ИНСТИТУТ ЯДЕРНОЙ ФИЗИКИ СО АН СССР

Ю.И. Мерзляков, И.А. Ткаченко

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРОЦЕССОРА АП-32.

2. Ассемблер. Редактор связей

ПРЕПРИНТ 90-30

НОВОСИБИРСК

Программное обеспечение процессора АП-32.

2. Ассемблер. Редактор связей

Ю.И. Мерзляков, И.А. Ткаченко

Институт ядерной физики
630090, Новосибирск 90, СССР

АННОТАЦИЯ

В препринте описаны ассемблер А32 и редактор связей LINKAP, входящие в комплект программного обеспечения АП-32. Рассмотрены отличия от аналогичных программ для других ЭВМ. Описан язык ассемблера АП-32, структура загрузочного модуля, распределение памяти программ и памяти данных АП-32. В Приложении приводится описание ключей ассемблера и редактора связей, список сообщений об ошибках.

4. А32 — АССЕМБЛЕР АП-32

4.1. Общее описание

Как и большинство ЭВМ, процессор АП-32 можно программировать на уровне машинных команд. Это необходимо для написания оптимизированного по времени исполнения и объему кодов матобеспечения, которое в дальнейшем не модифицируется, а также программ, с помощью которых нужно настраивать процессор или анализировать его состояние. Сюда относятся:

1) тесты аппаратуры процессора;
2) библиотечные математические функции;
3) системные подпрограммы.

Для машинно-ориентированного программирования АП-32 был разработан язык и написан ассемблер А32. Ассемблер обеспечивает следующие возможности:

1) управление функциями трансляции с помощью командных строк и программных директив;
2) создание объектного машинного кода;
3) использование глобальных символов для компоновки независимых объектных модулей;
4) использование директив секционирования программ;
5) возможность спецификации устройств и имен файлов для входных и выходных файлов трансляции;
6) вывод сообщений об ошибках трансляции на терминал;
7) вывод таблицы символов программы;
8) управление функциями распечатки с помощью программных директив.

4.1.1. Особенности А32

Ассемблер в целом сходен с аналогичными средствами других ЭВМ, но имеет и ряд особенностей.

1. При разработке ассемблера ставилась задача обеспечения гибкости в определении новых машинных команд и директив языка. Это вытекало из природы системы команд АП-32 (команды типа микрокоманд), допускающей расширение набора команд по мере необходимости. Эта задача была решена путем применения табличного метода лексического и синтаксического анализа исходного текста программы. Включение новой команды или директивы сводится к модификации макрокоманд генерации таблиц разбора и дописыванию, если нужно, семантических подпрограмм, что занимает непродолжительное время.

2. Для скорейшего освоения языка ассемблера АП-32 и удобства работы с ним реализована расширенная диагностика ошибок трансляции с указанием в листинге позиции синтаксической ошибки в операторе языка, а также вывод справочной информации на уровне командной строки.

3. Ассемблер генерирует два вида кода — перемещаемый объектный код для последующей сборки программы при помощи редактора связей и готовый образ задачи АП-32 для непосредственной загрузки и исполнения в том же виде, что создает редактор связей. Второй вид используется в основном для написания тестов АП-32. Выходной информацией является также листинг генерируемого кода и сообщения об ошибках, выводимые на терминал.

4.1.2. Конструкция ассемблера А32

Ассемблер работает по двухпроходной схеме, т. е. исходный текст программы считывается дважды.

В начале работы ассемблер примет командную строку, содержащую спецификации файлов, которые должны будут использоваться при трансляции. После синтаксической проверки командной строки ассемблер открывает входной файл и инициирует выходной файл, определенные в командной строке, а затем, начиная непосредственно сам процесс трансляции, ассемблер вводит исходные строки программы из входного файла.

Основной целью первого прохода трансляции является создание таблицы символов и таблицы программных секций; одновременно с этим выполняется частичная трансляция исходных операторов.

В конце первого прохода трансляции ассемблер выводит информацию, которая будет использована редактором связей при обработке объектных модулей. В объектный файл выводится такая информация, как имя объектного модуля и словарь глобальных символов каждой программной секции. Словарь глобальных символов содержит записи о глобальных точках входа каждой секции программного модуля и записи об используемых в секции глобальных символах.

На втором проходе трансляции ассемблер создает объектный выходной файл и, одновременно с этим, листинг, который содержит и таблицу символов программы.

В основном второй проход трансляции состоит из тех же шагов, что и первый, с той разницей, что при создании листинга трансляции строки исходной программы, содержащие ошибки, отмечаются флагами ошибок. Объектный файл, создаваемый в результате второго прохода ассемблера, наряду с объектным кодом, содержит информацию о перемещаемых символах и выражениях, необходимую редактору связей для обработки объектного файла. Информация, передаваемая редактору связей в объектном файле, позволяет ему связать глобальные символы с абсолютными адресами памяти и сформировать программу в формате загрузки.

Центральным элементом первого и второго проходов является процедура разбора оператора языка ассемблера. Принимая на входе текст оператора, эта процедура возвращает его специальное внутреннее представление, предназначенное для последующей обработки, а также диагностику ошибок.

Процедура разбора состоит из трех частей, написанных на языке ассемблера MACRO-11 [1]: таблицы разбора, подпрограммы анализа и набора семантических подпрограмм.

Таблица разбора представляет собой грамматику языка ассемблера АП-32 (набор правил построения операторов), записанную при помощи специальных макрокоманд MACRO-11. Подпрограмма анализа сканирует строку исходного текста по одному знаку (букве, цифре и т. д.), в соответствии с таблицей выделяет элементы оператора (лексемы) и определяет, принадлежит ли оператор языку (т. е. нет ли ошибки). Одновременно происходит преобразование текстовой информации в двоичную (мнемокоды операций — в коды команд АП-32, численные константы — в двоичные эквиваленты и т. д.) и при помощи семантических подпрограмм формируется полезная выходная информация (например, заполняются поля машинной команды АП-32). При выявлении синтакси-

ческой ошибки позиция ошибочного элемента оператора передается как выходной параметр. В качестве выходных параметров возвращается внутреннее представление — код оператора и коды метки и выражения в поле операнда.

Подпрограммы обоих проходов написаны на ФОРТРАНе. Они считывают текст программы по одной строке, вызывают процедуру разбора, а затем обрабатывают внутреннее представление в зависимости от кода оператора. Для машинных команд, например, вычисляются выражения в поле операнда и завершается формирование машинного слова АП-32. Из этих же подпрограмм вызываются функции работы с таблицей символов и записи сгенерированной информации в файлы кодов и листинга.

В таблице символов содержатся только метки, символы, определенные через прямое присваивание и ссылки, постоянных символов нет (они содержатся в таблице разбора). Емкость таблицы установлена на 1000 символов, при необходимости ее можно увеличить. Метод поиска в таблице линейный.

Выбранный метод реализации обеспечивает приемлемую скорость трансляции, хотя и не оптимален по этому параметру. Однако АП-32 рассчитан на решение вычислительных задач, которые пользователи пишут на ФОРТРАНе, и создание каких-либо программ большого объема на языке ассемблера (операционные системы, например) не предполагается.

К настоящему времени на языке ассемблера написано и оттранслировано с помощью A32 все библиотечное и системное матобеспечение АП-32, а также ряд тестов, которые невозможно написать на ФОРТРАНе АП-32. В целом это составляет около 15 000 ассемблерных операторов.

4.1.3. Вызов и прекращение работы A32

Для вызова следует воспользоваться командой R A32 или RUN DEV:A32 для запуска A32 с системного устройства или устройства DEV:, после чего ассемблер ожидает ввода строки команды.

Формат командной строки следующий:

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

где

FILE.OBT спецификация двоичного объектного файла, который получается в процессе трансляции;
FILE.LST спецификация файла листинга программы;
/SWT набор ключей и их аргументов;
FILE.ASM спецификация исходного файла.

Следующая командная строка вызывает трансляцию и получение объектного файла BINF.OBT и листинга. Используется один исходный файл, листинг выдается на устройство печати.

*DK:BINF.OBT,LP:=DK:SRC.ASM

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. Чтобы опустить объектный файл, надо начать командную строку с запятой. Запятая после спецификации последнего выходного файла в командной строке не нужна.

Если вызван A32, но еще не введена командная строка, то работу ассемблера 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.

4.2. Язык ассемблера A32

Программа состоит из последовательности символьных строк. Каждая строка содержит один оператор языка ассемблера и имеет длину не более, чем 80 знаков.

Оператор языка ассемблера может содержать до четырех полей. Назначение этих полей определяется порядком их появления в операторе и/или разграничительными знаками между ними. Формат оператора языка ассемблера имеет вид:

[метка:] операция операнды [;комментарий]

поле метки и поле комментария не обязательны.

Поле операции и поле операнда являются взаимозависимыми, т. е. если эти поля присутствуют в исходном операторе, то семантика полей взаимосвязана. Оператор может содержать поле операции и не содержать поля операнда, но не наоборот.

Ассемблер интерпретирует и обрабатывает операторы исходной программы последовательно один за другим, генерируя одну дво-

ичную инструкцию, слово или полуслово данных или выполняя указанные действия по трансляции и/или созданию листинга.

Пустые строки допустимы и не влияют на процесс трансляции, но распечатываются в листинге.

Оператор языка ассемблера должен занимать не более одной исходной строки и не может иметь строк продолжения.

Для размещения поля согласно приведенному формату, в качестве ограничителя поля может использоваться знак горизонтальной табуляции ⟨TAB⟩.

4.2.1. Поле метки

Метка представляет собой определяемое пользователем имя (символ), значение которого равно текущему значению счетчика адреса и заносится ассемблером в таблицу символов.

Счетчик адреса является средством, с помощью которого ассемблер определяет адреса памяти для операторов исходной программы по мере их трансляции. Значение адреса метки будет абсолютным или относительным в зависимости от того, является ли программная секция, в которой эта метка определяется, абсолютной или перемещаемой. В случае абсолютной программной секции значение счетчика адреса является абсолютным. Аналогично, значение счетчика адреса в перемещаемой программной секции является относительным. Чтобы установить фактический абсолютный адрес метки, редактор связей в этом случае вычисляет абсолютное смещение программной секции и прибавляет его к значению счетчика адреса.

Метка является средством символического описания ячейки внутри программы. Если метка имеется, она всегда стоит первой в операторе и должна быть ограничена двоеточием.

Метка может быть любой длины, однако значащими являются только первые шесть знаков. Поэтому они должны быть уникальными для всех меток в исходной программе. Все метки заканчиваются двоеточием (:), которое является ограничением, а не частью метки. Если первые шесть знаков в двух или более метках совпадают, то в листинге трансляции появляется флаг ошибки (M).

Символ, используемый как метка, не может быть переопределен внутри исходной программы. Многократное переопределение символа порождает в листинге трансляции флаг ошибки (M) (см. Приложение). Более того, любой оператор в исходной программе, который ссылается на многократно определенную метку, дает в листинге трансляции флаг ошибки (D).

4.2.2. Поле операции

Поле операции в исходном операторе следует за полем метки. Это поле может содержать мнемоническое изображение машинной команды или директиву ассемблера.

Поле операции не может предшествовать метке; ему могут предшествовать одна или несколько меток, а за ним могут следовать один или несколько операндов и/или комментарий. Пробелы и знаки табуляции в начале и конце поля операции не несут смыслового значения и служат только для отделения поля операции от предшествующего и следующего за ним полей.

Если оператор является мнемоническим изображением команды, то генерируется код машинной команды (мнемоники и коды описаны в [2]), и затем ассемблер вычисляет адреса операндов. Если оператор является директивой, то ассемблер выполняет соответствующие этой директиве действия. Ограничителями оператора могут быть пробел, знак табуляции, а также любой знак, не подлежащий кодировке RADIX-50.

4.2.3. Поле операнда

Если поле операции содержит мнемоническое изображение команды (код операции), то поле операнда определяет те параметры программы, над которыми оператор будет производить действия. Поле операнда может быть использовано также для передачи соответствующих аргументов директив ассемблера.

Операнды могут быть выражениями или символическими аргументами. Если в поле операндов указывается несколько символических аргументов, то они должны быть разделены любыми из установленных разделителей: запятой, знаком табуляции и/или пробелом. Операнду должно предшествовать поле операции; если поле операции опущено, то оператор интерпретируется ассемблером как неявная директива .WORD.

Если поле операции содержит код операции, то операнды такого оператора всегда являются выражениями. Например:

```
LD   R0, A+2(R1)
```

Если поле операции содержит директиву ассемблера, то соответствующие операнды являются символическими аргументами.

4.2.4. Поле комментария

Поле комментария всегда начинается знаком «точка с запятой» (;). Это поле необязательно и может содержать любые знаки

символьного кода за исключением знаков забой, возврат каретки, перевод строки, вертикальная табуляция, перевод формата. Все другие знаки, появляющиеся в поле комментария, даже специальные знаки, зарезервированные для использования в ассемблере, проверяются только на допустимость в символьном коде и включаются в листинг трансляции в том виде, в каком они встречаются в исходном тексте.

5. РЕДАКТОР СВЯЗЕЙ LINKAP

5.1. Общее описание

В результате работы языковых процессоров (ассемблеров и компиляторов языков высокого уровня) из исходного файла, содержащего текст программы, генерируются коды задачи, которые могут быть непосредственно загружены в машину (абсолютный код) или же требуют дополнительной обработки (перемещаемый код), связанной с вычислением физических адресов. В случае АП-32 возможна генерация как абсолютного, так и перемещаемого кода (ассемблер A32) или только перемещаемого (компилятор FORTAP). Абсолютные коды используются только для написания тестов аппаратуры.

В отличие от абсолютного, перемещаемый код предоставляет следующие возможности:

1) сборка рабочей программы из нескольких объектных модулей, созданных в разное время;
2) создание и использование библиотек подпрограмм;
3) настройка программы на любые нужные пользователю физические адреса памяти без перетранслирования программы.

Для создания из одного или нескольких объектных модулей загрузочного модуля служит программа, называемая редактором связей, которая написана для АП-32 на языке ассемблера MACRO-11. Редактор связей АП-32 LINKAP:

1) создает абсолютные программные секции и распределяет память для перемещаемых программных секций;
2) разрешает глобальные ссылки и присваивает абсолютные адреса перемещаемым глобальным символам;
3) просматривает указанные в командной строке библиотечные файлы и извлекает из них модули, необходимые для разрешения глобальных ссылок;
4) формирует контекстный блок задачи;

5) создает битовую карту занятости памяти (BITMAP);
6) создает карту распределения памяти.

В задачу создания средств построения программ из перемещаемых объектных модулей входила также разработка структур объектного кода, загрузочного файла и распределения памяти программы АП-32. Все эти способы организации данных для АП-32 уникальны.

При создании LINKAP не планировалось построение оверлейных задач для АП-32, и никаких средств для этого нет.

5.1.1. Конструкция редактора связей

Редактор связей LINKAP работает по трехпроходной схеме, т. е. исходные файлы считываются программой трижды.

В первом проходе из указанных входных файлов извлекается информация о глобальных символах, определяемых в модуле, и глобальных ссылках и помещается в таблицу символов. Аналогично все имена программных секций заносятся в соответствующую таблицу. При обработке глобальных символов делаются попытки замкнуть ссылки на определенные глобальные символы, устанавливая тем самым связь между модулями. Если после обработки всех указанных в командной строке модулей остались неразрешенные глобальные ссылки, редактор связей просматривает каталог системной библиотеки APLIB для разрешения этих ссылок. Из библиотек извлекаются только те модули, которые требуются программе. Имея в конце первого прохода полную таблицу символов, редактор связей вычисляет размеры и базовые адреса программных секций, присваивает абсолютные значения перемещаемым глобальным символам и создает абсолютные программные секции .IABS. для памяти программ и .DABS. для памяти данных.

Второй проход редактора связей посвящен созданию битовой карты занятости памяти (BITMAP). В это время просматривается объектный код и помечаются занятые области памяти, по размеру соответствующие дисковым блокам (512 байтов), после чего распределяется дисковая память загрузочного файла. Этот проход определяет особенность LINKAP относительно аналогичных редакторов связей, как правило, двухпроходных.

Во время третьего прохода редактор связей заполняет контекстный блок задачи, записывает в загрузочный файл карту занятости и генерирует абсолютный код, пользуясь накопленной в таблицах информацией. Код также копируется в загрузочный файл.

На последнем этапе создается карта распределения памяти, выводимая в соответствующий файл.

Таблицы данных редактора связей организованы в виде связных списков, когда элементы таблиц связаны ссылками. Таблица программных секций, кроме того, связана с таблицей символов. При этом элемент таблицы секций имеет цепочку ссылок на все символы, определенные в описываемой им программной секции. Этот прием позволяет создавать несколько списков переменной длины в линейном буфере LINKAP. Поиск в таблицах секций и символов ускорен при помощи хеширования.

5.2. Структура объектного модуля

Объектный код АП-32 по назначению и структуре сходен с объектным кодом ЭВМ семейства Электроника/СМ и ЕС ЭВМ. Он содержит информацию следующих типов:

1) словарь глобальных символов для установления связи между разными объектными модулями;
2) частично транслированные коды с незаполненными адресными частями команд и адресными константами данных;
3) описатели перемещений, при помощи которых эти адресные части настраиваются на абсолютные величины.

Словарь глобальных символов содержит:

1) имя объектного модуля;
2) имена внутренних (неглобальных) символов для символических отладчиков программы (необязательный элемент);
3) стартовый адрес программы;
4) глобальные определения или ссылки на символы;
5) имена программных секций и их длины;
6) код идентификации модуля.

Частично транслированные коды (текстовые блоки) — поток команд и данных, которые представляют основу для построения задачи.

Описатели перемещений указывают настраиваемые слова в текстовых блоках и способы этой настройки. Возможны следующие типы перемещений:

1) внутреннее перемещение — сложение величины адресной части с базой текущей программной секции;
2) глобальное аддитивное перемещение — помещение в адресную часть величины глобального символа и прибавление константы, если она есть;

3) определение или модификация текущего адреса программной секции;
4) перемещение программной секции — вычисление адресной части со ссылкой в другую программную секцию;
5) вычисление параметров программы — верхних и нижних пределов памяти.

Объектный код такого вида содержит полную информацию для построения задачи АП-32.

5.3. Структура загрузочного модуля

Загрузочный модуль, создаваемый редактором связей, содержит всю информацию, необходимую для загрузки задачи в память АП-32, ее запуска и состоит из карты занятости и кодов задачи.

Использование карты занятости позволяет максимально компактно расположить информацию в файле, содержащем загрузочный модуль. При этом дисковый образ задачи АП-32 занимает, как правило, меньшее пространство, чем сама задача в памяти АП-32, так как в него не входят объявленные неинициированные массивы, просто резервирующие память. Для мини- и микроЭВМ, к которым подключается АП-32, требования к объему диска становятся мягче, что важно при объеме памяти АП-32 2 Мбайта.

Карта занятости памяти размещена в нулевом блоке файла. Половина блока (адреса 0—376) отображает загруженность памяти данных, вторая половина (адреса 400—776) — загруженность памяти программ. Каждый бит карты соответствует блоку (512 байтов) памяти, подлежащему загрузке из файла.

Во время загрузки программы в память сканируется каждое слово карты занятости и происходит вычисление физического адреса загрузки в память АП-32 для каждого дискового блока.

Бит 0 слова с адресом 0 относится к блоку памяти данных с адресами 0—376, бит 1 того же слова — к блоку памяти данных 400—776 и т. д. Использование карты занятости становится понятным из следующего примера.

Пусть карта имеет вид:

```
000/ 000105
. . . . . .
376/ 000000
400/ 000022
. . . . . .
776/ 000000
```

В соответствии с ней загрузка памяти АП-32 произойдет следующим образом:

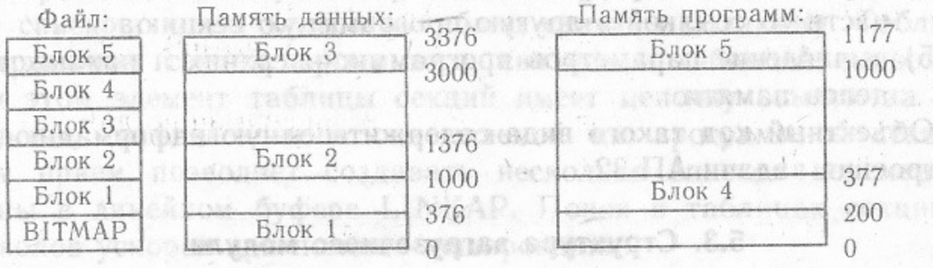

Рис. 2.

В слове с адресом 0 установлены биты 0, 2, 6, что соответствует блокам памяти данных с адресами (0—376), (1000—1376) и (3000—3376).

В части карты, «отвечающей» за загруженность памяти программ, установлены биты 1 и 4. Соответствующие блоки загружаются в память программ по адресам (200—377) и (1000—1177).

Использование карты занятости позволяет создавать компактные загрузочные модули, а, кроме того, позволяет загружать в память АП-32 несколько не перекрывающих друг друга программ, что является необходимым условием для реализации многозадачного режима в АП-32.

5.4. Распределение памяти для задачи АП-32

Как отмечалось выше, редактор связей создает абсолютные программные секции памяти программ и памяти данных. Они носят имена .IABS. и .DABS., соответственно, и содержат служебную информацию.

Абсолютная программная секция памяти данных .DABS. всегда размещается в блоке 1-го файла, содержащего загрузочный модуль. Физический номер блока для абсолютной программной секции памяти программ .IABS. не фиксирован и вычисляется в соответствии с картой занятости.

5.4.1. Распределение памяти данных

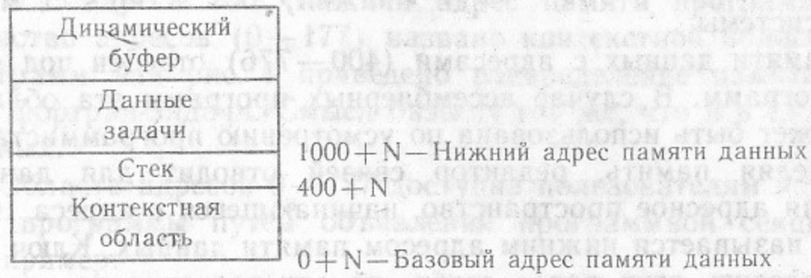

Рис. 3.

Служебная информация, содержащаяся в программной секции .DABS. и загружаемая в пространство адресов памяти данных (0—376), называется контекстной областью задачи. Распределение данных в контекстной области фиксировано, ниже приводится ее описание.

Адрес	Содержимое
0	Зарезервировано
2	Стартовый адрес задачи
4	Нижний адрес памяти данных
6	Верхний адрес памяти данных
10	Нижний адрес памяти программ
12	Верхний адрес памяти программ
14	Размер динамического буфера
16	Адрес начала стека задачи
20	Имя задачи в коде RADIX-50
22	Зарезервировано
24	Адрес системных данных фортран-программы
26	Длина секции кодов $CODE фортран-программы
30	Код пользовательской ошибки
32—376	Зарезервировано

Область адресов 0—376 доступна пользователям из ассемблерной программы путем объявления программной секции .DABS.. Например:

```
        .PSECT  .DABS.  ABS,D
.=0
        .WORD   12345
        .HWORD  111
```

Однако пользователям не рекомендуется размещать данные в

пространстве адресов 0—376, так как это может вызвать порчу системных данных, а в случае фортран-программ — ошибку исполнительной системы.

Блок памяти данных с адресами (400—776) отведен под стек фортран-программ. В случае ассемблерных программ эта область адресов может быть использована по усмотрению программиста.

Распределяя память, редактор связей отводит для данных пользователя адресное пространство, начинающееся с адреса 1000. Этот адрес называется нижним адресом памяти данных. Ключ /D: может определить этот адрес особо, но его значение не должно быть меньше 400 (восьмеричное).

Выше данных задачи в адресном пространстве памяти данных отводится динамический буфер, предназначенный для возможного расширения области данных задачи. В фортран-программах, например, динамический буфер используется для буферизации данных при операциях ввода/вывода. Размер этого буфера задается при сборке, размер по умолчанию — 400_8 полуслов.

Редактор связей позволяет перемещать задачу пользователя как единое целое в памяти программ и памяти данных путем задания физической базы памяти программ и памяти данных. Если значение базы N не задано, N=0 (рис. 3)

5.4.2. Распределение памяти программ

Абсолютная секция памяти программ .IABS. располагается в пространстве адресов памяти программ 0—177 и содержит часть системных кодов, служащих для коммуникации между АП-32 и ЭВМН.

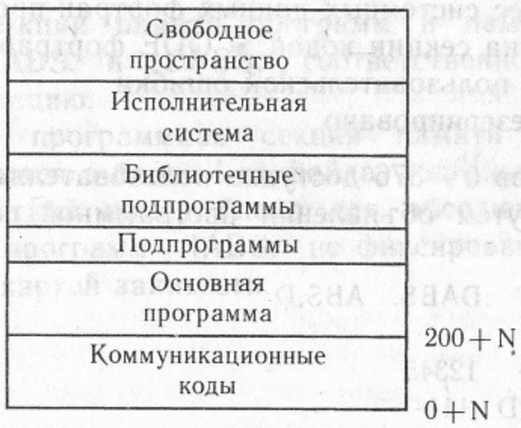

Рис. 4.

Коды программы пользователя размещаются в памяти программ с адреса 200 (нижний адрес памяти программ), а пространство адресов (0—177) названо контекстной областью памяти программ. На рис. 4 приведено распределение памяти программ для фортран-задачи. Смысл базы N тот же, что и в случае памяти данных.

Область адресов 0—177 доступна пользователям из ассемблерной программы путем объявления программной секции .IABS.. Например:

```
        .PSECT  .IABS.  ABS,I
.=0
        LXI     R1,2
        MOV     R0,R2
        ADD     R1,R2
```

Однако использовать адресное пространство памяти программ (0—177) не рекомендуется, так как это может привести к нарушению нормальной работы исполнительной системы ФОРТРАН.

5.5. Карта распределения памяти

Карта распределения памяти, создаваемая редактором связей LINKAP, содержит следующую информацию, которая может быть полезна пользователю при работе с программой:

1. Номер версии редактора связей, создавшего карту памяти.
2. Дата и время создания карты памяти.
3. Имена программных секций памяти программ и памяти данных.
4. Базовые адреса программных секций.
5. Размеры программных секций.
6. Атрибуты программных секций.
7. Список имен глобальных символов для каждой программной секции.
8. Значения абсолютных глобальных символов.
9. Адреса перемещаемых глобальных символов.
10. Стартовый адрес задачи.
11. Базовый адрес памяти данных..
12. Базовый адрес памяти программ.
13. Размер стека (для фортран-программ).
14. Размер буфера (для фортран-программ).
15. Размер в блоках файла, содержащего загрузочный модуль.

16. Размер в блоках сегмента, содержащего данные, которые будут загружены в память данных.
17. Размер в блоках сегмента, содержащего данные, которые будут загружены в память программ.
18. Размер области памяти данных, занятой данными задачи.
19. Размер области памяти программ, занятой кодами задачи.
20. Список неопределенных глобальных символов.

5.6. Вызов и прекращение работы редактора связей

Для вызова программы LINKAP необходимо дать команду:

.R LINKAP или RUN DEV:LINKAP

при этом после загрузки редактор связей печатает «*» и ожидает ввод командной строки.

Формат командной строки следующий:

[FIL.JOB] [,FIL.MAP] [,FIL.STB] =FIL1.OBT,FIL2.OBT[/SWT]

Выходными файлами редактора связей являются (в порядке указания в командной строке) файлы типов:

1) .JOB — загрузочный модуль задачи;
2) .MAP — карта распределения памяти;
3) .STB — двоичная таблица символов.

Входные файлы (тип .OBT) могут быть объектными модулями и библиотеками объектных модулей.

В каждой спецификации файла устройство должно быть устройством с произвольным доступом (диск), за исключением файла карты памяти. Файл карты памяти может быть выведен на любое внешнее устройство системы. Если устройство не указано, редактор связей использует по умолчанию устройство DK:.

Следующая командная строка вызывает создание загрузочного модуля MODUL.JOB на устройстве XD1: и карты распределения памяти. Используется два исходных файла, карта распределения памяти выдается на терминал:

*XD1:MODUL.JOB,TT:=DK:FIL1.OBT,DK:FIL2.OBT

Все спецификации выходных файлов не обязательны. Система не формирует выходной файл, если командная строка не содержит спецификацию этого файла.

Система определяет тип файла в спецификации выходного файла по позиции его в командной строке, указанной числом

запятых в строке. Кроме того, можно не указывать тип входных файлов, если этот тип OBT. Чтобы опустить файл, содержащий загрузочный модуль, надо начать командную строку с запятой.

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

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

ЛИТЕРАТУРА

1. PDP-11 MACRO-11 Language Reference Manual. Order No. AA-5075B-TC. December 1981. DEC, Maynard, Massachusetts.
2. *Аксенов Г.А. и др.* Универсальный арифметический процессор АП-32. Архитектура, система команд, технические характеристики.—Препринт ИЯФ 89-175, 1989 г.

Приложение 1

КЛЮЧИ АССЕМБЛЕРА A32

Ключи ассемблера влияют на вид генерируемого двоичного кода. Используются ключи:

1. /H—на экран терминала выводится список ключей и ошибок трансляции с кратким описанием.
2. /J—объектный файл генерируется в виде образа задачи .JOB с абсолютными адресами. Код готов к загрузке в АП-32.
3. /R—объектный код генерируется в виде файла двоичных записей с абсолютными адресами. Предназначен для служебного пользования.
4. /I—в объектном коде присутствует директория внутренних символов (используется символическим отладчиком).
5. Если ключей нет, генерируется нормальный объектный код, являющийся исходной информацией для редактора связей.

Приложение 2

СООБЩЕНИЯ ОБ ОШИБКАХ A32

Когда в тексте программы появляется ошибка, строка с ошибкой помечается в листинге трансляции знаком «*» в первой позиции. Ассемблер печатает коды диагностики ошибок в качестве вто-

рого знака той исходной строки, на которой он обнаружил ошибку. Код ошибки указывает на тип ошибки; например, код «M» указывает на многократное описание метки. В листинге трансляции могут появиться следующие ошибки:

1. A — ошибка адресации или перемещения; она появляется, когда операнды команды имеют неверный адрес.
2. B — ошибка границы; текущее содержимое счетчика адреса вызывает трансляцию слов данных с нечетным адресом памяти, чтобы исправить это, система увеличивает счетчик адреса на 1.
3. D — ссылка на символ с многократным описанием; программа ссылается на метку, которая определяется более, чем один раз.
4. E — нет директивы «.END»; ассемблер достиг конца исходного файла и не нашел директивы «.END»; система генерирует «.END» и продолжает работу.
5. L — строка исходного текста содержит более 80 знаков.
6. M — многократное описание метки; встреченная метка эквивалентна (по первым шести знакам) ранее встречавшейся метке.
7. O — ошибка кода операции; неправильная директива. В следующей строке листинга выводится знак «*» непосредственно перед элементом оператора, вызвавшим ошибку.
8. P — ошибка фазы; описание или значение метки меняется от одного прохода к другому.
9. Q — оператор не соответствует объявленному типу памяти АП-32.
10. T — ошибка усечения; используется число, занимающее более 19 разрядов операнда.
11. U — неопределенный символ; при вычислении выражения встретился неопределенный символ; ассемблер присваивает неопределенному символу значение нуля.

Приложение 3

СИНТАКСИС РЕЖИМОВ АДРЕСАЦИИ

В табл. 1 приняты следующие обозначения:

1) R — целое число от 0 до 31, задающее номер регистра;
2) E — выражение (один или два операнда и знак операции).

Таблица 1

Формат	Наименование режима адресации	Значение
R	Прямая адресация	Регистр «R» содержит операнд.
(R)	Косвенная адресация	Регистр «R» содержит адрес операнда.
E(R)	Индексная адресация	«E» плюс содержимое регистра «R» дают адрес операнда, «E» находится в слове команды.
E	Непосредственная адресация	«E» — операнд, находится в слове команды.
E	Прямая адресация	«E» — адрес операнда находится в слове команды.

Приложение 4

СПЕЦИАЛЬНЫЕ ЗНАКИ ЯЗЫКА

Таблица 2

Знак	Функция
Вертикальная табуляция	Ограничитель исходной строки.
:	Ограничитель метки.
=	Указатель прямого присваивания.
%	Признак терма регистра-аккумулятора.
Горизонтальная табуляция	Ограничитель отдельного элемента или ограничитель поля.
Пробел	Ограничитель отдельного элемента или ограничитель поля.
#	Признак терма регистра плав. запятой.
(	Начальный указатель регистра.
)	Конечный указатель регистра.
, (запятая)	Разделитель полей операндов.
;	Признак комментариев.
+	Знак арифметической операции сложения.
-	Знак арифметической операции вычитания.
' (апостроф)	Признак для размещения одного знака в символьном коде.
.	Счетчик адресов программы.

Приложение 5

ДИРЕКТИВЫ АССЕМБЛЕРА

Краткое описание директив ассемблера:

.BLKH EXPR — резервирует блок памяти, длина блока в полусловах равна значению выражения.

.BLKW EXPR — резервирует блок памяти, длина блока в словах равна значению выражения.

.BYTE ARG1,ARG2... — вычисляет и записывает в последовательно расположенные байты полуслов значения указанных аргументов. Аргумент — численная константа или апостроф, за которым следует один знак; генерирует полуслово, содержащее 7-разрядное представление знака и нуль в старшем байте этого полуслова. 1 или 2 аргумента генерируют полуслово, 3 или 4 аргумента — слово, больше 4 аргументов не бывает.

.END [EXPR] — указывает конец исходной программы; аргумент, если он задан, определяет адрес передачи управления при запуске программы.

.EVEN — обеспечивает четность счетчика ячеек программы увеличением его на 1 в случае его нечетности.

.FLT ARG — генерирует словное представление чисел с плавающей запятой.

.GLOBL NAME — объявление глобальных символов «NAME».

.LIMIT — резервирует четыре слова, в которые редактор связей записывает нижние и верхние адреса обоих типов памяти загрузочного модуля программы.

.LIST [ARG] — директива .LIST увеличивает содержимое счетчика-флага распечатки на 1; формирует листинг программы ниже данной директивы.

.NLIST — уменьшает счетчик-флаг распечатки на 1; не создается часть листинга ниже этой директивы.

.ODD — обеспечивает нечетность значения счетчика адресов программы добавлением 1 в случае его четности.

.PSECT [NAME],ARG1,ARG2,... — начинает или продолжает именованную или неименованную программную секцию, имеющую указанные аргументами характеристики.

.RAD50 /STRING/ — генерирует блок данных, содержащий эквивалент в коде RADIX-50 знаковой строки, заключенной между ограничителями. «STRING» может иметь длину до 6 знаков; 1—3 знака генерируют полуслово, 4—6 знаков генерируют слово памяти данных.

.TITLE STRING — присваивает объектному модулю имя — первые шесть знаков указанной строки; заданная строка распечатывается на каждой странице листинга.

.WORD EXPR — генерирует слово, содержащее значение указанного выражения.

Приложение 6

КЛЮЧИ РЕДАКТОРА СВЯЗЕЙ LINKAP

Ниже приводится краткое описание ключей, которые могут быть использованы в командной строке редактора связей.

/I:NUM — ключ позволяет установить нижний адрес памяти программы. Аргумент «NUM» — восьмеричное число без знака, указывающее нижний адрес памяти программ:

```
*OUTPUT,LP:=INPUT/I:500
```

Если ключ не указан, то нижний адрес равен 200.

/D:NUM — ключ позволяет установить нижний адрес памяти данных. Аргумент «NUM» — восьмеричное число без знака, указывающее нижний адрес памяти данных:

```
*OUTPUT,LP:=INPUT/D:500
```

Если ключ не указан, то нижний адрес равен 1000.

/C или // — ключи позволяют продолжить команду на нескольких строках, например:

```
*OUTPUT,LP:=INPUT1,INPUT2/C
*INPUT3,INPUT4/C
*INPUT5
```

Знак «//» печатается в конце первой строки, компоновщик продолжает принимать дополнительные строки до тех пор, пока не встретится другой ключ //. Например:

```
*LINK,LINK=LINK0/I:700//
*LNK1
*LNKGSD
*LNKHDR
*LNKMAP
*LNKSAV
*LNKEM
*//
```

/S:NUM — ключ позволяет задавать размер стека программы. по умолчанию размер стека равен 400.

/U:NUM — ключ позволяет задавать размер динамического буфера исполнительной системы программы. По умолчанию размер буфера равен 400.

/Z:NUM — ключ позволяет задавать адресную базу памяти данных, т. е. перемещать данные вместе с контекстным блоком задачи по памяти данных.

/R — ключ позволяет переместить программную секцию в любое место памяти, если только при этом не происходит наложение данной программной секции на какую-нибудь другую:

```
*LIN=LIN/R
*SECTION NAME? __DATA
*BASE ADDRESS? 10000
```

/T:[NUM] — ключ позволяет задать значение стартового адреса:

```
*FIL=FIL/T:200
```

или имя глобального символа, адрес которого будет стартовым адресом программы:

```
*FIL=FIL/T
*TRANSFER ADDRESS? $GLOB
```

Приложение 7

СООБЩЕНИЯ ОБ ОШИБКАХ LINKAP

Редактор связей во время работы сообщает пользователю об ошибках, которые делятся на два класса: фатальные, и предупреждения. Тексту сообщения о фатальной ошибке предшествует надпись «?LINKAP-F-». После сообщения о фатальной ошибке редактор связей прекращает работу и переходит в режим ожидания ввода командной строки.

Тексту сообщения о предупреждении предшествует надпись «?LINKAP-W-», после сообщения редактор связей продолжает работу.

Тексты сообщений выводятся по-русски и не требуют расшифровки.

Ю.И. Мерзляков, И.А. Ткаченко

Программное обеспечение процессора АП-32.

2. Ассемблер. Редактор связей

Ответственный за выпуск С.Г.Попов

Работа поступила 28 февраля 1990 г.
Подписано в печать 5.03 1990 г. МН 02149
Формат бумаги 60×90 1/16 Объем 2,0 печ.л., 1,6 уч.-изд.л.
Тираж 250 экз. Бесплатно. Заказ № 30

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.