Тема 12. Організація переходів у програмах.

Команды операций ветвления

Команды ветвления составляют четвертую группу команд, которой снабжен типовой МП. Они приведены в табл. 12.1. Заметим, что описание относится к командам перехода. Действительно, термины *ветвление* и *переход* приводятся в этой главе как синонимы. Однако некоторые разработчики усматривают разницу между ними. Эти команды называют иногда *командами передачи управления.* Там, где они будут встречаться, мы все-таки их будем квалифицировать как команды перехода, следуя договору с фирмой Intel.

Обычно микро-ЭВМ выполняет команды последовательно. Шестнадцатиразрядный счетчик команд типового микропроцессора хранит всегда адрес следующей извлекаемой из памяти команды до ее выполнения. Содержимое его обычно повышается в каждом счете. Команды ветвления или перехода являются средством изменения значения содержимого счетчика команд и, следовательно, изменения нормальной последовательности выполнения программы. Эти команды разделены на две группы; *безусловного перехода* и *условного перехода.*

Первая команда в табл. 12.1 является командой безусловного перехода. Команда ПЕРЕЙТИ непосредственной адресации является трехбайтовой, используемой для изменения специфического адреса
в счетчике команд МП. На рис. 12.1 приведен пример использования такой команды безусловного перехода. Здесь адрес 2000H загружен в счетчик команд, информация о нем следует непосредственно за КОП, поэтому адресация называется непосредственной. Заметим на рис. 12.1, что младшая часть адреса находится во 2-м байт памяти,

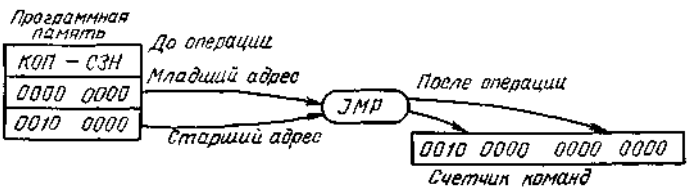


Рис. 12.1. Команда безусловного перехода

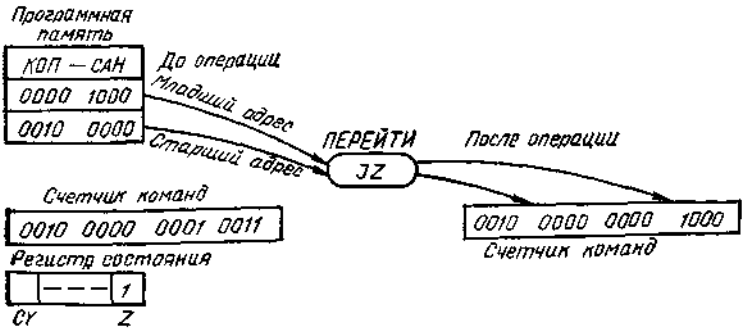
а старшая является содержимым 3-го байт памяти. Одна команда этого типа будет использована для запуска счетчика команд в момент начала выполнения новой программы. Мы можем рассматривать команду ветвления или безусловного перехода как способ загрузки новой информации об адресе в счетчик команд.

Таблица 12.1. Команды ветвления или перехода

Операция	Адресация	Мнемоника	код	Байты	Формат команды	Символика
Перейти в LOC	Непосредственная	JMP	C3	3	код мл.адрес ст. адрес	(PC) ← (адрес)
Перейти в LOC, если 0	»	JZ	CA	3	КОД мл.адрес ст. адрес	Если Z = 1, то (PC) ← (адрес)
Перейти в LOC, если не 0	»	JNZ	C2	3	КОД мл.адрес ст. адрес	Если Z = 0, то (PC) ← (адрес)
Перейти в LOC, если перенос	»	JC	DA	3	КОД мл.адрес ст. адрес	Если CY = 1, то (PC) ← (адрес)
Перейти в LOC, если нет переноса	»	JNC	D2	3	код мл.адрес ст. адрес	Если CY = 0, то (PC) ← (адрес)

Четыре последние команды ветвления в табл. 12.1 являются командами условного перехода. Эти команды повлекут за собой непосредственно загрузку адреса, только если будут выполнены специальные условия. В

противном случае счетчик команд будет инкрементирован нормально. На рис. 12.2 показан пример команды перейти, если 0. В этом случае счетчик команд 2013H до операции будет нормально инкрементирован, если только индикатор нуля не установлен в 1. Микропроцессор проверяет это и находит 1, значит, результатом последней логической или арифметической операции был 0. Условия перехода выполнены, и МП загружает новый адрес 2008H в счетчик команд. Этот новый адрес поступает из памяти программы; команда опроса использует непосредственную адресацию. Следующей выполняемой командой будет команда размещения данных в памяти по адресу 2008H (но не по адресу 2013H), как мы могли бы предположить при последовательном выполнении программы. Наш МП перешел к новой ячейке памяти программы. Отметим, что пример на рис. 12.2 показывает переход назад, что случается наиболее часто.



Рнс. 12.2. Команда перехода, если нулевой результат

Команды перехода или ветвления существуют практически во всех программах МП. Они очень эффективны как средство принятия решений и удобны для формирования циклов программы.

Ветвление программ

Рассмотрим простую задачу сравнения двух чисел. Нужно найти наибольшее и поместить его в определенную ячейку памяти (см. структурную схему решения этой задачн на рис. 12.3). В двух первых прямоугольных кадрах представлена операция загрузки двух чисел в регистры А (аккумулятор) и (данные). Следующий прямоугольный кадр соответствует операции сравнения, где содержимое регистра вычитается из содержимого регистра А. Команды сравнения

не разрушают содержимого регистров, но влияют на индикатор.

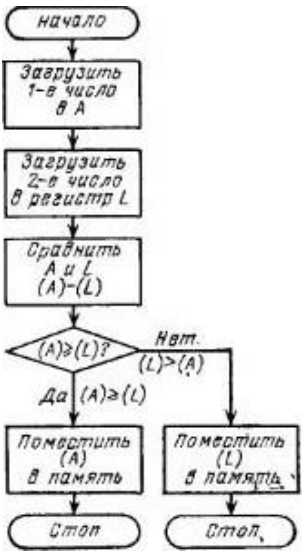


Рис.12.3Структурная схема программы сравнения двух чисел и помещения в память

Следующий кадр называется знаком принятия решения. Он завершает в программе этап решения и содержит поставленный вопрос [(A)≥(L)?]. Если ответ на этот вопрос Да, программа продолжается последовательно и содержимое регистра А помещается н память, затем процессор останавливается. Если Нег, программа ответвляется вправо и содержимое регистра помещается в память, затем МП

останавливается, Использование знаков принятия решения приводит в программах к тому, что называется внутренним ветвлением программы.

В табл. 12.2 представлена на ассемблере задача, которую мы только что описали и в которой наибольшее число помещено в память 2040H. Заметим, что здесь аккумулятор A загружен числом 15H, а регистр L числом 6H. Сравнивая (третья команда в этом примере), находим, что (A)>(L), и, следовательно, индикатор переноса (CY) сброшен в 0. Команда перехода (мнемоника JC в примере) проверяет индикатор переноса находит (CY)=0. Микропроцессор обращается тогда к следующей последовательной команде ПОМЕСТИТЬ (A) в память по адресу 2040H и наибольшее число помещено командой STA в память по адресу 2040H. Затем МП останавливается (команда HLT). Три последние команды не были использованы в этом примере.

Таблица 12.2 Ветвящаяся программа на ассемблере

Метка	Мнемоника	Операнд	Комментарий
	MVI	*A*, OFH	Загрузить первое число (15_{10}) в аккумулятор
	MVI	*L*, 06H	Загрузить второе число (6_{10}) в регистр *L*
	CMP	*L*	Сравнить (A) и (L). Индикатор *CY*=1, если (A)<(L)
	JC	STORE *L*	Перейти к STORE *L*, если *CY*=1 (если A<*L*); если нет, продолжать последовательно
	STA	2040H	Поместить (A) в ячейку памяти 2040H
	HLT		Остановить МП после помещения (A)
STORE *L*	MOV	*A*, *L*	Передать (L) в аккумулятор
	STA	2040H	Поместить (A) в ячейку памяти 2040H
	HLT		Остановить МП после помещения

Если обратиться к рис. 12.3, мы установим, что программа прошла последовательно.

Рассмотрим теперь выполнение такой же программы, но в случае, когда содержимое аккумулятора меньше содержимого регистра L. Программа остается та же (табл.12.2). Содержимое аккумулятора — 0AH (10_{10}), содержимое регистра L — 0EH (14_{10}) здесь больше, Команда сравнения устанавливает индикатор CY в 1, так как (A)<(L). Команда ПЕРЕЙТИ, если индикатор переноса установлен в 1 (JC здесь), проверяет индикатор CY и определяет, установлен ли он, При этом изменяется содержимое счетчика команд, т. е. загружается адрес следующей выполняемой команды (STORE

L, в нашем примере). Заметим, что МП перешел через две команды для того, чтобы обратиться туда, где находится символический адрес STORE L. Микропроцессор выполняет тогда три последние команды, которые помещают содержимое регистра L, т. е. наибольшее число 0EH в ячейку памяти по адресу 2040H. Обратившись к схеме на рис. 12.3, мы установим, что программа выполнена последовательно до знака принятия решения, затем ответвилась вправо, заканчиваясь оператором Стоп.

Рассмотрим снова три последние команды в табл. 12.1. Целью этих команд является размещение содержимого регистра L в память по адресу 2040H. Наш макропроцессор не позволяет поместить содержимое регистра L прямо в память. Нужно последовательно передать сначала содержимое L в A, затем поместить содержимое A в память. Таким образом, часто оказывается, что МП с ограниченным составом команд (как в случае типового МП) использовать сложнее, чем устройства, более мощные по своим возможностям.

Таким образом, рассмотренный на структурной схеме способ ветвления осуществляется командами перехода (или ветвления), согласно которым принимаются

решения, основанные на состоянии специальных индикаторов. Мы встретились здесь с использованием символического адреса (метки) при команде перехода. Ветвление очень широко используется при программировании.

Циклы

Микропроцессоры особенно эффективны в случае выполнения повторяющихся задач. Структурная схема на рис. 6.44, например, представляет программу, которая будет производить счет от 0 до 254 (0—FEH) и выводить результат счета на печать. Исходя из вершины, она является обычной для установки регистров. В этом примере содержимым аккумулятора *A* является 00H. Затем содержимое аккумулятора выводится на избранную периферию: это действие является повторяющимся процессом. Затем содержимое аккумулятора инкрементируется или изменяется. Знак сравнения и принятия решения выполняет процедуру проверки, чтобы определить, содержит ли аккумулятор значение FFH. Если ответ на поставленный в знаке принятия решения вопрос (будет ли A=FFH?) отрицателен (*Нет*), программа ветвится влево от знака принятия решения и возвращается в блок введения. Это составляет цикл. Программа будет продолжаться, повторяясь 254_{10} раз.

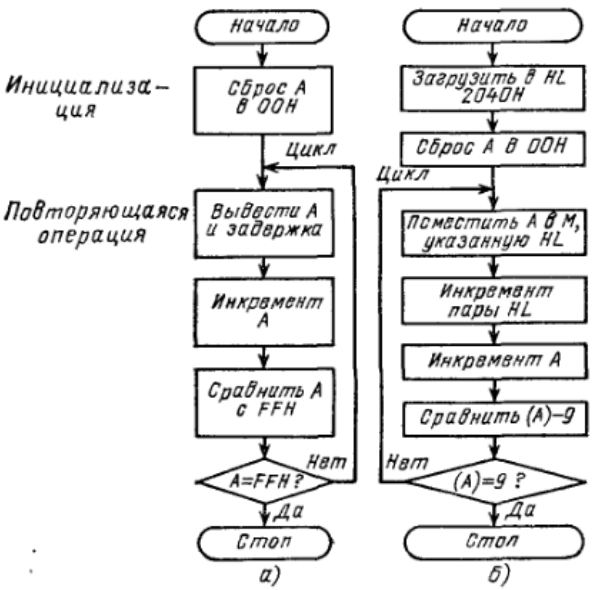


Рис. 6.44. Структурная схема последовательного счета с использованием циклов (*а*) и ее развитие (*б*)

Структурная схема на рис. 6.44, *а* представляет собой программу, которая может содержать только 20 или 30 команд. Если бы 255_{10} пропусков через программу были запрограммированы последовательно, их список составил бы тысячи команд. Циклы в программе являются очень эффективным методом ее сокращения.

На рис. 6.44, *б* приведена другая циклическая программа размещения ряда чисел (от 0 до 8) последовательно в память с адресами от 2040H до 2048H. Две первые прямоугольные рамки соответствуют начальной загрузке пары регистров *HL* и аккумулятора *A* значениями 2040H и 00H соответственно. Третья рамка соответствует процессу размещения данных в памяти, который будет повторен 9 раз в ходе выполнения этой программы. Микропроцессор, повторяя свои действия, разместит содержимое аккумулятора в памяти по адресу, указанному парой *HL*. При первом прохождении цикла содержимое (00H) аккумулятора будет помещено в память по адресу 2040H.

Четвертый и пятый кадры представляют операции, которые изменяют адрес в паре *HL* и счет в регистре *A*. Например, в ходе первого прохождения пара *HL* инкрементируется до 2041H, а аккумулятор — до 01H.

Прямоугольник сравнения и знак принятия решения составляют операцию тестирования. Команда сравнения вычитает 09H из содержимого *A* для восстановления или сброса индикатора нуля. Если (A)<9, индикатор нуля будет сброшен, если (A)=9, индикатор нуля будет установлен в 1. Знак принятия решения опрашивает: Равно ли (A) девяти? Если ответ *Да*, программа выходит из цикла и заканчивается, если, однако, ответ *Нет*, ветвится снова к команде размещения по соседству с вершиной структурной схемы. Команды сравнения и условного перехода используются для проверки изменяющегося счета для определения момента выхода из цикла. Программа, показанная на рис. 6.44, *б*, повторит цикл 9 раз прежде, чем выйти на знак *Стоп*.

В табл. 6.12 приведена версия программы на ассемблере, соответствующая структурной схеме на рис. 6.44, *б*.

Эта программа последовательно поместит числа от 0 до 9 в память по адресам от 2040H до 2048H. Две первые команды эквивалентны двум первым кадрам схемы и предназначены для установки в паре *HL* и в аккумуляторе начальных значений 2040H и 00H соответственно. Команда MOV *M*, *A* помещает содержимое аккумулятора в ячейку

Таблица 6.12. *Версия на ассемблере программы, соответствующей рис. 6.44, б*

Метка	Мнемоника	Операнд	Комментарий
	LXI	*H*, 2040H	Загрузить 2040H в пару *HL* (указатель адреса)
	XRA	*A*	Сброс аккумулятора в 00H, т. е. (A)⊕(A)=00H в аккумуляторе
LOOP	MOV	*M*, *A*	Поместить содержимое аккумулятора ячейку памяти, указанную парой *HL*
	INX	*H*	Инкрементировать пару *HL*
	INR	*A*	Инкрементировать аккумулятор
	CPI	09H	Сравнить (A)=09H? Если (A)=09H, индикатор нуля установлен в 1
	JNZ	LOOP	Перейти к LOOP, если *Z*==0, т. е. если (A)=09H; если нет, продолжать последовательно
	HLT		Остановить МП

памяти, указанную парой *HL*. Две следующие команды (INX *H* и INR *A*) инкрементируют пару *HL* и *A*. Команда CPI сравнивает содержимое *A* с константой 09H. Если *A* содержит значение между 0 и 8, индикатор нуля сброшен в 0, но если *A* содержит 9, индикатор нуля устанавливается в 1.

Команда JNZ проверяет индикатор нуля. Если индикатор нуля сброшен в 0, значит результат вычитания (A) — 09H не равен 0, следовательно, осуществляется переход по символическому адресу LOOP. Если индикатор нуля установлен в 1, результат вычитания (A) — 09H нулевой; при этих условиях программа выходит из цикла и выполняет следующую последовательную команду (HLT), по которой завершается выполнение программы.