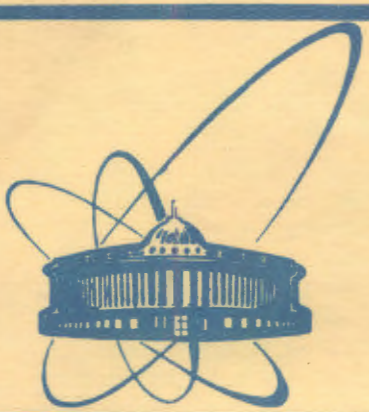

сообщения
объединенного
института
ядерных
исследований
дубна

10 - 12679

И.Н.Чурин

ДИАЛОГОВЫЙ РЕДАКТОР ТЕКСТА И АССЕМБЛЕР ДЛЯ ЭВМ НА ОСНОВЕ МИКРОПРОЦЕССОРА ИНТЕЛ-8080

1979

10 - 12679

И.Н.Чурин

ДИАЛОГОВЫЙ РЕДАКТОР ТЕКСТА И АССЕМБЛЕР ДЛЯ ЭВМ НА ОСНОВЕ МИКРОПРОЦЕССОРА ИНТЕЛ-8080

Чурин И.Н. 10 - 12679

Диалоговый редактор текста и ассемблер для ЭВМ на основе микропроцессора Интел-8080

Описывается способ подготовки новых программ, применяемый при отсутствии дорогостоящих магнитных накопителей данных и отличающийся тем, что позволяет для хранения промежуточной информации вместо перфоленты интенсивно использовать ОЗУ микро-ЭВМ. Этот способ реализуется в разработанной компактной резидентной программе диалогового характера, включающей редактор текста и ассемблер для микропроцессора Интел-8080. Программа предназначена для работы с ЭВМ на основе этого микропроцессора, снабженной памятью типа ОЗУ емкостью не менее 16 Кбайт и перфоленточным оборудованием. Она занимает объем памяти 5 Кбайт и может быть помещена в ПЗУ.

Работа выполнена в Лаборатории ядерных проблем ОИЯИ.

Сообщение Объединенного института ядерных исследований. Дубна 1979

Churin I.N. 10 - 12679

Dialogue Text Editor and Assembler for the Microcomputer Based on Intel-8080 Microprocessor

A program development method for the small computer systems not provided by expensive magnetic storage devices is described. The feature of the method is an intensive use of the microcomputer's RAM as a storage medium for the temporary intermediate datum but not the conventional paper tape. This method is implemented in the described small-size resident dialogue-type program, which includes text editor and assembler for the Intel-8080 microprocessor. This program runs on Intel-8080 based microcomputers equipped by the 16 K bytes RAM and paper tape devices. The program takes 5 K bytes of the memory space. It may be allocated in PROM's.

The investigation has been performed at the Laboratory of Nuclear Problems, JINR.

Communication of the Joint Institute for Nuclear Research. Dubna 1979

ВВЕДЕНИЕ

По мере увеличения степени интеграции больших интегральных схем расширяются их функциональные возможности и снижается стоимость. В то же время значительно возрастают объем и сложность программного обеспечения, создаваемого для каждой новой микропроцессорной системы. На разработку программного обеспечения тратится основная часть усилий при создании новой системы с микро-ЭВМ, поэтому она ведется параллельно с разработкой ее аппаратной части.

В настоящей работе описывается способ подготовки новых программ, основанный на использовании для хранения промежуточной информации вместо перфоленты ОЗУ микро-ЭВМ в условиях, когда отсутствуют дорогостоящие магнитные накопители данных. При наличии дешевого полупроводникового ОЗУ размером не менее 16 Кбайт микро-ЭВМ можно снабдить эффективной программной системой для разработки новых программ. При этом использование бумажной ленты в качестве носителя информации сводится к минимуму.

Обычно в микро-ЭВМ часть ПЗУ служит для хранения программы-монитора, которая обеспечивает при включении питания инициализацию ЭВМ, позволяет загружать программы в ОЗУ с перфоленты, запускать программы, останавливать их выполнение, выводить содержимое памяти на перфоленту. Кроме этого, монитор содержит ряд полезных, часто вызываемых подпрограмм: для обработки прерываний, для связи с основным терминалом и другими внешними устройствами. С помощью монитора можно отлаживать загруженную в ОЗУ программу: расставлять точки останова, выводить на экран дисплея и изменять с клавиатуры содержимое регистров ЭВМ и любых ячеек памяти. Программа-монитор использует часть ОЗУ для хранения значений переменных величин, а также для размещения зоны стека.

При разработке новых программ обычно применяют сначала редактор текста для подготовки текстовой информации на перфоленте /рис. 1/. При этом с помощью монитора редактор текста загружается в ОЗУ, затем с клавиатуры дисплея или

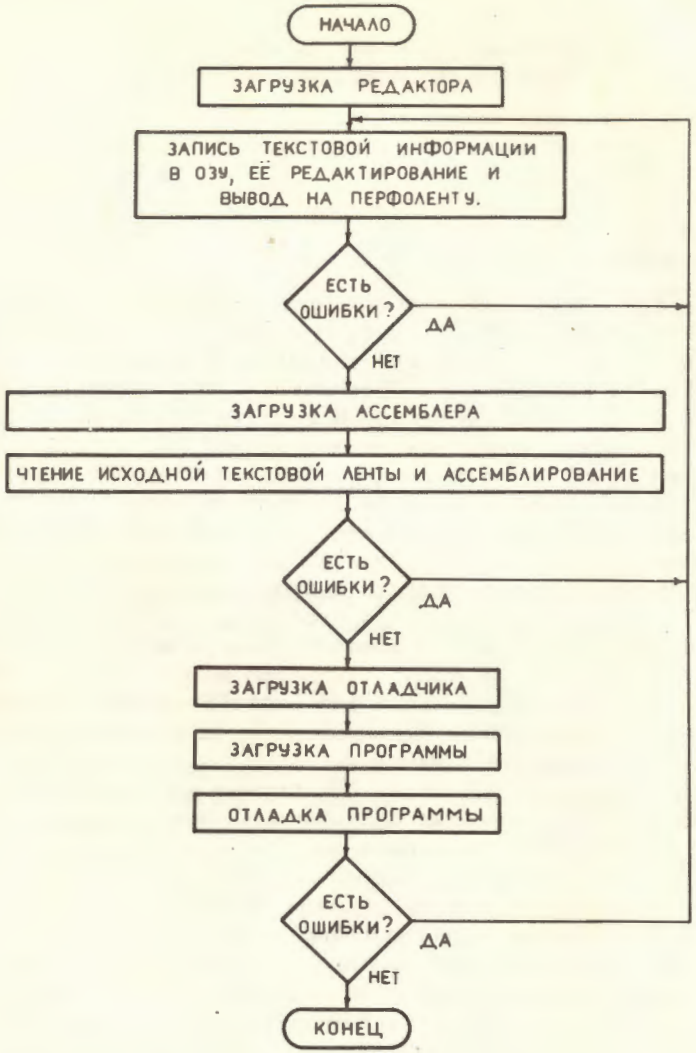

Рис. 1. Обычный способ разработки новых программ с поочередной загрузкой в ОЗУ программ редактора текста, ассемблера и отладчика.

с перфоленты производится запись текстовой информации в соответствующий буфер ОЗУ /текстовую рабочую зону/, ее редактирование и вывод на перфоленту. Далее в ОЗУ загружается программа ассемблера, которая считывает построчно исходную текстовую информацию с перфоленты и ассемблирует разрабатываемую программу. Считывание текстовой информации происходит два раза, так как ассемблеры обычно требуют двух просмотров. Обычно программа ассемблера использует ОЗУ только для хранения таблицы символов, созданной во время первого просмотра. Результаты ассемблирования сразу выводятся на внешние устройства: на перфоратор и на АЦПУ. Потом с перфоленты загружается программа отладки, если ее нет в ПЗУ, и загружается двоичная информация с перфоленты в двоичную рабочую зону. Далее новая программа отлаживается, пока не будут найдены ошибки. После этого в ОЗУ снова загружается редактор текста и весь процесс подготовки новой программы повторяется до тех пор, пока не будут устранены все ошибки.

Работая таким образом, невозможно избежать многократных прогонов через считыватель длинных перфолент, что сопряжено с потерей времени и появлением ошибок. Разработка даже короткой новой программы растягивается на большой срок.

С другой стороны, уже сейчас аппаратная часть микро-ЭВМ позволяет разместить в памяти как все три программы /редактор текста, ассемблер и монитор/, так и основные буфера данных: текстовую рабочую зону, двоичную рабочую зону и таблицу символов. Процедура подготовки новых программ в этом случае будет аналогична рассмотренной выше с той разницей, что в качестве промежуточного носителя информации вместо перфоленты используется ОЗУ микро-ЭВМ /рис. 2/. Программы редактора текста, ассемблера и отладчика можно хранить в ПЗУ микро-ЭВМ или загружать в ОЗУ с перфоленты один раз в начале работы. При этом переход от одного пункта процедуры к другому происходит легко и быстро, так как не надо работать с перфолентой. Текстовая рабочая зона служит для хранения и редактирования текста новой программы. Содержимое текстовой рабочей зоны после редактирования является исходной информацией для ассемблера. Результат ассемблирования в виде объектных /машинных/ кодов заносится в двоичную рабочую зону. Во время отладки управление может быть передано отлаживаемой программе, находящейся в двоичной рабочей зоне. Отлаженную работающую программу можно вывести на перфоленту, при этом адрес будущей загрузки программы может быть сделан любым.

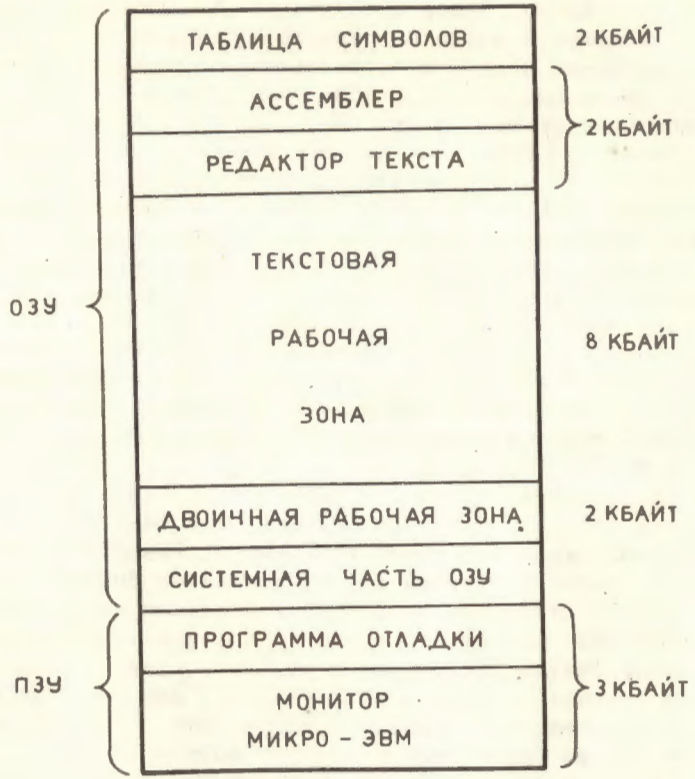

Рис. 2. Распределение памяти микро-ЭВМ при работе описываемого диалогового редактора текста и ассемблера.

РАБОТА ПРОГРАММ РЕДАКТОРА ТЕКСТА И АССЕМБЛЕРА

Программа содержит интерпретатор директив оператора, подпрограммы для создания, редактирования и копирования текстовой информации, находящейся в ОЗУ ЭВМ или на бумажной перфоленте, а также подпрограммы, выполняющие ассемблирование разрабатываемой программы, размещенной в текстовой рабочей зоне, выдачу двоичных кодов программы на перфоленту и распечатку таблицы символов. Программа работает следующим образом. После загрузки программы управление передается интерпретатору директив оператора /рис. 3/. Приняв с клавиатуры алфавитно-цифрового дисплея директиву оператора, он вызывает соответствующую подпрограмму, которая выполняет требуемую операцию с информацией в текстовой рабочей зоне. После выполнения вызванной подпрограммы интерпретатор директив сообщает оператору о готовности принять следующую директиву, выводя на экран знак (#). При выполнении директив, вызывающих ассемблирование, объектные двоичны коды загружаются в двоичную рабочую зону ОЗУ, кроме этого, можно выводить текстовую информацию и результаты ассемблирования на экран дисплея, перфоленту и АЦПУ.

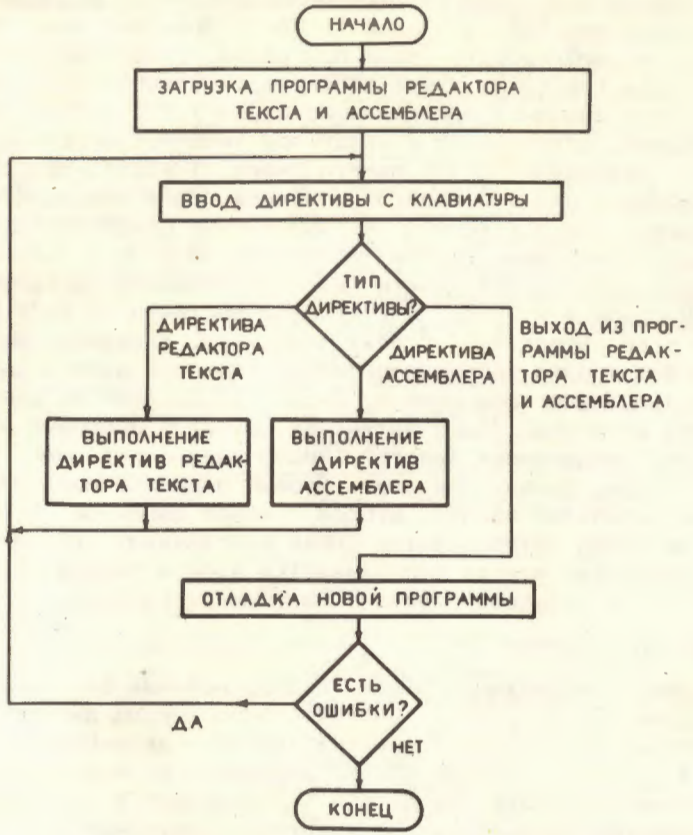

Рис. 3. Способ разработки новых программ с помощью диалогового редактора текста и ассемблера.

Размеры текстовой рабочей зоны зависят от размеров ОЗУ. Например, при ОЗУ величиной 16 Кбайт возможно подготавливать и ассемблировать новые программы, содержащие до 800 строк текста на языке ассемблера Интел-8080, что соответствует величине программы примерно 1,5-2 К байт. Подпрограммы, относящиеся к редактору текста, требуют для своего размещения объем памяти около 1 Кбайт, подпрограмма-ассемблер - около 2 Кбайтов, таблица кодов операций и стандартных меток - 2 Кбайта. На выполнение одной директивы при редактировании текста тратится от 0,5 до 1 секунды. Ассемблирование без печати листинга проходит со скоростью 60-70 строк в секунду, а с печатью листинга на АЦПУ Дaро-1156 - со скоростью 1 строка в секунду. Указанное быстродействие вполне удовлетворительно, так как работа выполняется в диалоговом режиме. Директивы состоят из идентификатора директивы /одна латинская буква/ и цифровых параметров, разделенных запятой. Число параметров может быть равно нулю, одному или двум. Первый параметр означает номер начальной строки, второй - номер конечной. В случае ошибки ввода интерпретатор снова запрашивает ввод директивы. Директива всегда заканчивается вводом знаков "возврат каретки" и "перевод строки". Например, директива

#D100, 120 CR LF

означает вычеркивание строк текста, начиная от 100-ой и кончая 120-ой строкой включительно. Список директив, выполняемых программой редактора текста и ассемблера, приведен в таблице. Указаны формат директив, их мнемоника и выполняемые операции. Директива V применяется при ассемблировании программ, когда не требуется получения распечатки, но желателен контроль результатов ассемблирования на отдельных участках программы. Директива E полезна для "быстрого ассемблирования", когда во время диалоговой отладки программы не нужно получения промежуточных распечаток.

ОПИСАНИЕ ПОДПРОГРАММЫ, ВЫПОЛНЯЮЩЕЙ АССЕМБЛИРОВАНИЕ

При ассемблировании используются в основном традиционные приемы. При первом проходе программы строится таблица символов. Происходит это следующим образом. После инициализации всех рабочих переменных ассемблера одна за другой просматриваются все строки исходной программы. Процесс просмотра каждой строки состоит из нескольких операций. Сначала вызывается подпрограмма SCAN, которая переносит

Таблица

Директивы, выполняемые программами редактора текста и ассемблера

Формат	Мнемоника	Выполняемая операция
CXX	CLEAR	Очистка текстовой рабочей зоны
DXX,YY	DELETE	Удаление группы строк из текста
IXX,YY	INSERT	Вставка группы строк в текст
RXX,YY	REPLACE	Замена группы строк в тексте
FXX		Загрузка текстовой информации с перфоленты в текстовую рабочую зону
LXX	LIST	Вывод строк текста на экран дисплея
WXX,YY	WRITE	Печать текстовой информации на АЦПУ
SXX,YY	SOURCE TAPE	Вывод текстовой информации на перфоленту
T	TYPE	Ассемблирование программы, текст которой находится в текстовой рабочей зоне, загрузка объектных двоичных кодов в двоичную рабочую зону, вывод листинга и диагностики ошибок на АЦПУ
V	VISUALISATION	Ассемблирование, выполняемое так же, как при директиве T, но с выводом листинга и диагностики ошибок на дисплей
E	ERRORS ONLY	Ассемблирование, выполняемое так же, как при директиве T, но с выводом на дисплей только диагностики ошибок. Листинг не выводится.
B	BINARY TAPE	Выдача двоичных кодов программы на перфоленту
M	MAP LABELS	Распечатка таблицы меток на АЦПУ

очередную строку из текстовой рабочей зоны в специально отведенный буфер строки, одновременно преобразуя свободный формат строки в фиксированный. Начало метки перемещается в первую позицию текстовой части буфера строки, начало мнемонического обозначения кода операции - в седьмую позицию, начало цепочки операндов - в позицию, следующую за пробелом после кода операции. Далее, если обнаружена метка, то с помощью подпрограммы FIND выполняется поиск метки в таблице символов. Если метка не была определена ранее, то ей присваивается текущее значение счетчика команд и она запоминается вместе с этим значением в таблице символов. С целью упрощения программы поиск в таблице идет линейно, путем последовательного перебора всех элементов таблицы. Стандартная часть таблицы включает в себя все машинные коды микропроцессора, псевдокоманды и стандартные метки используемой микро-ЭВМ. Добавляемые новые метки с их значениями образуют нестандартную часть таблицы символов. Для сокращения среднего времени поиска порядок расположения элементов таблицы в ее стандартной части оптимизирован так, что наиболее часто встречающиеся коды команд находятся в начале таблицы. После занесения метки в таблицу символов проверяется код команды, находящийся в данной строке, а число байтов, равное длине найденной команды, прибавляется к значению счетчика команд. Если была обнаружена псевдокоманда, то управление передается подпрограмме обработки псевдоопераций. Этим заканчивается обработка каждой строки. При первом проходе программы обнаруживаются такие ошибки, как дважды определенная метка, неправильный формат строки, неправильный код операции и др.

При втором проходе каждая строка обрабатывается вновь. Проверяются и вычисляются коды операций и аргументы, генерируются двоичные объектные коды программы, которые сразу загружаются в двоичную рабочую зону ОЗУ ЭВМ. Одновременно формируются и, по требованию, выводятся строки листинга. Как и при первом проходе, обработка строки начинается вызовом подпрограммы SCAN, при этом заполняется текстовая часть буфера строки. Далее формируется числовая часть буфера строки. С первой позиции начинается номер строки листинга, с седьмой - текущее значение счетчика команд, с 14-ой, 18-ой и 20-ой позиций начинаются значения соответственно 1-го, 2-го и 3-го байтов команды. Далее, через пробел следует текстовая часть буфера строки. Для обработки операндов вызывается подпрограмма GET, которая анализирует вид операнда и находит его числовое значение. Если операнд символический, то происходит поиск его значения в таблице символов.

Если операнд представлен в виде восьмеричного, десятичного или шестнадцатиричного числа, то его внутреннее представление будет найдено с помощью соответствующих программ преобразования целых чисел. Допускается также представление операнда в виде текстовой константы, при этом текст заключается между двумя знаками апострофа. Найденное значение аргументов подставляется во 2-й и 3-й байты команды и преобразуется для заполнения полей 2-го и 3-го байтов в буфере строки. Строка выводится на экран дисплея или на АЦПУ. Как первый, так и второй проходы заканчиваются при обнаружении псевдокоманды END выводом диагностического сообщения о количестве ошибок во время просмотра. Выполнение псевдокоманды ORG сводится к присвоению начального значения счетчику команд. В данной версии ассемблера псевдокоманда ORG может быть использована один раз только в начале программы, что связано с наличием одной двоичной рабочей зоны. Если эта псевдокоманда не применялась, то счетчик команд первоначально получает нулевое значение. Псевдокоманда DS служит для резервирования памяти и декларирования массивов. Псевдокомандами DB и DW можно задавать значения числовым, адресным или текстовым константам в программе. Длина этих констант может быть равна одному или двум байтам. Псевдокоманда EQU позволяет на период трансляции присвоить заданным символам необходимое значение /при этом переопределение символов не допускается/.

Запуск программы осуществляется передачей управления на первую команду программы. Прерывание работы программы возможно в любое время с клавиатуры дисплея, повторный запуск не изменяет содержимого текстовой рабочей зоны, что удобно при отладке программы. Имеются два варианта ассемблера с распечаткой чисел на листинге как в восьмеричном, так и в шестнадцатиричном представлении.

•Описанная программа редактора текста и ассемблера, предназначенная для разработки новых программ, успешно эксплуатируется в течение двух лет в Лаборатории ядерных проблем ОИЯИ на микро-ЭВМ, основанных на микропроцессоре Интел-8080. Программа проста в изучении и пользовании.

Автор приносит благодарность А.Н.Синаеву и В.Т.Сидорову за большую помощь в работе.

Рукопись поступила в издательский отдел
18 июля 1979 года.