Российская академия наук
Сибирское отделение
Институт систем информатики
им. А. П. Ершова

В. А. Евстигнеев

ОБЗОР ДЕЯТЕЛЬНОСТИ НОВОСИБИРСКИХ УЧЕНЫХ В ОБЛАСТИ ПРОГРАММИРОВАНИЯ (ПО МАТЕРИАЛАМ КОМИССИИ ПО СИСТЕМНОМУ МАТЕМАТИЧЕСКОМУ ОБЕСПЕЧЕНИЮ ККВТ АН СССР)

Препринт
83

Новосибирск 2002

Настоящая публикация основана на материалах Координационного Комитета по вычислительной технике АН СССР, связанных с заседанием его Комиссии по системному математическому обеспечению. Это заседание было посвящено обсуждению итога работ новосибирской школы системного и теоретического программирования за более, чем двадцатилетнюю историю ее существования. При рассмотрении итогов работы школы затрагивалась и вся ее история, но главное внимание уделялось результатам 70-х и планам на 80-е. Перспективы, намеченные на 80-е годы, в общем были верными. Эта публикация очень хорошо дает срез состояния работ новосибирской школы на начало 80-х годов и в этом отношении является хорошим и естественным дополнением к издаваемому сборнику "Становление новосибирской школы программирования (мозаика воспоминаний)".

Работа поддержана грантом РГНФ № 02-05-12010в и издается на средства этого фонда.

И. В. Поттосин

Siberian Division of the Russian Academy of Sciences
A. P. Ershov Institute of Informatics Systems

V. A. Evstigneev

OVERVIEW OF WORKS OF NOVOSIBIRSK SCIENTISTS IN PROGRAMMING (BASED ON MATERIALS OF THE COMMISSION ON SYSTEM SOFTWARE OF THE COORDINATING COMMITTEE ON COMPUTING MACHINERY OF THE USSR AC.SCI.)

Preprint
83

Novosibirsk 2002

This issue is based on materials of the Commission on system software of the Coordinating Committee on Computing Machinery of the USSR Ac.Sci. One of its sessions was devoted to works of the Novosibirsk school of system and theoretic programming over the period of more than 20 years. When discussing the activities, the main attention was paid to the results of the 70-ties and future plans for the 80-ties. Perspectives for the 80-ties turned out to be quite properly determined. This preprint gives us a good presentation of works of the Novosibirsk school at the beginning of 80-ties and in this aspect it can be considered as a successful and natural supplement to the collection of papers "Formation of the Novosibirsk school of programming (mosaic of memoirs)".

The work is supported by grant № 02-05-12010в of the Russian Scientific Humanitarian Foundation.

I. V. Pottosin

1. ВВЕДЕНИЕ

К началу деятельности Координационного комитета по вычислительной технике (ККВТ АН СССР) в Новосибирском научном центре сложился мощный коллектив специалистов по информатике и программированию, работавших в ВЦ СО АН СССР под руководством А. П. Ершова (в то время члена-корреспондента АН СССР), а также в НФ ИТМ и ВТ АН СССР. Кроме того, большие коллективы сложились в Институте математики СО АН СССР, в Институте автоматики и электрометрии СО АН СССР и других. В дальнейшем, говоря о новосибирских ученых, мы будем иметь в виду только Вычислительный центр СО АН СССР и в меньшей степени Новосибирский филиал ИТМ и ВТ АН СССР, поскольку только они принимали участие в работе ККВТ и Комиссии по СМО. (Заметим здесь, что Комиссию по элементной базе возглавлял также ученый из Академгородка академик А. В. Ржанов.)

Новосибирские ученые участвовали в работе Пленумов ККВТ, заседаний комиссий, рабочих групп и целевых подгрупп, на которых обсуждались текущее состояние и перспективы различных направлений в программировании и информатике. Заседания Комиссии и ее рабочих групп формировали общественное мнение по тем или иным вопросам вычислительной техники (в частности по персональным компьютерам, ЭВМ пятого поколения, технике ЕС ЭВМ и пр.), теоретического и системного программирования, ППП и др.

Кроме того, на заседаниях Комиссии по СМО рассматривались отчеты ведущих институтов АН СССР, в которых программирование было одним из ведущих направлений деятельности, а также отчеты республиканских АН. В настоящей работе мы покажем становление, развитие и состояние работ в Вычислительном центре СО АН СССР (по материалам Комиссии по СМО) на время проведения 6-го заседания Комиссии, проходившего в Новосибирске в мае 1981 г. С докладами тогда выступили заместитель директора ВЦ СО АН СССР В. Е. Котов и директор НФ ИТМ и ВТ АН СССР Г. Д. Чинин. Ниже приводится полный текст доклада В. Е. Котова и сокращенный доклад Г. Д. Чинина в том виде, в каком это было отражено в материалах Комиссии. Тексты докладов не корректировались, изменена только рубрикация доклада В. Е. Котова и исправлены явные погрешности стиля и опечатки.

2. В. Е. КОТОВ. РАЗВИТИЕ РАБОТ ПО ПО В ВЦ СО АН СССР

2.1. Введение

Работы по теоретической и прикладной информатике в СО АН СССР были начаты в связи с созданием в 1958 году отдела программирования в Институте математики с вычислительном центром СО АН СССР, ядра будущего отделения информатики ВЦ СО АН СССР. Первой широко известной работой, которой заявил о себе новый коллектив, было создание языка высокого уровня АЛЬФА и оптимизирующего транслятора для него. С начала зарождения работы ВЦ СО АН СССР по информатике объединяли и сочетали теоретические исследования и практические проекты. Такое соединение фундаментальных работ и их приложений, характерное для проектов ВЦ СО АН СССР, стимулировало, с одной стороны, постановки и решения перспективных проблем, а с другой, определяло принципиальную ценность практических разработок.

С 60-х годов исследования по информатике в ВЦ СО АН СССР шли широким фронтом. В них сочетались такие взаимодействующие направления, как теория схем программ, параллельное программирование, теория и практика трансляции и оптимизации программ, создание операционных систем и архитектура вычислительных средств. Развитие и интеграция исследований привели к появлению крупных проектов ВЦ СО АН СССР, в которые включились не только большие коллективы программистов внутри ВЦ СО АН СССР, но и ряда сотрудничающих организаций, в первую очередь Новосибирского филиала ИТМ и ВТ АН СССР, коллектив которого зародился внутри ВЦ СО АН СССР. Первым примером такого большого комплексного проекта был проект АИСТ, в результате которого была создана многомашинная вычислительная система и программное обеспечение (ПО) развитой системы разделения времени для нее.

В конце 70-х годов возникли мощные комплексные проекты по созданию архитектуры перспективных вычислительных средств и современного ПО для них — проекты Вычислительного Центра Коллективного Пользования (ВЦКП СО АН СССР) и модульной асинхронной развиваемой системы (МАРС). Эти проекты, также как и проект создания методологии многоязыковых систем программирования (БЕТА), следуя традициям ВЦ СО АН СССР, включают в себя и теоретическую проработку фундаментальных проблем, создание экспериментальных систем

и, наконец, разработку практических систем.

В последнее время (1981 г.) в ВЦ СО АН СССР появились важные направления исследований по новым проблемам информатики — верификации и синтезу программ, применению естественного языка для контактов с ЭВМ, созданию программного и технического обеспечения информационной деятельности, применению ЭВМ в школьном образовании.

2.2. Теория схем программ

Исследования по математическим моделям вычислений традиционно занимают важное место среди работ по информатике, ведущихся в Вычислительном центре. Начавшись в 60-х годах с изучения схем Янова, они развились дальше в сторону усложнения моделей и расширения круга исследуемых проблем. В настоящее время для различных подклассов стандартных, рекурсивных и параллельных схем ведутся исследования по алгоритмическим проблемам, по разработке систем эквивалентных преобразований, по сравнительной и структурной схематологии. Итоги первых этапов развития теории схем программ, и в частности достижения ВЦ в этой области, суммированы в монографиях "Введение в теорию схем программ" (В. Е. Котов) и "Введение в теоретическое программирование" (А. П. Ершов).

В теории стандартных (т.е. алголоподобных) схем программ важное место занимают исследования алгоритмических проблем пустоты (зацикливается ли схема при любой интерпретации), тотальности (останавливается ли схема при любой интерпретации) и эквивалентности.

Рассматриваются классы стандартных схем программ, которые задаются базисами, т.е. множествами исследуемых операторов и тестов. Построен класс схем программ с двумя переменными с неразрешимой проблемой пустоты, который минимален в том смысле, что эта проблема разрешима в любом подбазисе данного базиса. Исследовано влияние структуры базиса на разрешимость проблем пустоты и тотальности. Построен класс монадических схем, имеющих неограниченное число переменных, в котором эти проблемы разрешимы. Главная особенность этого класса — использование в левой части всех пересылок единственной выделенной переменной. Также указан класс монадических схем с двумя переменными с неразрешимыми проблемами пустоты и эквивалентности, но с разрешимой проблемой тотальности.

Изучены проблемы, связанные с распознаванием эквивалентности в

различных классах стандартных схем программ и с разработкой эквивалентных преобразований для этих схем. Дана топологическая классификация вершин стандартных схем и установлены условия совпадения функциональной эквивалентности с логико-термальной эквивалентностью. Также найдены некоторые условия разрешимости комбинированной эквивалентности, представляющей собой замыкание операторной (яновской) и логико-термальной эквивалентностей. Построены полные системы преобразований для операционной и логико-термальной эквивалентностей стандартных схем и для функциональной эквивалентности сквозных схем. Развита методика разметки, позволяющая строить полиномиальные по сложности алгоритмы для распознавания схемных свойств программ. На основе методики разметки разработаны полиномиальные по сложности алгоритмы распознавания операционной и логико-термальной эквивалентностей стандартных схем. Построен каталог оптимизирующих преобразований, выполняемых на уровне стандартных схем, предложен алгоритм элиминации в стандартных схемах программ, сохраняющий операционную эквивалентность и не увеличивающий число переменных исходной схемы.

Для класса рекурсивных схем исследовалось отношение древесной эквивалентности, в частности, был построен полиномиальный по сложности алгоритм распознавания древесной эквивалентности в классе линейных рекурсивных схем. Было обнаружено, что ряд свойств рекурсивных программ, интересных с точки зрения эквивалентных преобразований и оптимизации, можно выразить через отношение древесной эквивалентности. Были построены алгоритмы, основанные на методике разметки, для распознавания таких свойств, как древесная пустота и древесная тотальность, наличие бесполезных и несущественных термов во всем классе рекурсивных схем.

Развивался трансформационный подход к разработке программных процессоров, который основан на наблюдении, что оптимизацию, исполнение, синтез, трансляцию, смешанные вычисления и т.д. можно получить как разные композиции небольшого числа базовых трансформаций, в совокупности задающих трансформационную семантику языка, а в операционном выражении образующих своего рода трансформационную машину. Первые результаты, полученные в данном направлении, показывают перспективный и фундаментальный характер этого подхода.

Обоснование использования в алгоритмических языках различных

программных конструкций требует не только изучения их влияния на разрешимость перечисленных выше алгоритмических проблем. Важно также проанализировать, как сказывается использование таких средств на выразительную силу языков, на возможность преобразования программ, на структуру программ.

Работы в этом направлении были проведены для разных моделей алгоритмических языков. Такими моделями, наряду со стандартными схемами, являются рекурсивные схемы программ (возможно параллельные). Исследовано, как влияет на выразительную силу различных классов схем присоединение к их базисным средствам разнообразных параллельных функций. В частности, показано, что существует бесконечно много различных по выразительности классов параллельных схем, охарактеризованы параллельные функции, обеспечивающие максимальную выразительную силу. Разработаны также эквивалентные преобразования выражений, построенных с применением параллельных функций, получены результаты о регуляризации структуры параллельных выражений в рамках разработанной системы преобразований.

В рамках работ по исследованию структурированности программ даны критерии принадлежности управляющего графа программы трем классам графов, использующих итерации WHILE – DO, REPEAT – UNTIL, FOR – WHILE – DO соответственно. На основе предложенной аксиоматики введены новые классы структурированных управляющих графов программ, в том числе и параллельных. Установлены преимущества итерации FOR – WHILE – DO при достижении информационной структурированности программ. Установлены некоторые условия замкнутости класса структурированных схем Дейкстры относительно модификаций “игнорирования и подсказки”. Доказано, что композиционная структурированность способствует совпадению функциональной и логико-термальной эквивалентностей стандартных схем программ.

Наряду с исследованиями по схемам последовательных программ проводились работы по теории параллельных схем. Естественной моделью для асинхронных процессов являются сети Петри (специальный класс параллельных схем). Разработана алгебра сетей Петри, в которой любая сеть Петри может быть получена при помощи выделенных операций из класса элементарных сетей. В качестве моделей для иерархических асинхронных систем предлагаются структурированные сети, свойства и выразительная мощность которых подробно исследованы.

Установлено, что класс структурированных сетей строго мощнее класса сетей Петри. Это означает, что с помощью структурированных сетей моделируются более сложные взаимодействия между асинхронными процессами. В настоящее время исследования сконцентрированы на изучении фундаментальных свойств сетей, обеспечивающих адекватное описание "хорошо организованных" процессов. Алгебраизация анализа и конструирования сетей Петри позволила применить аналитический аппарат для описания в параллельных программах сложных структур управления.

2.3. Смешанные вычисления

А. П. Ершовым проанализированы некоторые общие тенденции, характерные для информатики 70-х годов, в частности в формулировании в программировании некоторых фундаментальных основ, усиливающих связь программирования с математическими основами.

Убедительным примером фундаментализации программирования является выяснение сущности трансляции на основе понятия смешанных вычислений. Было обнаружено, что объектный код программы можно получить из интерпретатора входного языка, подвергнув его направленной последовательности оптимизирующих преобразований макроподстановок, управляемых деревом разбора входной программы. Было также обнаружено, что этому процессу можно придать характер универсального вычисления, имеющего смешанный характер, поскольку в нем некоторое (частичное) вычисление по программе сопровождается ее переработкой в новую (остаточную) программу. Особенностью смешанного вычисления является то, что оно осуществляет частичное вычисление по заданным аргументам, порождая остаточную программу как функцию незаданных аргументов.

Аналогичные наблюдения, как оказалось, были в разные годы сделаны разными авторами, но не получили развития. В ВЦ СО АН СССР смешанные вычисления были тщательно изучены, в частности были определены правила смешанных вычислений в разных моделях программ: граф-схем с памятью и рекурсивных программ. Были произведены многообещающие эксперименты по применению смешанных вычислений к систематическому построению языково-ориентированных синтаксических анализаторов из универсального анализатора. Было далее выяснено, что смешанные вычисления входят в органическую связь с другими формами манипуляций с программами через так называемые

базовые трансформации программного текста, которые становятся одновременно еще одним способом формализации семантики языков системного программирования и технологической основой конструирования программных процессоров с помощью так называемых трансформационных машин.

Трансформационный подход к программированию привел к потребности изучения инвариантов программ, сохраняющихся при преобразованиях. Были найдены и исследованы некоторые важные классы таких инвариантов. Был выдвинут подход к определению функций, вычислимых в произвольных системах элементарных команд и условий, в котором схемный инвариант берется за определение вычислимой функции, не предопределяя синтаксис ее программы.

2.4. Верификация и синтез

Исследования по верификации, синтезу и распараллеливанию программ относятся к важным новым направлениям работ по информатике.

Наиболее распространенный метод Флойда–Хора верификации программ предполагает выбор языка для аннотирования программ, запись в этом языке выходных условий и индуктивных утверждений (инвариантов цикла), вывод условий правильности с помощью аксиоматической системы и доказательство этих условий. Для компактного аннотирования программ целесообразно применять специальные языковые понятия, которые сопровождаются набором аксиом, используемых для доказательства условий правильности. Этот подход реализован в экспериментальной системе СПРУТ. На вход системы поступает аннотированная программа на языке Паскаль (не использующая файлов и других сложных структур данных). После автоматической генерации условий правильности система пытается их упростить и доказать.

Предложена аксиоматическая система вывода условий правильности в случае паскалевских программ с файлами и множествами, которые ориентированы на следующие классы программ: сортировки одномерных массивов, линейной алгебры, обработки файлов с последовательном доступом.

Предложена методика доказательства частичной правильности программ линейной алгебры, включающая язык спецификаций, стандартную аксиоматическую систему вывода условий правильности, специальную систему вывода для получения условий правильности без инва-

риантов циклов. С помощью данной методики доказана правильность двух программ линейной алгебры: вычисления определителей и решения системы линейных уравнений.

Проведена приближенная верификация большой программы-интерпретатора языка Бейсик, реализованного на языке Паскаль. Даны специальные понятия для аннотирования интерпретатора и проведено формальное доказательство небольшой части условий правильности. В результате стандартного тестирования интерпретатора не было обнаружено ошибок, что подтверждает высокое качество формального аннотирования.

Разрабатывается формальная система правил вывода для статического контроля типов значений переменных в алголоподобных программах, не содержащих описаний переменных. Формальная система получена модификацией системы верификации Хоара и не требует сопоставлений индуктивных высказываний циклам.

2.5. Работы по системному программированию

В 60-х и в первой половине 70-х годов в области системного программирования проводились работы по созданию оптимизирующих трансляторов для языков высокого уровня, систем разделения времени в многопроцессорной обстановке, машинно-ориентированных языков для системного программирования. Ниже перечислены созданные в этот период системы программирования.

1. АЛЬФА — система автоматизации программирования для ЭВМ типа М-20, входной язык АЛЬФА. Объем 50 тыс. команд. Оптимизирующая система с высоким качеством получаемых программ. Эксплуатируется более в чем 100 организациях 37 городов СССР.

2. АЛГИБР — система для получения рабочих программ для ЭВМ БЭСМ-6 на ЭВМ типа М-20, входной язык — АЛЬФА. Объем 70 тыс. команд. Эксплуатируется в 4 городах Советского Союза.

3. АЛЬФА-6 — оптимизирующая система программирования для ЭВМ БЭСМ-6. Входной язык АЛЬФА-6. Объем 150 тыс. команд. Отличается большим набором сервисных возможностей и высоким качеством получаемых программ. Эксплуатируется в 20 организациях 11 городов Советского Союза.

4. АИСТ-0 — экспериментальная многопроцесорная универсальная система коллективного пользования на базе ЭВМ М-220 и “Минск-32”. Объем 100 тыс. команд. Внедрена в г. Кемерово.

5. ЭПСИЛОН-М3 — система программирования для ЭВМ типа М-220. Объем 11 тыс. команд. Эксплуатируется в 16 городах Советского Союза.

6. ЭПСИЛОН-Б — система программирования для ЭВМ БЭСМ-6. Входной язык — язык системного программирования ЭПСИЛОН. Объем 10 тыс. команд. Система находится в производственной эксплуатации.

7. МАКРОЭПСИЛОН — расширение системы ЭПСИЛОН-Б для БЭСМ-6. Включает средства комплексации программ. Объем 20(?) тыс. команд. Система находится в опытной эксплуатации.

8. СЕТЛ — экспериментальная система программирования для теоретико-множественного языка СЕТЛ сверхвысокого уровня. Разработана для БЭСМ-6. Объем 70 тыс. команд. Эксплуатируется в трех организациях.

9. ЛИСП — экспериментальная система для задач символьной обработки. Разработана для ЭВМ БЭСМ-6. Входной язык ЛИСП. Объем 4 тыс. команд. Находится в опытной эксплуатации.

Работа в период 1975–1980 гг. велась по направлениям, описанным ниже.

2.5.1. Разработка многоязыковой системы БЕТА

Общая характеристика системы БЕТА и методология ее разработки

Методика реализации трансляторов в рамках многоязыковой системы БЕТА основана на развитом модульном подходе. Общие принципы построения системы заключаются в следующем:

- Для реализации входных языков выбрана общая система трансляции, включающая этап декомпозиции (перевода на внутренний язык), этап оптимизации программ на внутреннем языке (с возможностью выключения этого этапа, управляемой пользователем) и этап генерации выходной программы.
- Существует единый внутренний язык, являющийся выходным языком этапа декомпозиции и входным языком этапа генерации.
- Этап оптимизации представляет собой универсальный оптимизатор и включает большое число глобальных преобразований, не зависящих от входных языков и осуществляемых над программами на внутреннем языке.

В настоящее время в системе реализованы языки Симула-67, Фортран и Модула. Применение модульного подхода для совместной реали-

зации этих языков показало, что в значительной мере (от 60 процентов текста и выше) трансляторы с входного языка на внутренний строятся из библиотечных модулей. В экспериментальной версии системы реализована генерация на ЭВМ БЭСМ-6. Эта версия снабжена блоком локальной оптимизации.

В полной мере модульный подход был применен к этапу декомпозиции. Следует отметить, что реализация модульного подхода с необходимостью требует принятия определенных стандартов — стандартов на представление и способы работы с основными информационными структурами, стандартов на интерфейсы между частями алгоритмов, стандартов на общую схему алгоритмов в данной модульной системе. Одним из источников модульности этапа декомпозиции была, с одной стороны, фиксация представления общих структур данных (таблиц и основной программы) на всех просмотрах декомпозиции, а с другой стороны, экранирование этого представления от собственно алгоритмов декомпозиции с помощью выделенного набора процедур работы с этими структурами. Другим источником модульности стала фиксация общей схемы алгоритма декомпозиции в виде трех последовательных просмотров для соответственно синтаксического анализа, идентификации и синтеза программы на внутреннем языке.

Для воплощения удобной модульности существенными были возможности, представляемые языками реализации (им был язык ЯРМО), концепция модульности в котором соответствует современным языкам модульного программирования.

Внутренний язык системы БЕТА — это частично интерпретируемый язык, часть операторов которого является операторами с неполной определяемой семантикой. Такой подход кажется естественным для многоязыковых систем, аналогичным БЕТА, т.е. таких, входные языки которых относятся хоть и к широкому (с точки зрения использования) кругу языков, но достаточно близки друг к другу. Круг языков, на которые рассчитывает система, очерчивается ориентацией на последовательное программирование (без глубокого параллелизма), компиляционный характер, отсутствие существенной машинной ориентации, общую (а не распределенную) память, традиционные структуры управления. Все это дает возможность во внутреннем языке выразить полностью семантику большинства всех входных языков.

Основные решения по внутреннему языку (ВЯЗу) системы БЕТА определялись следующими особенностями роли этого языка в системе:

- ВЯЗ является единым выходным языком для этапов декомпозиции входных языков и должен представлять универсальный базис для выражения свойств конструкций всех языков;
- ВЯЗ является входным языком генерации для различным машинных языков, в связи с чем исходная текстовая иерархия конструкции языка должна быть упрощена;
- ВЯЗ предоставляет возможности комплексации программ и включение отладочных действий на этом отладочном уровне, с чем, в частности, связана необходимость иметь внешнее представление ВЯЗа;
- ВЯЗ дает языковый уровень оптимизирующих преобразований. Программа на ВЯЗе представляет собой совокупность соответственно программного массива и таблиц. Таблицы служат для хранения атрибутов объектов ВЯЗа.

Программный массив есть дерево фрагментов ВЯЗа. Висячие вершины дерева фрагментов представляют собой последовательность операторов. Кратко характеризуя типы операторов внутреннего языка, выделим следующие типы:

1. Вычислительные операторы для общих базисных вычислительных операций, среди которых выделены операторы с многомерными операциями — мультикоманды, выражающие коммутативные и ассоциативные операции.
2. Ящики (черные ящики) для представления частично интерпретируемых операций — семантика этих операторов ВЯЗа выражает только отношения этих операторов к другим, необходимые для оптимизации. Полная семантика ящиков раскрывается лишь при генерации.
3. Операторы доступа (к компонентам структур или по указателю), отражающие общую модель адресации.
4. Управляющие операторы вызовов, переходов или условных переходов (в том числе и по переменной метке).
5. Операторы синхронизации, основанной на сигналах и ситуациях.

Наличие таких операторов, как ящики делает ВЯЗ открытым, способным к расширению. Другая возможность, представляемая ящиками, заключается в том, что можно регулировать степень детализации вычислительных операторов, исходя из нужд оптимизации.

В системе БЕТА задача создания оптимизирующих трансляторов решается тем, что этот этап во всех трансляторах представлен уни-

версальным, не зависящим от входных (и выходных) языков оптимизатором. Этот процесс оптимизации преобразует полученные на этапе декомпозиции ВЯЗ-программы, не меняя языкового уровня представления программ.

Большинство глобальных оптимизирующих преобразований требует предварительного потокового анализа — нахождения управляющего и информационного графов программы. В связи с этим выделен специальный предварительный этап потокового анализа программы на ВЯЗе.

Набор выбранных преобразований включает общие, такие как экономия выражений, чистка циклов, удаление несущественных вычислений и т.п., и специальные — ориентированные на некоторые конструкции языка преобразований, в первую очередь преобразования процедур. При выполнении ряда глобальных преобразований применяется факторизация в соответствии с иерархией фрагментов в ВЯЗ-программе. За некоторыми исключениями каждое преобразование реализуется в своем, отдельном просмотре ВЯЗ-программы, причем оптимизация процедур предшествует остальным оптимизациям. Предполагается организовать циклическое повторение последовательности оптимизаций, а сама начально выбранная последовательность может быть изменена на основе опыта эксплуатации.

Реализованная в экспериментальной версии системы БЕТА генерация является машинно-зависимой и предусматривает единственный выходной язык — язык БЭСМ-6. Дальнейшее развитие идей генерации предполагается вести в сторону многоязыковой генерации. Это движение подготовлено уже существующими в системе БЕТА решениями — в первую очередь наличием единого внутреннего языка, а также накопленном при разработке системы СИГМА опытом формального описания машинных языков.

Проведение машинно-зависимой оптимизации требует привлечения потокового анализа, а значит — использования результатов этапа анализа оптимизатора системы БЕТА.

Сама необходимость многоязыковой генерации в значительной мере подкрепляется наличием и развитием мини- и микро-ЭВМ. Перенос накопленного ПО на эти машины предполагает использование кросс-систем программирования, причем такие системы должны быть многоязыковыми.

В рамках проекта БЕТА были проведены исследования по включению отладочных средств. Основные решения, вытекающие из проведен-

ных исследований, представлены ниже.

- Основой для включения отладочных средств являются не входные языки, а ВЯЗ. Это дает возможность ввести единые отладочные средства для всех входных языков.
- Набор отладочных действий включает такие средства, как задание трассировки, динамических проверок, включение заплат и шлейфов, а также отладочных переисполнений при авостных ситуациях.
- Наличие в системе потокового анализа дает хорошую возможность для проверки правдоподобности программ, нахождения маршрутов. Наличие оптимизации дает возможность безболезненно включать в текст программы при декомпозиции динамических проверок семантики, имея в виду, что очевидные — при анализе программы — проверки будут выполнены при оптимизации.

Перевод на внутренний язык

При переводе на ВЯЗ применяется трехсмотровая система трансляции. Первый просмотр производит синтаксический разбор программы, строит таблицы (определения) лексем, преобразует в постфиксную нотацию.

Второй просмотр идентифицирует использующие вхождения лексем и вставляет приведения.

Третий просмотр осуществляет синтез программы на ВЯЗе. Все три просмотра перевода на ВЯЗ используют общую базу данных, называемую промежуточным языком (ПЯ).

На входе просмотра синтаксического разбора получают исходную программу — последовательный файл. На выходе получается программа на ПЯ, файл синтаксических ошибок, файл распечатки.

Служебные процедуры разбора включают прежде всего процедуры чтения входного потока, с редакцией или без нее, обеспечивающие формирование файла распечатки и подготовку контекста для выдачи сообщений об ошибках. Сравнительный анализ входных языков дал возможность выделить несколько общих процедур свертки лексем (идентификаторов, констант, служебных слов).

При использовании языка реализации, представляющего приемлемую эффективность рекурсивных процедур (как это имеет место для языка ЯРМО), прямая реализация синтаксического анализа оказывается не более сложной, чем перепись грамматики для приведения ее в

соответствие с требованиями универсального анализатора. Единственно сложной частью оказывается анализ выражений.

Второй просмотр получает на входе множество лексем и последовательный файл с программным массивом. Он обрабатывает множество лексем, выполняя разного рода транзитивные замыкания, контролирует последовательность атрибутов; переписывает программный массив, заменяя ссылки на имена ссылками на лексемы и вставляя приведения; пополняет файл ошибок. Логика семантической индукции для большинства языков очень проста, хотя этот просмотр в наибольшей степени языково-ориентирован.

Все пользовательские ошибки, нормально контролируемые трансляторами, обнаруживаются на первых двух просмотрах. Третий просмотр и вся дальнейшая обработка транслируемого модуля выполняются только при отсутствии таких ошибок.

Синтез завершает семантическую унификацию разноязычных транслируемых программ. На этом просмотре происходит существенная перестройка программы.

Синтез есть самая сложная часть перевода; правда, благодаря стандартности выхода он располагает весьма развитой библиотекой служебных процедур, обеспечивающих заведение объектов ВЯЗа и установление связей между ними.

Реализация Симулы-67 в системе БЕТА

Из всех языков системы БЕТА Симула-67 наиболее оригинальна. Прежде всего это специализированный язык. Однако его специализированность особая: он содержит полный набор изобразительных средств так называемых "универсальных языков", но многие из них расширены с ориентацией на специальное использование. Далее, указанные расширения придают языку в значительной степени интерпретационный характер, требуют развитой динамической поддержки — и это для таких фундаментальных конструкций как блок, вызов, переход по метке.

Эти особенности Симулы-67 резко контрастируют с принципиальными разработками системы БЕТА, поэтому реализация Симулы-67 сопряжена с особыми трудностями и особенно показательна для возможностей системы.

Симула-67 очень плохо документирована.

В нашей реализации во всех неясных случаях выделяется бесспорный вариант, и об отклонениях от него сообщается как об ошибке. В результате выделяется не весь язык, а пересечение всех вариантов.

Обычно вызовы процедур Симулы-67 программируют почти как запуск сопрограмм, отводя каждому вызову рабочую зону в куче и оставляя заботу об освобождении памяти сборщику мусора. В результате работа средств управления квазипараллелизмом оказывается, пожалуй, даже более эффективной, чем вызов процедур.

Мы проделали эксперимент по такой блочной структуре в рамках БЕТА. Однако число команд, исполняемых при каждом вызове самой регулярной процедуры, оказывается столь велико, что мы предпочли усложнить работу квазипараллелизма, но за счет этого добиться максимальной эффективности гораздо более распространенного и фундаментального процедурного механизма.

Конструкция внутреннего языка гораздо проще, чем конструкция класса в Симуле-67. Поэтому перевод классов на ВЯЗ содержит раскрытие выражения классов в более элементарных понятиях.

Массивы и тесты выносятся в кучу, в рабочих секциях остаются только указатели на них. При отведении им памяти они подвешиваются в список кучевых объектов ближайшего экземпляра блока, ограничивающего их область существования, причем в них отмечается номер поколения (динамическая глубина) этого экземпляра. При передаче агрегата фактическим параметром по ссылке в более мелкую (широкую) область существования на основании сравнения глубин происходит перевешивание объекта в список более широкой области. При уничтожении такой области освобождается память всех объектов, попавших в его список кучевых объектов. Таким образом, возможна чистка кучи и помимо сборщика мусора.

Все только что перечисленные действия программируются на ВЯЗе при синтезе ВЯЗ-программы, а не встраиваются во внутренний язык в качестве примитива.

Реализация языков Модула и Фортран в системе БЕТА

Общая трансляция языка Модула достаточно традиционна и соответствует подходу, принятому в системе. Транслятор совершает три просмотра: на первом просмотре производится лексическая свертка, синтаксический анализ и заполнение таблиц определяющих вхождений; второй просмотр производит идентификацию использующих вхождений и контроль типов; третий просмотр синтезирует программу на ВЯЗе.

Лексический анализ практически полностью покрывается стандартными средствами системы БЕТА. Единственный синтаксис Модула не

создает сложных проблем для анализа. Возможность использовать рекурсивные процедуры в языке реализации позволяет написать синтаксический анализатор, работающий по методу рекурсивного спуска.

Второй просмотр — это наименее унифицированная часть транслятора. Основу этого просмотра составляет модуль ПОИСКЛЕК и процедура ВЫРАЖЕНИЕ. Процедуры, собранные в модуле ПОИСКЛЕК, обеспечивают поиск определяющего вхождения.

Процедура ВЫРАЖЕНИЕ обрабатывает выражение, которое в ПЯ1 представлено в обратной польской записи. С помощью процедур поиска лексемы и подтяжки ссылок она производит идентификацию, а затем контролирует типы на совместимость.

Цель третьего просмотра — синтез программы на внутреннем языке. Задачу синтеза можно разбить на три части:

1) синтез ВЯЗ-объектов,

2) синтез выражений,

3) синтез операторов.

Выходом для третьего просмотра является программа на ВЯЗе, которая может быть передана на фазу оптимизации или генерации.

Опыт, накопленный при разработке системы, овеществлен прежде всего в библиотеке процедур и спецификаций.

Библиотека предоставляет разные возможности для каждого просмотра транслятора. С большим успехом библиотечные процедуры использованы на первом просмотре. Процедуры чтения и лексической свертки практически полностью обеспечивают лексический анализ и позволяют разработчику забыть о таких вещах, как формирование листинга и накопление контекста для выдачи ошибок.

Второй просмотр автоматизирован значительно меньше. Это связано с сильной зависимостью его от языка. На этом просмотре используются только общетехнические процедуры.

На третьем просмотре библиотека используется интенсивней, чем на втором. Очень полезными и удобными оказываются процедуры для работы с ВЯЗом.

В языке Модула имеется понятие параллельного процесса, события, операции над событием. Во внутреннем языке понятия параллельного процесса нет, но имеются средства для организации квазипараллельного исполнения, поэтому для таких действий, как *запуск процесса*, *совершение события*, *ожидание события* на внутреннем языке написаны процедуры, имитирующие работу параллельных процессов с помощью

квазипараллельного механизма.

Модульная структура языка сказывается на алгоритмах всех трех просмотров транслятора. На первом просмотре модуль оформляется как особый блок, его внешние связи выражаются ссылками на имена импортируемых объектов (внутри модуля) и ссылкой на экспортированное имя и на его модуль-хозяина (в охватывающем блоке). Для этих ссылок выделены лексемы особого рода РПОРТ.

На втором просмотре традиционный алгоритм идентификации для языков с блочной локализацией претерпевает некоторые изменения. Во-первых, модуль ограничивает движение вверх по охватывающим блокам; во-вторых, когда лексема найдена, она может оказаться лишь ссылкой, что потребует продолжения поиска в новом направлении.

После того как идентификация сделана, модули становятся ненужными.

По выполненной реализации можно сделать несколько итоговых выводов.

1. Язык Модула хорошо укладывается в схему трансляции, принятую в системе БЕТА, и весьма удобно представляется в виде промежуточного языка.
2. Семантика языка Модула достаточно просто и естественно выражается на внутреннем языке системы.
3. Существующий набор библиотечных процедур является хорошим помощником, освобождает разработчиков от значительной части шаблонной работы и позволяет сосредоточить внимание на нестандартных особенностях реализуемого языка.

Реализация Фортрана в системе БЕТА имела две версии. Первая версия была создана для конкретизации основных понятий, связанных с ним. Недостатком этой версии является изолированность реализации Фортрана от реализации других языков, т.е. оторванность от модульного подхода.

Вторая версия реализуется с учетом модульного подхода и состоит из четырех независимых (в смысле трансляции, загрузки и вызова) модулей, каждый из которых есть просмотр. Привязка к общей схеме трансляции потребовала введения предварительного просмотра. Этот просмотр преобразует транскрипции Фортрана так, чтобы входной текст мог обрабатываться общими процедурами чтения. Модуль предварительного просмотра является уникальным.

Первый просмотр наряду с преобразованием программного массива

в массив на первом промежуточном языке ПЯ1 осуществляет развернутый синтаксический контроль. В силу однопроходности Фортрана на первом просмотре осуществляется также идентификация, связанная с учетом всех определяющих вхождений. Наряду с идентификацией производится и частичный семантический контроль.

Второй просмотр завершает идентификацию, осуществляя семантическую индукцию в выражениях и операторах управления.

В настоящее время реализация Фортрана (во второй версии) завершена в отношении вышеупомянутых просмотров. Собственно синтез не завершен, однако уже существовавшая первая версия предлагает опробованные решения по отображению конструкций Фортрана на конструкции ВЯЗа.

Генерация

Задача генерации в общей постановке: получив программу (модуль компиляции) на ВЯЗе, построить по ней рабочую (выходную) программу для данной вычислительной системы.

Мы реализовали выход на ассемблер (автокод БЕМШ в версии для мониторной системы ДУБНА).

Реализация этой фазы с самого начала включала возможность породить программу с распечаткой сбойной выдачи; во входные языки была включена стандартная процедура распечатки состояния памяти, не оставляющая иных следов, кроме файла распечатки.

Хотя эффективность выходной программы в огромной мере зависит от тщательности, с которой реализована генерация, логическая сложность этой фазы значительно меньше, чем у фазы синтеза ВЯЗ-программы. Это объясняется, с одной стороны, близостью ВЯЗа к машинному уровню, а с другой стороны, отсутствием необходимости передавать фазе оптимизации информацию о возможных использованиях программируемой конструкции.

Фаза генерации начинается с просмотра используемых в программе видов и фрагментов, а также локальных в них переменных (фактически при этом просматриваются лишь заголовки фрагментов, а не весь программный массив). В результате вычисляются относительные адреса и размеры памяти под переменные и блоки, а также, если заказана распечатка сбойной и/или сборка мусора, строится выходной модуль образцов (прототипов), состоящий из одной длинной текстовой константы для управления обходом памяти и печати внешних имен.

Стремясь уменьшить число базовых регистров и “статизировать” адреса, мы заводим свои базы только в блоках для рекурсивных процедур и ветвей.

При всех конфликтах скорости и объема предпочтение отдается экономии кода. Естественно, рабочие переменные, изображенные в ВЯЗе информационными связями, реализованы через магазин, что значительно сокращает код и косвенно — время.

Возможность реализации параметра цикла индекс-регистром должна быть установлена на уровне ВЯЗа. Другой способ, которым оптимизация универсально может улучшить использование индекс-регистров, состоит в выделении, слиянии и сужении базисов — фрагментов внутреннего языка, соответствующих присоединяющим операторам некоторых входных языков.

Блок локальной оптимизации

Блок осуществляет как локальные оптимизации ВЯЗ-программы, так и “чистку” излишних конструкций, возникающих из универсальности алгоритмов синтеза и желания разгрузить алгоритмы оптимизации.

Работа блока осуществляется за один просмотр программного массива. Выполнение локальных оптимизаций осуществляется обращением к одной из доступных извне процедур блока:

– процедуре, реализующей удаление оператора и всех его связей в ВЯЗ-программе;
– процедуре, осуществляющей преобразования одного оператора;
– процедуре, реализующей преобразования последовательности операторов;
– процедуре, выполняющей преобразования с фрагментами.

Реализация некоторого из преобразований создает возможность реализации других преобразований. В связи с этим взаимодействие процедур носит существенно рекурсивный характер, и вызов хотя бы одного преобразования может привести к преобразованию всей ВЯЗ-программы.

Оптимизация

При проектировании блока оптимизации как наиболее важные рассматривались следующие требования:

1) каждое применяемое преобразование должно быть корректным;
2) недопустимость пессимизации;
3) строгая обоснованность как преобразования, так и алгоритмов анализа;

4) универсализация везде, где это возможно.

Прямая работа по реализации оптимизатора системы БЕТА потребовала проведения ряда теоретических исследований по обоснованию анализа и преобразований программ.

Был разработан единый подход с использованием специального вида нумераций операторов программы, позволяющий получать эффективные (по трудоемкости) алгоритмы анализа управляющих связей в программе, включая и те, которые необходимы для построения теней системы БЕТА.

Для обоснования оптимизирующих преобразований, связанных с перемещением или вычеркиванием операторов в программе (например, чистка циклов или втягивание в гамак), была предложена модель, основанная на рассмотрении программы в виде линейной последовательности структурных операторов.

В качестве единого базиса для исследования различных существующих алгоритмов анализа и преобразований программ была предложена модель так называемой крупноблочной программы. Эта модель, получившая название крупноблочной схемы, основана на рассмотрении программы (или ее фрагмента) в виде совокупности структурных операторов, действующих над совокупностью структурных переменных. В рамках указанной модели была решена задача перераспределения памяти, что, в частности, может использоваться для обоснования преобразования программ, связанных с перераспределением памяти не только под статические, но и динамические величины. Также была решена задача одновременного вынесения групп операторов из многократно повторяемого (исполняемого) участка программы (например, тела цикла или рекурсивной процедуры); при этом было показано, что такое одновременное вынесение позволяет производить более полную оптимизацию, чем алгоритмы пооператорного вынесения, которые были разработаны при исполнении линейной схемы.

2.5.2. Разработка общих принципов трансляции и создание систем программирования и систем аналитических преобразований

Теоретические основы обработки программ

Задача теории обработки программ состоит в поиске и изучении фундаментальных элементов и процессов обработки программ, которые лежат в основе тех или иных конкретных видов обработки, таких как исполнение, трансляция, оптимизация, генерация, сборка, адаптация и

анализ.

◇ *Исследования по сравнительной схематологии.* В рамках работ по сравнительной схематологии получены методы и результаты для исследования семантики различных новых программных конструкций, главным образом для анализа их выразительной силы и допускаемых ими эквивалентных преобразований. Эти проблемы изучались для различных функциональных моделей последовательных алгоритмических языков, обогащаемых параллельными функциями. Получены результаты о сравнении выразительной силы различных параллельных функций; найдены максимальные по выразительности функции как в языках, запрещающих рекурсивные определения, так и в языках, допускающих рекурсию; для ряда важных классов функций построены полные по выразительности базисы параллельных функций, для модели рекурсивных схем программ в терминах степеней параллелизма установлено наличие богатой иерархии различных по выразительности параллельных функций. Установлены такие свойства этой иерархии, как бесконечность и неплотность.

Разработаны эквивалентные преобразования, допускаемые рядом параллельных функций. Описан каталог таких преобразований, позволяющих в языке, обогащенном соответствующими средствами параллелизма, производить оптимизацию, трансляцию (там, где это возможно) более мощных по выразительности программных конструкций в менее мощные.

◇ *Преобразования схем программ.* Исследование эквивалентных преобразований приводит к методам и результатам для оптимизации программ, для проверки правильности, контроля и синтеза программ по их спецификациям. Преобразования изучались на двух моделях программ — для стандартных схем и для рекурсивных схем. Были построены функционально полные системы преобразований для логико-термальной эквивалентности стандартных схем и для функциональной эквивалентности сквозных схем. Была развита методика разметки как средства сбора нелокальной информации в интересах преобразований, с помощью которой удалось построить полиномиальный по сложности алгоритм распознавания логико-термальной эквивалентности стандартных схем, а также набор простых алгоритмов оптимизации.

Методика разметки была перенесена и на рекурсивные схемы. Это позволило сформулировать полиномиальный по сложности алгоритм распознавания древесной эквивалентности линейных рекурсивных схем.

На том же методе основаны и алгоритмы распознавания таких свойств, как древесная пустота и древесная тотальность, наличие бесполезных и несущественных термов, которые оказываются очень важными для формулирования эквивалентных преобразований.

Для логических фрагментов программ исследовалась сложность проблемы минимизации размера программы. Оказалось, что в общей постановке это — переборная задача, т.е. она содержит NP-полную подпроблему. Построен довольно эффективный алгоритм для уменьшения размера дерева (не обязательно до оптимального).

◇ *Смешанные вычисления и концепция трансформационной машины.* Исследование смешанных вычислений выявило его связи с рядом фундаментальных понятий теории алгоритмов, его значение для решения некоторых проблем технологического характера. В частности, изучались вопросы применения смешанных вычислений к получению оптимизированных языково-ориентированных анализаторов из универсального. Выяснилось, что многие процессы обработки программ, такие как оптимизация, исполнение, синтез, трансляция, смешанные вычисления и т.д., можно получить как разные композиции небольшого числа базовых трансформаций, в своей совокупности образующих так называемую трансформационную семантику языка, а в своем операционном выражении образующих так называемую трансформационную машину. Эти базовые трансформации сохраняют эквивалентность обрабатываемых программ, так что для корректности программных процессоров (т.е. программ трансформационной машины) достаточно, чтобы они всегда останавливались. Уже первые проработки в этом направлении показывают перспективный и фундаментальный характер этого подхода к программированию.

Средства комплексации в системе АЛЬФА-6

Работа по модернизации системы АЛЬФА-6 выполнялась в рамках совместной работы ВЦ СО АН СССР и НФ ИТМиВТ АН СССР. Целями разработки были:

- включение системы АЛЬФА-6 в мониторную систему ДУБНА;
- обеспечение стандартной системы для получения модулей загрузки системой АЛЬФА-6;
- создание языковых средств, управляющих раздельной трансляцией блоков, составных операторов, описаний процедур (функций) языка АЛЬФА.

Попутно была найдена расширенная семантика для некоторых операторов. Оператор MARG позволяет учитывать ширину листа АЦПУ и определяет расположение страниц вывода. Каналы операторов INPUT и OUTPUT являются едиными и допускают ввод информации, выведенной по OUTPUT. И наконец, оператор LIBRARY является средством динамической загрузки подпрограмм во время выполнения программы.

Отдельно транслируемую процедуру или функцию можно выполнить во время исполнения другой отдельно транслируемой программы. Для этого нужно специфицировать ее описателем внешней процедуры в начале блока и затем использовать в этом блоке как обычную процедуру (функцию) языка АЛЬФА-6. В описателе внешней процедуры указывается также тип процедуры (если она функция) и вид языка, на котором она написана (АЛЬФА-6, Фортран, Алгол-ГДР). Информация от модуля к модулю может передаваться с помощью параметров внешней процедуры или внешней памяти. Общая память задается в АЛЬФА-6 программе аналогично COMMON в языке Фортран и стыкуется с соответствующими COMMON-блоками, определенными в фортрановской программе.

Каждый канал АЛЬФА-6 представляет собой два совмещенных на внешнем носителе последовательных файла, один из которых используется для чтения операторами INPUT транслированных (или выделенных операторами OUTPUT) данных, второй — для записи информации, выводимой операторами OUTPUT. При окончании задачи содержимое второго файла преобразуется в текст и структурируется на страницы согласно семантики операторов OUTPUT и MARG, а затем выдается на АЦПУ. Каждый канал позволяет контролировать потоки вводимых данных и выдачу информации на АЦПУ.

В АЛЬФА-6 имеются два расширения алголовских средств вывода. Первое дает возможность размещать несколько страниц на одном развороте листа. Второе — это возможность управления курсором, а также перераспределение текста на странице, как это делается на видеотоне в режиме off-line. Указанные добавления позволяют кратко записывать выдачу графиков, таблиц, диаграмм и т.д.

Модификации блоков объемом в 13 тыс. команд (около 10 тыс. инструкций БЕМШ) производились разработчиками блоков в течение 1975–76 гг. и составили 14 человеко-лет. Вновь созданные компоненты (блоки 8–10, блок ГД и библиотека служебных подпрограмм) объемом 56,5 тыс. команд (14791 инструкций ASTRA) разрабатывались и отла-

живались двумя программистами с привлечением одного лаборанта с конца 1975 по 1980 гг. Затраты составляют приблизительно 15 человеколет. Кроме того, за период разработки комплексации были выпущены следующие издания:

1. Эскизный проект разработки средств компиляции (Отчет № АТИ–Н129/06.00, 1975. НФ ИТМ и ВТ, 118 с.)
2. Технический проект разработки средств компиляции (Отчет № АТИ–Н145/15.00, 1976, НФ ИТМ и ВТ,)
3. Руководство к пользованию системой АЛЬФА-6 в системе Дубна / Аникеева И.Н. и др. — М.: Наука, 1979.
4. Рабочие материалы, блок-схемы блоков 8–10 и блок ГД.

В настоящее время система АЛЬФА-6 (комплексирующая версия) сдана в производственную эксплуатацию в ГПВЦ СО АН СССР. (Акт о сдаче от 25.06.1980), а также передана в ряд организаций СССР (ВЦ АН СССР в г. Москве, НИИПМ в г. Перми, НИИИ и НИЭП в г. Новосибирске, ЦКБМ в г. Реутове, НИРФИ в г. Горьком, ЦСКБ в г. Куйбышеве, Объединение "Ленинец" в г. Ленинграде, ИММ в г. Свердловске, Институт ядерных реакций в г. Димитровграде, ИПФ и ИТПМ в г. Новосибирске).

Система Сигма

В 1967 г. в ВЦ СО АН СССР был разработан алгоритмический язык Сигма, основной особенностью которого является наличие параметров для создания выходного языка системы. Система Сигма реализована на БЭСМ-6 и предназначена для:

– получения переносимых программных продуктов;
– получения новых проблемно-ориентированных языков;
– быстрого освоения новых вычислительных машин путем описания их в качестве выходных языков системы.

Объектами языка являются константы и переменные. Существует понятие вида, под которым понимается задание разбиения на поля группы подряд стоящих ячеек памяти выходной машины. Поля при описании вида задаются только своей длиной, а не положением внутри ячейки памяти выходной машины. Распределение по ячейкам выполняется транслятором автоматически.

В языке имеется два типа процедур: открытые и замкнутые. Описание открытой процедуры устраняется из текста программы, а каждый вызов ее заменяется на тело процедуры с заменой формальных параметров на фактические. Тело замкнутой процедуры вставляется в

объектную программу один раз на место описания процедуры, а вместо каждого ее вызова вставляется группа макросов. Параметры замкнутой процедуры могут подставляться аргументом, результатом, аргументом-результатом и адресом. Кроме того, в качестве фактического параметра можно подставлять любую группу макросов.

Возможность иметь в качестве выходного языка системы некоторый алгоритмический язык или модуль загрузки позволяет осуществлять сборку отдельно оттранслированных частей Сигма-программы в одну объектную программу.

Макросы языка Сигма делятся на два класса: описанные (вызовы процедур) и неописанные. Неописанные макросы (за исключением небольшого числа служебных макросов, осуществляющих работу с динамическими объектами) в языке не фиксированы; их набор разрабатывается пользователем и задается системе при описании выходного языка с помощью параметров.

Система Сигма включает четыре части:

– монитор,
– программа настройки Сигма-транслятора на конкретные входной и выходной языки,
– универсальный Сигма-транслятор,
– административная система.

Монитор осуществляет все общение пользователя с системой (как в пакетном, так и в диалоговом режиме), обеспечивает хранение и редактирование текстовых программ. Монитор реализован на языке Эпсилон.

Программа настройки предназначена для обработки параметров выходного языка и описания макросов конкретного представления Сигма-языка и записи в Сигма-архив обработанной информации в виде, требуемом для универсального Сигма-транслятора.

Универсальный Сигма-транслятор предназначен для перевода текстов, написанных на заданом конкретном представлении, в тексты на описанном с помощью параметров выходном языке системы. Он состоит из трех просмотров: синтаксического анализа, синтеза программы на внутреннем языке и генератора объектного кода. Между вторым и третьим просмотрами работает программа распределении памяти статическим объектам. Программа настройки и Сигма-транслятор реализованы на языке БЕМШ.

Администрирование системы представляет собой динамическую поддержку системы Сигма и предназначена для работы с динамическими

объектами Сигма-программы во время счета. Административная система создана на языке Сигма.

Помимо административной системы на языке Сигма был реализован ряд тестовых программ. В настоящее время ведется разработка производственного транслятора с Фортрана для ЭВМ Электроника НЦ-80.

В качестве выходных языков системы были описаны следующие языки:

– автокод БЕМШ,
– коды команд БЭСМ-6,
– ассамблер ЕС ЭВМ,
– ассамблер ЭВМ Хьюлетт-Паккард.

В настоящее время в качестве выходного языка описываются коды команд ЭВМ Электроника НЦ-80.

Система Макроэпсилон

Эпсилон является одним из самых первых языков системного программирования. Опыт использования языка показал, что он является удобным инструментом системного программирования. Однако бедность языка в отношении средств управления и отсутствие в нем средств компиляции создавали определенную трудность при выборе его в качестве языка реализации для больших систем.

С 1975 г. началась работа по обогащению языка Эпсилон средствами комплексации и управления. Расширенный язык, получивший название Макроэпсилон, реализован для ЭВМ БЭСМ-6. Реализация представляет собой мониторную систему, позволяющую:

– транслировать с получением программы на макрокоде;
– транслировать с получением модуля загрузки;
– объединять оттранслированные программы;
– объединять Макроэпсилон-программы с программами на любых языках, совместимых с макрокодом;
– выполнять программы.

Система функционирует в рамках ОС Диспак/Диспак.

Программа на языке Макроэпсилон состоит из двух частей: описание действий и описание данных. Каждый объект, входящий в некоторый оператор, должен быть введен описанием объекта.

В языке Макроэпсилон различаются четыре класса объектов:

1) коды,
2) переменные,
3) переменные с начальным значением,

4) процедуры.

С помощью кодов можно задавать кодировку любых объектов, объединять их в множества и организовывать классификацию объектов. Значения переменных перед началом выполнения программы являются неопределенными. К этому классу объектов относятся: скаляр, слово, слог, заголовок списка, элемент списка, заголовок списка слов, элемент списка слов.

Слова используются для записи разнородной информации в одну ячейку. Слова разбиваются на слоги.

Список есть упорядоченная совокупность элементов списка.

Список слов есть упорядоченная совокупность элементов списка слов.

С каждым списком и с каждым списком слов связан его заголовок.

Слог слова, элемент списка и элемент слова могут быть указаны с помощью индекса, который может быть как простым, так и вычисляемым.

К переменным с начальным значением относятся константы и метки. Значения констант в начальный момент задаются с помощью описания. Метку имеет смысл использовать в качестве переменной лишь тогда, когда она метит команду. При выполнении программы константы и метки могут изменяться. Это дает возможность формировать информационные слова, формировать и переадресовывать команды.

Каждая процедура в языке задается описанием. По способу выполнения процедуры делятся на замкнутые и открытые. В первом случае фактические параметры подставляются значением, а во втором — только именем.

В языке отсутствует блочная структура и локализация переменных. Даже тела процедур не являются блоком и их формальные параметры имеют смысл глобальных переменных.

Ввиду того, что язык Макроэпсилон не предназначен для описания задач вычислительного характера, в нем присутствуют лишь простые арифметические выражения и все действия выполняются над целыми неотрицательными числами.

Набор операций языка включает арифметические операции, логические операции, операции отношения, операции сдвига, операции извлечения адреса.

В языке определены оператор присваивания, оператор перехода, условный оператор, оператор цикла, команда, пустой оператор, оператор останова, а также предусмотрены специальные средства для объедине-

ния отдельно оттранслированных программ.

В конце 1976 г. транслятор с языка Макроэпсилон был передан в опытную эксплуатацию. Этот период совпал с появлением в рамках ОС Диспак универсальных средств хранения информации, например таких, как “хамелеон”, и достаточного количества терминалов, в частности “Видеотон-340”. Практически отпала необходимость ведения перфокарточного хозяйства, поэтому был разработан язык управления заданиями, который сам был реализован на языке Макроэпсилон.

Выбор директив языка управлением заданиями происходил так, что если некоторая директива имела аналог в управляющих перфокартах, то она получала такое же имя. Директивы сопровождаются параметрами и краткими комментариями. Ниже следует перечень основных директив с краткими описаниями.

1) ЬПЕЧАТЬ — выдается построчная распечатка исходного текста;
2) ЬТРАНСЛИРОВАТЬ — исходный текст программы транслируется;
3) ЬБЕМШ — эта директива является разновидностью директивы ЬТРАНСЛИРОВАТЬ. По этой директиве текст, полученный после трансляции на макрокод, записывается в архив;
4) ЬКОНЕЦ — эта директива означает, что кончился текст исходной программы;
5) ЬКОМПЛЕКСИРОВАТЬ — происходит объединение отдельных загрузочных модулей в одну программу, готовую к выполнению;
6) ЬВЫПОЛНИТЬ — происходит загрузка программы в оперативную память и ее выполнение.

Система Макроэпсилон включает следующие сегменты:

– программа системного ввода,
– монитор,
– транслятор с языка Макроэпсилон на Макрокод,
– программа системного вывода,
– транслятор с Макрокода,
– редактор внешних связей (РВС),
– программа выборки.

Последние размещают готовую к работе программу в оперативной памяти и инициируют выполнение программы.

За период с 1978 г. система Макроэпсилон передана в некоторые организации Советского Союза. В течение этого времени язык Макроэпсилон использовался в качестве инструментального языка для макроас-

семблера системы ЛИТТЛ, преобразователя внутренних таблиц транслятора с языка Паскаль во внешнее представление списков языка ЛИСП, макропроцессора для языка ЛИТТЛ, программы стыковки СЕТЛ-транслятора с системой проверки правильности утверждений (СПРУТ), монитора системы Макроэпсилон, информационно-поисковых систем и т.д.

Базовая система аналитических преобразований

◇ *Входной язык.* Системы аналитических преобразований можно рассматривать как мощный инструмент для решения задач с хорошо определенным вычислительным процессом, но требующих большого рутинного труда по проведению громоздких выкладок, а также для решения задач, очень чувствительных к потере точности при численном решении.

Работа по базовой системе аналитических выкладок проводилась в рамках совместной работы ВЦ СО АН СССР и НФ ИТМиВТ АН СССР.

Разработка базового языка универсальной системы аналитических преобразований явилась естественным продолжением работ по аналитическим преобразованиям на ЭВМ, проводимых в рамках системы разделения времени АИСТ-0.

Кратко опишем базовый язык аналитических преобразований АУМ (Аналитическая Универсальная Машина).

Основным действием этого языка является вычисление S-объекта. S-объект рассматривается как комплекс, состоящий из обязательного основного объекта, который может быть правилом, блоком, компонентой или процедурной и факультативной последовательностью условных и специфицирующих предложений.

Совокупность правил, введенных пользователем в рабочую область системы, называется рабочей областью. Правила рабочей области делятся на две части: пользовательскую и системную части. Системная часть является неподвижной частью рабочей области. Правила, наполняющие подвижную часть, делятся на открытые, закрытые и локальные.

Локальными являются правила, введенные через предложения спецификаций. Они используются только в момент вычисления основного правила. Жизнь локальных правил и рабочей области определяется основным правилом.

Закрытыми являются правила, связанные либо со специальными переменными, либо с некоторой именованной группой правил. Эти пра-

вила участвуют в процессе вычисления, когда имена групп правил и управляющих переменных явно задаются в преобразующих командах.

Все прочие правила рабочей области называются открытыми.

Процесс вычисления выражения контролируется пользователем с помощью специального сигнала 'ФВ' (фаза вычислений), который в стандартном состоянии является именем вектора с компонентами, равными номерам выполняемых фаз, один, два, три, четыре, пять, шесть, соответственно для фаз предпостановки, раскрытие скобок, общее применение, постподстановки, упрощение, реконструирование.

Пользователь может изменять фиксированный процесс вычисления выражения либо установкой специальных значений вектора ФВ, либо меняя значения флагов преобразования и значения специальных переменных.

Процедуры служат для определения собственных преобразований пользователя, отсутствующих в общем процессе аналитического вычисления, и их многократного использования.

Блоки служат для заданий именований группы преобразующих правил. Правила, входящие в блок, не изменяют рабочую область пользователя.

Команды служат для выполнения наиболее часто встречающихся преобразований пользователя, для выполнения преобразований с выражениями специального вида, а также для выполнения системных преобразований. К командам общего назначения относятся команды *вычислить, подставить, раскрасить, упростить, коэффициент, сравнить, применить, выбрать, часть, показать, убрать, обозначить.* Специальные команды обеспечивают манипулирование с полиномиальными и рациональными функциями и содержат команды *псложс, пвыч, пумн, пдел, пстеп, пнод, порядок, рацслж, рацдел, рстеп, рацвыч, рацумн, рвычисл, факторизовать, чдробъ, решить, решитьур.* Системные команды предназначены для запуска системы, уничтожения правил из рабочей области, связи с файлами и системой программирования.

Поскольку общий процесс вычисления рассматривается как последовательная трансформация исходного выражения до тех пор, пока к преобразуемому выражению нельзя будет применить ни одного правила из множества правил. Алгоритм применения правил к выражению рассматривается как главная составляющая процесса вычисления.

◇ *Состав системы.* Базовый язык АУМ реализован в экспериментальной системе АУМ.

Система АУМ написана на языке высокого уровня ЯРМО. Общий объем системы составляет около 21 тыс. ЯРМО-предложений, в том числе 7 тыс. предложений составляет текст аналитического вычислителя, 1 тыс. — модуль подготовки таблиц, 13 тыс. — тексты библиотеки преобразований.

В систему АУМ входят входной анализатор, модуль декомпозиции, комплексный аналитический вычислитель (КАНВА), библиотека преобразующих модулей, выходной транслятор и планировщик. Входной транслятор выполняет лексический и синтаксический анализы входной строки языка, передает управление модулю декомпозиции, если строка правильная, и входному транслятору, если в строке была допущена ошибка. Модуль декомпозиции преобразует S-объект в структурную форму. Структурированная величина — мультикоманда для выражения, сегмент для правила, поступает на вход аналитическому вычислителю. Аналитический вычислитель производит аналитическое вычисление S-объекта, тесно взаимодействуя в процессе вычисления с библиотекой преобразующих модулей; вычисленный результат поступает на вход выходному транслятору, который показывает его пользователю.

Выражение в системе АУМ может быть представлено в мультикомандном виде с помощью общей системы представления данных либо в виде рекурсивной канонической формы, если выражение является аргументом полиномиальной или рациональной команды.

Системную поддержку аналитического вычислителя обеспечивают редакционные действия с рабочей областью, работу с файлами, действия по получению рабочих версий программы, запуск и сохранение нужных версий, обеспечение символьно-численного интерфейса и способы реализации допускаемых системных расширений.

В процессе опытной эксплуатации отрабатывалась схема взаимодействия комплексного аналитического вычислителя с библиотекой преобразований, исследовались различные схемы внутреннего применения модельных правил к выражению и исследовалась эффективность алгоритмов для различных представлений рациональных функций. Были получены сравнительные оценки эффективности для обработки полиномов с разными видами представлений. В процессе опытной эксплуатации были приняты решения о введении ряда дополнений в язык АУМ.

2.6. Программное обеспечение ввода-вывода информации на графические устройства

К настоящему времени в лаборатории машинной графики разработаны три графические системы:

- система математического обеспечения графических устройств СМОГ,
- система графических архивов и модулей СИГАМ,
- система программного обеспечения графического диалога СПО ГД.

Первая система широко распространена и эксплуатируется на различных ЭВМ, обеспечивая вывод информации практически на любое графическое устройство координатного типа. Остальные две системы созданы сравнительно недавно и опыт их эксплуатации невелик. Сразу отметим, что возможности системы СИГАМ покрывают возможности системы СМОГ, но обе системы не предоставляют средств графического ввода. Система СПО ГД разработана для ЭВМ БЭСМ-6, сопряженной с комплексом АРМ-Р/СМ-3, ее подпрограммы расположены на обеих ЭВМ (БЭСМ-6 и СМ-3) и обеспечивают ввод-вывод информации на устройства, входящие в состав комплекса. Системы СПО ГД и СМОГ совместимы, т.е. информация, подготовленная системой СМОГ, может быть визуализирована на экране графического дисплея, а изображение, созданное средствами СПО ГД, может быть выведено на графопостроители с помощью подпрограмм СМОГ.

Универсальная графическая система СМОГ в настоящее время эксплуатируется на трех ЭВМ: БЭСМ-6, ЕС-1052 и Барроуз-6700. С точки зрения пользователя, все три версии идентичны, достаточно просты для изучения и использования. Различие систем связано прежде всего с особенностями систем программирования на указанных ЭВМ и возможностью вывода информации на тот или иной графический прибор, имеющийся в институте.

Наиболее полно отработана версия СМОГ БЭСМ-6, предоставляющая средства вывода информации из любой системы программирования на любое имеющееся графическое устройство: графопостроители, фотопостроитель, устройства микрофильмирования. Вся система может быть разделена на две части: собственно графическая система и система управления графическим выводом. Подпрограммы первой части обеспечивают построение символов, различных графиков, векторных полей, карт изолиний и т.д. Результат работы этих подпрограмм

кодируется и хранится в архиве системы, недоступном программисту.

Система управления графическим выводом предназначена для перекодировки информации в управляющие коды конкретных графических устройств и вывода этой информации либо на магнитные носители, либо непосредственно на устройство. Такая организация системы позволяет, во-первых, разделить во времени процессы подготовки данных и их вывода на графические устройства, во-вторых, достаточно просто включать в систему новые графические устройства. По версиям СМОГ БЭСМ-6 и СМОГ ЕС имеются инструкции по их установке и отладке для системных программистов.

2.7. Применение ЭВМ в издательском деле

В области применения ЭВМ при подготовке печатных изданий разрабатывается редактирующая система в ОС ЕС ЭВМ для издания книг фотонабором. Эта система, называемая САПФИР, находится в настоящее время в стадии комплексной отладки и подготавливается к сдаче заказчику — Первой образцовой типографии им. Жданова — ее первая очередь, позволяющая осуществлять автоматизированный набор формульного и табличного текстов. Система САПФИР предназначена для работы в пакетном режиме, позволяющем редактору вести правку издания по распечаткам, выдаваемым ЭВМ.

Отметим, что чрезвычайно бедные возможности воспроизведения текстов на периферийных устройствах ЕС ЭВМ в настоящее время вносят дополнительные большие осложнения как для разработчиков системы, так и, в еще большей степени, для ее пользователей. Дело в том, что неизобразимые при распечатке особенности шрифтового оформления и способов расположения сложных текстов буквально “загромождают” текст распечатки управляющими символами команд. Так при наборе книги “АЛГОЛ-68” в ее синтаксической части оказалось, что управляющих знаков в распечатке более 200 процентов от видимых знаков, остающихся после выпуска текстов.

В этом году (1981 г.) начато выполнение темы “Создание информационно-вычислительной системы РУБИН”, являющейся заданием № 11 проблемы 0.80.02 программы важнейших научно-исследовательских работ ГКНТ. Научные работы по проекту РУБИН начаты в ВЦ СО АН СССР по инициативе издательства “Правда” в 1977 г. и проходили, в основном, в качестве соцобязательств.

За эти годы был проведен системный анализ процессов подготов-

ки и выпуска газеты “Правды”, изучены принципы работы редакции и структура ее информационного обеспечения.

Была разработана генеральная схема создания и развития системы РУБИН, утвержденная в 1979 г. главной редакцией газеты. Генеральная схема системы РУБИН предусматривает несколько необычный порядок разработки и внедрения системы: от периферийной сети автономно работающих терминалов к центральному информационно-справочному комплексу, а затем уже периферия объединяется с центром. Такой порядок запуска системы позволит наилучшим способом “вписать” ее в сложившийся сложный порядок работы редакции и создать условия для ее освоения сотрудниками за время, необходимое для разработки информационно-справочных систем большого объема на центральном вычислительном комплексе.

Работы по проекту РУБИН ведутся в кооперации с несколькими организациями, возглавляемыми издательством “Правда”; в этой кооперации ВЦ СО АН СССР осуществляет научное и конструкторское руководство. В ВЦ СО АН СССР ведется проработка ПО периферийных терминалов, специально разрабатываемых для проекта РУБИН совместно с заводом Мера-Блоне в ПНР.

2.8. Искусственный интеллект

Лаборатория искусственного интеллекта ВЦ СО АН СССР ведет работы, группирующиеся по трем основным направлениям (в этих работах лаборатория искусственного интеллекта возглавляет комплексный коллектив, в который помимо самой лаборатории входят целевая группа лаборатории вычислительных методов НИС НГУ и группа проблемной лаборатории при ВЦ СО АН СССР).

А. Взаимодействие с ЭВМ на естественном языке.

Б. Средства математического обеспечения задач искусственного интеллекта.

В. Системы управления недетерминированного типа.

Внутри этих направлений работы организованы по проектам. Направление, связанное с естественным языком, включает следующие проекты:

1. Экспериментальная разработка формальной модели русского языка. Задачей проекта является разработка формального описания, охватывающего достаточно содержательное подмножество русского языка и ориентированного на создание на его базе лингвистического обеспечения автоматических систем анализа и синтеза текста.

Работы по проекту ведутся в сотрудничестве с группой прикладной лингвистики МГУ.

2. Проект ВОСТОК охватывает проблематику, связанную с представлением знаний и моделированием процесса понимания.

- Разработка формального аппарата для представления содержания сообщения на семантическом уровне и представления знаний о предметной области.
- Создание экспериментальных программных систем, реализующих ограниченные модели понимания сообщений.

В рамках проекта была реализована экспериментальная вопросно-ответная система ВОСТОК-0, включающая интерфейс на ограниченном естественном языке, простой логический вывод и встроенную модель времени. Начата разработка экспериментальной системы ВОСТОК-1.

Исследования по проекту ВОСТОК ведутся в сотрудничестве с лабораторией вычислительной лингвистики МГУ.

3. Проект ЗАПСИБ объединяет работы по созданию математического обеспечения диалога человек—ЭВМ на естественном языке. Эти работы группируются по трем основным направлениям.

- Разработка модульной серии лингвистических процессоров ЗАПСИБ, предназначенных в первую очередь для анализа обращения пользователя к прикладной базе данных. Процессоры ЗАПСИБ строятся по общей схеме, основные модули которой реализуются в нескольких последовательно расширяющихся версиях. Такой принцип позволяет выбирать конфигурацию процессора в соответствии с требованиями заказчика. Модули процессора включают ряд “точек настройки”, упрощающих адаптацию системы к конкретной области применения. К настоящему времени создан ряд экспериментальных версий лингвистического процессора (СЕТЛ, БЭСМ-6) и прототип прикладного процессора ЗАПСИБ—ЕС (ПЛ/1). Ведется опытное внедрение процессора ЗАПСИБ—ЕС в нескольких предметных областях.
- Разработка программного комплекса СТЕНД, предназначенного для введения процедуры адаптации процессоров ЗАПСИБ к предметной области (формирование, заполнение и редактирование словаря, определение и редактирование правил лексического и основного анализа, спецификация предметной области — базы данных пользователя).
- Разработка на основе комплекса СТЕНД технологического паке-

та, ориентированного на создание диалоговых процессоров (ограниченный естественный язык, проблемно-ориентированные языки сверхвысокого уровня). Входящие в состав СТЕНДа набор модулей и системные средства программирования, настройки и отладки позволяют разработчику выбирать конфигурацию процессора, с помощью средств высокого уровня определять функции входящих в нее специализированных модулей (разные уровни анализа, преобразования, генерация выходного представления, диалог и т.д.), вести тестирование и отладку, создавать новые модули.

4. Проект BUMP имеет целью разработку универсального многовариантного программного процессора типа "снизу-вверх", управляемого системой правил-продукций. Процессор предназначен для использования в качестве базового матобеспечения при создании специализированных систем анализа и переработки сложных информационных структур, а также логического вывода. Проект предусматривает следующие этапы:

– разработка нескольких версий универсального процессора;
– использование этих версий в системах ЗАПСИБ (в качестве процессора анализа/синтеза) и ВОСТОК (в качестве процессора логического вывода;
– разработка проблемно-ориентированных версий; сейчас, в частности, завершается первый вариант процессора символьной обработки (взятие неопределенных интегралов от элементарных функций).

Планируется создание интеллектуального монитора для ППП и включение его в состав варианта пакета СТЕНД, ориентированного на создание диалоговых интерфейсов для ППП.

5. Проект СЕТЛ (реализация теоретико-множественного языка сверхвысокого уровня) в ВЦ СО АН СССР с самого начала был ориентирован на практичность. Система строилась как развиваемая, с повышенной модифицируемостью. Ее компилирующая часть представляет собой упрощенную систему построения трансляторов, а абстрактная машина основана на легко пополняемом наборе специализированных операций, позволяющих сравнительно эффективно отобразить входные конструкции.

Ядро системы (написанное на языке Эпсилон) за последние годы пополнилось значительным набором сервисно-системных программ (диа-

логовый процессор, архив, библиотека процедур, отладочные компоненты и др.), построенных уже средствами самого языка СЕТЛ.

Более пяти лет система СЕТЛ-БЭСМ является основным инструментом программирования в лаборатории искусственного интеллекта ВЦ СО АН СССР. Это оказалось полезным столкновением нового языка и области информатики, для которой характерны рассмотрение особо сложных нетрадиционных алгоритмов и необходимость быстрого изучения целого спектра различных решений. При этом сложилась своеобразная двухступенчатая технология создания прикладных систем, когда стадия проектирования неразрывно связана с макетированием ключевых фрагментов средствами языка СЕТЛ. По имеющимся оценкам это позволило уменьшить суммарное время изготовления программного продукта в 3–4 раза.

Опыт применения системы СЕТЛ-БЭСМ показывает бесспорную перспективность языка. Для будущих, существенно более эффективных, чем БЭСМ-6, вычислительных систем, язык типа СЕТЛ может стать одним из базовых универсальных средств программирования.

В настоящее время ведутся работы по реализации языка СЕТЛ для ЭВМ ЕС и ВС ЭВМ.

6. Система управления недетерминированного типа. В подходе, развиваемом в рамках данного проекта, в противоположность традиционным методам, проблема управления многопараметрической системой в условиях нестационарной внешней среды решается не как поиск априорной оптимальной или квазиоптимальной линии поведения, а как задача определения наиболее широкого "коридора поведения", обеспечивающего системе управления максимальную свободу маневра при изменении внешних условий. Методика подхода была с успехом опробована на системах управления движением шагающего автомата и манипулятора. Предполагается рассмотрение недетерминированных моделей управления промышленными объектами.

2.9. Проект МАРС создания модульной асинхронной развиваемой системы

Теоретические исследования методов организации параллельных вычислений, проводившиеся в Вычислительном центре в течении многих лет, позволили в середине 70-х годов приступить к формированию долгосрочной программы фундаментальных и прикладных работ в области архитектуры перспективных высокопроизводительных ЭВМ и про-

граммного обеспечения для них. Результатом этих работ должны стать обоснованные рекомендации по практической реализации новых архитектурных принципов, обеспечивающих

- существенное увеличение общей производительности вычислительных машин;
- уменьшение сложности проектирования, разработки и применения ЭВМ;
- возможность быстрой адаптации вычислительных систем к различным сферам и режимам применения.

Первый этап программы, завершившийся в 1978 г., состоял в формировании (на основе результатов предшествующих исследований и анализа тенденций в развитии вычислительной техники) концепции МАРС. Ниже представлены ее основные архитектурные принципы.

1. Иерархическое распределение и децентрализация вычислительной системы и всех протекающих в ней процессов — процессов обработки, процессов управления, процессов хранения и пересылки данных.
2. Модульность системы, представляющей собой иерархию вычислительных модулей, имеющих собственные средства обработки, хранения данных, локальную коммутацию и автономное управление.
3. Специализация и программируемость модулей на определенные прикладные и системные функции-модули для обработки данных разных типов (скаляры, векторы, тексты и т.п.), спецпроцессор, модули для реализации функций операционной системы, аппаратные средства повышения уровня машинного языка.
4. Развиваемость системы, базирующаяся на регулярности общей структуры системы, стандартизации интерфейса между модулями, специализируемости модулей и подсистем.

Таким образом, наряду с дальнейшим распараллеливанием всех процессов в системе, особое внимание обращено на развитие и аппаратную реализацию системных функций, на возможность перехода к "крупноблочному" проектированию системы, позволяющему существенно упростить и удешевить ее разработку, конструирование и адаптацию к пользователю.

Дальнейшее развитие и реализация концепции системы МАРС ведется по схеме "модель вычислений — язык программирования — архитектура системы". Реализация перечисленных выше архитектурных

принципов становится возможной, если система базируется на асинхронной модели параллельных вычислений, в которой все действия считаются изначально независимыми, а вычислительный процесс формируется в результате ограничений, накладываемых на порядок их исполнения. Эти ограничения формулируются в виде условий готовности, связанных с каждым действием и зависящих от наличия данных для выполнения действия, программных или системных событий или от значений данных.

Базовый параллельный язык реализует в языковой форме разработанную и исследованную модель асинхронных вычислений. Он имеет развитые средства для описания разнообразных форм параллелизма, сложных структур данных и параллельных операций высокого уровня над этими структурными данными. Для упрощения грамматической структуры язык с такими богатыми возможностями устроен как система четырех ортогональных языков: С-язык для описания структур управления, Е-язык для вычислительных операций, М-язык для описания работы с памятью и К-язык для конструирования составных объектов в предшествующих трех подъязыках. Все подъязыки имеют простую унифицированную структуру, описываемую в терминах единого метаязыка.

В настоящее время базовый язык находится в завершающей стадии разработки, продолжается работа по конкретизации архитектуры системы МАРС. В стадии инженерного проектирования находится макет высокопроизводительного модуля процессора для обработки числовой, скалярной и векторной информации с входным языком высокого уровня. Развиваются работы в области формализации методов проектирования архитектуры ЭВМ, в том числе разработка языков описания архитектуры. Вместе с тем активно продолжаются теоретические исследования методов описания семантики структур управления и структур данных в параллельных языках, разработка практических вариантов алгебры сетей Петри, ориентированных на моделирование иерархических асинхронных систем со сложными взаимодействиями. Разрабатывается язык описания дидикциональной архитектуры средств вычислительной техники. Этот язык является диалектом базового языка и будет составлять основу верхнего уровня систем автоматизации проектирования средств вычислительной техники.

Наибольший эффект при распараллеливании вычислений достигается тогда, когда распараллеливание ведется на всех этапах прохожде-

ния задачи и на всех уровнях алгоритмических конструкций. Работы, выполненные в ВЦ в рамках создания методической и алгоритмической базы для систем параллельного программирования, полностью учитывают это. Они ведутся в трех основных направлениях.

1. Статическое распараллеливание алгоритмов. Типичная постановка задачи статического распараллеливания — построить алгоритм, сопоставляющий каждой программе или программной конструкции эквивалентную ей и обладающую максимальной (оптимальной) параллельностью. Соответствующий блок распараллеливания может быть встроен в оптимизирующий транслятор.
2. Динамическое распараллеливание. Задача заключается в нахождении алгоритма, который по произвольной программе и начальным данным генерирует непосредственно максимально (оптимально) параллельное вычисление, минуя стадию параллельной программы.
3. Синтез параллельных программ на вычислительных моделях. Эта проблематика связана с генерацией максимально (оптимально) параллельных программ или вычислений на основе специального описания предметной области — вычислительной модели. Методика синтеза может лечь в основу системы генерации интеллектуальных ППП.

Работы в первом направлении были начаты в 60-е годы. Задача состояла в том, чтобы исследовать общие принципы распараллеливания, сформировать необходимый понятийный аппарат, формализовать его и на его основе исследовать теоретически возможности и пределы автоматического распараллеливания. Эта задача решалась в рамках теории схем программ. Были проанализированы структурные (информационные и логические) зависимости между операциями схем последовательных программ и выделены те отношения, которые определяют возможность распараллеливания. Было показано, что при определенных ограничениях на класс схем программ, существует алгоритм, позволяющий построить максимально параллельную схему асинхронной программы, эквивалентную исходной схеме последовательной программы. В настоящее время работы, выполняемые в первом направлении, касаются в основном распараллеливания циклических участков программ. Такой выбор не случаен, т.к. в циклах скрыта большая часть потенциальной параллельности программ. В ВЦ разработаны два метода распараллеливания циклических участков — метод параллелепипедов и метод пи-

рамид. Первый ориентирован на матричные процессоры, второй — на МВК типа Эльбрус-Бэрроуз. Первый метод реализован программно в экспериментальной версии. Кроме перечисленных работ в ВЦ велись также некоторые работы по распараллеливанию процедур, выделению оптимальных ветвей в ациклических конструкциях.

Появление работ второго направления вызвано тем, что разветвление программы с циклами, образованными операторами перехода, плохо поддаются распараллеливанию в статическом режиме. Для достижения максимальной параллельности необходим анализ и выявление параллелизма в процессе вычисления с учетом складывающейся обстановки. В ВЦ разработан ряд методов, позволяющих вести такой анализ с различной глубиной упреждения и для разных понятий эквивалентности и асинхронности. При необходимости получить максимально параллельное вычисление эти методы должны удовлетворять следующему принципу: некоторому оператору разрешается включиться, начиная с некоторого момента; если момент становится известным, то оператор неизбежно (возможно) выполнится и для него вычислена информация, которой он обязательно (возможно) воспользуется. Включение "неизбежных" операторов гарантирует максимальную параллельность вычисления. При этом оно не содержит неиспользуемых операторов. Включение же "по принципу возможности" позволяет достичь абсолютно максимальной параллельности: ни при каких других режимах, не использующих свойств конкретной интерпретации, вычисление нельзя провести быстрее. Однако, при этом допускаются вычислительные акты, которые не будут задействованы.

Задача синтеза программ на основе вычислительных моделей является известной, и есть действующие системы синтеза программ. Новое в работах ВЦ заключается в том, что

а) программы синтезируются с заданной степенью надежности, которая достигается за счет возможности дублирования вычислений одних и тех же переменных разными группами модулей;

б) проводится оптимизация по объему программы, числу операций ввода–вывода, общему числу тактов;

в) синтезируемая программа является максимально параллельной.

Получены соответствующие результаты как для простых вычислительных моделей, так и для моделей с массивами.

Велись также некоторые работы, примыкающие к этому направлению. Так, был введен и обоснован формализм для описания предметных областей и манипуляций над ними.

2.10. Проект "ВЦКП" создания вычислительного комплекса (центра) коллективного пользования

Программой работ по проекту "ВЦКП" предусматриваются следующие направления.

1. Проведение научно-исследовательских работ с целью изучения различных аспектов более эффективного использования вычислительных средств за счет комплексирования их в единую систему, использования адекватного возможностям вычислительных средств общесистемного и прикладного программного обеспечения, организации дистанционных связей с абонентами, а также за счет организации коллективного режима использования средств вычислительной техники.
2. Проведение опытно-конструкторских работ по созданию вычислительного комплекса коллективного пользования (ВККП) с целью его практической эксплуатации в Новосибирском научном центре СО АН СССР.
3. Выработка технических предложений по созданию типовых вычислительных центров коллективного пользования с учетом особенностей их использования в различных областях народного хозяйства. Проектом предусматривается создание многомашинного территориально-распределенного вычислительного комплекса типа локальной сети ЭВМ, функционирующего в режиме коллективного использования. При этом главными целями реализации являются обеспечение пользователей вычислительными и информационными мощностями, создание технической и технологической базы для развертывания прикладных программных систем (пакетов прикладных программ), ориентированных на обеспечение и автоматизацию научных исследований, развитие автоматизированных систем территориального управления и управления производством, а также исследование различных аспектов построения, реализации и эксплуатации подобных центров автоматизированной обработки информации.

Основу ВЦКП составляет вычислительный комплекс коллективного пользования (ВККП), он рассматривается как единство комплекса

технических средств и общесистемной программной поддержки, являющихся технической и технологической базой для развертывания прикладных систем в интересах проблемных применений.

На базе ВККП создается вычислительная система коллективного пользования ВСКП, при этом ВККП и функционирующие на его основе прикладные программные системы рассматриваются как единое целое.

И наконец, лишь при погружении вычислительной системы коллективного пользования в конкретные условия административно-правового и технологического обеспечения мы вправе говорить о появлении Вычислительного Центра Коллективного Пользования (ВЦКП).

В ВККП различаются три функциональных уровня — уровень взаимодействия с пользователем, коммуникационный уровень и уровень исполнения.

Уровень взаимодействия с пользователем представлен в системе периферийными центрами обработки (ПЦО), коммуникационный уровень — системой передачи данных (СПД) и уровень исполнения — базовыми вычислительными комплексами (БВК).

Таким образом, ВККП является многомашинным территориально-распределенным вычислительным комплексом общего назначения типа сети ЭВМ, базирующемся на разнородных средствах отечественной вычислительной техники и представляющем для массового пользователя единую вычислительную систему, обеспечивающую использование ее технических средств и информационно-вычислительных ресурсов в режимах разделения времени при контакте с системой и пакетной обработки при решении основных задач.

Абонентом центра коллективного пользования (АЦКП) может быть абонентский пункт, терминальное устройство, отдельная ЭВМ или развитый ведомственный вычислительный центр, удовлетворяющий условиям технического и программного интерфейса с ВККП. Абонент центра является для ВЦКП источником и приемником информации и управляющих воздействий (директив). Связь абонента с ВЦКП обеспечивается локальной абонентской сетью периферийного центра обработки (ПЦО). Периферийный центр обработки, разворачиваемый на базе мини-ЭВМ, концентрирует абонентов в основном по территориальному признаку. Проектирование ВЦКП ведется в предположении, что в развитом его варианте на территории Академгородка будет расположено не менее десяти периферийных центров обработки информации.

Функционально ПЦО обеспечивает: прием, предварительную обработку и преобразование к каноническому для системы виду поступающей от абонента информации и управляющих директив; первичную диспетчеризацию потоков заданий и сбор находящейся в его компетенции статической информации для ВЦКП в целом; прием адресуемой абоненту информации и ее выдачу в требуемом формате.

Одним из периферийных центров обработки является комплекс обработки аэрокосмических изображений ВЦ СО АН СССР. Функции процессора ввода-вывода выполняет мини-ЭВМ М-6000, к которой подключены микроденситор МАРК-2, фототелеграфный аппарат НЕВА, цветной полутоновый дисплей с памятью регенерации, прямой канал приема изображений с ИСЗ Метеор–природа. Создан пакет программ предварительной обработки аэрокосмических изображений, включающий более пятидесяти программных модулей.

Таким образом, периферийные центры обработки обеспечивают как доступ абонентов к основным вычислительным и информационным мощностям, так и связь абонентов между собой. В последнем случае обеспечивается лишь транзит сообщений абонентов друг другу без анализа синтаксиса и семантики. Это открывает принципиальную возможность кооперации заинтересованных организаций с целью решения ведомственных задач вне рамок ВЦКП.

Основные же вычислительные и информационные мощности сосредотачиваются в базовых вычислительных комплексах (БВК) на основе ЭВМ ЕС, БЭСМ-6. При этом принципиальных ограничений для увеличения числа БВК и ПЦО в составе ВЦКП не имеется. Связь базового вычислительного комплекса с остальными компонентами ВЦКП реализуется через связной процессор (СВП), разворачиваемый на базе отдельной мини-ЭВМ. В рамках БВК связной процессор обеспечивает прием адресуемой БВК информации и ее приведение к каноническому для БВК формату, формирование заданий и диспетчеризацию работы БВК, сбор статистической информации о работе БВК, прием и передачу адресуемой другим компонентам ВЦКП информации. Функционально базовый вычислительный комплекс обеспечивает эффективные вычисления и обработку информационных данных в режиме дистанционной пакетной обработки по заданиям, формируемым связным процессором.

Каждый из БВК (ЕС, БЭСМ-6) и ПЦО, а также их возможные комбинации могут быть автономной технико-технологической системой, обеспечивающей функционирование в режиме коллективного исполь-

зования. На основе этих систем могут быть созданы самостоятельные ВККП различного назначения и производительности.

Связующим звеном комплекса является система передачи данных (СПД), обеспечивающая передачу информации и управляющих воздействий между мини-ЭВМ, реализующими функции периферийных центров обработки информации и связных процессоров базовых вычислительных комплексов. СПД состоит из нескольких узлов, соединенных линиями связи и обеспечивает установление соединения и передачу информации по методу коммутации пакетов. Узел СПД реализуется на основе мини-ЭВМ и специального канального оборудования.

При реализации проекта ВЦКП ведется ориентация на использование штатного программного обеспечения в части ЭВМ ЕС, БЭСМ-6 и разрабатывается комплекс общесистемного программного обеспечения для мини-ЭВМ, включаемых в вычислительный комплекс коллективного пользования в качестве периферийных центров обработки (ПЦО) и связных процессоров (СВП) БВК. Основными компонентами разрабатываемого ОСПО являются:

– операционная система Дирак,

– язык и система автоматизации программирования МАСМ.

Работы по проекту будут продолжены со следующей пятилетки с целью создания Академической региональной сети (АРС), распространяющейся на основные научные центры СО АН СССР. При этом выделяются следующие основные направления развития:

– увеличение числа ПЦО, расширение сети абонентов и адаптация нового терминального оборудования;

– развитие БВК ЕС с включением в его состав ЭВМ ЕС-1060, ЕС-1065 и процессора внешней памяти на базе мини-ЭВМ;

– развитие сети передачи данных за счет создания новых узлов СПД, в том числе и ориентированных на использование спутниковых каналов связи;

– расширение функциональных возможностей операционной системы Дирак, системы программирования МАСМ и их перенос на мини-ЭВМ типа СМ-3/4;

– дальнейшее проведение научно-исследовательских и опытно-конструкторских разработок по проблеме создания больших банков данных, автоматизированной системы ведения классификаторов (по компонентам ВЦКП) обработки информации;

– расширение списка прикладных программных систем (ППС), ори-

ентированных, в первую очередь, на обеспечение научных исследований и создание автоматизированных систем управления предприятиями, территориально-промышленными комплексами, районами и другими хозяйственными объектами;
– разработка новых прикладных программных систем в соответствии с технологией, обеспечивающей адекватность реализуемых программных комплексов архитектуре ВЦКП.

3. Г. Д. ЧИНИН. О РАБОТАХ НОВОСИБИРСКОГО ФИЛИАЛА ИНСТИТУТА ТОЧНОЙ МЕХАНИКИ И ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ ИМ. С. А. ЛЕБЕДЕВА АН СССР

Новосибирский филиал Института точной механики и вычислительной техники (НФ ИТМиВТ АН СССР) образован в 1972 г. Основная специализация — научно-исследовательские и опытно-конструкторские работы в области системного программирования и создание общего системного программного обеспечения для МВК "Эльбрус".

Кадровую основу НФ в начальный период его существования составили сотрудники Вычислительного центра СО АН СССР (ВЦ) — специалисты с большим опытом исследовательской и практической работы в области системного программирования.

Эти два обстоятельства (основная специализация и изначальная кадровая основа) определили и конкретные работы НФ на достаточно длительный период времени.

Так, из ВЦ в НФ перешли следующие крупные разработки конкретного характера и поисковые работы, некоторые из которых были завершены в 1972–1975 гг.:

- Система коллективного пользования АИСТ-0. Работа проводилась совместно с ВЦ, где выполнялась аппаратная часть проекта.
- Специализированная система программного обеспечения "Корунд" для ЭВМ "Урал-14".
- Система программирования "Альфа-6".

К 1975 г. сложились три основные направления работ, которые определяют 95% тематики НФ:

1) системы программирования для МВК "Эльбрус".
2) технологические системы на ЭВМ БЭСМ-6.
3) экспериментальные работы.

3.1. Первое направление

Для МВК "Эльбрус-1,2" создаются трансляторы с известных языков Алгол, Фортран, Кобол, ПЛ/1, Симула-67, Алгол-68, Альфа.

В настоящее время в штатном общесистемном программном обеспечении (ОСПО) функционируют первые версии трансляторов с Алгола и Фортрана. Вторая очередь работ в направлении этих языков — создание системы программирования Алгол-Эльбрус и Фортран-Эльбрус.

Алголовская система программирования допускает в качестве входных языки Алгол-60, Алгол-ГДР, Алгол-ЕС, Алгамс, Альфа и Алгол-Эльбрус. Последний элемент этого перечня представляет собой некоторое расширение Алгола-60 в сторону Эльбруса. Язык дополняется новыми типами (и форматами) данных, операциями и языковыми процедурами (функциями), аппаратно реализуемыми в "Эльбрусе". Кроме того, в язык включаются основные интерфейсные процедуры для взаимодействия с операционной системой. Транслятор с языка Симула строится на основе системы Алгол-Эльбрус.

Расширяется номенклатура входных языков в системах программирования на Фортране. К теперешнему Фортран-стандарту добавляются Фортран-Дубна, Фортран-77, Фортран-ЕС и Фортран-Эльбрус. Фортран-Эльбрус является расширением Фортрана-77 средствами распараллеливания вычислений и некоторыми возможностями "Эльбруса". Фортрановский комплекс предусматривает несколько уровней оптимизации. Максимальный уровень оптимизации соответствует Фортрану ОЕ для машины ЕС. Кроме того, в фортрановский комплекс включаются адаптированные библиотеки процедур численного анализа и линейной алгебры.

Трансляторы с языков Кобол и ПЛ/1 учитывают существующие версии соответствующих входных языков для машин ЕС.

Алгол-68 разрабатывается средствами системы построения трансляторов, которая представляет собой набор специализированных языково-программных средств, ориентированных на типовые, характерные этапы и элементы трансляции.

Система программирования с перечисленных языков дополняется системой языковых файлов, которая, будучи надстройкой над системой файлов низкого уровня, обеспечивает унифицированную реализацию средств обмена во всех языках и допускает связь программ на разных языках через файлы.

Перечень конструкторских разработок для "Эльбруса" завершает

базовый пакет машинной графики, доступной из всех реализуемых языков, и система аналитических выкладок. Для обеих этих систем созданы прототипы на БЭСМ-6.

Отдельное ответвление в работах по "Эльбрусу" составляет программное обеспечение спецпроцессора с системой команд БЭСМ-6. Здесь созданы язык системного программирования ЯРМО и операционная система, которая сопровождается и эксплуатируется в практической работе.

Дальнейшее развитие этого направления идет по пути создания новых версий ЯРМО, операционной системы и трансляторов с Алгола и Фортрана, увязанных в единый комплекс, подобно тому, как это сделано в "Эльбрусе", с учетом многообразного и многолетнего практического опыта применения ЯРМО и последних современных тенденций.

3.2. Второе направление

Технологические системы на ЭВМ БЭСМ-6. Сюда относится упоминавшаяся система ЯРМО с соответствующими комплексами поддержки, наиболее мощная из которых — комплекс "Интеграл". Непосредственно для "Эльбруса" созданы два инструментальных комплекса "Стрела" и "Темп", отражающие соответствующие этапы разработки "Эльбруса". В настоящее время комплекс "Темп" используется более чем в двадцати предприятиях.

Дальнейшее развитие технологического направления предусматривает создание новой (третьей) версии ЯРМО (для БЭСМ-6 и "интегральной БЭСМ") и комплекса обеспечения процесса организации и создания (проектирования, программирования, тиражирования и т.д.) больших программных систем.

3.3. Третье направление

Экспериментальные работы. Это работы, которые проводятся в интересах "Эльбруса" и преследуют цель создания на БЭСМ-6 действующих прототипов. Сюда относятся уже завершенные разработки нескольких компонент машинной графики, системы аналитических выкладок, пакетов прикладных программ, системы построения трансляторов и некоторые другие. Разработаны системы "БОЯЗ-6" и "Сеть", направленные на создание информационных систем. На их основе созданы библиотечно-справочная система, система автоматизации кадрового уче-

та, система канцелярской работы и некоторые другие.

Разрабатывается система высокоуровневой обработки данных (на базе языка СЕТЛ) и система автоматизации логического проектирования блоков и узлов ЭВМ.

Кроме перечисленных, НФ проводит ряд совместных работ с ВЦ, которые отражены в докладе В. Е. Котова (работы по проектам МАРС, Академсеть и некоторым другим).

4. ВЫСТУПЛЕНИЕ (КРАТКИЕ ТЕЗИСЫ) А.П. ЕРШОВА ПЕРЕД ПРИНЯТИЕМ РЕШЕНИЯ

Создание Сибирского отделения поставило научные коллективы, в том числе и программистов, в уникальные условия. Комплекс из институтов АН СССР, учебных заведений и отраслевых институтов создал предпосылки для взаимодействия Академии, Минвуза и промышленности. Создалась богатая инфраструктура на очень динамичной основе. Но эти моменты налагают и очень большую ответственность за развитие комплексного подхода к решаемым проблемам. Мы обнаружили, что эту комплексность реализует системное математическое обеспечение в своем продукте. Такую комплексную проработку можно осуществить не везде. Это можно только в ИК АН УССР и в СО АН СССР. Даже в Москве нет таких возможностей. В силу этого в ВЦ СО АН СССР сложились предпосылки для ведения работ по всему фронту системного программирования. Здесь есть свои проблемы. В целом покрытие всей тематики высокое, но есть и несбалансированность. Традиционно развивались академические отделы, посвященные теоретическим вопросам, например, вопросам трансляции, но в последнее время стали появляться конструкторские отделы с целевым назначением, которые не занимаются фундаментальными исследованиями. В итоге у нас ощущается нехватка фундаментальных исследований по некоторым направлениям. Мы, например, ничего не можем сказать о реляционных базах данных, начинает чувствоваться отставание в теории операционных систем. Эти бреши нужно закрывать. Для этого нужна целеустремленная работа руководства Института. Требуется также усиление работ по подготовке докторских диссертаций. Слишком много появилось народу, которому хорошо живется без защиты докторской диссертации. Только защиты такого уровня поставят программирование на уровень остальных направлений в институте. В целом в институте накоплен большой потенциал, но он недореализован.

5. ЗАКЛЮЧЕНИЕ О РАБОТАХ ВЫЧИСЛИТЕЛЬНОГО ЦЕНТРА СО АН СССР В ОБЛАСТИ СИСТЕМНОГО МАТЕМАТИЧЕСКОГО ОБЕСПЕЧЕНИЯ

21–22 мая 1981 г. Комиссия по системному математическому обеспечению Координационого комитета по вычислительной технике при Президиуме Академии Наук СССР провела комплексное ознакомление с работами ВЦ СО АН СССР в области системного программирования. Был заслушан и обсужден доклад заместителя директора к.ф.-м.н. В. Е. Котова о работах в этой области за время существования ВЦ СО АН СССР. При этом особое внимание было уделено результатам работ в последней пятилетке, рассмотрены планы работ на следующую пятилетку и некоторые научно-организационные вопросы.

С самого основания Института научная деятельность ВЦ в области системного программирования явилась весомым вкладом в развитие программирования в нашей стране. Для проводимых в Институте работ характерна широта тематики, актуальность направлений исследований, наличие тесных связей с прикладными программистами Сибирского отделения и практическими задачами Главного производственного вычислительного центра (ГПВЦ СО АН СССР).

Институт завоевал международный авторитет своими работами по фундаментальным проблемам системного программирования, он активно взаимодействует с отраслевыми организациями по внедрению новых разработок.

Широкое признание и высокую оценку специалистов получили работы Института в области теории схем программ, теории параллельных вычислений, теории трансляции и методов оптимизации программ, теории операционных систем, искусственного интеллекта.

Известность получили разработанные и внедренные во многих организациях страны системы программирования АЛЬФА и АЛЬФА-6 (системы программирования с высоким качеством получаемых программ), ЭПСИЛОН-М3, ЭПСИЛОН-Б, МАКРОЭПСИЛОН (системы программирования для разработки программного обеспечения).

Одной из первых в стране была создана экспериментальная многомашинная система коллективного пользования АИСТ-0. Завершается проект БЕТА — многоязыковая система программирования. Сдано межведомственной комиссии общесистемное программное обеспечение первой очереди вычислительного центра коллективного пользования.

Продолжая активно разрабатывать фундаментальные проблемы системного программирования, Институт сформировал новые крупные

проекты (проект МАРС, ВЦКП 2-й очереди, региональная АКАДЕМСЕТЬ, проект "Рубин" для издательства "Правда"), реализация которых будет осуществляться в 80-е годы. Все эти проекты и ряд других ведутся по постановлениям директивных органов.

Институт ведет совместные научно-исследовательские и опытно-конструкторские работы в области системного программирования в рамках координационных планов, договоров о сотрудничестве и хоздоговоров с ведущими в стране министерствами, разрабатывающими средства вычислительной техники — с Минрадиопромом, Минприбором, Минпромсвязью, а также с Минвузом и некоторыми другими ведомствами. Особо следует отметить роль ВЦ СО АН СССР в создании Новосибирского филиала ИТМ и ВТ им. С. А. Лебедева, научное руководство которым осуществляет ВЦ. Опыт совместной работы ВЦ СО АН СССР и НФ ИТМ и ВТ показывает, что содружество академических и промышленных организаций является на данном этапе необходимым условием успешного внедрения новых передовых идей в практику системного программирования.

ВЦ СО АН СССР активно участвует в реализации планов двухстороннего и многостороннего научного сотрудничества между Академией наук СССР, академиями наук социалистических стран и научными организациями Франции, США, Англии и др. стран. Институт успешно пропагандирует за рубежом достижения советской науки, в частности системного и теоретического программирования.

Одним из наиболее важных результатов содружества явилось создание в Институте по инициативе члена-корреспондента АН СССР А. П. Ершова библиотеки системного программирования — уникального собрания актуальной информации в этой области, включающей оперативные материалы, получаемые из ведущих зарубежных и советских научных организаций.

Вычислительный центр СО АН СССР на протяжении многих лет является активным организатором научных совещаний, симпозиумов и конференций по разным аспектам системного программирования, в том числе ряда крупных всесоюзных и международных совещаний. Эти совещания сыграли большую роль в деле распространения передового опыта разработки программных систем. Особо следует отметить инициативу Института в регулярном издании и широком распространении тематических сборников научных трудов по системному программированию.

Вычислительный центр СО АН СССР ведет большую педагогическую работу по подготовке молодых специалистов в области системного программирования в Новосибирском государственном университете им. Ленинского комсомола. Эта работа ведется в основном в рамках курсовых и дипломных работ. К сожалению, программа обучения по специальности "системное программирование" в НГУ не отвечает требованиям современного уровня развития этой науки и требованиям практики.

Вместе с тем Комиссия считает:

1. Работы в области системного математического обеспечения в ВЦ СО АН СССР не обеспечены в достаточной мере вычислительными ресурсами и ассортиментом терминального оборудования.
2. В Институте неудовлетворительно поставлена работа по формальной научной аттестации квалификации ведущих сотрудников — специалистов в области системного программирования. Число докторов и кандидатов наук в ВЦ СО АН СССР не соответствует количественным и качественным параметрам коллектива системных программистов.
3. Дальнейшему внедрению научных разработок, выполненных в ВЦ СО АН СССР, препятствует отсутствие при Институте собственного опытно-конструкторского подразделения, способного разрабатывать большие программные системы.
4. Программа обучения по специальности "системное программирование" в НГУ не соответствует современному уровню развития этой дисциплины и требованиям практики, поэтому в Институте затрачиваются дополнительные усилия на начальную подготовку приходящих на практику студентов и молодых специалистов.

Комиссия постановляет:

1. Одобрить деятельность Вычислительного центра Сибирского отделения АН СССР в области системного программного обеспечения.
2. Просить Институт уделять большее внимание подготовке докторов и кандидатов наук в области системного математического обеспечения.
3. Просить Новосибирский госуниверситет им. Ленинского комсомола открыть отделение "системное программирование" на математическом факультете с программой, соответствующей рекомендациям Минвуза СССР.

6. ЗАКЛЮЧЕНИЕ

В докладах В.Е.Котова и Г.Д.Чинина не были указаны ответственные исполнители этих работ, а также не были названы ведущие ученые-теоретики. Мы не будем углубляться в эти вопросы, боясь пропустить или неверно указать вклад в решение тех или иных проблем. Скажем только, что под руководством чл.-корр. АН СССР (позднее академика) А. П. Ершова успешно трудились И. В. Поттосин, Г. И. Кожухин, В. Э. Иткин, В. Н. Касьянов, В. А. Непомнящий, В. Е. Котов, А. С. Нариньяни, В. А. Вальковский, А. А. Берс, М. Б. Трахтенброт, В. К. Сабельфельд и др. В НФ ИТМ и ВТ АН СССР трудились В. Л. Катков, Г. Д. Чинин, Б. Г. Чеблаков, Л. Б. Эфрос, М. М. Бежанова, И. С. Голосов, А. В. Замулин, В. И. Фишелев и др.

Указанные ученые Сибирского отделения СО АН СССР принимали активное участие в работе ККВТ АН СССР, как в составе комиссии (И. В. Поттосин, В. Е. Котов, В. Л. Катков, Г. Д. Чинин) или рабочих групп (В. Н. Касьянов, Г. Г. Степанов, и др.), так и в качестве приглашенных специалистов. С докладами на заседаниях Комиссии по СМО выступали И. В. Поттосин, Г. Д. Чинин, М. М. Бежанова, Б. Г. Чеблаков, Л. Б. Эфрос, В. А. Непомнящий, В. А. Дебелов.

В. А. Евстигнеев

ОБЗОР ДЕЯТЕЛЬНОСТИ НОВОСИБИРСКИХ УЧЕНЫХ В ОБЛАСТИ ПРОГРАММИРОВАНИЯ (ПО МАТЕРИАЛАМ КОМИССИИ ПО СИСТЕМНОМУ МАТЕМАТИЧЕСКОМУ ОБЕСПЕЧЕНИЮ ККВТ АН СССР)

Препринт
83

Рукопись поступила в редакцию 15.01.2001

Рецензент И. В. Поттосин
Редактор З. В. Скок

Подписано в печать 15.02.2002
Формат бумаги 60×84 1/16 Объем 3,3 уч.-изд.л., 3,6 п.л.
Тираж 50 экз.

НФ ООО ИПО "Эмари" РИЦ, 630090, г. Новосибирск, пр. Акад. Лаврентьева, 6