# ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

**Введение**. Микропроцессорная система (или микро-ЭВМ), состоящая из микропроцессора, источников питания, модулей памяти, интерфейса и устройств ввода-вывода, представляет собой аппаратуру для обработки информации или как принято говорить в вычислительной технике, аппаратную часть системы обработки. Но чтобы осуществить требуемую обработку, недостаточно располагать только тем, на чем это можно выполнить, необходимо еще имей указания, как это выполнить. Такие указания содержатся в наборе программ, представляющих программную часть системы обработки.

Совокупность, организованный набор программ различного назначения обеспечивающих функционирование микропроцессорной системы, автоматизации программирования и решение задач, называют *программным или математическим обеспечением.*

Различают системное программное обеспечение, не зависящее от конкретного применения (его поставляет изготовитель микропроцессора или ЭВМ), и программное обеспечение пользователя, разрабатываемое последним для решения определенных задач. Это две частя единого программного обеспечения.

Создание программного обеспечения микропроцессорной системы (микро ЭВМ)—трудоемкая и дорогостоящая процедура. Стоимость программной части во много раз превышает затраты на аппаратную часть или, как образна сказано в одном из журналов «Electronics», стоимость программного обеспечения «затмевает стоимость всех других элементов системы». Для снижения затрат программное обеспечение стремятся строить так, чтобы его можно было использовать многократно, в микропроцессорных системах различных видов. Системы, для которых применимо общее программное обеспечение, называют программно-совместимыми. Для них характерны единые набор команд, форма представления данных, система адресации.

Как уже отмечалось, программа—это совокупность команд для выполнения определенной операции или функции. Нередко программа складывается из определенного числа стандартных программ или подпрограмм, согласно которым производятся специфические вычисления, решаются частные задачи. Значимость некоторых стандартных программ столь велика или они так часто применяются, что в микропроцессорной системе (или микро-ЭВМ) эти программы хранятся в закодированной форме. Ее называют микрокомандой.

Для значительной части выпускаемых микропроцессоров, а также микро-ЭВМ и микроконтроллеров характерно «индивидуальное» программное обеспечение, что усложняет применение микропроцессоров и построение микропроцессорных систем.

**Задача и этапы ее решения**. Произнося слово «задача», мы не всегда задумываемся над вопросом: «А что же такое задача?» С позиций использования ЭВМ вряд ли нас может удовлетворить объяснение, содержащееся в толковом словаре: «Вопрос, требующий разрешения, то, что задано для решения». Более приемлемым представляется такое определение: задача—это совокупность исходных данных и четко поставленный вопрос, какой результат должен быть получен на их основе; отыскание этого результата и является решением задачи.

Процесс решения задачи на ЭВМ состоит из следующих этапов: математической постановки задачи, выбора или разработки алгоритма решения, представления алгоритма в виде структурной схемы, программирования, отладки программы, решения задачи. Кратко осветим каждый из перечисленных этапов. *Математическая постановка задачи* предполагает запись условия задачи с помощью математических выражений, обозначений, четкое формулирование исходных данных и искомого результата вычислений, определение формы представления результата.

Сущность разработки алгоритма решения задачи вытекает из его определения. *Алгоритмом* называют точное предписание о выполнении в определенном порядке совокупности элементарных операций с целью решения любой задачи из некоторого заданного класса задач. Иначе говоря, алгоритм — это правило решения задачи, строгое выполнение которого приводит к искомому результату. Осуществление последовательности действий, предписанных алгоритмом, связано главным образом с вычислениями. Поэтому процедуру выполнения алгоритма называют вычислительной процедурой. Итак, для решения задачи необходимо разработать алгоритм и провести соответствующие вычисления. Процесс создания алгоритма решения задачи называется алгоритмизацией решения. Ее проводит не ЭВМ, а человек.

*Программирование,* т. е. написание программы, — это запись алгоритма с помощью формализованной системы знаков, принятой для описания процедур решения задач на ЭВМ. Такие знако-

вые системы называются языками программирования. Программу, как и алгоритм, пишет человек — программист Для составления эффективной программы требуется специалист высокой квалификации, не только в совершенстве владеющий принципами программирования, но и хорошо понимающий решаемую задачу, а также знающий применяемую микропроцессорную технику.

*Отладка программы* — это процедура проверки правильности составленной программы, выявления ошибок и устранения их.

*Решение задачи* — автоматическое выполнение вычислительной процедуры согласно отлаженной программе, которая введена в ЭВМ.

**Алгоритм и его структурная схема.** Из изложенного видно, что составление алгоритма — это не только первый, но и очень важный этап решения задачи. Для многих задач может быть получено несколько алгоритмов решения. Из них стремятся выбрать тот, который оптимален по заданному критерию. Нередки ситуации, когда алгоритмизация решения связана с выполнением большого объема работы, обусловленной необходимостью четкого определения последовательности операций, установления логических связей между отдельными операциями или их группами, формулирования условий, от выполнения или невыполнения которых зависит дальнейший ход вычислений, и т. д. Поэтому следует иметь представление об основных свойствах алгоритма и способах его наглядного изображения.

Отметим три основных свойства, которыми должен обладать алгоритм: *определенность*— однозначность толкования элементов, предписывающих выполняемые операции;

*массовость*—возможность использования данного алгоритма для решения широкого класса аналогичных задач с различными исходными данными;

*результативность* — возможность получения определенного результата при допустимых исходных данных за конечное число шагов.

Наиболее удобная и наглядная форма представления алгоритма решения задачи, логики построения программы — графическая, в виде структурной схемы. Она состоит из упорядоченной совокупности условных геометрических фигур (блоков), несущих информацию об определенном характере операций, накладываемом условии, принятых обозначениях. Описания содержания операции или условия помещаются внутри фигур, которые нумеруют. На рис. 1.11 приведены стандартные обозначения фигур. Поясним их.

Фигуру, показанную на рис. 1.11,а, используют для обозначения начала или конца программы. Изображенная на рис. 1.11,б фигура служит для обозначения ввода данных в ЭВМ и вывода данных из нее. Трапеция (рис. 1.1l,в) отмечает ручную операцию. В прямоугольник (рис. 1.11,г) помещают выполняемые операции или группы операций (процесс). Ромб (рис. 1.1,д) обозначает условный оператор — логический блок, им пользуются при проверке условий тили сравнении величин. Направление перехода от данной операции к следующей, т. е. развития вычислительной процедуры, указывают с помощью стрелки (рис. 1.11,е). Окружность малого диаметра, приведенную на рис. 1.11,ж, называют страничным соединителем, указывающим, что продолжение схемы алгоритма находится на той же странице. Наконец, значок, изображенный на рис 1.11 з и называемый межстраничным соединителем, служит для сообщения о том, что продолжение схемы алгоритма находится на другой странице. Внутри соединителей обычно проставляют метки (цифры), облегчающие поиск продолжения схемы алгоритма.

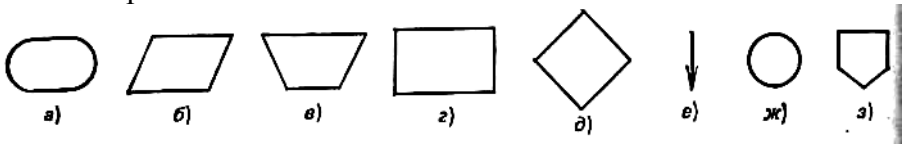


*Рис. 1.11*

Существуют три типа *структурных схем алгоритмов:* системная схема, которую строят для того, чтобы показать, какие устройства ввода-вывода требуются при решении задачи, а само решение в этой схеме представляют одним блоком;

основная схема, более подробно, но все же в общих чертах описывающая решение задачи;

детальная схема, которая составлена настолько подробно, что в одном блоке помещаются одна или две команды из системы команд данной ЭВМ. Эта схема начинается с блока «Начало» и завершается блоком «Конец».

**Языки программирования.** Крылатые слова «найти общий язык», пришедшие, как утверждают лингвисты, из дипломатической практики, означают «достичь взаимопонимания». В человеческом обществе язык — это средство общения между людьми. Но не при всяком общении используется язык слов — литературный язык. Нередки ситуации, когда для передачи мысли, замысла одного человека другому человеку, который должен воспринять высказанную мысль, эффективнее формализованный язык, состоящий из символов, знаков. Так, музыканту для воплощения замысла композитора нужны не слова, а совокупность нотных знаков, токарю — чертеж, составленный конструктором, а радиомонтажнику — электрическая функциональная или принципиальная схема разработанного устройства.

Общение человека с ЭВМ осуществляется с помощью специализированных формальных языков, называемых языками программирования. Они служат для описания совокупности инструкций, выполнение которых гарантирует правильное решение задачи. Опираясь на ранее введенные понятия, можно сказать, что языки программирования применяют для описания вводимых в ЭВМ данных и алгоритмов их обработки. Основное назначение языка программирования — хранение и передача алгоритма. Классифицируют эти языки по нескольким признакам. Однако, прежде чем говорить о классификации, необходимо отметить следующее.

В ЭВМ операции выполняются в двоичной системе счисления. Машина «понимает» только двоичную форму команды: совокупность нулей и единиц. Поэтому каждый элемент данных, вводимых в память микропроцессора или микропроцессорной системы (микро-ЭВМ, микроконтроллера), должен быть представлен в двоичной форме. Команды, выполняемые системой согласно программе, и адреса операндов также представлены двоичными кодами. Однако при составлении (написании) программы они могут быть записаны в любой Другой форме с использованием различной символики.

В зависимости от уровня языка, т. е. степени детализации шагов при выполнении программы, различают машинный язык, язык ассемблера и языки высокого уровня. Охарактеризуем каждую из этих языковых разновидностей.

*Машинный язык* — язык низкого уровня. Для него характерна высокая степень детализации шагов. Это единственный язык, который непосредственно воспринимает, «понимает» микропроцессор. Язык представляет собой систему команд, выраженных в двоичных кодах, или систему инструкций.

Для программиста машинный язык неудобен. Дело в том, что программа представляет собой длинную последовательность нулей и единиц: все коды операций, операнды и адреса операндов записываются в двоичном коде. Написание большой программы трудоемко, а чтение затруднительно. Программист должен помнить двоичные коды команд и стремиться к рациональному распределению памяти микропроцессора. На составление программы приходится затрачивать много времени, возможно значительное число ошибок. Отладка программы получается сложной и продолжительной.

Оправдано составление на машинном языке простых программ. Такие программы могут быть написаны с минимальной затратой времени и использованием малого объема памяти.

К достоинствам машинного языка относят: сравнительно высокую производительность при написании короткой программы (примерно до 250 байт), минимальный объем занимаемой ею памяти, непосредственный ввод результатов в устройство подготовки данных ППЗУ, возможность обходиться без дополнительных аппаратных средств, отсутствие необходимости расходовать машинное время для трансляции программ (сущность трансляции поясняется ниже). Недостатками этого языка являются: большая трудоемкость и низкая производительность составления объемных программ, высокая вероятность ошибок и сложность их выявления, трудности развития или сокращения ранее написанных программ [31].

*Язык ассемблера* (от англ. to assemble — собирать, компоновать) — это символическое представление машинного языка. Команды языка ассемблера позволяют существенно упростить составление, чтение и отладку программ функционирования микропроцессорной системы по сравнению с командами на машинном языке. Язык ассемблера, образованный в результате мнемонического (от греч. mnemonikon — искусство запоминать) или символического кодирования, считают наиболее простым языком программирования. **Имеется практически взаимооднозначное соответствие между командами языка ассемблера и машинными командами микропроцессора**. Каждая машинная команда обозначается символом, представляющим собой сокращенную форму полной за-

писи наименования данной команды на английском языке. Символ кодирования названия и содержания команды запоминается и воспринимается намного легче, чем сочетание нулей и единиц. Примеры типичных операций и их мнемонических обозначений приведены в табл. 1.

При записи полного набора команд программы мнемонические обозначения применяют как для кода операции, так и для операндов и их адресов (последние представляются символическими именами).

Так как микропроцессор «понимает» только язык двоичных кодов, то необходим языковый транслятор [от англ. Translator — переводчик (с одного языка на другой)], преобразующий программу, написанную на определенном языке программирования, в машинные коды. Подготовка программы на машинном языке путем замены символических названий операций на машинные коды и символических адресов на абсолютные или относительные адреса называется ассемблированием (абсолютный адрес — это адрес, который непосредственно определен в самой команде; относительным называют адрес, определяемый относительно адреса счетчика команд: для получения адреса операнда указанный в команде адрес суммируется с содержимым счетчика команд). Программист получает распечатку программ, называемую листингом, который содержит и исходную, и транслированную программы.

| Операция | Английское наименование | Мнемоническое обозначение |
|---|---|---|
| Записать в память | STORE | ST |
| Сложить | ADD | AD |
| Вычесть | SUBSTRACT | SUB или SU |
| Переслать . | MOVE | MOV |
| Загрузить | LOAD | LD или L |
| Увеличить на 1 | INCREMENT | INR |
| Уменьшить на 1 | DECREMENT | DCR |
| Логическое И | AND | AN |
| Логическое ИЛИ | OR | OR |
| Исключающее ИЛИ | EXCLUSIVE OR | XOR или XO |
| Сравнить два числа | COMPARE | CMP |
| Сдвинуть влево на 1 разряд | ROTATE LEFT in CARRY | RLC |

Следует сказать, что языковые трансляторы вообще облегчают процедуру программирования, упрощают отладку программ. Они способствуют расширению круга пользователей, которые составляют программы самостоятельно, без привлечения профессиональных программистов. Процесс трансляции выполняется с помощью ЭВМ (не обязательно той, для которой пишется данная программа). Транслятор, который осуществляет перевод программы, написанной на языке ассемблера, в машинный код, называют ассемблером.

Язык ассемблера — наиболее распространенный язык программирования. Его главное преимущество заключается в возможности эффективного использования особенностей микропроцессора. Но следует иметь в виду, что этот язык ориентирован на конкретную микро-ЭВМ, микропроцессорную систему. При необходимости работы с другой микро-ЭВМ, относящейся к иному классу машин, язык ассемблера, сохраняя общие свои черты, будет иным. Различия сводятся в основном к терминам, используемым в командах, и способам адресации операндов. Необходимо учитывать специфические особенности конкретного микропроцессора при пользовании языком ассемблера, т. е. программа на нем составляется для микропроцессора определенного типа. Таким образом, язык ассемблера относится к машинно-ориентированным языкам программирования. Из этого следует, что специалист, пишущий программу для данной микропроцессорной системы, должен хорошо знать ее устройство и особенности построения, ясно представлять ее архитектуру.

Программа на языке ассемблера—это последовательность команд (операторов), описывающая решение задачи. Все команды имеют одинаковую структуру. Команда, написанная на языке ассемблера, делится на четыре части, которые называют полями:

*Метка Операция Операнд Комментарий.*

Кратко охарактеризуем их.

Метка — это имя, присваиваемое тем командам, на которые в программах имеются ссылки. Иначе говоря, помечают те операции, к которым хотят обращаться в программе по ходу ее выполнения. Но так как ссылаются не на каждую команду, то отдельные поля остаются пустыми. Поле метки отделяют» от соседнего поля двоеточием.

Операция — это действие, которое предписывается командой. Поле операции содержит мнемонический код операции.

Операнды — данные, над которыми выполняется операция. В поле операнда может быть помещен не сам операнд, а его адрес. Если в этом поле занять два операнда, то их разделяют запятой.

Комментарий—это информация, полезная программистам и служащая для пояснения команды. Он повышает удобство чтения программы, но не воспринимается вычислительной машиной, не используется ассемблером в процесс трансляции. Поле комментария обычно отделяется от предыдущего поля точкой с запятой.

Резюмируя изложенное, к достоинствам языка ассемблера следует отнести: легкое восприятие и запоминание символических кодов и ссылок, сравнительно высокую производительность составления программ средней длины (д( 1000 байт), возможность использования средств, облегчающих программирование, относительную простоту изменения программы в процессе ее отладки или при модернизации. Недостатки, с которыми сталкиваются составители про грамм, сводятся к следующему: необходимо изучить правила и форматы языка его машинно-ориентированный характер, требующий знания структуры микропроцессора и микропроцессорной системы, для которой пишется программа; значительные затраты времени на программирование отдельных операций; потребность иметь дополнительную ЭВМ для трансляции программ в некоторых ситуациях

*Языки программирования высокого уровня* позволяют упростить и ускорить составление программ. Эти языки являются машинно-независимыми, чт( означает возможность использования программы, составленной на подобном, языке, для различных ЭВМ. Различают процедурно-ориентированные и проблемно-ориентированные языки высокого уровня, хотя четкую грань между ни ми провести трудно. Процедурно-ориентированные языки содержат средств; выражения характерных алгоритмических действий: вычисления выражений, про верки условий, выполнение процедур циклических вычислений, включение под программ (подпрограммой называют часть программы, представляющую собой последовательность команд, неоднократно выполняемых в различных раз делах программы). Эти языки освобождают программиста от записи программ, похожих на машинные. Проблемно-ориентированные языки ориентированы на определенные классы однотипных задач и представляют своим пользователя средства определения более или менее широкого набора функций, подлежащих выполнению. Такие языки определяют задание для ЭВМ в терминах функций, которые надо выполнить без детализации шагов, позволяющих осуществить эти функции. Проблемное ориентирование достигается в результате использования пакетов прикладных программ (иначе называемых проблемными или функциональными), т. е. программ, осуществляющих решение задачи, необходимой непосредственно пользователю вычислительной системы.

К языкам высокого уровня относятся:

АЛГОЛ [от англ. Algo(ritmic) L(Language) язык алгоритмов — алгоритмический язык, ориентированный на решение задач численного анализа;

ФОРТРАН [от англ. Formular Translation) трансляция формулы — язык программирования, разработанный специально для решения научных и инженерных задач;

КОБОЛ (от англ. Cobol—от аббревиатуры слов Common business oriented Language — общепринятый деловой язык) — язык, ориентированный на решение экономических, коммерческих задач, связанных с организацией больших массивов и их обработкой;

БЕЙСИК(от англ. Basic — аббревиатуры слов Beginners all purpose symbolic istruction code — универсальный символический код команд для начинающих) — наиболее широко распространенный диалоговый язык, т. е. язык взаимодействия человека с машиной, сравнительно простой в употреблении;

ПАСКАЛЬ (название — в честь известного французского математика, физика и философа Блеза Паскаля, который в 1642 г. сконструировал счетную машину) — язык программирования,

ориентированный на обучение программированию как систематической дисциплине, которая основана на ряде фундаментальных понятий, отраженных в этом языке;

ПЛ/1 и ПЛ/М [PL — аббревиатура английских словProgramming Language (язык программирования), а буква М указывает, что язык предназначен для программного обеспечения микропроцессоров — язык для алгоритмизации широкого класса научно-технических и информационных задач (язык ПЛ/М—подмножество языка ПЛ/1), он представляет попытку синтеза лучших свойств Фортрана, Алгола, Кобола, но имеет и ряд новых свойств, существенно расширяющих область его применения;

ЛИСП (от англ.LISP — аббревиатуры слов List Processing — обработка списков) — язык программирования, ориентированный на обработку списков; широко используется для программирования задач искусственного интеллекта. Кроме перечисленных используются и другие языки высокого уровня, в том числе и появившиеся за последнее время.

Естественно, что для ввода программы, написанной на языке высокого уровня, в микропроцессорную систему (микро-ЭВМ) необходимы трансляторы. Получаемые в результате трансляции программы, представленные на машинном языке, требуют существенно большего объема памяти, чем ассемблированные программы. Мера превышения зависит от опыта программиста. Важно подчеркнуть, что высококвалифицированный программист составляет программу на языке высокого уровня в 5—10 раз быстрее, чем на языке ассемблера.

Таким образом, достоинством языков программирования высокого уровня являются значительное сокращение продолжительности написания программы, уменьшение затрат на программирование, возможность использования программы, составленной для одной ЭВМ, при решении этой задачи на другой ЭВМ, более простое управление программами и сравнительно легкая передача в эксплуатацию. Однако следует ясно представлять и недостатки: необходимы программисты высокой квалификации (с большим опытом), нередко — дополнительная затрата средств и времени для трансляции составленных программ на машинный язык, осуществляемой с помощью больших ЭВМ, значительный объем памяти, занимаемый программой, получаемой после трансляции. Учитывая достоинства и недостатки программирования на языках высокого уровня, специалисты полагают, что его следует применять при больших программах (объёмом 1000 байт и более), для опытных образцов и изделий, выпускаемых мелкими сериями, для изделий, намеченных к производству большими сериями, программы составляют программисты высокого класса [31].

**Виды программ**. Как отмечалось в начале параграфа, программное обеспечение представляет совокупность программ различного назначения. Приведем названия основных разновидностей программ и их определения.

*Исходной* называется программа работы микропроцессорной системы (микро-ЭВМ), написанная на языке ассемблера или языке высокого уровня. С помощью транслятора исходная программа преобразуется в *объектную программу*— программу, представленную на машинном языке. Перевод с языка ассемблера осуществляется с помощью программы ассемблера, а трансляция программы, написанной на языке высокого уровня, в объективную выполняется посредством программы-компилятора После проверки и корректировки объектной программы получается *рабочая программа.* Указанные программы — исходная, объектная и рабочая — составляют программное обеспечение пользователя.

Для выполнения трансляции исходной программы в объектную, получения рабочей программы и ввода ее в микропроцессорную систему изготовитель микропроцессора или микро-ЭВМ снабжает пользователя набором вспомогательных (служебных, сервисных) программ. Они относятся к *системному программному обеспечению*, которое разделяют на **резидентное** программное обеспечение и **кроссовое** программное обеспечение. Резидентное обеспечение состоит из набора служебных программ, реализованных в той же микро-ЭВМ (микропроцессорной системе), которая будет работать согласно прикладной программе пользователя. Кросс - программное обеспечение — совокупность служебных программ, предназначенных для создания рабочих программ на микро-ЭВМ, отличающейся от той микро-ЭВМ, которая будет выполнять, рабочую программу. В состав кросс-программного обеспечения входят кросс- ассемблер, кросс-компилятор и имитатор.

Программа-имитатор — это служебная программа, позволяющая имитировать (моделировать) работу различных микропроцессорных систем с помощью одной и той же системы разработки — оборудования, которое преобразует исходную программу в рабочую.

Служебная программа, обеспечивающая загрузку (размещение) рабочей программы в отве-

денной области памяти и контроль правильности расположения в ней адресов начала и конца программы, называется программой-загрузчиком.

Управление функционированием микро-ЭВМ при трансляции, проверке, корректировке и вводе прикладных программ пользователя проводится с помощью резидентной программы-монитора.

Специальная служебная программа, применяемая для проверки объектной программы, носит название программы-отладчика. Эта программа входит в состав имитатора. Корректировка. и редактирование программы пользователя после того, как она нанесена на носитель информации, достигаются с помощью служебной программы, называемой редактором текста, или просто редактором.

Отладка программы. При составлении программы могут быть допущены ошибки. Их необходимо выявить и исключить. Эта процедура называется *отладкой программы.* Для проведения отладки используют аппаратные и программные средства. Как уже указывалось, существует специальная служебная программа-отладчик. Ее записывают в ПЗУ микропроцессорной системы. Отладчик позволяет проверить содержимое ячеек внешней памяти и регистров микропроцессора, которое при необходимости может быть изменено. Если в проверяемой программе указаны контрольные точки, то по достижении такой точки выполнение программы приостанавливается. Программист получает возможность сверить состояние микропроцессорной системы с ожидаемым состоянием и сделать заключение о том, что между какими двумя контрольными точками находится ошибка. Уменьшением расстояния между двумя точками удается локализовать ошибку. Иногда для исправления ошибки программы в нее требуется вставить несколько команд. Такую вставку делают в исходной программе, и после трансляции получается новая версия объектной программы.

Касаясь аппаратных средств, применяемых для отладки, следует сказать об эмуляторе. Под эмуляцией [от англ. Emulation — подражание (примеру)] в вычислительной технике понимают метод приспособления одних вычислительных машин к решению задач, подготовленных для других машин. Иначе говоря, эмуляция — это имитация одной системы средствами другой системы без какой-либо потери функциональных возможностей или искажения получаемых результатов. Эмулятор представляет собой специализированный программируемый контроллер, который при выполнении отладки заменяет микропроцессор в микропроцессорной системе. Подобный блок позволяет без перепрограммирования выполнять на проверяемой микропроцессорной системе программу, использующую коды или способы выполнения операций, отличные от данной микропроцессорной системы.

Ввод программы в память микропроцессорной системы. Если программа находится в ПЗУ, то она в него введена изготовителем и ее изменить нельзя. Однако нередко в составе микропроцессорной системы содержатся ППЗУ или РППЗУ, которые могут программироваться пользователем. Для ввода данных в ячейку запоминающего устройства требуется указать адрес ячейки и помещаемое в нее число.

Задачу ввода информация в микро-ЭВМ решают с помощью внешних ЗУ, системы переключателей и клавиатуры. Так как последний способ используется в измерительных приборах, содержащих микропроцессорные системы, то рассмотрим ввод данных посредством шестнадцатеричной клавиатуры . При такой клавиатуре ускоряется процедура ввода и уменьшается вероятность ошибки по сравнению с системой переключателей.. В ее составе 16 клавиш шестнадцатеричного кода (обозначенные цифрами от 0 до 9 и буквами *A, B, C, D, E, F* и .девять функциональных клавиш. Процедура ввода заключается в следующем :

После набора адреса начальной ячейки памяти (посредством соответствующих кодовых клавиш) нажимают клавишу УСТ АДР (установить адрес). Затем с помощью кодовых клавиш формируют данные, которые требуется записать в указанную ячейку памяти. Для ввода данных прикасаются к клавише ЗП с ИНКР (запись с инкрементированием). При этом данные направляются в ячейку указанного адреса и формируется адрес следующей ячейки (номер предыдущего адреса увеличивается на единицу).

Завершив ввод программы и данных в запоминающие устройства, для проверки правильности результатов ввода нажимают клавиши ЧТ с ДЕКР (чтение с декрементированием) и ЧТ с ИНКР (чтение с инкрементированием). Нажатие первой клавиши приводит к выборке предыдущего адреса, а нажатие второй — сопровождается выборкой следующего адреса.

В случае констатации отсутствия ошибок ввода, набрав адрес первого байта программы, по-

следовательно нажимают клавиши УСТ АДР в ПУСК- Начинается выполнение программы.

Так как с нажатием клавиши ПУСК клавиатура отключается от микропроцессорной системы, то, когда требуется возобновить возможность управления с помощью клавиатуры, нажимают клавишу СБР (сброс). Она связана непосредственно с микропроцессором. Клавишами ЗАПОМ ДАН (запомнить данные) и ЗАГР ДАН (загрузить данные) пользуются тогда, когда необходимо записать данные носитель и переслать их в основную память. При нажатии первой из них данные пересылаются из основной памяти в кассетное запоминающее устройство, а после нажатия второй — пересылка осуществляется противоположным образом: с кассетного запоминающего устройства в основную память. Управление микропроцессорной системой во время ввода рабочей программы производит программа-монитор.

Автоматизация программирования. Создание программного обеспечения, как уже отмечалось, — трудоемкая и дорогостоящая процедура. Ее осуществление облегчает наличие стандартизованных прикладных программ, однако для согласования их с требованиями конкретного пользователя необходимо участие программиста. Автоматизация программирования, при которой сама микро-ЭВМ выполняет многие функции программиста, открывает возможности резкого повышения производительности, скорости разработки программных средств. При этом постановка или описание задачи обычно осуществляется в диалоговом режиме в соответствии с ответами пользователя на «вопросы-подсказки» (отображаемые на дисплее ЭВМ), которые позволяют полно описать задачу. После этого начинают работать средства автоматического программирования, транслируя описание задачи в отлаженную (рабочую) программу. Помимо уменьшения трудоемкости и продолжительности создания программы при автоматизации программирования отпадает необходимость проверки и отладки программы, так как они получаются практически безошибочными. Однако следует иметь в виду, что в настоящее время техника автоматизации создания программного обеспечения находится в фазе своего развития. Пока еще она охватывает ограниченный круг прикладных задач, хотя перечень появляющихся прикладных программ расширяется и перспективы автоматического программирования оцениваются оптимистически.