

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
Санкт-Петербургский государственный университет
аэрокосмического приборостроения

Кафедра вычислительных систем и сетей

Структуры и системы команд микропроцессорных систем

Учебно-методическое пособие

Санкт-Петербург
2016

Подготовлено кафедрой вычислительных систем и сетей.

Составители: кандидат технических наук, доцент Н.М. Иванов
кандидат технических наук, доцент Т.Н. Соловьева

Содержатся сведения о структуре и системе команд процессорного модуля К-580, а также описание программно-аппаратного комплекса поддержки разработки микроконтроллерных систем на базе МК SAB80C515 и системы команд микроконтроллера семейства MCS51 на примере выполнения простейших программ.

Учебно-методическое пособие может быть использовано при изучении дисциплин «ЭВМ и периферийные устройства» и «Микропроцессорные системы» студентами, обучающимися по направлению 09.03.01 «Информатика и вычислительная техника», направленность «Вычислительные машины, комплексы, системы и сети» очной, очно-заочной (вечерней) и заочной форм обучения.

Содержание

1 Основные сведения о микропроцессоре KP580BM80A	3
2 Система команд микропроцессора KP580BM80A	6
3 Использование команд перехода	11
4 Организация аппаратных прерываний в микропроцессоре	12
5 Структурная схема МК SAB 80C515	14
6 Организация памяти МК SAB 80C515	17
7 Внутренняя память данных	19
Контрольные вопросы	28
Приложения	29

1 Основные сведения о микропроцессоре КР580ВМ80А

1.1 Структура микропроцессора

Структурная схема процессорного модуля К-580 приведена на рис 1. В Состав кристалла процессора (ПР) входит устройство синхронизации (УС), устройство управления (УУ), устройство обработки (УО), блок регистров (БР), буфер адреса (БА) и буфер данных (БД).

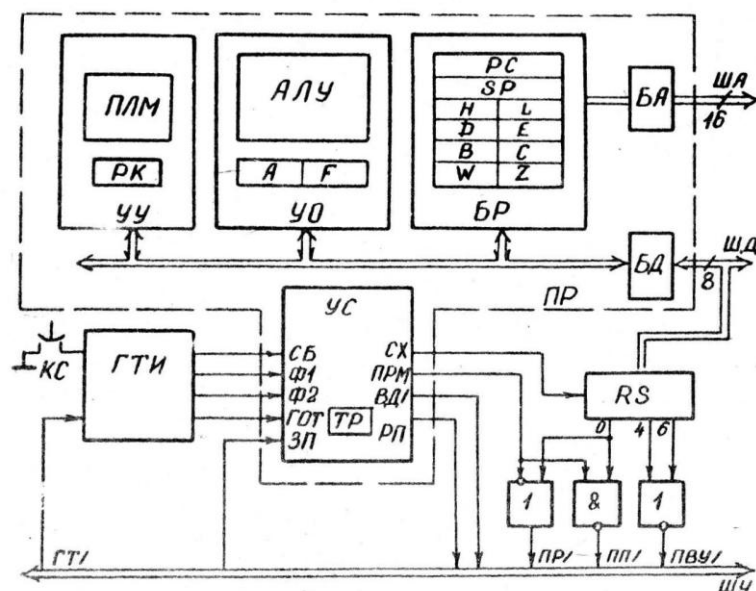


Рис. 1 Структурная схема микропроцессора

В ходе выполнения программы процессор устанавливает адреса памяти на 16-разрядной шине адреса (ША). По 8-разрядной двунаправленной шине данных (ШД) из памяти принимаются коды команд и данные, по этой же шине данные записываются в память. Коды команд поступают по внутренней магистрали в регистр команд (РК) и дешифруются с помощью программируемой логической матрицы (ПЛМ). Для каждой команды устройство управления генерирует необходимую последовательность сигналов, управляющих обработкой и пересылкой информации в кристалле.

Устройство обработки с помощью арифметико-логического устройства (АЛУ) выполняет задаваемые кодом команды арифметические и логические операции. При этом один из операндов находится в аккумуляторе А, а второй подается через внутреннюю магистраль из блока регистров БР или с шины данных через буфер данных БД. Результат операции фиксируется в аккумуляторе А, а признаки результата – в регистре F. Отдельные биты регистра F характеризуют результат операции следующим образом (см. таблицу 1).

16-разрядный счетчик команд РС находится в блоке регистров и после выборки каждого байта команды его значение увеличивается на 1. В случае программных переходов в РС заносится адрес перехода.

16-разрядный указатель стека SP используется для адресации находящейся в ОЗУ стековой области памяти. При записи информации в стек содержимое SP

уменьшается на число переданных байт, а при выборке информации – увеличивается.

Таблица 1. Значения битов в регистре F

Номер разряда	Обозначение признака	Параметр результата, при котором признак = 1
7	S	7 (старший) разряд результата равен 1
6	Z	Результат равен нулю
5	0	
4	W	Перенос из разряда 3 результата
3	0	
2	P	Четный паритет (число единичных разрядов)
1	1	
0	Y	Перенос из 7 разряда результата

Примечание: биты 5 и 3 регистра F всегда равны нулю; бит 1 всегда равен 1.

8-разрядные регистры H, L, D, F, B, C, W, Z используются для хранения промежуточных результатов. Заметим, что регистры W и Z, так же как регистр команд РК, не являются программно-управляемыми. Выдачей информации из регистров БР на шину адреса ША и обменом с шиной данных управляет УУ. В состав БР входит также не показанная на рис.1 схема инкремента – декремента, позволяющая увеличивать или уменьшать, содержимое регистров.

Устройство синхронизации УС осуществляет тактирование работы кристалла и совместно с УУ – обработку и выдачу внешних управляющих сигналов. Функции управляющих сигналов рассмотрим в следующем разделе.

1.2 Выполнение команд

Процедуру выполнения команд микропроцессором (МК) рассмотрим на примере команды OUT 1Ch, которая выдает байт информации из аккумулятора А в буферный регистр внешнего устройства (ВУ) с адресом 1С. Все числа далее указываются в 16-ричном коде, при записи на ассемблере этот код заканчивается символом h (например, 1Ch).

Сама команда OUT 1Ch занимает в памяти 2 байта. В первом байте находится код команды D3, во втором – адрес внешнего устройства (1С). Будем для определенности полагать, что первый байт команды расположен в ячейке 007F, и программный счетчик указывает на нее (РС = 007F), а содержимое аккумулятора перед выводом равно 3В. Коды команд микропроцессора приведены в Приложении Б.

Количество циклов, в течение которых выполняется каждая команда, равно числу необходимых обращений к памяти и/или внешним устройствам во время выполнения команды. Команда OUT выполняется в течение 3 циклов (рис. 2) – выборки кода операции, чтения адреса ВУ и вывода содержимого аккумулятора. Каждый цикл состоит из нескольких тактов. Тактирование осуществляется подачей внешних импульсов частотой 2 МГц на входы Ф1 и Ф2 с генератора тактовых

импульсов (ГТИ) (рис. 1). Число циклов и тактов в них для каждой команды приведено в Приложении А.

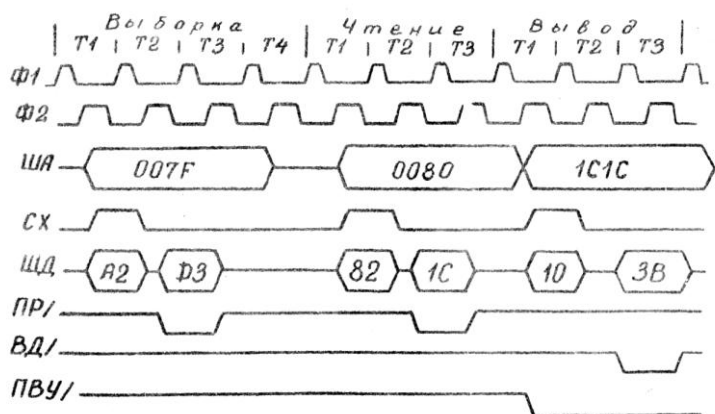


Рис. 2 Временные диаграммы выполнения команды OUT 1Ch

В первом такте Т1 каждого цикла на шине ША выставляется необходимый адрес, а на шину данных из УУ выдается байт состояния, который фиксируется во внешнем регистре состояния RS с помощью сигнала СХ. Байт состояния определяет тип выполняемого цикла. Его отдельные биты S7, ... , S0 имеет следующий смысл:

- S7 = 1 чтение информации из памяти;
- S6 = 1 ввод информации с внешнего устройства;
- S5 = 1 цикл выборки кода операции;
- S4 = 1 вывод информации на внешнее устройство;
- S3 = 1 цикл останова (после команды HLT);
- S2 = 1 адрес стека (на ША выдано содержимое SP);
- S1 = 0 выдача информации (в память или внешнее устройство);
- S0 = 1 цикл прерывания (только от внешнего устройства).

Примечание: если команда обращается к внешнему устройству или работает со стеком, это обязательно указано в описании команды (раздел 2).

Согласно схеме (рис. 1) биты 4 и 6 регистра состояния RS используются для получения признака внешнего устройства ПБУ/, а 0 бит – для получения сигнала подтверждения прерывания ПП/ (символ / отмечает, что сигнал активен по низкому уровню). В цикле выборки кода операции байт состояния равен A2 (активны биты 7 и 5).

Во втором такте Т2 процессор проверяет сигнал готовности ГОТ, который поступает с шины управления (ШУ) через ГТИ по цепи ГТ/ – ГОТ. Если ГОТ=0 (ГТ/=1), процессор переходит в фазу ожидания, и выполнение программы приостанавливается до момента восстановления высокого уровня на входе ГОТ. В конце второго такта процессор выдает сигнал приема ПРМ или выдачи ВД/. Они определяют направление и момент передачи информации по шине данных. В цикле выборки кода операции выдается сигнал ПРМ, который используется для формирования системного сигнала прием ПР/ на шине управления. В цикле прерывания (S0 = 1) формируется системный сигнал подтверждения прерывания

ПП/=S0^PPM. Если сигнал готовности ГОТ активен, процессор переходит к такту ТЗ, в котором завершается прием или выдача информации по шине данных.

Четвертый такт Т4 в цикле выборки используется для внутренних операций процессора, в частности, для дешифрования кода команды и межрегистровых передач. В некоторых командах такие передачи требуют пятого такта работы процессора.

Таким образом, в первом цикле М1 – выборка кода операции – процессор считывает из РС и выдает на ША адрес первого байта команды 007F и принимает по ШД код операции (КОП) D3 (содержимое первого байта команды) в регистр команд РК. Во втором цикле – чтение памяти – процессор считывает из РС и выдает на ША адрес второго байта команды 0080 и принимает по ШД адрес внешнего устройства 1Ch (содержимое второго байта команды, оно записывается в регистры W и Z). В третьем цикле – вывод информации на ВУ – процессор выдает на ША адрес ВУ (1C1C) из регистров W и Z, содержимое аккумулятора 3B по ШД выводится в буферный регистр ВУ. Так как адреса ВУ в отличие от адресов памяти восьмиразрядные, то при их выдаче по 16-разрядной шине адреса младший и старший байты дублируются.

В последнем такте последнего цикла каждой команды процессор анализирует состояние входа запроса прерывания ЗП (рис. 1), если триггер разрешения прерывания ТР установлен в 1. Состояние ТР выдается на выход разрешения прерывания РП. Если ТР = 1 и ЗП = 1, процессор переходит к выполнению особого цикла прерываний М1 (раздел 4).

При нажатии кнопки сброса КС (рис. 1) на вход СБ через ГТИ подастся высокий логический уровень, что вызывает сброс счетчика команд РС. В результате при снятии сигнала сброса программа всегда начинает выполняться с команды, записанной в нулевой ячейке памяти.

2 Система команд микропроцессора КР580ВМ80А

Рассмотрим теперь более подробно систему команд процессора К-580, которая приведена в Приложении А. В графе 1 таблицы А.1 указывается мнемокод операции, в графе 2 – операнды. При записи программ на ассемблере символы R1, R2, R в поле операндов заменяются обозначениями регистров A, B, C, D, E, H, L, а символ RP – обозначениями регистровых пар SP, B, D, H (пары B-C, D-E, H-L обозначаются по старшему регистру).

Символы * и ** в поле операнда при записи на ассемблере заменяются однобайтными (*) и двухбайтными (**) выражениями, которые в простейшем случае могут быть числами или идентификаторами. В процессе трансляции определяются значения этих выражений, и полученные числа размещаются во втором (*) или втором и третьем (**) байтах команды (второй – младший байт операнда, третий – старший байт операнда). Таким образом, в Приложении А символы * и ** являются признаком двухбайтной (*) или трехбайтной (**) команды. Остальные команды – однобайтные. Код операции всегда находится в первом байте команды.

Символы ?? в поле кода операции (команды условной передачи управления)

заменяются обозначениями условий, приведенными в таблице А.2.

В графе 3 таблицы А.1 приводится краткое описание операции, в графе F отмечается формирование признаков в регистре F, в графе t указывается количество тактов в каждом цикле команды. Формирование признаков в соответствии с обозначениями графы F поясняется в примечаниях.

Команды, использующие символ M в поле операнда, работают с байтом памяти, адрес которого предварительно должен быть установлен в регистровой паре H-L (например, командой LXI H, ADR).

Коды команд приведены в Приложении Б.

Поясним особенности выполнения отдельных команд.

2.1 Команды пересылки

MOV R1, R2 – информация из регистра R2 пересылается в регистр R1. Содержимое R2 не изменяется.

MOV R, M MOV M, R – пересылка из памяти в регистр и из регистра в память.

MVI R, * MVI M, * – непосредственная засылка байта * в регистр (R) или ячейку памяти (M).

LXI RP, ** – непосредственная засылка двухбайтного числа ** в одну из регистровых пар B, D, H, SP.

LDA ** – байт из памяти засылается в аккумулятор A.

STA ** – содержимое аккумулятора загружается в память, адрес памяти указывается аналогично команде LDA.

LHLD ** – содержимое ячейки ** засылается в регистр L а следующей ячейки – в регистр H.

SHLD ** – содержимое регистра L записывается по адресу **, а регистра H – в следующую ячейку.

LDAX RP – содержимое ячейки памяти, адрес которой находится в регистровой паре B-C или D-E (вместо RP подставляется B или D), записывается в аккумулятор.

STAX RP – содержимое аккумулятора записывается в ячейку памяти, адрес которой находится в регистровой паре B-C или D-E.

XCHG – обмен информацией между регистровыми парами H-L и D-E.

SPHL – содержимое пары H-L. засылается я указатель стека SP.

PCHL – содержимое пары H-L засылается в счетчик команд PC.

2.2 Арифметические команды.

При выполнении арифметических команд один из операндов находится в аккумуляторе, местонахождение другого указывается в команде; результат фиксируется в аккумуляторе, по значению результата формируются все признаки. При вычислении разности вычитаемое инвертируется и увеличивается на 1. Затем производится сложение с содержимым аккумулятора, **однако, в отличие от обычного сложения признак переноса Y устанавливается инверсно (Y = 1 при отсутствии переноса из старшего (знакового) разряда, при возникнове-**

нии переноса $Y = 0$).

ADD R ADD M ADI * – содержимое аккумулятора складывается с содержимым регистра R, ячейки памяти M (адресуемой парой H-L) или со вторым байтом команды *.

ADC R ADC M ACI * – эти команды действуют аналогично командам ADD и ADI, но в сложении также участвует значение бита переноса Y, установленное в регистре признаков F до выполнения команды.

SUB R SUB M SUI * – из содержимого аккумулятора вычитается содержимое регистра R, ячейки памяти M или непосредственно заданное значение байта *.

SBB R SBB M SBI * – эти команды действуют аналогично командам SUB, SUI, но в вычитании участвует ранее установленный бит переноса Y.

Рассмотренные арифметические команды формируют все признаки в регистре F.

INR R INR M – увеличение на 1 содержимого регистра R или ячейки памяти.

DCR R DCR M – уменьшение на 1 содержимого регистра R или ячейки памяти.

Команды INR R (M) и DCR R (M) формируют все признаки кроме признака переноса Y.

INX RP DCX RP – увеличение (INX) или уменьшение (DCX) на 1 содержимого регистровой пары. Символы RP в поле операнда заменяются обозначениями регистровых пар B, D, H, SP. Эти команды не формируют признаков (регистр F не меняется).

DAD RP – содержимое указанной регистровой пары (RP=B, D, H, SP) складывается с содержимым пары H-L. Результат сложения фиксируется в паре H-L. Формируется только бит переноса Y. **Внимание!** Команда DAD выполняется в течение трех машинных циклов, в течение двух последних машинных циклов на шинах ШД и ША информация **не формируется**.

DAA – корректируется находящийся в аккумуляторе результат сложения двух операндов, представленных в двоично-десятичном коде. В результате коррекции сумма в аккумуляторе также преобразуется в двоично-десятичную форму. Для этого если величина младшего полубайта аккумулятора больше 9 или установлен флаг вспомогательного переноса W в регистре F, то к A прибавляется 06. Если величина старшего полубайта аккумулятора больше 9 или установлен флаг переноса Y, к A прибавляется 60. Если обе тетрады A больше 9 или установлены оба переноса, к A прибавляется 66.

Примечание: Двоично-десятичный код BCD (binary-coded decimal) — форма записи целых чисел, когда каждый десятичный разряд числа записывается в виде его четырёхбитного двоичного кода. Например, десятичное число 311_{10} будет записано в двоичном коде как $1\ 0011\ 0111_2$, а в двоично-десятичном коде как $0011\ 0001\ 0001_{BCD}$.

2.3 Логические команды и сдвиги

При выполнении сдвигов и всех логических команд результат фиксируется в аккумуляторе.

ANA R ANA M ANI * – конъюнкция. Содержимое аккумулятора поразрядно логически умножается на содержимое регистра R, ячейки памяти M или непосредственно заданный байт *. Формируются все признаки за исключением признака перекоса Y, который сбрасывается в ноль.

XRA R XRA M XRI * – исключающее ИЛИ. Содержимое аккумулятора поразрядно суммируется по модулю 2 с содержимым регистра R, ячейки памяти M или непосредственно заданным байтом *. Формируются все признаки, Y сбрасывается в ноль.

ORA R ORA M ORI * – дизъюнкция. Содержимое аккумулятора поразрядно логически складывается с содержимым регистра R, ячейки памяти M или непосредственно заданным байтом *. Формируются все признаки. Y сбрасывается в ноль.

CMP R CMP M CPI * – сравнение. Из содержимого аккумулятора вычитается содержимое регистра R, ячейки памяти M или непосредственно заданный байт *. Содержимое аккумулятора в отличие от предыдущих команд не изменяется, однако в регистре признаков F фиксируются все признаки результата вычитания.

RLC – кольцевой сдвиг содержимого аккумулятора влево (рис. 3). Старший бит аккумулятора при сдвиге устанавливает признак Y.

RRC – кольцевой сдвиг содержимого аккумулятора вправо (рис. 3). Младший бит при сдвиге устанавливает признак Y.

RAL – кольцевой сдвиг содержимого аккумулятора и признака Y влево (рис. 3).

RAR – кольцевой сдвиг содержимого аккумулятора и признака Y вправо (рис. 3).

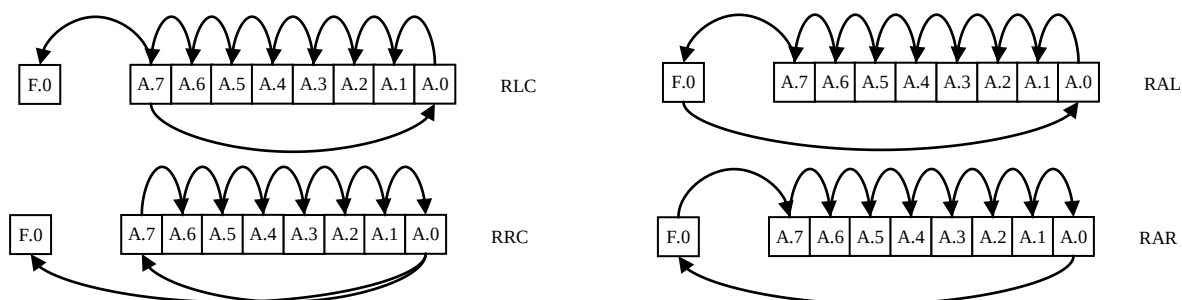


Рис. 3 Операции сдвигов

CMA – инверсия содержимого аккумулятора, признаки не формируются.

CMS – инверсия признака Y.

STC – установка признака Y равным 1.

2.4 Стековые операции и управление внешним обменом

При записи байта информации в стек сначала уменьшается на 1 значение указателя стека SP, затем по адресу, находящемуся в SP, производится запись в ОЗУ. При чтении байта информации из стека сначала выполняется чтение по адресу, находящемуся в SP, затем содержимое SP увеличивается на 1. При записи и чтении двухбайтных чисел или содержимого регистровых пар указанные операции повторяются дважды, причем старший байт всегда располагается в стеке по большему адресу. Аналогично записывается в стек содержимое счетчика команд при выполнении команд перехода.

PUSH RP – содержимое регистровой пары (RP=B, D, H) записывается в стек.

PUSH PSW – содержимое аккумулятора A и регистра признаков F записывается в стек (A – по большему адресу).

POP RP – содержимое регистровой пары (RP=B, D, H) задается из стека.

POP PSW – содержимое аккумулятора A регистра признаков F задается из стека (A располагается по большему адресу).

XTHL – из стека задается значение регистровой пары H-L (сначала L, затем H), прежнее значение H-L заносится в стек (сначала H, затем L).

IN * – информация из внешнего устройства (вместо * указывается адрес внешнего устройства) вводится в аккумулятор.

OUT * – информация из аккумулятора выводится на внешнее устройство с адресом *.

EI – разрешение прерывания.

DI – запрет прерывания.

HLT – останов до появления прерывания.

NOP – нет операции, переход к следующей команде.

2.5 Команды передачи управления.

При записи команд перехода символы ** заменяются двухбайтным выражением – адресом перехода, значение которого при трансляции размещается во 2 и 3 байтах команды. Для условных переходов символы ?? в имени команды заменяются обозначениями условий, приведенными в таблице А.2 (приложение А). Если условие не выполнено, переход не производится.

JMP ** – значение выражения ** заносится в счетчик команд, т.е. происходит безусловный переход по указанному адресу.

J?? ** – ** заносится в счетчик команд, если выполнено условие ??.

CALL ** – содержимое счетчика команд спасается в стеке, а в счетчик команд заносится значение ** (вызов подпрограммы).

C?? ** – команда действует как CALL, если выполнено условие ??.

RET – значение счетчика команд восстанавливается из стека (возврат из подпрограммы).

R?? – аналогично RET, если выполнено условие ??.

RST N – вызов подпрограммы обслуживания аппаратных прерываний. При записи команды N заменяется целым числом от 0 да 7. При выполнении команды содержимое счетчика команд спасается в стеке и в счетчик заносится адрес

0000000000bbb000 (bbb – двоичный код числа N), что обеспечивает переход в один из 8 начальных адресов памяти. Коды 8 модификаций команды приведены в Приложении Б.

Более подробно выполнение команд передачи управления описано в разделах 3 и 4.

3 Использование команд перехода

По принципу работы с адресами команды перехода могут быть разделены на две группы.

1) *Без запоминания адреса точки возврата.* Это команды вида JMP ADR, где ADR – символическое имя двухбайтового адреса команды, которой передается управление. Действие данной команды иллюстрирует карта памяти (рис. 4, слева) (здесь слева указывается адрес, справа – содержимое ячейки памяти с данным адресом).

Команды первой группы применяются для организации ветвлений.

Для организации подпрограмм используются команды второй группы.

2) *С запоминанием адреса точки возврата* – команды вызова подпрограммы (возврата из подпрограммы) – это команды типа CALL ADR, RET, где ADR – символическое имя начального адреса подпрограммы. После выполнения команды CALL ADR управление передается команде, находящейся в памяти по адресу ADR – первой команде подпрограммы.

Последней командой любой подпрограммы является команда RET, после выполнения которой управление передается команде, следующей за CALL (назовем ее точкой возврата). Запоминание адреса точки возврата (АТВ) осуществляется в стеке.

Таким образом, при выполнении команды CALL процессор выполняет следующие действия: 1) содержимое PC записывает в стек. 2) в PC записывается адресная часть команды CALL.

При выполнении команды RET процессор считывает содержимое PC из стека.

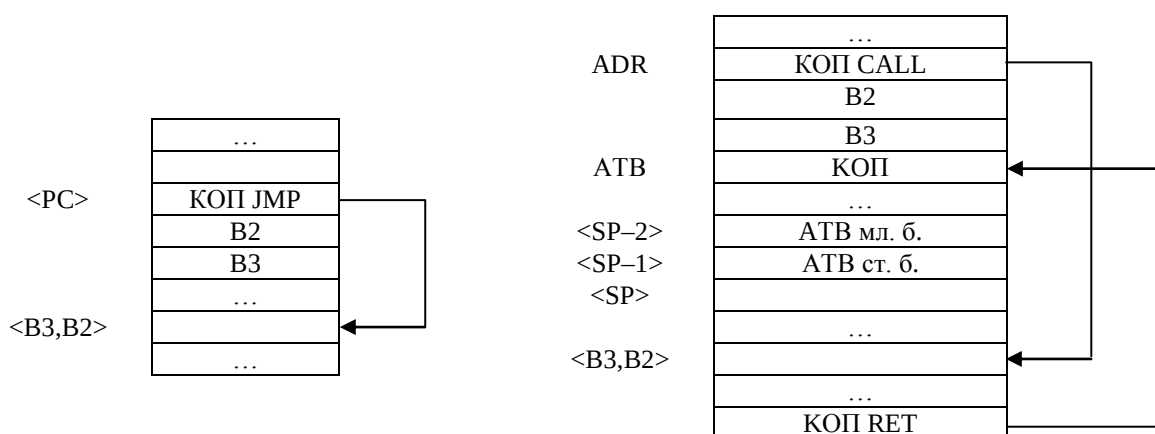


Рис. 4 Схема перехода без запоминания АТВ (слева) и с запоминанием АТВ (справа).

По наличию условий команды перехода можно разделить на две группы.

1) *Безусловные*: JMP ADR; CALL ADR; RET.

2) *Команды условного перехода*: J?? ADR; C?? ADR; R??.

При выполнении условия ?? команда выполняется, как команда безусловной передачи управления, в противном случае процессор лишь считывает команду из памяти, после чего управление передается команде, следующей за данной.

4 Организация аппаратных прерываний в микропроцессоре

Различают программные и аппаратные прерывания. Программное прерывание имеет место при использовании в программе команды CALL (или C??). Данный раздел посвящен аппаратным прерываниям.

Аппаратное прерывание вызывается появлением на входе ЗП (INT) МП активного сигнала. Он приходит асинхронно по отношению к команде, т.е. в этот момент процессор может выполнять какую-то команду. В конце выполнения каждой команды, если прерывания разрешены (РП=1), МП опрашивает входной сигнал запроса прерывания ЗП. Если выполняется условие $ЗП \wedge РП = 1$, процессор завершает выполнение текущей команды и переходит в особый цикл М1 выборки кода операции. В этом цикле принимается команда вызова подпрограммы обслуживания прерывания (CALL для случая А или RST для случая Б). При приеме этой команды РС не инкрементируется. Принятая команда выполняется.

Команда вызова подпрограммы обслуживания прерываний формируется контроллером прерываний (А) или внешним устройством (Б).

А) Если в состав комплекта входит контроллер прерываний, он определяет ВУ с максимальным приоритетом (из всех ВУ, запросивших прерывание) и формирует команду CALL AD_max – вызов подпрограммы, где AD_max – начальный адрес подпрограммы обслуживания прерывания от источника с максимальным приоритетом. Когда МП перейдет в особый цикл М1, он будет принимать три байта команды CALL AD_max, стробируя прием каждого байта сигналом разрешения прерывания РП (INTE). В течение трех машинных циклов байт состояния будет равен 23h.

Б) Если контроллера прерываний нет, внешнее устройство, имеющее номер N, формирует код команды RST N, т.е. ВУ само формирует код вызова подпрограммы. Формат команды RST N: **11bbb111** (bbb – двоичное представление N).

Для этого в состав каждого ВУ включен специальный регистр. Пример такого регистра для N=5 приведен на рис. 5.

Для N=5 код вызова подпрограммы RST N **11101111**. Этот код формируется в регистре и по сигналу РП (INTE) от МП поступает по ШД в МП в особом цикле М1. На основании этого кода формируется начальный адрес подпрограммы обслуживания прерываний от данного ВУ.

В начальной области адресного пространства имеется 8 зон, по 8 байтов в каждой зоне. Каждая зона соответствует ВУ с определенным номером: 0...7 – «0»; 8...0F «1»; 10...17 «2»; 18...1F «3»; 20...27 «4»; 28...2F «5». В нашем примере команда RST 5 устанавливает РС = 0028h.

Каждая зона содержит либо всю подпрограмму обслуживания прерывания

(если она содержит не больше 8-ми байтов) или команду JMP ADDR (она занимает 3 байта) ADDR – символическое имя начального адреса подпрограммы обслуживания прерывания.

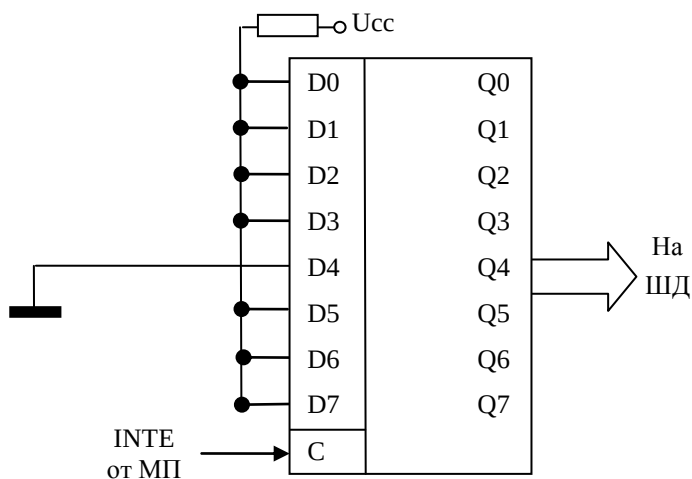


Рис. 5 Регистр ВУ №5 для формирования кода RST 5

Цикл M1 называется особым т.к. 1) Источником команды вызова подпрограммы обслуживания прерывания является внешнее устройство, а не память, как в обычном цикле M1. 2) При приеме каждого байта команды вызова подпрограммы обслуживания прерывания программный счетчик не инкрементируется.

Рассмотрим пример. Пусть прерывания разрешены (РП=1). Внешнее устройство № 5 сформировало код вызова подпрограммы RST 5. Оно посылает сигнал запроса прерывания на вход ЗП (INT) МП, т.е. INT=1. В это время МП выполняет команду STA 0920h (назовем ее текущей):

0806	32	} STA 0920h
0807	20	
0808	09	
0809	47	
080A	55	

МП завершает выполнение текущей команды, после чего входит в особый цикл M1, во время которого принимает команду RST 5. На момент входа в цикл M1 программный счетчик содержит адрес следующей команды PC=0809h. Это адрес команды, которой должно быть передано управление после обработки прерывания. При приеме байта команды вызова обработчика прерываний изменение содержимого программного счетчика блокируется, т.к. в противном случае после обработки прерывания управление будет передано совсем другой команде (команде с адресом 080Ah).

Когда команда RST 5 принята, МП приступает к ее выполнению: содержимое PC 0809h спасается в стеке, а PC устанавливается равным 0028h. Процессор начинает выполнять команды подпрограммы обработки прерывания. При этом PC инкрементируется как обычно. В конце подпрограммы должна быть команда RET, которая восстанавливает содержимое программного счетчика 0809h из стека, и прерванная программа продолжает свою работу.

5 Структурная схема МК SAB 80C515

Структурная схема МК SAB 80C515 приведена на рисунке 6.

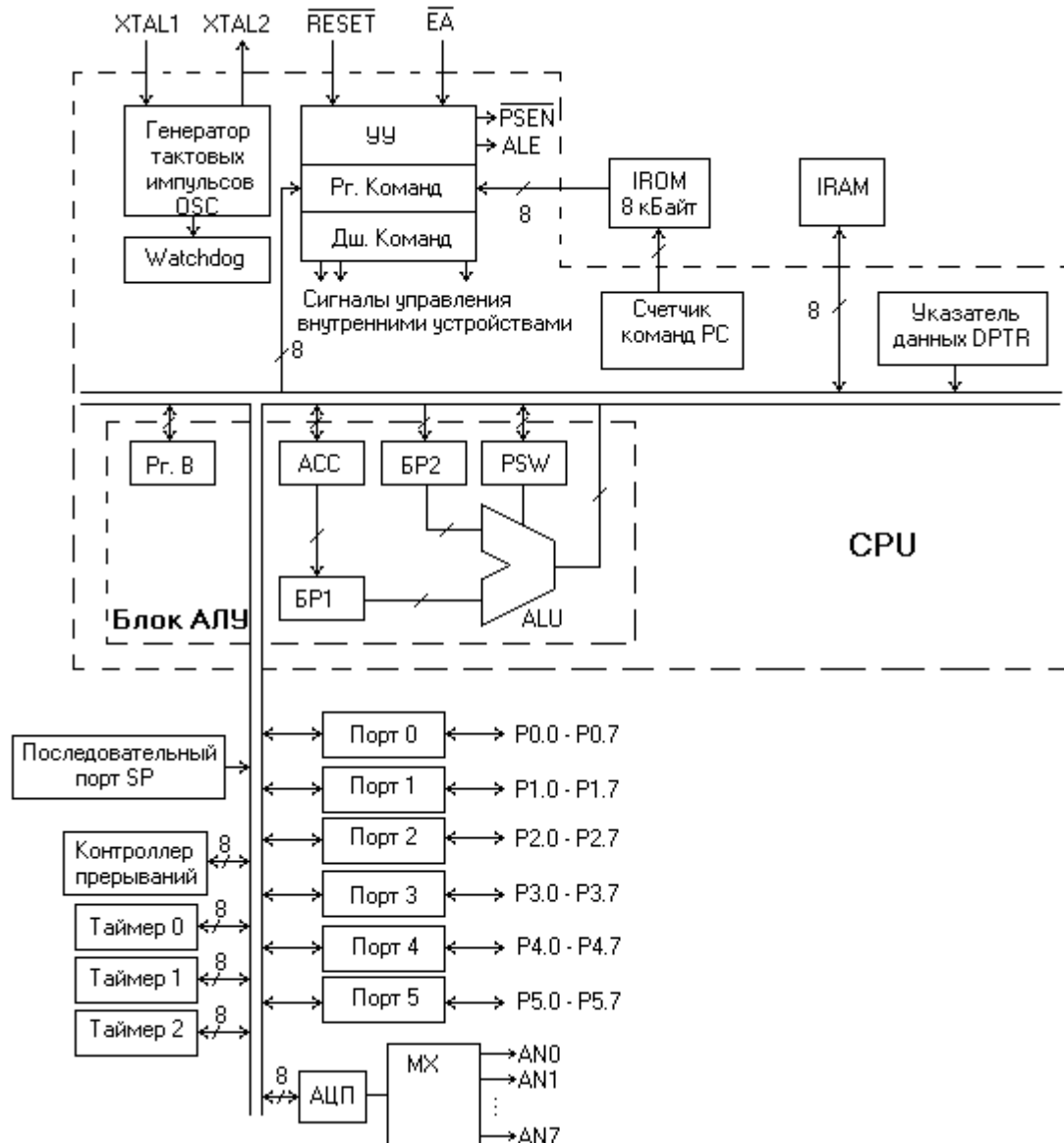


Рис. 6 Структурная схема МК SAB 80C515

Основные характеристики 8-битного МК SAB 80C515 следующие:

- 8-разрядное АЛУ и схемы аппаратной реализации умножения и деления;
- внутреннее ПЗУ (IROM) программ и констант объемом 8 Кбайт;
- возможность расширения ПЗУ за счет подключения внешних микросхем памяти до 64 Кбайт;
- внутреннее ОЗУ (IRAM) данных объемом 256 байт;
- возможность подключения внешнего ОЗУ объемом до 64 Кбайт;
- шесть программируемых портов ввода-вывода (P1 - P5);
- восьмивходовой порт ввода аналоговых сигналов (P6);
- полный дуплексный последовательный порт SP с фиксированной и переменной скоростью обмена;

- три программируемых 16-битных таймера/счетчика;
- 4-канальный блок быстрого ввода-вывода внешних событий, обладающий дополнительными возможностями формирования ШИМ-сигналов;
- 8-канальный аналого-цифровой преобразователь со встроенным блоком программируемых эталонных напряжений;
- сторожевой таймер (WDT);
- два режима пониженного энергопотребления: режим холостого хода и режим микропотребления;
- 4-уровневая система прерываний от 12 источников прерываний;
- внутренний стек глубиной 256 байт.

CPU МК 80C515 представляет собой 8-разрядный микропроцессор с фиксированной разрядностью и списком команд. Взаимодействие отдельных блоков как внутри CPU, так и с различными внутренними периферийными устройствами МК осуществляется по 8-разрядной шине данных. Связь МК с внешними устройствами, например при расширении системы за счет подключения внешней памяти и дополнительных устройств ввода-вывода, осуществляется с помощью мультиплексируемой шины адрес/данные, для организации которой используются линии порта P0 (младший байт адреса и данные) и порта P2 (старший байт адреса).

Арифметико-логический блок предназначен для выполнения арифметических и поразрядных логических операций, а также операций с отдельными битами.

В состав блока АЛУ входят: собственно АЛУ, регистр результата (аккумулятор) ACC, регистр временного хранения операндов BP1 и BP2, дополнительный регистр В, регистр признаков результата (регистр слова состояния) PSW и ПЗУ констант.

Регистры ACC, В и PSW размещаются в блоке регистров специальных функций SFR и являются программно доступными.

Регистры временного хранения BP1 и BP2, предназначены для запоминания операндов на время выполнения операции в АЛУ и программно недоступны.

8-разрядное АЛУ реализовано на базе комбинационного сумматора с последовательным переносом.

8-разрядный регистр В используется в качестве регистра-расширителя аккумулятора при выполнении операций умножения и деления. В других командах регистр В может использоваться как дополнительный регистр блока РОН.

ПЗУ констант содержит коды масок битовых операций и, кроме того, с его помощью обеспечивается выработка корректирующего кода для двоично-десятичного представления результата сложения двоично-десятичных чисел.

При выполнении арифметических и логических команд на выходе АЛУ формируются четыре признака (флага), отражающего значение результата выполненной операции: флаг переноса C, флаг межтетрадного переноса AC, флаг переполнения OV и флаг паритета P (см. рис.)

Формат регистра PSW.

7	6	5	4	3	2	1	0
C	AC	F0	RS1	RS0	OV	F1	P

Флаги C (Carry) и OV (Overflow) используются для выявления признаков пе-

реполнения разрядной сетки при выполнении арифметических операций с беззнаковыми и знаковыми числами соответственно.

Флаг С выполняет функции флага переноса в арифметических командах с числами без знака и функции "булевого аккумулятора" в командах работы с битами. Устанавливается и сбрасывается аппаратно. Система команд МК предусматривает возможность программного доступа к флагу С.

Флаг межтетрадного переноса АС фиксирует межтетрадный перенос при выполнении команд сложения и вычитания. Устанавливается и сбрасывается аппаратно.

С помощью флага переполнения OV контролируется правильность выполнения арифметических операций со знаковыми числами. Флаг OV устанавливается аппаратно, указывая на ошибку выполнения арифметической операции, если результат сложения/вычитания не укладывается в семи битах и старший бит результата не может интерпретироваться как знаковый.

При делении флаг OV аппаратно сбрасывается, а в случае деления на 0 - устанавливается. При умножении флаг OV аппаратно устанавливается, если результат превышает значения 255.

Использование флагов С и OV определяет пользователь.

Флаг паритета Р, устанавливаемый/сбрасываемый аппаратно, предназначен для указания нечетного/четного количества единичных разрядов аккумулятора. В 9-разрядном слове, состоящем из восьми разрядов аккумулятора и бита Р, всегда содержится четное число единичных битов.

Кроме флаговых бит результата, регистр PSW содержит два флага состояния F0 и F1, определяемых (программируемых) пользователем, и два бита указателя RS1 и RS0, используемых в качестве программируемого указателя рабочего банка регистров общего назначения. Код, определяемый битами RS1 и RS0, однозначно определяет номер рабочего (используемого) банка РОН: 00 - банк 0, 01 - банк 1, 10 - банк 2, 11 - банк 3.

В отличие от большинства стандартных МП, CPU МК SAB80C515 содержит в своем составе четыре банка рабочих регистров по восемь регистров общего назначения (R0 - R7) в каждом. Банки регистров размещаются в блоке регистров специального назначения. Для указания адреса регистров R0 - R7 блока РОН допускается использовать как прямую адресацию, так и прямую регистровую. Команды с прямой регистровой адресацией R0 - R7 всегда короче команд с прямой адресацией. Наличие четырех банков РОН обеспечивает быстрое переключение контекста программ при использовании ограниченного числа подпрограмм или обработчиков прерываний. По умолчанию после инициализации АЛУ взаимодействует с банком 0.

К программно доступным регистрам, поддерживающим работу CPU, относятся еще три регистра блока специальных функций: 16-разрядный счетчик команд PC, 8-разрядный указатель стека SP и 16-разрядный регистр-указатель данных DPTR.

16-разрядный счетчик команд PC предназначен для формирования текущего адреса программной памяти (внутренней и внешней). После чтения (выборки)

очередного байта команды из программной памяти содержимое РС инкрементируется на 1.

С помощью 16-разрядного указателя данных DPTR обеспечивается косвенная адресация операндов при обращении к ячейкам внешней памяти данных и считывание констант из программной памяти. Механизм обращения к внешней памяти программ и внешней памяти данных рассматривается ниже.

8-разрядные регистры SFR DPL и DPH программно доступны. При отсутствии необходимости обращения к внешней памяти они могут использоваться как два отдельных регистра PОН.

В МК SAB 80C515 используется стек, моделируемый во внутренней памяти данных IRAM. 8-битный указатель стека SP, в качестве которого используется регистр блока SFR, может адресовать любую область ОЗУ. Глубина стека лимитируется только объемом IRAM и не может превышать 256 байт. Однако в реальных системах фактическая глубина стека определяется непосредственно пользователем, который должен позаботиться о том, чтобы стек не перекрывал другие области данных программы, так как при загрузке стека эти данные будут испорчены. При загрузке данных в стек сначала осуществляется инкремент SP, а затем выполняется собственно запись в стек. При чтении стека данные извлекаются из стека, а затем производится декремент SP.

6 Организация памяти МК SAB 80C515

В отличие от процессоров персональных компьютеров, которые, как правило, имеют традиционную структуру фон Неймана, предполагающую расположение команд и данных в едином адресном пространстве памяти, микроконтроллеры семейства MCS-51, в том числе и SAB 80C515 реализованы в соответствии с Гарвардской архитектурой. Они имеют отдельные (физически и логически) адресные пространства памяти – память программ и память данных, которые выполняют различные функции. Для доступа к указанным областям памяти используются отличные механизмы адресации и различные управляющие сигналы: $\overline{PSEN}$ для доступа к ПЗУ и $\overline{RD}, \overline{WR}$ при обращении к ОЗУ.

Полное пространство памяти МК SAB 80C515 показано на рисунке 7.

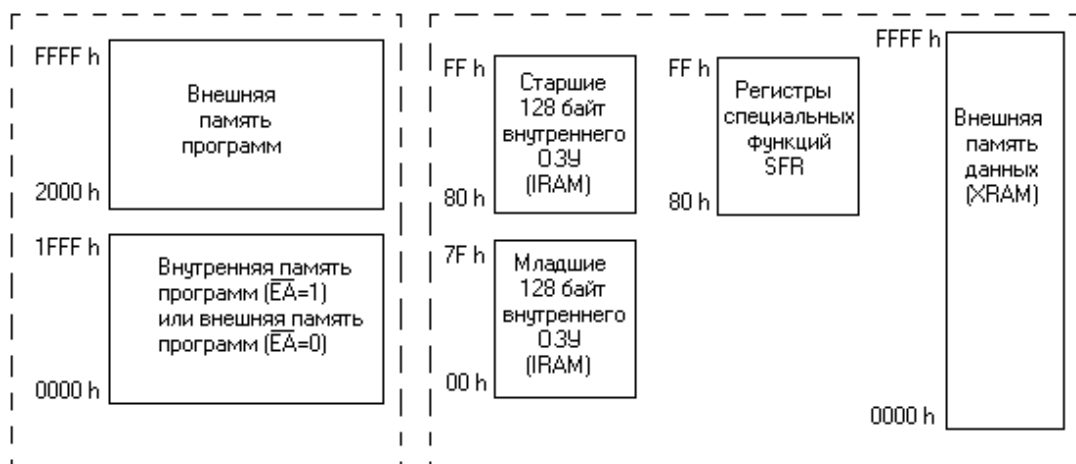


Рис. 7 Полное пространство памяти МК SAB 80C515

Память программ с байтовой организацией и объемом до 64 Кбайт доступна только по чтению. Младшие 8 Кбайт программной памяти МК SAB 80C515 размещаются на кристалле МК. Как правило, элементы программной памяти адресуются с помощью 16-битного счетчика команд PC, но в некоторых командах (при выборке констант) могут косвенно адресоваться с использованием 16-разрядного указателя DPTR (команды вида $MOV\ C\ A, @+DPTR$).

Доступ к внешней памяти программ (ВПП) разрешен либо если активизирован вход $\overline{EA}$ отключения внутреннего (резидентного) ПЗУ ($\overline{EA} = 0$), либо, если содержимое счетчика команд PC превышает значение 1FFFh, т.е. происходит обращение к программной памяти вне первых 8 Кбайт ее адресного пространства.

Существующие различия в механизме доступа к ВПП и внешней памяти данных (ВПД) учитываются на аппаратном уровне: при доступе к ВПД формируются входные сигналы $\overline{RD}$, $\overline{WR}$, а при доступе к ВПП - сигнал $\overline{PSEN}$ (Program Store ENable).

Сигнал $\overline{PSEN}$, выполняющий функцию строба чтения ВПП, активизируется только при обращении к ВПП и остается пассивным при выборке команд из внутренней программной памяти.

При доступе к ВПП для организации внешних шин адреса и данных используются линии портов P0 и P2, при этом с помощью линий порта P0 реализуется мультиплексированная внешняя шина адрес/данные.

При доступе к ВПП 16-разрядный адрес формируется на линиях порта P0 (младший байт) и порта P2 (старший байт). Для демultipлексирования адреса на выходах порта P0 предназначен выходной сигнал ALE (Address Latch Enable), с помощью которого младший байт адреса фиксируется во внешнем регистре адреса.

На рисунке 8 показана функциональная схема включения МК SAB и внешней памяти программ. Временные диаграммы сигналов при работе МК с внешней памятью программ приведены на рисунке 9.

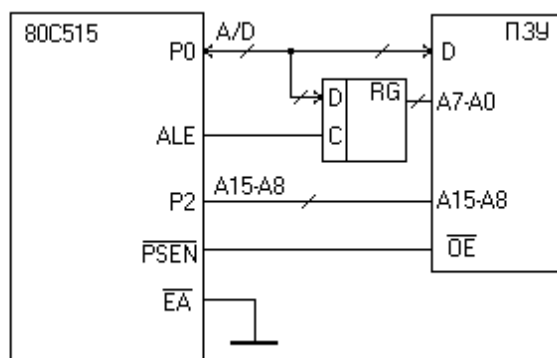


Рис. 8 Функциональная схема включения МК SAB и внешней памяти

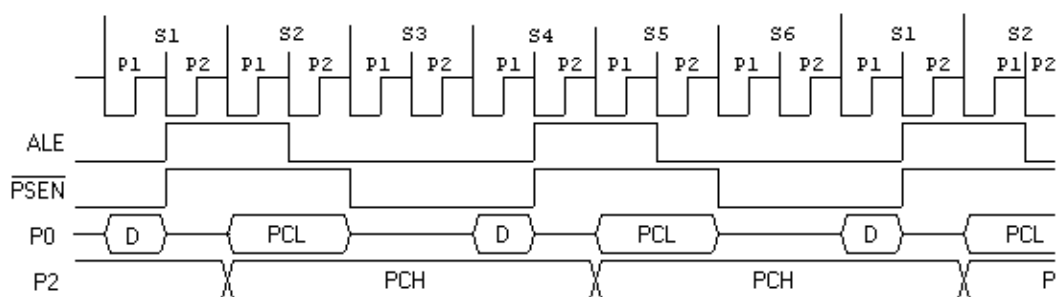


Рис.9 Временные диаграммы сигналов при работе МК с внешней памятью

7 Внутренняя память данных

Внутренняя память данных микроконтроллеров семейства MCS-51 в адресном пространстве занимает 256 байт. На кристалле размещаются три физически разделенных блока внутренней памяти - блок младших 128 байт ОЗУ с адресами 00 - 7Fh, блок старших 128 байт ОЗУ с адресами 80h - FFh и блок регистров специальных функций SFR, размещенный по адресам 80h - FFh. Так как старшие 128 байт внутренней памяти данных и область SFR имеют одинаковое адресное пространство, для их различия используют разные способы адресации: для обращения к регистрам специальных функций можно использовать только прямую адресацию, а доступ к старшим 128 байтам ОЗУ возможен только с использованием косвенной адресации, при этом младшие 128 байт ОЗУ могут адресоваться и прямо, и косвенно. Все обращения к ячейкам внутренней памяти данных всегда осуществляются с использованием короткого 8-разрядного адреса.

Младшие 128 байт ОЗУ в отличие от других областей внутренней памяти данных в свою очередь подразделяются на три области, обладающие специальными свойствами (рис.).

Первые (младшие) 32 байта внутреннего ОЗУ с адресами 00 - 1Fh сгруппированы в четыре банка по 8 регистров общего назначения с именами R0 - R7 в каждом. Выбор рабочего (текущего) банка регистров осуществляется программно с помощью управляемых бит RS0 и RS1 слова состояния PSW. Отличительной особенностью блока рабочих регистров является возможность использования, наряду с прямой и косвенной адресацией, прямой регистровой адресации. Наличие четырех банков рабочих регистров обеспечивает быстрое переключение контекста программы при использовании ограниченного числа подпрограмм или обработчиков внешних прерываний. После инициализации активен банк 0.

Следующие 16 байт внутреннего ОЗУ с адресами 20h - 2Fh образуют область ОЗУ, зарезервированную для хранения 128 программно-управляемых пользовательских флагов. Каждый из указанных флагов имеет прямой 8-битовый адрес в диапазоне 00 - 7Fh. Команды побитовой обработки, работающие с этими флагами, допускают только прямую адресацию бит (косвенная адресация бит невозможна). Естественно, при необходимости область пользовательских флагов можно использовать как область ОЗУ, допускающую прямую и косвенную адресации 8-битных ячеек памяти.

Оставшиеся 80 байт из области младших 128 байт ОЗУ и вся область старших 128 байт ОЗУ - это область обычного ОЗУ, использующего косвенную адресацию всех данных и допускающего прямую адресацию 80 байт из области младших 128 байт.

Система команд МК SAB 80C515.

МК SAB 80C515 оперирует с операндами четырех типов: битами, тетрадами (4-битными словами), байтами и 16-битными словами, при этом большинство операндов команд являются байтами.

Для их адресации используются следующие типы адресации: неявная регистровая, прямая, непосредственная, косвенная и разновидность косвенной $\square$ базово-индексная адресация. Указанные способы адресации обеспечивают обращение к операндам-источникам. При обращении к операндам-приемникам непосредственная и базово-индексная адресации не используются.

Неявная адресация, т.е. способ адресации, в котором адрес операнда не указывается, а сам операнд неявно адресуется кодом команды, используется в большинстве команд, оперирующих с аккумулятором. Кроме того, неявная адресация используется для адресации регистра В в командах умножения и деления, а также для обращения к флагу С в командах битовых операций, при обращениях к регистру DPTR и в командах работы со стеком.

Регистровая адресация применяется только для адресации 8 регистров R0 $\square$ R7 текущего (рабочего) банка регистров. Так как эти регистры принадлежат пространству младших 128 байт внутреннего ОЗУ, то к ним также можно обратиться, используя прямую и косвенную адресацию через регистр Ri ($i = 0, 1$) текущего банка регистров.

Прямая адресация, т.е. адресация с указанием 8-битного адреса операнда в коде команды используется для задания адресов младших 128 байт IRAM и адресов регистров блока SFR.

Косвенная адресация используется для обращения к операндам внутренней памяти данных IRAM через регистр Ri ($i = 0, 1$) и в командах пересылок из внешней/во внешнюю память данных через регистр Ri или DPTR.

Для адресации бит используется только прямая адресация, т.е. адресация, когда в команде содержится 8-разрядный адрес бита. Косвенная адресация бит невозможна.

При обращении к внешней памяти программ для считывания констант используется базово-индексная адресация, обеспечивающая формирование адреса выбираемого байта путем сложения 8-битного беззнакового содержимого аккумулятора с содержимым 16-битного базового регистра, который может быть либо DPTR либо PC.

Операндом команд с непосредственной адресацией может быть только источник непосредственных данных. Непосредственные данные $\square$ это константы или прямые адреса, для представления которых используется второй или второй и третий байт команды.

Синтаксис большинства команд ассемблерного языка состоит из мнемонического обозначения функций команды, вслед за которым могут размещаться опе-

ранды, указывающие методы адресации и типы данных. Для обозначения типов данных и способов адресации в мнемониках команд применяются следующие обозначения:

Rn □ регистры R0 □ R7 текущего (рабочего) банка регистров;

bit □ прямо адресуемый бит, находящийся в IRAM или регистре специальных функций;

#data или #d □ 8-битная константа, входящая в состав команды;

#data16 или #d16 □ 16-битная константа, входящая в состав команды;

direct □ 8-битный адрес младших 128 байт IRAM или адрес регистра блока SFR (старшие 128 байт внутренней памяти SFR);

@Ri □ обозначение косвенной адресации через регистр Ri (i = 0, 1);

@DPTR □ обозначение косвенной адресации через регистр DPTR;

addr16 □ 16-битный адрес безусловной передачи управления;

addr11 □ 11-битный адрес безусловной передачи управления в пределах 2-Кбайтного блока памяти программ, содержащем код команды блока передачи управления;

rel □ относительный адрес в командах условной передачи управления и командах короткого безусловного перехода. Rel □ 8-битное значение смещения со знаком, которое суммируется с содержимым PC для формирования адреса перехода;

/bit □ слэш перед операндом в языке ассемблера указывает, что используется инвертированное значение бита-источника, хотя сам бит источника не инвертируется.

Машинные коды команд МК представляются одним, двумя или тремя байтами. Команды выполняются за один или два машинных цикла. Исключение составляют команды умножения и деления, длительность исполнения которых равна 4 машинных цикла.

Систему команд МК удобно представить в виде нескольких групп:

- команды пересылок;
- команды арифметических и логических операций;
- команды битового процессора;
- команды передачи управления.

7.1 Группа команд пересылок (передачи данных)

Команды этой группы не модифицируют флаги результата. Исключение составляет команда загрузки PSW и аккумулятора. При загрузке PSW модифицируются все флаги, а при загрузке ACC □ флаг паритета.

Мнемокод	КОП		Б	Ц	Функция
MOV A, Rn	1110	1rrr	1	1	(A) := (Rn)
MOV A, direct	1110	0101	2	1	(A) := (direct)
MOV A, @Ri	1110	011i	1	1	(A) := ((Ri))
MOV A, #data	1110	0100	2	1	(A) := #data
MOV Rn, A	1111	1rrr	1	1	(Rn) := (A)

MOV Rn, direct	1010	1rrr	2	2	(Rn) := (direct)
MOV Rn, #data	0111	1rrr	2	1	(Rn) := #data
MOV direct, A	1111	0101	2	1	(direct) := (A)
MOV direct, Rn	1000	1rrr	2	2	(direct) := (Rn)
MOV direct, direct	1000	0101	3	2	(direct) := (direct)
MOV direct, @Ri	1000	011i	2	2	(direct) := ((Ri))
MOV direct, #data	0111	0101	3	2	(direct) := #data
MOV @Ri, A	1111	011i	1	1	((Ri)) := (A)
MOV @Ri, direct	1010	011i	2	1	((Ri)) := (direct)
MOV @Ri, #data	0111	011i	2	1	((Ri)) := #data
MOV DPTR, #data16	1001	0000	3	2	((DPTR)) := #data16
MOVC A, @A+DPTR	1001	0011	1	2	(A) := ((A)+(DPTR)) Чтение внешней памяти программ
MOVC A, @A+PC	1000	0011	1	2	(A) := ((A)+(PC)) Чтение внешней памяти программ
MOVX A, @Ri	1110	001i	1	2	(A) := ((Ri)) Чтение внешней памяти данных
MOVX @Ri, A	1111	001i	1	2	((Ri)) := (A) Запись внешней памяти данных
MOVX A, @DPTR	1110	0000	1	2	(A) := ((DPTR)) Чтение внешней памяти данных
MOVX @DPTR, A	1111	0000	1	2	((DPTR)) := (A) Запись внешней памяти данных
PUSH direct	1100	0000	2	2	(SP) := (SP)+1 ((SP)) := (direct) запись в стек
POP direct	1101	0000	2	2	(direct) := ((SP)) (SP) := (SP)-1 чтение из стека
XCH A, Rn	1100	1rrr	1	1	(A) <-> (Rn)
XCH A, direct	1100	0101	2	1	(A) <-> (direct)
XCH A, @Ri	1100	011i	1	1	(A) <-> ((Ri))
XCHD A, @Ri	1101	011i	1	1	(A ₃₋₀) <-> ((Ri) ₃₋₀)
SWAP A	1100	0100	1	1	Обмен тетрад аккумулятора

7.2 Группа команд арифметических и логических операций

При выполнении большинства арифметических и логических команд аккумулятор является источником одного из операндов и приемником результата. Вторым операндом может быть регистр Ri или ячейка внутренней памяти IRAM.

Многие команды этой группы модифицируют признаки результата в регистре

PSW. Команды других групп, как правило, на флаги результата не влияют. В таблице представлены команды, воздействующие на флаги регистра PSW.

Команда	Флаг			Команда	Флаг		
	C	OV	AC		C	OV	AC
ADD	+	+	+	ANL C, bit	+	☐	☐
ADDC	+	+	+	ANL C, /bit	+	☐	☐
SUBB	+	+	+	ORL C, bit	+	☐	☐
MUL	0	+	☐	ORL C, /bit	+	☐	☐
DIV	0	+	☐	MOV C, bit	+	☐	☐
DA	+	☐	☐	CLR C	0	☐	☐
RRC	+	☐	☐	CPL C	+	☐	☐
RLC	+	☐	☐	CJNE	+	☐	☐
SETB C	1	☐	☐	POP PSW	+	+	+

Выполнение операций сложения и вычитания чисел со знаком, как и одноименных операций с беззнаковыми числами, также выполняется в АЛУ. Однако, правильность выполнения в этом случае необходимо программно контролировать с помощью флага переполнения OV регистра PSW. Установленный флаг OV указывает на ошибочный результат (отрицательное значение при выполнении вычитания отрицательного числа из положительного или положительного значения при вычитании положительного числа из отрицательного).

Операции умножения и деления реализуются встроенным аппаратным блоком умножения/деления CPU с использованием аккумулятора A и регистра B. В команде умножения MUL AB содержимое ACC умножается на содержимое регистра B и 16-битный результат размещается в регистре B (старший байт) и в аккумуляторе (младший байт). При выполнении команды деления DIV AB реализуется деление 8-битового беззнакового числа в аккумуляторе на 8-битовое беззнаковое число в регистре B. Результат операции заносится в аккумулятор (целая часть частного) и в регистр B (остаток).

Мнемокод	КОП		Б	Ц	Функции
ADD A, Rn	0010	1rrr	1	1	$(A) := (A) + (Rn)$
ADD A, direct	0010	0101	2	1	$(A) := (A) + (\text{direct})$
ADD A, @Ri	0010	011i	1	1	$(A) := (A) + ((Ri))$
ADD A, #data	0010	0100	2	1	$(A) := (A) + \#data$
ADDC A, Rn	0011	1rrr	1	1	$(A) := (A) + (C) + (Rn)$
ADDC A, direct	0011	0101	2	1	$(A) := (A) + (C) + (\text{direct})$
ADDC A, @Ri	0011	011i	1	1	$(A) := (A) + (C) + ((Ri))$
ADDC A, #data	0011	0100	2	1	$(A) := (A) + (C) + \#data$
SUBB A, Rn	0011	1rrr	1	1	$(A) := (A) - (C) - (Rn)$
SUBB A, direct	0011	0101	2	1	$(A) := (A) - (C) - (\text{direct})$
SUBB A, @Ri	0011	011i	1	1	$(A) := (A) - (C) - ((Ri))$
SUBB A, #data	0011	0100	2	1	$(A) := (A) - (C) - \#data$
DA A	1101	0100	1	1	Десятичная коррекция содержи-

Мнемокод	КОП		Б	Ц	Функции
					мого аккумулятора
INC A	0000	0100	1	1	$(A) := (A) + 1$
INC Rn	0000	1rrr	1	1	$(Rn) := (Rn) + 1$
INC direct	0000	0101	2	1	$(direct) := (direct) + 1$
INC @Ri	0000	011i	1	1	$((Ri)) := ((Ri)) + 1$
INC DPTR	1010	0011	1	2	$(DPTR) := (DPTR) + 1$
DEC A	0001	0100	1	1	$(A) := (A) - 1$
DEC Rn	0001	1rrr	1	1	$(Rn) := (Rn) - 1$
DEC direct	0001	0101	2	1	$(direct) := (direct) - 1$
DEC @Ri	0001	011i	1	1	$((Ri)) := ((Ri)) - 1$
MUL AB	1010	0100	1	4	$(B)_{15-8}, (A)_{7-0} := (A) * (B)$
DIV AB	1000	0100	1	4	$(A)_{\text{цел}}, (B)_{\text{ост}} := (A) * (B)$
ANL A, Rn	0101	1rrr	1	1	$(A) := (A) \& (Rn)$
ANL A, direct	0101	0101	2	1	$(A) := (A) \& (direct)$
ANL A, #data	0101	0100	2	1	$(A) := (A) \& \#data$
ANL A, @Ri	0101	011i	1	1	$(A) := (A) \& ((Ri))$
ANL direct, A	0101	0010	2	1	$(direct) := (direct) \& (A)$
ANL direct, #data	0101	0011	3	3	$(direct) := (direct) \& \#data$
ORL A, Rn	0100	1rrr	1	1	$(A) := (A) \vee (Rn)$
ORL A, direct	0100	0101	2	1	$(A) := (A) \vee (direct)$
ORL A, @Ri	0100	011i	1	1	$(A) := (A) \vee ((Ri))$
ORL A, #data	0100	0100	2	1	$(A) := (A) \vee \#data$
ORL direct, A	0100	0010	2	1	$(direct) := (direct) \vee (A)$
ORL direct, #data	0100	0011	3	3	$(direct) := (direct) \vee \#data$
XRL A, Rn	0110	1rrr	1	1	$(A) := (A) \text{ XOR } (Rn)$
XRL A, direct	0110	0101	2	1	$(A) := (A) \text{ XOR } (direct)$
XRL A, @Ri	0110	011i	1	1	$(A) := (A) \text{ XOR } ((Ri))$
XRL A, #data	0110	0100	2	1	$(A) := (A) \text{ XOR } \#data$
XRL direct, A	0110	0010	2	1	$(direct) := (direct) \text{ XOR } (A)$
XRL direct, #data	0110	0011	3	3	$(direct) := (direct) \text{ XOR } \#data$
CLR A	1110	0100	1	1	Обнуление аккумулятора $(A) := 0$
CPL A	1111	0100	1	1	Инверсия аккумулятора
RL A	0010	0011	1	1	Сдвиг аккумулятора влево циклический
RLC A	0011	0011	1	1	Сдвиг аккумулятора влево через перенос
RR A	0000	0011	1	1	Сдвиг аккумулятора вправо циклический
RRC A	0001	0011	1	1	Сдвиг аккумулятора вправо через перенос
SWAP A	1100	0100	1	1	Циклический сдвиг АСС на 4 разряда $(A_{3-0}) \leftrightarrow (A_{7-4})$

7.3 Команды битового процессора

Битовый процессор является частью архитектуры МК, и его можно рассматривать как независимый процессор побитовой обработки. Битовый процессор выполняет свой набор команд, имеет побитно адресуемое ОЗУ и свой ввод-вывод.

Команды, оперирующие с битами, обеспечивают прямую адресацию 128 бит в шестнадцати ячейках IRAM (ячейки с адресами 20h □ 2Fh) и прямую адресацию бит регистров SFR с адресами, кратными 8.

Каждый из отдельно адресуемых битов может быть установлен в 1, сброшен в 0, инвертирован и проверен. Между любым прямо адресуемым битом и флагом переноса C может быть произведены логические операции И, ИЛИ с запоминанием результата во флаге C. Любой бит может быть перезаписан в/из флага C.

Команда условной передачи управления битового процессора реализуют переходы, если проверяемый бит установлен (со сбросом или без сброса этого бита) или если проверяемый бит не установлен.

Мнемокод	КОП		Б	Ц	Функции
MOV C, bit	1010	0010	2	1	(C) := bit
MOV bit, C	1001	0010	2	2	(bit) := (C)
CLR C	1100	0011	1	1	(C) := 0
CLR bit	1100	0010	2	1	(bit) := 0
SETB C	1101	0011	1	1	(C) := 1
SETB bit	1101	0010	2	1	(bit) := 1
CPL C	1011	0011	1	1	
CPL bit	1011	0010	2	1	
ANL C, bit	1000	0010	2	2	
ANL C, /bit	1011	0000	2	2	
ORL C, bit	0111	0000	2	2	
ORL C, /bit	1010	0000	2	2	
JB bit, rel	0010	0000	3	2	Переход, если бит установлен (PC) := (PC) + 3 if (bit) = 1 then (PC) := (PC) + rel
JBC bit, rel	0001	0000	3	2	Переход, если бит установлен, и сброс этого бита (PC) := (PC) + 3 if (bit) = 1 then (bit) := 0, (PC) := (PC) + rel
JNB bit, rel	0011	0000	3	2	Переход, если бит не установлен (PC) := (PC) + 3 if (bit) = 0 then (PC) := (PC) + rel
JC rel	0100	0000	2	2	Переход, если перенос установлен (PC) := (PC) + 2 if (C) = 1 then (PC) := (PC) + rel
JNC rel	0101	0000	2	2	Переход, если перенос не установлен (PC) := (PC) + 2 if (C) = 0 then (PC) := (PC) + rel

7.4 Команды ветвления и передачи управления

Команды передачи управления бывают двух типов: безусловной и условной передачи.

Адресное пространство памяти программ МК не имеет страничной организа-

ции, что позволяет свободно перемещать фрагменты программ внутри адресного пространства.

Мнемокод	КОП	Б	Ц	Функции
LJMP addr16	0000	0010	3 2	(PC) := addr ₁₅₋₀
AJMP addr11	a ₁₀ a ₉ a ₈ 0	0001	2 2	(PC) := (PC) + 2 (PC) ₁₀₋₀ := addr ₁₁₋₀
SJMP rel	1000	0000	2 2	(PC) := (PC) + 2 (PC) := (PC) + rel
JMP @A + DPTR	0111	0011	1 2	Косвенный переход: (PC) := (A) + (DPTR)
LCALL addr16	0001	0010	3 2	(PC) := (PC) + 3 (SP) := (SP) + 1 ((SP)) := (PC ₇₋₀) (SP) := (SP) + 1 ((SP)) := (PC ₁₅₋₈)(PC) := addr ₁₅₋₀
ACALL addr11	a ₁₀ a ₉ a ₈ 1	0001	2 2	(PC) := (PC) + 2 (SP) := (SP) + 1 ((SP)) := (PC ₇₋₀) (SP) := (SP) + 1 ((SP)) := (PC ₁₅₋₈)(PC ₁₀₋₀) := addr ₁₀₋₀
RET	0010	0010	1 2	(PC ₁₅₋₈) := ((SP)) (SP) := (SP) – 1 (PC ₇₋₀) := ((SP)) (SP) := (SP) – 1
RETI	0011	0010	1 2	(PC ₁₅₋₈) := ((SP)) (SP) := (SP) – 1 (PC ₇₋₀) := ((SP)) (SP) := (SP) – 1

Все команды условного перехода (табл.) реализуют только короткие относительные переходы, позволяющие передать управление в пределах -128 □ 128 байт относительно адреса команды, следующей по порядку за командой условного перехода.

Мнемоника	КОП	Б	Ц	Функции
JZ rel	0110	0000	2 2	(PC) := (PC) + 2 if (A) = 0 then (PC) := (PC) + rel
JNZ rel	0111	0000	2 2	(PC) := (PC) + 2 if (A) ≠ 0 then (PC) := (PC) + rel
JC rel	0100	0000	2 2	(PC) := (PC) + 2 if (C) = 1 then (PC) := (PC) + rel
JNC rel	0101	0000	2 2	(PC) := (PC) + 2 if (C) = 0 then (PC) := (PC) + rel
DJNZ Rn, rel	1101	1rrr	2 2	(PC) := (PC) + 2 (Rn) := (Rn) – 1 if (Rn) > 0 или (Rn) < 0 then (PC) := (PC) + rel
DJNZ direct, rel	1101	0101	2 2	(PC) := (PC) + 2 (direct) := (direct) – 1 if (direct) > 0 или (direct) < 0 then (PC) := (PC) + rel
CJNE A, direct, rel	1011	0101	3 2	(PC) := (PC) + 3 if (A) > < (direct) then (PC) := (PC) + rel if (A) < (direct) then (C) := 1 else (C) := 0
CJNE A, #data, rel	1011	0100	3 2	(PC) := (PC) + 3 if (A) > < #data then (PC) := (PC) + rel if (A) < #data then (C) := 1 else (C) := 0
CJNE Rn, #data, rel	1011	1rrr	3 2	(PC) := (PC) + 3 if (Rn) > < #data then (PC) := (PC) + rel if (Rn) < #data then (C) := 1 else (C) := 0
CJNE @Ri, #data, rel	1011	011i	3 2	(PC) := (PC) + 3 if ((Ri)) > < #data then (PC) := (PC) + rel if ((Ri)) < #data then (C) := 1 else (C) := 0

Приведем два примера программирования на языке ассемблера МК SAB80C515.

Пример 1. Написать программу вычисления логического выражения $F = \neg(x * y + z)$,

знаки « / », « * », « v » и « + » обозначают логические операции отрицания,

конъюнкции, дизъюнкции и сложения по модулю два соответственно. Пусть операнды x, y, z и бит промежуточного результата (buf) находятся в младших разрядах (0,1,2,3) ячейки с адресом 23h, а бит с значением результата (rez) – в старшем разряде ячейки с адресом 2Dh.

В программе будем использовать символические имена для всех операндов и результатов, поэтому перед командами программы необходимо записать псевдокоманды вида equ. Кроме того, необходимо учесть то обстоятельство, что в битовых командах МК нет

операции сложения по модулю два, поэтому исходное выражение придется видоизменить:

$$F = x * y * z \vee (/x * y) * /z .$$

```

x:  equ  18h
y:  equ  19h
z:  equ  1ah
buf: equ  1bh
rez: equ  6fh
      org  8000h
; программа вычисления логического выражения
      mov  c, y    ; y → c
      anl  c, /x    ; /x * y → c
      anl  c, z    ; /x * y * z → c
      mov  buf, c   ; /x * y * z → buf
      mov  c, y    ; y → c
      anl  c, /x    ; /x * y → c
      cpl  c        ; /(x * y) → c
      anl  c, /z    ; /(x * y) * /z → c
      orl  c, buf   ; F → c
      mov  rez, c   ; F → rez
      ret

```

Пример 2. Написать программу заполнения 16 ячеек верхней половины резидентной (встроенной) памяти данных (РПД) с адресами 0A0h .. 0AFh последовательностью байтов 0FFh .. 0F0h соответственно.

Предварительно отметим, что доступ к ячейкам памяти в указанной области осуществляется с использованием косвенной адресации. В качестве указателя адреса используем регистр R0.

Программа имеет следующий вид:

```

      Org  8100h
      Mov  R0, #0A0h; HA → R0
      Mov  A, 0FFh   ; Нач. значение  A →
m1:   mov  @R0, A
      inc  R0
      dec  A
      cjne A, #0EFh, m1
      ret

```

Контрольные вопросы

1. В чем заключаются отличия особого цикла M1, выполняемого при обработке прерываний, от стандартного?

2. Почему в особом цикле M1 не инкрементируется программный счетчик?

3. Каково назначение и разрядность программно-управляемых регистров процессора?

4. Задано следующее исходное состояние регистров и памяти (в 16-ричном коде)

A=80 F=43 B=80 C=33 D=80 E=32 H=80 L=20 SP=87D0 PC=8006

(8020)=38 (8021)=00 (87D0)=30 (87D1)=40

Нарисуйте и поясните временные диаграммы выполнения одной из следующих команд:

CNZ 102h XTHL LHLD 8020h INR M STA 8030h IN 20h POP B

5. В аккумуляторе A находится код 3E, в регистре признаков F – код 02. Каково будет состояние A и F после выполнения одной из следующих команд:

AD1 3Fh SUI 40h ANI 7XRI 5Eh ORI 25h CPI 77h RAL RRC

6. Предложите несколько вариантов взаимного обмена содержимого регистровых пар B-C и D-E. Оцените быстродействие (в тактах) предложенных вариантов.

7. Составьте программу для определения паритета находящегося в аккумуляторе кода.

8. Предложите 4 варианта записи двухбайтового числа из ячеек 8030, 8031 в ячейки 8050, 8051 соответственно. Оцените быстродействие (в тактах) предложенных вариантов.

9. Объясните назначение сигналов ALE, PSEN/, RD/, WR/; через какие внешние выводы МК передаются эти сигналы.

10. Пусть R0 = 0A5h и выполняется одна из команд: а) MOV A,@R0; б) MOVX A,@R0; к какой области памяти будет происходить обращение в каждом случае.

11. Программа осуществляет вычисление логического выражения, результат записывается в ячейку с символическим именем res. Определить местоположение результата в РПД, если используется одна из следующих псевдокоманд: а) res: equ 5Fh; б) res: equ 3Ah; в) res: equ 62h; г) res: equ 1Fh; д) res: equ 6Eh; е) res: equ 7Fh.

12. В памяти данных МК имеются 3 области с одинаковым диапазоном адресов. О каком диапазоне идет речь и с помощью каких команд осуществляется обращение в каждую область.

ПРИЛОЖЕНИЕ А

Таблица А.1

Система команд микропроцессора

1	2	3	F	t	1	2	3	F	t
MOV	R1, R2	R1=R2	-	5	ANA	R	$A=A \wedge R$	×	4
MOV	R, M	R=M	-	4, 3	ANA	M	$A=A \wedge M$	×	4, 3
MOV	M, R	M=R	-	4, 3	ANI	*	$A=A \wedge *$	×	4, 3
MVI	R, *	R=*	-	4, 3	XRA	R	$A=A \oplus R$	×	4
MVI	M, *	M=*	-	4, 3, 3	XRA	M	$A=A \oplus M$	×	4, 3
LXI	RP, □ **	RP=**	-	4, 3, 3	XRI	*	$A=A \oplus *$	×	4, 3
LDA	**	$A=<*>$	-	4, 3, 3, 3	ORA	R	$A=A \vee R$	×	4
STA	**	$<*>=A$	-	4, 3, 3, 3	ORA	M	$A=A \vee M$	×	4, 3
LHLD	**	$L=<*>$, $H=<*>+1$	-	4, 3, 3, 3, 3	ORI	*	$A=A \vee *$	×	4, 3
SHLD	**	$<*>=L$, $<*>+1>=H$	-	4, 3, 3, 3, 3	CMP	R	$A-R$	+	4
LDAX	RP	$A=<RP>$	-	4, 3	CMP	M	$A-M$	+	4, 3
STAX	RP	$<RP>=A$	-	4, 3	CPI	*	$A-*$	+	4, 3
XCHG		$HL \leftrightarrow DE$	-	4	RLC		$\underline{A} : Y=A.7, A.0=A.7$	Y	4
SPHL		$SP=HL$	-	5	RRC		$\underline{A} : Y=A.0, A.7=A.0$	Y	4
PCHL		$PC=HL$	-	5	RAL		$\underline{A} : A.0=Y, Y=A.7$	Y	4
ADD	R	$A=A+R$	+	4	RAR		$\underline{A} : A.7=Y, Y=A.0$	Y	4
ADD	M	$A=A+M$	+	4, 3	CMA		$A = \overline{A}$	-	4
ADI	*	$A=A+*$	+	4, 3	CMC		$Y = \overline{Y}$	Y	4
ADC	R	$A=A+R+Y$	+	4	STC		$Y=1$	Y	4
ADC	M	$A=A+M+Y$	+	4, 3	JMP	**	$PC=**$	-	4, 3, 3
ACI	*	$A=A++Y$	+	4, 3	J??	**	if (??) JMP	-	4, 3, 3
SUB	R	$A=A-R$	+	4	CALL	**	$<SP-1>=PC, PC=**$	-	5, 3, 3, 3, 3
SUB	M	$A=A-M$	+	4, 3	C??	**	if (??) CALL	-	5, 3, 3, (3, 3)
SUI	*	$A=A-*$	+	4, 3	RET		$PC=<SP>$	-	4, 3, 3
SBB	R	$A=A-R-Y$	+	4	R??		if (??) RET	-	5, (3, 3)
SBB	M	$A=A-M-Y$	+	4, 3	RST	N	$<SP-1>=PC, PC=8N$	-	5, 3, 3
SBI	*	$A=A--Y$	+	4, 3	PUSH	RP	$<SP-1>=RP$	-	5, 3, 3
INR	R	$R=R+1$	!	5	PUSH	PSW	$<SP-1>=A, <SP-2>=F$	-	5, 3, 3
INR	M	$M=M+1$	!	4, 3, 3	POP	RP	$RP=<SP>$	-	4, 3, 3
DCR	R	$R=R-1$	!	4	POP	PSW	$F=<SP>, A=<SP+1>$	-	4, 3, 3
DCR	M	$M=M-1$	!	4, 3, 3	XTHL		$L \leftrightarrow <SP>, H \leftrightarrow <SP+1>$	-	5, 3, 3, 3, 4
INX	RP	$RP=RP+1$	-	5	IN	*	$A=<*>$	-	4, 3, 3
DCX	RP	$RP=RP-1$	-	5	OUT	*	$<*>=A$	-	4, 3, 3
DAD	RP	$HL=HL+RP$	Y	4, 3, 3	EI		$TP=1$	-	4
DAA		КОРРЕКЦ.	+	4	DI		$TP=0$	-	4
					HLT		ОЖИДАН.	-	4, 3
					NOP		НЕТ ОПЕР.	-	4

Примечание: в таблице приняты следующие обозначения:

R, R1, R2 – A, B, C, D, E, H, L (регистры).

RP – B, D, H, SP (регистровые пары BC, DE, HL и указатель стека).

*, ** – однобайтное и двухбайтное выражения соответственно.

Таблица А.2

Формирование признаков в регистре F:
 рехода

Обозначения в командах условного пе-

- признаки сохраняются
- + формируются все признаки
- ! формируются все признаки, кроме Y
- Y формируется только Y
- × формируются все признаки, Y=0

??	Условие	Признак в регистре F
C	перенос из старшего разряда	Y=1
NC	нет переноса	Y=0
PE	четный паритет	P=1
P0	нечетный паритет	P=0
Z	признак нулевого результата	Z=1
NZ	результат не равен нулю	Z=0
M	результат отрицательный	S=1
P	результат положительный	S=0

ПРИЛОЖЕНИЕ Б

Коды операций

MOV															MVI		
A, A	7F	B, A	47	C, A	4F	D, A	57	E, A	5F	H, A	67	L, A	6F	M, A	77	A, *	3E
A, B	78	B, B	40	C, B	48	D, B	50	E, B	58	H, B	60	L, B	68	M, B	70	B, *	06
A, C	79	B, C	41	C, C	49	D, C	51	E, C	59	H, C	61	L, C	69	M, C	71	C, *	0E
A, D	7A	B, D	42	C, D	4A	D, D	52	E, D	5A	H, D	62	L, D	6A	M, D	72	D, *	16
A, E	7B	B, E	43	C, E	4B	D, E	53	E, E	5B	H, E	63	L, E	6B	M, E	73	E, *	1E
A, H	7C	B, H	44	C, H	4C	D, H	54	E, H	5C	H, H	64	L, H	6C	M, H	74	H, *	26
A, L	7D	B, L	45	C, L	4D	D, L	55	E, L	5D	H, L	65	L, L	6D	M, L	75	L, *	2E
A, M	7E	B, M	46	C, M	4E	D, M	56	E, M	5E	H, M	66	L, M	6E			M, *	36

ADD		ADC		SUB		SBB		INR		DCR		ANA		XRA		ORA		CMP	
A	87	A	8F	A	97	A	9F	A	3C	A	3D	A	A7	A	AF	A	B7	A	BF
B	80	B	88	B	90	B	98	B	04	B	05	B	A0	B	A8	B	B0	B	B8
C	81	C	89	C	91	C	99	C	0C	C	0D	C	A1	C	A9	C	B1	C	B9
D	82	D	8A	D	92	D	9A	D	14	D	15	D	A2	D	AA	D	B2	D	BA
E	83	E	8B	E	93	E	9B	E	1C	E	1D	E	A3	E	AB	E	B3	E	BB
H	84	H	8C	H	94	H	9C	H	24	H	25	H	A4	H	AC	H	B4	H	BC
L	85	L	8D	L	95	L	9D	L	2C	L	2D	L	A5	L	AD	L	B5	L	BD
M	86	M	8E	M	96	M	9E	M	34	M	35	M	A6	M	AE	M	B6	M	BE

				RST	RET	C9	JMP **	C3	CALL **	CD			
LDAX B	0A	ADI *	C6	RLC	07	0	C7	RNZ	C0	JNZ **	C2	CNZ **	C4
LDAX D	1A	ACI *	CE	RRC	0F	1	CF	RZ	C8	JZ **	CA	CZ **	CC
LHLD **	2A	SUI *	D6	RAL	17	2	D7	RNC	D0	JNC **	D2	CNC **	D4
LDA **	3A	SBI *	DE	RAR	1F	3	DF	RC	D8	JC **	DA	CC **	DC
STAX B	02	ANI *	E6	DAA	27	4	E7	RPO	E0	JPO **	E2	CPO **	E4
STAX D	12	XRI *	EE	CMA	2F	5	EF	RPE	E8	JPE **	EA	CPE **	EC
SHLD **	22	ORI *	F6	STC	37	6	F7	RP	F0	JP **	F2	CP **	F4
STA **	32	CPI *	FE	CMC	3F	7	FF	RM	F8	JM **	FA	CM **	FC

LXI		DAD		INX		DCX		PUSH		POP					
B, **	01	B	09	B	03	B	0B	B	C5	B	C1	XCHG	EB	OUT *	D3
D, **	11	D	19	D	13	D	1B	D	D5	D	D1	XTHL	E3	IN *	DB
H, **	21	H	29	H	23	H	2B	H	E5	H	E1	SPHL	F9	DI	F3
SP, **	31	SP	39	SP	33	SP	3B	PSW	F5	PSW	F1	PCHL	E9	EI	FB
														NOP	00
														HLT	76