**1. Обзор МПС**

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.

Структурная схема МК48 показана на рисунке 1.1. Основу структуры МК образует внутренняя двунаправленная 8-битная шина, которая связывает между собой все устройства БИС: арифметико-логическое устройство (АЛУ), устройство управления, память и порты ввода/вывода информации.

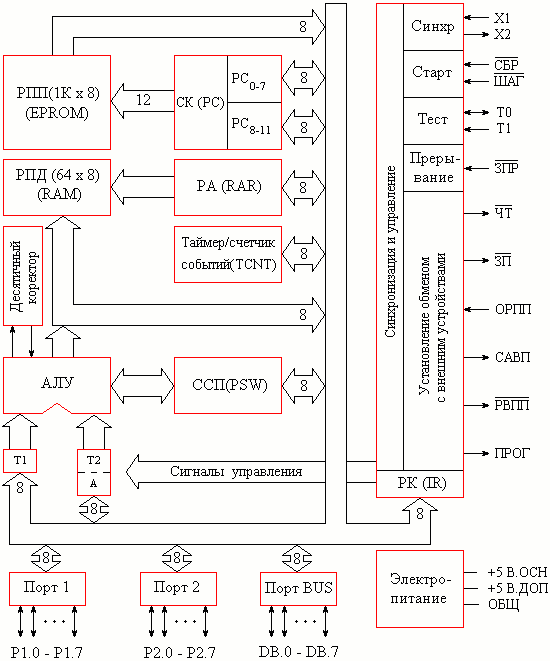


Рисунок 1.1 - Структурная схема МК48

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**

**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+1) – (X5 V X6-1) / 2

**Алгоритм:**

Начало

Конец

4\*(X1&X2) -> R1

-(X3-X4+1) -> R2

-(X5 V X6-1) / 2 -> A

A+R1 -> A

A+R2 -> A

A -> <10h>

**Код программы:**

Org 0

Jmp Program\_start ; прыжок в начало программы

Program\_start:; начало программы

IN A,P1 ;ввод данных из порта P1

MOV R1,A ;R1🡨X1

IN A,P1 ;A🡨X2

CLR C ;обнуление С

ANL A,R1 ;Логическое И

RLC A ;2\*(X1&X2)

CLR C ;обнуление С

RLC A ;4\*(X1&X2)

MOV R1,A ;4\*(X1&X2) 🡪 R1

IN A,P1 ;ввод данных из порта P1

MOV R2,A ;R1 🡨 X3

IN A,P1 ;A 🡨 X4

CPL A ;инвертирование А (-Х4)

ADD A,1 ; -X4+1

ADD A,R2 ;-X4+1+X3

CPL A ;инвертирование А -(X3-X4+1)

MOV R2, A ;R2🡨 -(X3-X4+1)

IN A,P1 ;ввод данных из порта P1

MOV R3,A ;R3🡨X5

IN A,P1 ;A🡨X6

XRL A,R3 ;Логическо ИЛИ (X5 V X6)

DEC A ; X5 V X6 – 1

CPL A ; -(X5 V X6 – 1)

RRC A ; -(X5 V X6 – 1)/2

ADD A,R1

ADD A,R2

SEL RB0

MOV R0,#10

MOV @R0,A

**3.1.2 Команды передачи управления.**

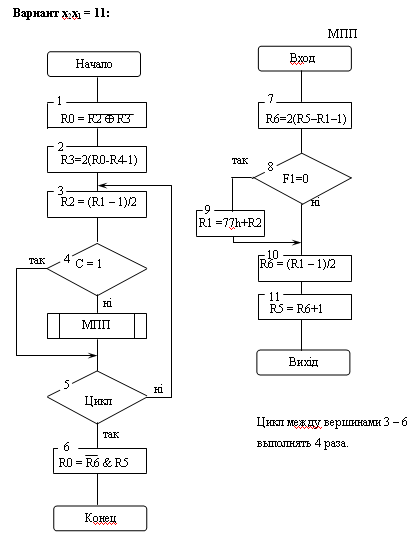
Группа команд передачи управления. Данную группу образуют 19 команд передачи управления, из них две команды безусловного перехода, 14 команд условного перехода, команда вызова подпрограмм и две команды возврата из подпрограмм. Пример некоторых операций показан в таблице 3.1.2.1:

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:**

In A, P1

Mov R1, A

In A, P1

Mov R2, A

In A, P1

Mov R3, A

In A, P1

Mov R4, A

In A, P1

Mov R5, A

In A, P1

Mov R6, A

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

; ВЫПОЛНЕНИЕ ПЕРВОГО БЛОКА

Mov A, R2 ; R0 --> R2

Xrl A, R3 ; R0 XOR R2

Clr A ; NOT R0

Mov R0, A

; ВЫПОЛНЕНИЕ 2-ГО БЛОКА

Mov A, R4

Cpl A

;DEC R3

Add A, R4

Add A, R0

Clr C

Rlc A

Mov R3, A

Mov R7, 4 ; КОЛИЧЕСТВО ЦИКЛОВ

Ll1:; ТОЧКА НАЧАЛА ЦИКЛА

;ВЫПОЛНЕНИЕ 3-ГО БЛОКА

Mov A, R1

Dec A

Clr C

Rrc A

Mov R2, A

Jc Ll2 ;ПРОВЕРКА ФЛАГА НА 1

CALL MMP

Ll2:;ТОЧКА ПЕРЕХОДА ПО УСЛОВИЮ С=1

Djnz R7, LL1 ; ПРОВЕРКА НА ОКОНЧАНИЕ ЦИКЛА И ПЕРЕХОД

;ВЫПОЛНЕНИЕ 6-ГО БЛОКА

Mov A, R6

Cpl A

Anl A, R5

Mov R0, A

End

MMP :

;ВЫПОЛНЕНИЕ 7-ГО БЛОКА

Mov A, R1

Cpl A

;DEC R6

Add A, R5

Clr C

Cpl A

Mov R6, A

;ВЫПОЛНЕНИЕ 8-ГО БЛОКА

Jf1 Ll4 ; ПРОВЕРКА F1 НА 0

; ВЫПОЛНЕНИЕ 9-ГО БЛОКА

Mov A, 77h

Add A, R2

Mov R1, A

Ll4:

; ВЫПОЛНЕНИЕ 10-ГО БЛОКА

Mov A, R1

Dec A

Clr C

Rrc A

Mov R6, A

; ВЫПОЛНЕНИЕ 11-ГО БЛОКА

Mov A, R6

Inc A

Mov R1, A

Retr

**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 4.5:

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:**

**Y1 Y5**

**Y2**

**Y1 Y2 Y5**

**Y1 Y5**

**X2**

**Y3 Y2 Y4**

**Конец**

**Начало**

**X1**

;Настройка порта на входные данные

ANLD P4,#10000000b ; X1

MOVD A,P4

JB7 LL2 ;ПРОВЕРЯЕМ Х1

;Х1=0 ФОРМИРУЕТСЯ СИГНАЛ У1,У5 С ЗАДЕРЖКОЙ 222мс

;ФОРМИРУЕМ ЗАДЕРЖКУ 222мс

LL0:MOV A,#0FDH ;(-2) ;160 мс

MOV T,A

ORLD P4,#1010b ;ФОРМИРОВАНИЕ СИГНАЛОВ У1,У5

STRT T

LLA:JTF LLB

JMP LLA

LLB:

MOV R5,0AH

LLQ:DJNZ R5,LLQ ;60мс

NOP ;2мс

ANLD P4,#10000000B ;

;УСЛОВНЫЙ ПЕРЕХОД ПО Х2

ANLD P5,#01000000B ;X2

MOVD A,P5

JB6 LL8 ; ПРОВЕРЯЕМ Х2

;Х2 = 0 И ФОРМИРУЕМ СИГНАЛ У2 У3 У4 С ЗАДЕРЖКОЙ 720мс

;ФОРМИРУЕМ ЗАДЕРЖКУ 720мс

ANLD P4,#01000000B ;X2

MOV A,#0F7H ;(-9)

MOV T,A

ORLD P4,#100b ;ФОРМИРОВАНИЕ СИГНАЛОВ У2

ORLD P5,#101000b ;ФОРМИРОВАНИЕ СИГНАЛОВ У3,Y4

STRT T

LLC:JTF LLD

JMP LLC

LLD:

ANLD P4,#01000000b ; X2

ANLD P5,#01000000b ; X2

JMP LL0 ; ПРИЖОК СОГЛАСНО АЛГОРИТМУ НА 1-Ю ВЕРШИНУ

;X1 = 1 ФОРМИРОВАНИЕ СИГНАЛА У2 С ЗАДЕРЖКОЙ 15мс

LL2:MOV R5,#3

ORLD P4,100B ;ЗАГРУЖАЕМ У2

LLL:DJNZ R5,LLL

ANLD P4,#10000000b ; X1

;X2=0 ИЛИ ПРОДОЛЖЕНИЕ УСЛОВИЯ Х1=1 ФОРМИРУЕТСЯ СИГНАЛ У1 У2 У5 С ЗАДЕРЖКОЙ 500

LL8:MOV A,#0FAH ;(-6) 480мс

MOV T,A

ORLD P4,1110B ; ФОРМИРОВАНИЕ СИГНАЛОВ У1 У2 У5

STRT T

LLE:JTF LLF

JMP LLE

LLF:

MOV R5,#3

LLK:DJNZ R5,LLK ;18мс

NOP ; 2мс

ANLD P4,#10000000b ; X1

; ФОРМИРОВАНИЕ У2 У5 С ЗАДЕРЖКОЙ 222мс

MOV A,FEH ;(-2) 160 мс

MOV T,A

ORLD P4,110B

STR T

LLT:JTF LLR

JMP LLT

LLR:

MOV R5,0AH

LLW:DJNZ R5,LLW ;60мс

NOP ;2мс

ANLD P4,#10000000b ; X1

**4. Программная часть**

Х3\*Х4 Считывание с порта Р1

R1 - старшие разряды Х3

R2 - младшие разряды Х3

R3 - старшие разряды Х4

R4 - младшие разряды Х4

R6=0 R5 = 0

R7 = 10H - счётчик

R0= 10H - адрес начала

CALL MUL

Результат: в памяти по адресам

13h 12h 11h 10h

Х7/Х8 Считывание с порта Р1

R2 - старшие разряды Х7

R3 - младшие разряды Х7

R4 - старшие разряды Х8

R5 - младшие разряды Х8

R6 = 0 R7 = 0

CALL DIVI

результат получаем в **R6.R7**

и записываем его в память

**R6 14h**

**R7 9h**

Х9\*Х9 Считывание с порта Р1

R1 - старшие разряды Х9

R2 - младшие разряды Х9

R3 - старшие разряды Х9

R4 - младшие разряды Х9

R6=0 R5 = 0

R7 = 10H - счётчик

R0= 15H - адрес начала

CALL MUL

Результат: в памяти по адресам

18h 17h 16h 15h

Х10\*Х10 Считывание с порта Р1

R1 - старшие разряды Х10

R2 - младшие разряды Х10

R3 - старшие разряды Х10

R4 - младшие разряды Х10

R6=0 R5 = 0

R7 = 10H - счётчик

R0= 15H - адрес начала

CALL MUL

Результат: в памяти по адресам

22h 21h 20h 19h

Суммирование результатов

Которые находятся по адресам:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Х3\*Х4 | 13H | 12H | 11H | 10H |
| Х7/Х8 |  |  |  | 14H . 9H |
| Х9\*Х9 | 18H | 17H | 16H | 15H |
| Х10\*Х1 | 22H | 21H | 20H | 19H |

Итоговый результат будет находится по адресам

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| X | 13H | 12H | 11H | 10H . 9H |

**4.1 Алгоритм программы**

Ниже приведён алгоритм основной программы, а также алгоритмы

умнжения и деления 2-х числе.

Х = Х3\*Х4 + Х7/Х8 + Х9\*Х9 + Х10\*Х10

Ниже представлен алгоритм выполнения функции Х

|  |  |
| --- | --- |
| … | … |
| 9h | Остача от деления Х7/Х8 |
| 10h | Старшие разряды Х3\*Х4 |
| 11h | Старшие разряды Х3\*Х4 |
| 12h | Младшие разряды Х3\*Х4 |
| 13h | Младшие разряды Х3\*Х4 |
| 14h | Частное от деления Х7/Х8 |
| 15h | Старшие разряды Х9\*Х9 |
| 16h | Старшие разряды Х9\*Х9 |
| 17h | Младшие разряды Х9\*Х9 |
| 18h | Младшие разряды Х9\*Х9 |
| 19h | Старшие разряды Х10\*Х10 |
| 20h | Старшие разряды Х10\*Х10 |
| 21h | Младшие разряды Х10\*Х10 |
| 22h | Младшиие разряды Х10\*Х10 |
| … | … |

**4.2 Алгоритм умножения двух чисел**

Ниже приведён алгоритм умножения двух 16 разрядных чисел

P3:=L1(P3)

P2:=R1(P2)

C:= C - 1

C := 16

C = 0

Х1\*=1

P2 := P2 + P1

Нет

Да

Да

Нет

Где Р1 – множитель, Р3 – множитель, Р2 – результат

C = 16 – счётчик

Р3 – 16 разрядный

Р1, Р2 – 32 разрядный

Х1\* - старший разряд Р3

L1(P3) – сдвиг Р3 влево на 1 разряд

R1(P2) - сдвиг Р2 вправо на 1 разряд

**4.3 Алгоритм деления двух чисел**

**4.4 Программа вычисления функции**

Программа вычисления функции содержит в себе подпрограммы вычисления деления и умножения двух чисел разрядностью 16

;Основная программа данные считываем с порта Р1 используем банк регистра нулевой

SEL RBO

;========Считываем с порта Р1 значение Х3============

IN A,P1

MOV R1,A ; старшие разряды Х3

IN A,P1

MOV R2,A; младшие разряды Х3

;========Считываем с порта Р1 значение Х3============

IN A,P1

MOV R3,A ; старшие разряды Х4

IN A,P1

MOV R4,A; младшие разряды Х4

;========Производим подготовку к умножению Х3\*Х4===

MOV R6,#0

MOV R5,#0

MOV R7,#10H ; счётчик в умножении

MOV R0,#10H ; адрес начала младших разрядов результата

CALL MULL; Вызов подпрограммы умножения

|  |  |  |  |
| --- | --- | --- | --- |
| 13h | 12h | 11h | 10h |

;результат получили Х3\*Х4=

;========Считываем с порта Р1 значение Х7============

IN A,P1

MOV R2,A ; старшие разряды Х7

IN A,P1

MOV R3,A; младшие разряды Х7

;========Считываем с порта Р1 значение Х8============

IN A,P1

MOV R4,A ; старшие разряды Х8

IN A,P1

MOV R5,A; младшие разряды Х8

;========Производим подготовку к делению Х7/Х8===

MOV R6,#0

MOV R7,#0

CALL DIVI ; вызов подрограммы деления

; результат получаем в **R6.R7**

; записуем результат в память

MOV RO,#14H

MOV A,R6

MOV @RO,A ; запись частного в память

MOV RO,#9H

MOV A,R7

MOV @RO,A ; запись остатка в память

;========Считываем с порта Р1 значение Х9============

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

|  |  |  |  |
| --- | --- | --- | --- |
| 18h | 17h | 16h | 15h |

; результат получили Х9\*Х9=

;========Считываем с порта Р1 значение Х10============

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,#19H ; адрес начала младших разрядов результата

CALL MULL; Вызов подпрограммы умножения

|  |  |  |  |
| --- | --- | --- | --- |
| 22h | 21h | 20h | 19h |

; результат получили Х9\*Х9=

;========Производим суммирование результатов===

; Х = Х3\*Х4 + Х7/Х8 + Х9\*Х9 + Х10\*Х10

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Х3\*Х4 | 13H | 12H | 11H | 10H |
| Х7/Х8 |  |  |  | 14H . 9H |
| Х9\*Х9 | 18H | 17H | 16H | 15H |
| Х10\*Х1 | 22H | 21H | 20H | 19H |
| X | 13H | 12H | 11H | 10H . 9H |

;========Производим суммирование двух первых операндов Х3\*Х4 + Х7/Х8 ===

MOV RO,#10H

MOV A,@RO ; считывание младших разрядов результата умножения Х3\*Х4

MOV R2,A

MOV R1,#14H

MOV A,@R1 ; считывание частного от деления Х7/Х8

ADD A,R2 ;суммирование результата и фиксация признака переноса

MOV @R0,A ; запись результата в память в ячейку 10h

MOV RO,#11H

MOV A,@RO ; считывание разрядов результата умножения Х3\*Х4

ADDС A,#0 ; суммирование результата и переноса

MOV @R0,A ; запись результата в память в ячейку 11h

MOV RO,#12H

MOV A,@RO ; считывание разрядов результата умножения Х3\*Х4

ADDС A,#0 ; суммирование результата и переноса

MOV @R0,A ; запись результата в память в ячейку 12h

MOV RO,#13H

MOV A,@RO ; считывание разрядов результата умножения Х3\*Х4

ADDС A,#0 ; суммирование результата и переноса

MOV @R0,A ; запись результата в память в ячейку 13h

;========Производим Суммирование двух первых операндов (Х3\*Х4 + Х7/Х8) и Х9\*Х9 ===

CLR C

MOV RO,#16H

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

MOV R2,A

MOV R1,#10H

MOV A,@R1 ; считывание разрядов результата cуммирования (Х3\*Х4 + Х7/Х8)

ADD A,R2 ; суммирование результататов и фиксация признака переноса

MOV @R1,A ; запись результата в память в ячейку 10h

MOV RO,#17H

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

MOV R2,A

MOV R1,#11H

MOV A,@R1 ; считывание разрядов результата cуммирования (Х3\*Х4 + Х7/Х8)

ADDC A,R2 ; суммирование результататов и фиксация признака переноса

MOV @R1,A ; запись результата в память в ячейку 11h

MOV RO,#18H

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

MOV R2,A

MOV R1,#12H

MOV A,@R1 ; считывание разрядов результата cуммирования (Х3\*Х4 + Х7/Х8)

ADDC A,R2 ; суммирование результататов и фиксация признака переноса

MOV @R1,A ; запись результата в память в ячейку 12h

MOV RO,#19H

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

MOV R2,A

MOV R1,#13H

MOV A,@R1 ; считывание разрядов результата cуммирования (Х3\*Х4 + Х7/Х8)

ADDC A,R2 ; суммирование результататов и фиксация признака переноса

MOV @R1,A ; запись результата в память в ячейку 13h

;========Производим Суммирование трёх первых операндов (Х3\*Х4 + Х7/Х8 + Х9\*Х9) и Х10\*Х10 ===

CLR C

MOV RO,#19H

MOV A,@RO ; считывания разрядов результата умножения Х10\*Х10

MOV R2,A

MOV R1,#10H

MOV A,@R1 ; считывание разрядов результата cуммирования (Х3\*Х4 + Х7/Х8 + Х9\*Х9)

ADD A,R2 ; суммирование результататов и фиксация признака переноса

MOV @R1,A ; запись результата в память в ячейку 10h

MOV RO,#20H

MOV A,@RO ; считывания разрядов результата умножения Х10\*Х10

MOV R2,A

MOV R1,#11H

MOV A,@R1 ; считывание разрядов результата cуммирования (Х3\*Х4 + Х7/Х8 + Х9\*Х9)

ADDC A,R2 ; суммирование результататов и фиксация признака переноса

MOV @R1,A ; запись результата в память в ячейку 11h

MOV RO,#21H

MOV A,@RO ; считывания разрядов результата умножения Х10\*Х10

MOV R2,A

MOV R1,#12H

MOV A,@R1 ; считывание разрядов результата cуммирования (Х3\*Х4 + Х7/Х8 + Х9\*Х9)

ADDC A,R2 ; суммирование результататов и фиксация признака переноса

MOV @R1,A ; запись результата в память в ячейку 12h

MOV RO,#22H

MOV A,@RO ; считывания разрядов результата умножения Х10\*Х10

MOV R2,A

MOV R1,#13H

MOV A,@R1 ; считывание разрядов результата cуммирования (Х3\*Х4 + Х7/Х8 + Х9\*Х9)

ADDC A,R2 ; суммирование результататов и фиксация признака переноса

MOV @R1,A ; запись результата в память в ячейку 13h

;======== В результате получили : ==================

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| X | 13H | 12H | 11H | 10H . 9H |

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

; R0 – адрес начала млаших 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 ; Старшиер 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 сумируем. Резльтат записываем в память

MOV A,@R0 ; считали данные

ADD A,R6 ; просумировали

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 ; просумировали 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 ; просумировали 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 ; просумировали 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 декримент R7

RET ; Возврат из подрограммы

;======== ПОДПРОГРАММ ДЕЛЕНИЯ ДВОХ 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 с 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 с памяти разрядов

CPL A ; Инвертирование А

ADDC A,R4

MOV R4,A ;Промежуточные данные в R4

MOV R1,#33H

MOV A,@R1 ;Считывание с памяти разрядов

CPL A ; Инвертирование А

ADDC A,R3

MOV R3,A ;Промежуточные данные в R3

MOV R1,#32H

MOV A,@R1 ;Считывание с 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 с памяти разрядов

ADD A,R5

MOV R5,A ;Промежуточные данные в R5

MOV R1,#34H

MOV A,@R1 ;Считывание с памяти разрядов

ADDC A,R4

MOV R4,A ;Промежуточные данные в R4

MOV R1,#33H

MOV A,@R1 ;Считывание с памяти разрядов

ADDC A,R3

MOV R3,A ;Промежуточные данные в R3

MOV R1,#32H

MOV A,@R1 ;Считывание с 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 с памяти разрядов

RRC A

MOV @R1,A ;Запись обратно в память

MOV R1,#34H

MOV A,@R1 ;Считывание с памяти разрядов

RRC A

MOV @R1,A ;Запись обратно в память

MOV R1,#33H

MOV A,@R1 ;Считывание с памяти разрядов

RRC A

MOV @R1,A ;Запись обратно в память

MOV R1,#32H

MOV A,@R1 ;Считывание с памяти разрядов

RRC A

MOV @R1,A ;Запись обратно в память

CLR C

JF1 SUBI

JMP ADDI

ENDING:

RETR ; Возврат из подрограммы