Министерство образования и науки Украины

Национальный технический университет Украины «КПИ»

**Расчетно-графическая работа**

по дисциплине «Архитектура компьютеров»

Выполнил:

студент ІІІ курса, группа ІО-02

Грицив А. В.

№зачетной книжки 20610, 110011102

Руководитель:

Ткаченко В. В.

Киев–2012

**Содержание**

1.Обзор существующих решений МПС

2. Разработка архитектуры.

2.1 Структурная схема микроконтроллера 1816ВЕ48.

2.2 Память программ.

2.3Память данных.

2.4 Внешние устройства.

2.5 Подключение дополнительных портов.

2.6Подключение ППА.

2.7Режим прерываний.

2.8 Прямой доступ к памяти.

3. Система команд

3.1 Основные команды

3.1.1 Команды арифметические и логические

3.1.2 Команды передачи управления.

3.1.3 Команды управления режимом работы МК

4. Программная часть

4.1 Алгоритм выполнения функции

4.2 Алгоритм умножения

4.3 Код микропрограммы

Додаток A

**1. Обзор существующих решений МПС.**

**Intel**

Intel 8051 — это однокристальный микроконтроллер гарвардской архитектуры, который был впервые произведен Intel в 1980 году, для использования во встраиваемых системах. В течение 1980-х и начале 1990-х годов был чрезвычайно популярен, однако позже устарел и был вытеснен более современными устройствами, также с 8051-совместимыми ядрами, производимыми более чем 20 независимыми производителями такими как Atmel, Maxim IC (дочерняя компания Dallas Semiconductor), NXP , Winbond, Silicon Laboratories, Texas Instruments и Cypress Semiconductor). Официальное название 8051-семейства микроконтроллеров Intel — MCS 51. Существует также советский клон данной микросхемы, КР1816ВЕ51.

*Особенности :*

Состоит из процессорного ядра (CPU), ОЗУ, ПЗУ, последовательного порта, параллельного порта, логики управления прерываниями, таймер и т. д.

Шина данных — 8-ми битная шина данных. Возможность обработки 8 бит данных за одну операцию. Обуславливает название 8-битный микропроцессор

Шина адреса — 16 битная адресная шина. Возможность доступа к 216 адресам памяти, то есть 64 кБ адресное пространство в ОЗУ и ПЗУ

Встроенное ОЗУ — 128 байт (Памяти данных)

Встроенное ПЗУ — 4 КБ (Памяти программ)

Четыре порта ввода/вывода: один двунаправленный и три квазидвунаправленных

Последовательный интерфейс UART (Универсальный асинхронный приёмопередатчик)

Два 16-битных таймера

Два уровня приоритета прерываний

Порядка 60 тысяч транзисторов на кристалле площадью 5,85 мм²

Энергосберегающий режим

**Atmel**

AVR - самая обширная производственная линии среди других флэш-микроконтроллеров корпорации Atmel. Atmel представила первый 8-разрядный флэш-микроконтроллер в 1993 году и с тех пор непрерывно совершенствует технологию. Прогресс данной технологии наблюдался в снижении удельного энергопотребления (мА/МГц), расширения диапазона питающих напряжений (до 1.8 В) для продления ресурса батарейных систем, увеличении быстродействия до 16 млн. операций в секунду, встраиванием эмуляции в реальном масштабе времени, реализации функции самопрограммирования, совершенствовании и расширении количества периферийных модулей, встраивании специализированных устройств (радиочастотный передатчик, USB-контроллер, драйвер ЖКИ, программируемая логика, контроллер DVD, устройства защиты данных) и др. Другой особенностью AVR-микроконтроллеров, которая способствовала их популяризации, это использование RISC-архитектуры, которая характеризуются мощным набором инструкций, большинство которых выполняются за один машинный цикл. Это означает, что при равной частоте тактового генератора они обеспечивают производительность в 12 (6) раз больше производительности предшествующих микроконтроллеров на основе CISC-архитектуры (например, MCS51). С другой стороны, в рамках одного приложения с заданным быстродействием, AVR-микроконтроллер может тактироваться в 12 (6) раз меньшей тактовой частотой, обеспечивая равное быстродействие, но при этом потребляя гораздо меньшую мощность. Таким образом, AVR-микроконтроллеры представляют более широкие возможности по оптимизации производительности/энергопотребления, что особенно важно при разработке приложений с батарейным питанием. Микроконтроллеры обеспечивает производительность до 16 млн. оп. в секунду и поддерживают флэш-память программ различной емкости: 1… 256 кбайт. AVR-архитектура оптимизирована под язык высокого уровня Си, а большинство представителей семейства megaAVR содержат 8-канальный 10-разрядный АЦП, а также совместимый с IEEE 1149.1 интерфейс JTAG или debugWIRE для встроенной отладки. Кроме того, все микроконтроллеры megaAVR с флэш-памятью емкостью 16 кбайт и более могут программироваться через интерфейс JTAG.

*Особенности:*

Производительность, приближающаяся к 1 MIPS/МГц

Усовершенствованная AVRa RISC архитектура

Раздельные шины памяти команд и данных, 32 регистра общего назначения

Внутрисхемно программируемая Flash-память программ, 1000 циклов записи/стирания

Память данных EEPROM, 100000 циклов записи/стирания

Блокировка режима программирования

Встроенные аналоговый компаратор, сторожевой таймер, порты SPI и UART, таймеры/счетчики

Полностью статические приборы - работают при тактовой частоте от 0 Гц до 20 МГц

Диапазон напряжений питания от 1,8 В до 6,0 В

Режимы энергосбережения: пассивный (idle) и стоповый (power down)

**2. Разработка архитектуры.**

Разрабатывается микропроцессорная система (МПС), ядром которой является процессор 1816ВЕ48. Cмотреть схему *ИАЛЦ 462637 004. Е1*(додаток A)

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 гарвардскую архитектуру. Объем внешней памяти программ 4К, внешней памяти данных – 32К. Шина адреса и данных разделены, КПП и КПДП - децентрализованные. Каждое ВУ имеет фиксированный адрес в адресном пространстве периферийных устройств.

**2.1. Структурная схема микроконтроллера 1816ВЕ48.**



Рис. 2.1 Структурная схема МК48



Рис. 2.2 Модель программиста

|  |  |
| --- | --- |
|  |  |
| Рис. 2.3. Организация памяти программ | Рис. 2.4. Карта распределения адресов внутренней памяти данных |



Рис. 2.5. Организация стека

**2.2. Память программ.**

Память программ реализована в резидентном СППЗУ емкостью 1 Кбайт. Максимальное адресное пространство, отводимое для программ, составляет 4 Кбайт. Адресовать можно и больший объем, если использовать постраничную адресацию внешней памяти.

Для подключения внешней памяти данных используются выходы портов BUS[7..0] иP2[3..0]. Для хранения адреса обращения к памяти используется внешний регистр РА. В процессе обращения к внешней памяти программ МК48 формирует 12-разрядный адрес, в котором старшие битыА[11..8] формируются на выходах P2[3..0], а младшие A[7..0] на порте BUS. Запись адреса в регистр РА стробируется сигналом ALE. После этого порт BUSпереключается в режим считывания информации и выдается сигнал разрешения чтения PME. По этому сигналу данные из памяти считываются в микроконтроллер.



Рис. 2.2. Схема подключения внешней памяти программ.

**2. 3. Память данных.**

Память данных (или - оперативная память, ОЗУ) - резидентная память данных емкостью 64 байта. Имеет в своем составе два банка рабочих регистров 0-7 и 24-31 по восемь регистров в каждом.

Выбор одного из двух банков выполняется по команде "SEL RB". Рабочие регистры доступны командам с прямой адресацией, а все ячейки ОЗУ доступны по командам с косвенной адресацией. В качестве регистров косвенного адреса используются регистры R0, R1.

В режиме работы МК48 с внешней памятью данных используются дополнительные ОЗП емкостью 256 байт. Если адресное пространство больше 256 байт, необходима постраничная организация внешней памяти данных.



Рис. 2.3.1 Схема подключения внешней памяти данных.

## ppa.png

Рис. 2.3.2 Карта распределения последней страницы памяти

За выбор памяти данных или внешних устройств отвечает старший бит адреса.

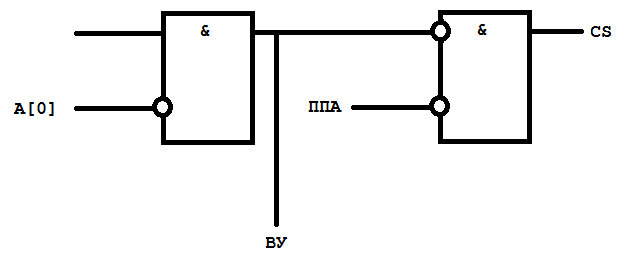


Рис. 2.3.3. Селектор адреса для последней страницы памяти данных

**2.4. Внешние устройства.**

ВУ имеет фиксированный адрес в адресном пространстве периферийных устройств.

Всего внешних устройств 54, которыеадресуются с 00h по 35h. Также подключён ППА с адресами D8h, D9h, DAh, DBh, которые относятся к 64 странице ПД. Составлен селектор адреса для подключения ППА или ПД64,если выбрана 64 страница. Если старшие 6 бит совпадают с адресами для ППА и выбрана 64 страница то подключается ППА, если адрес не совпадает и выбрана 64 страница, то подключается ПД64.

Помимо регистра данных и регистра состояния каждое ВУ содержит регистр режима, регистр начального адреса ОП, регистр-счетчик. Эти регистры обеспечивают прямой доступ ВУ к памяти. Доступ к этим регистрам пользователь может получить через команды IN и OUT микропроцессора в адресном пространстве ВУ. Организация памяти MFT (с фиксированными разделами) с защитой по маскам.

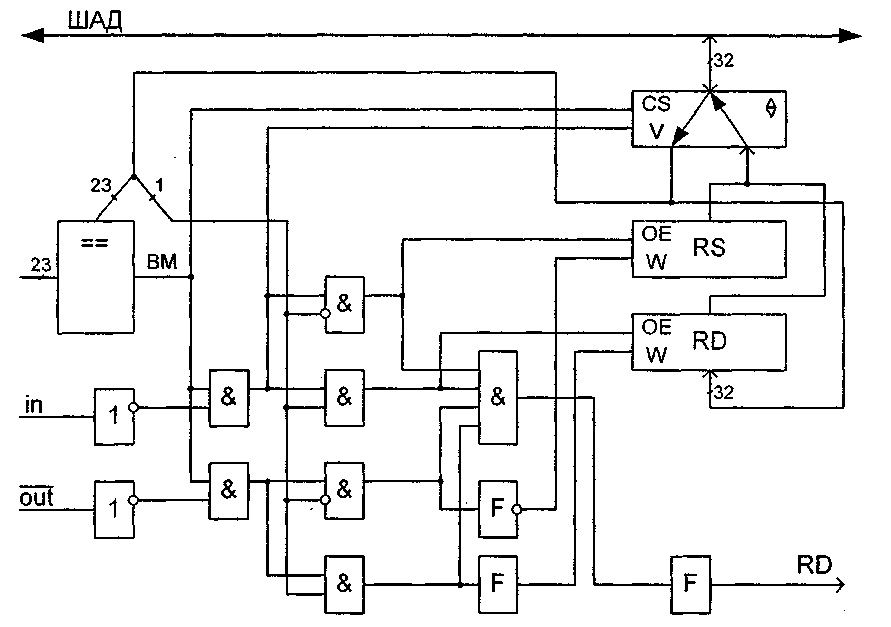


Рис. 2.4. Структурная схема интерфейса ВУ.

**2.5. Подключение дополнительных портов.**

Для увеличенияколичествалинийсвязи МК48 с объектомуправленияподключаютдополнительныечетырехразрядные порти P4, P5, P6, P7.

Команды передачи информации между МК48 и дополнительными портами выполняются за 2 цикла. В первом цикле на выходах P2[3..0] устанавливается управляющее слово, во втором цикле – через эти же выходы происходит обмен информацией между МК48 и одним из дополнительных портов.



Рис. 2.5. Схема соединения МК48 с дополнительными портами.

**2.6. Подключение ППА.**

Рис. 2.6. Схема подключения ППА к МК48.



**2.7. 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.

Микроконтроллеры семейства МК-51 обеспечивают поддержку пяти источников прерываний:двух внешних прерываний, поступающих по входам *INT0* и *INT1* (линии порта Р3: Р3.2 и Р3.3 соответственно);двух прерываний от таймеров/счетчиков*Т/С0* и *Т/С1*;прерывание от последовательного порта.

Запросы на прерывание фиксируются в регистрах специальных функций микроконтроллера: флаги *IE0, IE1, TF0, TF1* запросов на прерывание от *INT0, INT1, T/C0* и *T/C1* соответственно содержатся в регистре управления *TCON*, а флаги *RI* и *TI* запросов на прерыва­ние от последовательного порта - в регистре *SCON* управления последовательным портом (см. рис.4).

Флаги *TF0* и *TF1* устанавливаются аппаратно при переполнении соответствующего таймера/счетчика (переход *T/Cx* из состояния “все единицы” в состоянии “все нули”).

Флаги *IE0* и *IE1* устанавливаются аппаратно от внешних прерываний *IT0* и *IT1* соответственно. Установка *ITx*=0 настраивает систему прерывания на запрос по низкому уровню сигнала, *ITx*=1 - запрос на прерывание по спаду сигнала.

Флаги *TI* и *RI* устанавливаются аппаратно схемой последовательного интерфейса соответственно после окончания передачи или приема байта

Все указанные флаги запросов на прерывания программно доступны для установки и сброса. Программная установка флага запроса на прерывание приводит к такой же реакции микроконтроллера, что и аппаратная установка того же самого флага.

Флаги *TF0* и *TF1* сбрасываются аппаратно при передаче управления программе обработки соответствующего прерывания.

Сброс флагов *IЕ0* и *IЕ1* выполняется аппаратно при обслуживании прерывания только в том случае, если прерывание было настроено на восприятие спада сигнала *INTx*. Если прерывание было настроено на восприятие уровня сигнала запроса, то сброс флага *IEx* должна выполнять программа обслуживания прерывания, воздействуя на источник прерывания для снятия им запроса.

Флаги *TI* и *RI* сбрасываются только программным путем.

Каждый вид прерывания индивидуально разрешается или запрещается установкой или сбросом соответствующих бит регистра разрешения прерывания *IE*. Этот регистр содержит также и бит общего запрещения всех прерываний.

При одновременном поступлении запросов прерывания от источников, имеющих различные приоритеты, сначала обрабатывается запрос от более приоритетного источника.

В случае одновременного поступления нескольких запросов на прерывания с одинаковым приоритетом порядок их обработки определяется аппаратными средствами микроконтроллера и не может быть изменен программно. Этот порядок соответствует последовательности опроса флагов запросов прерываний, имеющей следующий вид: *IT0 → TF0 → IT1 → TF1 → (RI,TI).*

При переходе на подпрограмму обработки прерывания автоматически независимо от состояния регистра *IE* запрещаются все прерывания, имеющие уровень приоритета, равный уровню приоритета об­служиваемого прерывания, то есть вложенные прерывания с равным уровнем приоритета запрещены. Таким образом, низкоприо­ри­тетное прерывание (имеющее “0” в соответствующем разряде регистра *IP*) может прерываться высокоприоритетным (имеющим “1” в соответствующем разряде регистра *IP*), но не низкоприоритетным. Обслуживание высокоприоритетного прерывания не может быть прервано другим источником.

Возврат из обработчика прерываний осуществляется с по­мощью команды *RETI*, которая восстанавливает из стека значение *PC* и логику приоритетов прерываний.

**2.8. 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.

Пояснения к рисунку 2.8:

РАППi– распределенный арбитр КПП

РАПДПi - распределенный арбитр КПДП

ТПДП – требование ПДП

ПОД ПДП – подготовка ПДП

ПП ПДП – подтверждение ПДП

ПВ – подтверждение выборки

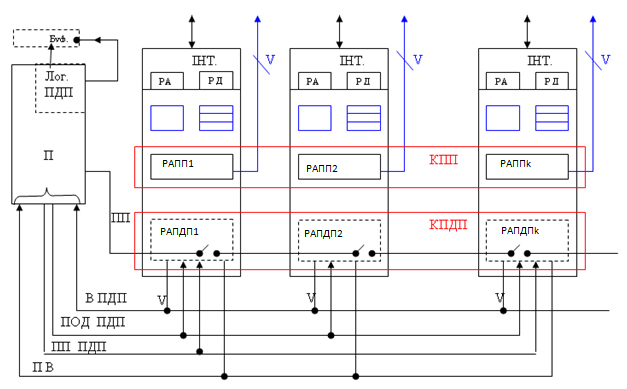


Рис. 2.8. Реализация децентрализованного КПДП.

**3. Система команд.**

**3.1 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 |

**3.1.1 Команды арифметические и логические**

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 3.1.1.1:

Таблица 3.1.1.1

|  |  |
| --- | --- |
| **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 3.1.1.2

|  |  |
| --- | --- |
| **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 приведенным алгоритмом:

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

;установка начальных значений



Mov R1, #FFH

Mov R2, #FFH

Mov R3, #FFH

Mov R4, #F0H

Mov R5, #FH

Mov R6, #FH

Mov R7, #0H

Mov A, R1

Add A, R2

Mov R1, #0H

Jnc M0

Inc R1

M0:

Dec A

Rlc A

Jnc M1

Inc R1

Dec A

M1:

Rlc A

Jnc M2

Inc R1

Dec A

M2:

Mov R0, A

;=================R1:=4\*(x1+x2-1)

Mov A, R3

Add A, R4

Mov R4, #0H

Jnc L1

Inc R4

L1:

Cpl A

Inc A

Mov R3, A

Рис. 3.1.1 Блок-схема алгоритмуMovA, R4

Cpl A

Mov R4, A

Mov A, R0

Add A, R3

Mov R0, A

Mov A, R1

Addc A, R4

Mov R1, A

;=================R1:=R1-(x3+x4)

Mov A, R5

Anl A, R6

Rrc A

Clr C

Rrc A

Clr C

Cpl A

Inc A

Add A,R0

Mov R0,A

Jnc L2

Mov A,R1

Inc A

Mov R1,A

;=================R1:=R1-(x5&x6)/4

L2:

**3.1.2 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 3.1.2.1:

Таблица 3.1.2.1

|  |  |
| --- | --- |
| **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 |

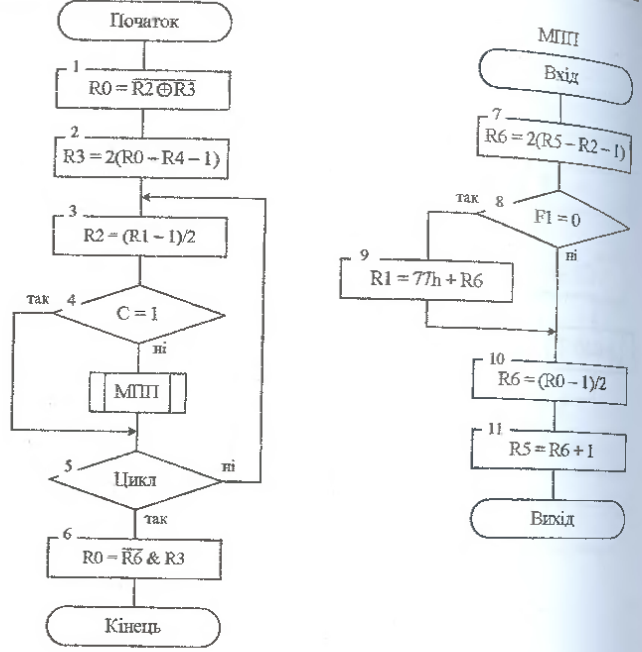


Рис. 3.1.2Блок-схема алгоритму

InsA, Bus

Mov R0, A

Ins A, Bus

Mov R1, A

Ins A, Bus

Mov R2, A

Ins A, Bus

Mov R3, A

Ins A, Bus

Mov R4, A

Ins A, Bus

Mov R5, A

Ins A, Bus

Mov R6, A

Ins A, Bus

Mov R7, A

Mov A, R2

Xrl A, R3

Cpl A

Mov R0, A ; opr\_1

Mov A, R4

Cpl A

Inc A

Add A, R0

Dec A

Clr C

Rlc A

Mov R3, A

Mov R4, #0H

Mov A, R4

Rlc A

Mov R4, A

Clr C ; opr\_2

L3:

Mov A, R1

Dec A

Mov R2, A

Sel Rb1

Mov A, R1

Rrc A

Mov R2, A

Sel Rb0

Mov A, R2

Rrc A

Mov R2, A ;opr\_3\_2byte

Jc L2 ;MPP

Call Func

L2:;MPP

Djnz R7, L3 ;cycle

Mov A, R6

Cpl A

Anl A, R3

Mov R0, A

Mov A, R4

Sel Rb1

Mov R0, A

Sel Rb0

Jmp Eend

;==========================

Func:

Mov A, R2

Inc A

Mov R6, A

Sel Rb1

Mov A, R2

Jnc M1

Inc A

M1:

Mov R6, A

Sel Rb0 ;r2+1

Mov A, R6

Cpl A

Inc A

Mov R6, A

Sel Rb1

Mov A, R6

Cpl A

Jnc M2

Inc A

Sel Rb0

M2:

Mov R6, A

Sel Rb0 ;not(r2+1)+1

Mov A, R6

Add A, R5

Mov R6, A

Sel Rb1

Mov A, R6

Jnc M3

Inc A

M3:

Add A, R5

Mov R6, A

Sel Rb0 ;r5-r2-1

Clr C

Mov A, R6

Rlc A

Mov R6, A

Sel Rb1

Mov A, R6

Rlc A

Mov R6, A

Sel Rb0 ;2\*(r5-r2-1) opr\_4

Jf1 Ff1

Mov A, R6

Clr C

Add A, #77H

Mov R1, A

Sel Rb1

Mov A, R6

Jnc F2

Inc A

F2:

Mov R1, A

Sel Rb0 ;opr \_5

Ff1:

Mov A, R0

Dec A

Clr C

Rrc A

Mov R6, A

Sel Rb1

Mov R6, #0H

Sel Rb0 ;opr\_6

Mov A, R6

Inc A

Mov R5, A

Sel Rb1

Mov R5, #0H

Sel Rb0 ;opr\_7

Ret

Eend:

**3.1.3 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 3.1.3.1:

Таблица 3.1.3.1

|  |  |
| --- | --- |
| 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 приведенным алгоритмом:

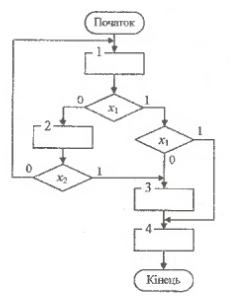


Рис. 3.1.3Блок-схема алгоритму

MovR1, #0H

Mov R2, #1H

Begg:Mov A, #0H

Anld P4, A

Mov A, #111B

Orld P4, A

Nop

Nop

Nop

Mov A, #110B

Anld P4, A

Mov R0, #9H

M1: Djnz R0, M1

Mov A, #10B

Nop

Anld P4, A

Mov A, #FEH

Mov T, A

Strt T

M2: Jtf M3

Jmp M2

M3: Stop Tcnt

Mov R0, #1H

M4: Djnz R0, M4

Nop

Mov A, #0H

Anld P4, A ;y2(fin:y1y2y3)

Mov R1, A

Jz M5

JmpEdd

M5:

Mov A, #10B

Orld P4, A

Mov A, #FDH

Mov T, A

Strt T

M6: Jtf M7

Jmp M6

M7: Stop Tcnt

Mov R0, #1H

M8: Djnz R0, M8

Nop

Mov A, #0H

Anld P4, A ;y2

Mov A, R2

JzBegg

Mov A, #1001B

Orld P4, A

Nop

Nop

Nop

Mov A, #1000B

Anld P4, A

Mov A, #F8H

Mov T, A

Strt T

M9: Jtf M10

Jmp M9

M10: Stop Tcnt

Mov R0, #5H

M11: Djnz R0, M11

Nop

Nop

Mov A, #0H

Anld P4, A ;y4y1

Edd:Mov A, #1000B

Orld P4, A

Mov A, #F8H

Mov T, A

Strt T

M12: Jtf M13

Jmp M12

M13: Stop Tcnt

Mov R0, #8H

M14: Djnz R0, M14

Nop

Nop

Mov A, #0H

Anld P4, A ;y4

**4. Программнаячасть**

**4.1 Алгоритм выполнения функции**



Рис. 4.1Блок-схема алгоритму функции

**4.2 Алгоритм умножения**

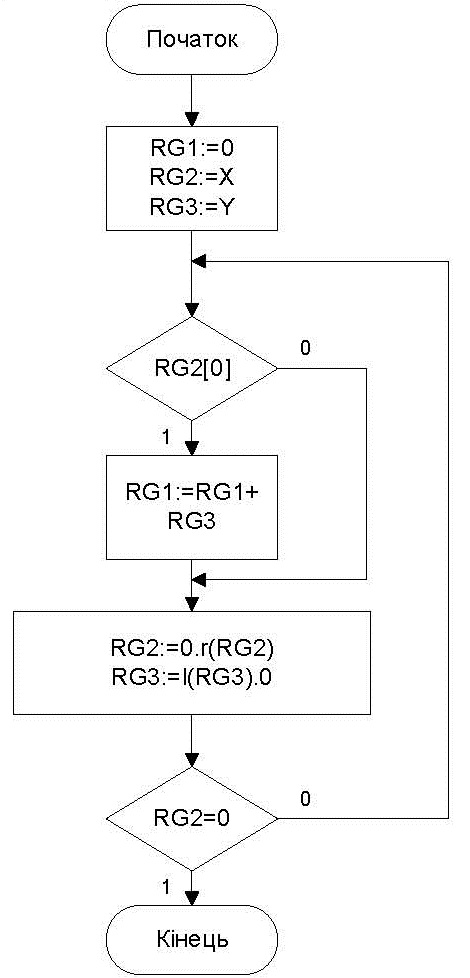


Рис. 4.2Микроалгоритмумножения ІІ способом

**4.3 Кодмикропрограммы**

jmp cross

multiply:

Mov A, R1

Rlc A

Mov A, R1

Jc M1

Jmp Bz1

M1:

Sel Rb1

Mov R0, #1H ;znak in R0

Sel Rb0

Anl A, #01111111B

Mov R1, A

Bz1:

Mov A, R3

Rlc A

Mov A, R3

Jc M2

Jmp Bz2

M2:

Sel Rb1

Mov A, R0

Inc A

Mov R0, A

Sel Rb0

Mov A, R3

Anl A, #01111111B

Mov R3, A

Bz2:

Bz4:

Bz5:

Mov A, R0 ;Rx[o] check

Rrc A

Mov A, R0

Jc M4

Jmp Bz3

M4:

Mov A, R2

Clr C

Add A, R4

Mov R4, A

Mov A, R3

Addc A, R5

Mov R5, A

sel rb1

Mov A, R4

sel rb0

Addc A, R6

Mov R6, A

sel rb1

Mov A, R5

sel rb0

Addc A, R7

Mov R7, A ;z=z+y

Bz3:

Mov A, R1

Clr C

Rrc A

Mov R1, A

Mov A, R0

Rrc A

Mov R0, A ;Rx=0.r(Rx)

Mov A, R2

Clr C

Rlc A

Mov R2, A

Mov A, R3

Rlc A

Mov R3, A

sel rb1

Mov A, R4

Rlc A

Mov R4, A

Mov A, R5

Rlc A

Mov R5, A

sel rb0 ;Ry=L(Ry).0

Mov A, R1

Jz M7

Jmp Bz4

M7:

Mov A, R0

Jz M8

Jmp Bz5

M8:

sel rb1

Mov A, R0

sel rb0

ret

mysub:

Mov A, R1

Rlc A

Mov A, R1

Jc L1

Jmp Lz1

L1:

Anl A, #01111111B

Mov R1, A

Mov A, R0

Cpl A

Inc A

Mov R0, A

Mov A, R1

Cpl A

Addc A,#00h

Mov R1,A ;check x znak

Lz1:

Mov A, R3

Rlc A

Mov A, R3

Jc L2

Jmp Lz2

L2:

Mov A, R3

Anl A, #01111111B

Mov R3, A

jmp D1

Lz2:

Mov A, R2

Cpl A

Inc A

Mov R2, A

Mov A, R3

Cpl A

Addc A,#00h

Mov R3,A ;check y znak

D1:

Clr C

Mov A, R0

Add A, R2

Mov R0, A

Mov A, R1

Addc A, R3

Mov R1, A ; add x and y

Clr C

Mov A, R1

Rlc A

Jc L3

Jmp Lz3

L3:

Clr C

Mov A, R0

Cpl A

Inc A

Mov R0, A

Mov A, R1

Cpl A

Addc A,#00h

orl A, #10000000b

Mov R1, A ;convert from DK in PK

Lz3:

ret

mul2:

Clr C

Mov A, R4

Rlc A

Mov R4, A

Mov A, R5

Rlc A

Mov R5, A

Mov A, R6

Rlc A

Mov R6, A

Mov A, R7

Rlc A

Mov R7, A

ret

outZPD:

Movx @R0, A

Inc R0

ret

copyXinY:

Mov A, R0

Mov R2, A

Mov A, R1

Mov R3, A ; copy data for mul

ret

clean:

Mov R0, #00h

Mov R1, #00h

Mov R2, #00h

Mov R3, #00h

Mov R4, #00h

Mov R5, #00h

Mov R6, #00h

Mov R7, #00h

ret

cross:

start:

Mov R0, #FFH ;x1

Mov R1, #0AH

Mov R2, #05H ;x2

Mov R3, #80H

callmysub

callcopyXinY

call multiply

call mul2

call mul2

call mul2 ; multiply 8

Mov R0, #00h

Mov A, R7

calloutZPD

Mov A, R6

calloutZPD

Mov A, R5

calloutZPD

Mov A, R4

calloutZPD ; write in ZPD

call clean

sel rb1

call clean

sel rb0

Mov R0, #0AH ;x3

Mov R1, #00H

Mov R2, #05H ;x4

Mov R3, #00H

callmysub

callcopyXinY

call multiply

Clr C

Mov R0, #03h

Movx A,@R0

Addc A, R4

Mov R4, A

Dec R0

Movx A,@R0

Addc A, R5

Mov R5, A

Dec R0

Movx A,@R0

Add A, R6

Mov R6, A

Dec R0

Movx A,@R0

Add A, R7

Mov R7, A ; add kvadrats

Mov R2, #0AH ;x7

Mov R3, #00H

Mov A, R3

Rlc A

Mov A, R3

Jc Lab2

Jmp Lzz2

Lab2:

Mov A, R3

Anl A, #01111111B

Mov R3, A

jmp Ddd1

Lzz2:

Mov A, R2

Cpl A

Inc A

Mov R2, A

Mov A, R3

Cpl A

Addc A,#00h

Mov R3,A ;check x7 znak

Ddd1:

Clr C

Mov A, R2

Addc A, R4

Mov R4, A

Mov A, R3

Addc A, R5

Mov R5, A

Mov A, R6

Addc A,#00h

Mov R6, A ;f=8\*(x1-x2)^2+(x3-x4)-x7 in DK