Национальный Технический Университет Украины

“Киевский Политехнический Институт”

Факультет Информатики и Вычислительной Техники

Кафедра вычислительной техники

Расчетно-графическая работа

по дисциплине “Архитектура компьютера”

**Выполнил:**

студент ІІІ-го курса

группы ИВ-81

Рудзик Д.И.

Руководитель:

Ткаченко В.В.

Киев 2010

Описание альбома

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *N п.п.* | *Формат* | *Обозначение* | | | *Наименование* | *К. листов* | | *№ экз.* | | *Примечание* | | |
|  |  |  | | | Документацияобщая |  | |  | |  | | |
|  |  |  | | |  |  | |  | |  | | |
|  |  |  | | | Разработана заново |  | |  | |  | | |
|  |  |  | | |  |  | |  | |  | | |
|  | А4 | ИАЛЦ 463617.001 ОА | | | Микропроцессорная система | 1 | |  | |  | | |
|  |  |  | | | *Описание альбома* |  | |  | |  | | |
|  |  |  | | |  |  | |  | |  | | |
|  | А4 | ИАЛЦ 463617002 ТЗ | | | Микропроцессорная система | 1 | |  | |  | | |
|  |  |  | | | *Техническое задание* |  | |  | |  | | |
|  |  |  | | |  |  | |  | |  | | |
|  | А3 | ИАЛЦ 463617.004 Е2 | | | Устройство ввода-вывода | 1 | |  | |  | | |
|  |  |  | | | *Схема функциональная* |  | |  | |  | | |
|  |  |  | | |  |  | |  | |  | | |
|  | А4 | ИАЛЦ 463617.003ПЗ | | | Пояснительная записка | 23 | |  | |  | | |
|  |  |  | | |  |  | |  | |  | | |
|  |  |  | | |  |  | |  | |  | | |
|  |  |  | | |  |  | |  | |  | | |
|  |  |  | | |  |  | |  | |  | | |
|  |  |  | | |  |  | |  | |  | | |
|  |  |  | | |  |  | |  | |  | | |
|  |  |  | | |  |  | |  | |  | | |
|  |  |  | | |  |  | |  | |  | | |
|  |  |  | | |  |  | |  | |  | | |
|  |  |  | | |  |  | |  | |  | | |
|  |  |  | | |  |  | |  | |  | | |
|  |  |  | | |  |  | |  | |  | | |
|  |  |  | | |  |  | |  | |  | | |
|  |  |  | | |  |  | |  | |  | | |
|  |  |  | | |  |  | |  | |  | | |
|  |  |  | | |  |  | |  | |  | | |
|  |  |  | | |  |  | |  | |  | | |
|  |  |  | | |  |  | |  | |  | | |
|  |  |  | | |  |  | |  | |  | | |
|  |  |  |  |  | *ИАЛЦ 463617.001 ОА* | | | | | | | |
|  |  |  |  |  |
| *Изм.* | *Лист* | *№ докум.* | *Подпись* | *Дата* |
| *Выполнил* | | *Рудзик Д.И.* |  |  | Микропроцессорная система  *Описание альбома* | | *Лит.* | | | | *Лист* | *Листов* |
| *Проверил* | | *Ткаченко В.В.* |  |  |  | |  |  | *1* | *1* |
|  | |  |  |  | *НТУУ «КПИ» ФИВТ*  *Гр. ИВ-81* | | | | | |
| *Н. контр.* | |  |  |  |
| *Зав. каф.* | |  |  |  |

Техническоезадание

Техническое задание на расчетную работу

Проектирование микропроцессорных систем

Студентка Потушинского Я. С.

ГруппыИВ-81

№ зачетной книжки

|  |  |
| --- | --- |
| Выбор элементной базы | 1816ВЕ51 |
| Организация шины | С объединенными шинами адреса и данных |
| Выбор системы команд | Комплексная |
| КПП, КПДП | Централизованный |
| Способ умножения, деления, разрядность операндов | Умножение – байтовое  Деление – байтовое  Разрядность – 16(деление – 8) |
| Количество ВУ | 116 |
| Функция |  |
| Вспомогательные порти, периферийные адаптеры | ВВ55, Р4,Р7 |
| Адреса для интерфейса внешнего устройства | Самостоятельно |
| Адреса для ППА | F0h, F1h, F2h, F3h |
| Внешняя память данных | по 64К(5 страниц) |
| Внешняя память программ | по 64К(2 страницы) |
| Функциональная схема | Устройство ввода + устройство вывода (интерфейсы) |

Задание выдал

Задание принял

Ткаченко В.В.

Пояснительная записка

СОДЕРЖАНИЕ

[1. Обзор микроконтроллера 3](#_Toc280296569)

[2. Обзор микропроцессорной системы 5](#_Toc280296570)

[2.1 Описание архитектуры микропроцессорной системы 5](#_Toc280296573)

[2.2 Память программ 5](#_Toc280296574)

[2.3 Память данных 7](#_Toc280296575)

[2.4 Внешние устройства и ППА 8](#_Toc280296576)

[2.5 Режим прерываний 9](#_Toc280296577)

[2.6 Прямой доступ к памяти 11](#_Toc280296578)

[2.7 Подключение дополнительных портов 14](#_Toc280296578)

[3. Система команд 15](#_Toc280296584)

[4. Программная часть 22](#_Toc280296585)

[4.1 Блок-схема алгоритма вычисления основной функции 23](#_Toc280296590)

[4.2 Листинг программы 24](#_Toc280296591)

[5. Функциональная схема 31](#_Toc280296592)

[Выводы 32](#_Toc280296593)

**Вступление.**

Микропроцессоры (МП) представляют собой автономные функционально законченные устройства, состоящие из одной или нескольких программно-управляемых интегральных микросхем высокой степени интеграции, включающие все средства, необходимые для обработки информации и управления данными рассчитанные на совместную работу с устройствами памяти и ввода-вывода информации.

Микроконтроллер — микросхема, предназначенная для управления электронными устройствами. Типичный микроконтроллер сочетает в себе функции процессора и периферийных устройств, может содержать ОЗУ и ПЗУ. Большая часть выпускаемых в современном мире процессоров — микроконтроллеры.

Основным классификационным признаком микроконтроллеров является разрядность данных, обрабатываемых арифметико-логическим устройством (АЛУ). По этому признаку они делятся на 4-, 8-, 16-, 32- и 64-разрядные.

При проектировании микроконтроллеров приходится соблюдать баланс между размерами и стоимостью с одной стороны и гибкостью и производительностью с другой. Для разных приложений оптимальное соотношение этих и других параметров может различаться очень сильно. Поэтому существует огромное количество типов микроконтроллеров, отличающихся архитектурой процессорного модуля, размером и типом встроенной памяти, набором периферийных устройств, типом корпуса и т. д.

Программирование микроконтроллеров обычно осуществляется на языке ассемблера или Си. Для отладки программ используются программные симуляторы (специальные программы для персональных компьютеров, имитирующие работу микроконтроллера), внутрисхемные эмуляторы и интерфейс JTAG.

1. **Обзор микроконтроллера**

### С развитием электроники сменилась элементная база ЭВМ – появились машины на транзисторах, а затем на микросхемах. Однако по традиции продолжали разрабатываться большие и мощные ЭВМ. И вот в середине 60-х годов появился новый класс вычислительных машин – однокристальные ЭВМ.

### Разработчики ОЭВМ исходили из того, что не везде нужны все (максимальные) возможности больших ЭВМ, не всегда требуется большая точность вычислений, большие объемы памяти или длительное хранение промежуточных результатов. Зато для целого ряда применений, таких, как управление производственным оборудованием или научным экспериментом, необходимо вводить и выводить специальные сигналы, учитывать течение времени, реагировать на случайно происходящие события.

### Вместе с этим, есть то минимальное ядро, без которого аппаратура еще не ЭВМ. Эти компоненты уже давно определились: арифметико-логическое устройство (АЛУ), процессор, оперативно запоминающее устройство (ОЗУ), устройства ввода/вывода.

### Простейшая в серии 1816 ОЭВМ МК48, имеет на кристалле следующие аппаратурные средства: процессор разрядностью 1 байт; стираемое программируемое ПЗУ программ ёмкостью 1 Кбайт, ОЗУ данных ёмкостью 64 байта; программируемый 8-битный таймер/счетчик; программируемые схемы ввода/вывода; блок векторного прерывания от двух источников; генератор; систему синхронизации и управления.

### Микроконтроллер МК48 конструктивно выполнен в корпусе БИС с 40 внешними выводами. Все выводы электрически совместимы с элементами ТТЛ, входы представляют собой единичную нагрузку, а выходы могут быть нагружены одной ТТЛ-нагрузкой.

### Структуры ОЭВМ серии 1816 и их команд таковы, что в случае необходимости функционально-логические воз­можности могут быть расширены. С использование внеш­них дополнительных БИС постоянной и оперативной па­мяти адресное пространство может быть расширено, а путем подключения различных интерфейсных БИС число линий связи ОЭВМ с объектом управления мо­жет быть увеличено практически без ограничений.

### ОЭВМ серии 1816 требуют одного источника электро­питания напряжением +5В ± 10%, рассеивают мощность около 1,5 Вт и работают в диапазоне температур от 0 до 700С. по входам и выходам серии 1816 электрически совместимы с интегральными схемами ТТЛ.

### ОЭВМ МК 48 может работать в диапазоне частот син­хронизации от 1 до 6 МГц, а минимальное время выпол­нения команды составляет 2,5 мкс.

### Микроконтроллер состоит из следующих узлов:

### — однокристальной ЭВМ со схемой внешнего тактового генератора и схемой формирования сигнала «сброс»;

### — регистра-защелки младшего байта адреса внешнего запоминающего устройства;

### — памяти программ, объемом 4 Кбайта;

### — памяти данных, объемом 1 Кбайт со страничной адресацией 256 байт на страницу и схемой выбора ОЗУ;

### — схемы управления записью-чтением внешних устройств;

### — адаптера параллельного интерфейса со схемами приемника и передатчика по стандарту ИРПС;

### — трехканального таймера;

### — контроллера клавиатуры и индикации;

### — схемы прерываний.

1. **Построение микропроцессорной системы (МПС)**
   1. **Описание архитектуры МПС**

### Разрабатывается микропроцессорная система (МПС), ядром которой является процессор 1816ВЕ48.

### В состав разрабатываемой МПС должны входить процессор (П), основная память (ОП), содержащая ОЗУ и ПЗУ, а также внешние устройства (ВУ), контроллеры прерываний и прямого доступа к памяти.

### Микропроцессор имеет 8-разрядную операционную часть. Объем внешней памяти программ 4К, внешней памяти данных – 8К. Шина адреса и данных разделённая, КПП и КПДП децентрализированное. Каждое ВУ имеет фиксированный адрес в адресном пространстве периферийных устройств.

* 1. **Память программ**

### Память программ и память данных в микроЭВМ разделены. Максимальное адресное пространство памяти программ составляет 4 Кбайт. Память программ разделяется на внутреннюю (находящуюся внутри ИС) и внешнюю, для реализации которой требуются дополнительные ИС памяти. Внутренняя память программы представляет собой перепрограммируемое ПЗУ(ППЗУ) ёмкостью 1Кбайт (адреса от 0 до 1023). Адреса 0, 3 и 7 имеют специальное назначение. С адреса 0 начинает выполняться программа по системному сбросу. Ячейка 3 предназначена для хранения начального адреса подпрограммы обслуживания внешнего прерывания, а ячейка 7 – для прерывания от таймера/счётчика.

### ome1816v

### **Рис.2.2.1 Организация памяти программ**

### Старший разряд РС[11] определяется признаком МВ, который устанавливается программно / командами SEL MB0 и SEL MB1/. Память программы в связи с этим рассматривается в виде двух банков (нулевого и первого). Если РС[11] = 0, то выборка слов осуществляется из нулевого банка (адреса от 0 до 2047), а при РС[11] = 1 из первого банка (адреса от 2047 до 4095). Программные средства для проверки признака банка памяти МВ отсутствуют.

### Кроме разделения на банки, память программ делится на страницы по 256 байт. Это связано с тем, что команды условных переходов модифицируют только 8 младших разрядов адреса, т.е. обеспечивают переход внутри страницы. При переходе к подпрограммам обслуживания прерываний автоматически обнуляется разряд РС[11] счётчика команд. В связи с этим подпрограммы обслуживания прерываний должны размещаться в нулевом банке памяти. Способы адресации операндов в памяти программ:

### - непосредственная;

### - косвенная с использованием аккумулятора.

### В первом случае операнд содержится в байте, следующем за кодом команды. При косвенной адресации в качестве указателя адреса операнда в текущей странице (или в странице 3) применяется аккумулятор А.

### На чертеже рис.1 элементы представляют страницы памяти программ 1К. Выбор страницы осуществляется при помощи DC, на который поступает адрес с порта Р2. Всего адресуется 4 страницы памяти программ по 1К.

* 1. **Память данных**

### Память данных, как и память программ, разделяется на внутреннюю и внешнюю. Внутренняя память данных представляет собой ОЗУ ёмкостью 64 байта. Память содержит два банка регистров общего назначения (рабочих регистров). Банк 0 включает регистры R0-R7 с адресами 0-7, а банк 1 – регистры R0-R7, которые имеют адреса 24-31. Выбор регистрового банка осуществляется командами SEL RB0 и SEL RB1, которые устанавливают признак RB, находящийся в 4-м разряде PSW. Специальная команда для проверки RB отсутствует, но признак можно проанализировать, переслав содержимое PSW в А и выполнив переход по признаку B4, который проверяется командой JB4. Ячейки с адресами 8-23 могут использоваться как 8-уровневый стек 16-разрядных слов или как ячейки ОЗУ данных с произвольным доступом.

### ome1816p

### **Рис.2.3.1 Карта распределения адресов внутренней памяти данных**

### Способы адресации для доступа к внутренней памяти данных:

### - прямая регистровая;

### - косвенная регистровая.

### Прямая регистровая адресация используется для обращения к регистрам общего назначения (R0-R7 при RB=0 или R0’-R7’ при RB=1). В этом случае 3-разрядный адрес регистра присутствует в коде команды.

### С помощью косвенной регистровой адресации можно обратиться к любому байту внутренней памяти данных (в том числе, к регистровым банкам и стеку). В качестве указателя адреса операции в этом случае используются регистры R0 и R1, если выбран нулевой регистровый банк или R0 и R1, если выбран 1 регистровый банк. Косвенная регистровая адресация применяется и при выполнении команд обращения к внешней памяти данных.

### Так как в качестве указателя адреса используются 8-разрядные регистры R0, R1, R0’ и R1’, то максимальный объём внешней памяти данных может составлять 256 байт. Увеличение объёма внешней памяти данных может быть достигнуто программно с применением страничной адресации.

### На чертеже рис.1 элементы представляют страницы памяти данных 256\*8 бит. Выбор страницы осуществляется при помощи DC, на который поступает адрес с порта Р1. Всего 32 страницы памяти данных, то есть в сумме 8К.

* 1. **Внешние устройства (ВУ) и периферийный программируемый адаптер (ППА)**

### В данном режиме инициатором обмена является процессор. Для синхронизации используется бит готовности в порте ВУ. Этот бит устанавливается контроллером ВУ, когда оно готово к обмену, и сбрасывается при обращении к порту данных.

### При наличии нескольких ВУ для их обслуживания используется программный поллинг, т.е. опрос ВУ в соответствии с их приоритетом. Режим опроса готовности имеет преимущества и недостатки по сравнению с другими режимами.

### Преимущества: простота реализации интерфейса ВУ, в процессе функционирования программы можно менять приоритеты ВУ. Недостатки: снижение производительности за счет непроизводительного расхода команд процессора на опрос ВУ, трудно предусмотреть аварийное или экстренное обслуживание некоторого ВУ.

### Всего внешних устройств 24, их адреса с 80h до AFh(адреса регистров состояний и регистров данных ВУ).

### Также подключён ППА с адресами С0h, С1h, С2h, С3h, которые относятся к 32 странице ПД. Составлен селектор адреса для подключения ППА или ПД32,если выбрана 32 страница. Если старшие 6 бит совпадают с адресами для ППА и выбрана 32 страница то подключается ППА, если адрес не совпадает и выбрана 32 страница, то подключается ПД32.

* 1. **Режим прерываний**

### Под прерыванием понимают временную приостановку выполнения программы и переход на другую подпрограмму с возможностью возврата на прерванную.

### Прерывания можно классифицировать следующим образом: внутренние и внешние.

### Внутренние делятся на аппаратные и программные.

### Внешние делятся на безвекторные и векторные.

### Аппаратные прерывания: требование прерывания формируется определенными схемами процессора при наступлении определенных событий (деление на ноль, зависание при обращении к памяти или ВУ и т.д.)

### Программные прерывания: вызываются при выполнении команд прерывания программы. Эти команды могут вводиться программистом в исходную программу или вставляться компилятором в процессе компилирования.

### Эти прерывания удобны в процессе отладки системы (они имитируют внешние прерывания), а также являются универсальным средством для вызова стандартных подпрограмм ОС.

### Внутренние прерывания имеют фиксированные начальные адреса подпрограмм для их обслуживания.

### Безвекторные прерывания: Процессор имеет специальные входы для поступления запросов на прерывания программы. Для некоторых входов существуют стандартные подпрограммы обслуживания с фиксированным начальным адресом (сбой по питанию постоянного тока, сбой по питанию переменного тока, сигнал от внешнего таймера).

### Векторные: любому ВУ можно разрешить прерывание программы. Для подачи сигнала такого прерывания используется один вход процессора. Идентификация устройства процессором осуществляется с помощью чтения на шине данных вектора (номера ВУ). Специальная процедура на аппаратном или программном уровне ставит в соответствие вектору начальный адрес подпрограммы обслуживания.

### В процессе инициализации системы процессор записывает в регистр состояния ВУ единицу в бит разрешения прерывания, если это устройство будет работать в режиме прерывания.

### Кроме этого процессор может записывать вектор в регистр вектора. Регистр вектора может быть тумблерным.

### Когда ВУ готово к обмену, устанавливается бит готовности в регистре состояния своим контроллером. По совпадении сигналов готовности и разрешения прерывания формируется низким уровнем сигнал требование прерывания на общей однопроводной шине.

### Процессор проверяет сигнал после выполнения команды и формирует последовательно два сигнала по шине управления: подготовка и подтверждение прерывания.

### По сигналу подготовка во всех ВУ запрещается изменение всех триггеров. В каждом интерфейсе коммутируется путь прохождения сигнала подтверждение прерывания. Если ВУ выставляло требование прерывания, то цепь прохождения дальше сигнала ПП разрывается, а в этом ВУ по данному сигналу выдается вектор на шине данных, который принимается процессором.

### В разрабатываемой ЭВМ реализован распределенный контроллер

### 

### На рисунке приняты обозначения:

### П - процессор, РАПП - распределенный арбитр приоритетных прерываний,

### ФВ - формирователь вектора,

### PC - регистр состояния ВУ, РД - регистр данных ВУ,

### ПД - сигнал подготовки, ЗП - запрос на прерывание,

### (in) - входящий сигнал подтверждение прерывания,

### (out) - выходящий сигнал подтверждение прерывания.

* 1. **Прямой доступ к памяти**

Одним из способов обмена данными с ВУ является обмен в режиме прямого доступа к памяти (ПДП). В этом режиме обмен данными между ВУ и основной памятью микроЭВМ происходит без участия процессора. Обменом в режиме ПДП управляет не программа, выполняемая процессором, а электронные схемы, внешние по отношению к процессору. Обычно схемы, управляющие обменом в режиме ПДП, размещаются в специальном контроллере, который называется контроллером прямого доступа к памяти.

Обмен данными в режиме ПДП позволяет использовать в микроЭВМ быстродействующие внешние запоминающие устройства, такие, например, как накопители на жестких магнитных дисках, поскольку ПДП может обеспечить время обмена одним байтом данных между памятью и ВЗУ, равное циклу обращения к памяти.Для реализации режима прямого доступа к памяти необходимо обеспечить непосредственную связь контроллера ПДП и памяти микроЭВМ. В целях сокращения количества линий в шинах микроЭВМ контроллер ПДП подключается к памяти посредством шин адреса и данных системного интерфейса. При этом возникает проблема совместного использования шин системного интерфейса процессором и контроллером ПДП. Можно выделить два основных способа ее решения: реализация обмена в режиме ПДП с "захватом цикла" и в режиме ПДП с блокировкой процессора.

Существуют две разновидности прямого доступа к памяти с "захватом цикла". Наиболее простой способ организации ПДП состоит в том, что для обмена используются те машинные циклы процессора, в которых он не обменивается данными с памятью. В такие циклы контроллер ПДП может обмениваться данными с памятью, не мешая работе процессора. Однако возникает необходимость выделения таких циклов, чтобы не произошло временного перекрытия обмена ПДП с операциями обмена, инициируемыми процессором. В некоторых процессорах формируется специальный управляющий сигнал, указывающий циклы, в которых процессор не обращается к системному интерфейсу. Более распространенным является ПДП с "захватом цикла" и принудительным отключением процессора от шин системного интерфейса. Для реализации такого режима ПДП системный интерфейс микроЭВМ дополняется двумя линиями для передачи управляющих сигналов "Требование прямого доступа к памяти" (ТПДП) и "Предоставление прямого доступа к памяти" (ППДП).

Управляющий сигнал ТПДП формируется контроллером прямого доступа к памяти. Процессор, получив этот сигнал, приостанавливает выполнение очередной команды, не дожидаясь ее завершения, выдает на системный интерфейс управляющий сигнал ППДП и отключается от шин системного интерфейса. С этого момента все шины системного интерфейса управляются контроллером ПДП. Контроллер ПДП, используя шины системного интерфейса, осуществляет обмен одним байтом или словом данных с памятью микроЭВМ и затем, сняв сигнал ТПДП, возвращает управление системным интерфейсом процессору. Как только контроллер ПДП будет готов к обмену следующим байтом, он вновь "захватывает" цикл процессора и т.д. В промежутках между сигналами ТПДП процессор продолжает выполнять команды программы. Тем самым выполнение программы замедляется, но в меньшей степени, чем при обмене в режиме прерываний.

Применение в микроЭВМ обмена данными с ВУ в режиме ПДП всегда требует предварительной подготовки, а именно: для каждого ВУ необходимо выделить область памяти, используемую при обмене, и указать ее размер, т.е. количество записываемых в память или читаемых из памяти байт (слов) информации. Следовательно, контроллер ПДП должен обязательно иметь в своем составе регистр адреса и счетчик байт (слов). Перед началом обмена с ВУ в режиме ПДП процессор должен выполнить программу загрузки. Эта программа обеспечивает запись в указанные регистры контроллера ПДП начального адреса выделенной ВУ памяти и ее размера в байтах или словах в зависимости от того, какими порциями информации ведется обмен. Сказанное не относится к начальной загрузке программ в память в режиме ПДП. В этом случае содержимое регистра адреса и счетчика байт слов устанавливается переключателями или перемычками непосредственно на плате контроллера.

Использование БИС ПДП позволяет существенно сократить аппаратные затраты при реализации прямого доступа к памяти.



Рисунок 2.4. Реализация централизованного КПДП

* 1. **Подключение дополнительных портов**

Для увеличения количества линий связи МК с объектом управления можно подключить дополнительные 4-разрядные порты P4-P7. Наиболее просто это достигается при использовании специальной ИС КР580 ВР43. В этом случае обеспечивается выполнение всех четырех команд с дополнительными портами, причем каждый вывод порта может быть настроен на ввод или вывод информации. Команды выполняются за 2 цикла. В первом цикле на выводы P4-P7 выдается управляющее слово, а во втором – через указанные выводы осуществляется обмен информацией между МК и портом.

1. **Система команд**
   1. **Основные команды**

### Основные команды МК48 включают в себя 96 основных команд и ориентированы на реализацию процедур управления. Все команды имеют формат один или два байта (70% команд однобайтные). Время выполнения команд составляет 2.5 или 5.0 мкс (один или дна машинных цикла соответственно) при тактовой частоте 6.0 МГц. Большинство команд выполняется за один машинный цикл. За два машинных цикла выполняются команды с непосредственным операндом, ввода/вывода и передачи управления.

### Основная группа команд пересылки данных. Данная группа состоит из 24 команд. Все команды (кроме MOV PSW, А) не оказывают воздействия на флаги. Команды пересылки данных внутри МК выполняются за один машинный цикл, обмен с внешней памятью и портами требует двух машинных циклов. Пример некоторых операций показан в таблице 3.1:

### **Таблица 3.1**

|  |  |
| --- | --- |
| Название команды | Мнемокод |
| Пересылка регистра в аккумулятор | MOV А,Rn |
| Пересылка байта из РПД в аккумулятор | MOV A,@Ri |
| Пересылка непосредственного операнда в аккумулятор | MOV A,#d |
| Пересылка аккумулятора в регистр | MOV Rn,A |
| Пересылка непосредственного операнда в регистр | MOV Rn,#d |
| Пересылка аккумулятора в РПД | MOV @Ri,A |
| Пересылка непосредственного операнда в РПД | MOV @Ri,#d |

### 

### **Пример**

; К микроконтроллеру подключено 5 страниц внешней памяти данных

; Из 5 страницы переслать массив из 8 байт по адресам C0h - C7h

; В банки регистров внутренней памяти данных.

mov r0, #C0h

mov r1, #24

mov r2, #8

orl p1, #11111111b

anl p1, #5; подключение 5 страницы. Производится через дешифратор.

ll: movx A, @r0 ; пересылка из внешней памяти данных в аккумулятор.

mov @r1, A; пересылка из аккумулятора в регистр.

inc r1

inc r0

djnz r2,ll

end; C0-C7- по адресам 24-31

* 1. **Арифметические и логические команды**

### Группа команд арифметических операции. Данная группа состоит из 12 команд и позволяет выполнять следующие операции над 8-битными целыми двоичными числами без знака: двоичное сложение (АDD), двоичное сложение с учетом переноса (АDDС) , десятичная коррекция (DA) , инкремент (INС) и декремент (DЕС). Пример некоторых операций показан в таблице 3.2.1:

### **Таблица** 3.2.1

|  |  |
| --- | --- |
| Название команды | Мнемокод |
| Сложение регистра с аккумулятором | ADD А,Rn |
| Сложение константы с аккумулятором | ADD A,#d |
| Сложение регистра с аккумулятором и переносом | ADDC A,Rn |
| Инкремент аккумулятора | INC А |
| Инкремент регистра | INC Rn |
| Декремент аккумулятора | DEC A |
| Декремент регистра | DEC Rn |

### Группа команд логических операций. Данная группа состоит из 28 команд и позволяет выполнять следующие операции над байтами: дизъюнкцию, конъюнкцию, исключающее ИЛИ, инверсию, сброс и сдвиг. Две команды (сброс и инверсия) позволяют выполнять операции над битами. Пример некоторых операций показан в таблице 3.1.1.2:

### **Таблица 3.2.2**

|  |  |
| --- | --- |
| Название команды | Мнемокод |
| Логическое И регистра и аккумулятора | ANL A,Rn |
| Логическое И константы и аккумулятора | ANL А,#d |
| Логическое ИЛИ регистра и аккумулятора | ORL A,Rn |
| Логическое ИЛИ константы и  аккумулятора | ORL A,#d |
| Исключающее ИЛИ регистра и аккумулятора | XRL A,Rn |
| Исключающее ИЛИ константы и аккумулятора | XRL A,#d |
| Сброс аккумулятора | CLR A |
| Инверсия аккумулятора | CPL A |
| Циклический сдвиг влево аккумулятора | RL А |
| Циклический сдвиг вправо аккумулятора | RR А |
| Сброс переноса | CLR С |
| Сброс флага F1 | CLR F1 |
| Инверсия переноса | CPL С |
| Инверсия флага F1 | CPL F1 |

### **Пример программы с приведённым алгоритмом:**

### F = 4\*(X1-X2)+(X3&X4) - (X5-X6-1) / 2

### **Алгоритм:**

Начало

4\*(X1-X2) -> R5

X3&X4 -> R6

-(X5-X6 - 1) -> A

A+R5 -> A

A+R6 -> A

Конец

**Листинг программы**

;---input x1 x2

in A, p1

mov r1, A

in A, p1

mov r2, A

;--x1 - x2

Cpl A

Inc A

Add A,r1

;--4\*()

clr c

rlc A

clr c

rlc A

Mov r5,A

;---input x3 x4

in A, p1

mov r1, A

in A, p2

mov r2, A

;--x3&x4

Mov A, r2

Anl A, r1

Mov r6, A

;---input x5 x6

in A, p2

mov r1, A

in A, p2

mov r2, A

;--(x5-x6-1)/2

Mov A,r2

Cpl A

Add A, r1

clr c

Rrc A

;--------

cpl A

inc A

Add A, r5

Add A, r6

End

* 1. **Команды передачи управления**

### Группа команд передачи управления. Данную группу образуют 19 команд передачи управления, из них две команды безусловного перехода, 14 команд условного перехода, команда вызова подпрограмм и две команды возврата из подпрограмм. Пример некоторых операций показан в таблице 3.3.1:

### Таблица 3.3.1

|  |  |
| --- | --- |
| Название команды | Мнемокод |
| Безусловный переход | JMP ad11 |
| Декремент регистра и переход, если не нуль | DJNZ Rn,ad |
| Переход, если перенос | JC ad |
| Переход ,если нет переноса | JNC ad |
| Переход, если аккумулятор содержит нуль | JZ ad |
| Переход, если аккумулятор содержит не нуль | JNZ ad |
| Переход, если флаг F1 установлен | JF1 ad |
| Возврат из подпрограммы | RET |

* 1. **Команды управления режимом работы МК**

### Группа команд управления режимом работы МК. В эту группу входят команды управления таймером/счетчиком, прерываниями и флагами переключения банков регистров и банков ПП. Пример некоторых операций показан в таблице 3.4.1:

### Таблица 3.4.1

|  |  |
| --- | --- |
| Название команды | Мнемокод |
| Запуск таймера | STRT T |
| Запуск счетчика | STRT CNT |
| Останов таймера/счетчика | STOP TCNT |
| Выбор нулевого банка регистров | SEL RB0 |
| Выбор первого банка регистров | SEL RB1 |
| Выбор нулевого банка ПП | SEL МВ0 |
| Выбор первого банка ПП | SEL МВ1 |
| Холостая команда | NOP |

### **Пример программы с приведённым алгоритмом:**

Y5

X1

Y1Y2Y3

X2

X2

Y2

Y4Y5

### 

; установка порта Р1

ANL P1, #11000000b ; условия X2 X1

ORL P1, #11100000b ; первая вершина - выдача y5

MOV A,#11110111b ; [-9] дк

MOV T,A

STRT T ; задержка 9\*80=720

l1: JTF l2

jmp l1

;----------------- проверка X1

l2: IN A,P1

m3: JB6 m1;

;-----------------

ORL P1,#11111111b

ANL P1, #11001110b; вторая вершина - выдача y1y2y3

MOV R5,#5; задержка 5\*5=25

for1: DJNZ R5, for1

;-----------------проверка X2

IN A,P1

JB7 m2

JMP m3

;-----------------проверка X2

m1: IN A,P1

JB7 m4;

;-----------------

m2: ORL P1,#11111111b

ANL P1, #11000100b ; третья вершина - выдача y2

MOV A,#11111011b ; [-5] дк

MOV T,A

STRT T ; задержка 5\*80=400

l3: JTF m4

jmp l3

;-----------------

m4: ORL P1,#11111111b

ANL P1, #11110000b; четвёртая вершина - выдача y4y5

MOV R5,#2; задержка 2\*5=10

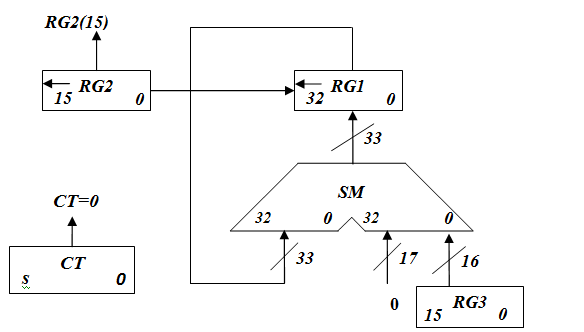
for2: DJNZ R5, for2

nop

1. **Программная часть**

Задача реализовать функцию .

Третий способ умножения

****

Операционная схема

**Общий алгоритм программы**



* 1. **Листинг программы**

In a,p1

Mov r1,a

In a,p1  
mov r2, a

Cpl a

Inc a

Add a,x1

Mov r2,a

;Вычисление корня - операнд восьмирозрядный.

; r2-операнд

; r1- для суммирования

; r0-для хранения результата

;Начальные значения

mov r1,#0

mov r0, #0

; количество циклов - точность вычисления= 4 знака

mov r3,#4h

ll3:

; r4=r0

mov a, r0

mov r4,a

; знак

mov r5,#0

; два сдвига влево r2.r1

clr c

mov a, r2

rlc a

mov r2, a

mov a, r1

rlc a

mov r1,a

clr c

mov a, r2

rlc a

mov r2,a

mov a, r1

rlc a

mov r1, a

; анализ знака

jb7 ll1

; r4=not(r0)

mov a, r4

cpl a

mov r4, a

;Два сдвига вправо + or 00000011b

ll1: mov a, r4

rlc a

rlc a

orl a, #3h

mov r4, a

; суммирование

add a,r1

mov r1, a

; проверка знака

jb7 ll2

mov r5, #1h

ll2:

; левый сдвиг и суммирование

clr c

mov a, r0,

rlc a

add a, r5

mov r0, a

; проверка цикла на точность.

djnz r3, ll3

mov a, r0

mov r0,#30h ; результат sqrt(x1-x2) в 30h

mov @r0,A

in a,p1 ; x7

mov r2,a

in a,p1 ; x8

mov r1,a

call divideR2\_R1 ; деление, результат в r0

clr c

mov a,r0

mov r0,#30h ;

add a,@r0 ; сумма (х7/x8) + sqrt(x1 – x2)

mov @r0,a

clr a

addc a,#0h ; старшие разряды в 31h

inc r0

mov @r0,a

;result 30 31h (х7/x8) + sqrt(x1 – x2)

in a,p1

mov r1,a ; старшие разряды х9

mov r3,a ; старшие разряды х9

in a,p1

mov r2,a ; младшие разряды х9

mov r4,a ; младшие разряды х9

;========производим подготовку к умножению х9\*х9===

mov r6,#0

mov r5,#0

mov r7,#10h ; счётчик в умножении

mov r0,#15h ; адрес начала младших разрядов результата

call mull; вызов подпрограммы умножения

in a,p1

mov r1,a ; старшие разряды х10

mov r3,a ; старшие разряды х10

in a,p1

mov r2,a ; младшие разряды х10

mov r4,a ; младшие разряды х10

;========производим подготовку к умножению х10\*х10===

mov r6,#0

mov r5,#0

mov r7,#10h ; счётчик в умножении

mov r0,#20h ; адрес начала младших разрядов результата

call mull; вызов подпрограммы умножения

;========Производим Суммирование двух операндов X10\*X10 и Х9\*Х9 ===

CLR C

MOV RO,#15H

MOV A,@RO ; считывания разрядов результата умножения Х9\*Х9

MOV R2,A

MOV R1,#20H

MOV A,@R1 ; считывание разрядов результата суммирования X10\*X10

ADD A,R2 ; суммирование результатов и фиксация признака переноса

MOV @R1,A ; запись результата в память в ячейку 20h

MOV RO,#16H

MOV A,@RO ; считывания разрядов результата умножения Х9\*Х9

MOV R2,A

MOV R1,#21H

MOV A,@R1 ; считывание разрядов результата суммирования X10\*X10

ADDC A,R2 ; суммирование результатов и фиксация признака переноса

MOV @R1,A ; запись результата в память в ячейку 21h

MOV RO,#17H

MOV A,@RO ; считывания разрядов результата умножения Х9\*Х9

MOV R2,A

MOV R1,#22H

MOV A,@R1 ; считывание разрядов результата суммирования X10\*X10

ADDC A,R2 ; суммирование результатов и фиксация признака переноса

MOV @R1,A ; запись результата в память в ячейку 22h

MOV RO,#18H

MOV A,@RO ; считывания разрядов результата умножения Х9\*Х9

MOV R2,A

MOV R1,#23H

MOV A,@R1 ; считывание разрядов результата суммирования X10\*X10

ADDC A,R2 ; суммирование результатов и фиксация признака переноса

MOV @R1,A ; запись результата в память в ячейку 23h

;result 20 21 22 23 hex (x9^2 + x10^2)

============== вся функция

CLR C

MOV RO,#30H

MOV A,@RO ;

MOV R2,A

MOV R1,#20H

MOV A,@R1 ;

ADD A,R2 ; суммирование результатов и фиксация признака переноса

MOV @R1,A ; запись результата в память в ячейку 20h

MOV RO,#31H

MOV A,@RO ;

MOV R2,A

MOV R1,#21H

MOV A,@R1 ;

ADDC A,R2 ; суммирование результатов и фиксация признака переноса

MOV @R1,A ; запись результата в память в ячейку 21h

MOV R1,#22H

MOV A,@R1 ;

ADDC A,#0h ;фиксация признака переноса

MOV @R1,A ; запись результата в память в ячейку 22h

MOV R1,#23H

MOV A,@R1 ;

ADDC A,#0h ; фиксация признака переноса

MOV @R1,A ; запись результата в память в ячейку 23h

Jmp endddd;

;Результат в 20 21 22 23h

;======== ПОДПРОГРАММ УМНОЖЕНИЯ ДВОХ ЧИСЕЛ Х\*У =======

; R0 – адрес начала младших разрядов результата

; R1 – старшие разряды Х

; R2 – младшие разряды Х

; R3 – старшие разряды У

; R4 – младшие разряды У

; R5 – дополнительный регистр

; R6 – дополнительный регистр

; R7 – счётчик

MUL:

LL0:

MOV A,R1

JB7 LL1 ; Проверка Х1\* (старшего разряда)

LL3: ; X1\*=0;

; P1 сдвигаем вправо ===================================

MOV A,R3 ; Старшие разряды в А

CLR C

RRC A ; Сдвиг А вправо А[7] = 0 фиксируем признак **с**

MOV R3,A

MOV A,R4 ; разряды в А

RRC A ; Сдвиг А вправо А[7] = **с** фиксируем признак **с**

MOV R4,A

MOV A,R5 ; разряды в А

RRC A ; Сдвиг А вправо А[7] = **с** фиксируем признак **с**

MOV R5,A

MOV A,R6 ; Младшие разряды в А

RRC A ; Сдвиг А вправо А[7] = **с** фиксируем признак **с**

MOV R6,A

CLR C ; очистка признак **с**

; P3 сдвигаем вправо ===================================

MOV A,R2

RLC A ; Сдвиг А вправо А[0] = 0 фиксируем признак **с**

MOV R2,A

MOV A,R1

RLC A ; Сдвиг А вправо А[0] = **с** фиксируем признак **с**

MOV R1,A

JMP CHECK ; Прыжок на метку

LL1:

; Х1\* = 1 ========================================================

; Р2 = Р2 + Р1

; Считываем данные и суммируем. Результат записываем в память

MOV A,R0 ; считали данные

ADD A,R6 ; просуммировали

MOV @RO,A ; результат обратно в память

INC R0 ; R0 = R0 + 1 - для получения адреса следующих разрядов

MOV A,R0 ; считали данные

ADDС A,R5 ; просуммировали с учётом переноса

MOV @RO,A ; результат обратно в память

INC R0 ; R0 = R0 + 1 - для получения адреса следующих разрядов

MOV A,R0 ; считали данные

ADDС A,R4 ; просуммировали с учётом переноса

MOV @RO,A ; результат обратно в память

INC R0 ; R0 = R0 + 1 - для получения адреса следующих разрядов

MOV A,R0 ; считали данные

ADDС A,R3 ; просуммировали с учётом переноса

MOV @RO,A ; результат обратно в память

JMP LL3

CHECK:

DJNZ R7,LLO ; прыжок на метку LLO если R7 не равно нулю и декремент R7

RET ; Возврат из подпрограммы

; z = x/y

; x < y

;y => R1 , x => R2

;в R0 маркерная единица

;результат в R0

divideR2\_R1:

mov r0, #01h ; Запись маркерной единицы

xch a, r1 ;формирование

cpl a; в регистре R1

inc a;делителя с отрицательным

xch a,r1;знаком.

mov a,r2;Делимое в А

CYCLE:

add a,r1;Вычитание делителя из остатка

jc c1; Проверка знака остатка

mov a,r2;Восстановление отрицательного остатка.

c1: xch a,r0;Запись цифры

rlc a;результата

xch a,r0;в R0

jc enda;Проверка конца операции

rlc a;Сдвиг (удвоение) остатка

mov r2,a;Сохранение положительного остатка

jmp cycle;Переход в начало цикла

enda:

retr

endddd:

nop

end.

1. **Функциональная схема подключения ПП и ПД**

# 

Выводы

Разработана микропроцессорная система на основе МК-48 с подключением внешней памяти данных, внешней памяти программ, периферийного адаптера, внешних устройств.

Литература

1. <http://ru.wikipedia.org/wiki/Intel_8048>
2. <http://digital.sibsutis.ru/content.htm>
3. <http://www.intuit.ru/department/hardware/microarch/8/1.html>
4. <http://grantronics.com.au/docs/8048inst.pdf>
5. Пухальский Г.И. Проектирование микропроцессорных систем. СП: Политехника,2001.
6. Самофалов К.Г., Корнейчук В.И., Тарасенко В.П., Жабин В.И. Цифровые ЭВМ. Практикум.- К.: Высш.шк. 1989. - 124 с.
7. GhoshalSubrata 8048 Microcontroller :Internals, Instructions, Programming&Interfacing