Тема 8. Машинна мова і Асемблер. Простий склад команд мікропроцесора.

Общие понятия об Ассемблере

Все языки программирования можно подразделить на три группы:проблемно-ориентированные, машинно-ориентированные и машинные.

Машинные языки - это языки цифр. Они занимают самый нижний уровень в иерархии языков программирования и весьма неудобны для программиста при написании и отладке программ.

К проблемно-ориентированным языкам, или языкам высокого уровня, относятся БЭЙСИК, ФОРТРАН, ПАСКАЛЬ, Си и др. Они обладают относительной машинной независимостью, и программа для них пишется, исходя из существа задачи. Проблемно-ориентированными языками для микроЭВМ являются БЭЙСИК и Си.

Машинно-ориентированный язык мнкроЭВМ называется языком Ассемблера. Поскольку микроЭВМ, производимые различными изготовителями, не похожи друг на друга, отличаются и языки Ассемблера, но незначительно. Свое название языки Ассемблера получили от имени программы транслятора» преобразующей исходную программу (написанную на языке Ассемблера) в объектную (программу на машинном языке» понятную ЭВМ). Каждый язык Ассемблера является машинно-зависимым и отражает аппаратурные особенности (в частности, состав программно* доступных регистров) той микроЭВМ, для которой он создан.

На своем рабочем уровне микропроцессор реагирует на список операций, называемый машинной программой. На рис. 8.1, *а* приведено содержимое памяти, являющееся программой на машинном языке. Эта программа начинается с адреса 2000Н с содержимым КОП 0011 1110_2 и оканчивается адресом 2006Н с содержимым 0111 0110_2. Человеку практически невозможно понять программу, представленную в такой форме.

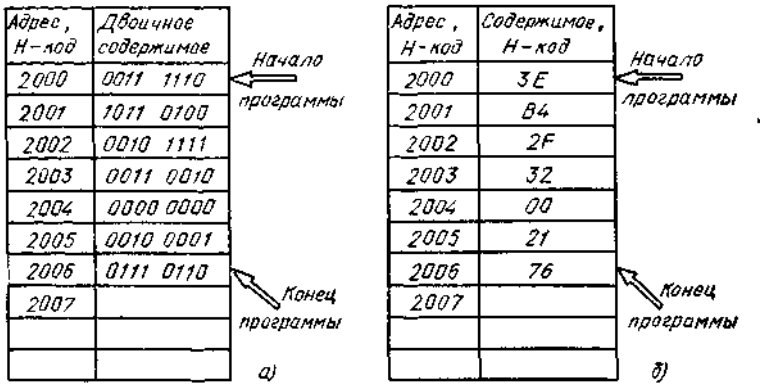

Рис. 8.1. Программы:
а — в двоичном машинном коде; *б* —в шестнадцатеричном машинном коде

Программа на машинном языке на рис. 8.1, *а* становится несколькопроще для восприятия, когда она представлена в шестнадцатеричном коде (Н-коде), как показано на рис. 8.1,б. Однако, хотя двоичные данные приведены в шестнадцатеричном коде, эта часть программы всегда рассматривается как заданная на машинном языке и оказывается трудной для понимания.

В более приемлемой форме записанная на машинном языке она могла бы выглядеть так:

1. Загрузить двоичное число (1011 0100) в аккумулятор.
2. Инвертировать каждый двоичный бит содержимого аккумулятора.
3. Поместить результаты инверсии в ячейку памяти данных 2100Н.

В этой части осуществляется перевод двоичного 8-разрядного числа в его эквивалент в инверсной форме.

Возникает вопрос: как перейти от этой формы человеческого языка, иногда длинной и сложной, к машинному языку? Ответ состоит в использовании языка простого программирования — от самого высокого уровня до машинного, представленного на рис. 8.1.

Ассемблер использует слова и фразы, преобразуя их в машинный код микропроцессора.

Обычно фраза или заданная величина на ассемблере будет соответствовать выражению длиной от одного до трех байт машинного языка. Суть и процедура ассемблирования показаны на рис. 8.2, где, например, вторая команда программы представлена единственной *мнемоникой* из трех букв CMA (инвертировать содержимое аккумулятора). Сначала три буквы переведены в их эквивалент в коде ASCII, затем три кода ASCII преобразованы в определенный

порядок специальной программой ассемблера, которая выдает код инверсии содержимого аккумулятора на машинном языке, т. е. $0010\ 1111_2$ в данном случае или 2FH. Мнемоника преобразована в один единственный байт машинного языка.

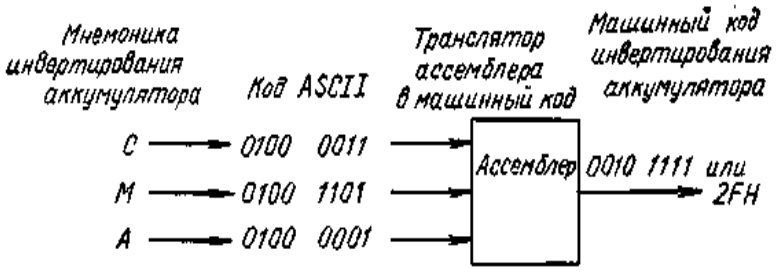

Рис. 8.2 Трансляция мнемоники ассемблера в машинный код программы

Программа на языке ассемблер, записанная человеком, могла бы быть представлена в виде табл. 8.1.

Обычным является деление объявлений на машинномязыке на четыре поля: метка; мнемоника; операнд и комментарий. *Поле метки* используется не всегда и в этой программе остается пустым. *Поле мнемоники* содержит точную мнемонику, установленную разработчиком, которая обычно указывает программе ассемблера операцию для выполнения. *Поле операнда* содержит информацию о регистрах, данных и адресах, объединенных соответствующей операцией. Используя информацию только полей мнемоники и операнда, ассемблер может выдать соответствующий код на машинном языке. Можно также назначить ячейки памяти списку команд. Поле комментариев не учитывается ассемблером и ограничивается его перепечаткой. Это поле очень важно, поскольку позволяет понять события в программе.

Таблица 8.1. Программа на языке ассемблер

Метка	Мнемоника	Операнд	Комментарии
	MVl *A,*	B4H	Загрузить в аккумулятор данные, Следующие непосредственно; B4H
	CMA	2100H	Инвертировать содержимое аккумулятора
	STA		Разместить содержимое аккумулятора
	HLT		Остановить МП

Как только программа составлена (см. табл. 8.1), она представляется затем в виде табл. 8.2. Таким образом, задача ассемблирования (или составление программы на ассемблере) состоит из этапов: 1) перевод мнемоники иоперандов на машинный язык; 2) назначение последовательно ячейки памяти каждому КОП и операнду. Переход от версии табл. 6.1 к ассемблированной версии табл. 8.2 может быть выполнен либо вручную, либо на машине при помощи специальной программы ассемблера.

Таблица 8 2. Программа в машинных кодах и на языке ассемблер

Адрес, Н-код	Содержимое, Н-код	Метка	Мнемоника	Операнд	Комментарий
2000	3E		MV1 *A,*	B4H	Загрузить аккумулятор данными, следующими непосредственно за КОП, B4H
2001	B4				
2002	2F		CMA		Инвертировать содержимое аккумулятора
2003	32		STA	2100H	Поместить содержимое аккумулятора в ячейку памяти 2100H
2004	00				
2005	21				
2006	76		HLT		Остановить МП

Программа, состоящая из символических команд (фрагмент показан в табл. 8.1), иногда называется *исходной программой,* а переведенная однажды па машинный язык — уже *объектной программой.*

Программирование на языке ассемблер является способом «очеловечивания» действий микропроцессора. Языки высокого уровня (БЕЙСИК, ФОРТРАН и т. д.) при их использовании делают программирование более удобным.

Система команд микропроцессора

Система команд микропроцессорного устройства служит для связи между микропроцессором, аппаратурой и программным обеспечением и представляет ту часть системы, которая видна программисту.

Все команды процессора можно разделить на несколько основных групп команд. Не смотря на большое число разновидностей МП и МК, на самом низком уровне системы их команд имеют много общего. Любая микропроцессорная система содержит следующие группы команд:

- команды передачи данных (копирования), копирующие информацию из одного места в другое;
- арифметические команды, производящие арифметические преобразования операндов;
- логические команды, позволяющие компьютеру производить анализ получаемой информации. Примерами могут служить сравнение, логические операции И, ИЛИ, НЕ, а так же анализ отдельных битов кода, их сброс и установка. Сдвиги двоичного кода влево и вправо. Операции сдвига используются, например, при выполнении умножения и деления чисел;
- команды ввода и вывода информации для обмена с внешними устройствами;

 команды управления, к которым следует отнести все виды переходов. Сюда же включают операции по управлению процессором.

Команды процессора обычно включают две части – операционную и адресную. Операционная часть (её называют кодом операции – КОП) указывает, какое действие необходимо выполнить с информацией. Адресная часть (адрес) указывает, где хранится используемая в операции информация и куда поместить результат. У некоторых команд управления работой процессора адресная часть может отсутствовать. Операционная часть команды имеется всегда. По количеству адресов, записываемых в команде, команды делятся на одно-, двух-, трех- и четырехадресные. Существует связь таких параметров процессора, как длина адресного пространства, адресность, разрядность. Увеличение разрядности позволяет увеличить адресность команды и длину адреса (т. е. объем памяти, доступной данной команде). Увеличение адресности, в свою очередь, приводит к повышению быстродействия обработки (за счет снижения числа требуемых команд).

Наибольшее количество адресов было в четырехадресных ЭВМ. В командах принята следующая последовательность указания частей – код операции – КОП, адрес первого операнда – А1, адрес второго операнда – А2, адрес результата – А3 и адрес следующей команды – А4. Такие команды применялись в первых ЭВМ (рис.8.5).

КОП – код операции	А1 – адрес первого операнда	А2 – адрес второго операнда	А3 – адрес результата	А4 – адрес следующей команды

Рис. 8.5. Типовая структура четырехадресной команды

В микропроцессорах большое количество адресов снижает производитель-ность. Поэтому, сначала отказались от использования адреса следующей команды, заменив его счетчиком команд. Затем результат начали помещать по адресу пер-вого операнда. Это привело к настоящему времени к использованию двух-, одно- и безадресных команд (рис 8.6).

а)

КОП – код операции	А1 – адрес первого операнда	А2 – адрес второго операнда

б)

КОП – код операции	А1 – адрес первого операнда

в)

КОП – код операции

а) двухадресная команда; б) одноадресная команда; в) безадресная команда.

Рис 8.6. Типовая структура современных команд

Таким образом, программирование в машинных командах требует знания системы команд конкретной ЭВМ и их адресности. При этом реализация даже довольно несложных вычислений требует разложения их на простые операции, что значительно увеличивает общий объем программы и затрудняет ее чтение и отладку.

Рассмотрим особенности команд используемых в универсальных микропроцессорах и микроконтроллерах.

Например, система команд одного из первых универсальных микропроцессоров i8080 состоит из 78 базовых команд, которые можно разделить на пять групп:

- команды передачи данных — используются для передачи данных из реги-стра в регистр, из памяти в регистр, из регистра в память;
- арифметические команды — используются для сложения, вычитания, инкре-мента или декремента содержимого регистров или ячейки памяти;
- логические команды: И, ИЛИ, исключающее ИЛИ, сравнение, сдвиги;
- команды переходов — используются для условных и безусловных перехо-дов, вызова подпрограмм и возврата из них;
- команды управления, ввода/вывода и работы со стеком — используются для управления прерыванием, регистром признаков, ввода и вывода информации.

В микропроцессоре принят формат информационного слова, представляющего собой 8-разрядное двоичное слово (байт). Формат информационного слова (данных):

7	6	5	4	3	2	1	0

где D7 — старший разряд слова, D0 — младший разряд.

Формат команды зависит от типа операции и может быть одно-, двух-, или трехбайтовым. Байты двух- и трехбайтовых команд должны храниться в ячейках памяти, следующих одна за другой. Адрес первого байта всегда является адресом кода операции.

Формат команд процессора:

Однобайтовая команда

Байт В1	D7	D6	D5	D4	D3	D2	D1	D0	Код операции

Двухбайтовая команда

Байт В1	D7	D6	D5	D4	D3	D2	D1	D0	Код операции
Байт В2	D7	D6	D5	D4	D3	D2	D1	D0	Данные или адрес

Трехбайтовая команда

Байт В1	D7	D6	D5	D4	D3	D2	D1	D0	Код операции
Байт В2	D7	D6	D5	D4	D3	D2	D1	D0	Данные или адрес
Байт В3	D7	D6	D5	D4	D3	D2	D1	D0	Данные или адрес

Как видно из формата команды универсального процессора, команды имеют фиксированный размер кода операции и переменную длину команды.