Цифровые вычислительные устройства и микропроцессоры приборных комплексов

Микроконтроллеры-II: особенности архитектуры (конвейеризация, стек, прерывания, периферия)

Соловьёв Сергей Юрьевич, канд. техн. наук, доцент кафедры 303
Ушаков Андрей Николаевич, ассистент кафедры 303

1. Конвейеризация

- **Конвейеризация** (или конвейерная обработка – **pipelining**) – способ повышения производительности (быстродействия) процессора, основанный на разделении подлежащей исполнению команды (инструкции) на более мелкие части, называемые этапами или ступенями, и выделении для каждой из них отдельного блока аппаратуры.
- Для выполнения каждой команды процессора требуется осуществить некоторое количество однотипных операций, таких как:
  - выборка команды из ОЗУ,
  - дешифрация команды,
  - адресация операнда в ОЗУ,
  - выборка операнда из ОЗУ, выполнение команды,
  - запись результата в ОЗУ.
- Следовательно, имея отдельные блоки аппаратуры (**ступени конвейера**) для выполнения каждого этапа и организовав передачу данных от одного этапа к другому (от одной ступени к другой), можно совместить выполнение нескольких разных команд во времени. Производительность при этом возрастает благодаря тому, что одновременно на различных ступенях конвейера выполняются несколько команд.

Конвейеризация (2)

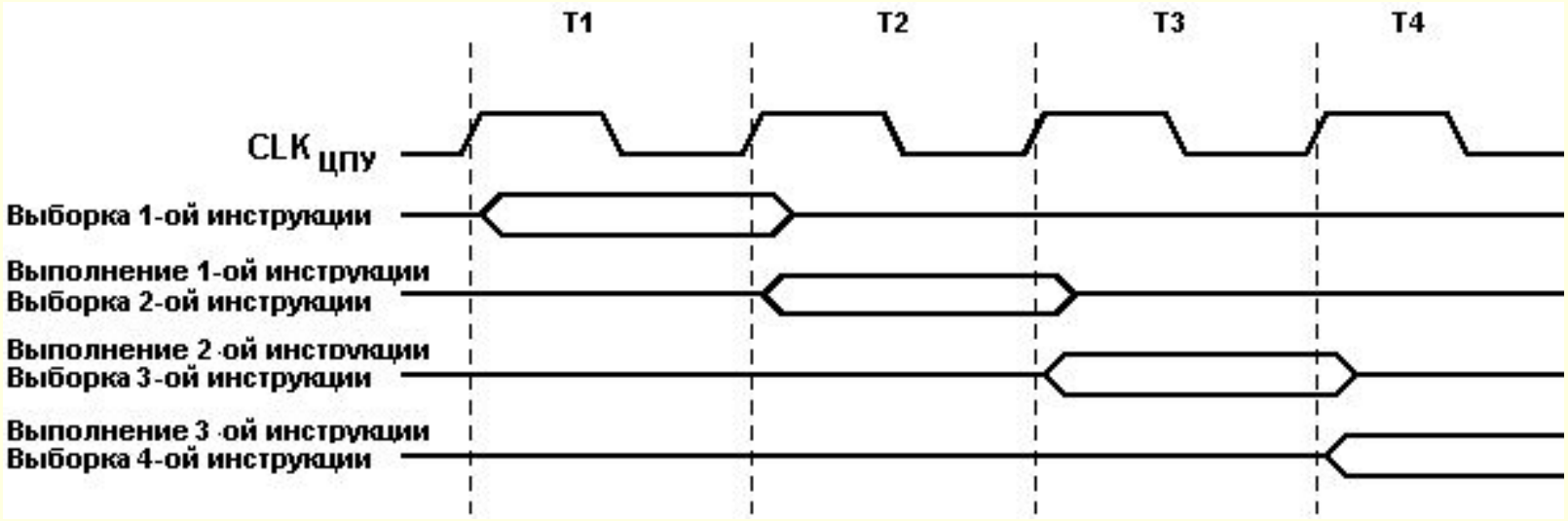
- После освобождения *k*-й ступени конвейера она сразу приступает к работе над следующей командой. Если предположить, что каждая ступень конвейера тратит единицу времени на свою работу, то выполнение команды на конвейере длиной в *n* ступеней займёт *n* единиц времени, однако в самом оптимистичном случае **результат выполнения каждой следующей команды будет получаться через каждую единицу времени**.
- Действительно, **при отсутствии конвейера** выполнение команды займёт *n* единиц времени (так как для выполнения команды по прежнему необходимо выполнять выборку, дешифрацию и т. д.), и для исполнения *m* команд **понадобится *n*·*m* единиц времени**; **при использовании конвейера** (в самом оптимистичном случае) для выполнения *m* команд понадобится **всего лишь *n* + *m* единиц времени**.

Конвейеризация (3)

- Факторы, снижающие эффективность конвейера:
  - **простой конвейера**, когда некоторые ступени не используются (например, адресация и выборка операнда из ОЗУ не нужны, если команда работает с регистрами);
  - **ожидание**: если следующая команда использует результат предыдущей, то последняя не может начать выполняться до выполнения первой (это преодолевается при использовании внеочередного выполнения команд, out-of-order execution);
  - **очистка конвейера** при попадании в него команды перехода (эту проблему удаётся сгладить, используя предсказание переходов).
- **Наряду с конвейеризаций** для повышения производительности процессора на основе идеи совмещения операций, при котором аппаратура компьютера в любой момент времени выполняет одновременно более одной базовой операции, **используют параллелизм**.
- При **параллелизме** совмещение операций достигается путём воспроизведения в нескольких копиях аппаратной структуры. Высокая производительность обеспечивается одновременной работой всех элементов структур, осуществляющих решение различных частей задачи.

Конвейеризация в AVR-микроконтроллерах (1)

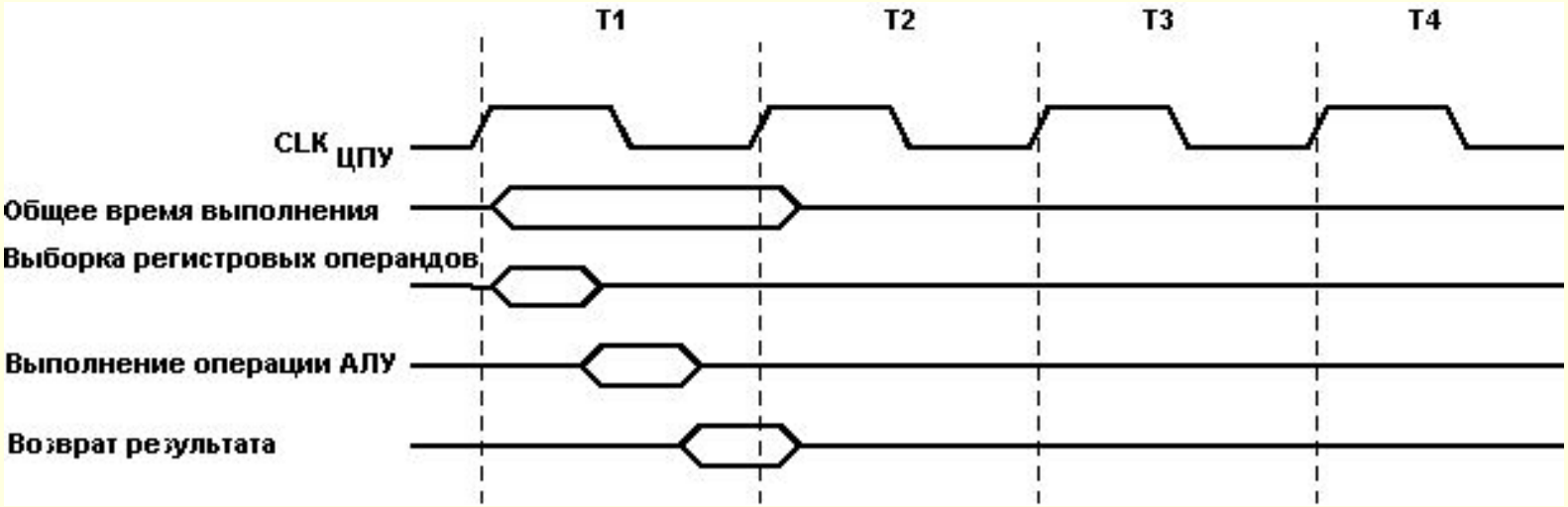
- Конвейеризации в AVR-микроконтроллерах обеспечивает удельную производительности до 1 миллиона операций в секунду на каждый МГц тактовой частоты.



Выборка и выполнение команд в AVR-микроконтроллерах

Конвейеризация в AVR-микроконтроллерах (2)

- За один такт синхронизации АЛУ выполняет действие над двухрегистровым операндом и возвращает результат обратно в регистр-получатель.

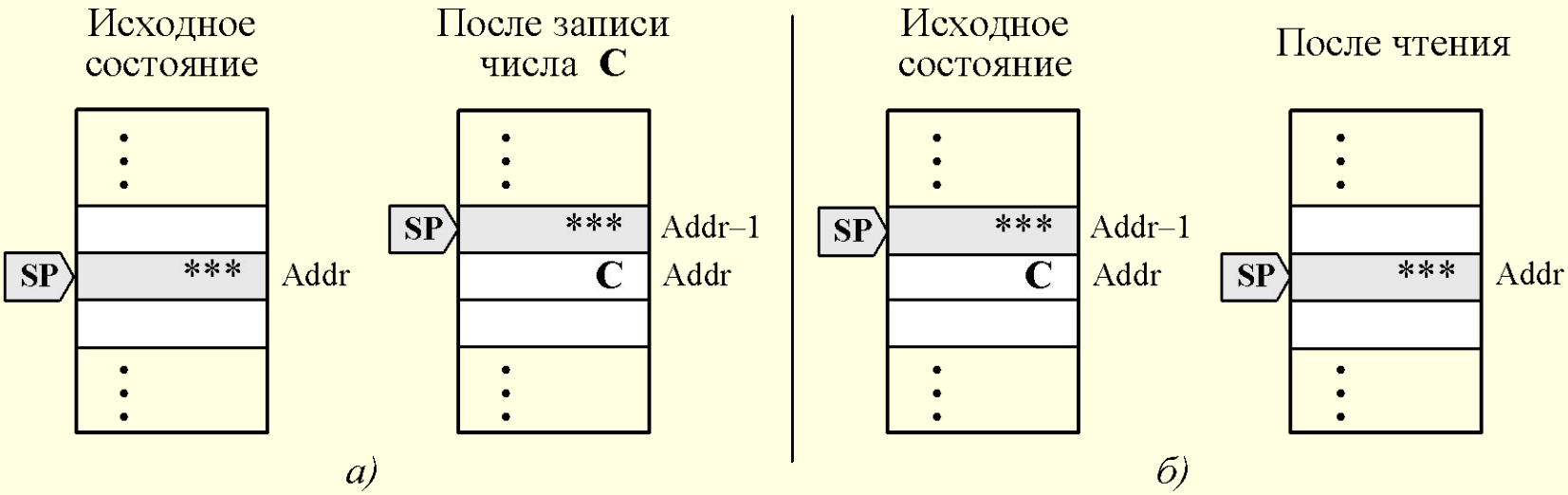


Этапы конвейерного выполнения команд в AVR-микроконтроллерах

2. Стек

- **Стек** – специальным образом организованная последовательность ячеек памяти с дисциплиной обслуживания «последним пришёл – первым вышел» (**LIFO**: Last-In – First-Out).
- При занесении в стек новых данных предыдущие данные сохраняются, но становятся временно недоступными («опускаются»). Выгружаются из стека («поднимаются», «выталкиваются») данные в обратном порядке.
- Стек может быть реализован как аппаратно, так и программно.
- **Аппаратный стек** (Hardware Stack) выполняется непосредственно в составе процессора в виде набора специальных регистров и, как правило, имеет небольшую глубину.
- **Программный стек** (Software Stack) организуется в оперативной памяти; глубина определяется размером и степенью использования памяти.
- Адрес очередной свободной ячейки стека содержится в специальном регистре – указателе стека **SP** (Stack Pointer).
- При записи в стек число помещается в ячейку с адресом, содержащимся в указателе стека, после чего содержимое указателя стека уменьшается на единицу (точнее – на размер записываемых в стек данных).
- При чтении из стека производится выборка содержимого ячейки по адресу, на единицу большему содержимого указателя стека.
- Таким образом, при записи стека и чтении из него содержимое указателя стека изменяется.

Стек (2)



Операции записи в стек (а) и чтения из стека (б)

*** – очередная свободная ячейка стека; Addr – адрес

3. Прерывания

- При работе реальной микропроцессорной системы в ней или вне её могут произойти события, требующие немедленной реакции. Такая реакция обеспечивается **процедурой прерывания** (interrupt), которая состоит в том, что выполнение текущей программы приостанавливается, запоминается состояние на момент прерывания, выполняется другая программа, после чего восстанавливается сохранённое до прерывания состояние процессора и продолжается выполнение прерванной программы.
- Сигнал, вызвавший прерывание текущей программы, называется **запросом на прерывание** (interrupt request – IRQ); источник этого сигнала – **источником прерывания**; последовательность действий, выполняемая по запросу на прерывание, – **обслуживанием прерывания**, а выполняемая по прерыванию программа – **подпрограммой обработки прерывания** (interrupt handler, interrupt routine).
- Различают два типа источников прерывания – аппаратные и программные.
- Источниками **аппаратных прерываний** служат внешние и внутренние периферийные устройства.

Прерывания (2)

- **Запросом** на прерывание **от внешнего источника** является активный сигнал на соответствующем выводе процессора; источник прерывания определяется по выводу, на котором появляется такой сигнал.
- К источникам **программного прерывания** относятся специальные команды прерываний (trap) – управляемые программные прерывания и особые условия (exception – исключение) – неуправляемые программные прерывания, являющиеся реакцией процессора на исключительную ситуацию, возникшую при выполнении некоторой команды (переполнение, деление на нуль и т. п.).
- **Запросом** на прерывание **от программного источника** является непосредственно команда прерывания или установка бита (битов), фиксирующих возникновение особого условия. Общее количество источников аппаратных и программных прерываний может быть различным – от единиц до нескольких десятков.
- **Процедура обслуживания прерываний** по запросам от нескольких источников в различных процессорах выполняется по-разному. Тем не менее основные принципы реализации механизма прерываний являются общими.

Прерывания (3)

- **Управление процедурой прерываний** осуществляется специальными устройствами в составе аппаратного обеспечения процессора (контроллерами, схемами управления и т. п.).
- **Основными средствами управления прерываниями** являются:
  - векторы прерываний;
  - приоритеты прерываний;
  - операция маскирования прерываний;
  - флаги прерываний.
- В микроконтроллерах указанные средства управления прерываниями реализуются следующим образом.
- Для управления прерываниями от *N* источников в адресном пространстве памяти программ выделяется специальная область из *N* ячеек памяти (или *N* блоков, состоящих из нескольких ячеек). В каждой из этих ячеек размещаются команды перехода к соответствующей подпрограмме обработки прерывания или (в случае блока из нескольких ячеек) непосредственно команды, которые необходимо выполнить по запросу на прерывание.

Прерывания (4)

- Эти ячейки памяти (блоки) называются **векторами прерываний** (или просто векторами), адрес ячейки (первой ячейки каждого блока) – адресом вектора прерывания. Таким образом, каждому источнику прерывания ставится в соответствие свой адрес вектора прерывания. Совокупность *N* векторов образует **таблицу векторов прерываний**, которая обычно располагается начиная с нулевого адреса памяти программ.
- **Приоритеты прерываний** (interrupt priority) определяют очерёдность обслуживания запросов на прерывания. Введение приоритетов необходимо, если возможно одновременное (в течение одного периода тактовой частоты) поступление запросов на прерывание от различных источников или поступление нового запроса на прерывание во время обслуживания прерывания по ранее поступившему запросу.
- Виды и структура приоритетов прерываний определяются архитектурой процессора.
- Наиболее простым способом задания приоритетов является последовательное присвоение значений приоритетов в таблице векторов прерываний от высшего к низшему.
- Высший приоритет всегда имеет аппаратный сброс; далее располагаются векторы прерываний от других источников.

Прерывания (5)

- Для того, чтобы запретить обслуживание неиспользуемых прерываний, служит операция **маскирования**.
- В зависимости от возможности маскирования источники прерывания делятся на маскируемые (maskable), прерывания от которых могут разрешаться или запрещаться, и немаскируемые (nonmaskable), прерывания от которых не могут запрещаться.
- Маскирование может быть общим и индивидуальным. При общем (глобальном) маскировании все прерывания, кроме немаскируемых, запрещены независимо от их индивидуального маскирования. Индивидуальное маскирование позволяет запрещать (разрешать) прерывание от каждого источника отдельно.
- **Флаги прерываний** представляют собой разряды специальных регистров, устанавливающиеся при поступлении запроса на прерывание от некоторого источника.
- Процедура обслуживания прерывания может быть упрощённо представлена состоящей из следующих этапов:
  - приёма запросов на прерывание;
  - арбитража прерываний;
  - выполнения подпрограммы обслуживания прерывания.

Прерывания (6)

- При **приёме запроса на прерывание** от немаскируемого источника сразу осуществляется переход к следующему этапу его обслуживания – арбитражу.
- Запрос на прерывание от маскируемого источника обрабатывается по более сложному алгоритму.
  - **При поступлении запроса** устанавливается соответствующий флаг прерывания.
  - **Проверяется наличие** общего **маскирования** прерываний. Если режим общего маскирования установлен, то запросы на прерывания от всех маскируемых источников игнорируются и продолжается выполнение текущей программы. Если режим общего маскирования не задан, то запрещение или разрешение данного прерывания определяется наличием (отсутствием) индивидуального маскирования. Если данное прерывание замаскировано, то запросы на прерывание от данного источника запрещены и продолжается выполнение текущей программы.
  - В противном случае прерывания от данного источника разрешены и для него начинается следующий этап обслуживания – **арбитраж**.

Прерывания (7)

- **Арбитраж прерываний** служит для определения прерывания с наивысшим приоритетом из очереди поступивших запросов на прерывание. После арбитража начинается выполнение выбранного запроса на прерывание.
- **Выполнение прерывания** состоит в переходе к подпрограмме обслуживания прерывания, её выполнении и возврате к выполнению текущей программы. Перед выполнением прерывания производится общее маскирование, т. е. запрещение всех прерываний, кроме немаскируемых, а также очищается флаг обслуживаемого прерывания. Собственно выполнение прерывания начинается с обращения к вектору прерывания обслуживаемого источника.
- Обслуживаемое прерывание может быть прервано по запросам от источников, имеющих более высокий приоритет. Прерывания, для обслуживания которых прерывается выполнение подпрограммы обработки другого прерывания, называются **вложенными**. Процедура их обслуживания аналогична обслуживанию обычных прерываний; отличие состоит лишь в том, что прерывается выполнение не основной программы, а подпрограммы обработки прерывания от источника с более низким приоритетом.

Прерывания (8)

- В микропроцессорных системах механизм прерываний используется для обмена информацией с различными устройствами ввода-вывода. Такой способ обмена данными называется **обменом по прерываниям**.
- Типичными примерами запросов на прерывание являются запросы по готовности результата аналого-цифрового преобразования, готовности устройства к приёму (передаче) информации, переполнению некоторого регистра и т. п.
- Использование механизма прерываний позволяет значительно повысить производительность системы при работе с медленно действующими устройствами, обслуживание которых в таком случае занимает процессорное время только при их готовности к обмену.

Прерывания (9)

- **В AVR-микроконтроллерах** механизм прерываний реализуется следующим образом.
- Управление прерываниями осуществляется с помощью схемы прерываний.
- Область векторов прерываний размещается в начале памяти программ; каждый вектор состоит из одной ячейки. При необходимости область векторов прерываний может быть перемещена в другое место памяти программ.
- Прерывания с младшими адресами имеют бóльший уровень приоритета. Источниками всех прерываний являются аппаратные средства (внешние или внутренние); источники программных прерываний отсутствуют. Все источники прерываний являются маскируемыми. Общее маскирование осуществляется очисткой бита **I** глобального разрешения прерываний в регистре состояния **SREG**.
- Количество векторов прерываний в **AVR**-микроконтроллерах составляет от 3 до 35 в зависимости от типа. Например, в микроконтроллере **ATmega8535** имеется 21 вектор прерывания: 3 от внешних источников и 18 от внутренней периферии.

4. Периферийные устройства

- В состав микроконтроллера помимо процессорного ядра и блоков памяти входят периферийные устройства, обеспечивающие различные дополнительные функции.
- К внутренней периферии относятся аналого-цифровые и цифро-аналоговые преобразователи, порты, таймеры, контроллеры интерфейсов и другие устройства. Кроме внутренних периферийных устройств, дополнительные функции могут обеспечиваться подключением к микроконтроллеру внешних периферийных устройств, выполненных в виде самостоятельных микросхем.
- Внешняя периферия представлена различными запоминающими устройствами, внешними аналого-цифровыми и цифро-аналоговыми преобразователями, средствами сопряжения с каналом связи, устройствами отладки и рядом других.
- Одними из наиболее важных внутренних периферийных устройств микроконтроллера являются порты ввода-вывода (I/O port), представляющие собой средства информационного обмена с внешними устройствами.

Периферийные устройства (2)

- **В зависимости от способа обмена данными** порты ввода-вывода бывают
  - последовательными и
  - параллельными.
- Последовательный порт имеет одноразрядный формат и передаёт (принимает) информацию по принципу «один бит за другим».
- Параллельный порт имеет формат в несколько разрядов (обычно 8, 16 или 32), которые передаются или принимаются одновременно.
- **По виду синхронизации** (под синхронизацией в данном случае понимается временно́е согласование работы устройств передачи и приёма информации) порты делятся на синхронные и асинхронные.
- **Синхронными** называют порты, передача и приём информации с помощью которых осуществляется с жёсткой временно́й синхронизацией.
- **Асинхронными** называют порты, которые передают (принимают) данные с различной скоростью. С точки зрения возможностей использования различают специализированные и универсальные порты.

Периферийные устройства (3)

- Специализированные порты предназначены для реализации определённых интерфейсов обмена данными (**RS232/422/485**, **USB**, **SCI**, **SPI**, **I^2C**, **CAN** и др.).
- Интерфейсы типа **RS-232/422/485** обеспечиваются универсальными асинхронными приёмопередатчиками – УАПП (UART – Universal Asynchronous Receiver-Transmitter).
- **Синхронные** интерфейсы реализуются универсальными синхронно-асинхронными приёмопередатчиками – УСАПП (USART – Universal Synchronous-Asynchronous Receiver-Transmitter).
- **Универсальные** порты позволяют программно управлять основными параметрами процесса обмена информацией (временными характеристиками, форматом данных и др.).
- Порты ввода-вывода могут использоваться для обмена информацией в различных режимах: программно-управляемого обмена, обмена по прерываниям и обмена в режиме прямого доступа к памяти (ПДП).
- При **программно-управляемом обмене** (program-driven I/O) все операции ввода-вывода выполняются в соответствии с заложенной программой с проверкой готовности внешнего устройства к обмену.

Периферийные устройства (4)

- **Обмен по прерываниям** (Interrupt-driven I/O) осуществляется с использованием механизма прерываний.
- **Обмен в режиме ПДП** (Direct Memory Access – DMA) осуществляется с помощью аппаратных средств независимо от процессора, который в данном случае только инициирует процесс ввода-вывода и управляет соответствующими ресурсами.
- Ввод и вывод данных с помощью портов, а также управление портами осуществляется посредством чтения и записи специальных регистров или ячеек памяти. Доступ к другим периферийным устройствам производится аналогично. Периферийные устройства могут иметь собственное адресное пространство или для них выделяется часть адресов пространства памяти. В последнем случае говорят о **вводе-выводе, отображённом на память** (memory-mapped I/O).

Периферийные устройства (5)

- **Порты ввода-вывода AVR-микроконтроллеров.** В **AVR**-микроконтроллерах в зависимости от типа имеется от двух до шести универсальных двунаправленных портов ввода-вывода (порты **A**…**F**), содержащих семь или восемь сигнальных линий, которые соединены с соответствующими выводами БИС микроконтроллера. Например, в микроконтроллере **ATmega8535** имеется четыре 8-разрядных порта **A**, **B**, **C** и **D**. Сигнальные линии порта **A** обозначаются **PA0**…**PA7**, порта **B** – **PB0**…**PB7** и т. д.
- С каждым портом связаны три регистра ввода-вывода: регистр **DDR*x*** направления передачи данных, регистр **PORT*x*** данных порта и регистр **PIN*x*** выводов порта, где ***x*** – имя порта (**A**…**F**). Содержимое разрядов регистра **DDR*x*** определяет направление передачи данных по каждой сигнальной линии порта (0 – порт используется для ввода данных, 1 – для вывода данных). Регистр **PORT*x*** позволяет задавать состояние сигнальных линий порта при выводе информации; регистр **PIN*x*** – считывать состояние сигнальных линий порта при вводе информации.

Периферийные устройства (6)

- С помощью указанных регистров реализуется требуемый протокол обмена данными (протокол – совокупность правил передачи кодированной информации).
- Основными процедурами при реализации некоторого протокола являются операции инициализации порта, вывода данных в порт, ввода данных из порта, а также формирование временных задержек.