Санкт-Петербургский Политехнический Университет

Институт Информационных Технологий и Управления

Кафедра Компьютерных Систем и Программных Технологий

Отчет о лабораторной работе

по курсу «Микропроцессорные системы»

«Изучение вычислительных возможностей МКSAB 80C515»

Работу выполнили студенты группы 43501/4

Чеботарев Г.М.

Алексеев Д.М

Работу принял преподаватель

Санкт-Петербург

2015.

**Цель работы**

* знакомство с программно-аппаратным комплексом поддержки проектирования микроконтроллерных систем на базе МКSAB80C515;
* изучение системы команд МК семействаMCS51 на примере выполнения простейших программ.

**Программа работы**

Изучите состав и назначение основных устройств и блоков программно-аппаратного комплекса поддержки проектирования микроконтроллерных систем на базе МК семейства MCS51 (среды проектирования Shell51).

**Описание лабораторного стенда**

Лабораторный стенд (рис. 1) предназначен для изучения особенностей построения и функционирования встраиваемых микроконтроллерных систем.

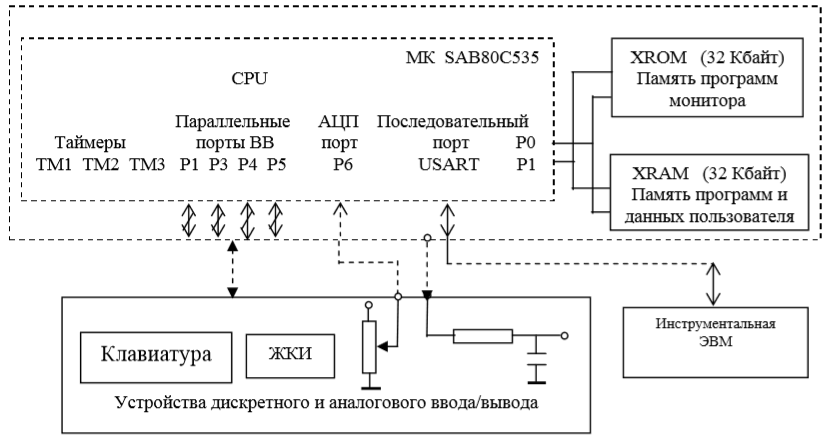


Рис.1

В состав стенда входят одноплатный контроллер, совокупность устройств дискретного и аналогового ввода/вывода, два генератора тестовых воздействий и коммутационное поле. На плате контроллера размещены МК SAB 80C535, микросхемы внешней памяти программ и данных (ОЗУ объемом 32 Кбайт), микросхемы внешней памяти программ «резидентного» монитора, схемы физического последовательного канала с оптоэлектронной развязкой и набор вспомогательных микросхем сопряжения МК с внешними устройствами.

Лабораторный стенд работает в составе системы поддержки проектирования МК-систем на базе ПК. С помощью ПК и установленной на нем оболочки проектирования Shell51 решаются следующие задачи разработки прикладного ПО МК-систем:

* подготовка исходного текста программ на языке ассемблера;
* трансляция «ассемблерных» программ в машинный код МК;
* обнаружение и коррекция синтаксических ошибок программы;
* запуск и отладка программы на симуляторе;
* загрузка программы в исполняемом формате в МК;
* контроль содержимого внутренней памяти программ и данных МК.

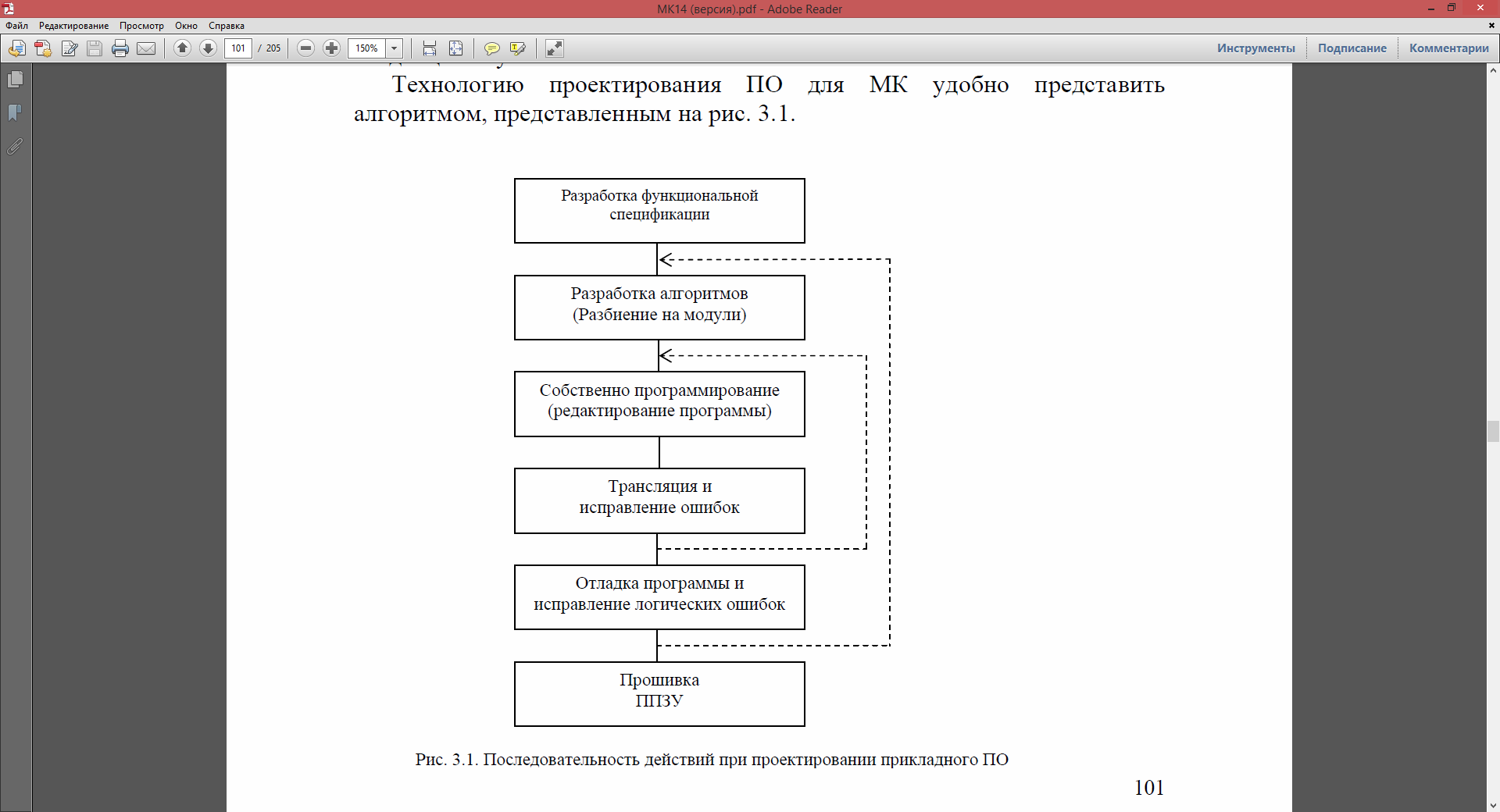
Интерфейс среды проектирования Shell51 предоставляет необходимые средства загрузки и отладки прикладных программ.

Совокупность устройств дискретного ввода-вывода в стенде представлена клавиатурой 4 х4 (устройство ввода) и блоком жидкокристаллических индикаторов ЖКИ (устройство вывода). Поскольку и клавиатура, и ЖКИ требуют программного управления, то по отношению к МК они являются объектами управления, что и отражено на схеме в их обозначении. Совокупность устройств аналогового вводавывода представлена регулируемым источником постоянного напряжения (потенциометром) и интегрирующим RC-звеном.

В качестве генераторов тестовых воздействий использованы источник гармонического сигнала «C» и источник импульсных сигналов «П». При формировании гармонических сигналов обеспечена возможность регулирования частоты и амплитуды. Регулируемыми параметрами генератора импульсных сигналов является частота и скважность;

Коммутационное поле позволяет задавать структуру аппаратных связей между элементами стенда и портами ввода-вывода МК. Внешние соединения реализуются с помощью набора проводников (специальных «шин»), прилагаемого к аппаратуре стенда.

При проектировании прикладного ПО имеет место следующая последовательность действий:



Мы ознакомились с этим циклом при выполнении задания 1.

**Задание 1**

На примере тестовой программы proba.asm познакомьтесь с полным циклом создания прикладного ПО. Программа proba.asm осуществляет обнуление ячеек заданной области внутренней памяти данных МК.

Алгоритма тестовой программы:

1. запись 0 в аккумулятор
2. запись адреса в регистр
3. начало цикла
   1. запись содержимого аккумулятора в адрес, указанный в регистре
   2. инкремент адреса
4. выполнять пока адрес не равен 0x60h
5. возврат из подпрограммы

Тестовая программа proba.asm

org 8400h ;размещение программы

;в памяти микроконтроллера

;начиная с адреса 0х8400

mov a,#0h ;запись в аккумулятор числа 0

mov r0,#50h ;запись в регистр

;первого адреса для обнуления

m1: mov @r0,a ;запись в ячейку по адресу,

;указанному в регистре

;числа из аккумулятора

inc r0 ;инкремент аккумулятора

;для получения следующего адреса

cjne r0,#60h,m1 ;условный переход

;с инкрементом счетчика

ret ;возврат из подпрограммы

Ответы на вопросы:

1. Какие способы адресации операндов используются в командах программы?

*Непосредственная и регистровая адресация для первой команды.  
Непосредственная и регистровая адресация для второй команды.  
Косвенная и регистровая адресация для третьей команды.*

*Регистровая для четвертой команды.*

*Регистровая и непосредственная для пятой команды.*

1. Какую функцию реализует директива org 8400h?  
   *Размещение программы в памяти микроконтроллера, начиная с адреса 0х8400.*
2. С помощью какой команды в программе реализуется цикл?

*С помощью команды cjne. Она увеличивает счетчик команд на 3, а затем сравнивает число в регистре R0 с числом 0х60. Если равенство не выполняется, то в счетчик команд загружается адрес начала цикла, то есть метки m1.*

В случае неисправности программы, появится влкдака «Листинги», указывающая на ошибку и номер строки, в которой произошла ошибка(рис.2).

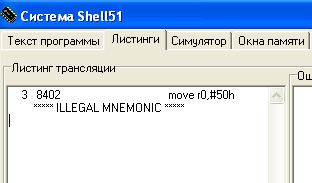


Рис.2

В случае возникновения ошибки необходимо исправить программу и еще раз скомпилировать. В данном примере использована некорректная мнемоника «move», вместо «mov». После исправления программа заработала.

**Задание 2**

Для заполнения ячеек заданной области внутренней памяти данных МК линейно возрастающими (линейно уменьшающимися) значениями модифицируйте программу proba.asm и выполните ее. Диапазон области адресуемой памяти и значения, записываемые в конкретные ячейки памяти, выберите самостоятельно.

Измененная тестовая программа proba.asm

org 8400h ;размещение программы в памяти микроконтроллера,

;начиная с адреса 0х8400

mov a,#0FFh ; записали в аккумулятор число FFh

mov r0,#40h ; записали в регистр число 40h

m1: mov @r0, #a ; записываем число, находящиеся в аккумуляторе по адресу из

; регистра r0

dec a ; уменьшаем число в аккумуляторе

inc r0 ; инкрементируем адрес, нах-ся в регистре r0

cjne r0,#60h,m1 ; выполняем до тех пор, пока r0!=60h

ret ; возврат из подпрограммы

Модификация программы заключается в добавлении команды инкремента содержимого аккумулятора на каждой итерации цикла. Таким образом, в ячейках памяти, начиная с адреса 0х40, будет находиться линейная убывающая последовательность чисел от FF до DF.

**Задание 3.1**

Разработайте и выполните программу вычисления арифметического выражения

F = (A\*B-B)/A\*(2\*A+B), где A!=0

Исходные данные – байтовые переменные. Операнды и результат вычисления разместите в ячейках внутренней памяти данных. Код программы для вычисления арифметического выражения:

org 8400h

mov 50h, #7

mov 51h, #3

mov A, 50h

mov B, 51h

mul AB ; A<=A\*B;

subb A, 51h ; A<=A-B;

mov B, 50h

div AB ; A<=A/B;

mov 55h, A ; save A in 0x55h

mov A, 50h;

rl a ; A<=2\*A;

add A, 51h ; A<=A+B;

mov B, 55h

mul AB ; A<=A\*B;

mov 57h, A ;save A in 0x57h

ret

Выражение было разделено и выполнено по действиям. Промежуточные данные сохранялись по мере необходимости в память, итоговый результат помещался по адресу 0x57

Результат работы программы:

|  |  |  |
| --- | --- | --- |
| Исходные данные | Результат теоретический | Результат практический |
| А = 1, В = 1 | 0 | 00000000 (00) |
| А = 6, В = 6 | 90 | 01011010 (5A) |
| А = 5, В = 9 | 133 | 10000101 (85) |

Результат работы программы удовлетворительный, все полученные данные совпадают с теоретическими расчетами.

**Задание 3.2**

Разработайте и выполните программу вычисления логического выражения

Y = ((not A) and C) xor (B and D)

с использованием команд булевого процессора. Исходные данные – битовые переменные. Операнды и результат вычисления разместите в ячейках внутренней памяти данных битового процессора (в диапазоне адресов 20h–2Fh).

*Исполнение:*

Примерный алгоритм работы программы:

1. Загрузка данных в 20,21,22,23 адрес
2. Логическое выражение разбито на несколько действий:
   1. not A
   2. x=nA and C, сохраняем полученный результат в 20-ый бит
   3. y=B and D, сохраняем в 28 бит
   4. Выполняется операция X xor Y.
3. Результат записывается в 38 бит.

Код программы представлен ниже:

org 8400h

mov 20h, #0 ; A

mov 21h, #0 ; B

mov 22h, #1 ; C

mov 23h, #0 ; D

;1

cpl 0h ; nA

;2

mov C, 0h

anl C, 10h ; x = nA and C

mov 20h,C ; 20 бит= x

;3

mov C, 8h ; B

anl C, 18h ; y = B and D

mov 28h,C ; 28 бит = y

;4 ; операция xor

mov C, 20h ; x

orl C,28h ; x or y

mov 30h,C ;30

cpl 20h ;not x

mov C, 20h

orl C,/28h ;not x or not y

anl C, 30h ;(!x or !y) and (x or y)

mov 38h, C ;0x27h

ret

Для проверки программы были использованы три набора данных. Результат работы удовлетворительный, все совпало. Данные работы приведены ниже.

|  |  |  |
| --- | --- | --- |
| Исходные данные | Результат | Рисунок |
| a = 1, b = 1, c = 1, d = 1 | y = 1 | Рис.3 |
| a = 1, b = 0, c = 1, d = 0 | y = 1 | Рис.4 |
| a = 1, b = 0, c = 1, d = 0 | y = 0 | Рис.5 |

**Задание 3.3**

Разработайте и выполните программу, которая осуществляет заполнение последовательных ячеек внешней памяти значениями, линейно изменяющимися в заданном диапазоне: 30h – 20h –30h.

*Исполнение:*

Была написана следующая программа:

org 8400h

mov a, #30h

mov r0, #0h

mov DPTR, #8460h

down: movx @DPTR,a

inc DPTR

dec a

inc r0

cjne r0, #10h, down

mov r0, #0h

up: movx @DPTR, a

inc DPTR

inc a

inc r0

cjne r0, #11h, up

ret

Для ясности приведем алгоритм работы программы:

1. Установка начальных данных: А=30, R0= 0, установка начального адреса;
2. Начало цикла:
   1. @DPTR = A;
   2. DPTR увеличиваем
   3. A=A-1;
   4. R0=R0+1;
3. Пока R0!=10 -> п.2
4. R0=0;
5. Начало цикла:
   1. @DPTR = A;
   2. DPTR инкрементируем
   3. A=A+1;
   4. R0=R0+1;
6. Пока R0!=11 -> п.5
7. ret.

Результат работы программы приведен на рисунках 6 - 8:

|  |  |  |
| --- | --- | --- |
| Рис.6 | Рис.7 | Рис.8 |

Как видно из представленных рисунков, программа работает. В адресах 84В0 – 84С0 – происходит уменьшение чисел от 30 до 20, а начиная с 84С1 и до 84D0 увеличение.

**Задание 3.4**

Разработайте программу, подсчитывающую число вхождений заданной константы в массив.

Код программы и алгоритм представлены ниже.

org 8400h

mov r0, #0h ;For circle

mov R5, #7h ;Const

mov R1, #0h ;Counter

mov 20h, #0 ;0 for clear C

; Ini mass

mov a, #5h

mov DPTR, #8461h;

movx @DPTR, a

mov DPTR, #8467h;

mov a, #7h

movx @DPTR, a

mov DPTR, #846Ah;

movx @DPTR, a

mov DPTR, #8460h ;Massive

mas: movx A, @DPTR

inc R0

inc DPTR

mov C, 0h ;clear C

subb A, R5

jz incr

cjne R0, #Fh, mas

sjmp ext

incr: inc R1

sjmp mas

ext: mov 50h, R1

ret

Алгоритм программы:

1. Инициализация регистров: R0=0- кол-во циклов

R5 = 7 – константа для поиска

R1 = 0 - регистр для счетчика

Инициализация пространства массива( в несколько случайных адресов записана константа) – коды между комментарием Ini mass и меткой mas

1. Цикл
   1. Запись числа из массива в аккумулятор
   2. R0 = R0+1;
   3. DPTR инкрементируем
   4. A – R5, если результат равен нулю, то увеличиваем счетчик
   5. Пока не пройдемся по всем массиву(длина F) -> 2.1
2. Конец программы

Результат работы:

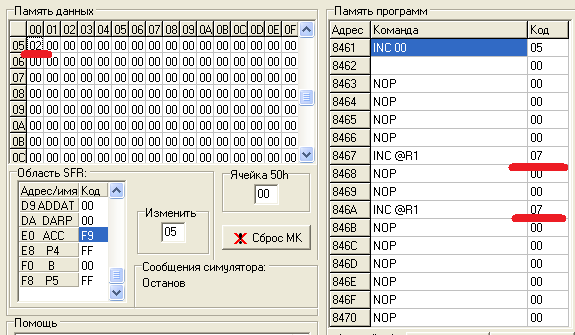


Рис.9 – результат работы 1.

Как видно, в память были записаны две семерки. Программа смогла подсчитать кол-во вхождений, и записало в ячейку 50h их кол-во = 2. Аналогичный результат на рисунке 10. Программе удалось правильно подсчитать число вхождений пятерок = 3.

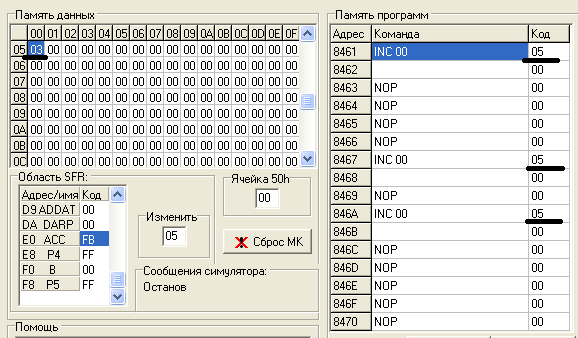


Рис.10 – результат работы 2.

**Выводы**

В данной лабораторной работе были проделаны первые шаги в понимании работы программно-аппаратного комплекса поддержки проектирования микроконтроллерных систем на базе микроконтроллера МКSAB80C515, а так же изучены системы команд МК семейства MCS51 на примере выполнения простейших программ, таких как:

1. обнуление заданной области внутренней памяти;
2. вычисление арифметических выражений и логических выражений с помощью
3. команд битового процессора;
4. заполнение области памяти линейно меняющими значениями;
5. функциональная обработка данных.

Данный микроконтроллер выглядит достаточно привлекательным:

1. не сложная структура
2. простата работы
3. схожесть с ранее изученными процессорами: 580-ым и 184-ым.

Однако есть определенная проблема, которая не может не насторожить: аппаратные комплексы сильно изношены. К сожалению, это может отразится на скорости выполнения работ.