Тема 13. Робота з підпрограмами і команди решти операцій.

Во многих программах часто приходится сталкиваться с необходимостью в нескольких ее местах решать одну и ту же задачу. В этом случае обычно пишут подпрограмму (п/п), решающую эту задачу, и обращаются к ней по мере необходимости, не повторяя одни и те же команды в разных точках программы. При использовании п/п необходимо, нарушить обычную последовательность выполнения команд, перейти к п/п, а также обеспечить запоминание адреса возврата в основную программу. Для запоминания адреса возврата в основную программу и для других целей в области оперативного запоминающего устройства (ОЗУ) отводится так называемая стековая область, или стек. Стек - это совокупность регистров, которые принимают и выдают информацию по принципу "последним записан - первым считан" (*last input first Output-LIFO*), т.е. доступ к стеку возможен только с одного конца. Для приема и хранения адреса ячейки стека, к которой было последнее обращение, предназначен шестнадцатиразрядный указатель стека. Содержимое указателя стека уменьшается на единицу при поступлении данных в стек и соответственно увеличивается при выборке из него.

Состав команд операций вызова подпрограмм и возврата в основную программу

Эти команды составляют пятую категорию состава команд типового МП. Их только две, и они приведены в табл. 13.1, Команды вызова (CALL) и возврата (RET) всегда используются парами. При их выполнении

индикаторы не изменяются.

Трехбайтовая команда CALL используется основной программой для перехода МП (или ветвления) к подпрограмме. В примере на рис. 13.1 подпрограмма является короткой последовательностью команд, целью которой является создание интервалов времени в течение 1 с. Когда МП передает первую команду CALL по адресу 1000H, он находит адрес перехода в двух следующих байтах программы. Адрес следующей команды за CALL отправляется в стек (не показанный на рис. 13.1), и МП переходит тогда в начало подпрограммы по адресу 1000H. Команды, которые составляют эту программу счета времени, выполняются, пока МП не передаст команду возврата (RET).

Таблица 13.1. Команды вызова подпрограмм и возврата типового МП

Операция	Адресация	Мнемоника	коп	Байты	формат команды	Символика
Вызов подпрограммы	Непосредственная, косвенная регистровая	CALL	CD	3	КОП мл.адрес ст. адрес	((SP) - 1) ← (PCH) ((SP) - 2) ← (PCL) (SP) ← (SP - 2) (PC) ← (адрес)
Возврат из подпрограммы	Косвенная регистровая	RET	C9	1	КОП	(PCL) ← ((SP)) (PCH) ← ((SP) + 1) (SP) ← (SP) + 2

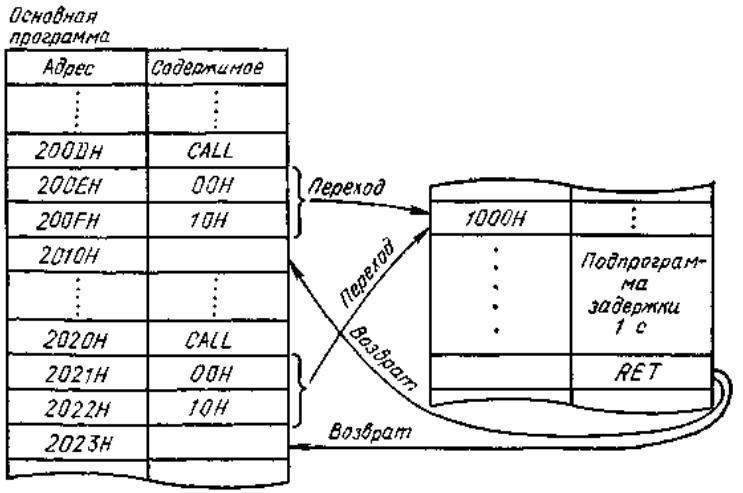

Рис. 13.1. Взаимодействие основной программы и подпрограммы при командах CALL (ВЫЗОВ) и RET (ВОЗВРАТ)

Сохраняющийся в стеке адрес (2010H) отыскивается счетчиком команд, и МП продолжает выполнение основной программы, принимая ее там, где он ее покинул. Это нормальное выполнение продолжается до тех пор, пока МП не встретит другую команду вызова по адресу 2020H. Микропроцессор сохраняет адрес следующей команды (2023H) в стеке и переходит на подпрограмму, начинающуюся адресом 1000H. После завершения выполнения этой подпрограммы команда возврата извлекает из стека адрес следующей команды основной программы и загружает его в счетчик команд. Данная подпрограмма может быть использована много раз в ходе выполнения одной и той же основной программы. Подпрограмма может быть расположена в ОЗУ или ПЗУ.

Команда вызова сочетает функции операций загрузки в стек и перехода. Использование ее показано на

рис. 13.2. Сначала она загружает в стек содержимое счетчика команд. Затем счетчик команд должен быть загружен новым адресом для выполнения перехода в подпрограмму.

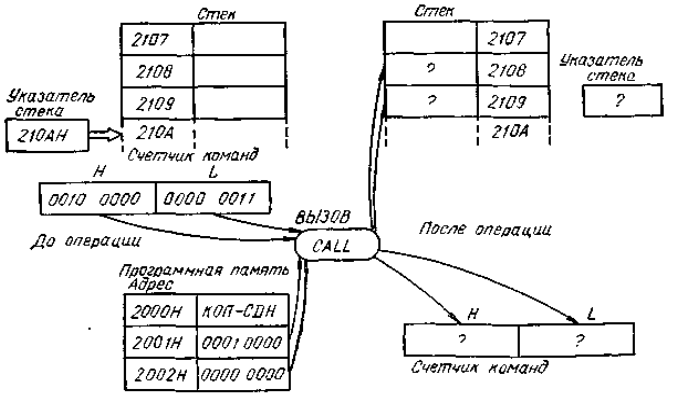

Рис. 13.2. Команда вызова подпрограммы

1. Указатель стека декрементирован от 210AH до 2109H

2. Старший байт счетчика команд загружается в стек по адресу 2109H.

3. Указатель стека декрементирован от 2109H до 2108H.

4. Младший байт счетчика команд загружается в ячейку памяти по адресу 2108H.

5. Младший байт адреса (второй байт памяти) загружается в младший байт счетчика команд.

6. Старший байт адреса (третий байт памяти) загружается в старший байт счетчика команд.

Здесь МП ответвляется по адресу, на который указывает счетчик команд (в приведенном примере 1000H), являющемуся адресом начала подпрограммы.

Рассмотрим теперь пример применения команды

возврата в основную программу RET (рис. 13.3). По командевозврата содержимое стека должно быть передано в счетчик команд. Проследим последовательность выполнения команды по номерам, взятым на рис. 13.3 в кружки.

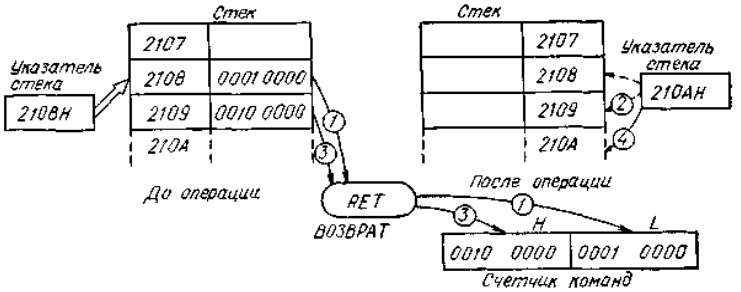

Рис. 13.3. Команда возврата из подпрограммы

1. Вершина стека (адрес 2108H) извлекается, ее содержимое передается в младший байт счетчика команд.
2. Указатель стека инкрементируется от 2108H до 2109H.
3. Вершина стека (теперь ее адрес 2109H) извлекается, ее содержимое передается в старший байт счетчика команд.
4. Указатель стека инкрементируется от 2109H до 210AH.

Теперь счетчик команд содержит 16-разрядный адрес (2010H) следующей извлекаемой из памяти команды.

Рассмотрим программу которая преднозначена для образования циклов неопределенно долго и принимает новые наборы данных в каждом цикле. Она покидает цикл для процедуры аварийной сигнализации только в случае возникновения соответствующей ситуации. В этом случае

оператор или другая программа выдает процедуру отработки аварийного состояния. Это показано на структурной схеме кружком.

На рис. 13.4,а приведен пример функциональной структурной схемы. В задачу МП входит ввести два отобранных числа, образовать их сумму и сохранить ее. Этой процедуре соответствуют три кадра в начале схемы. Затем сумма должна быть умножена на масштабный коэффициент и результат размещается в памяти. Это связано с тем, что сумма (в третьем кадре) должна быть восстановлена в аккумуляторе. Затем она проверяется знаком принятия решения. В том случае, если сумма равна или больше 10Н (16_{10}), программа выводится на сигнализацию тревоги, если меньше 10Н циклится и снова вводит выборку чисел. Такой тип программ мог бы быть использован в промышленных устройствах, когда сумма двух входов должна постоянно контролироваться; быть подвержена масштабированию, а затем размещена в памяти; контролироваться по крайней мере один раз в секунду для определения, не приняла ли она опасного значения, и если да, то программа должна выдать аварийный сигнал АС.

На рис. 13.4,б представлена схема программы. Каждый кадр ее эквивалентен одной операции МП. Выделенная рамкой ячейка, представляет собой группу команд, называемую подпрограммой умножить. В табл. 13.1 представлена программа на ассемблере к рис. 13.4, б. Этот список привлекает только команды, которые составляют часть состава команд нашего типового МП (список, относящийся к подпрограммам умножения, на рисунке не приведен).

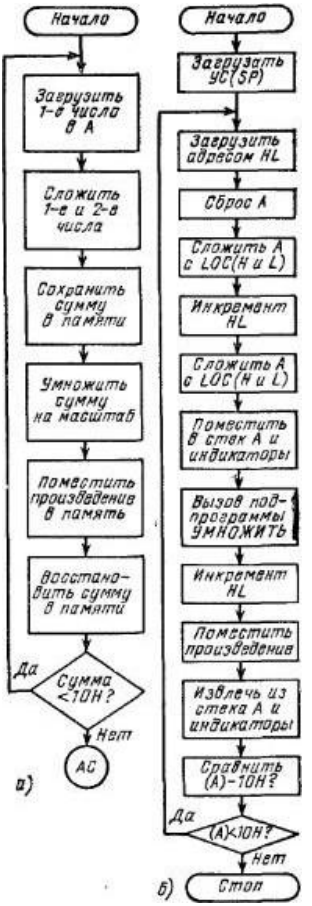

Рис. 13.4. Структурная схема программы контроля оборудования (а) и ее развитие (б)

Таблица 13.1. Праграмма на ассемблере к рис.13.4,б

Метка	Мнемоника	Операнд	Комментарий
	LXI	*P*, 20C0H	Поместить 20C0H в указатель стека
START	LXI	*H*, 2040H	Поместить 2040H в пару *HL* (указатель адреса основной программы)
	XPA	*A*	Сброс аккумулятора в 00H
	ADD	*M*	Сложить (A) с содержнмым ячейки памяти 2040H (сложнть первое слагаемое с содержимым ячейкн памяти 2040H)
	INX	*H*	Инкрементировать пару *HL* до 2041H
	ADD	*M*	Сложить *A* с содержнмым ячейки памяти 2041H (сложить второе число с содержимым ячейкн памяти 2041H)
	PUSN	*PSW*	Поместнть в стек (A) и индикаторы
	CALL	MULTIPLY	Вызов подпрограммы MULTIPLY в ячейку памятн 2050H
	INX	*H*	Инкрементнровать пару *HL* до 2042H
	MOV	*M*, *A*	Поместить произведение в ячейку памяти 2042H
	POP	*PSW*	Извлечь нз стека и восстановить *A* и нндикаторы
	CPI	10H	Сравнить (A) и 10H, т. е. (A) — 10H. Если (A) < 10H, *CY* = 1; если нет, *CY* = 0
	JC	START	Перейти к START (ячейка памяти 2003H), если *CY* = 1; если нет, продолжать последовательно
	HLT		Остановить МП

На рис. 13.4,б представлена схема программы. Каждый кадр ее эквивалентен одной операции МП. Выделенная рамкой ячейка, представляет собой группу команд, называемую подпрограммой умножить.

В табл. 13.1 представлена программа на ассемблере к рис. 13.4, б. Этот список привлекает только команды, которые составляют часть состава команд нашего типового МП (список, относящийся к подпрограммам умножения, на рисунке не приведен).

Три первые команды (табл. 13.1) восстанавливают указатель стека, пару HL и аккумулятор A соответственно до 20C0H, 2040H и 00H. Четвертая складывает содержимое (A) (т. е. 00H) с содержимым ячейки памяти, на которую указывает пара HL (адрес 2040H). Обратимся к рис. 13.5, который представляет собой основную программу, данные, подпрограмму память стека, использованные в этом примере. Заметим, что содержимым памяти по адресу 2040H, сложенным с (A) является 05H. После выполнения четвертой команды содержимым (A) является 05H (00H + 05H=05H).

Пятая команда основной программы инкрементирует пару HL. Шестая складывает содержимое памяти по адресу 2041H с (A). Эта ячейка памяти содержит 09H (см, рис.13.5), A содержит 05H, сумма в A после второго ADDM–0EH является суммой двух выбранных чисел.

Следующей командой основной программы является PUSH PSW, которая сохраняет содержимое аккумулятора и индикатора. Это должно быть выполнено, потому что следующая команда вызова разрушит содержимое этих регистров, выполняя подпрограмму умножения. Операция вызова помещает адрес команды в последовательности основной программы (в нашем примере 200EH) в стек, затем переходит к адресу первой команды подпрограммы (адрес 2050H).

Этот вызов показан на рис. 13.4, б. Теперь рассмотрим команду вызова как команду, которая просто умножает содержимое A на два. Подпрограмма в нашем примере

осуществляет умножение 0EHx2=1CH, и произведение, помешенное в аккумулятор к моменту возврата, будет lCH.

После возврата в основную программу адресом следующей последовательной команды становится 200EH. Команда INX H инкрементирует пару HL, и произведение, выполненное подпрограммой, помещается теперь в памяти по адресу 2042H командой MOV M, A.

Вспомним, что команды помещения в стек и извлечения из него парные. Только что мы поместили в стек содержимое аккумулятора и индикаторов командой PUSH PSW, мы их восстановим теперь командой POP PSW. Содержимое аккумулятора является не произведением, а суммой 0EH в нашем примере). Эта сумма будет тестирована последующей командой.

Важно отметить, что (0EH) отправлена в подпрограмму для обработки. Напротив, именно произведение было передано подпрограммой и, следовательно, размещено в памяти. На жаргоне информатики эта операция составляет прохождение параметра. В данном случае использован именно регистр A, но в других случаях это могло бы быть выполнено посредством памяти.

Две следующие команды основной программы предназначены для тестирования суммы в аккумуляторе. Команда сравнить непосредственно выполняет операцию вычитания 0EH—10H=FEH с использованием дополнительного кода числа —210. Содержимое A меньше чем 10H индикатор переноса CY установлен в 1. Команда ПЕРЕЙТИ, если перенос, проверяет индикатор, он установлен в 1, н МП переходит к символическому адресу START, который является не чем иным, как началом цикла команды LX1 H (адресация 2003H, см, рис. 13.5).

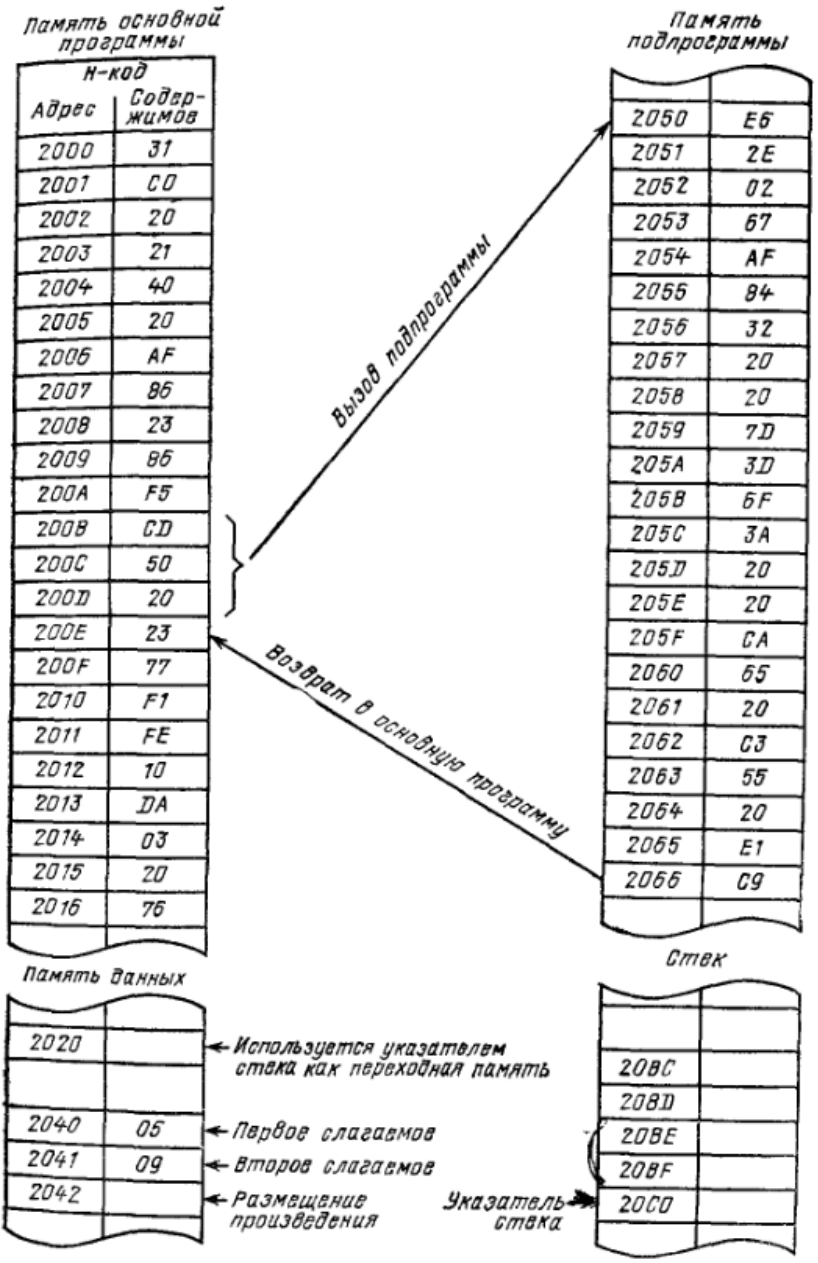

Рис.13.5 Воображаемая память программы, приведенной в табл. 13.1.

Для операции двоичного умножения могут быть использованы различные способы на базе команд типового МП. Вытекающие непосредственно из правил умножения методы приводят к повторяющемуся сложению. Рассмотрим типичное решение на примере выполнения операции $5x3=15_{10}$.

Множимое Множитель Произведение

$5 \quad x \quad 3 \quad = \quad 15_{10}$

При повторяющемся сложении эта операция выполнится иначе:

Множитель = 3 Произведение

$5+5+5= \quad 15_{10}$

Множимое 5

В этом случае множимое повторяется слагаемым число раз, равное множителю, а результатом этой операции будет произведение. В нашем примере подпрограмма эффективно выполняет операцию сложения (вместо 0Ex2=1CH). Множимое складывается с самим собой (0E+0E=lCH) , что дает искомое произведение.

На рис. 13.6 приведена подробная структурная схема подпрограммы умножения, и каждый ее шаг соответствует одной команде программы на ассемблере, приведенной в табл. 13.2. Первый шаг на рис.13.6 обеспечивает хранение текущего содержимого пары HL операцией помещения в стек. Это характерно для многих подпрограмм — помещать в содержимое регистров МП, потому что они могут быть использованы во время выполнения подпрограммы, что изменит их содержимое. Совершенно очевидно, что содержимое пары HL будет восстановлено в конце программы командой извлечения из стека.

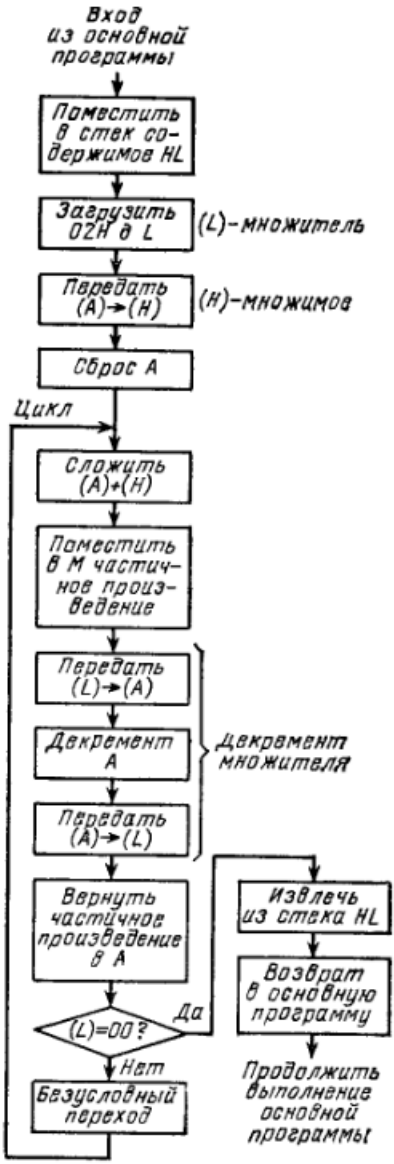

Рис. 13.6. Структурная схема подпрограммы умножения в задаче, приведенной на рис. 13.4.

Три следующих шага восстанавливают регистры L, H и A. Регистр L - будет содержать множитель (здесь 02H) и он будет декрементирован до 00H в ходе программы умножения. Регистр H будет содержать множимое (здесь 0EH). т. е. сумму, посланную в подпрограмму основной программой. Аккумулятор здесь сбрасывается в 00H.

Пятый шаг соответствует сложению множимого (здесь 0EH) с A). Частичное произведение 0EH затем сохраняется во временной памяти по адресу 2020H, тогда как аккумулятор используется для декрементирования содержнмого L. Один раз множитель декрементируется от 02H до 01H и помещается в L. Затем частичное произведение восстанавливается в A.

На шаге принятия решения возникает вопрос: (L)=00? Если ответ отрицательный, программа переходит снова в цикл, если положительный — ветвится вправо. Содержимое пары HL восстанавливается командой извлечения из стека, и, наконец, команда возврата возвращает нас в основную программу. Согласно рис. 13.5 возврат осуществляется по адресу 200EH основной программы. В случае, когда множителем является 02H, программа дважды пройдет в последовательности сложить–поместить–передать–декрементировать —передать—загрузить перед возвратом в основную программу. Отметим, что окончательное произведение 1CH поступит в основную программу, оставаясь в аккумуляторе.

В табл. 13.2 приведена программа на ассемблере. Каждая команда соответствует одному шагу структурной схемы на рис. 13.6. Рекомендуем читателю попытаться проследить поток данных сначала по ней, а затем по программе, приведенной в табл. 13.2.

Таблица 13.2. Программа на ассемблере, соответствующая рис.13.6.

Метка	Мнемоника	Операнд	Комментарий
	PUSH	*H*	Поместить в стек содержимое регистров *H* и *L* для сохранения содержимого пары *HL*
	MVI	*L*, 02H	Поместнть 02H (масштабный коэффнциент) в *L*; 02H — множнтель
	MOV	*H*, *A*	Передать (A)→(H); содержимое *H* — множнмое
	XRA	*A*	Сброс *A*
LOOP	ADD	*H*	Сложить (H)+(A); сумму поместить в *A*
	STA	2020H	Поместить (A) в ячейку памяти 2020H (временная память)
	MOV	*A*, *L*	Передать (*L*) в *A*
	DCR	*A*	Декрементировать содержнмое *A*
	MOV	*L*, *A*	Передать (A) в *L*
	LDA	2020H	Поместить содержимое ячейки памятн 2020H (временной памяти) в *A* (восстановленне *A* из ячейкн памяти 2020H)
	JZ	D0NE	Перейти к DONE (ячейка памятн 2065H), еслн *Z*=1, т. е. декрементировано до 00H; еслн нет — продолжать последовательно
	JMP	LOOP	Перейтн (всегда) к LOOP (адрес 2055H)
DONE	POP	*H*	Извлечь нз стека содержимое *HL*
	RET		Возврат в основную программу

Используя команды вызова и возврата, следует быть очень внимательным и использовать их парами. Нужно следить за тем, чтобы требуемые регистры были правильно инициализированы, многократно проверять состояние указателя стека. Следует также убедиться, что парно используются команды загрузки и извлечения из стека, а также глубоко осознать способы обмена параметрами между подпрограммами и основными программами.

Пример Светодиодный индикатор соединен с выходным портом 2 (рис.14.4). Светодиод загорается, если $в_0$ = = 1. Написать программу включения включения и выключения индикатора. Частота включений не имеет принципиального значения.

Можно заставить индикатор зажигаться и гаснуть, формируя на выходе $в_0$ выходного порта 2 сигналы соответственно 1 и 0 . Занесем в аккумулятор (А) число 01Н и по команде *OUT 02H* передадим его в выходной порт 2. После заданной временной задержки сделаем содержимое А равным 00Н. Если снова воспользуемся командой *OUT 02H,* то индикатор погаснет. После заданной временной выдержки снова зашлем в аккумулятор *A01H* число и опять отправим его содержимое в выходной
порт 2.

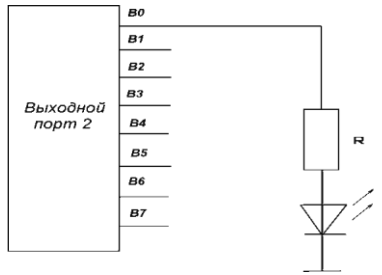

Рисунок 14.4- Схема включения светодиода

Для получения временной задержки можно использовать подпрограмму DELAY, описанную в примере 29. Но поскольку

период 110 мкс слишком мал для различения сигнала светодиода, необходимо сформировать большую задержку, взяв соответственно большое значение *x.* Составляем блок-схему решения задачи (рис. 14.5).

Назовем данную подпрограмму *ALARM,* используя это символическое имя в качестве метки адреса первой команды.

Подпрограмма *DELAY* вызывается по команде *CALL DELAY* (блок 4) и используется для того, чтобы светодиод горел в течение определенного времени. Содержимое *A* изменяется (блок 5) с помощью операции ИСКЛЮЧАЮЩЕЕ ИЛИ и операнда 00000001B=01H.

Рисунок 14.5- Блок-схема алгоритма решения задачи

Подпрограмма:

Адрес	Ассемблерный код	Комментарий
	ORG 0000H	; Первую команду поместить по адресу *0000H*
00	*ALARM: MVI A,O1H*	; Загрузить в *A 01H*
01		;
02	*FLASH: OUT 02H*	; Содержимое *A* выдать
03		; в порт 2

```
04      CALL DELAY      ; Задержка
05                      ;
06                      ;
07      XRI   01H       ; Изменить значение в0
08                      ; в A
09      JMP FLASH       ; Переход
0A                      ;
0B                      ;
```