АРХИТЕКТУРА ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ С ПРОГРАММИРУЕМОЙ СТРУКТУРОЙ
(Вычислительные системы)
1980 год Выпуск 82

УДК 681.3.06

# ПРИНЦИПЫ ПОСТРОЕНИЯ СИСТЕМЫ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛИТЕЛЬНЫХ СТРУКТУР НА СПЕЦИАЛИЗИРОВАННЫХ МИКРОПРОЦЕССОРАХ

В.Ф.Гузик, А.И.Питерский

## Введение

Для эффективной организации вычислений в многопроцессорных вычислительных комплексах [1] необходимы новые средства организации и управления как проблемными процессами пользователей, так и системными процессами в комплексах. Поскольку системные процессы организуются на том же оборудовании, что и проблемные, то время на подготовку задач пользователей в общем времени работы комплекса увеличивается. Для устранения отмеченного недостатка необходимо предъявить новые требования к функциям систем программирования разрабатываемых вычислительных комплексов, в том числе параллельных многопроцессорных структур на специализированных микропроцессорах [2], являющихся естественным развитием однородных цифровых интегрирующих структур [3].

Трудности, возникающие при эксплуатации интегрирующих структур на цифровых интеграторах, в частности при программировании задач, поставили вопрос о построении новых проблемно-ориентированных средств вычислительной техники с программным обеспечением, совместимым с программным обеспечением ЭВМ общего назначения.

Так, при программировании задач для цифровых интегрирующих структур можно переводить исходные функциональные зависимости в инверсную польскую запись или другую форму бесскобочной записи, расчленять их на элементарные составляющие и строить схемы решения задач по системам уравнений Шеннона для элементарных составляющих, имеющихся в библиотеке стандартных подпрограмм, с последующим переводом систем уравнений в коммутационную структуру [4].

Применяемые методы программирования интегрирующих структур при разработке программного обеспечения сопровождаются значительными трудностями, основные из которых связаны с необходимостью организации коммутации микропроцессоров по полному графу, автоматизации расчета масштабных соотношений и начальных значений, а также оптимизации полученной объектной программы. Следовательно, программное обеспечение многопроцессорных структур специализированного типа нуждается в упрощении программирования задач и в повышении эффективности использования ресурсов многопроцессорных структур [5].

Целью данной работы является рассмотрение принципов построения системы программного обеспечения для многопроцессорных структур, основных особенностей программирования и организации системных процессов в них.

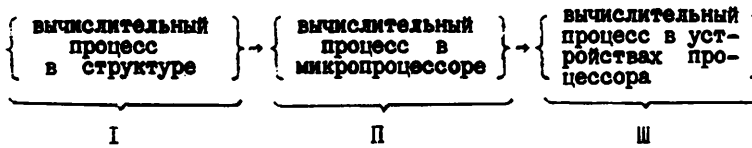
Решение данной задачи наиболее целесообразно провести с помощью развитой операционной системы ЭВМ общего назначения методом кросс-разработки. Другими словами, наиболее эффективно разработку программного обеспечения можно выполнить, используя ЭВМ общего назначения на языке программирования высокого уровня [6].

## §1. Структура, состав и назначение системы программного обеспечения многопроцессорных структур

В основу построения системы программного обеспечения положены принципы модульности, иерархичности и инвариантности описания программных и аппаратных средств многопроцессорных структур, а также реализуемость в аппаратуре базовых конструкций языка программирования.

Принцип модульности предполагает наличие в системе программного обеспечения специализированных по назначению (или реализованных аппаратно) модулей, выполняющих системные функции: трансляцию исходной задачи, загрузку и т.д.

Принцип иерархичности предполагает наличие в системе программного обеспечения определенной зависимости между программными модулями. Так, например, модуль транслятора состоит из сканера, синтаксического анализатора, генератора объектного кода, которые представляют собой функционально законченные подпрограммы. Кроме того, иерархия существует в организации системных и проблемных процессов:




I-й процесс организуется с помощью информационной и управляющей программ; II-й - с помощью алгебры операторов [7]; III-й - организуется в устройствах, входящих в состав операционного автомата процессора с помощью какого-либо метода численного интегрирования по Стилтьесу.

Переход от первого процесса ко второму производится при трансляции входной программы в объектный программный модуль; переход от второго к третьему организуется путем синтеза внутренней структуры процессора на определенных аппаратных средствах.

Принцип инвариантности описания программных и аппаратных средств предполагает представление модулей программного обеспечения и аппаратуры многопроцессорных структур на определенном языке спецификации (например, в форме Бэкуса) так, чтобы их настройка и функционирование не зависели от способа реализации - программного или аппаратного.

Структура системы программного обеспечения приведена на рис.I. В состав системы входят следующие модули: кросс-компилятор, загрузчик, диспетчер, программа символьного дифференцирования, программа расчета начальных данных, программа выбора оперативного базиса, модель структуры на высокоуровневом языке и формирователь микрокоманд операционного автомата микропроцессоров.

Система программного обеспечения предназначена для работы с языком программирования высокого уровня, построенного на основе арифметических и рекуррентных соотношений, имеющего полный набор управляющей логики: простую структуру типа do-od , предикативную структуру типа if-then-else-fi , итеративные структуры типа while-do и repeat-until , распределительную структуру типа case-of . Кроме того, в языке заложены принципы структурного программирования, позволяющие непосредственно реализовать базовые конструкции языка на микропроцессорах.

Теоретической основой построения языка программирования является алгебра схем Шеннона, или Т-операторов, с помощью которых исходные задачи представляются в операторном виде [7]. Оператор T для непрерывной на отрезке $[a,b]$ и $l-1$ раз дифференцируемой

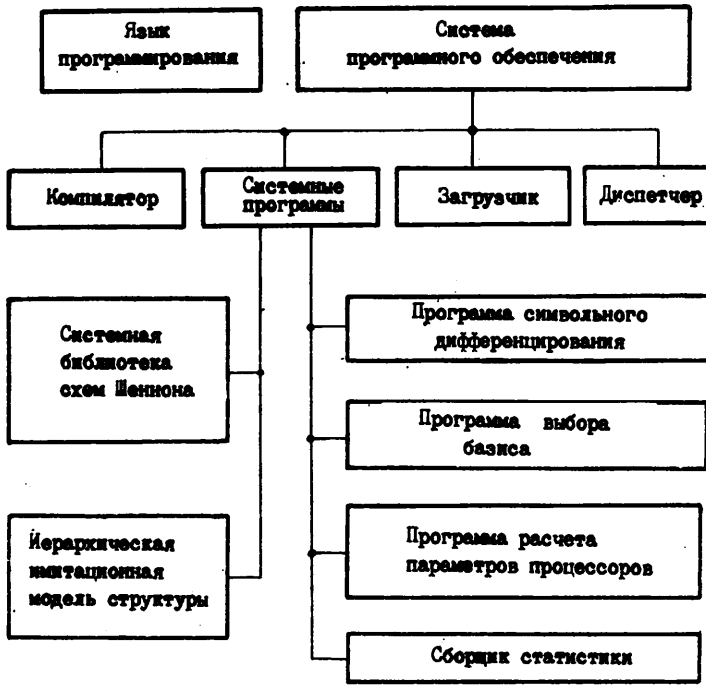



Рис.I. Структура системы программного обеспечения.

функции $f(x)$ , определяется как многозначное отображение $T: f(x) \to \to E_{f(x)}$ , где $E_{f(x)}$ - множество решений системы дифференциальных уравнений Шеннона для этой функции; $f(x)$- главное значение оператора; $E_{f(x)} \setminus \{ f(x)\}$ - множество его побочных значений.Для класса непрерывных задач операторный базис строится по специальному алгоритму [7], с помощью которого оптимальным образом представляется исходная задача.

Каждый микропроцессор реализует один из базисных операторов путем перестройки внутренней структуры. Таким образом, при выборе базисного набора Т-операторов решается вопрос о функциональной полноте микропроцессоров в данном классе задач [7]. В системе программного обеспечения программа выбора операторного базиса реализуется на основе алгоритма, осуществляющего выбор базиса за число шагов меньшее, чем при полном переборе, так как для определения

базиса используется матрица инцидентций семейства множества $E = \{E_{f_1}, E_{f_2}, \ldots, E_{f_n}\}$ при условии $\forall f_i (f_i \in E) \rightarrow (f_i \in E_{f_i}, |E_{f_i}| \geq 1)$.

Дадим краткую характеристику программам системы. Кросс-компилятор отображает множество терминальных цепочек входного языка программирования в множество терминальных цепочек объектного кода. Он состоит из следующих логических частей: сканера (лексического анализатора), синтаксического анализатора, генератора внутреннего кода (семантического анализатора), генератора объектного кода и информационных таблиц. Внутренняя форма исходной программы в кросс-компиляторе имеет следующий вид: ⟨оператор⟩, ⟨операнд⟩, ⟨операнд⟩, ⟨результат⟩. Объектный модуль представляет собой список соединений микропроцессоров структуры и содержит дополнительную информацию о переключении вычислительного процесса с одной ветви на другую при выполнении логических условий.

Программа символьного дифференцирования каждой функции $f$ из заданного класса ставит в соответствие множество $E_f$ – решение системы уравнений Шеннона для этой функции. Алгоритм этой программы использует таблицу производных основных элементарных функций. В этой таблице аргументом является имя функции, а значением – ее первая производная. При дифференцировании используются следующие правила: вынесение константы за знак производной, равенство производной от суммы сумме производных, закон дифференцирования сложной функции и т.д. Найдя заданную функцию в таблице, программа ставит ей в соответствие ее производную и получает в конечном итоге дифференциал для этой функции. Далее программа выделяет из дифференциала новую функцию, находит в таблице ее аналог и таким образом получает вторую производную исходной функции. Процесс продолжается до тех пор, пока не будет получена система Шеннона для заданной функции.

Автоматизация расчета начальных значений в операционном автомате микропроцессора является важной ступенью в процессе подготовки задачи к решению. Расчет начальных значений осуществляется в два этапа. На первом – на основе списка соединений формируется массив начальных значений в микропроцессорах. На втором этапе с использованием встроенных функций системной библиотеки операционной системы ЭВМ общего назначения рассчитываются массивы начальных значений для операционных автоматов, входящих в микропроцессоры.

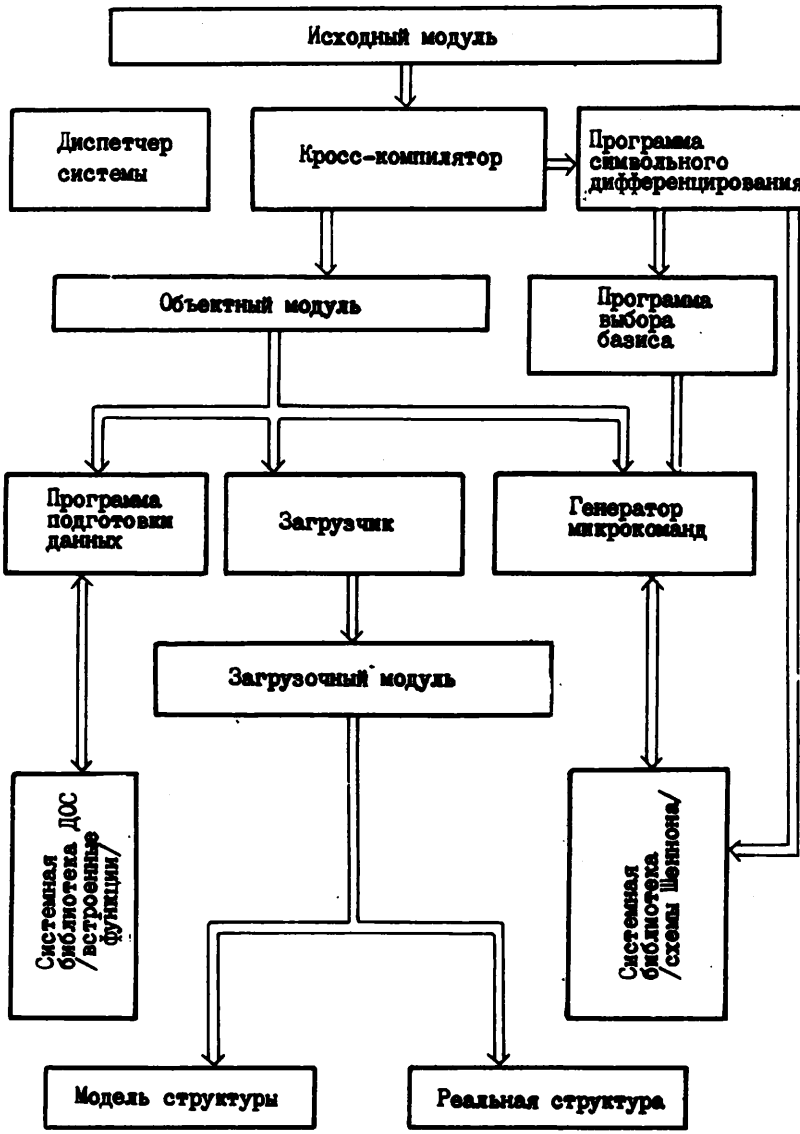



Рис. 2. Взаимодействие модулей программного обеспечения.

Загрузчик подготавливает объектный модуль и инициирует его выполнение на многопроцессорной структуре [8]. В соответствии со списком соединений он выполняет функции коммутации микропроцессоров и настройку всех величин объектного модуля, зависящих от физических адресов микропроцессоров. При замене реальной многопроцессорной структуры имитационной моделью загрузчик выполняет функцию коммутации микропроцессоров, связывая необходимые программы модели. В силу специфики данного типа структур, за счет структурной реализации вычислительного процесса загрузчик получается намного проще, чем в ЭВМ общего назначения.

Основной задачей диспетчера (управляющей программы) в системе программного обеспечения являются управление вызовами кросс-компилятора, загрузчика и других системных модулей, а также сбор статистической информации при работе системы - определение времени работы каждого системного модуля и времени прохождения программ пользователей и всей системы в целом. Кроме того, через диспетчер происходит передача информации между системными модулями. Поскольку диспетчер предназначен для управления работой только обрабатывающих программ (компилятора, загрузчика, модуля выбора базиса и т.д.), то для обработки других системных программ, которые подготавливают аппаратно-программные средства к работе, требуется самостоятельное управление. Этим достигается децентрализация управления в системе программного обеспечения и во всей структуре в целом, оправдываемая, например, тем, что частота использования обрабатывающих программ с диспетчером гораздо выше, чем остальных программ.

Рассмотрим взаимодействие программ в системе (рис.2). В режиме настройки исходные задачи подаются на вход кросс-компилятора. Список функций, встречающихся в этих задачах, поступает с выхода компилятора на модуль символьного дифференцирования, в котором каждой функции $f(x)$ ставится в соответствие множество $E(f)$. Полученные множества передаются в модуль выбора операторного базиса и в системную библиотеку. Базисный набор Т-операторов из модуля выбора базиса передается в модуль формирования микропрограмм, в котором на микропроцессорах структурным образом реализуются все функции из множества. На этом режим настройки заканчивается.

В режиме решения исходная задача, записанная на входном языке, поступает на вход кросс-компилятора. Объектный код, генерируемый в компиляторе, поступает на входы модулей соответственно подготовки начальных данных, формирования микропрограмм и загрузки.

Начальные данные, рассчитанные с помощью системной библиотеки операционной системы ЭВМ, и полученные микропрограммы реализации функциональных зависимостей подаются на вход загрузчика, который на основании этой информации, а также объектного модуля, поступающего с выхода компилятора, формирует загрузочный модуль. Далее полученная программа пользователя загружается в аппаратуру или при наличии модели структуры пересылается в рабочую зону памяти модели.

## §2. Аппаратная реализация базовых конструкций языка программирования

Если аппаратные средства многопроцессорных структур содержат в качестве коммутационных устройств микропроцессоров регистровые элементы [9], то возможна непосредственная реализация базовых конструкций языка программирования в микропроцессорах на этапе подготовки задач к решению. Это достигается тем, что наряду с функциями коммутации, регистровый элемент может вычислять логические функции от двух переменных в базисе $\{\neg, \vee, \wedge\}$, а совокупность таких элементов позволяет реализовать произвольную логическую функцию от n переменных [10]. Следовательно, на регистровых элементах при реализации базовых конструкций языка программирования можно воспроизводить логические условия P(x) для языковых структур типа while-do, if-then-else-fi, repeat-until , при этом часть элементов имитирует входы и выходы структур вышеперечисленных типов и простой последовательности do-od .

На рис.3 приведен пример реализации на микропроцессорах выражения while P(x) do F(y), где F(y) - некоторая функциональная зависимость, P(x) - логическое условие организации итеративной структуры. Блок, вычисляющий F(y) , представляет собой набор микропроцессоров, операционные автоматы которых подключаются к регистровым элементам и осуществляют коммутацию микропроцессоров между собой для вычисления F(y) пока предикат P(x) истинен. Блок вычисления предиката P(x) выполнен на регистровых элементах, отключенных от операционных автоматов. Последние вычисляют логическое условие P(x) в базисе $\{\neg, \vee, \wedge\}$, причем РЭ1 имитирует вход в итеративную структуру, РЭ2 - выход из структуры. При значении предиката P(x) = "ложь" на РЭ2 подается управляющее слово, которое отключает выход $Y_1$ и подключает $Y_2$ к общей схеме решения задачи, как бы завершая данный цикл и приступая к новой ветви вычи-

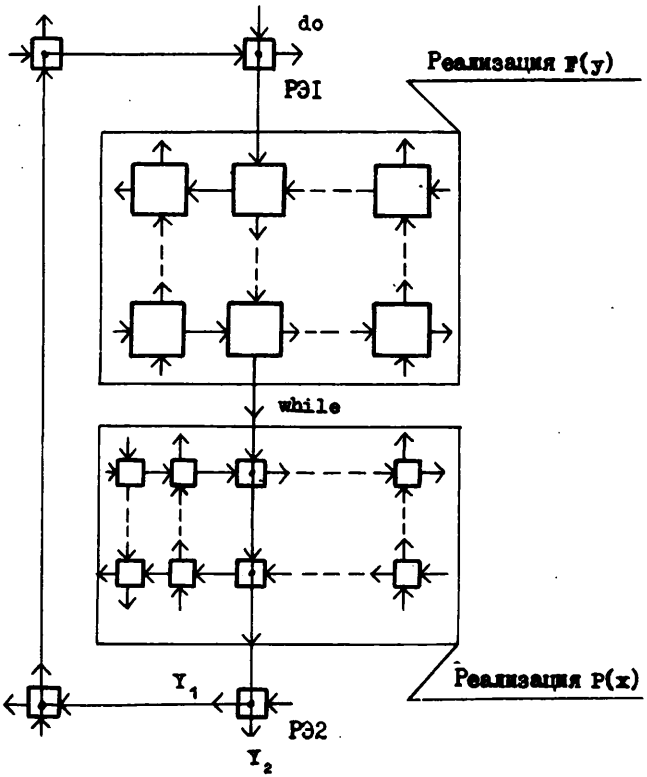



Рис. 3. Реализация конструкции типа while - do.

слений. Кроме конструкций языка программирования, из регистровых элементов можно создавать устройства для выполнения системных функций программного обеспечения путем соответствующей коммутации регистровых элементов между собой.

## Заключение

Применение метода кросс-разработки для создания системы программного обеспечения многопроцессорных структур специализированного типа позволяет снижать трудоемкость разработки и реализовать систему, совместимую с операционными системами ЭВМ общего назначения. Кроме того, появляется возможность использования данной системы для проектирования структуры микропроцессоров на ЭВМ при наличии соответствующих моделей и эмуляторов.

Первый вариант системы программного обеспечения был реализован в рамках дисковой операционной системы ЕС ЭВМ с использованием инструментальной машины ЕС-1020 на языке программирования ПЛ/I. Приведем характеристики производительности системы с использованием программы регистрации системного времени, необходимого на подготовку к решению задачи управления движением шагающего аппарата.

Уравнения, описывающие закон движения робота, имеют вид:

$$r_1^{(i)}(t) = \int_0^t [\omega_3 r_2^{(i)}(t) - \omega_2 r_3^{(i)}(t) + u_1^{(i)}(t) - V_1]dt + r_1^{(i)}(t_0);$$

$$r_2^{(i)}(t) = \int_0^t [\omega_1 r_3^{(i)}(t) - \omega_3 r_1^{(i)}(t) + u_2^{(i)}(t) - V_2]dt + r_2^{(i)}(t_0);$$

$$r_3^{(i)}(t) = \int_0^t [\omega_2 r_1^{(i)}(t) - \omega_1 r_2^{(i)}(t) + u_3^{(i)}(t) - V_3]dt + r_3^{(i)}(t_0);$$

$$\Delta x_1 = [l_0 + l_1 \sin\alpha_2 + l_2 \sin(\alpha_2 - \alpha_3)]\sin\alpha_1 - \tilde{x}_1;$$

$$\Delta x_2 = [l_0 + l_1 \sin\alpha_2 + l_2 \sin(\alpha_2 - \alpha_3)]\cos\alpha_1 - \tilde{x}_2;$$

$$\Delta x_3 = l_1 \cos\alpha_2 + l_2 \cos(\alpha_2 - \alpha_3).$$

Время трансляции задачи составляет 68 сек, время подготовки числовых данных - 35 сек, время загрузки - 25 сек. Для сокращения времени работы системы программного обеспечения по подготовке задачи к решению необходимо использовать более производительную ЭВМ. Так, например, использование ЭВМ типа ЕС-1022 позволяет сократить системный цикл в два раза.

## Литература


1. МАРЧУК Г.И., КОТОВ В.Е. Модульная асинхронная развиваемая система - МАРС. -Новосибирск, 1978. (Препринт/ВЦ СО АН СССР; №87).

2. КАЛЯЕВ А.В., ГУЗИК В.Ф., ПИТЕРСКИЙ А.И. Специализированный микропроцессор для параллельных интегрирующих вычислительных структур. -Автоматика и вычислительная техника, 1979, №1, с.76-81.

3. КАЛЯЕВ А.В. Теория цифровых интегрирующих машин и структур. -М.: Сов.радио, 1970. - 472 с.

4. ЛУКИЕНКО В.И., ДРОВЯННИКОВ А.Я. Автоматизация программирования цифровых интегрирующих структур. -В кн.: Автоматизация программирования. Киев, Наукова думка, 1974, с.108-121.

5. ГУЗИК В.Ф. Математическое обеспечение модульных цифровых вычислительных структур. -Изв. Северо-Кавказского научн. центра высшей школы. Серия технических наук, 1977, №4, с. 23-25.

6. BASS Ch., BROWN D. A perspective on microcomputer software.-Proc.of the IEEE,1976,v.64, N 6,p.54-73.

7. ГУЗИК В.Ф. Построение базиса в операторном пространстве для модульных цифровых вычислительных структур. -Изв. АН СССР. Техн.кибернетика, 1978, № 3, с.129-134.

8. БАРРОН Д. Ассемблеры и загрузчики. -М.: Мир, 1974. - 78с.

9. КАЛЯЕВ А.В. Плоские однородные автоматические коммутационные регистровые структуры. - Кибернетика, 1975, №3, с.1-10.

10. ГУЗИК В.Ф., ПИТЕРСКИЙ А.И. О построении интегрирующих вычислительных структур на специализированных микропроцессорах с расширенными логическими возможностями.-В кн.: Теория автоматов и ее приложения. Варна, 1979, т.2, с.445-452.



Поступила в ред.-изд.отд.
10 ноября 1979 года