**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**СЫКТЫВКАРСКИЙ ЛЕСНОЙ ИНСТИТУТ –
ФИЛИАЛ ГОСУДАРСТВЕННОГО ОБРАЗОВАТЕЛЬНОГО УЧРЕЖДЕНИЯ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКАЯ ГОСУДАРСТВЕННАЯ
ЛЕСОТЕХНИЧЕСКАЯ АКАДЕМИЯ ИМЕНИ С. М. КИРОВА»**

**КАФЕДРА АВТОМАТИЗАЦИИ ТЕХНОЛОГИЧЕСКИХ ПРОЦЕССОВ И ПРОИЗВОДСТВ**

# МИКРОПРОЦЕССОРНЫЕ СИСТЕМЫ УПРАВЛЕНИЯ

**Методическое пособие
для студентов специальности 220301
«Автоматизация технологических процессов и производств»
и направления бакалавриата 220200 «Автоматизация и управление»
всех форм обучения**

***Самостоятельное учебное электронное издание***

**СЫКТЫВКАР 2010**

УДК 621.38
ББК 32.973.2
М59

Утверждено к опубликованию в электронном виде методическим советом лесотранспортного факультета Сыктывкарского лесного института 15 сентября 2010 г. (протокол № 1).

**Составители:**
**В. И. Семёновых**, кандидат технических наук, доцент;
**Е. Ю. Сундуков**, кандидат экономических наук

**Рецензент:**
**В. М. Сбоев**, кандидат технических наук (Вятский государственный университет)

**Ответственный редактор:**
**В. И. Чудов**, кандидат технических наук, профессор

М59 МИКРОПРОЦЕССОРНЫЕ СИСТЕМЫ УПРАВЛЕНИЯ [Электронный ресурс] : методическое пособие для студентов специальности 220301 «Автоматизация технологических процессов и производств» и направления бакалавриата 220200 «Автоматизация и управление» всех форм обучения : самост. учеб. электрон. изд. / Сыкт. лесн. ин-т ; сост. В. И. Семёновых, Е. Ю. Сундуков. – Электрон. дан. (1 файл в формате pdf: 0,4 Мб). – Сыктывкар : СЛИ, 2010. – Режим доступа: http://lib.sfi.komi.com. – Загл. с экрана.

УДК 621.38
ББК 32.973.2

Рассмотрены типы микроконтроллеров, архитектура процессоров с анализом их достоинств и недостатков, а также современные типы памяти, используемые при программировании микроконтроллеров непосредственно с ПК. Детально описан микроконтроллер МК48 с ядром i51 относительно его архитектуры, системы команд и способов адресации, а также система моделирования SCM 1.38, позволяющая программировать микроконтроллер в виртуальной среде, отлаживать прикладные программы с последующим их выполнением на учебном микропроцессорном комплексе УМПК-48. Проанализированы технико-экономические аспекты разработки микропроцессорных систем управления, связанные с процедурой выбора по множеству критериев наиболее подходящего микроконтроллера для решения прикладных задач. Приведены задания контрольной работы и методические указания по их выполнению.

Предназначено для студентов специальности 220301 «Автоматизация технологических процессов и производств» и направления бакалавриата 220200 «Автоматизация и управление» всех форм обучения.

* * *

*Самостоятельное учебное электронное издание*

Составители: Семёновых Владимир Иванович, Сундуков Евгений Юрьевич

**МИКРОПРОЦЕССОРНЫЕ СИСТЕМЫ УПРАВЛЕНИЯ**

**Методическое пособие для студентов специальности 220301 «Автоматизация технологических процессов и производств» и направления бакалавриата 220200 «Автоматизация и управление» всех форм обучения**

Электронный формат – pdf
Разрешено к публикации 26.10.10. Объем 4,8 уч.-изд. л.; 0,4 Мб

Сыктывкарский лесной институт – филиал государственного образовательного учреждения
высшего профессионального образования
«Санкт-Петербургская государственная лесотехническая академия имени С. М. Кирова» (СЛИ)
167982, г. Сыктывкар, ул. Ленина, 39
institut@sfi.komi.com, www.sli.komi.com

Редакционно-издательский отдел СЛИ. Заказ № 55.

# ОГЛАВЛЕНИЕ

# ВВЕДЕНИЕ

Одной из характерных особенностей нынешнего этапа научно-технического прогресса является все более широкое применение микроэлектроники в различных отраслях народного хозяйства. Роль микроэлектроники в развитии общественного производства определяется ее практически неограниченными возможностями в решении различных задач во всех областях народного хозяйства, глубоким влиянием на культуру и быт современного человека.

Особое внимание в настоящее время уделяется внедрению микропроцессоров, обеспечивающих решение задач автоматизации управления механизмами, приборами и аппаратурой. Адаптация микропроцессора к особенностям конкретной задачи осуществляется в основном путем разработки соответствующего программного обеспечения, заносимого затем в память программ. Аппаратная адаптация в большинстве случаев осуществляется путем подключения необходимых интегральных схем обрамления и организации ввода-вывода, соответствующих решаемой задаче.

В микропроцессорной технике выделился самостоятельный класс больших интегральных схем (БИС) – однокристальные микроЭВМ (ОМЭВМ), которые предназначены для «интеллектуализации» оборудования различного назначения. Архитектура однокристальных микроЭВМ – результат эволюции архитектуры микропроцессоров и микропроцессорных систем, обусловленной стремлением существенно снизить их аппаратные затраты и стоимость. Как правило, эти цели достигаются как путем повышения уровня интеграции БИС, так и за счет поиска компромисса между стоимостью, аппаратными затратами и техническими характеристиками ОМЭВМ.

ОМЭВМ представляют собой приборы, конструктивно выполненные в виде одной БИС и включающие в себя все устройства, необходимые для реализации цифровой системы управления минимальной конфигурации: процессор, запоминающее устройство данных, запоминающее устройство команд, внутренний генератор тактовых сигналов, а также программируемые интегральные схемы для связи с внешней средой. Использование ОМЭВМ в системах управления обеспечивает достижение исключительно высоких показателей эффективности при столь низкой стоимости (во многих применениях система может состоять только из одной БИС ОМЭВМ), что им, видимо, нет в ближайшем времени альтернативной элементной базы для построения управляющих и/или регулирующих систем. В настоящее время более двух третей мирового рынка микропроцессорных средств составляют именно БИС ОМЭВМ. В некоторых публикациях однокристальную микроЭВМ (ОМЭВМ) называют «микроконтроллер». Обосновывается это тем обстоятельством, что такие микросхемы имеют незначительные емкости памяти, физическое и логическое разделение памяти программ (ПЗУ) и памяти данных (ОЗУ), упрощенную и ориентированную на задачи управления систему команд, примитивные методы адресации команд и данных. Специфическая организация ввода-вывода информации предопределяет область их применения в качестве специализированных вычислителей, включенных в контур управления объектом или процессом. Структурная организация, набор команд и аппаратно-программные средства ввода-вывода информации этих микросхем лучше всего приспособлены для решения задач управления и регулирования в приборах, устройствах и системах автоматики, а не для решения задач обработки данных. Наименование «микроконтроллер» в описании используется в аббре-

виатуре МК для обозначения семейства ОМЭВМ, объединенных рядом общих признаков, например, разрядностью, системой команд, набором функциональных блоков и т. д.

В настоящее время очень распространенными являются 8-разрядные ОМЭВМ семейств МК48, МК51 и микросхемы ЭКР1847ВГ6. Каждая из моделей ОМЭВМ этих семейств имеет соответствующий аналог ОМЭВМ, входящих в состав широко известных в мире семейств MCS-48, MCS-51 и UPI-42 фирмы Intel (США). В предлагаемом учебном пособии рассматривается наиболее простой из микроконтроллеров с ядром i51 – МК48, что обусловлено рядом следующих обстоятельств:

– для данного микроконтроллера имеется бесплатно распространяемый симулятор SCM 1.38, позволяющий в виртуальной среде автоматизировать процесс разработки прикладных программ пользователя;

– наличием промышленно выпускаемого учебного микропроцессорного комплекса УМПК-48, с помощью которого можно на физическом уровне изучать как сам микроконтроллер МК48, так и реализовывать аппаратно-программные средства управления различного рода объектами;

– наличием большого количества литературы (как отечественной, так и переводной) для микроконтроллеров с ядром i51.

Основная цель учебного пособия – помочь студентам в выполнении контрольной работы и подготовке к зачету и экзамену по данной дисциплине. В результате изучения дисциплины студент должен знать особенности архитектур микропроцессорных устройств, организацию ввода/вывода информации, организацию связи в микропроцессорных системах, а также выполнять проектные работы по разработке устройств управления в технических системах с применением микропроцессорной техники.

Настоящее пособие предназначено для студентов специальности «Автоматизация технологических процессов и производств», а также может быть рекомендовано для студентов технических специальностей, связанных с разработкой микропроцессорных систем управления.

# 1. МИКРОКОНТРОЛЛЕР КАК ОСНОВНОЙ ЭЛЕМЕНТ АВТОМАТИЗАЦИИ ПРОИЗВОДСТВА НА НИЖНЕМ УРОВНЕ ИЕРАРХИИ УПРАВЛЕНИЯ

Развитие современного производства влечет за собой освобождение человека от ручного труда. С автоматизацией производства происходит передача машинам функций управления.

Технический базис современного производства поднимается на качественно новую ступень и освобождается от всех ограничений, которые связаны с естественными возможностями рабочей силы. В результате обеспечивается поистине безграничный рост производительности труда. Автоматизация коренным образом меняет место человека в производственном процессе. Труд из непосредственного в процессе производства превращается в функцию контроля и регулирования.

Сложность, многообразие, взаимосвязанность параметров современных технологических процессов технически и экономически оправдывают передачу части функций управления и общего контроля человеку-оператору, т. е. создание человеко-машинных систем управления, получивших название *автоматизированных систем управления технологическими процессами* (АСУТП).

Основными целями автоматизации технологических процессов в промышленности являются:

– увеличение производительности технологического оборудования с одновременным уменьшением числа рабочих при заданных объеме, качестве и номенклатуре продукции;

– экономия производственных ресурсов;

– максимальное снижение затрат живого труда при заданных других экономических показателях производства;

– увеличение объема производства продукции за счет роста производительности технологического оборудования без увеличения затрат живого труда;

– улучшение качества продукции при заданных затратах живого труда, объеме и номенклатуре;

– достижение оптимальной загрузки технологического оборудования и оптимальное ведение технологического процесса.

Автоматизация технологических процессов является одним из решающих факторов повышения производительности и улучшений условий труда. Все существующие и строящиеся промышленные объекты в той или иной степени оснащаются средствами автоматизации.

АСУТП – это система, реализованная на базе высокоэффективной вычислительной и управляющей технике, которая обеспечивает автоматизированное (автоматическое) управление технологическим объектом управления с использованием централизованно обработанной информации по заданным технологическим и технико-экономическим критериям, определяющим качественные и количественные результаты производства продукта, а также подготовку информации для решения задач организационно-экономического характера [1].

С появлением управляющих ЭВМ (УВМ) измерительная, регулирующая и управляющая техника получила очень мощное средство автоматизации. В настоящее время в системах автоматики с внедрением ОМЭВМ происходит переход от

централизованной к распределенной обработке информации. Как правило, АСУТП имеет иерархический принцип построения. На верхнем уровне находится организационно-экономическая система управления производством АСУП, имеющая в контуре управления среднюю или мини-ЭВМ, выбор которой зависит от объема переработки информации. На среднем уровне АСУТП находится УВМ (средняя, мини- или микроЭВМ), которая ведет управление полным технологическим процессом, рядом агрегатов, объектов, группой станков и т. п. через локальные системы автоматики. На этом уровне вырабатываются управляющие программы на основе оптимизации техпроцесса, адаптации к внешним факторам и другим параметрам для микропроцессорных средств (МПС) нижнего (локального) уровня. Локальные МПС встраиваются в сам объект автоматизации, имеют минимальные связи с датчиками и управляющими органами объекта, но могут находиться на достаточном удалении от УВМ среднего уровня. С нижнего уровня на средний ведется передача текущей уже обработанной информации или по запросу – текущая необработанная информация о состоянии объекта управления. Такая иерархическая структура, иногда ее называют треугольником или пирамидой иерархии, позволяет разгрузить верхний и средний уровни АСУТП от сбора, обработки и выдачи информации для управления объектом и возложить эту работу на локальные уровни. При выходе из строя УВМ среднего уровня локальная может работать по ранее заложенной программе или программе, задаваемой с пульта управления вручную.

Микроконтроллеры, используемые на нижнем уровне АСУТП, выполняют функции интерпретации данных, поступающих от датчиков, определяющих параметры объекта управления, обеспечивают связь между различными устройствами системы и передают данные другим устройствам.

Микроконтроллеры отличаются не только архитектурой и характеристиками, но и особенностями функционирования и реализации. Большинство микроконтроллеров представляют собой процессор, интегрированный с памятью и устройствами ввода-вывода данных. Практически все микроконтроллеры входят в состав определенных семейств, члены которых отличаются составом и характеристиками периферийных устройств, реализованных на кристалле.

Огромная номенклатура выпускаемых промышленностью микроконтроллеров, отличающихся различным сочетанием основных параметров, значительно усложняет процесс выбора прибора, наиболее подходящего для данного применения.

# 2. ТИПЫ МИКРОКОНТРОЛЛЕРОВ

Существует огромное количество разнообразных приборов этого класса. Все эти приборы можно разделить на следующие основные типы:

- встраиваемые 8-разрядные микроконтроллеры;
- 16- и 32-разрядные микроконтроллеры;
- цифровые сигнальные процессоры.

## 2.1. Встраиваемые микроконтроллеры

Промышленностью выпускается очень широкая номенклатура встраиваемых микроконтроллеров. В этих микроконтроллерах все необходимые ресурсы (память, устройства ввода-вывода и т. д.) располагаются на одном кристалле с процессорным ядром. Все, что необходимо сделать, – это подать питание и тактовые сигналы. Встраиваемые микроконтроллеры могут базироваться на существующем микропроцессорном ядре или на процессоре, разработанном специально для данного микроконтроллера. Это означает, что существует большое разнообразие функционирования даже среди устройств, выполняющих одинаковые задачи.

Основное назначение встраиваемых микроконтроллеров – обеспечить с помощью недорогих средств гибкое (программируемое) управление объектами и связь с внешними устройствами. Эти микроконтроллеры не предназначены для реализации комплекса сложных функций, но они способны обеспечить эффективное управление во многих областях применения. Недорогими считаются микроконтроллеры, стоимость которых составляет от 1 до 20 долларов за штуку (цена зависит от технических характеристик, количества выводов корпуса, объема закупок).

Встраиваемые микроконтроллеры содержат значительное число вспомогательных устройств, благодаря чему обеспечивается их включение в реализуемую систему с использованием минимального количества дополнительных компонентов. В состав этих микроконтроллеров обычно входят:

- схема начального запуска процессора (Reset);
- генератор тактовых импульсов;
- центральный процессор;
- память программ (ROM) и программный интерфейс;
- память данных (RAM);
- средства ввода-вывода данных;
- таймеры, фиксирующие число командных циклов.

Общая структура микроконтроллера показана на рис. 2.1. Эта структура дает представление о том, как микроконтроллер связывается с внешним миром.

Более сложные встраиваемые микроконтроллеры могут дополнительно реализовать следующие возможности:

- встроенный монитор/отладчик программ;
- внутренние средства программирования памяти программ;
- обработка прерываний от различных источников;
- аналоговый ввод-вывод;

– последовательный ввод-вывод (синхронный и асинхронный);
– параллельный ввод-вывод (включая интерфейс с компьютером);
– подключение внешней памяти.

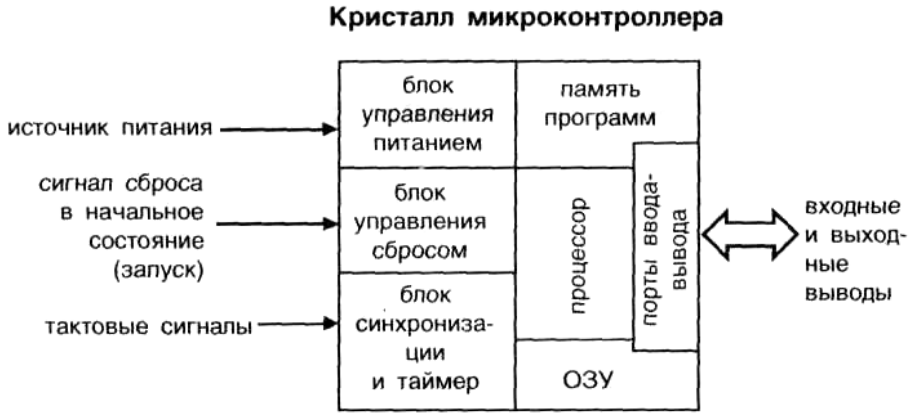



**Рис. 2.1.** Структура микроконтроллера

Все эти возможности значительно увеличивают гибкость применения микроконтроллеров и делают более простым процесс разработки систем на их основе. Следует заметить, что для реализации этих возможностей в большинстве случаев требуется расширение функций внешних выводов.

Типичные значения максимальной частоты тактовых сигналов составляют для различных микроконтроллеров от 10 до 20 МГц. Главным фактором, ограничивающим их скорость, является время доступа к памяти, применяемой в микроконтроллерах. Однако для типичных применений это ограничение не является существенным.

## 2.2. Микроконтроллеры с внешней памятью

Некоторые микроконтроллеры (особенно 16- и 32-разрядные) используют только внешнюю память, которая включает в себя как память программ (ROM), так и некоторый объем памяти данных (RAM), требуемый для данного применения. Структура микроконтроллера с внешней памятью показана на рис. 2.2.

Классическим примером такого микроконтроллера является Intel 80188. По существу он представляет собой микропроцессор 8088, который использовался в компьютерах IBM PC, интегрированный на общем кристалле с дополнительными схемами, реализующими ряд стандартных функций, таких как прерывания и прямой доступ к памяти (ПДП). Цель создания 80188 состояла в том, чтобы объединить в одном корпусе все устройства, необходимые инженеру для реализации систем, в которых могут использоваться функциональные возможности и программное обеспечение микропроцессора 8088.

Аналогичные цели достигаются при использовании микроконтроллера 80186, который имеет 16-разрядную внешнюю шину (80188 имеет 8-разрядную внешнюю шину) и представляет собой 16-разрядный процессор 8086, интегрированный на общем кристалле с дополнительными периферийными схемами (такими же, как в 80188). Так же как микропроцессор 8088 является упрощенной (8-разрядная внеш-

няя и 16-разрядная внутренняя шина) версией 8086 (16-разрядные внешняя и внутренняя шины), так и микроконтроллер 80188 является упрощенной версией 80186.

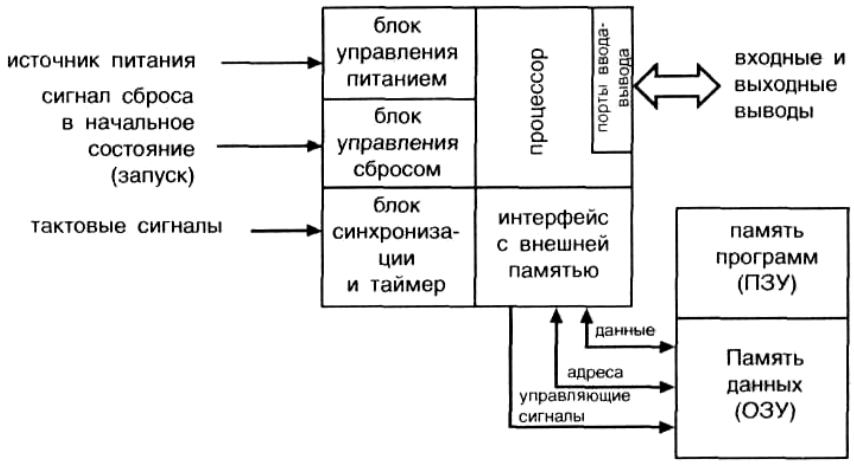



**Рис. 2.2.** Блок-схема микроконтроллера с внешней памятью

Микроконтроллеры с внешней памятью предназначены для других применений, нежели встраиваемые микроконтроллеры. Эти применения обычно требуют большого объема памяти (RAM) и небольшого количества устройств (портов) ввода-вывода. Для микроконтроллеров с внешней памятью наиболее подходящими являются приложения, в которых критическим ресурсом является память, а не число логических входов-выходов общего назначения, тогда как для встраиваемых микроконтроллеров имеет место противоположная ситуация. Типичным примером применения для микроконтроллера с внешней памятью является контроллер жесткого диска с буферной кэш-памятью, который обеспечивает промежуточное хранение и распределение больших объемов данных. Внешняя память дает возможность такому микроконтроллеру работать с более высокой скоростью, чем встраиваемый микроконтроллер.

## 2.3. Цифровые сигнальные процессоры

Цифровые сигнальные процессоры (DSP) – относительно новая категория процессоров. Назначение DSP состоит в том, чтобы получать текущие данные от аналоговой системы и формировать соответствующий отклик. DSP и их арифметико-логическое устройство, которое является аппаратным средством для выполнения вычислений, работают с очень высокой скоростью, что позволяет осуществлять обработку данных в реальном масштабе времени. DSP часто используются в активных шумоподавляющих микрофонах, которые устанавливаются в самолетах (второй микрофон обеспечивает сигнал окружающего шума, который вычитается из сигнала первого микрофона, позволяя таким образом подавить шум и оставить только голос) или для подавления раздвоения изображения в телевизионных сигналах.

В разнообразных DSP можно найти особенности, присущие как встраиваемым микроконтроллерам, так и микроконтроллерам с внешней памятью. DSP не предназначены для автономного применения. Обычно они входят в состав систем, используясь в качестве устройств управления внешним оборудованием, а также для обработки входных сигналов и формирования соответствующего отклика.

# 3. АРХИТЕКТУРА ПРОЦЕССОРОВ

Существует несколько архитектур, на базе которых построены микроконтроллеры. Это CISC (Complex Instruction Set Computers – компьютеры со сложной системой команд), RISC (Reduced Instruct Set Computers – компьютеры с сокращенной системой команд), Гарвардская, Принстонская.

Сложно сказать, какая из архитектур лучше – CISC или RISC, Гарвардская или Принстонская. Попытаемся объяснить различия между этими архитектурами и показать, какое отношение они имеют к микроконтроллерам.

## 3.1. CISC и RISC

В настоящее время существует множество RISC-процессоров, т. к. сложилось мнение, что RISC быстрее, чем CISC процессоры. Такое мнение не совсем верно. Имеется много процессоров, называемых RISC, но на самом деле относящихся к CISC. Более того, в некоторых приложениях CISC-процессоры выполняют программный код быстрее, чем это делают RISC-процессоры, или решают такие задачи, которые RISC-процессоры не могут выполнить.

Истинное различие между RISC и CISC в том, что CISC-процессоры выполняют большой набор команд с развитыми возможностями адресации (непосредственная, индексная и т. д.), давая разработчику возможность выбрать наиболее подходящую команду для выполнения необходимой операции, а в RISC-процессорах набор выполняемых команд сокращен до минимума. При этом разработчик должен комбинировать команды, чтобы реализовать более сложные операции.

Возможность равноправного использования всех регистров процессора называется «ортогональностью» или «симметричностью» процессора. Это обеспечивает дополнительную гибкость при выполнении некоторых операций. Рассмотрим, например, выполнение условных переходов в программе. В CISC-процессорах условный переход обычно реализуется в соответствии с определенным значением бита (флага) в регистре состояния. В RISC-процессорах условный переход может происходить при определенном значении бита, который находится в любом месте памяти. Это значительно упрощает операции с флагами и выполнение программ, использующих эти флаги.

Успех при использовании RISC-процессоров обеспечивается благодаря тому, что их более простые команды требуют для выполнения значительно меньшее число машинных циклов. Таким образом, достигается существенное повышение производительности, что позволяет RISC-процессорам эффективно решать чрезвычайно сложные задачи.

## 3.2. Гарвардская и Принстонская

Много лет назад правительство Соединенных Штатов дало задание Гарвардскому и Принстонскому университетам разработать архитектуру компьютера для военно-морской артиллерии. Принстонский университет разработал компьютер, ко-

торый имел общую память для хранения программ и данных. Такая архитектура компьютеров больше известна как ***архитектура фон Неймана*** по имени научного руководителя этой разработки. Структура компьютера с Принстонской архитектурой представлена на рис. 3.1.

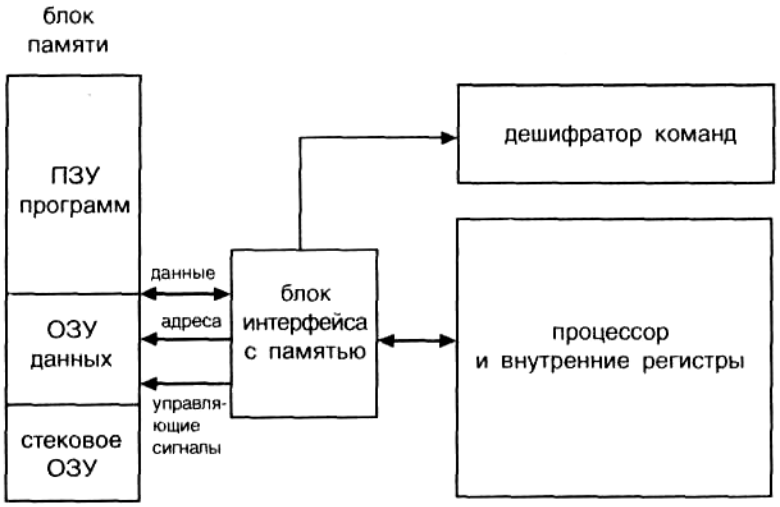



**Рис. 3.1.** Структура компьютера с Принстонской архитектурой

В этой архитектуре блок интерфейса с памятью выполняет арбитраж запросов к памяти, обеспечивая выборку команд, чтение и запись данных, размещаемых в памяти или внутренних регистрах. Может показаться, что блок интерфейса является наиболее узким местом между процессором и памятью, т. к. одновременно с данными требуется выбирать из памяти очередную команду. Однако во многих процессорах с Принстонской архитектурой эта проблема решается путем выборки следующей команды во время выполнения предыдущей. Такая операция называется *предварительной выборкой* («предвыборка»), и она реализуется в большинстве процессоров с такой архитектурой.

Гарвардский университет представил разработку компьютера, в котором для хранения программ, данных и стека использовались отдельные банки памяти. Структура компьютера с Гарвардской архитектурой представлена на рис. 3.2.

Принстонская архитектура выиграла соревнование, т. к. она больше соответствовала уровню технологии того времени. Использование общей памяти оказалось более предпочтительным из-за ненадежности ламповой электроники (это было до широкого распространения транзисторов), при этом возникало меньше отказов.

Гарвардская архитектура почти не использовалась до конца 70-х годов, когда производители микроконтроллеров поняли, что эта архитектура дает преимущества устройствам, которые они разрабатывали.

Основным преимуществом архитектуры фон Неймана является то, что она упрощает устройство микропроцессора, т. к. реализует обращение только к одной общей памяти. Для микропроцессоров самым важным является то, что содержимое ОЗУ (RAM – Random Access Memory) может быть использовано как для хранения данных, так и для хранения программ. В некоторых приложениях программе необходимо иметь доступ к содержимому стека. Все это предоставляет большую гибкость для разработчика программного обеспечения.

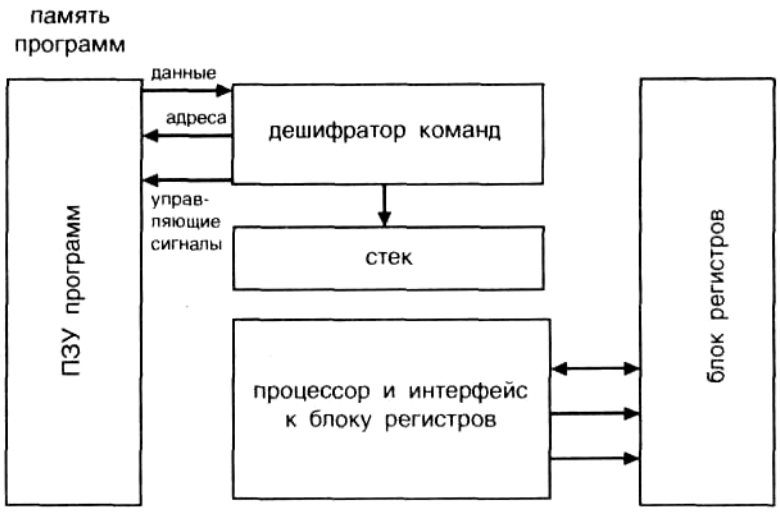



**Рис. 3.2.** Структура компьютера с Гарвардской архитектурой

Гарвардская архитектура выполняет команды за меньшее количество тактов, чем архитектура фон Неймана. Это обусловлено тем, что в Гарвардской архитектуре больше возможностей для реализации параллельных операций. Выборка следующей команды может происходить одновременно с выполнением предыдущей команды, и нет необходимости останавливать процессор на время выборки команды.

Например, если процессору с Принстонской архитектурой необходимо считать байт и поместить его в аккумулятор, то он производит последовательность действий, показанную на рис. 3.3. В первом цикле из памяти выбирается команда; в следующем цикле данные, которые должны быть помещены в аккумулятор, считываются из памяти.

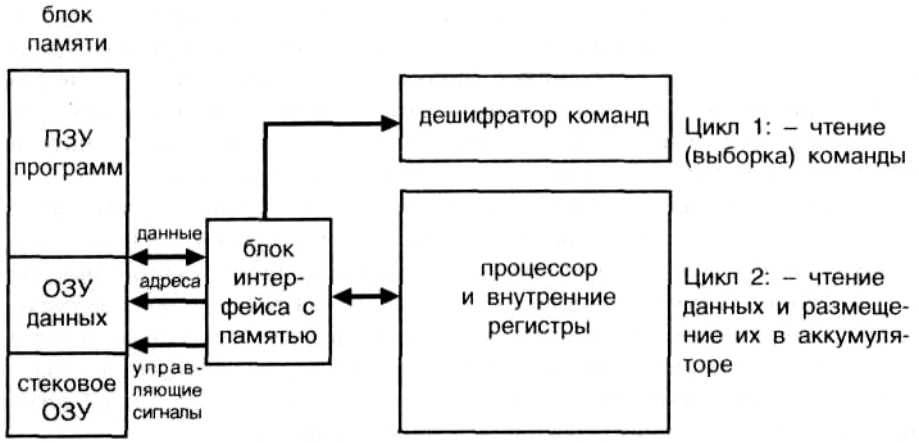



**Рис. 3.3.** Выполнение команды mov Acc, Reg в Принстонской архитектуре

В Гарвардской архитектуре, обеспечивающей более высокую степень параллелизма операций, выполнение текущей операции может совмещаться с выборкой следующей команды (рис. 3.4). Команда также выполняется за два цикла, но выборка очередной команды производится одновременно с выполнением предыдущей. Таким образом, команда выполняется всего за один цикл (во время чтения следующей команды).

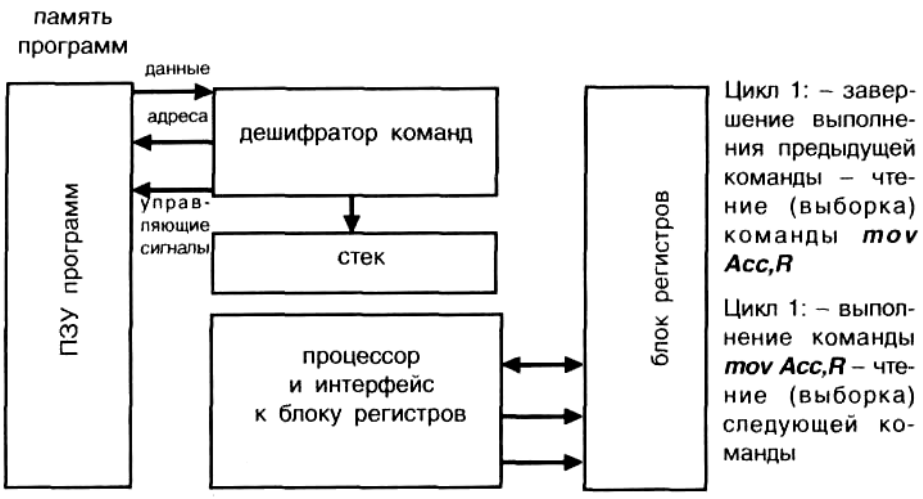



**Рис. 3.4.** Выполнение команды mov Acc, Reg в Гарвардской архитектуре

Этот метод реализации операций («параллелизм») позволяет командам выполняться за одинаковое число тактов, что дает возможность более просто определить время выполнения циклов и критических участков программы. Это обстоятельство является особенно важным при выборе микроконтроллера для приложений, где требуется строгое обеспечение заданного времени выполнения. Например, микроконтроллер PIC фирмы Microchip выполняет любую команду, кроме тех, которые модифицируют содержимое программного счетчика, за четыре такта (один цикл). Это упрощает реализацию критических ко времени процедур по сравнению с микроконтроллером Intel 8051, где для выполнения команд может потребоваться от 16 до 64 тактов. Из-за этого часто не удается подсчитать точное время выполнения программы вручную и приходится применять симуляторы или аппаратные эмуляторы.

Следует отметить, что такие общие способы сравнения производительности не следует использовать для всех процессоров и микроконтроллеров, в которых реализуются эти две архитектуры. Сравнение лучше проводить применительно к конкретному приложению. Различные архитектуры и устройства имеют свои специфические особенности, которые позволяют наилучшим образом реализовать те или иные приложения. В некоторых случаях конкретное приложение может быть выполнено только с использованием определенной архитектуры и специфических особенностей микроконтроллера.

На первый взгляд Гарвардская архитектура – это единственно правильный выбор. Но Гарвардская архитектура является недостаточно гибкой для некоторых программных процедур, которые требуются для реализации ряда приложений.

# 4. ТИПЫ ПАМЯТИ МИКРОКОНТРОЛЛЕРОВ

Можно выделить три основных вида памяти, используемой в микроконтроллерах:

– память программ – представляет собой постоянную память, предназначенную для хранения программного кода и констант. Эта память не изменяет своего содержимого в процессе выполнения программы;

– память данных – предназначена для хранения переменных в ходе выполнения программы;

– регистры микроконтроллера – этот вид памяти включает внутренние регистры процессора и регистры, которые служат для управления периферийными устройствами.

Объем памяти микроконтроллеров удивительно малый, но это не является их существенным недостатком.

## 4.1. Память программ

Для хранения программ обычно служит один из видов постоянной памяти:

– PROM – однократно программируемое ПЗУ;

– EPROM – электрически программируемое ПЗУ с ультрафиолетовым стиранием;

– EEPROM – ПЗУ с электрической записью и стиранием (к этому виду относятся также современные микросхемы Flash-памяти);

– ROM – масочно-программируемое ПЗУ.

Все эти виды памяти являются энергонезависимыми – это означает, что содержимое памяти сохраняется после выключения питания микроконтроллера. Такая память необходима, т. к. микроконтроллер не содержит каких-либо устройств массовой памяти (магнитных дисков), с которых загружается программа в компьютерах. Программа постоянно хранится в микроконтроллере.

В процессе выполнения программа считывается из этой памяти, а блок управления (дешифратор команд) обеспечивает ее декодирование и выполнение необходимых операций. Содержимое памяти программ не может меняться (перепрограммироваться) во время выполнения программы. Поэтому функциональное назначение микроконтроллера не может измениться, пока содержимое его памяти программ не будет стерто (если это возможно) и перепрограммировано (заполнено новыми командами).

Следует обратить внимание, что разрядность микроконтроллера (8, 16 или 32 бит) указывается в соответствии с разрядностью его шины данных. В Гарвардской архитектуре команды могут иметь бóльшую разрядность, чем данные, чтобы дать возможность считывать за один такт целую команду. Например, микроконтроллеры PIC в зависимости от модели используют команды с разрядностью 12, 14 или 16 бит. В микроконтроллерах AVR команда всегда имеет разрядность 16 бит. Однако все эти микроконтроллеры имеют шину данных разрядностью 8 бит.

В устройствах с Принстонской архитектурой разрядность данных обычно определяет разрядность (число линий) используемой шины. В микроконтроллерах Motorola 68HC05 24-разрядная команда размещается в трех 8-разрядных ячейках памяти программ. Для полной выборки такой команды необходимо произвести три цикла считывания этой памяти.

Когда говорится, что устройство является 8-разрядным, – это означает разрядность данных, которые способен обрабатывать микроконтроллер.

**Память ROM** используется тогда, когда программный код заносится в микроконтроллер на этапе его производства. Предварительно программа отлаживается и тестируется, после чего передается фирме-производителю, где программа преобразуется в рисунок маски на стеклянном фотошаблоне. Полученный фотошаблон с маской используется в процессе создания соединений между элементами, из которых состоит память программ. Поэтому такую память часто называют *масочно-программируемой ROM*.

ROM является самым дешевым типом постоянной памяти для массового производства. Однако она имеет ряд существенных недостатков, которые привели к тому, что в последние годы этот тип памяти почти не используется. Основными недостатками являются значительные затраты средств и времени на создание нового комплекта фотошаблонов и их внедрение в производство. Обычно такой процесс занимает около десяти недель и является экономически выгодным при выпуске десятков тысяч приборов. Только при таких объемах производства обеспечивается преимущество ROM по сравнению с EEPROM. Существует также ограничение, связанное с возможностью использования таких микроконтроллеров только в определенной сфере применения, т. к. его программа обеспечивает выполнение жестко фиксированной последовательности операций и не может быть использована для решения каких-либо других задач.

**Память PROM** может быть запрограммирована только один раз. Эта память обычно содержит плавкие перемычки, которые пережигаются во время программирования. В настоящее время такая память используется очень редко.

Электрически программируемая **память EPROM** состоит из ячеек, которые программируются электрическими сигналами и стираются с помощью ультрафиолетового света. Ячейка памяти EPROM представляет собой MOS-транзистор с плавающим затвором, который окружен диоксидом кремния ($SiO_2$). Сток транзистора соединен с «землей», а исток подключен к напряжению питания с помощью резистора. В стертом состоянии (до записи) плавающий затвор не содержит заряда, и MOS-транзистор закрыт. В этом случае на истоке поддерживается высокий потенциал, и при обращении к ячейке считывается логическая единица. Программирование памяти сводится к записи в соответствующие ячейки логических нулей.

Программирование осуществляется путем подачи на управляющий затвор высокого напряжения согласно рис. 4.1. Этого напряжения должно быть достаточно, чтобы обеспечить пробой между управляющим и плавающим затвором, после чего заряд с управляющего затвора переносится на плавающий. MOS-транзистор переключается в открытое состояние, закорачивая исток с землей. В этом случае при обращении к ячейке считывается логический нуль.

Чтобы стереть содержимое ячейки, она освещается ультрафиолетовым светом, который дает заряду на плавающем затворе достаточную энергию, чтобы он мог по-

кинуть затвор. Этот процесс может занимать от нескольких секунд до нескольких минут.

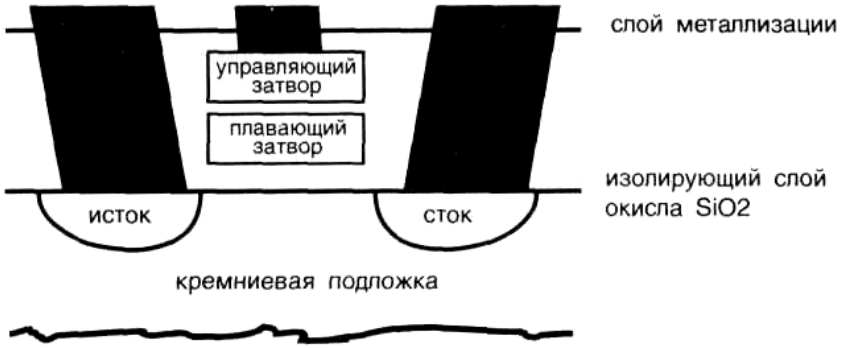



**Рис. 4.1.** Ячейка памяти EPROM

Обычно, микросхемы EPROM производятся в керамическом корпусе с кварцевым окошком для доступа ультрафиолетового света. Такой корпус довольно дорог, что значительно увеличивает стоимость микросхемы. Для уменьшения цены микросхемы EPROM заключают в корпус без окошка (версия EPROM с однократным программированием). Сокращение стоимости при использовании таких корпусов может быть настолько значительным, что эти версии EPROM в настоящее время часто используются вместо масочно-программируемых ROM.

**Память EEPROM** (Electrically Erasable Programmable Memory – электрически стираемая программируемая память) можно считать новым поколением EPROM памяти. В такой памяти ячейка стирается не ультрафиолетовым светом, а путем электрического соединения плавающего затвора с «землей». Использование EEPROM позволяет стирать и программировать микроконтроллер, не снимая его с платы. Таким способом можно периодически обновлять его программное обеспечение.

Память EEPROM более дорогая, чем EPROM (в два раза дороже EPROM с однократным программированием). EEPROM работает немного медленнее, чем EPROM.

Основное преимущество использования памяти EEPROM заключается в возможности ее многократного перепрограммирования без удаления из платы. Это дает огромный выигрыш на начальных этапах разработки систем на базе микроконтроллеров или в процессе их изучения, когда масса времени уходит на многократный поиск причин неработоспособности системы и выполнение последующих циклов стирания-программирования памяти программ.

Раньше микроконтроллеры программировались только с помощью параллельных протоколов, достаточно сложных для реализации. В настоящее время протоколы программирования современной EPROM и EEPROM памяти существенно изменились, что позволило выполнять программирование микроконтроллера непосредственно в составе системе, где он работает. Такой способ программирования получил название «in-system programming» или «ISP». ISP-микроконтроллеры могут быть запрограммированы после того, как их припаяли на плату. При этом сокращаются расходы на программирование, т. к. нет необходимости в использовании специального оборудования – программаторов.

Функционально Flash-память мало отличается от EEPROM. Основное различие состоит в способе стирания записанной информации. В памяти EEPROM стирание производится отдельно для каждой ячейки, а во Flash-памяти стирание осуществляется целыми блоками. Если необходимо изменить содержимое одной ячейки Flash-памяти, то потребуется перепрограммировать целый блок (или всю микросхему). В микроконтроллерах с памятью EEPROM можно изменять отдельные участки программы без необходимости перепрограммировать все устройство.

Часто указывается, что микроконтроллер имеет Flash-память, хотя на самом деле он содержит EEPROM. В настоящее время между этими типами памяти имеется мало различий, поэтому некоторые производители используют эти термины как эквивалентные.

## 4.2. Память данных

При первом знакомстве с описанием микроконтроллера многих удивит малый объем их оперативной памяти данных RAM, который обычно составляет десятки или сотни байт. Если микроконтроллер использует для хранения данных память EEPROM, то ее объем также не превышает нескольких десятков байт. Возникает вопрос, что можно сделать с таким маленьким объемом памяти.

Дело в том, программирование для микроконтроллера выполняется по несколько другим правилам, чем программирование PC. Применяя некоторые несложные правила, можно решать многие задачи с использованием небольшого объема памяти RAM. При программировании микроконтроллеров константы, если возможно, не хранятся как переменные. Максимально используются аппаратные возможности микроконтроллеров (такие, как таймеры, индексные регистры), чтобы по возможности ограничить размещение данных в RAM. Это означает, что при разработке прикладных программ необходимо предварительно позаботиться о распределении ресурсов памяти. Прикладные программы должны ориентироваться на работу без использования больших массивов данных.

В микроконтроллерах RAM используется для организации вызова подпрограмм и обработки прерываний. При этих операциях содержимое программного счетчика и основных регистров (аккумулятор, регистр состояния, индексные регистры и т. д.) сохраняется и затем восстанавливается при возврате к основной программе. *Стек* – это электронная структура данных, которая функционирует аналогично своей физической копии – стопки бумаг. Когда что-либо помещается в стек, то оно остается там до тех пор, пока не будет вынуто обратно. Когда данные удаляются, то происходит их перемещение в обратном порядке. По этой причине стек часто называют очередью типа LIFO (Last In, First Out) – «последний пришел, первый ушел».

В Принстонской архитектуре RAM используется для реализации множества аппаратных функций, включая функции стека. При этом снижается производительность устройства, т. к. для доступа к различным видам памяти требуются многократные обращения, которые не могут выполняться одновременно. По этой же причине Принстонская архитектура обычно требует большего количества тактов на выполнение команды, чем Гарвардская.

Процессоры Гарвардской архитектуры могут иметь три области памяти, которые адресуются параллельно (в одно и тоже время): 1) память программ, 2) память данных, включающая пространство ввода-вывода, и 3) стек.

В Гарвардской архитектуре стековые операции могут производиться в памяти, специально выделенной для этой цели. Это означает, что при выполнении команды вызова подпрограммы «call» процессор с Гарвардской архитектурой выполняет несколько действий одновременно. В Принстонской архитектуре при выполнении команды «call» следующая команда выбирается после того, как в стек будет помещено содержимое программного счетчика.

Необходимо помнить, что микроконтроллеры обоих архитектур имеют ограниченную емкость памяти для хранения данных. Превышение этого предела может вызвать проблемы при выполнении программы.

Если в процессоре выделен отдельный стек, и объем записанных в него данных превышает его емкость, то происходит циклическое изменение содержимого указателя стека, и указатель стека начинает ссылаться на ранее заполненную ячейку стека. Это означает, что после слишком большого количества команд «call» в стеке окажется неправильный адрес возврата, который был записан вместо правильного адреса. Если микропроцессор использует общую область памяти для размещения данных и стека, то существует опасность, что при переполнении стека произойдет запись в область данных либо будет сделана попытка записи загружаемых в стек данных в область ROM.

## 4.3. Регистры микроконтроллера. Пространство ввода-вывода

Подобно всем компьютерным системам, микроконтроллеры имеют множество регистров, которые используются для управлением различными устройствами, подключенными к процессору. Это могут быть регистры процессора (аккумулятор, регистры состояния, индексные регистры), регистры управления (регистры управления прерываниями, регистры управления таймером) или регистры, обеспечивающие ввод-вывод данных (регистры данных и регистры управления параллельным, последовательным или аналоговым вводом-выводом). Обращение к этим регистрам может производиться различными способами.

Реализуемые микроконтроллером способы обращения к регистрам оказывают существенное влияние на их производительность. Поэтому очень важно понять, как происходит обращение к регистрам, чтобы писать эффективные прикладные программы для микроконтроллеров. В процессорах с RISC-архитектурой все регистры (часто и аккумулятор) располагаются по явно задаваемым адресам. Это обеспечивает более высокую гибкость при работе процессора.

Используя процессор, который может непосредственно обращаться к любому регистру, можно получить преимущество при разработке простых прикладных программ.

Одним из важных вопросов является размещение регистров в адресном пространстве. В некоторых процессорах все регистры и RAM располагаются в одном адресном пространстве. Это означает, что память совмещена с регистрами. Такой подход называется «отображением устройств ввода-вывода на память».

В других процессорах адресное пространство для устройств ввода-вывода отделено от общего пространства памяти. Основное преимущество размещения регистров ввода-вывода в отдельном пространстве адресов состоит в том, что при этом упрощается схема подключения памяти программ и данных к общей шине. Устройства ввода-вывода обычно занимают маленький блок адресов, что делает неудобным декодирование их адреса совместно с большими блоками основной памяти. Отдельное пространство ввода-вывода дает некоторое преимущество процессорам с Гарвардской архитектурой, обеспечивая возможность считывать команду во время обращения к регистру ввода-вывода.

После всего вышесказанного можно сказать, что Гарвардская архитектура с регистрами и переменными, размещенными в отдельных адресных пространствах, является наиболее эффективной. Однако имеется ряд причин, по которым использование микроконтроллеров Принстонской архитектуры с отображением устройств ввода-вывода на память может оказаться для некоторых применений более предпочтительным.

## 4.4. Внешняя память

Несмотря на огромные преимущества использования внутренней встроенной памяти, в некоторых случаях необходимо подключение к микроконтроллеру дополнительной внешней памяти (как памяти программ, так и данных). Существуют два основных способа подключения внешней памяти. Первый способ – подключение внешней памяти к микроконтроллеру как к микропроцессору. Многие микроконтроллеры содержат специальные аппаратные средства для такого подключения. Второй способ состоит в том, чтобы подключить память к устройствам ввода-вывода и реализовать обращение к памяти через эти устройства программными средствами. Такой способ позволяет использовать простые устройства ввода-вывода без реализации сложных шинных интерфейсов. Выбор наилучшего из этих способов зависит от конкретного приложения.

# 5. МИКРОПРОЦЕССОРНЫЕ КОНТРОЛЛЕРЫ МК48

В настоящее время основной элементной базой при разработке локальных систем управления являются средства микропроцессорной техники. Поэтому знание структуры, системы команд, схемотехнического и программного обеспечения, порядка функционирования и сопряжения с внешними устройствами микропроцессорных систем на базе того или иного микроконтроллера, используемого в качестве основного элемента системы управления объектом, является необходимым условием эффективного их использования при решении конкретных прикладных задач.

## 5.1. Семейство МК48

Семейство МК48 включает ряд моделей ОМЭВМ, функциональный состав и технические характеристики которых отражают как различие в идеологическом подходе к применению ОМЭВМ, так и прогресс технологии СБИС. Все модели, входящие в семейство МК48, являются полностью совместимыми по системе команд, назначению и разводке выводов, совокупности основных функциональных устройств из базового набора семейства.

Первое поколение ОМЭВМ семейства МК48 – БИС КМ1816ВЕ48 и КР1816ВЕ35 – являются функционально-конструктивными аналогами БИС соответственно 8748 и 8035 фирмы Intel. Выполнены по *n*-канальной МОП-технологии, что обусловлено следующими ограничениями: уровень интеграции до 18 тыс. транзисторов на кристалле, частота следования тактовых сигналов – 6.0 MHz, объемы внутреннего ОЗУ – 64 байта, ППЗУ – 1 Кбайт и минимальное время цикла – 2.5 мкс.

Второе поколение – БИС КР1816ВЕ49, КР1816ВЕ39 (аналоги БИС 8049 и 8039 фирмы Intel) – выполнено по *n*-канальной МОП-технологии с пропорциональным масштабированием, что позволило повысить уровень интеграции до 36 тыс. транзисторов на кристалле, частоту следования тактовых импульсов до 11.0 MHz, увеличить объемы ОЗУ до 128 байт, ПЗУ – 2 Кбайт и снизить минимальное время цикла до 1.36 мкс.

Третье поколение семейства МК48 – БИС ОМЭВМ серии К1830: КР1830ВЕ48, КР1830ВЕ35 (аналоги БИС 80С48, 80С35 фирмы Intel) – выполнено по КМОП-технологии, что позволило на порядок уменьшить ток потребления по сравнению с БИС КМ1816ВЕ48 и соответственно КР1816ВЕ35 при сохранении остальных параметров.

ОМЭВМ КМ1816ВЕ48, КР1816ВЕ35, КР1830ВЕ48 и КР1830ВЕ35 полностью идентичны в части структурной реализации. При этом в БИС КМ1816ВЕ48 программная память размещается во внутреннем ППЗУ с ультрафиолетовым стиранием, а в БИС КР1830ВЕ48 – во внутреннем ПЗУ масочного типа. Таким образом, оперативность программирования ППЗУ позволяет использовать ОМЭВМ КМ1816ВЕ48 при создании контроллеров единичных экземпляров или мелкосерийных изделий. Потребители БИС КР1830ВЕ48 лишены такой возможности, т. к. программирование ПЗУ осуществляется в процессе изготовления БИС по данным «прошивки» заказа потребителя.

В микросхемах КР1816ВЕ35 и КР1830ВЕ35, в отличие от БИС КМ1816ВЕ48, КР1830ВЕ48, память программ реализуется только за счет подключения внешней памяти любого типа (ОЗУ, ППЗУ, ПЗУ) общим объемом до 4 Кбайт. Эта особенность позволяет использовать их в качестве отладочного варианта, когда память программ реализуется в ОЗУ, что позволяет легко модифицировать отлаживаемые программы.

## 5.2. Описание микроконтроллера МК48

В настоящее время микроконтроллер МК48 имеет скорее не практическое, а академическое значение, т. е. представляет интерес с точки зрения изучения студентами основ микропроцессорной техники. Это объясняется тем, что, как отмечалось выше, МК48 является наиболее простым из микроконтроллеров с ядром i51, только для него имеется бесплатная версия симулятора SCM1.38, распространяемая через Интернет (специально для студентов) и используемая для автоматизации программирования в виртуальной среде и, кроме того, имеется большое количество литературы. Получив основы знаний по МК48, студент в дальнейшем может легко переходить к изучению и практическому применению самых современных, но одновременно и самых сложных микроконтроллеров.

Основные компоненты:

- ✓ структура микроконтроллера МК48;
- ✓ система памяти;
- ✓ система ввода/вывода данных;
- ✓ структура информационных связей с точки зрения программиста;
- ✓ система команд и способы адресации;
- ✓ система моделирования Single-Chip Machine (SCM 1.38);
- ✓ порядок работы с учебным микропроцессорным комплексом УМПК-48;
- ✓ программная реализация типовых задач контрольной работы по дисциплине «МПСУ».

### *5.2.1. Структура МК48*

Структура МК48 показана на рис. 5.1. Основу структуры образует внутренняя двунаправленная 8-битная шина, которая связывает между собой все элементы микроконтроллера: арифметико-логическое устройство (АЛУ), устройство управления (УУ), резидентную память данных (RAM) и память программ (EPRON), программный счетчик (PC), регистр адреса (RAR), таймер/счетчик (TCNT), слово состояния программы (PSW), регистр команд (РК) и порты ввода/вывода информации (P1, P2, DB).

В состав АЛУ входят следующие блоки: комбинационная схема обработки байтов, регистры T1 и T2, регистр-аккумулятор (A), схема десятичного корректора и схема формирования признаков (PSW). Аккумулятор используется в качестве регистра операнда и регистра результата. Регистр временного хранения операнда T1 программно недоступен и используется для временного хранения второго операнда при выполнении двухоперандных команд.

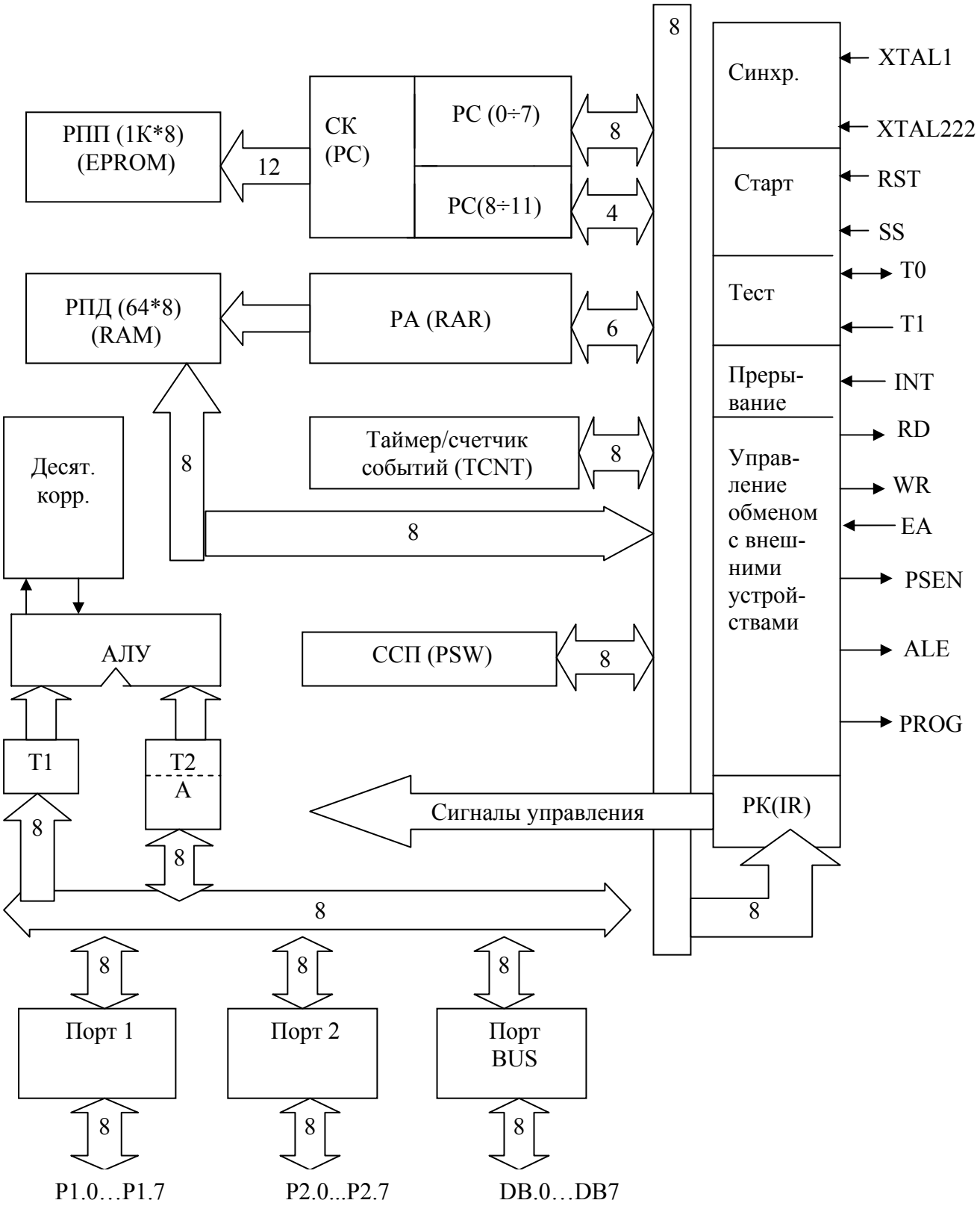



**Рис. 5.1.** Структура МК48

При выполнении операций обработки данных в АЛУ вырабатываются флаги (признаки), которые формируются на комбинационной схеме и не фиксируются на триггерах. К таким флагам относятся флаг нулевого содержимого аккумулятора и флаг наличия единицы в селектируемом бите аккумулятора. Логика условных пере-

ходов по указанным флагам позволяет выполнять команды передачи управления (JZ, JNZ, JB0 - IB7) без их фиксации на триггерах.

Флаги переноса и вспомогательного переноса (перенос «единицы» из младшей тетрады в старшую) фиксируются на триггерах, входящих в состав регистра слова состояния программы (PSW). Формат PSW показан на рис. 5.2.

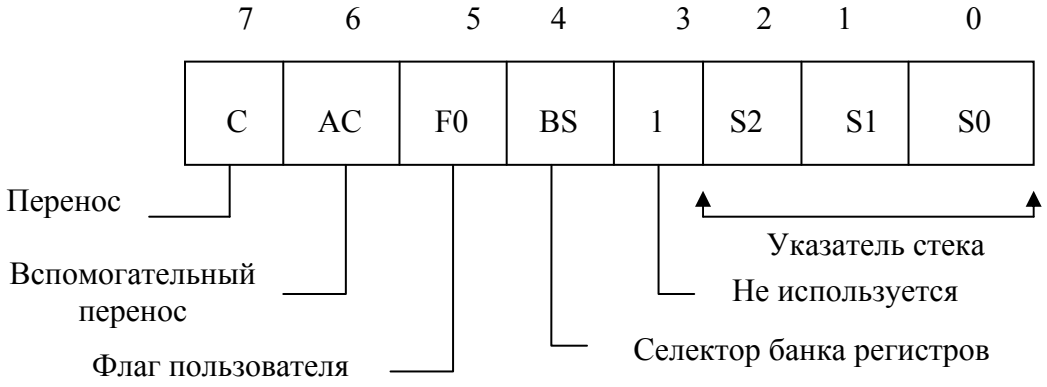



**Рис. 5.2.** Формат PSW

Кроме перечисленных признаков, логика условных переходов МК48 оперирует флагами F0 и F1, функциональное назначение которых определяется разработчиком; флагом переполнения таймера TF, сигналами на входах T0 и T1. Программистом могут быть также использованы флаги рабочего банка регистров BS и выбранного банка внешней памяти программ MB. Кроме того, логикой переходов после окончания каждого машинного цикла опрашивается флаг разрешения/запрета прерываний.

Память программ и память данных в МК48 физически и логически разделены. Память программ реализована в резидентном ППЗУ объемом 1 Кбайт. Максимальное адресное пространство, отводимое для программ, составляет 4 Кбайт. Счетчик команд (PC) содержит 12 бит, но инкрементируются в процессе счета только младшие 11 бит. Поэтому PC из предельного состояния 7FFH (если только по этому адресу не расположена команда передачи управления) перейдет в состояние 000H. Состояние старшего бита счетчика команд может быть изменено специальными командами (SEL MB0, SEL MB1). Подобный режим работы позволяет создать два банка памяти емкостью по 2 Кбайт каждый. В свою очередь, оба банка памяти разделены на страницы по 256 байт в каждой. В командах условного перехода задается 8-битный адрес передачи управления в пределах текущей страницы. Межстраничные переходы обеспечиваются модификацией 11-го бита PC при вызове подпрограмм в пределах выбранного банка памяти программ. Карта адресов памяти программ показана на рис. 5.3.

В резидентной памяти программ имеются три специализированные ячейки с адресами 00H, 03H и 07H, в которых должны находиться соответственно вектор безусловного перехода к началу программы после окончания сигнала RST, вектор прерывания от внешнего источника (вход INT) и вектор прерывания от таймера или начальная команда подпрограммы обслуживания прерывания по признаку переполнения TCNT.


**Рис. 5.3.** Карта адресов памяти программ

Резидентная память данных емкостью 64 байта имеет в своем составе два банка рабочих регистров 0 - 7 и 24 - 31 (рис. 5.4) по восемь регистров в каждом. Выбор банка регистров осуществляется по команде SEL RB0 или SEL RB1. Рабочие регистры доступны по командам с прямой адресацией, а все ячейки РПД – по командам с косвенной адресацией. В качестве регистров косвенного адреса используются регистры R0 и R1 и R0* и R1*.

Ячейки RAM с адресами 8 - 23 адресуются указателем стека из PSW и могут быть использованы в качестве 8-уровневого стека. В случае если уровень вложенности программ меньше восьми, незадействованные в стеке ячейки могут использоваться как ячейки RAM. КМ1816ВЕ48 не имеет команд загрузки байта в стек или его извлечения из стека и в нем фиксируются только содержимое PC и старшая тетрада PSW. В силу этого необходимо следить за тем, чтобы вложенные подпрограммы не использовали одни и те же рабочие регистры.

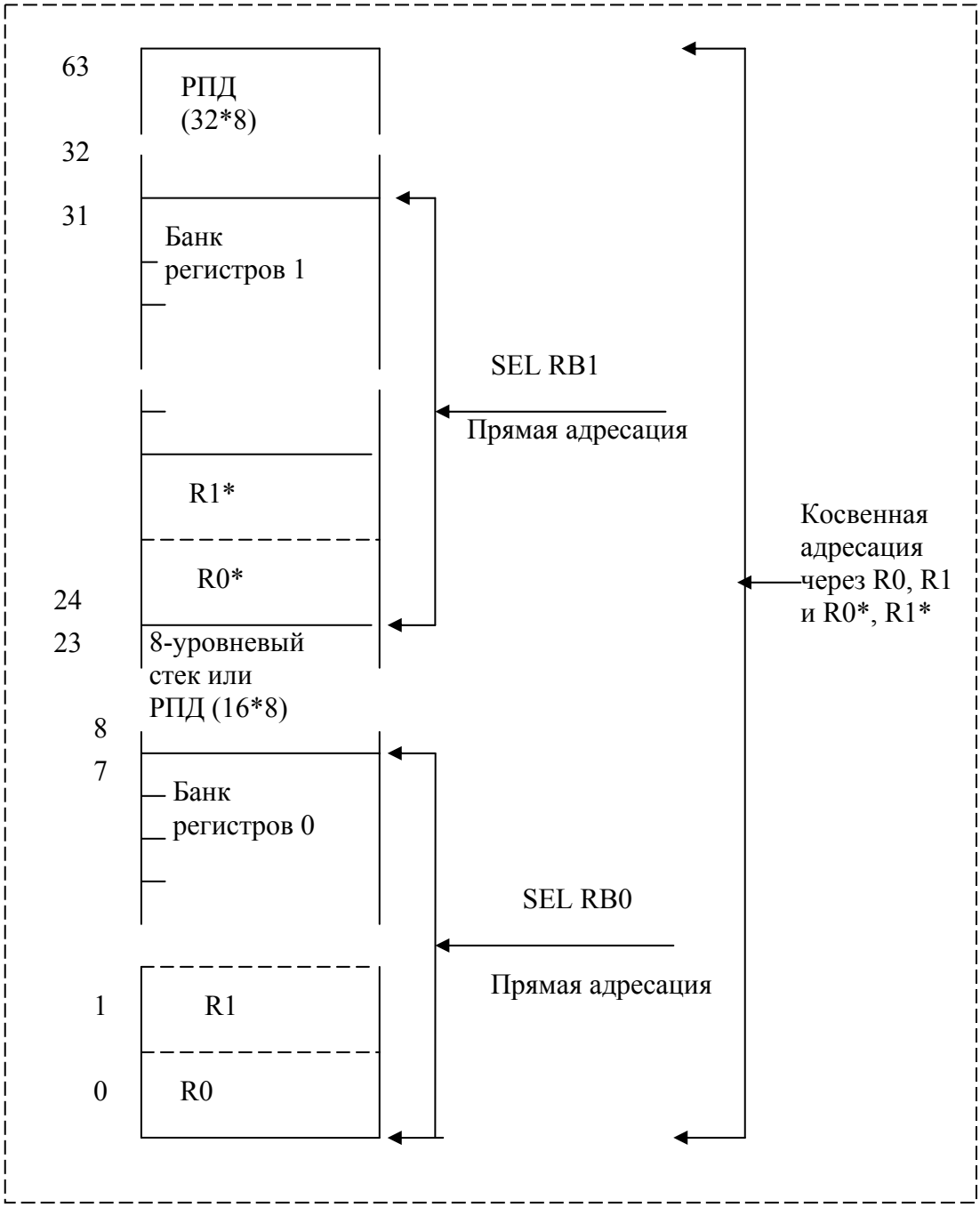



**Рис. 5.4.** Карта адресов памяти данных

Для связи МК48 с объектом управления используются 27 линий. Эти линии сгруппированы в три порта (P1, P2 и DB) по восемь линий в каждом и могут быть использованы для ввода, вывода или для ввода/вывода информации через двунаправленные линии. Кроме портов, имеются три линии, сигналы на которых могут изменять ход программы по командам условного перехода (INT, T0, T1).

Порты P1 и P2 позволяют осуществлять побитную обработку информации и являются квазидвунаправленными. Последнее означает, что для того чтобы настроить некоторую линию на режим ввода в МК48, необходимо перед этим в буферный триггер этой линии записать «1». Сигнал RST автоматически записывает во все линии портов P1 и P2 сигнал «1». Квазидвунаправленная структура портов P1 и P2 специфична тем, что в процессе ввода информации выполняется операция логического И

над вводимыми данными и текущими (последними) введенными данными. Порт P2 отличается от порта P1 тем, что его младшие четыре бита могут быть использованы для расширения микропроцессорной системы по вводу/выводу информации.

Порт DB представляет собой двунаправленный буфер с тремя состояниями и предназначен для побайтного ввода/вывода информации. Данный порт является системным: с его помощью можно расширить систему ввода/вывода информации.

Внутренний 8-битный двоичный таймер/счетчик TCNT может быть использован для формирования временных задержек и для подсчета внешних событий. Из максимального состояния FFH TCNT переходит в начальное состояние 00H. При этом устанавливается в «1» флаг переполнения, который может вызвать прерывание, если оно разрешено. Если прерывание запрещено, то флаг переполнения может быть опрошен по команде условного перехода JTF. Выполнение команды JTF, как и переход к подпрограмме обработки прерывания по вектору с адресом 7, сбрасывает флаг переполнения TCNT.

В режиме таймера на вход TCNT поступают основные синхросигналы машинного цикла ALE (400 кГц), и счетчик увеличивает свое состояние на «1» через каждые 80 мкс (12,5 кГц). Путем программной установки TCNT в исходное состояние и анализа флага переполнения могут быть реализованы различные временные задержки, лежащие в диапазоне от 80 мкс до 20,48 мс. Большие по времени задержки могут быть получены накоплением переполнений в рабочем регистре под управлением программы.

В режиме счетчика событий TCNT увеличивает свое состояние на «1» каждый раз, когда сигнал на входе T1 переходит из состояния «1» в состояние «0».

Устройство управления МК48 совместно с логической схемой переходов в каждом цикле команды формирует последовательность сигналов, управляющих функциями всех элементов микроконтроллера и системой их взаимосвязи. Функциональное назначение основных управляющих и синхронизирующих сигналов УУ (см. рис. 5.1) следующие:

- ✓ XTAL1, XTAL2 – выводы для подключения кварцевого резонатора (6 МГц);
- ✓ RST – вход сигнала общего сброса при запуске МК48;
- ✓ SS – сигнал, который совместно с сигналом ALE позволяет организовать пошаговый режим выполнения программы;
- ✓ T0 – входной сигнал, опрашиваемый по командам условного перехода JT0 или JNT0; может быть использован для вывода сигнала синхронизации после команды ENT1 CLK;
- ✓ T1 – входной сигнал, опрашиваемый командами условного перехода JT1 или JNT1; может использоваться в качестве входа внутреннего счетчика внешних событий после исполнения команды STRT CNT;
- ✓ INT – сигнал запроса прерывания от внешнего источника;
- ✓ RD,WD – стробирующие сигналы при чтении/записи информации во внешнюю память данных или УВВ;
- ✓ EA – входной сигнал отключения резидентной EPROM, позволяющей организовать выборку команд из внешней памяти программ;
- ✓ ALE – строб адреса внешней памяти; используется для приема и фиксации адреса внешней памяти на внешнем регистре; является идентификатором машинного цикла;

✓ PSEN – входной сигнал разрешения работы с внешней памятью программ;

✓ PROG – вход для подачи программируемого импульса +25 В при загрузке резидентной EPROM; выходной стробирующий сигнал для БИС расширителя ввода/вывода информации КР580ВР43.

Система команд МК48 содержит 96 основных команд, ориентированных на реализацию процедур управления. Формат команд – один или два байта (70 % – однобайтные). МК48 оперирует с командами четырех типов.

Тип 1: Команда – однобайтная, и все разряды байта занимает код операции.

Тип 2: Команда – двухбайтная; первый байт занимает код операции, а второй – непосредственное число;

Тип 3: Команда – двухбайтная; три старших разряда первого байта определяют номер страницы в пределах одного из двух банков памяти программ, а остальные пять – код операции; второй байт команды определяет адрес перехода в пределах страницы памяти программ.

Тип 4: Команда – двухбайтная; все разряды первого байта занимает код операции, а второй байт определяет адрес перехода в пределах страницы памяти программ.

Все множество команд МК48 можно разбить на пять групп по функциональному признаку: команды пересылки данных; команды арифметических операций, команды логических операций, команды передачи управления и команды управления режимами работы МК48. Все группы системы команд МК48 представлены в приложении 1.

С точки зрения программиста можно выделить 9 типов операндов, между которыми выполняется информационный обмен (рис. 5.5). При этом основным элементом, через который остальные устройства обмениваются данными, является аккумулятор А.

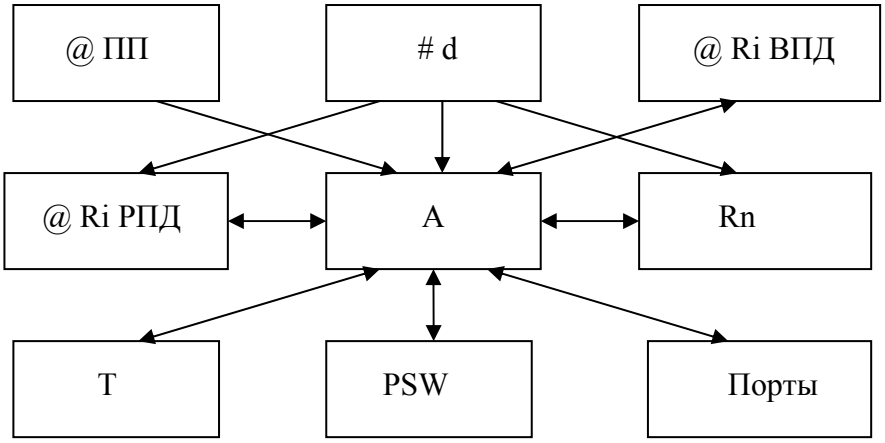



**Рис. 5.5.** Схема информационных связей в МК48

В МК48 возможна передача данных в двух режимах: пересылки (загрузки) и обмена. Пересылка предполагает передачу данных в направлении от источника (второй операнд команды) к приемнику (первый операнд команды). При этом источник не изменяет своего содержимого. Обмен предполагает одновременную передачу данных в двух направлениях: в результате операции обмена изменяются значения обоих операндов, участвующих в операции.

В МК48 предусмотрены три способа адресации:

1) прямая, когда адрес операнда содержится в теле самой команды:

```
MOV A ,R3 ; номер регистра, содержимое которого пересылается
          ; в А, указывается в 3-х младших битах кода операции
```

2) непосредственная, когда 8-битный операнд (число) располагается непосредственно в теле команды (второй байт команды):

```
MOV A , # 9FH ; содержимое второго байта команды – число 9FH –
              ; пересылается в А
```

3) косвенная, при которой адрес операнда располагается в регистре R0 или R1 каждого из двух банков регистров:

```
MOV A , @ R1 ; содержимое ячейки РПД по адресу, хранимому в
             ; регистре R1, пересылается в А
```

### *5.2.2. Система моделирования Single-Chip Machine*

Система моделирования Single-Chip Machine (SCM 1.38) предназначена:

- ✓ для моделирования работы ОЭВМ КМ1816ВЕ48 в совокупности с микросхемой-расширителем портов ввода вывода КР580ВР43 и блоком внешней памяти данных объемом 256 байт;
- ✓ разработки и отладки программ для микроконтроллеров серии МК48;
- ✓ исследования поведения внутренних и внешних сигналов указанных микросхем.

***Возможности программы***

Программа SCM (Single-Chip Machine) выполнена в виде независимого запускаемого модуля, работоспособного под управлением операционной системы MS Window 95/98/2000/NT/XP. SCM включает средства отладки и редактирования программ на ассемблере со встроенным интерпретатором, что делает ввод программ намного удобнее и эффективнее, чем в других эмуляторах подобного класса.

Выполнение программы пользователя осуществляется с максимальным приближением к действительности с помощью имитационной модели, уровень детализации которой равен одному такту (1 т = 0,5 мкс). Доступны следующие режимы моделирования:

- на один такт вперед;
- на один машинный цикл вперед;
- на один шаг вперед;
- выполнение шага до изменения регистра адреса микроконтроллера;
- выполнение до ближайшей точки останова;
- выполнение до конца программы;

– выполнение до первой пустой ячейки памяти;
– на один машинный цикл назад;
– на один такт назад.

Кроме того, пользователю предоставляются такие средства, как:

– временные диаграммы внутренних и внешних сигналов;
– имитация внешних сигналов с отображением изменений на условно-графическом отображении микросхем;
– возможность изменения значений узлов микроЭВМ в процессе работы модели и др.

Встроенный редактор-компилятор позволяет набирать программы на ассемблере МК48, а затем с помощью кнопки «компиляция» перевести текст программ в машинные кода и записать его как в файл ПЗУ с расширением «.MPM», так и в ПЗУ микроконтроллера для отображения в отладчике. Кроме формата ПЗУ, «.MPM» расшифровывается как Microcontroller Program Memory. Существует еще более старый формат представления памяти программ – так называемый формат HEX, который поддерживается ПО всех моделей программаторов, SCM по умолчанию работает с «.MPM» форматом ПЗУ (более удобен для отладки, содержит массу отладочной информации, например: точка входа в программу, точка выхода, тип данных, данные инициализации и др.), но также может работать с форматом ПЗУ «.HEX». Таким образом, SCM полностью совместим с промышленными эмуляторами (например, AVSIM).

SCM позволяет найти и удалить все временные бесполезные файлы, созданные в процессе работы. Кроме того, поддерживаются следующие функции распределенного моделирования (на нескольких компьютерах):

– прием библиотек;
– загрузка системы команд;
– сопряжения с другими программами-эмуляторами;
– обмена сообщений между пользователями, подключенными к одному серверу.

***Настройки программы***

Настройки можно изменить двумя путями. Первый – с помощью меню «Настройки» и выбором соответствующего пункта. Второй – с помощью самостоятельного редактирования файла SCMF.CFG, текстового файла конфигурации программы SCM.

Рассмотрим, какие настройки можно изменять и как они будут влиять на работу программы. При выборе в главном меню пункта «Настройки» на экране появятся следующее пункты меню.

Пункт «Загружать программу на входе» – если находится в выделенном состоянии, то при нажатии на кнопку «Питание» автоматически будет загружена последняя модифицированная программа в редактор и в модель. Если программу загрузить невозможно, то будет выдано соответствующее сообщения, поясняющее причину.

Пункт «Выдавать запрос на выходе» – при выборе данного пункта на выходе из программы при наличии модифицированной программы будет выдаваться пригла-

шение ее сохранить. В противном случае закрытие программы будет игнорировать изменения в Вашей программе.

Пункт «Интенсивное автосохранение» – в выбранном состоянии исходный текст будет время от времени записываться, что немного тормозит работу, но позволяет избавиться от полной потери программы в случае системного сбоя.

***Директивы и выражения***

Ассемблер МК48 допускает применение директив:

EQU – объявление именованных констант;

DB – определение байта;

DW – определение слова (2 байта);

ORG – указание абсолютного адреса следующей команды.

Кроме того, в мнемониках команд и директивах допускаются арифметические выражения.

**EQU**

Синтаксис: <имя> EQU <численное значение>.

Имя – последовательность символов языка, не являющихся числом или зарезервированным словом. Имена не могут повторяться.

Численное значение: число, записанное согласно правилам языка и лежащее в диапазоне [0…255].

Действие директивы: при компиляции встреченное в исходном тексте программы имя будет заменено указанным числом.

```
Пример:  A1   EQU   #20h
         A    EQU   #40h ;     Ошибка – A – зарезервированное слово
         A2   EQU   #300 ;     Ошибка – 300>255
```

**DB**

Синтаксис: DB <численное значение>.

Численное значение: число, записанное согласно правилам языка и лежащее в диапазоне [0…255].

Действие директивы: в соответствующее место программы в машинных кодах компилятор поместит указанный байт.

```
Пример:  DB   #20h
         DB   #250
         DB   #10111111b
```

**DW**

Синтаксис: DW <численное значение>.

Численное значение: число, записанное согласно правилам языка и лежащее в диапазоне [0…65535].

Действие директивы: в соответствующее место программы в машинных кодах компилятор поместит два байта (старший байт раньше младшего).

```
Пример:  DW   #20FFh
         DW   #2500
         DW   #1011111100001111b
```

**ORG**

Синтаксис: ORG <адрес>.

Адрес представляет собой число, находящееся в пределах адресного пространства памяти команд.

Действие директивы: следующие команды будут размещаться в памяти, начиная с указанного адреса. Пересечение адресов (две команды, расположенные по одному адресу) транслятор воспримет как ошибку.

Пример:
```
ORG 30
ORG 250
ORG 0
```

***Арифметические выражения***

В мнемониках команд вместо численных констант допускается ввод арифметических выражений. Численное значение такого выражения вычисляется в процессе трансляции. В выражениях можно использовать операторы (по убыванию приоритета):

Not (поразрядная инверсия)

And, *, /

Or, Xor, +, –

В качестве операндов могут выступать численные константы и имена, определенные директивой EQU.

# 6. ТЕХНИКО-ЭКОНОМИЧЕСКИЕ АСПЕКТЫ РАЗРАБОТКИ МИКРОПРОЦЕССОРНЫХ СИСТЕМ УПРАВЛЕНИЯ

От правильного выбора микроконтроллера (МК) зависит успех или провал всего проекта. При выборе МК существуют качественные и количественные критерии.

Основная цель – выбор наименее дорогого микроконтроллера (для снижения общей стоимости изделия), но в то же время удовлетворяющего системной спецификации, т. е. требованиям по производительности, надежности, условиям применения. Общая стоимость изделия включает: инженерные исследования и проработки, разработку, производство (комплектующие и труд), гарантийный ремонт, обновление, обслуживание, совместимость, простоту в обращении.

## 6.1. Процесс и критерии выбора МК

Приступая к выбору МК, нужно первоначально задаться вопросом: «Что должен делать микроконтроллер?». Ответ определяет необходимые в конкретной системе характеристики МК и, таким образом, является решающим фактором в процессе выбора.

Второй шаг – проведение поиска МК, которые удовлетворяют всем системным требованиям. Обычно этот этап включает подбор литературы, т. е. технических описаний желаемого МК, и демонстрационные консультации. Самым оперативным источником получения информации о новейших МК являются коммерческие журналы и Интернет.

Хорошо, если идеально подошел однокристальный МК, предпочтительный по критериям цены и надежности, в противном случае придется продолжить выбор. Повторный поиск может касаться МК, которые имели бы минимальную избыточность и при этом наилучшим образом удовлетворяли предъявляемым требованиям: имели минимум внешних электрических цепей и подходили по стоимости и габаритам.

Выбор МК значительно сужается, если вопросы преемственности разработок и программной совместимости диктуют применение МК конкретного производителя.

Последняя стадия выбора состоит из нескольких этапов, цель которых – сузить список приемлемых МК до одного. Эти этапы включают в себя: анализ цены, доступности и средств разработки, поддержку производителя, стабильность поставок и наличие других производителей. Чтобы прийти к оптимальному решению, возможно, весь процесс придется повторить несколько раз.

Основные критерии выбора МК в порядке значимости представлены ниже. Более детально они будут рассмотрены в дальнейшем.

1. Пригодность для прикладной системы. Возможность создания подобной системы на однокристальном МК или для максимального соответствия спецификации требуется несколько дополнительных микросхем:

– имеет ли МК необходимое количество контактов-портов ввода-вывода (в случае их недостатка решение поставленных задач становится невозможным, в случае избытка цена может оказаться слишком высокой);

– имеются ли в наличии требуемые периферийные устройства (такие, как последовательные порты ввода-вывода, RAM, ROM, АЦП, ЦАП);

– удовлетворяет ли производительность ядра CPU (вычислительная мощность, позволяющая обрабатывать системные запросы с учетом программирования на выбранном прикладном языке) предъявляемым требованиям;

– правильно ли сделан экономический расчет бюджета и достаточно ли средств для применения МК именно этого производителя, т. к. зачастую вопрос цены является определяющим.

2. Доступность:

– наличие необходимого количества данных устройств;

– состояние производства выбранного МК на данный момент (серийное, опытные образцы, снят с производства);

– прогноз на будущее.

3. Поддержка разработчика:

– ассемблеры, компиляторы, средства отладки, отладочные мониторы, внутрисхемные эмуляторы;

– утилиты, в том числе «бесплатные» ассемблеры;

– отладчики программ в исходных текстах;

– оценочный модуль (EVM);

– насадки для логических анализаторов;

– исполнительная система реального времени;

– примеры применения, сообщения об ошибках;

– Интернет;

– образцы исходных текстов.

4. Поддержка применений:

– наличие специальной группы, которая занимается только поддержкой применений;

– персонал компании: инженеры, техники или продавцы;

– компетентность и квалификация поддерживающего персонала.

5. История производителя:

– компетентность, подтвержденная разработками;

– надежность микросхем (качество продукции);

– срок и надежность поставок;

– продолжительность работы в данной области;

– финансовый отчет.

## 6.2. Системные требования

Проведение системного анализа проекта позволит определить требования к микроконтроллеру. Какие требуются периферийные устройства? Применяются ли битовые операции или только числовые? Сколько требуется манипуляций для обработки данных? Должна ли система управляться по прерываниям, по готовности, требуется ли синхронная обработка по таймерам? Каким количеством устройств-

битов ввода-вывода необходимо управлять? Какие устройства из числа возможных типов I/O-устройств должны контролироваться-управляться: RS-232C-терминалы, выключатели, реле, клавиши, сенсоры (температура, свет, напряжение), звуковые устройства, визуальные индикаторы (LCD-дисплеи), аналого-цифровые преобразователи (АЦП) или цифроаналоговые преобразователи (ЦАП)? Одно или несколько напряжений питания требуется для системы? Насколько требователен МК к качеству источника питания, каковы диапазон допустимых уровней напряжения и допустимые пульсации напряжения питания? Изделие должно работать от сети или от батарей? Во втором случае необходимо определить тип батарей и их технические характеристики. Существуют ли ограничения по размеру, весу и дизайну? Существуют ли какие-либо специфические требования к условиям окружающей среды (температура, влажность, взрывоопасная атмосфера, давление, высота)? Будут ли использоваться в системе диски либо только Flash или ROM? Будет ли система применяться для решения задач в реальном масштабе времени, и если да – нужно ли создавать или приобретать специальное операционное ядро реального времени или, возможно, будет достаточно простейшего доступного и широко используемого системного программного обеспечения? Достаточно ли персонала и времени для разработки собственного операционного ядра и оправданы ли экономически затраты на такую разработку? Как будут оплачиваться авторские права и программное обеспечение? Для решения задач реального времени требуется большая исследовательская работа.

***Основные особенности МК***

Микроконтроллеры в целом можно разделить на группы 8-, 16- и 32-разрядных по размеру их арифметических и индексных регистров. Впрочем, некоторые разработчики считают, что 8/16/32-разрядную архитектуру определяет разрядность шины.

Тем не менее требования к МК остаются прежними. Способен ли дешевый МК удовлетворить требованиям системы или требуется дорогой 16-или 32-разрядный? Возможна ли программная эмуляция особенностей 16/32-разрядного МК на недорогом 8-разрядном МК, жертвуя размером исполняемого кода и скоростью? Например, может ли 8-разрядный МК быть использован с программным макросом, чтобы эмулировать 16-разрядный аккумулятор и операции индексирования? Выбор прикладного языка (высокого уровня вместо Ассемблера) может сильно повлиять на производительность системы, которая затем может диктовать выбор 8-/16-/32-разрядной архитектуры, но ограничение по цене может отвергнуть этот выбор.

Тактовая частота МК и скорость передачи данных по шине определяет, сколько вычислений может быть выполнено за единицу времени. Некоторые МК имеют узкий диапазон возможной тактовой частоты, в то время как другие могут работать вплоть до нулевой частоты. Иногда выбирается специфическая тактовая частота, чтобы сгенерировать другую тактовую частоту, требуемую в системе, например для задания скоростей последовательной передачи. В основном вычислительная мощность, потребляемая мощность и стоимость системы увеличиваются с повышением тактовой частоты. При повышении частоты цена системы увеличивается не только по причине стоимости МК, но и требующихся дополнительных микросхем, таких как RAM, ROM и контроллеры шины.

Рассмотрим также технологию, с использованием которой изготовлен микропроцессор: NMOS (N-канальный металл-окисел-полупроводник) в сравнении с HCMOS (комплементарным MOS высокой плотности интеграции). В отличие от ранних NMOS-процессоров, в HCMOS сигналы изменяются в диапазоне от 0 до значения напряжения питания. Так как это обстоятельство может значительно влиять на уровень помех в схеме, обычно предпочтение отдается HCMOS-процессорам. Кроме того, HCMOS потребляют меньшую мощность и, таким образом, меньше нагреваются. Габаритные размеры элементов в HCMOS меньше, что позволяет иметь более плотные схемы и работать при более высоких скоростях. Более плотный дизайн также уменьшает стоимость, т. к. на кремниевой пластине того же размера можно произвести большее количество чипов. По этим причинам сегодня большинство МК изготавливаются с использованием HCMOS-технологии.

***Возможности МК***

Чтобы достичь более высокого уровня интеграции и надежности при более низкой цене, все МК имеют встроенные дополнительные устройства. Эти устройства под управлением микропроцессорного ядра МК выполняют определенные функции. Встроенные устройства повышают надежность системы, потому что они не требуют никаких внешних электрических цепей. Они предварительно тестируются производителем и освобождают место на плате, т. к. все соединительные электрические цепи выполнены на кристалле в МК. Наиболее популярные среди внутрисхемных устройств – устройства памяти, таймеры, системные часы-генератор и порты ввода-вывода (I/O). Устройства памяти включают оперативную память (RAM), постоянные запоминающие устройства (ROM), перепрограммируемую ROM (EPROM), электрически перепрограммируемую ROM (EEPROM) и электрически стираемую память (EEM). Термин EEM на самом деле относится к инженерно развиваемой версии МК, где EEPROM заменяет ROM, чтобы снизить время разработки. Таймеры представляют собой часы реального времени и таймеры периодического прерывания. Следует принимать во внимание диапазон и разрешение таймера, так же, как и другие подфункции (сравнение таймера или входных линий измерения длительности сигнала). I/O-устройства включают последовательные порты связи, параллельные порты (I/O линии), аналого-цифровые и цифроаналоговые преобразователи, драйверы жидкокристаллического экрана (LCD) и драйверы вакуумного флуоресцентного экрана (VFD).

Другими, реже используемыми встроенными ресурсами являются внутренняя/внешняя шина, таймер слежения за нормальным функционированием системы (COP), сторожевая схема (WatchDog), система обнаружения отказов тактового генератора, выбираемая конфигурация памяти и системный интеграционный модуль (SIM). SIM обычно заменяет внешнюю логику, необходимую для взаимодействия с внешними устройствами через выбранные контакты микросхемы.

В большинство МК с внутрисхемными ресурсами включается блок конфигурационных регистров для управления этими ресурсами. Иногда этот блок размещается в различных местах карты памяти. Имеется пользовательский или фабричный тестовый регистр, наличие или отсутствие которого косвенно говорит о том, какое значение производитель придает качеству. Наличие конфигурационных регистров приводит к проблеме случайного изменения конфигурации системы «блуждаю-

щим» кодом. Для предотвращения этого используется механизм «замка» (до того как конфигурационный регистр может быть изменен, биты в другом регистре должны быть изменены в определенной последовательности). Впрочем, сложность конфигурационных регистров не должна пугать – благодаря гибкости при низкой стоимости они крайне ценны, так что одному МК можно найти различные применения.

***Набор команд МК***

Необходимо внимательно изучить набор команд и регистров каждого МК, т. к. они играют важнейшую роль в определении возможностей системы в целом. Отдельная задача – изучение возможностей режимов адресации для целей системы. Особое внимание обратить на наличие:

– специальных команд, которые будут использоваться в системе (умножение, деление и табличное интерполирование);

– мало потребляющих режимов для экономии батарейного питания (остановка, остановка с низким потреблением мощности или ожидание);

– команд битовых манипуляций (установка, очистка, тест и изменение бита, команды перехода по установленному – очищенному биту), облегчающих применение МК или команд манипуляции с битовыми полями.

Реальным критерием производительности является количество тактовых циклов, требуемое для выполнения задачи, а не количество исполненных команд. Для справедливого сравнения лучше закодировать одинаковую программу и сравнить полное число выполненных тактовых циклов и использованных байтов. Есть ли в карте операционных кодов нереализованные инструкции и что получится, если они случайно выполнятся? Обработает ли система подобную ситуацию корректно обработчиком «исключительных» событий или это приведет к выходу системы из строя?

***Прерывания МК***

Проверка структуры прерываний необходима всегда, когда создается система реального времени. Сколько линий или уровней прерывания имеется и сколько их требуется для системы? Имеется ли маска уровней прерывания? Когда уровень прерывания подтвержден, есть ли индивидуальные векторы для программы обработчика прерывания, или должны опрашиваться все возможные источники прерывания, чтобы определить источник? В критических по скорости применениях, таких как управление принтером, критерием выбора подходящего МК может быть время реакции на прерывание, то есть время от начала прерывания (в худшем случае, фазированного относительно тактового генератора МК) до выполнения первой команды соответствующего обработчика прерывания.

***Характеристика поставщика***

Следующий шаг в сокращении списка технически приемлемых МК – проверка производителей и поставщиков МК. Поставщик может быть производителем МК или дилером, который является полномочным представителем нескольких производителей. Наилучшим образом удовлетворит запросы поставщик с более широким ассортиментом продуктов и репутацией высокого качества, надежности, обслуживания и своевременной поставки при справедливой цене. При этом работа с одним

поставщиком и заказ больших объемов компонентов предусматривает дополнительные условия: льготная цена, специальные услуги и поддержка. Поставщики, которые снабжают не только микроконтроллерами, но и памятью (RAM, ROM), дискретными устройствами, стандартными цифровыми логическими устройствами, специальными микросхемами, заказными устройствами (CSIC), устройствами для специфических применений (ASIC) и программируемыми логическими устройствами (PLD), смогут лучше удовлетворить постоянно растущие запросы.

***Характеристика производителя***

Другими критериями в выборе производителя-поставщика МК являются стабильность, монопольное положение, дополнительные сведения о нем из литературы и техническая поддержка. Стабильность может быть надежно проверена путем установления стажа работы в этой области. Монопольное положение поставщика, к сожалению, обычно норма, т. к. большинство производителей МК редко пересекаются в производстве с другими производителями. Если производитель имеет хорошие показатели в снабжении, доставке и цене, то его монопольное положение не должно являться препятствием.

***Поддержка производителя***

Прямая поддержка производителя включает маркетинг продажи и прикладную инженерную поддержку. Необходимо оценить сервис и качество телефонной связи со специалистами, а также наличие электронной почты и web-сайта. Количество и доступность телефонных линий, а также система «Voice Mail» определяют возможность оперативного получения нужной информации. «Voice Mail», сложная компьютерная система, в которой каждый пользователь имеет свой собственный автоответчик, защищенный паролем, с расширенными возможностями (например, переадресацию сообщений). Необходимо также обратить внимание на график работы персонала поддержки. Имеют ли они другие обязанности? Каков количественный состав специалистов по обслуживанию? Возможно ли получение консультации у специалистов-производственников: по готовой продукции, по производству, по качеству, электронщиков, программистов? Каким образом организуется взаимодействие заводских инженеров и персонала поддержки?

Возможно ли получение технических консультаций и поддержки через web-сайты сети Интернет? Какую информацию можно получить на web-сайте производителя (прикладные программы, новости о продуктах, свежие программы, исходные тексты, сообщения об ошибках, электронную почту, проведение конференций)? Насколько удовлетворяет работа с данным сайтом (низкая скорость передачи данных, неудобный интерфейс, плохая поисковая система)? Существует ли у производителя специальная служба технической поддержки и какие мероприятия проводятся с целью повышения уровня обслуживания клиентов?

***Литературная поддержка***

Литература охватывает широкий набор печатных материалов, которые могут помочь сделать правильный выбор. Это различные буклеты производителя (технические описания и рекомендации по применению), а также технические издания, доступные в местном книжном магазине или библиотеке. Специализированные

журналы не только указывают на популярность производителя МК, но и публикуют беспристрастные мнения независимых экспертов.

***Окончательный выбор***

Для окончательного шага в процессе выбора строится таблица, содержащая рассматриваемые МК в одной графе, а их важные характеристики – в другой. Затем прикладываются бланки технических описаний производителей, чтобы получить объективное наглядное сравнение. Некоторые производители имеют предварительно сделанные сравнительные описания их МК. Среди возможных характеристик – цена (на ожидаемый объем продукции, включая предсказание будущей цены в процессе предполагаемого жизненного цикла вашего изделия), RAM, ROM, EPROM, EEPROM, таймеры, АЦП, ЦАП, последовательные порты, параллельные порты, скорость шины, специальные команды (умножение, деление), число доступных прерываний, время отклика прерывания (время от начала прерывания до выполнения первой команды, управляемой прерыванием), размер и тип корпуса (керамический DIP или LCC, пластиковый 0,3" DIP или 0,6" DIP, сжатый DIP; расстояние между контактами для поверхностного монтажа), требования по питанию и другие немаловажные характеристики.

Если на данном этапе выбора в списке фигурирует более одного МК, нужно рассмотреть возможности расширения требований к МК и стоимость, а также какие особенности МК могут понадобиться в будущем.

# 7. МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ВЫПОЛНЕНИЮ КОНТРОЛЬНОЙ РАБОТЫ

Для выполнения контрольной работы студентам выдается дистрибутив для установки на ПЭВМ симулятора SCM 1.38. После запуска программы на экран монитора выдается рабочая среда для разработки и отладки прикладных программ пользователя, как показано на рис. 7.1.

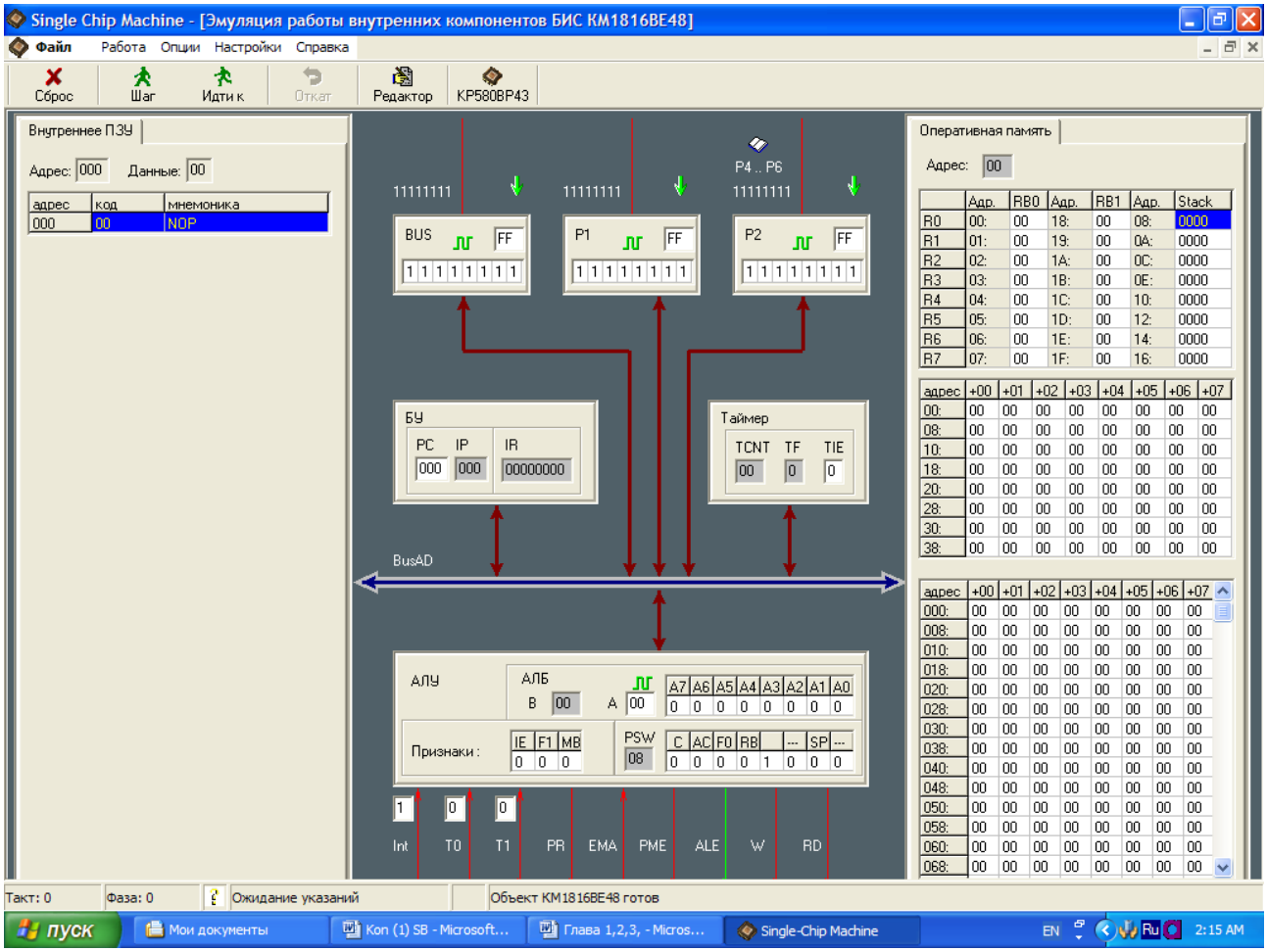



**Рис. 7.1.** Рабочая среда симулятора SCM 1.38

Рабочее поле симулятора содержит панель меню, панель инструментов, адресное пространство памяти программ и памяти данных, а также структуру МК48 со всеми компонентами: портами, блоком управления, таймером, АЛУ, аккумулятором, словом состояния программы (ССП) и набором управляющих сигналов, с которыми пользователь может активно взаимодействовать при разработке и отладке прикладных программ. Используя данную виртуальную среду, студент пишет прикладную программу на языке ассемблера, затем компилирует ее на язык шестнадцатиричных кодов с целью ввода последних на учебный микропроцессорный комплекс УМПК-48 и реализует программу уже в реальной физической среде. Приемы работы с симулятором SCM 1.38 изложены в его справочной системе.

## 7.1. Порядок работы с модулем УМПК-48

Внешний вид модуля показан на рис. 7.2.


**Рис. 7.2.** УМПК-48

Для подготовки к работе необходимо перевести переключатели на плате модуля в следующие положения:

- верхние четыре переключателя – с первого по третий – в нижнее, четвертый – в верхнее положение;
- восемь нижних правых переключателей – в нижнее;
- левый нижний переключатель – в нижнее (в этом положении переключателя в момент нажатия кнопки «R» проводится инициализация (обнуление) памяти команд, внешней и внутренней памяти данных и регистров МК48; в верхнем (замкнутом) положении переключателя инициализация не производится);
- второй слева нижний переключатель – в верхнее.

После включения питания провести начальную установку модуля нажатием на кнопку «R». При этом выдается звуковой сигнал и на знаковом дисплее высвечивается надпись НАЧАЛО, что свидетельствует о готовности модуля к работе и возможности ввода директив с помощью функциональной клавиатуры, а команд данных – с помощью 16-ричной клавиатуры.

Функциональная клавиатура определяет следующие режимы работы модуля.

***1) Режим просмотра и изменения содержимого памяти команд.***

При нажатии на клавишу «АПК» (Адрес памяти команд) во 2, 3 и 4-м разрядах дисплея загораются нижние сегменты, означающие возможность ввода адреса памяти команд с помощью последовательного нажатия трех 16-ричных числовых кла-

виш. При этом на дисплей выводится адрес открытой ячейки памяти команд и ее содержимое. Просмотр памяти команд вперед осуществляется с помощью клавиши «ЗпУв» (Запись/Увеличить); после каждого нажатия адрес памяти команд увеличивается на единицу. Просмотр памяти команд назад осуществляется нажатием на клавишу «Ум» (Уменьшить); после каждого нажатия адрес памяти команд уменьшается на единицу.

Ввод нового значения ячейки по выбранному адресу производится нажатием на соответствующие 16-ричные числовые клавиши. При этом в 6-м разряде дисплея загорается запятая, являющаяся признаком записи. Введенное значение записывается в память команд клавишей «Зп/Ув», при этом осуществляется автоматический переход к следующей ячейке.

***2) Вывод на дисплей содержимого программного счетчика.***

Данный режим осуществляется нажатием на клавишу «ПрСч» (Программный счетчик). На дисплей выводится значение программного счетчика на момент последнего выхода из программы пользователя и содержимое памяти команд по этому адресу.

***3) Просмотр и изменение содержимого внутренней памяти данных.***

Для ввода адреса внутренней памяти данных необходимо нажать клавишу «АПД» (Адрес памяти данных) и после сообщения 0 - 1 нажать клавишу «0». На дисплей выводится сообщение I - _ _ , после чего необходимо ввести две цифры требуемого адреса. При этом на дисплее индицируются адрес и содержимое ячейки внутренней памяти данных и возможны просмотр и изменение содержимого памяти.

***4) Просмотр и изменение содержимого внешней памяти данных.***

В этом случае необходимо также нажать клавишу «АПД» и после вывода сообщения 0 - 1 нажать на клавишу «1». На дисплей выводится сообщение E - _ _ , после чего необходимо ввести две цифры требуемого адреса. При этом на дисплее индицируются адрес и содержимое ячейки внешней памяти данных и возможны просмотр и изменение содержимого памяти.

***5) Просмотр и изменение содержимого внутренних регистров МК48***

Данный режим осуществляется нажатием на клавишу «РГ» (Регистры), после чего на дисплей выводятся мнемоническое обозначение регистра и его содержимое в следующем порядке:

A – аккумулятор;
F – регистр флагов и указатель стека (PSW);
t – таймер/счетчик;
b 0 r 0 – регистр 0 нулевого банка регистров;
. . . . . .
b 0 r 7 – регистр 7 нулевого банка регистров;
b 1 r 0 – регистр 0 первого банка регистров;
. . . . . .
b 1 r 7 – регистр 7 первого банка регистров;
PCL – младший байт программного счетчика;
PCH – старший байт программного счетчика.

Просмотр содержимого регистров вперед в указанном порядке осуществляется с помощью клавиши «ЗпУв», назад – с помощью клавиши «Ум». Изменение содер-

жимого выбранного регистра может быть произведено набором требуемого значения с помощью числовых клавиш и последующим нажатием на клавишу «ЗпУв».

***6) Режим запуска программы пользователя.***

Запуск программы пользователя может быть произведен в двух режимах: пошаговом и автоматическом. Для запуска программы необходимо ввести начальный адрес прикладной программы с помощью директив «АПК» или «ПрСч» и нажать клавишу «П» (Пуск). На индикаторах дисплея выводится сообщение 0 - 1, определяющее один из двух режимов запуска программы: нажатием на клавишу «0» осуществляется автоматический запуск программы, а на «1» – пошаговое выполнение команд программы. Выполнение очередной команды программы реализуется с помощью клавиши «ШК» (Шаг команды). В пошаговом режиме 12 верхних светодиодов модуля индицируют адрес выполняемой команды.

***7) Режим выхода из программы пользователя.***

Данный режим осуществляется нажатием на клавишу «Ст» (Стоп). При этом сохраняется содержимое ячеек внутренней памяти данных и регистров МК48, останавливается таймер и на дисплей выводится адрес останова. Это дает возможность просмотра результатов выполнения программы. Продолжить выполнение программы с адреса останова можно последовательно выполнив директивы «ПрСч» и «П».

## 7.2. Указания к выполнению контрольной работы

Прежде чем приступить к выполнению контрольной работы по дисциплине «Микропроцессорные устройства систем управления», необходимо:

- ознакомиться со структурной схемой МК48, с функциональным назначением всех его элементов и узлов;
- установить на компьютер и ознакомиться по справочной системе с работой симулятора SCM 1.38;
- ознакомиться с системой команд МК48 по функциональным группам и способами адресации (см. приложение 1);
- ознакомиться с порядком работы на УМПК-48;
- получить задание на выполнение контрольной работы у преподавателя. Результаты выполнения задания (прикладной программы) должны быть выведены в квазидвунаправленный порт P1 и расширитель ввода/вывода информации – порты P4, P5, P6 и P7. Состояние квазидвунаправленного порта ввода/вывода P1 МК48 может задаваться поразрядно переключателями S8.1...S8.8 и индицироваться светодиодами HL3...HL10. Для достоверной индикации состояния разрядов порта P1, включенных на вывод, соответствующие секции переключателя S8 должны быть разомкнуты.

Контрольная работа должна содержать: подробную схему алгоритма выполнения прикладной программы, программу на языке Ассемблера и программу на языке машинных (16-ричных) кодов, которую можно непосредственно реализовать на микропроцессорном комплексе УМПК-48.

Контрольная работа оформляется в соответствии с требованиями, изложенными в [3].

## 7.3. Краткие пояснения к выполнению заданий

На базе SCM1.38 и УМПК-48 необходимо выполнить три задания по следующим темам:

1) ознакомление с архитектурой и системой команд микроконтроллера КМ 1816 ВЕ 48;

2) программное управление двигателем постоянного тока (ДПТ НВ) по заданной тахограмме;

3) программные модели элементов цифровой техники.

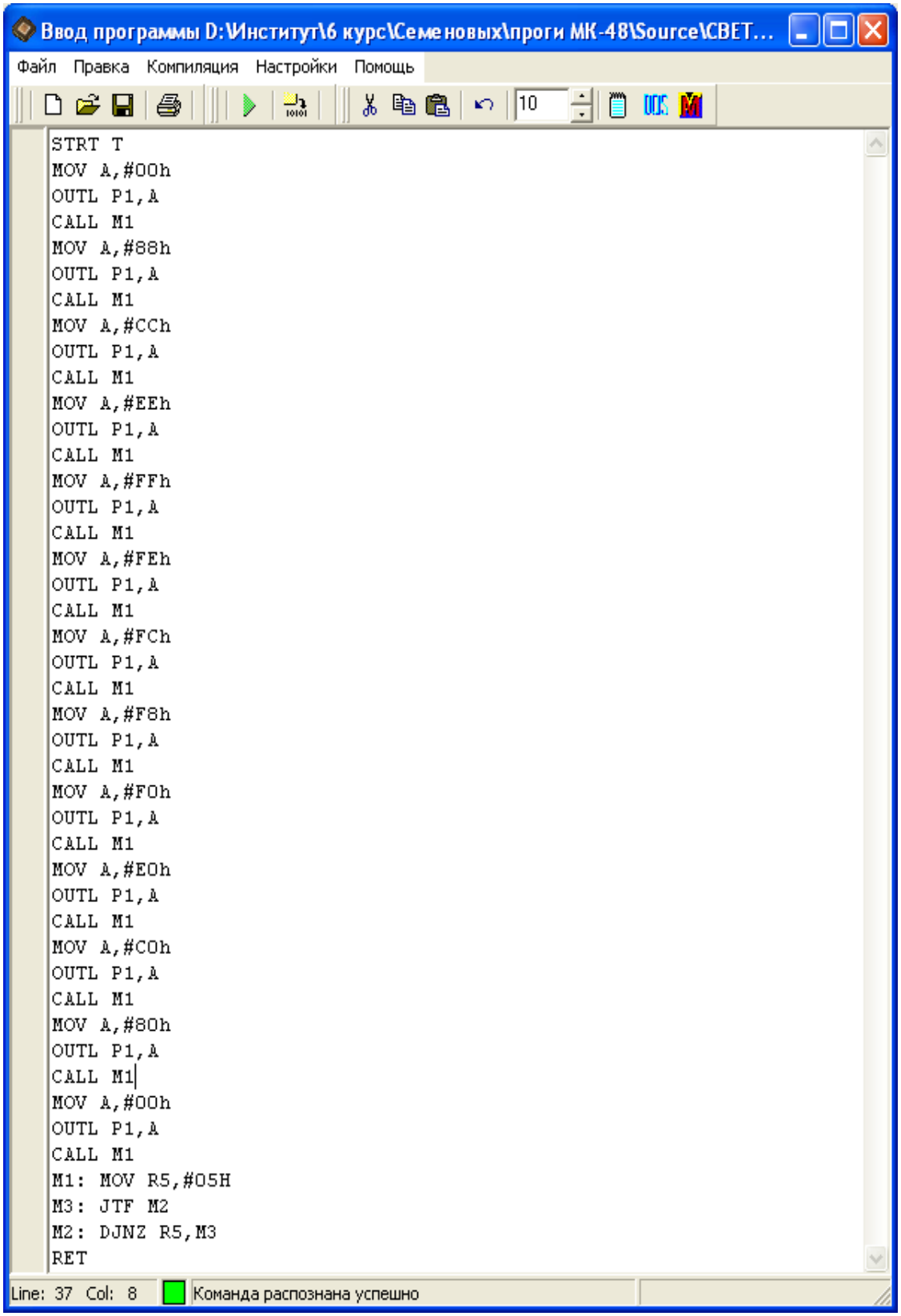
### *Задание № 1. Ознакомление с архитектурой МК48*

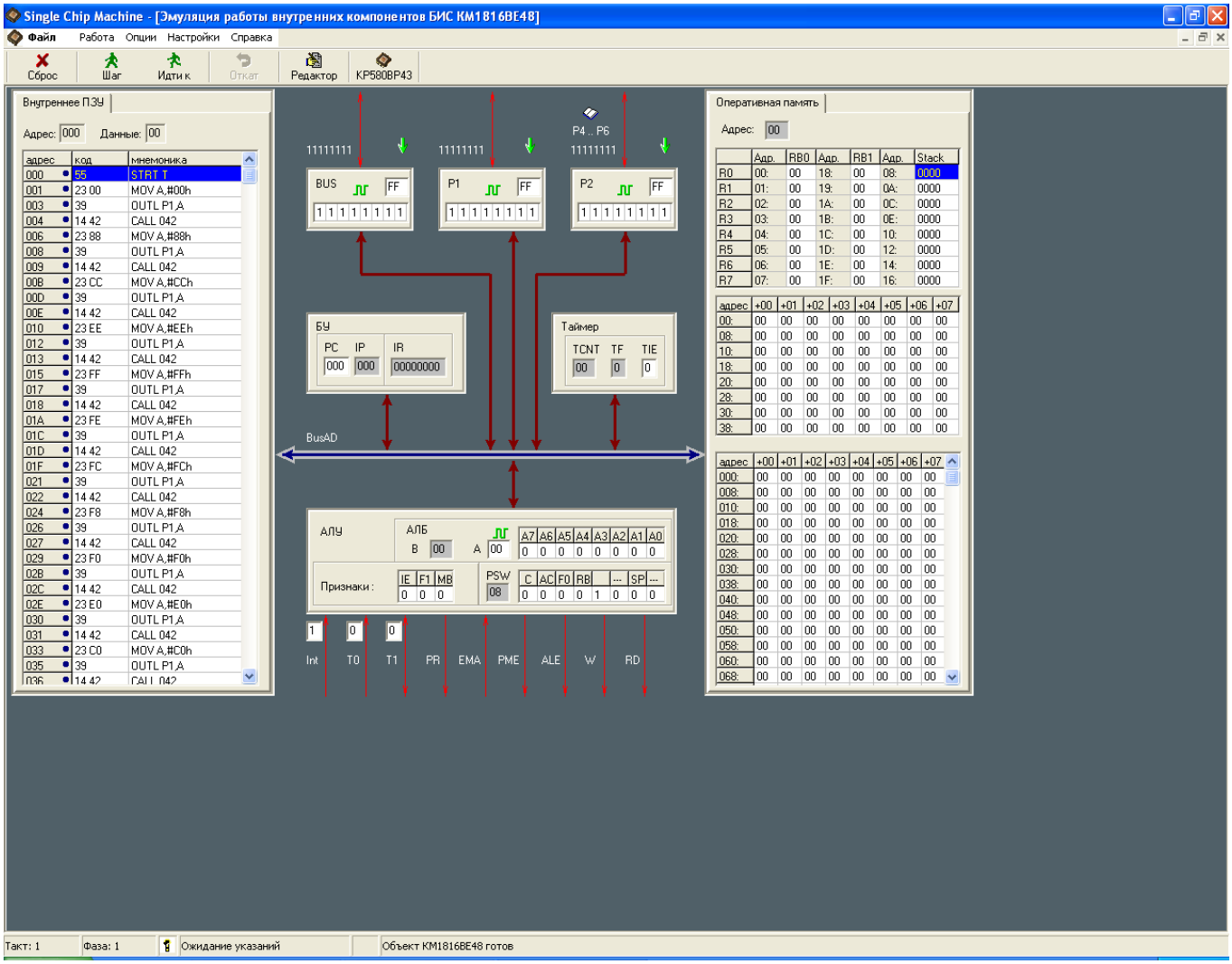
***Цель:*** ознакомление с архитектурой МК48 и функциями всех его элементов, системой команд и способами адресации, в том числе косвенной при работе с памятью данных, с системой моделирования Single-Chip Machine и порядком работы с учебным микропроцессорным комплексом УМПК-48.

***Задание.*** Используя систему моделирования SCM 1.38, написать на языке ассемблера программу типа «бегущие огни», отладить ее в редакторе симулятора, скомпилировать на язык шестнадцатиричных кодов и запрограммировать УМПК-48 для получения результатов выполнения программы. Последние выводятся в восьмиразрядный порт P1 МК48, к которому подключены восемь светодиодов.

***Пояснения к выполнению задания.*** Словесно алгоритм формулируется следующим образом: горят 8 светодиодов порта P1. Сначала их гасим. Затем зажигаем попарно одновременно с задержкой $\tau = 0{,}2$ с. Светодиоды подключены к разрядам порта P1.7 и P1.3, P1.6 и P1.2, P1.5 и P1.1, P1.4 и P1.0. Когда все светодиоды зажглись, последовательно их гасим с задержкой $\tau = 0{,}2$ с, начиная со светодиода, подключенного к разряду порта P1.0 до P1. Затем цикл повторяется.

Ниже представлен текст программы на языке ассемблера, скомпилированная программа с приведением 16-ричных кодов, а также диаграммы состояния аккумулятора, содержимое которого выводится в порт P1 с различными масштабами.


Ввод программы D:\Институт\6 курс\Семеновых\проги МК-48\Source\СВЕТ...
Файл Правка Компиляция Настройки Помощь
10
STRT T
MOV A,#00h
OUTL P1,A
CALL M1
MOV A,#88h
OUTL P1,A
CALL M1
MOV A,#CCh
OUTL P1,A
CALL M1
MOV A,#EEh
OUTL P1,A
CALL M1
MOV A,#FFh
OUTL P1,A
CALL M1
MOV A,#FEh
OUTL P1,A
CALL M1
MOV A,#FCh
OUTL P1,A
CALL M1
MOV A,#F8h
OUTL P1,A
CALL M1
MOV A,#F0h
OUTL P1,A
CALL M1
MOV A,#E0h
OUTL P1,A
CALL M1
MOV A,#C0h
OUTL P1,A
CALL M1
MOV A,#80h
OUTL P1,A
CALL M1
MOV A,#00h
OUTL P1,A
CALL M1
M1: MOV R5,#05H
M3: JTF M2
M2: DJNZ R5,M3
RET
Line: 37 Col: 8
Команда распознана успешно


Single Chip Machine - [Эмуляция работы внутренних компонентов БИС КМ1816ВЕ48]
Файл
Работа
Опции
Настройки
Справка
Сброс
Шаг
Идти к
Откат
Редактор
КР580ВР43
Внутреннее ПЗУ
Адрес:
Данные:
адрес
код
мнемоника
Оперативная память
BUS
P1
P2
P4 .. P6
БУ
PC
IP
IR
Таймер
TCNT
TF
TIE
BusAD
АЛУ
АЛБ
Признаки:
PSW
Int
T0
T1
PR
EMA
PME
ALE
W
RD
Такт: 1
Фаза: 1
Ожидание указаний
Объект КМ1816ВЕ48 готов

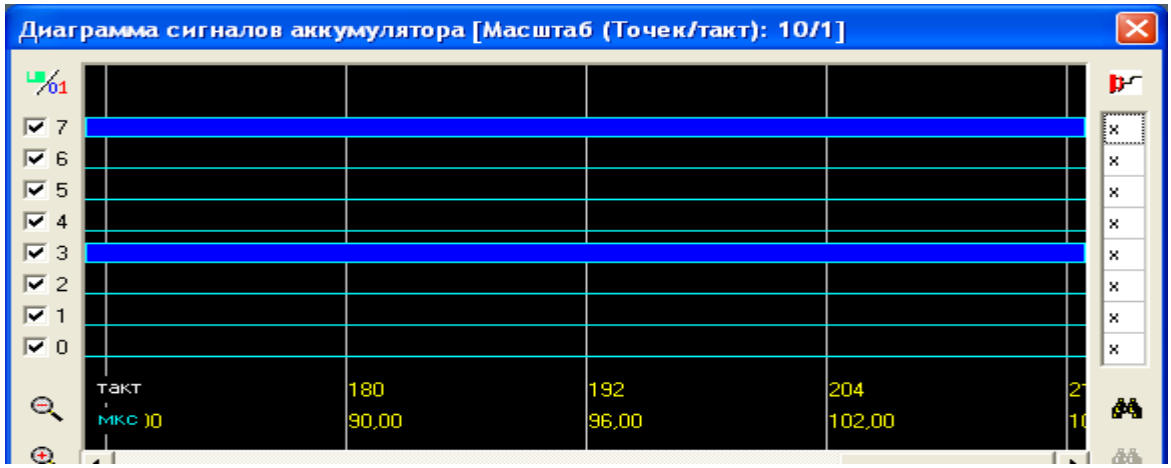


Диаграмма сигналов аккумулятора [Масштаб (Точек/такт): 10/1]
7
6
5
4
3
2
1
0
такт
мкс
180
90,00
192
96,00
204
102,00

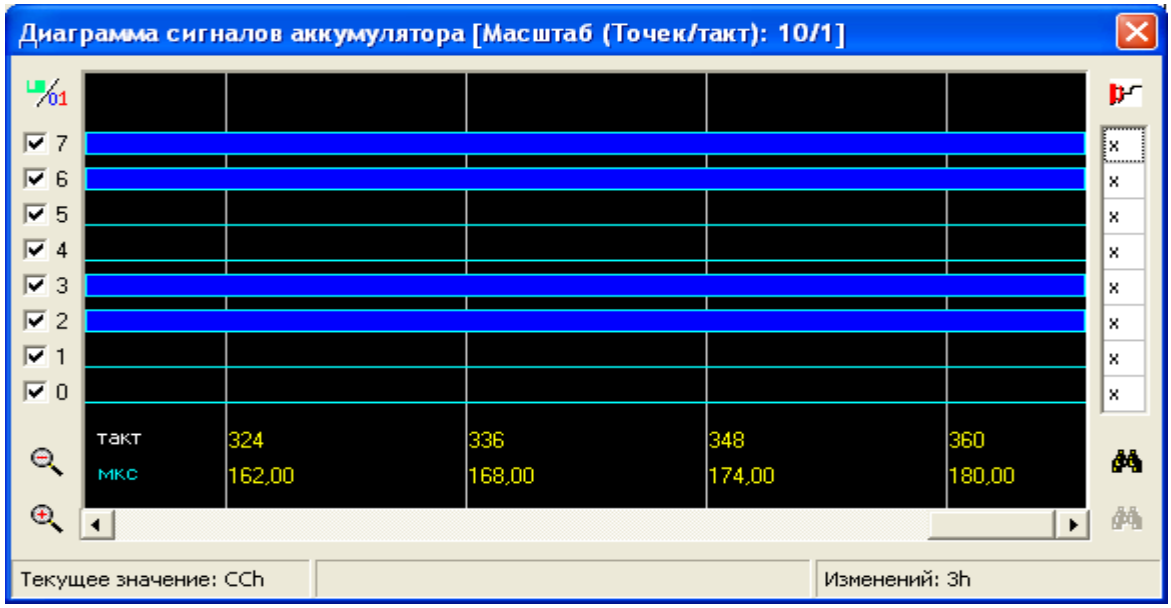


Диаграмма сигналов аккумулятора [Масштаб (Точек/такт): 10/1]
7
6
5
4
3
2
1
0
такт
324
336
348
360
мкс
162,00
168,00
174,00
180,00
Текущее значение: CCh
Изменений: 3h

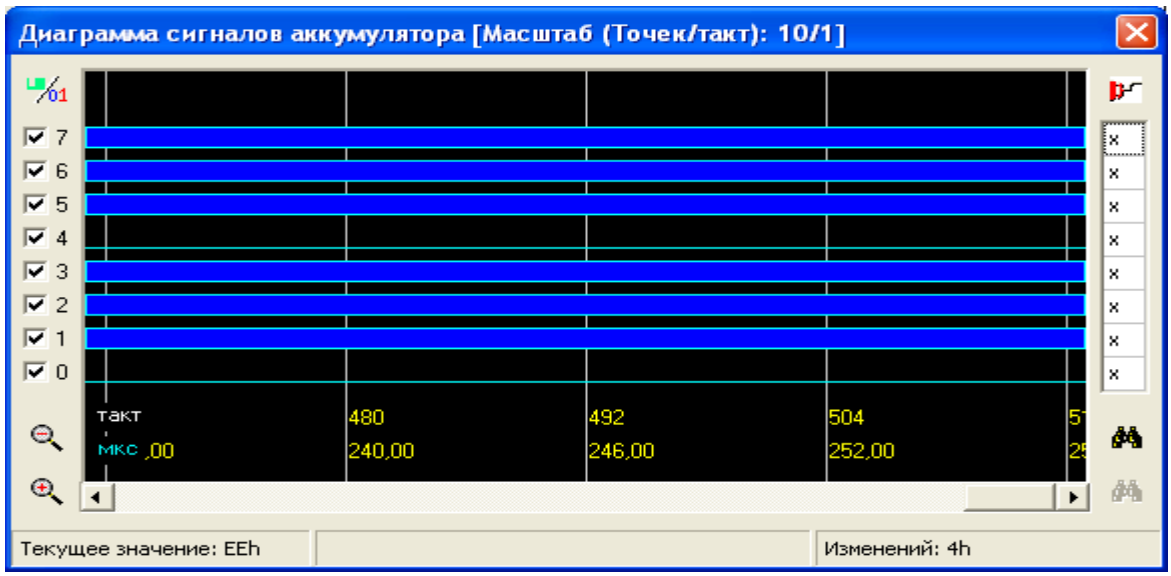


Диаграмма сигналов аккумулятора [Масштаб (Точек/такт): 10/1]
7
6
5
4
3
2
1
0
такт
480
492
504
мкс
240,00
246,00
252,00
Текущее значение: EEh
Изменений: 4h

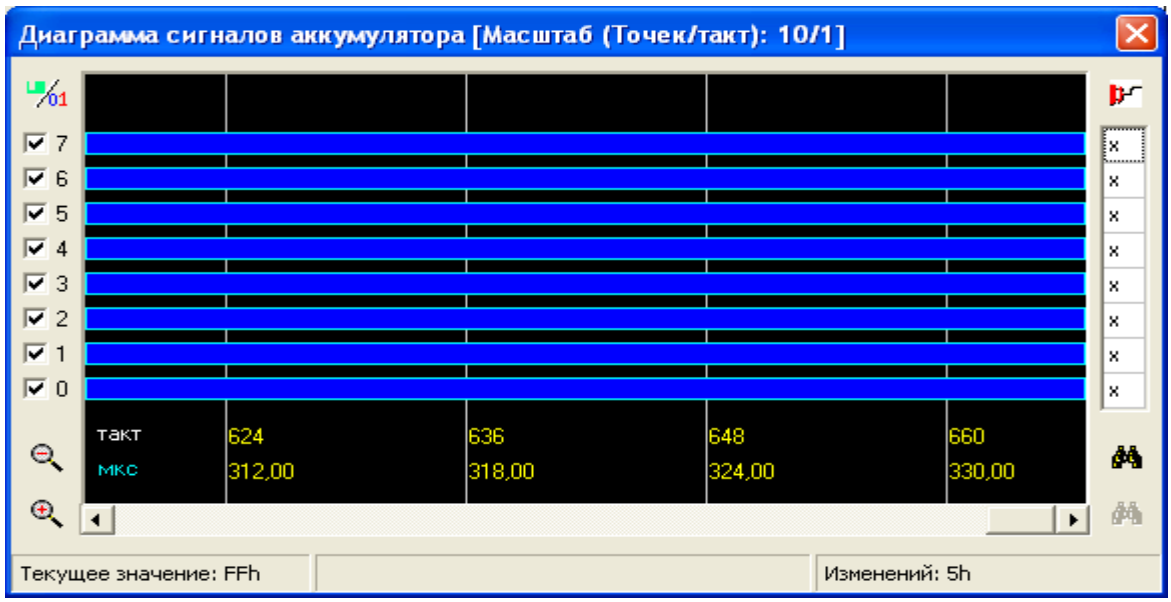


Диаграмма сигналов аккумулятора [Масштаб (Точек/такт): 10/1]
7
6
5
4
3
2
1
0
такт
624
636
648
660
мкс
312,00
318,00
324,00
330,00
Текущее значение: FFh
Изменений: 5h


Диаграмма сигналов аккумулятора [Масштаб (Точек/такт): 10/1]
7
6
5
4
3
2
1
0
такт
780
792
804
мкс ,00
390,00
396,00
402,00
Текущее значение: FEh
Изменений: 6h

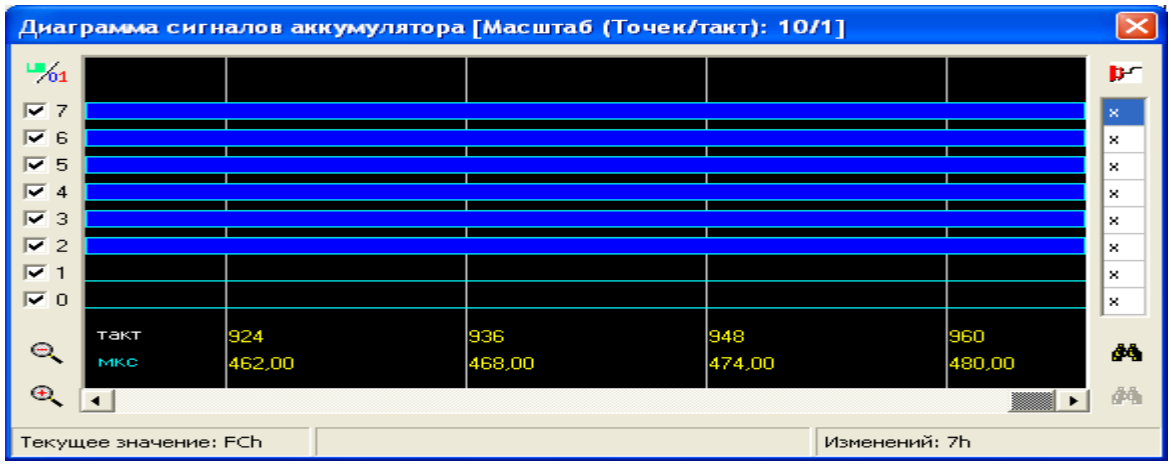


Диаграмма сигналов аккумулятора [Масштаб (Точек/такт): 10/1]
7
6
5
4
3
2
1
0
такт
924
936
948
960
мкс
462,00
468,00
474,00
480,00
Текущее значение: FCh
Изменений: 7h

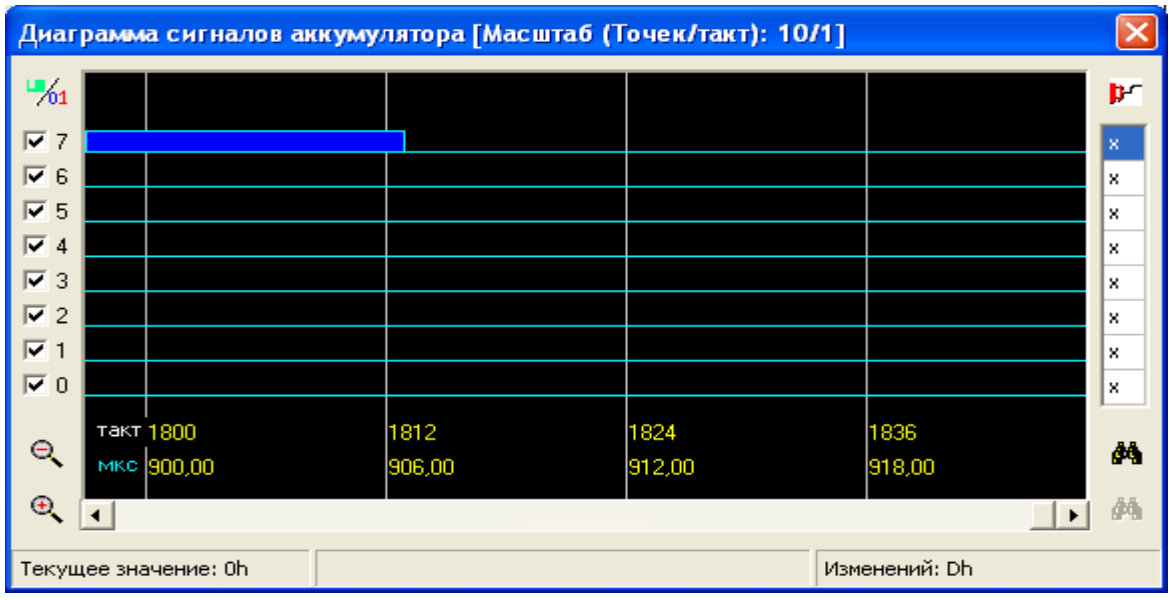
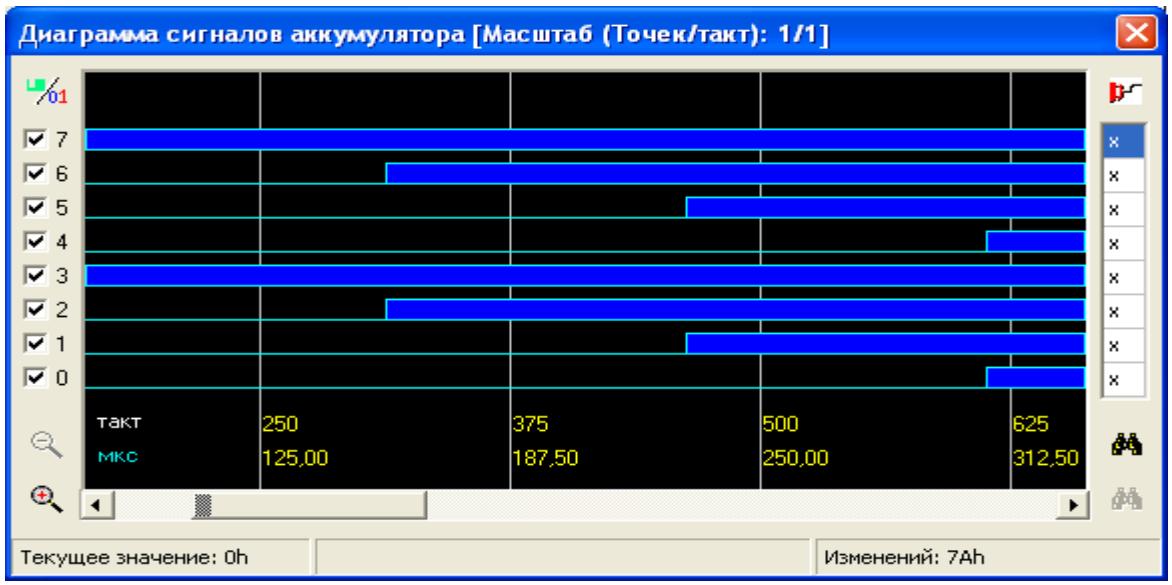
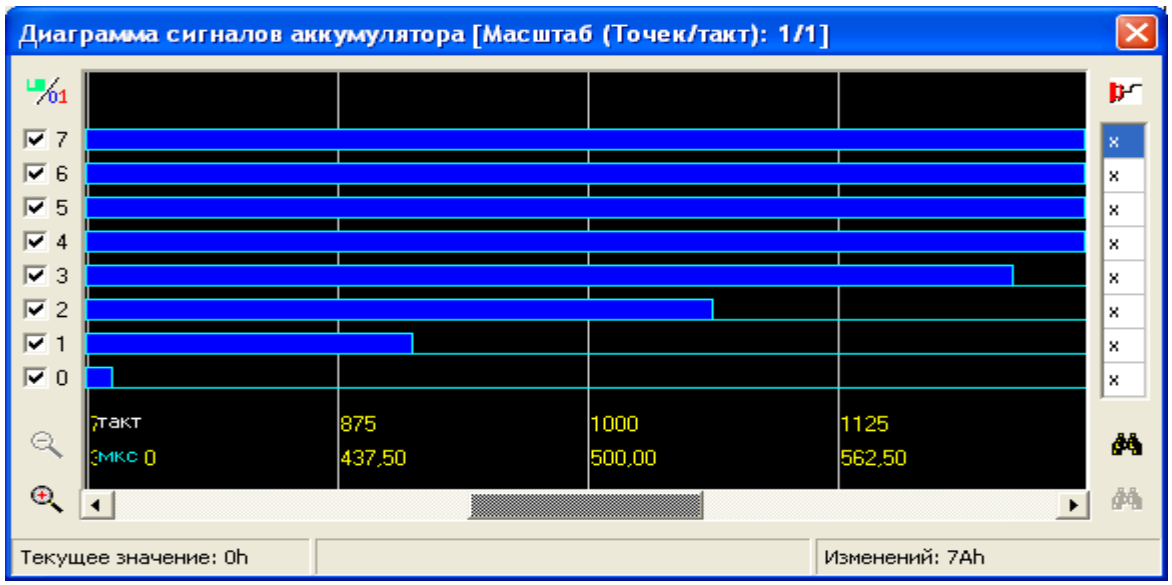
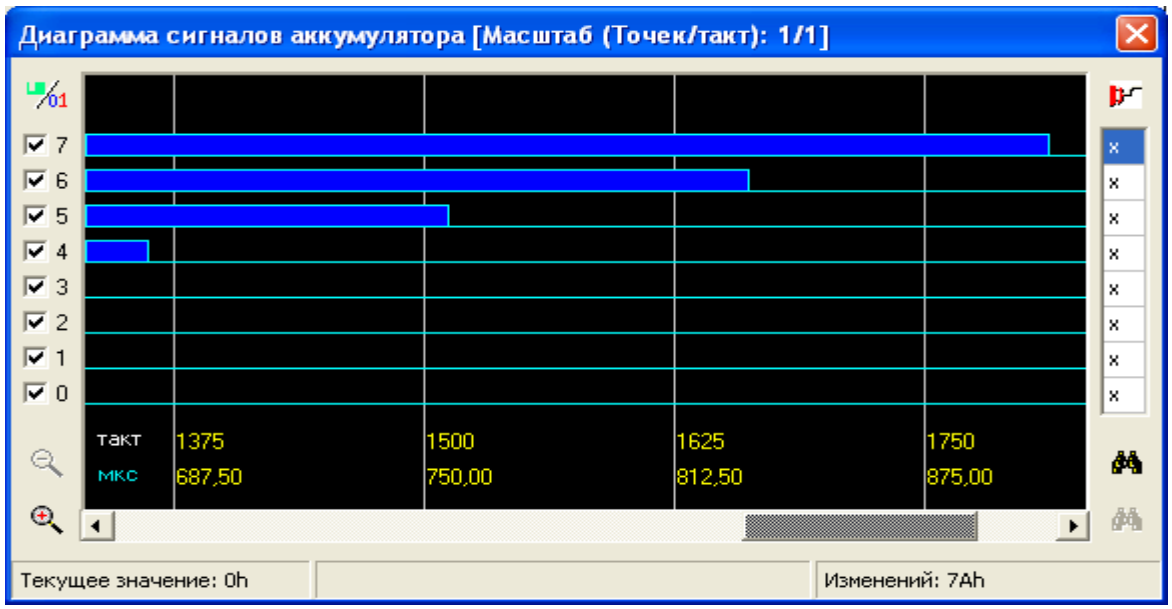


Диаграмма сигналов аккумулятора [Масштаб (Точек/такт): 10/1]
7
6
5
4
3
2
1
0
такт 1800
1812
1824
1836
мкс 900,00
906,00
912,00
918,00
Текущее значение: 0h
Изменений: Dh

**Общая картина диаграммы в уменьшенном масштабе**



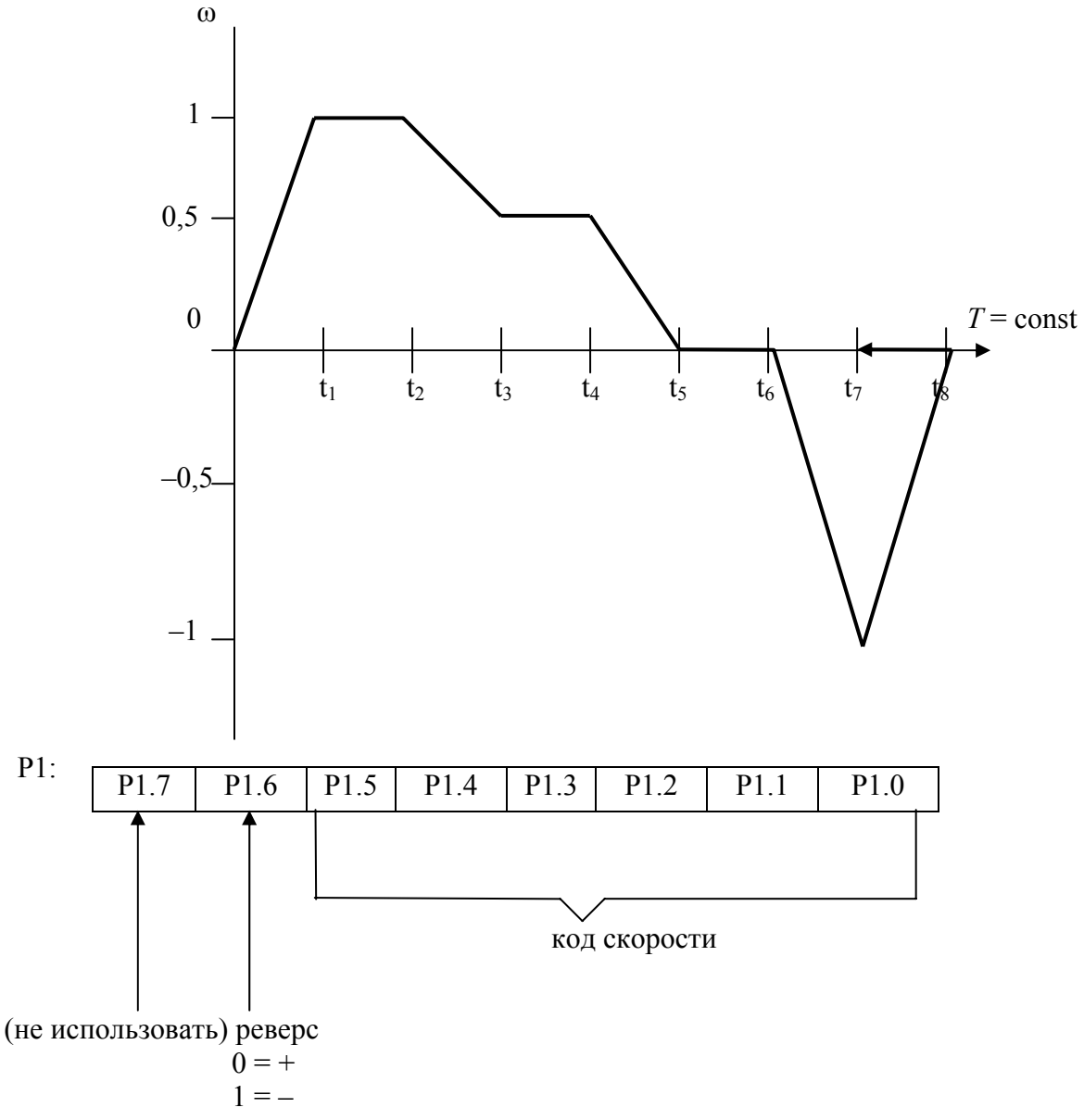



## *Задание № 2.*
## *Программное управление двигателем по заданной тахограмме*

***Цель:*** формирование у студентов навыков использования микроконтроллеров в качестве устройства управления объектом, в качестве которого выступает двигатель постоянного тока независимого возбуждения (ДПТНВ).

***Задание.*** Используя систему моделирования SCM 1.38, написать на языке ассемблера программу реализации тахограммы работы ДПТНВ, отладить ее в редакторе симулятора, скомпилировать на язык 16-ричных кодов и запрограммировать УМПК-48 для получения результатов выполнения программы (варианты тахограмм см. в приложении 2). Последние выводятся в восьмиразрядный порт P1 МК48, к которому подключены широтно-импульсный преобразователь и силовая часть. В разряды порта P1.0 – P1.5 выводится код скорости, а в разряд P1.6 – знак скорости (0 – положительная скорость, 1 – отрицательная).

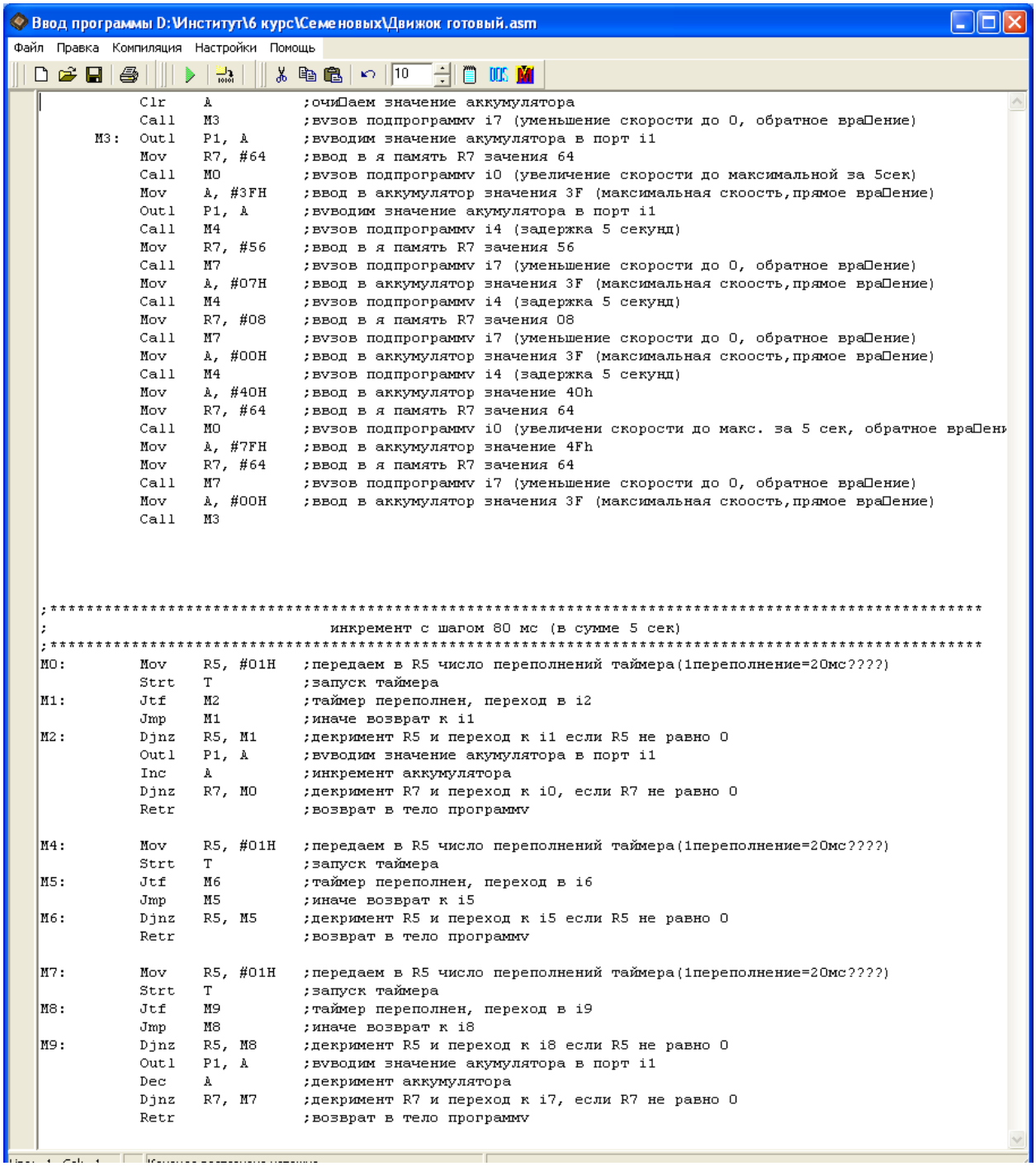
***Пояснения к выполнению задания.***



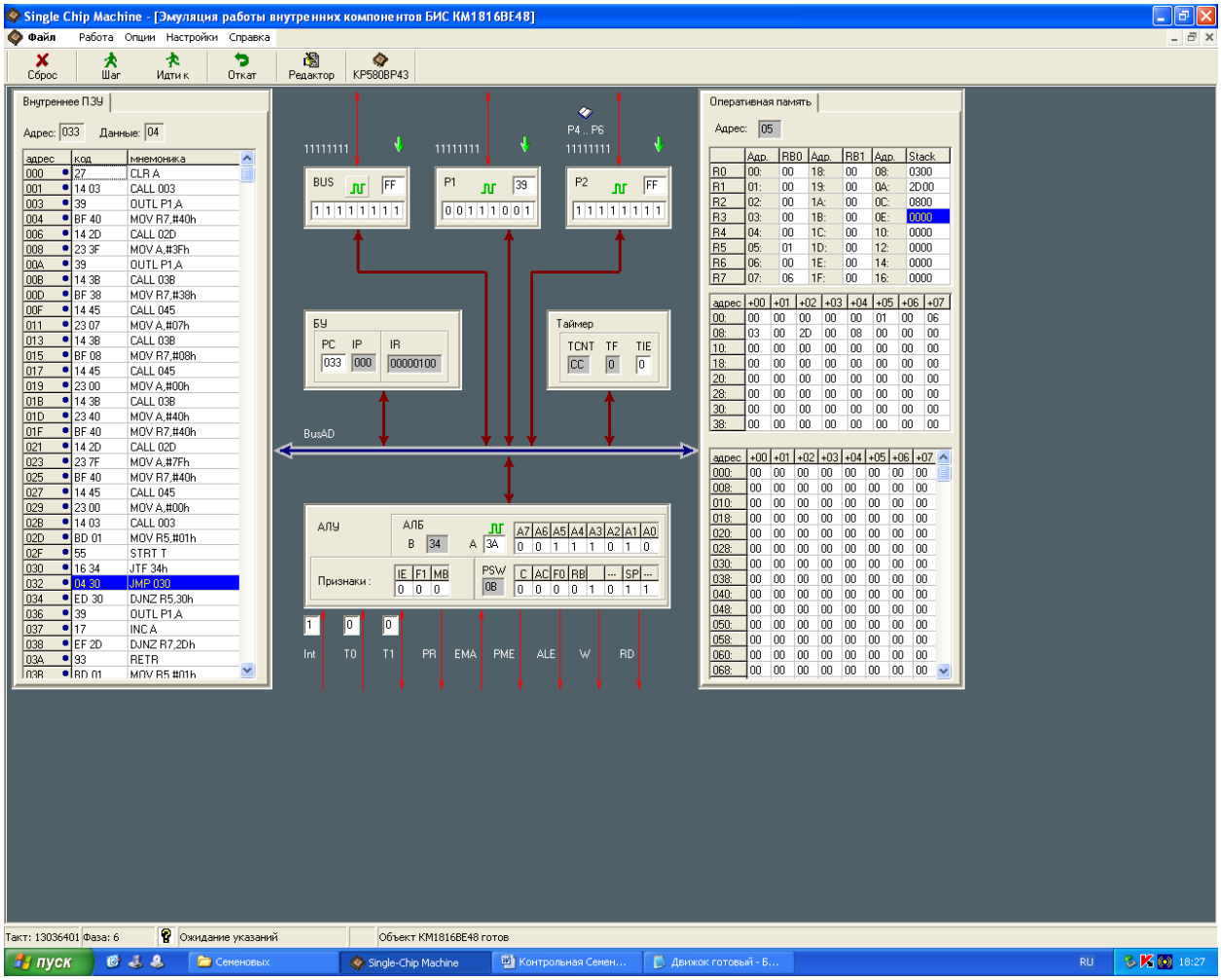
**$t_1 = t_2 = 5$ с**
**$\omega_{max}$ = 3F; 0,5 $\omega_{max}$ = 1F; 0 $\omega_{max}$ = 00 – неподвижен.**

**$\tau_1$ = 80 мс**
**$\tau_2$ = 5 с** **– это три подпрограммы формирования временной задержки**
**$\tau_3$ = 160 мс**

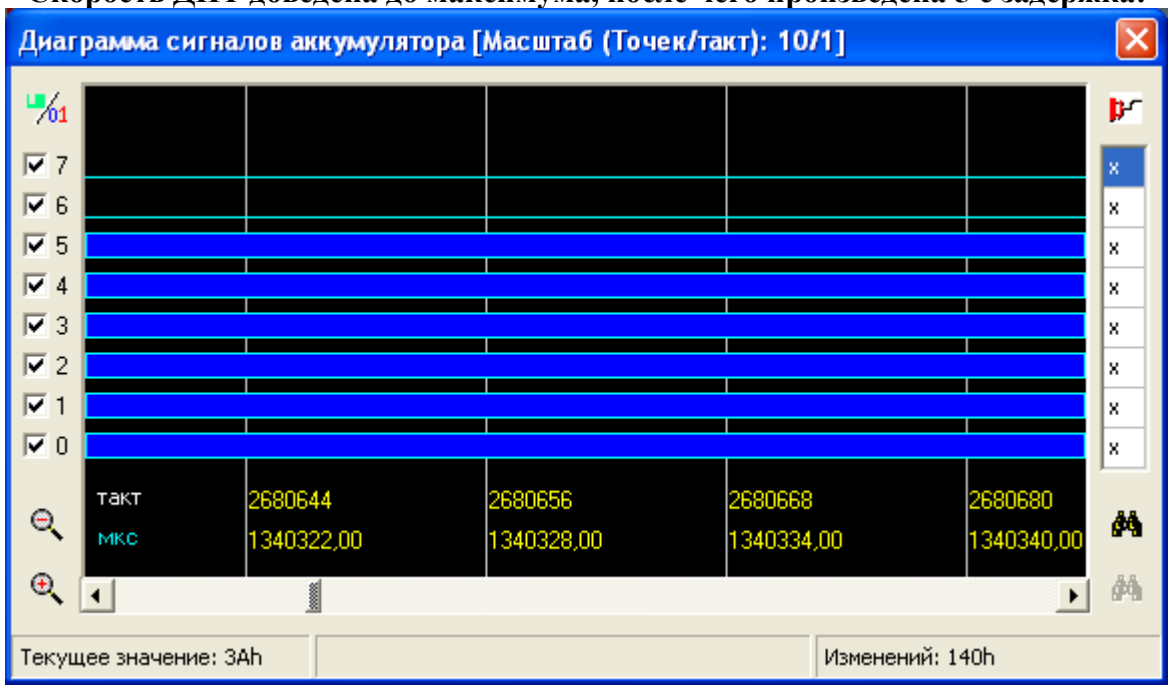
**Создаем программу управления ДПТ:**



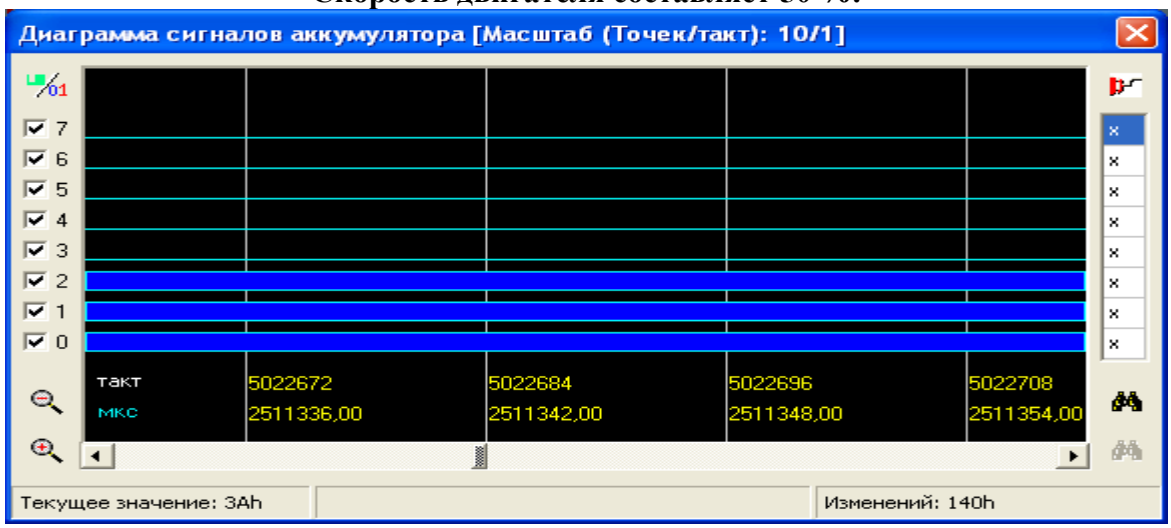

```
            Clr    A           ;очи□аем значение аккумулятора
            Call   M3          ;вызов подпрограммы i7 (уменьшение скорости до 0, обратное вра□ение)
      M3:   Outl   P1, A       ;выводим значение акумулятора в порт i1
            Mov    R7, #64     ;ввод в я память R7 зачения 64
            Call   M0          ;вызов подпрограммы i0 (увеличение скорости до максимальной за 5сек)
            Mov    A, #3FH     ;ввод в аккумулятор значения 3F (максимальная скоость,прямое вра□ение)
            Outl   P1, A       ;выводим значение акумулятора в порт i1
            Call   M4          ;вызов подпрограммы i4 (задержка 5 секунд)
            Mov    R7, #56     ;ввод в я память R7 зачения 56
            Call   M7          ;вызов подпрограммы i7 (уменьшение скорости до 0, обратное вра□ение)
            Mov    A, #07H     ;ввод в аккумулятор значения 3F (максимальная скоость,прямое вра□ение)
            Call   M4          ;вызов подпрограммы i4 (задержка 5 секунд)
            Mov    R7, #08     ;ввод в я память R7 зачения 08
            Call   M7          ;вызов подпрограммы i7 (уменьшение скорости до 0, обратное вра□ение)
            Mov    A, #00H     ;ввод в аккумулятор значения 3F (максимальная скоость,прямое вра□ение)
            Call   M4          ;вызов подпрограммы i4 (задержка 5 секунд)
            Mov    A, #40H     ;ввод в аккумулятор значение 40h
            Mov    R7, #64     ;ввод в я память R7 зачения 64
            Call   M0          ;вызов подпрограммы i0 (увеличени скорости до макс. за 5 сек, обратное вра□ени
            Mov    A, #7FH     ;ввод в аккумулятор значение 4Fh
            Mov    R7, #64     ;ввод в я память R7 зачения 64
            Call   M7          ;вызов подпрограммы i7 (уменьшение скорости до 0, обратное вра□ение)
            Mov    A, #00H     ;ввод в аккумулятор значения 3F (максимальная скоость,прямое вра□ение)
            Call   M3



;*************************************************************************************************************
;                                инкремент с шагом 80 мс (в сумме 5 сек)
;*************************************************************************************************************
M0:         Mov    R5, #01H    ;передаем в R5 число переполнений таймера(1переполнение=20мс????)
            Strt   T           ;запуск таймера
M1:         Jtf    M2          ;таймер переполнен, переход в i2
            Jmp    M1          ;иначе возврат к i1
M2:         Djnz   R5, M1      ;декримент R5 и переход к i1 если R5 не равно 0
            Outl   P1, A       ;выводим значение акумулятора в порт i1
            Inc    A           ;инкремент аккумулятора
            Djnz   R7, M0      ;декримент R7 и переход к i0, если R7 не равно 0
            Retr               ;возврат в тело программу

M4:         Mov    R5, #01H    ;передаем в R5 число переполнений таймера(1переполнение=20мс????)
            Strt   T           ;запуск таймера
M5:         Jtf    M6          ;таймер переполнен, переход в i6
            Jmp    M5          ;иначе возврат к i5
M6:         Djnz   R5, M5      ;декримент R5 и переход к i5 если R5 не равно 0
            Retr               ;возврат в тело программу

M7:         Mov    R5, #01H    ;передаем в R5 число переполнений таймера(1переполнение=20мс????)
            Strt   T           ;запуск таймера
M8:         Jtf    M9          ;таймер переполнен, переход в i9
            Jmp    M8          ;иначе возврат к i8
M9:         Djnz   R5, M8      ;декримент R5 и переход к i8 если R5 не равно 0
            Outl   P1, A       ;выводим значение акумулятора в порт i1
            Dec    A           ;декримент аккумулятора
            Djnz   R7, M7      ;декримент R7 и переход к i7, если R7 не равно 0
            Retr               ;возврат в тело программу
```



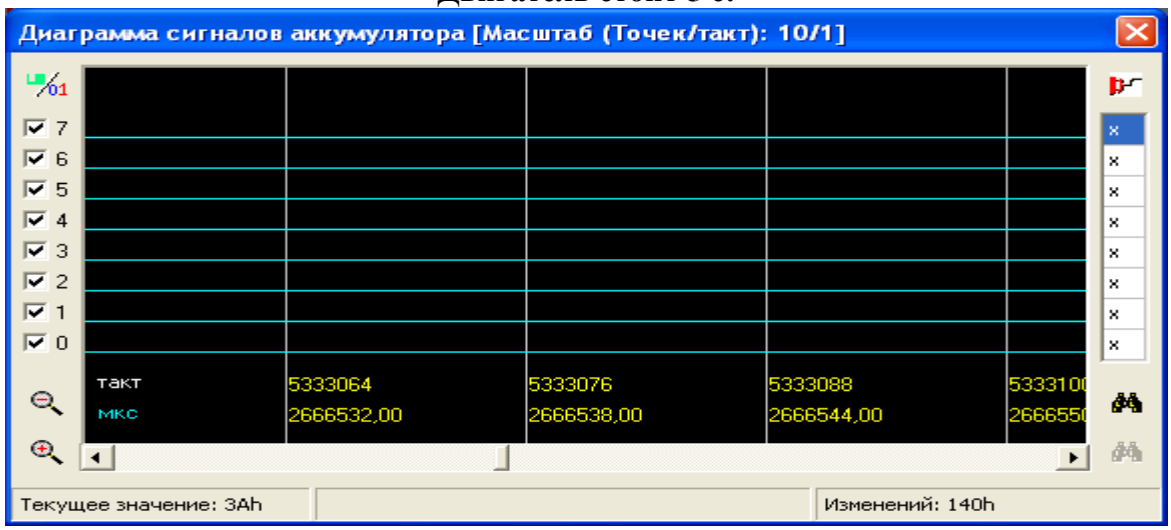

**Скорость ДПТ доведена до максимума, после чего произведена 5 с задержка:**



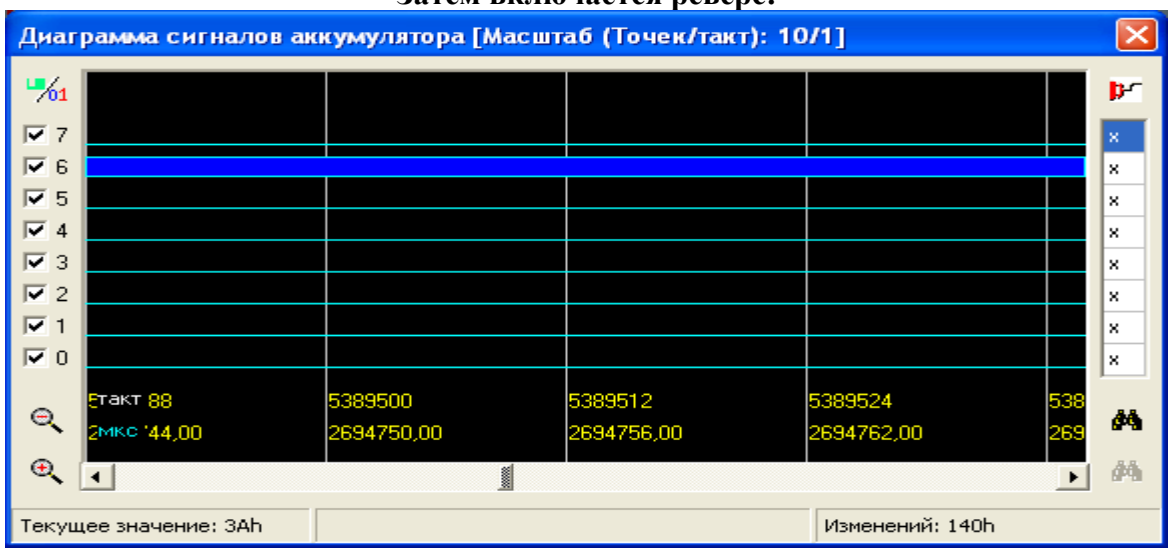
**Скорость двигателя составляет 50 %:**



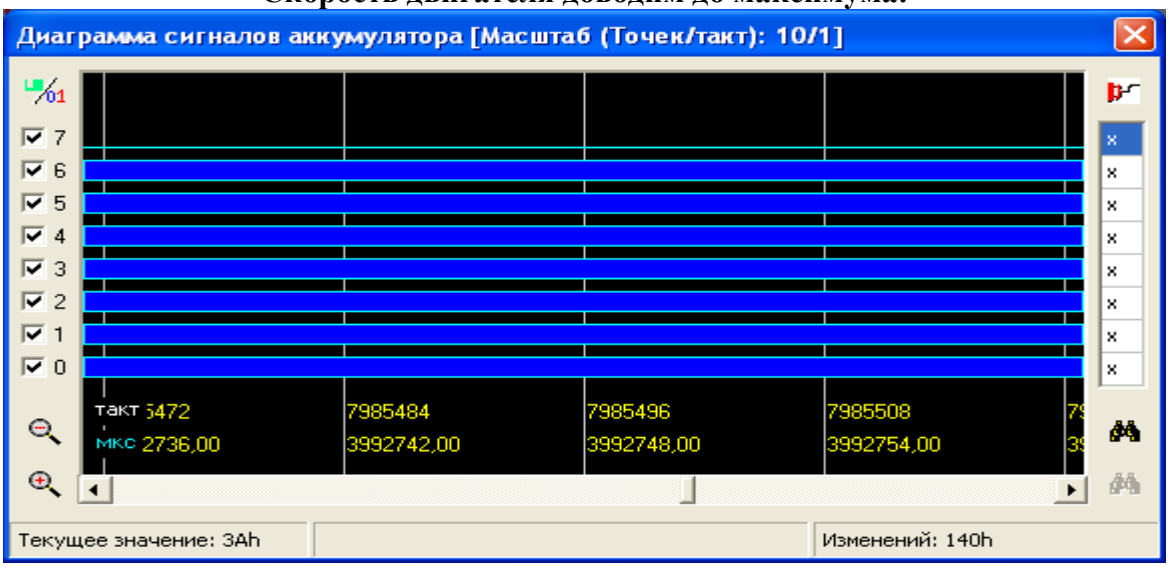

**Двигатель стоит 5 с:**



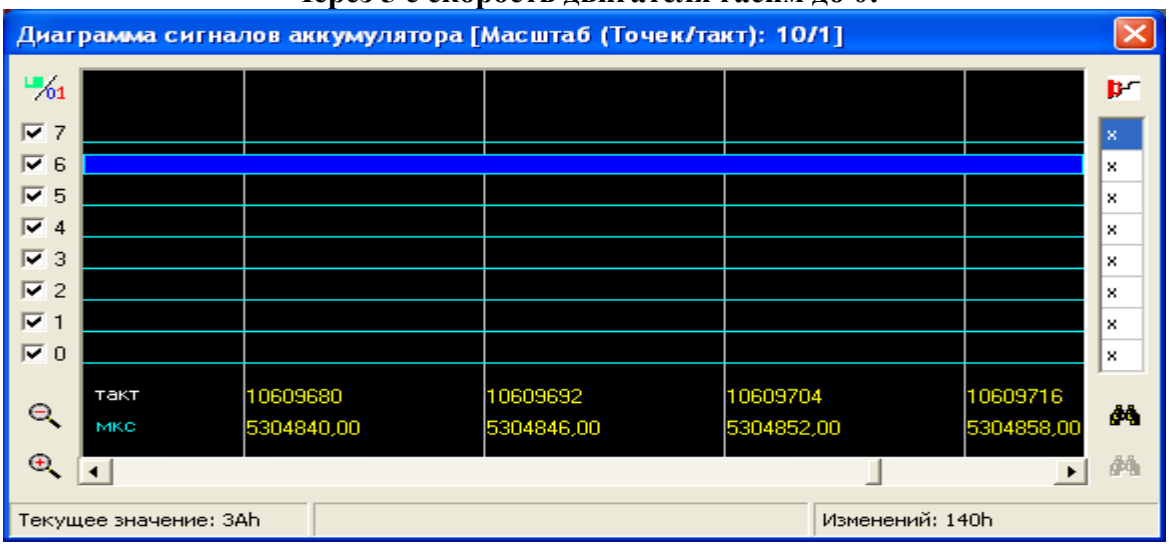

**Затем включается реверс:**



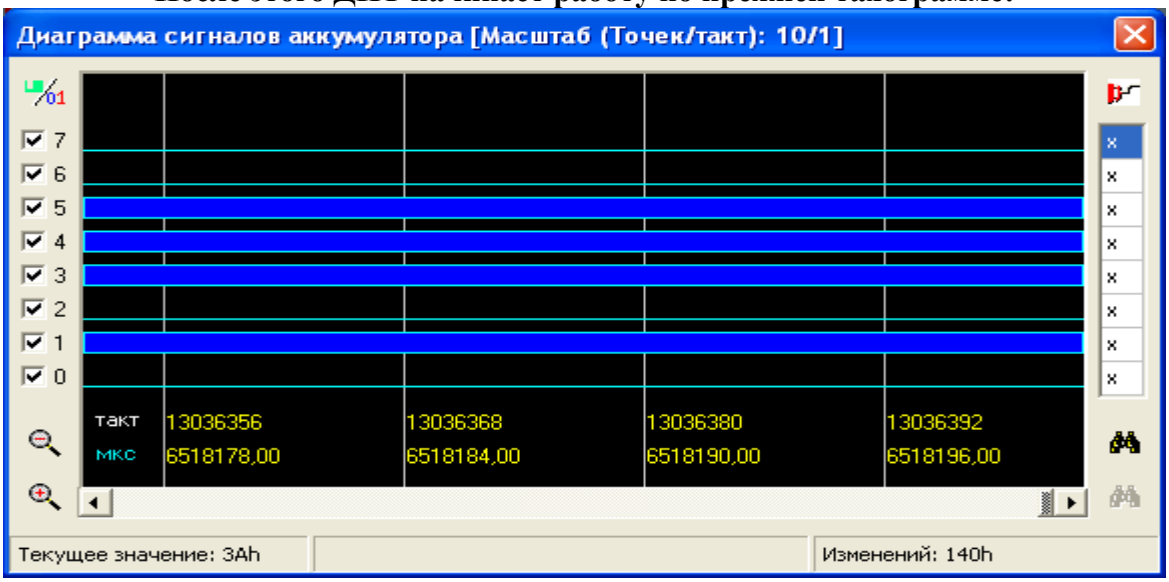
**Скорость двигателя доводим до максимума:**




**Через 5 с скорость двигателя гасим до 0:**




**После этого ДПТ начинает работу по прежней тахограмме:**



### *Задание № 3. Программные модели элементов цифровой техники*

***Цель:*** формирование у студентов навыков создания программных моделей элементов цифровой техники (счетчиков, дешифраторов, мультиплексоров, триггеров и т. д.). Каждый студент получает индивидуальное задание и выполняет в соответствии с двумя вышерассмотренными примерами.

***Задание.*** Используя систему моделирования SCM 1.38, реализовать на языке ассемблера программную модель четырехразрядного двоичного суммирующего счетчика.

***Пояснения к выполнению задания.*** Суммирование производится по переднему фронту импульсов, поступающих с микропереключателя, подключенного к разряду P1.0 порта P1. Необходимо предусмотреть предустановку счетчика потенциальным сигналом низкого уровня по входу разряда P1.7 порта P1. Результаты модели счетчика вывести в порт расширителя ввода/вывода P.4 (микросхема КР580ВР43).

# 8. ВОПРОСЫ К ЗАЧЕТУ

1. Информационная модель микропроцессорной системы управления.
2. Особенности проектирования микропроцессорных систем управления объектом.
3. Структура МП-системы управления.
4. Особенности разработки аппаратурных средств МП-систем.
5. Особенности разработки прикладного программного обеспечения.
6. Микроконтроллеры на базе ядра i8051. Структурная схема МК48.
7. Слово состояния МК48.
8. Организация памяти программ и памяти данных в МК48.
9. Структура информационных связей в МК48. способы адресации.
10. Организация портов ввода/вывода.
11. Организация системной шины в МК48.
12. Организация таймера/счетчика.
13. Организация системы прерываний в МК48.
14. Процедуры и подпрограммы.
15. Правила записи программ на языке ассемблера.
16. Отладка прикладного программного обеспечения.
17. Организация взаимодействия МК с объектом управления.
18. Ввод информации с датчиков:

– опрос двоичного датчика;
– ожидание события;
– устранение дребезга контактов;
– подсчет числа импульсов;
– опрос группы двоичных датчиков.

19. Вывод управляющих сигналов из МК:

– формирование статических сигналов;
– формирование импульсных сигналов.

20. Реализация функций времени:

– программное формирование временной задержки;
– формирование временной задержки на основе таймера.

21. Сопряжение МК с медленно действующим АЦП.
22. Сопряжение МК с быстродействующим АЦП.
23. Сопряжение МК с семисегментным индикатором.
24. Организация последовательного интерфейса RS-232.
25. Организация последовательного интерфейса RS-485.

# 9. ВОПРОСЫ К ЭКЗАМЕНУ ПО ДИСЦИПЛИНЕ

1. Назначение, особенности и отличия МПСУ.
2. Классификация и развитие МПСУ.
3. Структурные построения АСУ предприятия и АСУ ТП, их особенности.
4. Особенности периферийных устройств, используемых в промышленной автоматике.
5. Требования, предъявляемые к корпусам МПСУ.
6. Особенности программного обеспечения МПСУ.
7. Пути повышения надежности МПСУ.
8. Достоинства архитектуры IBM PC, выбор между ПК и ПЛК.
9. Организация внешних интерфейсов.
10. Канал общего пользования.
11. Основные особенности интерфейса RS-485.
12. Конфигурация систем на базе интерфейса RS-485.
13. Способы защиты аппаратуры сети от помех.
14. Методика выбора кабеля для последовательных интерфейсов.
15. Назначение, особенности и характеристики полевых шин.
16. Промышленный интерфейс CAN.
17. Основы обработки дискретных сигналов. Схемы сопряжения.
18. Принципы построения модулей дискретного ввода/вывода. Типы дискретных входов.
19. Основы обработки аналоговых сигналов. Параллельная и последовательная обработка.
20. Выбор частоты опроса датчиков и частот среза входных/выходных фильтров.
21. Принципы построения модулей аналогового ввода. Состав, назначение, элементная база.
22. Принципы построения модулей аналогового вывода. Состав, назначение, элементная база.
23. Сопряжение ЦАП с МПУ.
24. Сопряжение медленных АЦП с МПУ.
25. Сопряжение быстродействующих АЦП с МПУ.
26. Пути повышения надежности и помехозащищенности контура измерения.
27. УСО. Назначение, классификация.
28. УСО. Особенности и применение.
29. Принципы построения дискретных УСО.
30. Принципы построения аналоговых УСО.
31. Принципы построения интеллектуальных УСО.
32. Программируемые контроллеры. Классификация, особенности, область применения.
33. Микроконтроллеры. Назначение, особенности и характеристики.
34. Микроконтроллеры на базе ядра i8051. Организация системных шин.
35. Микроконтроллеры на базе ядра i8051. Сопряжение с внешними устройствами через порты индивидуальных линий.
36. PIC-контроллеры. Архитектура. Система команд. Защита кода.

37. PIC-контроллеры. Внутренние устройства.

38. Микроконтроллер MSC96. Особенности архитектуры, АЦП, генератор ШИМ.

39. Цифровые сигнальные процессоры. Назначение, особенности и характеристики.

40. Цифровые сигнальные процессоры. Особенности архитектуры.

41. Обобщенная структура модуля аналогового ввода/вывода на базе ЦСП.

42. Обобщенная структура АСУ ТП. Состав и назначение технических средств АСУ ТП.

43. Принципы построения централизованных систем управления.

44. Принципы построения распределенных систем управления.

45. Способ построения отказоустойчивой системы управления.

## БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Кузин, А. В.* Микропроцессорная техника [Текст] : учебник / А. В. Кузин, М. А. Жаворонков. – М. : Академия, 2004. – 304 с.

2. Микропроцессорные системы управления. Самостоятельная работа студентов [Текст] : метод. указ. для подготовки дипломированного специалиста по направлению 651900 «Автоматизация и управление», спец. 220301 «Автоматизация технологических процессов и производств» / В. И. Семёновых ; СЛИ. – Сыктывкар : СЛИ, 2008. – 24 с.

3. Положении о дипломном проектировании : ч. 1. Единые требования к текстовым документам / сост. В. А. Паршукова, А. А. Митюшов ; СЛИ. – Сыктывкар, 2009. – 36 с.

4. Схемотехника электронных систем. Микропроцессоры и микроконтроллеры [Текст] : учебник / В. И. Бойко [и др.]. – СПб. : БХВ-Петербург, 2004. – 464 с.

5. Электроника и микропроцессорные средства (аналоговая и цифровая техника) [Текст] : сб. описаний виртуальных лабораторных и практ. для студ. спец. 110301, 110302, 150405, 190603 всех форм обучения / сост. К. Ф. Майер. – Сыктывкар : СЛИ, 2006. – 76 с.

6. Электроника и микропроцессорные средства [Текст] : учеб.-метод. комплекс по дисц. для студ. спец. 110102 «Электрификация и автоматизация сельского хоз-ва» всех форм обучения / сост. К. Ф. Майер. – Сыктывкар : СЛИ, 2006. – 116 с.

# ПРИЛОЖЕНИЕ 1
# СИСТЕМА КОМАНД МК48

Группа команд передачи управления
(Т – тип команды, Б – формат в байтах, Ц – число машинных циклов)

| Название команды | Мнемокод | КОП | Т Б Ц | Операция |
|---|---|---|---|---|
| Безусловный переход ad11 | JMP ad 11 | $a_{10}a_9a_8$ 00100 | 3 2 2 | (PC__0-10__) <- (PC__11__) <- DBF |
| Косвенный переход в текущей странице | JMPP @A | 10110011 | 1 1 2 | (PC__0-7__) <- ((A)) |
| Декремент регистра и переход, если не нуль | DJNZ Rn,ad | 11101rrr | 4 2 2 | (Rn)<-(Rn)-1, если (Rn)/0, то (PC__0-7__) <- ad, иначе (PC)<-(PC)+2 |
| Переход, если есть перенос | JC ad | 11110110 | 4 2 2 | если (C)=1, то (PC__0-7__) <- ad, иначе (PC)<-(PC)+2 |
| Переход, если нет переноса | JNC ad | 11100110 | 4 2 2 | если (C)=0, то (PC__0-7__) <- ad, иначе (PC)<-(PC)+2 |
| Переход, если аккумулятор содержит нуль | JZ ad | 11000110 | 4 2 2 | если (A)=0, то (PC__0-7__) <- ad, иначе (PC)<-(PC)+2 |
| Переход, если аккумулятор содержит не нуль | JNZ ad | 10010110 | 4 2 2 | если (A)=0, то (PC__0-7__) <- ad, иначе (PC)<-(PC)+2 |
| Переход, если на входе T0 высокий уровень | JT0 ad | 00110110 | 4 2 2 | если (T0)=1, то (PC__0-7__) <- ad, иначе (PC)<-(PC)+2 |
| Переход, если на входе T0 низкий уровень | JNT0 ad | 00100110 | 4 2 2 | если (T0)=0, то (PC__0-7__) <- ad, иначе (PC)<-(PC)+2 |
| Переход, если на входе T1 высокий уровень | JT1 ad | 01010110 | 4 2 2 | если (T1)=1, то (PC__0-7__) <- ad, иначе (PC)<-(PC)+2 |
| Переход, если на входе T1 низкий уровень | JNT1 ad | 01000110 | 4 2 2 | если (T1)=0, то (PC__0-7__) <- ad, иначе (PC)<-(PC)+2 |
| Переход, если флаг F0 установлен | JF0 ad | 10110110 | 4 2 2 | если (F0)=1, то (PC__0-7__) <- ad, иначе (PC)<-(PC)+2 |
| Переход, если флаг F1 установлен | JF1 ad | 01110110 | 4 2 2 | если (F1)=1, то (PC__0-7__) <- ad, иначе (PC)<-(PC)+2 |
| Переход, если флаг переполнения таймера установлен | JTF ad | 00010110 | 4 2 2 | если TF=1,то TF=0, (PC__0-7__) <- ad, иначе (PC)<-(PC)+2 |
| Переход, если на входе ЗПР низкий уровень | JNI ad | 10000110 | 4 2 2 | если ЗПР = 0, то (PC__0-7__) <- ad, иначе (PC)<-(PC)+2 |

| Название команды | Мнемокод | КОП | Т Б Ц | Операция |
|---|---|---|---|---|
| Переход, если бит аккумулятора ра вен единице (b__0-7__) | JBb ad | bbb10010 | 4 2 2 | если (Bb)=1,<br>то (PC__0-7__) <- ad,<br>иначе (PC)<-(PC)+2 |
| Вызов подпрограммы | CALL ad11 | $a_{10}a_9a_8$ 10100 | 3 2 2 | ((SP))<-(PC),<br>(PSW__4-7__)<br>(SP)<-(SP)+1,<br>(PC__11__)<-DBF<br>(PC__0-10__)<-ad11 |
| Возврат из подпрограммы | RET | 10000011 | 1 1 2 | (SP)<-(SP)-1<br>(PC)<-((SP)) |
| Возврат из подпрограммы и восстановление ССП | RETR | 10010011 | 1 1 2 | (SP)<-(SP)-1<br>(PC)<-((SP))<br>(PSW__4-7__)<-((SP)) |

## Группа команд управления режимами работы МК48

| Название команды | Мнемокод | КОП | Т Б Ц | Операция |
|---|---|---|---|---|
| Запуск таймера | STRT T | 01010101 | 1 1 1 | |
| Запуск счетчика | STPT CNT | 01000101 | 1 1 1 | |
| Останов таймера/счетчика | STOP TCNT | 01100101 | 1 1 1 | |
| Разрешение прерывания от таймера/счетчика | EN TCNTI | 00100101 | 1 1 1 | |
| Запрещение прерывания от таймера/счетчика | DIS TCNTI | 00110101 | 1 1 1 | |
| Разрешение внешнего прерывания | EN I | 00000101 | 1 1 1 | |
| Запрещение внешнего прерывания | DIS I | 00010101 | 1 1 1 | |
| Выбор нулевого банка регистров | SEL RB0 | 11000101 | 1 1 1 | (BS)<-0 |
| Выбор первого банка регистров | SEL RB1 | 11010101 | 1 1 1 | (BS)<-1 |
| Выбор первого банка ПП | SEL MB1 | 11110101 | 1 1 1 | (DBF)<-1 |
| Выбор нулевого банка ПП | SEL MB0 | 11100101 | 1 1 1 | (DBF)<-0 |
| Разрешение выдачи синхросигнала на выходе ТО | ENTO CLC | 01110101 | 1 1 1 | ТО-синхросигнал (2 мгц) |
| Холостая команда | NOP | 00000000 | 1 1 1 | (PC)<-(PC)+1 |

## Группа команд логических операций

| Название команды | Мнемокод | КОП | Т Б Ц | Операция |
|---|---|---|---|---|
| Логическое И регистра и аккумулятора | ANL A,Rn | 01011rrr | 1 1 1 | (A)<-(A)^(Rn) |
| Логическое И байта из РПД и аккумулятора | ANL A,@Ri | 0101000i | 1 1 1 | (A)<-(A)^((Ri)) |
| Логическое И константы и аккумулятора | ANL A,#d | 01010011 | 2 2 2 | (A)<-(A)^#d |
| Логическое ИЛИ регистра и аккумулятора | ORL A,Rn | 01001rrr | 1 1 1 | (A)<-(A)v(Rn) |
| Логическое ИЛИ байта из РПД и аккумулятора | ORL A,@Ri | 0100000i | 1 1 1 | (A)<-(A)v((Ri)) |
| Логическое ИЛИ константы и аккумулятора | ORL A,#d | 01000011 | 2 2 2 | (A)<-(A)v#d |

| Название команды | Мнемокод | КОП | Т Б Ц | Операция |
|---|---|---|---|---|
| Исключающее ИЛИ регистра и аккумулятора | XRL A,Rn | 11011rrr | 1 1 1 | (A)<-(A)v(Rn) |
| Исключающее ИЛИ байта из РПД и аккумулятора | XRL A,@Ri | 1101100i | 1 1 1 | (A)<-(A)v((Ri)) |
| Исключающее ИЛИ константы и аккумулятора | XRL A,#d | 11010011 | 2 2 2 | (A)<-(A)v#d |
| Сброс аккумулятора | CLR A | 00100111 | 1 1 1 | (A) <- 0 |
| Инверсия аккумулятора | CPL A | 00110111 | 1 1 1 | (A) <- (A) |
| Обмен тетрад в аккумуляторе | SWAP A | 01000111 | 1 1 1 | (A__0-3__)<->(A__4-7__) |
| Циклический сдвиг влево аккумулятора | RL A | 11100111 | 1 1 1 | (A__n+1__)<-(A__n__), n = 0...6<br>(A__0__) <- (A__7__) |
| Сдвиг влево аккумулятора через перенос | RLC A | 11110111 | 1 1 1 | (A__n+1__)<-(A__n__), n = 0...6<br>(A__0__)<-(C);<br>(C)<-(A__7__) |
| Циклический сдвиг вправо аккумулятора | RR A | 01110111 | 1 1 1 | (A__n__)<-(A__n+1__), n = 0...6<br>(A__7__) <- (A__0__) |
| Сдвиг вправо аккумулятора через перенос | RRC A | 01100111 | 1 1 1 | (A__n__)<-(An__+1__), n = 0...6<br>(A__7__)<-(C);<br>(C)<-(A__0__) |
| Логическое И константы порта Pp(p=1,2) | ANL Pp,#d | 100110pp | 2 2 2 | (Pp)<-(Pp)^#d |
| Логическое И константы и порта BUS | ANL BUS,#d | 10011000 | 2 2 2 | (BUS)<-(BUS)^#d |
| Логическое И аккумулятора порта Pp(p=4..7) | ANLD Pp,A | 100111pp | 1 1 2 | (Pp)<-(Pp)^(A__0-3__) |
| Логическое ИЛИ константы и порта Pp(p=1,2) | ORL Pp,#d | 100010pp | 2 2 2 | (Pp)<-(Pp)v#d |
| Логическое ИЛИ константы и порта BUS | ORL BUS,#d | 10001000 | 2 2 2 | (BUS)<-(BUS)v#d |
| Логическое ИЛИ аккумулятора и порта Pp (p=4..7) | ORLD Pp,A | 100011pp | 1 1 2 | (Pp)<-(Pp)v(A__0-3__) |
| Сброс переноса | CLR C | 10010111 | 1 1 1 | (C) <- 0 |
| Сброс флага F0 | CLR F0 | 10000101 | 1 1 1 | (F0) <- 0 |
| Сброс флага F1 | CLR F1 | 10100101 | 1 1 1 | (F1) <- 0 |
| Инверсия переноса | CPL C | 10100111 | 1 1 1 | (C) <- (C) |
| Инверсия флага F0 | CPL F0 | 10010101 | 1 1 1 | (F0) <- (F0) |
| Инверсия флага F1 | CPL F1 | 10110101 | 1 1 1 | (F1) <- (F1) |

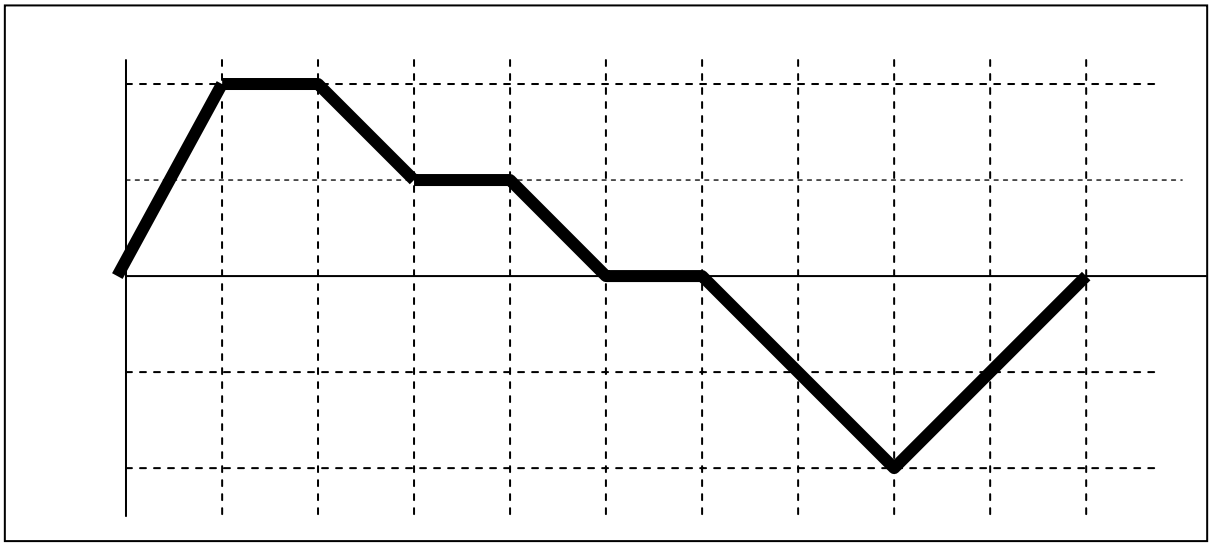
Группа команд пересылки данных

| Название команды | Мнемокод | КОП | Т Б Ц | Операция |
|---|---|---|---|---|
| Пересылка регистра в аккум. | MOV A,Rn | 11111rrr | 1 1 1 | (A) <- (Rn) |
| Пересылка байта из РПД в аккумулятор | MOV A,@Ri | 1111000i | 1 1 1 | (A) <- ((Ri)) |
| Пересылка непосред. операнда в аккумулятор | MOV A,#d | 00100011 | 2 2 2 | (A) <- #d |
| Пересылка аккум. в регистр | MOV Rn,A | 10101rrr | 1 1 1 | (Rn) <- (A) |
| Пересылка непосред. операнда в регистр | MOV Rn,#d | 10111rrr | 2 2 2 | (Rn) <- #d |
| Пересылка аккум. в РПД | MOV @Ri,A | 1010000i | 1 1 1 | ((Ri)) <- (A) |
| Пересылка непосред. операнда в РПД | MOV @Ri,#d | 1011000i | 2 2 2 | ((Ri)) <- #d |
| Пересылка ССП в аккум. | MOV A,PSW | 11000111 | 1 1 1 | (A) <- (PSW) |
| Пересылка аккум. в ССП | MOV PSW,A | 11010111 | 1 1 1 | (PSW) <- (A) |
| Пересылка содержимого таймера/счетчика в аккум. | MOV A,T | 01000010 | 1 1 1 | (A) <- (T) |
| Пересылка аккумулятора в таймер/счетчик | MOV T ,A | 01100010 | 1 1 1 | (T) <- (A) |
| Пересылка байта из ВПД в аккумулятор | MOV XA,@Ri | 1000000i | 1 1 2 | (A) <- ((Ri)) |
| Пересылка аккум. в ВПД | MOV X@Ri,A | 1001000i | 1 1 2 | ((Ri)) <- (A) |
| Пересылка байта из текущей строки программной памяти в аккумулятор | MOV PA,@A | 10100011 | 1 1 2 | ((PC__0-7__)) <- (A)<br>(A) <- ((PC)) |
| Пересылка байта из третьей строки программной памяти в аккумулятор | MOV P3A,@A | 11100011 | 1 1 2 | ((PC__0-7__)) <- (A)<br>((PC__8-11__))<-0011 |
| Обмен регистра с аккум. | XCH A,Rn | 00101rrr | 1 1 1 | (A) <-> (Rn) |
| Обмен аккум. с РПД | XCH A,@Ri | 0010000i | 1 1 1 | (A) <-> ((Ri)) |
| Обмен младших тетрад аккумулятора и байта РПД | XCHD A,@Ri | 0011000i | 1 1 1 | (A__0-3__)<->((Ri)__0-3__) |
| Пересылка данных из порта P__p__(p=1,2) в аккумулятор | IN A,P__p__ | 000010pp | 1 1 2 | (A) <- (P__p__) |
| Стробируемый ввод данных из порта BUS | INS A,BUS | 00001000 | 1 1 2 | (A) <- (BUS) |
| Пересылка аккумулятора порт P__p__ (p=1,2) | OVTL P__p__,A | 001110pp | 1 1 2 | (P__p__) <- (A) |
| Строб. вывод данных из аккумулятора в порт BUS | OVTL BUS,A | 00000010 | 1 1 2 | (BUS) <- (A) |
| Ввод тетрады из порта P__p__ (p=4...7) схемы расширителя | MOVD A,P__p__ | 000011pp | 1 1 2 | (A__0-3__) <- (P__p__)<br>(A__4-7__) <- 0000 |
| Вывод тетрады в порт P__p__ (p=4...7) схемы расширителя | MOVD P__p__,A | 001111pp | 1 1 2 | (P__p__) <- (A__0-3__) |

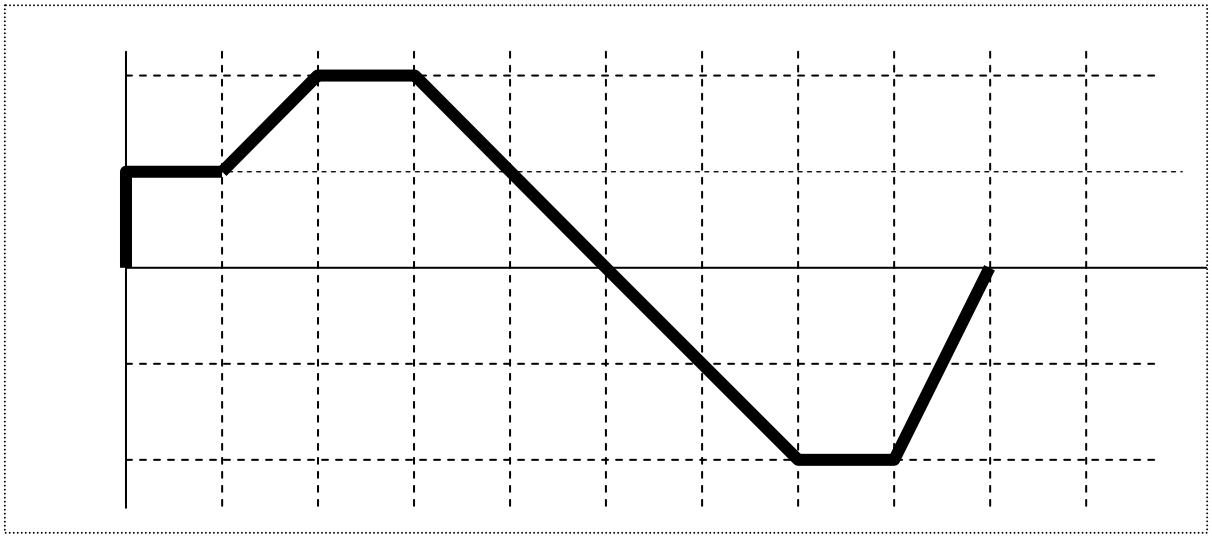
Группа команд арифметических операций

| Название команды | Мнемокод | КОП | Т Б Ц | Операция |
|---|---|---|---|---|
| Сложение регистра с аккумулятором | ADD A,R_n | 01101rrr | 1 1 1 | (A) <- (A)+(R_n) |
| Сложение байта из РПД с аккумулятором | ADD A,@R_i | 0110000i | 1 1 1 | (A) <- (A)+((R_i)) |
| Сложение константы с аккумулятором | ADD A,#d | 00000011 | 2 2 2 | (A) <- (A)+#d |
| Сложение регистра с аккумулятором и перенос | ADDC A,R__i__ | 01111rrr | 1 1 1 | (A)<-(A)+(R__n__)+(C) |
| Сложение байта из РПД с аккумулятором и перенос | ADDC A,@R__i__ | 0111000i | 1 1 1 | (A)<-(A)+((R__i__))+(C) |
| Сложение константы с аккумулятором и перенос | ADDC A,#d | 00010011 | 2 2 2 | (A)<-(A)+#d+(C) |
| Десятичная коррекция аккумулятора | DA A | 01010111 | 1 1 1 | если ((A__0-3__)>9)v((AC)=1),<br>то (A__0-3__)<-(A__0-3__)+6,<br>затем<br>если ((A__4__-__7__)>9)v((C)=1),<br>то (A__4-7__)<-(A__4-7__)+6 |
| Инкремент аккумулятора | INC A | 00010111 | 1 1 1 | (A) <- (A)+1 |
| Инкремент регистра | INC R__n__ | 00011rrr | 1 1 1 | (R__n__) <- (R__n__)+1 |
| Инкремент байта в РПД | INC @Ri | 0001000i | 1 1 1 | ((Ri)) <- ((Ri))+1 |
| Декремент аккумулятора | DEC A | 00000111 | 1 1 1 | (A) <- (A)-1 |
| Декремент регистра | DEC R__n__ | 11001rrr | 1 1 1 | (Rn) <- (Rn)-1 |

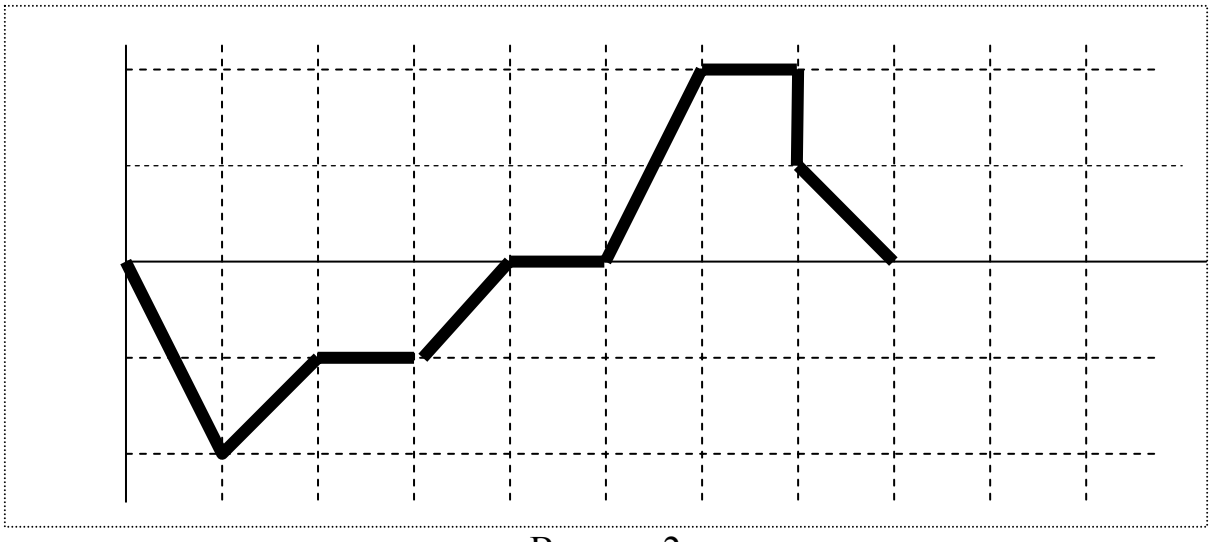
## ПРИЛОЖЕНИЕ 2
## ТАХОГРАММЫ ПО ВАРИАНТАМ



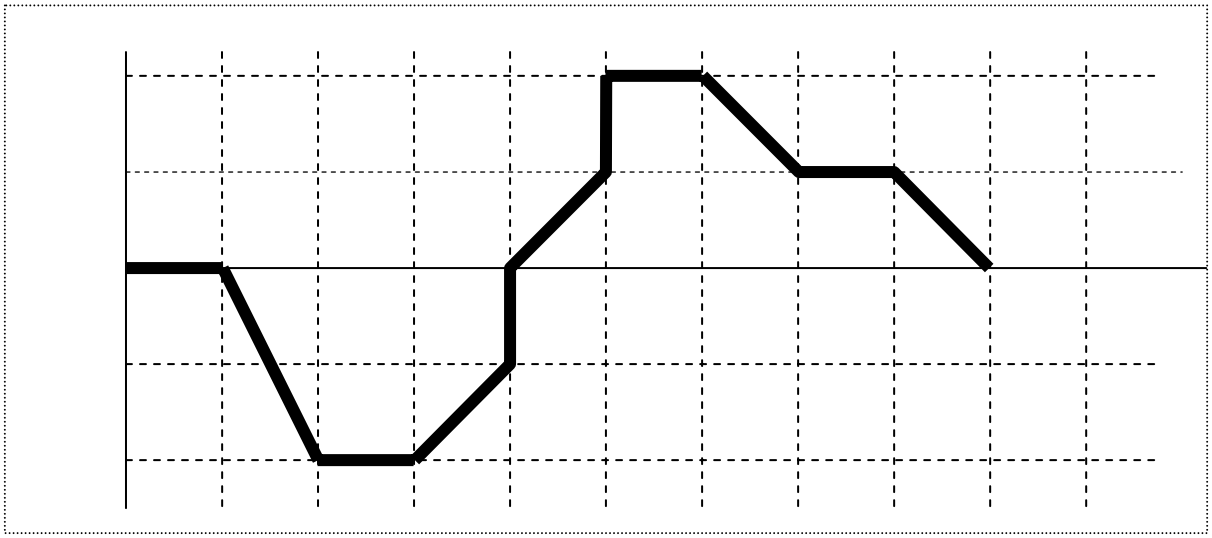
Вариант 0



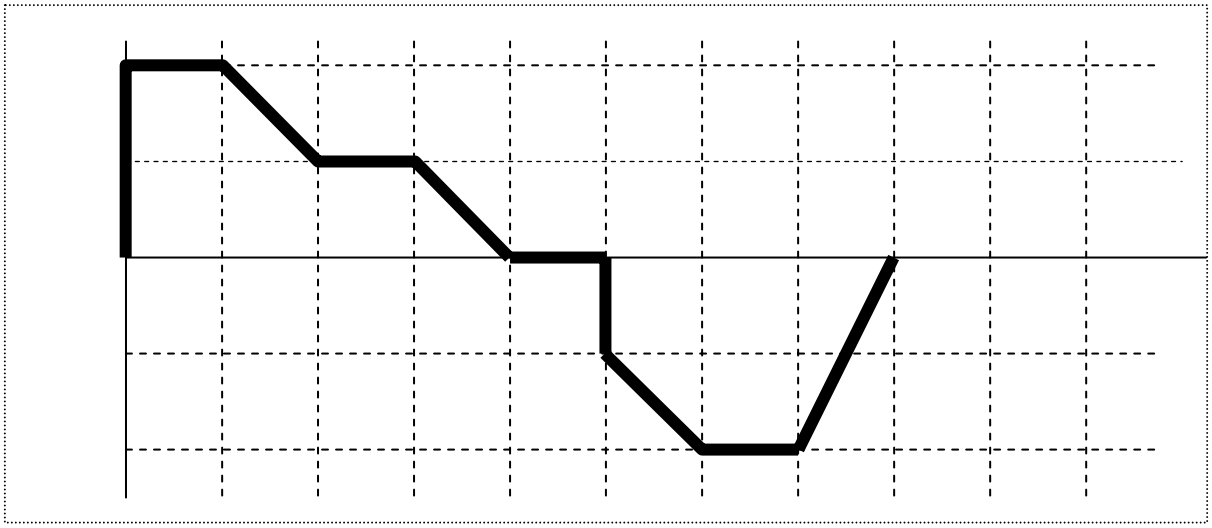
Вариант 1



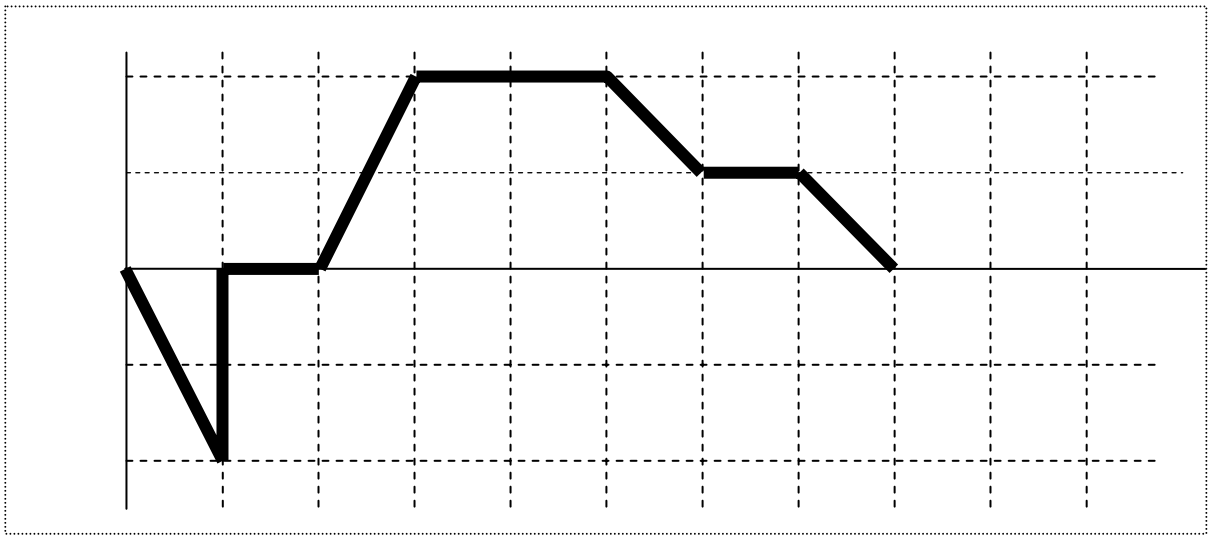
Вариант2



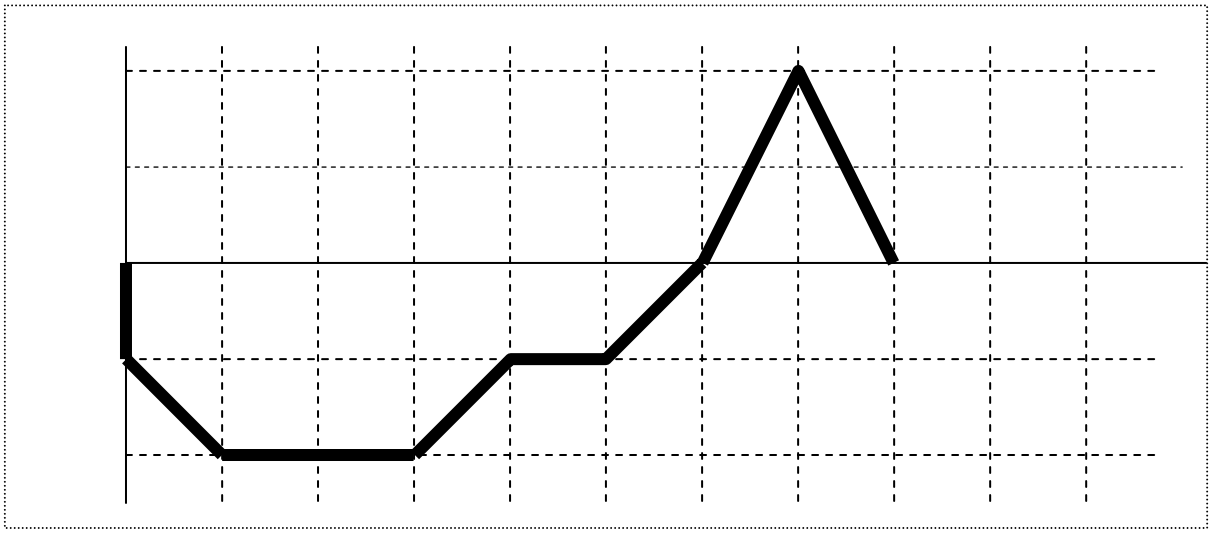
Вариант 3



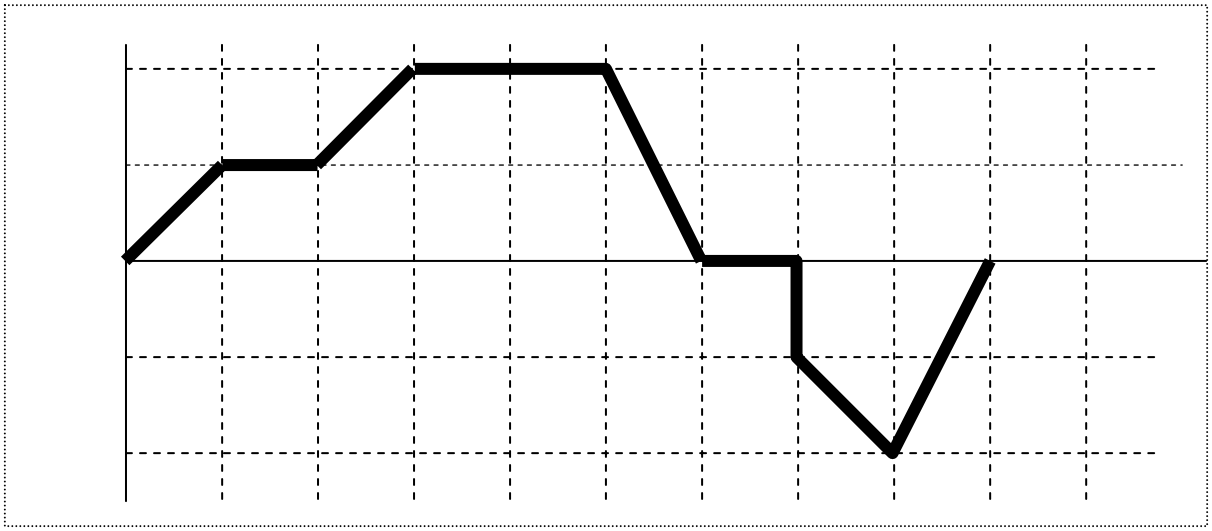
Вариант 4



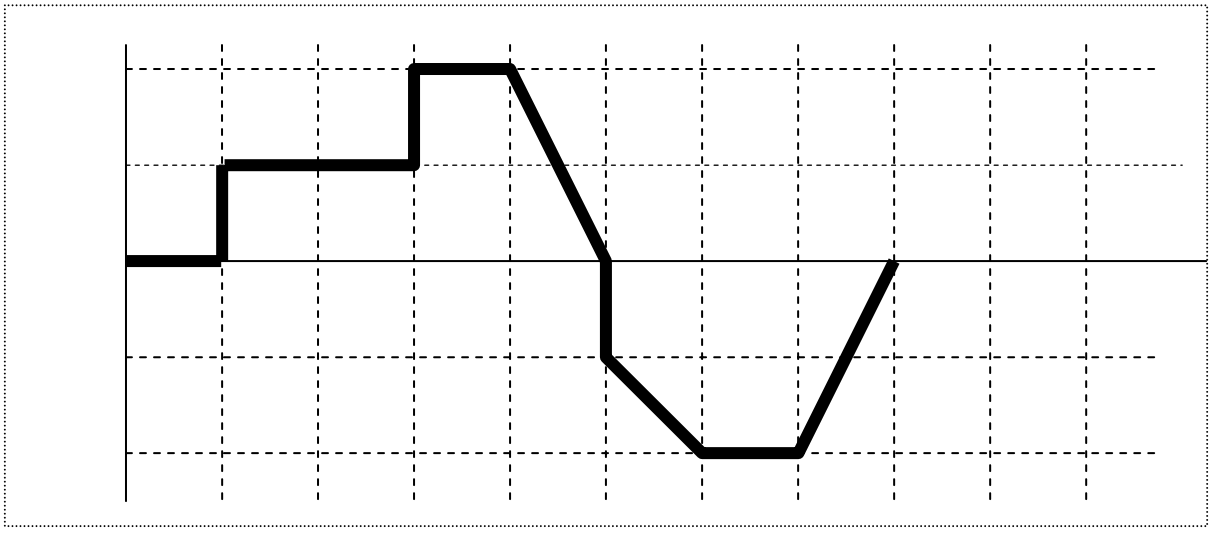
Вариант 5



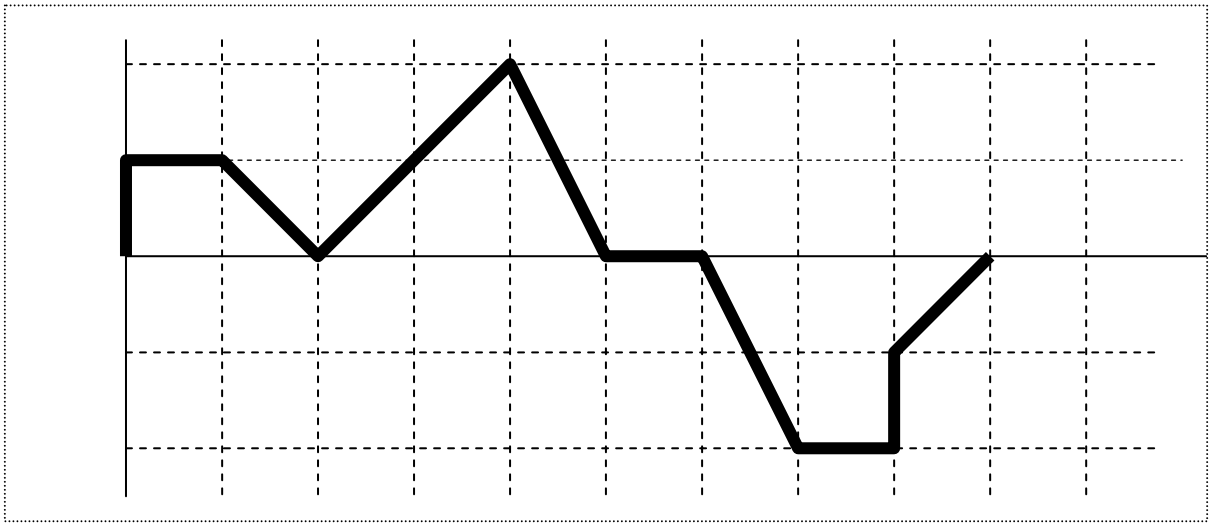
Вариант 6



Вариант 7



Вариант 8



Вариант 9