Системное программное обеспечение вычислительной системы «Электроника ССБИС»

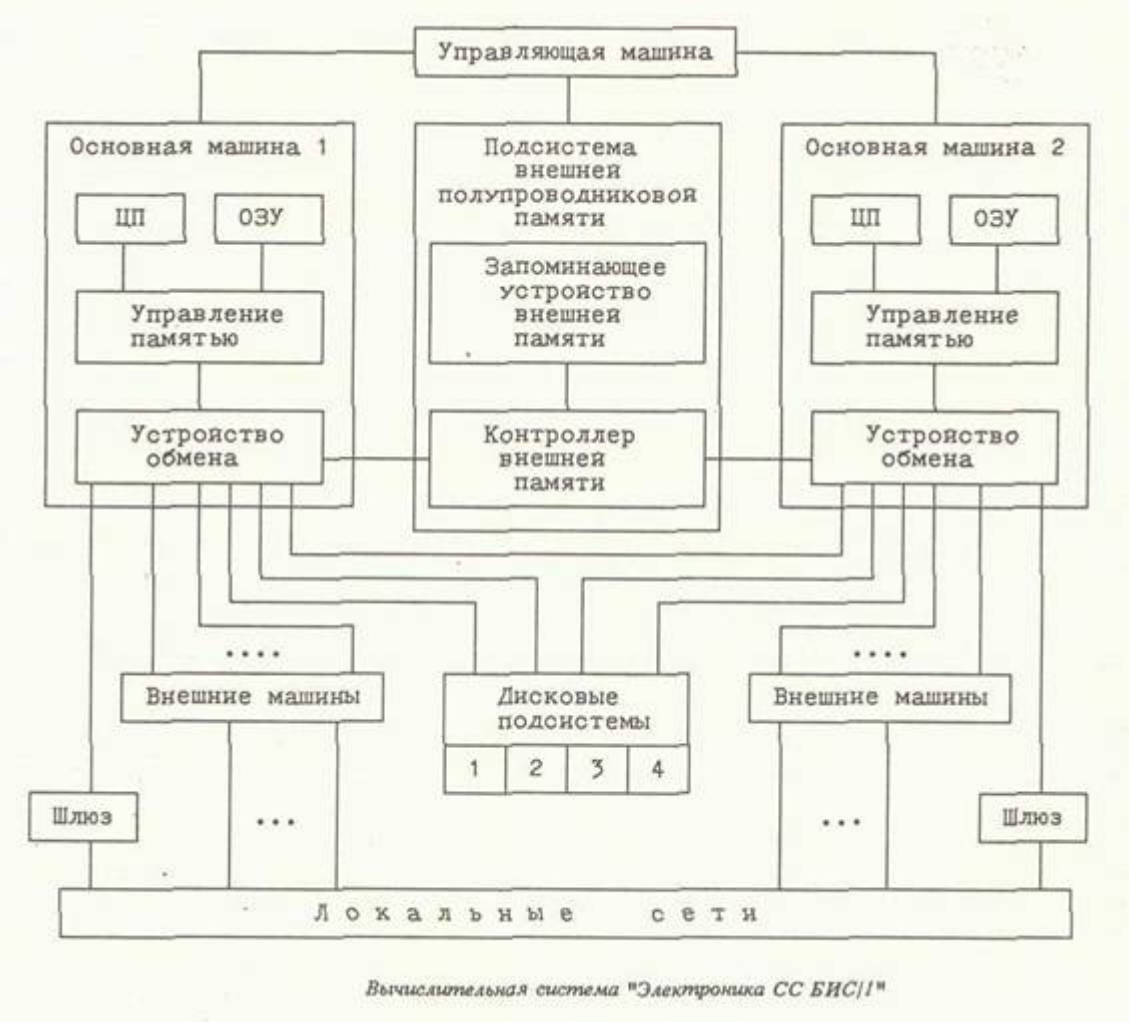
Иванников В.П., Гайсарян С.С., Томилин А.Н.

Институт системного программирования РАН
tom11@bk.ru

Ivannikov V., Gaisaryan S., Tomilin A.
System Software for the computing system "Electronica SSBIS"
RAS Institute for System Programming

К 1991 году были разработаны и произведены в нескольких экземплярах две векторно–конвейерные суперЭВМ: «Электроника ССБИС» (разработка НИИ «Дельта» МЭП СССР и Института проблем кибернетики АН СССР – главный конструктор Владимир Андреевич Мельников) и «Модульный конвейерный процессор (МКП)» (разработка Института точной механики и вычислительной техники имени С.А. Лебедева – главный конструктор Андрей Андреевич Соколов).

На рисунке представлен вариант конфигурации вычислительной системы "Электроника ССБИС/1".



Вычислительная система "Электроника СС БИС/1"

В первые системы «Электроника ССБИС/1» входили:

- основная машина векторно–конвейерного типа с локальным оперативными запоминающим устройством емкостью 8–32 Мб и производительностью 250 млн. операций в секунду над числами с плавающей запятой;
- подсистема внешней полупроводниковой памяти емкостью 256 Мб с двухпортовым контроллером для подключения двух основных машин;
- от 2 до 8 дисковых подсистем общей емкостью 20 Гб;
- подсистема внешних машин, используемая для работы с периферийным оборудованием и для подготовки заданий для основной машины;

- управляющая машина для выполнения управляющих и диагностических функций всей системы;
- программируемые устройства доступа для построения локальной вычислительной сети, обеспечивающей работу с рабочими станциями и внешними ЭВМ пользователя.

Высокая производительность основных машин достигается за счет широкого использования принципа конвейерной обработки с малым тактом синхронизации, высокоскоростной регистровой памяти большого объема и большого числа специализированных функциональных устройств, обеспечивающих параллельную обработку данных, объединения в рамках одной ЭВМ возможностей скалярной и векторной обработки.

Архитектура комплекса суперЭВМ, базирующаяся на объединении функционально–специализированных вычислительных средств подготовки, передачи и обработки данных, предъявляет к системному программному обеспечению компонент и комплекса в целом требования адаптивности к подключению в комплекс различных аппаратных средств, к включению суперЭВМ в состав различных крупных вычислительных центров.

В соответствии с этим требованием системное программное обеспечение комплекса суперЭВМ обеспечивало работу его компонентов локальной сети ЭВМ, в которую входят основные машины, предназначенные для выполнения большого объема вычислений, и внешние ЭВМ, используемые как для подготовки данных, передачи их для выполнения счета на основные машины, приема от них и выдачи на устройства вывода результатов вычислений, так и для совместного использования с основными машинами в системах распределенной обработки информации, в том числе в системах обработки информации в реальном времени.

Рассмотренные требования привели к децентрализованной архитектуре программного обеспечения системы, в которой независимые операционные системы разных функциональных компонент дополнены средствами работы в составе локальной сети системы. В эти средства входят транспортная служба сети, обеспечивающая обмен информацией между процессами в разных ЭВМ системы (характер взаимодействия между ними относится к типу удаленный вызов процедур), и службы, реализующие функциональные протоколы сети – протоколы ввода заданий и вывода результатов счета, пересылки файлов, операторской службы системы. Над транспортной службой сети могут создаваться пользовательские распределенные системы, например, распределенные информационно–поисковые системы, системы автоматизации проектирования, системы обработки информации в реальном времени.

Основная машина имела развитую операционную систему (ОС), выполнявшую наряду с традиционными функциями операционных систем управление данными на двухуровневой внешней памяти, поддержку транспортных и функциональных протоколов сети.

Основные особенности архитектуры высокопроизводительной векторно–конвейерной ЭВМ – наличие возможностей векторной обработки данных, многоуровневая организация регистров процессора, параллельно работающие конвейерные устройства – потребовали создания оптимизирующих трансляторов с языков высокого уровня.

В состав системного программного обеспечения основной машины вычислительной системы "Электроника ССБИС" входили следующие компоненты:

- операционная система;
- базовая система программирования;
- система программирования на языке ФОРТРАН–77;
- система программирования на языке Си;
- система программирования на языке Паскаль;
- имитационный комплекс основной машины, предназначенный для разработки и отладки ее программного обеспечения на инструмен тальных ЭВМ ("Эльбрус 1КВ", IBM PC/AT);
- стандартное прикладное программное обеспечение.

Операционная система основной машины

Выбор архитектуры операционной системы определяется ее функциональным назначением, архитектурой комплекса и средой, в которой ОС предстоит работать. Аппаратные особенности основной машины (ОМ) и режимы работы — пакетный и реального времени — определяют выбор базовых объектов ОС: задач (пользовательских, системных), выполняющихся в отдельных адресных пространствах, и сообщений как средства взаимодействия и синхронизации задач. Ядро ОС осуществляет передачу сообщений и управление задачами.

Операционная система состоит из ядра и совокупности системных задач. Некоторые системные задачи, называемые псевдозадачами, работают в привилегированном режиме, обладают доступом ко всей оперативной памяти. В виде псевдозадач оформлены: транспортная станция, задачи управления оперативной и внешней памятью. Псевдозадачи взаимодействуют с задачами ОС, используя стандартный аппарат обмена сообщениями.

Аппарат обмена сообщениями между задачами реализует примитивы: "послать сообщение синхронно", "послать сообщение асинхронно", "послать ответ на сообщение", "ждать сообщение", "ждать ответ на сообщение", "ждать истечения интервала времени" и примитивы с различными комбинациями таких указаний.

Центральной задачей ОС является "Инициатор–терминатор", выполняющий следующие функции:

- образование новых пользовательских задач и системных нерезидентных задач ОС;
- извещение всех системных задач о появлении новой задачи, в результате чего все системные задачи настраиваются на работу с новым абонентом;

- нормальное или аварийное завершение задачи, о котором оповещаются все системные задачи, после чего системные задачи отстраиваются от абонента;
- планирование выполнения введенных заданий и шагов заданий;
- разрешение конфликтных ситуаций при не хватке ресурсов и тупиках в системе;
- управление "контрольными точками".

Задания готовятся на внешних машинах (ВМ). Задание состоит из совокупности наборов данных. Обязательным в задании должен быть набор данных – управляющая программа на языке управления заданиями. Задание принимается системной задачей ввода заданий и помещается в файловую систему ОС ОМ. Планировщик инициатора–терминатора выбирает задание на выполнение. Задание состоит из последовательности шагов. Для выполнения каждого шага создается задача. При создании задачи для нее выделяется оперативная память. В случае нехватки памяти выстраивается очередь шагов заданий, ожидающих для своего выполнения предоставления оперативной памяти. Запущенная задача оканчивается либо специальным сообщением от нее об окончании выполнения шага задания, либо аварийно.

При выполнении шагов задания выводные данные заносятся в выводные файлы задания. Инициатор–терминатор передает системной задаче вывода результатов эти файлы. После завершения задания системная задача вывода результатов передает их на внешние машины.

При выполнении шага задания инициатор–терминатор может получить от него сообщение–просьбу о построении контрольной точки. Инициатор–терминатор сообщит всем системным задачам о том, что для данной задачи строится контрольная точка. В ответ инициатор–терминатор получит информацию, по которой впоследствии можно возобновить выполнение задачи. Эту информацию инициатор–терминатор заносит в "журнал" контрольных точек, куда также поступает информация о состоянии регистров и памяти задачи. После аварии инициатор–терминатор выполняет обратные действия, в результате чего счет задачи может быть продолжен с последней ее контрольной точки.

При работе нескольких задач они могут попасть в тупик из–за совместного использования файлов или из–за нехватки массовой памяти. Инициатор–терминатор, учитывая приоритеты задач и занимаемые ими ресурсы, исключает одну из выполняемых задач, разрешая тем самым тупик. Исключенная задача "откатывается" к своей последней контрольной точке, с которой впоследствии продолжится ее выполнение.

Последовательность шагов задания описывается на языке управления заданиями (ЯУЗ).

Для обеспечения гибкого конструирования и изменения операционной среды выполняемой программы в системе программирования ОМ реализован механизм абстракций при помощи понятия "кластер". Абстрактный объект характеризуется уровнем спецификации (представляется набором операций со специфицированным интерфейсом) и уровнем представления (с описанием локальных структур данных; процедур, реализующих операции; других абстрактных объектов). Программе, использующей абстрактный объект, он доступен только на уровне спецификаций, уровень представления скрыт от нее. Кластер порождается конструкцией объявления, в которой задается имя экземпляра и его тип. Тип содержит всю информацию, характеризующую объект. Существуют эквивалентные типы кластеров (порождающие кластеры с одинаковым уровнем спецификаций, но с различными уровнями представления). Совокупность кластеров задает операционную среду выполнения программы.

Кластер порождается на этапе загрузки (компоновки программы из объектных модулей). Программа, использующая кластеры, может содержать конструкции объявления экземпляров в собственном теле. В этом случае во время компоновки загрузчик по этим конструкциям породит необходимые экземпляры кластеров и свяжет с ними исходную программу. Программа может содержать только конструкции обращения к кластерам, а объявления экземпляров могут осуществляться в других объектных модулях (в том числе извлеченных из библиотеки объектных модулей), которые совместно с данной программой, также подготовленной в виде объектного модуля, поступят на вход загрузчику. Существует возможность объявить экземпляр кластера и на языке управления заданиями.

При компоновке программы в ее пространство загружается и компонент управления заданиями – интерпретатор шага, которому доступны все специфицированные точки входа в программу, представленную в виде двоичного кода. После загрузки шага задания в оперативную память инициатор–терминатор передает управление интерпретатору шага для интерпретации файла управления заданиями во внутреннем представлении, задающем последовательность обращения к точкам входа данного шага. Обычная последовательность обращений заключается в инициации кластеров (подготовка операционной среды к выполнению) и в запуске собственно программы. Встретив в файле обращение к другому шагу, интерпретатор шага обратится к инициатору–терминатору с указанием следующего шагаю и смещения по файлу управления заданиями во внутреннем представлении. По данному сообщению будет закончена задача, в рамках которой выполнялся шаг, освобождена оперативная память и другие ресурсы. Инициатор–терминатор образует новую задачу, соответствующую новому шагу, передает управление интерпретатору шага в ней с указанием смещения по файлу управления заданиями во внутреннем представлении. Интерпретатор шага продолжит интерпретацию задания.

Объекты файловой системы ОС ОМ хранятся на внешней памяти ОМ, которая является двухуровневой: первичная внешняя память (интегральная массовая память) и вторичная (дисковая память). Основное назначение массовой памяти — сглаживание дисбаланса между скоростью работы процессора и темпом обмена с дисковой памятью. Особенный выигрыш при использовании массовой памяти может быть получен при реализации непоследовательных методов доступа – прямых, индексных, ключевых и т.п.

Максимально выигрывают от использования массовой памяти получают задачи, обрабатывающие большие массивы информации с произвольным доступом к элементам массивов данных (например по строкам, столбцам, диагоналям матрицы т.д.).

Обычный режим работы — обработка активных файлов в массовой памяти, предварительно переписанных в нее из дисковой памяти. Существует также возможность работы и с файлами на дисковой памяти, например с файлами последовательной организации.

Принципиальной особенностью массовой памяти является наличие процессора управления ее работой. Процессор массовой памяти может обеспечивать выполнение передачи данных между ней и оперативной памятью по любому алгоритму выборки ячеек, например, передавать строки, столбцы или диагонали матриц. Это повышает общую эффективность работы системы, так как освобождает процессор ОМ от рутинной работы, уменьшает объем информации, обмениваемой с внешней памятью, поскольку поток данных между ОМ и массовой памятью будет содержать только полезную информацию.

Основная машина по каналам связана с внешними машинами. ОС ОМ по этим каналам может связываться с программным обеспечением на ВМ, в частности через него получать доступ к файлам в архивах внешних машин.

Функции файловой системы: • именование объектов • распределение внешней памяти • хранение объектов на внешней памяти и перемещение объектов между уровнями внешней памяти (массовой и дисковой), защита объектов от несанкционированного доступа • синхронизация доступа к объектам • сохранность объектов при авариях внешней памяти и отказах системы • реализация различных методов доступа к файлам • поддержка контрольных точек.

Файловая система ОС ОМ состоит из системной задачи "Архив", псевдозадач "Обмен с массовой памятью" и «Обмен с дисковой памятью», кластеров методов доступа к файлам, выполняющихся в адресных пространствах задач пользователя. В файловой системе всякому методу доступа соответствует свой кластер управления файлом данной структуры.

Расположение части файловой системы в задаче пользователя (кластеров, реализующих доступ к файлам и содержащих буферы для обмена с внешними устройствами) объясняется стремлением снизить накладные расходы по доступу к отдельной записи файла. Реализация задачи "Обмен" со статусом псевдозадачи объясняется тем, что для задания обменов по каналам требуется привилегированный режим работы программы.

Совокупность объектов файловой системы на внешней памяти является деревом. Узлы дерева представляют справочники, листья — файлы или пустые справочники, корень – корневой справочник файловой системы. Каждый элемент дерева имеет имя, которое уникально относительно предыдущего узла. Любой объект дерева однозначно именуется полным составным именем относительно корня.

Перемещением объектов с одного уровня внешней памяти на другой, выполняя команды открытия файлов, заведует задача "Архив". При закрытии файлов, если они располагались в массовой памяти, модифицированные объекты отображаются на дисковую память, освобождая массовую. Если при этом велась работа только на чтение, массовая память освобождается без отображения объекта на дисковую.

Для того чтобы файл стал доступен некоторой задаче, необходимо открыть его. Если файл открывается в массовой памяти, то выполняются следующие действия: выделяется пространство в массовой памяти, файл переписывается с дисковой памяти в массовую, задача "Обмен с массовой памятью" настраивается на данную область в массовой памяти. Если файл открывается на диске, задача "Обмен с дисковой памятью" настраивается на те области на диске, где находится требуемый файл.

При открытии файла по полному составному имени задача "Архив", начиная от корня, просматривает всю последовательность справочников, выбирая на каждом следующем шаге соответствующий элемент по имени. Существует возможность использования коротких имен, которую поддерживает в "Архиве" среда поиска, задаваемая последовательностью имен справочников. При обращении по короткому имени поиск осуществляется последовательно по указанному списку. При инициации задания устанавливается стандартная среда поиска, которую можно сменить соответствующим обращением к "Архиву".

Взаимодействие компонентов программных систем, в том числе операционных систем, работающих в разных машинах происходит по динамически устанавливаемым соединениям. На каждое установление соединения приходится несколько обменов информацией по этому соединению. В период взаимодействия компоненты посылают друг другу сообщения.

Характер взаимодействия между компонентами относится к типу «удаленный вызов процедур». То есть, из пары взаимодействующих по соединению компонентов один является активной стороной, и инициатива в посылке сообщений принадлежит ему. Он посылает запросы, которые пассивный компонент обрабатывает и отвечает на них. Архитектура сетевого обеспечения локальной сети ВС с прямым подключением внешних ЭВМ, то есть с подключением каждой внешней ЭВМ через отдельный канал к основной машине, соответствует нижним уровням архитектуры эталонной модели открытых соединений МОС. Сетевой уровень как уровень маршрутизации отсутствует. Часть функций канального уровня выполняется аппаратурой. Транспортировка информации между ОМ и ВМ происходит при помощи высокоскоростного канала. В связи с тем, что надежность работы какала велика, принята схема работы без подтверждений на нижнем уровне о приеме сообщений. В случае сбоя последствия устраняются за счет верхнего уровня. Цена восстановления при такой схеме– выше, чем при схеме с подтверждениями на нижнем уровне, но при нормальной работе (то есть при отсутствии сбоев) экономится по одному прерыванию ОМ на каждое сообщение.

Функции транспортного уровня состоят в установлении транспортного соединения между точками доступа, в транспортировке данных по этому соединению, в управлении потоком данных.

Сетевое программное обеспечение локальной сети ВС создает базовый транспортный уровень, выше которого строятся служебные и прикладные функциональные системы. Таковой является система пакетной обработки заданий на ОМ. Пользователь, желающий выполнить свое задание на ОМ, должен сформировать пакет с заданием. Для этого в его распоряжении на ВМ имеются редакторы, библиотеки файлов и пр. Сформированный пакет состоит из заголовка и набора файлов, первый из которых является файлом с программой задания, написанной на языке управления заданиями ОС ОМ. Этот пакет передается, как файл, служебной задаче передачи заданий на ВМ. Задача передачи заданий на ВМ устанавливает транспортное соединение со служебной задачей ввода заданий в ОМ. Задача ввода заданий в ОМ не является резидентной, она образуется при попытке установления соединения с ней.

Между задачей передачи заданий в ВМ и задачей ввода заданий в ОМ устанавливается два транспортных соединения. В одном из них активной стороной является задача передачи заданий в ВМ, в другой – задача ввода заданий в ОМ. По первому соединению происходит передача файла с пакетом, по второму – обмен служебными сообщениями. В состав набора служебных входят сообщения: "приглашение", разрешающее передачу; "сброс", извещающее о происшедшей ошибке и требующее повторного ввода; "квитанция", подтверждающее прием пакета и постановку его в очередь на выполнение. Задача передачи заданий в ВМ после получения "квитанции" сообщает пользователю, используя системные средства ВМ, о вводе его задания.

Файл с пакетом задания может быть запомнен в библиотеке ВМ и использован при попытке повторной передачи пакета, если ОМ будет перезагружена до вывода результатов.

Протокол вывода результатов выполнения задания симметричен протоколу ввода задания.

Базовая система программирования ОМ

Базовая система программирования на языке ассемблера, позволяет вручную писать высокоэффективные программы, в полной мере учитывающие особенности архитектуры ОМ. Она содержит макроассемблер, кластерный оверлейный загрузчик, символьный отладчик.

Язык макроассемблера позволяет программисту выражать в символьной форме все функции центрального процессора ОМ.

Модули на языке ассемблера не должны вкладываться друг в друга. Предусмотрена возможность совместного ассемблирования группы модулей, причем между модулями и перед первым модулем могут находиться описания и операторы, образующие глобальный контекст. Внутри любого модуля может существовать свой локальный контекст, причем он может как дополнять глобальный контекст, так и переопределять его. Язык ассемблера предоставляет широкий набор макросредств.

Особенностью базовой системы программирования ОМ является возможность использования абстрактных типов данных, поддерживаемая ассемблером, загрузчиком и языком управления заданиями ОС. Механизм абстракций (описанный выше механизм «кластеров») был разработан для обеспечения независимости программ от операционной среды выполнения; он широко использовался при реализации операционной системы ОМ и, в частности, ее файловой системы.

Независимость программ от операционной среды выполнения основывается на понятии эквивалентных кластеров. У них одинаковые наборы операций, но могут быть различные внутренняя структура и реализация операций. Программы, хранящиеся в виде объектных модулей, можно компоновать с теми или иными эквивалентными кластерами, не изменяя исходной программы.

Кластерный оверлейный загрузчик предназначен для формирования двоичного образа рабочей программы, пригодного для хранения в библиотеке, а также для непосредственной записи в оперативную память и последующего выполнения.

Результатом работы любого транслятора с языка высокого уровня и ассемблера является последовательность объектных модулей, запоминаемая в библиотеке модулей. Формирование двоичного кода, предназначенного для выполнения процессором, производится загрузчиком путем редактирования связей и настройки на конкретные физические адреса объектных модулей.

Загрузчик извлекает из библиотек все объектные модули, необходимые для разрешения внешних ссылок, порождает требуемые кластеры, используя объектные модули соответствующих классов в качестве прототипов, распределяет память для всех полученных объектов. Приписав адреса всем внешним ссылкам и внутренним объектам каждого модуля, загрузчик осуществляет загрузку кода.

В общем случае загружаемые модули состоят из обычных модулей и модулей–классов. Модуль–класс копируется для всех кластеров, порождаемых по этим классом. Кластеры делятся на глобальные и локальные. Глобальный кластер доступен из любого модуля программы, а локальный – только из модуля, в котором определен.

Передача информации от ассемблера или других трансляторов загрузчику осуществляется через объектные модули, которые хранятся в библиотеках модулей. Для каждого объекта компиляции (модуля, программы, файла) ассемблер или транслятор генерирует один или несколько объектных модулей.

Объектный модуль состоит из заголовка, таблиц и тела модуля. Заголовок модуля содержит специальный признак, указывающий, является ли данный модуль обычным модулем или модулем–классом. Кроме этого признака заголовок включает в себя дескрипторы всех таблиц и некоторую другую информацию. Тело модуля со-

стоит из порций определенного размера, называемых фрагментами, каждый из которых содержит кодовую часть и командную информацию, интерпретируя которую, загрузчик осуществляет редактирование и загрузку кода из текущего фрагмента.

Загрузчик ОМ извлекает из библиотек все модули и модули–классы, необходимые для порождения кластеров и разрешения внешних ссылок, и осуществляет редактирование связей между модулями. Порождение кластеров происходит путем копирования загрузчиком модулей–классов с подстановкой указанных фактических параметров.

Результатом работы загрузчика является файл, содержащий двоичный код, не допускающий в дальнейшем настройки адресов. Перемещаемость этого кода обеспечивается аппаратным базированием. Результат работы загрузчика может быть переписан из файла на любой участок памяти, где он сможет выполняться после установки начало этого участка в регистр базового адреса.

Если для выполнения программы требуется больше оперативной памяти, чем может быть выделено в ее распоряжение, загрузчик ОМ позволяет организовать разбиение программы на части, называемые разделами и сменяющие друг друга в памяти во время работы. Каждый раздел может состоять из одного или нескольких модулей или кластеров. При этом связывание модулей, кластеров и настройка на конкретные адреса всех разделов производится загрузчиком до начала выполнения программы (так называемый статический оверлей).

- Символьный интерактивный отладчик ОМ обеспечивает выполнение следующих требований:
- наличие режима имитации, позволяющего осуществлять более детальный контроль за выполнением программы, чем это возможно с помощью одних лишь аппаратных прерываний;
- возможность управления режимом реализации прерываний и перехода от режима аппарат ной реализации к режиму имитации и обратно;
- возможность задавать условия прерывания, связанные с обращениями не только к памяти, но и к регистрам, так как регистры векторно–конвейерной ЭВМ могут использоваться для дли тельного хранения переменных;
- возможность обратного прослеживания значений переменной (то есть выяснения изменений значения переменной до достижения текущего значения) для переменных, хранящихся на регистрах;
- возможность замеров времени выполнения отдельных участков программы.

Контроль за выполнением отлаживаемой программы пользователь осуществляет с помощью отладочных операторов, которые можно разбить на следующие группы:

- планирование сеанса;
- управление выполнением программы;
- задание и отмена условий прерываний;
- управление выдачей информации об отлеживаемой программе;
- изменение значений регистров и переменных;
- задание трассировок;
- замеры времени выполнения отдельных учас тков программы;
- определение процедур отладки и нестандарт ных действий;
- установка контекста и переименования;
- –работа с архивом.

Для ОМ отладка программ часто включает в себя не только выявление и устранение ошибок, но и "подгонку" отдельных наиболее критичных участков программы под архитектуру процессора для более быстрого выполнения. Поэтому отладчик предоставляет возможность измерять время выполнения отдельных участков программы.

Система программирования на языке ФОРТРАН–77

Основными компонентами, поддерживающими программирование в этой системе являются компилятор программ с языка ФОРТРАН–77 и административная система управления операциями ввода/вывода.

Компилятор с языка ФОРТРАН–77 для ОМ является прямым. Это позволяет эксплуатировать его как на основной машине, так и в рамках имитационного комплекса.

Результаты работы компилятора могут быть получены как в виде загрузочных модулей, так и в виде текстового файла, содержащего результирующие программы на языке ассемблера ОМ.

Компилятор имеет три уровня оптимизации рабочих программ. Применение описываемых режимов оптимизации позволяет в значительной степени задействовать все преимущества векторно–конвейерной архитектуры, которые обеспечивают высокую производительность процессора. К наиболее важным особенностям устройства процессора относятся многоуровневая регистровая память, конвейеризованные функциональные устройства и векторная обработка данных. Каждый из описываемых уровней оптимизации ориентирован на эффективное использование этих свойств аппаратуры.

Нулевой уровень оптимизации выполняется в компиляторе по умолчанию и не имеет режимов управления. Первый и второй уровни оптимизации включаются пользователем посредством задания режимов компиляции. На нулевом уровне компилятором выполняются машинно–зависимые приемы оптимизации генерируемого кода\ Планирование выполнения потока команд параллельными конвейерными функциональными устройствами

составляет задачу первого уровня оптимизации. Второй уровень преобразует выполнение соответствующих итеративных циклов в векторный код и называется распараллеливанием циклов и векторизацией.

Система программирования на языке Си

Система программирования языка Си включает в себя препроцессор, компилятор и библиотеку стандартных функций, определенных в Стандарте языка. Система программирования на языке Си почти целиком написана на этом языке.

Препроцессор языка Си осуществляет макроподстановку, условную макрогенерацию, а также обеспечивает включение текстов именованных файлов в компилируемую программу. На вход препроцессора поступает текст исходной программы на языке Си, который после обработки препроцессором уже не содержит директив препроцессора. Эта программа затем компилируется. Использование директив препроцессора сокращает текст исходной программы и тем самым экономит время разработчиков.

Компилятор языка Си (далее компилятор) предназначен для преобразования Си–программы, полученной после обработки препроцессором, в загрузочный модуль ОМ или в программу на языке ассемблера ОМ. Кроме того, компилятор проверяет корректность транслируемой Си–программы, и, если в этой программе имеются ошибки, печатает информацию о характере и месте ошибки (в терминах исходной программы). В случае обнаружения ошибок формирование загрузочного модуля (программы на языке ассемблера) не производится. Компилятор может выдавать также предупредительные сообщения, которые свидетельствуют не об ошибках в программе, а о случаях некорректного (но допускаемого языком Си) использования конструкций этого языка.

Язык Си не имеет операторов ввода/вывода, управления памятью и других операторов, обеспечивающих взаимодействие с операционной средой выполнения. Взаимодействие программы с операционной средой осуществляется с помощью библиотеки стандартных функций, которая включает в себя следующие библиотеки: ввода/вывода, математических функций, обработки символов, обработки строк, обработки локализаций, управления сигналами.

Система программирования на языке Паскаль

Система включает в себя в качестве компонент компилятор программ с языка Паскаль, а также административную систему поддержки работы Паскаль–программ на ОМ. Результатом работы компилятора с языка Паскаль является стандартный загрузочный модуль ОМ.

Компилятор реализует стандартное множество языка Паскаль, расширенное, в основном, средствами независимой компиляции Паскаль–программ.

При реализации административной системы языка Паскаль были использованы функции стандартной библиотеки языка Си. Поэтому можно считать, что среда программирования языка Паскаль погружена в среду программирования языка Си. Паскаль–программы и Си–программы имеют одинаковый интерфейс межпроцедурных взаимодействий.

Имитационный комплекс основной машины

Имитационный комплекс ОМ предоставляет пользователю средства программирования ОМ путем использования инструментальной машины "Эльбрус 1КБ" или IBM PC/AT. Обеспечиваются возможности трансляции, загрузки, интерпретации, диалоговой и пакетной отладки программ, написанных на языке ассемблера ОМ. Файлы пользователя хранятся в специальным образом организованных областях на внешней памяти инструментальной ЭВМ – архивах, причем обеспечивается доступ к файлам по именам.

Директивы имитационного комплекса обеспечивают:

- создание архива;
- печать справочника архива;
- создание, ликвидацию, печать файла в архиве;
- копирование файла;
- создание, обновление, печать пакета модулей;
- создание и модификацию библиотеки модулей;
- загрузку модулей из библиотеки в пакет;
- печать справочника библиотеки;
- перенос пакета модулей из памяти в архив и обратно;
- трансляцию с языка ассемблера;
- загрузку пакета модулей;
- интерпретацию.

Стандартное прикладное программное обеспечение

Стандартное прикладное обеспечение ВС "Электроника ССБИС" организовано в виде расширяемого множества эффективно реализованных подпрограмм, объединенных в библиотеки и пакеты. Его состав охватывает как наиболее часто встречающиеся общеупотребительные задачи численного анализа и дискретной математики, так и задачи, связанные с отдельными предметными областями. Разработка стандартного обеспечения отделена от пользователя и является неотъемлемой частью реализации всего проекта создания новой машины.

Отличительной особенностью разработки в целом является то, что создание стандартного прикладного обеспечения осуществлялось одновременно с разработкой аппаратуры и системного обеспечения, поэтому тестирование, отладка и оптимизация разрабатываемых программ велись на имитационном комплексе основной машины.

Разработка стандартного прикладного обеспечения для векторно–конвейерной ЭВМ "Электроника ССБИС" была начата с ядра, охватывающего первоочередные задачи общего назначения. К их числу относилось создание следующих библиотек и пакетов программ.

- Библиотека стандартных подпрограмм вычисления элементарных математических функций (108 программных модулей, макро ассемблер). Отдельные разделы:
    - вычисление элементарных функций вещественного аргумента (скалярный и векторный варианты);
    - вычисление элементарных функций комплексного аргумента (скалярный и векторный варианты);
    - –вычисления с двойной точностью (арифметические операции, скалярный и векторный варианты вычисления элементарных функций).
- Библиотека стандартных программ вычисления специальных функций скалярного и векторного вещественного аргумента (27 программных модулей, макроассемблер и ФОРТРАН).
- Библиотека элементарных операций линейной алгебры СПЛАВ (17 программных модулей, макроассемблер, вещественные и комплексные аргументы). Отдельные разделы:
    - скалярное произведение векторов;
    - умножение вектора на скаляр и сложение с другим вектором;
    - копирование вектора;
    - перемещение векторов;
    - вычисление евклидовой нормы;
    - вычисление суммы абсолютных значений элементов вектора;
    - умножение вектора на константу (масштабирование);
    - вращение Гивенса;
    - – поиск максимального по абсолютному значению элемента вектора.
- Библиотека программ нечисловой обработки (36 программных модулей, макроассемблер).
- Библиотека программ внутренней сортировки.
- Библиотека стандартных программ генерирования псевдослучайных чисел (10 программных модулей, макроассемблер, скалярный и векторный варианты).
- Библиотека стандартных программ элементарной статистики, предварительная обработка данных, вычисление функций распределения и критериев согласия (32 программных мо дуля, макроассемблер).
- Библиотека программ интерполирования, аппроксимации и сглаживания функций (12 программных модулей, макроассемблер).
- Библиотека стандартных программ численного интегрирования (12 программных модулей, макроассемблер и ФОРТРАН).
- Библиотека программ решения систем обыкновенных дифференциальных уравнений первого порядка (макроассемблер и ФОРТРАН).
- Библиотека программ решения интегральных уравнений (6 программных модулей, IIJl–1 и макроассемблер).
- Библиотека базовых программ решения задач комбинаторной вычислительной геометрии (11 программных модулей, макроас семблер).
- Библиотека программ решения задач на графах (21 программный модуль, макроассемблер).
- Пакет программ быстрого преобразования Фурье (37 программных модулей, ПЛ/1 и макроассем блер).
- Пакет программ решения задач линейной алгебры с заполненными квадратными и прямо угольными матрицами общего вида, симметричными и треугольными матрицами (50 программных модулей, ФОРТРАН и макроассемблер).
- Пакет программ решения задач линейной алгебры с использованием внешней полупроводниковой памяти (30 программных модулей, ФОРТРАН и макроассемблер).
- Пакет программ решения задач на собственные значения для заполненных симметричных положительно определенных матриц (4 программных модуля, ФОРТРАН и макроассемблер).
- Пакет программ решения задач линейной алгебры с разреженными матрицами (29 программных модулей, ФОРТРАН и макроассемблер).
- Пакет программ решения задач линейной алгебры с ленточными матрицами прямыми методами (38 программных модулей, ФОРТ РАН и макроассемблер).
- Пакет программ решения систем линей ных уравнений с ленточными матрицами ите рационными методами (12 программных модулей, ФОРТРАН
- Пакет программ решения задач спектрального анализа для симметричных трехдиагональных матриц (8 программных модулей, ФОРТРАН и макроассемблер).

- Пакет программ решения задач оптимального распределения ресурсов на сетях большой размерности (39 программных модулей, ФОРТРАН и макроассемблер).

В создании описанного программного продукта, получившего название Научная Библиотека "Электроники ССБИС" помимо сотрудников Отдела базового программного обеспечения Института проблем кибернетики РАН принимали участие сотрудники Института математики АН Беларуси и кафедры хибернетики Московского института электронного машиностроения.

Список литературы

1. Программное обеспечение высокопроизводительной системы // Вопросы кибернетики. М., 1986. Вып. 127.
2. Системы программирования векторно–конвейерной ЭВМ // Вопросы кибернетики. М., 1990. Вып. 162.
3. Архитектура высокопроизводительной вычислительной «Электроника ССБИС/1» // Программные продукты и системы. 1991. № 1.
4. Системное программное обеспечение основной машины ВС «Электроника ССБИС» // Программные продукты и системы. 1991. № 1.
5. Стандартное прикладное программное обеспечение основной машины ВС «Электроника ССБИС» // Программные продукты и системы. 1991. № 1.