, что мы этим термином выделяем каждого пользователя (или даже отдельную его задачу). В настоящее время такая постановка не является чересчур утопической, так как уже появляются машины с настраиваемой системой команд [II], Формально, используя такой метод глобальной специализации, мы можем достичь оптимального значения величины Э или, по крайней мере, приблизиться к нему. Более того, часто решения, принимаемые при таком подходе, являются практически целесообразными. Примером этого могут служить: кассовый аппарат, являющийся простейшей вычислительной машиной; ЭВМ МИР [7], предназначенная для инженерных расчетов; сверхбыстродействующая машина ИЛЛИАК IУ [29,37,48], предназначенная для быстрой однородной обработки массивов. Фактически мы имеем вычислительные машины, обладающие определенной квалификацией в заданной области, некоторыми профессиональными знаниями. Но специалист в одной области не может

хорошо выполнять работу, не соответствующую его профилю. Такое обстоит дело и с вычислительными машинами.

Заметим, что чрезмерная специализация приводит к утрате всякой квалификации. Каждая профессия характеризуется своим языком, соответствующим системе понятий, сложившейся в ней. Этот язык отражает множество K_0, относящееся уже не к отдельному индивидууму, а к целому их классу, который определяется природой решаемых задач. Появление такого языка знаменует качественно новый уровень развития профессии. Закрепленные в виде понятий языка термины, емкие и точные, обычно открывают новые возможности. История программирования, хотя и не очень большая, ярко свидетельствует о том, что эволюция вычислительных систем в точности следует такому же способу развития.По счастью, аппаратная реализация более сложных понятий языка обеспечивает, как было показано ранее, выигрыш в эффективности. Впрочем, и это в точности соответствует приобретению профессионального опыта. Из приведенных рассуждений следует, что нельзя злоупотреблять теми новыми возможностями, которые обещают нам машины с настраиваемой системой команд. Их следует рассматривать прежде всего как метод, упрощающий введение аппаратных решений в вычислительную систему.

3.2. Выбор базовых элементов вычислительной системы

Продолжим теперь нашу аналогию в отношении процесса выбора базовых элементов, характеризующих данную профессию. Процесс этот крайне сложный и противоречивый. Он может сопровождаться ошибками и подвержен поэтому резким изменениям. При создании обеспечения вычислительных машин, мы в значительной степени должны следовать такому же извилистому пути, который неизбежен, если "профессия" только создается.

В таких случаях хорошо работает аппарат постепенного насыщения, опирающийся на метод стандартных программ и макрогенерации. Методы эти в настоящее время хорошо разработаны и могут с успехом применяться для конкретной вычислительной машины.

Они дают возможность в процессе эволюции развивать как саму сферу "профессии", так и выяснять ее K_0 (либо его приближение). Однако, здесь имеются и существенные трудности. До последнего времени реализация K_0 в основном делалась программно. При этом на

данной вычислительной машине накапливались огромные программные системы, которые определяли квалификацию машины (t_{φ}) и которые становились совершенно бесполезными при переходе на вычислительную машину с новой системой команд. Положение усугублялось еще и тем, что соображения эффективности приводили к тому, что даже алгоритмы, записанные в сложных операторах языков высокого уровня, хранились в машине в форме близкой к системе команд для избежания неэффективности, обусловленной переводом (см.раздел 2.3.I.I). Аппаратная реализация элементов K_0 была в большинстве случаев неразумной из-за дороговизны аппаратных решений. В настоящее время развитие элементной базы, методика микропрограммирования и появление вычислительных машин с настраиваемой архитектурой обещает существенный сдвиг на пути аппаратного закрепления элементов K_0. Однако вопрос о выборе некоторого единого базисного языка при этом не только не отменяется, а напротив, становится более сложным и важным.

3.2.I. Необходимость обеспечения сопряжения языков

Использование универсальных вычислительных машин предполагает возможность развития системы в любом направлении, а принцип специализации несколько противоречит этому. Например, язык АЛГОЛ-60 [I9] оказался мало пригодным для решения задач из области обработки экономической информации, в которых сравнительно небольшой счет сочетается с большой работой по нарифметическому преобразованию информации (редактирование, упорядочивание, печать и т.д.).

Появились языки типа КОБОЛ [22] . Очень часто возникает потребность использовать при решении задачи сильные стороны нескольких языков или их реализаций. Тогда приходится обращаться к системе команд машины, устраивая сопряжение между языками. Отсюда следует, что задача сопряжения различных языков является в современных вычислительных системах чрезвычайно важной. Если мы стремимся к сбалансированному повышению уровня входного языка вычислительной системы, то это повышение должно затрагивать и эту задачу. Следовательно, мы вправе говорить о необходимости введения в современную вычислительную систему специального органа, обеспечивающего эффективное сопряжение различных языковых компонент. При этом в сферу действия этого органа попадает и обеспечение диалога

с человеком, то есть создаются предпосылки эффективного восприятия машиной языка, близкого к естественному.

Задача создания такого органа программным путем не нова. Имеется несколько языков, которые могли бы служить базисом для его создания [17,32,34,50]. По нашему мнению, наиболее подходящим является РЕФАЛ [23,24]. Он обеспечивает переход от произвольного структуризованного языкового объекта к другому аналогичному объекту. Этот язык прошел хорошую проверку (в идейном смысле), ибо с успехом применялся для решения задач трансляции, конвертирования и аналитических преобразований [4,5,26]. Таким образом, множество $К_О$ для этого важнейшего органа можно считать хорошо известным. Конечно, программная реализация этого языка приводит к определенному замедлению (5–8 раз) соответствующих процессов преобразования текстов, по сравнению с прямыми методами, так как его $К_О$ плохо согласуется с $К_М$ современных вычислительных машин. Однако лаконичность языка поразительна, что является признаком его соответствия указанному классу задач. В работе [13] показано, что за счет аппаратных решений можно без особых ухищрений (следуя схемам программной реализации) получить на порядок более быстрое выполнение программ, написанных на РЕФАЛ'е, чем при программной реализации на современных ЭВМ при одинаковых элементных параметрах. При этом остаются не меньшие резервы. Заметим, что поскольку РЕФАЛ можно рассматривать как универсальный макрогенератор, мы должны были бы включить его в базис и на основании того, что он позволяет наращивать программное обеспечение над любым набором команд, что естественно требовать от любой универсальной схемы.

3.2.2. Необходимость закрепления в машине общеобразовательных знаний

Рассмотрим теперь, какие еще элементы следует включить в базис универсальной вычислительной системы. Снова вернемся к аналогии с профессиями. Причем, поскольку речь идет о системе, предназначенной для автоматизации нетехнической деятельности, рассмотрим подготовку специалиста-человека. Его учат сначала в школе, где он получает общеобразовательные знания, составляющие тот необходимый фундамент, на основе которого можно затем получить факультативные знания, пройдя курс обучения в высшем или специальном учебном заведении. Мы уже видели, что язык, подобный РЕФАЛ'у, является необходимым (обязательным) предметом в "средней школе" для универсальной вычислительной машины.

Кроме этого, машина должна понимать математические обозначения, выполнять вычисления по арифметическим и логическим формулам, уметь запоминать информацию, организуя архивные справочные системы, иметь возможность сопряжения с внешней средой. Почему мы уверенно говорим о том, что такие знания являются первичными и заслуживают того, чтобы найти свое место в K_0, отражающем универсальность машины? Дело в том, что в любом проблемно-ориентированном языке мы сталкиваемся с их воплощением в том или ином виде. Однако, из тех или иных соображений некоторые из этих необходимых элементов отражены в отдельном языке (или машине) более полно, а другие менее полно. Сейчас можно ставить вопрос о максимальном внедрении в аппаратуру свойств, отвечающих перечисленным элементам.

3.2.3. Компоненты вычислительной системы

Наиболее радикальной постановкой проблемы внесения общеобразовательных знаний в машину является следующее: каждый из перечисленных предметов закладывается в вычислительную систему в виде отдельного процессора. А именно, предлагается сделать систему, состоящую из следующих процессоров:

1. Арифметический процессор
2. Процессор символьных преобразований
3. Процессор работы со справочными системами (архив)
4. Процессор связи с внешними органами
5. Процессор управления данными и программами.

Прежде всего следует отметить, что предложенный состав процессоров не совпадает с органами, рассмотренными во втором разделе. Это связано с тем, что каждый из процессоров (в силу их усложнения) может иметь свою собственную память и свое управление. Общее управление вынесено в отдельный процессор: процессор управления данными и программами (см.ниже). Предполагается, что часть его памяти предоставляется для временного использования другим процессорам.

Во-вторых, при конструировании всех этих процессоров предполагается, что любой из них основан на именах, а не на адресном принципе. Это просто означает, что адресная система, даже если она имеется в аппаратуре, недоступна для программиста. Обоснование этого принципа дается в работе [57].

В-третьих, процессор управления данными и программами, который ранее не обсуждался, представляет собой организующий процессор, который вводится для согласования программы пользователя с внешней средой, ресурсами, адресным окружением, задачным окружением и т.д. для реализации мультипрограммной работы; работы в режиме разделения времени; организации распределения наличных ресурсов; задания параллельных действий на уровне работ других процессоров и т.д.

В-четвертых, каждый из перечисленных выше процессоров, во всяком случае, их части, уже воспроизводился в аппаратном решении. Так, например, известны аппаратные реализации символьной адресации [55,61], арифметического процессора [30,51,62], процессора символьных преобразований [59], процессора работы с архивом [52] процессора связи с внешней средой (ввод-вывод) и управления данными и программами [51,54,55,63,65]. При каждой такой реализации получался существенный выигрыш в производительности. Наши исследования, указанные в первой главе, позволяют надеяться, что за счет предлагаемого целого структурного подхода можно будет увеличить производительность вычислительной системы по меньшей мере на порядок. Чтобы закончить аналогию с обучением, надо отметить, что факультативные знания можно вкладывать в вычислительную систему либо программным, либо аппаратным методом, в зависимости от поставленных целей и стоимостных соображений.

Кстати говоря, даже в нашей схеме вычислительной системы желательно жестко фиксировать только языковое описание предлагаемых процессоров, а не способ их реализации. Это дает возможность согласовывать вычислительную систему с требуемыми эксплуатационными параметрами за счет варьирования долей аппаратных и программных средств, сохраняя программное обеспечение.

Целесообразность введения перечисленных предметов в общеобразовательный курс можно считать обоснованной. Полнота этой системы на теперешнем этапе довольно очевидна. Не исключено, что определенные обстоятельства потребуют введения в общеобразовательный курс дополнительных предметов. Хотя такая возможность будет предусмотрена в нашей вычислительной системе, мы считаем, что частота этого процесса будет по теперешним меркам мала. Такая вера в качественно новый уровень устойчивости языка вычислительной системы опирается на то, что предметы выбраны из самых общих соображений и лишь в

весьма малой степени отражают субъективные свойства "обучаемых индивидуумов".

Отметим еще раз, что эффективность при этом не только не теряется, а наоборот, на порядок возрастает. Причины такого феномена были обсуждены ранее.

Понятно, что при таком подходе решающее значение играет выбор входного языка каждого процессора (предметный K_0). Задача эта, конечно, весьма сложная, однако, большое количество типов операндов, операций и дисциплин выполнения алгоритмов, накопленных к настоящему времени, в системном программировании, позволяют надеяться на удовлетворительное ее решение. Например, для арифметического процессора можно в качестве входного языка выбрать язык, в который легко отображаются арифметические элементы языков APL , АЛГОЛ-60 и ФОРТРАН, дополнив его некоторыми элементами из универсальных языков, таких как PL /I, АЛГОЛ-68 и СИМУЛА-67. В качестве входного языка для процессора символьных преобразований можно взять язык РЕФАЛ, дополнив его некоторыми машинными операциями и т.д.

Интересно отметить, что деление на указанные процессоры имеет не только методическое значение. Например, в процессоре управления данными и программами предполагается реализовать древовидную структуру программы, сходную с описанной в работе [36] , которая позволяет отразить последовательно-параллельное выполнение операций, характерное для операционных систем. Обособленность процессоров позволяет более точно и более экономно решать для них вопросы отображения элементов языка в памяти процессора, а также применять различные методы повышения эффективности , такие как поточный метод обработки команд, использование кэша и т.д.

Более того, система становится более терпимой к смене физических принципов. Так, исследования, проводимые в настоящее время показывают, что применение опто-электронных принципов может оказаться в будущем весьма эффективным для создания процессора символьной обработки. Вопрос о включении такого процессора в систему в условиях нашей архитектуры является лишь вопросом преобразования текстовой информации из одной физической формы в другую и обратно. Причем такие паразитные преобразования должны сопровождать довольно емкую операцию (типа трансляции).

Наконец, следует отметить, что форма организации такова, что можно включать в работу несколько одинаковых процессоров с тем,

чтобы увеличить мощность системы за счет параллельной работы однотипных процессоров – если того потребует специфика решаемых задач.

4. СОПОСТАВЛЕНИЕ ПРИНЦИПОВ ОРГАНИЗАЦИИ СОВРЕМЕННЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

4.I. Специализированные машины и системы

Во втором разделе мы рассмотрели в общих чертах, как эволюционировали отдельные компоненты вычислительной машины. В результате этой эволюции появился широкий спектр вычислительных систем, при создании которых был сделан разный акцент на те или иные особенности. Это разнообразие явилось отражением различных точек зрения на необходимость изменения тех или иных компонентов системы, на важность совершенствования тех или иных функций вычислительной системы.

В конечном счете как изменение компонентов, так и совершенствование функций проходило под знаком стремления к повышению комплексной эффективности вычислительной системы на всех этапах выполнения вычислительного процесса.

Теперь мы кратко остановимся на особенностях некоторых вычислительных систем, в которых наиболее выпукло, с нашей точки зрения, отразились изменения в структуре как системы в целом, так и ее отдельных компонентов.

4.I.I. Специализированные супермашины

Один из путей резкого повышения производительности вычислительных машин заключается в создании систем, специализированных на решение узкого класса задач. Для этих задач выделяются достаточно крупные операции, такие как операции над векторами или матрицами, и для них предусматривается эффективная аппаратная реализация. Сама структура таких машин делается хорошо приспособленной под задачи определенного класса, скажем, предусматривается возможность глубокого распараллеливания алгоритма. За счет этого удается значительно (на I–2 порядка) поднять производительность. Наиболее яркими примерами такого подхода могут служить STAR-I00 [40,43,47,53,66], ILLIAC-IV [28,29,37,48], STARAN [67],

TIASC [68] . Такие системы с успехом используются для решения ряда специализированных задач. Однако ввиду их явной специализации они плохо подходят для повышения производительности систем общего назначения, поскольку далеко не во всех задачах можно использовать преимущества их организации и высокую скорость выполнения специфических операций.

4.I.2. Машины под проблемно-ориентированные языки

Другой путь специализации с целью повышения производительности с одновременным повышением удобства использования состоит в создании вычислительных машин на базе одного из проблемно-ориентированных языков. Сюда относятся попытки аппаратной реализации таких языков, как ФОРТРАН [30] , АЛГОЛ-60 [7] , APL [27,44,62], СНОБОЛ [I7,35,59] и т.д. Такой подход, вполне оправданный как способ повышения эффективности и поднятия уровня входного языка машин для некоторого класса задач, очевидно не подходит в качестве базы для создания универсальной системы. Отличие этого пути от предыдущего носит не принципиальный характер. Если предыдущий способ повышения производительности основывался на специализации машины на задачах определенного класса, то последний имеет в своей основе специализацию на языках, которые в свою очередь отражают специфику некоторого класса задач.

4.I.3. Специализированные сети

Для решения специализированных задач предназначены и некоторые сети вычислительных машин, которые в последнее время получают все большее распространенное решение. Диапазон специализированных сетей очень широк, как по мощности - от очень больших, типа OCTOPUS [69], до маленьких типа DCS в Калифорнийском университете [70] , так и по применению - от чисто научных типа СДС SYBERNET [7I] до коммерческих типа MARK Ш [72]. Но так как мы стремимся создать универсальную базовую систему, то специализированные вычислительные сети, так же как вычислительные машины под проблемно-ориентированные языки и системы под узкий класс задач остаются несколько в стороне от нашей цели. Гораздо больший интерес представляют для нас системы общего назначения. К обсуждению ряда таких систем мы сейчас и перейдем.

4.2. Системы общего назначения

4.2.1. Вычислительная система MU-5

Впервые система описана в работе [45]. Цель разработки заключалась в обеспечении высокой скорости выполнения как скомпилированных, так и ручных программ за счет сближения структуры машинного кода и языков программирования и приближения скорости работы памяти к скорости работы арифметического устройства. В качестве типовых языков выбраны АЛГОЛ и ФОРТРАН.

Одним из важных свойств, выделяющих аппаратную реализацию этой системы, является развитая система адресации (по именам и к элементам массива), отражающая возможности согласования с адресным окружением в соответствии с соглашениями, которые используются в языках высокого уровня: локализация, вложенность и т.д., особые возможности, облегчающие доступ к отдельным элементам массива, возможности работы со структуризованными элементами.

Кроме того, очень важно, чтобы структура системы команд учитывала статистические характеристики вычислительных потоков, а дисциплина выполнения команд была выбрана так, что при поточной их обработке минимизировались потери, вызванные ветвлением.

Доклад Самнера на конгрессе IFIP 1974г. [61] подвел итоги эксплуатации системы. Они весьма поучительны. Самнер отмечает, что рабочие программы языков АЛГОЛ-60 и ФОРТРАН, оказываются весьма эффективными, однако компиляция производится менее эффективно, а для описания операционных систем принятая система команд и вовсе мало пригодна.

Предлагается разработать и подключить к системе специализированные процессоры для реализации трансляции и операционной системы. Фактически предлагается продвинуться в сторону вычислительной системы, предложенной нами, с тем же набором процессоров (заметим, что они введены в рассмотрение еще в 1971г. [12]), за исключением процессора работы со справочными системами, который предполагается, видимо, реализовывать либо как часть основной машины, либо как часть процессора, который выполняет функции операционной системы. Отметим, что в докладе Самнера указывается один из недостатков системы, состоящей из специализированных процессоров. Мы обсудим его в конце этой главы (см. 4.2.5).

4,2,2, Машины B5700/B6700

Мы выбрали эти машины из-за того, что они являются наиболее типичными и развитыми системами, которые отражают стремление фирмы создать аппаратную поддержку для алголоподобных языков. Фирма решилась на очень смелый шаг, начав одной из первых решительное внедрение аппаратных решений в архитектуру.

Во многих местах аппаратные схемы следуют широко известной реализации АЛГОЛ'а-60, описанной в [2I]. Архитектура таких систем и их развитие является для нас чрезвычайно интересным.

Подробное описание рассматриваемых систем и многочисленные ссылки на соответствующую литературу можно найти в книге [5I]. Достоинства их хорошо отражены автором, который не скрывает, что является их ярым приверженцем.

Выбранный авторами системы метод реализации поднятия уровня входного языка привел к следующему парадоксу: аппаратная реализация языка, ориентированного на описание решения численных задач, не дала преимущества в скорости вычисления, более того, в некоторых важных случаях (а именно при работе с массивами) получается существенный проигрыш по сравнению с машинами традиционной архитектуры. По нашему мнению, это связано с тем, что уровень счетных команд остался прежним, а в отношении работы с массивами даже измельчился.

Изящная схема адресации, которая хорошо и экономно решает задачи сопряжения с адресным окружением в пределах работы с объектами алголоподобных языков, теряет свою стройность, а главное эффективность, при отображении других языков, например, ФОРТРАНа. Автор упомянутой выше книги отмечает, например, трудности, связанные с реализацией операторов ФОРТРАНа: EQUIVALENCE и COMMON . Этот факт является признаком превышения разумной границы между аппаратным и программным оборудованием. ФОРТРАН настолько популярен в вычислительном мире, что заслуживает тщательного учета, несмотря на спорность некоторых конструкций.

Отметим также, что совершенствование операций неарифметического плана проводилось фирмой традиционным постепенным методом и в этом отношении такие операции выглядят иногда как заплатки, и по уровню сложности не превосходят тех, которые приняты в традиционных машинах. В результате вся система команд в целом выглядит неоднородной и тяжеловесной, что обязательно должно приводить к

ненужному усложнению аппаратуры и ухудшению технических параметров.

4.2.3. Вычислительная машина GE -645

Фактически рассмотрение этой машины вызвано тем, что она является аппаратной частью системы Малтикс. Описание этой системы дано в книге [52]. Там же имеется богатая библиография. Цель разработки состояла в создании вычислительной системы с мультидоступом (коллективного пользования) для возможности систематической работы на ней большого количества пользователей. Для этого особое внимание было обращено на удобство организации на ней информационных систем: машина характеризуется огромной виртуальной памятью, средствами автоматического динамического объединения программных модулей, специальными средствами для работы с информационными массивами (заведение, доступ, многоуровневая защита), особыми средствами, упрощающими создание операционных систем и подсистем и т.д.

С нашей точки зрения, серьезной критики эта машина заслуживает лишь за недостаточное внимание к аппаратной поддержке языковых средств (процессор символьной обработки), которые для таких систем должны иметь первостепенное значение. Ведь эта система мыслилась базой для экспериментов над моделями сложных систем и подсистем.

Кроме того, заметно, что авторы недооценивали возможности достижения быстродействия за счет усложнения аппаратуры, хотя смело пользовались ею при создании удобств для разработки программного обеспечения.

4.2.4. Попытки выбора аппаратно реализуемой базы универсальной системы

4.2.4.I. Машины под универсальные языки высокого уровня.

Можно поставить следующий вопрос: не следует ли взять за основу один из универсальных языков высокого уровня типа ПЛ/I [25,4I], АЛГОЛ-68 [6] и СИМУЛА-67 [9] и сделать под него аппаратуру. Такой путь обсуждался, например, в [I,49]. Нам кажется, что идти по нему не следует. Во-первых, наличие трех довольно разных претендентов говорит о том, что аргументы в пользу каждого из них не всегда достаточно убедительны. Во-вторых, изящная закон-

ченность и глобальная общность двух последних языков обещает потерю эффективности в реализации, связанную с наличием редко используемых компонент, а первый слишком очевидно выражает концепции операционной системы и структуры машин 360 и 370 серии фирмы IBM. Переработка его, предпринятая ECMA (Европейская ассоциация производителей вычислительных машин) и ANSI (Американский национальный институт стандартов), затягивается: издана уже II версия (февраль 1974г.). В-третьих (и это самое главное): пользователей, по-видимому, больше устраивают проблемно-ориентированные языки, которые и используются с успехом для решения задач.

На наш взгляд, большое практическое значение приобретают системы, обеспечивающие накопление знаний в виде банков данных, дающих возможность усложнять языковые конструкции в конкретных областях (факультативные знания) при помощи довольно простых базовых операций.

С этой точки зрения, возможно, более оправдана реализация некоторого базового языка не столь высокого уровня, обеспечивающего упомянутые возможности. Примером подобной реализации может служить вычислительная машина B.L.M.

4.2.4.2. Вычислительная машина B.L.M.

Идеология этой машины наиболее выпукло изложена в работе [57]. Методы реализации проиллюстрированы в книге [I] . Позволим себе привести несколько цитат из работы [57] . Первая относится к методу выбора элементов, которые целесообразно реализовать аппаратно. Нужно обратиться к элементарным понятиям математики, чтобы определить, подходит ли данная конструкция для включения в постоянную логику системы, и включать ее с учетом существующих языков "высокого уровня", каждый из которых отражает, так или иначе, черты взаимосвязи с машинами на протяжении двадцати лет." Вторая цитата указывает главного претендента на аппаратное решение. "Основная идея, положенная в основу машины с Базовым языком (B.L.M), состоит в том, что с самого начала отбрасывается система фиксированных ссылок и заменяется на систему ссылок, основанную на именах, аналогичную той, которая так долго служит человечеству. Наиболее существенным свойством "имени" является то, что оно не изменяется, если элемент передвигается с одной позиции на другую внутри памяти". Далее "...необходимо, чтобы структура информации явно отража-

лась в машине". В настоящее время накоплен некоторый опыт работы на машине с базовым языком (символический язык, являющийся языком программирования на BLM), который авторы расценивают весьма положительно с точки зрения величины Э.

Мы полностью согласны с подходом авторов к архитектуре вычислительных машин, отраженном в приведенных цитатах, однако считаем, что они сделали лишь небольшой шаг по пути, намеченном первой из приведенных цитат, выбрав довольно низкий уровень базового языка. Предложенный нами метод позволяет определить сразу разумный предел языковой границы между аппаратным и программным оборудованием, а не продвигаться к нему небольшими шагами.

4.2.4.3. Вычислительная машина SYMBOL

Другая попытка создания аппаратно реализуемой языковой базы универсальной системы отражена в выборе входного языка для машины SYMBOL . В качестве такого входного языка взят оригинальный язык довольно высокого уровня [3I] , разработанный на основе АЛГОЛа, ФОРТРАНа, ПЛ/I, ЛИСПа и ЭЙЛЕРа с учетом его последующей аппаратной реализации. Этот последний факт позволяет избежать многих неэффективностей, обязательно возникающих при аппаратной реализации языков, при создании которых возможность последующей реализации не учитывалась. Надо отметить, что вообще машина SYMBOL отличается от других вычислительных машин необычайно глубоким внедрением аппаратных решений [55] . В работе [54] так формулируется задача этой системы:

"Основная цель исследовательского проекта SYMBOL заключалась в том, чтобы продемонстрировать путем создания полнокровной работающей системы, что универсальный, процедурный, весьма высокого уровня язык, отражающий современное состояние дел в программировании, а также большая часть операционной системы, допускающей разделение времени, может быть реализована аппаратно с существенным увеличением скорости вычислений". Следующие черты, получившие аппаратное воплощение, подчеркивает автор работы:

Динамическое распределение памяти,
динамическое восстановление памяти,
динамически переменная длина поля,
динамически переменные структуры,
автоматическое управление виртуальной памятью,
автоматическое преобразование типов,

автоматическое управление разделением времени,
непосредственная символьная адресация,
арифметические действия с управлением точностью,
непосредственная компиляция программ,
работа с символьными полями,
непосредственное редактирование текста.

Несмотря на такое глубокое внедрение аппаратных решений, система доведена, в первом варианте, до работающей установки. Уже описан некоторый опыт работы с ней, накопленный в университете штата Айова [63] . Интересно, что для системы характерна многопроцессорная структура. Однако эти процессоры не носят самостоятельного гранично-языкового характера, как в предлагаемой нами схеме, а являются подсобными блоками процесса обработки задачи на входном языке в режиме разделения времени: от ввода задачи с терминала до ее решения. В связи с этим они носят слишком ограниченный характер и утрачивают эффективность в самых важных случаях. Например, блок символьной обработки представлен лишь обычными для традиционных машин операциями со строками, правда, расширены операции редактирования. Арифметический блок связан с неэффективным по памяти представлением программы, обязательной списковой структурой операндов, которая оказывается часто неэффективной при работе с типовыми стандартными массивами. Кроме того, арифметические операции выполняются по последовательной схеме, что не может не привести к потере скорости. Только при работе со сложными динамически меняющимися структурами и данными с малой точностью, такая стратегия может привести к выигрышу в производительности. Однако такой стиль вычислений не характерен для применения высокопроизводительных вычислительных машин.

Мало выражена в аппаратной части SYMBOL возможность организации внешних массивов-файлов, их идентификации, защиты и т.п.

Следует однако отметить, что эта система блестяще иллюстрирует огромные возможности аппаратных решений в рамках процессора универсального назначения.

4.2.5. Сети общего назначения

В заключение остановимся еще на одном, с нашей точки зрения, принципиальном вопросе, а именно: нельзя ли решить поставленную нами проблему с помощью сети специализированных машин, являющейся в целом универсальной системой? Тогда возникает следующий

вопрос: какова должна быть специализация объединяемых в такую сеть машин? Ответ на эти вопросы может быть таков: да, создание сети из функционально-специализированных машин (где функциональная специализация понимается в смысле описанной в предыдущей главе "общеобразовательной школы") является в идейном смысле подходом, эквивалентным предлагаемому нами. При этом наиболее существенным моментом является именно функциональная специализация отдельных машин, а не способ их объединения. Мы предлагаем объединять несколько (а именно, пять) функционально специализированных процессоров в рамках одной машины, учитывая необходимость обеспечения эффективной связи между ними, вместо создания сети автономных функционально-специализированных машин, считая, что при большой интенсивности обменов между отдельными частями системы, неизбежной при функциональной специализации, когда для решения большинства задач требуется привлечение всех частей системы, немаловажную роль будет играть пространственная близость или разобщенность этих частей.

Что касается сетей из проблемно-специализированных машин под отдельные проблемно-ориентированные языки программирования, то для создания из них действительно универсальной эффективно работающей сети, необходимо было бы вводить в сеть новые машины всякий раз при появлении принципиально новых проблем, решение которых с помощью старого набора языков скорее всего оказалось бы не эффективным, а в ряде случаев и не возможным. Такая сеть должна была бы разрастаться с неизбежным при этом усложнением общего управления. Этот подход кажется нам не оправданным.

5. ЗАКЛЮЧЕНИЕ

В данной работе предложена схема вычислительной машины, позволяющая резко и сбалансировано поднять уровень входного языка, приблизив его к естественному. Одновременно с повышением уровня языка могут быть достигнуты сразу три основные цели, которые оказываются противоречивыми при программном решении:

1. Качественно новый уровень устойчивости языка.
2. Существенное повышение эффективности.
3. Открытие новых резервов внедрения аппаратных решений.

Выделение указанных пяти процессоров является логически, методически и физически обоснованным. Они логически необходимы как безусловные компоненты всякой способной к развитию системы обработки информации. Они разумны в методическом плане, так как позволяют для каждого в отдельности процессора, независимо от других, поднять языковую границу как можно выше в пределах общеобразовательных знаний. Наконец, оправдано их физическое разделение, так как не исключено, что их можно будет реализовать на различных физических принципах и элементах. Кроме того, как уже указывалось, на отдельных процессорах можно более разумно решать вопросы оптимизации, надежности, диагностики и т.д.

Конечно, задача создания вычислительной системы на описанных принципах вовсе не проста и не лишена недостатков, так, например:

1) Задача правильного выбора входного языка каждого процессора является чрезвычайно трудной и ответственной. Разделение на процессоры еще больше усложняет ее.

2) Неизбежно определенное дублирование функций в каждом из процессоров.

3) При решении определенных смесей задач, некоторые процессоры могут простаивать и быть балластом установки.

4) Для возможности внесения мелких доделок или усовершенствований придется, вероятно, оставлять "старые" средства в одном или нескольких процессорах.

Вскроются, конечно, и дополнительные сложности. Однако результаты довольно продолжительной работы (около 3 лет) в этом направлении являются весьма обнадеживающими.

ЛИТЕРАТУРА

I. Дж.Айлиф "Принципы построения базовой машины". Перевод с английского под редакцией И.Б.Задыхайло. Библиотека кибернетического сборника. Изд. "Мир", 1973, Москва.

2. Ю.М.Баяковский "Тенденция развития вычислительной техники и программирования (обзор)": Изд.ИПМ АН СССР, М., 1971.

3. А.Беркс, Г.Голдстейн, Дж.Нейман "Предварительное рассмотрение логической конструкции электронного вычислительного устройства" Кибернетический сборник № 9. Сборник переводов под редакцией Ляпунова А.А. и Лупанова О.Б. Изд."Мир", М., 1964г.

4. А.П.Будник, Е.В.Гай, Н.С.Работнов, А.В.Климов, В.Ф.Турчин, И.Б.Щенков "Базисные волновые функции и матрицы операторов коллективной модели ядра". Ядерная физика, т.14, вып.2, 1971.

5. С.П.Бычков, В.А.Фисун, С.Н.Флоренцев, Л.К.Эйсымонт СИМУЛА (описание языка и инструкция по использованию компилятора). Изд. ИПМ АН СССР, 1974.

6. Ван Вейнгарден А. (редактор), Б.Дж.Майу, Дж.Э.Л.Пек, К.Г.А.Костер "Сообщение об алгоритмическом языке АЛГОЛ-68". Ж."Кибернетика" № 6 (1969) и № 1 (1970), Киев.

7. В.М.Глушков, А.А.Стогний, И.Н.Молчанов "Алгоритмический язык малой электронной цифровой вычислительной машины МИР", Том II книга I, Изд."Наукова думка", Киев, 1971.

8. Глушков В.М. и др. "Вычислительные машины с развитыми системами интерпретации". Изд."Наукова думка", Киев, 1970.

9. У.И.Дал, Б.Мюрхауг, К.Нюгорд "СИМУЛА-67 - универсальный язык программирования". Изд. "Мир", М., 1969.

10. Джермейн К. "Программирование на IBM/360". Перевод с английского под редакцией В.С.Штаркмана. Изд. "Мир", М., 1973.

11. Э.В.Евреинов, И.В.Прангишвили "Цифровые автоматы с настраиваемой структурой (однородные среды)" Изд."Энергия", М., 1974.

12. И.Б.Задыхайло, С.С.Камынин, Э.З.Любимский "Вопросы конструирования вычислительных машин из блоков повышенной квалификации" Препринт ИПМ АН СССР № 68, М., 1971.

13. И.Б.Задыхайло, А.Н.Мямлин, Е.И.Котов, А.Г.Красовский, В.К.Смирнов "О повышении эффективности символьных преобразований." Сб. "Методы построения трансляторов на базе метаалгоритмического языка" (в печати).

I4. И.Б.Задыхайло, А.Н.Мямлин, В.К.Смирнов, Л.К.Эйсымонт "Об аппаратной реализации языка для описания объектов на уровне понятий". Сб. "Искусственный интеллект. Итоги и перспективы". М., МДНТП им.Ф.Э.Дзержинского, I974.

I5. И.Б.Задыхайло, Л.М.Немировская, Л.А.Поздняков "Об одной модели быстрой справочной системы, использующей расслоение памяти". Часть I. Препринт ИПМ АН СССР № 44, М., I974.

I6. Иванова О.Л. "Анализ особенностей работы вычислительных машин с массивами". Дипломная работа. Факультет ВМ и К МГУ, М., I974

I7. Лавров С.С. "СНОБОЛ-А" Язык для обработки строк". М., ВЦ АН СССР, I968.

I8. А.Н.Мямлин, В.К.Смирнов, "Входной язык вычислительной машины с магазинной памятью". Ж."Кибернетика" № 6, I967.

I9. П.Наур (под редакцией) "Алгоритмический язык АЛГОЛ-60" Пересмотренное сообщение. Изд. "Мир", М., I965.

20. Поздняков Л.А. "Об одной модели быстрой справочной системы, использующей расслоение памяти". Часть 2. "Схема использования предлагаемой модели, учитывающая размещение переполняющих ключей". Препринт ИПМ АН СССР № 46, М., I974.

2I. РенделлБ., Л.Рассел, "Реализация АЛГОЛа-60". Перевод с английского под редакцией В.М.Курочкина. Изд."Мир", М., I967.

22. Д.Саксон "КОБОЛ". Изд."Статистика", М., I970.

23. В.Ф.Турчин "Базисный РЕФАЛ". Описание языка и основные приемы программирования. (Методические рекомендации)". М., ЦНИПИАСС, I974, Фонд алгоритмов и программ для ЭВМ (в отрасли "Строительство"), Вып.У-33.

24. В.Ф.Турчин. Программирование на языке РЕФАЛ. I. Неформальное введение в программирование на языке РЕФАЛ. Изд.ИПМ АН СССР, М., I97I.

25. Универсальный алгоритмический язык РД/I. Перевод с английского под редакцией В.М.Курочкина. Изд."Мир", М.

26. Флоренцев С.Н., А.А.Хромов, Л.К.Эйсымонт "Транслятор с алгоритмического языка ФОРТРАН ЦЕРН в Ассемблер, написанный на РЕФАЛе". Сб."Математическое обеспечение АСУ", МДНТП им.Ф.Э. Дзержинского, I972.

27. Abrams, P.S, An APL Machine, Stanford Electronics Lab., Tech. Rent. No 3, February I970.
28. Barnes G.H., et al., ILLIAC IV Computer, IEEE Trans. on Comp. vol. C-I7, p 746 (I968).
29. Barnes G.H., The use of ILLIAC IV, Compcon, I972.
30. Bashkow T.R., Sasson A., Kornfeld A., System Design of a FORTRAN Machine, IEEE Trans. on Electronic Comp., vol EC-I6 I967.
31. Chesley C.D., Smith W.R., The Hardware-Implemented High-Level Machine Language for SYMBOL, Proceeding SJCC I97I.
32. Christense C., AMBIT/2 (programming language7, Wakefield, Applied Data Research, Mass, I970.
33. Glushkov V.M., Ignatyev M.B., Myasnikov V.A., Torgashev V.A., Recursive machines and computing technology, IFIP Congress 74, Stockholm august 5-I0, I974.
34. Criswold R.E., Poage J.F., Polonsky I.P., The SNOBOL 4 Programming Language, Prentice-Hall, Englewood Cliffs, N.J, I968.
35. Criswold R.E., Poage J.F. and Polonsky I.P., The SNOBOL 4 Programming Language second edition, Prentice Hall, I97I.
36. Dennis J.B., Programming generality, parallelism and computer architecture, IFIP Congress 68 (Software 2), Edinburgh, August I968.
37. Down H.R., Real-time algorithms and data management on ILLIAC-IV, Compcon, I972.
38. Fenstel Edward A., On the advantages of tagged architecture, IEEE Trans. on Comp., No 7, I973.
39. Goldstine H.H., J.von Neumann, On the principles of large scale computing machines, I946. J.von Wenmann, Collected Works, vol.5, I-33, Pergamon, I96I.
40. Hintz R.G. and Tate D.P., Control Data STAR-I00 Processor Design, Compcon, I972.
41. PL/I Language Specification, revised ed., IBM Systems Reference Library, Form No.C 28-657I I966.
42. Computer Hardware and Architecture (Session III), IFIP Congress 74, Stockholm, august 5-I0, I974.
43. Jones P.D., Implicit Storage Management in the Control Data STAR-I00, Compcon, I972.

44. Tverson K.E., A Programming Language, Wiley, New York, I962.
45. Kilburn T., Morris D., Rohl J.S. and Sumner F.H., A system design proposal, IFIP-68, North-Holland, Amsterdam, I969.
46. Knuth D.E., An empirical study of Fortran programs, Software-Practice & Expirience, v.I, April-June, I97I.
47. Krider I.D., STAR-A De-Education Problem, Compcon, I972.
48. Kuck David J., ILLIAC-IV Software and Application Programming, IEEE Trans. on Comp., vol.C-I7, p.758 (I968).
49. Linsey C.H., Making the Hardware Suit the Language. Algol 68 Implementation, North Holland, J.E. Peck (Ed).
50. McCarthy, J. et all, LISP I.5. Programmers Manual, MIT Press, Cambridge, Mass. I962.
5I. Organick E.I., Computer System Organization. The B 5700/B 6700 Series, Academic press, I973, New York and London.
52. Organick E.I., The Multics System. An Examination of its Structure, The Mit Press Cambridge, Massachusetts, and London, England.
53. Requa J.E., STAR-A system Programmer's View, Compcon, I972.
54. Rice, R. A project overview,Comcon, I972.
55. Rice R., Smith W.R., Symbol- A major Departure from classic software dominated von Neumann Computing Systems, Proceeding SJCC,I97I.
56. Ruggiero J.F. and Coryell D.A., An auxiliary processing system for array calculations, IBM System Journal No 2,I969.
57. Scarrott G.G. and Iliffe J.K., The Basic Language Project, IFIP Congress 68, Edinburgh, august I968.
58. Schid E., Rechenzeitenvergleich bei Digitalrechner, Computing, VS, I970.
59. Shapiro Michael D., A Snobol machine: a Higher-level language processor in a conventional hardware framework, Compcon, I972.
60. Sugimoto A., PL/I Reducer and Direct Processor,Proc. ACM Nat. conf.I969, pp 5I9-538.
6I. Sumner F.H., MU-5- an assessment of the design, IFIP congress 74, august 5-I0, I974, Stockholm Sweden - preprint.
62. Thurber K.J., Myrna J.W., System Design of Cellular APL computer, IEEE Trans.on Comp, v. C-I9, N4, April I970.
63. Zingg R.J. and Richards H., Operational experience with Symbol, Compcon,I972.

64. Г.Катцан, Вычислительные машины системы 370, пер.с англ. под ред. В.К.Левина и Л.Д.Райкова. "Мир", Москва, 1974.
65. Dickinson R.V., Orr W.K., System Ten - a new approach to multiprogramming, Fall Joint Computer Conf., 1970.
66. Purcell C.J., The Control Data STAR-100 - Performance measurement, Compcon 74, pp.385-387.
67. Batcher K.E., STARAN parallel processor system hardware, Compcon 74, pp.405-410.
68. Watson W.J., Carr H.M., Operational experiences with the TI Advanced Scientific Computer, Compcon 74, pp.389-397.
69. Fletcher J., OCTOPUS communication structure, Computer networks from minis trough maxis - are they real, Digest of papers, (Compcon 73) pp.21-23.
70. Riley W.B., Computer networks are taking on the heavy-weight computations, Electronics, May 2, 1974.
71. Aschim F., Data base networks - an overview, Management Information, v.3 (1974), No1.
72. Wedberg G.H., Hanschild L.W., The General Electric network monitor system, IFIP Congress 74, august 5-10, 1974, Stockholm Sweden - preprint.

И.Б. Задыхайло, Е.И. Котов, А.Н. Мямлин, Л.А. Поздняков, В.К. Смирнов. "Вычислительная система с внутренним языком повышенного уровня". Редактор В.К. Смирнов. Корректор Л.А. Поздняков.

№ Т-07945 от 29.04.75г. Заказ № 3160. Тираж 150 экз. Формат бумаги 60X90, 1/16. Объем 2,7 п/л. 2,1 уч. изд. л. Цена 15 коп.

055(02)2 N

Отпечатано на ротапринтах в Институте прикладной математики АН СССР Москва, Миусская пл. 4.

Цена 15 коп.