Тема 5. Організація роботи мікропроцесорної системи.

Обмен информацией в микропроцессорах и микропроцессорных системах

Обмен информацией в микропроцессорных системах происходит в циклах обмена информацией. Под циклом обмена информацией понимается временной интервал, в течение которого происходит выполнение одной элементарной операции обмена по шине. Например, пересылка данных из процессора в память или же пересылка данных из устройства ввода/вывода в процессор. В пределах одного цикла также может передаваться и несколько кодов данных.

Циклы обмена информацией делятся на два основных типа:

• Цикл записи (вывода), в котором процессор записывает (выводит) информацию;

• Цикл чтения (ввода), в котором процессор читает (вводит) информацию.

Во время каждого цикла устройства, участвующие в обмене информацией, передают друг другу информационные и управляющие сигналы в строго установленном порядке или, как еще говорят, в соответствии с принятым протоколом обмена информацией.

Длительность цикла обмена может быть постоянной или переменной, но она всегда включает в себя несколько периодов сигнала тактовой частоты системы. То есть даже в идеальном случае частота чтения информации процессором и частота записи

информации оказываются в несколько раз меньше тактовой частоты системы.

Обмен информацией происходит с использованием системной магистрали (системной шины, интерфейса) микропроцессорной системы. Системный интерфейс содержит три основные информационные шины: адреса, данных и управления. Шина данных — это основная шина по которой передается информация. Количество ее разрядов (линий связи) определяет скорость и эффективность обмена, а также максимально возможное количество команд. Шина данных всегда двунаправленная, так как предполагает передачу информации в обоих направлениях. Наиболее часто встречающийся тип выходного каскада для линий этой шины — выход с тремя состояниями. Обычно шина данных имеет 8, 16, 32 или 64 разряда. За один цикл обмена по 64-разрядной шине может передаваться 8 байт информации, а по 8-разрядной — только один байт. Разрядность шины данных определяет и разрядность всей магистрали. Например, когда говорят о 32-разрядном системном интерфейсе, подразумевается, что он имеет 32-разрядную шину данных.

Шина адреса — вторая по важности шина, которая определяет максимально возможную сложность микропроцессорной системы, то есть допустимый объем памяти и, следовательно, максимально возможный размер программы и максимально возможный объем запоминаемых данных. Количество адресов, обеспечиваемых шиной адреса, определяется как 2N, где N — количество разрядов. Например, 16-разрядная шина адреса

обеспечивает 65536 адресов. Разрядность шины адреса обычно кратна 4 и достигает 64 разрядов.

Шина адреса может быть однонаправленной (когда магистралью всегда управляет только процессор) или двунаправленной (когда процессор может временно передавать управление магистралью другому устройству). Наиболее часто используются типы выходных каскадов с тремя состояниями или обычные.

Для снижения общего количества линий связи процессора применяется мультиплексирование шин адреса и данных. То есть одни и те же линии связи используются в разные моменты времени для передачи как адреса, так и данных (сначала — адрес, затем — данные). Для фиксации этих моментов (стробирования) служат специальные сигналы на шине управления. Мультиплексированная шина адреса/данных обеспечивает меньшую скорость обмена, требует более длительного цикла обмена.

Шина управления — это вспомогательная шина, управляющие сигналы на которой определяют тип текущего цикла и фиксируют моменты времени, соответствующие разным частям или стадиям цикла. Кроме того, управляющие сигналы обеспечивают согласование работы процессора с работой памяти или устройства ввода/вывода. Управляющие сигналы также обслуживают запрос и предоставление прерываний, запрос и предоставление прямого доступа к памяти.

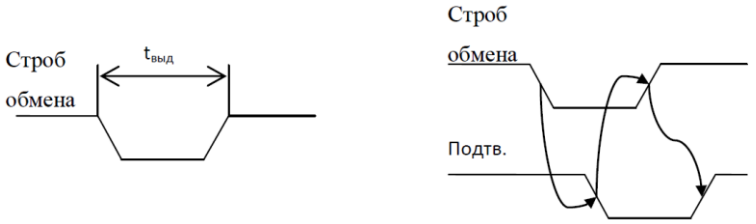
Линии шины управления могут быть как однонаправленными, так и двунаправленными.

Основные управляющие сигналы — это стробы обмена, то есть сигналы, формируемые процессором и определяющие моменты времени, в которые производится пересылка данных по шине данных, обмен данными. Чаще всего в магистрали используются два различных строба обмена:

• Строб записи (вывода), который определяет момент времени, когда устройство-исполнитель может принимать данные, выставленные процессором на шину данных;

• Строб чтения (ввода), который определяет момент времени, когда устройство-исполнитель должно выдать на шину данных код данных, который будет прочитан процессором.

При этом большое значение имеет то, как процессор заканчивает обмен в пределах цикла, в какой момент он снимает свой строб обмена. Возможны два пути решения (рис. 5.1):



а) синхронный обмен б) асинхронный обмен

Рис. 5.1. Синхронный обмен и асинхронный обмен

• при синхронном обмене процессор заканчивает обмен данными самостоятельно, через раз и навсегда установленный временной интервал выдержки ($t_{выд}$), то есть без учета реального быстродействия остальных устройств участвующих в обмене;

• при асинхронном обмене процессор заканчивает обмен только тогда, когда устройство-исполнитель подтверждает выполнение операции специальным сигналом ответным сигналом.

Достоинства синхронного обмена — более простой протокол обмена, меньшее количество управляющих сигналов. Недостатки — отсутствие гарантии, что исполнитель выполнил требуемую операцию, а также высокие требования к быстродействию исполнителя.

Достоинства асинхронного обмена — более надежная пересылка данных, возможность работы с самыми разными по быстродействию исполнителями.

Недостаток — необходимость формирования сигнала подтверждения всеми исполнителями, то есть дополнительные аппаратурные затраты.

Синхронный и асинхронный типы обмена обеспечивают разную производительность работы микропроцессорной системы. С одной стороны, при асинхронном обмене требуется какое-то время на выработку, передачу дополнительного сигнала и на его обработку процессором. С другой стороны, при синхронном обмене приходится искусственно увеличивать длительность строба обмена для соответствия требованиям большего числа исполнителей, чтобы они успевали обмениваться информацией в темпе процессора. Поэтому иногда в магистрали предусматривают возможность как синхронного, так и асинхронного обмена, причем синхронный обмен является основным и довольно быстрым, а асинхронный применяется только для медленных исполнителей.

Обмен информацией в контроллерах устроен принципиально по-другому без использования внешнего системного интерфейса. Каждый МК имеет некоторое количество линий ввода/вывода, которые объединены в многоразрядные (чаще 8-разрядные) параллельные порты ввода/вывода. В памяти МК каждому порту ввода/вывода соответствует свой адрес регистра данных. Обращение к регистру данных порта ввода/ вывода производится теми же командами, что и обращение к памяти данных. Кроме того, во многих МК отдельные разряды портов могут быть опрошены или установлены командами битового процессора.

В зависимости от реализуемых функций различают следующие типы параллельных портов:

• однонаправленные порты, предназначенные только для ввода или только для вывода информации;

• двунаправленные порты, направление передачи которых (ввод или вывод) определяется в процессе инициализации МК;

• порты с альтернативной функцией (мультиплексированные порты). Отдель-ные линии этих портов используются совместно со встроенными периферийными устройствами МК, такими как таймеры, АЦП, контроллеры последовательных интерфейсов;

• порты с программно управляемой схемотехникой входного/выходного буфера.

Порты выполняют роль устройств временного согласования функционирования МК и объекта управления, которые в общем случае работают асинхронно. Различают три типа алгоритмов обмена

информацией между МК и внешним устройством через параллельные порты ввода/вывода:

• режим простого программного ввода/вывода;

• режим ввода/вывода со стробированием;

• режим ввода/вывода с полным набором сигналов подтверждения обмена.

В современных МК, как правило, обеспечивается индивидуальный доступ к портам, что позволяет использовать каждую линию независимо в режиме ввода или вывода.

Необходимо обратить особое внимание на то, что при вводе данных с порта считывается значение сигнала, поступающее на внешний вывод. При выводе информации на порт поступает содержимое регистра данных порта ассоциированное с одним из регистров оперативной памяти.

Использование регистра адреса/данных

Использование пары регистров *HL*в качестве указателя адреса является интересным свойством типового МП. Обычно рассматривают ее использование в качестве указателя адреса, когда она временно берет на себя роль основного счетчика команд, указывая адрес ячейки памяти или УВВ. Многие широко распространенные МП (например, Intel 8080/8085, Z 80) содержат регистры такого типа. Регистры адреса/данных в рассматриваемом типовом МП называются также парой *HL*.-регистров, регистром адреса, счетчиком данных или указателем адреса.

Рассмотрим простую задачу сложения содержимого трех последовательных ячеек памяти с размещением суммы в следующей ячейке памяти

(выполнение ее показано на рис. 5.2).

Программа загружена в ячейки памяти 2000H — 200AH, а три слагаемых (0CH + 0AH + 07H) — в ячейки памяти 2100H—2102H. Программа содержит 6 команд, записанных справа на рис. 5.2. Не следует забывать, что текущая сумма будет всегда помещаться в аккумулятор, содержащий вначале первое слагаемое (0CH).

Команда *1* имеет КОП 3AH (рис. 5.2) и приказывает МП ЗАГРУЗИТЬ (LOAD) в аккумулятор содержимое ячейки памяти 2100H. Выполнение этой команды прямой загрузки аккумулятора показано на рис. 5.3, *а.* После выполнения команды аккумулятор будет содержать первое слагаемое (0CH).

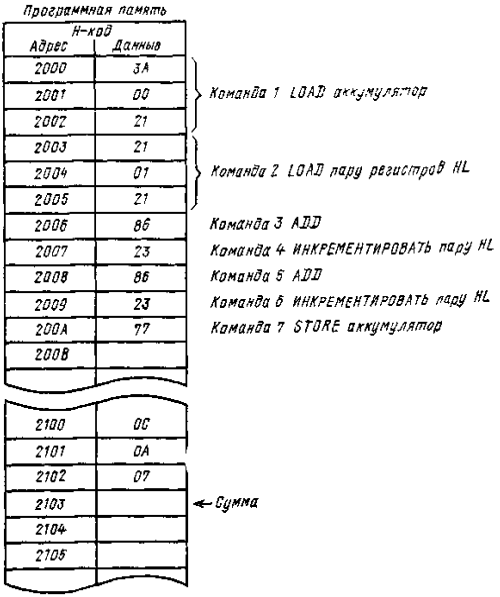


Рис. 5.2. Воображаемая память и команды в примере сложения

Команда *2* приказывает МП загрузить (LOAD) 2101H в пару регистров *HL*,емкость которых 16 бит. Это число представляет собой адрес памяти данных. Более точно команду *2* можно сформулировать так: загрузить пару регистров *HL*непосредственно следующими за КОП данными, ее выполнение приведено на рис. 5.3, *б.* Заметим, что содержимое первой ячейки памяти (2004H) загружается в младший байт L, следующей за ней — в старший байт *H* пары регистров *HL.*

Команда *3* приказывает МП выполнить операцию сложить (ADD) содержимое аккумулятора с содержимым ячейки памяти, адрес которой содержится в паре регистров *HL.* Ее выполнение приведено на рис. 5.3, *в* (команда ADD). Пара регистров *HL*указывает на ячейку памяти 2101H, и АЛУ складывает свое содержимое ($0000\ 1010_2$) с содержимым аккумулятора ($0000\ 1100_2$), что дает сумму ($0001\ 0110_2$), помещаемую в аккумулятор.

Команда *4* приказывает МП инкрементировать (увеличить на 1) содержимое пары регистров *HL*(см. рис. 5.3, *г).* Заметим, что изменился только младший байт пары регистров *HL.*

Команда *5* снова приказывает МП сложить (ADD) содержимое аккумулятора и ячейки памяти с адресом 2102H, на которую указывает пара регистров *HL*(см. рис. 4.3,*б*) .Оба содержимых складываются, что дает сумму ($0001\ 1101_2$), помещаемую в аккумулятор.

По команде *6* содержимое пары регистров *HL*снова инкрементируется (см. рис. 5.3, *е).*

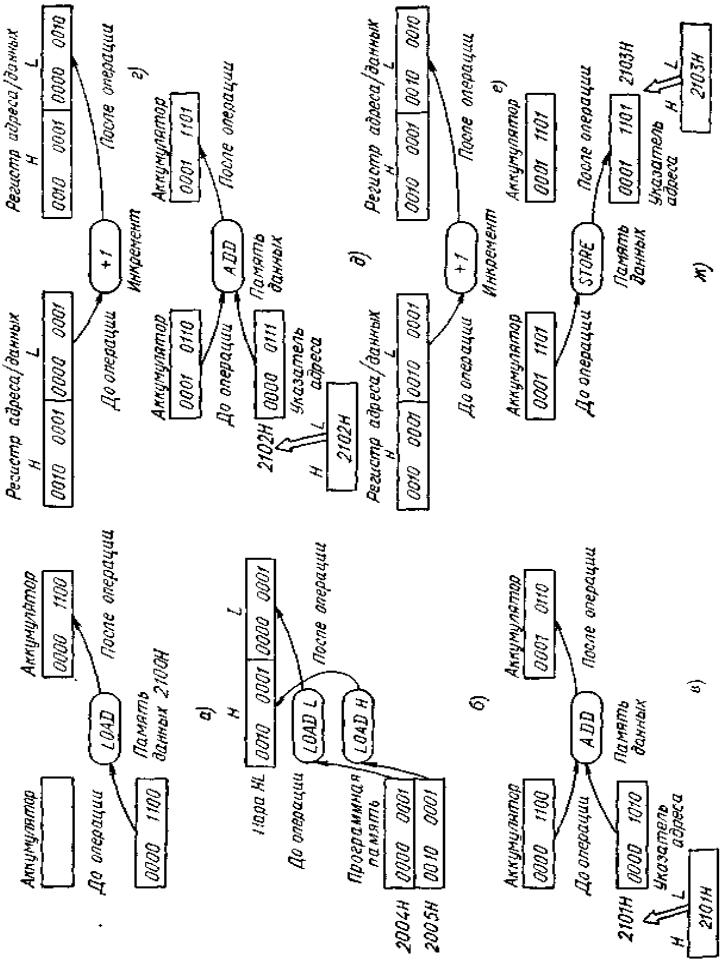


Рис. 5.3. Выполнение операций в соответствии с программой рис. 5.2:

а — команда 1 загрузки аккумулятора; б — команда 2 загрузки пары регистровHL\ в — команда 3 сложения, г — команда 4 инкремента пары регистров; д —команда 5 сложения; е — команда 6 инкремента пары регистров HL\ ж — команда 7 размещения в памяти содержимого акумулятора.

Команда *7* приказывает МП поместить (STORE) содержимое аккумулятора, т. е. окончательную сумму ($0001\ 1101_2$) в ячейку памяти, на которую указывает пара регистров *HL* (см. рис. 5.3, *ж), т.е.* по адресу 2103H.

Команды *3, 5, 7,* взаимодействующие с парой регистров *HL*как с указателем адреса, используют косвенно-регистровый способ адресации. Его мы изучим в следующей главе.

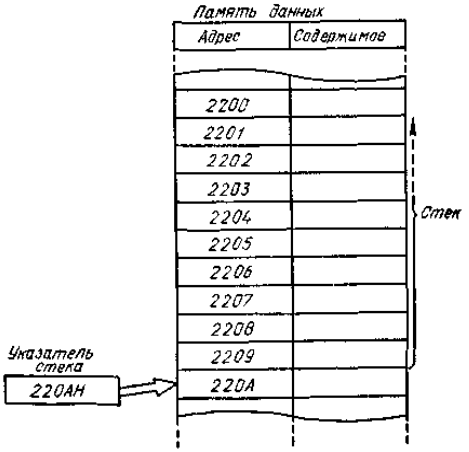
Использование указателя стека

Типовый микропроцессор содержит указатель стека — специализированный 16-разрядный регистр-счетчик, содержимым которого всегда является адрес. Этот адрес принадлежит особой группе ячеек памяти данных, которая называется *стеком.* В некоторых МП стек может быть составлен из группы физически локализованных на кристалле МП ячеек памяти. Когда микропроцессором выполнялась подпрограмма обслуживания прерывания, текущие данные во всех регистрах МП должны были временно сохраняться. Эта сохранность обеспечена стеком. А когда подпрограмма полностью выполнена, содержимое счетчика команд должно быть сохранено таким образом, чтобы МП мог возвратиться в соответствующее место в программной памяти. Зона временной памяти является стеком. Напомним, что подпрограмма является короткой, часто используемой, специализированной программой (например, умножить).

Стек типового микропроцессора будет содержаться в ОЗУ, и его положение определяется

программистом. Указатель стека загружается старшим адресом, представляющим собой вершину стека (рис. 5.4). В этом случае указательстека содержит 220АН, что на единицу старше первой ячейки памяти стека 2209Н.

Данные можно записать в стек, используя команды PUSH (поместить) или CALL (вызвать). Они могут быть считаны из стека по командам POP (извлечь) или RETURN (возврат). Стек функционирует как память с последовательным доступом по типу: данные, поступившие последними, извлекаются первыми (тип LIFO от *Last In — First Out* — последний входит — первый выходит, или FILO от *First In — Last Out* —первый входит — последний выходит).



Рнс. 5.4. Расположение стека в ОЗУ

Команда загрузки в стек (PUSH) приводит к результату, показанному на рис. 4.5, *а.* Содержимое пары регистров *HL*помещается в стек. Отметим, что двухбайтовая пararегистров *HL*должна быть размещена в двух ячейках памяти стека.

Последовательность событий может быть описана в соответствии с номерами, показанными в кружках на рис. 5.5.

1. Указатель стека МП декрементируется от 220АН до 2209Н.

2. Указатель стека показывает на ячейку памяти 2209Н по адресной шине и старший байт ($0000\ 0000_2$) помещается в стек.

3. Указатель стека снова декрементируется от 2209Н до 2208Н.

4. Указатель стека указывает на ячейку памяти 2208Н (по адресной шине системы) и младший байт из регистра данных ($0000\ 1111_2$) загружается в стек.

На рис. 5.5, *б* показано выполнение другой операции загрузки. На этот раз в стек загружается содержимое аккумулятора и регистра состояния. Проследим снова за событиями, отмеченными цифрами в кружках.

5. До операции указатель стека указывает на последнюю ячейку стека. Ее называют вершиной стека. Затем указатель стека декрементируется до 2207Н.

6. Указатель стека указывает на ячейку памяти 2207Н, и содержимое аккумулятора ($0101\ 0101_2$) загружается в стек по этому адресу.

7. Указатель стека декрементируется от 2207Н до 2206Н.

8. Указатель стека указывает на ячейку памяти 2206Н. Содержимое регистра состояния ($1111\ 1111_2$) загружается по этому адресу.

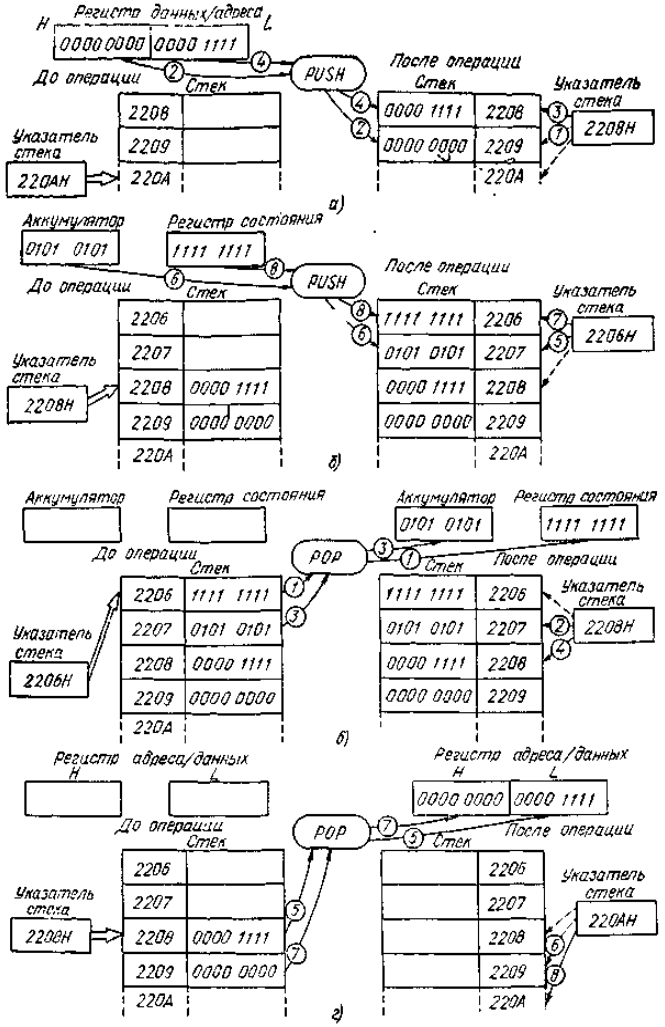


Рис. 5.5. Операции со стеком:

а — помещение в стек содержимого регистра адреса/данных; б — помещение в стек содержимого аккумулятора и регистра состояния; в — извлечение из стека содержимого аккумулятора н регистра состояния; г ~ извлечение из стека содержимого регистра адреса/данных

Стек может продолжать расти, пока длится процесс загрузки в него (на рис. 5.4 показано, что стек растет вверх).Стек не имеет ограничений, за исключением тех, которые обусловлены наличием других программ в ОЗУ.

Обычно каждой команде загрузки в стек (PUSH) позже будет соответствовать команда извлечения из стека (POP), по которой данные берутся из вершины стека. Поскольку стек является памятью типа LIFO (последний вошел — первый вышел), данные должны извлекаться из стека в порядке, обратном загрузке. На рис. 5.5, *в* и *г* подробно показаны операции извлечения из стека.

Рассмотрим команду POP на рис.5.5, *в*. Аккумулятор и регистр состояния свободны до операции извлечения из стека. Следуем последовательности, указанной цифрами в кружках.

1. Указатель стека указывает на вершину стека, т. е. на адрес 2206H. Содержимое регистра состояния ($1111\ 1111_2$) извлечено из стека и переслано в АЛУ.
2. Указатель стека инкрементирован с 2206H до 2207H.
3. Указатель стека указывает на адрес 2007H стека. Вершина стека извлекается, и ее содержимое пересылается в аккумулятор АЛУ.
4. Указатель стека инкрементирован до 2208H и указывает теперь на следующий адрес извлечения из стека.

Содержимое аккумулятора и регистра состояния было восстановлено до тех значений, которые были до операции PUSH, показанной на рис. 5.5, *б*.

Затем (на рис. 5.6, *г)* содержимое регистра адреса/данных в свою очередь извлекается из стека. Снова последуем согласно заключенным в кружки цифрам:

5. Указатель стека указывает на вершину стека (адрес 2208H). Содержимое этой ячейки памяти стека извлекается и пересылается в младший байт пары регистров *HL.*

6. Указатель стека инкрементируется до 2209H.

7. Указатель стека показывает на вершину стека, т. е. теперь адрес 2209H, содержимое которого ($0000\ 0000_2$), передается в старший байт пары регистров *HL.*

8. Указатель стека инкрементируется от 2209H до 220AH для последующей операции загрузки в стек (PUSH) или извлечения из стека (POP).

Извлечение данных из стека и их восстановление в регистре адреса/данных является действием, обратным операции загрузки в стек (PUSH), выполненной на рис. 5.5, *а.* Команды PUSH и POP используются всегда совместно, однако между ними располагаются другие команды, которые меняют данные, содержащиеся в регистрах МП.

Наш типовой МП загружает в стек и извлекает содержимое пары регистров. Некоторые МП осуществляют эти операции только с однобайтовыми регистрами, единственной командой. В других МП указатель стека может указывать на пустую ячейку памяти по ближайшему большему по отношению к вершине стека адресу вместо указания на вершину стека, как в предыдущем случае. При этих условиях имеется возможность сохранить в памяти то, что

программист загрузил в начале (адрес 220АН в нашем случае) в указатель стека для определения его адреса