Федеральное агентство по образованию РФ

Государственное образовательное учреждение высшего профессионального образования

АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(ГОУВПО «АмГУ»)

УТВЕРЖДАЮ
Зав. кафедрой АПЭиЭ
_____ А.Н. Рыбалев
«___» _____ 200__г.

Энергетический факультет

кафедра «Автоматизация производственных процессов и электротехники»

Учебно-методический комплекс дисциплины

МИКРОПРОЦЕССОРНЫЕ СИСТЕМЫ УПРАВЛЕНИЯ

для специальности

22.03.01 «Автоматизация технологических процессов и производств»

для специализации

«Автоматизация технологических процессов тепловых электрических станций»

Составитель: Д.А. Теличенко

Благовещенск 2009

Печатается по решению
редакционно-издательского совета
энергетического факультета
Амурского государственного
университета

Микропроцессорные системы управления для специальности 220301 «Автоматизация технологических процессов и производств», для специализации «Автоматизация технологических процессов тепловых электрических станций»: учебно-методический комплекс дисциплины. / Теличенко Д.А. – Благовещенск. Изд-во Амурского гос. ун-та, 2009. 258 с.

Учебно-методический комплекс дисциплины «Микропроцессорные системы управления» представляет собой совокупность учебно-методических документов, призванных обеспечить организацию и содержательную целостность системы методов и средств обучения. Основной целью данного комплекса является систематизация содержания дисциплины, улучшение ее методического обеспечения, правильное планирование и организация работы и контроля знаний студентов.

ОГЛАВЛЕНИЕ

ПРЕДИСЛОВИЕ

Учебно-методический комплекс дисциплины Микропроцессорные системы управления для специальности 22.03.01 «Автоматизация технологических процессов и производств», специализации «Автоматизация технологических процессов тепловых электрических станций»:

составлен на основании Государственного образовательного стандарта ВПО 657900 «Автоматизированные технологии и производства» и учебного плана специальности 22.03.01 «Автоматизация технологических производств»: блок дисциплин специализации ДС.02 «Микропроцессорные системы управления»,

обсужден на заседании кафедры автоматизации производственных процессов и электротехники
«__» _____200__ г., протокол № ___

Заведующий кафедрой _____А.Н. Рыбалев

одобрен на заседании УМС 22.03.01 Автоматизация технологических процессов и производств
«__» _____200__ г., протокол № ___

Председатель_____А.Н. Рыбалев

СОГЛАСОВАНО
Начальник УМУ
_____Г.Н. Торопчина
(подпись, И.О.Ф.)
«__» _____200___ г.

СОГЛАСОВАНО
Председатель УМС факультета
_____Ю.В. Мясоедов
(подпись, И.О.Ф.)
« » 200 г.

СОГЛАСОВАНО
Заведующий выпускающей кафедрой
_____А.Н. Рыбалев
(подпись, И.О.Ф.)
«__» _____200___ г.

1 РАБОЧАЯ ПРОГРАММА

по дисциплине «Микропроцессорные системы управления»
для специальности 22.03.01 «Автоматизация технологических процессов и производств»
для специализации «Автоматизация технологических процессов тепловых электрических станций»

Курс 4,5	Семестр 7,8,9		
Лекции 61 (час.)	Экзамен 7,8		
Практические (семинарские) занятия	29 (час.)	Зачет	9
Лабораторные занятия 45 (час.)		Курсовой проект	8
Самостоятельная работа 135 (час.)			

Всего часов: 270.

ЦЕЛИ И ЗАДАЧИ ДИСЦИПЛИНЫ, ЕЕ МЕСТО В УЧЕБНОМ ПРОЦЕССЕ

Цель преподавания дисциплины – сформировать у студентов знания о методах и способах использования микропроцессорных систем управления для решения различных задач в области автоматизации производственных процессов.

Задача изучения дисциплины: привить навыки по оценке, выбору и использованию современных микропроцессорных систем управления; развить умение проектирования систем управления различной сложности на основе современных микроконтроллеров.

Дисциплина базируется на курсах: «Информатика», «Обшая электротехника и электроника», «Вычислительные машины, системы и сети».

Знания и умения, приобретенные студентами при изучении дисциплины, используются в курсовом проектировании по данному предмету, а так же при выполнении курсового проекта по автоматизации технологических процессов и производств, дипломного проекта и в последующей практической деятельности.

ТРЕБОВАНИЯ К УРОВНЮ ОСВОЕНИЯ СОДЕРЖАНИЯ ДИСЦИПЛИНЫ

В результате изучения дисциплины студенты должны:

знать:

- принципы построения и функционирования микропроцессоров и микроконтроллеров;
- принципы построения и функционирования микропроцессорных систем управления;
- основы организации связей в микропроцессорных системах управления;

уметь:

-анализировать работу микропроцессорных систем управления;
- проектировать системы микроконтроллерные системы управления;
- программировать и отлаживать системы с микроконтроллерами.

СОДЕРЖАНИЕ ДИСЦИПЛИНЫ

1. ЛЕКЦИОННЫЙ КУРС (61 час)

7 семестр (32 часа)

1.1. Микропроцессорные системы – определение, структура, типы – 2ч. Основные определения. Системы с жесткой логикой и гибкой логикой. Понятие о системе команд. Состав простейшего микропроцессора. Организация связей в микропроцессорных системах. Организация выходных каскадов в цифровых схемах. Структура микропроцессорной системы с шинной организацией. Общий принцип работы микропроцессорной системы и информационные потоки, их предназначения. Режимы работы микропроцессорной системы. Понятие архитектуры. Архитектура современных микропроцессорных систем. Системы с общей памятью. Архитектура систем с разделяемой памятью. Сравнительные характеристики обеих архитектур.

1.2. Организация обмена информацией в МПС – 4ч.

Понятие и элементарные циклы обмена. Двухнаправленность и разрядность шин, мультиплексированные шины, особенности передачи информации, понятие асинхронного и синхронного обмена. Циклы обмена информацией: цикл программного обмена (чтение, запись, мультиплексированные асинхронные шины, временные диаграммы, фаза адреса, фаза данных, основные сигналы, модифицированные циклы, немультеплексированные магистрали и их особенности, особенности асинхронного и синхронного обмена). Циклы обмена информацией: цикл обмена по прерываниям (прерывания в ситеме; организация шин при векторных прерываниях: временная диаграмма, принцип работы, основные сигналы; организация шин при радиальных прерываниях: схема, временная диаграмма, принцип работы, основные сигналы; особенности векторных и радиальных прерываний). Циклы обмена информацией: цикл обмена в режиме прямого доступа к памяти (особенности организации режима прямого доступа к памяти,

основные сигналы, принцип работы, структура связей, временные диаграммы). Особенности организации обмена по шинам в микропроцессорной системе: прохождение сигналов по шинам, улучшение организации обмена по шинам.

1.3. Арбитраж шин – 2ч.

Определение и понятия арбитраж шин, его предназначение. Распределение приоритетов, применение схем. Распределение по схеме с динамическим приоритетом, схемы арбитража: централизованный (различные схемы) и децентрализованный опрос (различные схемы)– предназначение и особенности организации. Комбинированные схемы арбитража. Ограничение времени контроля над шиной. Опросные схемы арбитража, централизованный и децентрализованный опрос. Схемы основных опросных методов арбитража, принцип организации и работы.

1.4. Методы повышения эффективности шин – 2ч.

Пакетный режим пересылки информации, его особенности и временная диаграмма работы, преимущества и недостатки, примеры использования. Конвейеризация транзакций, ее особенности, временная диаграмма. Протокол с расщеплением транзакций, особенности, принцип работы, временная диаграмма. Увеличение полосы пропускания шин. Ускорение транзакций. Повышение эффективности шин с множеством ведущих. Надежность и отказоустойчивость, стандартизация шин.

1.5. Основные элементы МПС. Микропроцессор – 2ч.

Микропроцессор основной принцип работы: предназначение, основные сигналы, шины, структура и принцип работы. Характеристики микропроцессора, особенности работы, кварцевый резонатор и тактовая частота и ее влияние на производительность, понятие перегрева процессора и особенности обмена информацией. Организация начального пуска и сброса микропроцессора. Организация питания микропроцессора. Использование буферных регистров в микропроцессоре. Функции микропроцессора. Функциональная структура микропроцессора. Аккумуляторная структура микропроцессора, структура с равноправными регистрами. Служебные функции микропроцессора. Особенности выполнения команд и предназначение счетчика команд. Особенности использования и предназначения регистра признаков. Схемы управления прерыванием и прямым доступом к памяти – предназначение принцип работы. Логика работы.

1.6. Основные элементы МПС. Память и устройства ввода/вывода – 2ч.

Память в микропроцессорной системе: предназначение, виды, разрядность, особенности организации, пространство памяти, схема подключения, особенности организации оперативной и постоянной памяти, области памяти, стек, таблица векторов прерываний, память устройств подключенных к системной шине, подключение внешних устройств, разделение адресного пространства. Устройства ввода/вывода: особенности, обмен информацией, дополнительные устройства для организации обмена, функциональная схема подключения, предназначение основных блоков.

Порты ввода/ввода, последовательная и параллельная организация, принцип работы, устройства интерфейса пользователя, устройства для длительного хранения информации, таймерные устройства.

1.7. Функционирование МПС. Адресация и ее особенности, регистры – 2ч.

Основы программного режима работы микропроцессорной системы: принцип, программы, операнды, код команды, понятие методов адресации. Методы адресации: непосредственная адресация, прямая адресация, регистровая адресация, косвенно-регистровая, относительная адресация. Особенности адресации: автоинкрементирование (декрементирование), неявная адресация, использование различных методов адресации. Сегментное разбитие памяти. Особенности адресации данных. Регистры микропроцессора: универсальные и специализированные, виды, особенности использования, примеры. Особенности использование аккумулятора.

1.8. Программные основы работы МП – 2ч.

Основные группы команд в микропроцессоре, предназначение и их операнды. Команды пересылки данных, функции, примеры, особенности применения и использования на некоторых процессорах. Арифметические команды, их группы, функции, примеры, особенности применения и использования. Логические команды и команды переходов, их группы, функции, примеры, особенности применения и использования на некоторых процессорах.

1.9. Микроконтроллеры. Основы организации – 4ч.

Понятие микроконтроллеров, основные элементы. Структура микроконтроллеров: классы микроконтроллеров (8, 16, 32 –х разрядные, сигнальные процессоры DSP), производители современных микроконтроллеров. Особенности микроконтроллеров (модульная организация, закрытая архитектура, типовые и расширенные функциональные периферийные модули). Типовая структура микроконтроллера. Процессорное ядро и изменяемый функциональный блок. Предназначение основных элементов. Процессорное ядро, его характеристики. Процессоры с CISC-архитектурой, RISC-архитектурой – особенности, отличия, сравнение. Особенности организации памяти в микроконтроллерах: структуры с фон-неймановская (принстонская) и гарвардской архитектурой – особенности, отличия, сравнение. Система команд микроконтроллеров, ее особенности. Схема синхронизации и организации памяти микроконтроллеров, их особенности. Память программ, типы модулей памяти и их особенности. Память данных. Особенности хранения данных и программ. Регистровая и стековая память – предназначение и особенности. Внешняя память.

1.10. Внутренние и внешние связи в микроконтроллерах – 4ч.

Порты ввода/вывода: параллельные и последовательные порты, типы портов, их предназначение, алгоритмы работы. Типичная схема двунаправленного порта ввода/вывода микроконтроллера. Таймеры и процессоры событий: предназначение, структура типичного 16-разрядного таймера/счетчика,

основные недостатки данной схемы, пути усовершенствования данной схемы и современные направления. принцип действия канала входного захвата таймера/счетчика, его схема, типы сигналов, функциональная схема. Структура аппаратных средств канала выходного сравнения – основные сигналы, схема, принцип работы, аппаратные и программные усовершенствования. Модули процессоров событий – предназначение, принцип работы, основные сигналы, реализация режима широтно-импульсной модуляции. Модуль прерываний: принцип работы, источники прерываний, схема приоритетов.

1.11. Аппаратные средства микроконтроллеров – 4ч.

Особенности режимов энергопотребления, минимизация данного режима: активный режим, режим ожидания, режим останова – особенности, предназначение. Тактовые генераторы микроконтроллеров: определения тактовой частоты генератора с помощью кварцевого резонатора, керамического резонатора и внешней RC-цепи, схемы подключения тактовых генераторов, используемые входы, сравнительная характеристика каждого способа подключения. Аппаратные средства обеспечения надежной работы микроконтроллера: схема формирования сигнала сброса, ее предназначение, основные сигналы, принцип работы, типовые схемы формирования сигнала внешнего сброса; блок детектирования пониженного напряжения питания: предназначение, особенности применения, принцип работы; сторожевой таймер: принцип действия, основные используемые сигналы, предназначение, особенности работы. Дополнительные модули, используемые в микроконтроллерах: модули последовательного и параллельного вода/вывода, задачи решаемые данными модулями, их типы, основы функционирования, протоколы интерфейса, современное состояние проблемы передачи информации через порты ввода/вывода; модули аналогового ввода/вывода, основные схемы, принцип работы, схема типового модуля АЦП, основы работы ЦАП и средства реализации данной функции в современных микроконтроллерах.

1.12. Проектирование устройств на микроконтроллерах – 2ч.

Основные этапы разработки: составление технического задания и функциональной спецификации; разработка алгоритма управления, формулирование требований к аппаратной и программной части; выбор типа ядра контроллера, условия, виляющие на данный выбор; разработка структуры аппаратной части, перераспределения функций между аппаратными и программными средствами; возможные модификации технического задания и функциональной спецификации, этапы доработки. Разработка и отладка аппаратных средств, програмного обеспечения. Методы и средства совместной отладки. Заключительный этап проектирование цифровых систем на основе микроконтроллеров.

8 семестр (15 часов)

1.13. Введение: микроконтроллеры серии PIC и AVR – 2ч.

Общие сведенья о микроконтроллерах серии PIC и AVR: состав и назначения семейств, их особенности; история появления и развития; особенности

центрального процессора; отличительные особенности семейств микроконтроллеров; общие сведенья о сопутствующих программных продуктах. Технические характеристики отдельных подгрупп семейства. Представление микроконтроллера с программной точки зрения. Архитектура и характеристики базовых моделей отдельных подгрупп. Структурная схема базовых моделей отдельных подгрупп. Назначение выводов корпусов.

1.14. Принципы работы, организация памяти и особенности выполнения команд для микроконтроллеров PIC и AVR – 2ч.

Принцип работы, временные диаграммы, схема тактирования и выполнения команды, принцип выборки команды. Организация памяти: программ и стека, памяти данных. Особенности выполнения команд: выборка команд, выполнение команд условного и безусловного переходов, аппаратный стек. Особенности методов адресации в микроконтроллерах PIC и AVR.

1.15. Организация обмена с внешними устройствами, память, прерывания для микроконтроллеров PIC и AVR – 2ч.

Порты ввода-вывода, назначение, специфика, принцип организации, схемы портов и отдельных линий, особенности практического использования (программирование и электрическое соединение). Таймеры, структурная схема таймера(счетчика) для микроконтроллеров PIC и AVR, состав, предназначение, принцип работы, реакция на прерывания, особенности программного и аппаратного использования, структурные схемы возможных вариантов использования. Память данных, запись и чтение из памяти данных, используемые регистры и назначение отдельных битов в них. Примеры программного кода, поясняющие принцип работы с памятью данных. Организация прерываний, возможные виды прерываний, особенности, схема логики прерываний микроконтроллера, прерывания отдельных устройств.

1.16. Специальные функции и система команд микроконтроллеров PIC и AVR – 2ч.

Набор специальных функций расширяющих возможности системы: сброс, сторожевой таймер, режим пониженного энергопотребления, выбор типа генератора, защита от кода считываний, биты идентификации, последовательное программирование. Система команд: перечень и формат команд отдельных моделей микроконтроллеров PIC и AVR, описание полей команд, основные форматы команд, количество циклов, изменяемые биты состояния. Специфичные команды: работы с байтами и битами, управления и работы с константами.

1.17. Особенности программирования и отладки, разработка программного кода для микроконтроллеров PIC и AVR – 2ч.

Особенности программирования и отладки: особенности загрузки констант, арифметическо-логических операций, конвейер команд, инструкции для организации ветвлений, стек, память программ и данных, ограниченность ресурсов. Разработка программного кода: различные ассемблеры, цели использования, принципы применения; компоновщики объектного кода; основной текст программы на ассемблере, метки, мнемоники, операнды, формат представления чисел, основные арифметические операторы,

комментарии. Расширения файлов используемых при создании программного кода. Листинг. Директивы языка ассемблер, синтаксис при написании программного кода, поясняющие примеры.

1.18. Разработка программного обеспечения для микроконтроллеров PIC и AVR – 2ч.

Особенности компоновщиков. Особенности менеджеров библиотек. Особенности симуляторов, примеры использования. Завершенные программные пакеты для создания, и исследования программного кода: примеры, принцип работы, использование, преимущества и недостатки, заключительные замечания.

1.19. Макет микропроцессорной системы и программирование простейших задач для микроконтроллеров PIC и AVR – 3ч.

Описание макета, электрическая схема соединения, параметры основных элементов, предназначение и выполняемые функции. Особенности инициализации и запуска в работы. Примеры завершенных программ (текст программы, комментарии к ней, описание принципа работы): программа считывания состояния кнопки и вывода на светодиодный индикатор; программа считывания состояния кнопки и вывода на светодиодный индикатор при определенных условиях; программа для работы с семисегментным индикатором и кнопками; пограмма для работы вывода на семисегментный индикатор числа; подпрограммы формирования задержки; программа для работы с звуковым динамиком; программа для работы с мигающим светодиодом; программа для борьбы с дребезгом контактов.

9 семестр (14 часов)

1.20. Системы малой автоматизации на x-51 совместимых микроконтроллерах – 2ч.

Предпосылки создания, история, направления развития, представители. Распределенные системы управления. Микроконтроллеры с классической архитектурой, коммутирующие и логические возможности, автономный подход к решению сложных задач управления, стоимость и габаритные размеры, децентрализация задач управления в микропроцессорных системах. Локальные вычислительные сети, стандарты, классификация локальных сетей, режим реального времени, форматы фреймов. Факторы способствующие развитию систем малой автоматизации, сетевые технологии, альтернативный подход к построению микропроцессорных систем управления на x-51 совместимых микроконтроллеров, магистрально-модульные системы и типовые интерфейсы.

1.21. Командные и информационные сети на x-51 совместимых микроконтроллерах – 2ч.

Типовая структура, алгоритм работы и основные параметры, базовые интерфейсы, основные диспетчеры, способы передачи и приема информации, примеры микросхем для работы в сетевом режиме, основные элементы схем, их технические характеристики и примеры. Способы управления доступом к каналам связи, диспетчер станции, активный и пассивный доступ. Определение скорости передачи информации. Диспетчеры периферийных

станций: топология сети, структурная схема диспетчера, назначение основных линий и элементов, комбинированный диспетчер. Формат фреймов и общий алгоритм работы: адресация, принцип организации, формат вопроса и ответа, минимизация формата фреймов. Тестовая программа контроллера станций.

1.22. Организация универсальных технологических x-51 контроллеров: введение – 2ч.

Предпосылки создания универсальных технологических контроллеров: аспекты архитектуры, затраты на производство и эксплуатацию, различная номенклатура, и т.п. Основные понятия и тенденции развития универсальных технологических контроллеров: аспекты децентрализации, аспекты сетевого режима работы, слотовая технология, интеллектуальный и магистрально-модульный подход, многоконтроллерные системы, рабочие станции, основной состав систем. Общие технологические требования к главному микроконтроллерному модулю: ядро, питание и сторожевые таймеры, оперативная память, таймеры реального времени, АЦП и ЦАП, сверхоперативная память, порты ввода-вывода, интерфейсы, в том числе и сетевые, питание и отдельные режимы работы. Обобщенная функциональная схема центрального модуля.

1.23. Организация универсальных технологических x-51 контроллеров – супервизоры питания и охранные таймеры – 2ч.

Основные понятия о супервизорах питания и таймерах. Критерии выбора супервизора питания. Предварительный анализ и выбор микросхем супервизоров. Схемы включения микросхем супервизоров. Заключительный этап выбора микросхем супервизоров.

Функциональные характеристики таймеров реального времени. Критерии выбора таймеров реального времени. Анализ и выбор микросхем таймеров реального времени. Микросхемы дополнительной памяти для технологических контроллеров. Схемы включения таймеров реального времени.

1.24. Организация универсальных технологических x-51 контроллеров: устройства ввода-вывода, расширения, ЦАП и АЦП – 2ч.

Устройства ввода-вывода и расширения. Последовательные интерфейсы. Регистры ввода-вывода. Модификаторы адреса. Оптически развязанные узлы. Символьные жидкокристаллические индикаторы.

Аналого-цифровые преобразователи. Критерии выбора АЦП и различные микросхемы. Вспомогательные микросхемы узлов АЦП. Принципиальная схема АЦП на различных составных элементах. Цифро-аналоговые преобразователи. Критерии выбора ЦАП и различные микросхемы. Отбор микросхем ЦАП и их отличия. Параллельные ЦАП. Последовательные ЦАП. Принципиальные схемы подсистем ЦАП.

1.25. Проектирование систем малой автоматизации на основе x-51 совместимых микроконтроллеров – 2ч.

Основные критерии выбора конкретной модели микроконтроллера (сопоставительный анализ моделей x-51 и других). Разработки фирмы Atmel.

Разработки фирмы MAXIM. Разработки фирмы Cygnal. Отдельные вопросы обеспечения пиковой производительности микроконтроллеров.

1.26. Завершенные примеры макетов микропроцессорных систем управления на базе x-51 совместимых микроконтроллеров – 2ч.

Принципиальная схема основных узлов. Соединение линий ввода-вывода. Подсистема аналогового ввода-вывода: схема соединения, основные элементы и их параметры, принцип работы. Подсистема интерфейсов: схема соединения, основные элементы и их параметры, принцип работы. Универсальный технологический контроллер на базе микроконтроллера *C8051F020*. Состав, схема, принцип работы, основные элементы и их параметры. Варианты слотового исполнения системы сбора и обработки данных. Специализированные контроллеры. Специализированный технологический контроллер для работы в составе информационной сети.

2. ПРАКТИЧЕСКИЕ ЗАНЯТИЯ (29 часов)

8 семестр (15 часов)

2.1. Изучение основ работы с контроллером Mega-128 – 3 часа.

Изучение команд ввода/вывода и логических операций. Изучение основ написания программ на ассемблере, реализующих логические функции. Моделирование работы программы в отладчике PLC. Компилирование программы в машинный код. Пересылка программ в контроллер при помощи программатора.

2.2. Изучение работы контроллера Mega-128 при реализации основных алгоритмов – 6 часов.

Изучить основные команды сравнения. Изучить работу таймера. Создать завершенные команды работы контроллера по: поддержанию уровня, по формированию задержек и заданной установке выхода в требуемое значение, формирования аварийной сигнализации по заданным условиям.

2.3. Изучение работы контроллера Mega-128 при реализации завершенных алгоритмов управления – 6 часов.

Создание программ поддержания уровня в баке в двух заданных диапазонах с учетом задания работы контроллера в автоматическом и в ручном режиме. Создание программ поддержания уровня в резервуаре в двух заданных диапазонах, с учетом задержки на включени и выключение в автоматическом и ручном режиме.

9 семестр (14 часов)

2.4. Основы построения системы регулирования на базе микроконтроллера Siemens Simatic S7 200 и SCADA системы TM5 – 4 часа.

Построение системы регулирования уровня. Подключение драйверов контроллера к SCADA системе. Создание проекта, основных узлов проекта, принципы подключения сигналов к TM5 на примере Siemens Simatic S7 200.

2.5. Разработка завершенного проекта системы регулирования на базе микроконтроллера Siemens Simatic S7 200 и SCADA системы TM5 – 4 часа.

Настройка каналов управления микроконтроллера Siemens Simatic S7 200. Создание монитора реального времении для микроконтроллера Siemens

Simatic S7 200. Создание графической базы узлов и разработка графического интерфейса для системы регулирования.

2.6. Разработка вспомогательных элементов проекта системы регулирования на базе микроконтроллера Siemens Simatic S7 200 и SCADA системы TM5 – 4 часа.

Настройка системы архивирования, каналов для архивирования. Создание СПАД-архивов. Создание отчетов тревог. Разработка различных отчетов. Разработка шаблонов, создание сценариев и генерация отчетов.

2.7. Разработка системы управления контроллером с использованием возможностей FBD-программирования на базе микроконтроллера Siemens Simatic S7 200 и SCADA системы TM5 – 2 часа.

Создание и разработка FBD-программ реализующих конкретную задачу управления. Подключение FBD-программы к каналам управления.

3. ЛАБОРАТОРНЫЕ ЗАНЯТИЯ (45 часов)

7 семестр (16 часов)

3.1. Знакомство с учебным стендом УМК – 4 часа.

3.2. Запись и выполнение простых программ на учебном стенде УМК – 4 часа.

3.3. Программная реализация типовых управляющих воздействий и вычислительных процедур на учебном стенде УМК – 4 часа.

3.4. Организация циклов, обработка массивов и маскирование данных на учебном стенде УМК 4 часа.

8 семестр (15 часов)

3.5. Изучение стенда Siemens Simatic S7 200 – 3 часа.

3.6. Создание базовых программ на стенде Siemens Simatic S7 200 – 3 часа.

3.7. Создание программ регулирования по условию на стенде Siemens S7 200 – 3 часа.

3.8. Создание программ автоматического и ручного регулирования по условию на стенде Siemens S7 200 – 3 часа.

3.9. Создание программ автоматического и ручного регулирования по условию с задержкой срабатывания на стенде Siemens S7 200 – 3 часа.

9 семестр (14 часов)

3.10. Основные приемы программирования на Ассемблере для ПК – 5 часов.

3.11. Обработка прерываний на Ассемблере для ПК – 5 часов.

3.12. Создание резидентных программ на Ассемблере для ПК – 4 часа.

4. САМОСТОЯТЕЛЬНАЯ РАБОТА (135 часов).

Самостоятельная работа состоит в подготовке к выполнению лабораторных работ и составлению отчетов по ним (95 часов, в т.ч. 32 часа в 7 семестре, 30 часов в 8 семестре и 33 часа в 9 семестре); выполнение курсового проекта по дисциплине (40 часов, 8 семестр).

Формы контроля самостоятельной работы: защита отчетов по лабораторным работам; защита курсового проекта; итоговый контроль (экзамены и зачеты).

5. КУРСОВОЕ ПРОЕКТИРОВАНИЕ (8 семестр, 40 часов).

Курсовой проект выполняется по индивидуальным (формулировка изменяется каждый год) вариантам в рамках общей темы: «Разработка завершенного цифрового устройства на базе однокристального микроконтроллера с RISC-архитектурой». При этом перед каждым студентом формулируется задача, которую предстоит разрешить с помощью создаваемого устройства. Курсовой проект предусматривает:

- разработку принципиальных и детальных структурных схем для аппаратной и программной части с выбором путей решения (тип микроконтроллера, способы программирования)
- разработку электрической схемы соединения и монтажной схемы печатной платы
- разработка программы на языке низкого уровня, ее компилирование и отладка
- исследование работы всего устройства в целом (с помощью специализированных аппаратно-программных средств)
- составление необходимой документации (технического задания, спецификаций, описаний).

Варианты студентам выдаются в начале семестра, тогда же устанавливается график консультаций и сдачи отдельных частей проекта. Готовый проект сдается преподавателю на проверку не позднее, чем за три дня до установленного срока защиты. Для защиты курсовых проектов создается комиссия из преподавателей кафедры, оценка на которой выставляется согласно действующему положению Амурского государственного университета.

6. ПРОМЕЖУТОЧНЫЕ ФОРМЫ КОНТРОЛЯ ЗНАНИЙ.

Промежуточный контроль знаний студентов по дисциплине предусматривает две контрольные точки в 7 семестре, две в 8 семестре и одну в 9 семестре, оценки по которым выставляются на основе информации о выполнении лабораторных работ и индивидуальных заданий на практических занятиях, а также на основе тестирования теоретических знаний, полученных за прошедший период обучения. Предусматривается тестирование по темам, изученным ко дню проведения текущего тестирования со дня проведения предыдущего тестирования (или начала курса).

7. ВОПРОСЫ К ЭКЗАМЕНУ.

7 семестр

1. Микропроцессорные системы. Основные определения, структуры с гибкой и жесткой логикой, основной элемент системы, организация связей.
2. Структура МП системы с шинной организацией. Общий принцип работы МП системы.

3. Режимы работы МП системы. Архитектура современных микропроцессорных систем.
4. Обмен информации по шинам в МП системе. Понятие и характеристики циклов обмена. Особенности шин.
5. Обмен информации по шинам в МП системе. Цикл программного обмена.
6. Обмен информации по шинам в МП системе. Цикл обмена по прерываниям.
7. Обмен информации по шинам в МП системе. Цикл обмена в режиме прямого доступа к памяти. Особенности организации обмена по шинам.
8. Арбитраж шин. Схемы распределения приоритетов.
9. Схемы арбитража: централизованный арбитраж.
10. Схемы арбитража: децентрализованный арбитраж.
11. Схемы арбитража: опросные схемы.
12. Методы повышения эффективности обмена по шинам в микропроцессорной системе. Пакетный режим, конвейеризация транзакций.
13. Методы повышения эффективности обмена по шинам в микропроцессорной системе. Протокол с расщеплением транзакций, увеличение полосы пропускания, ускорение транзакций, повышение эффективности с множеством ведущих.
14. Микропроцессор – основной принцип работы.
15. Микропроцессор – функциональная структура.
16. Память в микропроцессорной системе.
17. Устройства ввода/вывода в микропроцессорной системе.
18. Функционирование микропроцессорной системы. Основы программного режима работы, методы адресации.
19. Функционирование микропроцессорной системы. Сегментное разбитие памяти.
20. Особенности адресации данных. Регистры микропроцессора.
21. Основные команды микропроцессора. Команды пересылки данных и арифметические команды.
22. Основные команды микропроцессора. Логические команды и команды переходов.
23. Основы организации микроконтроллеров: структура, особенности, типы.
24. Функционирование процессорного ядра микроконтроллеров, различные архитектуры микроконтроллеров.
25. Система команд микроконтроллеров. Схема синхронизации в микроконтроллерах и основы организации памяти.
26. Внутренние и внешние связи в микроконтроллерах. Порты ввода/вывода.
27. Внутренние и внешние связи в микроконтроллерах. Таймеры.
28. Внутренние и внешние связи в микроконтроллерах. Процессоры событий и модуль прерываний.
29. Аппаратные средства микроконтроллеров. Особенности режимов энергопотребления. Тактовые генераторы.

30. Аппаратные средства обеспечения надежной работы микроконтроллеров. Схема формирования сигнала сброса, блок детектирования, сторожевой таймер.
31. Дополнительные модули микроконтроллера. Модули последовательного и параллельного вода/вывода, АЦП и ЦАП.
32. Проектирование цифровых систем на основе микроконтроллеров. Основные этапы.
33. Проектирование цифровых систем на основе микроконтроллеров. Разработка и отладка аппаратных средств.
34. Проектирование цифровых систем на основе микроконтроллеров. Разработка и отладка программных средств. Заключительные этапы проектирования, совместная отладка.

Примечание: В билете два теоретических вопроса (представлены выше) и одна задача, имеющая следующую формулировку:

Пользуясь системой команд микропроцессора КР580ИК80 (лабораторного стенда «УМК») создать программу по заданию, предложенному ниже. При этом необходимо:

а) Написать законченную программу на языке «Ассемблер»;

б) Перевести полученную программу в машинный код;

в) Описать последовательность действий по записи и выполнению программы на лабораторном стенде «УМК»;

г) Для каждого этапа выполнения программы записать содержимое используемых: ячеек памяти, регистров, регистра флагов;

д) Предложить возможные альтернативы по решению задачи.

8 семестр

1. Состав и назначение семейств PIC и AVR контроллеров.
2. Микроконтроллеры отдельных подгрупп семейств PIC и AVR, особенности и отличия.
3. Основные технические характеристики PIC и AVR.
4. Особенности архитектуры базовых моделей PIC и AVR.
5. Принципы работы микроконтроллеров PIC и AVR: временная диаграмма тактирования и циклов выполнения программы, организация памяти программ и стека.
6. Микроконтроллеры PIC и AVR: организация памяти данных.
7. Микроконтроллеры PIC и AVR: регистры специального назначения.
8. Микроконтроллеры PIC и AVR: особенности выполнения команд (выборка команд, прямая и косвенная адресации).
9. Микроконтроллеры PIC: организация работы с внешними устройствами – порты ввода/вывода.
10. Микроконтроллеры AVR: организация работы с внешними устройствами – порты ввода/вывода.
11. Микроконтроллеры PIC и AVR: модуль таймера и регистр таймера.
12. Микроконтроллеры PIC и AVR: память данных в РПЗУ (EEPROM).
13. Микроконтроллеры PIC и AVR: организация прерываний.
14. Микроконтроллеры PIC и AVR: специальные функции.

15. Микроконтроллеры PIC и AVR: система команд – перечень и формат команд.
16. Микроконтроллеры PIC и AVR: система команд – команды работы с байтами и битами.
17. Микроконтроллеры PIC и AVR: система команд – команды управления и работы с константами.
18. Микроконтроллеры PIC и AVR: особенности программирования и отладки.
19. Микроконтроллеры PIC и AVR: разработка программного кода – различные ассемблеры, общие сведения, режимы работы по умолчанию.
20. Микроконтроллеры PIC и AVR: разработка программного кода – метки, мнемоники, операнды комментарии, расширения файлов.
21. Микроконтроллеры PIC и AVR: разработка программного кода – директивы языка.
22. Микроконтроллеры PIC и AVR: разработка программного обеспечения – компоновщики; менеджеры библиотек; симуляторы.
23. Микроконтроллеры PIC и AVR: простейший микроконтроллерный макет – архитектура, схема, принцип организации и работы.
24. Микроконтроллеры PIC и AVR: создание завершенных программ – опрос состояния кнопки и вывод его на индикатор (светодиод).
25. Микроконтроллеры PIC и AVR: создание завершенных программ – использование семисегментного индикатора для контроля за состоянием тумблеров.
26. Микроконтроллеры PIC и AVR: создание завершенных программ – программные методы формирования задержки.
27. Микроконтроллеры PIC и AVR: создание завершенных программ – инициирование звуковых сигналов.
28. Микроконтроллеры PIC и AVR: создание завершенных программ – подавление дребезга контактов и подсчет количества нажатий на кнопку.

Примечание: В билете два теоретических вопроса (представлены выше) и один вопрос теоретическо-практический. Здесь необходимо: описать разработку устройств на базе однокристальных микроконтроллеров для решения конкретных задач (стадии, этапы, особенности, пример).

7. ВОПРОСЫ К ЗАЧЕТУ.

9 семестр

1. Предпосылки создания систем малой автоматизации.
2. Распределенные системы управления в системах малой автоматизации.
3. Локальные вычислительные сети в системах малой автоматизации.
4. Основные понятия и определения систем малой автоматизации.
5. Командные и информационные сети – основные определения и понятия.
6. Диспетчеры периферийных станций.
7. Формат фреймов и общий алгоритм работы.
8. Предпосылки создания универсальных технологических контроллеров.

9. Основные понятия и тенденции развития универсальных технологических контроллеров.
10. Общие технологические требования к главному микроконтроллерному модулю.
11. Обобщенная функциональная схема центрального модуля.
12. Основные понятия о супервизорах питания и таймерах.
13. Критерии выбора супервизора питания.
14. Предварительный анализ и выбор микросхем супервизоров.
15. Схемы включения микросхем супервизоров.
16. Заключительный этап выбора микросхем супервизоров.
17. Функциональные характеристики таймеров реального времени.
18. Критерии выбора таймеров реального времени.
19. Анализ и выбор микросхем таймеров реального времени.
20. Микросхемы дополнительной памяти для технологических контроллеров.
21. Схемы включения таймеров реального времени.
22. Устройства ввода-вывода и расширения.
23. Аналого-цифровые преобразователи.
24. Цифро-аналоговые преобразователи.
25. Основные критерии выбора конкретной модели микроконтроллера.
26. Разработки фирмы Atmel (x-51 контроллеры).
27. Разработки фирмы MAXIM (x-51 контроллеры).
28. Разработки фирмы Cygnal (x-51 контроллеры).
29. Отдельные вопросы обеспечения пиковой производительности микроконтроллеров.
30. Принципиальная схема основных узлов макетов микропроцессорных систем управления на базе x-51 совместимых микроконтроллеров.
31. Подсистема аналогового ввода-вывода макетов микропроцессорных систем управления на базе x-51 совместимых микроконтроллеров.
32. Подсистема интерфейсов макетов микропроцессорных систем управления на базе x-51 совместимых микроконтроллеров.
33. Универсальный технологический контроллер на базе микроконтроллера *C8051F020.*
34. Варианты слотового исполнения системы сбора и обработки данных.
35. Специализированные контроллеры-фотодатчики.
36. Специализированный технологический контроллер для работы в составе информационной сети.

УЧЕБНО-МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ ПО ДИСЦИПЛИНЕ

1. ПЕРЕЧЕНЬ ОБЯЗАТЕЛЬНОЙ (ОСНОВНОЙ) ЛИТЕРАТУРЫ

1.1. Схемотехника электронных систем. Микропроцессоры и микроконтроллеры: Учеб. / В.И. Бойко, А.Н. Гуржий , В.Я. Жуйков и др. – СПб. : БХВ-Петербург, 2004. – 453 с.

1.2. Новиков Ю.В. Основы микропроцессорной техники: курс лекций: Учеб. пособие: Рек. УМО в обл. прикладной информатики / Ю.В. Новиков, П.К. Скоробогатов. – 2-е изд., испр. – М. : Интернет-Ун-т Информ. Технологий, 2004. – 433 с.

1.3. Киреева Э.А. Микропроцессорные устройства, повышающие надежность работы защиты и автоматики: учеб. пособие / Э. А. Киреева. – М. : Изд-во Моск. энергет. ин-та, 2004. – 28 с.

1.4. Электротехнический справочник: в 4 т. / Под общ. ред. В.Г. Герасимов, Под общ. ред. А.Ф. Дьяков, Под общ. ред. Н.Ф. Ильинский, Гл. ред. А.И. Попов. - 8-е изд., испр. и доп. - М. : Изд-во Моск. энергет. ин-та, 2002

1.5. Новиков Ю.В. Основы цифровой схемотехники: базовые элементы и схемы. Методы проектирования: Учеб. / Ю.В. Новиков. - М. : Мир, 2001. - 380 с.

1.6. А.П. Пятибратов, Л.П. Гудыно, А.А. Кириченко. Вычислительные системы, сети и телекоммуникации. Учебник: Рек. Мин. обр. РФ. – М.: Финансы и статистика, 2004, 512 с.

1.7. Н.Л. Прохоров, Г.А. Егоров, В.Е. Красовский и др. Управляющие вычислительные комплексы. Учеб. пособие: Рек. Мин. обр. РФ. – М.: Финансы и статистика, 2003, 352 с.

1.8. Б.Я. Цилькер, С.А. Орлов. Организация ЭВМ и систем. Учеб.: рек. Мин. обр. РФ. – Спб.: Питер, 2006, 668 с.

1.9. А. В. Богданов [и др.] Архитектуры и топологии многопроцессорных вычислительных систем. Курс лекций: учеб. пособие: Рек. УМО вузов. – М.: Мир, 2006, 480 с. – 20 экз.

2. ПЕРЕЧЕНЬ ДОПОЛНИТЕЛЬНОЙ ЛИТЕРАТУРЫ

2.1. Поворознюк А.И. Архитектура компьютеров. Архитектура микропроцессорного ядра и системных устройств: Учеб пособие. Ч.1. – Харьков: Торнадо, 2004, 355 с.

2.2. Поворознюк А.И. Архитектура компьютеров. Архитектура внешней памяти, видеосистемы и внешних интерфейсов: Учеб пособие. Ч.2. – Харьков: Торнадо, 2004, 296 с.

2.3. Теличенко Д. А., Романова М. В. Цифровые узлы и элементы организации вычислительных систем. Лабораторный практикум. Благовещенск: Амурский гос. ун-т, 2004, 104 с.

УЧЕБНО-МЕТОДИЧЕСКАЯ (ТЕХНОЛОГИЧЕСКАЯ) КАРТА ДИСЦИПЛИНЫ

Учебно-методическая карта дисциплины «Микропроцессорные системы управления», соответствующая рабочей программе отражена в виде таблицы 1.

Здесь в графе 6, имеют место обозначения следующих наглядных и методических пособий:

1* – *Теличенко Д.А.* Микропроцессорные системы управления – Простейшие микропроцессоры. Лабораторный практикум.

2.1* Микронтроллер Mega-128. Системное руководство. Электронный вариант.

2.2* Микронтроллер SIEMENS SIMATIC S7-200. Системное руководство. Электронный вариант.

3* – *Теличенко Д.А.* Микропроцессорные системы управления – Программирование микропроцессоров. Лабораторный практикум.

Таблица 1 – Учебно-методическая карта дисциплины.

Номер недели	Номер темы	Вопросы, изучаемые на лекции	Занятия (номера)		Используемые нагляд. и метод. пособия	Самостоятельная работа студентов		Формы контроля
			практич. (семин.)	лаборат.		содержание	час.	
1	2	3	4	5	6	7	8	9
7 семестр								
1	1	Микропроцессорные системы – определение, структура, типы				Подготовка к выполнению лабораторных работ и отчетов по ним	2	Защита лабораторной работы 1, контрольная точка (тест), экзамен
2	2	Организация обмена информацией в МПС		1	1*	Подготовка к выполнению лабораторных работ и отчетов по ним	2	Защита лабораторной работы 1, контрольная точка (тест), экзамен
3	2	Организация обмена информацией в МПС				Подготовка к выполнению лабораторных работ и отчетов по ним	2	Защита лабораторной работы 1, контрольная точка, экзамен
4	3	Арбитраж шин		1	1*	Подготовка к выполнению лабораторных работ и отчетов по ним	2	Защита лабораторной работы 1, контрольная точка (тест), экзамен
5	4	Методы повышения эффективности шин				Подготовка к выполнению лабораторных работ и отчетов по ним	2	Защита лабораторной работы 2, контрольная точка (тест), экзамен
6	5	Основные элементы МПС. Микропроцессор		2	1*	Подготовка к выполнению лабораторных работ и отчетов по ним	2	Защита лабораторной работы 2, контрольная точка (тест), экзамен
7	6	Основные элементы МПС. Память и устройства ввода/вывода				Подготовка к выполнению лабораторных работ и отчетов по ним	2	Защита лабораторной работы 2, контрольная точка (тест), экзамен
8	7	Функционирование МПС. Адресация и ее особенности, регистры		2	1*	Подготовка к выполнению лабораторных работ и отчетов по ним	2	Защита лабораторной работы 2, контрольная точка (тест), экзамен

Продолжение таблицы 1.

1	2	3	4	5	6	7	8	9
9	8	Программные основы работы МП				Подготовка к выполнению лабораторных работ и отчетов по ним	2	Защита лабораторной работы 3, контрольная точка (тест), экзамен
10	9	Микроконтроллеры. Основы организации		3	1*	Подготовка к выполнению лабораторных работ и отчетов по ним	2	Защита лабораторной работы 3, контрольная точка (тест), экзамен
11	9	Микроконтроллеры. Основы организации				Подготовка к выполнению лабораторных работ и отчетов по ним	2	Защита лабораторной работы 3, контрольная точка (тест), экзамен
12	10	Внутренние и внешние связи в микроконтроллерах		3	1*	Подготовка к выполнению лабораторных работ и отчетов по ним	2	Защита лабораторной работы 3, контрольная точка (тест), экзамен
13	10	Внутренние и внешние связи в микроконтроллерах				Подготовка к выполнению лабораторных работ и отчетов по ним	2	Защита лабораторной работы 4, контрольная точка (тест), экзамен
14	11	Аппаратные средства микроконтроллеров		4	1*	Подготовка к выполнению лабораторных работ и отчетов по ним	2	Защита лабораторной работы 4, контрольная точка (тест), экзамен
15	11	Аппаратные средства микроконтроллеров				Подготовка к выполнению лабораторных работ и отчетов по ним	2	Защита лабораторной работы 4, контрольная точка (тест), экзамен
16	12	Проектирование устройств на микроконтроллерах		4	1*	Подготовка к выполнению лабораторных работ и отчетов по ним	2	Защита лабораторной работы 4, контрольная точка (тест), экзамен

Продолжение таблицы 1.

1	2	3	4	5	6	7	8	9
8 семестр								
1			1	5	2.1*, 2.2*	Подготовка к выполнению лабораторных работ и отчетов по ним. Курсовое проектирование.	2 3	Защита лабораторных работ (курсового проекта), контрольная точка (тест), экзамен
2	13	Введение: микроконтроллеры серии PIC и AVR				Подготовка к выполнению лабораторных работ и отчетов по ним. Курсовое проектирование.	2 3	Защита лабораторных работ (курсового проекта), контрольная точка (тест), экзамен
3			1,2	5,6	2.1*, 2.2*	Подготовка к выполнению лабораторных работ и отчетов по ним. Курсовое проектирование.	2 3	Защита лабораторных работ (курсового проекта), контрольная точка (тест), экзамен
4	14	Принципы работы, организация памяти и особенности выполнения команд для микроконтроллеров PIC и AVR				Подготовка к выполнению лабораторных работ и отчетов по ним. Курсовое проектирование.	2 3	Защита лабораторных работ (курсового проекта), контрольная точка (тест), экзамен
5			2	6	2.1*, 2.2*	Подготовка к выполнению лабораторных работ и отчетов по ним. Курсовое проектирование.	2 3	Защита лабораторных работ (курсового проекта), контрольная точка (тест), экзамен
6	15	Организация обмена с внешними устройствами, память, прерывания для микроконтроллеров PIC и AVR				Подготовка к выполнению лабораторных работ и отчетов по ним. Курсовое проектирование.	2 3	Защита лабораторных работ (курсового проекта), контрольная точка (тест), экзамен
7			2	7	2.1*, 2.2*	Подготовка к выполнению лабораторных работ и отчетов по ним. Курсовое проектирование.	2 3	Защита лабораторных работ (курсового проекта), контрольная точка (тест), экзамен
8	16	Специальные функции и система команд микроконтроллеров PIC и AVR				Подготовка к выполнению лабораторных работ и отчетов по ним. Курсовое проектирование.	2 3	Защита лабораторных работ (курсового проекта), контрольная точка (тест), экзамен
9			2,3	7,8	2.1*, 2.2*	Подготовка к выполнению лабораторных работ и отчетов по ним. Курсовое проектирование.	2 3	Защита лабораторных работ (курсового проекта), контрольная точка (тест), экзамен

Продолжение таблицы 1.

1	2	3	4	5	6	7	8	9
10	17	Особенности программирования и отладки, разработка программного кода для микроконтроллеров PIC и AVR				Подготовка к выполнению лабораторных работ и отчетов по ним. Курсовое проектирование.	2 3	Защита лабораторных работ (курсового проекта), контрольная точка (тест), экзамен
11			3	8	2.1*, 2.2*	Подготовка к выполнению лабораторных работ и отчетов по ним. Курсовое проектирование.	2 3	Защита лабораторных работ (курсового проекта), контрольная точка (тест), экзамен
12	18	Разработка программного обеспечения для микроконтроллеров PIC и AVR				Подготовка к выполнению лабораторных работ и отчетов по ним. Курсовое проектирование.	2 3	Защита лабораторных работ (курсового проекта), контрольная точка (тест), экзамен
13			3	9	2.1*, 2.2*	Подготовка к выполнению лабораторных работ и отчетов по ним. Курсовое проектирование.	2 3	Защита лабораторных работ (курсового проекта), контрольная точка (тест), экзамен
14	19	Макет микропроцессорной системы и программирование простейших задач для микроконтроллеров PIC и AVR				Подготовка к выполнению лабораторных работ и отчетов по ним. Курсовое проектирование.	2 1	Защита лабораторных работ (курсового проекта), контрольная точка (тест), экзамен
15			3	9	2.1*, 2.2*	Подготовка к выполнению лабораторных работ и отчетов по ним.	2	Защита лабораторных работ (курсового проекта), контрольная точка (тест), экзамен
9 семестр								
1	20	Системы малой автоматизации на x-51 совместимых микроконтроллерах				Подготовка к выполнению лабораторных работ и отчетов по ним.	2,5	Защита лабораторных работ, контрольная точка (тест), экзамен
2			4	10	2.1*, 2.2*	Подготовка к выполнению лабораторных работ и отчетов по ним.	2,5	Защита лабораторных работ, контрольная точка (тест), экзамен
3	21	Командные и информационные сети на x-51 совместимых микроконтроллерах				Подготовка к выполнению лабораторных работ и отчетов по ним.	2,5	Защита лабораторных работ, контрольная точка (тест), экзамен

Продолжение таблицы 1.

1	2	3	4	5	6	7	8	9
4			4	10	2.1*, 2.2*	Подготовка к выполнению лабораторных работ и отчетов по ним.	2,5	Защита лабораторных работ, контрольная точка (тест), экзамен
5	22	Организация универсальных технологических x-51 контроллеров: введение				Подготовка к выполнению лабораторных работ и отчетов по ним.	2,5	Защита лабораторных работ, контрольная точка (тест), экзамен
6			5	10,11	2.1*, 2.2*	Подготовка к выполнению лабораторных работ и отчетов по ним.	2,5	Защита лабораторных работ, контрольная точка (тест), экзамен
7	23	Организация универсальных технологических x-51 контроллеров – супервизоры питания и охранные таймеры				Подготовка к выполнению лабораторных работ и отчетов по ним.	2,5	Защита лабораторных работ, контрольная точка (тест), экзамен
8			5	11	2.1*, 2.2*	Подготовка к выполнению лабораторных работ и отчетов по ним.	2,5	Защита лабораторных работ, контрольная точка (тест), экзамен
9	24	Организация универсальных технологических x-51 контроллеров: устройства ввода-вывода, расширения, ЦАП и АЦП				Подготовка к выполнению лабораторных работ и отчетов по ним.	2,5	Защита лабораторных работ, контрольная точка (тест), экзамен
10			6	11	2.1*, 2.2*	Подготовка к выполнению лабораторных работ и отчетов по ним.	2,5	Защита лабораторных работ, контрольная точка (тест), экзамен
11	25	Проектирование систем малой автоматизации на основе x-51 совместимых микроконтроллеров				Подготовка к выполнению лабораторных работ и отчетов по ним.	2,5	Защита лабораторных работ, контрольная точка (тест), экзамен
12			6	12	2.1*, 2.2*	Подготовка к выполнению лабораторных работ и отчетов по ним.	2,5	Защита лабораторных работ, контрольная точка (тест), экзамен
13	26	Завершенные примеры макетов микропроцессорных систем управления на базе x-51 совместимых микроконтроллеров				Подготовка к выполнению лабораторных работ и отчетов по ним.	2,5	Защита лабораторных работ, контрольная точка (тест), экзамен
14			7	12	2.1*, 2.2*	Подготовка к выполнению лабораторных работ и отчетов по ним.	0,5	Защита лабораторных работ, контрольная точка (тест), экзамен

2 ГРАФИК САМОСТОЯТЕЛЬНОЙ РАБОТЫ СТУДЕНТОВ

График самостоятельной работы студентов (соответствующий учебной программе и учебно-методической карте, см. табл.1) по дисциплине «Микропроцессорные системы управления», отражен в виде таблицы 2. Здесь время выполнения каждого пункта соответствует графику учебного процесса, т.е. происходит до момента проверки работы.

Таблица 2 – График самостоятельной работы студентов.

Самостоятельная работа студентов		Формы контроля
содержание	час.	
1	2	3
7 семестр		
Подготовка и защита лабораторной работы № 1	8	Защита лабораторной работы 1, контрольная точка (тест), экзамен
Подготовка и защита лабораторной работы №2	8	Защита лабораторной работы 2, контрольная точка (тест), экзамен
Подготовка и защита лабораторной работы №3	8	Защита лабораторной работы 3, контрольная точка (тест), экзамен
Подготовка и защита лабораторной работы №4	8	Защита лабораторной работы 4, контрольная точка (тест), экзамен
8 семестр		
Подготовка и защита лабораторной работы №1(5)	6	Защита лабораторной работы 1(5), контрольная точка (тест), экзамен
Подготовка и защита лабораторной работы №2(6)	6	Защита лабораторной работы 2(6), контрольная точка (тест), экзамен
Подготовка и защита лабораторной работы №3(7)	6	Защита лабораторной работы 3(7), контрольная точка (тест), экзамен
Подготовка и защита лабораторной работы №4(8)	6	Защита лабораторной работы 4(8), контрольная точка (тест), экзамен
Подготовка и защита лабораторной работы №5(9)	6	Защита лабораторной работы 5(9), контрольная точка (тест), экзамен
Разработка раздела 1 курсового проекта	3	Предварительная по графику, на защите
Разработка раздела 2 курсового проекта	3	Предварительная по графику, на защите
Разработка раздела 3 курсового проекта	3	Предварительная по графику, на защите
Разработка раздела 4 курсового проекта	3	Предварительная по графику, на защите
Разработка раздела 5 курсового проекта	3	Предварительная по графику, на защите
Разработка раздела 6 курсового проекта	6	Предварительная по графику, на защите
Написание текста пояснительной записки к курсовому проекту (введение, заключение, реферат, список литературы)	12	Предварительная по графику, на защите
Составление приложений к пояснительной записке курсового проекта	3	Предварительная по графику, на защите
Составление листов к курсовому проекту	3	Предварительная по графику, на защите
Написание доклада на защиту курсового проекта	1	На защите курсового проекта
9 семестр		
Подготовка и защита лабораторной работы №1(10)	11	Защита лабораторной работы 1(10), контрольная точка (тест), зачет
Подготовка и защита лабораторной работы №2(11)	11	Защита лабораторной работы 2(11), зачет
Подготовка и защита лабораторной работы №3(12)	11	Защита лабораторной работы 3(12), зачет

3 МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ПРОВЕДЕНИЮ ЗАНЯТИЙ

3.1 САМОСТОЯТЕЛЬНАЯ РАБОТА И КУРСОВОЕ ПРОЕКТИРОВАНИЕ

Самостоятельная работа в соответствии с учебно-методической картой дисциплины (см. п.2, табл. 1,2) состоит в:

- Проработке материалов лабораторных (7, 8, 9 семестр) и практических занятий (8, 9 семестр). При этом студенты по выдаваемому материалу предварительно готовятся к занятию и составляют отчет. В случае необходимости задают возникшие вопросы на лабораторном (практическом) занятии. Основной формой контроля проработки материала является опрос, проводимый при допуске к лабораторной работе (ее защите), контроль при выполнении практических занятий, так же результаты контрольных работ (тестов)
- Защита курсового проекта.

3.2 ЛАБОРАТОРНЫЕ РАБОТЫ

Лабораторный практикум представляет собой 12 лабораторных работ. Из них 4 выполняются в 7 семестре, 5 в 8 семестре и 3 в 9 семестре.

Для выполнения лабораторных работ студентам предварительно предлагается самостоятельно ознакомиться с краткой тсорисй к каждой выполняемой работе. Это даст необходимую теоретическую основу и облегчит выполнение работ, позволив на занятии уделить большее внимание вопросам, обычно вызывающим наибольшее затруднение. Сами задания в каждом семестре расположены в возрастающем порядке сложности.

Все занятия делятся на два цикла: выполнение работы и защита работы. Циклы повторяются для каждой работы, в порядке следования, без нарушения очередности.

При проведении лабораторных работ рекомендуется: совместить в рамках проведения одного занятия темы исследования так, что бы равномерно распределить студентов (группы студентов) по лабораторным стендам.

Допускается после выполнения очередного цикла всеми студентами (группой студентов) в случае оставшегося времени уделить время на ликвидацию образовавшихся задолжностей, если таковые имеются.

На вводном занятии: студенты группы делятся на бригады по два – три человека, им присваиваются варианты, номера которых сохраняются за ними на протяжении всего курса.

На первом этапе цикла (снятия работы):

- преподавателем осуществляется допуск к работе, на котором проверяется: знание студентов краткой теории по выполняемой работе; наличие заготовки отчета;
- выясняются вопросы, вызвавшие у студентов затруднения, даются необходимые пояснения по ним;

- даются комментарии по методике проведения работы;
- контролируется выполнение работы каждой бригады и всеми студентами в целом.

Работа считается снятой, если: студенты одной бригады, и каждый в отдельности, выполнили все задания работы, согласно вариантам; зафиксировали необходимые данные в заготовке отчета.

На втором этапе цикла (защита работы):
- преподавателем проверяется личный отчет каждого из студентов, задаются вопросы по ходу выполнения работы; задаются контрольные вопросы.

Работа считается защищенной, если: правильно выполнен отчет по работе; даны корректные ответы на вопросы преподавателя.

Представляемый отчет (после успешной защиты работы отчет сдается преподавателю и сохраняется до успешной сдачи студентом экзамена или зачета) должен удовлетворять следующим требованиям:
- отчет выполняется на одной стороне белого листа формата А4 в рукописной или печатной форме, в варианте возможном для прочтения (почерк, шрифт, размер, интервал);
- титульный лист должен содержать следующие сведения: название предмета; тему работы, с ее порядковым номером; фамилию студента выполнившего работу с указанием номера группы и вариантов (личного и на бригаду); фамилию преподавателя, осуществляющего прием работы; дату снятия и защиты (дата защиты заполняется преподавателем лично).
- основная часть работы должна содержать следующие сведения: краткую теорию; цель работы; элементы, приборы и инструменты, используемые в работе; ход работы с необходимыми рисунками, схемами, таблицами и формулами.

В случае если студент не снял или не защитил работу, он может приступать к следующей работе. Ликвидировать возникшую задолжность можно на оставшемся времени после проведения очередной лабораторной работы или на дополнительных занятиях. Если ликвидировать задолжность по лабораторным работам в течение семестра не удается, студент является на экзамен с отчетами по несданным работам, где ему до ответа на экзаменационные вопросы дается возможность защитить каждую работу.

3.3. ПРАКТИЧЕСКИЕ ЗАНЯТИЯ

Предварительно студенты знакомятся со списком всех изучаемых тем, рассматриваемых на практических занятиях. Практические занятия проводятся в 8 и 9 семестре (в 7 семестре не предусмотрены). Характер вопросов, прорабатываемых здесь, является дополнением к материалу, изучаемому на лекциях и на лабораторных работах. Целью занятий является получение практических навыков по работе с микропроцессорными системами управления на базе современных микроконтроллеров. Так в 8 семестре время уделя-

ется решению задач создания систем управления на основе RISC-контроллера «Mega-128», где студенты проектируют простейшие автономные устройства и алгоритмы управления им. В 9 семестре с использованием микроконтроллера Siemens Simatic S7 200 и SCADA системы TraceMode студенты учатся строить комплекнсые микропроцессорные системы управления.

Все задачи решаются студентами самостоятельно (либо по подгруппам). Для проведения практических занятий рекомендуется использование лабораторий кафедры АППиЭ, оснащенных соответствующими средствами вычислительной техники (компьютерами и микроконтроллерами). Преподавателем контролируется не только правильность решения, но и даются: практические рекомендации по решению подобных заданий, применимость рассматриваемых тем к практике, а так же предлагается другие варианты решения. Каждому студенту (группе), по результатам занятия выставляется оценка.

3.4. КОНТРОЛЬНЫЕ РАБОТЫ

Контрольные работы предназначены для проверки текущей успеваемости студентов и являются одним из важнейших показателей при выставлении аттестационных оценок в семестре.

Основу контрольных работ составляют тесты, по темам указанным в п.7.4. Сам перечень вопросов формируется исходя из комплексного теста, представленного в разделе 10.2.

Контрольные работы рекомендуется проводить в рамках учебного плана, по порядку вопросов рассмотренных на лекциях, лабораторных и практических занятиях. В связи с большим объемом лабораторных и практических работ (а так же отсутствие в некоторых семестрах либо лабораторных, либо практических занятий) контрольные работы рекомендуется проводить на лекционных занятиях. Минимальное количество контрольных работ – 5, из них 2 в 7 семестре, 2 в 8 семестре и 1 в 9 семестре, с длительностью каждой работы – 20 минут. При этом количество работ может быть увеличено, например по 1 работе, содержащей 3 тестовых вопроса. В этом случае контрольные работы проводятся в конце каждого лекционного занятия, и общая длительность работы не превышает 5 минут.

4 СПИСОК ЛИТЕРАТУРЫ

Полный список рекомендованной (основной и дополнительной литературы) приведен ниже. Отдельно выделена литература, имеющаяся у ведущего лекционные занятия преподавателя и предназначенная для выполнения курсового проекта

1. ПЕРЕЧЕНЬ ОБЯЗАТЕЛЬНОЙ (ОСНОВНОЙ) ЛИТЕРАТУРЫ

1.1. Схемотехника электронных систем. Микропроцессоры и микроконтроллеры: Учеб. / В.И. Бойко, А.Н. Гуржий , В.Я. Жуйков и др. – СПб. : БХВ-Петербург, 2004. – 453 с.

1.2. Новиков Ю.В. Основы микропроцессорной техники: курс лекций: Учеб. пособие: Рек. УМО в обл. прикладной информатики / Ю.В. Новиков, П.К. Скоробогатов. – 2-е изд., испр. – М. : Интернет-Ун-т Информ. Технологий, 2004. – 433 с.

1.3. Киреева Э.А. Микропроцессорные устройства, повышающие надежность работы защиты и автоматики: учеб. пособие / Э. А. Киреева. – М. : Изд-во Моск. энергет. ин-та, 2004. – 28 с.

1.4. Электротехнический справочник: в 4 т. / Под общ. ред. В.Г. Герасимов, Под общ. ред. А.Ф. Дьяков, Под общ. ред. Н.Ф. Ильинский, Гл. ред. А.И. Попов. - 8-е изд., испр. и доп. - М. : Изд-во Моск. энергет. ин-та, 2002

1.5. Новиков Ю.В. Основы цифровой схемотехники: базовые элементы и схемы. Методы проектирования: Учеб. / Ю.В. Новиков. - М. : Мир, 2001. - 380 с.

1.6. А.П. Пятибратов, Л.П. Гудыно, А.А. Кириченко. Вычислительные системы, сети и телекоммуникации. Учебник: Рек. Мин. обр. РФ. – М.: Финансы и статистика, 2004, 512 с.

1.7. Н.Л. Прохоров, Г.А. Егоров, В.Е. Красовский и др. Управляющие вычислительные комплексы. Учеб. пособие: Рек. Мин. обр. РФ. – М.: Финансы и статистика, 2003, 352 с.

1.8. Б.Я. Цилькер, С.А. Орлов. Организация ЭВМ и систем. Учеб.: рек. Мин. обр. РФ. – Спб.: Питер, 2006, 668 с.

1.9. А. В. Богданов [и др.] Архитектуры и топологии многопроцессорных вычислительных систем. Курс лекций: учеб. пособие: Рек. УМО вузов. – М.: Мир, 2006, 480 с. – 20 экз.

2. ПЕРЕЧЕНЬ ДОПОЛНИТЕЛЬНОЙ ЛИТЕРАТУРЫ

2.1. Поворознюк А.И. Архитектура компьютеров. Архитектура микропроцессорного ядра и системных устройств: Учеб пособие. Ч.1. – Харьков: Торнадо, 2004, 355 с.

2.2. Поворознюк А.И. Архитектура компьютеров. Архитектура внешней памяти, видеосистемы и внешних интерфейсов: Учеб пособие. Ч.2. – Харьков: Торнадо, 2004, 296 с.

2.3. Теличенко Д. А., Романова М. В. Цифровые узлы и элементы организации вычислительных систем. Лабораторный практикум. Благовещенск: Амурский гос. ун-т, 2004, 104 с.

3. ПЕРЕЧЕНЬ ИМЕЮЩЕЙСЯ ЛИТЕРАТУРЫ

3.1. Баранов В.Н. Применение микроконтроллеров AVR: схемы, алгоритмы, программы. – М.: Издательский дом Додэка-XXI, 2004. – 288 с.

3.2. Белов А.В. Конструирование устройств на микроконтролллерах. – СПб.: Наука и Техника, 2005. – 256 с.

3.3. Белов А.В. Микроконтроллеры AVR в радиолюбительской практике. – СПб.: Наука и Техника, 2007. – 352 с.

3.4. Белов А.В. Самоучитель по микропроцессорной технике. – СПб.: Наука и Техника, 2003. – 224 с.

3.5. Бродин В.Б., Калинин А.В. Системы на микроконтроллерах и БИС программируемой логики. – М.: Издательство ЭКОМ, 2002. – 400 с.

3.6. Голубцов М.С. Микроконтроллеры AVR: от простого к сложному. – М.: СОЛОН-Пресс, 2003. – 288 с.

3.7. Гребнев В.В. Микроконтроллеры семейства AVR фирмы Atmel. – М.: ИП РадиоСофт, 2002. – 176 с.

3.8. Евстифеев А.В. Микроконтроллеры AVR семейства Classic фирмы ATMEL. – М.: Издательский дом Додэка-XXI, 2006. – 266 с.

3.9. Евстифеев А.В. Микроконтроллеры AVR семейства Tiny и Mega фирмы ATMEL. М.: Издательский дом Додэка-XXI, 2004. – 560 с.

3.10. Евстифеев А.В. Микроконтроллеры AVR семейства Mega. Руководство пользователя. – М.: Издательский дом Додэка-XXI, 2007. – 592 с.

3.11. Мортон Дж. Микроконтроллеры AVR. Вводный курс. /Пер. с англ. – М.: Издательский дом Додэка-XXI, 2006. – 272 с.

3.12. Трамперт В. AVR-RISC микроконтроллеры.: Пер. с нем. – К.: МК-Пресс, 2006. – 464 с.

3.13. Трамперт В. Измерение, управление и регулирование с помощью AVR-микроконтроллеров.: Пер. с нем. – К.: МК-Пресс, 2006. – 208 с.

3.14. Заяц Н.И. Радиолюдительские конструкции на PIC-микроконтроллерах. С алгоритмами работы программ и подробными комментариями к исходным текстам. – М.: СОЛОН-Прес, 2003. – 368 с.

3.15. Заяц Н.И. Радиолюдительские конструкции на PIC-микроконтроллерах. С алгоритмами работы программ и подробными комментариями к исходным текстам. Книга 2. – М.: СОЛОН-Прес, 2005. – 192 с.

3.16. Заяц Н.И. Радиолюдительские конструкции на PIC-микроконтроллерах. Книга 3. – М.: СОЛОН-Прес, 2006. – 240 с.

3.17. Шпак Ю.А. Программирование на языке С для AVR и PIC микроконтроллеров. – К.: МК-Пресс, 2006. – 400 с.

3.18. Голобов В.Н. Умный дом своими руками. – М.: НТ Пресс, 2007. – 416 с.

3.19. Катцен С. PIC-контроллеры. Все, что вам необходимо знать. – М.: Додэка-XXI, 2008. – 556с.

3.20. Кенинг А, Кеннинг М. Полное руководство по PIC-контроллерам.: Пер. с нем. – К.: МК-Пресс, 2006. – 256 с.
3.21. Предко М. 123 эксперимента по робототехнике.: Пер. с англ. – М.: НТ Пресс, 2007. – 544 с.
3.22. Предко М. Руководство по микроконтроллерам. Том 1. М: Постмаркет, 2001. – 416 с.
3.23. Предко М. Руководство по микроконтроллерам. Том 2. М: Постмаркет, 2001. – 488 с.
3.24. Предко М. Справочник по PIC-микроконтроллерам.: Пер. с англ. – М.: ДМК Пресс, 2002 – 512 с.
3.25. Предко М. Устройства управления роботами: схемотехника и программирование. М.: ДМК Пресс, 2004. – 404 с.
3.26. Тавернье К. PIC-микроконтроллеры. Практика применения.: Пер. с фр. – М.: ДМК Пресс, 2004. – 272 с.
3.27. Уилмсхерст Т. Разработка встроенных систем с помощью микроконтроллеров PIC. Принципы и практические примены.: Пер. с англ. – К.: МК-Пресс, СПб.: КОРОНА-ВЕК, 2008. – 544 с.
3.28. Ульрих В.А. Микроконтроллеры PIC16X7XXX. – СПб.: Наука и техника, 2002. – 320 с.
3.29. Хелибайк Ч. Программирование PIC-микроконтроллеров на PicBasic. М.: Додэка-XXI, 2008. – 321 с.
3.30. Яценков В.С. Микроконтроллеры Microchip. Практическое руководство. М.: Горячая линия-Телеком, 2002. – 296 с.
3.31. Яценков В.С. Микроконтроллеры Microchip rfPIC со встроенными маломощным радиопередатчикомм. – М.: Горячая линия-Телеком, 2006. – 344 с.
3.32. Гель П. Как превратить персональный компьютер в универсальный программатор.: Пер. с фр. – М.: ДМК, 2000. – 168 с.
3.33. Граф Р.Ф., Шиитс В. Энциклопедия электронных схем. Том 7. Часть 1: пер с англ. – М.: ДМК, 2000. – 304 с.
3.34. Граф Р.Ф., Шиитс В. Энциклопедия электронных схем. Том 7. Часть 2: пер с англ. – М.: ДМК, 2000. – 416 с.
3.35. Граф Р.Ф., Шиитс В. Энциклопедия электронных схем. Том 7. Часть 3: пер с англ. – М.: ДМК, 2001. – 384 с.
3.36. Кауфман М., Сидман А.Г. Практическое руководство по расчетам схем в электронике: Справочник. В 2-х т. Т. 1: Пер. с англ. М.: Энергоатомиздат, 1991. – 368 с.
3.37. Кауфман М., Сидман А.Г. Практическое руководство по расчетам схем в электронике: Справочник. В 2-х т. Т. 2: Пер. с англ. М.: Энергоатомиздат, 1993. – 288 с.
3.38. Маркировка электронных компонентов. – 9-е изд., стер. – М.: Издательский дом Додэка-XXI, 2004. – 208 с.
3.39. Быстродействующие интегральные сикросхемы ЦАП и АЦП и измерение их параметров / А.-Й. К. Марцинкявичус, Э.-А. К. Багданскис,

Р.Л. Пощюнас и др.; Под ред. А.-Й. К. Марцинкявичуса, Э.-А. К. Багданскиса. – М.: Радио и связь, 1988. – 224 с.
3.40. Пирогова Е.В. Проектирование и технология печатных плат: Учебник. – М.: ФОРУМ: ИНФРА-М, 2005. – 560 с.
3.41. Рабаи Жан М., Чандракасан А., Николич Б. Цифровые интегральные схемы. Методология проектирования. – М., К., СПб: Вильямс, 2007. – 911 с.
3.42. Стешенко В.Б. ACCEL EDA. Технология проектирования печатных плат. – С.: Нолидж, 2000. – 512 с.
3.43. Уваров А.С. Программа P-CAD. Электронное моделирование. – М.: Издательство Диалог-МИФИ, 2008. – 192 с.
3.44. Уваров А.С. P-CAD 2002 и SPECCTRA. Разработка печатных плат. – М.: СОЛОН-Пресс, 2003. – 544 с.
3.45. Шрайбер Г. Справочник по микросхемам. Том 1: Пер. с фр. – М.: ДМК Пресс, 2005. – 208 с.
3.46. Шрайбер Г. Справочник по микросхемам. Том 2: Пер. с фр. – М.: ДМК Пресс, 2005. – 200 с.
3.47. Шрайбер Г. Справочник по микросхемам. Том 3: Пер. с фр. – М.: ДМК Пресс, 2005. – 208 с.
3.48. Шрайбер Г. Справочник по микросхемам. Том 4: Пер. с фр. – М.: ДМК Пресс, 2005. – 136 с.

5. ПЛАН КОНСПЕКТ ЛЕКЦИЙ ПО КАЖДОЙ ТЕМЕ

СЕМЕСТР 7

Лекция №1 «Микропроцессорные системы – определение, структура, типы»

Определения: микропроцессор, микропроцессорная система, интерфейс.

Системы с жесткой логикой: понятие, структура, недостатки и преимущества.

Системы с гибкой логикой: понятие, структура, недостатки и преимущества.

Микропроцессор как основной элемент системы с гибкой логикой. Определение программы. Понятие о системе команд. Состав простейшего микропроцессора.

Организация связей в микропроцессорных системах: обычная, шинная. Организация выходных каскадов в цифровых схемах: стандартный выход с двумя состояниями, выход с открытым коллектором, выход с третьим состоянием.

Структура микропроцессорной системы с шинной организацией: основные элементы, схема соединений, предназначения основных шин в системе (шина адреса, шина данных, шина управления, шина питания).

Общий принцип работы микропроцессорной системы и информационные потоки, их предназначения (проходящие через устройства ввода-вывода, память, микропроцессор).

Режимы работы микропроцессорной системы: командный, по прерываниям, режим прямого доступа к памяти (организация, принцип работы основных элементов).

Понятие архитектуры. Архитектура современных микропроцессорных систем. Системы с общей памятью: структура с единой системой шин, структура с выделением шин ввода-вывода и дополнительная шина расширения. Архитектура систем с разделяемой памятью. Сравнительные характеристики обеих архитектур.

Лекция №2 «Организация обмена информацией в микропроцессорных системах I»

Понятие циклов обмена. Элементарные циклы обмена. Некоторые особенности шин микропроцессорных систем: двухнаправленность разрядность, мультиплексированные шины, особенности передачи информации, понятие асинхронного и синхронного обмена.

Циклы обмена информацией: цикл программного обмена:

Циклы чтения/записи на примере мультиплексированной асинхронной шины *Q-bus*. Временные диаграммы, фаза адреса, фаза данных при циклах чтения записи. Основные сигналы на шине при различных фазах. Модифи-

цированные циклы чтения/записи на шине *Q-bus* (чтение-модификация-запись).

Циклы чтения/записи на примере синхронной немультиплексированной магистрали *ISA*. Временные диаграммы, фаза адреса, фаза данных при циклах чтения записи. Основные сигналы на шине при различных фазах. Особенности реализации синхронного и асинхронного режима.

Лекция №3 «Организация обмена информацией в микропроцессорных системах II»

1. Циклы обмена информацией: цикл обмена по прерываниям:

Принципы работы в данном режиме. Прерывания в системе: векторные, радиальные – основные понятия.

Организация шин при векторных прерываниях: схема, временная диаграмма, принцип работы, основные сигналы.

Организация шин при радиальных прерываниях: схема, временная диаграмма, принцип работы, основные сигналы.

Особенности векторных и радиальных прерываний. Сравнительная характеристика.

2. Циклы обмена информацией: цикл обмена в режиме прямого доступа к памяти:

Особенности организации режима прямого доступа к памяти.

Прямой доступ к памяти в магистрали *Q-bus*. Основные сигналы, принцип работы.

Прямой доступ к памяти в магистрали в магистрали *ISA*. Структура связей, основные сигналы, принцип работы, временны диаграммы.

3. Особенности организации обмена по шинам в микропроцессорной системе. Факторы, влияющие на прохождение сигналов по шинам. Основные подходы, применяемые для улучшения организации обмена по шинам, оконечные согласователи.

Лекция №4 «Арбитраж шин»

Определение понятия арбитраж шин, его предназначение.

1. Распределение приоритетов:

Организация статического и динамического приоритетов, применение данных схем.

Распределение по схеме с динамическим приоритетом: простая циклическая смена приоритетов, циклическая смена приоритетов с учетом последнего запроса, смена приоритетов по случайному закону, схема равных приоритетов, алгоритм наиболее давнего использования – особенности организации и предназначение.

2. Схемы арбитража: централизованный и децентрализованный опрос – предназначение и особенности организации.

Централизованный арбитраж, параллельная и последовательная схема, предназначение шин и особенности работы, основные схемы.

Децентрализованный арбитраж, способа организации, применение, схема подсоединения устройств при данном виде арбитража.

3. Общие замечания по схемам организации арбитража.

Преимущества схем децентрализованного арбитража, надежность данных схем. Комбинированные схемы арбитража. Ограничение времени контроля над шиной.

Опросные схемы арбитража, централизованный и децентрализованный опрос. Схемы основных опросных методов арбитража, принцип организации и работы.

Лекция №5 «Методы повышения эффективности шин»

Методы повышения эффективности шин:

Пакетный режим пересылки информации, его особенности и временная диаграмма работы, преимущества и недостатки, примеры использования.

Конвейеризация транзакций, ее особенности, временная диаграмма.

Протокол с расщеплением транзакций, особенности, принцип работы, временная диаграмма.

Увеличение полосы пропускания шин: отказ от мультиплексирования шин адреса и данных увеличение ширины шины данных, повышение тактовой частоты шины, использование блочных транзакций (предназначение и принцип работы).

Ускорение транзакций: арбитраж с перекрытием, арбитражу с удержанием шины, расщепление транзакций (предназначение и принцип работы).

Повышение эффективности шин с множеством ведущих: способы достижения, особенности организации.

Надежность и отказоустойчивость, стандартизация шин.

Лекция №6 «Основные элементы микропроцессорной системы. Микропроцессор»

1. Микропроцессор – основной принцип работы:

Предназначение микропроцессора, основные сигналы использующиеся в микросхеме, шины микропроцессора, его структура и принцип работы.

Важнейшие характеристики микропроцессора.

Особенности работы микропроцессора, кварцевый резонатор и тактовая частота и ее влияние на производительность, понятие перегрева процессора и особенности обмена информацией.

Организация начального пуска и сброса микропроцессора, сигнал сброса, его предназначение и использование.

Организация питания микропроцессора.

Использование буферных регистров в микропроцессоре.

Программа начального пуска.

Функции микропроцессора.

2. Функциональная структура микропроцессора:

Схема управления выборкой команд: состав, принцип работы. Организация конвейера, использование кэш памяти.

Арифметическо-логическое устройство: состав принцип работы, предназначение. Сопроцессоры.

Регистры процессора: предназначение, принцип организации и использования. Аккумуляторная структура микропроцессора, структура с равноправными регистрами. Служебные функции микропроцессора. Особенности выполнения команд и предназначение счетчика команд. Особенности использования и предназначения регистра признаков.

Схемы управления прерыванием и прямым доступом к памяти - предназначение принцип работы. Логика работы.

Лекция №7 «Основные элементы микропроцессорной системы. Память и устройства ввода/вывода»

1. Память в микропроцессорной системе:

Предназначение памяти, ее виды, разрядность, особенности организации, пространство памяти.

Схема подключения памяти, основные элементы схемы подключения, их предназначение.

Особенности организации оперативной и постоянной памяти.

Области памяти: память начального пуска, память стека, основные команды для работы со стеком, таблица векторов прерываний (аппаратное и программное прерывание), память устройств подключенных к системной шине.

Особенности организации адресного пространства с точки зрения подключения внешних устройств, разделение адресного пространства.

2. Устройства ввода/вывода:

Особенности работы устройств ввода/ввода. Обмен информацией через устройства ввода/ввода и памятью, микропроцессором, дополнительные устройства для организации обмена через устройства ввода/ввода.

Функциональная схема подключения устройств ввода/ввода в микропроцессорной системе. Предназначение основных блоков.

Порты ввода/ввода, последовательная и параллельная организация.

Принцип работы устройств ввода/ввода.

Группы устройств ввода/ввода: устройства интерфейса пользователя, устройства ввода/вывода для длительного хранения информации, таймерные устройства. Предназначение, принцип работы данных устройств.

Лекция №8 «Функционирование МПС. Адресация и ее особенности, регистры»

1. Основы программного режима работы микропроцессорной системы: принцип программного режима, программы, операнды (входные и выходные), код команды, понятие методов адресации.

2. Методы адресации:

Непосредственная адресация, прямая адресация, регистровая адресация, косвенно-регистровая, относительная адресация – принцип организации.

Особенности адресации: автоинкрементирование (декрементирование), неявная адресация, использование различных методов адресации.

3. Сегментное разбитие памяти: особенности представления и использования, формирование физического адреса, смещение, адресация в защищенном режиме, таблица дескрипторов, особенности адресации для различных типов микропроцессоров, эффективный, линейный адрес.

4. Особенности адресации данных.

5. Регистры микропроцессора:

Понятие универсальных и специализированных регистров, их виды, особенности использования регистров в некоторых микропроцессорах.

Характеристика, предназначение и использование регистров на примере простейшего микропроцессора (например, Intel 8086), регистры данных, сегментные регистры, указатели, регистр флагов. Особенности использование аккумулятора.

Лекция №9 «Программные основы работы МП

Основные группы команд в микропроцессоре, предназначение, их операнды, различные подвиды: команды пересылки данных, арифметические команды, логические команды, команды переходов.

1. Команды пересылки данных, выполняемые функции, примеры команд на языке Ассемблера, особенности применения и использования на некоторых процессорах.

2. Арифметические команды, их группы (команды операций с фиксированной запятой, команды операций с плавающей запятой, команды очистки, команды инкремента и декремента, команда сравнения) – выполняемые функции, примеры команд на языке Ассемблера, особенности применения и использования на некоторых процессорах.

3. Логические команды, их группы (логическое И, логическое ИЛИ, сложение по модулю 2; логические, арифметические и циклические сдвиги; проверка битов и операндов; установка и очистка битов регистра состояния процессора) – выполняемые функции, примеры команд на языке Ассемблера, особенности применения и использования на некоторых процессорах

4. Команды переходов, их группы (команды условных и безусловных переходов) – выполняемые функции, примеры команд на языке Ассемблера, особенности применения и использования на некоторых процессорах.

Лекция №10 «Микроконтроллеры. Основы организации I»

Понятие микроконтроллеров, основные элементы.

1. Структура микроконтроллеров:

Классы микроконтроллеров (8, 16, 32 –х разрядные, сигнальные процессоры DSP), производители современных микроконтроллеров.

Особенности микроконтроллеров (модульная организация, закрытая архитектура, типовые и расширенные функциональные периферийные модули).

Типовая структура микроконтроллера. Процессорное ядро и изменяемый функциональный блок. Предназначение основных элементов.

2. Процессорное ядро, его характеристики. Процессоры с CISC-архитектурой, RISC-архитектурой – особенности, отличия, сравнение. Особенности организации памяти в микроконтроллерах: структуры с фон-неймановская (принстонская) и гарвардской архитектурой – особенности, отличия, сравнение.

Лекция №11 «Микроконтроллеры. Основы организации II»

1. Система команд микроконтроллеров, ее особенности.

2. Схема синхронизации и организации памяти микроконтроллеров. Особенности данных схем. Память программ, типы модулей памяти и их особенности. Память данных. Особенности хранения данных и программ. Регистровая и стековая память – предназначение и особенности. Внешняя память.

Лекция №12 «Внутренние и внешние связи в микроконтроллерах I»

Порты ввода/вывода:

Параллельные и последовательные порты, типы портов, их предназначение, алгоритмы работы. Типичная схема двунаправленного порта ввода/вывода микроконтроллера.

Лекция №13 «Внутренние и внешние связи в микроконтроллерах II»

1. Таймеры и процессоры событий:

Предназначение, структура типичного 16-разрядного таймера/счетчика, основные недостатки данной схемы, пути усовершенствования данной схемы и современные направления.

Принцип действия канала входного захвата таймера/счетчика, его схема, типы сигналов, функциональная схема.

Структура аппаратных средств канала выходного сравнения – основные сигналы, схема, принцип работы, аппаратные и программные усовершенствования.

Модули процессоров событий – предназначение, принцип работы, основные сигналы, реализация режима широтно-импульсной модуляции.

2. Модуль прерываний: принцип работы, источники прерываний, схема приоритетов.

Лекция №14 «Аппаратные средства микроконтроллеров I»

1. Особенности режимов энергопотребления, минимизация данного режима: активный режим, режим ожидания, режим останова – особенности, предназначение.

2. Тактовые генераторы микроконтроллеров:

Определения тактовой частоты генератора: с помощью кварцевого резонатора, керамического резонатора и внешней RC-цепи. Схемы подключения тактовых генераторов, используемые входы. Сравнительная характеристика каждого способа подключения.

3. Аппаратные средства обеспечения надежной работы микроконтроллера:

Схема формирования сигнала сброса: предназначение, основные сигналы, принцип работы, типовые схемы формирования сигнала внешнего сброса.

Блок детектирования пониженного напряжения питания: предназначение, особенности применения, принцип работы.

Лекция №15 «Аппаратные средства микроконтроллеров II»

1. Сторожевой таймер: принцип действия, основные используемые сигналы, предназначение, особенности работы.

2. Дополнительные модули, используемые в микроконтроллерах:

Модули последовательного и параллельного вода/вывода: задачи решаемые данными модулями, их типы, основы функционирования, протоколы интерфейса, современное состояние проблемы передачи информации через порты ввода/вывода.

Модули аналогового ввода/вывода: основные схемы, принцип работы, схема типового модуля АЦП, основы работы ЦАП и средства реализации данной функции в современных микроконтроллерах.

Лекция №16 «Проектирование устройств на микроконтроллерах»

1.Основные этапы разработки:

Составление технического задания и функциональной спецификации, их особенности.

Этап разработки алгоритма управления, формулирование требований к аппаратной и программной части.

Выбор типа ядра контроллера, условия, виляющие на данный выбор.

Этап разработки структуры аппаратной части, перераспределения функций между аппаратными и программными средствами. Возможные модификации технического задания и функциональной спецификации, этапы доработки.

2. Разработка и отладка аппаратных средств: разработка общей принципиальной схемы, разводка топологии плат, монтаж макета, автономная отладка, системы проектирования.

3. Разработка и отладка программного обеспечения: интегрированные средства разработки и отладки, их состав предназначение, особенности.

4. Методы и средства совместной отладки аппаратных и программных средств: отладка в реальном масштабе времени, внутрисхемные эмуляторы, платы развития, мониторы отладки, эмуляторы

5. Заключительный этап проектирование цифровых систем на основе микроконтроллеров.

СЕМЕСТР 8

Лекция №1 «Введение: микроконтроллеры серии PIC и AVR»

1. Общие сведенья о микроконтроллерах серии PIC и AVR

Состав и назначения семейств, особенности памяти, энергопотребления, производительности, архитектуры и размеров. История появления и развития. Особенности центрального процессора. Отличительные особенности семейств микроконтроллеров. Общие сведенья о сопутствующих программных продуктах.

Технические характеристики отдельных подгрупп семейства, объем памяти, количество команд, тактовая частота, интерфейс ввода-вывода. Представление микроконтроллера с программной точки зрения.

2. Архитектура и характеристики базовых моделей отдельных подгрупп: максимальная частота; Flash-память программ; ПЗУ программ; память данных; память данных в РПЗУ (EEPROM); таймеры, число источников прерываний; число линий ввода-вывода, диапазон напряжения питания; число выводов и тип корпуса, специализированные микроконтроллерные функции. Структурная схема базовых моделей отдельных подгрупп. Назначение выводов корпусов.

Лекция №2 «Принципы работы, организация памяти и особенности выполнения команд для микроконтроллеров PIC и AVR»

1. Принцип работы: временные диаграммы тактирования и циклов выполнения программы, схема тактирования и выполнения команды, принцип выборки команды.

2. Организация памяти.

Организация памяти программ и стека, счетчик команд, адресное пространство, физические ограничения адресации, схема организации памяти и программ стека, вектор сброса, подпрограмма идентификации и обработки прерываний.

Организация памяти данных, области памяти, особенности адресации. Особые регистры, регистры специального назначения, назначение бит специальных регистров

3. Особенности выполнения команд.

Выборка команд из памяти, выполнение команд условного и безусловного переходов, аппаратный стек.

Особенности методов адресации: прямая и косвенная адресация в микроконтроллерах PIC и AVR.

Лекция №3 «Организация обмена с внешними устройствами, память, прерывания для микроконтроллеров PIC и AVR»

1. Порты ввода-вывода, назначение, специфика, принцип организации, схемы портов и отдельных линий, особенности практического использования (программирование и электрическое соединение).

2. Таймеры. Структурная схема таймера(счетчика) для микроконтроллеров PIC и AVR, состав, предназначение, принцип работы основных элементов схемы, реакция на прерывания, особенности программного и аппаратного использования, структурные схемы возможных вариантов использования.

3. Память данных, запись и чтение из памяти данных, используемые регистры и назначение отдельных битов в них. Примеры программного кода, поясняющие принцип работы с памятью данных.

4. Организация прерываний, возможные виды прерываний, особенности, схема логики прерываний микроконтроллера, прерывания отдельных устройств.

Лекция №4 «Специальные функции и система команд микроконтроллеров PIC и AVR»

1. Набор специальных функций расширяющих возможности системы: сброс, сторожевой таймер, режим пониженного энергопотребления, выбор типа генератора, защита от кода считываний, биты идентификации, последовательное программирование. Характеристика отдельных «особых режимов». Биты конфигурации.

2. Система команд.

Перечень и формат команд отдельных моделей микроконтроллеров PIC и AVR. Описание полей команд, основные форматы команд. Перечень команд, их описание, количество циклов за которые они исполняются, изменяемые биты состояния.

Команды работы с байтами, преимущества, особенности использования, примеры применения.

Команды работы с битами, преимущества, особенности использования, примеры применения.

Команды управления и работы с константами, преимущества, особенности использования, примеры применения.

Лекция №5 «Особенности программирования и отладки, разработка программного кода для микроконтроллеров PIC и AVR»

1. Особенности программирования и отладки.

Отдельные замечания по особенностям программирования: особенности загрузки констант, арифметическо-логических операций, конвейер команд, инструкции для организации ветвлений, стек, память программ и данных, ограниченность ресурсов.

2. Разработка программного кода.

Различные ассемблеры для разработки программного кода, цели использования, принципы применения.

Компоновщики объектного кода, опции использования, режимы работы.

Основной текст программы на ассемблере, метки, мнемоники, операнды, формат представления чисел, основные арифметические операторы, комментарии.

Расширения файлов используемых при создании программного кода. Листинг.
Директивы языка ассемблер, синтаксис при написании программного кода, поясняющие примеры.

Лекция №6 «Разработка программного обеспечения для микроконтроллеров PIC и AVR»

1. Особенности компоновщиков.
2. Особенности менеджеров библиотек.
3. Особенности симуляторов, примеры использования.
4. Завершенные программные пакеты для создания, и исследования программного кода: примеры, принцип работы, использование, преимущества и недостатки, заключительные замечания.

Лекция №7 «Макет микропроцессорной системы и программирование простейших задач для микроконтроллеров PIC и AVR»

1. Описание макета, электрическая схема соединения, параметры основных элементов, предназначение и выполняемые функции
2. Особенности инициализации и запуска в работы
3. Примеры завершенных программ:

Программа считывания состояния кнопки и вывода на светодиодный индикатор (текст программы, комментарии к ней, описание принципа работы).
Программа считывания состояния кнопки и вывода на светодиодный индикатор при определенных условиях (текст программы, комментарии к ней, описание принципа работы).
Программа для работы с семисегментным индикатором и кнопками (текст программы, комментарии к ней, описание принципа работы).
Программа для работы вывода на семисегментный индикатор числа (текст программы, комментарии к ней, описание принципа работы).
Подпрограммы формирования задержки (текст программы, комментарии к ней, описание принципа работы).
Программа для работы с звуковым динамиком (текст программы, комментарии к ней, описание принципа работы).
Программа для работы с мигающим светодиодом (текст программы, комментарии к ней, описание принципа работы).
Программа для борьбы с дребезгом контактов (текст программы, комментарии к ней, описание принципа работы).

СЕМЕСТР 9

Лекция №1 «Системы малой автоматизации на x-51 совместимых микроконтроллерах»

1. Предпосылки создания систем малой автоматизации

История развития, направления развития, основные представители направления, понятие малых систем автоматизации.

2. Распределенные системы управления в системах малой автоматизации.

Микроконтроллеры с классической архитектурой, коммутирующие и логические возможности, автономный подход к решению сложных задач управления, стоимость и габаритные размеры, децентрализация задач управления в микропроцессорных системах.

3. Локальные вычислительные сети.

История развития, основные стандарты, классификация локальных сетей – современный подход, режим реального времени, форматы фреймов.

4. Основные понятия и определения систем малой автоматизации.

Определение систем малой автоматизации, факторы способствующие развитию систем малой автоматизации, сетевые технологии, альтернативный подход к построению микропроцессорных систем управления на x-51 совместимых микроконтроллеров, магистрально-модульные системы и типовые интерфейсы.

Лекция №2 «Командные и информационные сети на x-51 совместимых микроконтроллерах»

1. Основные понятия

Типовая структура, алгоритм работы и основные параметры, базовые интерфейсы *RS232C* и *RS485*, основные диспетчеры, способы передачи и приема информации, примеры микросхем для работы в сетевом режиме, основные элементы схем, их технические характеристики и примеры. Способы управления доступом к каналам связи, диспетчер станции, активный и пассивный доступ. Определение скорости передачи информации.

2. Диспетчеры периферийных станций: CI-LAN «SISNET», топология сети, структурная схема диспетчера, назначение основных линий и элементов; CI-LAN «MISNET», структурная схема, назначение основных элементов, принцип работы, комбинированный диспетчер.

3. Формат фреймов и общий алгоритм работы CI-LAN: адресация, принцип организации, формат вопроса и ответа, минимизация формата фреймов. Тестовая программа контроллера станций.

Лекция №3 «Организация универсальных технологических x-51 контроллеров: введение»

1. Предпосылки создания универсальных технологических контроллеров: аспекты архитектуры, затраты на производство и эксплуатацию, различная номенклатура, и т.п.

2. Основные понятия и тенденции развития универсальных технологических контроллеров: аспекты децентрализации, аспекты сетевого режима работы, слотовая технология, интеллектуальный и магистрально-модульный подход, многоконтроллерные системы, рабочие станции, основной состав систем.

3. Общие технологические требования к главному микроконтроллерному модулю: ядро, питание и сторожевые таймеры, оперативная память, таймеры реального времени, АЦП и ЦАП, сверхоперативная память, порты ввода-вывода, интерфейсы, в том числе и сетевые, питание и отдельные режимы работы.

4. Обобщенная функциональная схема центрального модуля (элементы и их взаимосвязь, основные сигналы).

Лекция №4 «Организация универсальных технологических х-51 контроллеров: супервизоры питания и охранные таймеры»

1. Основные понятия о супервизорах питания и таймерах.

2. Критерии выбора супервизора питания (выработка различных сигналов, компараторы предупреждений, возможность программирования, резервное питание, цветовая и звуковая сигнализации, дополнительные выходы, корпусы исполнения).

3. Предварительный анализ и выбор микросхем супервизоров (микросхемы различных производителей, примеры, характеристики и параметры, использующиеся сигналы).

4. Схемы включения микросхем супервизоров.

5. Заключительный этап выбора микросхем супервизоров (сравнительный анализ, формализация этапов выбора, окончательные выводы).

6. Функциональные характеристики таймеров реального времени (фирмы производителя, форматы записи данных, генерация различных кодов, программирование времени, встроенные супервизоры питания и сторожевые таймеры, внешние выводы, энергонезависимая память, подсчет времени между событиями, корпусы и варианты исполнения, доступ к данным и используемые шины).

7. Критерии выбора таймеров реального времени (выделение необходимых преимущественных характеристик для решения задач малой автоматизации).

8. Анализ и выбор микросхем таймеров реального времени (примеры некоторых микросхем).

9. Микросхемы дополнительной памяти для технологических контроллеров (виды, предназначение, характеристики).

10. Схемы включения таймеров реального времени (способ подключения, примеры).

Лекция №5 «Организация универсальных технологических х-51 контроллеров: устройства ввода-вывода, расширения, ЦАП и АЦП»

1. Устройства ввода-вывода и расширения

Последовательные интерфейсы (принципиальная схема конкретного комбинированного диспетчера, основные элементы, параметры, способ работы).
Регистры ввода-вывода (примеры, принцип работы, схемы).
Модификаторы адреса (предназначение, примеры, принцип работы).
Оптически развязанные узлы (основные параметры, принцип работы и предназначение, примеры).
Символьные жидкокристаллические индикаторы (основные параметры, принцип работы и предназначение, примеры).

2. Аналого-цифровые преобразователи.

Критерии выбора АЦП и различные микросхемы (фирмы производители, основные продукты, важнейшие параметры - тип интерфейса, быстродействие, тип корпуса, характеристики питания, примеры).
Вспомогательные микросхемы узлов АЦП (операционные усилители, мультиплексоры, и т.п. – примеры, принцип работы и основные параметры).
Принципиальная схема АЦП на различных составных элементах (принцип работы, состав, характеристика).

3. Цифро-аналоговые преобразователи.

Критерии выбора ЦАП и различные микросхемы (фирмы производители, основные продукты, важнейшие параметры - тип интерфейса, быстродействие, тип корпуса, характеристики питания, тип выхода, примеры).
Отбор микросхем ЦАП и их отличия (примеры различных микросхем).
Параллельные ЦАП (преимущества, недостатки, и различные параметры).
Последовательные ЦАП (преимущества, недостатки, и различные параметры).
Принципиальные схемы подсистем ЦАП (пример, основные элементы и сигналы).

Лекция №6 «Проектирование систем малой автоматизации на основе x-51 совместимых микроконтроллеров»

1. Основные критерии выбора конкретной модели микроконтроллера (сопоставительный анализ моделей x-51 и других).

2. Разработки фирмы Atmel (примеры, основные параметры, отличительные особенности).

3. Разработки фирмы MAXIM (примеры, основные параметры, отличительные особенности).

4. Разработки фирмы Cygnal (примеры, основные параметры, отличительные особенности).

5. Отдельные вопросы обеспечения пиковой производительности микроконтроллеров.

Лекция №7 «Завершенные примеры макетов микропроцессорных систем управления на базе x-51 совместимых микроконтроллеров»

1. Принципиальная схема основных узлов: интерфейс *RS232C* на микросхеме *MAX202/232* или *ADM202*; интерфейс *RS485* на микросхеме *MAX485/487* или *ADM485;* узел управления интерфейсами: *74HC32* и

74HC03; супервизор питания с охранным таймером *WDT* на микросхеме *TL7705;* микроконтроллер*;* таймер реального времени на микросхеме *DS12(C)887(A);* регистры вывода *74HC573;* управляемый канал ввода/вывода на микросхеме *74HC24*. Соединение линий ввода-вывода.

2. Подсистема аналогового ввода-вывода (на основе восьмиканального 12-разрядного аналого-цифрового преобразователя *MAX1270* и двух микросхем восьмиканальных цифро-аналоговых преобразователей 8-разрядного *LTC1665* и 10-разрядного *LTC1660*): схема соединения, основные элементы и их параметры, принцип работы.

3. Подсистема интерфейсов (содержащая супервизор питания с функциями охранного таймера *WDT* на микросхеме *TL7705*, входной регистр модификатора *74HC245,* дешифратор сигналов чтения/записи памяти и узлов ввода/вывода на микросхеме дешифратора *74ИС138,* микроконтроллер, микросхему оперативной памяти *61C256*, микросхемы шинного формирователя *74HC245* и регистра защелки младшего байта адреса *74HC573,* микросхему таймера реального времени со встроенной литиевой батарейкой и энергонезависимой оперативной памятью *DS1644,* логику выборки старшего адреса *74HC00,* выходной регистр управления *74HC374,* логику выбора адресов регистров на микросхемах *74HC30* и *74HC32,* два выходных изолированных оптически сигнала на оптронах *4N35,* двухцветный индикатор состояния на светодиоде, модификатор сетевого адреса, системный разъем, разъем для подключения жидкокристаллического индикатора, переменный резистор для выбора и установки напряжения контрастности): схема соединения, основные элементы и их параметры, принцип работы.

4. Универсальный технологический контроллер на базе микроконтроллера *C8051F020.* Состав, схема, принцип работы, основные элементы и их параметры.

5. Варианты слотового исполнения системы сбора и обработки данных (схема, печатная плата, основные элементы).

6. Специализированные контроллеры-фотодатчики (принципиальная схема, примеры программирования).

7. Специализированный технологический контроллер для работы в составе информационной сети (принципиальная схема, основные элементы, принцип работы, разработка отдельных программных модулей - работа в режиме реального времени, режим обслуживания последовательных портов UART, главный программный модуль).

6. ПЕРЕЧЕНЬ ПРОГРАММНЫХ ПРОДУКТОВ И МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ПРИМЕНЕНИЮ СОВРЕМЕННЫХ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ ДЛЯ ПРЕПОДАВАНИЯ УЧЕБНОЙ ДИСЦИПЛИНЫ

Для преподавания дисциплины «Микропроцессорные системы управления» рекомендуется использовать два программных продукта: «MPLAB IDE v. 7.42», компании Microchip, «Proteus v. 7.2», компании Labcenter Electronics. Первый программный продукт поставляется компанией-разработчиком микроконтроллеров PIC и является универсальным средством создания программного обеспечения для микроконтроллеров этого типа. Основы работы с программой описываются на лекционных занятиях, интуитивно понятны и далее приводится, не будут. Весь представленный ниже материал касается, прежде всего, универсальной программы-симулятора микропроцессорных устройств , «Proteus» [1]

6.1 ОБЩЕЕ ОПИСАНИЕ

«Proteus» - программа-симулятор микропроцессорных устройств. Поддерживает микроконтроллеры следующих марок: PIC, 8051, AVR, HC11, ARM7/LPC2000 и другие. Более 6000 аналоговых и цифровых моделей устройств. Работает с большинством компилятором и ассемблерами. «Proteus» позволяет очень достоверно моделировать и отлаживать достаточно сложные устройства, в которых может содержаться несколько микроконтроллеров одновременно и даже разных семейств в одном устройстве.

«Proteus» содержит огромную библиотеку электронных компонентов. Отсутствующие модели можно сделать самостоятельно. Если компонент не программируемый, то на сайте производителя возможно скачать его модель и добавить в подходящий корпус. Помимо базовых возможностей «Proteus» имеет дополнительные преимущества в виде наличия следующих инструментов: USBCONN - этот инструмент позволяет подключиться к реальному USB порту компьютера. COMPIM - этот компонент позволяет вашему виртуальному устройству подключится к реальному COM-порту вашего компьютера.

«Proteus» - великолепно работает с популярными компиляторами Си для микроконтроллеров: CodeVisionAVR (для AVR), IAR (для любых МК), ICC (для МК AVR, msp430, ARM7, Motorola), WinAVR (для МК AVR), Keil (для МК архитектуры 8051 и ARM), HiTECH (для МК архитектуры 8051 и PIC от Microchip).

[1] Сайт http://www.eldigi.ru

6.2 ПРИНЦИП РАБОТЫ

Начало работы с программной начинается с запуска модуля ISIS, где создается новый проект, в котором путем выбора элементов из библиотеки формируется перечень используемых элементов DEVACE (см. рис. 6.1). В последствие данные элементы размещаются в рабочей области программы, создаются эклектические соединения элементов. Для микроконтроллера в окне его свойств возможно указание откомпилированной рабочей программы, которая будет исполняться в дальнейшем.

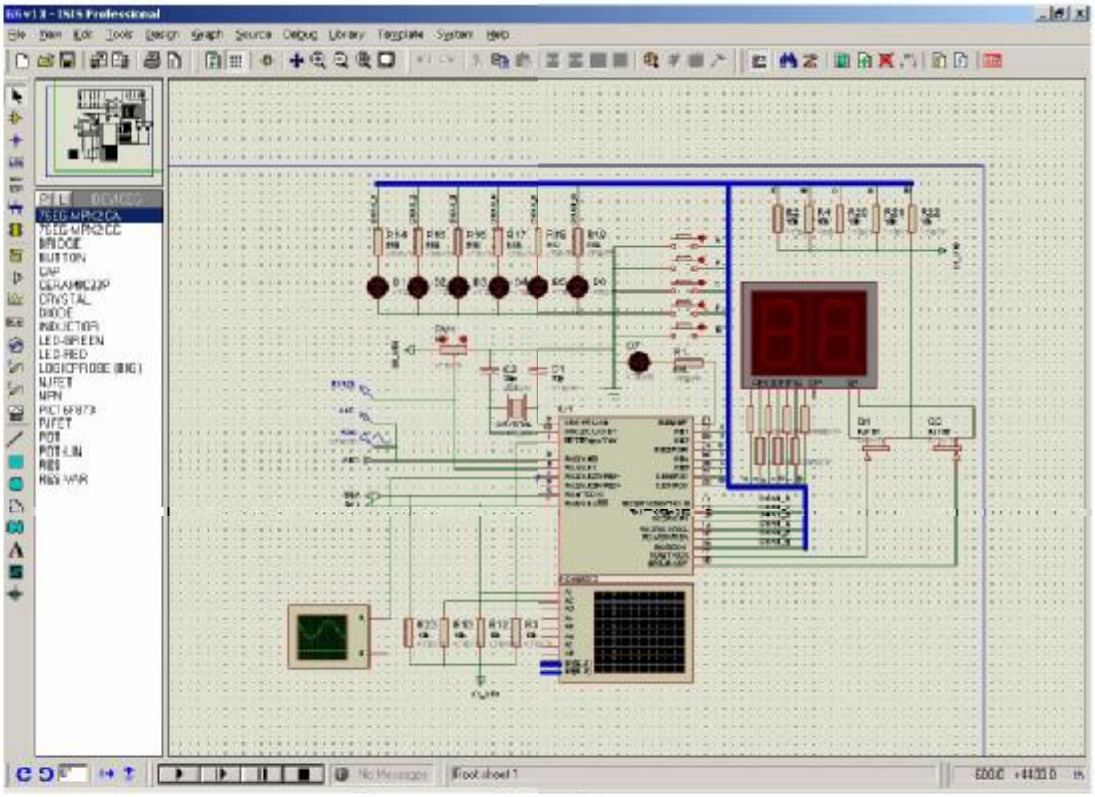


Рисунок 6.1 – Созданная схема

Для дальнейшего исследования схемы проект запускается в режим моделирования. При этом возможно проведение манипуляций с переключением различных элементов, отслеживание состояния на выходах и входах микроконтроллера, а так же исполнение записанной в микроконтроллер программы.

Для создания печатной платы спроектированного и исследованного устройства необходимо каждому элементу схемы сопоставить какой-то корпус. Это параметр – PCB Package (см. рис. 6.2). Тип корпуса можно изменить, нажав на [?]. Для резисторов и керамических конденсаторов он обычно называется RES40 или C20, где число – это расстояние между выводами. 10th = 2,5мм (тогда RES40 => 10мм между выводами). Некоторым элементам корпус изначально не сопоставлен (светодиоды), поэтому придется это сделать самостоятельно (см. рис. 6.3).

Рисунок 6.2 – Задание типа корпуса

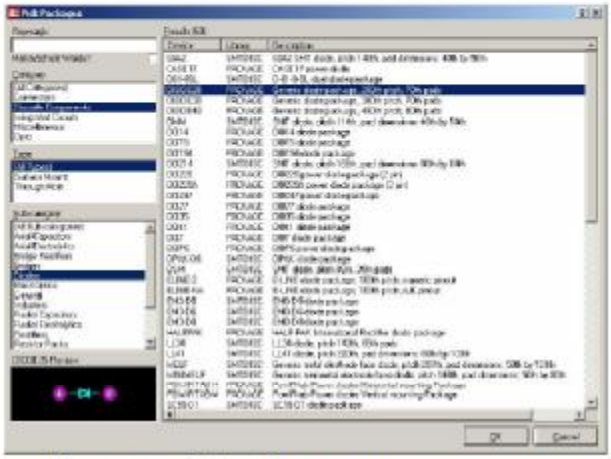


Рисунок 6.3 – Различные корпуса

У некоторых элементов пока невозможно сопоставить корпус, это, например светодиодные индикаторы, катушки индуктивности, кнопки. Если корпус есть в стандартной библиотеке, можно сопоставить при экспортировании в ARES (этот способ подходит, например, для потенциометра) – модуль для проектирования печатных плат. Если же в стандартных библиотеках нет подходящего корпуса, то возможно взять нужные корпуса из нестандартных библиотек. Однако рекомендуется добавить в схему дополнительные разъемы и соединить с выводами этих элементов, а потом в ARES расставлять не сами элементы, а эти разъемы, соответственно цоколевке. Например, как показано на рис. 6.4 для индикатора:

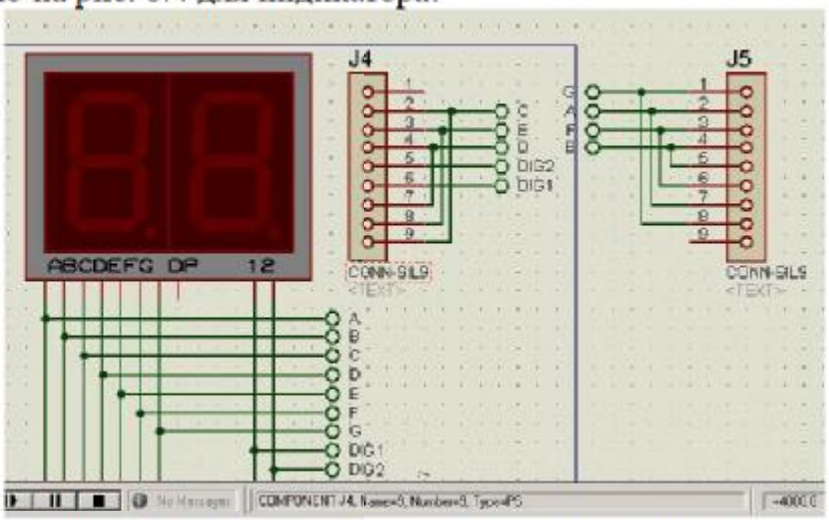


Рисунок 6.4 – Задание разъемов для индикатора

Полученная в итоге преобразованная схема (рис. 6.1) приобретет вид (рис. 6.5).

Далее при нажатии на кнопку [ARES] вызывается соответствующий модуль и в случае если каким-то элементом не был сопоставлен корпус (на рис. 6.5) это все кнопки, катушки, индикатор и потенциометр), то появится окно с предложением сделать это сейчас (рис. 6.6).

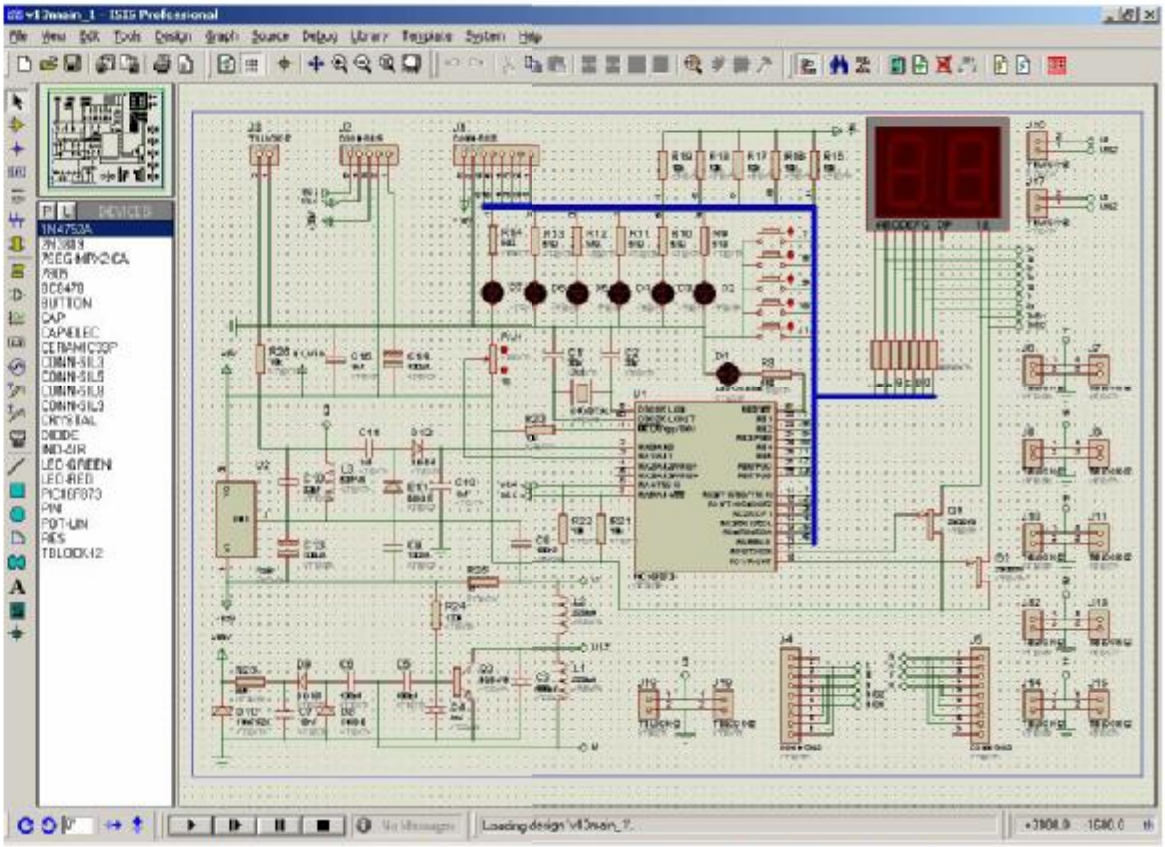


Рисунок 6.5 – Схема для передачи в модуль ARIES

Рисунок 6.6 – Окно с предложением сопоставления корпусов

В этом случае необходимо пропустить все элементы, исключая потенциометр (для предложенного примера).

Далее нужно ограничить размер печатной платы. Это делается на слое Board Edge с помощью инструментов 2d-графики (наиболее удобен прямоугольник) – см. рис. 6.7.

Элементы на плате можно расставить вручную или автоматически. После этого возможно проведение автотрассировки, кнопка [auto-placer]. Далее появляется запрос вида рис. 6.8. Здесь выбираются элементы, которые надо расставить, шаг сетки, предпочтительное расположение элементов, степень группирования. Для настройки стратегии трассировки необходимо зайти в меню *system* → *set strategies,* см. рис. 6.9.

После настройки стратегии необходимо нажать кнопку [autorouter], и выставить шаг сетки, полученная в итоге схема будет иметь вид – см. рис. 6.10.

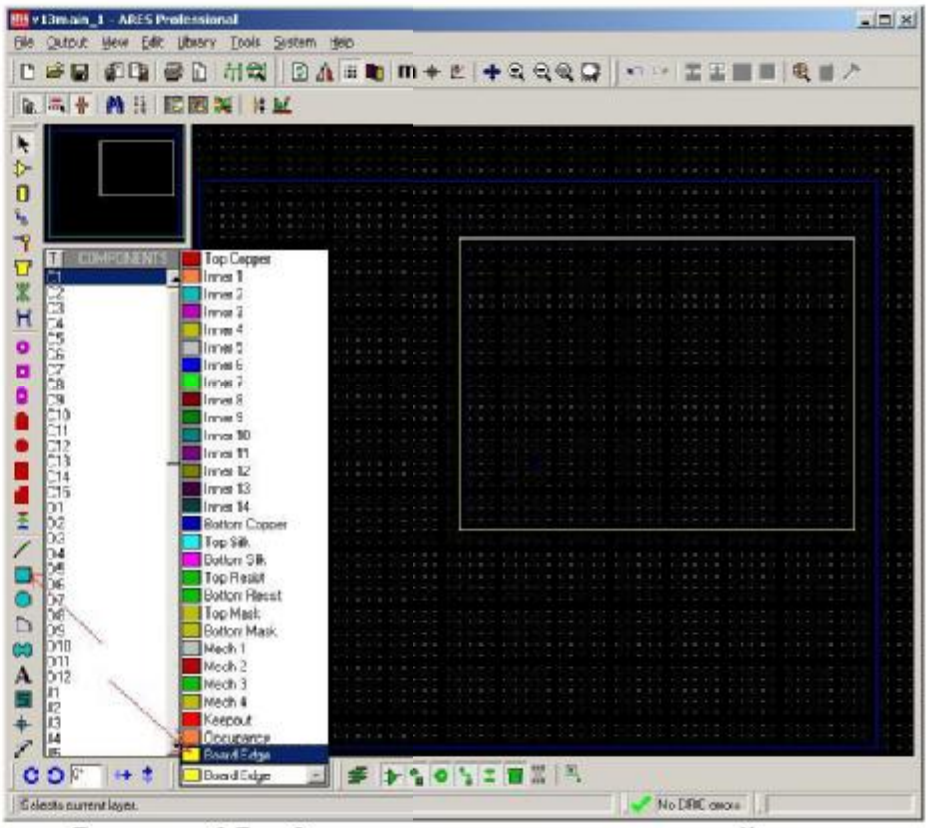


Рисунок 6.7 – Ограничение размера печатной платы

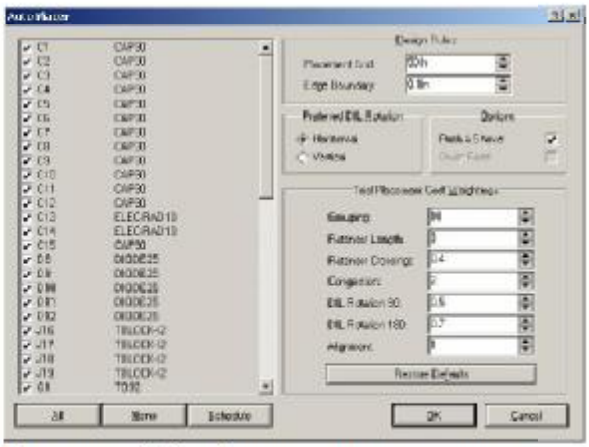


Рисунок 6.8 – Запрос на расстановку элементов

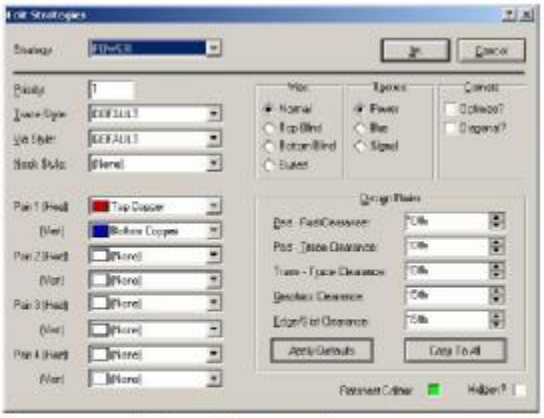


Рисунок 6.9 – Настройка стратегии

Если ARES не смог развести дорожки, то необходимо переставить элементы и по новой. Если же ARES развел плату, но выдал предупреждение об ошибках, то это значит, что расстояние между площадками и дорожками где-то меньше допустимого. В этом случае необходимо оценить степень критичности.

Полученный в итоге вариант печатной платы можно распечатать: *Output → Set Output Aria*, выделив площадь печати, и непосредственно отправив на печать *Output → Print*. При этом возможно задание сторон распечатки (зеркальное отображение).

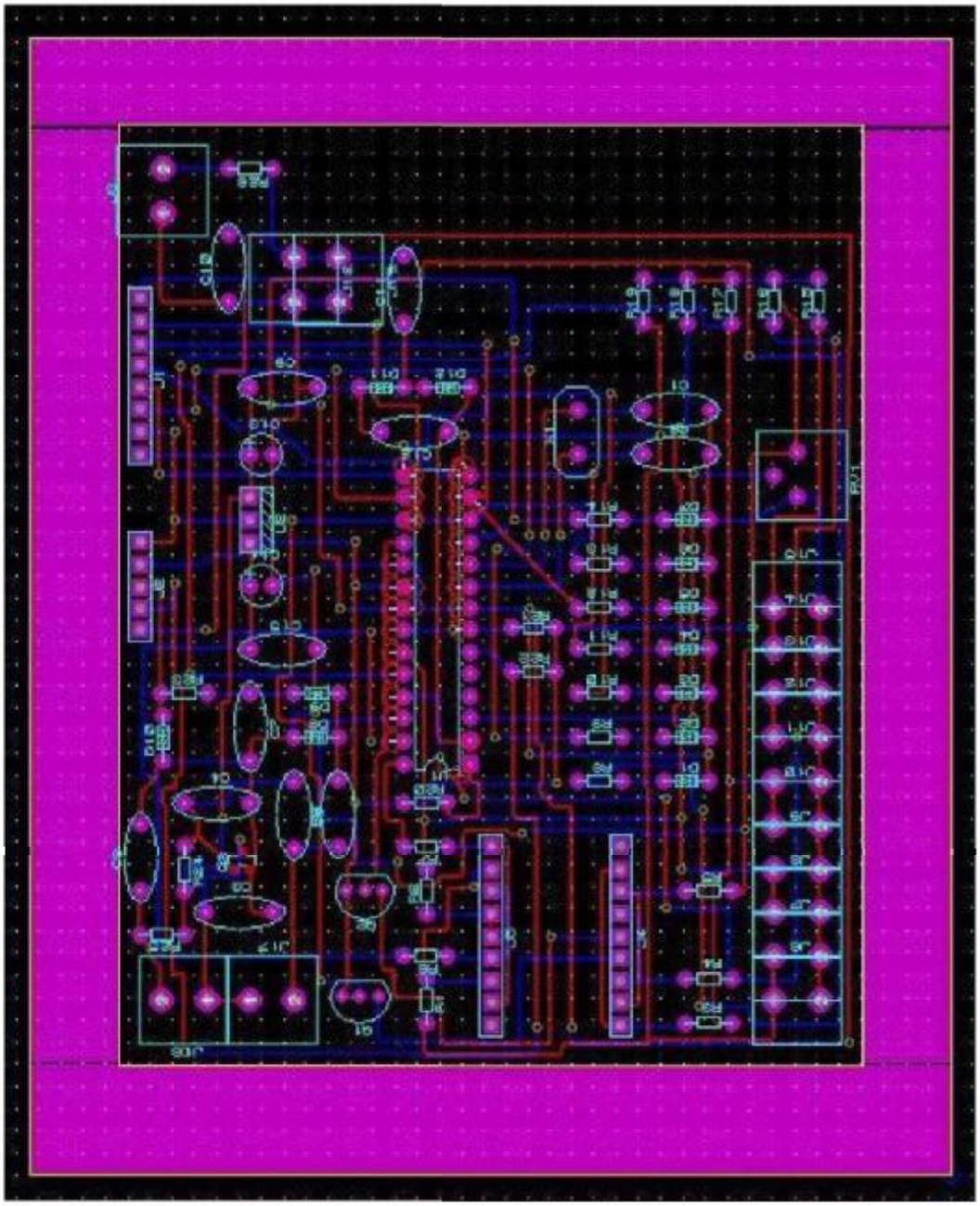


Рисунок 6.10 – Вид печатной платы

7. МЕТОДИЧЕСКИЕ УКАЗАНИЯ

7.1 САМОСТОЯТЕЛЬНАЯ РАБОТА И КУРСОВОЕ ПРОЕКТИРОВАНИЕ

Самостоятельная работа студентов проводится согласно рекомендациям – см. п.3.1 и в рамках графика – см. п.2. Ниже представлены методические указания касающиеся курсового проектирования.

ТЕМА: Разработка завершенного цифрового устройства на базе однокристального микроконтроллера с RISC-архитектурой.

1. *Общие принципы организации работы*

Задания для курсового проекта представляют большую возможность студентам самостоятельно проявить навыки по созданию завершенных устройств на базе микроконтроллеров.

Несмотря на то, что материал лекционного курса, в части касающейся детального рассмотрения работы микроконтроллеров базируется на серии PIC (AVR), а лабораторный и практический курс основывался на микропроцессорах КР580 и микроконтроллеров Siemens и Mega – решение поставленных задач можно осуществить на любых микроконтроллерах с RISC архитектурой. При этом, конечно же, выбор используемых производителей и серий всей аппаратной и программной части должен быть детально обоснован. Не уменьшая общности представленного материала, необходимо отметить, что в качестве базовой серии микроконтроллеров здесь и далее рассматривается PIC(AVR)XXX.

В любом случае поощряются разработки, отличные от рекомендаций предложенных здесь, если они приводят к повышению объема и качества выполняемых работ.

Особое внимание стоит обратить на несколько моментов.

Во-первых: Лекционного курса и подобранного материала достаточно для выполнения работы в полном и достаточном объеме самостоятельно каждым студентом.

Во-вторых: Студентам необходимо жестко выполнять указанные временные рамки своей работы (см. график сдачи). Данные временные рамки составлены с учетом всех возможных трудностей при выполнении работы, а так же базируются на том, что разработку курсового проекта невозможно проводить без необходимого лекционного материала.

В-третьих: График сдачи заданий непрерывно модифицируется – в него вносится конкретное время, когда каждый студент выполнил указанную работу (данный график доступен в режиме on-line на сайте кафедры).

В-четвертых: Необходимо следить за возможным изменением представленного материала – доступного в режиме on-line.

2. *Содержание текстовой, расчетной и графической части*

2.1. *Содержание тестовой и расчетной части*

Ниже представлена общая структура пояснительной записки к курсовому проекту. Курсивом выделены аспекты, которые необходимо раскрыть в данных разделах.

Реферат

Выполняется согласно требованиям к курсовому проектированию Амурского государственного университета.

Введение

Дается краткая характеристика микроконтроллеров с RISC архитектурой, преимущества данной архитектуры, область применения микроконтроллеров, и т.п.

1. Разработка и характеристика основных решений

1.1. Постановка и описание задачи разработки

Приводится полное словесное описание разрабатываемой проблемы. Данный раздел должен показать насколько глубоко понимается исходная задача.

1.2. Разработка исходной структурной схемы устройства

Данная структурная схема используется в дальнейшем (детализируется на каждом уровне разработки). В общем случае представляет собой структурное представление задач, которые необходимо решить для создания устройства и необходимых для этого действиях (касается как аппаратной части, так и программной).

1.3. Технические характеристики

Приводится необходимые или обязательные параметры, которым должно удовлетворять устройство; анализируется, каким образом это может повлиять на выбор решений и на каком этапе разработки должно учитываться.

1.4. Пути решения поставленной задачи

Описывается не только предлагаемый и развиваемый в работе подход, но и возможные альтернативы решений (касается как аппаратной, так и программной части), обосновывается выбор основных частей (микроконтроллера).

1.5. Техническое задание на разработку

Техническое задание разрабатывается согласно требованиям ГОСТ-19.2001-78 «Техническое задание. Требование к содержанию и оформлению» (данный ГОСТ представлен в литературе, см. ниже). Сам текст Технического задания приводиться в Приложении А. В данном пункте описываются и обосновываются пункты технического задания – а именно почему был выбран такой путь.

2. Разработка и анализ структуры устройства

2.1. Принципиальный алгоритм реализуемой программы

В данном пункте необходимо проанализировать полученную структуру в 1.2 и 1.3. А именно, в части касающейся программной реализации – описать принципиальный алгоритм работы реализуемой программы и те технические характеристики (производительность, количество портов, наличие АЦП и ЦАП и т.п.), необходимые для выбора аппаратной части.

2.2. Разработка детальной структурно-аппаратной схемы устройства и ее функциональных подблоков.

Здесь на уровне каждого элемента (контроллера, индикатора, резистора и т.п) представляется общая схема устройства с точки зрения наличия основных компонентов и их взаимосвязи.

2.3 Технические характеристики и описание основных аппаратных частей.

Здесь приводятся только основные элементы и их характеристики, которые выбираются без расчета их параметров.

3. Принципиальная схема устройства

Здесь приводится разрабатываемая схема уже со связанными между собой компонентами, объясняются принципы аппаратной реализации каждой функциональной связи, приводиться расчет и обоснование выбора параметров всех электрических элементов. В обязательном порядке снабжается расчетом мощности, потребляемой схемой и расчет временных параметров работы схемы. Т.е. описывается разработка которая в дальнейшем может быть смоделирована – например в программе «Proteus».

4. Программная реализация

4.1. Разработка полного алгоритма программы

Здесь полученный в 2.1. принципиальный алгоритм представляется в виде полного алгоритма работы программы, вычерчивается алгоритмическая схема. Принципиальный алгоритм должен содержать наименование используемых регистров, анализируемых битов и т.п.

4.2. Исходный текст программы

Здесь приводится в исходном текстовом коде (ассемблере) сам текст программы, и даются детальные комментарии по ее выполнению.

4.3. Компиляция программы

Описывается общий принцип компиляции и создание завершенного программного продукта. Приводится так же вариант создания перемещаемого объектного кода (вариант при котором создается отдельный модуль, для дальнейшего использования в других программах). Приводятся полученные в ходе компиляции листинги, и тому подобные файлы.

4.4. Отладка программы

Описывается принцип симуляции программы, осуществляемой с помощью модуля MPSIM, ее результат, делаются выводы.

5. Расчет и создание монтажной схемы

Описываются принципы трассировки и требования к размещению элементов. Автоматическая трассировка печатной платы осуществляется с помощью представленных ниже программных средств.

6. Испытания устройства

Приводятся результаты моделирования схемы устройства с записанной в микроконтроллер разработанной программой. Приводятся необходимые рисунки и графики. В обязательном порядке строятся временные диаграммы работы устройства (количество и наименование осей по временным диаграммам выбирается самостоятельно каждым студентом). Представленное описание, диаграммы должны полностью иллюстрировать принцип работы устройства и его работоспособность.

Модель до защиты в обязательном порядке должна быть продемонстрирована преподавателю.

Заключение

Делаются заключительные выводы о работе устройства, приводятся возможные рекомендации по дальнейшему использованию разработки. В случае необходимости возможны рекомендации по тематике проекта (работы) или варианты работ, которые можно проделать в будущем.

Список используемой литературы

Приводится полный перечень используемой литературы. Возможно указание собственно найденной и использованной литературы

Приложения:

А. Техническое задание на разработку

Б. Структурная схема устройства

В. Алгоритмическая схема работы программы

Г. Листинг текста программы

Д. Результаты моделирования работы устройства

Е. Монтажная схема со спецификацией к используемым элементам

Ж. Другие материалы по желанию

2.2. *Содержание графической части*

Представленные рекомендации касаются разработки графической части при выполнении курсового проекта. Общее количество листов – 3, формата А1. Рекомендации по размещению материала ориентировочные, указанный перечень выносимого материала может быть дополнен, либо изменено размещение материала по листам.

Однако, указанные ниже материалы должны в обязательном порядке присутствовать на листах.

Лист 1: Детальная структурная схема устройства (условное обозначение с указанием линий взаимодействия основных элементов); Алгоритмическая схема разработанной программы;

Лист 2: Схема модели устройства и полученные результаты моделирования (временные диаграммы, рисунки и т.п.) полученные, например, в программе «Proteus»;

Лист 3: Монтажная схема и спецификации к устройству.

2.3. *Примечания*

Оформление текста и графической части работы необходимо осуществить согласно действующим требованиям Амурского государственного университета.

3. *Источники информации и их характеристика*

Ниже приведены источники информации и их краткая характеристика, которые можно использовать при выполнении курсового проекта и расчетно-графической работы. Данные материалы разбиты на группы, содержание каждой группы представлено ниже, курсивом даны рекомендации по использованию (материл доступен у ведущего преподавателя).

Общие документы:

3.1. ГОСТ-19.2001-78 «Техническое задание. Требование к содержанию и оформлению» *(для составления технического задания)*

3.2. ГОСТ-2.102-68 «Едина система конструкторской документации. Виды и комплектность конструкторских документов» *(для оформления графической части и текстовых документов)*

3.3. В.В. Скороделов «Проектирование устройств на однокристальных микроконтроллерах с RISC-архитектурой» *(методичка, представляет общую информацию по проектированию, есть описание PIC – архитектура, принципы программирования, и т.п.)*

Книги:

3.4. Группа «!!!_AVR_ATMEL» - 13 книг по микроконтроллерам фирмы ATMEL. (*Представлена для желающих выполнить задания не пользуясь серией PIC. Замечание: некоторые книги по данной тематике есть в папке PIC – это касается справочников, в которых описаны многие серии*)

3.5. Группа «!!!_PIC» - 23 книги и статьи по микроконтроллерам PIC и не только. Самая полная папка по содержанию (*наиболее полное собрание книг по данной тематике, есть книги с приложенными к ним дисками, на которых представлены примеры программ. Подборка составлялась из книг, в которых есть готовые схемы аппаратной части и примеры программ*)

3.6. Группа «!!!_Монтаж и вспомогательное по МПСУ» - 19 книг по общей тематике. Имеется рекомендации по изготовлению и принципам расчета печатных плат, выбору и расчету параметров схем, изготовление программаторов, маркировке компонентов и хороший справочник по микросхемам

Программы:

Содержание данной папки детально описано в следующем разделе.

Полезные ссылки:

http://www.microchip.ru/ - "Про PIC контроллеры"
http://pic16f84.narod.ru/ - "Есть программаторы и возможна покупка"
http://eldigi.ru/ - "Сайт Элдиги, очень полезный"
http://easyelectronics.ru/ - по AVR

4. *Программные продукты и материалы*

4.1. Программа MPLAB IDE, версии 7.42 от производителя Microchip. Содержит все необходимое для создания и написания программ для микроконтроллеров серии PIC на ассемблере и перевода написанной программы в шестнадцатеричный код (*рекомендуемая программа для создания исполняемого модуля*).

4.2. Программа «Proteus». Данная программа предназначена для создания и моделирования схем на базе различных элементов, в том числе и микроконтроллеров. Содержит большое число демонстрационных примеров. С помощью нее можно проводить и автоматическую разводку печатной платы. (*рекомендуемая программа для моделирования и создания микроконтроллерных устройств).*

7.2 ЛАБОРАТОРНЫЕ РАБОТЫ

Лабораторные работы проводятся согласно рекомендациям и в форме изложенной в п.3.2. Тематика и рассматриваемые вопросы по каждому заня-

тию представлены ниже. Полный комплект заданий для лабораторных работ представлен в разделе 9.2.

7 семестр (16 часов)

Первое занятие (2 часа) – вводное, выполнение работы.

Тема: «Знакомство с учебным стендом УМК»

Цель работы:

Знакомство с устройством и принципом работы учебного стенда УМК

Рассматриваемые вопросы:

1. Знакомство с устройством, принципом работы стенда УМК
2. Заполнение массива ОЗУ последовательностью чисел арифметической прогрессии
3. Заполнение массива ОЗУ константой
4. Перемещение массива в памяти
5. Определение контрольной суммы массива чисел

Второе занятие (2 часа) – защита работы.

Тема: «Знакомство с учебным стендом УМК»

Цель работы:

1. Выполнить отчет по лабораторной работе №1.
2. Ответить на вопросы по методике снятия работы.
3. Дать ответы на вопросы преподавателя.

Третье занятие (2 часа) – выполнение работы.

Тема: «Запись и выполнение простых программ на учебном стенде УМК»

Цель работы:

1. Знакомство с системой команд микропроцессора КР580ИК80.
2. запись и исследование выполнения простых программ.

Рассматриваемые вопросы:

1. Исследование программы 1: вычисление простейшей формулы
2. Проанализировать результат выполнения программы 1 с помощью регистра словосостояния
3. Исследовать процесс выполнения программы 1 в пошаговом режиме
4. Исследовать варианты модернизированной программы 1

Четвертое занятие (2 часа) – защита работы.

Тема: «Запись и выполнение простых программ на учебном стенде УМК»

Цель работы:

1. Выполнить отчет по лабораторной работе №2.
2. Ответить на вопросы по методике снятия работы.
3. Дать ответы на вопросы преподавателя.

Пятое занятие (2 часа) – выполнение работы.

Тема: «Программная реализация типовых управляющих воздействий и вычислительных процедур на учебном стенде УМК»

Цель работы:

Ознакомление с принципами формирования микропроцессором управляющих воздействий различными технологическими устройствами, их программирования, а так же программной реализацией вычислительных процедур

Рассматриваемые вопросы:

1. Анализ работы микропроцессора в режиме ожидания события
2. Анализ работы микропроцессора в режиме формирования управляющего сигнала
3. Анализ работы микропроцессора в режиме формирования временной задержки
4. Реализация процедур сложения и умножения

Шестое занятие (2 часа) – защита работы.

Тема: «Программная реализация типовых управляющих воздействий и вычислительных процедур на учебном стенде УМК»

Цель работы:

1. Выполнить отчет по лабораторной работе №3.
2. Ответить на вопросы по методике снятия работы.
3. Дать ответы на вопросы преподавателя.

Седьмое занятие (2 часа) – выполнение работы.

Тема: «Организация циклов, обработка массивов и маскирование данных на учебном стенде УМК»

Цель работы:

1. Изучение способов организации циклов с помощью условных переходов.
2. обработка массивов данных и маскирование данных.

Рассматриваемые вопросы:

1. Исследование программы 1: суммирование элементов массива
2. Исследование программы 2: нахождение максимума
3. Исследование программы 3: реализация переключательной функции

Восьмое занятие (2 часа) – защита работы.

Тема: «Организация циклов, обработка массивов и маскирование данных на учебном стенде УМК»

Цель работы:

1. Выполнить отчет по лабораторной работе №4.
2. Ответить на вопросы по методике снятия работы.
3. Дать ответы на вопросы преподавателя.

8 семестр (15 часов)

Первое занятие (2 часа) – выполнение работы.

Тема: «Изучение стенда Siemens Simatic S7 200»

Цель работы:

Научится запускать и работать с учебным стендом Siemens Simatic S7 200

Рассматриваемые вопросы:

1. Изучить команды ввода/вывода и логических действий
2. Написать программу, реализующую логическую функцию
3. Переслать программу в контроллер
4. Смоделировать работу программы на стенде.

Второе занятие (2 часа) – выполнение работы.

Тема: «Создание базовых программ на стенде Siemens Simatic S7 200»

Цель работы:

Научится создавать и исследовать простейшие программы на стенде Siemens Simatic S7 200

Рассматриваемые вопросы:

1. Изучить таймер
2. Написать программу на ассемблере, реализующую заданный алгоритм
3. Переслать программу в контроллер
4. Смоделировать работу программы на стенде.

Третье занятие (2 час) – защита работ.

Тема 1: «Изучение стенда Siemens Simatic S7 200»

Тема 2: «Создание базовых программ на стенде Siemens Simatic S7 200»

Цель работы:

1. Выполнить отчеты по лабораторным работам №1,2.
2. Ответить на вопросы по методике снятия работ.
3. Дать ответы на вопросы преподавателя.

Четвертое занятие (2 часа) – выполнение работы.

Тема: «Создание программ регулирования по условию на стенде Siemens S7 200»

Цель работы:

Создать и исследовать программу регулирования по условию на стенде Siemens S7 200

Рассматриваемые вопросы:

1. Создать программу на ассемблере, реализующую заданный алгоритм
2. Переслать программу в контроллер
3. Смоделировать работу программы на стенде.

Пятое занятие (2 часа) – выполнение работы.

Тема: «Создание программ автоматического и ручного регулирования по условию на стенде Siemens S7 200»

Цель работы:

Создать и исследовать программу автоматического и ручного регулирования по условию на стенде Siemens S7 200

Рассматриваемые вопросы:

1. Создать программу на ассемблере, реализующую заданный алгоритм
2. Переслать программу в контроллер
3. Смоделировать работу программы на стенде.

Шестое занятие (2 час) – защита работ.

Тема 3: «Создание программ регулирования по условию на стенде Siemens S7 200»

Тема 4: «Создание программ автоматического и ручного регулирования по условию на стенде Siemens S7 200»

Цель работы:

1. Выполнить отчеты по лабораторным работам № 3,4.
2. Ответить на вопросы по методике снятия работ.
3. Дать ответы на вопросы преподавателя.

Седьмое занятие (3 часа) – выполнение и защита работы.

Тема: «Создание программ автоматического и ручного регулирования по условию с задержкой срабатывания на стенде Siemens S7 200»

Цель работы:

1. Создать и исследовать программу автоматического и ручного регулирования по условию с задержкой на стенде Siemens S7 200.
2. Выполнить отчет по лабораторной работе № 5.
3. Ответить на вопросы по методике снятия работ.
4. Дать ответы на вопросы преподавателя.

Рассматриваемые вопросы:

1. Создать программу на ассемблере, реализующую заданный алгоритм
2. Переслать программу в контроллер
3. Смоделировать работу программы на стенде.

9 семестр (14 часов)

Первое и второе занятия (4 часа) – вводное, выполнение работы.

Тема: «Основные приемы программирования на Ассемблере для ПК»

Цель работы:

1. Изучение структуры программы на Ассемблере.
2. Изучение способов программирования ветвлений.
3. Изучение способов программирования циклов.
4. Изучение способов обработки массивов.

Рассматриваемые вопросы:

1. Создание программы на Ассемблере для ввода и вывода массива чисел
2. Компилирование программы для ввода и вывода массива чисел

3. Проверка правильности работы программы для ввода и вывода массива чисел

Третье и четвертое занятия (4 часа) – выполнение работы.

Тема: «Обработка прерываний на Ассемблере для ПК»

Цель работы:

1. Изучение способов обработки прерываний.
2. Обработка прерываний от системного таймера.
3. Обработка прерываний нажатия клавиш клавиатуры.

Рассматриваемые вопросы:

1. Создание программы на Ассемблере для демонстрации прерывания системного таймера по выводу текущего времени
2. Компилирование программы для демонстрации прерывания системного таймера по выводу текущего времени
3. Проверка правильности работы программы для демонстрации прерывания системного таймера по выводу текущего времени
4. Создание программы на Ассемблере для обработки прерывания от клавиатуры
5. Компилирование программы для обработки прерывания от клавиатуры
6. Проверка правильности работы программы для обработки прерывания от клавиатуры

Пятое занятие (2 часа) – защита работ.

Тема 1: «Основные приемы программирования на Ассемблере для ПК»

Тема 2: «Обработка прерываний на Ассемблере для ПК»

Цель работы:

1. Выполнить отчеты по лабораторным работам № 1,2.
2. Ответить на вопросы по методике выполнения работ.
3. Дать ответы на вопросы преподавателя.

Шестое занятие (2 часа) – выполнение работы.

Тема: «Создание резидентных программ на Ассемблере для ПК»

Цель работы:

1. Изучение способов предотвращения повторной загрузки резидента в память.
2. Изучение способов выгрузки резидентов из памяти.
3. Изучение способов передачи параметров резидентам.

Рассматриваемые вопросы:

1. Создание программы на Ассемблере для самовыгружающегося резидентного обработчика прерывания
2. Компилирование программы для самовыгружающегося резидентного обработчика прерывания
3. Проверка правильности работы программы для самовыгружающегося резидентного обработчика прерывания

Седьмое занятие (2 часа) – защита работы.

Тема: «Создание резидентных программ на Ассемблере для ПК»

Цель работы:

1. Выполнить отчет по лабораторной работе № 3.
2. Ответить на вопросы по методике выполнения работы.
3. Дать ответы на вопросы преподавателя.

7.3 ПРАКТИЧЕСКИЕ ЗАНЯТИЯ

Практические занятия проводят согласно рекомендациям и в форме изложенной в п.3.3. Тематика и рассматриваемые вопросы по каждому практическому занятию приведены ниже.

8 семестр (15 часов)

Первое занятие (3 часа).

Тема: «Изучение основ работы с контроллером Mega-128»

Рассматриваемые вопросы:

1. Изучение команд ввода/вывода и логических операций.
2. Изучение основ написания программ на ассемблере, реализующих логические функции.
3. Моделирование работы программы в отладчикс PLC.
4. Компилирование программы в машинный код.
5. Пересылка программ в контроллер при помощи программатора.

Примечание: для каждого рассматриваемо вопроса варианты заданий формулируются согласно п.9.3.

Второе, третье, четвертое занятия (6 часов).

Тема: «Изучение работы контроллера Mega-128 при реализации основных алгоритмов»

Рассматриваемые вопросы:

1. Изучить основные команды сравнения.
2. Изучить работу таймера.
3. Создать завершенные команды работы контроллера по поддержанию уровня.
4. Создать завершенные команды работы контроллера по формированию задержек.
5. Создать завершенные команды работы контроллера по заданной установке выхода в требуемое значение.
6. Создать завершенные команды работы контроллера по формирования аварийной сигнализации по заданным условиям.

Примечание: для каждого рассматриваемо вопроса варианты заданий формулируются согласно п.9.3.

Пятое, шестое, седьмое занятия (6 часов).

Тема: «Изучение работы контроллера Mega-128 при реализации завершенных алгоритмов управления»

Рассматриваемые вопросы:

1. Создание программ поддержания уровня в баке в двух заданных диапазонах с учетом задания работы контроллера в автоматическом и в ручном режиме.
2. Создание программ поддержания уровня в резервуаре в двух заданных диапазонах, с учетом задержки на включени и выключение в автоматическом и ручном режиме.

Примечание: для каждого рассматриваемо вопроса варианты заданий формулируются согласно п.9.3.

9 семестр (14 часов)

Первое, второе занятие (4 часа).

Тема: «Основы построения системы регулирования на базе микроконтроллера Siemens Simatic S7 200 и SCADA системы TM5»

Рассматриваемые вопросы:

1. Построение системы регулирования уровня.
2. Подключение драйверов контроллера к SCADA системе.
3. Создание проекта, основных узлов проекта, принципы подключения сигналов к TM5 на примере Siemens Simatic S7 200.

Примечание: для каждого рассматриваемо вопроса варианты заданий формулируются согласно п.9.3.

Третье, четвертое занятие (4 часа).

Тема: «Разработка завершенного проекта системы регулирования на базе микроконтроллера Siemens Simatic S7 200 и SCADA системы TM5»

Рассматриваемые вопросы:

1. Настройка каналов управления микроконтроллера Siemens Simatic S7 200.
2. Создание монитора реального времени для микроконтроллера Siemens Simatic S7 200.
3. Создание графической базы узлов и разработка графического интерфейса для системы регулирования.

Примечание: для каждого рассматриваемо вопроса варианты заданий формулируются согласно п.9.3.

Пятое, шестое занятие (4 часа).

Тема: «Разработка вспомогательных элементов проекта системы регулирования на базе микроконтроллера Siemens Simatic S7 200 и SCADA системы TM5»

Рассматриваемые вопросы:

1. Настройка системы архивирования, каналов для архивирования.

2. Создание СПАД-архивов.

3. Создание отчетов тревог. Разработка различных отчетов.

4. Разработка шаблонов, создание сценариев и генерация отчетов.

Примечание: для каждого рассматриваемо вопроса варианты заданий формулируются согласно п.9.3.

Седьмое занятие (2 часа).

Тема: «Разработка системы управления контроллером с использованием возможностей FBD-программирования на базе микроконтроллера Siemens Simatic S7 200 и SCADA системы TM5»

Рассматриваемые вопросы:

1. Создание и разработка FBD-программ реализующих конкретную задачу управления.

2. Подключение FBD-программы к каналам управления.

Примечание: для каждого рассматриваемо вопроса варианты заданий формулируются согласно п.9.3.

7.4 КОНТРОЛЬНЫЕ РАБОТЫ

Контрольные работы проводят согласно рекомендациям и в форме, изложенной в п.3.4. Темы и количество вопросов в каждом варианте теста представлены ниже (полный перечень см.п. 9.4). Темы пронумерованы в соответствии с учебной программой дисциплины.

1.1. Микропроцессорные системы – определение, структура, типы

пять вопросов;

1.2. Организация обмена информацией в МПС

пять вопросов;

1.3. Арбитраж шин; 1.4. Методы повышения эффективности шин

пять вопросов;

1.5. Основные элементы МПС. Микропроцессор

пять вопросов;

1.6. Основные элементы МПС. Память и устройства ввода/вывода

четыре вопроса;

1.7. Функционирование МПС. Адресация и ее особенности, регистры

пять вопросов;

1.8. Программные основы работы МП

пять вопросов;

1.9. Микроконтроллеры. Основы организации

шесть вопросов;

1.10. Внутренние и внешние связи в микроконтроллерах

девять вопросов;

1.11. Аппаратные средства микроконтроллеров

девять вопросов;

1.12. Проектирование устройств на микроконтроллерах

девять вопросов;

1.13. Введение: микроконтроллеры серии PIC и AVR

шесть вопросов;

1.14. Принципы работы, организация памяти и особенности выполнения команд для микроконтроллеров PIC и AVR

шесть вопросов;

1.15. Организация обмена с внешними устройствами, память, прерывания для микроконтроллеров PIC и AVR

восемь вопросов;

1.16. Специальные функции и система команд микроконтроллеров PIC и AVR

восемь вопросов;

1.17. Особенности программирования и отладки, разработка программного кода для микроконтроллеров PIC и AVR

семь вопросов;

1.18. Разработка программного обеспечения для микроконтроллеров PIC и AVR

1.19. Макет микропроцессорной системы и программирование простейших задач для микроконтроллеров PIC и AVR

четырнадцать вопросов.

8. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ОРГАНИЗАЦИИ МЕЖСЕССИОННОГО И ЭКЗАМЕНАЦИОННОГО (ЗАЧЕТНОГО) КОНТРОЛЯ ЗНАНИЙ СТУДЕНТОВ

Организация аттестации студентов, проводится в соответствии с положением АмГУ о курсовых экзаменах и зачетах (далее приведена выписка из положения):

2.1. Организация аттестации студентов в университете по специальностям и направлениям высшего профессионального образования регламентируется рабочим учебным планом, расписанием учебных занятий и программами учебных дисциплин, утверждаемыми в установленном в университете порядке.

Контроль за качеством освоения образовательных программ осуществляется путем текущей внутрисеместровой аттестации, ректорской контрольной аттестации, промежуточной аттестации студентов в форме курсовых экзаменов и зачетов и итоговой аттестации выпускников.

2.2. Курсовые экзамены и зачеты проводятся по дисциплинам утвержденного учебного плана по соответствующим специальностям и направлениям высшего профессионального образования. Знания, умения и навыки обучающегося определяются оценками "отлично", " хорошо", " удовлетворительно", " неудовлетворительно", "зачтено" и "незачтено".

2.3. Студенты, обучающиеся по основным программам высшего профессионального образования, сдают в течение учебного года не более 10 экзаменов и 12 зачетов. В это число не входит аттестация по физической культуре и факультативным дисциплинам.

Студенты, обучающиеся в сокращенные сроки (по индивидуальным планам), в течение учебного года сдают не более 20 экзаменов и 24 зачетов.

2.4. Сроки проведения курсовых зачетов и экзаменов (экзаменационная сессия) и начало очередного учебного семестра устанавливаются графиком учебного процесса, утвержденным проректором по учебной работе.

Расписание экзаменов составляется в соответствии с графиком учебного процесса, утверждается проректором по учебно-научной работе и доводится до сведения преподавателей и студентов не позднее, чем за две недели до начала сессии. Расписание составляется таким образом, чтобы на подготовку к экзаменам по каждой дисциплине было отведено не менее 3 дней, исключая день предыдущего экзамена. По согласованию с деканами и заведующими соответствующих кафедр отдельные экзамены (зачеты) могут проводиться в течение семестра по завершении преподавания дисциплины.

8.1 ОРГАНИЗАЦИЯ МЕЖСЕССИОННОГО КОНТРОЛЯ

В соответствии с вышеприведенным положением АмГУ рекомендуется следующий способ межсессионной аттестации студентов. Аттестация проводится дважды в семестр. Аттестационная оценка складывается из следующих критериев:

- оценок полученных на соответствующих контрольных работах;
- оценки характеризующей выполнение и защиту лабораторных работ;
- оценки характеризующей работу студентов на практических и семинарских занятиях.

При этом преимущественным весом обладают оценки, характеризующие персональное усвоение материала студентом (оценки по контрольным работам и оценки, характеризующая выполнение лабораторных работ). Оцен-

Оценки, характеризующие работу студента на практических занятиях, в большинстве случае может учитываться в роли повышающей, если таковая работа студента имеется.

8.2 ОРГАНИЗАЦИЯ ЭКЗАМЕНАЦИОННОГО (ЗАЧЕТНОГО) КОНТРОЛЯ

В соответствии с вышеприведенным положением АмГУ итоговые знания и умения студента определяются оценками «отлично», «хорошо», «удовлетворительно» и «неудовлетворительно», «зачтено» и «не зачтено». Учебным планом предусматривается устная сдача экзамена (7, 8 семестр) и зачет (9 семестр). Основные вопросы, на которые студенту предстоит ответить на экзамене, определяются экзаменационным или зачетным билетом (п.11.1, п.11.2).

Экзаменационный билет состоит из двух теоретических вопросов и третьего – практического (в рамках которого студенту предлагается решить предложенные задачи). В рамках второго экзаменационного вопроса студенту представляется возможность самостоятельно выбрать необходимую схему для ответа на вопрос. Данная схема предназначена: с одной стороны для облегчения сдачи студентом экзамена (к схемам имеется свободный доступ, и нет необходимости заучивания большого количества ненужной информации), с другой стороны для оценки полного объемом знаний студента (может быть выбрано произвольное количество схем, но необходимость и достаточность выбора так же оценивается на экзамене).

Зачетный билет состоит из двух теоретических вопросов.

Помимо ответа по экзаменационному (зачетному) билету в случае наличия неликвидированных задолжностей по лабораторным работам, студентом на экзамене (зачете) так же защищаются и несданные работы. Оценка, полученная по результатам защиты лабораторных работ, учитывается при проставлении итоговой.

Студенты, проявившие особые успехи в освоении дисциплины (стопроцентная посещаемость занятий, успешное выполнение плана по сдаче лабораторных работ, отличная работа на практических занятиях, получившие оценку отлично на контрольных работах) могут быть по результатам выполнения теста (см. п.10.2) освобождены от ответа на один или несколько экзаменационных вопросов.

9. КОМПЛЕКТЫ ЗАДАНИЙ

9.1 КУРСОВОЙ ПРОЕКТ

Вариант 1-2008.

На основе микроконтроллера разработать устройство, выполняющее следующие функции:

Подсчет количества входных дискретных сигналов, поступающих от кнопки SB1. Количество поступивших импульсов необходимо отразить на 7-сегментном дисплее (таким образом, отображаемые числа от «0» до «9»). В разработке предусмотреть кнопку сброса SB2 и схему автоматического сброса счетчика при достижении числа «10».

Вариант 1а-2009.

На основе микроконтроллера разработать устройство, выполняющее следующие функции:

Отображения на 7-сегментном дисплее шестнадцатеричных чисел от «0» до «F». При запуске устройства должно отображаться «A». При нажатии кнопки «SB+» отображаемое число увеличивается на единицу. При нажатии кнопки «SB-» отображаемое число уменьшается на единицу. При одновременном нажатии кнопок «SB+» и «SB-» схема сбрасывается. Факт достижения «0» сигнализировать зажиганием светодиода «VD-min», а достижения «F» сигнализировать зажиганием светодиода «VD-max».

Вариант 1б-2009.

На основе микроконтроллера разработать устройство, выполняющее следующие функции:

Сравнения 2-х введенных чисел. Первое число вводится при нажатии кнопки SB1 и выводится на 1 первый семисегментный дисплей, второе - при нажатии кнопки SB2 и выводится на 2 первый семисегментный дисплей. Порядок ввода чисел: при нажатии кнопки и удержании ее на протяжении 1 сек – число увеличивается на «1», и так до «F», далее следует «0» и процедура повторяется. При нажатии кнопки SB3 введенные числа сравниваются, и максимальное число продолжает отображаться на соответствующем дисплее, минимальное удаляется (соответствующий семисегментный дисплей гаснет). При нажатии кнопки SB4 – повторяется та же процедура по сравнению, только ищется минимальное число.

Вариант 2-2008.

На основе микроконтроллера разработать устройство, выполняющее следующие функции:

Звучание динамика в трех независимых режимах. *Первый режим* - непрерывный звук (при нажатой кнопке SB1). *Второй режим* - прерываюшиеся сигналы длительностью 2 сек, через интервал 5 сек (режим активизируется однократным нажатием кнопки SB2). *Третий режим* - сигналы длительностью 5 сек, через интервал в 1 сек, количество которых предварительно сохранено в памяти (выдача сигналов активизируется однократным нажатием

на кнопку SB3). Количество сохраненных сигналов задается путем нажатия кнопки SB4, при этом считывается и сохраняется в памяти 3-х разрядное двоичное число определяющее количество выдаваемых импульсов в *третьем режиме.*

Вариант 2а-2009.

На основе микроконтроллера разработать устройство, выполняющее следующие функции:

Озвучивание композиции «Metallic – The Unforgiving». При разработке использовать 3 динамика (и больше). Запуск композиции осуществить при нажатии кнопки SB1. Проигрывание «заново» реализовать при нажатии кнопки SB2. Паузу в проигрывании реализовать при нажатии кнопки SB3.

Вариант 3-2008.

На основе микроконтроллера разработать устройство, выполняющее следующие функции:

Управления движением через перекресток. Считать, что установлено 4-е четырех сторонних светофора. Длительность разрешающего (запрещающего) сигнала равна 15 сек. Длительность предупреждающего сигнала 2 сек. При необходимости использовать дополнительные логические схемы.

Вариант 3а-2009.

На основе микроконтроллера разработать устройство, выполняющее следующие функции:

Управления движением через перекресток с одним светофором и индикатором оставшегося времени горения сигнала. Длительность разрешающего (запрещающего) сигнала равна 15 сек. Длительность предупреждающего (желтого) сигнала 2 сек. Индикация оставшегося времени должна осуществляться с помощью 2 семисегментных индикаторов (десятки секунд и целые секунды). Дополнительно предусмотреть совместное горение «красный+желтый» при переключении на «зеленый» («зеленый+желтый» при переключении на «красный») и трехразовое мигание «зеленого» при переключении на «красный».

Вариант 4-2008.

На основе микроконтроллера разработать устройство, выполняющее следующие функции:

Бегущая строка из 5-ти светодиодов. При этом при нажатии на кнопку SB1 – загораются все светодиоды. При нажатии на кнопку SB2 – тухнут все (сброс). При нажатии на кнопку SB3 реализуется режим одного «бегущего диода». При одновременном нажатии на кнопки SB1, SB3 – последовательно загораются сначала один, затем второй и т.д. светодиоды; тухнуть диоды начинают аналогично, начиная с первого. При нажатии одновременно на SB2, SB3 – реализуется режим мигания – первый диод мигает с частотой 1 раз в секунду, второй – 1 раз в 2 секунды, третий – 1 раз в 4 сек. и т.д.

Вариант 4а-2009.

На основе микроконтроллера разработать устройство, выполняющее следующие функции:

Бегущая строка из 8-и светодиодов. При нажатии на кнопку SB1 – последовательно загораются VD1, затем VD1 и VD2 и так далее; затем тухнут, начиная с VD8 (интервал между событиями 1 сек). При нажатии на кнопку SB2 – последовательность меняется. При нажатии на кнопку SB3 – организуется сброс. При нажатой кнопке SB4 – работают только четные диоды, при нажатой SB5 – работают только нечетные диоды.

Вариант 5-2008.

На основе микроконтроллера разработать устройство, выполняющее следующие функции:

Отображение имени (на основе LCD-матрицы), активизируемое при нажатии кнопки SB1. И завершающиеся при нажатии кнопки SB2.

Вариант 5а-2009.

На основе микроконтроллера разработать устройство, выполняющее следующие функции:

Опрос состояния 2-х кнопок SB1 и SB2. Запрос на нажатие и результат опроса отображать на LCD-матрицу. Так сначала бегущей строкой выводится запрос «Нажмите кнопку 1», если после запроса была нажата правильная кнопка, то выводится сообщение «Правильно!», если кнопка была нажата не правильная, то выводится сообщение «Не правильно! Нажмите кнопку 1». Аналогично производится работа со второй кнопкой. Если в течение 5 секунд не нажата никакая кнопка (на соответствующий запрос) то выводится сообщение «Время истекло!». Запросы на нажатие какой-либо кнопки повторять случайным образом.

Вариант 6-2008.

На основе микроконтроллера разработать устройство, выполняющее следующие функции:

Отображение номера нажатой кнопки (на основе LCD-матрицы). При этом необходимо непрерывно проводить опрос 7-ми кнопок (датчиков). Если не нажата никакая кнопка – выводить на дисплей запрос нажатия (сам запрос сформулировать произвольным образом). Если нажата какая-либо кнопка – на дисплей вывести номер нажатой кнопки. Кнопки в устройстве реализовать с фиксацией нажатия. Предусмотреть возможность анализа ситуации, когда нажато несколько кнопок.

Вариант 6а-2009.

На основе микроконтроллера разработать устройство, выполняющее следующие функции:

Отображение номера нажатой кнопки на семисегментном дисплее. При этом необходимо непрерывно проводить опрос 8-ми кнопок (датчиков). Если не нажата никакая кнопка – выводить на дисплей запрос нажатия (буква F). Если нажата какая-либо кнопка – на дисплей вывести номер нажатой кнопки. Кнопки в устройстве реализовать с фиксацией нажатия. Если нажато несколько кнопок – то необходимо выводить последовательно номера нажатых кнопок с интервалом в 1 сек (при этом, если нажата кнопка SB1 – номера выводятся, начиная с минимального, если кнопка SB1 не нажата, номера выводятся, начиная с максимального).

Вариант 7-2008.

На основе микроконтроллера разработать устройство, выполняющее следующие функции:

Разработать секундомер реального времени на основе 7-и сегментного дисплея. На секундомере отображать минуты (один индикатор) и секунды (два индикатора). Секундомер активизируется при нажатии на кнопку SB1. Запоминает время – при нажатии на SB2 (при этом отсчет времени не останавливается, а текущее время запоминается). Общий стоп секундомера реализовать на кнопке SB3. Количество запоминаемых отсчетов равно 4-ем. Отображение запомненных отсчетов времени реализовать при нажатии одновременном нажатой кнопке SB3, и при нажатии SB1 (если SB1 нажимается один раз – первое запомненное число; если SB1 нажимается 2 раза подряд – второе запомненное число и т.д.)

Вариант 7а-2009.

На основе микроконтроллера разработать устройство, выполняющее следующие функции:

Разработать секундомер реального времени на основе 2-х семисегментных дисплеев (первый отображает секунды, второй минуты). Пуск отсчета реализовать при нажатии кнопки SB1. Остановку отсчета реализовать при нажатии кнопки SB2. При этом предусмотреть реализацию прямого и обратного отсчета: так при работе схемы если однократно нажимается кнопка SB3, то отсчет меняет свое направление (если время увеличивалось, то после нажатия умешьшается, и наоборот).

Примечание: контроллер должен быть PIC16F84, а сама разработка без дополнительных схем.

Вариант 8-2008.

На основе микроконтроллера разработать устройство, выполняющее следующие функции:

Устройство управления температурным режимом. Считать что сигналы от «датчика температуры» и от «задающего устройства» представляет собой токовые сигнал 0-5мА. Управление (дискретный сигнал на открытие в сторону больше и меньше) подавать на исполнительный механизм постоянной скорости – клапан. Управляющее воздействие сформировать по закону «пропорционального регулирования»

Вариант 8а-2009.

На основе микроконтроллера разработать устройство, выполняющее следующие функции:

Устройство управления температурным режимом. Считать что сигналы от «датчика температуры» и от «задающего устройства» представляет собой аналоговый сигнал (их изменение смоделировать с помощью потенциометра). Управление подается на исполнительный механизм постоянной скорости (клапан) и формируется по закону пропорционального регулирования. Если клапан должен быть «открыт», то загорается светодиод VD1, если требуется клапан «закрыть» то загорается светодиод VD2. Клапан «открывается» если сигнал от «датчика температуры» меньше чем от «задающего устройства» и

наоборот. Минимальная длительность сигнала на «открытие» или «закрытие» 1 сек при рассогласовании 10%, 2 сек при рассогласовании 20% и т.д.

Вариант 9-2008.

На основе микроконтроллера разработать устройство, выполняющее следующие функции:

Условное отображение уровня входного напряжения. Разработку провести БЕЗ использования завершенных схем АЦП. При этом заданное в программе базовое значение напряжения сравнивать с входящим. Если напряжение входное напряжение «меньше» заданного загорается светодиод VD1 – «отрицательное напряжение»; в противном случае светодиод VD1 не горит. При этом предусмотреть отображение процентной разности на 7-и сегментном дисплее (так разница до 10% соответствует цифре «1», до 20% соответствует цифре «2», и т.д.)

Вариант 9а-2009.

На основе микроконтроллера разработать устройство, выполняющее следующие функции:

Отображение уровня входного напряжения. Разработку провести на микроконтроллере без встроенного АЦП (с дополнительным модулем АЦП). Диапазон входного анализируемого напряжения от 0В до 1,5В (задается потенциометром). Результат обработки выводить на 2 семисегментных индикатора (первый индикатор отображает целые единицы, второй - десятые).

Примечание: Реализовать схему на основе микроконтроллера PIC16F84, схема АЦП любая.

Вариант 9б-2009.

На основе микроконтроллера разработать устройство, выполняющее следующие функции:

Регулирование освещенности. Считать, что освещенность в комнате анализируется с помощью фотодатчика, имеющего аналоговый выход. В зависимости от уровня освещенности должны быть включены: одна лампа (светодиод VD1), две лампы (светодиод VD1, VD2) и т.д. Всего ламп – восемь.

Примечание: Реализовать схему на основе любого микроконтроллера с встроенным АЦП, например - PIC16F8734.

Вариант 10-2008.

На основе микроконтроллера разработать устройство, выполняющее следующие функции:

Принцип управления кодовым замком. При нажатой кнопке SB1 – последовательно вводиться комбинация из трех цифр. Факт ввода цифры сигнализируется световым диодом VD1. При разомкнутой кнопке SB1 введенная комбинация проверяется. Если число совпадает с заданным, то загорается светодиод VD2.

Вариант 10а-2009.

На основе микроконтроллера разработать устройство, выполняющее следующие функции:

Кодовый замок. Индикация режима программирования осуществляется с помощью диода VD1, режима работы с помощью диода VD2. Если замок открыт, то должен гореть диод VD3. Ввод комбинации осуществляется с помощью кнопок «SB1, SB2,…SB8». Окончание ввода (проверка комбинации) осуществляется при нажатии кнопки SB0. При первом включении схема переходит в режим «работа», правильной комбинацией на открытие является «00000000». В схеме предусмотреть режим неоднократного автоматического перепрограммирования: при вводе «правильной комбинации» и нажатии SB0 (проверка), загорается VD1 и VD3 (замок открыт), можно изменить состояние «SB1, SB2,…SB8» (новая комбинация) и отпускании кнопки SD0, диод VD3 тухнет – система переходит в режим проверки.

Вариант 11-2008.

На основе микроконтроллера разработать устройство, выполняющее следующие функции:

«Азбука Морзе». При этом кнопочными выключателями «с фиксированным нажатием» задается символ, который надо передать (задание передаваемого сигнала соответствует принципу 7-и сегментного дисплея). Передачи сигнала соответствует горение «зеленного индикатора» (светодиода VD1). Отсутствие передачи соответствует горение «красного индикатора» (светодиода VD2). Сигналы передаются по нажатию кнопки SB1. Выходные сигналы формируются через звуковой динамик. Всего анализируемых символов десять – цифры от «0» до «9».

Вариант 11а-2009.

На основе микроконтроллера разработать устройство, выполняющее следующие функции:

Отображение 3-х введенных символов. При этом кнопочными выключателями «с фиксированным нажатием» задается символ, который надо отобразить (задание передаваемого символа соответствует принципу 7-и сегментного дисплея). Отображение введенного символа осуществить на основе семисегментного дисплея. Предусмотреть кнопку SB0 (если нажата, то символ вводится, отпущена - выводится).

При активном режиме «ввода»: если нажата SB1 – считывается символ и запоминается как 1-ый введенный, если нажата SB2 - считывается символ и запоминается как 2-ой введенный, если нажата SB3 - считывается символ и запоминается как 3-ий введенный. При активном режиме «вывода»: последовательно на дисплее отображаются введенные символы, начиная с первого с интервалом около 1 сек.

Примечание: Реализовать схему на основе любого микроконтроллера, например PIC16F873, обладающего достаточным количеством портов.

Вариант 12-2008.

На основе микроконтроллера разработать устройство, выполняющее следующие функции:

«Азбука Морзе». При этом кнопочными выключателями «с фиксированным нажатием» задается символ, который надо передать (задание передаваемого сигнала соответствует принципу 7-и сегментного дисплея). Переда-

чи сигнала соответствует мигание с частотой в 2 сек. светодиода VD1. Отсутствие передачи соответствует горение светодиода VD1. Сигналы передаются по нажатию кнопки SB. Выходные сигналы формируются через индикатор – световой диод VD2. Всего анализируемых символов 10, любых заранее определенных букв.

Вариант 13-2008.

На основе микроконтроллера разработать устройство, выполняющее следующие функции:

Рисование продолжающейся линии. Отображать линию необходимо на 7–и сегментном дисплее. Первоначально загорается средний индикатор. Имеются 2 кнопки кратковременного нажатия: SB1 – считать направление; SB2 – сброс рисунка. Направление рисования задается кнопками с фиксированным нажатием: Слево Справо Сверх Сниз. При рисовании линии руководствоваться следующими правилами: загорается «правый верх» если нажаты и Справо и Сверх (и т.п.). Таким образом, задается первоначально направление, а затем нажимается SB1 – загорается (и не тухнет) соответствующий индикатор. Затем при разомкнутой SB1 – задается новое направление рисования, и снова нажимается SB1 – линия продолжается и т.д.

Вариант 13а-2009.

На основе микроконтроллера разработать устройство, выполняющее следующие функции:

Рисование продолжающейся линии. Отображать линию необходимо на двух 7 и сегментных дисплеях. Первоначально оба дисплея неактивны, отправная точка – «левый низ». Имеется кнопка кратковременного нажатия SB0 – сброс рисунка. Направление рисования задается кнопками с кратковременным нажатием: Слево Справо Сверх Сниз. При рисовании линии руководствоваться следующими правилами: линия рисуется в направлении указанном нажатием соответствующей кнопки, но без прерывания; если направление указано не корректно загорается светодиод VD1 «ошибка».

Примечание: Реализовать схему на основе любого микроконтроллера, например PIC16F873, обладающего достаточным количеством портов.

Вариант 14-2008.

На основе микроконтроллера разработать устройство, выполняющее следующие функции:

Распознать введенное число. Считать, что задан 7-сегментный дисплей, каждая сегмента которого это определенная кнопка (SB1-SB7). Первоначально активизируется кнопка SB0, которая сигнализирует, что в замкнутом положении – вводится распознаваемое число, а при разомкнутом состоянии – выводится распознанное. Диод VD1 загорается в случае если число распознано. Распознанное число отобразить любым способом (дисплей, LCD-матрица).

Вариант 14а-2009.

На основе микроконтроллера разработать устройство, выполняющее следующие функции:

Распознать введенную букву (буквы «A», «B», «C», «D», «E», «F», «G», «H», «I»). Считать, что задан 7-сегментный дисплей, каждая сегмента которого это определенная кнопка (SB1-SB7). Первоначально активизируется кнопка SB0, которая сигнализирует, что в замкнутом положении – вводится распознаваемая буква, а при разомкнутом состоянии – выводится распознанная. Диод VD1 загорается, в случае если буква распознана, VD2 – если буква не распознана. Распознанное число отобразить с помощью семисегментного дисплея.

Вариант 15-2008.

На основе микроконтроллера разработать устройство, выполняющее следующие функции:

Управление температурным режимом. Всего предусмотреть 2 режима работы, по заданному алгоритму. Режим программирования активизируется НАЖАТИЕМ кнопки SB1 (режим отработки задания активизируется при ОТЖАТОЙ кнопке SB1). В режиме программирования: задать общую длительность режима работы кнопкой SB2 (количество нажатий соответствует времени в минутах); задать длительность простоя в секундах кнопкой SB3. При этом предусмотреть индикацию в режиме программирования факта считывания контроллером нажатых кнопок. Таким образом, в режиме работы необходимо в соответствии с заданной длительностью включать и отключать нагреватель (фактом включения является горение светодиода VD1).

Вариант 15а-2009.

На основе микроконтроллера разработать устройство, выполняющее следующие функции:

Управление нагревательным элементом. В проектируемом устройстве 2 режима работы: «программирование» и «работа». Режим «работа» активизируется однократным нажатием кнопки SBпуск. В этом режиме светодиод VD1 должен «гореть» и быть «потушенным» запрограммированное время (цикл повторяется многократно). В режиме программирования (первоначальный режим при пуске) предусмотреть задание в секундах (заданное время выводится на 7-и сегментный индикатор):

- длительности «горения VD1»: при НАЖАТОЙ кнопке «SBрежим» путем последовательного нажатия кнопки «SBзад» время увеличивается на 1 сек и так до 9 сек.;
- длительности «потушенного VD1»: при ОТПУЩЕННОЙ кнопке «SBрежим» путем последовательного нажатия кнопки «SBзад» время увеличивается на 1 сек и так до 9 сек.

Вариант 16-2008.

На основе микроконтроллера разработать устройство, выполняющее следующие функции:

Последовательную прорисовку цифр на 7-и сегментном дисплее. Всего прорисовываемых цифр – 10: от «0» до «9». При этом: при нажатой кнопке SB1 – прорисовываются четные цифры, при отжатой кнопке SB1 – прорисовываются нечетные цифры. Прорисовку каждой цифры проводить последовательно, путем загорания соответствующего индикатора. Так при прорисов-

ке «0» сначала загорается 1-ый сегмент, затем через 2 сек. – 2-ой сегмент, затем через 2 сек. – 3 сегмент, и т.д. Интервал между прорисовкой каждой цифры – 5 сек. (между сегментами 0,5 сек; между цифрами 2 сек.)

Вариант 16а-2009.

На основе микроконтроллера разработать устройство, выполняющее следующие функции:

Последовательную прорисовку букв на 7-и сегментном дисплее. Всего прорисовываемых букв – 9: «A», «B», «C», «D», «E», «F», «G», «H», «I». При этом: при нажатой кнопке SB1 – прорисовываются гласные, при отжатой кнопке SB1 – прорисовываются согласные. Прорисовку каждой буквы проводить последовательно, путем загорания соответствующего индикатора. Так при прорисовке «A» сначала загорается 1-ый сегмент, затем через 1 сек. – 2-ой сегмент, и т.д. Интервал между прорисовкой каждой буквы – 2 сек.

Вариант 17-2009.

На основе микроконтроллера разработать устройство, выполняющее следующие функции:

Замер количества нажатия кнопки SB0. Работа устройства циклична, и повторяется, каждые 15 сек. При этом считывается количество нажатий кнопки SB0. Каждые 15 сек., после окончания процедуры подсчета на дисплей (7-и сегментный или LCD-матрицу) выводиться число нажатий на кнопку SB0, и сохраняется, следующие 15 сек. Процедура повторяется.

Вариант 17а-2009.

На основе микроконтроллера разработать устройство, выполняющее следующие функции:

Замер интервалов между нажатиями кнопки SB0. Работа устройства циклична, и повторяется, каждые 20 сек. При этом считывается первое нажатие кнопки и после этого начинается отсчет времени до второго нажатия кнопки SB0. Каждые 20 сек., после окончания процедуры подсчета на дисплей (7-и сегментный или LCD-матрицу) выводиться интервал времени между первым нажатием и вторым в секундах, и сохраняется, следующие 20 сек. Процедура повторяется. Первое нажатие кнопки SB0 приводит к свечению светодиода VD1, при втором нажатии загорается светодиод VD2.

Вариант 17б-2009.

На основе микроконтроллера разработать устройство, выполняющее следующие функции:

Замер времени нажатиями кнопки с фиксацией SB0. Работа устройства циклична, и повторяется, каждые 10 сек. При этом нажатое состояние кнопки и после этого начинается отсчет времени до ее отпускания. Каждые 10 сек., после окончания процедуры подсчета на дисплей (7-и сегментный или LCD-матрицу) выводиться интервал времени нажатия в секундах, и сохраняется, следующие 10 сек.

Вариант 18-2008.

На основе микроконтроллера разработать устройство, выполняющее следующие функции:

Отгадай число. Пуск программы происходит нажатием и удерживанием кнопки SB0, при этом последовательно прокручиваются цифры, начиная от «0» и заканчивая «9» При отжатой кнопке SB0 прокручивание последовательности цифр останавливается, число запоминается. Факт запоминания числа индицируется свечением светодиода VD1. Пользователь вводит число (с помощью нажатия кнопок SB1…SB7, которые фиксируются в нажатом положении, по принципу 7-и сегментного дисплея). Окончание ввода числа и проверки его на совпадение активизируется кнопкой SB8. Если число совпало, светодиод VD1 начинает мигать с частотой и длительностью 1 сек. В противном случае процедура опроса повторяется.

Вариант 18а-2009.

На основе микроконтроллера разработать устройство, выполняющее следующие функции:

Успей нажать. Пуск программы происходит по нажатию кнопки SB0, при этом произвольно на 7-и сегментном индикаторе одновременно загораются сегменты (всего таких вариантов не менее 10). Пользователь в течении 5 сек. должен активизировать нужные сегменты (с помощью нажатия кнопок SB1…SB7, которые фиксируются в нажатом положении, по принципу 7-и сегментного дисплея). По истечении 5 сек. проверяется соответствие активированных сегментов и заданных в рамках текущего варианта. Если комбинации равны – то загорается и тухнет светодиод «VD+» и процедура повторяется (задается новый вариант). Если введенная комбинация не совпадает с заданным вариантом начинает гореть VD1 (первая попытка) и процедура опроса через 5 сек. повторяется – если по-прежнему введенная комбинация не совпадает, загорается VD2 (вторая попытка), так же с VD3. По факту истечения 3 попыток (попытки суммируются на протяжении всех прокручиваемых 10-ти вариантов) программа останавливается и загорается светодиод «VD-». Если все 10-ть комбинаций нажаты вовремя, и попытки не исчерпаны загорается светодиод «VDwin» и программа останавливается.

Вариант 19-2008.

На основе микроконтроллера разработать устройство, выполняющее следующие функции:

Сравнения 2-х двухразрядных двоичных чисел, поступающих на микроконтроллер. Процедура сравнения активизируется нажатием на кнопку SB1. Если поступающие числа равны – активизируется светодиод VD1, если первое число больше второго – мигает светодиод VD2, если второе больше первого – мигает светодиод VD3. Числа задавать с помощью кнопок с фиксированным нажатием – первое число S10, S11; второе число – S20, S21.

Вариант 19а-2009.

На основе микроконтроллера разработать устройство, выполняющее следующие функции:

Сравнения 2-х восьми разрядных двоичных чисел, поступающих на микроконтроллер. Процедура сравнения активизируется нажатием на кнопку SB1. Если поступающие числа равны – активизируется светодиод VD=, если первое число больше второго – мигает светодиод VD12 (с частотой в 3 сек,

длительность горения 1 сек), если второе больше первого – мигает светодиод VD21 (с частотой в 1 сек, длительность горения 3 сек). Числа задавать с помощью кнопок с фиксированным нажатием.

Вариант 20-2008.

На основе микроконтроллера разработать устройство, выполняющее следующие функции:

Последовательного сложения чисел. Вводимые числа (каждое число находиться в интервале от «0» до «7») задавать кнопками с фиксированным нажатием SB0, SB1, SB2. Факт ввода инициализировать нажатием кнопки SB3. Полученную сумму графически отображать (с помощью 7-и сегментного дисплея или LCD-матрицы).

Вариант 20а-2009.

На основе микроконтроллера разработать устройство, выполняющее следующие функции:

Последовательного сложения чисел. Вводимые числа (каждое число находиться в интервале от «0» до «255») задавать кнопками с фиксированным нажатием SB0, SB1,… SB7. Факт ввода инициализировать нажатием кнопки SB+. Полученную трехразрядную сумму в виде шестнадцатеричного числа графически отображать с помощью 3-х семисегментных дисплеев.

Вариант 21-2008.

На основе микроконтроллера разработать устройство, выполняющее следующие функции:

Последовательного вычитания из числа 41 вводимых через порт чисел. Каждое вводимое число находиться в интервале от «0» до «7» и задается кнопками с фиксированным нажатием SB0, SB1, SB2. Факт ввода инициализировать нажатием кнопки SB3. Полученную разность графически отображать (с помощью 7-и сегментного дисплея или LCD-матрицы).

Вариант 21а-2009.

На основе микроконтроллера разработать устройство, выполняющее следующие функции:

Последовательного вычитания из числа «999» (десятичная система исчисления) числа введенного через порт. Вводимые числа (каждое число находиться в интервале от «0» до «255») задавать кнопками с фиксированным нажатием SB0, SB1,… SB7. Факт ввода инициализировать нажатием кнопки SB-. Полученную трехразрядную разность в виде десятичного числа графически отображать с помощью 3-х семисегментных дисплеев.

Вариант 22-2008.

На основе микроконтроллера разработать устройство, выполняющее следующие функции:

Сложения двух 2-х разрядных двоичных числа. Каждое число вводить через порт путем нажатия кнопок с фиксацией - SB0, SB1. Факт ввода первого числа инициируется при нажатии кнопки SB2, а второго SB3. Полученный результат при нажатии кнопки SB4 выводиться на 7-сегментный дисплей.

Вариант 22а-2009.

На основе микроконтроллера разработать устройство, выполняющее следующие функции:

Умножения двух 3-х разрядных двоичных числа. Каждое число вводить через порт путем нажатия кнопок с фиксацией – SB2, SB1, SB0. Факт ввода первого числа инициируется при нажатии кнопки SB3, а второго SB4. Полученный результат при нажатии кнопки SB5 выводиться на два 7-сегментных дисплея (десятки, единицы).

Вариант 23-2008.

На основе микроконтроллера разработать устройство, выполняющее следующие функции:

Вычитания двух 2-х разрядных двоичных чисел. Каждое число вводить через порт путем нажатия кнопок с фиксацией - SB0, SB1. Факт ввода первого числа инициируется при нажатии кнопки SB2, а второго SB3. Полученный результат при нажатии кнопки SB4 выводиться на 7-сегментный дисплей. При этом если результат отрицательный загорается светодиод VD1.

Вариант 23а-2009.

На основе микроконтроллера разработать устройство, выполняющее следующие функции:

Деления двух 3-х разрядных двоичных числа. Каждое число вводить через порт путем нажатия кнопок с фиксацией – SB2, SB1, SB0. Факт ввода первого числа инициируется при нажатии кнопки SB3, а второго SB4. Полученный результат при нажатии кнопки SB5 выводиться на два 7-сегментных дисплея (десятки, единицы). При этом если в результате получается не целое число выводить целый остаток, и зафиксировать этот факт путем светодиода VD1.

Вариант 24-2008.

На основе микроконтроллера разработать устройство, выполняющее следующие функции:

Составить программу по вычислению формулы: $a4 = (a1 + a2) \times a3$. Числа 3-х разрядные, двоичные. Каждое число вводить через порт путем нажатия кнопок с фиксацией – SB4, SB5, SB6. Факт ввода первого числа инициируется при нажатии кнопки SB1, а второго SB2, третьего SB3. Полученный результат при нажатии кнопки SB0 выводиться на 7-сегментный дисплей.

Вариант 25-2008.

На основе микроконтроллера разработать устройство, выполняющее следующие функции:

Составить программу по нахождению среди введенных чисел четного. Вводимые числа 4-х разрядные, двоичные. Каждое число вводить через порт путем нажатия кнопок с фиксацией – SB1, SB2, SB3, SB4. Факт проверки инициировать нажатием кнопки SB0. Если число четное – зажечь светодиод VD1, в противном случае VD2.

Вариант 25а-2009.

На основе микроконтроллера разработать устройство, выполняющее следующие функции:

Составить программу по нахождению среди введенных чисел четного. Вводимые числа 8-и разрядные, двоичные. Каждое число вводить через порт путем нажатия кнопок с фиксацией – SB1, SB2,…SB8. Факт проверки инициировать нажатием кнопки SB0. Если число четное – зажечь светодиод VD1, в противном случае VD2.

Вариант 26-2008.

На основе микроконтроллера разработать устройство, выполняющее следующие функции:

Опроса двоичных датчиков. Считать, что имеется 5 комнат, в каждой, из которой имеется 3 дискретных датчика. Информация с датчиков каждой комнаты поступает на порт каждые 4 секунды, последовательно начиная с первой комнаты. Предполагать так же, что «первый» датчик подключен к контакту D0, «второй» к D1, «третий» к D2. соответствующего порта. Если в какой либо из комнат сработали хотя бы 2 датчика, то сигнал «пожар» необходимо сформировать с помощью светодиода VD1, а на 7-и сегментный дисплей вывести номер комнаты со сработавшими датчиками.

Вариант 26а-2009.

На основе микроконтроллера разработать устройство, выполняющее следующие функции:

Опроса двоичных датчиков. Считать, что имеется 8 комнат, в каждой, из которой имеется 2 дискретных датчика. Реализовать процедуру опроса комнат, т.е. считать что датчики адресные их опрос производится выставлением на шине A3 A2 A1 A0 адреса (эти линии так же подключить к семисегментному дисплею). Опрос проводить каждые 10 сек., каждую комнату опрашивать 3 сек. При проведении опроса учесть следующие ситуации: если в комнате сработал 1 датчик, номер комнаты выводится на семисегментный дисплей, и загорается желтый светодиод VD1 (внимание); если в комнате сработали оба датчика, номер комнаты выводится (и мигает с частотой 1 раз в секунду) на семисегментный дисплей, и загорается красный светодиод VD2 (пожар). Процедура опроса циклично повторяется, если при опросе обнаружился пожар светодиод VD2 при опросе других комнат не тухнет.

Вариант 27-2008.

На основе микроконтроллера разработать устройство, выполняющее следующие функции:

Подсчет равных 3-х разрядных двоичных чисел. Исходное число ввести через порт с помощью кнопок с фиксацией – SB01, SB02, SB03 (данное число остается без изменения). Второе число (которое сравнивается) вводиться с помощью кнопок с фиксацией – SB11, SB12, SB13. Процедуру начала сравнения инициировать по нажатию кнопки SB0. Если числа равны, необходимо просигнализировать звуковым сигналом и мигающим светодиодом VD1.

Вариант 27а-2009.

На основе микроконтроллера разработать устройство, выполняющее следующие функции:

Подсчет равных 8-и разрядных двоичных чисел. Исходное число ввести через порт с помощью кнопок с фиксацией – SB01, SB02…SB07 (данное число остается без изменения). Второе число (которое сравнивается) вводиться с помощью кнопок с фиксацией – SB11, SB12,…SB17. Процедуру начала сравнения инициировать по нажатию кнопки SB0. Если числа равны, необходимо просигнализировать звуковым сигналом и мигающим светодиодом VD1.

Вариант 28-2008.

На основе микроконтроллера разработать устройство, выполняющее следующие функции:

Управление частотой мигания диодов. Считать, что имеется 7 светодиодов. Реализовать программу управления частотой мигания светодиодов с помощью кнопок SB1, SB2, SB3, SB4. При нажатии кнопки SB1 – частота мерцания первого, третьего, пятого и седьмого светодиодов увеличивается в 2 раза (при нажатии SB2 наоборот уменьшается в 2 раза). При нажатой SB3 – второй, четвертый, и шестой светодиоды мерцают по очереди, в противном случае одновременно с заданной частотой. При нажатии SB4 – частота мигания всех светодиодов уменьшается в два раза.

Вариант 28а-2009.

На основе микроконтроллера разработать устройство, выполняющее следующие функции:

Светомузыка. Считать, что имеется 16 светодиодов. Реализовать программу управления частотой их мигания по произвольному принципу в зависимости от нажатия кнопки SB0. Частота мигания диодов должна быть разная, каждое нажатие кнопки – она либо увеличивается, либо уменьшается.

Вариант 29-2009.

На основе микроконтроллера разработать устройство, выполняющее следующие функции:

Опроса адресных датчиков. Всего датчиков в системе 99. Опрос осуществляется выставлением на адресной шине (A1, A2,…,A7) номера датчика (номера меняются последовательно), при этом датчик должен ответить (моделируется кнопкой SB-ответ) в течении 2 сек. Если ответа не произошло необходимо сформировать сигнал неисправность (горит диод VD-неиспр), и в течение 3 сек опрос приостановить, а номер датчика выводить на 2 семисегментых индикатора. При опросе датчиков учесть факт их срабатывания: так при опросе если проходит сигнал SB-ответ (соответствующая кнопка) проверить факт его срабатывания (моделируется кнопкой SB-сраб). Если датчик сработал, то сформировать сигнал срабатывание (моделируется звучанием динамика), дальнейший опрос датчиков прекратить.

Вариант 30-2009.

На основе микроконтроллера разработать устройство, выполняющее следующие функции:

Работа пожарного инфракрасного датчика. При этом необходимо непрерывно отслеживать значение аналогового сигнала на входе (потенциометром), который информирует о возгорании. Если значение аналогового сигнала возросло до «напряжения срабатывания» (определить самостоятельно), то необходимо сформировать сигнал «пожар» (светодиодом VD). Каждые 10 сек проводить тест датчика (сам тест длится 5 сек.), при этом сформировать тестирующий сигнал (светодиод VD-тест), если в течении 2 сек значение аналогового сигнала возросло до «напряжения срабатывания» считать что тест прошел успешно, сигнал пожар не формировать, тест прекратить, датчик перевести в рабочее состояние.

Вариант 31-2009.

На основе микроконтроллера разработать устройство, выполняющее следующие функции:

Нахождения среди введенных чисел минимального. Всего чисел 9. Семиразрядные двоичные числа вводятся через порт с помощью кнопок с фиксированным нажатием SB7, SB6,…SB1, и считываются каждые 3 сек. Начало считывания сопровождается горением светодиода «VD-счит» (в течение этого времени пользователь вводит число – нажимает соответствующие кнопки). После считывание в течение 1 сек горит светодиод «VD-анализ», на 2-х семисегментных дисплеях (в 16-тиричном коде) выводится последнее введенное число. По факту завершения анализа всех чисел должен загораться светодиод «VD-min» и на 2 семисегментных дисплея (в 16-тиричном коде) выводится минимальное число.

Вариант 32-2009.

На основе микроконтроллера разработать устройство, выполняющее следующие функции:

Работа теплового максимально-дифференциального датчика. В устройстве предусмотреть 2 режима: «программирование» - кнопка SB нажата, и «работа» - кнопка SB отпущена. В режиме «программирование» задать БАЗОВОЕ значение, с которым потом будет сравниваться входное напряжение (в виде двоичного числа, кнопками SB7, SB6,…SB1). В режиме «работа» непрерывно считывать аналоговое значение, поступающее от чувствительного элемента (моделируется потенциометром). Срабатывание датчика (диод VD горит) должно происходит в 2-х случаях: аналоговое значение на входе больше БАЗОВОГО, или когда в течение 10 сек аналоговое значение на входе (конечное на время 10 сек) возросло в 2 раза, по сравнению с аналоговым значением на входе (исходным на время 0 сек).

9.2 ЛАБОРАТОРНЫЕ РАБОТЫ

9.2.1 Часть 1

РАБОТА 1. ЗНАКОМСТВО С УЧЕБНЫМ СТЕНДОМ УМК

1. Цель работы

Знакомство с устройством и принципом работы учебного стенда УМК.

2. Основные теоретические положения

Микропроцессорные системы (МПС) управления, несмотря на достаточно широкую область применения (а в этой связи и отличия в конструкции), имеют в своем составе общие элементы. Структура простейшей МПС (типичной для 8-и разрядных систем CISC архитектуры) представлена на рис. 1

Рис. 1

Шина адреса: предназначена для передачи адресов в системе, например для адресации ячеек памяти, адресации устройство ввода/вывода.

Шина данных: предназначена для передачи данных, например от микропроцессора к памяти, от устройств ввода/вывода к микропроцессору.

Шина управления: предназначена для передачи различных управляющих сигналов, например сигналов разрешения, сигналов синхронизации.

Блок памяти: микросхемы памяти типа ОЗУ, ПЗУ предназначенные для временного или постоянного хранения информации.

Микропроцессор: центральный элемент системы выполняющий функции управления работой всей системы, выполняющий различные арифметические и логические операции

Устройство ввода/вывода: комплекс устройств предназначенных для сопряжения МПС с внешней средой, например порты ввода/вывода.

Дисплей и клавиатура: простейшие внешние устройства, с которыми должна работать МПС.

Структура центрального элемента системы – микропроцессора представлена на рис. 2.

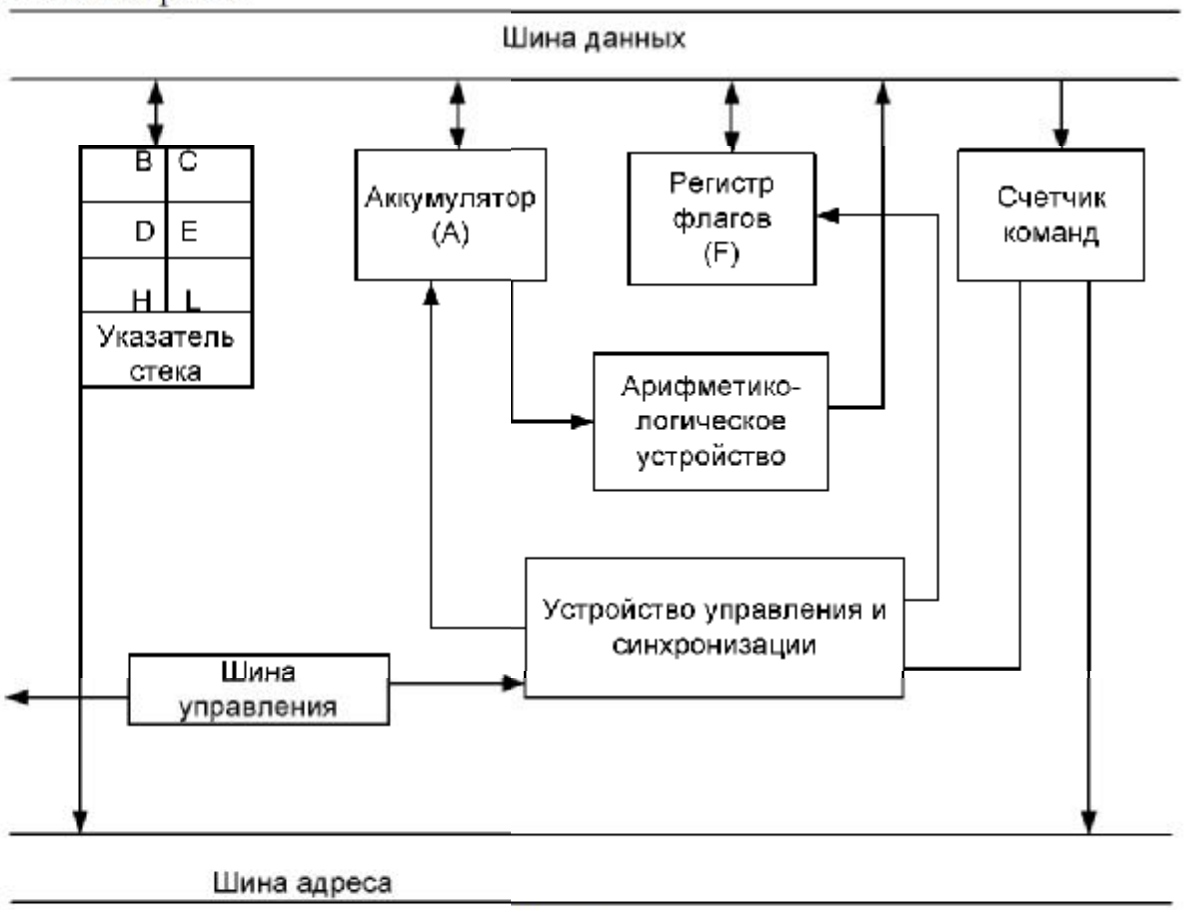


Рис.2

Аккумулятор (A) – основной регистр системы, используется при выполнении арифметических, логических операций и операций сдвига. Служит источником операнда (числа, участвующего в операции); в него помещается результат выполненной операции.

Регистры B, C, D, E, H, L образуют так называемый блок регистров общего назначения (РОН) – то есть, эти регистры могут использоваться для хранения, как данных, так и адресов. Они могут применяться как одиночные 8-разрядные регистры. В случаях, когда возникает необходимость хранить 16-разрядные двойные числа, они объединяются в пары BC, DE, HL.

Стек – специально организованная область памяти, используемая микропроцессором для временного хранения данных или адресов. Число, записанное в стек последним, извлекается из него первым (стек магазинного типа).

Указатель СТЕКа – регистр (16-разрядный) для хранения последнего по времени поступления адреса ячейки памяти.

Счетчик команд - специальный регистр, обеспечивающий смену команд, последовательное их выполнение. В него загружают адрес первой команды (программа – это и есть набор последовательных команд). После выбора из ОЗУ текущей команды, содержимое счетчика увеличивается на единицу и таким образом формируется адрес очередной команды (при отсутствии безусловных и условных переходов).

При обращении к памяти в качестве адреса может использоваться и содержимое любой пары регистров блока РОН.

Регистр флагов F (регистр признаков) предназначен для хранения определенных признаков, выявляемых в числе, которое представляет собой результат выполнения некоторых операций. Результат может быть положительным, отрицательным, равным нулю и другим. Эти признаки могут использоваться последующими командами как для изменения последовательности выборки команд из памяти, так и для модификации обрабатываемых данных.

Устройство управления и синхронизации формирует управляющие сигналы, под действием которых выполняются микрооперации в отдельных узлах.

Арифметическо-логическое устройство (АЛУ) предназначено для непосредственного выполнения простейших арифметических операций типа сложение, вычитание и логических.

Все операции в микропроцессоре выполняются в двоичной, системе счисления. Поэтому и все исходные данные, с которыми оперирует МПС, должны быть представлены в двоичной форме. Двоичная форма записи исходных данных и команд является единственной, которую понимает МПС.

Данные и команды в лабораторном УМК представляются и виде 8-разрядных двоичных чисел, называемых байтами и имеющих следующий формат:

D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0

(байт данных)

Например: 11001101; 10100010.

Однако для удобства и сокращения записи будем записывать их в шестнадцатеричной форме. Это соответствует и клавиатуре ввода. Тогда 8-разрядное двоичное число может быть представлено в виде 2-разрядного шестнадцатеричного числа. Например, двоичное число 11000101 будем записывать как C5. Перевод в двоичные, шестнадцатеричные и десятичные числа показан в табл. 1. Таким образом, 1 байт информации соответствует 8-разрядному двоичному числу или 2-разрядному шестнадцатеричному числу.

3. Описание лабораторной установки

В качестве лабораторной установки используется микропроцессорный комплект (УМК), представляющий собой законченную микроЭВМ. УМК предназначен для обучения микропроцессорной технике и основам программирования на базе микропроцессора КР580ИК80А.

Таблица 1 – Формы записи чисел

Двоичное число	16-ричное число	Десятичное число	Двоичное число	16-ричное число	Десятичное число
0000	0	0	10001	11	17
0001	1	1	10010	12	18
0010	2	2	10011	13	19
0011	3	3	10100	14	20
0100	4	4	10101	15	21
0101	5	5	10110	16	22
0110	6	6	10111	17	23
0111	7	7	11000	18	24
1000	8	8	11001	19	25
1001	9	9	11010	1A	26
1010	A	10	11011	1B	27
1011	B	11	11100	1C	28
1100	C	12	11101	1D	29
1101	D	13	11110	1E	30
1110	E	14	11111	1F	31
1111	F	15	100000	20	32
10000	10	16	100001	21	33

Основой любой микроЭВМ является микропроцессор, который производит все операции по обработке информации. Исходным состоянием микропроцессора является чтение информации по нулевому адресу ПЗУ. МП принимает это состояние после нажатия управляющей кнопки сброс «СБ».

В ПЗУ хранятся неизменяющиеся программы и данные.1 Кбайт ПЗУ (1 байт - одно двухразрядное шестнадцатеричное число) занимает программа "Монитор", обеспечивающая ввод информации с клавиатуры пульта оператора и вывод ее на дисплей. Еще 1 Кбайт ПЗУ используется для вспомогательных программ.

ОЗУ применяется для хранения изменяющихся программ и данных. ОЗУ занимает ячейки памяти с адреса 0800H и имеет емкость 2 Кбайта. (Символ H в конце адреса указывает, что число 0800 представлено в шестнадцатеричном коде).

Адрес 0800H является начальным адресом ОЗУ, куда могут записываться программы пользователя.

Пульт оператора (рис. 3) состоит из клавиатуры, шестиразрядного дисплея, световой индикации (позиции 1. 2. 3) и управляющих кнопок: сброс СБ, прерывание ПР, шаг ШГ, а также переключателей работа/шаг РБ/ШГ и команда/цикл КМ/ЦК.

Клавиатура состоит из 8 *директивных клавиш* (слева) и 16 *информационных клавиш* шестнадцатеричной системы (от 0 до F). Клавиши с 4/РН по F служат также для вызова регистров микропроцессора.

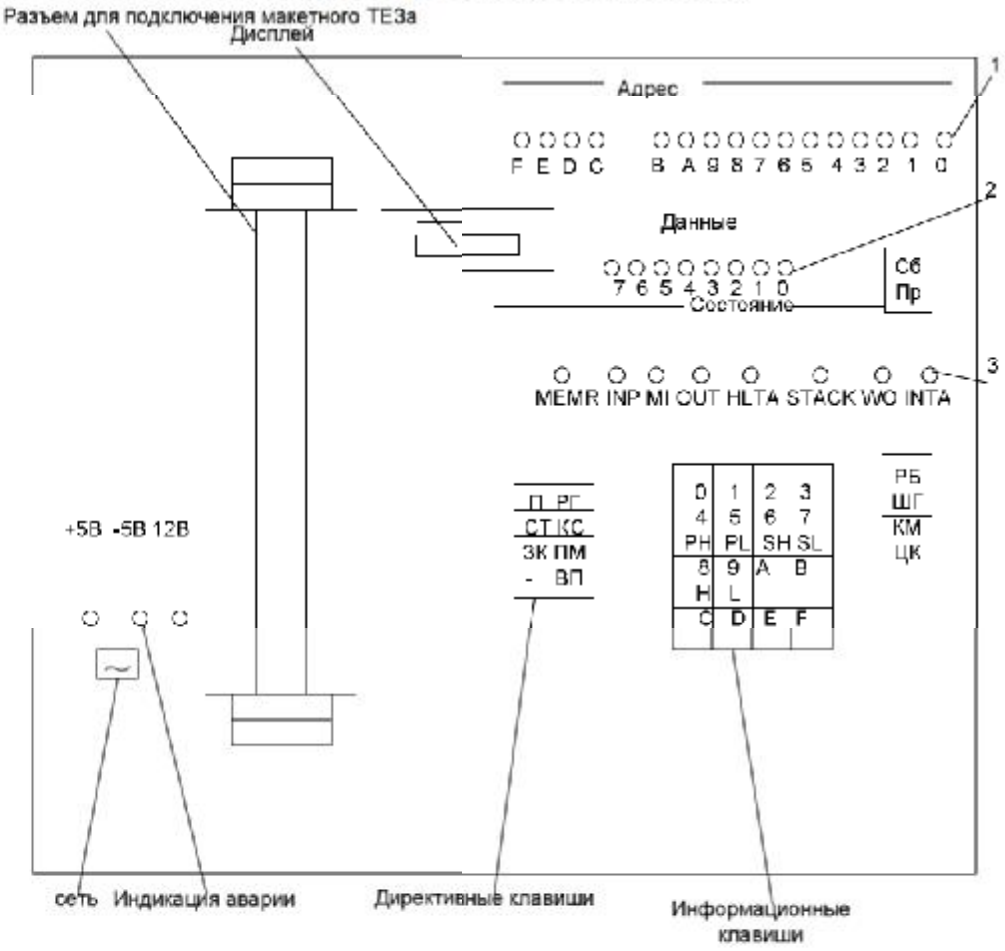


Рис.3

Описание клавиатуры

Директивные клавиши:

П – чтение и изменение содержимого памяти;

РГ – чтение и изменение содержимого регистров;

СТ – передача управления программе пользователя;

КС – определение контрольной суммы массива памяти;

ЗК – заполнение массива памяти константой;

ПМ – перемещение массива памяти в адресном пространстве;

_ – клавиша пробела служит для разделения нескольких переменных при вводе;

ВП – выполнить, означает конец директивы.

Информационные клавиши:

Клавиши с 0 по F служат для ввода чисел в шестнадцатеричном коде.

Клавиши с 4/РН по F служат для вызова регистров МП.

Регистры МП:

A – аккумулятор (регистр A);

B – регистр B;

C – регистр C;

D – регистр D;

E – регистр E;

H – регистр старших разрядов адреса;

L – регистр младших разрядов адреса;

F – регистр флагов;

SL(SPL) – младший байт указателя стека;

SH(SPH) – старший байт указателя стека;

PL(PCL) – младший байт счетчика команд;

PH(PCH) – старший байт счетчика команд.

Кнопки:

СБ – сброс;

ПР – прерывание;

ШГ – шаг.

Переключатели:

«РБ/ШГ» – работа/шаг;

«КМ/ЦК» команда/цикл.

Отображение вводимой и выводимой информации в шестнадцатеричном коде происходит шестиразрядном дисплее. Четыре разряда слева используются для отображения информации типа «Адрес», два справа – типа «Данные».

Для выполнения той или иной команды оператор должен ввести с клавиатуры соответствующую директиву и необходимые параметры. Результат выполненной команды будет отображен на дисплее.

Команда, задаваемая пользователем, состоит из директивы управления, параметров директивы и директивы ВП и может быть представлена в общем виде:

ДИР [ПАР1,_, ПАР2,_, ПАР3]ВП,

где

ДИР – соответствует нажатию одной из директивных клавиш: П, РГ, СТ, КС, ЗК, ПМ;

ПАР1, ПАР2, ПАР3, – числа, набираемые на информационной клавиатуре и являющиеся параметрами директив (возможно от одного до трех параметров в зависимости от директивы);

_ – клавиша пробела служит для разделения нескольких параметров при вводе;

ВП – клавиша «выполнить», означает конец команды.

Команды УМК, используемые в этой работе, приведены в табл. 2.

Таблица 2 – Формат и примеры команд УМК

Формат команд	Примеры	Пояснения
П, $А_н$, _, $Д_1$, $Д_2$,…, _, $Д_н$, ВП, где $А_н$ – начальный адрес ввода данных (4 цифры); $Д_1$, $Д_2$, … , $Д_н$, - данные (2 цифры)	П, 0803, _ , 21, _ , 3D, _, 2A, ВП	В ячейку памяти 0803Н записано число 21Н; в следующую ЯП (0804Н) –число 3DH и в ЯП 0805Н – число 2АН
РГ, И, Д, ВП, где И – идентификатор регистра	РГ, А, 27, ВП	В регистр А записано число 27Н
СТ, $А_1$, _ , $А_к$, ВП, где $А_1$ – начальный адрес $А_к$ – конечный адрес	СТ, 0801, _ , 0807, ВП	Выполняется программа пользователя, записанная в ячейках памяти с 0801Н по 0807Н. Результат выполненной программы высветится на двух правых индикаторах дисплея в шестнадцатеричном коде
КС, $А_1$, _ , $А_к$, ВП	КС, 0927, _ , 0945, ВП	На двух правых индикаторах высветится контрольная сумма содержимого ЯП с 0927Н по 0945Н
ЗК, $А_1$, _ , $А_к$, _ , К, ВП где К – константа (2 цифры)	ЗК, 0800, _ , 0805, _ , 12, ВП	В ячейки памяти с 0800Н по 0805Н включительно записана константа 12Н
ПМ, $А_1$, _ , $А_к$, _ , $А_н$, ВП. где $А_н$ – начальный адрес массива размещения	ПМ, 0927, _ , 093А, _ , 0806, ВП	Массив памяти, ограниченный адресами 0927Н и 093АН включительно, переписывается в область памяти, начиная с адреса 0806Н

4. Порядок выполнения работы

1. Заполнить массив ОЗУ последовательностью чисел арифметической прогрессии

$$a_n=a_1+(n-1)*d,$$

где

a_n – n-й член арифметической прогрессии;

a_1 – первый член арифметической прогрессии (задается преподавателем);

d=3 – шаг прогрессии;

n=1, … , 10 – количество членов арифметической прогрессии.

Например, при a_1=2 будем иметь следующую последовательность чисел: 2, 5, 8, 11, 14, 17, 20, 23, 26, 29.

Для того чтобы ввести числа в УМК, необходимо перевести их в шестнадцатеричные коды и занести каждое в ячейку памяти, начиная, например, с адреса 080111. При этом необходимо заполнить таблицу по форме 1.

Форма 1

№ п/п	Адрес ячейки памяти	Десятичное число	Шестнадцатеричное число	Двоичное число
1	081H	2	2	10
…	.	.	.	.
…	.	.	.	.
…	.	.	.	.
10	080AH			
		сумма =	сумма =	сумма =

Включить УМК в сеть.

Нажать кнопку «~». При этом в крайней левой позиции дисплея должен появиться знак «-», свидетельствующий о том, что УМК готов к работе. В противном случае нажать управляющую кнопку «СБ».

Повторное включение УМК производить не менее чем через 10 с после выключения.

Записать в отчет формат команды для занесения арифметической прогрессии в память УМК. Для нашего примера формат команды будет выглядеть так:

П, 0801, _ , 02, _ , 05, _ , 08, _ , … , _ , 1A, _ , 1D, ВП.

Нажать клавиши на УМК согласно команде:

- нажать директивную клавишу «П» (при этом знак «-» исчезнет с экрана дисплея), а затем набрать адрес нужной ЯП с помощью информационных клавиш, в качестве адреса на экране дисплея слева фиксируются последние 4 введенные цифры;

- нажать клавишу «_» - при этом УМК высветит на двух правых индикаторах дисплея содержимое указанной ячейки, для изменения содержимого ЯП наберите на информационных клавишах ее новое значение, в качестве данных фиксируются последние две введенные цифры;
- нажать клавишу «_» при этом на дисплее инициируется адрес и содержимое следующей ячейки памяти: ввести данные, соответствующие этой ЯП, и, нажав клавишу «_», перейти к следующей ячейки памяти, для перехода к следующей ЯП без изменения его содержимого необходимо только нажать клавишу «_», закончив ввод данных, нажать клавишу «ВП».

При неправильной работе с клавиатурой в крайней правой позиции дисплея высветится знак «?». В этом случае следует нажать кнопку «СБ» и повторить команду, начиная с директивной клавиши.

Проверить произведенную запись. Для этого нажать клавишу «П», вызвать первую ЯП (0801Н) и затем многократным (10 раз) нажатием только клавиши «_», вызвать все последующие ячейки до 080АН и проверить правильность записи содержимого этих ячеек. Если в содержимом какой-нибудь ячейки имеется ошибка, перед очередным нажатием клавиши «_» необходимо внести правильное значение. Проверив содержимое последней ячейки (080АН), нажать клавишу «ВП».

2. Определить контрольную сумму данного массива ОЗУ.

Формат команды:

КС, A1 , _, Ak, ВП,

где

КС, _, ВП – директивные клавиши на лицевой панели УМК;

A_1, A_k – соответственно начальный и конечный адреса суммируемого массива памяти.

Записать в отчет формат команды со своими данными.

Проверить готовность УМК к выполнению следующей директивы: в крайнем левой позиции дисплея должен быть знак «-». В противном случае нажать кнопку «СБ»

Нажать последовательно клавиши:

КС, 0801, _, 080А, ВП.

При этом на двух правых индикаторах дисплея высвечивается контрольная сумма массива памяти с адреса 0801Н по адрес 080АН включительно в шестнадцатеричном коде. Результат занести в форму 1.

Определить сумму арифметической прогрессии по формуле

$$\frac{(a_1 + a_n)}{2} * n$$

где

a_1 – первый член арифметической прогрессии;

a_n – последний член арифметической прогрессии;

n – количество членов арифметической прогрессии.

Проверить совпадение теоретического и практического результатов.

3. Заполнить массив ОЗУ константой.

Константа задается преподавателем.

Формат команды:

ЗК, A_1, _ , A_K , _ , К, ВП,

где

ЗК, _ , ВП – директивные клавиши;

A_1 и A_K – соответственно начальный и конечный адреса массива памяти, в который следует занести константу;

К – константа.

Принять A_1=0918Н, A_K=0921Н.

Допустим К=А5Н, тогда для заполнения массива памяти константой А5Н необходимо нажать следующие клавиши:

ЗК, 0918, _ , 0921, _ , А5.ВП.

Записать в отчет формат команды с заданной преподавателем константой.

Произвести заполнение массива памяти с адреса 0918Н по адрес 0921Н константой, указанной преподавателем.

Проверить правильность произведенной в ОЗУ записи. Для этого нажать клавишу «П», вызвать первую ЯП (0918 Н) и затем, нажимая клавишу «_» проверить содержимое ячеек до 0912Н включительно. Закончив проверку, нажать клавишу «ВП». Записать в отчет все адреса, в которых находиться константа.

Записать в отчет константу в шестнадцатеричном, десятичном и двоичном кодах и теоретически определить сумму массива с адреса 0918Н по 0921Н включительно.

4. Переместить массив памяти с адреса 080111 по адрес 080АН включительно в область памяти, начиная с адреса 0922Н.

Формат команды:

ПМ, A_1, _ , $A_к$, _ , $A_н$, ВП,

где

ПМ – директивная клавиша «Перемещение массива памяти в адресном пространстве»;

A_1=0801Н и $A_к$=080АН – соответственно начальный и конечный адреса переменного массива;

$A_н$=0922Н – начальный адрес массива размещения.

При этом массив памяти, ограниченный адресами A_1 и $A_к$ включительно, переписывается в области памяти, начиная с адреса $A_н$. Массивы перемещения и назначения не должны перекрываться. В противном случае происходит потеря информации.

Нажать последовательно клавиши:

ПМ, 0801, _ , 080А, _ , 0922, ВП.

Проверить правильность перемещения массива. Для этого нажать клавишу «П», вызвать ЯП=0922Н и затем, нажимая клавишу «_», проверить содержимое ячеек до 092ВН включительно. Нажать «ВП».

5. Определить контрольную сумму массива памяти с адреса 0918Н по адрес 092ВН.

Самостоятельно записать формат этой команды с конкретными адресами (аналогично тому, как было сделано при выполнении п.2).

Произвести вычисление контрольной суммы на УМК. Записать в отчет результат, учитывая, что ответ на УМК дан по модулю 256 без учета переполнения. Поскольку для «данных» у нас имеется всего два разряда 16-ричного кода, мы сможем увидеть на дисплее только два младших разряда полученного результата.

Например, мы должны получить результат 1258:

1258=4ЕАН, то есть 1258=4*256+14*16+10.

Значит, на экране дисплея мы увидим только два младших разряда шестнадцатеричного числа 4ЕА, то есть ЕА.

Определить контрольную сумму теоретически и сравнить с ответом на УМК.

5. Содержание отчета

1. Название работы.
2. Цель работы.
3. Структурная схема МПС и МП.
4. Используемые в работе формулы.
5. Таблица по форме 1.
6. Формат всех выполняемых команд.
7. Теоретические и практические результаты вычислений.

Вопросы для самопроверки.

1. Что такое аккумулятор?
2. Чем отличается ОЗУ от ПЗУ?
3. Привести пример слова данных размером в 1 байт.
4. Объяснить значение клавиш КС, ПМ, РГ.
5. Для чего регистры объединяются в пары?
6. Перевести число 136 в двоичный и шестнадцатеричный коды.

Литература: [1], с. 5..13; 16..26; 30..37; 60..71; 74..79.

РАБОТА 2. ЗАПИСЬ И ВЫПОЛНЕНИЕ ПРОСТЫХ ПРОГРАММ

1. Цель работы

Знакомство с системой команд микропроцессора КР580ИК80, запись и исследование выполнения простых программ.

2. Основные теоретические положения

Общий принцип работы микропроцессорного устройства заключается в следующем. Из микропроцессора на шину адреса счетчик выдает адрес очередной команды. Считанная по этому адресу из памяти (например, из ПЗУ) команда поступает на шину данных и принимается в микропроцессор, где она исполняется. В счетчике команд МП автоматически формируется адрес следующей команды. После окончания исполнения данной команды на шину адреса поступает адрес следующей команды и т.д.

При записи команд все числа представляются в шестнадцатеричной системе счисления. Однако в таком виде запомнить команды очень трудно, поэтому чаще их записывают сокращенными английскими словами. Такая запись составляет язык низшего уровня – Ассемблер. Перевод команд Ассемблера в шестнадцатеричный код для нашего УМК производится по таблицам вручную, а дальнейший перевод в двоичный код (единственно понятный для микропроцессора) – программой «Монитор».

Язык Ассемблера упрощает запись команд, обеспечивает лучший обзор программы и внесение исправлений в записанную при этом языке программу, облегчает поиск в ней ошибок, позволяет существенно уменьшить зависимость программы от типа микропроцессора, т.к. при переходе на другой МП изменяется лишь таблица соответствия.

Команды могут быть одно-, двух- и трехбайтовыми.

Большинство команд являются однобайтовыми и содержат в себе только код операции. Например:

78 – содержимое регистра В переслать в регистр А.

81 – сложить содержимое аккумулятора с содержимым регистра А.

В двухбайтовой команде в первом байте указывается вид выполняемой операции, во втором приводится число, являющееся операндом при выпол-

нении операции либо номером устройства ввода или вывода при обмене данными. Например:

06 А5 – здесь: 06 – код операции «Переслать в регистр В»;

А5 – шестнадцатеричное число (операнд), которое следует переслать в регистр В.

Байты трехбайтовой команды имеют следующее назначение:
- первый байт – код операции;
- второй байт – младшие разряды;
- третий байт – старшие разряды.

Второй и третий байты используются для указания двухбайтового адреса команды или адреса ячейки ОЗУ, содержимое которой является операндом, или двухбайтового операнда.

Например,

3А 2В 09,

здесь

3А – код операции «Переслать в аккумулятор содержимое ЯП, адрес которой (092В) указан во втором и третьем байтах команды»;

2В – младшие разряды адреса ЯП ОЗУ;

09 – старшие разряды адреса ЯП ОЗУ.

Все команды МП КР580ИК80 раздсляются на 5 групп.

Команды пересылки данных осуществляют обмен данными между регистрами и памятью. Основные команды этой группы имеют обозначение MOV или MVI. По команде MOV происходит пересылка содержимого одного регистра в другой или содержимого регистра в память, или содержимого памяти в регистр. Например, по команде MOV B,H происходит передача числа из регистра H в регистр B. По команде MVI осуществляется непосредственная запись числа в аккумулятор, регистр или память (то есть само число непосредственно присутствует в команде). Например, команда MVI B, 35 позволяет непосредственно записать число 35 (в шестнадцатеричном коде) в регистр B.

При записи на Ассемблере аккумулятор обозначается буквой A, оперативное запоминающее устройство (память) – M, ячейка памяти, адрес которой записан в паре регистров H, L, т.е. H, L – сдвоенный регистр косвенного адреса (M). Шестнадцатеричные коды некоторых команд пересылки данных приведены в таблице 3.

Команды арифметические предназначены для выполнения сложения, вычитания, увеличения или уменьшения содержимого регистров или ячеек памяти на единицу.

Команды сложения и вычитания всегда выполняются между первым числом, находящимся в аккумуляторе, и вторым числом, находящимся в регистре или в памяти. Результат помещается в аккумулятор.

Шестнадцатеричные (машинные) коды некоторых арифметических команд приведены в таблице 4.

Таблица 3 – Команды пересылки данных УМК

Ассемблерный (мнемокод команды)	**16-ричный код (машинный код)**	**Количество байт в команде**	**Комментарий**
MOV A, B	78	1	Содержимое регистра B переслать в регистр A: A ← B
MOV C, A	4F	1	Содержимое аккумулятора переслать в регистр C: C ← A
MOV M, A	77	1	Содержимое аккумулятора переслать в ЯП, адрес которой указан в регистрах H, L: ЯП(H, L) ← A
MOV B, M	46	1	Содержимое ЯП, адрес которой указан в паре регистров H, L, переслать в регистр B: B ← ЯП(H, L)
MVI A, 2EH	3E 2E	2	Шестнадцатеричное число 2E переслать в A: A ← 2E
MVI B, A5H	06 A5	2	Шестнадцатеричное число A5 переслать в регистр B: B ← A5
MVI M,36H	36 36	2	Шестнадцатеричное число 36 переслать в ЯП, адрес которой указан в паре регистров H, L: ЯП(H, L) ← 36
LDA 092BH	3A 2B 00	3	Содержимое ЯП, адрес которой указан во втором и третьем байтах команды, пересылается в A, т.е. содержимое ЯП с адресом 092B пересылается в A: A ← ЯП(092B)
STA 0803H	32 03 08	3	Содержимое A пересылается в ЯП, адрес которой указан во втором и третьем байтах команды (в данном примере в ЯП с адресом 0803): ЯП(0803) ← A

Таблица 4 – Арифметические команды УМК

Ассемблерный (мнемокод команды)	16-ричный код (машинный код)	Количество байт в команде	Комментарий
ADD B	80	1	Сложение содержимого аккумулятора с содержимым регистра B:
ADD M	86	1	Сложение содержимого A с содержимым ЯП, адрес которой указан в паре регистров HL: A ← [A+ЯП(H,L)]
ADI 32H	C6 32	2	Сложение содержимого A с числом 32 (шестнадцатеричный код), которое указано во втором байте команды:
SUB C	91	1	Содержимое регистра C вычитается из аккумулятора A ← (A-C)
SUI C2H	Д6 C2	2	Содержимое второго байта команды вычитается аккумулятора: A ← (A-C2H)
INR M	34	1	Содержимое ЯП, адрес которой указан в паре регистров H,L увеличивается на единицу: ЯП(H,L) → [ЯП(H,L)+1]
DCR A	3D	1	Содержимое регистра A уменьшается на единицу: A ← (A-1)
DAD B	09	1	Содержимое пары регистров B, C складывается с содержимым регистров H,L; результат помещается в пару регистров H,L: (HL) → [(H,L)+(B,C)]

Логические команды выполняют логические операции над данными, хранящимися в регистрах и ячейках памяти.

Основными логическими операциями (и них отсутствует перенос из разряда в разряд) являются И, ИЛИ, исключающие ИЛИ, сравнение. Некоторые логические команды приведены в табл. 5.

Таблица 5 – Логические команды

Ассемблерный (мнемокод команды)	16-ричный код (машинный код)	Количество байт в команде	Комментарий
ANI C8H	E6C8	2	Над содержимым аккумулятора и второго байта команды выполняется логическая операция «И»: A ← (A^C8H)
XRI C7H	EE C7	2	Над содержимым A и второго байта команды выполняется логическая операция «исключающее ИЛИ»:
RLC	07	1	Содержимое A сдвигается влево на одну позицию. Содержимое старшего бита заносится в младший бит: $A_{n+1} \leftarrow A_n$; $A_0 \leftarrow A_7$
CMA	2F	1	Дополнение аккумулятора (содержимое A инвертируется): A ← A'
CMP B	B8	1	Содержимое регистра B сравнивается с содержимым регистра A. A не изменяется

Команды переходов (условных и безусловных), *вызова подпрограмм* и *возвращении из подпрограмм* изменяют нормальную последовательность исполнения команд.

Команды ввода-вывода, управления и работы со СТЕКом предназначены для выполнения операций ввода-вывода, работы со СТЕКом, управления флагами, разрешения и прерываний.

Напишем простейшую программу для следующей формулы:

$$X\text{^}C8H+Y,$$

где

X, Y – содержимые ячеек памяти с адресами 0B00H и 0B01H соответственно;

C8H – заданное число в шестнадцатеричном коде ($C8_{16}=200_{10}$);

^ – условное обозначение логической операции «И» (логическое умножение).

В соответствии с этой формулой требуется принять из адреса) 0B00H число X, логически умножить его на число C8H и сложить с числом Y из ячейки памяти 0B01H.

Результат поместить в ЯП 0B02H

Схема алгоритма решения данной задачи, построенная в операциях, выполняемых микропроцессором серии КР580, приведена на рис. 4.

Рассмотрим операции, выполняемые в каждом из блоков схемы алгоритма.

Блок 1 производит прием в аккумулятор А содержимого ячейки ОЗУ с адресом 0B00HI. Эта операция может быть выполнена командой прямой загрузки аккумулятора, мнемокод этой команды – LDA [адрес] (в нашем примере LDA 0B00H).

Блок 2 производит операцию «И» над содержимым аккумулятора и числом C8H. Результат помещается в аккумулятор. Мнемокод команды – ANI C8H. В этой команде число C8H входит в саму команду и заносится с клавиатуры.

Блок 3 служит для того, чтобы произвести сложение промежуточного результата, оставшегося после предыдущей операции (блок 2) в аккумуляторе, с каким-либо числом из ЯП. Используя команды Ассемблера, мы можем сделать это, либо поместив число из ОЗУ непосредственно в какой-то из регистров, либо использовав пару регистров H,L (при записи на Ассемблере они обозначаются «M») для записи в них адреса операнда.

Согласно этому варианту блок 3 производит запись в регистры H, L адреса 0B01H, в котором находится следующий операнд. Мнемокод такой команды LXI H[адрес] – непосредственная загрузка пары регистров H, L. В нашем примере команда записывается следующим образом: LXI H 0B01H.

Блок 4 производит сложение содержимого А с содержимым ячейки ОЗУ M, адресом которой служит содержимое пары регистров H, L. Операция сложения выполняется командой ADD M.

Блок 5 пересылает в ЯП 0B02H полученную в А сумму. Мнемокод команды – STA [адрес] (в нашей программе STA 0B02H).

Для записи программы в память микроЭВМ необходимо перевести мнемокоды команд Ассемблера в машинные коды, где все команды и числа представляются в шестнадцатеричной системе счисления.

Команды в программе могут быть одно-, двух- или трехбайтовые и должны в ОЗУ занимать соответственно один, два или три адреса, т. к. в каждую ячейку ОЗУ можно поместить только один байт информации. Например, наша трехбайтовая команда LDA 0B00H располагается в трех ячейках ОЗУ.

Размещение команд, произведено начиная с ячейки, имеющей адрес 0800H (программа 1).

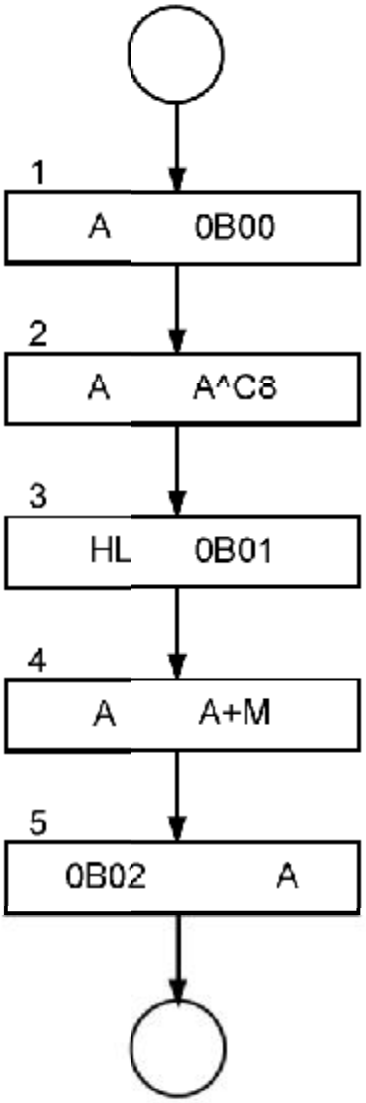


Рис. 4 – Алгоритм программы

Программа 1

№ блока	Адрес	Мнемокод	Число байт в команде	Машинный код	Комментарий
1	0800	LDA 0B00H	3	3A	Код команды LDA: содержание ЯП, адреса которых указаны во втором и третьем байтах команды загрузить в А
	0801			00	Младший байт адреса
	0802			0B	Старший байт адреса

2	0803	ANI C8H	2	E6	Код команды ANI: над содержимым A и второго байта команды выполняется логическая операция «И»
	0804			C8	Второй байт команды (операнд)
3	0805	LXI H, 0B01H	3	21	Код команды LXI H: Непосредственная загрузка пары регистров H, L
	0806			01	Младший байт адреса, который пересылается в регистр L
	0807			0B	Старший байт адреса, который пересылается в регистр H
4	0808	ADD M	1	86	Код команды ADD M: содержимое ЯП, адрес которой в регистрах H,L складывается с содержимым A; результат помещается в A
5	0809	STA 0B02H	3	32	Код команды STA: содержание A пересылается в ЯП, адрес которой указан во втором и третьем байтах команды
	080A			02	Младший байт адреса
	080B			0B	Старший байт адреса
	080C	HLT	1	76	Код команды HLT: останов

Предварительную запись программ удобно проводить в более компактной форме. В программе указывается начальный адрес команды и при этом понимается, что в зависимости от длинны команды (одно-, двух- или трехбайтовая) в памяти будут занимать от одной до трех последовательных ячеек (адресов). При такой записи в левом столбце указываются лишь адреса

первого байта команд. Это позволяет сократить объем при описании программ и сделать более простым их анализ.

В укороченном виде наша программа примет вид программы 2.

Программа 2

Адрес	Машинный код	Мнемокод	Комментарий
0800	3A 000B	LDA 0B00H	Содержимое ЯП с адресом 0B00 переслать в аккумулятор
0803	E6 C8	ANI C8H	Над содержимым A и шестнадцатеричным числом C8 выполнить логическую операцию «И». Результат поместить в A
0805	21 010B	LXI H, 0B01H	Загрузить в пару регистров H,L адрес 0B01
0808	86	ADD M	Сложить содержимое A с содержимым ЯП, которая указана в паре регистров H,L (M) Результат поместить в A
0809	32 020B	STA 0B02H	Содержимое A переслать в ЯП с адресом 0B02
080C	76	HLT	Останов

Результат программы можно проанализировать по состоянию регистра флагов F. Как и все регистры МП серии КР580, регистр F имеет 8 разрядов, однако 3 разряда не используются, а 5 разрядов этого регистра устанавливаются по определенному правилу в соответствии с выполнением последней команды.

Признаки результата размещаются в регистре флагов следующим образом:

Пять разрядов регистра F имеют следующие значения.

Разряд знака S – SIGN. В него записывается 1, ели при выполнении арифметической или логической команды в старшем, седьмом, разряде аккумулятора записана 1, в противном случае в разряд записывается 0.

Разряд нулевого результата Z – ZERO. Если результат равен нулю, в него записывается 1, в противном случае Z=0.

Дополнительный разряд переполнения AC – AUX. CARRY. В него записывается 1, если при выполнении команд в аккумуляторе возникает единица переноса из третьего разряда чисел (перенос между тетрадами байта).

Разряд четности P – PARITY. В него записывается 1, если при выполнении команды количество единиц в разрядах аккумулятора будет четным. В противном случае P=0.

Разряд переполнения C – CARRY. В него записывается 1, если при выполнении арифметической команды или команды сдвига было переполнение аккумулятора, в противном случае C=0.

Рассмотрим подробнее процесс выполнения команды. Он разбивается на циклы, обозначаемые M_1, M_2, M_3, M_4, M_5. Выполнение каждой команды занимает от одного до пяти машинных циклов. Машинный цикл требуется всякий раз, когда микропроцессор обращается к памяти или устройствам ввода-вывода (портам).

Если команда занимает несколько байтов, то для выбора каждого байта требуется по одному машинному циклу.

Например, команда LDA 0B00 (см. выше) занимает три ячейки памяти 0800, 0801, 0802, в которых хранятся коды 3A, 00, 0B. Выполнение команды начинается, когда на программном счетчике имеется число 0800. В первом машинном цикле по адресу 0800 выбирается первый байт команды 3A, передается в микропроцессор, где производится дешифрация этого кода. Во втором машинном цикле из следующей ячейки 0801 выбирается число 00 и записывается в регистр L. В третьем машинном цикле из ячейки 0802 выбирается число 0B и записывается в регистр H. Таким образом, в паре регистров H, L сформировался адрес ЯП, содержимое которой по команде 3A необходимо передать в аккумулятор. Поэтому в четвертом машинном цикле число из ячейки 0B00 передается в аккумулятор. На этом выполнение команды заканчивается и к содержимому счетчика прибавляется единица, то есть микропроцессор переходит к выполнению следующей команды. Первым машинным циклом при извлечении любой команды является цикл M_1 – выбор первого байта команды. Самые длинные по времени исполнения команды выполняются в пять циклов.

Каждый цикл включает в себя несколько тактов, обозначаемых T_1, T_2, T_3, T_4, T_5. Циклы могут содержать от трех до пяти тактов. Первые три такта во всех циклах используются для организации обмена с памятью и УВВ, такты T_4 и T_5 (если они присутствуют в цикле) – для выполнения внутренних операции и микропроцессоре.

Рассмотрим цикл M_1. В такте T_1 содержимое счетчика команд выдается на шину адреса, адрес принимается памятью, где начинается процесс чтения байта команды из указанной ячейки. В такте T_2 проверяется наличие сигнала

(уровня логической единицы) на входе ГОТОВНОСТЬ. Этот сигнал подается на вход микропроцессора через интервал времени, достаточный для завершения процесса чтения из памяти. Если на входе ГОТОВНОСТЬ сигнал отсутствует (действует уровень логического нуля), то микропроцессор устанавливается в режим ожидания до тех пор, пока не появится сигнал на входе готовность. С приходом этого сигнала микропроцессор выходит из режима ожидания, переходя в такт T_3. в этом такте выданный из памяти байт команды с шины данных принимается в микропроцессор. В такте T_4 анализируется принятый байт команды и выясняется, нужны ли дополнительные обращения в оперативную память. Если такие обращения не требуются (команда однобайтовая и операнды находятся в регистрах микропроцессора), то в этом же такте T_4 либо с использованием дополнительного такта T_5 выполняется предусматриваемая командой операция.

Если необходимы дополнительные обращения в оперативную память, то после такта T_4 цикл M_1 завершается и происходит переход к циклу M_2.

Информация о состоянии микропроцессора (о том, какой именно машинный цикл выполняется в данный момент) фиксируется в регистре состояний в начале каждого машинного цикла. В зависимости от состояния данного регистра формируются сигналы, управляющие работой всей микроЭВМ. В таблице 6 приведены возможные состояния микропроцессора.

Таблица 6 – Содержание регистра состояний МП

Состояние МП (машинный цикл)	**Байты регистра состояний**							
	D_7	**D_6**	**D_5**	**D_4**	**D_3**	**D_2**	**D_1**	**D_0**
	MEMR	**INP**	**MI**	**OUT**	**HLTA**	**STACK**	**WO**	**INTA**
Выбор первого байта команды	1	0	1	0	0	0	1	0
Чтение памяти	1	0	0	0	0	0	1	0
Запись в память	0	0	0	0	0	0	0	0
Чтение стека	1	0	0	0	0	1	1	0
Запись в стек	0	0	0	0	0	1	0	0
Ввод	0	1	0	0	0	0	1	0
Вывод	0	0	0	1	0	0	0	0
Прерывание	0	0	1	0	0	0	1	1
Останов	1	0	0	0	1	0	1	0

Прерывание в останове	0	0	1	0	1	0	1	1

Определение битов регистра состояния

INTA – сигнал подтверждения запроса прерывания;
WO – в текущем машинном цикле выполняется запись в память или операция вывода;
STACK – означает наличие в шине адреса содержимого указателя СТЕКа;
HLTA – сигнал подтверждения команды HLT (останов);
OUT – в текущем машинном цикле выполняется операция вывода;
MI – текущий машинный цикл служит для выборки первого байта команды;
INP – в текущем машинном цикле выполняется операция ввода;
MEMR – в текущем машинном цикле будет производиться чтение памяти.

3. Порядок выполнения работы

1. Исследовать программу 1.

Включить УМК в сеть. Проверить готовность УМК к работе: в крайней левой позиции дисплея должен появиться знак «–».

Занести программу 1 в память УМК. Формат команды для занесения этой программы в память:

П, 0800, _ , 3A, _ , 00, _ , 0B, _ , …, 76, ВП.

Записать по адресам 0B00 и 0B01 числа по указанию преподавателя. Формат команды:

П, 0B00, _ , X, _ , Y, ВП,

где X, Y – шестнадцатеричные числа, которые необходимо записать в соответствующие ЯП.

Осуществить пуск программы 1.

Формат команды на выполнение программы:

СТ, 0800, _ , 080C, ВП

В соответствии с нашей программой результат будет помещен в ячейку памяти 0B02H. Поэтому для прочтения результата необходимо нажать следующие клавиши:

П, 0B02H, ВП

Занести результат в форму 2.

Форма 2

Исходные данные и результаты выполнения программы для вариантов							
Параметр программы	**Адрес**	**1**	**2**	**3**	**4**	**5**	**6**
X	0B00	C2	BA	36	2E	73	E4
Y	0B01	F6	6C	3E	17	E4	A5
Результат	0B02						
Регистр	F						

Получить этот результат вручную по формуле (1). Для этого перевести исходные данные в двоичный код и произвести соответствующие вычисления. Сравнить теоретические и практические результаты.

2. Проанализировать полученный результат по состоянию регистра флагов F.

Для этого необходимо нажать последовательно клавиши: РГ, F; записать содержание этого регистра в шестнадцатеричном коде, перевести это число в двоичный код и проанализировать.

Например, после нажатия клавишей РГ, F на дисплее справа высветилось число 92Н. Переведем его в двоичный код, и поставим в соответствие с битами регистра словосостояния:

92H=	1	0	0	1	0	0	1	0
Биты	S	Z		AC		P		C

Отсюда S=1 – старший разряд (седьмой) результата равен 1;
Z=0 – результат не равен нулю;
AC=1 – был перенос из третьего разряда;
P=0 – количество единиц в разрядах результата нечетное;
C=0 – переноса из старшего разряда не было.

3. Исследовать процесс выполнения программы 1 в пошаговом режиме работы УМК.

Устройство пошагового выполнения программ, переводящее МП в состояние «Ожидание» либо в каждом рабочем цикле (поцикловый шаг), пошаговый режим работы вызывается переключателем «РБ/ШГ», выбор величины шага – переключателем «КМ/ЦК» (см. рис.3). Для первого и последующего шага необходимо нажать кнопку «ШГ». При этом после выполнения очередного шага на световой индикации отображается состояние адресной шины, шины данных и регистра состояний микропроцессора в двоичном коде.

Нажать переключатели «РБ/ШГ» и «КМ/ЦК». При этом загорится световая индикация на УМК. Проверить готовность УМК к работе – наличие знака « - « в крайней левой позиции дисплея.

Произвести пуск (старт) программы 1:

СТ, 0800, _ , 080C, ВП.

При этом на световой индикации будет отображена следующая информация: (горящий светодиод соответствует 1 двоичного кода, не горящий – 0):

адрес – 0000 1000 0000 0000, что соответствует 0800H;

данные – 0011 1010, что соответствует 3AH;

состояние – 10100010 соответствует состоянию МП, при котором производится выбор первого байта команды из ЯП ОЗУ с адресом 0800.

Далее, нажимая кнопку шаг «ШГ», осуществить пошаговое выполнение программы 1, заполняя при этом форму 3.

Форма 3

Адрес в 16-ричном коде	Данные в 16-ричном коде	Состояние в двоичном коде	Комментарий о регистре состояния	Количество байт в команде	Количество циклов в команде
0800	3A	10100010	Выбор первого байта команды		
0801	00	10000010	Чтение памяти		
…	…	…	…	…	…
080C					

Заполняется 4-й столбец, 5-й столбец нам известен из программы 1.

Поскольку первым машинным циклом (регистр состояний фиксирует именно машинные циклы) при выполнении любой команды является «Выбор первого байта команды», то после заполнения первых четырех столбцов формы не составит труда отделить команды друг от друга и сосчитать количество циклов в данной команде.

Необходимо обратить внимание на последовательность передачи и преобразования информации в микроЭВМ при выполнении каждой команды.

4. Заменяя в программе 1 команду AN1 C8 на команды из табл. 7 (по указанию преподавателя), исследовать результат выполнения преобразованных программ.

Написать новую программу в соответствии с внесенными изменениями (за основу взять программу 1).

Таблица 7

№ варианта	Мнемокод команды	Машинный код	Комментарий
1	INR A	3C	Увеличение содержимого регистра A на 1: A←(A+1)
2	DCR A	3D	Уменьшение содержимого регистра A на 1: A←(A-1)

3	SUI 3BH	D6 3B	Непосредственное вычитание второго байта команды из содержимого аккумулятора. Результат помещается в аккумулятор: A←(A-3BH)
4	CMA	2F	Дополнение аккумулятора (содержимое А инвертируется: бит, равный 1, становится равным 0; бит равный 0, - единице): A←(A)'
5	XRI C7H	EE C7	Непосредственная операция «исключающее ИЛИ» (неравенство).
6	RLC	07	Циклический сдвиг влево. Содержимое А сдвигается влево на одну позицию. Содержимое самого старшего бита заносится в младший бит и бит флага переноса (разряд С регистра F): $A_{n+1} \leftarrow A_n$ $A_0 \leftarrow A_7$ $C \leftarrow A_7$

Примечание: Операция «дополнение» (инвертирование) используется для получения дополнительного кода числа при вычитании двоичных чисел.

Следует учесть, что команда ANI C8 – двухбайтовая и занимает в ОЗУ две ЯП 0803 и 0804. Если используемая команда окажется однобайтовой, то для устранения образовавшегося разрыва в программе по адресу 0804 следует поместить команду NOP (нет операции), ее машинный код – 00.

Ввести: в УМК измененную часть программы. Для этого необходимо нажать следующие клавиши:

П. 0803, __ , (...),__,(...). ВП,

где (...) - данные, соответствующие варианту новой программы.

Осуществить пуск программы. Световая индикация должна быть выключена (переключатели «РБ/ШГ» и «КМ/ЦК» – отжаты).

Формат команды:

СТ, 0800,__, 080C, ВП.

Результат занести в форму 4; здесь X – циклический сдвиг влево.

Форма 4

№ п/п	Формула новой программы	X, 0B00	Y, 0B01	Результат, 0B02	Теоретический
1	X+1+Y				
2	X-1+Y				
3	X-3B+Y				
4	X'+Y				
5	X ⊕ C7 + Y				

Получить результат новой программы вручную и занести его в форму 4.

4. Содержание отчета

1. Название и цель работы.
2. Алгоритм программы 1.
3. Программы (1 и новые).
4. Таблицы с результатами исследований.
5. Теоретические расчеты.
6. Форматы команд.

Вопросы для самопроверки

1. Привести примеры одно-, двух-, трехбайтовых команд
2. Определить количество байт в следующих командах:
   STA 0920H;
   MOV M,B;
   SUI 2AH.
3. При выполнении каких команд программы 1 задействуются разряды регистра F?
4. Перечислите признаки разряда регистра F.
5. Почему не всегда совпадает количество байт и количество циклов в команде?
6. Зачем нужен программный счетчик?
7. Что такое флаг нуля?
8. Что такое флаг переноса?
9. Напишите программу увеличения на 5 числа, записанного по адресу 0900H, и записи результата по адресу 0901H.

Литература: [1],с. 79…102, 110…120; [2], с. 19,20; 28…29.

РАБОТА 3. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ТИПОВЫХ УПРАВЛЯЮЩИХ ВОЗДЕЙСТВИИ И ВЫЧИСЛИТЕЛЬНЫХ ПРОЦЕДУР

1. Цель работы

Ознакомление с принципами формирования микропроцессором управляющих воздействий различными технологическими устройствами, их программирования, а так же программной реализацией вычислительных процедур.

2. Основные теоретические положения

При проектировании микропроцессорных контроллеров различных машин или технологических процессов возникает необходимость программирования таких типовых процессов, как опрос состояния двоичного датчика, ожидание события, формирование выходных управляющих сигналов, формирование временных задержек и т.п. В данном разделе рассмотрим некоторые способы программной реализации типовых управляющих воздействий применительно к однокристальному 8-разрядному микропроцессору (МП) КР580 ИК80А, а так же программы вычислительных процедур, разработка которых связанна с ограничениями, накладываемыми особенностями организации этого МП.

1. Опрос двоичного датчика.

На рис.5а показана схема подключения контакта двоичного датчика к входному порту МП-контроллера. Если контакт К разомкнут, то на выходе D_5 присутствует сигнал 1, если замкнут, то D_5=0 (т.к. в этом случае D_5 подключен на «землю»).

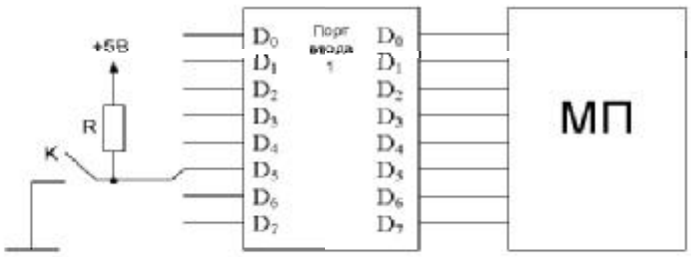


Рис. 5а

Необходимо в некоторой части управляющей программы контроллера опросить значение сигнала на входе D_5 порта 1 и в зависимости от него передать управление фрагменту программы с меткой LABEL A (если D_5 =0) или по адресу , отмеченному меткой LABEL R (D_5=1)

На рис.5б приводится алгоритм программы 3, реализующий процедуру опроса двоичного датчика. Программа имеет символическое имя INPKEY (ввод ключа), которое используется в качестве метки начальной команды этой программы.

Рис.5б

Текст программы в стандартной записи приводится ниже. В поле комментария приводится подробное описание работы программы.

Программа 3

Метка	Мнемокод	Комментарий
INPKEY	IN 01H ANI 20H	Ввод в аккумулятор из порта 1 Маскирование всех разрядов введенного байта, кроме D_5
	JZ LABEL A;	Переход к фрагменту LABEL A, если D5=0, иначе выполнение очередной команды
LABEL B:	…, …, …,	Начало фрагмента B
LABEL A:	…, …, …,	Начало фрагмента A

Организация условных переходов в микроЭВМ осуществляется с помощью регистра признаков микропроцессора.

Регистр признаков имеет пять разрядов, каждый из которых устанавливается по определенному правилу в соответствии с выполнением МП последней команды (см. работу 2).

Во многих случаях при выполнениях программ необходимо проверять или изменять (маскировать) состояние одного или нескольких разрядов числа в аккумуляторе. Это можно осуществить с помощью следующих команд логических операций:

- логического умножения числа в аккумуляторе и маски, очищающего разряда числа, если в соответствующем разряде маски будет записан 0, и не изменяющая его, если в разряде маски записана 1;
- логического сложения числа в аккумуляторе и маски, устанавливающего разряд числа в 1, если в таком же разряде маски будет записана 1, и не изменяющего его, если в этом разряде записан 0;
- логического «исключающего ИЛИ» числа в аккумуляторе и маски, инвертирующего содержание разряда числа, если в соответствующем разряде маски записана 1, и не изменяющая его, если в этом разряде записан 0.

Примеры использования этих команд приведены в табл.8.

Таблица 8

Мнемокод	Машинный код	Число в аккумуляторе	Маска	Комментарий	Результат в аккумуляторе
ANI<B1>	E6<B1>	0011 1010 1111 1111 0000 0000 1010 1010 1111 0000	1010 1100 0010 0010 0010 0010 0010 0010 1111 1111	Логическое умножение содержимого аккумулятора с байтом B1	0010 1000 0010 0010 0000 0000 0010 0010 1111 0000
ORI<B1>	F6<B1>	0011 1010 0000 1111 1111 0000	1010 1100 0000 1111 0000 1111	Логическое сложение содержимого аккумулятора с байтом B1	1011 1110 0000 1111 1111 1111
XRI<B1>	EE<B1>	0011 1010 0000 1111 1111 0000	1010 1100 0000 1111 0000 1111	Логическое «исключающее ИЛИ «содержимого аккумулятора с байтом B1	1001 0110 0000 0000 1111 1111

При выполнении всех логических команд задействуются разряды Z, S, P, AC регистра признаков (в разряде C записывается 0). Это позволяет прове-

рить состояния любого разряда числа и выполнять условные переходы в программах.

Проведение логических операций возможно также с содержимым аккумулятора и внутренними регистрами МП. В этом случае команды – однобайтные.

2. Ожидание события

Контроллеры технологических объектов работают в реальном масштабе времени, и, следовательно, их функционирование должно определяться событиями, происходящими в объекте управления. Чаще всего события в объекте управления фиксируются с использованием двоичных датчиков («Да «- «Нет»), например: замыкание или размыкание кольцевого переключателя, контролирующего перемещения детали на станке.

Пусть требуется по ходу выполнения управляющей программы приостановить продвижение по программе до тех пор, пока в результате процессов, происходящих в объекте управления, не замкнет контакт К некоторого двоичного датчика. На рис.6 показаны схема подключения контакта двоичного датчика к порту ввода контроллера и схема алгоритма реализации процедуры ожидания события. Алгоритм ожидания события отличается от алгоритма опроса двоичного датчика (см. рис.5) тем, что процедура опроса второго разряда порта 2 повторяется многократно до тех пор, пока не выполнится технологическое условие, например, не замкнется кольцевой переключатель. Таким образом, этот процесс носит циклический характер.

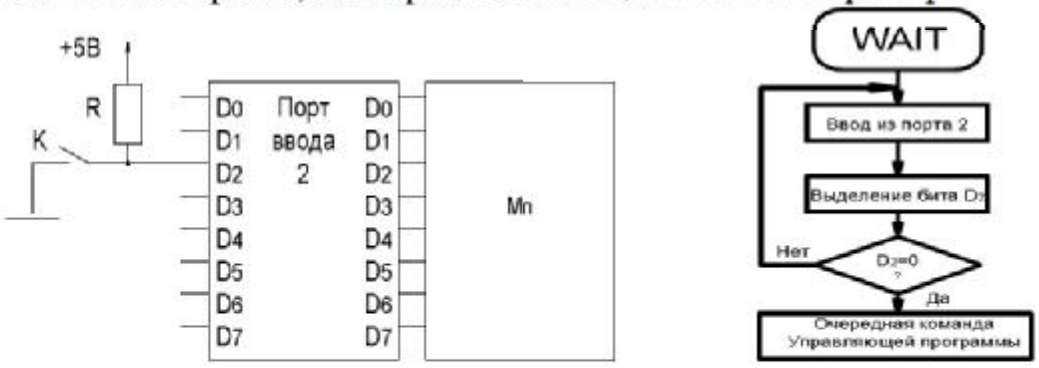


Рис.6

3. Формирование управляющего сигнала

На рис.7 показана схема подключения контроллера к некоторому исполнительному механизму объекта управления через порт вывода информации.

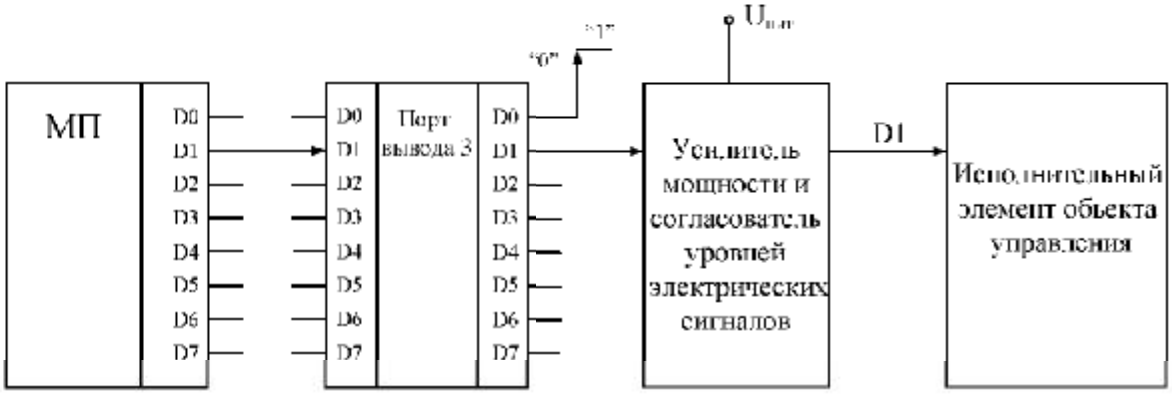


Рис. 7

Предположим, что данный исполнительный механизм работает по принципу «Включить – Выключить», т.е.может управляться двоичным выходным сигналом контроллера.

Программа формирования такого управляющего воздействия чрезвычайно проста и состоит всего из двух команд:

для включения исполнительного механизма

ON: MVI A, 02H;	загрузить в аккумулятор число, имеющее 1 во втором разряде (D1)
OUT 03H;	передать содержимое А в порт вывода 03H;

для выключения исполнительного механизма

OFF : XRA A;	произвести операцию «Исключающее ИЛИ» над содержимым А (в результате получим 00H – обнуление аккумулятора);
OUT 03H	передать содержимое А в порт вывода 03H.;

В том случае, если к остальным семи выводам выходного порта 3 подсоединяются другие исполнительные механизмы, формируется не двоичное управляющее воздействие, а байт управляющего слова, где каждому разряду ставиться в соответствие 0 или 1 в зависимости от того, какие исполнительные механизмы должны быть включены или отключены.

4. Формирование временной задержки

Программная реализация временной задержки использует метод программных циклов, при котором в некоторый рабочий регистр блока РОН загружается число, которое затем в каждом проходе цикла уменьшается на 1. Так продолжается до тех пор, пока содержимое рабочего регистра не станет равным 0, что интерпретируется программой как момент выхода из программного цикла. Время задержки при этом определяется числом, загружен-

ным в рабочий регистр, и временем выполнения команд, образующих программный цикл. Схема алгоритма такой программы показана на рис.8. Программа имеет символическое имя TIME и может использоваться в виде подпрограммы. Подпрограмма – это последовательность команд, выполнение которых может быть вызвано из любого места программы любое количество раз. Процесс передачи управления подпрограмме называется ее вызовом. Для вызова подпрограмм и возврата из них используются команды CALL (вызов) и RET (возврат).

Структура алгоритма отражена в программе 4.

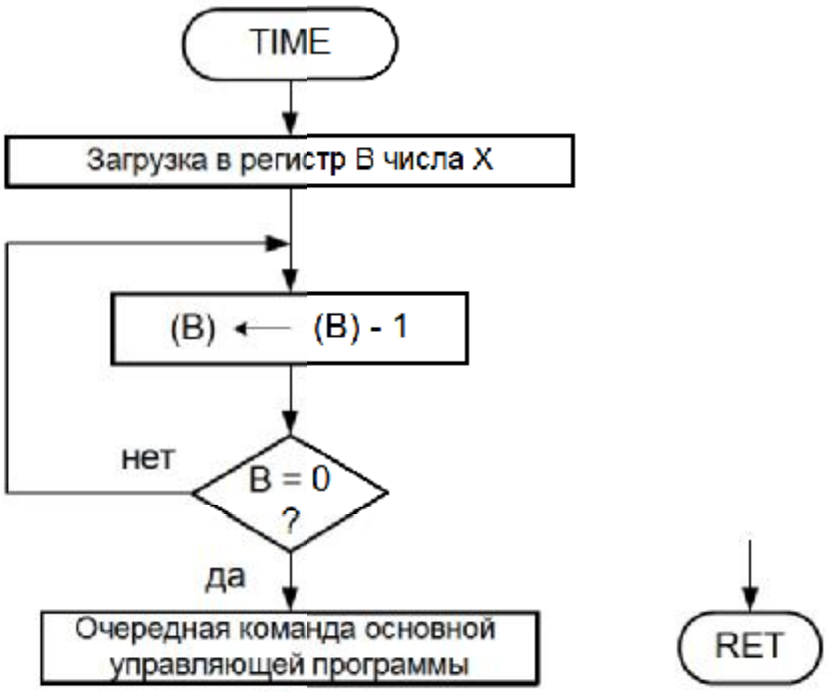


Рис. 8

Программа 4

Метка	Мнемокод	Комментарий
TIME:	CALL TIME; MVI B, X;	Вызов подпрограммы «TIME» Загрузка в регистр B числа X
COUNT:	DCR B;	Уменьшить на 1 содержимое регистра B
	JNZ COUNT;	Цикл (переход к фрагменту COUNT), если B не равно 0
	RET;	Возврат в основную программу, если B = 0

Для получения требуемой величины временной задержки необходимо определить значение числа X, загружаемого в рабочий регистр B. Число X определяется на основе расчета времени выполнения команд, образующих данную подпрограмму.

Предположим, что в программе, управляющей работой контроллера, микропроцессор которого работает с тактовой частотой 2 МГц (период составляет 0,5 мкс), необходимо реализовать временную задержку длительностью 250 мкс.

В описании системы команд МП К580ИК80А указывается, за сколько тактов основной частоты синхронизации исполняются команды МП. На основе этих данных можем записать:

CALL TIME – 17 тактов – 8,5 мкс;
MVI B, X – 7 тактов – 3,5 мкс;
DCR B – 5 тактов – 2,5 мкс;
JNZ COUNT – 10 тактов – 5 мкс;
RET – 10 тактов – 5 мкс.

Таким образом, однократно исполняемые команды (CALL, MVI, RET) в этой подпрограмме требуют 17 мкс (8,5+3,5+5,0). Для получения требуемой задержки в 250 мкс, следовательно, необходимо выполнить команды DCR B и JNZ COUNT столько раз, чтобы время их исполнения составило 233 мкс (250 – 17):

$$X = (233)/(2,5+2,0) \approx 31$$

При этом задержка составит 232,5 мкс.

Если точность программной реализации временной задержки длительностью 250 мкс с погрешностью 0,5 мкс удовлетворяет условию задачи, то на этом разработка программы заканчивается.

Исходя из проведенного расчета, можно записать текст программы TIME:

TIME	MVI B, 31
COUNT:	DCR B
	JNZ COUNT
	RET

Реализация временной задержки большей длительности при относительно высокой частоте синхронизации 2 МГц, с использованием описанного ниже метода не представляется возможной, т.к максимальной емкости регистровой пары (FFFFH) не хватит для того, чтобы представить число X, достаточное для формирования задержки, равной, например 1 сек. Сформировать столь большую (для МП) задержку можно с использованием метода вложенных циклов подобно тому, как это показано на рисунке 9 (программа 5).

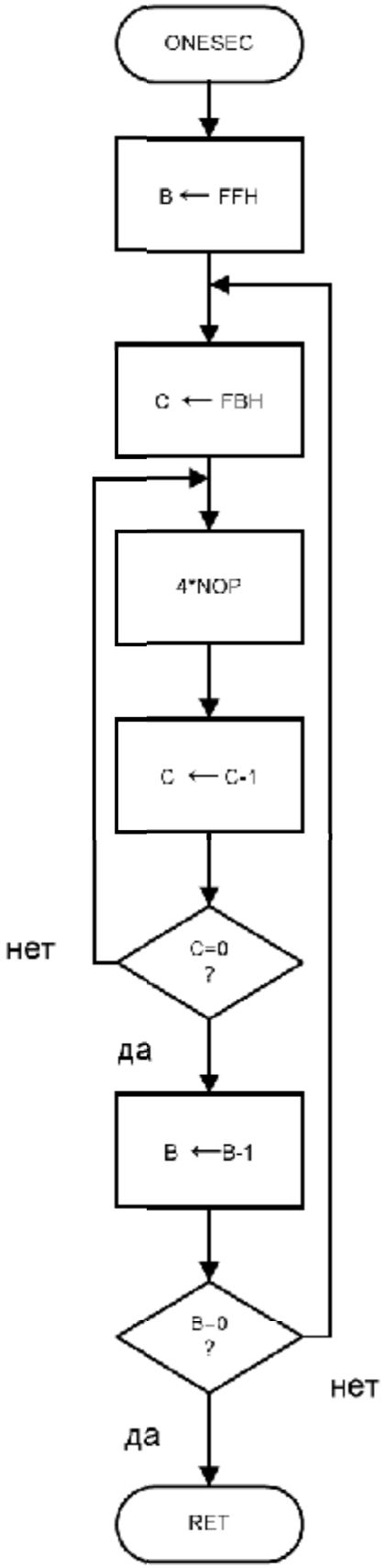


Рис. 9

Программа 5

Метка	Мнемокод	Комментарий
ONESEC:	MVI B, FFH;	Счетчик внешних циклов
L1:	MVI C, FBH;	Счетчик внутренних циклов
L2:	NOP;	Точная подгонка времени внутреннего цикла
	NOP;	
	NOP;	
	NOP;	
	DCR C;	Декремент счетчика внутренних циклов
	JNZ L2;	Возврат во внутренний цикл, если С не равно нулю
	DCR B;	Декремент счетчика внешних циклов
	JNZ L1	Возврат во внешний цикл, если В не равно нулю
	RET	

5. Программная реализация типовых вычислительных процедур

Возникающие в процессе разработки прикладных программ для МП-систем, построенных на основе МП К580ИК80А, сложности определяются: отсутствием в системе команд МП эффективных и часто используемых операций, например умножения и деления; малоразрядным форматом данных и, следовательно, низкой точностью их обработки; ограниченным диапазоном представления данных из-за отсутствия команд обработки чисел с плавающей точкой; отсутствием операций десятичной арифметики.

Эти ограничения не являются непреодолимыми, однако способы их преодоления являются во многих случаях довольно сложными.

На рис. 10 представлена блок-схема алгоритма сложения массива однобайтных чисел.

На рис. 11 представлена блок-схема алгоритма умножения двух однобайтных чисел.

3. Порядок выполнения работы

1. Рассмотрим работу микропроцессора в режиме ожидания события. Используя представленный алгоритм (см. рис.6,б), написать программу работы МП в режиме ожидания момента включения ключа К (см. рис.6, а).

Ознакомится с программой 6 для определения единицы в пятом разряде числа (D_5), принятого из памяти. Программа использует маскирование числа и условный переход.

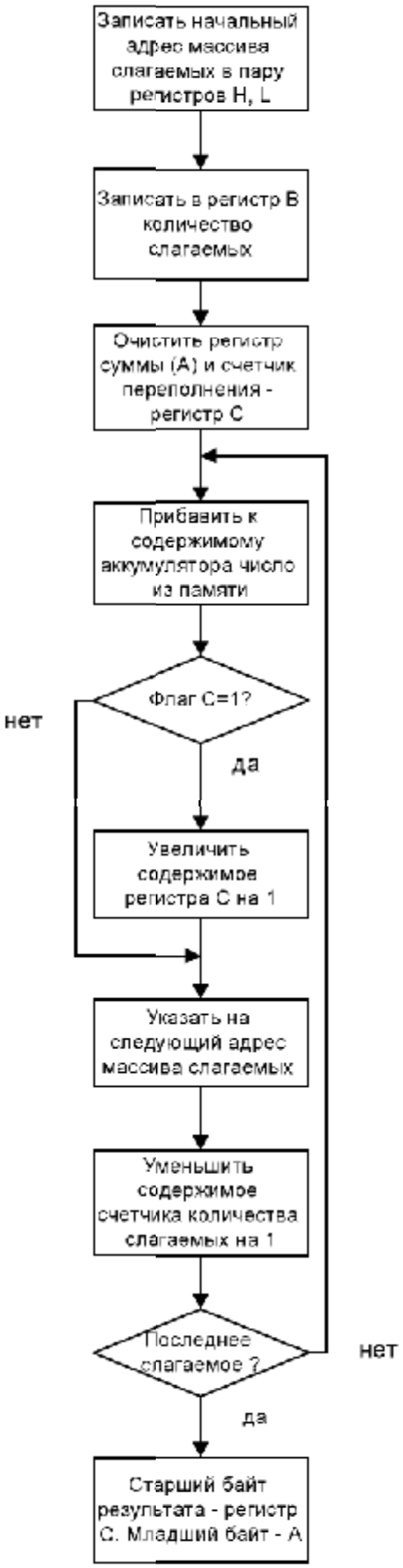

Записать начальный адрес массива слагаемых в пару регистров H, L
Записать в регистр B количество слагаемых
Очистить регистр суммы (A) и счетчик переполнения - регистр C
Прибавить к содержимому аккумулятора число из памяти
Флаг C=1?
нет
да
Увеличить содержимое регистра C на 1
Указать на следующий адрес массива слагаемых
Уменьшить содержимое счетчика количества слагаемых на 1
Последнее слагаемое ?
нет
да
Старший байт результата - регистр C. Младший байт - A

Рис. 10

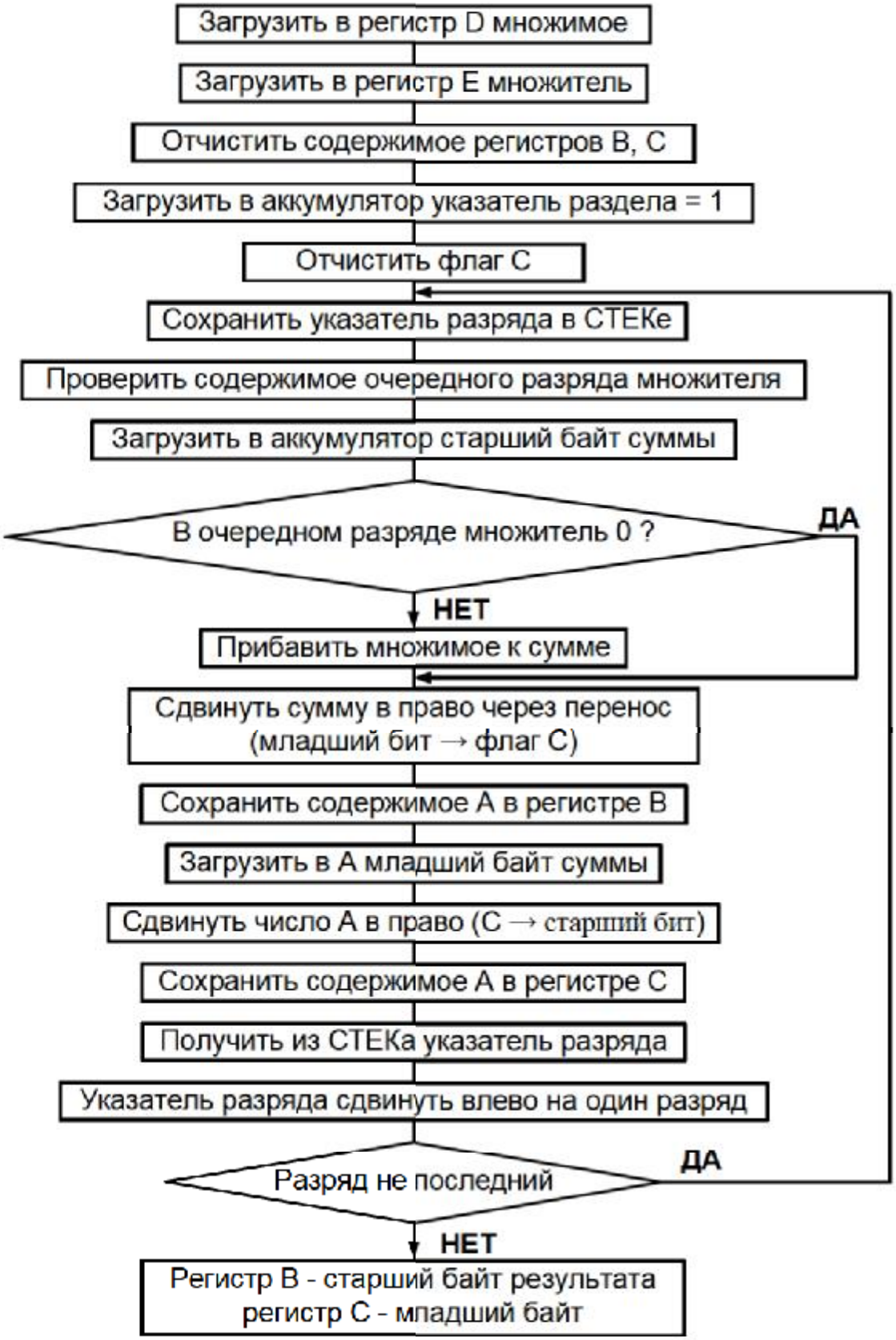

Загрузить в регистр D множимое
Загрузить в регистр E множитель
Отчистить содержимое регистров B, C
Загрузить в аккумулятор указатель раздела = 1
Отчистить флаг C
Сохранить указатель разряда в СТЕКе
Проверить содержимое очередного разряда множителя
Загрузить в аккумулятор старший байт суммы
В очередном разряде множитель 0 ?
ДА
НЕТ
Прибавить множимое к сумме
Сдвинуть сумму в право через перенос (младший бит → флаг C)
Сохранить содержимое A в регистре B
Загрузить в A младший байт суммы
Сдвинуть число A в право (C → старший бит)
Сохранить содержимое A в регистре C
Получить из СТЕКа указатель разряда
Указатель разряда сдвинуть влево на один разряд
Разряд не последний
ДА
НЕТ
Регистр B - старший байт результата
регистр C - младший байт

Рис. 11

Программа 6

Адрес	Машинный код	Метка	Мнемокод	Комментарии
0900	21 20 09		LXI, H 0920	Загрузить пару регистров H, L начальный адрес массива вводимых чисел
0903	7E	WAIT	MOV A, M	Переслать в аккумулятор число из памяти
0904	46		MOV B, M	Запомнить число из памяти в регистре B
0905	E6 20		ANI 20	Проверить состояние пятого разряда числа
0907	23		INX H	Сформировать в паре регистров H, L адрес следующего входного числа
0908	CA 03 09		JZ WAIT	Идти на WAIT если в пятом разряде был 0 (Z = 0)
090B	…		…	Очередная команда управляющей программы

Исследовать программу 6. Для этого:

- ввести в УМК программу 6

П, 0900, _, 21, _, 20, _, 09, _, …, CA, _, 03, _, 0, 9, ВП;

- ввести в УМК массив входных чисел из табл. 9 (по указанию преподавателя); начиная с адреса 0920H;

П, 9020, _, B_1, _, B_2, _, B_3, _, B_4, _, B_5, ВП;

Таблица 9 – Данные массива входных чисел

№ варианта	B_1	B_2	B_3	B_4	B_5
1	55	87	CC	6E	16
2	C7	9B	A3	56	BA
3	91	46	9E	C7	6C

- осуществить пуск программы:

СТ, 0900, _, 090B, ВП;

- исследовать результат выполнения программы по числу, по записанному в регистре В (после нажатия клавиши «_» записать содержимое регистра В в форму 5):

РГ, В, _, ВП;

- изменить программу так, чтобы проверялось состояние третьего разряда числа. Записать измененную программу в форму;

Форма 5 Измененная программа

Массив входных чисел	16-ричный код	Двоичный код	Число в регистре В	Адрес	Машинный код	Мнемокод	Число в регистре В
B_1							
B_2							
B_3							
B_4							
B_5							

-осуществить пуск измененной программы и исследовать результат ее выполнения по числу, записанному в регистре В.

Результат исследований занести в форму 5.

2. Рассмотрим работу микропроцессора в режиме формирования управляющего сигнала.

Используя представленную схему (см. рис. 9) написать программу работы МП в режиме управления исполнительными механизмами для вариантов указанных в табл. 10 и оформить в отчете по форме 6.

Таблица 10 – Состояние исполнительных механизмов для вариантов

№ исполнительного механизма	1	2	3	4	5	6	7
0 (D_0)	вкл.	откл.	вкл.	откл.	вкл.	откл.	вкл.
1 (D1)	откл.	вкл.	откл.	вкл.	откл.	вкл.	вкл.
2 (D2)	вкл.	откл.	вкл.	откл.	вкл.	откл.	вкл.
3 (D3)	откл.	вкл.	откл.	вкл.	вкл.	вкл.	вкл.
4(D4)	вкл.	откл.	откл.	откл.	вкл.	откл.	вкл.
5 (D5)	откл.	вкл.	вкл.	вкл.	откл.	вкл.	вкл.
6 (D6)	вкл.	вкл.	откл.	откл.	вкл.	откл.	вкл.
7 (D7)	вкл.	откл.	вкл.	вкл.	откл.	откл.	откл.

Форма 6

№ варианта	Байт управляющего слова	Программа		
	в двоичном коде	в 16-ричном коде	метка	мнемокод

3. Рассмотрим работу МП в режиме формирования временной задержки.

Используя представленный на рисунке алгоритм формирования задержки на 1 сек., рассчитать время выполнения программы ONESEC. Команда NOP выполняется за 2 мкс.

Изменить программу ONESEC таким образом, чтобы временная задержка была равна 10 сек., и записать ее в отчет по форме 7, начиная с адреса 0800H.

Форма 7

Адрес	Машинный код	Метка	Мнемокод	Комментарий
0800		TEN SEC		

Занести полученную программу в память УМК:

П, 0800, _ , … ,ВП.

Осуществить пуск программы:

СТ, 0800, _ , Ak, ВП,

где Ak – конечный адрес программы (адрес программы RET).

Определить по часам время выполнения программы: момент запуска программы соответствует нажатию клавиши ВП, а момент окончания работы программы – появлению световой индикации на дисплее. Результат занести в отчет.

4. Рассмотрим реализацию процедур сложения и умножения.

Используя блок-схему алгоритма сложения массива однобайтных чисел (см. рис. 10) написать программу и занести ее в отчет в виде таблицы по форме 7.

Исходные данные для выполнения данной программы представлены в табл. 11.

Таблица 11

№ варианта	1	2	3	4	5	6
Количество слагаемых	5	5	5	6	6	6
	20	25	10	25	30	40
	30	35	100	25	30	40
Массив слагае-мых	40	5	5	50	50	40
	50	100	35	80	70	30
	160	110	80	100	110	60
				25	100	150

Занести программу в память УМК:

П, 0800, _ , … , ВП.

Занести по указанию преподавателя в память УМК, начиная с адреса 0900Н, массив слагаемых из таблицы. Предварительно слагаемые, представленные в десятичной системе счисления, необходимо перевести в 16-ричный код записи и занести в форму 8.

Форма 8

Формы счисления	Десятичная	Шестнадцатеричная
Исходные данные		
Результат		

Осуществить пуск программы:

СТ, 0800, _ , Ak, ВП.

где Ak – конечный адрес программы.

Результат сложения (в соответствии с алгоритмом нашей программы) находится в регистрах С (старший байт) и А (младший байт). Записать результат в форму 8, нажимая последовательно клавиши:

РГ, С, _ , А, ВП.

Используя блок-схему алгоритма умножения двух однобайтных чисел, представленную на рис.11, написать программу и занести ее в отчет в виде таблицы по форме 7.

Исходные данные для выполнения данной программы представлены в табл. 12.

Таблица 12

№ варианта	1	2	3	4	5	6
Множимое	115	16	20	25	15	10

Множитель	20	19	25	30	42	120

Занести программу в память УМК

Исходные величины для работы программы брать из таблицы по указанию преподавателя, переводя их из десятичной системы счисления в 16-ричный код.

Осуществить пуск программы.

Результат умножения находится в регистровой паре B, C.

Записать результат в форму 8, нажимая последовательно клавиши:

РГ, B, _ , C, ВП.

В отчете должна быть подробная цифровая запись перевода результата из 16-ричной в десятичную.

4. Содержание отчета

1.Название работы.

2. Схема, алгоритм и программа работы МП в режиме ожидания момента включения ключа К (см. рис. 6,а).

3. Текст программы 6.

4. Таблица 5.

5. Программа работы МП в режиме управления исполнительными механизмами.

6. Таблица по форме 7.

7. Алгоритм и программа формирования временной задержки на 10 с.

8. Алгоритм и программа сложения массива однобайтовых чисел.

9. Программа умножения двух однобайтных чисел.

10. Таблица по форме 8.

Вопросы для самопроверки.

1. Как можно использовать команду логического «И»

2. Какую команду надо ввести в МП для обнуления А?

3. Напишите команду для определения 1 в четвертом разряде числа, занесенного в А.

4. Напишите команду для определения 1 в третьем и седьмом разрядах числа, записанного в аккумуляторе.

5. Как работают команды перехода: JMP – безусловного; JNZ – условного?

6. Как работает команда PUSH PSW?, POP PSW?

Литература: [1], с. 102 …110; [2], с. 26 …28; 31 …36; 40 …46.

РАБОТА 4. ОРГАНИЗАЦИЯ ЦИКЛОВ, ОБРАБОТКА МАССИВОВ И МАСКИРОВАНИЕ ДАННЫХ.

1. Цель работы

Изучение способов организации циклов с помощью условных переходов, обработка массивов данных и маскирование данных.

2. Теоретические сведения

Под алгоритмизацией понимается сведение задачи к последовательности этапов, выполняемых друг за другом так, что результаты предыдущих этапов используются при выполнении следующих. Результатом выполнения этапа алгоритмизации является алгоритм решения задачи, где под алгоритмом будем понимать систему правил, четко описывающую последовательность действий, которые необходимо выполнить для решения задачи. Алгоритм обладает следующими основными свойствами: дискретностью, определимостью, результативностью и массовостью.

Дискретность – процесс преобразования походных данных в результат осуществляется дискретно по шагам.

Определенность – алгоритм должен быть четким и однозначным, так, что значение величин, получаемых в какой-либо момент времени, определяют значением величин, полученными в предыдущие моменты времени.

Результативность – алгоритм должен приводить к решению задачи за конечное число шагов.

Массовость – алгоритм решения задачи разрабатывается в общем виде так, что бы его можно было применить для класса задач, различающимися исходными данными.

Этапы и условные изображения алгоритма

Типичные этапы алгоритмы можно представить следующим образом:

1. Этап обработки вычисления:

V = ВЫРАЖЕНИЕ,

где V – переменная.

Этот этап содержит вычисления выражения в правой части.

2. Проверка условия:

УСЛОВИЕ ?

Если условие выполняется, то осуществляется переход к этапу с номером (меткой) N. Если условие не выполняется, то переходя к следующему по порядку этапу.

3. Конец вычислений:

ОСТАНОВ

4. Переход к этапу с номером N:

ИДТИ К N

Использование условных отображений перечисленных ниже этапов позволяет представить алгоритм в наглядной форме.

- Этапы обработки (вычисления) обозначаются прямоугольником (рис. 12а), внутри которого записывается содержание этого этапа.

- Проверка условия изображается ромбом, внутри которого записывается условие. В результате проверки выбирается один из двух возможных путей вычислительного процесса (рис. 12б). Если условие выполняется, т.е. имеет значение ДА, то следующим выполняется переход по стрелки ДА (аналогично для НЕТ).

- Начало и конец вычислительного процесса изображают фигурой (рис. 12в). Внутри нее записывается НАЧАЛО или ОСТАНОВ.

- Ввод исходных данных и вывод результата изображается параллелограммом (рис.12г). Внутри него имеется слово ВВОД или ПЕЧАТЬ и перечисляются переменные, подлежащие вводу или выводу.

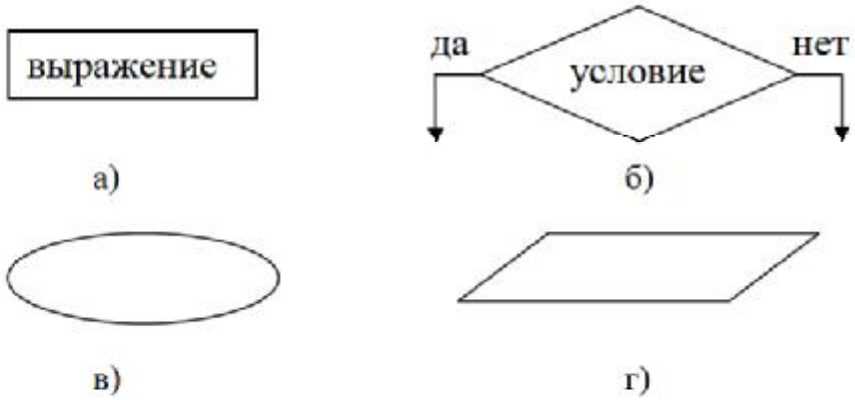


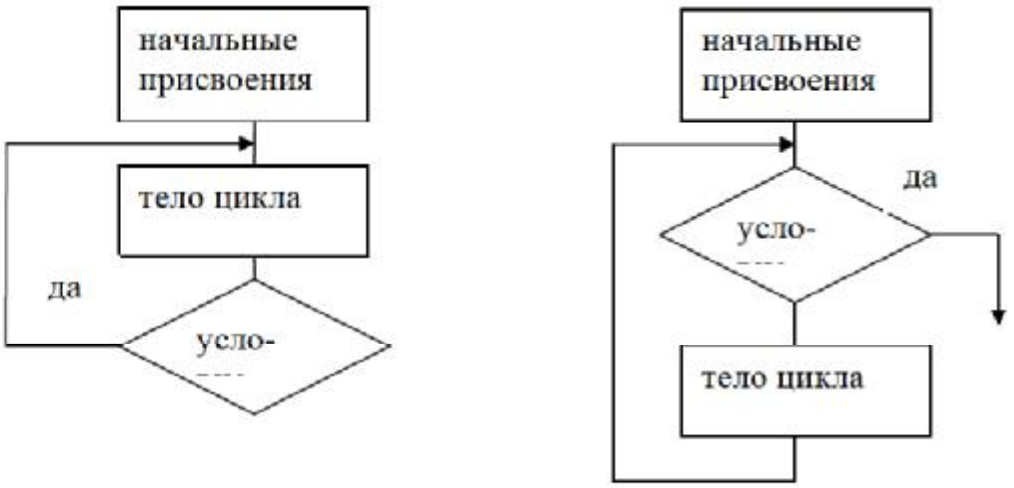
Рис. 12

Основные структуры алгоритма – это ограниченный набор стандартных способов соединения блоков этапов алгоритма для выполнения типичных последовательностей. Действительно, структурный метод предполагает использование следующих основных структур:

1) следование
2) цикл «До»
3) цикл «Пока»
4) разветвление
5) обход.

Следование – последовательное размещение блоков и групп блоков.

Цикл «До» – применяется при необходимости выполнить вычисление несколько раз до тех пор, пока выполняются некоторые условия (рис. 13а).



а) б)

Вошедшие в рисунок этапы:

Тело цикла – представляет последовательность действий, выполняемых многократно (в цикле).

Начальные присвоения – задание начальных значений тем переменным, которые используются в теле цикла.

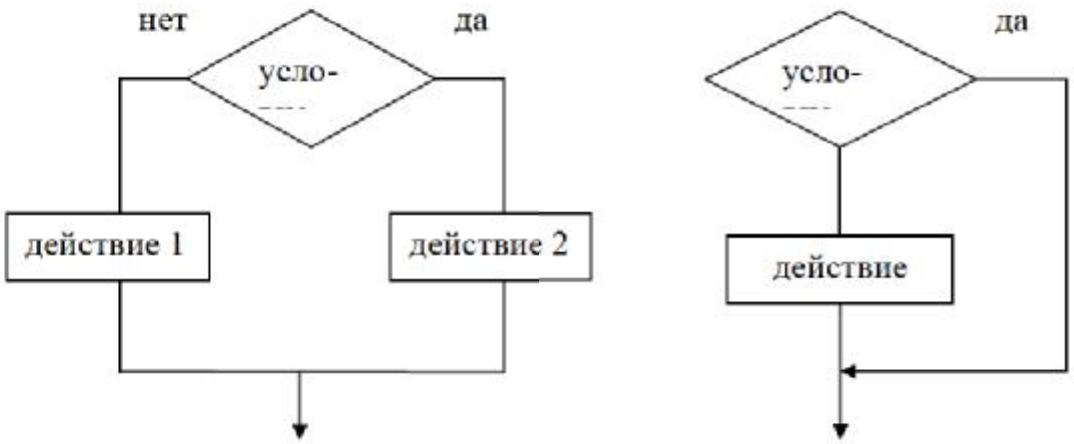
Рис. 13

Цикл «Пока» (рис. 13б) отличается от цикла «До» тем, что здесь проверка условия производится до выполнения тела цикла.

Разветвление – (рис. 14а) применяется, когда в зависимости от условия нужно выполнять либо одно, либо другое действие.

Обход – (рис. 14б) частный случай разветвления, когда одна ветвь не содержит никаких действий.

Особенностью приведенных структур является то, что они имеют один вход и один выход и их можно соединять друг с другом в любой последовательности.



а) б)

Рис. 14

Организация условных и безусловных переходов

В системе команд существует группа команд, играющая особую роль в организации выполнения программ микроЭВМ. Это команды передачи управления. Пока в программе не встречаются команды этой группы, счетчик команд PC постоянно увеличивает свое значение, и микропроцессор выполняет команду за командой в порядке их расположения в памяти.

Порядок выполнения программы может быть изменен, если занести в регистр счетчика команд PC код адреса, отличающийся от адреса очередной команды. Такая передача управления или переход в программе может быть выполнена с помощью трехбайтовой команды безусловного перехода JMP ADR.

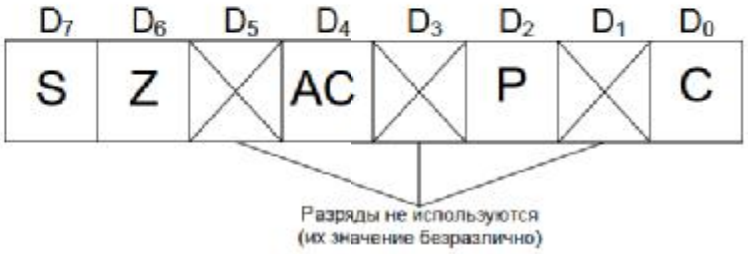
Безусловную передачу управления можно произвести также по команде PCHL, в результате выполнения которой произойдет передача управления по адресу, хранящемуся в регистровой паре HL.

Кроме команды безусловного перехода имеет 8 трехбайтовых команд условного перехода. При появлении команды условного перехода, передача управления по адресу, указанному в команде, происходит только в случае выполнения определенного условия. Если условие не удовлетворяется, то выполняется команда, непосредственно следующая за командой условного перехода.

Условия, с которыми оперируют команды условной передачи управления, определяются состоянием регистра признаков F.

Как было отмечено в лабораторной работе 2, разряды в регистре признаков распределены следующим образом:

Признаки результата размещаются в регистре флагов следующим образом:



Пять разрядов регистра F имеют следующие значения.

Разряд знака S – SIGN. В него записывается 1, ели при выполнении арифметической или логической команды в старшем, седьмом, разряде аккумулятора записана 1, в противном случае в разряд записывается 0.

Разряд нулевого результата Z – ZERO. Если результат равен нулю, в него записывается 1, в противном случае Z=0.

Дополнительный разряд переполнения AC – AUX. CARRY. В него записывается 1, если при выполнении команд в аккумуляторе возникает единица переноса из третьего разряда чисел (перенос между тетрадами байта).

Разряд четности P – PARITY. В него записывается 1, если при выполнении команды количество единиц в разрядах аккумулятора будет четным. В противном случае P=0.

Разряд переполнения C – CARRY. В него записывается 1, если при выполнении арифметической команды или команды сдвига было переполнение аккумулятора, в противном случае C=0.

Связь между состоянием регистра признаков и мнемоникой команды условного перехода:

NZ (NOT ZERO) – ненулевой результат Z=0
Z (ZERO) – нулевой результат Z=1
NC (NO CARRI) – отсутствие переноса CY=0
C (CARRI) – перенос, CH=1
PO (PARITY 0DD) – нечетный результат, P=0
PE (PARITY EVEN) – четный результат, P=1
P (PLUS) – число неотрицательное, S=0
M (MINUS) – число отрицательное, S=1

Дополнив данную мнемонику, мнемоникой JMP – переход, получим соответствующие команды условного перехода JNZ, JZ, JNC, JC, JPO, JPE, JP, JM. Применению данных команд в программе способствует таблица 13, где приведены указанные команды.

Для передачи управления по условию, записанного в левом столбце таблицы, (где A – аккумулятор), необходимо выполнить последовательно команды из 2-го и 3-го столбцов. Если сравниваются операции со знаком, то вместо команд JC, JNC следует использовать команды JM и JP соответственно. Здесь ADR – адрес перехода, D8 – непосредственный операнд (8-и разрядное число).

Таблица 13.

Условие	**Команда, устанавливающая бит регистра признаков F**	**Команда условной передачи управления**
Любой бит A=0	ANI D8 (1 в соответствующем разряде выбирает бит)	JZ ADR
Любой бит A=1	ANI D8 (1 в соответствующем разряде выбирает бит)	JNZ ADR
Бит 7 A=0	RAL, RLC или ADD	JNC ADR
Бит 7 A=1	RAL, RLC или ADD	JNC ADR
Бит 6 A=0	ADD A	JP ADR
Бит 6 A=1	ADD A	JM ADR
Бит 0 A=0	RAR или RRC	JNC ADR

Бит 0 A=1	RAR или RRC	JC ADR
Все биты A=0	ANA A или ORA A	JZ ADR
Содержимое A ≠0	ANA A или ORA A	JNZ ADR
Содержимое A положительно (ст. бит = 0)	ANA A или ORA A	JP ADR
Содержимое A отрицательно (ст. бит = 1)	ANA A или ORA A	JM ADR
Содержимое A=D8	CPI D8	JZ ADR
Содержимое A ≠D8	CPI D8	JNZ ADR
Содержимое A≥D8	CPI D8	JNC ADR
Содержимое A<D8	CPI D8	JC ADR
Содержимое A=R	CMP R	JZ ADR
Содержимое A≠R	CMP R	JNZ ADR
Содержимое A≥R	CMP R	JNC ADR
Содержимое A<R	CMP R	JC ADR

Циклические программы

Наиболее часто команды условного перехода используются для организации циклов. Рассмотрим более подробно организацию цикла на примере структуры цикла «До».

На рис. 15 представлена развернутая структурная схема циклической программы.

В блоке начального присвоения указываются начальные значения массивов, в которых размещается информация, и устанавливаются счетчики.

В теле цикла, условно разделенном на блоки обработки и управления циклом, производится обработка элементов массивов данных и производится изменение счетчиков перед выполнением операции по обработке следующего элемента массива.

В блоке условие производится проверка числа выполняемых переходов с помощью команды условного перехода. Если обработка всего массива не завершена, то переход повторяется.

Рассмотрим два примера обработки массива данных с использованием цикла.

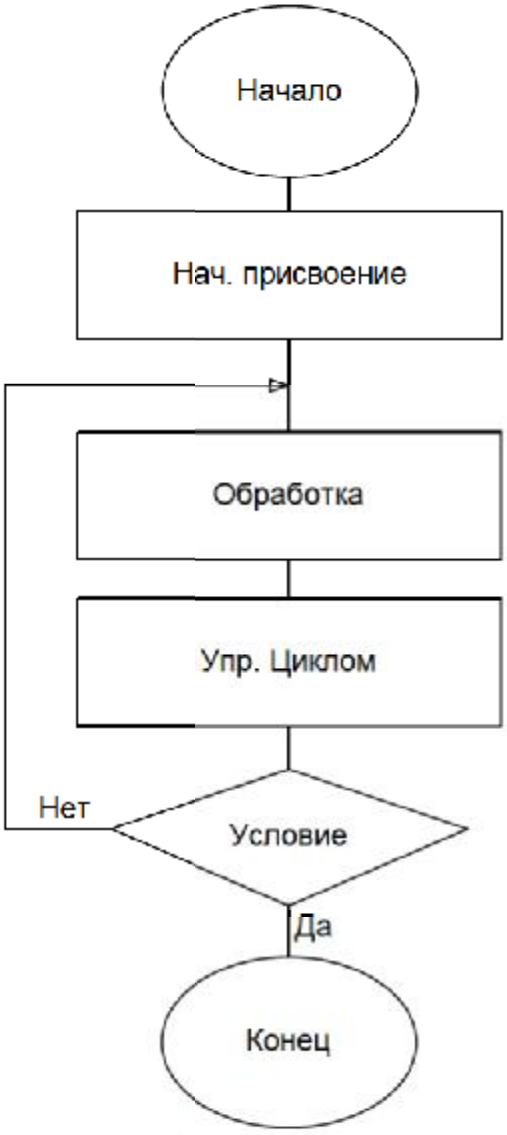


Рис. 15

Пример 1. Суммирование элементов массива. Необходимо осуществить суммирование трех элементов массива данных, расположенных в ячейках памяти, начиная с адреса 2142 и разместить результат в ячейке по адресу 2140.

Адрес ячейки памяти	Содержимое ячейки памяти
2141 H	03 H
2142 H	35 H
2143 H	72 H
2144 H	1D H

Ниже приведена программа суммирования.

Метка	Команда	Адреса, операнд	Комментарий
	SUB	A	Очистка аккумулятора
	LXI	H, 2141H	Установка адресного указателя
	MOV	B, M	Установка счетчика: чтение из ячейки 2141 числа элементов массива и запись в регистр B
1.00	INX	H	Подготовка в регистрах адреса следующего элемента массива данных
	ADD	M	Обработка элементов массива
	DCR	B	Модификация счетчика числа обработанных элементов массива
	JNZ	1.00	Проверка окончания цикла
	STA	2140H	Запомнить сумму в ячейке 2140
	HLT		Завершение цикла

В начале программы устанавливаются начало массива (ячейка 2142 H) и число его элементов (число 03 ячейка 2141 H). Этот адрес загружается в регистровую пару HL . Далее производится загрузка в регистр B, который служит счетчиком числа обработанных байт, содержимым ячейки 2141. На этом формирование начальных значений цикла заканчивается. Следующие три команды составляют собственно тело цикла, в котором команда ADD M производит суммирование содержимого аккумулятора с содержимым ячейки памяти указанной регистровой парой HL, а команда INX H подготавливает содержимое HL (увеличивает на единицу) к следующему циклу. При первом

проходе эта команда из адреса 2141 H формирует адрес 2142, в котором находится первый элемент массива, а в следующие проходы формирует адрес следующих операндов. Команда DCR B уменьшает содержимое счетчика на 1 после каждого прохода и, таким образом производится подсчет числа обработанных элементов массива. Эта команда, помимо указанных действий, управляет значением регистра признаков, в частности, триггером Z. Таким образом, с помощью данной команды осуществляется управление циклом.

Команда JNZ производит проверку выполнения условия Z=0 и если оно выполняется, то программа завершается записью результата в ячейку памяти с адресом 2140 H. Если же B≠0, то происходит переход к метке 1.00.

Пример 2. Нахождение максимума.

Необходимо отыскать максимальный по величине элемент массива, расположенного в ячейках, начиная с адреса 2142. Длина массива, предположим, задана в ячейке 2141, а результат операции помещается в ячейку 2140. элементы массива – восьмибитные числа без знака.

Массив содержит три элемента. В ячейках памяти исходных данных содержатся следующие записи:

Адрес	Содержимое
2141	03 H
2142	37 H
2143	12 H
2144	C6 H

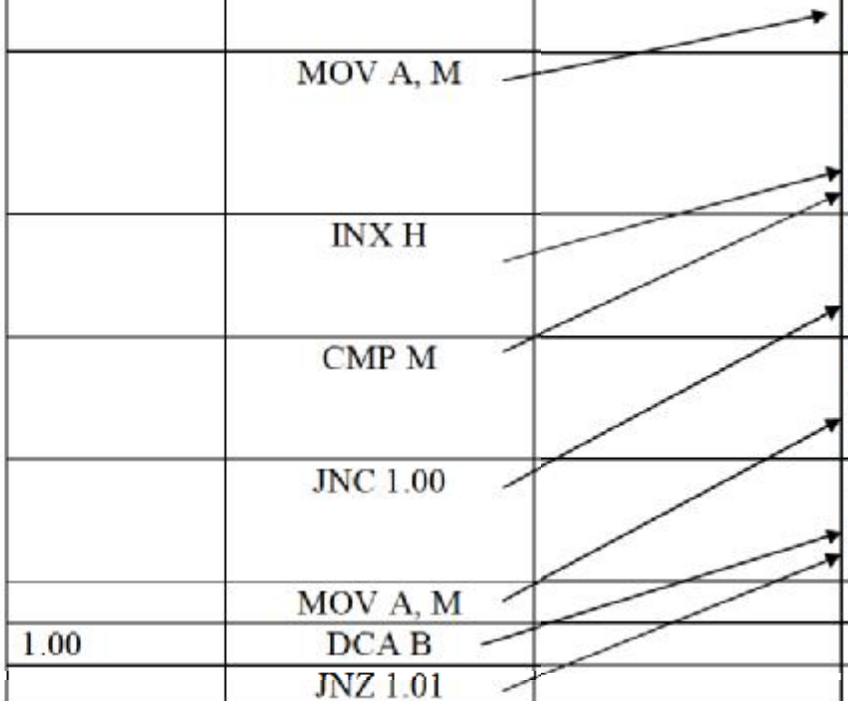
Начало программы во многом похоже на программу примера 1. Первоначально производится загрузка регистровой пары HL адресом первого элемента исходных данных – числом элементов массива. Затем производится считывание этого числа и загрузка его в регистр B, который будет служить счетчиком числа обработанных байт.

Блок обработки начинается с чтения первого элемента массива. Предполагается, что это число максимальное. Затем оно сравнивается со вторым числом с помощью команды сравнения CMP M и устанавливаются флаговые регистры. Отсутствие переноса указывает на то, что первое число больше второго и что необходимо осуществить выборку следующего числа. В противном случае необходимо завершить программу и записать результат в ячейку 2140.

Программа на Ассемблере К 580 имеет вид:

Метка	**Команда**	**Адрес**	**Комментарий**
	LXI H	2141 H	Загрузка начального адреса
	MOV B, M		Установка счетчика числа элементов массивов

	DCR B		Исключение из рассмотрения содержимого ячейки 2141
1.01	INX H		Считать первое число максимумом
	MOV A, M		Сравнение второго числа в памяти с первым в аккумуляторе
	INX H		Максимум больше текущего значения?
	CMP M		Нет, заменить максимум на текущий элемент
	JNC 1.00		Завершена ли обработка всего массива?
	MOV A, M		
1.00	DCA B		
	JNZ 1.01		
	STA 2140 H		Да, записать максимальное значение
	HLT		Закончить работу



Маскирование данных

Во многих случаях при выполнении программ необходимо проверять или изменять (маскировать) состояние одного или нескольких разрядов числа в аккумуляторе. В работе 3, табл.8 представлены команды для маскирования использующих непосредственное задание маски. Ниже приведем примеры касающиеся задания маски в регистре:

1. Логического умножения числа в аккумуляторе и маски, которая очищает разряд числа, если в соответствующем разряде маски будет записан «0» и не изменяет его, если в разряде маски записана «1».

Пример:

Команда	**Число в аккумуляторе**	**Маска (регистр R)**	**Результат**
ANA (R)	0011 1010	0000 1111	0000 1010
	1111 1111	1010 1010	1010 1010
	0000 0000	1111 1111	0000 0000

2. Логическое сложение числа в аккумуляторе и маски, которое устанавливает разряд числа в «1», если в таком же разряде маски будет записана «1», и не изменяет разряд числа, если в этом же разряде маски будет записан «0».

Пример:

Команда	Число в аккумуляторе	Маска (регистр R)	Результат
ORA (R)	0011 1010	1010 1100	1011 1110
	0000 1111	1111 0000	1111 1111
	1111 0000	0000 1111	1111 1111

3. Логическое «исключающее или» числа в аккумуляторе и маски, которое инвертирует содержание разряда числа, если в соответствующем разряде маски записана «1» и не изменяет его, если в этом разряде записан «0».

Пример:

Команда	Число в аккумуляторе	Маска (регистр R)	Результат
XRA(R)	0011 1010	1010 1100	1001 0110
	0000 1111	0000 1111	0000 0000
	1111 0000	0000 1111	1111 1111

Реализация переключательных функций

В совокупности с командами перехода, эти логические команды позволяют реализовать любую комбинационную логическую схему и любой конечный автомат с памятью. При программной реализации переключательных функций f(x) с помощью МП обычно принимают решение последовательно, о значении каждого числа записи f(x), что позволяет судить о значении всей функции на заданном наборе аргументов. При этом надо учитывать, что МП обрабатывает данные побайтно, а логические операции над двумя операндами выполняются поразрядно.

Пример 3: Реализовать переключательную функцию m=8 аргументов, записанную в СДНФ следующим образом:

$$Y(x)=X_1\,X_3\,X_5\,X_7+X_2'\,X_3\,X_4\,X_8'=Y_1(x)+Y_2(x)$$

Набор аргументов предварительно вводится в ячейку памяти с символическим обозначением DRGOM.

Значения каждого из двух дизъюнктивных членов $Y_1(x)$ и $Y_2(x)$ определим с помощью следующих операций:

Маскирование, т. е. установка в нулевое положение аргументов, которые не входят в данный член $Y_i(x)$. Это производится с помощью логическо-

го умножения на соответствующие маски Y_1 и Y_2, которым присвоим символические обозначения GAMMA 1 и GAMMA 2 , в данном примере Y1 = 10101010 и Y2=01110001.

Инвертирование тех аргументов, которые входят в данный член $Y_i(x)$ без знаков инверсии. Эта операция осуществляется путем суммирования по модулю два полученного ранее набора аргументов с константами б1 и б2, которые содержат «1» на позициях, соответствующих переменным без инверсии, и, «0» на позициях, соответствующих переменным либо отсутствующих в данном члене $Y_i(x)$, либо входящих туда с инверсией.

Анализ результата. Если результат выполнения описанных ранее операций равен 00000000, то данный член $Y_i(x)$ =1, в противном случае переходим к испытанию следующего члена в выражении для Y(x) либо принимаем решение, что $Y_i(x)$=0, если испытанный член $Y_i(x)$ был последним.

Метка	Команда	Адреса операндов	Комментарий
	LDA	ARGUM	Запись в аккумулятор содержимого ячейки ARGUM
	MOV	C, A	Дублирование числа в регистре C
	ANI	GAMMA 1	Маскирование отсутствующих в $Y_1(x)$ аргументов
	XRI	SIGMA 1	Инвертирование аргументов, которые входят в $Y_1(x)$ без знака инверсия
	JZ	M1	Переход, если A=0, т. е. Y(x)=1
	MOV	A, C	Восстановление содержимого аккумулятора
	ANI	GAMMA 2	Маскирование отсутствующих в $Y_2(x)$ аргумен-

	XRI	SIGMA 1	Инвертирование аргументов, которые входят в $Y_2(x)$ без знака инверсия
	JZ	M1	Переход, если A=0, т. е. Y(x)=1
	MVI	A, 0	Формирование нулевого сигнала, т.е. Y(x)=0
	JMP	M2	Переход к выводу сигнала
M1	MVI	A, 1	Формирование сигнала Y(x)=1
M2	STA	RESULT	Занесение содержимого A в ячейку RESULT

3. Задания для подготовки.

1. Изучить группы логических команд и команд условной передачи управления.

2. Ознакомиться с примерами, приведенными в лабораторной работе, и подготовить их для введения в УМК

3. Изучить организацию циклических программ и роль в них команд условной передачи управления.

4. Уяснить роль программных счетчиков и способы формирования адресов команд при обработке массивов данных.

4. Задания к лабораторной работе.

1. Исследовать программу первого примера:

- ввести в память УМК программу начиная с адреса, указанного преподавателем
- ввести исходные данные, полученные у преподавателя
- осуществить запуск программы
- осуществить чтение результата
- выполнить повторный запуск программы, считывая содержимое РОН после выполнения каждой команды

2. Исследовать программу второго примера (порядок выполнения аналогичен предыдущему)

3. Исследовать программу реализации логической функции

Вопросы для самопроверки.

1. Понятие алгоритма, свойства алгоритма
2. Условное изображение алгоритмов
3. Основные структуры в разработке алгоритмов
4. Регистр признаков, назначение и формат команд
5. Команды условного перехода
6. Относительная адресация с использованием индексного регистра для работы с массивами и таблицами
7. С помощью каких команд производиться маскирование данных
8. Программная реализация комбинационных логических схем
9. Циклические программы

ЛИТЕРАТУРА

1. Электронные вычислительные машины, в 8-ми кн. Учеб. Пособие для вузов /Под ред. А.Я. Савельева. Кн. 3 Алгоритмизация и основы программирования / Г.И. Светозаров. – М.: Высш. шк., 1987. – 128 с.

2. Микропроцессоры. В 3-х кн. Средства отладки. Лабораторный практикум и задачник. Учеб. для вузов /Под ред. Л.Н. Преснухина. – М.: Высш. шк., 1986. – 383 с.

ПРИЛОЖЕНИЕ

НАБОР КОМАНД МИКРОПРОЦЕССОРА КР580ИК80

Символы и сокращения

A – аккумулятор (регистр A);
ADR -16-битовый операнд;
D8 – однобайтовый операнд;
R – один из регистров A, B, C, D, E, H, L;
M – ячейка памяти, адрес которой указан в регистрах H, L;
YZ – регистровая пара (BC, DE, HL, SP);
SP – 16-битовый регистр указателя СТЕКа;
PC – 16-битовый регистр программного счетчика;

PSW – двухбайтовое содержимое регистров A с F;
N – номер порта ввода/вывода.

Мнемокод	Байты	Циклы	Такты	Выполняемая операция	Комментарий	Признаки s z ac p c
Команды пересылки						
MOV R1,R2	1	1	5	Пересылка данных из регистра в регистр	R1←R2 Содержимое регистра R2 пересылается в регистр R1	- - - - -
MOV R,M	1	2	7	Пересылка данных из памяти	R← ЯП(H,L) Содержимое ЯП, адрес которой хранится в регистрах H,L, пересылается в регистр R	- - - - -
MOV M, R	1	2	7	Пересылка данных в память	ЯП(H,L) ←R Содержимое регистра R пересылается в ЯП, адрес которой содержится в регистре H,L	- - - - -
MVI R,D8	2	2	7	Непосредственная пересылка	R1←D8 Второй байт команды (D8) пересылается в регистр R	- - - - -
MVI M,D8	2	3	10	Непосредственная пересылка в память	ЯП(H,L)←D8 Второй байт команды (D8) пересылается в ЯП(H,L), адрес которой указан в регистрах H,L	- - - - -
LXI IZ, D16	3	3	10	Непосредственная загрузка пары регистров	YZ←D16 Второй и третий байты команды пересылаются в пару регистров YZ	- - - - -

LDA ADR	3	4	13	Прямая загрузка аккумулятора	A←ЯП(ADR) Содержимое ЯП, адрес которой указан в команде (ADR) пересылается в A	- - - - -
STA ADR	3	4	13	Прямая запись содержимого A в память	ЯП(ADR) ←A Содержимое A пересылается в ЯП по указанному в команде адресу	- - - - -
LHLD ADR	3	5	16	Прямая загрузка регистров	L←ADR H←(ADR+1) Содержимое ячейки памяти (ADR) пересылается в регистр L; Содержимое ЯП со следующим адресом (ADR+1) пересылается в регистр H	- - - - -
LDAX YZ	1	2	7	Косвенная загрузка аккумулятора	A←ЯП(YZ) Содержимое ЯП, адрес которой указан в паре регистров YZ помещается в A	- - - - -
STAX YZ	1	2	7	Косвенная запись содержимого аккумулятора в память	ЯП(YZ) ←A Содержимое A пересылается в ЯП, адрес которой указан в паре регистров YZ	- - - - -
XCHG	1	1	4	Обмен данными между регистрами H,L и D,E	(H)↔(D) (L)↔(E) Содержимое регистров H,L обмениваются содержимым регистров D,E	- - - - -
Арифметические команды						
ADD R	1	1	4	Сложение содержимого A с содержимым	A←(A+R) Результат помещается в A	+ + + + +

				регистра		
ADD M	1	2	7	Сложение содержимого А с содержимым ЯП	А←[Ф+ЯП(H,L)] Содержимое ЯП, адрес которой указан в (H,L), складывается с содержимым А. Результат помещается в А	+ + + + +
ADI D8	2	2	7	Непосредственное сложение	А←(А+D8) Содержимое второго байта команды складывается с содержимым А. Результат помещается в А.	+ + + + +
SUB M	1	2	7	Вычитание содержимого ЯП	А←[А-ЯП(H,L)] Содержимое ЯП с адресом (H,L) вычитается из содержимого А.Результат помещается в А.	+ + + + +
SUI D8	2	2	7	Непосредственное вычитание	А←(А-D8) Содержимое второго байта команды вычитается из А. Результат помещается в А.	+ + + + +
SBB R	1	1	4	Вычитание содержимого регистра и бита флага переноса	А←(А-R-C) Содержимое регистра R и бит флага переноса С вычитаются из А. Результат помещается в А.	+ + + + +
SBB M	1	2	7	Вычитание содержимого ЯП и бита флага переноса	А←[А-ЯП(H,L)-C] Содержимое ЯП с адресом (H,L) и бит флага переноса С вычитаются из А.Результат помещается в А.	+ + + + +
SBI D8	2	2	7	Непосредственное вычитание с займом	А←(А-D8-C) Содержимое второго байта команды и бит флага переноса	+ + + + +

					С вычитаются из аккумулятора. Результат в регистре А.	
ADC R	1	1	4	Сложение содержимого А с содержимым регистра R и битом флага переноса	А←(A+R+C) Содержимое регистра R и бит флага переноса С складывается с содержимым А. Результат помещается в А.	+ + + + +
ADC M	1	2	7	Сложение содержимого А с содержимым ЯП и битом флага переноса	А←[A+ЯП(H,L)+C] Содержимое ЯП с адресом (H,L) и содержимое бита флага переноса складывается с содержимым А. Результата помещается в А.	+ + + + +
ACI D8	2	2	7	Непосредственное сложение с битом флага переноса	А←(A+D8+C) Содержимое второго байта команды и бита флага переноса складываются с содержимым А. Результат помещается в А.	+ + + + +
SUB R	1	1	4	Вычитание содержимого регистра	А←(A-R) Содержимое регистра R вычитается из А. Результат помещается в А.	+ + + + +
INR R	1	1	5	Увеличение содержимого регистра R	R←(R+l) Содержимое регистра R увеличивается на единицу.	+ + + + +
INR M	1	3	10	Увеличение содержимого ЯП	ЯП(H,L) ←[ЯП(H,L)+1] Содержимое ЯП с адресом (H,L) увеличивается на единицу.	+ + + + -
DCR R	1	1	5	Уменьшение содержимого	R←(R-l) Содержимое реги-	+ + + + -

				регистра R	стра R уменьшается на единицу.	
DCR M	1	3	10	Уменьшение содержимого ЯП	ЯП(H,L) ←[ЯП(H,L)-1] Содержимое ЯП с адресом (H,L) уменьшается на единицу.	+ + + + -
INX YZ	1	1	5	Увеличение содержимого пары регистров	YZ←(YZ+1) Содержимое пары регистров увеличивается на единицу.	- - - - -
DCX YZ	1	1	5	Уменьшение содержимого пары регистров	YZ←(YZ+1) Содержимое пары регистров уменьшается на единицу.	- - - - -
DAD YZ	1	3	10	Сложение содержимого пары регистров с содержимым регистров H,L	(H,L) ←[(H,L)+YZ] Содержимое пары регистров складывается с содержимым регистров H,L. Результат помещается в пару регистров H,L.	- - - - +
DAA	1	1	4	Десятичное дополнение аккумулятора	8-битовое число A дополняется до представления в виде двух 4-битовых чисел в двоично-десятичном коде с помощью спец. операций	+ + + + +
Логические команды						
ANA R	1	1	4	Операция «И» над содержимым регистра и аккумулятора	A←(A^R) Результат помещается в A.	+ + 1 + 0
ANA M	1	2	7	Операция «И» над содержимым ЯП(H,L) и A	A←[A ∧ ЯП (H,L)] Результат помещается в A.	+ + 0 + 0
ANI D8	2	2	7	Непосредст-	A←(A ∧ D8)	+ + 0 + 0

				венная операция «И»	Над содержимым А и второго байта (D8) выполняется логическая операция «И». Результат помещается в А.	
XRA R	1	1	4	Операция «исключающее ИЛИ» над содержимым R и А	А←(А⊽ R) Результат помещается в А	+ + 0 + 0
XRA M	1	2	7	Операция «исключающее ИЛИ» над содержимым ЯП и А	А←[А⊽ ЯП(H,L)] Результат помещается в А	+ + 0 + 0
XRI D8	2	2	7	Непосредственная операция исключающее ИЛИ	А←(А⊽ D8) Результат помещается в А	+ + 0 + 0
ORA R	1	1	4	Логическая операция «сложение (ИЛИ)» содержимого А с содержимым регистра	А←(А ∨ R) Результат помещается в А	+ + 0 + 0
ORA M	1	2	7	Логическая операция «ИЛИ» содержимого ЯП (H,L) и А	А←[А ∨ ЯП(H,L)] C=0; AC=0 Результат помещается в А	+ + 0 + 0
ORI D8	2	2	7	Логическая операция «ИЛИ»	А←(А ∨ D8) C=0; AC=0	+ + 0 + 0
CMP R	1	1	4	Сравнение содержимого регистра с содержимым А	А – R Содержимое R сравнивается с содержимым А. А не изменяется. По результату сравнения устанавливаются флаги: Z=1, если A=R;	+ + + + +

					C=1, если A<R.	
RAL	1	1	4	Сдвиг влево через перенос	$A_{n+1} \leftarrow A_n$ $C \leftarrow A_7$ $A_0 \leftarrow C$; Содержимое A сдвигается на одну позицию влево через бит флага переноса C. Младший бит устанавливается равным флагу переноса, а бит флага переноса C равным старшему биту A.	- - - - +
RAR	1	1	4	Сдвиг вправо через перенос	$A_n \leftarrow A_{n+1}$ $C \leftarrow A_0$ $A_7 \leftarrow C$ Содержимое A сдвигается на одну позицию вправо через бит флага переноса C. Старший бит A=C, а C = младшему биту A	- - - - +
CMA	1	1	4	Дополнение аккумулятора	$A \leftarrow \overline{A}$ Содержимое A инвертируется (бит, равный единице, становится равным нулю; бит, равный нулю – единице)	- - - - -
CMC	1	1	4	Дополнение флага переноса	$C \leftarrow \overline{C}$ Инвертируется бит флага переноса C	- - - - +
CMP M	1	2	7	Сравнение содержимого ЯП(H,L) с содержимым A	A – (ЯП(H,L)) Содержимое ЯП, адрес которой указан в регистрах H,L, сравнивается с содержимым A. A не изменяется. По результатам сравнения устанавливаются флаги: Z = 1, если A =	+ + + + +

					ЯП(H,L); C =1, если A < ЯП(H,L)	
CPI D8	2	2	7	Непосредственное сравнение	A – D8 Содержимое второго байта команды вычитается из содержимого A. По результатам устанавливаются флаги: Z = 1, если A = D8; C = 1, если A < D8	+ + + + +
RLC	1	1	4	Циклический сдвиг влево	$A_{n+1} \leftarrow A_n$ $A_0 \leftarrow A_7$ $C \leftarrow A_7$; Содержимое A сдвигается влево на одну позицию. Содержимое старшего бита заносится в младший бит и бит флага переноса.	- - - - +
RRC	1	1	4	Циклический сдвиг вправо	$A_n \leftarrow A_{n+1}$ $A_7 \leftarrow A_0$ $C \leftarrow A_0$ Содержимое A сдвигается на одну позицию вправо. Содержимое младшего бита заносится в самый старший бит и бит флага переноса	- - - - +
STC	1	1	4	Установка флага переноса	C ← 1 Бит флага переноса устанавливается = 1	- - - - 1

Команды переходов						
JMP ADR	1	3	10	Переход безусловный	PC←ADR Управление передается команде, адрес которой указан во втором и третьем байтах команды перехода	- - - - -
J (условие) ADR, например: JNC ADR JP ADR	3	3	10	Условие перехода	Если указанное условие истинно, то управление передается команде, адрес который указан во втором и третьем байтах команды перехода. Если условие ложное, то последовательный ход программы не изменяется	
CALL ADR	3	5	17	Вызов	(SP – 1)← PCH; (SP – 2)← PCL; SP← (SP – 2); PC ADR Старше 8 бит адреса следующей команды пересылаются в ячейку памяти, адрес которой на единицу меньше содержимого указателя СТЕКа SP. Младшие 8 бит адреса следующей команды пересылаются в ЯП, адрес которой на 2 меньше содержимого указателя СТЕКа SP. Содержимое указателя СТЕКа уменьшается на 2. Управление передается команде, адрес которой указан во втором и третьем байтах команды вызова	
C (условие) ADR, например: CNZ ADR CP ADR	3	5	17	Условный вызов подпрограммы	Если указанное условие истинно, то выполняются действия, описанные в команде COLL, в противном случае последовательность выполнения команд не меняется	

RET	1	3	10	Возврат из подпрограммы	PCL←SP PCH← (SP + 1) SP←(SP + 2) Содержимое ЯП, адрес которой находится в указателе СТЕКа SP, пересылается в младшие 8 бит программного счетчика PC. Содержимое ЯП, адрес которой на 1 больше содержимого указателя СТЕКа пересылается в старшие 8 бит программного счетчика. Содержимое указателя СТЕКа увеличивается на 2.
R (условие) например: RPO RM	1	3	12	Условный возврат из подпрограммы	Если указанное условие истинно, то выполняются действия, описанные в команде RET, а в противном случае последовательность выполнения команд не нарушается.
RST n	1	3	12	Рестарт	(SP-1)←PCH; (SP-2)← PCL; SP←(SP-2) PC←8*NNN Старшие 8 бит адреса следующей команды пересылаются в ячейку памяти, адрес которой на единицу меньше содержимого указателя СТЕКа. Младшие 8 бит адреса следующей команды пересылаются в ЯП, адрес которой на 2 меньше содержимого указателя СТЕКа. Содержимое указателя СТЕКа уменьшается на 2.Управление передается команде, адрес которой соответствует коду NNN.

PCHL	1	1	6	Загрузка содержимого регистров H и L в программный счетчик	PCH← H; PCL ← L; Содержимое регистра H пересылается в старшие 8 бит программного счетчика PC. Содержимое регистра L пересылается в младшие 8 бит программного счетчика.
PUSH YZ	1	3	12	Засылка в СТЕК содержимого пары регистров	(SP-l) ←YZH; (SP-2) ← YZL; SP← (SP-2) Содержимое старшего регистра пары регистров YZH пересылается в ЯП, адрес которой на 1 меньше содержимого указателя СТЕКа SP. Содержимое младшего регистра пары регистров YZL пересылается в ЯП, адрес которой на 2 меньше содержимого указателя СТЕКа. Содержимое указателя СТЕКа уменьшается на 2.
PUSH PSW	1	3	12	Засылка в СТЕК слова состояния программы	(SP-1)← A (SP-2)0← C (SP-2)2← P (SP-2)4← AC (SP-2)6← Z (SP-2)7← S SP← (SP-2) Содержимое A пересылается в ЯП, адрес которой на 1 меньше содержимого указателя СТЕКа SP. Содержимое флагов объединяется в слово состояния, это слово пересылается в ЯП, адрес которой на 2 меньше содержимого указателя Стека. Содержимое указателя СТЕКа уменьшается на 2.

POP YZ	1	3	10	Считывание из СТЕКа содержимого пары регистров	YZL ←SP YZH← (SP+1) SP←(SP+2) Содержимое ЯП, адрес которой определяется содержимым указателя СТЕКа SP, пересылается в младший регистр пары регистров YZL. Содержимое ЯП, адрес которой на единицу больше содержимого указателя СТЕКа, пересылается в старший регистр пары регистров YZH. Содержимое указателя СТЕКа увеличивается на 2.
POP PSW	1	3	10	Считывание из СТЕКа слова состояния программы	C ← (SP)0; P ← (SP)2; AC ← (SP)4; Z ← (SP)6; S ← (SP)7; A ← (SP+1); SP ← (SP+2); Содержимое ЯП, адрес которой определяется содержимым указателя СТЕКа SP, используется для восстановления состояния флагов. Содержимое ЯП, адрес которой на единицу больше содержимого указателя СТЕКа, пересылается в A.Содержимое указателя СТЕКа увеличивается на 2
XTHL	1	5	16	Обмен содержимого верхушки СТЕКа и регистров H,L	L ← SP; H ← (SP+1) Содержимое регистра L обменивается на содержимое Я11, адрес которой содержится в указателе СТЕКа SP. Содержимое регистра H обменивается на содержимое ЯП, адрес которой на 1 больше содержимого указателя СТЕКа.

SPHL	1	1	6	Пересылка содержимого регистров H,L в указатель СТЕКа	SP ← HL; Содержимое регистров H,L (16 бит) пересылается в указатель СТЕКа
Команды ввода-вывода					
IN N	2	3	10	Ввод	A ← D8; Данные, имеющиеся в 8 бит двунаправленной шины данных указанного порта, пересылаются в A.
OUT N	2	3	10	Вывод	D8 ← A; Содержимое A помещается на двунаправленную шину данных для передачи в указанный порт.
Общие команды					
E_1	1	1	4	Разрешение прерываний	Система прерываний разрешается при выполнении следующей за EI команды
D_1	1	1	4	Запрещение прерываний	Система прерываний запрещается при выполнении следующей за D1 команды
HLT	1	1	5	Останов	В счетчик PC заносится адрес следующей команды. Процессор затем бездействует до прихода команды прерывания. Состояние регистров не изменяется
NOP	1	1	4	Нет операции	В счетчик PC заносится адрес, следующей команды, и микропроцессор переходит к её обработке. Состояние регистров не изменяется.

Примечание:

В командах условных переходов параметр «условие» может иметь следующий вид:

NZ – не ноль (Z = 0)

Z – ноль (Z = 1)

NC – нет переноса (C = 0)

C – есть перенос (C = 1)

PO – нечетный результат (P = 0)

PE – четный результат (P = 1)

P – плюс (S = 0)
M – минус (S = 1).

НАШИННЫЕ КОДЫ КОМАНД МП КР580ИК80

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	NOP	LXI B,&	ctax B	INX B	INR B	DCR B	MVI B,#	RLC	-	DAD B	ldax B	DCX B	INR C	DCR C	MVI C, #	RRC
1	-	LXI D,&	stax D	INX D	INR D	DCR D	MVI D,#	RAL	-	DAD D	ldax D	DCX D	INR E	DCR E	MVI E, #	RAR
2	-	LXI H, &	Shld *	INX H	INR H	DCR H	MVI H, #	DAA	-	DAD H	lhld *	DCX H	INR L	DCR L	MVI L, #	CMA
3	-	LXI SP,&	STA *	INX SP	INR M	DCR M	MVI M,#	STC	-	DAD SP	LDA *	DCX SP	INR A	DCR A	MVI A,#	CMC
4	MOV B,B	MOV B,C	MOV B,D	MOV B,E	MOV B,H	MOV B,L	MOV B,M	MOV B,A	MOV C,B	MOV C,C	MOV C,D	MOV C,E	MOV C,H	MOV C,L	MOV C,M	MOV C,A
5	MOV D,B	MOV D,C	MOV D,D	MOV D,E	MOV D,H	MOV D,L	MOV D,M	MOV D,A	MOV E,B	MOV E,C	MOV E,D	MOV E,E	MOV E,H	MOV E,L	MOV E,M	MOV E,A
6	MOV H,B	MOV H,C	MOV H,D	MOV H,E	MOV H,H	MOV H,L	MOV H,M	MOV H,A	MOV L,B	MOV L,C	MOV L,D	MOV L,E	MOV L,H	MOV L,L	MOV L,M	MOV L,A
7	MOV M,B	MOV M,C	MOV M,D	MOV M,E	MOV M,H	MOV M,L	HLT	MOV M,A	MOV A,B	MOV A,C	MOV A,D	MOV A,E	MOV A,H	MOV A,L	MOV A,M	MOV A,A
8	ADD B	ADD C	ADD D	ADD E	ADD H	ADD L	ADD M	ADD A	ADC B	ADC C	ADC D	ADC E	ADC H	ADC L	ADC M	ADC A
9	SUB B	SUB C	SUB D	SUB E	SUB H	SUB L	SUB M	SUB A	SBB B	SBB C	SBB D	SBB E	SBB H	SBB L	SBB M	SBB A
A	ANA B	ANA C	ANA D	ANA E	ANA H	ANA L	ANA M	ANA A	XRA B	XRA C	XRA D	XRA E	XRA H	XRA L	XRA M	XRA A
B	ORA B	ORA C	ORA D	ORA E	ORA H	ORA L	ORA M	ORA A	CMP B	CMP C	CMP D	CMP E	CMP H	CMP L	CMP M	CMP A
C	RNZ	POP B	JNZ *	JMP *	CNZ *	push B	ADI #	RST 0	RZ	RET	JZ *	-	CZ *	call *	ACI *	RST 1
D	RNC	POP D	JNC *	OUT N	CNC *	push D	SUI #	RST 2	RC	-	JC *	IN N	CC *	-	SBI #	RST 3
E	RPO	POP H	JPO *	xthl	CPO *	push H	ANI #	RST 4	RPE	pchl	JPE *	xchg	CPE *	-	XRI #	RST 5
F	RP	POP PSW	JP *	DI	CP *	Push PSW	ORI #	RST 6	RM	sphl	JM *	EI	CM *	-	CPI #	RST 7

ТЕОРЕТИЧЕСКИЕ ОСНОВЫ

1. Архитектура микропроцессоров Intel (8086)

1.1. Структура микропроцессора

На рис. 1 представлена структурная схема микропроцессора 8086, в состав которого входят:

устройство управления (УУ);

арифметико-логическое устройство (АЛУ);

блок преобразования адресов;

регистры.

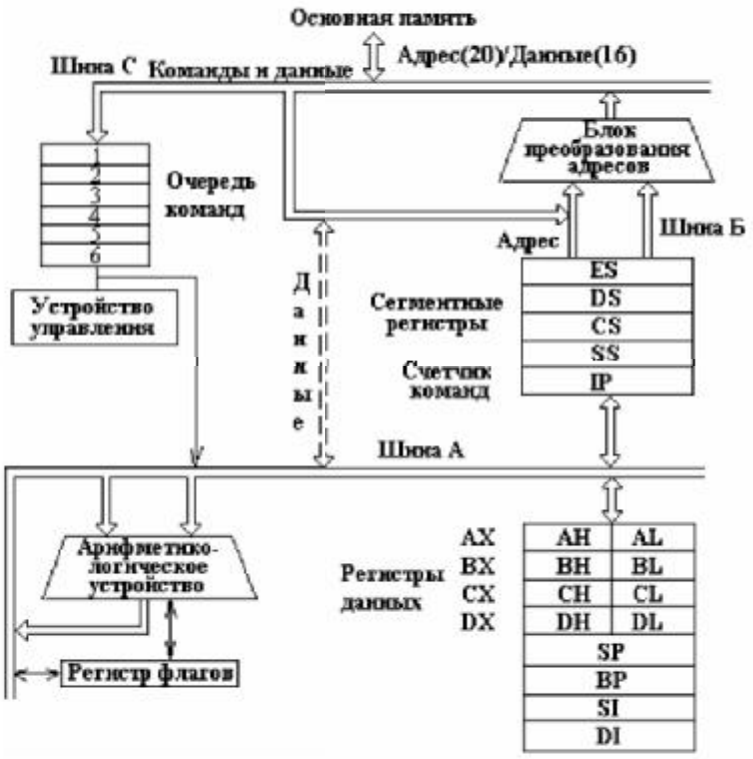


Рис. 1 – Структура микропроцессора i8086

Устройство управления дешифрирует коды команд и формирует необходимые управляющие сигналы. Арифметико-логическое устройство осуществляет необходимые арифметические и логические преобразования данных. В блоке преобразования адресов формируются физические адреса данных, расположенных в основной памяти. Регистры используются для хранения управляющей информации: адресов и данных.

Всего в состав микропроцессора i8086 входит четырнадцать 16-битовых регистров (см. рис. 1). Их можно условно разбить на четыре группы:

а) Регистры общего назначения (РОН):

AX – регистр-аккумулятор,
BX – базовый регистр,
CX – счетчик,
DX – регистр-расширитель аккумулятора;

б) Регистры адреса:
SI – регистр индекса источника,
DI – регистр индекса результата,
BP – регистр-указатель базы;

в) Управляющие регистры:
SP – регистр-указатель стека,
IP – регистр-счетчик команд,
F – регистр флагов;

г) четыре сегментных регистра:
CS – регистр сегмента кодов,
DS – регистр сегмента данных,
ES – регистр дополнительного сегмента данных,
SS – регистр сегмента стека.

1.2. Организация основной памяти

Минимальной адресуемой единицей основной памяти ПЭВМ является байт, состоящий из 8 бит. Доступ к байтам основной памяти осуществляется по номерам (номер байта является ого физическим адресом в устройстве памяти).

Для адресации основной памяти в микропроцессоре i8086 предусматриваются 20-битовые адреса, что позволяет работать с основной памятью до 1 Мбайта.

Физический адрес формируется из 16-битового смещения и содержимого 16-битового сегментного регистра, сдвинутого влево на 4 бита (см. рис. 2).

Рис. 2 – Формирование физического адреса

Для размещения программ и данных в основной памяти выделяются специальные области – сегменты. Адреса этих областей хранятся в специальных сегментных регистрах.

Каждый из четырех сегментных регистров используется для хранения адреса определенного сегмента (см. рис. 3).

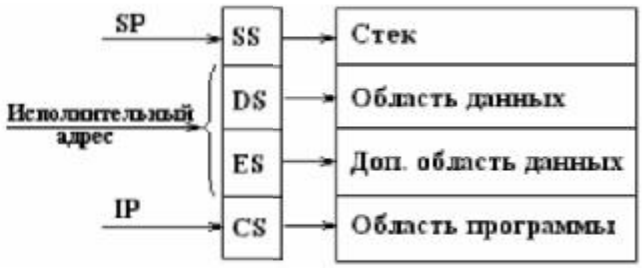


Рис. 3 – Назначение сегментных регистров

Стек представляет собой специальным образом организованную область памяти, допускающую последовательную запись элементов данных длиной 2 байта (слово) и чтение их в порядке, обратном порядку записи. Для хранения адреса последнего слова, занесенного в стек, служит регистр-указатель стека *SP* (см. рис. 4).

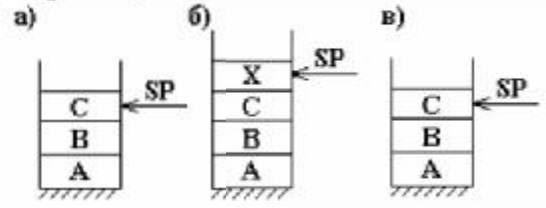


Рис. 4 – Работа со стеком

Здесь на рис. 4:
а) – текущее состояние стека,
б) – запись X,
в) – чтение X.

Стек используется для временного хранения данных и адресов, например при вызове подпрограмм, когда в стек заносится адрес возврата и значения параметров, передаваемых в подпрограмму.

Формат команд микропроцессора 8086 позволяет указывать в команде только один операнд, размещенный в основной памяти, т. е. одной командой нельзя, например, сложить содержимое двух ячеек памяти,

Принципиально допускается 8 способов задания смещения (исполнительного адреса) операндов, размещенных в основной памяти:

SI + <индексное смещение>
DI + <индексное смещение>
BP + <индексное смещение>
BX + <индексное смещение>
BP + *SI* + < индексное смещение>
BP + *DI* +< индексное смещение>
BX + *SI* + <индексное смещение>
BX + *DI* + <индексное смещение>

Во всех случаях исполнительный адрес операнда определяется как сумма содержимого указанных регистров и индексного смещения, представляющего собой некоторое число (одно- или двухбайтовое).

1.3. Выполнение программы

Содержимое регистров *CS* и *IP*, в которых хранится базовый адрес сегмента кодов и смещение очередной команды относительно начала сегмента, определяет физический адрес команды, которая должна быть выполнена на следующем шаге.

По указанному адресу из основной памяти считывается команда и пересылается в микропроцессор. Команда длиной от 1 до 8 байт помещается в очередь команд, откуда поступает в устройство управления, где дешифрируется.

Если при выполнении команды требуются данные, расположенные в основной памяти, то специальные полем кода команды определяется способ адресации и вычисляется исполнительный, и затем и физический адрес данных.

Данные, считанные из основной памяти по указанному адресу, пересылаются в регистр данных или в арифметико-логическое устройство и обрабатываются в соответствии с кодом команды. Результат помещается либо в регистры, либо (в соответствии с командой) в какую-либо область основной памяти.

Если выполненная команда не являлась командой передачи управления, то содержимое регистра *IP* увеличивается на длину выполненной команды, в противном случае в регистр *IP* заносится исполнительный адрес команды, которая должна выполняться следующей. И затем процесс повторяется.

1.4. Регистр флагов

На рис. 5 представлен флажковый регистр микропроцессора i8086, в котором в виде однобитовых признаков по принципу ДА – НЕТ (ВКЛЮЧЕНО – ВЫКЛЮЧЕНО) фиксируется информация о результатах выполнения некоторых команд, например арифметических.

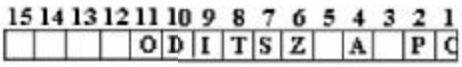


Рис. 5 – Регистр флагов

Значения битов регистра флагов (рис. 5) следующее:

O – признак переполнения;

D – признак направления;

I – признак прерывания;

T – признак трассировки;

S – признак знака: «1» – число «< 0», «0» – число «> 0»;

Z – признак нуля: «1» – число «= 0»;
A – признак переноса из тетрады;
P –признак четности;
C – признак переноса.

В последующем эта информация может использоваться, например, командами условной передачи управления.

2. Язык Ассемблер

2.1. Формат операторов

Операторы языка ассемблера ПЭВМ имеют следующий формат:

[< *метка* >:] < *код операции* > [< *список операндов* >] [< *комментарии* >]

Примечание. Здесь и далее по тексту:
1. выражение, находящееся между знаками «< … >» заменяется при составлении программы в кодах на какое либо название по усмотрению разработчика, причем сами символы «<» и «>» не указываются;
2. выражения присутствующие в «[…]» не обязательны, так же сами символы «[» и «]» не указываются.

Запись программы выполняется по свободному формату, т. е. специально не оговариваются правила заполнения каких бы то ни было позиций строки. Каждый оператор размещается на отдельной строке. Точка с запятой означает следование после нее комментариев, до конца строки.

Программа может записываться как заглавными, так и строчными буквами. Метку произвольной длины следует записывать с начала строки и отдалять от кода операции двоеточием, за которым может следовать произвольное количество пробелов (вплоть до конца строки).

Код операции должен отделяться от списка операндов хотя бы одним пробелом. Операнды отделяются один от другого запятой.

2.2. Определение участков памяти для размещения данных

Для определения данных в основной памяти и резервирования полей памяти под данные, размещаемые в основной памяти в процессе выполнения программы, используются следующие операторы:
DB – определить однобайтовое поле, *DW* – определить слово (двухбайтовое поле), *DD* – определить двойное слово (четырехбайтовое поле).
Формат команды:

$$[<\text{имя поля}>]\ \begin{cases} DB \\ DW \\ DD \end{cases}\ [<\text{количество}>]\ \begin{cases} DUP\ (\ [<\text{список чисел}>]\ ) \\ ? \end{cases},$$

где *<количество >* – количество полей памяти указанной длины, которое определяется данной командой (указывается, если определяется не одно поле памяти);

«?» – используется при резервировании памяти.

Пример 1. Записать в байт памяти десятичное число 23 и присвоить этому байту имя *a*:

```
a  db     23
```

Пример 2. Зарезервировать 1 байт памяти:

```
db       ?
```

Пример 3. Записать в слово памяти шестнадцатеричное число «1234»:

```
dw       1234H
```

Пример 4. Определить 31 байт памяти, повторяя последовательность «1, 2, 3, 4, 5, 1, 2, 3, 4,...»:

```
db31     dup   (1,2,3,4,5)
```

Примечание. При записи слов в память младший байт записывается в поле с младшим адресом. Например, в примере 3, если запись выполнялась начиная с адреса «100», то по адресу «100» будет записано «34H», а по адресу «101» – «12H».

2.3. Операнды команд

Операнды команд ассемблера могут определяться непосредственно в команде, находиться в регистрах или в основной памяти.

Данные, непосредственно записанные в команде, называются литералами. Так, в команде

```
mov      ah, 3
```

«3» – литерал.

Если операнды команд ассемблера находятся в регистрах, то в соответствующих командах указываются имена регистров (если используемые регистры особо не оговариваются для данной команды. Например, в приведенном выше примере *ah* - имя регистра аккумулятора.

Адресация операндов, расположенных в основной памяти, может быть прямой и косвенной.

При использовании прямой адресации в команде указывается символическое имя поля памяти, содержащего необходимые данные, например:

inc OPND

Здесь *OPND* – символическое имя поля памяти, определенного оператором ассемблера

OPND dw ?

При трансляции программы ассемблер заменит символическое имя на исполнительный адрес указанного поля памяти (смещение относительно начала сегмента) и занесет этот адрес на место индексного смещения. Адресация в этом случае выполняется по схеме: *BP + <индексное смещение>*, но содержимое регистра *BP* при вычислении исполнительного адреса не используется (частный случай).

В отличие от прямого косвенный адрес определяет не местоположение данных в основной памяти, а местоположение компонентов адреса этих данных. В этом случае в команде указываются один или два регистра в соответствии с допустимыми схемами адресации и индексное смещение, которое может задаваться числом или символическим именем. Косвенный адрес заключается в квадратные скобки весь или частично, например:

[OPND +SI]
OPND [SI]
OPND + [SI]
[OPND] +[SI]

Приведенные выше формы записи косвенного адреса интерпретируются одинаково.

При трансляции программы ассемблер определяет используемую схему адресации и соответствующим образом формирует машинную команду, при этом символическое имя заменяется смещением относительно начала сегмента так же, как в случае прямой адресации.

Примечание. При использовании косвенной адресации по схеме *BP + <индексное смещение>* индексное смещение не может быть опущено, так как частный случай адресации по данной схеме с нулевой длиной индексного смещения используется для организации прямой адресации. Следовательно, при отсутствии индексного смещения в команде следует указывать нулевое индексное смещение, т.е. *[BP + 0]* .

Пример. Зададим адрес двумя способами:

[a + bx]
[bp] + [si] +6.

В первом случае исполнительный адрес операнда определяется суммой содержимого регистра *bx* и индексного смещения, заданного символическим

именем "*a*", а во втором - суммой содержимого регистров *bp, si* и индексного смещения, равного «6».

Длина операнда может определяться:

а) кодом команды – в том случае, если используемая команда обрабатывает данные определенной длины, что специально оговаривается;

б) объемом регистров, используемых для хранения операндов (1 или 2 байта);

в) специальными указателями *byte ptr* (1 байт) и *word ptr* (2 байта), которые используются в тех случаях, когда длину операнда нельзя установить другим способом. Например,

```
mov     byte ptr x, 255
```

т. е. операнд пересылается в поле с именем "*x*" и имеет длину 1 байт.

2.4. Команды пересылки/преобразования данных

2.4.1. Команда пересылки данных

Формат команды передачи данных следующий:

MOV < адрес приемника >, < адрес источника >

Данная команда используется для пересылки данных длиной «1» или «2» байта из регистра в регистр, из регистра в основную память, из основной памяти в регистр, а также для записи в регистр или основную память данных, непосредственно записанных в команде. Все возможные пересылки представлены на рис. 6.

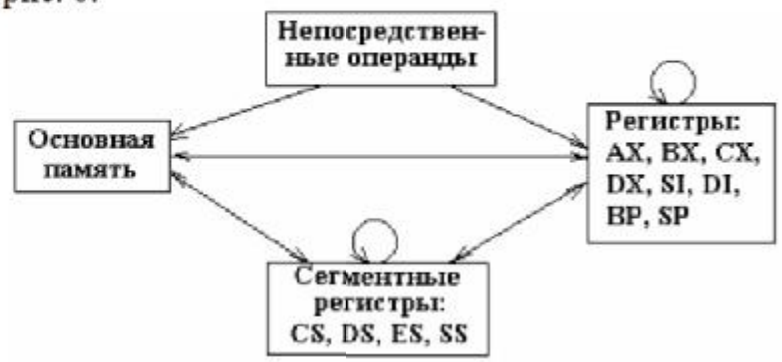


Рис. 6 – Возможности команды *MOV*

Примеры:

```
mov     ax, bx          ;пересылка содержимого регистра bx в
                         регистр ax
mov     cx, exword      ;пересылка 2 байт, расположенных в
                         поле exword, из основной памяти в
                         регистр cx
```

```
mov     si, 1000              ;запись числа 1000 в регистр si;
mov     word ptr [di+515], 4;запись числа 4 длиной 2 байта в
                              основную память по адресу [di+515]
```

Для загрузки «прямого» адреса в сегментный регистр используются две команды пересылки:

```
mov     ax, code
mov     ds, ax
```

2.4.2. Команда обмена данных.

Формат команды обмена данных следующий:

XCHG < операнд 1 >, < операнд 2 >

Данная команда организует обмен содержимого между двумя регистрами (кроме сегментных) или регистра и поля основной памяти.
Например:

```
xchg    bx, cx   ;обмен содержимого регистров bx и cx.
```

2.4.3. Команда загрузки исполнительного (эффективного) адреса.

Формат команды загрузки исполнительного адреса следующий:

LEA < операнд 1 >, < операнд 2 >

Данная команда вычисляет исполнительный адрес второго операнда и помещает его в область, на которое указывает первый операнд.

Примеры:

```
lea     bx, exword  ;в регистр bx загружается исполнительный ад-
рес
                    exword;
lea     bx, [di+10] ;в регистр bx загружается адрес 10-го байта
                    относительно  точки, на которую указывает
                    адрес в регистре di.
```

2.4.4. Команды загрузки указателя

Формат команды загрузки указателя следующий:

LDS < регистр >, < операнд 2 >
LES < регистр >, < операнд 2 >

Команда *LDS* загружает указатель (то есть полный адрес). При этом в указанный *< регистр>* загружается смещение, находящееся по адресу *< опе-*

ранд 2> а в *DS* загружается значение, находящееся по адресу < *операнд 2*> +2.

Команда *LES* делает те же операции, только теперь в качестве сегментного регистра используется дополнительный регистр ES.

Например:

lds ***si, exword***

2.4.5 Команда записи в стек

Формат команды записи в стек следующий:

PUSH < *операд* >

Данная команда организует запись в стек либо непосредственно содержимого <*операнда*> или слова, адресом которого служит операнд.

Например;

push ***dx*** ;запомнить содержимое регистра *dx* в стеке;
push ***[si]*** ;запомнить содержимое ячейки памяти адресуемой смещением *si* в стеке.

2.4.6. Команда восстановления из стека

Формат команды извлечения из стека следующий:

POP < *операд* >

Данная команда организует восстановление из стека последнего записанного элемента и помещает его либо непосредственно в <*операнд*> либо по адресу, на который он указывает.

Например;

pop ***dx*** ;поместить в *dx* значение из стека.

2.4.7. Команды сложения

Формат команд сложения следующий:

ADD < *операнд* 1 >, < *операнд* 2 >
ADC < *операнд* 1 >, < *операнд* 2 >

Данные команда устанавливает флаги четности, знака результата, наличия переноса, наличия переполнения.

По команде *ADD* выполняется сложение двух операндов. Результат записывается по адресу первого операнда. По команде *ADC* также выполнятся сложение двух операндов, но к ним добавляется еще значение, записанное в бите переноса, установленном предыдущей командой сложения.

На рис. 7 показаны возможные способы размещения слагаемых, где:
а) операнды – слова;
б) операнды – байты.

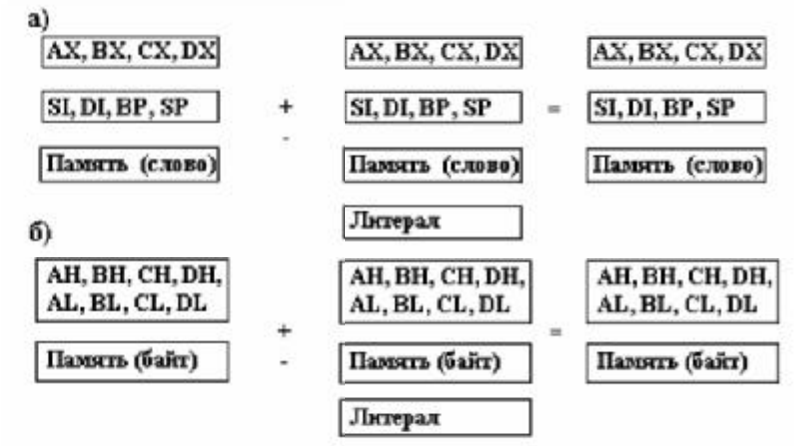


Рис. 7 – Размещение слагаемых

Приведем пример сложения двух 32-разрядных чисел:

```
mov     ax,value1
add     value2,ax
mo      ax,value1+2
adc     value2+2,ax
```

Исходные числа находится в основной памяти по адресам *value1* и *value2*, а результат записывается по адресу *value1*.

2.4.8. Команды вычитания

Формат команд вычитания следующий:

SUB < *уменьшаемое/ результат* >, < *вычитаемое* >
SBB < *уменьшаемое/ результат* >, < *вычитаемое* >

Данная команда устанавливает флаги четности, знака результата, наличия заема, наличия переполнения.

При выполнении операции по команде *SUB* заем не учитывается, а по команде *SBB* – учитывается. Ограничения на местоположение операндов такие же, как и у команды сложения.

2.4.9. Команда изменения знака

Формат команды следующий:

NEG < *операнд* >

Данная команда изменяет знак операнда на противоположный.

2.4.10. Команда добавления единицы

Формат команды следующий:

INC < *операнд* >

По данной команде значение операнда увеличивается на единицу.

2.4.11 Команда вычитания единицы

Формат команды следующий:

DEC < *операнд* >

По данной команде значение операнда уменьшается на единицу.

2.4.12 Команда сравнения

Формат команды следующий:

CMP < *операнд* 1 >, < *операнд* 2 >

По данной команде выполняется операция вычитания без записи результата, и устанавливаются признаки во флажковом регистре.

2.4.13. Команды умножения

Формат команд следующий:

MUL < *операнд* >

IMUL < *операнд* >

Данные команды устанавливают флаги наличия переноса или переполнения.

По команде *MUL* числа перемножаются без учета, и по команде – *IMUL* с учетом знака (в дополнительном коде).

На рис. 8 приведены возможные способы размещения сомножителей и результата. При этом один из сомножителей всегда расположен в аккумуляторе.

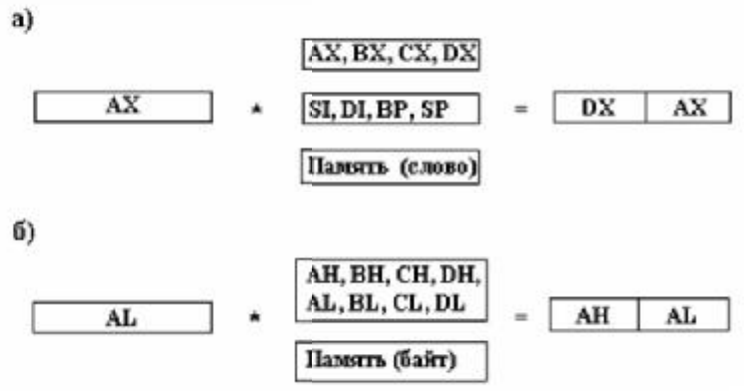


Рис. 8 – Способы размещения сомножителей

Здесь на рис. 8:

а) операнды – слова;

б) операнды – байты.

Рассмотрим пример:

imul word ptr c

В примере содержимое основной памяти по адресу «*c*» длиной в слово умножается на содержимое регистра *ax*. Младшая часть результата операции записывается в регистр *ax*, а старшая часть – в регистр *dx*.

2.4.14. Команда деления

Формат команд следующий:

DIV < *операнд* >

IDIV < *операнд* >

По команде *DIV* операция деления выполняется без учета, а по команде *IDIV* – с учетом знака (в дополнительном коде).

На рис. 9 приведены возможные способы размещения делимого, делителя и результата.

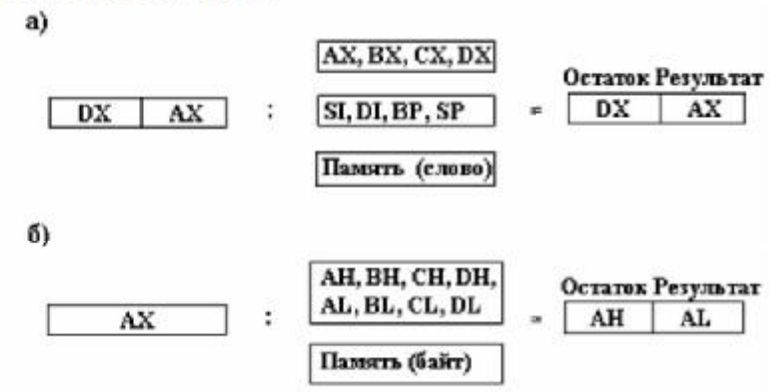


Рис. 9 – Способы размещения делимого, делителя и результата

Здесь на рис. 8:

а) операнды – слова;

б) операнды – байты.

2.4.15. Команда преобразования байта в слово, а слова – в двойное слово

Формат команд следующий:

CBW

CWD

По команде *CBW* число из *al* переписывается в *ax*. Аналогично по команде *CWD* число из *ax* переписывается в два регистра *dx* и *ax*.

2.5. Команды передачи управления

2.5.1. Команда безусловного перехода.

Формат команды следующий:

JMP < адрес перехода >

Данная команда имеет три модификации в зависимости от длины ее адресной части:

short – при переходе по адресу, который находится на расстоянии (-128…127) байт, относительно адреса данной команды (длина адресной части 1 байт);

near ptr – при переходе по адресу, который находится на расстоянии 32 Кбайта (-32768...32767 байт) относительно адреса данной команды (длина адресной части 2байта);

far ptr – при переходе по адресу, который находится на расстоянии превышающем 32 Кбайта (длина адресной части 4 байта).

При указании перехода к командам, предшествующим команде перехода, ассемблер сам определяет расстояние до метки перехода и строит адрес нужной длины.

При указании перехода к последующим частям программы необходимо ставить указатели *short, near ptr* и *far ptr*.

В качестве адреса команды перехода могут использоваться метки трех видов:

а) *< имя > : nop* (*nop* - команда «нет операции»);
б) *< имя> label near* (для внутрисегментных переходов);
в) *<имя> label far* (для вне сегментных переходов).

Приведем примеры:

```
jmp     short b     ;переход по адресу b
jmp     [bx]        ;переход по адресу в регистре bx (адрес
                     определяется косвенно)
a :     nop         ;описание метки перехода «a»
b       label near  ;описание метки перехода «b»
```

2.5.2. Команды условного перехода

Формат команды следующий:

JX < адрес перехода >

Здесь вместо *JX* необходимо указать необходимую команду, из списка, приведенного ниже:

JZ	переход если результат нулевой;
JE	переход если результат равный;
JNZ	переход если результат не нулевой;
JNE	переход если результат равный;
JL	переход если результат меньше;
JNG, JLE	переход если результат меньше или равный;
JG	переход если результат больше;
JNL, JGE	переход если результат больше или равный;

JA переход если результат больше по модулю;
JNA, JBE переход если результат не больше по модулю;
JB переход если результат меньше модулю;
JNB, JAE переход если результат не меньше по модулю.

Примечание: Все команды имеют однобайтовое поле адреса, следовательно, смешение не должно превышать -128...127 байт. Если смещение выходит за указанные пределы, то используется специальный прием:

вместо	программируется
jz zero	***jnz continue***
	jmp zero
	continue:

2.5.3. Команды организации циклической обработки

Примечание: в качестве счетчика цикла во всех командах циклической обработки используется содержимое регистра *cx*.

1) Команда организации цикла.

Формат команды следующий:

LOOP *< адрес перехода >*

При каждом выполнении данной команды уменьшается содержимое регистра *cx* на единицу, а управление при этом передается по указанному адресу, пока *cx* не равно «0»:

```
mov     cx, loop_count          ;загрузка счетчика
  begin_loop:
        ; тело цикла
  loop  begin_loop
```

Примечание: Если перед началом цикла в регистр *cx* загружен «0», то цикл выполняется 35536 раз.

2) Команда перехода по обнуленному счетчику.

Формат команды следующий:

JCXZ *< адрес перехода >*

При выполнении данной команды управление передается по указанному адресу, если содержимое регистра *cx* равно 0.

Например:

```
mov     cx, loop_count          ;загрузка счетчика
jcxz    end_of_loop             ;проверка счетчика
  begin_loop:
        ;тело цикла
  loop  begin_loop
  end_of_loop:
```

3) *Команды организации цикла с условием.*
Формат команд следующий:

LOOPE *< адрес перехода >*
LOOPNE *< адрес перехода >*

При выполнении данных команд содержимое *cx* уменьшается на единицу, управление передается по указанному адресу при условии, что содержимое *cx* отлично от нуля. В дополнение *LOOPE* дополнительно требует наличия признака «равно», а *LOOPNE* «'не равно», формируемых командами сравнения. Например:

```
mov     cx, loop_count          ;загрузка счетчика
jcxz    end_of_loop             ;проверка счетчика
  begin_loop:
        тело цикла ...
        cmp   al, 100           ;проверка содержимого al
        loopne      begin_loop  ;возврат в цикл, если cx>0 и al>100
  end_of_loop:
```

2.5.4. Команды вызова подпрограмм

1) Команда вызова процедуры.
Формат команды следующий:

CALL *< адрес перехода >*

Данная команда осуществляет передачу управления по указанному адресу, предварительно записав в стек адрес возврата.

При указании адреса процедуры, так же как и при указании адреса перехода в командах безусловного перехода, возникает необходимость определить удаленности процедуры от места вызова:

а) если процедура удалена не более чем на -128...127 байт, то специальных указаний не требуется;

б) если процедура удалена в пределах 32 кбант, то перед адресом по процедуры необходимо указать *near ptr*;

в) если процедура подпрограммы удалена более, чем на 32 кбайта, то перед адресом процедуры необходимо записать *far ptr*
Например:

```
call      near ptr p            ;вызов подпрограммы «p»
```

При этом текст процедуры оформляется в виде:

< имя процедуры > proc < указатель удаленности >

… ;тело процедуры …

< имя процедуры > end

Здесь указатель удаленности также служит дли определения длины адресов, используемых при обращении к процедуре: *near* – при использовании двухбайтовых адресов, *far* – при использовании четырехбайтовых адресов.

2) Команда возврата управления.

Формат команды следующий:

RET [< число >]

Данная команда извлекает из стека адрес возврата и передает управление по указанному адресу.

Если в команде указано значение счетчика, то после восстановления адреса возврата указанное число добавляется к содержимому регистра-указателя стека.

2.6. Команды обработки строк

Команды обработки строк используются для организации циклической обработки последовательностей элементов длиной «1» или «2» байта. Адресация операндов при этом выполняется с помощью пар регистров:

DS:SI – источник,

ES:DI – приемник.

Команды имеют встроенную корректировку адреса операндов согласно флагу направления *D*: «1» – уменьшение адреса на длину элемента, «0» – увеличение адреса на длину элемента. Корректировка выполняется после выполнения операции.

Установка требуемого значения флага направления выполняется специальными командами:

STD ;установка флага направления в единицу

CLD ;сброс флага направления в ноль

1) Команда загрузки строки LODS.

LODSB ;загрузка байта

LODSW ;загрузка слова

Команда загружает байт в *AL* или слово в *AX*. Для адресации операнда используются регистры *DS:SI*

2) Команда записи строки STOS.

STOSB ;запись байта

STOSW ;запись слова

Команда записывает в основную память содержимое *AL* или *AX* соответственно. Для адресации операнда используются регистры *ES:DI*.

3) Команда пересылки MOVS.

MOVSB ;пересылка байта

MOVSW ;пересылки слова

Команда пересылает элемент строки из области, адресуемой регистрами *DS:SI*, в область, адресуемую регистрами *ES:DI*.

4) Команда сканирования строки SCAS.

SCASB ;поиск байта

SCASW ;поиск слова

По команде содержимое регистра *AL* или *AX* сравниваются с элементом строки, адресуемым регистрами *DS:SI* и устанавливается значение флажков в соответствии с результатом *[DI] - AL* или *[DI]-AX*.

5) Команда сравнения строк CMPS.

CMPSB ;сравнение байт

CMPSW ;сравнение слов

По команде элементы строк, адресуемых парами регистров *DS:SI* и *ES:DI*, сравниваются и устанавливаются значения флажков в соответствии с результатом *[DI]-[SI]*.

6) Префиксная команда повторения.

Формат команды следующий:

REP < команда >

Команда позволяет организовать повторение указанной команды *CX* раз. Например:

rep stosb

Здесь поле, адресуемое парой регистров *ES:DI* длиной *CX* заполняется содержимым *AL* .

7) Префиксные команды «повторять, пока равно» и «повторять, пока не равно».

Формат команд следующий:

REPE < команда >

REPNE < команда >

Префиксные команды используются совместно с командами *CMPS* и *SCAS*. Префикс *REPE* означает повторять, пока содержимое регистра *cx* не равно нулю и значение флажка нуля равно единице, а *REPNE* – повторять, пока содержимое регистра *cx* не равно нулю и значение флажка нуля равно нулю.

2.7. Команда манипулирования битами

2.7.1. Логические команды

Формат команд следующий:

NOT *< операнд >* ;логическое НЕ

AND *< операнд1 >*, *< операнд2 >* ;логическое И

OR *< операнд1 >*, *< операнд2 >* ;логическое ИЛИ

XOR *< операнд1 >*, *< операнд2 >* ;исключающее ИЛИ

TEST *< операнд1 >*, *< операнд2 >* ;логическое И без записи результата

Пример: Выделить из числа в AL первый бит:

```
and     al, 10000000B
```

2.7.2. Команды сдвига

Формат команды следующий:

< код операции > *< операнд >*, *< счетчик >*

Счетчик записывается в регистр *cl*. Если счетчик равен «1», то его можно записать в команду.

Коды команд сдвига:

```
SAL     ;сдвиг влево арифметический
SHL     ;сдвиг влево логический
SAR     ;сдвиг вправо арифметический
SHR     ;сдвиг вправо логический
ROL     ;сдвиг влево циклический
ROR     ;сдвиг вправо циклический
RCL     ;сдвиг циклический влево с флагом переноса
RCR     ;сдвиг циклический вправо с флагом переноса
```

Например Умножить число находящееся в *ax* на 10:

```
mov     bx, ax
shl     ax, 1
shl     ax, 1
add     ax, bx
shl     ax, 1
```

2.8. Команды ввода – вывода

Обмен данными с внешней средой осуществляемся с помощью следующих команд:

IN *< регистр >*, *< порт >* ;ввод из порта в регистр

IN *< регистр >*, *< DX >* ;ввод из порта, номер которого ;указан в регистре *DX* в регистр

OUT *< порт >*, *< регистр >* ;вывод содержимого регистра в порт

OUT *< регистр >*, *< DX >* ;вывод содержимого регистра в порт, ;номер которого указан в *DX*

В качестве регистра можно указать *AL* или *AX* (соответственно будет обрабатываться байт или два байта). Порт отождествляется с некоторым внешним устройством (0...255).

Однако при организации ввода – вывода помимо самой операции необходимо осуществить ряд дополнительных действий, например, проверить готовность устройства. В связи с этим для типовых устройств разработаны стандартные программы организации ввода – вывода, которые вызываются по команде прерывания *int 21h*.

В таблице 1 приведен перечень основные функции, реализуемые подпрограммами ввода – вывода, и их коды. Код функции должен передаваться в подпрограмму в регистре **AH**.

Таблица 1 – Прерывания

Код функции	Функция
01	Ввод с клавиатуры одного символа в регистр *AL* (с проверкой на *Ctrl-Break*, с ожиданием, с озвучкой)
02	Вывод одного символа на экран дисплея из регистра *DL* (с проверкой на *Ctrl-Break*)
06	Непосредственный ввод – вывод: ввод в регистр *AL* (без ожидания, без озвучки, без проверки на *Ctrl-Break*, регистр *DL* должен содержать 0FFH), вывод из регистра *DL* (без проверки на *Ctrl-Break*).
07	Ввод в регистр *AL* (без проверки на *Ctrl-Break*, с ожиданием, без озвучки)
08	Ввод в регистр *AL* (с проверкой на *Ctrl-Break*, с ожиданием, без озвучки)
09	Вывод строки на экран (*DS:DX* – адрес строки, которая должна завершаться символом "$")
10(0Ah)	Ввод строки в буфер (*DS:DX* – адрес буфера, первый байт которого должен содержать размер буфера, после ввода – второй байт содержит количество введенных символов)
11(0Bh)	Чтение состояния клавиатуры (если буфер пуст, то *AL=0*, иначе *AL=0FFh*)

Примеры:

```
а)      mov  ah, 1          ;номер функции
        int  21h            ;ввод символа: символ в AL

б)      mov  ah, 2          ;номер функции
        mov  dl, 'A'
        int  21h            ;вывод символа из DL

в)      lea  dx, STRING     ;адрес буфера ввода
```

```
        mov  ah, 0Ah          ;номер функции
        int  21h              ;ввод строки: во втором байте буфера
                              ;количество введенных символов, далее в
                              ;буфере символы
STRING  db 50, 50 dup (?)
```

г)
```
        lea  dx, MSG          ;адрес выводимой строки
        mov  ah, 9            ;номер функции
        int  21h              ;вывод строки
        ...
MSG     db   'Пример вывода', 13, 10, 'S'
```

ЛАБОРАТОРНАЯ РАБОТА № 1

«Основные приемы программирования на Ассемблере для ПК»

Цель работы:

1. Изучение структуры программы на Ассемблере.
2. Изучение способов программирования ветвлений.
3. Изучение способов программирования циклов.
4. Изучение способов обработки массивов.

Краткие теоретические сведения

1. Структура программы на ассемблере

Структура программы на языке ассемблера выглядит следующим образом (**.exe**):

```
                    TITLE  <имя программы>
<имя сегмента стека>   SEGMENT  STACK
                       DB   3000  DUB   (?)
<имя сегмента стека>   ENDS
<имя сегмента данных > SEGMENT
                    <данные>
<имя сегмента данных > ENDS
<имя сегмента кодов>   SEGMENT
                    ASSUME  CS: <имя сегмента кодов>,  DS:<имя сегмента
данных>
                    EXTRN  <имя внешней процедуры >:<тип>
                    PUBLIC <имя внутренней процедуры>
<имя основной процедуры> PROC FAR
                       PUSH  DS
                       MOV   AX, 0
                       PUSH  AX
                       MOV   AX, <имя сегмента данных>
                       MOV   DS, AX
```

<тело процедуры>
RET
<имя основной процедуры> ***ENDP***
<имя внутренней процедуры> ***PROC NEAR***
<тело внутренней процедуры>
<имя внутренней процедуры> ***ENDP***
<имя сегмента кодов> ***ENDS***
END *<имя основной процедуры >*

Первая строка программы - заголовок, состоящий из служебного слова ***TITLE*** и имени программы.

Текст программы состоит из отдельных сегментов, каждый из которых начинается оператором ***SEGMENT*** и завершается оператором ***ENDS***:

<имя сегмента > ***SEGMENT***
... тело сегмента ...
<имя сегмента > ***ENDS***

Сегмент стека содержит специальный описатель ***STACK.***

Сегмент кодов, в котором располагается текст программы, начинается псевдокомандой ***ASSUME***, которая сообщает ассемблеру, какой сегментный регистр должен использоваться для адресации каждого сегмента.

За псевдокомандой ***ASSUME*** может следовать описание используемых программой внешних подпрограмм:

EXTRN *<имя внешней процедуры >:<тип* ***near*** *или* ***far****>*
PUBLIC *<имя внутренней процедуры>*

Сегмент кодов всегда адресуется сегментным регистром ***CS.*** Значение этого регистра операционная система устанавливает автоматически. Значения сегментного регистр ***DS*** загружается программистом:

MOV AX, *<имя сегмента данных>*
MOV DS, AX

При необходимости также загружается регистр **ES**:

MOV ES, AX

Команды

PUSH DS
MOV AX, 0
PUSH AX

организуют возможность возврата управления в MS DOS командой ***RET.***

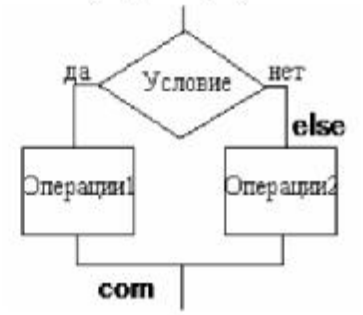
В этом случае в стек в качестве адреса возврата помещается адрес префиксной области программы, первые два байта которой содержат команду ***INT 20H*** возврата управления операционной системе.

2. Основные приемы программирования на Ассемблере.

Ассемблер, являясь языком низкого уровня не содержит операторов ветвления, циклов, не поддерживает автоматического формирования адресов для структур данных, не обеспечивает автоматического выполнения преобразований при вводе-выводе данных. Все перечисленные операции программируются "вручную" с использованием имеющихся команд ассемблера.

2.1. Программирование ветвлений.

Ветвления программируются с использованием команд условной и безусловной передачи управления.




```
        cmp   ...
        j<условие>    ELSE
    <операции 1>
        jmp   COM
ELSE:   <операции 2>
COM:    <продолжение>
```

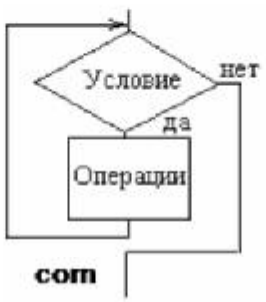
Пример: Написать процедуру вычисления ***X=max(A,B)***:

```
max     proc    near
        mov     ax, A
        cmp     ax, B     ; сравнение A и B
        jl      LESS      ; переход по меньше
        mov     X, ax
        jmp     CONTINUE  ; переход на конец ветвления
LESS:   mov     ax, B
        mov     X, ax
CONTINUE: ret
max     endp
```

2.2. Программирование циклических процессов.

Программирование циклических процессов осуществляется с использованием либо команд переходов, либо - в случае счетных циклов - с использованием команд организации циклов.

а) программирование итерационных циклов (цикл-пока):



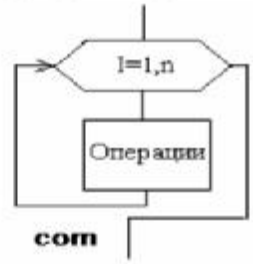

```
CYCL:   cmp  ...
        jne  COM
    <операции>
        jmp  CYCL
COM:    ....
```

Пример: Написать процедуру суммирования чисел от 1 до 10, используя итерационый цикл.

```
sum     proc    near
        mov     ax, 0     ; обнуление суммы
        mov     bx, 1     ; первое слагаемое
CYCL:   cmp     bx, 10    ; слагаемое больше 10
        jg      CONTINUE  ; выход из цикла
```

```
        add     ax, bx    ; суммирование
        inc     bx        ; следующее число
        jmp     CYCL      ; возврат в цикл
CONTINUE:  ret            ; выход, сумма - в ax
sum     endp
```

а) программирование счетных циклов:




```
        mov   cx, N
CYCL:   <операции>
        loop  CYCL
com:    .....
```

Пример: Написать процедуру суммирования чисел от 1 до 10, используя счетный цикл.

```
sum     proc    near
        mov     ax, 0     ; обнуление суммы
        mov     bx, 1     ; первое слагаемое
        mov     cx, 10    ; загрузка счетчика
CYCL:   add     ax, bx    ; суммирование
        inc     bx        ; следующее число
        loop    CYCL      ; возврат в цикл
continue:  ret            ; выход, сумма - в ax
sum     endp
```

2.3. Моделирование одномерных массивов.

Массив во внутреннем представлении - это последовательность элементов в памяти, например:

```
A       dw   10,13,28,67,0,-1   ; массив из 6 чисел длиной слово.
```

Программирование обработки выполняется с использованием адресного регистра, в котором хранится либо адрес текущего элемента, либо его смещение относительно начала массива. При переходе к следующему элементу адрес (или смещение) увеличивается на длину элемента.

Пример: Написать процедуру, выполняющую суммирование массива из 10 чисел размером слово.

Вариант 1 (используется адрес):

```
summas      proc
            mov     ax, 0
        lea     bx, MAS
        mov     cx, 10
CYCL:       add     ax, [bx]
        add     bx, 2
        loop    CYCL
```

Вариант 2 (используется смещение):

```
summas      proc
            mov     ax, 0
            mov     bx, 0
            mov     cx, 10
CYCL:       add     ax, MAS [bx]
            add     bx, 2
            loop    CYCL
```

```
        ret                                    ret
summas  endp                           summas  endp
```

Второй вариант позволяет получать более наглядный код и потому является предпочтительным.

В том случае, если элементы просматриваются непоследовательно, адрес элемента может рассчитываться по его номеру: $A_{исп} = A_{начала}$+(<номер>-1)*<длина элемента>. Полученный по формуле адрес записывается в один из адресных регистров (***BX, BP, DI, SI***) и используется для доступа к элементу.

Пример*:* Написать процедуру, которая извлекает из массива, включающего 10 чисел размером слово, число с номем n (n≤10).

```
n_mas   proc
        mov  bx, N     ; номер числа
        dec  bx        ; вычитаем 1
        sal  bx, 1     ; умножили на длину (сдвинули влево на 1)
        mov  ax, MAS [bx]  ; результат в ax
        ret
n_mas   endp
```

2.4. Моделирование матриц.

Значения матрицы могут располагаться в памяти по строкам и по столбцам. Для определенности будем считать, что матрица расположена в памяти построчно.

При моделировании обработки матрицы следует различать просмотр *по строкам*, *просмотр по столбцам*, *просмотр по диагоналям* и *произвольный* доступ.

Просмотр по строкам иногда может выполняться так, как в одномерном массиве (без учета перехода от одной строки к другой).

Пример: Написать процедуру определения максимального элемента матрицы A(3,5).

```
maxmatr   proc
          mov   bx, 0      ; смещение 0
          mov   cx, 14     ; счетчик цикла
          mov   ax, A      ; заносим первое число
CYCL:     cmp   ax, A[bx+2]  ; сравниваем числа
          jge   NEXT       ; если больше, то перейти к следующему
          mov   ax, A[bx+2]  ; если меньше, то запомнить
NEXT:     add   bx, 2      ; переходим к следующему числу
          loop  CYCL
          ret   ; результат в ax
maxmatr   endp
```

Просмотр по строкам при необходимости фиксировать завершение строки и просмотр по столбцам выполняются в двойном цикле: по строкам - во внешнем цикле, по столбцам - во внутреннем или наоборот. В этом случае обычно отдельно формируются смещения строки и столбца.

Пример**:** Определить сумму максимальных элементов столбцов матрицы A(3,5).

```
maxmatr     proc
            mov     ax, 0      ; обнуляем сумму
            mov     bx, 0      ; смещение элемента столбца в строке
            mov     cx, 5      ; количество столбцов
CYCL1:      push    cx         ; сохраняем счетчик
            mov     cx, 2      ; счетчик элементов в столбце
            mov     dx, A[bx]  ; заносим первый элемент столбца
            mov     si, 10     ; смещение второго элемента столбца
CYCL2:      cmp     dx, A[bx]+ [si]  ; сравниваем
            jge     NEXT       ; если больше или равно - к следующему
            mov     dx, A[bx]+[si]  ; если меньше, то сохранили
NEXT:       add     si, 10     ; переходим к следующему элементу
            loop    CYCL2      ; цикл по элементам столбца
            add     ax, dx     ; просуммировали макс. элемент
            pop     cx         ; восстановили счетчик
            add     bx, 2      ; перешли к следующему столбцу
            loop    CYCL1      ; цикл по столбцам
            ret                ; результат в ax
maxmatr     endp
```

2.5. Преобразования ввода-вывода.

При программировании операций ввода-вывода на ассемблере приходится вручную осуществлять преобразования чисел из символьного представления во внутренний формат (двоичный с фиксированной точкой, отрицательные числа записаны в дополнительном коде) и обратно.

Для облегчения преобразования во внутренний формат целесообразно оговорить возможные варианты ввода чисел в символьном виде. При этом также используется то, что при добавлении цифры к числу справа число меняется следующим образом: *<число>:=<число>*10+<цифра>*.

Обратное преобразование из внутреннего формата в символьный для вывода результатов обычно использует стандартное правило перевода числа из двоичной системы счисления в десятичную: деление на 10 с выделением остатков. В этом случае десятичные цифры получаются в обратном порядке.

Если среди вводимых или выводимых чисел могут быть отрицательные, то необходимо предусмотреть специальную проверку и преобразовывать отрицательные числа в дополнительный код при вводе и в прямой при выводе.

Ход работы

В ходе выполнения работы предлагается написать программу на Ассемблере, откомпилировать ее и проверить правильность работы. Код программы представлен ниже.

Задание: Написать процедуры ввода массива из ***n*** чисел размером слово и вывода того же массива. Числа должны вводиться каждое со своей строки, положительные числа должны вводиться без знака с первой позиции, от-

рицательные - со знаком в первой позиции. Вывод всех чисел должен осуществляться в одну строку через пробелы. Перед отрицательными числами необходимо выводить знак «-»

Код программы.

```
                title inout
code            segment
                assume cs:code, ds:code
N               equ     5     ; определяем константу для транслятора
A               dw      N dup (?)  ; резервируем место под массив
main            proc    far       ; основная процедура
                push    ds        ; обеспечиваем возврат управления в MS
DOS
                mov     ax, 0
                push    ax
                mov     ax, code  ;  загружаем сегментный адрес
                mov     ds, ax    ;        сегмента данных в DS
                call    input     ;  вызываем процедуру ввода
                call    output    ;  вызываем  процедуру вывода
                ret               ;  возврат управления DOS
main            endp              ; конец основной процедуры
input           proc    near      ; процедура ввода
                mov     cx, N     ; загрузка размерности массива
                mov     di, 0     ; загрузка смещения массива
CYCL1_IN:       push    cx        ; сохраняем счетчик внешнего цикла
                lea     dx, MES_IN  ; загружаем адрес запроса на ввод
                mov     ah, 9     ; загружаем номер функции DOS
                int     21h       ; вызываем функцию вывода строки
                lea     dx, BUF_IN  ; загружаем  адрес строки ввода
                mov     ah, 0ah   ; загружаем номер функции DOS
                int     21h       ; вызываем функцию ввода строки
                mov     byte ptr NEG_IN, 0  ; признак - "число положитель-
но"
                cld       ;    флаг направления - "возрастание адресов"
                mov     cl, BUF_IN+1  ; загружаем длину введенной стро-
ки
                mov     ch, 0   ; счетчик  - в CX
                lea     si, BUF_IN+2  ; загружаем адрес введенной стро-
ки
                cmp     byte ptr [si], '-' ;  число отрицательно ?
                jne     short UNSIGNED  ; если нет, то переход
                mov     byte ptr NEG_IN, 1  ; признак - "число отрицатель-
но"
                inc     si    ; пропускаем знак
                dec     cx    ; уменьшаем счетчик
UNSIGNED:       mov     bx, 0 ; исходное значение числа
```

```
CYCL2_IN:   mov   ax, 10  ; заносим константу 10
            mul   bx      ; умножаем текущее значение числа на 10
            mov   bx, ax  ; текущее значение числа в BX
            lodsb   ; загрузили очередную цифру
            sub   al, 30h  ; преобразовали из символа в двоичное чис-
ло
            cbw             ; преобразовали в слово
            add   bx, ax  ; добавили к текущему значению числа
            loop  CYCL2_IN  ; выполняем для всех введенных цифр
            cmp   NEG_IN, 0 ; число положительно ?
            je    short DONE   ; если да, то переход
            neg   bx  ; преобразуем в отрицательное число
DONE:       mov   A[di], bx  ; записываем результат в массив
            add   di, 2 ; корректируем смещение
            pop   cx  ; восстанавливаем счетчик внешнего цикла
            loop  CYCL1_IN ; выполняем для всех чисел
            ret   ; возвращаем управление основной процедуре
BUF_IN      db    10,10 dup (0)  ; буфер ввода (до 10 символов)
NEG_IN      db    0  ; признак "положительно/отрицательно"
MES_IN      db    13,10,'Введите число: $'  ; запрос на ввод
input       endp
output      proc  near  ; процедура вывода
            lea   dx, MES_OUT ; загружаем адрес заголовка вывода
            mov   ah, 9   ; загружаем номер функции DOS
            int   21h     ; вызываем функцию вывода строки
            mov   cx, N  ; загружаем количество чисел
            mov   si, 0  ; устанавливаем смещение в массиве
            mov   di, 7  ; устанавливаем смещение для строки вывода
cycl1_out:  mov   byte ptr NEGOUT, 0 ; признак - "число положительно"
            mov   bx, 10 ; загрузили константу 10
            mov   ax, A[si]  ; взяли очередное число из массива
            cmp   ax, 0  ; сравнили с нулем
            jge   short  AGAIN ; если положительно, то переход
            mov   byte ptr NEG_OUT, 1 ; признак "число отрицательно"
            neg   ax  ; преобразовали в положительное
AGAIN:      cwd   ; преобразовали в двойное слово
            div   bx  ; разделили на 10
            add   dl, 30h  ; преобразовали остаток в символ
            mov   BUF_OUT[di], dl ; записали в поле вывода
            dec   di  ; уменьшили смещение в поле вывода на 1
            cmp   ax, 0 ; сравнили результат деления с нулем
            jne   AGAIN  ; если не равно, то получаем следующую циф-
ру
            cmp   byte ptr NEG_OUT, 1 ; число отрицательно
            jne   short POSITIVE  ; если нет, то переход
```

```
        mov   BUF_OUT[di],  '-'  ; если да, то вставили знак "-"
POSITIVE: mov   ax,  N+2  ; считаем смещение следующего числа
        sub   ax,   cx          ; в поле вывода
        mov   bx,  7            ;
        mul   bx                ;
        mov   di,  ax           ;  записали смещение в DI
        add   si,   2        ;  перешли к следующему числу массива
        loop  CYCL1_OUT   ;  выполняем для всех чисел массива
        lea   dx,  BUF_OUT   ; загружаем адрес строки вывода
        mov   ah,  9     ; загружаем номер функции DOS
        int   21h      ; вызываем функцию вывода строки
        ret    ; возврат в основную процедуру
NEG_OUT   db  0  ;  признак "положительно/отрицательно"
BUF_OUT    db    13,10, N*7 dup (' '),'$'  ; поле вывода
MES_OUT   db    13,10,'Результат: $'  ; заголовок вывода
output      endp
code        ends
            end   main
```

Порядок действий при компиляции:

1. Создайте в любом текстом редакторе файл с кодом написанным выше.

2. Сохраните файл с расширением *.asm

4. Зайдите в каталог с комплектом программ TASM.

3. Запустите в командной строке MS DOS следующие наборы команд:

tasm /l <путь к вашему файлу /название_вашего_файла.asm>

После чего создастся объектный файл (расширение «obj») и файл листинга, который сдержит код программы и ошибки если они были.

Примечание: Если файл находится в том же каталоге что и tasm, то путь к нему указывать не надо. По умолчанию имя объектного файла и файла листинга те же что и у исходного файла.

tlink <имя.obj>

После чего создается рабочий файл.

ЛАБОРАТОРНАЯ РАБОТА № 2

«Обработка прерываний на Ассемблере для ПК»

Цель работы:

1. Изучение способов обработки прерываний.
2. Обработка прерываний от системного таймера.
3. Обработка прерываний нажатия клавиш клавиатуры.

Краткие теоретические сведения

Для IBM совместимых ЭВМ понятие прерывания является одним из базовых. Через систему прерываний в них реализованы доступ к внешним устройствам, взаимодействие со схемами контроля ЭВМ, управление процессом выполнения программы со стороны операционной системы, выполнение сервисных функций DOS и многое другое. Операционная система MS DOS позволяет пользователю разрабатывать собственные программы, которые могут вызываться через прерывания, дополняя и даже заменяя стандартные программы MS DOS.

1. Основные сведения о системе прерываний

Прерыванием называется временное переключение процессора на выполнение некоторой заранее определенной последовательности команд, после завершения которой процесс выполнения программы возобновляется.

Прерывания в IBM совместимых ЭВМ (см. Рисунок 1) могут инициироваться как специальными сигналами микропроцессора (внутренние), так и внешними сигналами, например, от внешних устройств (внешние).

Рис. 1 Система обработки прерываний.

Внешние сигналы на прерывание поступают на специальные микросхемы - контроллеры прерываний 8259A, имеющие 8 уровней приоритета. Начиная с PC AT ЭВМ включает два контроллера, что позволяет увеличить количество уровней приоритета до 16. Уровни приоритета определяются аппаратно в зависимости от места подключения внешнего устройства.

Как правило, самый высокий приоритет имеет системный таймер, затем следует клавиатура, что обеспечивает оперативное управление микроЭВМ, далее идут прочие внешние устройства. Завершают список обычно накопители на гибких магнитных дисках и устройство печати.

Прерывания, поступающие через контроллеры прерываний, также называют *аппаратными* в отличие от остальных, носящих название *программных*.

Выполнение всех прерываний, кроме прерывания 2, можно запретить установив в 0 флаг *IF* флажкового регистра. Прерывание 2 в связи с этим носит название *немаскируемого* и используется для того, чтобы выполнять обработку сигналов, наличие которых нельзя игнорировать, например, сигнала о недопустимом изменении напряжения в сети питания.

Адреса программ-обработчиков прерываний хранятся в специальной области основной памяти, которая обычно располагается с 0-го адреса и занимает 1кБ, то есть может содержать адреса 256 программ (каждый адрес занимает 4 байта). Местоположение адреса нужного обработчика в области векторов прерываний определяется по типу (номеру) прерывания:

$A=4*N$, где N - номер прерывания.

При наличии сигнала прерывания выполняется следующая последовательность действий:

1) проверяется установка флажка *IF* (для немаскируемых прерываний этот пункт игнорируется): *1* - прерывания разрешены, *0* - прерывания запрещены;

2) если прерывание разрешено, то после завершения выполнения текущей команды слово состояния программы(*PSW*), хранящееся во флажковом регистре микропроцессора, значение сегментного регистра кодов(*CS*) и значение счетчика команд(*IP*) заносятся в стек;

3) из области векторов прерываний в регистры *CS* и *IP* помещается адрес программы обработки прерываний.

После чего начинается выполнение программы обработки прерывания.

По завершению этой программы значения *PSW, CS* и *IP* восстанавливаются из стека, и выполнение прерванной программы возобновляется.

2. Классификация прерываний

Использование большинства прерываний определяется конкретным программным обеспечением и соответственно для каждой ЭВМ будет своим. Однако существуют номера прерываний, закрепленные за соответствующими функциями.

2.1. Прерывания микропроцессора

0 - прерывание от схем контроля микропроцессора – «Деление на 0»;

1 - прерывание специального режима работы микропроцессора, устанавливаемого, если флажок *TF*=1 – «Пошаговое выполнение»;

2 - немаскируемое прерывание;

3 - прерывание микропроцессора, осуществляемого при обнаружении адреса останова - «Точка останова»;

4 - инициируется по команде INTO, используемой после выполнения арифметической операции – «Переполнение»;

5 - печать содержимого экрана - инициируется нажатием клавиши *Print Screen*.

2.2. Прерывания микроконтроллера прерываний

8 - прерывание от таймера;

9 - прерывание от клавиатуры;

0BH - COM2;

0CH - COM1;
0EH - прерывание от НГМД (дискеты);
0FH - прерывание от печатающего устройства;
70H - прерывание от часов реального времени;
76H - прерывание от НЖМД (жесткий диск).

2.3. Процедуры BIOS

10H - управление дисплеем;
11H - определение конфигурации ПЭВМ;
12H - определение объема памяти ПЭВМ;
13H - управление дисковой памятью;
14H - управление асинхронной связью;
16H - управление клавиатурой;
17H - управление печатающим устройством;
1AH - управление часами реального времени.

2.4. Процедуры пользователя

1BH - возможность подключения при обнаружении *Ctrl-Break*;
1CH - возможность подключения к обработке кванта таймера.

2.5. Указатели системных таблиц

1DH - таблица параметров видео;
1EH - таблица параметров дискеты;
1FH - таблица символов для графического режима;
41H - таблица параметров жесткого диска.

2.6. Прерывания DOS

20H - нормальное завершение программы и возврат управления DOS;
21H - вызов диспетчера функций DOS;
22H - адрес пользовательской программы обработки нормального завершения программы;
23H - адрес пользовательской программы обработки завершения по *Ctrl-Break*;
24H - адрес пользовательской программы обработки завершения по ошибке;
25H - абсолютное чтение секторов с диска;
26H - абсолютная запись секторов на диск;
27H - завершение программы с сохранением в памяти.

2.7. Прерывания, зарезервированные для пользователей

Для определения конкретной конфигурации прерываний можно использовать, например, программу SYSINFO пакета Norton Utilities.

3. Структура обработчиков прерываний. Модификация области векторов прерываний.

Программа обработки прерывания в общем случае имеет следующую структуру:

```
<имя>        proc    far
        <сохранение содержимого регистров>
```

```
                    <обработка>
                    <восстановление содержимого регистров>
                              iret        ; возврат управления с
                                          ; восстановлением PSW,CS,IP
            <имя>             endp
```

Для подключения обработчика прерываний необходимо поместить его адрес в область векторов прерываний, а после завершения использования - восстановить старое содержимое вектора. Фрагменты программы, выполняющие эти операции приведены ниже.

```
                    title     inter
code                segment
                    assume    cs:code
old_adress          dw        2 dup (?) ;область для сохранения старого
                                        ;вектора
main                proc
                    .....
                    push      ES
                    mov       AH,35H    ;вызов функции DOS для
                    mov       AL,0FEH ; получения старого значения
                    int       21H       ; вектора прерывания
                    mov       old_adress,ES       ;  сохранение старого
                    mov       old_adress+2,BX     ;  значения вектора
                    push      DS
                    mov       DX,offset user  ; вызов функции DOS
                    mov       AX,seg user     ; для записи адреса
                    mov       DS,AX           ; пользовательской
                    mov       AH,25H          ; программы обработки
                    mov       AL,0FEH         ; прерывания в область
                    int       21H             ; векторов прерываний
                    pop       DS
                    .....
                    int       0FEH            ; вызов прерывания
                    .....
                    push      DS
                    mov       DX,old_adress+2 ; вызов функции DOS
                    mov       AX,old_adress   ; для восстановления
                    mov       DS,AX           ; старого адреса
                    mov       AH,25H          ; в области векторов
                    mov       AL,0FEH         ; прерываний
                    int       21H             ;
                    pop       DS
                    pop       ES
                    ........
main                endp
; обработчик прерываний
```

```
user            proc    far
                .....
                iret
user            endp
code            ends
end     main
```

В данном примере для получения и записи значений векторов прерывания использованы специальные функции прерывания *21H DOS* (*25H и 35H*). Прямая запись и прямое чтение области векторов прерываний не желательны, так как прямое обращение по данным адресам в некоторых операционных системах может привести к зависанию системы.

4. Пользовательские обработчики прерываний

В простейшем варианте пользовательские обработчики прерываний могут использоваться для вызова процедур, хотя такое их использование вряд ли целесообразно. Гораздо чаще пользовательские обработчики прерываний используются для замены или дополнения стандартных программ обработки прерываний, как программных, так и аппаратных. При этом возможны различные варианты подключения пользовательских программ.

1. Используется только пользовательская программа обработки прерывания.

В этом случае обработчик строится по схеме, приведенной в разделе 3, и завершается командой *IRET*. При программировании *аппаратных* прерываний перед *IRET* необходимо записать две команды, при выполнении которых сбрасываются флаги запрещения выполнения аппаратных прерываний более низкого уровня приоритета (если пишется обработчик прерывания уровней *8..15*, то добавляется третья команда, выполняющая те же действия для второго микроконтроллера):

```
mov     AL,20H
out     20H,AL
out     A0H,AL
```

2. Пользовательская программа выполняет предобработку прерывания и передает управление уже имеющейся программе обработки (см. **Ошибка! Источник ссылки не найден.**).

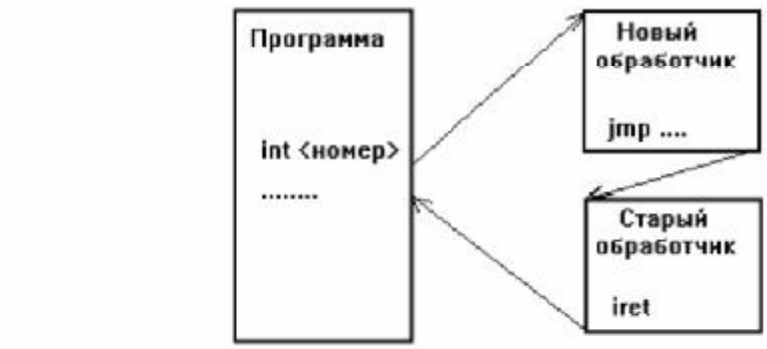


Рис. 2 – Вызов старого обработчика прерывания после нового

В этом случае пользователь, записывая адрес своей программы в область векторов прерываний, должен сохранить старое значение адреса в своей программе и после выполнения необходимых действий передать по нему управление:

jmp far CS:old_adress

Следовательно, команда *IRET* в программе пользователя в этом случае отсутствует.

3. Пользовательская программа вызывает старый обработчик прерываний из себя (см.**Ошибка! Источник ссылки не найден.**).

Рис. 3 Вызов старого обработчика прерываний из нового

В этом случае необходимо согласовать возврат управления из старого обработчика прерываний по *IRET* и в завершении выдать свой собственный *IRET*:

```
.......
pushf
call        far CS:old_adress
.......
iret
```

Согласование возврата по *IRET* выполняется за счет помещения в стек содержимого флажкового регистра перед вызовом старого обработчика прерывания. Таким образом, в стеке оказываются значения *PSW, CS* и адрес следующей команды. При выполнении *IRET* старого обработчика эти значения восстанавливаются во флажковый регистр, *CS* и *IP*, и продолжается выполнение пользовательской программы обработки прерывания.

Примечание. При желании можно использовать более сложный способ подключения пользовательской обработки, не требующий изменения соответствующего вектора. В этом случае любое место старого обработчика прерываний копируется в специальную область нового обработчика, и на это место пишется вызов нового обработчика. Получив управление, новый обработчик выполняет необходимые операции, затем выполняет скопированные команды и возвращает управление старому обработчику. Данный метод получил название «врезка» и в основном используется в вирусах и антивирусах.

При написании обработчиков прерываний следует иметь в виду, что при прерывании автоматически выдается команда *CLI*, устанавливающая в 0 флаг *IP* флажкового регистра, т.е. запрещающая маскируемые прерывания. Поэтому, если обработчик не меняет область векторов прерываний и не содержит фрагментов, выполнение которых жестко рассчитано по времени, по-

лучив управление желательно выдать команду *STI*, разрешив, таким образом, остальные прерывания.

Ход работы

В ходе выполнения работы предлагается написать две программы на Ассемблере, откомпилировать их и проверить правильность работы. Код программ представлен ниже.

Задание 1: Написать программу демонстрации перехвата прерывания системного таймера по выводу текущего времени в левом углу экрана

Код программы.

```
.model      tiny
.code
.186                    ; для pusha/popa и сдвигов
org    100h
start  proc   near
; сохранить адрес предыдущего обработчика прерывания 1Ch
       mov    ax,351Ch    ; AH = 35h, AL = номер прерывания
       int    21h         ; функция DOS: определить адрес обработчика
       mov    word ptr old_int1Ch,bx        ; прерывания
       mov    word ptr old_int1Ch+2,es      ; (возвращается в ES:BX)
; установить наш обработчик
       mov    ax,251Ch    ; AH – 25h, AL = номер прерывания
       mov    dx,offset int1Ch_handler ; DS:DX - адрес обработчика
       int    21h         ; установить обработчик прерывания 1Ch

; здесь размещается собственно программа, например вызов command.com
       mov    ah,1
       int    21h         ; ожидание нажатия на любую клавишу
; конец программы

; восстановить предыдущий обработчик прерывания 1Ch
       mov    ax,251Ch          ; AH = 25h, AL = номер прерывания
       mov    dx,word ptr old_int1Ch+2
       mov    ds,dx
       mov    dx,word ptr cs:old_int1Ch ; DS:DX - адрес обработчика
       int    21h

       ret

old_int1Ch dd     ? ; здесь хранится адрес предыдущего обработчика
start_position    dw 0  ; позиция на экране, в которую выводится текущее
                        ; время
start  endp
```

```
; обработчик для прерывания 1Ch
; выводит текущее время в позицию start_position на экране (в текстовом
; режиме)

int1Ch_handler  proc  far
        pusha           ; обработчик аппаратного прерывания
        push  es        ; должен сохранять ВСЕ регистры
        push  ds

        push  cs        ; на входе в обработчик известно только значение
        pop   ds        ; регистра CS
        mov   ah,02h        ; Функция 02h прерывания 1Ah:
        int   1Ah           ; чтение времени из RTC
        jc    exit_handler ; если часы заняты - в другой раз

; AL = час в BCD-формате
        mov   al,ch         ; Ch = час в BCD-формате
        call  bcd2asc       ; преобразовать в ASCII
        mov   byte ptr output_line[2],ah ; поместить их в
        mov   byte ptr output_line[4],al ; строку output_line

        mov   al,cl         ; CL = минута в BCD-формате
        call  bcd2asc
        mov   byte ptr output_line[10],ah
        mov   byte ptr output_line[12],al

        mov   al,dh         ; DH = секунда в BCD-формате
        call  bcd2asc
        mov   byte ptr output_line[16],ah
        mov   byte ptr output_line[18],al

        mov   cx,output_line_l  ; число байтов в строке - в CX
        push  0B800h
        pop   es                ; адрес в видеопамяти
        mov   di,word ptr start_position ; в ES:DI
        mov   si,offset output_line ; адрес строки в DS:SI
        cld
        rep movsb               ; скопировать строку
exit_handler:
        pop   ds            ; восстановить все регистры
        pop   es
        popa
        jmp   cs:old_int1Ch ; передать управление предыдущему
                            ; обработчику
```

```
; процедура bcd2asc
; преобразует старшую цифру BCD-числа в AL в ASCII-символ в AH,
; а младшую цифру - в ASCII-символ в AL
bcd2asc         proc  near
        mov   ah,al
        and   al,0Fh        ; оставить младшие 4 бита в AL
        shr   ah,4          ; сдвинуть старшие 4 бита в AH
        or    ax,3030h      ; преобразовать в ASCII-символы
        ret
bcd2asc         endp

; строка " 00h 00:00" с атрибутом 1Fh (белый на синем) после каждого симво-
ла
output_line db    ' ',1Fh,'0',1Fh,'0',1Fh,'h',1Fh
            db    ' ',1Fh,'0',1Fh,'0',1Fh,':',1Fh
            db    '0',1Fh,'0',1Fh,' ',1Fh
output_line_l     equ $-output_line

int1Ch_handler  endp

end   start
```

Задание 2: Написать программу обработки прерывания от клавиатуры. Резидентный перехватчик прерывания *int 9H*. Обнаруживает одновременное нажатие клавиш *<Ctrl+Alt+Home>* и устанавливает режим «озвучания» нажатия клавиш. При повторном нажатии той же комбинации сбрасывает режим «озвучания». После выполнения всех операций возвращает управление старому обработчику прерывания. Резидент загружается из COM-файла. При загрузке адреса в область векторов прерываний используется «прямая» запись без использования функции DOS. При передаче управления старому обработчику используется машинный код команды *jmp* и следующее за ним поле, где был размещен адрес старого обработчика прерывания.

Код программы.

```
                title           prer
code            segment
                assume          CS:code
                org             100h
start:          jmp             init
; резидентный обработчик
tsr_9:          push            AX
                in              AL,60H  ;читаем scan-код  из порта 60H
```

```
            push    BX
            push    CX
            push    DX
            push    DS
            cmp     AL,47H  ;код клавиши <Home>?
            jne     check   ;по «нет» переход на обработку
            xor     BX,BX           ; загрузка байта
            mov     DS,BX           ; состояния
            mov     AL,DS:[0417H]   ; клавиатуры
            mov     AH,AL   ; сохранение байта состояния
            and     AL,0cH  ; проверка на нажатие <Ctrl+Alt>
            cmp     AL,0cH
            jne     check   ; по «нет» переход на обработку
            xor     byte ptr CS:flag,1 ; установка/сброс «озвучания»
check:      cmp     byte ptr CS:flag,0 ; «озвучание» включено?
            je      exit ; по «нет» переход на конец
            mov     DX,800  ;
            in      AL,61H  ;
            and     AL,0feH ;       программа
n_cycl:     or      AL,2    ;
            out     61H,AL  ;
            mov     CX,150  ;       озвучания
cycl_u:     loop    cycl_u  ;
            and     AL,0fdH ;
            out     61H,AL  ;       нажатия
            mov     CX,150  ;
cycl_d:     loop    cycl_d  ;
            dec     DX      ;       клавиши
            jnz     n_cycl  ;
exit:       pop     DS
```

```
                pop         DX
                pop         CX
                pop         BX
                 pop         AX
                 db     0eaH   ; передаем управление родному обработчику
original    dw          ?     ; адрес старого обработчика
            dw          ?     ;  9-го прерывания
flag        db          0     ; флаг  режима «озвучания»
mes         db          'tsr:',0ah,0dh,24h  ; сообщение о загрузке резидента
;   инсталлятор
init:       push        ES              ; получение
            mov         AX,3509H        ; старого значения
            int         21H             ;  вектора 9-го прерывания
            mov         original,BX       ;  сохранение этого вектора
            mov         original+2,ES     ;   в «теле» обработчика
            pop         ES
            xor         AX,AX           ;  «прямая» запись
            mov         DS,AX           ;  нового адреса в
            lea         AX,tsr_9        ;  область
            cli                         ;  векторов
            mov         DS:[0024H],AX   ;  прерываний
            mov         DS:[0026H],CS   ;
            sti                         ;
             push       CS          ; выдача сообщения
             pop        DS          ; о
             lea        DX,mes      ; загрузке
             mov        AH,9        ; в память
             int        21H         ; резидента
             lea        DX,init     ;   завершение и выход с сохранением
             int        27H         ;   обработчика прерываний в памяти
```

```
code        ends
end
```

ЛАБОРАТОРНАЯ РАБОТА № 3

«Создание резидентных программ на Ассемблере для ПК»

Цель работы:

1. Изучение способов предотвращения повторной загрузки резидента в память.
2. Изучение способов выгрузки резидентов из памяти.
3. Изучение способов передачи параметров резидентам.

Краткие теоретические сведения

В предыдущих лабораторных работах мы уже познакомились с резидентными программами, однако их работа в системе не является корректной. Дело в том, что помимо использования резидентных программ (их корректная загрузка) существует еще ряд принципиально важных моментов, на которые стоит обратить внимание: предотвращение повторной загрузки одного и того же резидента, передача параметров резидентам, выгрузка резидента из памяти.

Передача параметров резидентам не требует особого внимания и использование этой особенности резидентов подробно рассмотрено в ходе самой лабораторной работы.

1. Предотвращение повторной загрузки резидента в память

Достаточно часто от резидентной программы требуется, чтобы она могла определить свое наличие в памяти для предотвращения повторной загрузки.

Существует множество вариантов решения этой задачи. Как правило, используются следующие методы:

1.1. *Сигнатуры в памяти.* При инсталляции программы в некоторое место памяти пишется специальная сигнатура, по наличию которой в дальнейшем и определяют наличие резидента. Проблема заключается в том, чтобы найти такое место, где сигнатура не мешала бы нормальной работе. В качестве кандидата на такое место можно рассматривать: неиспользуемые вектора прерываний, область данных BIOS и т.п. Однако на практике найти такое место достаточно сложно, и нет никаких гарантий, что оно не будет использоваться другими программами.

1.2. *Сигнатура в тексте резидента* непосредственно перед обработчиком прерывания. В этом случае достаточно проверить 1-2 байта, расположенных по адресу перед считанным из вектора, и наличие данного резидента будет установлено. К сожалению, данный способ не работает, если после ус-

тановки резидента прерывание было перехвачено другой программой, даже, если сама обработка прерывания выполняется.

1.3. *"Магические" числа* в дополнительных обработчиках прерываний. Перед тем как оставить резидент в памяти программа инсталляции обращается к некоторому прерыванию с определенным "магическим" числом. Если существует резидентная копия обработчика, то она перехватывает это прерывание и отвечает другим заранее определенным "магическим" числом. Получив ответ, программа инсталляции завершает работу без сохранения резидента в памяти. Если получен любой другой ответ, программа инсталлируется. Поскольку обработчик дополнительного прерывания находится в одном модуле с обработчиком основного прерывания, появляется возможность изменения параметров резидентной программы. Метод не работает при полном перехвате дополнительного прерывания другой программой.

1.4. Использование *мультиплексного прерывания DOS* (int 2FH). Данное прерывание используется системой для связи с собственными резидентными процессами. При его вызове в регистре AH задается номер функции (номер резидентного процесса), а в регистре AL- номер подфункции. Функции с номерами от 00H до BFH зарезервированы для использования системой, а функции с C0H по FFH могут использоваться прикладными программами. При инсталляции обработчику присваивается оригинальный мультиплексный номер, по наличию которого в системе и определяется присутствие данного резидента. Естественно, гарантии, что другой резидент не использует тот же номер, нет. Так же как и в предыдущем случае, резидент должен включать дополнительный обработчик прерывания (int 2FH), который, помимо предотвращения повторной загрузки, может использоваться для связи с резидентным процессом, например, для передачи ему параметров. Далее в пункте 2 и приводится пример использования данного метода.

1.5. Запись *сигнатуры в PSP резидента*. При попытке повторной загрузки инсталлятор проверяет PSP всех загруженных программ. Для этого используется функция DOS 52H, которая позволяет получать сегментные адреса управляющих блоков памяти (MCB). Метод дает полную гарантию, но относительно сложен.

2. Удаление резидента из памяти

Все рассмотренные выше резидентные программы не обеспечивают возможности выгрузки из памяти, и единственная возможность освободить занимаемую ими память - перезагрузка машины. На практике же большинство серьезных резидентных программ имеют встроенные средства выгрузки.

Выгрузка резидентного обработчика прерывания из памяти выполняется в два этапа: сначала необходимо восстановить вектор прерывания, а затем освободить память, занимаемую резидентом. Причем, прежде чем восстанавливать вектор необходимо убедиться, что никакая другая программа не захватила данное прерывание и не вызывает наш обработчик в качестве "старого". Удаление обработчика в таком случае, скорее всего, приведет к "зависанию" машины. Достаточно надежно можно удалить лишь последний загруженный резидент.

Инициатором процесса выгрузки может служить запуск специальной выгружающей программы. Удобнее всего в качестве выгружающей программы использовать саму резидентную программу, которая при повторном запуске должна получить доступ к первой копии и выгрузить ее. Так же как при передаче параметров доступ к первой копии можно осуществить через перехват дополнительных прерываний (int 2fH или любого другого свободного прерывания).

Ход работы

В ходе выполнения работы предлагается написать законченную программу на Ассемблере, откомпилировать ее и проверить правильность работы. Код программы с заданием представлен ниже. Программа должна быть скомпилирована в виде COM файла.

Задание :

Написать программу самовыгружающегося резидентного обработчика прерывания *int 1cH*.

При запуске программы без параметра происходит проверка повторной загрузки обработчика и инсталляции его, если он не был инсталлирован ранее. Запуск с параметром *«off»* приводит к выгрузке обработчика.

Программа использует параметр, задаваемый в командной строке при ее запуске. Доступ к параметрам командной строки осуществляется через область PSP, по адресу 80H в которой записывается длина строки параметров, а начиная с адреса 81H располагаются сами параметры.

В качестве дополнительно перехватываемого прерывания используется прерывание *int 2fH*. Обработчик этого прерывания использует номер *0c8H* для идентификации резидентного процесса и выполняет две подфункции: 0 - проверка повторной загрузки и 1 - выгрузка резидента.

Обработчик основного прерывания *int 1cH* каждые десять тиков таймера выводит на экран символы *ASCII*.

```
text        segment  'code' public byte
            assume   CS:text,DS:text
            org      256
main        proc
            jmp      init
old_2fh  dd      0          ; старое значение вектора прерывания int 2fH
old_1ch  dd      0          ; старое значение вектора прерывания int 1cH
; обработчик int 2fH
new_2fh   proc
```

```
        cmp     AH,0c8H
        jne     out_2fh
        cmp     AL,00H          ; если резидент загружен
        je      inst            ; то выдать в AL специальный код
        cmp     AL,01h          ; если выгрузка
        je      uninstalled     ; то перейти к выгрузке
    jmp short out_2fh ; иначе передать управление старому обработчику
inst:   mov     AL,0ffH
        iret
out_2fh: jmp    CS:old_2fh
uninstalled:
        push    DS
        push    ES
        push    DX
        mov     AX,251cH                ;   восстановить
        lds     DX,CS:old_1ch           ;   старые
        int     21H                     ;   адреса
        mov     AX,252fH                ;   обработчиков
        lds     DX,CS:old_2fh           ;   прерываний
        int     21H                     ;   в области  векторов
        mov     ES,CS:2cH       ;   освободить
        mov     AH,49H          ;   область, занимаемую
        int     21H             ;   окружением DOS (адрес из PSP)
        push    CS                      ;   освободить
        pop     ES                      ;   область,
        mov     AH,49H                  ;   занимаемую
        int     21H                     ;   программой
        pop     DX
        pop     ES
        pop     DS
```

```
            iret
new_2fh     endp
; обработчик int 1cH
new_1ch     proc  ; основной обработчик: используя таймер пишет
            cli   ; в видеопамять коды символов ASCII
            inc   CS:tik
            cmp   CS:tik,10
            jl    a1
            push  ES
            push  AX
            mov   AX,0b800H
            mov   ES,AX
            xor   AL,AL
            mov   CS:tik,al
            inc   CS:nch
            mov   AL,CS:nch
            mov   ES:[0],AL
            mov   AL,1eH
            mov   ES:[1],AL
            pop   AX
            pop   ES
 a1:        sti
            push  AX
            pop   AX
            iret
tik         db    ?
nch         db    0
new_1ch     endp
main        endp
end_rez=$           ; адрес конца резидентной части
```

```
tail    db    'off' ; значение параметра, означающего выгрузку резидента
flag    db     0    ; флаг выгрузки
init    proc       ; инсталлятор
          mov    CL,ES:80H     ; длина области параметров из PSP
          cmp    CL,0          ; если нет параметров
          je     ahead         ; то проверяем загружен ли резидент
          xor    CH,CH         ; иначе формируем счетчик CX
          mov    DI,81H        ; заносим в DI смещение строки параметров
          mov    SI,offset tail ; в SI смещение "правильного" параметра
          mov    AL,' '          ; организует
repe      scasb                  ; пропуск всех
          dec    DI              ; пробелов и
          mov    CX,3            ; проверяем значение
repe      cmpsb                  ; параметра
          jne  ahead ; если не "off", переходим к проверке повторной загруз-
ки
          inc    flag            ; иначе - установим флаг выгрузки
ahead:    mov    AH,0c8H         ; обращаемся к функции
          mov    AL,0            ; определения повторной загрузки
          int    2fH             ; прерывания int 2fH
          cmp    AL,0ffH       ;  если ответ=ожидаемому
          je  installed ; то переходим к сообщению о повторной загрузке
          mov    AX,352fH              ; иначе - сохраняем старые
          int    21H                   ; значения векторов преры-
          mov    word ptr old_2fh,BX   ; ваний int 2fH и int 1cH и
          mov    word ptr old_2fh+2,ES ; записываем туда адреса
          mov    AX,351cH              ; своих обработчиков
          int    21H                   ;
          mov    word ptr old_1ch,BX   ;
          mov    word ptr old_1ch+2,ES ;
```

```
        mov     AX,252fH                ;
        mov     DX,offset new_2fh       ;
        int     21H                     ;
        mov     AX,251cH                ;
        mov     DX,offset new_1ch       ;
        int     21H                     ;
        mov     AH,09H                  ; печатаем сообщение о
        mov     DX,offset mes           ; загрузке резидента
        int     21H                     ;
        mov     AX,3100H                ; выходим, сохранив в
        mov     DX,(end_rez-main+100H+0fH)/16   ; памяти резидентную
        int     21H                     ; часть программы
installed:
        cmp     flag,1      ;   если необходимо выгрузить программу
        je      unins       ;   то переходим на выгрузку
        mov     AH,09H      ;   иначе - выдаем сообщение о том
        mov     DX,offset mes1  ; что программа уже
        int     21H             ; загружена в память
        mov     AX,4c01H        ; и выходим без
        int     21H             ; сохранения в памяти
unins:  mov     AX,0c801H   ; обращаемся функции удаления
        int     2fH         ; прерывания  int 2fH
        mov     AH,09H          ; выдаем сообщение
        mov     DX,offset mes2  ; о выгрузке программы
        int     21H             ; из памяти
        mov     AX,4c00H        ; и выходим без
        int     21H             ; сохранения в памяти
mes     db      'Программа загружена',10,13,'$'
mes1    db      'Программа уже загружена',10,13,'$'
mes2    db      'Программа выгружена из памяти',10,13,'$'
```

```
init    endp
text    ends
        end     main
```

9.3 ПРАКТИЧЕСКИЕ ЗАНЯТИЯ

Тема 1. Изучение основ работы с контроллером Mega-128

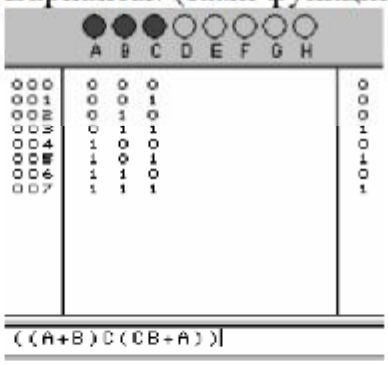
Рассматриваемые вопросы:

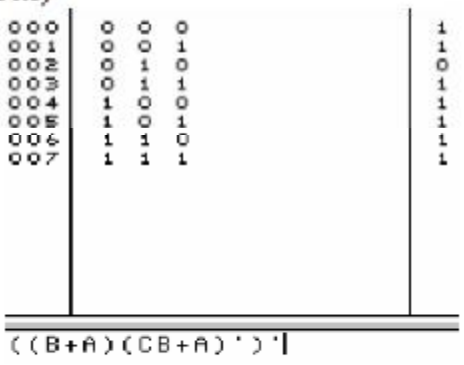
1. Изучение команд ввода/вывода и логических операций.
2. Изучение основ написания программ на ассемблере, реализующих логические функции.
3. Моделирование работы программы в отладчике PLC.
4. Компилирование программы в машинный код.
5. Пересылка программ в контроллер при помощи программатора.

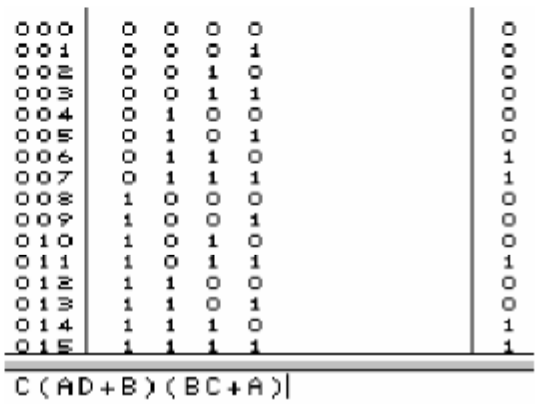
Задачи:

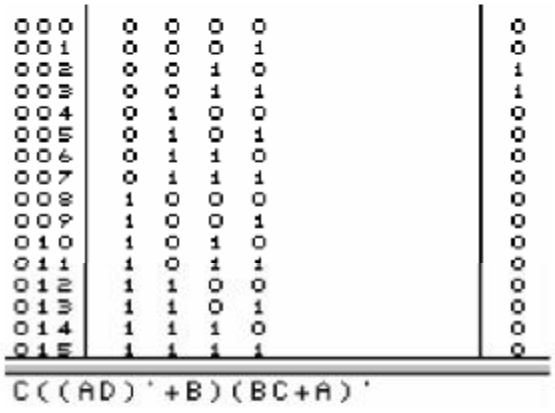
Задание1: Решить задачу для варианта:

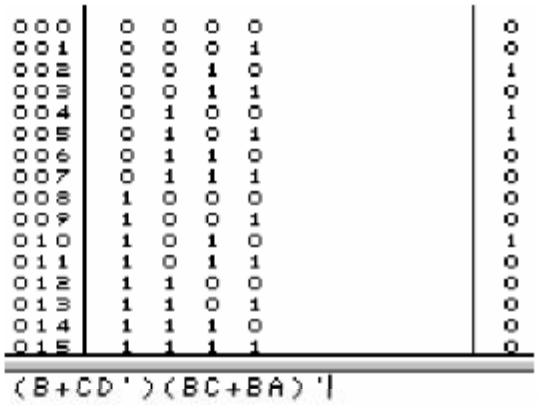
Варианты: (сами функции студентам не выдаются)

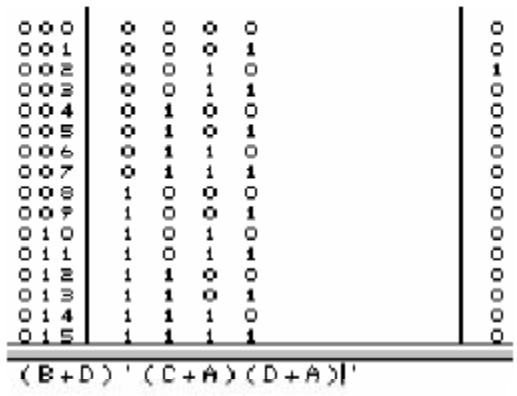


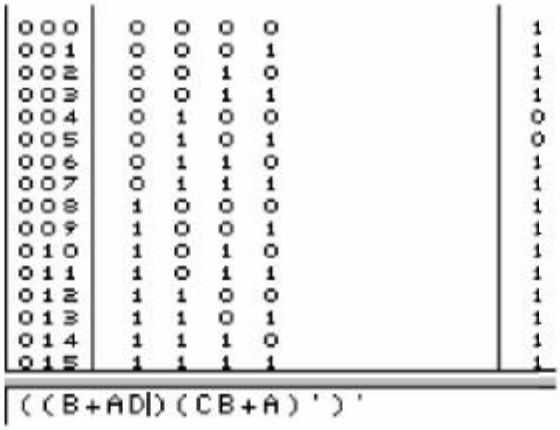


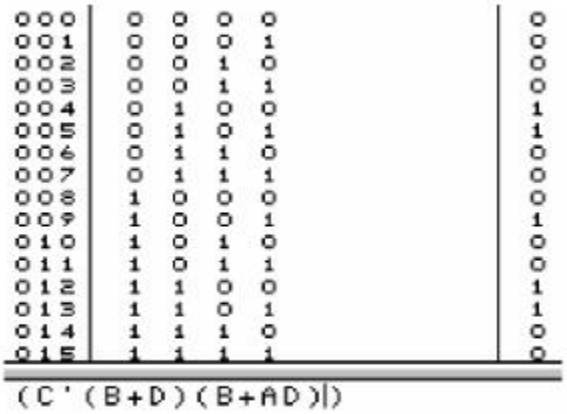












Задание2: Написать программу эмулирующую работу устройства контролирующего уровень в бункере с углем. Уровень не должен повышаться выше датчика X00 или опускаться ниже датчика X01. Считаем, что если датчик в угле, то на нем 1. Включение конвейера, подающего уголь в бункер, через ключ Y00, если 1, то конвейер включен.

Для этого необходимо зарезервировать переменную R00, которая будет запоминать предыдущее воздействие.

Логическая функция, описывающая данное устройство:

Таблица истинности

R00	X00	X01	Y00	Описание
1	1	1	0	Оба датчика в угле – выключить подачу
1	1	0	1	Нижний в угле и конвейер был включен - включить
1	0	1	0	Нижний чистый, верхний в угле – такого быть не может
1	0	0	1	Оба датчика чистые – включить подачу
0	1	1	0	Оба датчика в угле – выключить подачу
0	1	0	0	Нижний в угле, а конвейер был выключен - выключить
0	0	1	0	Нижний чистый, верхний в угле – такого быть не может
0	0	0	1	Оба датчика чистые – включить подачу

Тема 2. Изучение работы контроллера Mega-128 при реализации основных алгоритмов

Рассматриваемые вопросы:

1. Изучить основные команды сравнения.
2. Изучить работу таймера.
3. Создать завершенные команды работы контроллера по поддержанию уровня.
4. Создать завершенные команды работы контроллера по формированию задержек.
5. Создать завершенные команды работы контроллера по заданной установке выхода в требуемое значение.
6. Создать завершенные команды работы контроллера по формирования аварийной сигнализации по заданным условиям.

Задачи:

Задание1. Написать программу, эмулирующую работу устройства контролирующего уровень угля в котле. Уровень не должен повышаться выше или опускаться ниже заданных значений. И температура двигателя не должна быть выше 50 ^{0}C.

Задание 2: Написать программу, показывающую работу таймера. При подаче на вход X00 логической единицы на выходе Y00 через 4 сек. появляется единица.

Задание 3: Написать программу, имитирующую работу таймера. При подаче на вход X00 логической единицы на выходе Y00 через 4 сек. появляется единица. При подаче на вход X00 логического нуля на выходе Y00 появляется ноль, спустя 4 сек. (запаздывание, петля гистерезиса)

Тема 3. Изучение работы контроллера Mega-128 при реализации завершенных алгоритмов управления

Рассматриваемые вопросы:

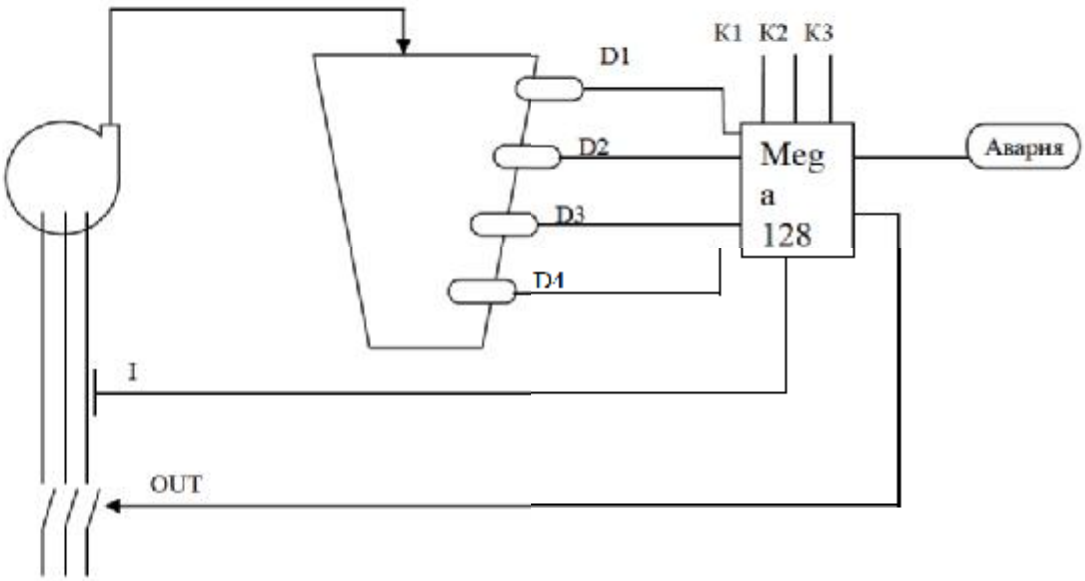
1. Создание программ поддержания уровня в баке в двух заданных диапазонах с учетом задания работы контроллера в автоматическом и в ручном режиме.
2. Создание программ поддержания уровня в резервуаре в двух заданных диапазонах, с учетом задержки на включение и выключение в автоматическом и ручном режиме.

Задачи:

Задание 1:

Необходимо поддерживать уровень в баке насосом в диапазоне D2-D3 с помощью автоматики и в ручном режиме. В автоматическом режиме насос включается, если уровень меньше D3 и отключается, если больше D2. Насос отключается если ток двигателя I больше 50. Предусмотреть аварийную сигнализацию, которая включается, если уровень меньше D4 или больше D1 или если произошел перегрев насоса I.

К1 – автоматика/ручное управление
К2 – вкл/откл насос при ручном режиме



Задание 2:

Необходимо поддерживать уровень в баке насосом в двух диапазонах D2-D3 или D3-D4. Вид диапазона определяется с помощью К3. Необходимо предусмотреть автоматическое и ручное управление.

К1 – автоматика/ручное управление
К2 – вкл/откл насос при ручном режиме
К3 – 1/2 диапазоны при автоматическом управлении

Задание 3:

Необходимо поддерживать уровень в баке насосом в диапазоне D2-D3 с помощью автоматики и в ручном режиме. В автоматическом режиме насос включается через 4 сек, если уровень меньше D3 и отключается через 4 сек, если больше D2.

Варианты:

Лаб	Вар	Д1	Д2	Д3	Д4	К1	К2	К3	I	OUT	Авар
3	1	X00	X01	X02	X03	X04	X05		A00	Y00	Y01
	2	X01	X02	X03	X04	X05	X06		A01	Y01	Y02
	3	X02	X03	X04	X05	X06	X07		A02	Y02	Y03
	4	X03	X04	X05	X06	X07	X08		A03	Y03	Y04
	5	X04	X05	X06	X07	X08	X09		A00	Y04	Y05
	6	X05	X06	X07	X08	X09	X10		A01	Y05	Y06

	7	X06	X07	X08	X09	X10	X11		A02	Y06	Y07
	8	X07	X08	X09	X10	X11	X12		A03	Y07	Y08
	9	X08	X09	X10	X11	X12	X13		A00	Y08	Y09
	10	X09	X10	X11	X12	X13	X00		A01	Y09	Y10
4	1	X00	X01	X02	X03	X04	X05	X06		Y00	
	2	X01	X02	X03	X04	X05	X06	X07		Y01	
	3	X02	X03	X04	X05	X06	X07	X08		Y02	
	4	X03	X04	X05	X06	X07	X08	X09		Y03	
	5	X04	X05	X06	X07	X08	X09	X10		Y04	
	6	X05	X06	X07	X08	X09	X10	X11		Y05	
	7	X06	X07	X08	X09	X10	X11	X12		Y06	
	8	X07	X08	X09	X10	X11	X12	X13		Y07	
	9	X08	X09	X10	X11	X12	X13	X00		Y08	
	10	X09	X10	X11	X12	X13	X00	X01		Y09	
5	1	X00	X01	X02	X03	X04	X05			Y00	
	2	X01	X02	X03	X04	X05	X06			Y01	
	3	X02	X03	X04	X05	X06	X07			Y02	
	4	X03	X04	X05	X06	X07	X08			Y03	
	5	X04	X05	X06	X07	X08	X09			Y04	
	6	X05	X06	X07	X08	X09	X10			Y05	
	7	X06	X07	X08	X09	X10	X11			Y06	
	8	X07	X08	X09	X10	X11	X12			Y07	
	9	X08	X09	X10	X11	X12	X13			Y08	
	10	X09	X10	X11	X12	X13	X00			Y09	

9.4 КОНТРОЛЬНЫЕ РАБОТЫ

Как отмечено в п.7.4 проведение контрольных работ предусматривается в виде тестов. В зависимости от времени проведения выбираются соответствующие темы. Комплект заданий к контрольным работам здесь не приводится. В разделе 10.2 представлен комплексный тест, в котором вопросы разбиты по темам лекционного курса (см. п.5). Текущие контрольные работы выбираются из данного комплексного теста.

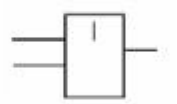
10. ФОНД ТЕСТОВЫХ И КОНТРОЛЬНЫХ ЗАДАНИЙ ДЛЯ ОЦЕНКИ КАЧЕСТВА ЗНАНИЙ ПО ДИСЦИПЛИНЕ

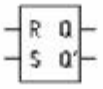
10.1. ВХОДЯЩИЙ КОНТРОЛЬ ЗНАНИЙ

Водящий контроль знаний представляет тест состоящий из 12 вопросов (по 4 произвольных вопроса из 3 частей). Вопросы каждой части представлены ниже.

Часть 1

1. Одним из способов описания цифрового устройства является…	а) Логическое выражение; б) Логическое тождество; в) Логическая функция; г) Правильного ответа нет;
2. Схема, представляющая собой управляемый переключатель, который подключает к выходу один из входов данных, называется:	а) Шифратор; б) Мультиплексор; в) Дешифратор; г) Сумматор;
3.В двоичной системе исчисления за единицу объема принят…	а) Гбайт; б) Бит; в) Байт; г) Мбайт;
4. Информация после какого-то свершившегося факта или действия называется….	а) Апостериорная неопределенность; б) Синтаксическая адекватность; в) Априорная неопределенность; г) Энтропия;
5. Под процессом автоматизированной обработки информации понимается…	а) Получение; б) Хранение; в) Преобразование; г) Передача;
6. Дисциплина, занимающаяся изучением работы устройств, работающих по принципу включено-выключено называется…	а) Геометрией; б) Булевой алгеброй; в) Математикой; г) Правильных ответов нет;
7. К комбинационным устройствам относятся:	а) Транзистор; б) Шифратор; в) Резистор; г) Сумматор;

8. Соответствие содержания и смысла называется…	а) Синтаксическая адекватность; б) Прагматическая адекватность; в) Семантическая адекватность; г) Правильных ответов нет;
9. Объем работ выполняемых в единицу времени называется…	а) Быстродействие; б) Достоверность; в) Точность; г) Производительность;
10. Простейшее устройство ВМ, выполняющая одну операцию над входными переменными, называется…	а) Узел; б) Элемент; в) Блок; г) Правильных ответов нет;
11. Для случая, когда все состояния системы равновероятны, энтропия определяется по формуле:	а) $H(\alpha) = \log N$ б) $H(\alpha) = -\sum Pi \log N$ в) $I\beta(\alpha)=H(\alpha)$ г) $N=m^n$
12. Какой из триггеров делит частоту входного сигнала пополам?	а) RS-триггер; б) T-триггер; в) D-триггер; г) JK-триггер;
13. Схемное представление соответствует операции: 	а) Конъюнкция; б) Дизъюнкция; в) Инверсия; г) Эквивалентность;
14. Показателем качества является…	а) Последовательность; б) Точность; в) Адекватность; г) Достаточность;
15. Управляемый переключатель, имеющий один выход и несколько входов называется…	а) Диод; б) Дешифратор; в) Резистор; г) Мультиплексор;

16. Какой комбинационной схемы представлено схемное представление?  	а) D-триггер; б) Т-триггер; в) RS-триггер; г) JK-триггер;
17. По назначению ВМ бывают:	а) Специализированные; б) Функциональные; в) Структурные; г) Универсальные;
18. Таблица истинности какого элемента представлена на рисунке? Входы: A, B; Выходы: Y 0 0 0 0 1 0 1 0 0 1 1 1	а) Конъюнкция; б) Инверсия; в) Эквивалентность; г) Дизъюнкция;
19. Как называется соответствие цели и содержания?	а) Синтаксическая адекватность; б) Прагматическая адекватность; в) Семантическая адекватность; г) Правильных ответов нет;
20. Переменные, рассматриваемые в алгебре логики, принимают значение:	а) ‘2’ б) ‘0’ в) ‘5’ г) ‘1’
21. . Какие из указанных входов имеются у триггера?	а) Информационные б) Управляющие в) Разрешения г) Установочные

22. Абстрактная модель, описывающая функциональные возможности ВМ, и предоставляемые ей услуги называется…	а) Структурная организация ВМ; б) Техническая организация ВМ; в) Функциональная организация ВМ; г) Правильных ответов нет;
23.Схемное представление какой функции представлено на рисунке? 	а) И; б) ИЛИ; в) И-НЕ; г) ИЛИ-НЕ;
24. Как называется комплекс программно-технических средств, предназначенных для обработки информации в процессе решения информационных и вычислительных задач?	а) Персональный компьютер; б) Микропроцессор; в) Комбинационная схема; г) Вычислительная машина;
25. Устройство, предназначенное для синхронизации всей системы, называется…	а) Процессор; б) Внешнее запоминающее устройство; в) Таймер; г) Тактовый генератор;
26. Как называется выделение в структуре ВМ, достаточно автономных, функционально и конструктивно независимых устройств?	а) Модульность; б) Точность; в) Доступность; г) Адекватность;
27. Шифратор - комбинационная схема, которая имеет…	а) 2^n выходов; б) n входов; в) 2 выхода; г) 2^n входов;
28. Что выполняет устройство управления УУ?	а) Выборку данных; б) Выборку команд; в) Дешифрирование команд г) Правильных ответов нет;

29. Адекватность – это…	а) Соответствие содержания и смысла; б) Соответствие содержания и цели; в) соответствие содержания информации образу исходного объекта; г) Соответствие формально-структурных показателей;
30. Связанные между собой по принципу работы АЛУ и УУ объединены единым устройством -…	а) Счетчик; б) Дешифратор; в) Вычислительная машина; г) Микропроцессор;
31. Фамилией, какого ученого названо данное правило? $y = \overline{x_1 \cdot x_2} = \overline{x_1} + \overline{x_2}$ $y = \overline{x_1 + x_2} = \overline{x_1} \cdot \overline{x_2}$	а) Глушкова б) Фон Неймана в) Де Моргана г) Карно
32. Структурой называется…	а) Элементарная ячейка памяти, представленная для внешнего хранения промежуточных результатов. б) Совокупность элементов и их связей. в) Контролер прямого доступа к памяти. г) Согласованность параметров элементарных сигналов.
33. В какой структуре множество процессорных элементов объединены с помощью коммутатора?	а) VLIW архитектура. б) RISC архитектура. в) CISC архитектура. г) SIMD архитектура.
34. К какому поколению относятся ВМ с жесткой структурой устройства управления?	а) 1-му. б) 2-му в) 3-му. г) 4-му.
35.Какая из предложенных подсистем осуществляет инициализацию, тестирование, отладку и контроль за эффективностью работы всей системы?	а) Подсистема ввода-вывода. б) Подсистема повышения производительности. в) Подсистема памяти. г) Подсистема управления и обслуживания.

36.Структурной организацией ВМ называется…	а) Абстрактная модель, описывающая функциональные возможности ВМ и представленные ей услуги. б) Физическая модель, устанавливающая состав, порядок и принципы взаимодействия основных функциональных частей ВМ. в) Физическая модель, описывающая возможности ВМ и представляющая ей услуги. г) Правильных ответов нет.
37. Основой ВМ в 70-е XX века являлись …	а) Полупроводниковые приборы. б) Интегральные схемы. в) Микропроцессоры. г) Вакуумные лампы.
38. Какие ЭВМ разрабатывались на основе микрокомпонентов?	а) Супер-ЭВМ. б) Большие ЭВМ. в) Мини ЭВМ. г) Микро ЭВМ.
39. Важнейшим свойством информации является…	а) Актуальность; б) Простота; в) Адекватность; г) Правильного ответа нет;
40.Предписание, определяющее содержание действий, называется…	а) Адрес; б) Архитектура; в) Программа; г) Команда;
41. На каком этапе VLIW архитектуры команды группируются в пакеты, содержимое, которых соответствует структуре процессора?	а) 3; б) 1; в) 4; г) 2;
42. Функция, представляющая собой зависимость выхода системы от входа $y = f(x)$ называется…	а) Логическим элементом; б) Логическим тождеством; в) Логическим выражением; г) логической функцией;

43. Как называется устройство, которое подключает один вход к нескольким выходам?	а) Счетчик; б) Шифратор; в) Регистр; г) Правильного ответа нет;
44.К преимуществам SIMD архитектуры относятся:	а) Экономичность; б) Выполнение одной или нескольких команд одновременно; в) Большое число внутренних регистров; г) Высокая производительность;
45. В каких системах человек присутствует только на этапе проектирования?	а) Автоматизированные системы; б) Автоматические системы; в) Функциональные системы; г) Подсистема управления;
46. Схемное представление какой функции представлено на рисунке? 	а) Эквивалентность; б) Конъюнкция; в) Дизъюнкция; г) Инверсия;
47. Таблица истинности какого элемента представлена на рисунке? Входы | Выход A | B | Y 0 | 0 | 1 0 | 1 | 1 1 | 0 | 1 1 | 1 | 0	а) ИЛИ; б) И; в) И-НЕ; г) ИЛИ-НЕ;
48. Арифметическое логическое устройство содержит:	а) Триггеры; б) Мультиплексоры; в) Шифраторы; г) Логические преобразователи;

49. В каком типе архитектуры ВМ оценивается быстрое действие, производительность, надежность.	а) Программный; б) Функциональный; в) Технический; г) Правильных ответов нет;
50. Какая из предложенных характеристик позволяет оценить правильность восприятия информации?	а) Надежность; б) Достоверность; в) Точность; г) Производительность;

Часть 2

1. Элементарная ячейка памяти, предназначенная для временного хранения промежуточных результатов.	a) регистр b) выборка c) чипсет d) свопинг
2. Предназначен для выполнения полного функционального набора арифметических и логических операций. Как правило в него входят: РОН, АЛУ, аккумулятор, буферный регистр и т.д.	a) интерфейсный блок b) блок управления c) операционный блок d) блок заполнения
3. В большинстве реальных моделях МП один из регистров выделяется в качестве главного и называется	a) БФУС b) РОН c) аккумулятор d) регистр команд
4. В состав интерфейсного блока входит	a) счётчик команд b) регистр сдвигатель c) буферный регистр данных d) всё выше перечисленное
5. Регистр предназначенный для хранения результатов	a) буферный регистр b) регистр признаков c) регистр команд d) регистр сдвигатель
6. Предназначен для адресации внутри стека. При выполнении одной операции увеличивается на 1-цу тем самым обеспечивает выполнение следующей команды.	a) указатель стека b) счётчик команд c) БФУС d) регистр команд
7. При каком виде адресации сам код команды подразумевает работу с неким адресом и этот код не указывается	a) прямая адресация b) неявная адресация c) косвенная адресация d) индексная адресация
8. Предназначен для организации взаимодействия МП с внешними устройствами, с устройством ввода-вывода, а также для организации обмена между ОП и операционным блоком	a) интерфейсный блок b) блок управления c) операционный блок d) блок заполнения
9. Применяется для временного хранения данных, организация прерывателя, вызова процедур и т.д.	a) сектор b) стек c) очередь d) блок

10. Обеспечивает внутренний обмен информации между регистрами МП, также обеспечивает обмен данных между устройствами ввода-вывода и МП	a) команды пересылки данных b) команды передачи управления c) команды работы со стеком d) команды поразрядного двоичного сдвига
11. При каком виде адресации адрес содержится в коде команды и обычно следует за кодом операции	a) непосредственная адресация b) прямая адресация c) косвенная адресация d) неявная адресация
12. Одна из фаз при выполнении любой команды, которая обеспечивает считывание команд из памяти и пересылку её в МП	a) выборка b) декодирование c) выполнение d) реализация
13. Какой вид адресации позволяет задавать фиксированные значения непосредственно в адресной части	a) косвенная адресация b) неявная адресация c) непосредственная адресация d) индексная адресация
14. При каком виде адресации команд адрес команд содержится в коде операции. При считывании очередной команды в СК автоматически загружается адрес следующей команды	a) относительная адресация b) абсолютная адресация c) косвенная адресация d) прямая адресация
15. При увеличении ёмкости памяти её быстродействие	a) увеличивается b) уменьшается c) не изменяется d) безразлично
16. Самая быстродействующая память	a) регистровая память b) сверхоперативная память c) управляемая память d) буферная
17. Какой вид адресации используется если необходимо считать список из ячеек памяти расположенный не подряд а с некоторым шагом	a) прямая адресация b) неявная адресация c) косвенная адресация d) индексная адресация
18. Тип памяти реализованной с помощью некого запоминающего устройства размещённого между основной памятью и процессором, предназначенная для сокращения обращений МП к основной памяти	a) управляемая память b) буферная c) регистровая память d) сверхоперативная память
19. Базовый тип оперативной памяти которая содержит ячейки памяти работающие по принципу конденсатора, наличие или отсутствие заряда	a) SRAM b) DRAM c) SDAM d) FRAM
20. Базовый тип оперативной памяти признаком хранения информации которой является типа открыто, закрыто. Реализуется на основе транзисторных схем	a) FRAM b) SRAM c) DRAM d) SDAM
21. Тип распределения памяти при котором виртуальное адресное пространство разбивается на сегменты, величина которых определяется программистом либо системой	a) страничное распределения b) сегментное распределения c) странично-сегментное распределения d) свопинг

22. Тип распределения памяти при котором некоторые задачи находятся в режиме ожидания, целиком отгружаются на жёсткий диск и в случаи необходимости могут быть снова загружены памятью	a) странично-сегментное распределения b) сегментное распределения c) свопинг d) страничное распределения
23. Ресурс обладающий гораздо большим объёмом чем ОП, но для пользователя представленная как единое целое	a) регистровая память b) виртуальная память c) буферная d) управляемая память
24. При каком виде адресации адрес указываемый в команде является указателем ячейки содержащий исполнительный адрес, фактически указывается адрес адреса	a) относительная адресация b) абсолютная адресация c) косвенная адресация d) прямая адресация
25. Принцип обмена информацией между периферийными устройствами и МП. Предполагает что одно из устройств является ведущим, а второе ведомым	a) принцип подчинения b) принцип квитирования c) принцип унификации характеристик d) принцип замены
26. Программно управляемая передача которая применяется при взаимодействии с быстродействующими устройствами для обмена с которыми не требуется дополнительный сигнал синхронизации	a) синхронная передача b) прямая передача c) асинхронная передача d) непрямая передача
27. Способ обмена данными между периферийными устройствами и вычислительным ядром системы при котором для операции ввода вывода используются специальные сигналы	a) Программно управляемая передача b) Передача инфор. с прерыванием c) Передача инфор. в режиме прямого доступа к памяти
28. Центральное устройство предназначенное для управления работой и выполнения арифметических, логических и др. операций	a) микропроцессор b) оперативная память c) системная шина d) DART
29. Обеспечивает сопряжение и связь всех устройств	a) системная шина b) прямая адресация c) RATS d) кэш-память
30. Предназначена для хранения и оперативного обмена информацией. Содержит ОЗУ и ПЗУ	a) оперативная память b) внешнее запоминающее устройство c) основная память d) кэш-память
31. Жёсткий диск, флэшка, CD, дискета и т.д.	a) оперативная память b) кэш-память c) основная память d) внешнее запоминающее устройство
32. Тип системной платы определяет	a) основная память b) микропроцессор c) оперативная память d) периферийные устройства
33. Шина расширения, имеет 8-ми разрядную шину данных, 20-ти разрядную шину адреса, может работать на частоте 4,77	a) ISA b) PC/XT c) PC/AT

МГц	d) AGP
34. Шина расширения, имеет 32-х разрядную шину данных и шину адреса, создана в 1989г. Работает на частоте 8-10 МГц. Имеет возможность подключать до 16 внешних устройств	a) PC/XT b) IESA c) VLB d) SCSJ
35. Локальная шина, имеет 32-х разрядную шину данных и шину адреса ,допускает подключение 10 внешних устройств. Тактовая частота 33 МГц. Может работать в 64 битных системах	a) PCI b) VLB c) PC/AT d) AGP
36. Тип памяти построенная на основе полупроводников, своеобразных конденсаторах	a) статическая память b) постоянная память c) динамическая память d) внешняя память
37. Используется для работы с жёстким диском	a) SATA b) RS-232 c) ATA d) IEEE 1284
38. Универсальная периферийная шина с возможностью подключения 126 устройств. Передача информации 12 Мбит/с	a) IEEE 1394 b) PCMCIA c) VLB d) USB
39. Модуль оперативной памяти, работает на частоте 600-800 МГц, имеет пропускную способность 1,6 Гбайт/с, время обращения 5 нс	a) DIM b) RIM c) SIM d) DIP
40. Тип оперативной памяти, пропускная способность при шине 100 МГц 1,6 Гбайт/с, выпускается в конструкции DIM	a) FPMDRAM b) RAMEDO c) DRAM d) DDR|SDRAM
41. Тип оперативной памяти, может работать на частоте 800 МГц, позволяет использовать технологию двухканального обмена	a) RAMEDO b) DRAM c) DR DRAM d) FPMDRAM
42. На какой фазе проводится процедура тестового контроля	a) на всех b) выборочно c) не на одной d) возможны все варианты
43. Если причиной неисправности являются ошибки, допущенные при проектировании, некорректный монтаж, нарушение режимов работы то неисправность называется	a) физическая неисправность b) субъективная неисправность c) объективная неисправность d) семеричная неисправность
44. Процесс обнаружения ошибки и определение источника его появления	a) диагностика неисправности b) отладка c) выкладка d) проектирование
45. В процессе отладки неисправности	a) полностью

устраняются	b) не устраняются вовсе c) возможен их пропуск d) устраняется первая и процесс заканчивается
46. Что должно: 1) управлять исполнением программы, останавливать и т.д. 2) собирать информацию о ходе выполнения программы 3) обеспечивать обмен информации между программой и ВМ 4) моделировать работу отдельных элементов системы	a) средства отладки программ b) диагностика неисправности c) тестовый сегмент d) средства фиксации
47. Программное обеспечение МП разрабатывается на	a) иностранных языках b) алгоритмических языках c) интернациональных языках d) возможны все варианты ответов
48. Исходную программу на ассемблере составляют с помощью	a) Microsoft Word b) блокнота c) Bred d) любого текстового редактора
49. Содержит исходную программу, программу в машинных кодах, а также обнаруженные ошибки с указанием страницы где они обнаружены	a) исполнительный файл b) листинг c) объектная модель d) транслятор
50. Сколько существует основных приёмов комплексной отладки микропроцессорных систем	a) 2 b) 7 c) 1 d) 5

Часть 3

1. Микроконтроллерная система, предназначенная для решения конкретных технических задач.	1. Персональный компьютер 2. Микропроцессор 3. Простейшая вычислительная система 4. Объект управления
2. Некий управляемый совокупностью используемых устройств процесс.	1. Объект управления 2. Вычислительная система 3. Микропроцессор 4. Устройство сопряжения с объектом
3. Это устройство обеспечивает согласование сигналов, как управления, так и информации между объектами управления и микропроцессорными системами.	1. Объект управления 2. Пульт управления 3. Микропроцессор 4. Устройство сопряжения с объектом
4. Это устройство представляет оператору формировать параметры процесса управления и при необходимости вносить коррективы.	1. Объект управления 2. Пульт управления 3. Цифровой регулятор 4. Устройство сопряжения с объектом
5. Это устройство получает информацию об объекте управления от датчиков.	1. Объект управления 2. Пульт управления 3. Цифровой регулятор 4. Устройство сопряжения с объектом
6. Нормальный процесс работы системы, который проходит после её создания.	1. Частота 2. Функционирование

	3. Быстродействие 4. Производительность
7. Система функционирующая без участия человека.	1. Автоматическая 2. Механическая 3. Автоматизированная 4. Независимая
8. Система функционирующая под воздействием человека.	1. Автоматическая 2. Механическая 3. Автоматизированная 4. Независимая
9. Как система управления с микроконтроллером обрабатывает информацию во времени.	1. Непрерывно 2. Дискретно 3. По закону sin 4. По закону cos
10. Структура трехуровневого аппаратно-программного управляющего комплекса (от низшего к высшему).	1. Микроконтроллер – пульт управления - цифровой регулятор 2. Пульт управления – цифровой регулятор – микроконтроллер 3. Цифровой регулятор – микроконтроллер - пульт управления 4. Микроконтроллер – цифровой регулятор – пульт управления
11. Система управления пространственно приближенная к датчикам и использующая устройства.	1. Локальная 2. Встраиваемая (централизованная) 3. Независимая 4. Зависимая
12. Для обеспечения надежности работы распределённых систем управления функции управления техническим процессом возлагаются на	1. Вычислительные системы высшего уровня 2. Микроконтроллеры низшего уровня 3. Пульт управления 4. Цифровой регулятор
13. Режим в котором система должна обеспечивать формирование управляющих сигналов.	1. Реального времени 2. Виртуального времени 3. Функционирования 4. Отказа
14. Программируемое однокристальное вычислительное устройство с встроенным набором средств ввода и вывода, применяемое для решения задач управления и первичной обработки данных технических системах.	1. Микроконтроллер 2. Пульт управления 3. Цифровой регулятор 4. Устройство сопряжения с объектом
15. Какой из вариантов не является методом увеличения производительности вычислительных систем параллельной обработки данных.	1. Совершенствование элементной базы 2. Повышение входного напряжения 3. Использование методов параллельной обработки (структурный) 4. Математический метод
16. Какой из вариантов не является особым типом ошибок вычислительных систем па-	1. Взаимные блокировки процессов (дэдлоки)

раллельной обработки.	2. Нехватка напряжения (вольтлоки) 3. Не использование альтернатив (ливлоки) 4. Невозможность получить ресурс (голодание)
17. Эта обработка осуществляется на нескольких параллельно работающих устройствах, причём каждый элемент осуществляет обработку соответствующей порции данных от начала и до конца (суперскалярная архитектура).	1. Многостадийная обработка 2. Многоэлементная обработка 3. Многомашинная обработка 4. Повремённая
18. В случае этой обработки процесс обработки данных разбивается на несколько фаз, которые выполняются последовательно.	1. Многостадийная обработка 2. Многоэлементная обработка 3. Многомашинная обработка 4. Повремённая
19. В чем сходство MISD и SISD систем.	1. Имеют одиночный поток данных 2. Имеют одиночный поток команд 3. Имеют множественный поток команд 4. Имеют множественный поток данных
20. В чем сходство MIMD и SIMD систем.	1. Имеют одиночный поток данных 2. Имеют одиночный поток команд 3. Имеют множественный поток команд 4. Имеют множественный поток данных
21. Какие компьютеры представляют собой регулярную структуру, состоящую из цепочки последовательно соединенных процессоров, образующих процессорный конвейер.	1. MISD 2. SISD 3. MIMD 4. SIMD
22. В каких компьютерах процесс обработки данных делится на последовательно выполняемые этапы, каждый этап выполняется на отдельном процессоре. А единый поток данных поступает на вход всего процессорного конвейера.	1. MISD 2. SISD 3. MIMD 4. SIMD
23. В каком классе вычислительных машин процессоры обмениваются информацией через единую память, при этом возможно конфликты процессоров.	1. Многопроцессорные вычислительные системы 2. Однопроцессорные вычислительные системы 3. Персональный компьютер 4. Многомашинные вычислительные системы
24. В каком классе многопроцессорных вычислительных систем информация между модулями передается в режиме разделения времени, то есть на шине присутствует в определённый момент времени информация от одного модуля.	1. С общей шиной 2. С многовходовой памятью 3. Модульной 4. Многомашинная
25. В каком классе многопроцессорных вычислительных систем память имеет персональные выходы и управляющие схемы для подключения каждого микропроцессо-	1. С общей шиной 2. С многовходовой памятью 3. Модульной 4. Многомашинная

ра.	
26. В системах данного типа реализуется асинхронный вычислительный процесс, а каждый микропроцессор выполняет свою собственную программу или её участок. В таких системах происходит постоянное распараллеливание вычислений.	1. Многопроцессорные вычислительные системы 2. Однопроцессорные вычислительные системы 3. Персональный компьютер 4. Многомашинные вычислительные системы
27. Такие системы относятся к системам со слабой связью состоят из нескольких компьютеров, объединённых на основе сетевых средств.	1. Сетевые 2. Системы массового параллелизма 3. Одноядерные 4. Многомашинные комплексы
28. Для чего используется многомашинная связь на уровне внешних устройств.	1. Для увеличения входов в памяти 2. Для организации общего объема адресного пространства для внешних запоминающих устройств 3. Для уменьшения объема внешних запоминающих устройств 4. Для организации общего объема данных во внутренних запоминающих устройствах
29. При помощи какого устройства осуществляется взаимодействие вычислительных систем на уровне канал - канал	1. Адаптер 2. Внешняя память 3. Внутренняя память 4. Датчик
30. Для этой системы характерно то, что большинство узлов имеют логически и физически распределённую между процессорами память, а каждый узел выполняет отдельный процесс. В узле хранится результат промежуточных вычислений.	1. Сетевые 2. Системы массового параллелизма 3. Одноядерные 4. Многомашинные комплексы
31. Как в системах массового параллелизма происходит передача данных между узлами.	1. По факту готовности некого процесса 2. Перед началом выполнения центральной программы 3. Под управлением центральной программы 4. По истечении некого времени
32. Сеть обмена и распределённой обработки информации, образуемая множеством взаимосвязанных абонентских систем и средствами связи.	1. Вычислительная сеть 2. Телекоммуникационная вычислительная сеть 3. Производственная сеть 4. Сеть микропроцессоров
33. Что не включает в себя телекоммуникационная система.	1. Бытовая техника 2. Физическая среда передачи 3. Аппаратные средства, обеспечивающие взаимодействие абонентских станций 4. Программные средства, обеспечивающие взаимодействие абонентских станций

34. Что не является важнейшей функцией телекоммуникационной системы.	1. Синхронизация взаимодействия абонентских систем при обмене информацией 2. Коммутация соединений 3. Маршрутизация сообщений 4. Создание положительной мотивации
35. Какая последовательность действий, называемая протоколом, верна.	1. Запрос данных – подтверждение запроса – ответ – подтверждение ответа 2. Запрос данных – ответ – подтверждение запроса - подтверждение ответа 3. Ответ - запрос данных - подтверждение запроса – подтверждение ответа 4. Подтверждение ответа – подтверждение запроса – ответ – запрос
36. Эти сети, объединяющие абонентов района, города, области. Удаленность абонентов обычно составляет десятки – сотни километров.	1. Глобальные сети (WAN) 2. Региональные сети (MAN) 3. Локальные сети (LAN) 4. Системные сети (SAN)
37. Это высоко производительные вычислительные сети, объединяющие до нескольких сотен узлов, с длиной связей до 100 м, располагаемых чаще всего в специальном машинном зале.	1. Глобальные сети (WAN) 2. Региональные сети (MAN) 3. Локальные сети (LAN) 4. Системные сети (SAN)
38. Логическая и техническая организация телекоммуникационных вычислительных сетей, включающая совокупность сетевых аппаратных и программных средств, методов доступа и используемых протоколов.	1. Архитектура сети 2. Сетевая совокупность 3. Детерминант сети 4. Протокольная сеть
39. Какой из вариантов не является верным типом обработки в логических структурах.	1. Переадресация данных 2. Изменение информационного содержания сообщения 3. Обработка сообщения 4. Информационная обработка
40. Эти интерфейсы обеспечивают связь между соседними иерархическими уровнями внутри одного узла.	1. Протокольный интерфейсы 2. Рассылочные интерфейсы 3. Обслуживающие интерфейсы 4. Интерфейсы доступа
41. Уровень предназначен для организации доступа каждого пользователя к программам удаленных пользователей. На нём предусмотрены такие функции как пересылка файлов, пересылка заданий обращение к базам данных.	1. Сеансовый уровень 2. Прикладной уровень 3. Физический 4. Сетевой
42. Уровень обеспечивает средства, необходимые абонентам для организации, синхронизации и административного управления обменом данных между ними, обеспечивает по запросам процессов создание	1. Сеансовый уровень 2. Прикладной уровень 3. Физический 4. Сетевой

портов для приёма и передачи сообщений, организацию соединений, а также выполнение универсальных функций управления для протоколов представительного уровня.	
43. Уровень обеспечивает передачу двоичных сигналов через передающую среду. На нём реализуются такие функции управления каналом связи, как подключение и отключение, формирование и приём сигналов.	1. Канальный уровень 2. Прикладной уровень 3. Физический 4. Сетевой
44. Дейтаграмма – это	1. Пакет, заголовок которого содержит адрес отправителя. 2. Пакет, заголовок которого содержит адрес получателя и необходимые маршрутные признаки 3. Совокупность маршрутов, необходимая для нормального функционирования канала данных 4. Совокупность заголовков, предназначенных для установления адреса отправителя
45. Устройство, осуществляющее поочерёдное подключение нескольких входных каналов связи на один выходной без изменения скорости передачи.	1. Повторитель 2. Коммутатор 3. Мост 4. Маршрутизатор
46. Устройство, выполняющее соединение на транспортном уровне. Обеспечивает соединение логически не связанных сетей, имеющих одинаковые протоколы на сеансовом уровне и выше, создает нужный логический канал и передаёт сообщение по назначению.	1. Повторитель 2. Коммутатор 3. Мост 4. Маршрутизатор
47. Какой кабель необходимо использовать, если необходимо обеспечить высокую помехозащищенность и наиболее высокую скорость передачи сигналов.	1. Витая пара 2. Волоконно-оптический 3. Инфракрасный 4. Коаксиальный
48. Код используется во многих системах передачи данных, обеспечивает изменение уровня сигнала при передаче каждого бита, что обеспечивает возможность самосинхронизации.	1. Код NRZ 2. Код RZ 3. Манчестерский код 4. Код AMI
49. Какая технология использует толстый (полдюйма) коаксиальный кабель, обеспечивает сегменты длиной до 500 м, с числом подключаемых к сегменту узлов до 100.	1. 10 Base-F 2. 10 Base-5 3. 10 Base-T 4. 10 Base-2
50. Протокол поддерживает передачу сообщений электронной почты между произвольными узлами сети Интернет, содержит механизмы промежуточного хранения почты и повышения надёжности доставки. Обеспечивает группирование сообщений в	1. Telnet 2. FTP 3. SMTP (Simple Mail Transfer Protocol) 4. HTTP (Hyper Text Transfer Protocol)

адрес одного получателя и размножение нескольких копий одного сообщения для передачи в разные адреса. Допускает использование транспортных служб.	

10.2. КОМПЛЕКСНЫЙ ТЕСТ

Комплексный тест разбит на темы, нумерация которых соответствует темам лекционных занятий

1.1. Микропроцессорные системы – определение, структура, типы

1	Микропроцессор – это ……	а) программно-управляемый элемент, выполняющий операции с буквами б) программно-управляемый элемент, выполняющий арифметические операции в) программно-управляемый элемент, выполняющий арифметические и логические операции над числами, имеющий средство взаимодействия с устройствами памяти и устройствами ввода-вывода
2	Гибкая структура – это……	а) структура, которая может меняться только на этапе создания системы б) структура, которая меняется в режиме работы системы путем подачи управляющих воздействий в) структура, которая не меняется ни на этапе создания системы, ни в режиме работы
3	Исключите функцию, которая не выполняется микропроцессором	а) арифметические и логические операции б) пересылка данных в) операции с буквами г) временное хранение, обрабатываемой информации д) управление режимом работы
4	Принцип организации шинной структуры связи	а) при данном способе организации связи все сигналы передаются по общим линиям в различных направлениях б) каждое устройство соединено с другим по своим линиям связи, передает информацию не зависимо от

		других и по своим протоколам передачи в) при данном способе организации связи все сигналы передаются по общим линиям в одном направлении
5	Какого способа соединения выходных каскадов микросхем не существует	а) выход 2С, 2S б) выход с общим коллектором в) высокое импенданстное: 3С, 3S г) выход 3С, 4S

1.2. Организация обмена информацией в МПС

1	Для чего предназначена шина питания	а) для организации питания всей системы, по ней проходит сигнал +- 5В+-12В б) для организации питания всей системы, по ней проходит сигнал +- 3В+-5В в) для организации питания всей системы, по ней проходит сигнал +- 0-4А
2	Какого режима работы микропроцессорной систбемы нет	а) программного б) по прерываниям в) через УВВ г) прямого доступа к памяти
3	Что делает микропроцессор в цикле записи	а) выводит информацию б) считывает информацию в) записывает информацию
4	Какое количество адресов, обеспечивается шиной адреса	а) 2^{N-1}, N- количество линий связи б) 2^{N+1}, N- количество линий связи в) 2^{N}, N- количество линий связи
5	В чем смысл мультиплексирования Шины адреса и Шины данных	а) т.е. одни и те же линии связи в разные моменты используются для передачи, как адреса, так и данных б) т.е. разные линии связи в одни моменты используются для передачи, как адреса, так и данных в) т.е. одни и те же линии связи в одни моменты используются для передачи, как адреса, так и данных

1.3. Арбитраж шин

1.4. Методы повышения эффективности шин

1	Для какой системы свойственны статические приоритеты	а) для систем с гибкой структурой б) для систем с жесткой структурой в) для систем с гибкой и жесткой струк-

		турой
2	В чем заключается децентрализованный арбитраж	а) единый арбитр отсутствует, вместо этого каждый ведущий содержит блок управления доступа к шине б) присутствует арбитр, а так же каждый ведущий содержит блок управления доступа к шине в) в системе имеется специальное устройство, называемое центральным арбитром или контроллер шин
3	В чем суть конвейеризации транзакций	а) один адресный цикл сопровождается множественными циклами данных б) очередная транзакция устройством А не может быть послана до того, как устройство В завершит считывание предыдущей транзакции в) очередная транзакция устройством А может быть послана до того, как устройство В завершит считывание предыдущей транзакции
4	Что определяет разрядность шины данных	а) допустимый объем памяти б) скорость работы системы в) эффективность обмена МП с другими устройствами
5	Выберете основной недостаток организации обычной структуры связи	а) произвольные протоколы передачи информации б) все устройства включаются параллельно; не правильная передача информации или сбой одного из устройств может привести к сбою всей системы в) протоколы обмена информации унифицируются

1.5. Основные элементы МПС. Микропроцессор

1	Для чего предназначен выход CLK в МП	а) для подключения периферии б) для подключения тактового генератора в) CLK заземляется
2	В каком случае АЛУ в работе не участвует	а) во всех случаях б) при записи данных в) если команда сводится к пересылке данных
3	Исключите функцию, не подходящую для регистров	а) определяют адрес памяти, где находится выполняемая программа в данный мо-

		мент времени б) определяют текущий адрес стека в) выбор выполняемых команд
4	Логика управления	а) обеспечивает отключение МП от шины на время представления прямого доступа к памяти б) определяет взаимодействие всех устройств МП, синхронизирует работу и организует процедуры ввода-вывода в) определяет адрес памяти, где находится выполняемая программа в данный момент времени
5	Синхронный обмен информации заключается:	а) МП заканчивает работу тогда, когда устройство – исполнитель подтверждает выполнение операции специальным сигналом б) МП самостоятельно завершает обмен данных в) МП заканчивает работу тогда, когда устройство – исполнитель не подтверждает выполнение операции специальным сигналом

1.6. Основные элементы МПС. Память и устройства ввода/вывода

1	Оперативная память:	а) обращается с системной магистралью в циклах чтения или записи б) обращается с системной магистралью только в циклах записи в) обращается с системной магистралью только в циклах чтения
2	Память первоначального запуска процессора выполняется:	а) на ОЗУ памяти б) на ОЗУ и ПЗУ памяти в) на ПЗУ или flash памяти
3	Для чего используется PUSH команда работы со стеком	а) извлечения из стека б) помещения в стек г) может использоваться как для извлечения, так и для помещения в стек
4	Выберите группу УВВ, которая не входит в состав МПС	а) УВВ для кратковременного хранения информации б) Устройство интерфейса пользователя в) УВВ для длительного хранения информации г) таймерное устройство

1.7. Функционирование МПС. Адресация и ее особенности, регистры

1	Абсолютная адресация предполагает:	а) что операнд источник находится в памяти непосредственно за кодом б) что операнд находится во внутренних регистрах процессора в) что операнд располагается в памяти по некоторому адресу
2	Регистр указателя или смещения определяет:	а) местоположение внутри сегмента б) смещение в) адрес начала сегмента, т.е. место расположения в памяти
3	Индекс – это…….	а) множитель, обычно =1,2,4,8 б) содержимое индексного регистра процессора в) 8,16,32-разрядное число, включенное в код команды
4	Для чего используется регистр BX	а) для операции умножения, деления, обмена с УВВ б) является счетчиком в) для вычисления адреса
5	Регистры SP, BP, SI, DI, IP являются	а) сегментными регистрами б) указательными регистрами в) регистрами флагов

1.8. Программные основы работы МП

1	Бит регистра флагов SF является	а) флагом знака б) флаг четности результата в) флаг переноса
2	Какие команды не требуют выполнения операций над операндами	а) арифметические команды б) логические команды в) команды пересылки данных
3	По сравнению с какой командой команда CLK является более быстродействующей	а) mov _, 0h б) mov A, B в) такой нет
4	Циклический сдвиг	а) значение старшего разряда остается прежним, а младший бит заносится в регистр флагов б) значение младшего разряда остается прежним, а старший бит заносится в регистр флагов в) позволяет сдвигать код операнда впра-

		во, т.е. в сторону младших разрядов или влево
5	Какая команда выполняет установку регистра слово состояния при проверке операнда	а) CLC _ б) SEC_ в) TST

1.9. Микроконтроллеры. Основы организации

1	CISC микроконтроллеры	а) отличаются системой команд, имеют развитые возможности адресации б) характеризуются сокращенным набором команд в) все команды имеют фиксированную длину
2	Какая память в микроконтроллере не используется	а) память программ б) память данных в) регистровая память г) flash память д) все используются
3	Какие регистры в микроконтроллере не используются	а) регистры программ б) регистры процессора в) регистры ввода-вывода г) регистры управления
4	В каком режиме микроконтроллера центральный процессор отключается, но продолжают функционировать периферийные устройства	а) активный режим б) режим останова в) режим ожидания
5	Какой способ не используется для задания тактовой частоты	а) с помощью LC цепи б) с помощью внешней RC цепи в) с помощью кварцевого резонатора г) с помощью керамического резонатора
6	Сколько выходных операндов формируют логические команды	а) 3 б) 4 в) 1

1.10. Внутренние и внешние связи в микроконтроллерах

1	Какое значение сигнала считывается при вводе данных с порта микроконтроллера	а) содержимое триггера данных б) содержимое триггера регистра управления в) логическое «И» над содержимым триг-

		гера данных и значением сигнала на внешнем выводе МК г) значение сигнала на внешнем выводе МК
2	При каких условиях триггер переполнения таймера/счетчика генерирует запрос на прерывание МК	а) при переполнении таймера/счетчика б) при сбросе таймера/счетчика в) при сбросе запроса на прерывание г) при переполнении таймера/счетчика, если прерывания от таймера разрешены
3	Для чего в первую очередь предназначен модуль выходного сравнения МК	а) для формирования временных интервалов заданной длительности б) для сравнения информации на двух портах МК в) для измерения интервалов между событиями на выходах МК г) для выдачи импульсов фиксированной частоты
4	Какой тип логической функции позволяет реализовать объединение «квазидвунаправленных» выводов МК	а) сложение по модулю два б) логическое «И» в) логическое «ИЛИ» г) константа «1»
5	Для чего в первую очередь предназначен модуль входного захвата МК	а) для отслеживание изменений сигнала на входе МК б) для подсчета количества событий на входе МК в) для выдачи импульсов фиксированной длительности
6	Какой параметр выходного сигнала изменяется при широтно-импульсной модуляции	а) частота б) уровень логического «0» в) скважность г) уровень логической «1»
7	Какова типичная разрядность таймера/счетчика в составе МК	а) 32 б) 64 в) 8 или 16 г) 4
8	Какие ошибки измерения позволяет исключить использование режима входного захвата таймера/счетчика МК	а) ошибки связанные с временем перехода к подпрограмме обработки прерывания б) потери времени на перезагрузку таймера/счетчика в) потери времени при фиксации события захвата г) потери времени при чтении содержимого регистра входного захвата
9	Что называется «вектором	а) уровень приоритета данного типа пре-

	прерывания» МК	рывания б) состояние линии приема запросов на прерывание в) адрес перехода к подпрограмме обработки прерывания г) состояние бита разрешения прерывания МК

1.11. Аппаратные средства микроконтроллеров

1	Какой модуль микроконтроллера прекращает работу в режиме ожидания	а) центральный процессор б) тактовый генератор в) таймер г) блок прерывания
2	Какой способ тактирования микроконтроллера обеспечивает наивысшую стабильность частоты	а) с использование RC-цепи б) с использованием кварцевого резонатора в) с использованием керамического резонатора
3	Что используется в качестве простейшего устройства аналогового ввода информации в МК	а) АЦП б) компаратор напряжения в) резистивный делитель г) емкостной делитель
4	Как зависит ток потребления микроконтроллера от напряжения питания	а) не зависит б) приблизительно линейно в) обратно пропорционально г) квадратично
5	Зачем нужна задержка времени при запуске тактового генератора микроконтроллера	а) для стабилизации частоты генератора б) для минимизации энергопотребления при запуске МК в) для перевода регистров МК в начальное состояние г) для исключения выдачи ложных сигналов на выходах МК
6	АЦП какого типа чаще всего используют в составе микроконтроллера	а) интегрирующие б) параллельные в) последовательного приближения г) на основе преобразователей напряжение-частота
7	Как зависит ток потребления КМОП-микроконтроллера от частоты тактового генератора	а) не зависит б) приблизительно линейно в) пропорционально корню квадратному от частоты г) квадратично

8	Что происходит при переполнении сторожевого таймера микроконтроллера	а) формирование сигнала запроса прерывания б) переход в режим пониженного энергопотребления в) сброс МК г) инкремент таймера/счетчика МК
9	Что используется в качестве простейшего ЦАП на выходе МК	а) широтно-импульсный модулятор с фильтром нижних частот б) операционный усилитель в) электронный ключ г) усилитель напряжения

1.12. Проектирование устройств на микроконтроллерах

1	Какая сфера применения является наиболее типичной для цифровых устройств на МК	а) обработка данных эксперимента б) решение задач математического моделирования в) задачи управления объектами г) распознавание образов
2	Что такое «программный симулятор»	а) программа, заменяющая МК в составе устройства б) средство для исполнения разработанной программы на программно-логической модели МК в) программа для оптимизации размещения данных в памяти МК г) программа, подменяющая внутреннее ЗУ программ МК
3	Какую функцию выполняет «монитор» на плате развития	а) устройство для контроля напряжения питания МК б) простейшее средство отладки в) устройство для контроля температуры корпуса МК г) средство для крепления платы развития в устройстве
4	Что включает в себя понятие «работа в реальном масштабе времени»	а) максимально достижимое на данный момент быстродействие б) обеспечение реакции на внешнее событие в течение определенного интервала времени в) возможность выдачи сигналов строго определенной длительности г) включение и выключение устройства по сигналам точного времени

5	Что такое «внутрисхемный» эмулятор	а) программа для контроля состояния внутренних регистров МК б) аппаратное устройство для реализации пошагового режима работы МК в) программно-аппаратное средство для замены МК в реальной схеме г) программа, подменяющее внутреннее ЗУ программ МК
6	Что такое «эмулятор ПЗУ»	а) устройство для расширения объема внешней памяти МК б) устройство для «вскрытия» содержимого памяти программ МК в) схема сопряжения МК с внешней памятью программ г) программно-аппаратное средство для замещения ПЗУ
7	Что включает в себя понятие «закрытая архитектура»	а) невозможность доступа к памяти программ МК б) невозможность доступа к памяти данных МК в) отсутствие возможности изменения тактовой частоты г) реализация большинства функций устройства внутренними средствами
8	Что такое «плата развития»	а) конструктор для макетирования электронных устройств б) устройство для увеличения тактовой частоты МК в) схема для сопряжения МК с внешними устройствами г) плата, выставленная на пробную продажу
9	Что такое «виртуальное» периферийное устройство МК	а) периферийный модуль, поставляемый на заказ б) периферийный модуль, реализованный программными средствами в) периферийный модуль, находящийся в стадии разработки г) периферийный модуль с изменяемыми режимами работы

1.13. Введение: микроконтроллеры серии PIC и AVR

1	В каком году появились первые микроконтроллеры ком-	а) в 1981 б) в 1982

	пании Microchip PIC16C5x	в) в конце 1980-х годов
2	За счет чего достигается высокая скорость выполнения команд в PIC-контроллерах	а) за счет использования одношинной гарвардской архитектуры б) за счет использования двухшинной гарвардской архитектуры в) за счет использования традиционной одношинной фон-неймовской архитектуры
3	В какой серии микроконтроллеров используется 58 команд	а) PIC 17CXXX б) PIC 18CXXX в) PIC 16C5x
4	Какова максимальная частоты PIC16F8X	а) 10 МГц б) 20 МГц в) 30 МГц
5	Исключите основную особенность, которая не является характерной для микроконтроллеров подгруппы PIC16F8X	а) раздельные шины данных (8 бит) и команд (14 бит) б) восьмиуровневый аппаратный стек в) используется только 36 простых команд
6	Чем различаются микроконтроллеры подгруппы PIC16F8X между собой	а) объемом ОЗУ данных, а также объемом и типом памяти программ б) объемом ПЗУ в) числом источников прерываний

1.14. Принципы работы, организация памяти и особенности выполнения команд для микроконтроллеров PIC и AVR

1	По какому адресу находится вектор сброса в микроконтроллерах PIC16F8X	а) 0004h б) 0000h в) 0001h
2	Какое количество банков содержит память данных PIC16F8X	а) 1 б) 3 в) 2
3	Что является счетчиком команд PIC16F8X	а) PLC и PCLATH б) LCP и HTA б) RETLN
4	Какие биты счетчика команд загружаются из PCLATH	а) старшие и младшие биты б) старшие биты в) младшие биты
5	К какой области принадлежит стек	а) не принадлежит ни к программной области, ни к области данных б) принадлежит к программной области в) принадлежит к области данных
6	Что является признаком кос-	а) обращение к регистру конфигурации

	венной адресации	б) обращение к регистру статуса в) обращение к регистру INDF

1.15. Организация обмена с внешними устройствами, память, прерывания для микроконтроллеров PIC и AVR

1	Какого типа является операции с портом А	а) тип «запись – модификация – чтение» б) тип «чтение – модификация – запись» в) тип «модификация – чтение – запись»
2	Когда происходит запись в порт вывода	а) в конце командного цикла б) в начале командного цикла в) в конце и в начале командного цикла
3	Какой компонент не содержит программный модуль TIMER0	а) 8 разрядный таймер/счетчик TMR0 с возможность чтения и записи как регистр б) мультиплексор входного сигнала для выбора внутреннего или внешнего тактового сигнала в) формирователь запроса прерывания по обнуления регистра г) схема выбора фронта внешнего тактового сигнала
4	Как должен быть сброшен бит запроса T0IF при обработке прерывания	а) программно б) аппаратно в) произвольным способом
5	Какой памятью является память данных EEPROM	а) ОЗУ б) ПЗУ в) РПЗУ
6	Для чего предназначен бит WRERR	а) бит управления записью б) бит признака ошибки записи в EEPROM в) бит управления чтением
7	Что дает чтение регистра EECON2	а) дает нули б) дает единицы в) дает нули и единицы
8	Какого источника прерываний не имеют МК подгруппы PIC16F8X	а) внешнее прерывание с выводом RB0/INT б) прерывание от переполнения счетчика/таймера TMR0 в) прерывание по окончании записи данных в EEPROM г) прерывание при чтении данных

1.16. Специальные функции и система команд микроконтроллеров PIC и AVR

1	Какие команды обнуляют сторожевой таймер	а) CLRWDT и SLEEP б) MCLR в) PLRT
2	XT – это	а) высокочастотный кварцевый генератор б) низкочастотный кварцевый генератор в) стандартный кварцевый генератор
3	Когда устанавливаются 5 или 6 бит конфигурации в микроконтроллере подгруппы PIC16F8X, которые хранятся в EEPROM	а) на этапе разработки МК б) на этапе программирования МК в) на этапе эксплуатации
4	Чем является поле TOS в системе команд МК семейства PIC16XXX	а) счетчик команд б) вершина стека в) бит разрешения всех прерываний
5	Для чего используется в PIC микроконтроллерах команды работы с байтами	а) они оперируют с однобитными операндами б) используют при выполнении операции явно заданные операнды, которые являются частью команды в) используются для пересылки данных между регистрами и выполняют математические операции над их содержимым
6	Что означает команда NOP	а) означает отсутствие операции б) циклический сдвиг в) меняет местами тетрады в регистре
7	Изменяется ли содержимое регистра STATUS при использовании команды MOVLW K	а) изменяется б) не изменяется в) в зависимости от K
8	Для чего используется команда RETURN	а) для возвращения данных из порта б) сбрасывает в 0 содержимое всех портов в) приводит к восстановлению адреса команды, следующей за командой вызова подпрограммы

1.17. Особенности программирования и отладки, разработка программного кода для микроконтроллеров PIC и AVR

1	Что размещается в поле метки	а) операнды, участвующие в операции б) символическое имя ячейки памяти, в

		которой хранится отмеченный операнд в) мнемонические обозначения команды, которые непосредственно транслируются в машинный код
2	. Оператор это……..	а) число, выраженное в некоторой системе счисления б) арифметические символы, подобные + и -, которые используются при формировании выражений в) это последовательность любых допустимых ASC‖ символов заключенная в двойные кавычки
3	Оператор % - это	а) проценты б) деление в) модуль
4	Возможно ли в поле комментария применять любые символы	а) да б) нет в) если они правильные
5	Что означает расширение .OBJ используемые по умолчанию в MPASM	а) выходной файл листинга, генерируемый по умолчанию в MPASM б) выходной файл перемещаемого объектного кода из MPASM в) входной файл ассемблера для MPASM
6	Какого типа директив в MPASM не существует	а) директива данных б) директива листинга в) директива записи/чтения г) управляющие директивы д) макро-директивы
7	Какие из директив управляют распределением памяти и обеспечивают доступ к символическим обозначениям данных	а) директивы листинга б) директивы данных в) директивы управления

1.18. Разработка программного обеспечения для микроконтроллеров PIC и AVR

1.19. Макет микропроцессорной системы и программирование простейших задач для микроконтроллеров PIC и AVR

1	Для чего нужен симулятор MPSIM	а) для отладки программного обеспечения PIC-контроллеров б) позволяет создавать и модифицировать файлы библиотек

		в) для генерирования кода программы непосредственного при ассемблировании
2	Что означает данная запись GOTO LOOP	а) вызов подпрограммы LOOP б) переход к метке LOOP для повторения процесса в) маскирование
3	Регистр конфигурации OPTION является доступным	а) по чтению и записи регистром б) только по чтению регистром в) только по записи регистром
4	Сколько памяти программ адресуют микроконтроллеры PIC16F83 и PIC16CR83	а) 256x14 б) 1Kx14 в) 512x14
5	Сколько мс составляет номинальная выдержка WDT (сторожевого таймера)	а) 15 мс б) 16 мс в) 18 мс
6	Регистр W является	а) рабочим регистром б) константой в) адресом регистра
7	Что означает команда МК подгруппы PIC16F8X COMF f, d	а) сброс регистра f б) инверсия регистра f в) декремент регистра f
8	Что означает команда МК подгруппы PIC16F8X «BTFSC f, b»	а) пропустить команду, если бит f равен 0 б) пропустить команду, если бит f равен 1 в) переход по адресу
9	В каком порядке следуют типы информации в ассемблерной строке	а) мнемоника, метки, операнды, комментарии б) метки, мнемоника, операнды, комментарии в) операнды, метки, мнемоника, комментарии г) метки, операнды, мнемоника, комментарии
10	На чем основан программный метод подавления «дребезга» контактов при вводе данных в микроконтроллер	а) на увеличении частоты опроса б) на использовании специальных команд подавления «дребезга» в) на блокировании соответствующего порта на время «дребезга» г) при повторном чтении через небольшой интервал времени и сравнении результатов
11	На чем основан метод формирования программной задержки	а) на подсчете времени выполнения участка программы и повторении ее определенное количество раз б) на задержке выполнения программы с

		помощью специальной команды задержки в) на использовании аппаратных прерываний г) ответа нет
12	Что означает команда МК подгруппы PIC16F8X «CLRWDT»	а) формирует программную задержку б) формирует аппаратную задержку в) опрашивает порт, и записывает результат в регистр W г) исключает влияние сброса по переполнению сторожевого таймера
13	Что означает директива «LIST P=16C84»	а) определяет формировать или нет файл листинга б) задает тип системы исчисления по умолчанию в) определяет тип процессора г) задает тип по умолчанию
14	Что означает команда МК подгруппы PIC16F8X «XORLW k»	а) исключающее ИЛИ константы k и W б) переносит содержимое W в k в) пересылает константу в W г) инкрементирует регистр W указанное k-раз

11. КОМПЛЕКТ БИЛЕТОВ

11.1. ЭКЗАМЕНАЦИОННЫЕ БИЛЕТЫ

11.1.1 СЕМЕСТР 7

ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ №1

1. Микропроцессорные системы. Основные определения, структуры с гибкой и жесткой логикой, основной элемент системы, организация связей.
2. Устройства ввода/вывода в микропроцессорной системе.
3. Задача.

Пользуясь системой команд микропроцессора КР580ИК80 (лабораторного стенда «УМК») создать программу по заданию, предложенному ниже. При этом необходимо:

а) Написать законченную программу на языке «Ассемблер»;

б) Перевести полученную программу в машинный код;

в) Описать последовательность действий по записи и выполнению программы на лабораторном стенде «УМК»;

г) Для каждого этапа выполнения программы записать содержимое используемых: ячеек памяти, регистров, регистра флагов;

д) Предложить возможные альтернативы по решению задачи.

Задание:

Написать программу сложения чисел массива, размещенного в памяти начиная с адреса «0C00h». Считать, что первым элементом массива является общее число складываемых чисел. Результат сложения чисел сохранить в памяти по адресу «0A00h».

ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ №2

1. Структура МП системы с шинной организацией. Общий принцип работы МП системы.
2. Функционирование микропроцессорной системы. Основы программного режима работы, методы адресации.
3. Задача.

Пользуясь системой команд микропроцессора КР580ИК80 (лабораторного стенда «УМК») создать программу по заданию, предложенному ниже. При этом необходимо:

а) Написать законченную программу на языке «Ассемблер»;

б) Перевести полученную программу в машинный код;

в) Описать последовательность действий по записи и выполнению программы на лабораторном стенде «УМК»;

г) Для каждого этапа выполнения программы записать содержимое используемых: ячеек памяти, регистров, регистра флагов;

д) Предложить возможные альтернативы по решению задачи.

Задание:

Написать программу опроса 7 датчиков, подключенных к порту «01h». Если срабатывает хотя бы один датчик, то результат опроса (число, считываемое с порта) сохранить по адресу «0E01h». Сами датчики необходимо опросить сто раз.

ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ №3

1. Микропроцессорные Режимы работы МП системы. Архитектура современных микропроцессорных систем.
2. Функционирование микропроцессорной системы. Сегментное разбитие памяти.
3. Задача.

Пользуясь системой команд микропроцессора КР580ИК80 (лабораторного стенда «УМК») создать программу по заданию, предложенному ниже. При этом необходимо:

а) Написать законченную программу на языке «Ассемблер»;

б) Перевести полученную программу в машинный код;

в) Описать последовательность действий по записи и выполнению программы на лабораторном стенде «УМК»;

г) Для каждого этапа выполнения программы записать содержимое используемых: ячеек памяти, регистров, регистра флагов;

д) Предложить возможные альтернативы по решению задачи.

Задание:

Составить программу нахождения разности чисел, расположенных в памяти. Считать, что числа располагаются, начиная с адреса «0801h». Общее количество чисел десять. Первое число является уменьшаемым, остальные числа вычитаемые. Если после вычитания всех чисел результат положительный, то его необходимо сохранить в памяти по адресу «0801h». Если на каком либо этапе появился отрицательный результат, то порядковый номер этого числа необходимо сохранить в регистре «D».

ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ №4

1. Обмен информации по шинам в МП системе. Понятие и характеристики циклов обмена. Особенности шин.
2. Особенности адресации данных. Регистры микропроцессора.
3. Задача.

Пользуясь системой команд микропроцессора КР580ИК80 (лабораторного стенда «УМК») создать программу по заданию, предложенному ниже. При этом необходимо:

а) Написать законченную программу на языке «Ассемблер»;

б) Перевести полученную программу в машинный код;

в) Описать последовательность действий по записи и выполнению программы на лабораторном стенде «УМК»;

г) Для каждого этапа выполнения программы записать содержимое используемых: ячеек памяти, регистров, регистра флагов;

д) Предложить возможные альтернативы по решению задачи.

Задание:

Составить программу по вычислению формулы: $a4 = (a1 + a2) \times a3$. Число «$a1$» расположено в регистре «B», число «$a2$» расположено в регистре «D», число «$a3$» в регистре «E». Результат – число «$a4$» сохранить в регистре «H».

ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ №5

1. Обмен информации по шинам в МП системе. Цикл программного обмена.
2. Основные команды микропроцессора. Команды пересылки данных и арифметические команды.
3. Задача.

Пользуясь системой команд микропроцессора КР580ИК80 (лабораторного стенда «УМК») создать программу по заданию, предложенному ниже. При этом необходимо:

а) Написать законченную программу на языке «Ассемблер»;

б) Перевести полученную программу в машинный код;

в) Описать последовательность действий по записи и выполнению программы на лабораторном стенде «УМК»;

г) Для каждого этапа выполнения программы записать содержимое используемых: ячеек памяти, регистров, регистра флагов;

д) Предложить возможные альтернативы по решению задачи.

Задание:

Написать программу по передаче 20 чисел, расположенных в памяти начиная с адреса «0900h», в порт «10h». Если среди чисел встретится нулевое (равное нулю), то управление необходимо передать подпрограмме, расположенной по адресу «08AAh».

ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ №6

1. Обмен информации по шинам в МП системе. Цикл обмена по прерываниям.
2. Основные команды микропроцессора. Логические команды и команды переходов.

3. Задача.

Пользуясь системой команд микропроцессора КР580ИК80 (лабораторного стенда «УМК») создать программу по заданию, предложенному ниже. При этом необходимо:

а) Написать законченную программу на языке «Ассемблер»;

б) Перевести полученную программу в машинный код;

в) Описать последовательность действий по записи и выполнению программы на лабораторном стенде «УМК»;

г) Для каждого этапа выполнения программы записать содержимое используемых: ячеек памяти, регистров, регистра флагов;

д) Предложить возможные альтернативы по решению задачи.

Задание:

Написать программу нахождения первого четного числа в элементах массива, считываемого из порта «03h». Порядковый номер первого считанного четного числа сохранить по адресу «08FFh».

ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ №7

1. Обмен информации по шинам в МП системе. Цикл обмена в режиме прямого доступа к памяти. Особенности организации обмена по шинам.

2. Основы организации микроконтроллеров: структура, особенности, типы.

3. Задача.

Пользуясь системой команд микропроцессора КР580ИК80 (лабораторного стенда «УМК») создать программу по заданию, предложенному ниже. При этом необходимо:

а) Написать законченную программу на языке «Ассемблер»;

б) Перевести полученную программу в машинный код;

в) Описать последовательность действий по записи и выполнению программы на лабораторном стенде «УМК»;

г) Для каждого этапа выполнения программы записать содержимое используемых: ячеек памяти, регистров, регистра флагов;

д) Предложить возможные альтернативы по решению задачи.

Задание:

Написать подпрограмму формирования задержки. Считать, что необходимое время задержки храниться в регистре «E». Начало подпрограммы расположить по адресу «0800h». После выполнения подпрограммы, управление необходимо передать участку основной программы, расположенной начиная с адреса «0C00h».

ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ №8

1. Арбитраж шин. Схемы распределения приоритетов.

2. Функционирование процессорного ядра микроконтроллеров, различные архитектуры микроконтроллеров.
3. Задача.
Пользуясь системой команд микропроцессора КР580ИК80 (лабораторного стенда «УМК») создать программу по заданию, предложенному ниже. При этом необходимо:

а) Написать законченную программу на языке «Ассемблер»;
б) Перевести полученную программу в машинный код;
в) Описать последовательность действий по записи и выполнению программы на лабораторном стенде «УМК»;
г) Для каждого этапа выполнения программы записать содержимое используемых: ячеек памяти, регистров, регистра флагов;
д) Предложить возможные альтернативы по решению задачи.

Задание:

Написать программу управления исполнительным механизмом, подключенным к порту «05h». Дискретный сигнал на движение в сторону «больше» соответствует наличию «единицы» на 5 выходе (D4) порта «05h». Дискретный сигнал на движение в сторону «меньше» соответствует наличию «единицы» на 4 выходе (D3) порта «05h». Информация о том, в какую сторону необходимо переместить механизм, считывается из порта «01h», так если считанное число больше нуля – то механизм необходимо передвинуть в сторону меньше (если считанное число с порта «01h» равно нулю, то механизм необходимо передвинуть в сторону меньше).

ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ №9

1. Схемы арбитража: централизованный арбитраж.
2. Система команд микроконтроллеров. Схема синхронизации в микроконтроллерах и основы организации памяти.
3. Задача.
Пользуясь системой команд микропроцессора КР580ИК80 (лабораторного стенда «УМК») создать программу по заданию, предложенному ниже. При этом необходимо:

а) Написать законченную программу на языке «Ассемблер»;
б) Перевести полученную программу в машинный код;
в) Описать последовательность действий по записи и выполнению программы на лабораторном стенде «УМК»;
г) Для каждого этапа выполнения программы записать содержимое используемых: ячеек памяти, регистров, регистра флагов;
д) Предложить возможные альтернативы по решению задачи.

Задание:

Написать программу «Пожарная сигнализация». Считать, что имеется 5 комнат, в каждой из которой имеется 3 дискретных датчика. Информация с датчиков 1 комнаты поступает на порт «01h», со второй на «02h» с третьей на

«03h». Предполагать так же, что «первый» датчик подключен к контакту D0, «второй» к D1, «третий» к D2 соответствующего порта.
Если в какой либо из комнат сработали хотя бы 2 датчика, то сигнал «пожар» (логическая единица) необходимо сформировать на контакте D0 порта «06h». Номер комнаты, в которой сработал ХОТЯ БЫ ОДИН датчик, сохранять в памяти начиная с адреса «09AAh».

ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ №10

1. Схемы арбитража: децентрализованный арбитраж.
2. Внутренние и внешние связи в микроконтроллерах. Порты ввода/вывода.
3. Задача.

Пользуясь системой команд микропроцессора КР580ИК80 (лабораторного стенда «УМК») создать программу по заданию, предложенному ниже. При этом необходимо:

а) Написать законченную программу на языке «Ассемблер»;
б) Перевести полученную программу в машинный код;
в) Описать последовательность действий по записи и выполнению программы на лабораторном стенде «УМК»;
г) Для каждого этапа выполнения программы записать содержимое используемых: ячеек памяти, регистров, регистра флагов;
д) Предложить возможные альтернативы по решению задачи.

Задание:

Написать программу «Управление двухпозиционной нагрузкой». Считать, что информация по «контролируемой величине», проходящей через модуль АЦП, считывается с порта «02h» (т.е. представлена числом от «00h» до «FFh»). Информация по «заданию величины», аналогичным образом считывается с порта «01h». Сам исполнительный механизм подключен к выходам порта «03h».
Управление реализовать: подачей сигнала на 3 выход (D2) порта «03h» если исполнительный механизм необходимо открыть в сторону «больше»; и подачей сигнала на 6 выход (D5) порта «03h» если исполнительный механизм необходимо открыть в сторону «меньше».

ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ №11

1. Схемы арбитража: опросные схемы.
2. Внутренние и внешние связи в микроконтроллерах. Таймеры.
3. Задача.

Пользуясь системой команд микропроцессора КР580ИК80 (лабораторного стенда «УМК») создать программу по заданию, предложенному ниже. При этом необходимо:

а) Написать законченную программу на языке «Ассемблер»;

б) Перевести полученную программу в машинный код;

в) Описать последовательность действий по записи и выполнению программы на лабораторном стенде «УМК»;

г) Для каждого этапа выполнения программы записать содержимое используемых: ячеек памяти, регистров, регистра флагов;

д) Предложить возможные альтернативы по решению задачи.

Задание:

Написать программу по нахождению положительных чисел расположенных в элементах массива. Массив располагается начиная с адреса «0BCCh» по адрес «0CFFh». Общее количество положительных чисел расположить в ячейке памяти по адресу «0A00h». Факт нахождения положительного числа сигнализировать подачей на выход порта «01h» единицы в старший байт (D6).

ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ №12

1. Методы повышения эффективности обмена по шинам в микропроцессорной системе. Пакетный режим, конвейеризация транзакций.
2. Внутренние и внешние связи в микроконтроллерах. Процессоры событий и модуль прерываний.
3. Задача.

Пользуясь системой команд микропроцессора КР580ИК80 (лабораторного стенда «УМК») создать программу по заданию, предложенному ниже. При этом необходимо:

а) Написать законченную программу на языке «Ассемблер»;

б) Перевести полученную программу в машинный код;

в) Описать последовательность действий по записи и выполнению программы на лабораторном стенде «УМК»;

г) Для каждого этапа выполнения программы записать содержимое используемых: ячеек памяти, регистров, регистра флагов;

д) Предложить возможные альтернативы по решению задачи.

Задание:

Создать программу по нахождению наибольшего числа в массиве чисел, расположенного в памяти начиная с адреса «0BACh» по адрес «0A00h». Найденное максимальное число сохранить в памяти по адресу «0BFFh».

ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ №13

1. Методы повышения эффективности обмена по шинам в микропроцессорной системе. Протокол с расщеплением транзакций, увеличение полосы пропускания, ускорение транзакций, повышение эффективности с множеством ведущих.
2. Аппаратные средства микроконтроллеров. Особенности режимов энергопотребления. Тактовые генераторы.

3. Задача.

Пользуясь системой команд микропроцессора КР580ИК80 (лабораторного стенда «УМК») создать программу по заданию, предложенному ниже. При этом необходимо:

а) Написать законченную программу на языке «Ассемблер»;

б) Перевести полученную программу в машинный код;

в) Описать последовательность действий по записи и выполнению программы на лабораторном стенде «УМК»;

г) Для каждого этапа выполнения программы записать содержимое используемых: ячеек памяти, регистров, регистра флагов;

д) Предложить возможные альтернативы по решению задачи.

Задание:

Создать программу по нахождению наименьшего числа в массиве чисел, расположенного в памяти начиная с адреса «0A11h» по адрес «0B00h». Найденное минимальное число сохранить в памяти по адресу «0BFFh».

ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ №14

1. Микропроцессор – основной принцип работы.

2. Аппаратные средства обеспечения надежной работы микроконтроллеров. Схема формирования сигнала сброса, блок детектирования, сторожевой таймер.

3. Задача.

Пользуясь системой команд микропроцессора КР580ИК80 (лабораторного стенда «УМК») создать программу по заданию, предложенному ниже. При этом необходимо:

а) Написать законченную программу на языке «Ассемблер»;

б) Перевести полученную программу в машинный код;

в) Описать последовательность действий по записи и выполнению программы на лабораторном стенде «УМК»;

г) Для каждого этапа выполнения программы записать содержимое используемых: ячеек памяти, регистров, регистра флагов;

д) Предложить возможные альтернативы по решению задачи.

Задание:

Составить программу по вычислению формулы: $a4 - a1 \times a2 - a3$. Число «$a1$» расположено в памяти по адресу «0FFB», число «$a2$» считывается с порта «01h» «D», число «$a3$» расположено в регистре «E». Результат – число «$a4$» переслать в порт «02h»».

ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ №15

1. Микропроцессор – функциональная структура.

2. Дополнительные модули микроконтроллера. Модули последовательного и параллельного вода/вывода, АЦП и ЦАП.
3. Задача.
Пользуясь системой команд микропроцессора КР580ИК80 (лабораторного стенда «УМК») создать программу по заданию, предложенному ниже. При этом необходимо:

а) Написать законченную программу на языке «Ассемблер»;
б) Перевести полученную программу в машинный код;
в) Описать последовательность действий по записи и выполнению программы на лабораторном стенде «УМК»;
г) Для каждого этапа выполнения программы записать содержимое используемых: ячеек памяти, регистров, регистра флагов;
д) Предложить возможные альтернативы по решению задачи.

Задание:

Создать программу по сортировке чисел считываемых с порта «03h». Общее количество обрабатываемых чисел тридцать. Если число больше 21h, то его необходимо разместить в массиве, расположенном начиная с адреса «0A00h». Если число меньше 21h то его необходимо разместить в массиве памяти начиная с адреса «0B00h». Если среди чисел найдется равное 21h, то программу необходимо остановить, а порядковый номер данного числа сохранить в регистре А.

ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ №16

1. Память в микропроцессорной системе.
2. Проектирование цифровых систем на основе микроконтроллеров. Основные этапы.
3. Задача.
Пользуясь системой команд микропроцессора КР580ИК80 (лабораторного стенда «УМК») создать программу по заданию, предложенному ниже. При этом необходимо:

а) Написать законченную программу на языке «Ассемблер»;
б) Перевести полученную программу в машинный код;
в) Описать последовательность действий по записи и выполнению программы на лабораторном стенде «УМК»;
г) Для каждого этапа выполнения программы записать содержимое используемых: ячеек памяти, регистров, регистра флагов;
д) Предложить возможные альтернативы по решению задачи.

Задание:

Написать программу по передаче 100 чисел, расположенных в памяти начиная с адреса «0B00h», в массив расположенный начиная с адреса «0F00». При этом необходимо провести подсчет нулевых чисел, общее количество которых, необходимо сохранить в регистре С.

ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ №17

1. Микропроцессор – функциональная структура.
2. Проектирование цифровых систем на основе микроконтроллеров. Разработка и отладка аппаратных средств.
3. Задача.

Пользуясь системой команд микропроцессора КР580ИК80 (лабораторного стенда «УМК») создать программу по заданию, предложенному ниже. При этом необходимо:

а) Написать законченную программу на языке «Ассемблер»;

б) Перевести полученную программу в машинный код;

в) Описать последовательность действий по записи и выполнению программы на лабораторном стенде «УМК»;

г) Для каждого этапа выполнения программы записать содержимое используемых: ячеек памяти, регистров, регистра флагов;

д) Предложить возможные альтернативы по решению задачи.

Задание:

Написать программу по анализу массива чисел, расположенных в памяти начиная с адреса «0B00». Общее количество обрабатываемых чисел равно 50. Количество положительных чисел сохранить по адресу «0F00», количество четных по адресу «0AA0», количество нулевых передать в порт 01h.

ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ №18

1. Память в микропроцессорной системе.
2. Проектирование цифровых систем на основе микроконтроллеров. Разработка и отладка программных средств. Заключительные этапы проектирования, совместная отладка.
3. Задача.

Пользуясь системой команд микропроцессора КР580ИК80 (лабораторного стенда «УМК») создать программу по заданию, предложенному ниже. При этом необходимо:

а) Написать законченную программу на языке «Ассемблер»;

б) Перевести полученную программу в машинный код;

в) Описать последовательность действий по записи и выполнению программы на лабораторном стенде «УМК»;

г) Для каждого этапа выполнения программы записать содержимое используемых: ячеек памяти, регистров, регистра флагов;

д) Предложить возможные альтернативы по решению задачи.

Задание:

Написать программу по обнаружению срабатывания любых 3 из 7 подключенных к порту 01h дискретных датчика. Посчитать общее количество таких срабатываний за 10 минут. Считать, что каждый такт команды микропроцес-

сор выполняет за 0,01 сек. Полученное количество срабатываний передать в порт 01h.

11.1.2 СЕМЕСТР 8

ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ №1

1. Состав и назначение семейств PIC и AVR контроллеров.
2. Микроконтроллеры PIC и AVR: система команд – перечень и формат команд.
3. Разработка устройств на базе однокристальных микроконтроллеров для решения конкретных задач (стадии, этапы, особенности, пример).

ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ №2

1. Микроконтроллеры отдельных подгрупп семейств PIC и AVR, особенности и отличия.
2. Микроконтроллеры PIC и AVR: система команд – команды работы с байтами и битами.
3. Разработка устройств на базе однокристальных микроконтроллеров для решения конкретных задач (стадии, этапы, особенности, пример).

ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ №3

1. Основные технические характеристики PIC и AVR.
2. Микроконтроллеры PIC и AVR: система команд – команды управления и работы с константами.
3. Разработка устройств на базе однокристальных микроконтроллеров для решения конкретных задач (стадии, этапы, особенности, пример).

ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ №4

1. Особенности архитектуры базовых моделей PIC и AVR.
2. Микроконтроллеры PIC и AVR: особенности программирования и отладки.
3. Разработка устройств на базе однокристальных микроконтроллеров для решения конкретных задач (стадии, этапы, особенности, пример).

ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ №5

1. Принципы работы микроконтроллеров PIC и AVR: временная диаграмма тактирования и циклов выполнения программы, организация памяти программ и стека.
2. Микроконтроллеры PIC и AVR: разработка программного кода – различные ассемблеры, общие сведения, режимы работы по умолчанию.
3. Разработка устройств на базе однокристальных микроконтроллеров для решения конкретных задач (стадии, этапы, особенности, пример).

ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ №6

1. Микроконтроллеры PIC и AVR: организация памяти данных.
2. Микроконтроллеры PIC и AVR: разработка программного кода – метки, мнемоники, операнды комментарии, расширения файлов.
3. Разработка устройств на базе однокристальных микроконтроллеров для решения конкретных задач (стадии, этапы, особенности, пример).

ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ №7

1. Микроконтроллеры PIC и AVR: регистры специального назначения.
2. Микроконтроллеры PIC и AVR: разработка программного кода – директивы языка.
3. Разработка устройств на базе однокристальных микроконтроллеров для решения конкретных задач (стадии, этапы, особенности, пример).

ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ №8

1. Микроконтроллеры PIC и AVR: особенности выполнения команд (выборка команд, прямая и косвенная адресации).
2. Микроконтроллеры PIC и AVR: разработка программного обеспечения – компоновщики; менеджеры библиотек; симуляторы.
3. Разработка устройств на базе однокристальных микроконтроллеров для решения конкретных задач (стадии, этапы, особенности, пример).

ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ №9

1. Микроконтроллеры PIC: организация работы с внешними устройствами – порты ввода/вывода.
2. Микроконтроллеры PIC и AVR: простейший микроконтроллерный макет – архитектура, схема, принцип организации и работы.

3. Разработка устройств на базе однокристальных микроконтроллеров для решения конкретных задач (стадии, этапы, особенности, пример).

ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ №10

1. Микроконтроллеры AVR: организация работы с внешними устройствами – порты ввода/вывода.
2. Микроконтроллеры PIC и AVR: создание завершенных программ – опрос состояния кнопки и вывод его на индикатор (светодиод).
3. Разработка устройств на базе однокристальных микроконтроллеров для решения конкретных задач (стадии, этапы, особенности, пример).

ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ №11

1. Микроконтроллеры PIC и AVR: модуль таймера и регистр таймера.
2. Микроконтроллеры PIC и AVR: создание завершенных программ – использование семисегментного индикатора для контроля за состоянием тумблеров.
3. Разработка устройств на базе однокристальных микроконтроллеров для решения конкретных задач (стадии, этапы, особенности, пример).

ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ №12

1. Микроконтроллеры PIC и AVR: память данных в РПЗУ (EEPROM).
2. Микроконтроллеры PIC и AVR: создание завершенных программ – программные методы формирования задержки.
3. Разработка устройств на базе однокристальных микроконтроллеров для решения конкретных задач (стадии, этапы, особенности, пример).

ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ №13

1. Микроконтроллеры PIC и AVR: организация прерываний.
2. Микроконтроллеры PIC и AVR: создание завершенных программ – инициирование звуковых сигналов.
3. Разработка устройств на базе однокристальных микроконтроллеров для решения конкретных задач (стадии, этапы, особенности, пример).

ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ №14

1. Микроконтроллеры PIC и AVR: специальные функции.

2. Создание завершенных программ: подавление дребезга контактов и подсчет количества нажатий на кнопку.
3. Микроконтроллеры PIC и AVR: создание завершенных программ – подавление дребезга контактов и подсчет количества нажатий на кнопку.

11.2. ЗАЧЕТНЫЕ БИЛЕТЫ

ЗАЧЕТНЫЙ БИЛЕТ №1

1. Предпосылки создания систем малой автоматизации.
2. Анализ и выбор микросхем таймеров реального времени.

ЗАЧЕТНЫЙ БИЛЕТ №2

1. Распределенные системы управления в системах малой автоматизации.
2. Микросхемы дополнительной памяти для технологических контроллеров.

ЗАЧЕТНЫЙ БИЛЕТ №3

1. Локальные вычислительные сети в системах малой автоматизации.
2. Схемы включения таймеров реального времени.

ЗАЧЕТНЫЙ БИЛЕТ №4

1. Основные понятия и определения систем малой автоматизации.
2. Устройства ввода-вывода и расширения.

ЗАЧЕТНЫЙ БИЛЕТ №5

1. Командные и информационные сети – основные определения и понятия.
2. Аналого-цифровые преобразователи.

ЗАЧЕТНЫЙ БИЛЕТ №6

1. Диспетчеры периферийных станций.
2. Цифро-аналоговые преобразователи.

ЗАЧЕТНЫЙ БИЛЕТ №7

1. Формат фреймов и общий алгоритм работы.
2. Основные критерии выбора конкретной модели микроконтроллера.

ЗАЧЕТНЫЙ БИЛЕТ №8

1. Предпосылки создания универсальных технологических контроллеров.
2. Разработки фирмы Atmel (x-51 контроллеры).

ЗАЧЕТНЫЙ БИЛЕТ №9

1. Основные понятия и тенденции развития универсальных технологических контроллеров.
2. Разработки фирмы MAXIM (x-51 контроллеры).

ЗАЧЕТНЫЙ БИЛЕТ №10

1. Общие технологические требования к главному микроконтроллерному модулю.
2. Разработки фирмы Cygnal (x-51 контроллеры).

ЗАЧЕТНЫЙ БИЛЕТ №11

1. Обобщенная функциональная схема центрального модуля.
2. Отдельные вопросы обеспечения пиковой производительности микроконтроллеров.

ЗАЧЕТНЫЙ БИЛЕТ №12

1. Основные понятия о супервизорах питания и таймерах.
2. Принципиальная схема основных узлов макетов микропроцессорных систем управления на базе x-51 совместимых микроконтроллеров.

ЗАЧЕТНЫЙ БИЛЕТ №13

1. Критерии выбора супервизора питания.
2. Подсистема аналогового ввода-вывода макетов микропроцессорных систем управления на базе x-51 совместимых микроконтроллеров.

ЗАЧЕТНЫЙ БИЛЕТ №14

1. Предварительный анализ и выбор микросхем супервизоров.
2. Подсистема интерфейсов макетов микропроцессорных систем управления на базе x-51 совместимых микроконтроллеров.

ЗАЧЕТНЫЙ БИЛЕТ №15

1. Схемы включения микросхем супервизоров.
2. Универсальный технологический контроллер на базе микроконтроллера *C8051F020.*

ЗАЧЕТНЫЙ БИЛЕТ №16

1. Заключительный этап выбора микросхем супервизоров.
2. Варианты слотового исполнения системы сбора и обработки данных.

ЗАЧЕТНЫЙ БИЛЕТ №17

1. Функциональные характеристики таймеров реального времени.
2. Специализированные контроллеры-фотодатчики.

ЗАЧЕТНЫЙ БИЛЕТ №18

1. Критерии выбора таймеров реального времени.
2. Специализированный технологический контроллер для работы в составе информационной сети.

12. КАРТА ОБЕСПЕЧЕННОСТИ ДИСЦИПЛИНЫ КАДРАМИ ПРОФЕССОРСКО-ПРЕПОДАВАТЕЛЬСКОГО СОСТАВА

Лекционный курс – к.т.н., доцент, Теличенко Денис Алексеевич;

Практические занятия – к.т.н., доцент, Теличенко Денис Алексеевич;

– к.т.н., ст.преп. Безруков Николай Сергеевич;

Лабораторные занятия – к.т.н., доцент, Теличенко Денис Алексеевич;

– к.т.н., ст.преп. Безруков Николай Сергеевич.