ИНСТИТУТ ЯДЕРНОЙ ФИЗИКИ СО АН СССР

В.Н. Козлов, Ю.И. Мерзляков,
И.А. Ткаченко, А.Г. Чертовских

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРОЦЕССОРА АП-32.

3. Реализация языка высокого уровня ФОРТРАН-IV

ПРЕПРИНТ 90-35

НОВОСИБИРСК

Программное обеспечение процессора АП-32.

3. Реализация языка высокого уровня ФОРТРАН-IV

В.Н. Козлов, Ю.И. Мерзляков,
И.А. Ткаченко, А.Г. Чертовских

Институт ядерной физики
630090, Новосибирск 90, СССР

АННОТАЦИЯ

Описан компилятор языка высокого уровня ФОРТРАН-IV для процессора АП-32, а также исполнительная система и библиотеки фортрановских программ.

6. РЕАЛИЗАЦИЯ ЯЗЫКА ВЫСОКОГО УРОВНЯ ФОРТРАН-IV

6.1. Выбор языка и способ его реализации

АП-32 — универсальный арифметический процессор. Это означает, что система его команд [1] не ориентирована на какие-то специализированные вычисления и позволяет программировать произвольные алгоритмы. Таким образом, в плане программирования АП-32 не отличается от любой универсальной ЭВМ для которой множество рабочих программ не сводится к библиотеке вычислительных алгоритмов, поставляемых разработчиками вместе с этой ЭВМ (как в случае спецпроцессоров), а разрабатывается самими пользователями сообразно их задачам. Тогда неизбежно встает вопрос о языке программирования высокого уровня как необходимой компоненте современной ЭВМ.

Сегодня можно назвать ряд популярных языков высокого уровня — ФОРТРАН, С, ПАСКАЛЬ, МОДУЛА-2, АДА и др. В задачах физического эксперимента в ИЯФ (как и во всем мире) предпочтение традиционно отдается ФОРТРАНу. Большинство программ ИЯФ написано на ФОРТРАНе-IV (ФОРТРАН-77 начали применять относительно недавно), а применение программ, написанных в других физических центрах, весьма ограничено спецификой экспериментальных установок. Все это и предопределило выбор ФОРТРАНа-IV в качестве основного языка высокого уровня для АП-32.

Следует отметить, что единого ФОРТРАНа для всех ЭВМ не

существует из-за особенностей каждой ЭВМ, на каждой из них реализован его диалект. Для диалектов является общим соответствие американскому стандарту ANSI X3.9-1966. Особенности, связанные с архитектурой, системой команд и операционными системами, проявляются в расширениях синтаксиса ФОРТРАНа (например, вводятся операторы для работы с файловой системой OPEN и CLOSE, новые типы данных и битовые операции), предоставляющих дополнительные относительно стандарта возможности. Некоторые особенности ЭВМ (длина машинного слова или порядок нумерации байтов в слове) приводят к непереносимости программ между разными машинами.

Для АП-32 также был разработан диалект ФОРТРАНа. В его основу был положен язык, соответствующий стандарту и включающий 16 пунктов расширений [2]. Этот язык использован на ЭВМ семейства Электроника/СМ, он почти полностью совместим с ФОРТРАНом ЕС ЭВМ (различия касаются некоторых расширений стандарта для этих семейств). Таким образом, этот язык позволяет переносить программы, разработанные на этих ЭВМ, на АП-32 и обратно на уровне текстов.

Диалект ФОРТРАНа для АП-32 также имеет особенности, связанные с системой команд АП-32. Например, отсутствие байтовой адресации и байтовых операций в АП-32 привело к отказу от данных типа BYTE или LOGICAL*1, которые, впрочем, отсутствуют и в упомянутом стандарте. Текстовые переменные (константы Холлерита или литералы) тем не менее в нашем языке существуют.

Для программирования на ФОРТРАНе-IV любой ЭВМ необходимо иметь два программных средства — компилятор и исполнительную систему. Компилятор предназначен для трансляции текстов программы в машинный код, а исполнительная система является обязательной частью любой фортрановской программы, создающей операционную среду при исполнении этой программы.

Для трансляции программ, написанных на ФОРТРАНе АП-32, в объектный код АП-32 был разработан компилятор, названный FORTAP. Компилятор генерирует прямые машинные коды АП-32 для большинства арифметико-логических операций с фиксированной и плавающей запятой. При создании FORTAP были использованы общая схема и отдельные машинно-независимые модули компилятора ЭВМ семейства Электроника/СМ, что позволило сократить время разработки.

Задача создания исполнительной системы сводилась к написанию библиотек функций и подпрограмм, к которым явно и неявно

обращается фортрановская программа. Эти обращения включаются компилятором в генерируемый код. Библиотека названа APLIB, в нее вошли подпрограммы ввода/вывода, файловых операций, вычисления встроенных функций и другие. Сложность задачи состояла в отсутствии у АП-32 канала ввода/вывода в общепринятом понимании, а также операционной системы, посредством которых можно осуществлять операции ввода/вывода [3]. Как следствие этого, необходимо было распределить исполнительную систему ФОРТРАН между АП-32 и ЭВМ «накачки» (ЭВМН), которая его обслуживает. Часть исполнительной системы в АП-32 формирует запросы на ввод/вывод, а другая — в ЭВМН — исполняет эти запросы относительно прикрепленных к ней внешних устройств. Это взаимодействие подробно описано в [4]. В результате, кроме библиотеки подпрограмм АП-32, потребовалось разработать специальное программное обеспечение ЭВМН.

6.2. Общее описание компилятора

Компилятор FORTAP — это программа, работающая на ЭВМ семейства Электроника под ОС RT-11 или совместимыми с ней ОС TSX-Plus и RT-эмуляторами. Компилятор в рамках принятого подхода к программированию АП-32 является кросс-средством, т. е. работает на ЭВМН и производит объектные коды АП-32. Написан компилятор на языке ассемблера MACRO-11 [5].

Главной проблемой при написании больших программ для ЭВМ Электроника является ее малый объем прямо адресуемой памяти (64 Кбайт). Это обстоятельство вынуждает создавать оверлейные программы и тратить много сил на их оптимальную компоновку. Организация FORTAP в значительной степени определена тем же.

Компилятор FORTAP построен таким образом, что исходная программная единица (основная программа, подпрограмма или подпрограмма-функция) считывается с внешнего носителя только один раз и помещается в оперативную память, а процесс трансляции разбит на фазы, выполненные как оверлейные сегменты. Один за другим загружаясь в память, они производят обработку текста и генерацию кода. Всего компилятор насчитывает 17 фаз. В каждой фазе происходит один или несколько проходов (просмотров и преобразований программы). Результатом деятельности компилятора является файл объектного кода АП-32 и файл листинга, включающий исходный текст с возможными сообщениями об

ошибках, карту распределения переменных и массивов и листинг сгенерированных машинных кодов АП-32.

В процессе работы компилятор должен выполнить анализ исходной программы и синтез соответствующей объектной программы. Логически всю деятельность компилятора можно разбить на несколько этапов:

1) инициация;
2) лексический анализ;
3) синтаксический анализ;
4) подготовка к генерации кода;
5) машинно-независимые оптимизации;
6) генерация кода.

Деление это достаточно условно, так как многие процессы исполняются параллельно, с определенной взаимной синхронизацией. Например, отдельные моменты генерации кода имеют место уже на этапе синтаксического и даже лексического анализа, что позволяет повысить скорость трансляции и оптимизировать распределение памяти и размер оверлейного сегмента.

6.2.1. Инициация

Распределение оперативной памяти компилятора изображено на рис. 1. Память делится на две компоненты: статическую, определяемую при построении компилятора, и динамическую. Динамическая память располагается выше верхнего адреса статической памяти и занимает максимально возможное пространство, выделяемое операционной системой.

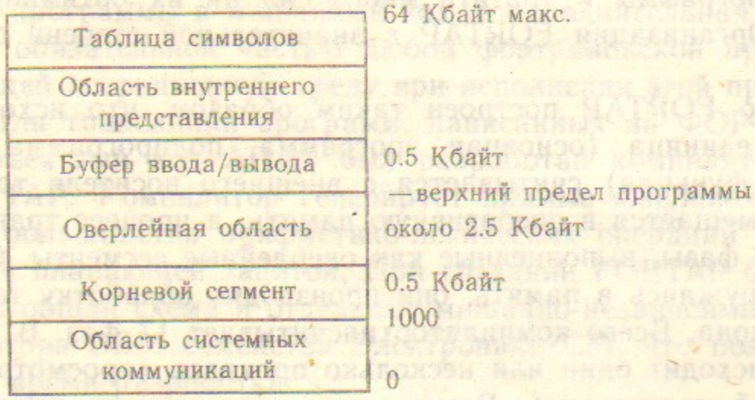

Рис. 1.

После запуска компилятора происходит распределение динамической памяти на буфер ввода/вывода, буфер внутреннего представления транслируемой программы и таблицу символов. При инициации двум последним областям отводится равное пространство, в дальнейшем граница между ними может перемещаться в ту или иную сторону. Область, отведенная для таблицы символов, заполняется в сторону уменьшения адресов, а область внутреннего представления — в сторону увеличения. Программная единица (основная программа, подпрограмма или подпрограмма-функция) считывается с внешнего устройства только один раз и полностью помещается в буфер внутреннего представления. Главное достоинство такого распределения памяти — возможность организации многопроходной работы без многократного обращения к внешним устройствам, что, естественно, повышает производительность компилятора.

Располагая оперативной памятью в 56 Кбайт, можно компилировать программные единицы размером около 2000 строк. В случае перекрытия таблицы символов и области внутреннего представления компилятор прекращает работу по ошибке, и в этом случае следует уменьшить размер программной единицы.

В буфере ввода/вывода накапливается необходимая для пересылки на внешнее устройство информация. На начальных этапах это генерируемые компилятором программные секции данных, векторы массивов, константы, а затем и сгенерированный объектный код.

При инициации происходит также разбор командной строки компилятора и установка значений глобальных переменных.

6.2.2. Лексический анализ

На этом этапе сканируется исходная программа с целью приведения текстовых строк операторов языка, состоящих из цепочек литер переменной длины, к виду, удобному для последующей обработки. При этом символы-разделители (ключевое слово, знак операции или знак пунктуации) преобразуются в числовые коды внутреннего представления. Имена переменных, массивов и т. д. заносятся в таблицу символов, а во внутреннем представлении заменяются ссылками на соответствующие элементы таблицы символов. На этой же стадии обнаруживаются грубые ошибки (недопустимые знаки, неравное количество открывающихся и закрывающихся скобок и т. д).

Операторы исходного текста, таким образом, преобразуются в

кодированные записи, и в дальнейшей работе участвуют именно эти структурные единицы. Информация о метках операторов помещается в таблицу символов.

6.2.3. Синтаксический анализ

Во время синтаксического анализа операторы программы проверяются на принадлежность языку ФОРТРАН/АП-32 и происходит окончательное преобразование текстовых данных в коды и частичная генерация объектного кода.

Операторы языка ФОРТРАН делятся на исполняемые и неисполняемые, их обработка происходит раздельно. В первую очередь разбираются неисполняемые операторы в следующей последовательности:

1) декларации размерности и типов данных;
2) конструкции EQUIVALENCE и COMMON;
3) операторы DATA и FORMAT;
4) создаются векторы многомерных массивов.

Извлеченная информация заносится в таблицу символов, в объектный файл записываются директивы резервирования памяти данных.

В процессе обработки из исполняемых операторов извлекаются численные константы и заменяются ссылками на таблицу символов. Элементы синтаксиса, имеющие неоднозначную семантику (например, скобки), в результате разбора получают однозначное толкование и заменяются кодами. В качестве основного метода синтаксического разбора применен рекурсивный спуск [6].

6.2.4. Подготовка к генерации кода

В результате работы предыдущих фаз получена вся необходимая информация о типах идентификаторов, содержимом COMMON-блоков и конструкций EQUIVALENCE, сгенерированы программные секции для данных и получена информация об операторах, содержащих ошибки. Таким образом, созданы все условия для преобразования внутреннего представления к виду, пригодному для генерации кода.

На данном этапе осуществляется перевод скобочной инфиксной записи исполняемых операторов в более удобную бесскобочную постфиксную (польскую) запись, обеспечивающую порядок действий в операторе на основе стекового механизма [6]. Так, выражение X=A*(B+C/D) в польской постфиксной записи будет выглядеть как XABCD/+*=. Однако, подобная запись еще недостаточно приспособлена для генерации кодов, поэтому она преобразуется в последовательность тетрад (четверок), более близких к типичным машинным командам или простым группам команд [6]. Тетрада в общем виде выглядит следующим образом (угловые скобки служат для разделения):

⟨оператор⟩⟨операнд1⟩⟨операнд2⟩⟨результат⟩

Для описанного выше примера последовательность тетрад будет следующей:

⟨ / ⟩⟨ C ⟩⟨ D ⟩⟨результат1⟩
⟨ + ⟩⟨ B ⟩⟨результат1⟩⟨результат2⟩
⟨ * ⟩⟨ A ⟩⟨результат2⟩⟨результат3⟩
⟨ = ⟩⟨ X ⟩⟨результат3⟩⟨ ⟩

6.2.5. Машинно-независимые оптимизации

Оптимизации проводились на всех этапах компиляции. Сюда можно отнести

1) свертку констант (вычисление выражений, состоящих из целых констант, во время разбора);
2) вычисление базового адреса элемента массива в случае индексов-констант;
3) устранение лишних преобразований типов данных в выражениях.

После приведения к тетрадному представлению становится удобно проводить еще несколько машинно-независимых оптимизаций, которые включаются в процесс компиляции по требованию:

1) оптимизация циклов: операции умножения и индексирования элементов массивов, содержащие переменные циклов, сводятся к сложению;
2) удаление общих подвыражений: повторяющиеся выражения вычисляются в первый раз, в дальнейшем используется результат этого вычисления;
3) удаление тетрад, приводящих к лишним пересылкам в память, и упрощение операций путем эквивалентных преобразований.

После этапа оптимизации начинается генерация кода.

6.2.6. Генерация кода

На этом этапе происходит окончательное формирование объектного кода данных и перевод внутреннего представления исходной программы (последовательности тетрад) на машинный язык

(объектный код и листинг на языке ассемблера АП-32). Генератор кода представляет собой набор процедур для интерпретации тетрад, распределения регистров АП-32 и формирования групп команд АП-32.

При генерации совершается от двух до трех проходов по внутреннему представлению (тетрадам). В первом из них по результату генерации команд разрешаются адреса переходов (меток исполняемых операторов). В каждом из следующих проходов формируется объектный код и листинг.

Каждый фортрановский оператор просматривается вперед до конца, чтобы выяснить, употребляются ли в нем функции, что влияет на распределение регистров. Затем для каждой тетрады вызывается отдельная процедура генерации кодов АП-32, подбирающая адекватную ей группу команд. Тетрады различаются по коду операции, типу операндов (прямо адресуемая память, непосредственная константа, стек, параметр подпрограммы и т. д), классу данных операндов (целые, реальные, логические, ...) и т. п., что приводит к большому разнообразию кода. Процедура распределения регистров АП-32 выделяет свободные регистры для формирования команд. Для приведенного выше примера компилятором FORTAP будет сгенерирован листинг следующего набора команд:

```
LD   R11, $DATA +000010   ; D → I-регистр R11
LD   R10, $DATA +000006   ; C → I-регистр R10
JSR  DIV$                 ; переход на п/п деления
ST   R10, $STACK+000002   ; рез-т деления → стек
LD   AD, $DATA +000004    ; B → сумматор ППЗ
LD   AR, $STACK+000002    ; стек → сумматор ППЗ
NOP                       ; ожидание результата
ST   AF, $STACK+000002    ; рез-т сложения → стек
LD   MP, $DATA +000002    ; A → умножитель ППЗ
LD   MR, $STACK+000002    ; стек → умножитель ППЗ
NOP                       ; ожидание результата
NOP
ST   MF,$DATA +000000     ; рез-т умножения → X
```

FORTAP генерирует прямые (inline) коды, т. е. непосредственно машинные команды. В ряде случаев, когда генерируемый код оказывался велик, а время его исполнения значительно по сравнению с временем вызова библиотечной подпрограммы, для экономии памяти принималось решение о помещении его в библиотеку в виде подпрограммы.

6.2.7. Средства диагностики ошибок компилятора

Компилятор FORTAP — большая программа, работающая как функция входных данных (исходной программы на ФОРТРАНе). Невозможно в разумное время проверить правильность работы FORTAP для всех возможных комбинаций операторов исходной программы. Поэтому, следуя общепринятой практике, в компилятор включили специальный модуль диагностики сбойных ситуаций. В случае сбоя FORTAP на экран терминала выводится исчерпывающая информация о его состоянии:

1) содержимое регистров процессора;
2) текущее распределение памяти FORTAP;
3) содержимое внутреннего представления кода;
4) содержимое таблицы символов;
5) восьмеричный дамп буфера ввода/вывода.

Информация представляет ценность для разработчиков при анализе ошибки. Этот же модуль использовался разработчиками в отладочных целях.

6.3. Особенности реализации компилятора

Качество компилятора определяется тем, насколько эффективна генерация кодов, т. е. насколько полученная последовательность команд близка к оптимальной по занимаемой памяти и времени исполнения. Эта эффективность связана со способами отображения внутреннего представления программы на систему команд конкретной ЭВМ. Для большинства современных ЭВМ, имеющих быструю сверхоперативную память (регистры) и медленную оперативную, центральное место занимает проблема размещения наиболее часто используемых данных в регистрах и минимизация числа обращений к памяти для общего увеличения производительности.

Архитектура и система команд АП-32, как отмечалось в [1], относится к типу RISC (машина с сокращенной системой команд). ЭВМ такого типа имеет ряд достоинств в плане разработки компиляторов:

1) команды несложны, что позволяет при генерации кода оптимально подобрать их нужную последовательность;
2) все арифметико-логические операции производятся на регистрах;
3) команды исполняются одинаковое время, т. е. команды обращения к памяти занимают то же время, что и регистровые операции.

Эти достоинства сглаживают проблему распределения регистров. Для АП-32 они еще усиливаются тем, что число регистров АП-32 сравнительно велико (32 I-регистра и более 10 F-регистров).

При программировании на ФОРТРАНе (особенно вычислительных задач) наибольший вклад в производительность вносят вычисления арифметических выражений и обращения к подпрограммам и функциям, и от оптимальности генерации их кодов зависит общая эффективность (другие операции, например, переходов, генерируются однозначно, а операции ввода/вывода некритичны по времени).

6.3.1. Генерация арифметических выражений

Для фортрановской арифметики характерно, что операнды извлекаются из памяти и результат также помещается в память. Промежуточные результаты вычислений во внутреннем представлении содержатся в стеке. Этот стек реализуется на конкретной аппаратуре целевой ЭВМ (аппаратный стек или память). В АП-32 аппаратного стека нет, и он моделируется на памяти.

Для АП-32 стек выражений реализуется по-разному, однако во всех случаях без моделирования указателя стека на регистре. Для целочисленной арифметики оказалось удобным реализовать стек непосредственно на I-регистрах, где уровень стека обозначается номером регистра и вычисляется при компиляции. Таким образом, все арифметико-логические операции внутри оператора выполняются исключительно на регистрах, обращения к памяти применяется только для необходимой загрузки операндов и результата. Такая машинно-зависимая оптимизация работает в пределах одного оператора ФОРТРАНа. Например, для оператора I=J+(K—L) генерируется

```
LD   R0,$DATA+000004 ;  загрузка K в стек R0
LD   R1,$DATA+000006 ;  загрузка L в стек R1
SUB  R1,R0           ;  стек R0 ← рез-т (K—L)
LD   R1,$DATA+000002 ;  загрузка J в стек R1
ADD  R1,R0           ;  стек R0 ← J + (R1)
ST   R0,$DATA+000000 ;  I ← результат из стека
```

При переходе от одного фортрановского оператора к другому I-регистры, принимающие участие в вычислениях, полностью освобождаются.

В отличие от целочисленной арифметики, стек операндов с плавающей запятой моделируется на памяти данных АП-32, так как F-регистры специализированы и не могут, подобно I-регистрам, служить регистрами операндов любых операций, что использовано в предыдущем случае. Здесь уровни стека соответствуют прямым адресам памяти, которые также вычисляются при компиляции, а обращение к стеку не требует вычислений адреса при исполнении программы и происходит за одну команду. Пример генерации кода для выражения с плавающей запятой приведен выше.

6.3.2. Обращение к подпрограммам и функциям

В ФОРТРАНе АП-32 есть несколько разновидностей подпрограмм и функций. Они реализуют ряд возможностей, обеспечивающих разный уровень сервиса ценой разных накладных расходов. Отличия состоят в способах передачи параметров, использования I-регистров и взаимодействия с исполнительной системой программы. Ниже приведены эти способы в порядке убывания сервиса и сокращения времени на обращение.

1. Фортрановские подпрограммы, функции и оператор-функции. При их вызове (кроме оператор-функций) происходит обращение к исполнительной системе, которая сохраняет о них полную информацию и выдает ее в случае программной ошибки. Адреса параметров, формируемые во время трансляции, передаются через специальную область памяти данных.

2. Ассемблерные подпрограммы и функции. Пользуются фортрановским способом передачи параметров. Обращение к исполнительной системе необязательно и может быть включено по необходимости. Для работы доступны 10 I-регистров, которые не нужно сохранять.

3. Ассемблерные системные функции. Имена должны начинаться на SYS. Исполнительная система не используется. Адреса параметров передаются через регистры. Применяются для программирования критичных по времени задач (например, машинная графика) и требуют сохранения и восстановления нужных I-регистров.

4. Стандартные функции. Вызываются без обращения к исполнительной системе, параметры передаются через регистры, издержки времени минимальны.

6.3.3. Типы данных

Основными компонентами операторов ФОРТРАНа АП-32 являются

1) константы;
2) переменные;
3) массивы;
4) ссылки к функциям;
5) выражения.

Каждая основная компонента представлена одним из перечисленных типов данных:

1. Целый. Реализуется двумя фортрановскими типами — INTEGER*2 (занимает машинное полуслово АП-32) и INTEGER*4 (слово АП-32). Представлен в дополнительном коде.
2. Реальный, соответствует REAL*4 (слово АП-32). Формат числа с плавающей запятой соответствует формату ЭВМ семейства Электроника/СМ (PDP-11 и VAX).
3. Логический, соответствует LOGICAL*4 (слово АП-32).

Байтовый тип (LOGICAL*1 или BYTE) в явном виде отсутствует, но применяется в литеральных или холлеритовых константах.

Комплексный тип (COMPLEX*4) и тип с удвоенной точностью (DOUBLE PRECISION) в данной версии языка не реализованы. Реализация требует незначительного расширения модулей генерации кодов компилятора и написания ряда библиотечных и стандартных функций.

6.4. Исполнительная система ФОРТРАН-программ АП-32

Исполнительная система (ИС) ФОРТРАН АП-32 создает операционную среду фортран-программы, предоставляя ей ряд возможностей по вводу/выводу, обработке ошибок, вычислению функций и т. д. Краткое описание системы и этих возможностей приведено в [4]. Исполнительная система написана на языке ассемблера АП-32 [7].

ИС включается в программу при сборке редактором связей, и при пуске программы ей передается управление. Система распределяет память данных задачи, инициирует служебные данные (рабочую область) и вызывает ФОРТРАН-программу как свою подпрограмму. По завершении программы управление возвращается обратно в ИС, которая сообщает ЭВМН о конце работы.

Во время работы программа может обращаться к подпрограммам ИС, получать необходимый сервис, а затем продолжать исполнение. При возникновении аварийной ситуации (аппаратная или программная ошибка) система обрабатывает ошибку.

ИС распределена между ЭВМН и АП-32 так, что все действия, зависящие от операционной системы ЭВМН (ввод/вывод, файловые операции), вынесены в фортрановскую программу ЭВМН по обслуживанию АП-32. Ее ИС работает «от имени» ИС АП-32.

Ниже описан состав и функционирование элементов системы.

6.4.1. Служебные данные ИС

Кроме данных программы (константы, переменные и массивы), в памяти размещаются также буферы и данные ИС, которые используются ее модулями.

1. Динамический буфер. Располагается выше данных программы и занимает пространство, выделяемое при сборке задачи. В нем при открытии файлов отводятся буферы ввода/вывода, это пространство освобождается при закрытии файлов.

2. Стек программы. В стеке находится информация об истории вызова текущей подпрограммы в виде списка структур, содержащих имя вызывающей подпрограммы, номер вызывающего фортрановского оператора и адрес возврата. Список описывает цепочку многократно вложенных вызовов и используется для возврата из подпрограмм и при обработке ошибок для вывода имен и номеров операторов всех вызывавших подпрограмм.

3. Таблица файловых дескрипторов. Содержит информацию обо всех одновременно открытых файлах (число записей, размер записи и т. п.).

4. Таблица состояний ошибок, позволяющая управлять принятием решения по ошибке (например, завершать ли программу при арифметической ошибке). Таблица на 18 ошибок доступна из программы.

5. Буфер запроса ЭВМН. Используется для обмена данными АП-32 — ЭВМН.

6. Рабочая область, содержащая номер текущего фортрановского оператора, адреса и указатели текущих позиций всех перечисленных структур памяти и области сохранения/восстановления I-регистров. Эта область инициируется при пуске программы и используется на всем протяжении ее работы.

7. Таблица часто используемых ИС констант.

6.4.2. Библиотека модулей ИС

Модули ИС помещены в библиотеку APLIB.OBT. Во время сборки программы только необходимые модули (не все) включаются в ее состав. Большая часть модулей полностью исполняется в АП-32, остальные связываются с ИС ЭВМН. Ниже описаны модули ИС, разбитые на группы по назначению.

1. Модуль инициации и завершения фортрановской программы и фортрановских подпрограмм и функций. Инициация программы описана выше. При входе в фортрановские подпрограммы модуль вызывается для заполнения стека программы и указателя на параметры, для функций сохраняется ряд I-регистров, на выходе стек освобождается.

2. Библиотечные и стандартные функции ФОРТРАН-IV. Написаны на языке ассемблера АП-32. Стандартные математические функции (SIN, ALOG и т. п.) опознаются компилятором и являются частью синтаксиса языка. Почти все они реализованы аппаратно-программным способом с применением блока СПЭФ АП-32 [8], что обеспечивает высокую скорость вычислений. Библиотечные функции (модуль числа, генератор случайных чисел и т. д.) составляют набор, существующий в любой реализации ФОРТРАНа.

3. Программная реализации некоторых арифметических операций. Сюда относятся, например, умножение и деление с фиксированной запятой и разные степенные функции, т. е. действия, не реализуемые аппаратно.

4. Подпрограммы файловых операций — открытие и закрытие файлов разных типов (с последовательным и прямым доступом, для форматного и бесформатного ввода/вывода, а также устройств без файловой структуры). Подпрограммы служат для формирования блоков параметров для ИС ЭВМН и их пересылки.

5. Подпрограммы ввода/вывода данных. Обслуживают операторы ввода/вывода типа READ и WRITE. Список данных для ввода/вывода этих операторов сканируется по одному элементу, для элемента выполняются, если нужно, форматные преобразования. Полученная информация для оператора типа WRITE копируется в выходную запись, которая после заполнения передается в ИС ЭВМН для физического вывода. Для оператора READ информация обрабатывается в обратном порядке — физический ввод с внешнего устройства в ЭВМН по запросу ИС АП-32, передача в АП-32, распаковка записи и копирование двоичных данных в переменные и массивы программы с возможным форматным преобразованием.

6. Подсистема форматных преобразований. Форматные преобразования типа «текст — двоичное представление» и наоборот производятся над списками ввода/вывода в соответствии с форматным дескриптором, задаваемым оператором FORMAT или форматом времени исполнения, содержащемся в массиве. В первом случае оператор формат преобразуется в таблицу управления преобразованием во время компиляции, во втором — во время исполнения, для чего в ИС написан модуль форматного транслятора. Полученная тем или иным способом таблица управления преобразованием используется модулем форматного процессора ИС, интерпретирующего эту таблицу и вызывающего конкретные подпрограммы преобразования для каждого элемента списка ввода/вывода. Все эти модули и подпрограммы реализованы в АП-32 и вызываются при обработке операторов типа READ, WRITE, ENCODE и DECODE.

7. Модуль трассировки программы и обработки ошибок. В начале исполнения каждого фортрановского оператора в рабочую область ИС заносится его номер, соответствующий номеру в листинге, выдаваемом компилятором. При переходе к подпрограмме он фиксируется в стеке программы. Если возникает ошибка, формируется текст сообщения о ней и сообщение об истории вызова подпрограммы. Эти сообщения передаются в ЭВМН для вывода на экран терминала. Сообщения о некоторых ошибках могут быть подавлены при помощи подпрограммы SETERR.

8. Модуль связи с ЭВМН. Применяется для формирования блока данных для запроса ЭВМН, запроса, анализа его результата и передачи информации в случае чтения в требующую связи подпрограмму АП-32. Блок данных состоит из заголовка (служебной информации), содержащего номер логического устройства, код запроса и некоторые параметры, и полезной информации. ИС ФОРТРАНа использует 16 кодов запроса, среди которых запросы на открытие/закрытие файла; поиск записи и позиционирование на запись вперед, назад и в начало файла; форматное и бесформатное чтение и запись; пауза, останов и обработка ошибки. Протокол обмена информацией и сам модуль связи может использоваться (и используется) не только ИС, но и другими системами обмена данными АП-32 — ЭВМН.

6.5. Переносимость ФОРТРАНА АП-32

В рамках принятого подхода, когда все программирование АП-32 ведется на специально подключенной к нему ЭВМН, вопрос о переносимости не имеет остроты, однако заслуживает внимания. Бо́льшая часть ИС ФОРТРАНа АП-32 написана на языке ассемблера АП-32 и вообще не зависит от ЭВМН, но компилятор FORTAP и часть программного обеспечения для эксплуатации программ АП-32 написаны на языке макроассемблера ЭВМ семейства Электроника/СМ, что ограничивает переносимость этим семейством, поэтому можно говорить о переносе только между разными ОС этих ЭВМ.

Компилятор слабо зависит от ОС ЭВМН, имея системную зависимость только в модулях инициации и ввода/вывода, которые локализованы и легко заменяются при переносе.

Системно-зависимые модули ИС находятся в составе программы ЭВМН по обслуживанию фортран-программ АП-32 и не подвергались модификации. Протокол связи ЭВМН—АП-32 выполнен в терминах фортрановских параметров и не зависит от ОС ЭВМН. Следовательно, при замене модулей ИС ЭВМН на аналогичные для другой ОС ЭВМН перенос произойдет автоматически (ФОРТРАН-IV реализован для всех ОС семейства Электроника/СМ: RT-11, RSX-11M и RSTS/E).

6.6. Вызов и прекращение работы FORTAP

Для вызова компилятора следует воспользоваться командой R FORTAP или RUN DEV:FORTAP для пуска с системного устройства или устройства DEV:, после чего он ожидает ввода строки команды. Формат командной строки следующий:

[FILE.OBT] [,FILE.LST] [/SWT] =FILE.FOR

где

FILE.OBT спецификация двоичного файла, который получается в процессе трансляции;
FILE.LST спецификация файла листинга программы;
/SWT набор ключей и аргументов, описаны ниже;
FILE.FOR спецификация файла с исходной программой.

Все спецификации выходных файлов не обязательны. Выходной файл не формируется, если командная строка не содержит спецификацию этого файла. Тип файла в спецификации выходного файла определяется по позиции его в командной строке, указанной

числом запятых в строке. Запятая после спецификации последнего выходного файла в командной строке не нужна.

Если вызван FORTAP, но еще не введена командная строка, то его работу можно завершить нажатием ⟨CTRL/C⟩.

После завершения набора командной строки трансляцию можно остановить, набрав ⟨CTRL/C⟩ дважды. Управление передается монитору, и на терминале появляется точка.

ЛИТЕРАТУРА

1. *Аксенов Г.А. и др.* Универсальный арифметический процессор АП-32. Архитектура, система команд, технические характеристики.—Препринт ИЯФ 89-175. Новосибирск, 1989.
2. PDP-11 FORTRAN Language Reference Manual. Order No.AA-1855D-TC. December 1979. DEC, Maynard Massachusetts.
3. *Аксенов Г.А. и др.* Универсальный арифметический процессор АП-32. 3. Канал управления и ввода/вывода. Драйвер операционной системы.—Препринт ИЯФ 90-3. Новосибирск, 1990.
4. *Аксенов Г.А. и др.* Программное обеспечение процессора АП-32. Комплекс программного обеспечения. Средства разработки программ. Операционная среда программы АП-32. —Препринт ИЯФ . Новосибирск, 1990.
5. PDP-11 MACRO-11 Language Reference Manual. Order No.AD-5075B-T. December 1981. DEC, Maynard Massachusetts.
6. *Грис Д.* Конструирование компиляторов для цифровых вычислительных машин. М.: Мир, 1975.
7. *Мерзляков Ю.И. и др.* Программное обеспечение процессора АП-32. 2. Ассемблер. Редактор связей.—Препринт ИЯФ 90-30. Новосибирск, 1990.
8. *Аксенов Г.А. и др.* Универсальный арифметический процессор АП-32. 4. Арифметика с плавающей запятой, сопроцессор элементарных функций.—Препринт ИЯФ 90-26. Новосибирск, 1990.

Приложение 1

КЛЮЧИ КОМАНДНОЙ СТРОКИ КОМПИЛЯТОРА FORTAP

Командные строки компилятора включают описанные ниже ключи с восьмеричными, десятичными или символическими аргументами в спецификациях входных и выходных файлов. Можно использовать следующие ключи:

/A—печатает статистику трансляции.

/D—трансляция строк с символом D в первой колонке (для целей отладки). Без этого ключа соответствующие строки воспринимаются как комментарий.

/E—позволяет использовать позиции 73—80 в исходных строках программы для текста оператора.

/L:N — определяет содержимое листинга; аргумент N кодируется следующим способом:

0 или NUL — распечатка диагностики;

1 или SRC — распечатка исходной программы и диагностики;

2 или MAP — распечатка карты памяти и диагностики;

4 или COD — распечатка генерируемого кода и диагностики.

Любая комбинация вышеприведенного списка параметров может быть определена как сумма числовых значений аргументов, например: 7 или ALL выдает исходный листинг, карту памяти и листинг генерируемого кода; если этот ключ опущен, то по умолчанию устанавливается /L:3 и выдается исходный листинг и карта памяти;

/M:xxx — подавляет указанный тип оптимизации программы; используются следующие аргументы xxx:

CSE — устранение общих подвыражений;

STR — оптимизацию циклов.

/N:n — определяет количество логических устройств, которые могут использоваться в программе одновременно; по умолчанию принимается число, введенное во время генерации компилятора;

/O — включает в листинг раздел «OPTION-IN-EFFECT» (используемые возможности компилятора);

/P:xxx — разрешает вид оптимизации по значению аргумента xxx:

CSE — устранение общих подвыражений;

STR — оптимизацию циклов.

/Q — запрещает печать имен программных модулей (операторы PROGRAM, FUNCTION, SUBROUTINE и BLOCK DATA) при трансляции каждого программного модуля.

/R:n — определяет максимальный размер записи (в байтах) во время форматного ввода/вывода ($4 < n < 4095$); по умолчанию принимается число, заданное во время генерации компилятора.

/S — подавляет включение кодов трассировки внутренних номеров операторов в транслируемую программу.

/V — подавляет векторизацию многомерных массивов.

/W — разрешает выдачу предупреждающей диагностики компилятора; используется в сочетании с ключом /L.

/Z — устанавливает для секций чистого кода и чистых данных атрибут «RO» («только для чтения»).

БИБЛИОТЕЧНЫЕ ФУНКЦИИ

ABS(X)	абсолютная величина реального числа X
IABS(I)	абсолютная величина целого числа I
FLOAT(I)	преобразование целого I в реальное
IFIX(X)	преобразование реального X в целое
AINT(X)	выделение целой части реального X
INT(X)	ближайшее к реальному целое число I
AMOD(X,Y)	реальный остаток X/Y
MOD(I,J)	целый остаток I/J
AMAX0(I,J,...)	реальный максимум списка целых чисел
AMAX1(X,Y,...)	реальный максимум списка реальных чисел
MAX0(I,J,...)	целый максимум списка целых чисел
MAX1(X,Y,...)	целый максимум списка реальных чисел
AMIN0(I,J,...)	реальный минимум списка целых чисел
AMIN1(X,Y,...)	реальный минимум списка реальных чисел
MIN0(I,J,...)	целый минимум списка целых чисел
MIN1(X,Y,...)	реальный минимум списка целых чисел
DIM(X,Y)	реальная положительная разность X — (MIN(X,Y))
IDIM(I,J)	целая положительная разность I — (MIN(I,J))
SIGN(X,Y)	перенос знака реального Y на реальное X
ISIGN(I,J)	перенос знака целого J на целое I
RAN(I,J)	генератор случайных чисел
RANDU(I,J,X)	генератор случайных чисел

Стандартные функции

SQRT(X)	квадратный корень
ALOG(X)	натуральный логарифм
ALOG10(X)	десятичный логарифм
EXP(X)	экспонента
SIN(X)	синус
COS(X)	косинус
TAN(X)	тангенс
ASIN(X)	арксинус
ACOS(X)	арккосинус
ATAN(X)	арктангенс
ATAN2(X,Y)	арктангенс X/Y
TANH(X)	гиперболический тангенс

Байтовые функции

BYTE	многоцелевая байтовая функция
IBYTE	многоцелевая байтовая функция
PUTBYT	укладка байта в массив
GETBYT	извлечение байта из массива
LEN	определение длины цепочки байтов
SCOMP	сравнение цепочек байтов
ISCOMP	сравнение цепочек байтов
SCOPY	копирование цепочек байтов
STRPAD	дополнение цепочек байтов пробелами

Прочие функции

R50ASC	преобразование кода RADIX-50 в ASCII
SETERR	установка режима обработки ошибок

В.Н. Козлов, Ю.И. Мерзляков,
И.А. Ткаченко, А.Г. Чертовских

Программное обеспечение процессора АП-32.

3. Реализация языка высокого уровня ФОРТРАН-IV

Ответственный за выпуск С.Г.Попов

Работа поступила 28 февраля 1990 г.
Подписано в печать 12.03 1990 г. МН 08433
Формат бумаги 60×90 1/16 Объем 1,8 печ.л., 1,5 уч.-изд.л.
Тираж 250 экз. Бесплатно. Заказ № 35

Набрано в автоматизированной системе на базе фотонаборного автомата ФА1000 и ЭВМ «Электроника» и отпечатано на ротапринте Института ядерной физики СО АН СССР,
Новосибирск, 630090, пр. академика Лаврентьева, 11.