**Техническое задание на рассчётно-графическую работу по дисциплине**

**«Архитектура компьютера»**

**студентки Поважной Н. гр. ИВ-61**

№ зачётной книжки (6114) 10, (1011111100010 ) 2

|  |  |
| --- | --- |
| Выбор элементной базы | 1816ВЕ48 |
| Организация шины | С объединёнными шинами данных и адреса |
| КПП. КПДП | Децентрализованный |
| Способ умножения, деления, разрядность операндов | Умножение - 3 способ, деление – 2 способ, разрядность – 16 |
| Количество ВУ | 24 |
| Функция |  |
| Адреса для интерфейса внешнего устройства | 64 адреса для ВУ |
| Внутренняя память программ | 4К |
| Внешняя память данных | 8К |
| Функциональная схема |  |
| Дополнительные порты, периферийные адаптеры | ВВ55, Р4, Р7 |

Задание выдал

Задание принял

**1. Обзор МПС**

Разработанная МикроЭВМ предназначена для решения задачи обращения квадратной невырожденной матрицы методом Гаусса-Жордана с выбором главного элемента по столбцу.

В микроЭВМ шины адреса и данных объединены в ШАД. Разрядность ШАД - 32 бита, что обеспечивает адресацию 8МБ оперативной памяти и работу с 32-битными данными. Для работы с ШАД имеются интерфейсные регистры РА и РД. Регистр РА может выдавать на ШАД свою младшую, либо старшую часть.

Оперативная память с изменяемыми разделами (MVT). Каждой задаче выделяется объём памяти, необходимый для её решения (т.е. динамическое разделение). Выделяется квант памяти – это параграф (2i); чаще бывает, что квант времени – это 16 ячеек памяти. Компиляция программ производится в условных адресах. При загрузке программ в память к условному адресу прибавляется базовый адрес. Существуют такие дисциплины выделения раздела: 1. первый, который подходит по размеру; 2. наименьший, подходящий по размеру. При таких выделениях разделов возникают не очень хорошие последствия - фрагментация памяти. Дефрагментация производится с помощью специальных Utility.

Обмен данных между внешними устройствами и оперативной памятью осуществляется через КПДП. Обработка векторных прерываний от ВУ выполняется КПП. КПДП и КПП совмещены и распределены по интерфейсам внешних устройств.

Арифметическая и логическая обработка данных осуществляется в БОД. В нем имеется набор рабочих регистров, которые могут адресоваться либо регистрами RA, RB, либо адресными полями AdA, AdB регистра микрокоманд РМК.

ПМК - память микрокоманд - служит для хранения микропрограмм.

ЭВМ обладает блоком совмещения и выборки команд. БМУ управляет выборкой, распаковкой и выполнением микрокоманд. Через мультиплексор внешних условий он взаимодействует с ВУ, БУ SVS, БОД, РАПП и РАПДП. Буферы М, Р и V обеспечивают интерфейс между локальной шиной, шиной адреса ветвления и БМУ.

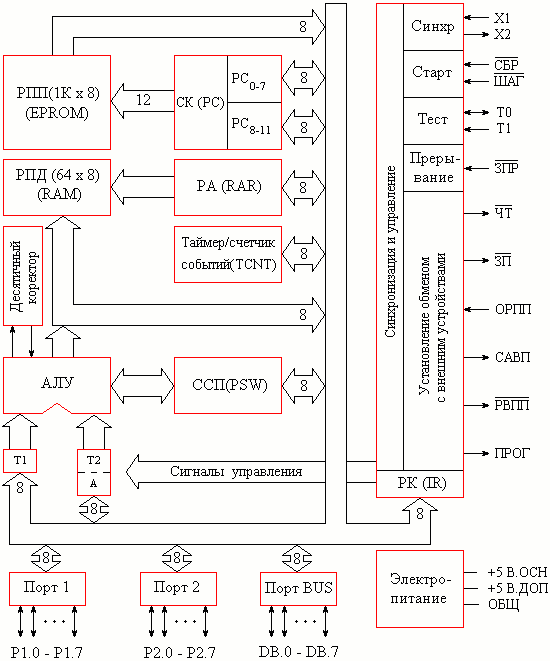


Рисунок 1.1 - Структурная схема МК48

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.

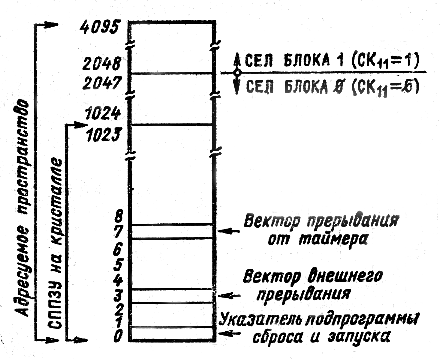
ОЭВМ серии 1816 требуют одного источника электро­питания напряжением +5В ± 10%, рассеивают мощность около 1,5 Вт и работают в диапазоне температур от 0 до 700С. по входам и выходам серии 1816 электрически совместимы с интегральными схемами ТТЛ.

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

**2. Архитектура МПС**

**2.1 Разработка структурной схемы**

**2.1.1 Память программ** (или - память команд, ПК) реализована в резидентном СППЗУ емкостью 1 Кбайт. Максимальное адресное пространство, отводимое для программ, составляет 4 Кбайт. Счетчик команд (СК, или программный счетчик - PC) содержит 12 разрядов, но инкрементируются в процессе счета только младшие 11 разрядов. Поэтому счетчик команд из предельного состояния 7FFH (если только по этому адресу не расположена команда передачи управления) перейдет в состояние 000H. Состояние старшего разряда счетчика команд может быть изменено специальными командами ("SEL МВ0" и "SEL MB1"). Подобный режим работы СК позволяет создать два блока памяти емкостью по 2 Кбайт каждый. Карта адресов памяти программ показана на рисунке



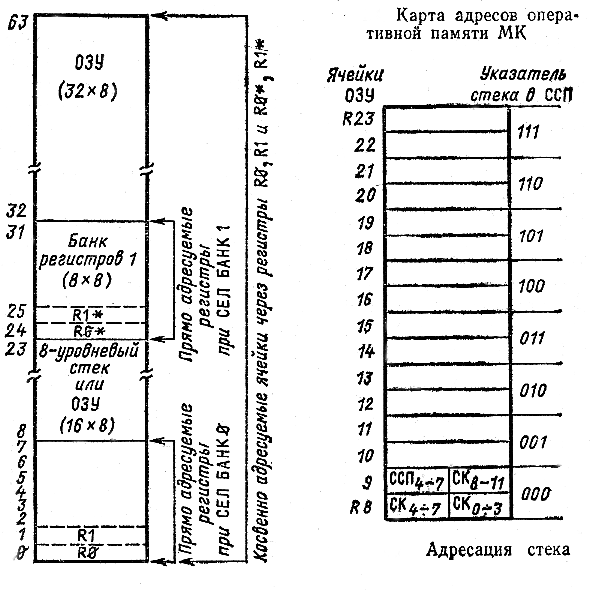
В резидентной памяти программ имеются три специализированных адреса:

адрес 0, к которому передается управление сразу после окончания сигнала системного сброса СБРОС; по этому адресу должна находиться команда безусловного перехода к началу программы (точка входа после начальной установки);

адрес 3, по которому расположен вектор прерывания от внешнего источника (точка входа при обработке внешних прерываний);

адрес 7, по которому расположен вектор прерывания от таймера или начальная команда подпрограммы обслуживания прерывания по признаку переполнения таймера/счетчика событий (точка входа при обработке прерываний по переполнению таймер-счетчика).

ПК служит для записи как команд, так и таблиц констант. Для отладки программ в микро-ЭВМ существует возможность отключения внутренней ПК подачей сигнала высокого уровня на вывод РВП (ЕА) микро-ЭВМ.

**2.1.2 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. Структура регистров (карта адресов) памяти данных МК представлена на следующем рисунке.   
  
  
  
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, R0\*, R1\*

Ячейки ОЗУ с адресами 8 - 23 адресуются указателем стека SP из ССП (PSW) [(см. рисунок регистра флагов в описании АЛУ)](http://www.kosmos.mk.ua/usmtu/mpk/k1816/k1816org2.html%22). Они могут быть использованы в качестве 8-уровневого стека (см. рис. карты адресов памяти МК выше). Если уровень вложенности подпрограмм меньше 8, то незадействованные в стеке регистры можно использовать как ячейки ОЗУ. Используемая ячейка определяется числом в 3-разрядном указателе стека SP, который является частью слова состояния программы. Порядок записи информации в стек приведен на рисунке. Загрузка в стек приводит к увеличению SP, чтение из стека - к уменьшению. При переполнении стека содержимое перезаписывается в регистры R8 и R9, т. е. SP изменяется со 111 на 000. При выполнении операций чтения из стека может происходить изменение SP с 000 на 111. Слово состояния программы РSW микро-ЭВМ может читаться и загружаться с помощью аккумулятора. Возможность загружать РSW необходима для перезапуска микро-ЭВМ после сбоя питания. Четыре старших бита PSW заносятся в стек и могут быть восстановлены при возврате с использованием специальной команды возврата.

МК 1816 не имеет команд загрузки байта в стек или его извлечения из стека, и в нем (в стеке) фиксируются только содержимое СК и старшая тетрада ССП (признаки). В связи с этим, не допускается во вложенных подпрограммах использование одноименных ячеек ОЗУ в качестве рабочих регистров. Если же такая необходимость возникает, то загрузку в стек и извлечение из него можно выполнять программно, путем передачи PSW в аккумулятор, выделения по маске указателя стека, передачи его в регистр косвенного адреса R0 или R1 с последующим обращением к ОЗУ по сформированому таким образом адресу вершины стека.

Практически все команды с обращением к ОЗУ оперируют с одним байтом. Однако по командам вызова и возврата осуществляется доступ к двухбайтным словам. В памяти данных слова хранятся так, что старший байт слова располагается в ячейке с большим адресом. Необходимо отметить, что в памяти программ порядок расположения байт по старшинству при хранении двухбайтных слов обратный.

В расширенных МК-системах, где используется внешнее ОЗУ, через регистры косвенного адреса R0 и R1 возможен доступ к внешней памяти объемом 256 байт. В случае необходимости в МК-системах можно применять внешнее ОЗУ требуемого объема, если, например, использовать 4 бита порта ввода - вывода информации для выбора одной из 16 страниц внешней памяти, каждая из которых имеет объем 256 байт. При этом 4-битный указатель страниц становится дополнением ССП и должен сохраняться в стеке при обработке прерываний.

**2.1.3 Внешнее устройство** - ВУ имеет фиксированный адрес в адресном пространстве периферийных устройств. Помимо регистра данных и регистра состояния каждое ВУ содержит регистр режима, регистр начальнога 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. Структурная схема интерфейса ВУ представлена на рисунке 2.1.3.1

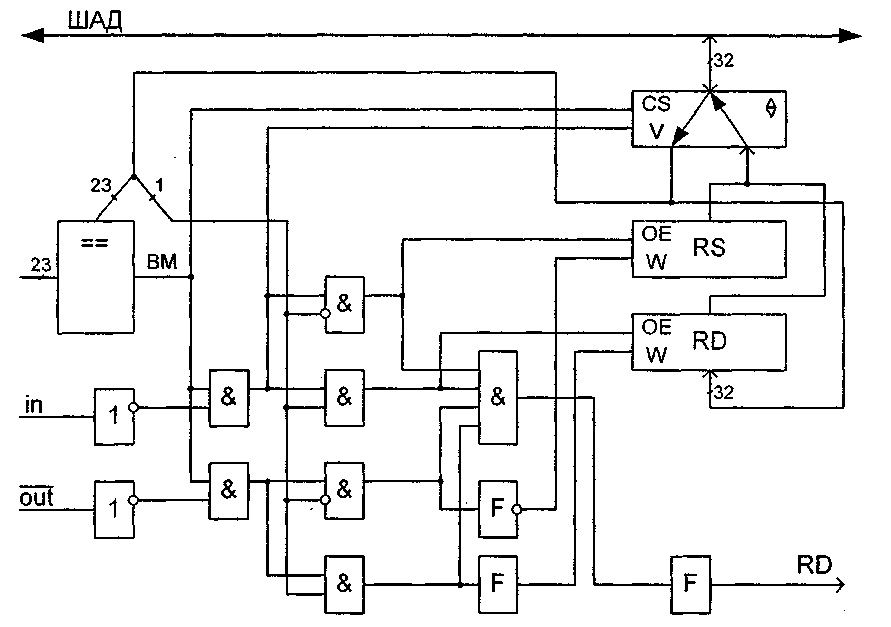


Рисунок 2.1.3.1 - Структурная схема интерфейса ВУ.

**2.1.4 Внешнее прерывание.**

Для использования безвекторных прерываний процессор имеет специальные входы для поступления запросов на прерывание программ. Для некоторых выходов существуют стандартные подпрограммы обслуживания с фиксированными начальными адресами. Безвекторные прерывания сигнализируют о сбое по питанию, сигнале от внешнего таймера.

Суть использования векторных прерываний в том, что любому внешнему устройству можно разрешить прерывание программы. Для подачи сигнала такого прерывания используется один вход процессора.

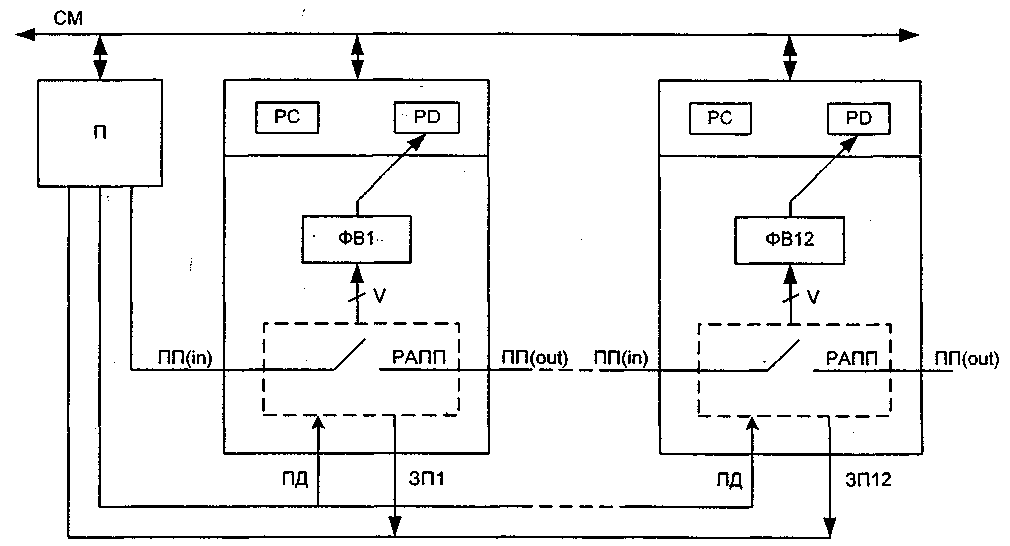
Идентификация устройства процессором осуществляется чтением по шине данных вектора (номера) внешнего устройства. Специальная процедура на аппаратном или микропрограммном уровне ставит в соответствие вектору начальный адрес подпрограммы обслуживания.

Требования прерывания проверяются после завершения каждой команды.

Векторные прерывания реализуются двумя методами:

1. использование централизованного контроллера приоритетных прерываний,
2. использование распределенного контроллера приоритетных прерываний.

В разрабатываемой ЭВМ реализован распределенный контроллер



На рисунке приняты обозначения:

П - процессор, РАГШ - распределенный арбитр приоритетных прерываний,

ФВ - формирователь вектора,

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

ПД - сигнал подготовки, ЗП - запрос на прерывание,

ITQ(in) - входящий сигнал подтверждение прерывания,

nn(out) - выходящий сигнал подтверждение прерывания.

**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**

1. 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 |

1. 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 приведённым алгоритмом:**

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

**Алгоритм:**

Начало

Конец

4\*(X1-X2) -> R2

X3&X4 -> R3

(X5-X6 - 1) -> A

A+R3 -> A

A+R2 -> A

A -> <14h>

**Код программы:**

Org 0

JMP Program\_start ; прыжок в начало программы

Program\_start:; начало программы

IN A,P1 ;подключение порта Р1 на ввод данных

MOV R1,A ;запись R1<-X1

IN A,P1 ;в аккумулятор A<-X2

CPL A

CLR C

ADDC A,1 ;перевод A в дополнительный код

ADD A,R1 ;разница (X1-X2)

RLC A ;4\*(X1-X2)

MOV R2,A ;разница (X1-X2)

IN A,P1

MOV R1,A ; запись в R1<-X3

IN A,P1 ; в аккумуляторA<-X4

ANL A,R1 ; X3&X4

MOV R3, A; результатR3<-X3&X4

IN A,P1

MOV R1,A ;запись R1<-X5

IN A,P1 ; A<-X6

CPL A

CLR C

ADDC A,1 ; A в доп.код

ADD A,R1 ; X5-X6

DEC A

MOV R3,A ; (X5-X6-1)

ADD A,R3

ADD A,R2

SEL RB0; выбираем регистр банка 0

MOV R0,#20

MOV @R0,A; косвенная адресация

**3.1.2 Команды передачи управления.**

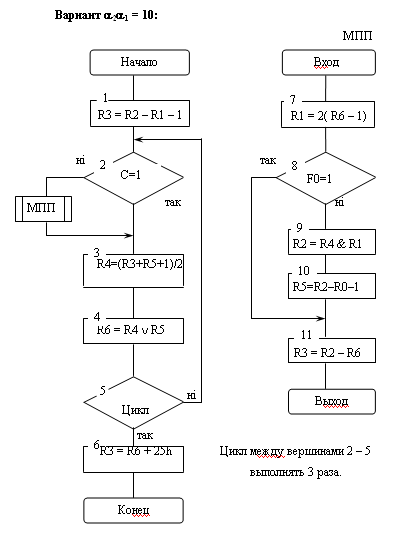
Группа команд передачи управления. Данную группу образуют 19 команд передачи управления, из них две команды безусловного перехода, 14 команд условного перехода, команда вызова подпрограмм и две команды возврата из подпрограмм. Пример некоторых операций показан в таблице 3.1.2.1:

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 |

**Пример программы с приведённым алгоритмом:**

**Алгоритм:**



**Листинг программы:**

;загрузка значений в регистры

IN A,BUS

MOV R0,A ;R0<-X0

IN A,BUS

MOЂV R1,A ;R1<-X1

IN A,BUS

MOV R2,A ;R2<-X2

IN A, BUS

MOV R3,A ;R3<-X3

IN A, BUS

MOV R4,A ;R4<-X4

IN A, BUS

MOV R5,A ;R5<-X5

IN A, BUS

MOV R6,A ;R6<-X6

IN A, BUS

MOV R7,A ;R7<-X7

;выполнение алгоритма

;блок 1

MOV A,R1

CPL A

CLR C

ADDC A,1;перевод в дополнительный код

MOV R1,A

MOV A,R2

ADD A,R1 ;R2-R1

DEC A ;R2-R1-1

MOV R3,A

;блок 2

;установка флага

ll2:

CLR C ;C=0

CPL C ;C=1

JC ll3 ;если C=1

CALL MPP: ;блок 7

MPP: MOV A,R6

DEC A ;R6-1

RLC A ;2\*(R6-1)

MOV R1,A

;блок 8

;установка флага

CLR F1 ;F1=0

;CPL F1 ;F1=1

JF1 ll11 ;если F1=1

ll9:

;блок 9

MOV A,R4

ANL A,R1 ;R4&R1

MOV R2,A ;R2<-R4&R1

;блок 10

MOV A,R0

CPL A

CLR C

ADDC A,1

MOV R0,A

MOV A,R2

ADD A,R0 ;R2-R0

DEC A ;R2-R0-1

MOV R5,A

ll11:

;блок 11

MOV A,R6

CPL A

CLR C

ADDC A,1

MOV R6,A

MOV A,R2

ADD A,R6 ;R2-R6

MOV R3,A

RET

JMP ll3

ll3:

;блок 3

MOV A,R3

ADD A,R5 ;R3+R5

INC A;R3+R5+1

CLR C

RRC A ;(R3+R5+1)/2

MOV R1,A ;R1<-(R3+R5+1)/2

;блок 4

MOV A,R4

ORL A,@R5;R4\/R5 косвенная адресация

MOV R6,A; R6<- R4\/R5

;блок 5

DJNZ R7, ll2

;блок 6

MOV A,R6

ADD A,#25H ;R6+#25H

MOV R3,A

END

**3.1.2 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 3.5:

Таблица 3.5

|  |  |
| --- | --- |
| 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:**

Y1Y5

X1

Y2

X2

X2

Y2 Y3 Y4

Y1 Y2 Y5

ANL P1, #40h

;формируем задержку 100 мкс

MOV R5, #20

ORL P1, #22h ;y1,y5

ll1: DJNZ R5, ll1

NOP

ANL P1, #40h ;сбрасываем y1,y5

;условие х1

X: INPUT A, P1

JB6 X1

;формирование задержки 725 мкс

MOV A,#FFh ; (-9) дк

MOV T, A

ORL P1,#4h ;у2

STRT T; включили счётчик

ll3: JTE ll2 ;анализируем TF

JMP ll0

ll2: ANL P1,#40h ;сбрасываем у2

JMP ll3

;анализмруем х2

ll0: INPUT A, P1

JB6 XX

JMP X

;условие х2

X1:

INPUT A, P1

JB6 X2

XX:

;формирование задержки 15 мкс

MOV R5, #3

ORL P1, #1Сh ; у2, у3, у4

ll4: DJNZ R5, ll4

NOP

ANL P1, #40h; сбрасываем у2, у3, у4

;формирование задержки 22 мкс

X2: MOV R5, #4h

ORL P1, #26h ;y1, y2, y5

ll: DJNZ R5, ll

NOP

ANL P1, #40h; сбрасываем y1, y2, y5

**4. Программная часть**

**4.1 Алгоритм программы**

Задание по варианту:

Y = X1-X2 + X5\*X6 + X9\*X9 + X10\*X10;

Операнды 16-разрядные, данные вводятся с порта Р1

Блок-схема алгоритма вычисления выражения

Вычисление (Х1- Х2)+Х5\*Х6, запись результата в 20h21h, а младшие разряды в 22h23h

Вычисление произведения Х5\*Х6, результат в R1R2, R5R6

Ввод Х5 в R1R2, Х6 в R3R4 c порта P2

Вычисление (Х1- Х2)+Х5\*Х6, запись результата в 20h21h, а младшие разряды в 22h23h

Вычисление произведения Х5\*Х6, результат в R1R2, R5R6

Ввод Х5 в R1R2, Х6 в R3R4 c порта P2

Вычисление (Х1- Х2)+Х5\*Х6, запись результата в 20h21h, а младшие разряды в 22h23h

Вычисление произведения Х5\*Х6, результат в R1R2, R5R6

Ввод Х5 в R1R2, Х6 в R3R4 c порта P2

Вычисление (Х1- Х2), запись в 16h17h

Ввод Х1 в R1R2, Х2 в R3R4 c порта P2

IN A, P1; ввод данных с порта Р1

MOV R1, A; старшие разряды X1 в R1;

IN A, P1;

MOV R2, A; младшие разряды X1 в R2;

IN A, P1;

MOV R3, A; старшие разряды X2 в R3;

IN A, P1;

MOV R4, A; младшие разряды X2 в R3;

CPL A; A= NOT A, инвертируем младшие разряды X2

CLR C; очистка признака С

ADD A, 1; окончательный перевод содержимого аккумулятора в дополнительный код

MOV R4, A; в R4 дополнительный код младших разрядов

MOV A, R3; теперь переходим к переводу старших разрядов Х2 в дополнительный код

CPL A; инвертирование содержимого аккумулятора

ADDC A, 0; окончательный перевод старших рязрядов X2 в доп.код

MOV R3, A; в R3R4 получили –Х2

MOV A, R4; переходим к вычислению разницы (Х1 – Х2)

ADD A, R2; начинаем с младших разрядов

MOV R0, #23; используем косвенную адресацию для записи результата в память:

MOV @R0, A; младших 8 разрядов, начиная с 23-й ячейки

MOV A, R3;

ADDC A, R1; с учётом переноса переходим к суммированию старших разрядов

MOV R0, #22; старшие разряды, начиная с 22-ой ячейки, записываем в память

MOV @R0, A;

IN A, P1; ввод данных с порта Р1

MOV R1, A; старшие разряды X5 в R1;

IN A, P1;

MOV R2, A; младшие разряды X5 в R2;

IN A, P1;

MOV R3, A; старшие разряды X6 в R3;

IN A, P1;

MOV R4, A; младшие разряды X6 в R4;

JMP MYMUL; подпрограмма умножения, Х5 в R1R2, Х6 в R3R4,

; результат получим 32-х разрядный, где старшие в R1R2, младшие R5R6

MOV R0, #23; для суммирования полученного результата умножения с имеющейся разницей

MOV A, @R0; считываем данные из памяти и суммируем

CLR C; очистка переноса

ADD A, R6; начинаем суммировать с младших разрядов

MOV @R0, A; результат вычислений записываем в память

MOV R0, #22;

|  |  |  |  |
| --- | --- | --- | --- |
| 14h | 15h | 16h | 17h |

MOV A, @R0; результат выражения (Х1-Х1)+Х5\*Х6 записали в память

ADDC A, R5; при этом использовали косвенную адресацию

MOV @R0, A;

MOV A, R2;

ADDC A, #0;

MOV R0, #21;

MOV @R0, A;

MOV A, R1;

ADDC A,#0;

MOV R0, #20;

MOV @R0, A;

IN A, P1; ввод данных с порта Р1

MOV R1, A; старшие разряды X9 в R1;

MOV R3, A; старшие разряды X9 в R3;

IN A, P1;

MOV R2, A; младшие разряды X9 в R2;

MOV R4, A; младшие разряды X9 в R4;

JMP MYMUL; подпрограмма умножения, Х9 в R1R2, и Х9 в R3R4,

; результат получим 32-х разрядный, где старшие в R1R2, младшие R5R6

MOV R0, #23; результат произведения 32-х разрядный

MOV A, @R0; суммируем с имеющимися результатами вычислений

CLR C;

ADD A, R6;

MOV @R0, A;

MOV R0, #22; в R0 записываем адрес необходимой ячейки для считывания данных

MOV A, @R0; в аккумулятор данные из памяти (косвенная адресация)

ADDC A, R5; суммирование с учётом переноса

MOV @R0, A; запись нового результата в память по тому же адресу

MOV R0, #21; адрес следующих 16-ти разрядов

MOV @R0, A;

ADDC A, R2;

MOV @R0, A;

MOV R0, #20;

MOV @R0, A;

ADDC A, R1; (Х1 – Х2) + Х5\*Х6 + Х9\*Х9

|  |  |  |  |
| --- | --- | --- | --- |
| 14h | 15h | 16h | 17h |

MOV @R0, A; получили опять конечные данные в памяти

IN A, P1; ввод данных с порта Р1

MOV R1, A; старшие разряды X10 в R1;

MOV R3, A; старшие разряды X10 в R3;

IN A, P1;

MOV R2, A; младшие разряды X10 в R2;

MOV R4, A; младшие разряды X10 в R4;

JMP MYMUL; подпрограмма умножения, Х10 в R1R2, и Х10 в R3R4,

; результат получим 32-х разрядный, где старшие в R1R2, младшие R5R6

MOV R0, #23; считываем имеющиеся данные из памяти и суммируем

MOV A, @R0; полученное произведение Х10\*Х10

CLR C; записываем полученный результат по таким же адресам

ADD A, R6; используя при этом косвенную адресацию

MOV @R0, A;

MOV R0, #22; адрес ячейки памяти

MOV A, @R0; считываем данные из памяти по указанному адресу

ADDC A, R5; сумма с учётом переноса

MOV @R0, A; записываем в память по указанному адресу полученный результат

MOV R0, #21;

MOV @R0, A;

ADDC A, R2;

MOV @R0, A;

MOV R0, #20;

MOV @R0, A; (Х1 – Х2) + Х5\*Х6 + Х9\*Х9 + Х10\*Х10

ADDC A, R1;

|  |  |  |  |
| --- | --- | --- | --- |
| 14h | 15h | 16h | 17h |

MOV @R0, A; в памяти получили результат

END;

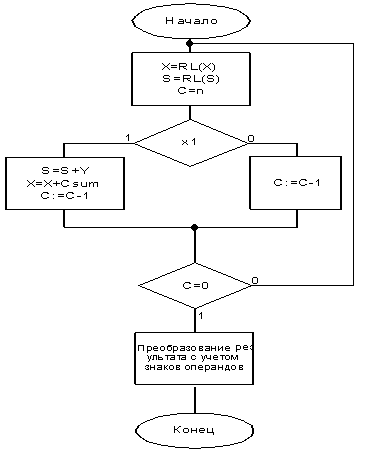
**4.2 Алгоритм умножения двух чисел**

Блок-схема алгоритма умножения 3-им способом:

1-й множитель Х

2-й множитель У

S – сумма накопления, операнды 16-разрядные



1

**X**🡨 n+1

2 **S**🡨 n+1

1 **СМ** n

1 **Y** n

X1

+1

MYMUL:

CLR F0; очистка признака F0

CLR F1; очистка признака F1

MOV A, R1; в аккумулятор старшие разряды 1го множителя для проверки знака

ANL A, #10000000B; если Х отрицательное,

JZ Point10; то инвертируем знаковый разряд

MOV A, R1; и инвертируем признак F0

ANL A, #01111111B; выделяем значимую часть множителя (старшие разряды)

MOV R1, A; в R1 положительное число

CPL F0; инвертируем признак F0 (признак 1-го множителя)

Point10:

MOV A, R3; загоняем в аккумулятор старшие разряды 2-го множителя для проверки знака

ANL A, #10000000B ; если Y отрицательное,

JZ Point4; то инвертируем знаковый разряд

MOV A, R3; и инвертирум признак F1

ANL A, #01111111B

MOV R3, A; в старших разрядах 2-го множителя положительное число

CPL F1; инвертируем признак F1

MOV A, R1

ANL A, #10000000B

JZ Point4

MOV A, R1

ANL A, #01111111B

MOV R1, A

CPL F0

JMP Point4

Point4:

MOV R7, #16; в счётчик записываем разрядность операндов - 16

Point3:

MOV A, R6; сдвиг младших разрядов S влево

RLC A

MOV R6, A

MOV A, R5; сдвиг влево накопителя суммы (старших разрядов)

RLC A

MOV R5, A

MOV A, R2; сдвиг влево младших разрядов 1-го множителя

RLC A

MOV R2, A

MOV A, R1; Сдвиг влево Х (старшие разряды)

RLC A

MOV R1, A

JC Point1

DEC R7; инкремент счётчика

MOV A, R7; проверяем следует ли заканчивать цикл

JNZ Point3; зацикливаем

JMP Next2

Point1:

MOV A, R6; если получили перенос = 1

ADD A, R4; к накопителю суммы прибавляем 2-ой множитель

MOV R6, A; назад в S

JC Point2

JMP Next1

Point2:

INC R5; при возникновении переноса, следует прибавить 1 к старшим разрядам S

Next1:

MOV A, R5; суммируем 2-ой множитель и S (старшие разряды)

ADD A, R3

MOV R5, A

JC Point8

JMP Next3

Point8:

INC R2; инкремент младших рязрядов 1-го множителя

JC Point9; при возникновении переноса

JMP Next3

Point9:

INC R1; инкремент старших разрядов 1-го множителя

Next3:

CLR C; очистка переноса

DEC R7; декремент счётчика

MOV A, R7; проверка счётчика на 0

JNZ Point3

Next2:

JF0 Point5; определение знака результата

JMP Point6

Point5:

JF1 Point7

JMP Point11

Point7:

MOV A, R1; формирование положительного результата

ANL A, #01111111B

MOV R1, A; если были одинаковые знаки множителей

JMP Progend

Point6:

JF1 Point11

JMP Progend

Point11:

MOV A, R1; если знаки множителей различны, то результат отрицательный

ORL A, #10000000B

MOV R1, A

Progend:

RET; возврат в программу

**4.3 Алгоритм деления двух чисел**

Ниже приведён алгоритм деления двух чисел

Подпрограмма деления приведена в программе ниже

P3 🡨

P2

2n

2n

CM

2n

P1 🡪

;======== ПОДПРОГРАММА ДЕЛЕНИЯ ДВУХ ЧИСЕЛ Х\*У =======

; R2 – старшие разряды Х

; R3 – младшие разряды Х

; R4 – старшие разряды У

; R5 – младшие разряды У

; R0 – счётчик

DIVI:

MOV R0, #10H

MOV R1,#32H

MOV A,R4

MOV @R1,A ; Пересылка во внутр. память старших разрядов

MOV R1,#33H

MOV A,R5

MOV @R1,A ; Пересылка во внутр. память старших разрядов

MOV R4,#0

MOV R5,#0

SUBI:

CLR C

MOV R1,#35H

MOV A,@R1 ;Считывание из памяти разрядов

CPL A ; Инвертирование А

INC A ; Получаем А в доп. коде

ADD A,R5

MOV R5,A ;Промежуточные данные в R5

MOV R1,#34H

MOV A,@R1 ;Считывание из памяти разрядов

CPL A ; Инвертирование А

ADDC A,R4

MOV R4,A ;Промежуточные данные в R4

MOV R1,#33H

MOV A,@R1 ;Считывание из памяти разрядов

CPL A ; Инвертирование А

ADDC A,R3

MOV R3,A ;Промежуточные данные в R3

MOV R1,#32H

MOV A,@R1 ;Считывание из памяти разрядов

CPL A ; Инвертирование А

ADDC A,R2

MOV R2,A ;Промежуточные данные в R2

JMP SH

ADDI:

CLR C

MOV R1,#35H

MOV A,@R1 ;Считывание из памяти разрядов

ADD A,R5

MOV R5,A ;Промежуточные данные в R5

MOV R1,#34H

MOV A,@R1 ;Считывание из памяти разрядов

ADDC A,R4

MOV R4,A ;Промежуточные данные в R4

MOV R1,#33H

MOV A,@R1 ;Считывание из памяти разрядов

ADDC A,R3

MOV R3,A ;Промежуточные данные в R3

MOV R1,#32H

MOV A,@R1 ;Считывание из памяти разрядов

ADDC A,R2

MOV R2,A ;Промежуточные данные в R2

JMP SH

SH:

NOP

CLR C

CLR F1

JB7 SHZ

CPL C

CPL F1

SHZ:

MOV A,R7

RLC A

MOV R7,A

MOV A,R6

RLC A

MOV R6,A

DJNZ R0, OK

JMP ENDING

OK:

MOV R1,#35H

MOV A,@R1 ;Считывание из памяти разрядов

RRC A

MOV @R1,A ;Запись обратно в память

MOV R1,#34H

MOV A,@R1 ;Считывание из памяти разрядов

RRC A

MOV @R1,A ;Запись обратно в память

MOV R1,#33H

MOV A,@R1 ;Считывание из памяти разрядов

RRC A

MOV @R1,A ;Запись обратно в память

MOV R1,#32H

MOV A,@R1 ;Считывание из памяти разрядов

RRC A

MOV @R1,A ;Запись обратно в память

CLR C

JF1 SUBI

JMP ADDI

ENDING:

RETR ; Возврат из подпрограммы