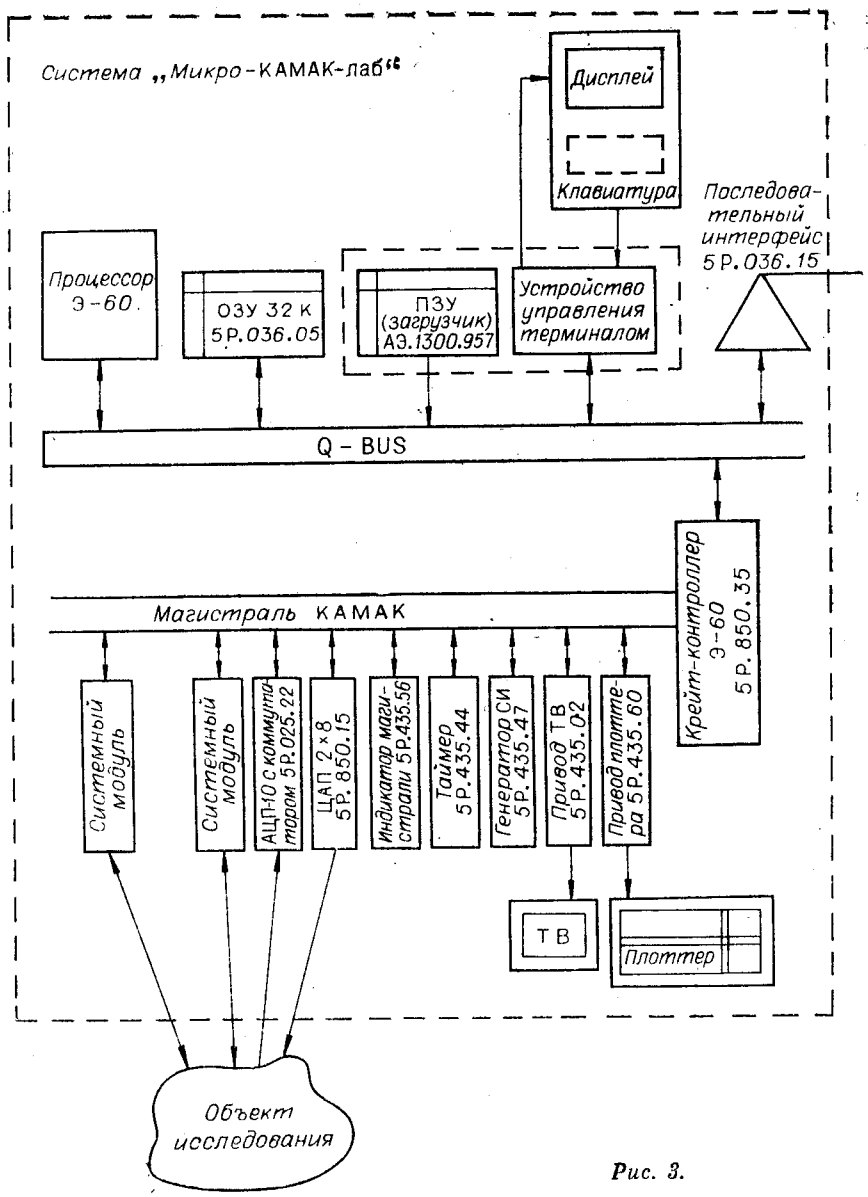


Рис. 3.

они приобретают необходимые навыки работы с автоматизированными системами, что упрощает процесс передачи и внедрения этих систем.

Поступила в редакцию 5 марта 1984 г.

УДК 683.3.06

Н. Г. ЩЕРБАКОВА

(Новосибирск)

КРОССОВОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ МИКРОПРОЦЕССОРНЫХ СИСТЕМ

Применение инструментальной вычислительной машины, на которую возлагаются функции подготовки, хранения, редактирования и трансляции программ для микропроцессорных систем, позволяет существен-

но сократить сроки разработки программного обеспечения. В настоящей текста программы, написанной на соответствующем языке, в объектный код. Процедура ассемблирования включает лексический и синтаксический анализ, генерацию кодов. При наличии в языке макроопределений необходим предварительный этап, на котором исходный текст программы преобразуется в развернутое представление.

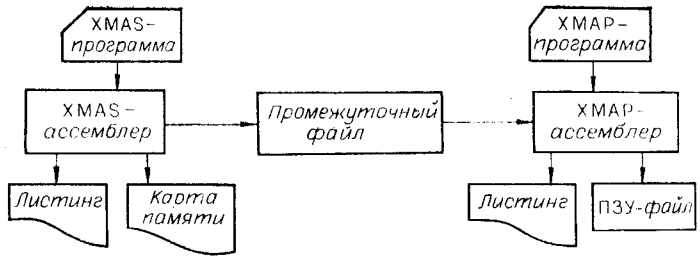
Кросс-ассемблер СА6800 осуществляет подготовку абсолютного объектного кода программы, написанной на языке Ассемблер [1] для микропроцессора МС6800. Предполагается, что файл с текстом исходной программы размещен в файловой системе SINTRAN-III (далее в аналогичных случаях этот факт не оговаривается). В результате двух просмотров исходного текста программы кросс-ассемблер генерирует файл объектного кода (который с помощью монитора-загрузчика может быть помещен в память микропроцессорного устройства), представляющий собой символьный файл специального формата. Кроме того, имеется возможность получения листинга программы, т. е. символьного файла, каждая строка которого содержит строку исходного текста программы и соответствующий ей сгенерированный код. Сообщения об ошибках, обнаруженных в процессе трансляции, также помещаются в листинг (непосредственно за строкой, в которой они обнаружены).

Кросс-ассемблер СА6800 реализован на языке «Nord PL», с одной стороны, обладающем возможностями языков высокого уровня, с другой — являющемся машинно-ориентированным, что позволило обеспечить высокую эффективность кросс-ассемблера (в частности, значительно снизить время трансляции). В качестве метода ассемблирования используется модифицированный метод диаграмм переходов [2], основывающийся на графическом представлении синтаксиса языка. Следует заметить, что непосредственное применение метода диаграмм затруднено из-за особенностей синтаксиса языков ассемблеров, для которых часто один и тот же набор лексем имеет разное синтаксическое значение в зависимости от местоположения в строке.

Кросс-ассемблер СА8748 предназначен для получения абсолютного кода программы, написанной на языке Ассемблер для микропроцессора серии MCS-48 [3]. По услугам, предоставляемым пользователям, этот кросс-ассемблер аналогичен кросс-ассемблеру СА6800. В результате работы за два просмотра программы ассемблер генерирует объектный файл в стандартном формате [4], который используется универсальным программатором ППЗУ [5].

В качестве языка реализации выбран Паскаль. Выбор языка в значительной степени обусловлен простотой перенесения программы на различные инструментальные машины, соблюдением в языке принципов структурного программирования и возможностью создавать динамические структуры. Последняя применяется при организации таблицы символов пользователя, имеющей списковую структуру. Скорость ассемблирования может быть увеличена за счет введения специальных методов записи и поиска в таблице, не влияющих на архитектуру программы.

В отличие от описанных выше инструментальных средств кросс-ассемблер СА3000 представляет собой систему из двух трансляторов, реализованную на инструментальной ЭВМ и предназначенную для программирования устройств, выполненных на базе микросхем серии К589. Основной функцией СА3000 является перевод микропрограммы, написанной на языке XMAS [6], в формат, пригодный для программирования микросхем ПЗУ, используемых в качестве памяти микропрограмм. Эта задача подразделяется на два этапа: трансляция программы с языка



XMAS; генерация файла для программирования ПЗУ. Такое функциональное деление обусловлено тем, что во время разработки микропрограммы может потребоваться неоднократное ассемблирование, прежде чем программа будет помещена в устройство постоянной памяти. Процесс создания ПЗУ-файла управляется с помощью специального языка XMAP, предложения которого включаются в исходный текст микропрограммы. Итак, система CA3000 состоит из двух кросс-ассемблеров: ассемблера с языка XMAS и ассемблера с языка XMAP, сообщающихся между собой с помощью промежуточного файла, формат которого зависит от реализации (см. рисунок).

Уникальной особенностью языка XMAS является его расширяемость. Формат микроинструкции строго не зафиксирован. Жестко определены только CPU-функции, JMP-функции и некоторые управляющие функции. Язык XMAS обеспечивает возможность расширения микроинструкции так называемыми полями пользователя, позволяя ему вводить для этих полей собственную мнемонику. Поэтому предложения языка включают как предложения-декларации, так и предложения-спецификации. В декларационной части пользователь может задать макроопределения.

При трансляции XMAS-программы сначала разбирается декларационная часть, на основе которой строятся необходимые внутренние таблицы, в том числе таблица макрорасширений, а затем в случае отсутствия ошибок проводится разбор предложений-спецификаций. В результате трансляции XMAS-программы формируется файл-листинг, где для каждого предложения-спецификации указываются значения всех полей, определяющих микроинструкцию. При наличии ошибок сообщения о них помещаются в листинг. Пользователь может запросить формирование файла, содержащего графическое представление содержимого микропрограммной памяти и список перекрестных ссылок для адресных идентификаторов («графическая карта памяти»). Кроме того, на этом этапе формируется промежуточный файл, необходимый для работы следующего этапа.

На втором этапе проводится разбор предложений языка XMAP, в результате чего формируется символьный файл в стандартном формате [4], который используется универсальным программатором ППЗУ [5]. Графически работа системы CA3000 представлена на рисунке.

На инструментальной машине реализованы два *инверсных ассемблера*. Инверсный ассемблер DA6800 по объектному коду восстанавливает «первоначальный» текст программы для микропроцессора MC6800, инверсный ассемблер DA8080 выполняет аналогичные функции для микропроцессора КР580. Поскольку в основу реализации положен один и тот же алгоритм, будем его называть алгоритмом инверсного ассемблера.

Процесс инверсного ассемблирования распадается на три этапа: отделение инструкций от данных; определение формата данных и расстановка меток; получение исходного текста.

Следует заметить, что основная проблема инверсного ассемблирования — отделение инструкций от данных — в общем случае алгоритмически неразрешима [7]. Кроме того, даже в случае успешного отделения инструкций существует неоднозначность их расшифровки (например, была использована инструкция LDX # ARG (при ARG = 200) или

LDX # 200). Следует отметить также, что некоторые данные могут вообще не получить признаков формата или иметь конфликтные признаки.

Для отделения инструкций от данных применяется алгоритм, который гарантирует полное решение задачи в предположении, что исходная программа не содержит кодов, модифицирующих инструкции; адреса переходов указываются явно; условное ветвление не используется там, где на самом деле осуществляется безусловный переход; возврат из подпрограммы происходит в точку, следующую за вызовом.

Алгоритм отделения инструкций от данных включает следующие этапы:

1. Поместить в стек все точки входа в программу.
2. Если стек пуст, алгоритм завершен.
3. Извлечь адрес $A1$ из стека.
4. Проверить, зарегистрирован ли адрес как инструкция. Если да, то возврат к 2.
5. Зарегистрировать $A1$ как инструкцию.
6. Декодировать инструкцию.
7. Если это инструкция ветвления, то поместить адрес перехода в стек; если это инструкция останова или безусловного перехода, то переход к 2.
8. Прибавить к $A1$ длину инструкции и возвратиться к 4.

Исходные данные для инверсного ассемблера — объектные коды в формате [4] — могут быть представлены в виде нескольких файлов. Эти файлы собираются в промежуточный файл, являющийся образом оперативной памяти рабочей ЭВМ. Он представляет собой упорядоченную по возрастанию адресов последовательность 16-разрядных слов общей емкостью 64 К. При этом один байт слова соответствует байту оперативной памяти рабочей ЭВМ с тем же адресом, а второй байт используется для размещения служебной информации.

Детрансляция выполняется за два прохода. Во время первого прохода проводится разбор промежуточного файла и расстановка в нем необходимых служебных признаков. Во время второго прохода происходит генерация файла с расшифрованным текстом программы на языке Ассемблер или (по желанию пользователя) с листингом этой программы. Промежуточный файл сохраняется после выхода из программы-дисассемблера, что, во-первых, позволяет исполнять второй проход отдельно от первого; во-вторых, избавляет от необходимости после выхода повторно загружать исходные файлы. Это может оказаться важным, так как исходных файлов в общем случае несколько, а получение удовлетворительного результата обычно связано с несколькими запусками инверсного ассемблера.

Инверсный ассемблер — это диалоговая программа, позволяющая программисту в интерактивном режиме управлять процессом детрансляции. Например, в процессе работы программист может ввести собственную таблицу адресных идентификаторов или новые точки входа в программу.

Необходимо отметить, что описанное в данной статье инструментальное программное обеспечение является частью распределенной системы, предназначенной для разработки аппаратных и программных средств микропроцессорных устройств [8, 9].

Пользуясь случаем, хочу поблагодарить П. М. Песляка за постоянное обсуждение работ, а также В. А. Мелешихина и Е. Г. Юрашанского за ценные консультации.

ЛИТЕРАТУРА

1. **Macro** Assembler Reference Manual 6800, 6801, 6805, 6809.— Motorola, Sept. 1979, M68MASR (D2).
2. **Конвей Е. М.** Проект делимого компилятора, основанного на диаграммах перехода.— В кн.: Современное программирование, языки для экономических расчетов. М.: Сов. радио, 1967.

3. **MCS-48. The Singlechip Family of Microcomputer Product Discription.**— INTEL, Jan. 1978.
4. **INTEL** Data Catalog, 1977.
5. **Universal** Prom Mapper Operators Manual.— INTEL, 1976.
6. **Series** 3000 Microprogramming Manual.— INTEL, 1976.
7. **Horspool R. N., Marovac N.** An approach to the problem of detranslation of computer programs.— Comput. J., 1980, vol. 23, N 3.
8. **Бобко В. Д. и др.** Распределенная система для разработки и отладки аппаратных и программных средств микропроцессорных устройств.— In: Proc. of the Symp. on Microcomputer and Microprocessor Application. Budapest, 1979, vol. 1, p. 311—315.
9. **Песляк П. М., Щербакова Н. Г.** Программное обеспечение распределенной системы микропроцессорных устройств.— В кн.: Тез. докл. VI Всесоюз. конф. по автоматизации научных исследований на основе применения ЭВМ. Новосибирск: ИАиЭ СО АН СССР, 1980.

Поступила в редакцию 27 декабря 1983 г.