1. Основные функциональные и технические характеристики 8-разрядных микроконтроллеров.

Функциональные характеристики означают состав микроконтроллера, т.е. из каких модулей он состоит. Технические характеристики определяют значения параметров микроконтроллера.

В состав микроконтроллера входят:

- блок АЛУ
- память программ
- оперативная память данных
- блок синхронизации и управления
- порты ввода-вывода
- регистры специальных функций
- таймеры/счетчики
- система прерываний

AT89C51. В состав микроконтроллера входят:

- блок АЛУ;
- перепрограммируемая Flash-память программ;
- оперативная память данных RAM;
- блок синхронизации и управления БСУ;
- четыре восьмиразрядных порта ввода-вывода P0, P1, P2 и P3 с альтернативными функциями;
- регистры специальных функций SFR.

Значения параметров:

- разрядность – 8
- управляющая память – до 64 Кбайт
- память данных – до 64 Кбайт
- резидентная память данных (RAM) – 128 байт

- память программ (ROM или EPROM) – 4 Кбайт
- тактовая частота работы – 12 МГц
- число линий ввода-вывода – 4 (по 8 линий)
- таймеры – 2 шт (16-разрядные)
- АЛУ – аккумуляторного типа

2. Структура и интерфейс микроконтроллеров семейства MCS-51 (например, AT89C51). Назначение устройств микроконтроллера.

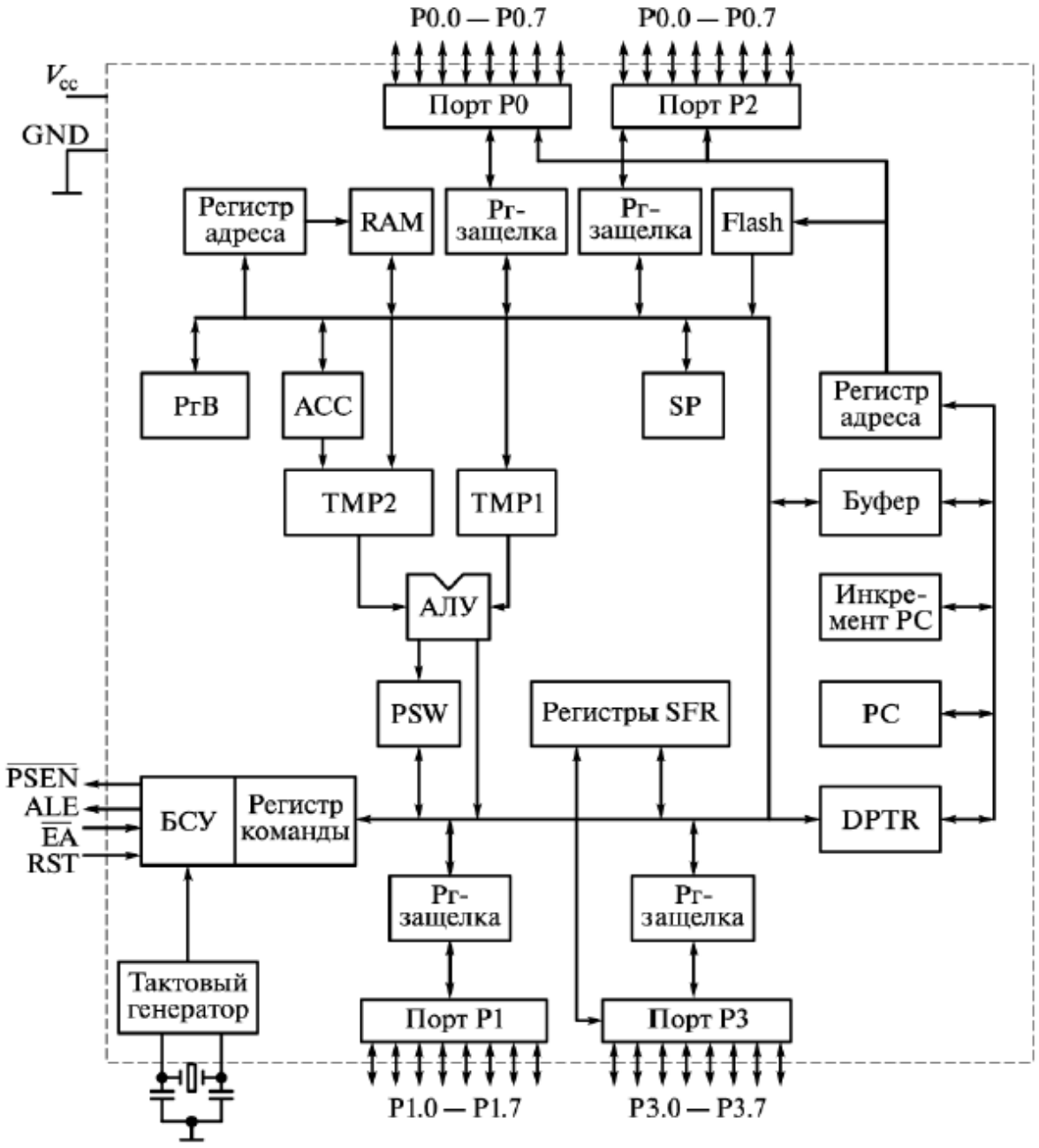


Рис. 3.2. Структурная схема микроконтроллера AT89C51

AT89C51. В состав микроконтроллера входят:

- блок АЛУ;
- перепрограммируемая Flash-память программ;
- оперативная память данных RAM;
- блок синхронизации и управления БСУ;
- четыре восьмиразрядных порта ввода-вывода P0, P1, P2 и P3 с альтернативными функциями;
- регистры специальных функций SFR.

Все устройства объединяет внутренняя восьмиразрядная шина данных. Внешний интерфейс представляют 32 линии портов ввода-вывода, выводы тактового генератора, именуемые XTAL1 и XTAL2 (на рисунке не обозначены), вход сброса RST, выходные сигналы стробирования адреса ALE при работе с внешней памятью и обращения к памяти программ $\overline{\text{PSEN}}$, сигнал отключения резидентной (внутренней) памяти программ $\overline{\text{EA}}$. Дополнительная группа внешних сигналов управления содержит сигналы, передаваемые через порт P3:

- сигналы записи $\overline{\text{WR}}$ и чтения $\overline{\text{RD}}$ внешней памяти данных;
- входные сигналы таймеров T0, T1;
- входы внешних прерываний $\overline{\text{INT0}}$, $\overline{\text{INT1}}$;
- вход приемника RxD и выход передатчика TxD последовательного канала ввода-вывода.

Блок АЛУ представляет собой устройство аккумуляторного типа. В него входят регистр-аккумулятор ACC, регистр расширения PrB, два программно-недоступных регистра временного хранения TMP1 и TMP2, двоично-десятичный корректор, регистр слова состояния программы PSW. Блок АЛУ обеспечивает выполнение целочисленных операций двоичного сложения-вычитания восьмиразрядных операндов, коррекцию результата при сложении двоично-десятичных чисел. Наличие в АЛУ аппаратных средств умножения и деления позволяет выполнять умножение и деление восьмиразрядных целых чисел.

Flash-память, применяемую в микроконтроллерах, можно неоднократно перепрограммировать, число допустимых циклов перепрограммирования достигает 10 000. Адресация ячеек памяти осуществляется через программный счетчик PC и 16-разрядный регистр-указатель DPTR. При превышении максимального адреса внутренней памяти адрес ячейки направляется в порты P0, P2 для поиска ячейки во внешней памяти. Ячейки внутренней памяти данных адресуются через регистр адреса RAM или указатель стека SP.

Блок управления содержит тактовый генератор, регистр команды и блок формирования сигналов синхронизации и управления БСУ для внутренних и внешних устройств микроконтроллерной системы.

Блок регистров специальных функций SFR включает в себя два 16-разрядных таймера T0, T1 с изменяемой конфигурацией, регистры системы прерываний, канал последовательного ввода-вывода и схему управления мощностью. Каждый таймер содержит по два восьмиразрядных регистра: THx, TLx (x = 0, 1). Для управления работой таймеров используют два регистра: режима TMOD и управления TCON (старшая тетрада). Систему прерываний обслуживают два регистра: маски IE и приоритетов IP, а также младшая тетрада регистра TCON. Последовательный порт представлен регистрами данных SBUF и управления SCON. Управление потребляемой мощностью микроконтроллера осуществляется из регистра PCON.

3) Логическая организация памяти микроконтроллеров MCS-51.

3.3. Логическая организация памяти

В микроконтроллерах с архитектурой MCS-51 используется память двух типов: постоянная, применяемая для хранения программ и констант, и оперативная для хранения данных. В микроконтроллерах с комплементарной МОП-технологией 80C51/87C51 (Intel) и Flash-памятью AT87F51/AT89C51 (Atmel) резидентная память программ имеет объем 4 Кбайт. При этом объем памяти можно расширить до 64 Кбайт за счет подключения внешней памяти. Микроконтроллеры новых поколений 80x52/54/58 имеют увеличенную память, в зависимости от модели ее объем составляет 8/16/32 Кбайт. Резидентная память данных в микроконтроллерах базового ряда 51 имеет объем 128 байт, ряда 52/54/58 — 256 байт.

Начальная область адресов постоянной памяти программ ROM, начиная с адреса 0003h, зарезервирована для таблицы прерываний (рис. 3.3, *а*). В связи с этим при использовании прерываний основную программу, вызываемые подпрограммы и обработчики прерываний следует размещать выше таблицы прерываний, а учитывая, что при сбросе микроконтроллера программный счетчик принимает значение 0000h, по указанному адресу следует разместить команду перехода к основной программе. Переход из резидентной области программ к внешней, выше 4 Кбайт, происходит автоматически. Используя управляющий сигнал интерфейса $\overline{EA} = 0$, можно отключить резидентную память программ и работать только с внешней памятью размером до 64 Кбайт, что удобно при отладке программного кода, когда требуется вносить изменения в программу. После завершения отладки программный код может быть запрограммирован во внутренней памяти микроконтроллера.

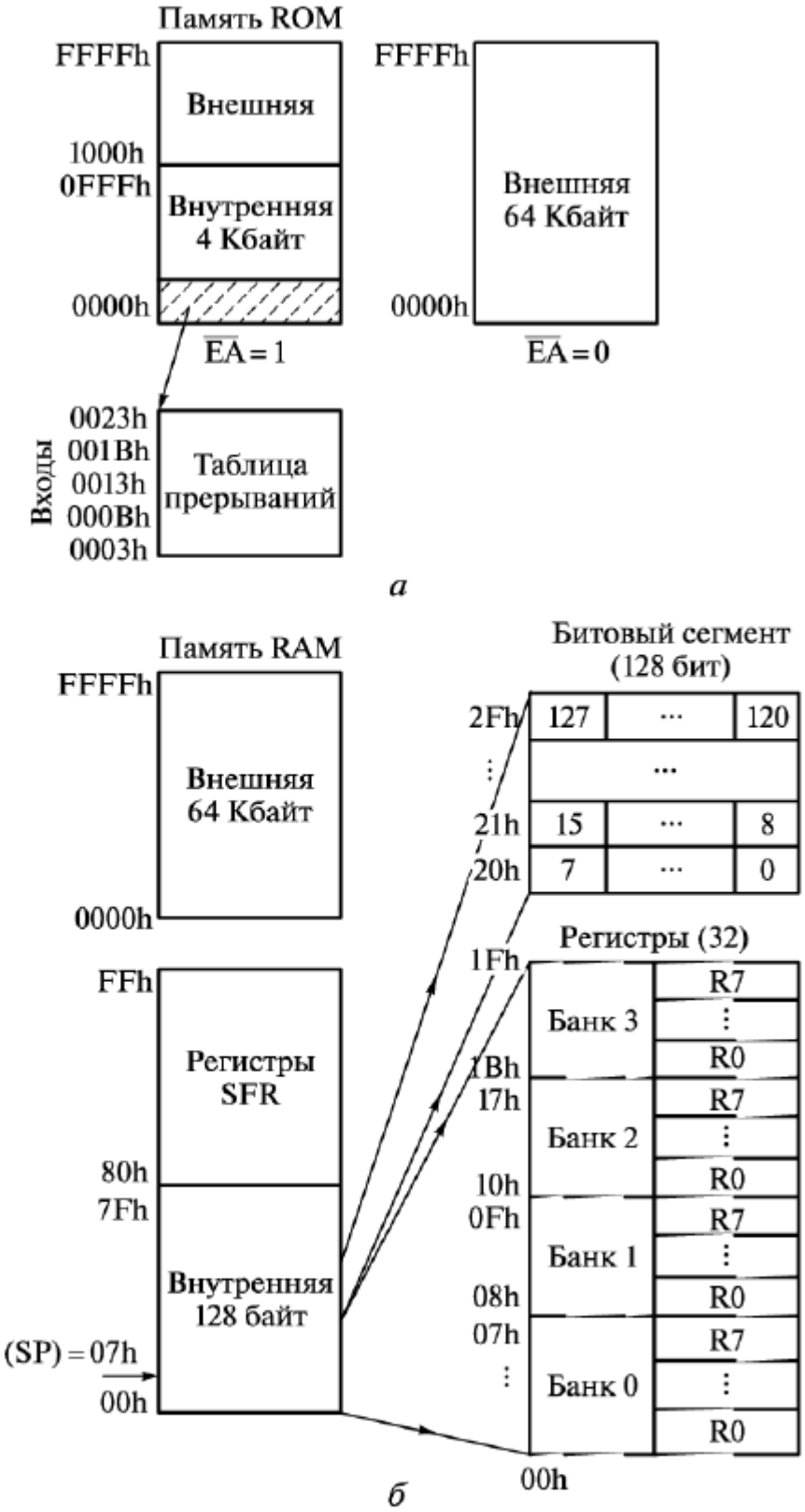


Рис. 3.3. Организация памяти микроконтроллера AT89C51:
а — постоянной; *б* — оперативной

Адресное пространство оперативной памяти данных RAM (рис. 3.3, *б*) охватывает регистровый сегмент (00h... 1Fh), битовый сегмент (20h... 2Fh), ячейки внутренней памяти данных (128 байт) и регистры специальных функций. Регистровый сегмент представлен 32 регистрами, организованными в четыре банка, по восемь регистров в каждом банке. Выбор банка регистров (0, 1, 2, 3) осуществляется с помощью двух битов слова и состояния программы PSW, выбор регистров в каждом банке — по именам R0 — R7. При работе с регистрами выбранного банка доступ к остальным регистрам можно произвести только как к ячейкам памяти адресного пространства памяти данных.

Выше регистрового сегмента по адресам 20h... 2Fh расположен битовый сегмент, занимающий 16 байт (128 бит). Доступ к элементам этого сегмента можно осуществить двояко: либо побитно с помощью команд битовых операций и восьмиразрядных адресов битов, либо побайтно, выбирая с помощью одной команды одновременно 8 бит и применяя в дальнейшем маску для выделения нужных битов.

Стек располагается во внутренней памяти данных. При сбросе микроконтроллера указатель стека SP принимает значение 0007h. Перед занесением данных в стек указатель стека увеличивается на единицу, после чтения из стека уменьшается на единицу. Таким образом, первые записи в стек оказываются в регистрах первого банка. Это требует от программиста при написании программы контроля вызова подпрограмм и использования регистров.

В старшей половине внутреннего адресного пространства (80h ... FFh) располагаются регистры специальных функций. Обращение к ним обычно осуществляется по именам. Также по именам можно обращаться к отдельным битам этих регистров, выполняя битовые операции. Все это необходимо учитывать при работе с ячейками для хранения данных при объеме памяти 256 байт. Ячейки резидентной памяти данных могут участвовать как операнды-источники и как операнды — приемники результатов во всех операциях: пересылки, арифметической и логической обработки данных, сравнения и перехода.

Память данных микроконтроллера является расширяемой за счет дополнительной внешней оперативной памяти объемом до 64 Кбайт. Доступ к ячейкам внешней памяти осуществляется только с помощью команд пересылки вида: MOVX A, @name; MOVX @name, A, где A — содержимое аккумулятора; name — имя одного из восьмиразрядных регистров (R0, R1) или 16-разрядного регистра-указателя DPTR.

Блочный обмен данными внешней памяти и внутренней (резидентной) памяти можно осуществить, только выполнив *N* раз, с учетом размера пересылаемого блока, циклическую процедуру пересылки в направлении (а) или (б):

а) из внешней ячейки памяти в аккумулятор, затем из аккумулятора во внутреннюю ячейку памяти;

б) из внутренней ячейки памяти в аккумулятор, затем из аккумулятора во внешнюю ячейку памяти.

Для адресации ячеек внешней и внутренней памяти используют косвенно-регистровую адресацию через регистры DPTR, R0 или R1.

4. Структура и интерфейс микроконтроллеров семейства AVR. Назначение устройств микроконтроллера. (инфа из сайта выглдит куда более структурированно чем в учебнике слишком много теста поэтому советую взять 5-6 предложений по пункту)

Все AVR имеют **Flash-память программ**, которая может быть загружена как с помощью обычного программатора, так и с помощью SPI-интерфейса, в том числе непосредственно на целевой плате. Число циклов перезаписи - не менее 1000. Последние версии кристаллов семейства "mega" выпуска 2001-2002 года имеют возможность самопрограммирования. Это означает, что микроконтроллер способен самостоятельно, без какого-либо внешнего программатора, изменять содержимое ячеек памяти программ. То есть, новые AVR могут менять алгоритмы своего функционирования и программы, заложенные в них, и далее работать уже по измененному алгоритму или новой программе. Например, Вы можете написать и сохранить несколько рабочих программ и менять их по мере надобности.
Все AVR имеют также блок энергонезависимой электрически стираемой памяти данных EEPROM. Этот тип памяти, доступный программе микроконтроллера непосредственно в ходе ее выполнения, удобен для хранения промежуточных данных, различных констант, таблиц перекодировок, калибровочных коэффициентов и т.п. EEPROM также может быть загружена извне как через SPI интерфейс, так и с помощью обычного программатора. Число циклов перезаписи - не менее 100000. Два программируемых бита секретности позволяют защитить память программ и энергонезависимую память данных EEPROM от несанкционированного считывания. Внутренняя оперативная память SRAM имеется у всех AVR семейств "classic" и "mega" и у одного нового кристалла семейства "tiny" - ATtiny26/L. Для некоторых микроконтроллеров возможна организация подключения внешней памяти данных объемом до 64K.
Внутренний тактовый генератор AVR может запускаться от нескольких источников опорной частоты (внешний генератор, внешний кварцевый резонатор, внутренняя или внешняя RC-цепочка). Поскольку AVR-микроконтроллеры полностью статические, минимальная допустимая частота ничем не ограничена (вплоть до пошагового режима). Максимальная рабочая частота определяется конкретным типом микроконтроллера. Верхние границы частотного диапазона, указанные в таблицах 1 – 3 (приложение 1), гарантируют устойчивую работу микроконтроллеров при работе во всем температурном диапазоне (хотя, например, AT90S8515 при комнатной температуре может быть "разогнан" до 14 МГц). Интересную аппаратную особенность имеет микроконтроллер ATtiny15L. Он содержит блок PLL для аппаратного умножения основной тактовой частоты в 16 раз. При номинальном значении последней 1,6 МГц получаемая вспомогательная периферийная частота равна 25,6 МГц. Эта частота может служить источником для одного из таймеров/счетчиков микроконтроллера, значительно повышая временное разрешение его работы.
Сторожевой (WATCHDOG) таймер предназначен для защиты микроконтроллера от сбоев в процессе работы. Он имеет свой собственный RC-генератор, работающий на частоте 1 МГц. Эта частота является приближенной и зависит прежде всего от величины напряжения питания микроконтроллера и от температуры. WATCHDOG-таймер снабжен своим собственным предделителем входной частоты с программируемым коэффициентом деления, что позволяет подстраивать временной интервал переполнения таймера и сброса микроконтроллера.
Микроконтроллеры AVR имеют в своем составе от 1 до 4 таймеров/счетчиков общего назначения с разрядностью 8 или 16 бит, которые могут работать и как таймеры от внутреннего источника опорной частоты, и как счетчики внешних событий с внешним тактированием. Общие черты всех таймеров/счетчиков следующие:

- наличие программируемого предделителя входной частоты с различными градациями деления. Отличительной чертой является возможность работы таймеров/счетчиков на основной тактовой частоте микроконтроллера без предварительного ее понижения, что существенно повышает точность генерации временных интервалов системы;
- независимое функционирование от режима работы процессорного ядра микроконтроллера (т.е. они могут быть как считаны, так и загружены новым значением в любое время);
- возможность работы или от внутреннего источника опорной частоты, или в качестве счетчика событий. Верхний частотный порог определен в этом случае как половина основной тактовой частоты микроконтроллера. Выбор перепада внешнего источника (фронт или срез) программируется пользователем;
- наличие различных векторов прерываний для нескольких различных событий (переполнение, захват, сравнение).

Система реального времени (RTC) реализована во всех микроконтроллерах семейства "mega" и в двух кристаллах семейства "classic" - AT90(L)S8535. Таймер/счетчик RTC имеет свой собственный предделитель,

который может быть программным способом подключен или к основному внутреннему источнику тактовой частоты микроконтроллера, или к дополниельному асинхронному источнику опорной частоты (кварцевый резонатор или внешний синхросигнал). Для этой цели зарезервированы два внешних вывода микроконтроллера. Внутренний осциллятор, нагруженный на счетный вход таймера/счетчика RTC, оптимизирован для работы с внешним "часовым" кварцевым резонатором 32,768 кГц.

Порты ввода/вывода AVR имеют число независимых линий "Вход/Выход" от 3 до 53. Каждый разряд порта может быть запрограммирован на ввод или на вывод информации. Мощные выходные драйверы обеспечивают токовую нагрузочную способность 20 мА на линию порта (втекающий ток) при максимальном значении 40 мА, что позволяет, например, непосредственно подключать к микроконтроллеру светодиоды и биполярные транзисторы. Общая токовая нагрузка на все линии одного порта не должна превышать 80 мА.

Интересная архитектурная особенность построения портов ввода/вывода у AVR заключается в том, что для каждого физического вывода существует 3 бита контроля/управления, а не 2, как у распространенных 8-разрядных микроконтроллеров (Intel, Microchip, Motorola и т.д.) Естественно возникает вопрос: а зачем необходимы именно 3 бита? Дело в том, что использование только двух бит контроля/управления порождает ряд проблем при операциях типа "чтение-модификация-запись". Например, если имеют место две последовательные операции "чтение-модификация-запись", то первый результат может быть потерян безвозвратно, если вывод порта работает на емкостную нагрузку и требуется некоторое время для стабилизации уровня сигнала на внешнем выводе микросхемы. Архитектура построения портов ввода/вывода AVR с тремя битами контроля/управления позволяет разработчику полностью контролировать процесс ввода/вывода. Если необходимо получить реальное значение сигнала на физическом выводе микроконтроллера - читайте содержимое бита по адресу PINx. Если требуется обновить выходы - прочитайте PORTx защелку и потом модифицируйте данные. Это позволяет избежать необходимости иметь копию содержимого порта в памяти для безопасности и повышает скорость работы микроконтроллера при работе с внешними устройствами. Особую значимость приобретает данная возможность AVR для реализации систем, работающих в условиях внешних электрических помех.

Аналоговый компаратор входит в состав большинства микроконтроллеров AVR. Типовое напряжение смещения равно 10 мВ, время задержки распространения составляет 500 нс и зависит от напряжения питания микроконтроллера. Так, например, при напряжении питания 2,7 Вольт оно равно 750 нс. Аналоговый компаратор имеет свой собственный вектор прерывания в общей системе прерываний микроконтроллера. При этом тип перепада, вызывающий запрос на прерывание при срабатывании компаратора, может быть запрограммирован пользователем как фронт, срез или переключение. Логический выход компаратора может быть программным образом подключен ко входу одного из 16-разрядных таймеров/счетчиков, работающего в режиме захвата. Это дает возможность измерять длительность аналоговых сигналов а также максимально просто реализовывать АЦП двухтактного интегрирования.

Аналого-цифровой преобразователь (АЦП) построен по классической схеме последовательных приближений с устройством выборки/хранения (УВХ). Каждый из аналоговых входов может быть соединен со входом УВХ через аналоговый мультиплексор. Устройство выборки/хранения имеет свой собственный усилитель, гарантирующий, что измеряемый аналоговый сигнал будет стабильным в течение всего времени преобразования. Разрядность АЦП составляет 10 бит при нормируемой погрешности ±2 разряда. АЦП может работать в двух режимах - однократное преобразование по любому выбранному каналу и последовательный циклический опрос всех каналов. Время преобразования выбирается программно с помощью установки коэффициента деления частоты специального предделителя, входящего в состав блока АЦП. Оно составляет 70...280 мкс для ATmega103 и 65...260 мкс для всех остальных микроконтроллеров, имеющих в своем составе АЦП. Важной особенностью аналого-цифрового преобразователя является функция подавления шума при преобразовании. Пользователь имеет возможность, выполнив короткий ряд программных операций, запустить АЦП в то время, когда центральный процессор находится в одном из режимов пониженного энергопотребления. При этом на точность преобразования не будут оказывать влияние помехи, возникающие при работе процессорного ядра.

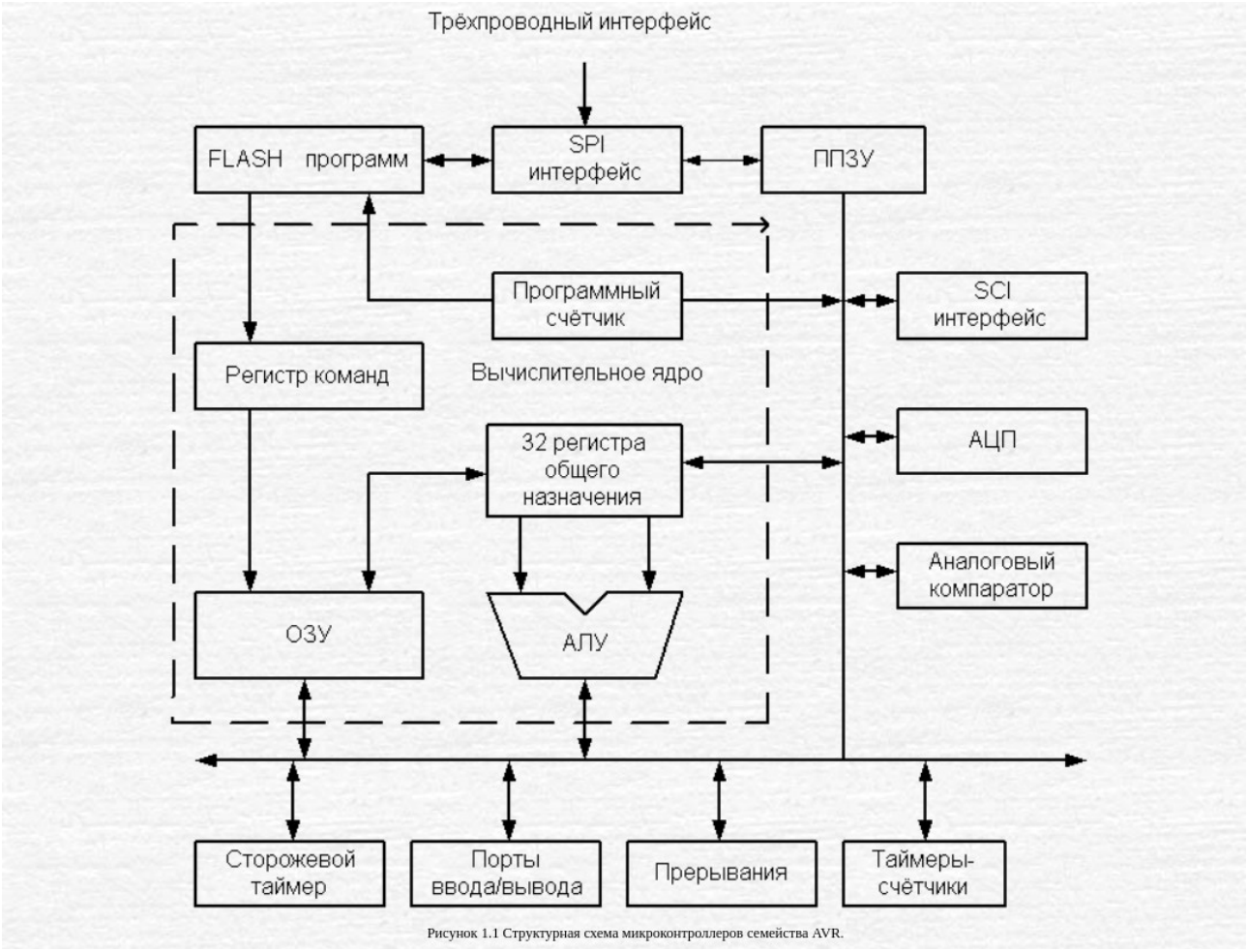


Рисунок 1.1 Структурная схема микроконтроллеров семейства AVR.

5. Карта памяти МК AVR(учебник)

Память организована как показано на рисунке.
1) Flash обособлена и её размер – 8Кбайт, каждая ячейка её составляет 16 разрядов.
2) Память данных – делится на 3 части -
2.1) Регистровая
2.2) Оперативная
2.3) Статическая

Регистровую память составляют 32 РОН + 64 Рег ввода-вывода(представляет из себя переферийные устройства)
Оперативная память(ОП) – 512 Кбайт – хранение данных программ.
Вообще регистровая и ОП образуют единое адр пр-во:
- $0000 - $001F РОН
- $0020 - $005F рег ввода-вывода
- $0060 - $025F ОП
возможно далее до FFFF расширение адр пр-ва, при подключении внешней ERAM.

Можно подключать EEPROM объемом 512 байт, которое имеет своё обособленное адр пространство, каждая ячейка содержит 8 разрядов. В данную память данные обычно записываются при программировании МК. Память энергонезависима.
РОН делятся на 2 группы R0-R15 , R16-R31
AVR имеют 44 рег ввода-вывода в составе периферии (+ 20 зарезервированных регистров), из них 14 – 8 разрядные регистры данных, остальные – это регистры управления и состояния.
ОП – служит для оперативного хранения данных при выполнении программы. Даные для записи поступают из регистра общ назначения. Память энергозависима.

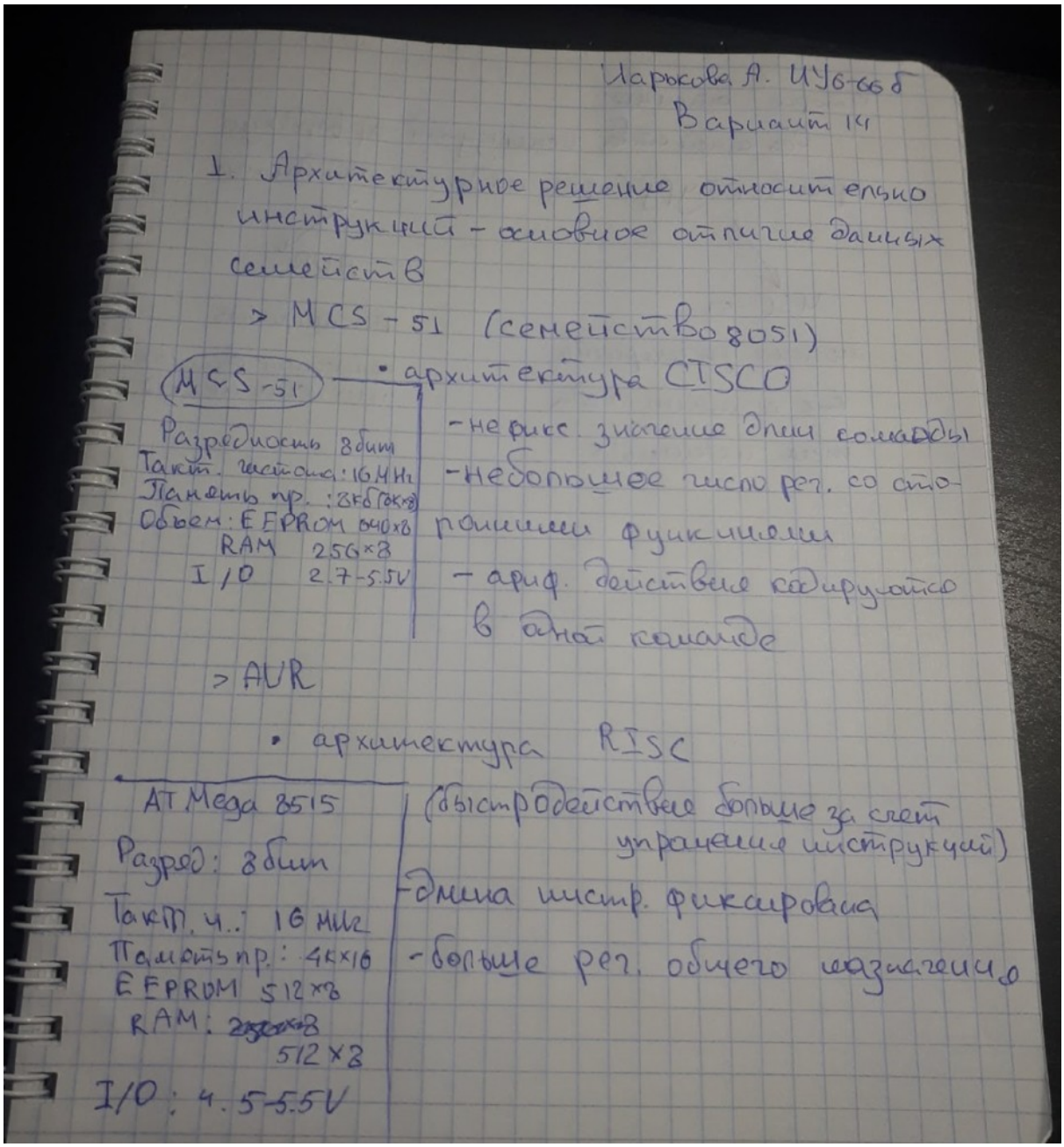
Стек – располагается в памяти SRAM обычно под него располагается область адресов, начиная с адреса $025F. Растет в сторону убывающих адресов.

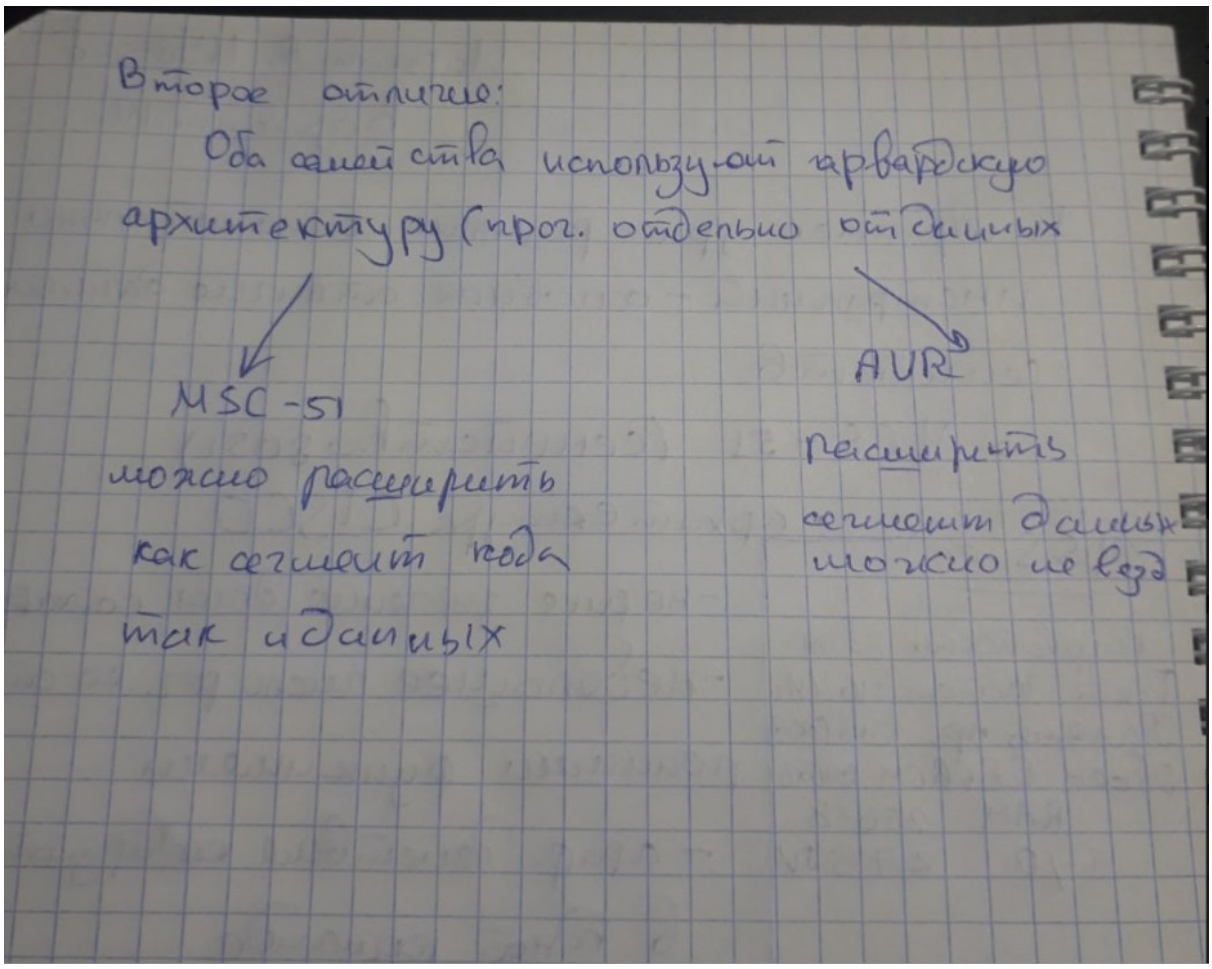
/

6. Отличительные признаки архитектуры (включая характеристики) микроконтроллеров, представляющих семейства MCS-51 и AVR.

Ядро MR AVR выполнено по усовершенствованной RISC- архитектуре в которой используется ряд решений направленных на повышение быстродействия. Это обеспечивает более высокую производительность по равной частоте по сравнению с MR на основе CISC-архитектуре MCS-51.

MR с CISC-архитектурой имеет большой набор команд с развитыми возможностями давая разработчику возможность выбрать наиболее подходящую команду для выполнения. Выборка команды на исполнение осуществляется побитно в течении нескольких циклов работы MR. Время выполнения команды может составлять от 1 до 12 циклов от генератора.





Привести примеры команд микроконтроллеров AVR, операндами которых являются регистры общего назначения, регистры ввода/ вывода, ячейки памяти SRAM, константы памяти Flash. Как передать данные из одного порта в другой, из порта в ячейку, из ячейки в порт, из ячейки в ячейку?

Примеры:

- Команды, которые используют два регистра общего назначения: d(Rd) и r(Rr). Этот вид адресации используют команды пересылки данных из регистра в регистр и большинство команд арифметических операций, ряд команд логических операций. При этих операциях результат операции сохраняется в регистре d (Rd).

MOV Rd, Rr Пересылка между регистрами

ADD Rd, Rr Сложение двух регистров
ADC Rd, Rr Сложение двух регистров и переноса

Данный вид адресации используют для выполнения обмена между регистром ввода/вывода, расположенным в адресном пространстве ввода/вывода, и одним из регистров общего назначения по командам IN и OUT (рис.).

IN Rd, P Чтение регистра ввода-вывода

OUT P, Rr Запись в регистр ввода-вывода

- Данный способ адресации применяется при обращении к любой ячейке адресного пространства SRAM. Имеются всего две команды: LDS и STS, каждая длиной в два слова (32 разряда). Первое слово содержит код операции и адрес регистра общего назначения, второе - 16-разрядный адрес ячейки, к которой направлено обращение

LDS Rd, k Прямая загрузка регистра

STS k, Rr Прямое сохранение

- Микроконтроллеры AVR позволяют обратиться к ячейкам памяти программ для считывания констант, а в моделях семейства Mega – и для записи данных во Flash-память программ, используя механизм косвенной адресации через регистр Z. При этом старшие 15 разрядов регистра определяют адрес слова, а младший нулевой разряд – младший или старший байт слова. В случае если младший разряд адреса равен 0, выбирается младший байт, в противном случае выбирается старший байт. Данный вид адресации используют команды считывания из ячейки памяти в регистр R0 (LPM) и записи в память из регистров R1:R0 (SPM).

  Кроме простой косвенной адресации при чтении константы в регистр Rd можно применить косвенную адресацию с постинкрементом (команда LPM Rd, Z+). Помимо команд, связанных с передачей данных, косвенная адресация должна быть использована в командах косвенного перехода по адресу в регистре Z (IJMP) и косвенного вызова подпрограммы через регистр Z (1CALL)

7) Типы данных, способы адресации данных в микроконтроллерах MCS-51.

Комментарий:

Источник: Учебник Хартова (страница 45)

Также можно смотреть сюда: http://www.gaw.ru/html.cgi/txt/doc/micros/mcs51/asm/start.htm

В части про типы данных я не очень уверен

Способы адресации:

Команды используют в основном четыре вида адресации: регистровую, косвенно-регистровую, прямую, непосредственную.

При *регистровой адресации* операнд находится в одном из восьми регистров общего назначения R0 — R7 выбранного для работы банка регистров. Команды с регистровой адресацией имеют длину один байт, в котором три разряда используют для указания номера регистра, остальные биты содержат код операции.

При *косвенно-регистровой адресации* в качестве указателя восьмиразрядного адреса ячейки памяти используется один из регистров R0, R1.

При *прямой адресации* команда содержит восьмиразрядный адрес ячейки памяти данных (второй байт команды) и позволяет адресовать ячейки внутренней памяти данных. Подобным образом можно адресовать также элементы битового сегмента.

Непосредственная адресация определяет операнд-константу, указываемый в самой команде. Операнды занимают один или два байта. Однобайтовые операнды могут быть использованы в арифметических, логических операциях и в командах пересылки, двухбайтовые операнды представляют собой адресные константы, загружаемые в указательный регистр DPTR.

При чтении констант из постоянной памяти применяют два вида косвенной адресации: относительную и базовую. В первом случае адрес ячейки с константой вычисляется путем сложения содержимого программного счетчика и смещения из аккумулятора, во втором — сложением содержимого указательного регистра DPTR и аккумулятора.

Относительная адресация широко применяется в операциях условной передачи управления. В этих случаях адрес перехода при выполнении условия перехода вычисляется путем сложения содержимого программного счетчика PC после выборки двухбайтовой команды условного перехода со смещением rel, определяемым после трансляции вторым байтом команды (PC = PC + 2 + rel).

Типы данных (?):

В качестве операндов могут быть использованы байты, биты, полубайты (тетрады) и 16-разрядные данные. В АЛУ аккумуляторного типа обрабатываются байтовые операнды. При этом данные представляют в виде целочисленных значений без знака или со знаком. Диапазон беззнаковых чисел находится в пределах от 0 до 255. Числа со знаком в старшем разряде лежат в диапазоне от –128 до +127. Дробные и смешанные числа с фиксированной точкой, а также с плавающей точкой обрабатываются с помощью специальных алгоритмов и программ.

При арифметических операциях:

Команда двоично-десятичной коррекции (DA A) применяется при сложении 2 — 10 операндов для получения результата также в двоично-десятичном коде. При сложении многобайтных операндов коррекция проводится каждый раз после сложения очередных байтов.

8 Логические и арифметические операции микроконтроллеров MCS-51

Арифметические операции (табл. 3.2) представлены набором из 24 команд. Сюда входят сложение ADD и сложение с переносом ADDC, вычитание с заемом SUBB, операции приращения на единицу INC (инкремента) и уменьшения на единицу DEC (декремента), умножения и деления байтов. Используя команды сложения с переносом, вычитания с заемом, легко построить программные процедуры сложения (вычитания) многобайтовых операндов. Команда умножения MUL осуществляет перемножение восьмиразрядных операндов, помещенных в регистры A и B. Полученное 16-разрядное произведение формируется в регистрах B (старший байт) и A (младший байт). Команда деления DIV осуществляет деление восьмиразрядного делимого в регистре A на восьмиразрядный делитель в регистре B. Частное от деления записывается в регистр A, остаток — в регистр B.

Команда двоично-десятичной коррекции (DA A) применяется при сложении 2—10 операндов для получения результата также в двоично-десятичном коде. При сложении многобайтных операндов коррекция проводится каждый раз после сложения очередных байтов.

При выполнении арифметических операций в зависимости от полученного результата устанавливаются (сбрасываются) флаги переноса C, межтетрадного переноса AC, переполнения OV, паритета P в регистре состояния PSW.

Логические операции (табл. 3.3) представлены операциями логического умножения ANL (И), логического сложения ORL (ИЛИ), XRL (исключающее ИЛИ), инверсии CPL и очистки CLR содержимого аккумулятора и выполняются поразрядно с восьмиразрядными операндами.

Операции сдвига, проводимые в аккумуляторе, представлены четырьмя операциями: RL — циклический сдвиг влево, RLC — циклический сдвиг влево через перенос, RR — циклический сдвиг вправо, RRC — циклический сдвиг вправо через перенос.

Арифметические операции (табл. 3.2) представлены набором из 24 команд. Сюда входят сложение ADD и сложение с переносом ADDC, вычитание с заемом SUBB, операции приращения на единицу INC (инкремента) и уменьшения на единицу DEC (декремента), умножения и деления байтов. Используя команды сложения с переносом, вычитания с заемом, легко построить программные процедуры сложения (вычитания) многобайтовых операндов. Команда умножения MUL осуществляет перемножение восьмиразрядных операндов, помещенных в регистры A и B. Полученное 16-разрядное произведение формируется в регистрах B (старший байт) и A (младший байт). Команда деления DIV осуществляет деление восьмиразрядного делимого в регистре A на восьмиразрядный делитель в регистре B. Частное от деления записывается в регистр A, остаток — в регистр B.

Команда двоично-десятичной коррекции (DA A) применяется при сложении 2—10 операндов для получения результата также в двоично-десятичном коде. При сложении многобайтных операндов коррекция проводится каждый раз после сложения очередных байтов.

При выполнении арифметических операций в зависимости от полученного результата устанавливаются (сбрасываются) флаги переноса C, межтетрадного переноса AC, переполнения OV, паритета P в регистре состояния PSW.

Логические операции (табл. 3.3) представлены операциями логического умножения ANL (И), логического сложения ORL (ИЛИ), XRL (исключающее ИЛИ), инверсии CPL и очистки CLR содержимого аккумулятора и выполняются поразрядно с восьмиразрядными операндами.

Операции сдвига, проводимые в аккумуляторе, представлены четырьмя операциями: RL — циклический сдвиг влево, RLC — циклический сдвиг влево через перенос, RR — циклический сдвиг вправо, RRC — циклический сдвиг вправо через перенос.

Таблица 3.2. **Команды арифметических операций**

Мнемоника	Описание команды	Б/Ц	Операция
ADD A, Rn	Сложение аккумулятора с регистром	1/1	(A) ← (A) + (Rn)
ADD A, ad	Сложение аккумулятора с прямо адресуемой ячейкой	2/1	(A) ← (A) + (ad)
ADD A, @Ri	Сложение аккумулятора с косвенно адресуемой ячейкой	1/1	(A) ← (A) + ((Ri))
ADD A, #data	Сложение аккумулятора с константой	2/1	(A) ← (A) + #data
ADDC A, Rn	Сложение аккумулятора с регистром и переносом	1/1	(A) ← (A) + (Rn) + (C)
ADDC A, ad	Сложение аккумулятора с прямо адресуемой ячейкой и переносом	2/1	(A) ← (A) + (ad) + (C)
ADDC A, @Ri	Сложение аккумулятора с косвенно адресуемой ячейкой и переносом	1/1	(A) ← (A) + ((Ri)) + (C)
ADDC A, #data	Сложение аккумулятора с константой и переносом	2/1	(A) ← (A) + #data + (C)
SUBB A, Rn	Вычитание из аккумулятора регистра и заема	1/1	(A) ← (A) – (Rn) – (C)
SUBB A, ad	Вычитание из аккумулятора прямо адресуемой ячейки и заема	2/1	(A) ← (A) – (ad) – (C)
SUBB A, @Ri	Вычитание из аккумулятора косвенно адресуемой ячейки и заема	1/1	(A) ← (A) – ((Ri)) – (C)
SUBB A, #data	Вычитание из аккумулятора константы и заема	2/1	(A) ← (A) – #data – (C)
INC A	Инкремент аккумулятора	1/1	(A) ← (A) + 1
INC Rn	Инкремент регистра	1/1	(Rn) ← (Rn) + 1
INC ad	Инкремент прямо адресуемой ячейки	2/1	(ad) ← (ad) + 1
INC @Ri	Инкремент косвенно адресуемой ячейки	1/1	((Ri)) ← ((Ri)) + 1
INC DPTR	Инкремент указателя данных	1/2	(DPTR) ← (DPTR) + 1
DEC A	Декремент аккумулятора	1/1	(A) ← (A) – 1
DEC Rn	Декремент регистра	1/1	(Rn) ← (Rn) – 1

Окончание табл. 3.2

Мнемоника	Описание команды	Б/Ц	Операция
DEC ad	Декремент прямо адресуемой ячейки	2/1	(ad) ← (ad) – 1
DEC @Ri	Декремент косвенно адресуемой ячейки	1/1	((Ri)) ← ((Ri)) – 1
MUL AB	Умножение аккумулятора на регистр B	1/4	(B).(A) ← (A)×(B)
DIV AB	Деление аккумулятора на регистр B	1/4	(A).(B) ← (A)/(B)
DA A	Десятичная коррекция аккумулятора	1/1	если (A.0—A.3) > 9 или (AC) = 1, то (A.0—A.3) ← (A.0—A.3) + 6, затем, если (A.4—A.7) > 9 или (C) = 1, то (A.4—A.7) ← (A.4—A.7) + 6

Таблица 3.3. **Команды логических операций**

Мнемоника	Описание команды	Б/Ц	Операция
ANL A, Rn	Логическое И аккумулятора и регистра	1/1	(A) ← (A) ∧ (Rn)
ANL A, ad	Логическое И аккумулятора и прямо адресуемой ячейки	2/1	(A) ← (A) ∧ (ad)
ANL A, @Ri	Логическое И аккумулятора и косвенно адресуемой ячейкой	1/1	(A) ← (A) ∧ ((Ri))
ANL A, #data	Логическое И аккумулятора и константы	2/1	(A) ← (A) ∧ #data
ANL ad, A	Логическое И прямо адресуемой ячейки и аккумулятора	2/1	(ad) ← (ad) ∧ A
ANL ad, #data	Логическое И прямо адресуемой ячейки и константы	3/2	(ad) ← (ad) ∧ #data
ORL A, Rn	Логическое ИЛИ аккумулятора и регистра	1/1	(A) ← (A) ∨ (Rn)
ORL A, ad	Логическое ИЛИ аккумулятора и прямо адресуемой ячейки	2/1	(A) ← (A) ∨ (ad)
ORL A, @Ri	Логическое ИЛИ аккумулятора и косвенно адресуемой ячейки	1/1	(A) ← (A) ∨ ((Ri))
ORL A, #data	Логическое ИЛИ аккумулятора и константы	2/1	(A) ← (A) ∨ #data
ORL ad,	Логическое ИЛИ прямо адресуемой ячейки и аккумулятора	2/1	(ad) ← (ad) ∨ A
A ORL ad, #data	Логическое ИЛИ прямо адресуемой ячейки и константы	3/2	(ad) ← (ad) ∨ #data
XRL A, Rn	Исключающее ИЛИ аккумулятора и регистра	1/1	(A) ← (A) ⊕ (Rn)
XRL A, ad	Исключающее ИЛИ аккумулятора и прямо адресуемой ячейки	2/1	(A) ← (A) ⊕ (ad)
XRL A, @Ri	Исключающее ИЛИ аккумулятора и косвенно адресуемой ячейки	1/1	(A) ← (A) ⊕ ((Ri))
XRL A, #data	Исключающее ИЛИ аккумулятора и константы	2/1	(A) ← (A) ⊕ #data
XRL ad, A	Исключающее ИЛИ прямо адресуемой ячейки и аккумулятора	2/1	(ad) ← (ad) ⊕ A
XRL ad, #data	Исключающее ИЛИ прямо адресуемой ячейки и константы	3/2	(ad) ← (ad) ⊕ #data
CLR A	Сброс аккумулятора	1/1	(A) ← 0
CPL A	Инверсия аккумулятора	1/1	(A) ← /(A)
RL A	Сдвиг аккумулятора влево циклический	1/1	(A.n + 1) ← (A.n), n = 0—6, (A.0) ← (A.7)
RLC A	Сдвиг аккумулятора влево через перенос	1/1	(A.n + 1) ← (A.n), n = 0—6, (A.0) ← (C), (C) ← (A.7)
RR A	Сдвиг аккумулятора вправо циклический	1/1	(A.n) ← (A.n + 1), n = 0—6, (A.7) ← (A.0)
RRC A	Сдвиг аккумулятора вправо через перенос	1/1	(A.n) ← (A.n + 1), n = 0—6, (A.7) ← (C), (C) ← (A.0)

9 **Типы данных, способы адресации данных в микроконтроллерах AVR.**

- Прямая адресация
    - Прямая регистровая адресация с одним регистром - адрес регистра содержится в самой команде (Пример - Команды работы со стеком, обмена тетрадами в регистре, некоторые логические и арифметические операции)
    - Прямая регистровая адресация с двумя регистрами - адреса регистров общего назначения содержатся в команде (Пример - Пересылка данных из регистра в регистр, большинство команд арифметических и логических операций, при этом результат сохраняется в первом регистре)
    - Прямая адресация регистра ввода-вывода - используются при пересылки данных из регистра ввода-вывода в регистры общего назначения и обратно.
    - Прямая адресация памяти данных - используются при обращении к ячейкам адресного пространства SRAM. Команды LDS и STS (32 разряда: 16 - команда + 16 адрес ячейки)
    - Косвенная адресация
    - Косвенная адресация памяти данных - обращение к ячейкам памяти, адрес которых находится в 16-разрядном индексном регистре X(R27:26),Y(R29:28),Z(R31:30)
    - Косвенная адресация памяти данных со смещением - обращение к ячейкам памяти, адрес которых находится в 16-разрядном индексном регистре X(R27:26),Y(R29:28),Z(R31:30) с 6-разрядным смещением (Пример - команды LDD, STD - пересылка из SRAM в регистр, из регистра в SRAM)
    - Косвенная адресация памяти данных с преддекрементом - сначала уменьшающие содержимое индексного регистра X,Y,Z на единицу, а затем производящие обращение по адресу (Пример LD,ST)
    - Косвенная адресация памяти данных с постинкрементом - сначала производящие обращение по адресу в индексном регистре X,Y,Z, а затем увеличивающее значение в регистре на единицу (Пример LD,ST)
    - Косвенная адресация памяти программ - используются при обращении к ячейкам памяти программы для считывания констант (в Mega и для записи во Flash память). Можно использовать в комбинации с постинкрементом и преддекриментом.
- Относительная адресация памяти программ - адрес вычисляется путем сложения программного счетчика PC и константы k задаваемой в команде. (Пример - RJMP, RCALL)
- Непосредственная адресация - с указанием одного из операндов (константы) в команде (Пример - LDI)
- Битовая адресация - с указанием одного из 8 бит любого из 32 регистров общего назначения или первой половины регистров ввода-вывода с номерами 0-31 и регистра SREG. (Пример SBI - установка бита в регистре и CBI - сброс)

Любой алгоритм, так или иначе, сводится к неким действиям над данными, будь то числа, буквы, звуки, сигналы и т.п. Так как язык программирования служит для реализации алгоритма, он имеет все необходимые средства для манипулирования данными. Для ра их по видам, введено понятие тип данных. Так как тип не может существовать без данных, а данными в языке Си может быть очень совершенно разных объектов, далее будем использовать термин объект для обозначения абстрактных данных.

Тип – это идентификатор, обозначающий принадлежность объекта к некой категории, однозначно определяющей содержимое и на действий, которые можно выполнить над ним. Например, все числа можно разделить на ряд категорий – целые, действительные, комплексные, отрицательные и т.п.

Так как работа процессора и микроконтроллера есть процесс цифровой обработки данных, поэтому так или иначе все разнообрази сводится к числам. Даже буквы есть не что иное, как числа, сопоставленные им (грубо говоря, это номер буквы по порядку в алфав этим причинам в языке Си основным видом данных являются числа, которые для удобства разделены на ряд стандартных типов (1
char – символ, то есть число, занимающее один байт (т.е. 8-битное число)
int – целое число, занимающее 2 байта (т.е. 16-битное число)
long – длинное целое число, занимающее 4 байта (т.е. 32-битное число)
long long – двойное длинное целое число, занимающее 8 байт (т.е. 64-битное число)
float – действительное число, т.е. число с плавающей точкой
double – действительное число удвоенной точности

При помощи дополнительных служебных слов определяются разновидности типов целых чисел:
signed – обозначает, что целое число имеет знак (т.е. может принимать и отрицательные значения)
unsigned – обозначает, что целое число не имеет знака, т.е. может быть только нулем или более
short – обозначает, что фактический размер целого числа вдвое меньше

Таким образом, получаются следующие типы:
unsigned char – число от 0 до 255
signed char – число от -127 до 127
unsigned int – число от 0 до 65535
signed int – число от -32767 до 32764
unsigned long – число от 0 до 4294967295
signed long – число от -2147483648 до 2147483648
и т.д.

10 Логические и арифметические операции микроконтроллеров AVR.

В группу арифметических команд входят:

1. MUL - умножение беззнаковых
2. MULS - умножение со знаком
3. MULSU - умножение беззнакового числа на число со знаком
4. FMUL - дробных беззнаковых
5. FMULS - дробных со знакомых
6. FMULSU - дробного беззнакового и дробного со знаком

Группа арифметических и логических операций по составу достаточно традиционна для архитектуры восьмиразрядных микроконтроллеров, за исключением микроконтроллеров семейства Mega. Так, в группу арифметических команд микроконтроллера ATmega8515 добавлены шесть операций умножения: беззнаковых чисел MUL, чисел со знаком MULS, беззнакового числа на число со знаком MULSU, дробных беззнаковых чисел FMUL, дробных со знаком FMULS, дробного беззнакового и дробного со знаком FMULSU. Дробные сомножители имеют формат 1.7, их произведение — формат 1.15, где справа от точки указано число дробных разрядов. Во всех операциях умножения источниками операндов являются регистры Rd и Rr, произведение формируется в регистрах R1:R0.

При выполнении операций сложения-вычитания приемником результата является один из регистров общего назначения, в котором до операции находится один из операндов. Таким образом, можно говорить о реализации АЛУ аккумуляторного типа по отношению к любому регистру общего назначения (заметим, микроконтроллеры с ядром MCS-51 имеют всего лишь один аккумулятор, что, безусловно, ухудшает эффективность обработки данных и ведет к снижению производительности в целом). Особенностью системы команд микро-

Rd, Rr — регистры общего назначения с номерами d и r;
Rdh:Rdl — пара регистров;
K — константа (данные);
P — обозначение регистра ввода-вывода;
P, b — разряд b (b = 0 — 7) регистра ввода-вывода P;
Rr(b) — разряд b (b = 0 — 7) регистра Rr;
(X), (Y), (Z) — содержимое ячеек, адресуемых регистровыми парами X, Y, Z соответственно;
s — номер разряда в регистре SREG;
PC — содержимое программного счетчика;
k — приращение в счетчике команд;
q — шестиразрядное смещение;
STACK — область памяти SRAM, адресуемая указателем стека SP;
C, Z, N, V, S, H, T, I — биты регистра состояния SREG;
d, r = 0 — 31 во всех случаях, кроме специально отмеченных.

Таблица 4.1. **Арифметические и логические операции**

Мнемоника	Описание команды	Операция	Признаки
ADD Rd, Rr	Сложение двух регистров	Rd ← Rd + Rr	Z, C, N, V, H
ADC Rd, Rr	Сложение двух регистров и переноса	Rd ← Rd + Rr + C	Z, C, N, V, H
ADIW Rdl, K	Сложение регистровой пары с константой	Rdh:Rdl ← Rdh:Rdl + K	Z, C, N, V, S
SUB Rd, Rr	Вычитание двух регистров	Rd ← Rd – Rr	Z, C, N, V, H
SUBI Rd, K	Вычитание константы из регистра	Rd ← Rd – K, d = 16 — 31	Z, C, N, V, H
SBC Rd, Rr	Вычитание двух регистров с заемом	Rd ← Rd – Rr – C	Z, C, N, V, H
SBCI Rd, K	Вычитание константы из регистра с заемом	Rd ← Rd – K – C, d = 16 — 31	Z, C, N, V, H
SBIW Rdl, K	Вычитание константы из регистровой пары	Rdh:Rdl ← Rdh:Rdl – K	Z, C, N, V, S
AND Rd, Rr	Логическое И двух регистров	Rd ← Rd ∧ Rr	Z, N, V
ANDI Rd, K	Логическое И регистра и константы	Rd ← Rd ∧ K, d = 16 — 31	Z, N, V
OR Rd, Rr	Логическое ИЛИ двух регистров	Rd ← Rd ∨ Rr	Z, N, V
ORI Rd, K	Логическое ИЛИ регистра и константы	Rd ← Rd ∨ K, d = 16 — 31	Z, N, V
EOR Rd, Rr	Логическое исключающее ИЛИ регистров	Rd ← Rd ⊕ Rr	Z, N, V
LSL Rd	Логический сдвиг влево	Rd(n + 1) ← Rd(n), Rd(0) ← 0	Z, C, N, V
LSR Rd	Логический сдвиг вправо	Rd(n) ← Rd(n + 1), Rd(7) ← 0	Z, C, N, V
ROL Rd	Сдвиг влево через перенос	Rd(0) ← C, R (n + 1) ← Rd(n), C ← Rd(7)	Z, C, N, V
ROR Rd	Сдвиг вправо через перенос	Rd (7) ← C, Rd (n) ← Rd(n + 1), C ← Rd(0)	Z, C, N, V
ASR Rd	Арифметический сдвиг вправо	Rd(n) ← Rd(n + 1), n = 0 — 6	Z, C, N, V
CP Rd, Rr	Сравнение регистров	Rd – Rr	Z, N, V, C, H
CPC Rd, Rr	Сравнение регистров с учетом заема	Rd – Rr – C	Z, N, V, C, H
CPI Rd, K	Сравнение регистра с константой	Rd – K, d = 16 — 31	Z, N, V, C, H
COM Rd	Инверсия регистра	Rd ← $FF – Rd	Z, C, N, V

Окончание табл. 4.1

Мнемоника	Описание команды	Операция	Признаки
NEG Rd	Изменение знака	Rd ← $00 – Rd	Z, C, N, V, H
SBR Rd, K	Логическое ИЛИ регистра и константы	Rd ← Rd ∨ K, d = 16 — 31	Z, N, V
CBR Rd, K	Логическое И Rd с инверсией константы	Rd ← Rd ∧ ($FF – K)	Z, N, V
INC Rd	Инкремент регистра	Rd ← Rd + 1	Z, N, V
DEC Rd	Декремент регистра	Rd ← Rd – 1	Z, N, V
TST Rd	Проверка регистра	Rd ← Rd ∧ Rd	Z, N, V
CLR Rd	Сброс регистра в нуль	Rd ← Rd ⊕ Rd	Z, N, V
SER Rd	Установка единицы в разрядах регистра	Rd ← $FF, d = 16 — 31	—

11. Привести примеры команд микроконтроллеров AVR, операндами которых являются регистры общего назначения, регистры ввода/ вывода, ячейки памяти SRAM, константы памяти Flash. Как передать данные из одного порта в другой, из порта в ячейку, из ячейки в порт, из ячейки в ячейку?

Примеры:

- Команды, которые используют два регистра общего назначения: d(Rd) и r(Rr). Этот вид адресации используют команды пересылки данных из регистра в регистр и большинство команд арифметических операций, ряд команд логических операций. При этих операциях результат операции сохраняется в регистре d (Rd).

MOV Rd, Rr Пересылка между регистрами

ADD Rd, Rr Сложение двух регистров

ADC Rd, Rr Сложение двух регистров и переноса

- Данный вид адресации используют для выполнения обмена между регистром ввода/вывода, расположенным в адресном пространстве ввода/вывода, и одним из регистров общего назначения по командам IN и OUT (рис.).

IN Rd, P Чтение регистра ввода-вывода

OUT P, Rr Запись в регистр ввода-вывода

- Данный способ адресации применяется при обращении к любой ячейке адресного пространства SRAM. Имеются всего две команды: LDS и STS, каждая длиной в два слова (32 разряда). Первое слово содержит код операции и адрес регистра общего назначения, второе - 16-разрядный адрес ячейки, к которой направлено обращение

LDS Rd, k Прямая загрузка регистра

STS k, Rr Прямое сохранение

- Микроконтроллеры AVR позволяют обратиться к ячейкам памяти программ для считывания констант, а в моделях семейства Mega – и для записи данных во Flash-

память программ, используя механизм косвенной адресации через регистр Z. При этом старшие 15 разрядов регистра определяют адрес слова, а младший нулевой разряд – младший или старший байт слова. В случае если младший разряд адреса равен 0, выбирается младший байт, в противном случае выбирается старший байт. Данный вид адресации используют команды считывания из ячейки памяти в регистр R0 (LPM) и записи в память из регистров R1:R0 (SPM).

Кроме простой косвенной адресации при чтении константы в регистр Rd можно применить косвенную адресацию с постинкрементом (команда LPM Rd, Z+). Помимо команд, связанных с передачей данных, косвенная адресация должна быть использована в командах косвенного перехода по адресу в регистре Z (IJMP) и косвенного вызова подпрограммы через регистр Z (1CALL) (рис.).

12) Назвать команды условной передачи управления микроконтроллеров AVR. Какие команды используют для проверки битов регистров и портов?

Таблица 5. Команды условной передачи управления

Мнемоника	Описание команды	Операция
SBRC Rr,b	Пропустить, если бит в регистре = 0	если (Rr(b)=0), то PC←PC + 2/3
SBRS Rr,b	Пропустить, если бит в регистре = 1	если (Rr(b)=1), то PC←PC + 2/3
SBIC P,b	Пропустить, если бит порта = 0	если (P(b)=0), то PC←PC + 2/3
SBIS P,b	Пропустить, если бит порта = 1	если (P(b)=1), то PC← PC + 2/3
BRBS s,k	Перейти, если флаг в SREG = 1	если (SREG(s)=1), то PC←PC + k + 1
BRBC s,k	Перейти, если флаг в SREG = 0	если (SREG(s)=0), то PC←PC + k + 1
BREQ k	Перейти, если равно	если (Z = 1), то PC←PC + k + 1
BRNE k	Перейти, если не равно	если (Z = 0), то PC←PC + k + 1
BRCS k	Перейти, если C = 1	если (C = 1), то PC←PC + k + 1
BRCC k	Перейти, если C = 0	если (C = 0), то PC←PC + k + 1
BRSH k	Перейти, если больше или равно	если (C = 0), то PC←PC + k + 1
BRLO k	Перейти, если меньше	если (C = 1), то PC←PC + k + 1
BRMI k	Перейти, если минус	если (N = 1), то PC←PC + k + 1
BRPL k	Перейти, если плюс	если (N = 0), то PC←PC + k + 1
BRGE k	Перейти, если больше или равно (со знаком)	если (N⊕V=0), то PC←PC+k+1
BRLT k	Перейти, если меньше (со знаком)	если (N⊕V=1), то PC←PC+k+1
BRHS k	Перейти, если межтетрадный перенос H = 1	если (H = 1), то PC←PC+k+1
BRHC k	Перейти, если межтетрадный перенос H = 0	если (H = 0), то PC ← PC + k + 1
BRTS k	Перейти, если флаг T = 1	если (T = 1), то PC ← PC + k + 1
BRTC k	Перейти, если флаг T = 0	если (T = 0), то PC ← PC + k + 1
BRVS k	Перейти, если флаг переполнения V= 1	если (V = 1), то PC ← PC + k + 1
BRVC k	Перейти, если флаг переполнения V = 0	если (V = 0), то PC ← PC + k + 1
BRIE k	Перейти, если флаг прерывания I = 1	если (I = 1), то PC ← PC + k + 1

Их много, но я бы указал лишь те, что про порты и регистры, а также про флаги C и T.

13. **Сложение и вычитание 16-разрядных двоичных и двоично-десятичных операндов в 8 разрядном процессоре**

Двоичные:

1. Складываются (вычитаются) младшие байты слов.
2. Затем складываются (вычитаются) старшие байты слов с учетом переноса (заема), если он возник на шаге 1.

Двоично-десятичные:

Если оба операнда имеют одинаковые знаки, то выполняют сложение модулей этих чисел, а знаковый разряд суммы определяют по знаку одного из слагаемых. Если операнды имеют разные знаки, то предварительно знак суммы устанавливают по знаку первого операнда А. Затем производят вычитание модулей чисел. Если полученная разность больше нуля, знак суммы сохраняется без изменений. Если разность меньше нуля, следует найти дополнительный код разности и изменить знак суммы на противоположный.

При сложении чисел в одном разряде суммы S можно получить результаты:

1) $s_i \leq 9$, не требует коррекции;

2) $10 \leq s_i \leq 15$, требует коррекции путем увеличения s, на 6 с об

образованием переноса из тетрады;

3) $s_i > 15$, требует коррекции путем увеличения s_i на 6. В этом случае переноса из тетрады образуется автоматически при сложении операндов до выполнения коррекции.

MCS-51:

В микроконтроллерах MCS-51 коррекция осуществляется аппаратно при выполнении команды двоично-десятичной коррекции содержимого аккумулятора DA A.

Программа для вычитания двухбайтовых двоично-десятичных чисел для MCS-51 представляет собой циклическую процедуру получения дополнительного кода вычитаемого и сложения его с уменьшаемым А с последующей коррекцией результата.

AVR:

При отсутствии схемы двоично-десятичной коррекции, как в микроконтроллерах AVR, поступают следующим образом. При сложении двоично-десятичных чисел добавляют число, каждый разряд которого равен б. В этом случае, если вычисляемая поразрядная сумма $s_i \leq 9$, перенос из тетрады не возникает и избыточное значение 6 подлежит удалению. Во всех остальных случаях добавленное в разряд значение 6 удаляется автоматически с переносом из тетради в процессе сложения.

Вычитание беззнаковых чисел A - B можно произвести по алгоритму:

1) выполняют вычитание A - B, образуя первый промежуточный результат R'. Если в результате операции образуется перенос из старшей тетрады, результат является положительным. При отсутствии переноса результат отрицательный и его следует перевести в дополнительный код R";
2) коррекция положительного результата осуществляется по правилу, сформулированному для сложения двоично-десятичных чисел. Коррекция отрицательного результата выполняется иначе: если произошел перенос из i-й тетрады при вычитании A-B, то из i-й тетрады r_i'' вычитается 6. При выполнении коррекции перенос из тетрады не должен изменять содержимое следующей тетриады промежуточного результата.

14. **Как перемножить 16-разрядные двоичные операнды в 8-разрядном микроконтроллере?**

Количество итераций умножения n определяется числом разрядов множителя. Поскольку в процессе умножения на каждой итерации осуществляется сдвиг множителя B на один разряд вправо, на месте освобождаемого разряда можно записать выталкиваемый при сдвиге вправо разряд произведения C. Таким образом, 2n-разрядное произведение можно получить, объединив содержимое n-разрядного регистра, в котором формируется старшая часть произведения, и регистр B, в котором после выполнения умножения окажется младшая часть произведения.
При умножении двухбайтовых сомножителей A (AH:AL) и B (BH:BL) необходимо принять во внимание, что

$$A \cdot B = (2^8 AH + AL)\ (2^8 BH + BL) = (AL \cdot BL) + (2^8 AH \cdot BL) + \\ + (2^8 AL \cdot BH) + (2^{16} AH \cdot BH).$$

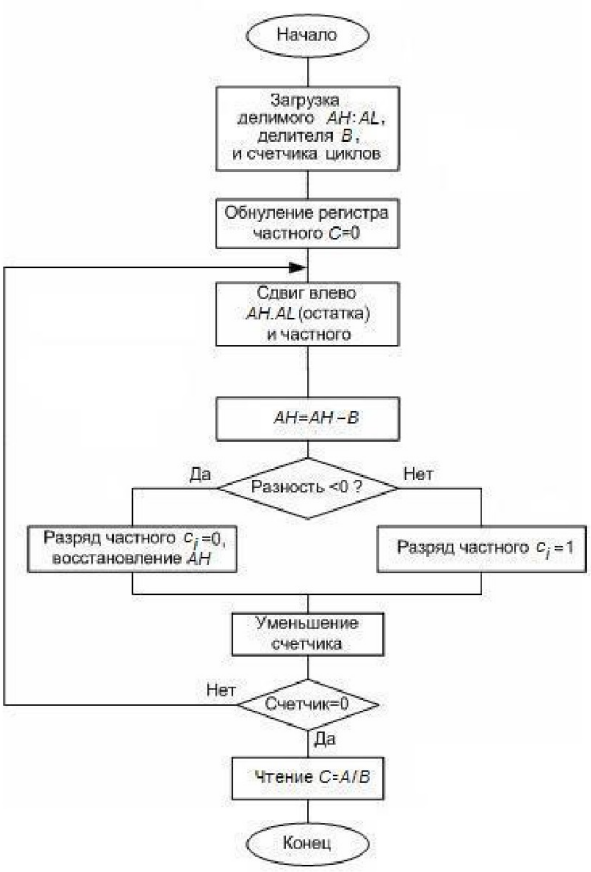
Из этого следует: что алгоритм умножения можно представить как последовательность из четырех операций умножения однобайтовых операндов $AL \cdot BL, AH \cdot BL, AL \cdot BH, AH \cdot BH$ **с последующим суммированием взвешенных частичных произведений.**

15) Описать процедуру деления с восстановлением остатка (на примере 16-разрядного делимого и 8-разрядного делителя).

Какие проверки выполняют перед началом деления?

Перед выполнением деления необходимо проверить условие – делитель должен быть больше старшего слова делимого (*B* > *AH*).
При делении целых чисел можно использовать алгоритм деления без восстановления остатка и алгоритм с восстановлением остатка.

Схема алгоритма с восстановление остатка приведена на рис. 3. Алгоритм деления представляет собой итерационную процедуру. На каждой итерации сначала удваивается делимое (на первой итерации) или остаток (на всех последующих) путем сдвига влево на один разряд, затем вычитается делитель и определяется цифра частного по знаку разности. Если разность положительная, определяемая на данной итерации цифра частного *ci* =1, если разность отрицательная, цифра частного *ci* =0. **Восстановление остатка** выполняется путем сложения делителя с остатком после вычитания на текущей итерации деления. Деление выполняется до получения всех цифр частного.



Ниже приведен пример деления 16-разрядного числа A на 8-разрядное число B с восстановлением остатка.

$A = 1024 = 00000100.00000000$, $B = 10 = 00001010$,

$-B = [-10]_{доп} = 11110110$,

$C = c_7c_6c_5c_4c_3c_2c_1c_0$ – частное, x – бит, свободно определяемый при сдвиге.

+	00000100.00000000	делимое A (AH AL)
	11110110	пробное вычитание B
	11111010	так как разность меньше 0, переполнения нет
+	00001000.0000000x	сдвиг A влево
	11110110	вычитание B
	11111110	1-й остаток меньше 0, разряд частного $c_7 = 0$
+	00010000.000000xx	сдвиг влево восстановленного AH
	11110110	вычитание B
	00000110	2-й остаток больше 0, разряд частного $c_6 = 1$
+	00001100.00000xxx	сдвиг остатка
	11110110	вычитание B
	00000010	3-й остаток, $c_5 = 1$
+	00000100.0000xxxx	сдвиг остатка
	11110110	вычитание B
	11111010	4-й остаток, $c_4 = 0$
+	00001000.000xxxxx	сдвиг восстановленного AH
	11110110	вычитание B
	11111110	5-й остаток, $c_3 = 0$
+	00010000.00xxxxxx	сдвиг восстановленного AH
	11110110	вычитание B
	00000110	6-й остаток, $c_2 = 1$
+	00001100.0xxxxxxx	сдвиг остатка
	11110110	вычитание B
	00000010	7-й остаток, $c_1 = 1$
+	00000100.xxxxxxxx	сдвиг остатка
	11110110	вычитание B
	11111010	8-й остаток, $c_0 = 0$
+	11111010	
	00001010	прибавление B
	00000100	восстановлен остаток $AH = 4$

$C = 0b01100110 = 102$

16. Схемы и описание работы портов ввода-вывода P0, P1 микроконтроллера MCS-51.

Порт P0 может быть использован для организации шины данных при работе микроконтроллера с внешней памятью данных или программ, при этом через него выводится младший байт адреса A0 - A7, выдается из микроконтроллера или принимается в микроконтроллер байт данных. Во время доступа к внешней памяти во все триггеры - защелки порта P0 аппаратно записываются "1" (т.е. содержимое порта теряется). Кроме того, через порт P0 передаюься данные при программировании внутреннего ППЗУ, и читается содержимое внутренней памяти программ при работе с программатором. **При сбросе микросхемы во все разряды порта записываются '1'.** Схема порта P0 отличается от всех других портов тем, что у этого порта нет внутреннего генератора тока. Поэтому при работе с этим портом приходится подключать внешние резисторы.

- **Порт P1** может быть использован для чтения внутренней памяти программ или для передачи младшего байта адреса при программировании внутреннего РПЗУ. В младших моделях микроконтроллера семейства других альтернативных функций у порта P1 нет. **При сбросе микросхемы во все разряды порта записываются '1'.**

Адрес	Ст. зн. Разр							Мл. зн. Разр	
090h	*CEX4* **P1.7**	*CEX3* **P1.6**	*CEX2* **P1.5**	*CEX1* **P1.4**	*CEX0* **P1.3**	ECI **P1.2**	**T2EX P1.1**	**T2 P1.0**	**P1**
	87h	**86h**	**85h**	**84h**	**83h**	**82h**	**81h**	**80h**	

<u>P1.0</u> T2 -внешний вход таймера/счетчика 2
<u>P1.1</u> T2EX -вход управления перезагрузки/захвата таймера
<u>P1.2</u> *ECI* - внешний вход набора программируемых счетчиков (PCA)
<u>P1.3</u> *CEX0* -внешний вход/выход для вывода ШИМ или сравнения/захвата модуля 0
<u>P1.4</u> *CEX1* -внешний вход/выход для вывода ШИМ или сравнения/захвата модуля 0
<u>P1.5</u> *CEX2* -внешний вход/выход для вывода ШИМ или сравнения/захвата модуля 0
<u>P1.6</u> *CEX3* -внешний вход/выход для вывода ШИМ или сравнения/захвата модуля 0
<u>P1.7</u> *CEX4* -внешний вход/выход для вывода ШИМ или сравнения/захвата модуля 0

Примечание: альтернативные функции, выделенные:

1. **<u>жирным подчеркнутым текстом</u>** - присутствуют во всех микросхемах
2. **жирным текстом** - отсутствуют в микросхемах 8051,8031, КР1816ВЕ31, КР1816ВЕ51, КР1816ВЕ751 КР1830ВЕ31, КР1830ВЕ51, КР1830ВЕ751;
3. *курсивом* - присутствуют только в микросхемах 8X51FA, FB, FC и GB

17. Структура порта для параллельного ввода-вывода в микроконтроллерах AVR.

В состав каждого порта Px входят три регистра с именами DDRx, PORTx и PINx. В микроконтроллерах семейства Classik регистр PINx не имеет аппаратной реализации. Это имя используется для чтения линий интерфейса. На рис. 7.11, *а* изображена общая структура восьмиразрядных портов Px, на рис. 7.11, *б* — структурная схема одного разряда порта PD.y (y = 0 – 7) микроконтроллера AT90S8515.

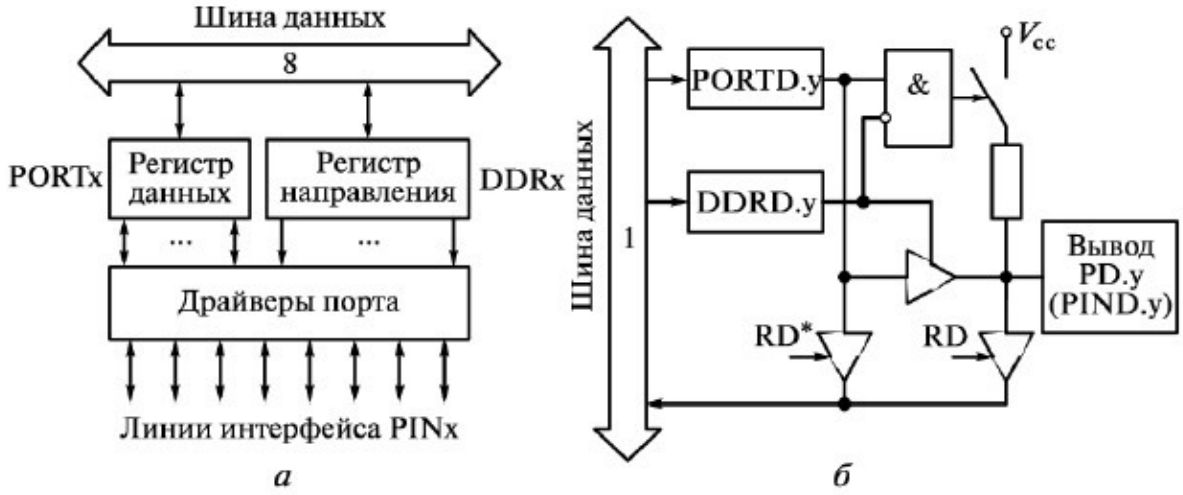


Рис. 7.11. Структура порта Px (*а*) и схема одного разряда порта PD (*б*)

Состояние разряда DDRD.y определяет направление передачи бита данных через вывод порта PD.y. При DDRD.y = 0 вывод порта PD.y является входом, при DDRD.y = 1 — выходом.

В режиме входа состояние разряда PORTD.y определяет состояние вывода PD.y. При PORTD.y = 1 вывод порта через внутренний резистор подключается к шине питания V_{cc}. При PORTD.y = 0 резистор отключается, вывод PD.y находится в высокоимпедансном состоянии (Z-состояние).

В режиме выхода состояние разряда PORTD.y определяет значение сигнала на выводе PD.y. При PORTD.y = 0 на выводе устанавливается напряжение низкого уровня, при PORTD.y = 1 — высокого.

При пуске и перезапуске микроконтроллера все разряды регистров DDRx и PORTx сбрасываются в нулевое состояние, вследствие чего выводы портов работают в режиме входа и находятся в Z-состоянии.

При совместном использовании всех разрядов порта для ввода байта данных используются команды с мнемоникой IN Rd, PINx, для вывода — OUT PORTx, Rr (d, r = 0—31).

Значение выходного сигнала на отдельном выводе порта можно задать с помощью команд установки нуля (CBI PORTD.y) и установки единицы (SBI PORTD.y). Значение входного сигнала на отдельном выводе порта можно проверить, используя команды условного перехода SBIC PIND.y или SBIS PIND.y, которые предусматривают пропуск следующей команды по нулевому или единичному значению PD.y.

18. Построить временные диаграммы протокола передачи данных от микроконтроллера (приема данных в микроконтроллер) по сигналу запроса от внешнего устройства с использованием квитирующих сигналов.

На диаграммах сигналов квитирования:

ЗПР и **ПОДТВ** – сигналы квитирования

ДАННЫЕ выставляются на шине связи

а) Передача данных от МК, запрос от ВУ

б) Прием данных в МК, запрос от ВУ

19. Построить временные диаграммы протокола приема данных от внешнего устройства (передачи данных во внешнее устройство) по сигналу запроса от микроконтроллера с использованием квитирующих сигналов.

а) Прием данных от ВУ, запрос от МК

б) Передача данных в ВУ, запрос от МК

20. Системы прерываний микроконтроллеров MCS-51, AVR. Идентификация прерываний в одноуровневой системе прерываний.

Обработка прерываний в микроконтроллерах MCS-51. Как уже говорилось, применяемую в микроконтроллерах с архитектурой MCS-51 систему прерываний по способу организации и принципу действия можно отнести к системам радиального типа. Все запросы, внешние и внутренние, обрабатываются путем аппаратного вызова прерывающих подпрограмм через таблицу прерываний по фиксированным адресам. Всего обрабатываемых запросов прерываний — 5. Для внешних прерываний зарезервированы адреса 0003h и 0013h, для внутренних прерываний от двух таймеров — адреса 000Bh и 001Bh, от последовательного канала ввода-вывода — 0023h. Для каждого прерывания в таблице прерываний отведено 8 байт. Из-за этого длинные процедуры прерывания размещаются в памяти программ за пределами таблицы прерываний, а в соответствующей секции таблицы размещается ссылка на процедуру (команда JMP).

Каждый из запросов прерываний может быть разрешен или запрещен с помощью регистра маски прерываний IE, формат которого приведен ниже.

Номер разряда	7	6	5	4	3	2	1	0
Имя	EA	—	—	ES	ET1	EX1	ET0	EX0

Здесь назначение битов: EX0, EX1 — маски внешних прерываний; ET0, ET1 — маски прерываний от таймеров T0 и T1; ES — маска запроса от последовательного канала; EA — общая маска прерываний.

При значении 1 соответствующего бита регистра прерывание разрешено, при 0 — запрещено. Бит EA при значении 1 разрешает работу системы прерываний, при 0 все прерывания запрещены.

Система прерываний MCS-51 является двухуровневой. Запросы верхнего уровня имеют более высокий приоритет по сравнению с запросами нижнего уровня. Особенностью системы прерываний микроконтроллеров MCS-51 является возможность «прикрепления» запроса к верхнему или нижнему уровню прерываний. Достигается это с помощью регистра приоритетов IP, имеющего следующий формат:

Номер разряда	7	6	5	4	3	2	1	0
Имя	—	—	—	PS	PT1	PX1	PT0	PX0

Здесь PX0, PX1 — приоритеты внешних прерываний $\overline{\text{INT0}}$, $\overline{\text{INT1}}$; PT0, PT1 — приоритеты прерываний от таймеров T0 и T1; PS — приоритет запроса от последовательного канала.

Система прерываний микроконтроллеров AVR. Встроенный контроллер прерываний обрабатывает внешние прерывания и прерывания от периферийных устройств микроконтроллера (таймеров, портов последовательного ввода-вывода, аналогового компаратора и др.). Число прерываний зависит от конкретной модели микроконтроллера. Например, в моделях ATmega64x, ATmega128x их число достигает 34, в том числе: 8 — внешних, 14 — от таймеров, 8 — от последовательных каналов ввода-вывода, 2 — от аналоговых устройств, по одному — от памяти EEPROM и Flash-памяти. Кроме того, в этих микроконтроллерах реализована многоуровневая система приоритетных прерываний. Таблица прерываний располагается в начальной области памяти программ. Размер вектора в таблице зависит от объема памяти программ и составляет в зависимости от модели одно или два слова (соответственно 2 или 4 байта). Для перехода к прерывающей программе в первом случае используются команды относительного перехода RJMP, во втором — JMP. При этом команды RJMP занимают соседние адреса, начиная с адреса $0001, а команды JMP располагаются по четным адресам, начиная с адреса $0002.

В качестве примера рассмотрим более подробно систему прерываний модели AT90S8515. Все прерывания, а их в модели 12, являются маскируемыми. Адреса, маски и флаги прерываний указаны в табл. 9.1.

Таблица 9.1. **Адреса и маски прерываний AT90S8515**

Запрос	Адрес	Маска	Флаг
RESET	000	—	—
INT0	001	GIMSK.6	GIFR.6
INT1	002	GIMSK.7	GIFR.7
T/C1 CAPT	003	TIMSK.3	TIFR.3
T/C1 COMPA	004	TIMSK.6	TIFR.6
T/C1 COMPB	005	TIMSK.5	TIFR.5
T/C1 OVF	006	TIMSK.7	TIFR.7
T/C0 OVF	007	TIMSK.1	TIFR.1
SPI STC	008	SPCR.7	SPSR.7
UART RXC	009	UCR.7	USR.7
UART DRE	00A	UCR.5	USR.5
UART TXC	00B	UCR.6	USR.6
ANA COMP	00C	ACSR.3	ACSR.4

Положение вектора в таблице прерываний определяет приоритет соответствующего прерывания. Запрос с меньшим адресом имеет более высокий приоритет. Флаг общего прерывания I расположен в регистре состояния микроконтроллера SREG (7-й бит). При I = 1 и единичном значении маски запроса прерывание данного типа разрешено. При поступлении запроса устанавливается флаг прерывания в одном из регистров ввода-вывода (GIFR, TIFR и др.), который может вызвать аппаратное прерывание. Состояние флага может быть также опрошено программой.

В микроЭВМ обычно используется одноуровневая система прерываний, т. е. сигналы "Запрос на прерывание" от всех ВУ поступают на один вход процессора. Поэтому возникает проблема идентификации ВУ, запросившего обслуживание, и реализации заданной очередности (приоритета) обслуживания ВУ при одновременном поступлении нескольких сигналов прерывания. Существуют два основных способа идентификации ВУ, запросивших обслуживания:

- программная идентификация (метод Полинга);

- аппаратная идентификация.

Организация прерываний с программным опросом готовности предполагает наличие в памяти микроЭВМ единой подпрограммы обслуживания прерываний от всех внешних устройств.

Обслуживание ВУ с помощью единой подпрограммы обработки прерываний производится следующим образом. В конце последнего машинного цикла выполнения очередной команды основной программы процессор проверяет наличие требования прерывания от ВУ. Если сигнал прерывания есть и в процессоре прерывание разрешено, то процессор переключается на выполнение подпрограммы обработки прерываний. После сохранения содержимого регистров процессора, используемых в подпрограмме, начинается последовательный опрос регистров состояния контроллеров всех ВУ, работающих в режиме прерывания. Как только подпрограмма обнаружит готовое к обмену ВУ, сразу выполняются действия по его обслуживанию. Завершается подпрограмма обработки прерывания после опроса готовности всех ВУ и восстановления содержимого регистров процессора.

Приоритет ВУ в микроЭВМ с программным опросом готовности внешнего устройства однозначно определяется порядком их опроса в подпрограмме обработки прерываний. Чем раньше в подпрограмме опрашивается готовность ВУ, тем меньше время реакции на его запрос и выше приоритет. Необходимость проверки готовности всех внешних устройств существенно увеличивает время обслуживания тех ВУ, которые опрашиваются последними. Это является основным недостатком рассматриваемого способа организации прерываний. Поэтому обслуживание прерываний с опросом готовности ВУ используется только в тех случаях, когда отсутствуют жесткие требования на время обработки сигналов прерывания внешних устройств.

Аппаратный способ реализации системы прерываний (Дейзи цепь) позволяет устранить указанный недостаток. При такой организации системы прерываний ВУ, запросившее обслуживания, само идентифицирует себя с помощью вектора прерывания - адреса ячейки основной памяти микроЭВМ, в которой хранится либо первая команда подпрограммы обслуживания прерывания данного ВУ, либо адрес начала такой подпрограммы. Таким образом, процессор, получив вектор прерывания, сразу переключается на выполнение требуемой подпрограммы обработки прерывания. В микроЭВМ с векторной системой прерывания каждое ВУ должно иметь собственную подпрограмму обработки прерывания.

Аппаратный опрос готовности ВУ производится гораздо быстрее, нежели программный (время распространения сигнала по линии – 1 машинный такт). Но если обслуживания запросили одновременно два или более ВУ, обслуживание менее приоритетных ВУ будет отложено на время обслуживания более приоритетных, как и в системе прерывания с программным опросом.