Оглавление

[1-й вопрос билета 2](#_Toc124302143)

[**1. Микропроцессор и микропроцессорная система.** 2](#_Toc124302144)

[**2. Структура МикроЭВМ.** 2](#_Toc124302145)

[**3. Однокристальная, одноплатная, многоплатная ЭВМ. Структура микрокомпьютера.** 3](#_Toc124302146)

[**4. Классификация микропроцессорных систем.** 3](#_Toc124302147)

[**5. Универсальные и специализированные микропроцессоры.** 4](#_Toc124302148)

[**6. Магистральная структура микропроцессорной системы с разделением по времени, с временным разделением сигналов, через общую память.** 5](#_Toc124302149)

[**7. Вычислительные системы, управляемые потоками команд.** 5](#_Toc124302150)

[**8. Вычислительные системы, управляемые потоками данных.** 5](#_Toc124302151)

[**9. Вычислительные системы, управляемые потоками запросов.** 6](#_Toc124302152)

[**10. Функционирование микропроцессорной системы.** 6](#_Toc124302153)

[**11. Задачи систем управления микропроцессорной системы.** 7](#_Toc124302154)

[**12. Интерфейсы микропроцессорной системы.** 8](#_Toc124302155)

[**13. Типы программного обеспечения микропроцессорной системы.** 9](#_Toc124302156)

[**14. Классификация программного обеспечения микропроцессорной системы.** 10](#_Toc124302157)

[**15. Фон-неймановская (принстонская) и гарвардская архитектуры.** 10](#_Toc124302158)

[**16. Организация пространств памяти и ввода-вывода.** 11](#_Toc124302159)

[**17. Микропроцессоры с RISC и CISC – архитектурами.** 12](#_Toc124302160)

[**18. Шина данных. Шина адреса. Шина управления.** 12](#_Toc124302161)

[2-й вопрос билета 14](#_Toc124302162)

[**19. Циклы чтения из памяти (из порта ввода). Цикл записи в память (в порт вывода).** 14](#_Toc124302163)

[**20. Двухшинная магистраль с совмещенными шинами передачи адреса и данных.** 14](#_Toc124302164)

[**21. Структура операционного блока 8-ми разрядных микропроцессоров.** 15](#_Toc124302165)

[**22. Структура операционного блока 16 разрядных микропроцессоров.** 15](#_Toc124302166)

[**23. Организация подсистемы прерываний в микропроцессорной системе.** 16](#_Toc124302167)

[**24. Понятие прерывания процессора. Контекстное переключение.** 17](#_Toc124302168)

[**25. Организация радиальной системы прерываний.** 17](#_Toc124302169)

[**26. Организация векторной системы прерываний. Вектор прерывания.** 18](#_Toc124302170)

[**27. Обработка обычных прерываний и прерываний при возникновении непредусмотренной (исключительной) ситуации.** 18](#_Toc124302171)

[**28. Функции памяти. Архитектура памяти.** 19](#_Toc124302172)

[**29. Многоуровневая организация памяти. Основные характеристики запоминающих устройств.** 20](#_Toc124302173)

[**30. Средства управления памятью в микропроцессорах. Линейная, сегментная и страничная адресация.** 20](#_Toc124302174)

[**31. Методы и способы снижения энергопотребления микропроцессорной системы.** 21](#_Toc124302175)

[**32. Выключение ядра микропроцессора и периферии (способы снижения энергопотребления).** 22](#_Toc124302176)

[**33. Структура микропроцессорной системы с аналого-цифровым и цифро-аналоговым преобразователем.** 23](#_Toc124302177)

[**34. Назначение и режимы работы таймеров в микропроцессорной системе.** 23](#_Toc124302178)

[**35. Универсальный синхронный/асинхронный приемопередатчик USART (UART, SPI, I2C) в микропроцессорной системе.** 24](#_Toc124302179)

[**36. Программные, программно-аппаратные способы отладки. Методы контроля правильности функционирования.** 25](#_Toc124302180)

# **1-й вопрос билета**

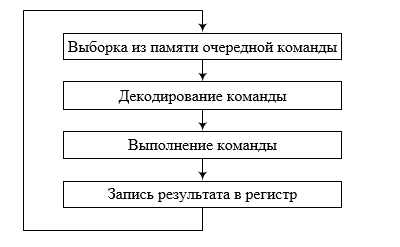
## **1. Микропроцессор и микропроцессорная система.**

Микропроцессор (МП) — программно-управляемое устройство, как правило, выполненное на одной большой интегральной схеме (БИС), осуществляющее цифровую обработку информации и управление этим процессом.

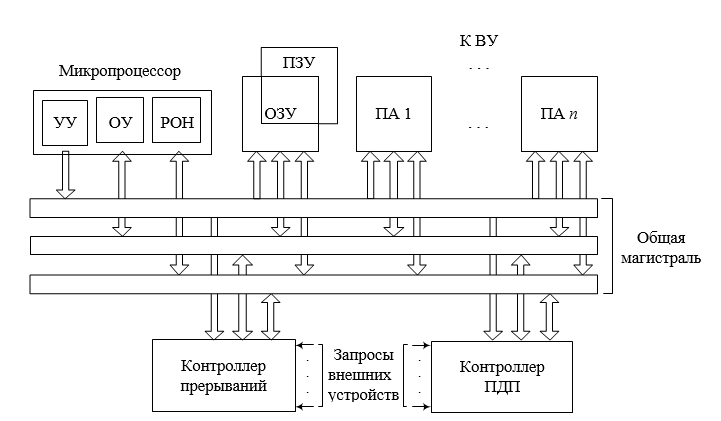
Долгое время процессоры создавались на основе отдельных микросхем малой и средней интеграции, содержащих от нескольких единиц до нескольких сотен транзисторов. Совокупность таких микросхем, совместимых по электрическим, информационным и конструктивным параметрам, называется микропроцессорным комплектом (МПК).

Микропроцессорное устройство (МПУ) — функционально и конструктивно законченное изделие, состоящее из нескольких микросхем, в состав которого входит микропроцессор. МПУ предназначено для выполнения определенного набора функций: получения, обработки, передачи, преобразования информации и управления.

Микроконтроллер (МК) — устройство, конструктивно выполненное на базе одной БИС и содержащее основные функциональные части МПК.



## **2. Структура МикроЭВМ.**



Все отдельные устройства (модули), входящие в состав такой системы, обмениваются данными по общей системной магистрали.

Основным модулем такой системы является микропроцессор, содержащий устройство управления, одно или несколько операционных устройств и блок регистров общего назначения (РОН). Модуль ОЗУ необходим для хранения выполняемой программы (или ее части) и обрабатываемых данных. ПЗУ предназначено для хранения выполняемой программы и констант. В процессорах общего назначения в ПЗУ хранится загрузчик — небольшая программа, которая загружает из внешнего запоминающего устройства в ОЗУ основную программу. В процессорах цифровой обработки сигналов в ПЗУ могут храниться таблицы часто используемых арифметических функций, например синусов.

Внешние устройства подключаются к общей шине посредством периферийных адаптеров (ПА), реализующих протоколы параллельного или последовательного обмена. В однокристальных микропроцессорных системах внешними устройствами могут быть различные таймеры, встроенные блоки АЦП и ЦАП, контроллеры внешней памяти, интерфейсы USB и др. Многие внешние устройства могут генерировать сигналы прерываний и запросы прямого доступа к памяти (ПДП). Для реализации всех возможностей таких устройств к общей шине подключаются также контроллер прерываний и контроллер ПДП.

## **3. Однокристальная, одноплатная, многоплатная ЭВМ. Структура микрокомпьютера.**

Однокристальные микроконтроллеры — функционально законченный МПК (микропроцессорный комплект), реализованный в виде одной СБИС (сверх-БИС). ОМК включает процессор, ОЗУ, ПЗУ, порты ввода-вывода для подключения внешних устройств, модули ввода аналогового сигнала АЦП, таймеры, контроллеры прерывания, контроллеры различных интерфейсов и т. д.

Одноплатный компьютер (SBC, англ. single-board computer) — самодостаточный компьютер, собранный на одной печатной плате, на которой установлены микропроцессор, оперативная память, системы ввода-вывода и другие модули, необходимые для функционирования компьютера. Одноплатные компьютеры изготавливаются в качестве демонстрационных систем, систем для разработчиков или образования, или для использования в роли промышленных или встраиваемых компьютеров.

В отличие от традиционных персональных компьютеров форм-фактора «desktop» (стандарты AT, ATX, и т. п.), одноплатные компьютеры часто не требуют установки каких-либо дополнительных периферийных плат. Некоторые одноплатные системы изготовлены в виде компактной платы с процессором и памятью, подключаемые к backplane для расширения возможностей, например, для увеличения количества доступных разъемов.

Чаще всего эти решения должны быть защищены или очень компактны, поэтому все компоненты должны располагаться на одной плате.

Такая экономия, с одной стороны, делает всё устройство более компактным и гораздо более дешевым за счет использования системы на кристалле, с другой стороны, расширение возможностей — смена процессора или памяти — затруднено, так как чаще всего эти компоненты напаяны на плату.

Общая структура микроЭВМ: ЦП – микропроцессор, ПЗУ – постоянная память для хранения программ, ОЗУ – оперативная память, УВВ – устройства ввода-вывода, Г – генератор тактовых импульсов. Отдельные модули соединены с помощью шины данных, шины адреса и шины управления. Количество модулей можно увеличивать, т.е. структура открыта и допускает расширение.

## **4. Классификация микропроцессорных систем.**

МПК может реализовываться на следующей элементной базе:

- однокристальных микропроцессорах (ОМП);

- секционных (многокристальных) МП;

- однокристальных микроконтроллерах (ОМК);

- сложных матричных программируемых логических схемах (ПЛИС, PLD, CPLD и др.).

Различают:

1) Периферийные (интерфейсные) ОМК предназначен для реализации простейших МП систем управления. Имеют малую производительность и малые габаритные размеры. В частности может использоваться периферийными устройствами ЭВМ (клавиатура, мышь и т.п.)

2) Универсальные 8–разрядные ОМК предназначены для реализации МП систем малой и средней производительности. Имеют простую систему команд и большую номенклатуру встроенных устройств.

3) Универсальный 16–разрядный ОМК. Предназначен для реализации систем реального времени средней производительности. Структура и система команд нацелены на скорейшую реакцию на внешние события. Наибольшее использование имеют в системах управления электродвигателями (мехатронные системы).

4) Специализированные 32–разрядные ОМКреализуют высокопроизводительную ARM архитектуру и предназначены для систем телефонии, передачи информации, телевидения и других, требующие высокоскоростной обработки информации.

5) Цифровые сигнальные процессоры (DSP – Digital Signal Processor) предназначен для сложной математической обработки измеряемых сигналов в режиме реального времени. Широко используются в телефонии и связи. Основные отличия DSP: повышенная разрядность обрабатываемых слов (16,32,64 бита) и высокая скорость в формате с плавающей точкой (16flops).

По области применения определилось три направления развития микропроцессоров:

* микроконтроллеры
* универсальные микропроцессоры
* сигнальные микропроцессоры

По внутренней структуре существует два основных принципа построения микропроцессоров:

* Гарвардская архитектура
* Архитектура Фон-Неймана

По системе команд микропроцессоры отличаются огромным разнообразием, зависящим от фирмы-производителя. Тем не менее можно определить две крайние политики построения микропроцессоров:

* Аккумуляторные микропроцессоры
* Микропроцессоры с регистрами общего назначения

По назначению различают универсальные и специализированные микропроцессоры.

По организации памяти МПС разделяют на:

* Локальная память
* Общая память
* Локальная основная + общая вспомогательная

По типу внутренней связи:

* Магистральная
* Матричная
* Иерархическая

## **5. Универсальные и специализированные микропроцессоры.**

Универсальные микропроцессоры могут быть применены для решения широкого круга разнообразных задач. При этом их эффективная производительность слабо зависит от проблемной специфики решаемых задач. Специализация МП, т.е. его проблемная ориентация на ускоренное выполнение определенных функций позволяет резко увеличить эффективную производительность при решении только определенных задач.

Среди специализированных микропроцессоров можно выделить различные микроконтроллеры, ориентированные на выполнение сложных последовательностей логических операций, математические МП, предназначенные для повышения производительности при выполнении арифметических операций за счет, например, матричных методов их выполнения, МП для обработки данных в различных областях применений и т. д. С помощью специализированных МП можно эффективно решать новые сложные задачи параллельной обработки данных.

## **6. Магистральная структура микропроцессорной системы с разделением по времени, с временным разделением сигналов, через общую память.**

Магистральная структура или, как ее еще называют, структура с общей шиной предполагает подключение всех составляющих микропроцессорной системы к микропроцессору по одной группе шин.

Магистральная организация обеспечивает все связи между устройствами ЭВМ, предусмотренные общей функциональной схемой, но не параллельно, а последовательно, с разделением во времени.

В каждый момент времени по общей шине взаимодействуют лишь два устройства вычислительной машины. Третье устройство отключено от шины с помощью выходов с открытым коллектором или тремя состояниями.

## **7. Вычислительные системы, управляемые потоками команд.**

Под потоком команд понимается последовательный ряд команд, выполняемых системой.

При управлении от потока команд, команды выполняются по ходу расположения в программе.

Сравнение управления от потока команд и управление от потока данных:

*От потока команд:*

* Последовательное выполнение
* Перезапись глобальных переменных
* Централизованное управление

*От потока данных:*

* Параллельность выполнения команд с готовыми операндами
* Последовательно выполняются зависимые команды
* Нет перезаписи глобальных переменных
* Децентрализация управления

## **8. Вычислительные системы, управляемые потоками данных.**

При управлении от потока данных команды выполняются, если все исходные данные готовы, команды могут выполняться параллельно (естественный параллелизм). В системах с управлением от потока данных, каждая команда, для которой имеются все необходимые операнды, немедленно выполняется. Однако для получения окончательного результата многие из этих вычислений оказываются ненужными.

Сравнение управления от потока команд и управление от потока данных:

*От потока команд:*

* Последовательное выполнение
* Перезапись глобальных переменных
* Централизованное управление

*От потока данных:*

* Параллельность выполнения команд с готовыми операндами
* Последовательно выполняются зависимые команды
* Нет перезаписи глобальных переменных
* Децентрализация управления

## **9. Вычислительные системы, управляемые потоками запросов.**

При управлении от потоков запросов, вычисления инициируются не по готовности данных, а на основе запроса на данные. Такая организация вычислительного процесса называется управлением вычислениями по запросу (demand-driven control). В ее основе лежит представление вычислительного процесса в виде графа, как и в потоковой модели (data-driven control).

В потоковой модели верхние узлы графа запускаются раньше, чем нижние (нисходящая обработка). Механизм управления по запросу состоит в обработке вершин потокового графа снизу вверх путем разрешения запуска узла, лишь когда требуется его результат.

При управлении от потока запросов, команда выполняется в тот момент, когда её результат оказывается нужен кому-то другому. В таком виде управления команды могут выполняться параллельно.

Достоинства такой реализации:

* Задача анализируется по мере необходимости
* Не анализируются лишние фрагменты
* Задача реализуется с верхнего уровне

Недостатки такой реализации:

* Реализация происходит с запаздыванием, так как на тот момент, когда данные уже необходимы, только начинается процесс их вычисления.

## **10. Функционирование микропроцессорной системы.**

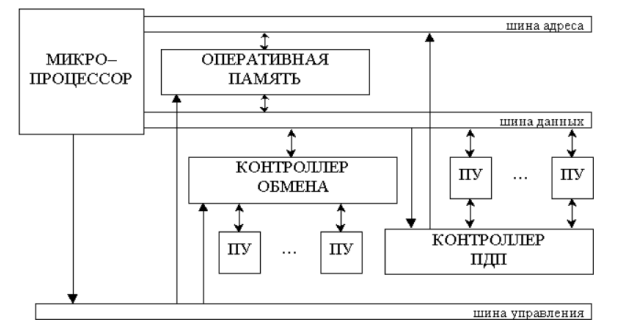
Функционирование МПС сводится к следующей последовательности действий:

1. Получение данных от различных периферийных устройств (с клавиатуры терминала, от дисплеев, из каналов связи, от различного типа внешних запоминающих устройств);
2. Обработка данных;
3. Выдача результатов обработки на периферийные устройства (ПУ).

При этом данные от ПУ, подлежащие обработке, могут поступать и в процессе их обработки. Для выполнения этих действий в МПС кроме микропроцессора предусматриваются следующие устройства:

- оперативная память (ОП), предназначенная для хранения и выдачи по запросам команд программ, определяющих работу микропроцессора, различных данных (исходных данных, промежуточных и конечных результатов обработки данных в микропроцессоре);

- контроллеры — устройства, обеспечивающие обмен данными различных ПУ с микропроцессором и ОП.



## **11. Задачи систем управления микропроцессорной системы.**

Можно сформулировать конкретные функции, которые выполняет МПСУ при управлении технологическим оборудованием:

1. Хранение программного обеспечения в постоянном запоминающем устройстве (ПЗУ).

2. Ввод и хранение управляющих программ (УП). В автономных МПСУ ввод осуществляется с перфоленты или по каналу от ЭВМ верхнего уровня. Может использоваться ручной ввод УП с пульта управления (ПУ) МПСУ. Память хранения УП должна быть энергонезависимой.

3. Реализация фиксированных и программно-технологических циклов, под которыми понимают повторяющиеся (стандартные) участки УП, что позволяет сократить трудоемкость программирования и объем УП.

4. Интерпретация кадра, под которой понимают проведение ряда предварительных процедур при отработке очередного кадра УП. Так, для обеспечения непрерывности контурного управления процедуры интерпретации (i+1)-кадра должны быть реализованы во время управления по i-кадру.

5. Интерполяция, под которой понимают получение с требуемой точностью координат промежуточных точек траектории движения по координатам крайних точек и заданной функции интерполяции.

6. Управление приводами подач рабочих органов станка. Данная задача сводится к организации следящих систем для каждой управляющей координаты.

7. Управление приводом главного движения, предусматривающее включение и отключение этого привода, стабилизацию скорости его вращения, а в некоторых случаях и управление углом поворота шпинделя станка как дополнительной управляемой координатой (доворот шпинделя для установки инструмента).

8. Логическое управление технологическими узлами и механизмами дискретного действия, входные сигналы которых производят операции типа “включено”,” выключено”.

9. Смена инструментов. Эта задача включает поиск гнезда магазина с требуемым инструментом и его замену.

10. Коррекция размеров инструмента в УП.

11. Коррекция погрешностей механических элементов привода подач и погрешностей измерительных преобразователей.

12.Адаптивное управление в процессе обработки заготовок.

13.Автоматический контроль обрабатываемых заготовок на станке.

14.Обмен информацией с ЭВМ верхнего уровня и системой управления промышленным роботом (СУПР).

15.Управление автооператором, транспортными устройствами и средствами автоматизации.

16.Связь с оператором посредством пульта управления, дисплея, перфоратора ленточного (ПЛ), фотосчитывающего устройства (ФСУ), внешней памяти на магнитных лентах (НМЛ) и магнитных дисках (НМД).

17.Техническая диагностика МПСУ.

## **12. Интерфейсы микропроцессорной системы.**

Одним из центральных моментов в проектировании микропроцессорной системы является выбор интерфейсов.

Интерфейсы представляют собой совокупность унифицированных аппаратных, программных и конструктивных средств, необходимых для реализации алгоритмов взаимодействия различных функциональных устройств. Стандартизации в интерфейсе обычно подлежат: форматы передаваемой информации, команды и состояния, состав и типы линий связи, алгоритм функционирования, передающие и приемные электронные схемы, параметры сигналов и требования к ним, конструктивные решения.

Скорость передачи данных по интерфейсу измеряется в Бодах. Бод определяет, сколько бит передается в единицу времени, т.е. в секунду.

К основным характеристикам интерфейса относят:

* Функциональное назначение;
* Принцип обмена информацией;
* Способ обмена;
* Режим обмена;
* Количество линий;
* Число линий для передачи данных (разрядность);
* Количество адресов;
* Количество команд;
* Быстродействие;
* Длину линий связи;
* Число подключаемых устройств (нагрузочная способность);
* Тип линии связи.

По функциональному назначению интерфейсы разделяют на внутренние (межплатные, межелементные и системные) и внешние (для периферийных устройств, для локальных сетей, для распределенных систем).

Внутренние интерфейсы определяют комплекс принятых соглашений по организации связей между микросхемами, модулями и блоками, составляющими микропроцессоную систему.

Внешние интерфейсы организуют связь между несколькими микропроцессорными системами.

По принципу обмена информацией выделяют интерфейсы с параллельной, последовательной и параллельно-последовательной передачей информации.

В параллельных интерфейсах в единицу времени данные пересылаются по нескольким сигнальным линиям одновременно. В последовательных – по одной линии бит за битом.

В последовательных интерфейсах кабели обычно представляют витую пару и их длину можно реализовывать значительно большую, чем в параллельных интерфейсах.

При проектировании специализированных микропроцессорных систем на базе микроконтроллеров разработчики часто сталкиваются с проблемой нехватки портов для подключения большого количества периферии. Поэтому выводы микросхем микроконтроллеров часто приходится экономить. По этой причине в микроконтроллерных микропроцессорных системах в настоящее время часто используются последовательные интерфейсы, особенно внутренние.

По режиму обмена информацией различают интерфейсы с симплексным, полудуплексным, дуплексным и мультиплексным режимами обмена.

Симплексный режим обмена характерен тем, что только один из абонентов может в любой момент времени инициировать передачу информации.

Полудуплексный режим обмена обеспечивает инициализацию связи любым абонентом, если линия связи интерфейса свободна.

Дуплексный режим обеспечивает связь любому абоненту в любой момент времени.

Мультиплексный режим позволяет связываться только между парами абонентов и в единственном направлении от одного к другому. Мультиплексный режим характерен для магистральной организации, когда поочередно разные устройства включаются в общую шину.

По способу передачи информации во времени различают интерфейсы с синхронной передачей данных (с постоянной временной привязкой в цикле сбора информации) и с асинхронной (без постоянной временной привязки к определенному временному интервалу цикла сбора). В первом случае передача синхронизируется специальными синхроимпульсами CLK в виде последовательности прямоугольных импульсов. Во втором – управляющими сигналами «ГОТОВНОСТЬ» к обмену, «НАЧАЛО», «КОНЕЦ» и «КОНТРОЛЬ» обмена.

## **13. Типы программного обеспечения микропроцессорной системы.**

Следует отметить, что необходимой составной частью архитектуры любой микропроцессорной системы является программное обеспечение (ПО). Различают два вида программного обеспечения в микропроцессорных системах:

* Внутреннее или резидентное ПО, которое образуют служебные программы, постоянно хранящиеся в памяти прикладной микропроцессорной системы;
* Кросс-программные средства, используемые на этапе разработки или модернизации самой системы или ее резидентного ПО, если это происходит с применением универсальных компьютеров.

Резидентное программное обеспечение — это совокупность программ, разработанных для конкретной микропроцессорной системы управления на языке используемого в ней процессора. К нему можно отнести:

* Программы самодиагностики для контроля правильности функционирования данной микропроцессорной системы управления;
* Прикладные программы, непосредственно реализующие функции, возлагаемые на систему пользователями (управление объектами, проектирование изделий и многое другое);
* Операционная система, обеспечивающая как организацию совместного функционирования всех элементов ее структуры (монитор, включающий в себя драйверы внешних устройств – программы, реализующие информационно-логическое сопряжение вычислительного устройства с ним), так и распределение ресурсов микропроцессорных систем управления между прикладными программами (процессами) пользователя (супервизорная часть);
* Среды разработки программного обеспечения, представляющие собой комплекс программ, необходимых для создания программного обеспечения микропроцессорных систем управления: текстовые редакторы; трансляторы, осуществляющие перевод исходного текста программы на формальном языке в исполняемую программу на языке процессора; отладчики, позволяющие различными методами находить ошибки в разрабатываемом программном обеспечении. При этом средства отладки должны управлять исполнением программы (останавливать, изменять порядок, запускать), собирать информацию о ходе ее выполнения, обеспечивать диалог между программистом и электронно-вычислительной машиной на уровне языка программирования.

Другую большую часть программного обеспечения микропроцессорного комплекта составляет кросс-обеспечение — это совокупность программ для разработки и отладки резидентного программного обеспечения микропроцессорных систем управления на микропроцессорную систему управления с другим типом микропроцесса. К ним относятся:

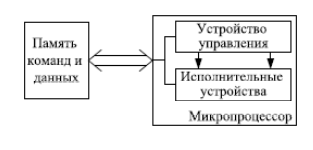
* Текстовый редактор, обеспечивающий ввод в кросс-систему и корректировку программы пользователя на исходном языке;
* Интерпретатор, осуществляющие перевод программы с исходного языка в команды микропроцессорных систем управления с одновременным их исполнением;
* Дизассемблер - транслятор, осуществляющий обратное преобразование исполняемой программы с языка команд микропроцессорных систем управления на формальный язык – Ассемблер;
* Программный эмулятор, позволяющие промоделировать работу микропроцессорной системы управления и прочих аппаратных средств разрабатываемой микропроцессорной системой управления на другой электронно-вычислительной машине;
* И многое другое

## **14. Классификация программного обеспечения микропроцессорной системы.**

**СМ ПРЕДЫДУЩИЙ ВОПРОС**

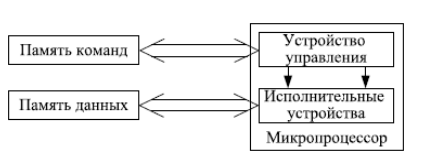
## **15. Фон-неймановская (принстонская) и гарвардская архитектуры.**

Архитектуру компьютеров, отвечающих этим принципам, называют принстонской, или фон Неймана. Однако обычно, когда говорят о принстонской архитектуре, речь идет о способе организации памяти, потому что большинство современных компьютеров отвечают принципам фон Неймана.

Согласно принципу адресуемости, память микросистемы представляет собой упорядоченный набор K-разрядных ячеек с произвольным доступом. Такая память называется линейной. Совокупность адресов памяти от 0 до 2n− 1, где n— количество двоичных разрядов адреса, называется адресным пространством. Обычно адресное пространство разделяется на два подмножества: пространство ввода/вывода и адресное пространство памяти. В этом случае ввод/вывод называют изолированным, в противном случае —совместным.

Характерной особенностью принстонской архитектуры является то, что в ее состав входит отделенная от исполнительных устройств общая память команд и данных. В системе с такой архитектурой исполнительные устройства и основная память соединены одним коммутационным трактом, передачи команд и данных разделены во времени. Достоинствами принстонской архитектуры стали гибкость вычислительной системы, простота реализации и отладки.

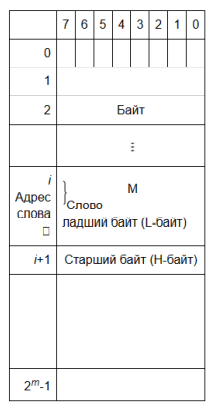
Простота отладки определяется наличием только одной системной шины. Однако в связи с увеличением производительности исполнительных устройств и ростом объема передаваемых данных единая системная шина стала ограничивать производительность таких систем.

В гарвардской архитектуре память команд и память данных физически разделены. Шины, соединяющие их с процессором, могут иметь как разную разрядность, так и разный объем памяти под команды и данные. Такое архитектурное решение дает выигрыш в быстродействии по сравнению с принстонской архитектурой, поскольку за один машинный цикл процессор может получить команды и данные.

Основным недостатком гарвардской архитектуры является сложность технической реализации двух независимых шин, требующая дополнительных аппаратных затрат. Другим недостатком является то, что объемы памяти команд и памяти данных фиксированы, нет возможности динамически перераспределять информацию между ними.

## **16. Организация пространств памяти и ввода-вывода.**

Память представляет собой линейно упорядоченный набор n-разрядных ячеек с произвольным доступом (одномерный массив) линейная память. Все ячейки пронумерованы, таким образом каждой ячейке набора соответствует число, называемое ее адресом. Все адреса занимают целочисленный диапазон от 0 до 2 m -1 (m разрядность адреса), который образует адресное пространство памяти. В большинстве случаев процессор может адресоваться к памяти с точностью до одного байта, т.е. наименьшей адресуемой единицей является байт и память имеет байтовую организацию.

Организация пространства памяти показана на рисунке. При этом память изображается таким образом, чтобы ячейки со старшими адресами располагались ниже, чем с младшими. Нумерация отдельных разрядов в ячейке производится справа налево начиная с нуля, при этом разряд с нулевым номером является младшим.

Программные объекты (команды и операнды) могут иметь длину, превышающую один байт, например, два байта – 16-разрядное слово или просто слово, четыре байта – 32-разрядное слово или двойное слово, восемь байтов – 64-разрядное слово или учетверенное слово. Такие объекты располагаются в смежных ячейках пространства памяти, причем обычно младший байт размещается в ячейке с меньшим адресом.

Пространство ввода/вывода представляет набор адресуемых буферных схем и регистров, которые называются портами и через которые осуществляется связь с внешними и внутренними аппаратными средствами микропроцессорной системы. Пространство ввода-вывода имеет такую же организацию, как и пространство памяти.

В микропроцессорной системе может использоваться два варианта организации доступа к пространству ввода/вывода:

* *Изолированный ввод/вывод.* Порты ввода/вывода размещены в специальном пространстве ввода/вывода (Input/Output Space – IOS), изолированном от других пространств данных. В этом случае МП имеет специальный набор команд ввода/вывода.
* *Совмещенный ввод/вывод* или *ввод/вывод с отображением на память*. В этом случае изолированное пространство ввода/вывода отсутствует, а в пространстве памяти данных DS выделяются области, в которых размещаются порты. Организация доступа к портам в такой микропроцессорной системе ничем не отличается от процесса обращения к данным в памяти.

## **17. Микропроцессоры с RISC и CISC – архитектурами.**

*Основные черты CISC-концепции:*

- Ранее других появились процессоры CISC. Термин CISC означает сложную систему команд и является аббревиатурой английского определения Complex Instruction Set Computer. Благодаря этому процессоры выполняют самые разнообразные задачи обработки данных.

- При разработке набора команд CISC заботились об удобстве программиста/компилятора, а не об эффективности исполнения команд процессором. В систему команд вводили много сложных команд (производящих по несколько простых действий). Часто эти команды представляли собой программы, написанные на микрокоде и записанные в ПЗУ процессора. Команды CISC имеют разную длину и время выполнения. Зато машинный код CISC-процессоров — язык довольно высокого уровня. В наборе команд CISC часто присутствуют, например, команды организации циклов, команды вызова подпрограммы и возврата из подпрограммы, сложная адресация, позволяющая реализовать одной командой доступ к сложным структурам данных. Основной недостаток CISC — большая сложность реализации процессора при малой производительности.

*Основные черты RISC-концепции:*

- Со временем стало необходимо повысить скорость работы процессоров. Одним из путей к этому стал процессор RISC, который характеризуется сокращенным набором быстро выполняемых команд и происходит от английского Reduced Instruction Set Computer.

-Одинаковая длина команд;

-Одинаковый формат команд — код команды регистр-приемник два регистра-источника;

-Операндами команд могут быть только регистры;

-Команды выполняют только простые действия;

-Большое количество регистров общего назначения (могут быть использованы любой командой);

-Конвейер(ы); ∙выполнение команды не дольше, чем за один такт; ∙простая адресация.

*Сравнение архитектур:*

Основная идея RISC-архитектуры — это тщательный подбор таких комбинаций кодов операций, которые можно было бы выполнить за один такт тактового генератора. Основной выигрыш от такого подхода — резкое упрощение аппаратной реализации ЦП и возможность значительно повысить его производительность.

Однако обычно выигрыш от повышения быстродействия в рамках RISC-архитектуры перекрывает потери от менее эффективной системы команд, что приводит к более высокой эффективности RISC-систем в целом по сравнению с CISC. Так, в процессоре CISC для выполнения одной команды необходимо, в большинстве случаев, 10 и более тактов.

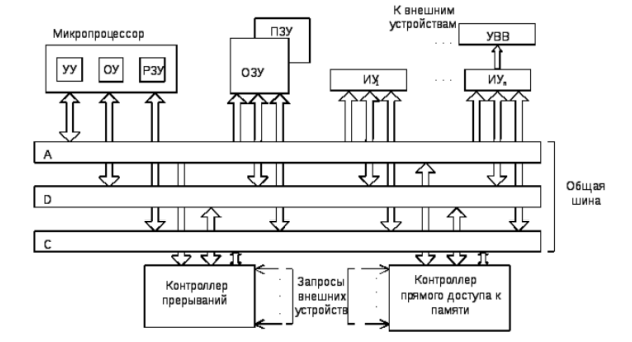
## **18. Шина данных. Шина адреса. Шина управления.**

Системная шина содержит несколько десятков (в сложных системах более 100) проводников, которые в соответствии с их функциональным назначением подразделяются на отдельные шины – адреса А, данных D и управления С.

Шина А служит для передачи адреса, который формируется микропроцессором и позволяет выбрать необходимую ячейку памяти ОЗУ (ПЗУ) или требуемое ИУ при обращении к внешнему устройству.

Шина D служит для выборки команд, поступающих из ОЗУ или ПЗУ в УУ микропроцессора, и для пересылки обрабатываемых данных (операндов) между микропроцессором и ОЗУ или ИУ (внешним устройством).

По шине С передаются разнообразные управляющие сигналы, определяющие режимы работы памяти (запись или считывание), интерфейсных устройств (ввод или вывод информации) и МП (запуск, запросы внешних устройств на обслуживание, информация о текущим режиме работы и другие сигналы).



# **2-й вопрос билета**

## **19. Циклы чтения из памяти (из порта ввода). Цикл записи в память (в порт вывода).**

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

Циклы обмена информацией делятся на два основных типа:

* Цикл записи (вывода), в котором процессор записывает (выводит) информацию;
* Цикл чтения (ввода), в котором процессор читает (вводит) информацию.

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

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

Чтение кодов команд из памяти системы также производится с помощью циклов чтения. Поэтому в случае одношинной архитектуры на системной магистрали чередуются циклы чтения команд и циклы пересылки (чтения и записи) данных, но протоколы обмена остаются неизменными независимо от того, что передается — данные или команды. В случае двухшинной архитектуры циклы чтения команд и записи или чтения данных разделяются по разным шинам и могут выполняться одновременно.

## **20. Двухшинная магистраль с совмещенными шинами передачи адреса и данных.**

В некоторых МП с целью сокращения ширины физической магистрали используют совмещение адресной шины с шиной данных. В течение первого такта цикла магистрали шина данных не используется, поэтому этот интервал можно использовать для передачи по шине данных адресных сигналов (адреса). Этап передачи адресной информации по совмещенной шине адреса/данных AD (Address/Data Bus) отделяется по времени от этапа передачи данных и стробируется специальным сигналом ALE (Address Latch Enable), который включается в состав шины управления. Данную магистраль называют двухшинной с совмещенными шинами передачи адреса и данных. Если разрядность данных меньше разрядности адреса, то по совмещенной шине передаются только младшие разряды адреса, а старшие разряды при этом передаются по адресной шине.

Входящий в состав шины управления сигнал ALE используется для разделения функций, выполняемых совмещенной шиной AD. По этому сигналу присутствующая на шине AD адресная информация должна быть принята (зафиксирована) во внешний (по отношению к МП) адресный регистр-фиксатор. Для этой цели обычно служит срез сигнала ALE (переход из высокого уровня в низкий). Обычно каждый модуль микропроцессорной системы с двухшинной магистралью (модуль памяти или интерфейс периферийного устройства) содержит локальный адресный регистр для запоминания адресной информации.

## **21. Структура операционного блока 8-ми разрядных микропроцессоров.**

Основные характеристики:

1. Длина слова - 8 бит (байт)

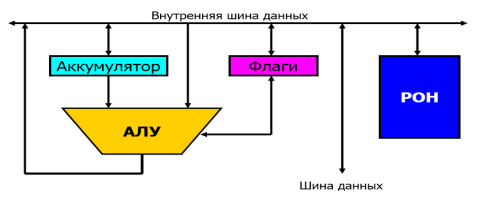
2. Адресуемое пространство памяти – 64 К байта

3. Операционные блок с аккумулятором

4. Равенство длинны слова и разрядность АЛУ, аккумулятора, регистров (данная характеристика условна, т.к. существуют исключения в виде разрядности аккумулятора, АЛУ и т.д., разрядность которых может быть больше 8).

5. Восьмиразрядные ЦП могут иметь возможность оперировать с 16-ти разрядными числами, если в них реализовано 16-ти разрядное АЛУ.

*Структура 8-ми разрядных ЦП:*



*Внутренняя шина данных – осуществляет передачу данных.*

*Шина данных – внешняя шина.*

*РОН – регистры общего назначения.*

*АЛУ – арифметико-логическое устройство, выполняет основные действия.*

Для выполнения действий в АЛУ нужно два операнда, но т.к. внутренняя шина данных только одна (по ней можно подать один из операндов), то для подачи второго операнда в АЛУ необходим аккумулятор, находящийся максимально близко к АЛУ. Операнд из РОН сначала попадает в аккумулятор, а уже затем в АЛУ.

К примеру, если выполняется сложение: один операнд передают в аккумулятор, а другой подается на вход АЛУ по внутренней шине данных. Второй операнд может поступать на внутреннюю шину данных либо из РОН, либо по внешней шине данных, если он находился во внешней памяти. Затем операнды синхронно загружаются в АЛУ. Результат выполнения операции может быть сохранен в РОН либо во внешнюю память. Это зависит от выполняемых действий.

## **22. Структура операционного блока 16 разрядных микропроцессоров.**

Основные характеристики:

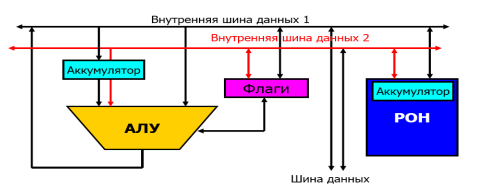
1. Длина слова – 16 бит (слово)

2. Адресуемое пространство памяти – 64 К байта и более

3. Операционные блок с аккумулятором или без него (шин стало больше, увеличилась степень интеграции)

4. Равенство длинны слова и разрядность АЛУ, аккумулятора, регистров. Данная характеристика условна, однако разрядность указанных элементов должна быть не меньше разрядности микропроцессора.

*Структура 16-и разрядных ЦП:*

**

Появление второй внутренней шины данных позволяет снабдить АЛУ сразу двумя операндами без занесения одного из них в аккумулятор. В данном случае аккумулятор имеет условное значение и расценивается как ячейка памяти.

При такой архитектуре ЦП можно выполнять действия как с операндами в РОН, так и с операндами во внешней памяти. Если операнды находятся в РОН, то результата по внутренней шине возвращается в РОН.

Производительность процессора может возрасти в два раза, если иметь дело только с РОН. Увеличение производительности происходит за счет наличия второй внутренней шины.

## **23. Организация подсистемы прерываний в микропроцессорной системе.**

Системой прерываний называется совокупность специальных аппаратных средств, осуществляющих прием запроса на прерывание и переход к подпрограмме обработки этого запроса, команд и программ обслуживания запросов прерывания. Кроме обслуживания ПУ (выполнения обмена) в микропроцессорной системе существуют и другие события, которые могут вызвать прерывание процессора. Типы прерываний зависят от конкретной микропроцессорной системы. Прерывания распадаются на два основных класса: внешние и внутренние.

*Внешние прерывания* вызываются асинхронными событиями, которые происходят вне прерываемой программы. Например, прерывания от таймера или подсистемы ввода/вывода.

*Внутренние прерывания* вызываются событиями, которые связаны с работой процессора и являются синхронными с его операциями. Например: арифметическое переполнение, попытка деления на 0, переполнение или исчезновение порядка при выполнении операций с плавающей запятой. Внутренние прерывания происходят при обращении к защищенным или несуществующим ячейкам памяти, а также к отсутствующему сегменту или странице. Прерывания вызывает попытка использовать незадействованный код операции, а также попытка использовать привилегированные команды в пользовательском режиме. Внутренние прерывания могут вызываться также сбоями системы, например, ошибкой четности, при выполнении специальных команд.

Прерывание программы не должно оказывать на прерванную программу никакого влияния кроме увеличения времени ее выполнения за счет приостановки на время выполнения подпрограммы обработки прерывания. Для этого после приема запроса на прерывание состояние МП необходимо сохранить. Для того чтобы прерванная программа могла быть продолжена после обслуживания очередного запроса на прерывание с того места, на котором она была приостановлена, состояние процессора должно быть восстановлено. Всякий раз, когда процессор воспринимает запрос на прерывание, он активизирует процедуру обслуживания, передавая ее стартовый адрес в программный счетчик. Чтобы не потерялось старое содержимое программного счетчика, которое является адресом возврата в прерванную программу, оно должно быть автоматически сохранено. Лучше всего для этой цели использовать системный стек, тогда возврат к прерванной программе будет заключаться в передаче управления по адресу, находящемуся на вершине стека. Обычно аппаратными средствами обработки прерывания автоматически сохраняется не только содержимое программного счетчика, но и регистра состояния процессора, а также его программно-доступных регистров.

Таким образом, хотя существует несколько различных способов обработки прерывания, следующая последовательность действий присуща большинству микропроцессорных систем:

- Фиксируются характеристики произошедшего прерывания (тип прерывания);

- Сохраняется состояние прерванной программы – состояние процессора;

- Анализируется тип прерывания и передается управление соответствующей подпрограмме обработки этого прерывания;

- Обрабатывается прерывание – выполняется соответствующая подпрограмма;

- Восстанавливается состояние процессора, что приводит к возобновлению выполнения прерванной программы.

Во всех системах прерываний предусмотрен механизм программно-управляемой блокировки запросов, который реализуется с помощью набора флажков, разрешающих или запрещающих восприятие запросов на прерывание процессором. Эти флажки образуют маску прерываний и либо упаковываются в отдельный регистр маски прерываний, либо входят в состав регистра флагов (признаков) процессора.

Существуют две системы прерываний: радиальная и векторная.

## **24. Понятие прерывания процессора. Контекстное переключение.**

Прерывание — это событие, при наступлении которого процессор должен приостановить выполнение текущего процесса, сохранить его состояние и начать обрабатывать другой процесс, благодаря процедуре, называемой обработчиком прерывания (interrupt handler). После завершения обработчика прерывания состояние прерванного процесса должно быть восстановлено, а в случае фатального прерывания (например, из-за отказа аппаратуры) процессор должен быть перезагружен или остановлен.

Когда процессор переключается на другой процесс, система должна сохранить состояние старого процесса и загрузить сохраненное состояние для нового процесса. Такое действие системы называется переключением контекста (context switch). Переключение контекста относится к накладным расходам (overhead), так как система не выполняет никаких полезных действий при переключении с одного процесса на другой.

При переключении контекста:

1. Фиксируются характеристики произошедшего прерывания (тип прерывания);
2. Сохраняется состояние прерванной программы – состояние процессора;
3. Анализируется тип прерывания и передается управление соответствующей подпрограмме обработки этого прерывания;
4. Обрабатывается прерывание – выполняется соответствующая подпрограмма;
5. Восстанавливается состояние процессора, что приводит к возобновлению выполнения прерванной программы.

## **25. Организация радиальной системы прерываний.**

Радиальные прерывания - прерывания, при которых номер прерывания (адрес вектора) определяется номером используемой линии запроса прерывания. При радиальном прерывании в магистрали имеется столько линий запроса прерывания, сколько всего может быть разных прерываний. То есть каждое устройство ввода / вывода, желающее использовать прерывание, подает сигнал запроса прерывания по своей отдельной линии. Процессор узнает о номере прерывания по номеру линии, по которой пришел сигнал запроса прерывания. Никаких циклов обмена по магистрали при этом не требуется.

В случае радиальных прерываний в систему обычно включается дополнительная микросхема контроллера прерываний, обрабатывающая сигналы запросов прерываний. Каждое радиальное прерывание требует введения дополнительной линии в шину управления системной магистрали. Но работать с радиальными прерываниями проще всего, так как все сводится только к выработке единственного сигнала IRQ, и никаких циклов обмена по магистрали не требуется.

Блок прерываний организует приоритетную систему прерываний в процессоре, принимает и обрабатывает внутренние и внешние запросы на прерывание. При одновременном появлении нескольких запросов все источники прерываний по взаимному приоритету образуют следующую иерархию:

1. Ошибка обращения к каналу;
2. Резервный или запрещенный код в регистре команд;
3. Т - разряд в слове состояния процессора;
4. Сигнал аварии сетевого питания ACLO;
5. Запросы радиального прерывания IRQ1, IRQ2, IRQ3;
6. Требование прерывания VIRQ.

Необходимо отметить, что блок прерывании реагирует на запросы ACLO, IRQ2, IRQ3 при их переходе от высокого уровня напряжения к низкому. При возникновении условий прерывания процессор микропрограммно осуществляет сохранение текущего значения счетчика команд и слова состояния и производит загрузку их нового значения из пары ячеек внешнего ОЗУ или ПЗУ. Прерывание процессора обычно допускается лишь в конце выполнения команды. Только прерывание «Ошибка обращения к каналу» может остановить выполнение команды на любой фазе ее выполнения.

## **26. Организация векторной системы прерываний. Вектор прерывания.**

Вектор прерывания — закреплённый за устройством номер, который идентифицирует соответствующий обработчик прерываний. Векторы прерываний объединяются в таблицу векторов прерываний, содержащую адреса обработчиков прерываний. Местоположение таблицы зависит от типа и режима работы процессора.

Аппаратный способ идентификации источников прерываний базируется на схемной реализации опроса Внешних устройств (ВУ). При обнаружении ВУ, запросившего прерывание, в процессор посылается его собственный код, который называют вектором прерывания. Название связано с тем, что двоичный n–разрядный код можно представить вектором в n–мерном пространстве. Вектор прерывания является начальным, или стартовым адресом подпрограммы. Системы прерываний с аппаратной идентификацией ВУ называют векторными системами прерываний. Простейшей векторной системой прерываний является система, к каждому входу запроса на прерывание которой подключено по одному ВУ. Такая система не требует идентификации ВУ.

## **27. Обработка обычных прерываний и прерываний при возникновении непредусмотренной (исключительной) ситуации.**

Механизм обработки прерываний независимо от архитектуры вычислительной системы включает следующие элементы механизма обработки прерываний:

1. Установление факта прерывания (прием сигнала на прерывание) и идентификация прерывания (в операционных системах иногда осуществляется повторно, на шаге 4).

2. Запоминание состояния прерванного процесса. Состояние процесса определяется прежде всего значением счетчика команд (адресом следующей команды, который, например, вi80x86 определяется регистрами CSиIP— указателем команды), содержимым регистров процессора и может включать также спецификацию режима (например, режим пользовательский или привилегированный) и другую информацию.

3. Управление аппаратно передается подпрограмме обработки прерывания. В простейшем случае в счетчик команд заносится начальный адрес подпрограммы обработки прерываний, а в соответствующие регистры — информация из слова состояния. В более развитых процессорах, например в том же i80286 и последующих 32-битовых микропроцессорах, начиная с i80386, осуществляется достаточно сложная процедура определения начального адреса соответствующей подпрограммы обработки прерывания и не менее сложная процедура инициализации рабочих регистров процессора.

4. Сохранение информации о прерванной программе, которую не удалось спасти на шаге 2 с помощью действий аппаратуры. В некоторых вычислительных системах предусматривается запоминание довольно большого объема информации о состоянии прерванного процесса.

5. Обработка прерывания. Эта работа может быть выполнена той же подпрограммой, которой было передано управление на шаге 3, но в ОС чаще всего она реализуется путем последующего вызова соответствующей подпрограммы.

6. Восстановление информации, относящейся к прерванному процессу (этап, обратный шагу 4).

7. Возврат в прерванную программу.

*Исключительная ситуация (exception)* - событие, возникающее в результате попытки выполнения программой команды, которая по каким-то причинам не может быть выполнена до конца. Примерами таких команд могут быть попытки доступа к ресурсу при отсутствии достаточных привилегий или обращение к отсутствующей странице памяти. Исключительные ситуации, как и системные вызовы, являются синхронными событиями, возникающими в контексте текущей задачи. Исключительные ситуации можно разделить на исправимые и неисправимые. К исправимым относятся такие исключительные ситуации, как отсутствие нужной информации в оперативной памяти. После устранения причины исправимой исключительной ситуации программа может выполняться дальше. Возникновение в процессе работы операционной системы исправимых исключительных ситуаций считается нормальным явлением. Неисправимые исключительные ситуации чаще всего возникают в результате ошибок в программах (например, деление на ноль). Обычно в таких случаях операционная система реагирует завершением программы, вызвавшей исключительную ситуацию.

## **28. Функции памяти. Архитектура памяти.**

Память микропроцессорной системы выполняет функцию временного или постоянного хранения данных и команд. Объем памяти определяет допустимую сложность выполняемых системой алгоритмов, а также в некоторой степени и скорость работы системы в целом. Модули памяти выполняются на микросхемах памяти (оперативной или постоянной).

Память микропроцессорной системы представляет собой иерархическую структуру. В основе реализации иерархии памяти современных компьютеров лежат два принципа: принцип локальности обращений и соотношение стоимость/производительность. Принцип локальности обращений говорит о том, что большинство программ, к счастью, не выполняют обращений ко всем своим командам и данным равновероятно, а оказывают предпочтение некоторой части своего адресного пространства.

Иерархия памяти современных компьютеров строится на нескольких уровнях, причем более высокий уровень меньше по объему, быстрее и имеет большую стоимость в пересчете на байт, чем более низкий уровень. Уровни иерархии взаимосвязаны: все данные на одном уровне могут быть также найдены на более низком уровне, и все данные на этом более низком уровне могут быть найдены на следующем нижележащем уровне и так далее, пока мы не достигнем основания иерархии.

Иерархия памяти обычно состоит из многих уровней, но в каждый момент времени мы имеем дело только с двумя близлежащими уровнями. Минимальная единица информации, которая может либо присутствовать, либо отсутствовать в двухуровневой иерархии, называется блоком. Размер блока может быть либо фиксированным, либо переменным. Если этот размер зафиксирован, то объем памяти является кратным размеру блока.

Успешное или неуспешное обращение к более высокому уровню называются соответственно попаданием (hit) или промахом (miss). Попадание - есть обращение к объекту в памяти, который найден на более высоком уровне, в то время как промах означает, что он не найден на этом уровне. Доля попаданий (hit rate) или коэффициент попаданий (hit ratio) есть доля обращений, найденных на более высоком уровне. Иногда она представляется процентами. Доля промахов (miss rate) есть доля обращений, которые не найдены на более высоком уровне.

Поскольку повышение производительности является главной причиной появления иерархии памяти, частота попаданий и промахов является важной характеристикой. Время обращения при попадании (hit time) есть время обращения к более высокому уровню иерархии, которое включает в себя, в частности, и время, необходимое для определения того, является ли обращение попаданием или промахом. Потери на промах (miss penalty) есть время для замещения блока в более высоком уровне на блок из более низкого уровня плюс время для пересылки этого блока в требуемое устройство (обычно в процессор). Потери на промах далее включают в себя две компоненты: время доступа (access time) - время обращения к первому слову блока при промахе, и время пересылки (transfer time) - дополнительное время для пересылки оставшихся слов блока. Время доступа связано с задержкой памяти более низкого уровня, в то время как время пересылки связано с полосой пропускания канала между устройствами памяти двух смежных уровней.

## **29. Многоуровневая организация памяти. Основные характеристики запоминающих устройств.**

**Основная информация в 28.** Основными характеристиками ЗУ являются информационная емкость, быстродействие и надежность. Информационная емкость ЗУ определяется количеством двоичных единиц информации (бит), которое может храниться в нем. Быстродействие ЗУ характеризуется его временными характеристиками, к которым относятся: время обращения к ЗУ при записи и считывании информации, время считывания или выборки информации. Время обращения (время цикла) характеризует максимальную частоту обращения к данному ЗУ при считывании или записи информации. Время считывания или выборки информации – это интервал времени обращения к ЗУ до получения выходного от подачи сигнала считывания. Время записи информации – это интервал времени от момента подачи сигнала обращения к ЗУ до момента готовности информации к считыванию.

Надежность ЗУ определяется числовыми значениями параметров конструктивной и информационной надежности. Под конструктивной или элементной надежностью понимают вероятность безотказной работы всех элементов или устройства в заданном интервале времени и в заданных условиях эксплуатации.

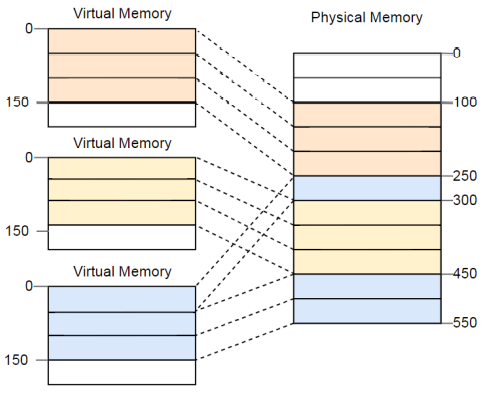
## **30. Средства управления памятью в микропроцессорах. Линейная, сегментная и страничная адресация.**

Блок управления памятью (MMU) представляет собой компьютерный аппаратный блок, через который проходят все ссылки на память, в основном выполняющий преобразование адресов виртуальной памяти в физические адреса.

MMU эффективно выполняет управление виртуальной памятью, одновременно обеспечивая защиту памяти, управление кэшем, арбитраж шины и, в более простых компьютерных архитектурах (особенно в 8-разрядных системах), переключение банков.

*Линейная адресация* – сплошная, без разделения на сегменты – сплошной массив ячеек, неразбиваемый на независимые участки. Благодаря введению механизма линейной адресации можно создавать любое (ограниченное только размерами оперативной памяти) количество адресных пространств. Причём каждая страница линейного адресного пространства может находиться по любому (естественно, выравненному по границе 4 КБайт) физическому адресу, а благодаря обработчику — и на любом накопителе.

*Сегментация* — это деление памяти на сегменты. Это механизм адресации, обеспечивающий существование нескольких независимых адресных пространств как в пределах одной задачи, так и в системе в целом для защиты задач от взаимного влияния. С точки зрения разработчиков программного обеспечения, сегментация дает удобный способ совместного использования информации несколькими процессами. Конкретный сегмент может использоваться совместно с другими без нарушения требований его защиты. Сегментация также предполагает естественное разделение программных строк и данных и отделение модуля от модуля.

*Страничная организация памяти*. Идея состоит в том, чтобы разделить пространство виртуальной и физической памяти на небольшие блоки фиксированного размера. Блоки виртуальной памяти называются страницами, а блоки физического адресного пространства — фреймами. Каждая страница индивидуально сопоставляется с фреймом, что позволяет разделить большие области памяти между несмежными физическими фреймами.

Преимущество становится очевидным, если повторить пример с фрагментированным пространством памяти, но на этот раз с использованием страниц вместо сегментации: В этом примере размер страницы 50 байт, то есть каждая из областей памяти разделена на три страницы. Каждая страница сопоставляется с отдельным фреймом, поэтому непрерывную область виртуальной памяти можно сопоставить с изолированными физическими фреймами. Это позволяет запустить третий экземпляр программы без дефрагментации.

## **31. Методы и способы снижения энергопотребления микропроцессорной системы.**

*Режимы сна* (обычно называемые режимами с низким энергопотреблением), возможно, являются наиболее популярным методом снижения энергопотребления микроконтроллеров. Обычно они включают отключение определенных схем или часов, которые управляют определенными периферийными устройствами микроконтроллеров.

*Режим ожидания / сна* — это самый простой из реализуемых разработчиками режимов пониженного энергопотребления. Этот режим позволяет микроконтроллеру вернуться к полной работе с очень высокой скоростью. Поэтому это не лучший режим, если цикл питания устройства требует, чтобы оно выходило из спящего режима очень часто, так как потребляется большое количество энергии, когда микроконтроллер выходит из спящего режима. Возврат в активный режим из режима ожидания обычно происходит по прерыванию. Этот режим реализуется на микроконтроллере путем отключения дерева тактовых импульсов, которое управляет схемой ЦП, в то время как основные высокочастотные тактовые импульсы микроконтроллера продолжают работать. При этом ЦП может возобновить работу сразу после активации триггера пробуждения. Стробирование тактовой частоты широко используется для отсечения сигналов в режимах низкого энергопотребления микроконтроллеров, и этот режим эффективно стробирует тактовые сигналы через ЦП.

*Режим глубокого сна*, как правило, включает в себя отключение высокочастотных часов и других схем в микроконтроллере, оставляя только схему тактовой частоты, используемую для управления критическими элементами, такими как сторожевой таймер, обнаружение отключения и схемы сброса при включении питания. Другие MCU могут добавлять к нему другие элементы для повышения общей эффективности. Потребляемая мощность в этом режиме может составлять всего 1 мкА в зависимости от конкретного микроконтроллера.

*Режим остановки / выключения*. Некоторые микроконтроллеры имеют разные варианты этого дополнительного режима. В этом режиме генераторы как высокого, так и низкого уровня обычно отключены, оставив включенными только некоторые регистры конфигурации и другие важные элементы.

*Динамическое изменение частоты процессора.* Это еще один широко распространенный метод эффективного снижения мощности, потребляемой микроконтроллером. Это, безусловно, самый старый метод и немного сложнее, чем режимы сна. Он включает в себя микропрограммное обеспечение, динамически управляющее тактовой частотой процессора, чередуя высокую и низкую частоту, поскольку зависимость между частотой процессора и количеством потребляемой мощности является линейной.

## **32. Выключение ядра микропроцессора и периферии (способы снижения энергопотребления).**

Методы сокращения энергопотребления:

- Уменьшение напряжения питания

- Применение внутренней памяти против внешней – не изменяем конструкцию системы, но не активизируем интерфейс ввода/вывода. Если больше использовать внутреннюю память, то будет меньше загрузок из внешней (а это требует больше энергии) уменьшится энергопотребление.

- Регистр управления холостым ходом (ICR) –возможность отключения внутренних модулей при условии, что в данный момент они не нужны; как только они потребуются для работы, подключаем модули.

- Автоматическое управление энергопотреблением (APM) – автоматическое отключение неиспользуемых интерфейсов, при необходимости – подключаются.

- Частота – снижение частоты вызывает снижение потребляемой энергии, однако необходимо учитывать и то, что снизиться производительность (чтобы организовать работу системы нужны ресурсы). Кроме того, нельзя забывать, что для сохранения данных в памяти, нужно определенное количество энергии – иначе данные не сохраняться.

- В спящем режиме сокращается энергопотребление, но опять же может возникнуть проблема с сохранением данных. Помимо этого, при выходе из спящего режима система может потреблять большие ресурсы, и если время выхода достаточно велико, то и потребление энергии в течение данного процесса будет значительным.

Общие принципы построения систем с низким энергопотреблением:

− максимально длительное время нахождения в режимах пониженного энергопотребления (особенно LPM3 и LPM4);

− использование прерываний для управления ходом выполнения программ; − включение периферии только по мере необходимости;

− использование интегрированной периферии с низким потреблением энергии вместо программного выполнения функций;

− вычисление ветвей и использование таблиц значений вместо опроса флагов и длительных программных вычислений;

− избегать частого вызова функций и процедур из-за дополнительных затрат; − использовать однотактные регистры ЦПУ в длинных процедурах;

− отключать недоступные и неиспользуемые сегменты памяти при помощи регистра управления ОЗУ.

● В состояние С0 ядро входит при загрузке, когда происходит прерывание, или после записи по адресу памяти, который отслеживается инструкцией MWAIT.

● Состояния C1/C1E достижимы с помощью инструкций HLT и MWAIT.

● Войти в состояние С3 можно с помощью инструкции MWAIT. Затем кэши L1 и L2 сбрасываются в кэш верхнего уровня (LLC), и все тактовые генераторы процессора останавливаются. Тем не менее, ядро сохраняет свое состояние, так как не обесточено.

● Вход в состояние С6 возможен через инструкцию MWAIT. Ядро сохраняет состояние на выделенную SRAM и напряжение на ядре снижается до нуля. В этом состоянии ядро обесточено. При выходе из C6 состояние ядра восстанавливается из SRAM.

Помимо главного “ядра” у микроконтроллера есть куча периферийных блоков (АЦП, компаратор, таймеры, интерфейсы связи), которые, по сути, работают отдельно и могут общаться с МК на аппаратном уровне. Каждый блок потребляет некоторый ток, и при желании некоторые блоки можно отключить. В режиме глубокого сна они отключаются автоматически, а вот в активной работе нужно отключить их вручную.

## **33. Структура микропроцессорной системы с аналого-цифровым и цифро-аналоговым преобразователем.**

Аналого-цифровой преобразователь (АЦП, англ. Analog-to-digital converter, ADC) — устройство, преобразующее входной аналоговый сигнал в дискретный код (цифровой сигнал). Обратное преобразование осуществляется при помощи ЦАП (цифро-аналогового преобразователя, DAC). Как правило, АЦП — электронное устройство, преобразующее напряжение в двоичный цифровой код.

В основном применяется три типа АЦП:

*- Параллельные* - входной сигнал одновременно сравнивается с эталонными уровнями набором схем сравнения (компараторов), которые формируют на выходе двоичное значение.

*- Последовательного приближения* – в котором при помощи вспомогательного ЦАП генерируется эталонный сигнал, сравниваемый с входным. Эталонный сигнал последовательно изменяется по принципу половинного деления. Это позволяет завершить преобразование за количество тактов, равное разрядности преобразователя, независимо от величины входного сигнала.

*- С измерением временных интервалов* - используются различные принципы преобразования уровней в пропорциональные временные интервалы, длительность которых измеряется при помощи тактового генератора высокой частоты. Иногда называются также считающими АЦП.

Цифро-аналоговый преобразователь (ЦАП) — это устройство для преобразования цифрового кода в аналоговый сигнал по величине, пропорциональной значению кода. ЦАП применяются для связи цифровых управляющих систем с устройствами, которые управляются уровнем аналогового сигнала. Также, ЦАП является составной частью во многих структурах аналого-цифровых устройств и преобразователей. ЦАП характеризуется функцией преобразования.

Основной принцип работы ЦАП – суммирование взвешенных токов или напряжений, когда каждый разряд входного слова вносит соответствующий своему двоичному весу вклад в общую величину получаемого аналогового сигнала; такие ЦАП называют также параллельными или многоразрядными (multibit).

## **34. Назначение и режимы работы таймеров в микропроцессорной системе.**

*Нормальный режим (Normal).* Это самый простой режим. В этом режиме таймер производит подсчет приходящих на его вход импульсов (от тактового генератора или внешнего устройства) и вызывает прерывание по переполнению.

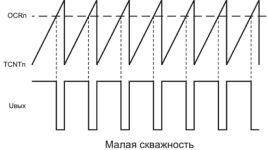
*Режим «Захвата» (Capture)*. Суть этого режима заключается в сохранении содержимого счетного регистра таймера в определенный момент времени. Запоминание происходит либо по сигналу, поступающему через специальный вход микроконтроллера, либо от сигнала с выхода встроенного компаратора. Этот режим удобен в том случае, когда нужно измерить длительность какого-либо внешнего процесса.

*Режим «Сброс при совпадении» (CTC)*. Для работы в режиме CTC используется специальный регистр - регистр совпадения. Если микроконтроллер содержит несколько таймеров, то для каждого из них, существует свой отдельный регистр совпадения. Как же используются регистры совпадения? Эти регистры включаются в работу только тогда, когда выбран режим CTC. В этом режиме, как и в предыдущем, таймер производит подсчет входных импульсов. Текущее значение таймера из его счетного регистра постоянно сравнивается с содержимым регистров совпадения. Если таймер имеет два регистра совпадения, то для каждого из этих регистров производится отдельное сравнение. Когда содержимое счетного регистра совпадет с содержимым одного из регистров совпадения, произойдет вызов соответствующего прерывания. Кроме вызова прерывания в момент совпадения может происходить одно из следующих событий:

● Сброс таймера (верно только для регистров совпадения OCR1 и OCR1 A)

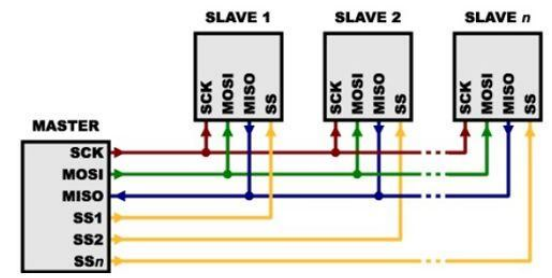
● Изменение состояния одного из выводов микроконтроллера (верно для всех регистров)

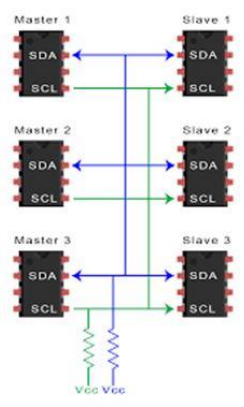
*Режим «Быстродействующий ШИМ» (Fast PWM)*. На вход таймера подаются импульсы от системного генератора. Таймер находится в состоянии непрерывного счета. При переполнении таймера его содержимое сбрасывается в ноль, и счет начинается сначала. В режиме ШИМ переполнение таймера не вызывает прерываний. На рис. 3.5 это показано в виде пилообразной кривой, обозначенной как TCNTn. Кривая представляет собой зависимость содержимого счетного регистра от времени.

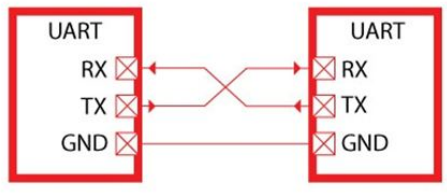


*Асинхронный режим*. В некоторых моделях микроконтроллеров таймер может работать в асинхронном режиме. В этом режиме на вход таймера подается либо частота от внутреннего кварцевого генератора, либо от внешнего генератора. Счетчик не вырабатывает никаких прерываний и дополнительных сигналов. В этом режиме он работает в качестве часов реального времени. Микроконтроллер может предустанавливать содержимое счетного регистра. А затем в любой момент он может считать это содержимое, получив, таким образом, текущее значение реального времени.

## **35. Универсальный синхронный/асинхронный приемопередатчик USART (UART, SPI, I2C) в микропроцессорной системе.**

*SPI (Serial Peripheral Interface)* — это протокол последовательной связи синхронного типа, который состоит из двух линий данных (MOSI и MISO), одной тактовой линии (SCK) и линии выбора подчиненных (SS). В случае с SPI в любой момент времени может быть только одно ведущее устройство и несколько других ведомых, которые отвечают только на вызов ведущего. Вся связь обрабатывается самим ведущим; ни один подчиненный не может отправлять данные по своей воле. Ведущий отправляет данные через MOSI, а ведомые отвечают через линию MISO. Во всем процессе SCK (последовательное тактирование) играет очень важную роль, каждое подчиненное устройство зависит от этих часов, чтобы читать данные из MOSI и отвечать через MISO. SS (выбор ведомого) используется для того, чтобы конкретное подчиненное устройство проснулось, с кем мастер хочет общаться. Ниже представлена иллюстрация принципа подключения посредством интерфейса SPI:

*I2C (Inter-Integrated Circuit) или двухпроводный интерфейс*. Другим очень полезным синхронным протоколом последовательной связи является протокол I2C или Inter-Integrated Circuit. В отличие от SPI, I2C использует только два провода для всего процесса, возможно, поэтому он также известен как протокол двухпроводного интерфейса (TWI). Эти два провода представляют собой SDA (последовательные данные) и SCL (последовательное тактирование). Протокол I2C может поддерживать несколько подчиненных устройств, но в отличие от SPI, который поддерживает только одно ведущее устройство, I2C может также поддерживать несколько мастер-устройств. Каждое устройство отправляет / принимает данные, используя только один провод, который является SDA. SCL поддерживает синхронизацию между устройствами через общую систему тактирования, которая реализуется активным ведущим устройством.

*UART / USART*. UART означает универсальный асинхронный приемник и передатчик, а USART – универсальный синхронный и асинхронный приемник и передатчик. Разница между ними заключается в том, что UART выполняет только асинхронную последовательную связь, в то время как USART может выполнять как синхронный, так и асинхронный последовательный коммуникационный процесс. Для асинхронного режима этот протокол использует только два провода, а именно Rx и Tx. Поскольку здесь не нужна синхронизация, оба устройства должны использовать свои независимые внутренние системы тактирования для функционирования. Тем не менее, существует термин «скорость передачи», который помогает этим устройствам оставаться в режиме синхронизации, фиксируя скорость обмена данными. Скорость передачи данных в бодах равно число бит данных, передаваемое в секунду, поэтому оба устройства должны работать с одинаковой скоростью передачи в бодах, чтобы поддерживать его надлежащее функционирование. Интерфейс UART/USART имеет большое ограничение, связанное с тем, что только два устройства могут обмениваться данными с помощью этого протокола одновременно. Линия Tx одного устройства передает данные на лннию Rx другого устройства и аналогично Tx последнего передает данные в Rx первого устройства. Так происходит обмен данными.

## **36. Программные, программно-аппаратные способы отладки. Методы контроля правильности функционирования.**

В целом средства отладки и диагностирования можно разделить на 2 основные группы:

1. Программные

2. Аппаратно-программные

*Программные средства:*

1. Программные симуляторы

2. Мониторы отладки

3. Интегрированная среда разработки

*Аппаратно-программные средства:*

1. Внутрисхемные эмуляторы

2. Платы развития (оценочные платы)

3. Эмуляторы ПЗУ

4. Логические анализаторы

5. Сигнатурные анализаторы

6. Комплексы диагностирования

*Симулятор* - программное средство, способное имитировать работу микроконтроллера и его памяти. Как правило, симулятор содержит в своем составе:

● Отладчик;

● Модель ЦПУ и памяти.

*Отладочный монитор* - специальная программа, загружаемая в память отлаживаемой системы. Она вынуждает процессор пользователя производить, кроме прикладной задачи, еще и отладочные функции:

● Загрузку прикладных кодов пользователя в свободную от монитора память;

● Установку точек останова;

● Запуск и останов загруженной программы в реальном времени;

● Проход программы пользователя по шагам;

● Просмотр, редактирование содержимого памяти и управляющих регистров.

*Интегрированная среда разработки (IDE)* - совокупность программных средств, поддерживающая все этапы разработки программного обеспечения от написания исходного текста программы до ее компиляции и отладки, и обеспечивающая простое и быстрое взаимодействие с другими инструментальными средствами (внутрисхемным эмулятором, программным симулятором и программатором).

*Внутрисхемный эмулятор* – программно-аппаратное средство, способное замещать собой эмулируемый процессор в реальной схеме. Внутрисхемный эмулятор — это наиболее мощное и универсальное отладочное средство.

*Отладчик* является своеобразным мостом между разработчиком и отладочным средством. Состав и объем информации, проходящей через средства ввода-вывода, доступность ее для восприятия, контроля, и, при необходимости, для коррекции и модификации напрямую зависят от свойств и качества отладчика.

Хороший отладчик позволяет осуществлять:

● Загрузку отлаживаемой программы в память системы;

● Вывод на монитор состояния и содержимого всех регистров и памяти, и при необходимости, их модификацию;

● Управление процессом эмуляции.

*Эмуляционная память*. Наличие эмуляционной памяти дает возможность использовать ее в процессе отладки вместо ПЗУ в отлаживаемой системе, и более того, отлаживать программу без использования реальной системы или ее макета. При необходимости внесения изменений в отлаживаемую программу достаточно загрузить новую или модифицированную программу в память эмулятора, вместо того чтобы заниматься перепрограммированием ПЗУ.

*Процессор точек останова.* Процессор точек останова позволяет останавливать выполнение программы или выполнять иные действия, например, запускать или останавливать трассировщик при выполнении заданных пользователем условий. В отличие от механизма обычных точек останова, процессор точек останова позволяет формировать и отслеживать условия практически любой степени сложности, и при этом эмулируемый процесс не выводится из масштаба реального времени.

*Профилировщик* (иначе анализатор эффективности программного кода) позволяет получить по результатам прогона отлаживаемой программы следующую информацию:

● Количество обращений к различным участкам программы;

● Время, затраченное на выполнение различных участков программы.

Анализ статистической информации, поставляемой профилировщиком, позволяет легко выявлять “мертвые” или перенапряженные участки программ, и в результате оптимизировать структуру отлаживаемой программы.

*Эмулятор ПЗУ* - программно-аппаратное средство, позволяющее замещать ПЗУ на отлаживаемой плате, и подставляющее вместо него ОЗУ, в которое может быть загружена программа с компьютера через один из стандартных каналов связи. Это устройство позволяет пользователю избежать многократных циклов перепрограммирования ПЗУ. Эмулятор ПЗУ имеет смысл только для микроконтроллеров, которые в состоянии обращаться к внешней памяти программ. Это устройство сравнимо по сложности и по стоимости с платами развития. Оно имеет одно большое достоинство: универсальность. Эмулятор ПЗУ может работать с любыми типами микроконтроллеров.

*Логические анализаторы* - контрольно-измерительные приборы, предназначенные для сбора данных о поведении дискретных систем, для обработки этих данных и представления их человеку на различных уровнях абстракции. Они работают независимо и незаметно для испытуемых дискретных систем и применяются для их отладки и диагностирования МПС на всех этапах жизненного цикла.

*Сигнатурный анализатор.* Поиск неисправности в МПС с помощью логического анализатора сопряжен, как правило, со значительными трудностями, прежде всего, это требование высокой квалификации оператора и необходимость знания им принципов работы МПС. Кроме того, для каждой конкретной неисправности необходимо заново разрабатывать процедуру испытаний, настраивать прибор и исследуемую систему на новый режим работы, который, по мнению оператора, позволит обнаружить и локализовать неисправность. Но даже при выполнении этих двух условий локализовать неисправность с точностью до компонента схемы часто не удается.

*Комплексы диагностирования* объединяют возможности логических анализаторов и генераторов слов (приборы, предназначенные для формирования и подачи входных воздействий): способны подавать входные воздействия на диагностируемую систему, собирать и анализировать ответные реакции системы. КД используют главным образом при проектировании как микропроцессорных, так и других дискретных систем, а также для проверки работоспособного состояния и диагностики неисправностей систем при их производстве и эксплуатации. К объекту диагностирования комплекс подключается с помощью выносных зондов, у каждого канала которых три состояния. Управление комплексом может осуществляться через интефейс IEEE-488.