Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий

Кафедра компьютерных систем и программных технологий

Отчет по лабораторной работе

по дисциплине «Микропроцессорные системы»

«Изучение вычислительных возможностей МК»

|  |  |  |
| --- | --- | --- |
| Работу выполнили студенты гр. 43501/3 | Козлов М.Н.  Воробей Е.А. | \_\_\_\_\_\_\_\_\_\_\_  *подпись*  \_\_\_\_\_\_\_\_\_\_\_  *подпись* |
| Работу принял преподаватель | \_\_\_\_\_\_\_\_\_\_\_\_  *ФИО* | \_\_\_\_\_\_\_\_\_\_\_  *подпись* |

Санкт-Петербург

2016

**Содержание**

[1. Цель исследования 2](#_Toc461349003)

[2. Программа работы 2](#_Toc461349004)

[3. Краткие теоретические сведения 2](#_Toc461349005)

[3.1 МК SAB80C515 2](#_Toc461349006)

[3.2 Программно-аппаратный комплекс SHELL51 3](#_Toc461349007)

[4. Выполнение заданий 4](#_Toc461349008)

[4.1 Цикл создания ПО на примере тестовой программы 4](#_Toc461349009)

[4.2 Вычисление арифметического выражения 5](#_Toc461349010)

[4.3 Вычисление логического выражения 7](#_Toc461349011)

[4.4 Заполнение ячеек внешней памяти 8](#_Toc461349012)

[4.5 Функциональная обработка данных 9](#_Toc461349013)

[Выводы 10](#_Toc461349014)

# Цель исследования

Цель данной работы является знакомство с программно-аппаратным комплексом поддержки проектирования микроконтроллерных систем на базе МК SAB80C515, а также изучение системы команд МК семейства MCS51 на примере выполнения простейших программ.

# Программа работы

Необходимо изучить систему команд на примере выполнения следующих задач:

1. На примере тестовой программы, осуществляющей обнуление ячеек заданной области внутренней памяти данных МК, ознакомиться с полным циклом создания прикладного программного обеспечения.
2. Разработать и выполнить программу вычисления арифметического выражения заданного вида.
3. Разработать и выполнить программу вычисления логического выражения заданного вида с использование команд битового процессора.
4. Разработать и выполнить программу, которая осуществляет заполнение последовательных ячеек внешней памяти значениями, линейно изменяющимися в заданных диапазонах.
5. Разработать и выполнить программу функциональной обработки данных.

# Краткие теоретические сведения

## МК SAB80C515

В состав 8-битного МК SAB 80C515 входят (рис. 3.1):

1. 8-разрядное АЛУ и схемы аппаратной реализации умножения и деления;
2. Внутреннее ПЗУ (IROM) программ и констант объемом 8 Кбайт;
3. Внутреннее ОЗУ (RRAM) данных объемом 256 байт;
4. Шесть программируемых портов ввода-вывода (P1 - P5);
5. Порт ввода аналоговых сигналов (P6);
6. Полный дуплексный последовательный порт SP с фиксированной и переменной скоростью обмена;
7. Три программируемых 16-битных таймера/счетчика;
8. 4-канальный блок быстрого ввода-вывода внешних событий, обладающий дополнительными возможностями формирования ШИМ-сигналов;
9. 8-канальный аналого-цифровой преобразователь со встроенным блоком программируемых эталонных напряжений;
10. Сторожевой таймер (WDT);
11. 4-уровневая система прерываний от 12 источников прерываний;
12. Внутренний стек глубиной 256 байт;

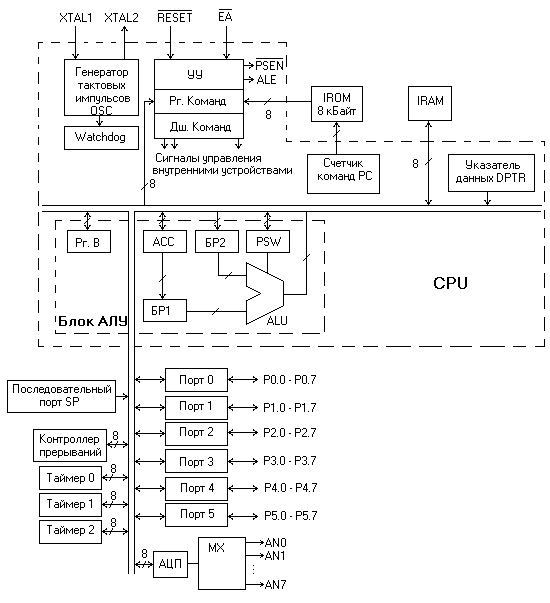


Рис. 3.1. Структурная схема МК SAB80C515

## Программно-аппаратный комплекс SHELL51

Проектирование микроконтроллерных систем проводится с использованием инструментальных программных и программно-аппаратных средств. Используемая программная среда SHELL51 ориентирована на применение в составе комплекса, включающего ЭВМ и микроконтроллер, подключенный к ЭВМ.

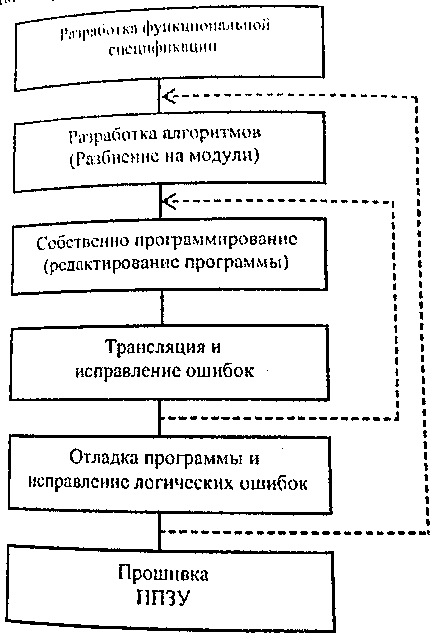
Среда Shell51 позволяет работать с исходным представлением программы в виде текста на языке ассемблера. С помощью кнопки “ЗАПУСК” активизируется выполнение процесса перевода исходного текста программы в представление, пригодное для загрузки в микроконтроллер. В случае появления ошибок и предупреждений со стороны транслятора детальная информация о результатах трансляции и компоновки будет расположена на специальной вкладке “ЛИСТИНГИ”.

Программный комплекс SHELL51 содержит симулятор, который позволяет загрузить оттранслированную программу в память и отследить ее выполнение по шагам, до установленного пользователем адреса или целиком.

Вкладка “Окна памяти” позволяет загрузить программу в микроконтроллер, запустить её и передать содержимое памяти микроконтроллера обратно на компьютер для анализа полученных результатов.

# Выполнение заданий

## Цикл создания ПО на примере тестовой программы

Цикл создания прикладного программного обеспечения для МК включает в себя следующие этапы:

1. Разработка функциональной спецификации
2. Разработка алгоритмов
3. Программирование (написание текста программы)
4. Трансляция и исправление ошибок
5. Отладка программы и исправление логических ошибок
6. Прошивка ППЗУ

Рассмотрим каждый этап подробнее на примере тестовой программы prob.asm.

**Разработка функциональной спецификации.** В нашем случае спецификация уже разработана. Программа prob.asm должна осуществлять обнуление ячеек заданной области внутренней памяти данных МК.

**Разработка алгоритмов.** Поскольку программа предоставляется в готовом варианте, мы также пропускаем этап разработки алгоритма работы программы. Алгоритм заключается в последовательном обращении к ячейкам внутренней памяти в определенном диапазоне и их обнулении.

**Программирование.** Необходимо текст программы, приведенной в учебном пособии, ввести в окно ввода программы. Текст программы prob.asm приведен ниже (листинг 4.1).

Листинг 4.1. Код программы prob.asm

|  |
| --- |
| org 8400h  mov a, #0h  mov r0, #50h  m1: mov @r0, a ; Косвенная адресация через регистр r0  inc r0  cjne r0, #60h, m1 ; Цикл  ret |

**Трансляция и исправление ошибок.** Чтобы выявить синтаксические ошибки, необходимо осуществить трансляцию написанной программы. В данном случае трансляция ошибок не выявила. Однако в качестве примера заменим в 4 строке “a” на “ab” и повторим операцию, то увидим, что она завершится сообщение об ошибке во вкладке «Листинги».

**Отладка программы и исправление логических ошибок.** Отладка программы осуществляется при помощи симулятора. Для этого необходимо загрузить программу во вкладке симулятор. Далее мы можем выполнить программу сразу или проанализировать её работу пошагово. При этом можно в реальном времени наблюдать изменения данных в ячейках памяти.

**Прошивка ППЗУ.** Убедившись, что программа работает верно и логические ошибки отсутствуют, можно загрузить её в микроконтроллер. Здесь мы не можем в реальном времени оценить работу программы, так как в компьютер загружается конечное состояние памяти МК – результат выполнения программы.

## Вычисление арифметического выражения

Необходимо разработать и выполнить программу вычисления арифметического выражения заданного вида (вариант 4):

*F = (A \* B – B) / A \* (2 \* A + B), где А ≠ 0*

Исходные данные – байтовые переменные. Операнды и результат вычисления необходимо разместить в ячейках внутренней памяти.

Алгоритм программы:

1. Чтение операнда А из внутренней памяти;
2. Проверка, что операнд А не равен 0, иначе завершение программы;
3. Чтение операнда В из внутренней памяти;
4. Последовательное выполнение арифметических операций с сохранением промежуточных результатов во внутренней памяти;
5. Помещение результата вычисления в ячейку памяти r5;
6. Завершение программы.

Текст программы приведен ниже (листинг 4.2). В комментариях указаны арифметические операции, которые выполняются в данном блоке кода.

Листинг 4.2. Код программы вычисления арифметического выражения

|  |
| --- |
| org 8400h  mov r0, #2h  mov a, r0  jz error ;проверка А на неравенство нулю  mov r1, #4h  mov b, r1 ;загрузка числа В в регистр b  mul ab ;A\*B  ;AB-B  clr c  subb a, r1 ;вычитание  mov r2, a  mov r3, b  ;(AB-B)/A  mov b, r0 ;загрузка А в регистр b  div ab ;деление  mov r2, a  ;2\*A  mov a, #2h  mov b, r0  mul ab  ;2A+B  clr c  add a, r1  ;F=(АВ-В)/А\*(2А+В)  mov b, r2 ;загрузка результата (AB-B)/A  mul ab  mov r5, a  error: mov r5, #ffh ;загрузка в r5 числа ff  ret |

Выполнив программу для разных входных значений, мы убедились, что результаты соответствуют теоретическим (табл. 4.1).

Таблица 4.1. Результаты работы программы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № теста | А | В | Ответ теор. | Ответ практ. |
| 1 | 02h | 04h | 10h | 10h |
| 2 | 03h | 06h | 30h | 30h |
| 3 | 0 | 06h | FFh | FFh |

## Вычисление логического выражения

Необходимо разработать и выполнить программу вычисления логического выражения с использованием команд битового процессора. Исходные данные – битовые переменные. Логическое выражение (вариант 4):

*Y = ((NOT a) AND c) XOR (b AND d)*

Алгоритм программы:

1. Задание исходных значений для бит, соответствующих операндам;
2. Последовательное вычисление результатов промежуточных операций;
3. Помещение результата логического выражения в нулевой бит.

Текст программы приведен ниже (листинг 4.3). В комментариях указаны логические операции, которые выполняются в данном блоке кода.

Листинг 4.3. Код программы вычисления логического выражения

|  |
| --- |
| org 8400h  ;Установление бит внутренней памяти  clr 0 ;A  setb 1 ;B  clr 2 ;C  setb 3 ;D  ;NOT A and C  mov c, 2  anl c, /0  mov 4, c ;результат в бит 4  ;B and D  mov c, 1  anl c, 3  mov 5, c ;результат в бит 5  ;XOR  clr 6  clr 7  mov c, 4  anl c, /5 ;А\*not В  mov 6, c ;результат в бит 6    mov c, 5  anl c, /4 ;not А\*В  mov 7, c ;результат в бит 7  mov c, 6  orl c, 7 ; А\*not В + not А\*В  mov 0, c ; результат в бит 0  ret |

Выполнив программу для разных входных значений, мы убедились, что результаты соответствуют теоретическим (табл. 4.2).

Таблица 4.2. Результаты работы программы

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № теста | А | В | С | D | Ответ теор. | Ответ практ. |
| 1 | 0 | 1 | 0 | 1 | 1 | 1 |
| 2 | 1 | 0 | 1 | 0 | 0 | 0 |

## Заполнение ячеек внешней памяти

Необходимо разработать и выполнить программу, которая осуществляет заполнение последовательных ячеек внешней памяти значениями, линейно изменяющимися в заданных диапазонах. Заданный диапазон (вариант 4):

*30h – 20h – 30h*

Алгоритм программы:

1. Инициализация счетчика значением 10h
2. Инициализация переменной значение 30h
3. Пока счетчик не равен нулю:
   1. Заполнить текущую ячейку внешней памяти значением переменной
   2. Уменьшить значение счетчика на единицу
   3. Уменьшить значение переменной на единицу
   4. Перейти к следующей ячейке
4. Инициализация счетчика значением 11h
5. Пока счетчик не равен нулю:
   1. Заполнить текущую ячейку внешней памяти значением переменной
   2. Уменьшить значение счетчика на единицу
   3. Увеличить значение переменной на единицу
   4. Перейти к следующей ячейке

Текст программы приведен ниже (листинг 4.4).

Листинг 4.4. Код программы заполнения ячеек внешней памяти

|  |
| --- |
| оrg 8400h  mov r0, #30h  mov dptr, #8500h ;задание адреса внешней памяти  mov r1, #10h ;счетчик  l1: mov a, r0  movx @dptr, a  dec r0 ;изменение числа  inc dptr ;след. адрес внешней памяти  dec r1 ;декремент счетчика  mov a, r1  jnz l1 ;проверка не равен ли счетчик 0  mov r1, #11h ;установка счетчика  l2: mov a, r0  movx @dptr, a  inc r0  inc dptr  dec r1  mov a, r1  jnz l2  ret |

Результат выполнения программы соответствует ожиданиям: ячейки внешней памяти в диапазоне *8500h – 8520h* заполняются значениями в заданном диапазоне *30h – 20h – 30h*.

## Функциональная обработка данных

Разработать и выполнить программу подсчета количества вхождений заданной константы в массив.

Алгоритм программы:

1. Инициализация счетчика количества вхождений значением 0
2. Задание размера массива (5 элементов)
3. Запись элементов массива во внешнюю память
4. Последовательное сравнение константы с каждым элементом массива, в случае равенства – увеличение счетчика на единицу

Текст программы приведен ниже (листинг 4.5).

Листинг 4.4. Код программы функциональной обработки данных

|  |
| --- |
| org 8400h  mov dptr, #8500h  ;Запись массива во внеш. память  mov a, #10h  movx @dptr, a  inc dptr  mov a, #1Ah  movx @dptr, a  inc dptr  mov a, #0Ah  movx @dptr, a  inc dptr  mov a, #0Ah  movx @dptr, a  inc dptr  mov a, #1Fh  movx @dptr, a  ;Константа  mov dptr, #8500h  mov r2, #0h ;счетчик нахождения константы  mov r1, #5h ;установка счетчика  l1: movx a, @dptr  clr c  subb a, #0Ah ;сравнение константы с элементом массива  jnz l2  inc r2  l2: dec r1  inc dptr ;след. адрес  mov a, r1 ;конец массива?  jnz l1  ret |

Результаты работы программы приведены ниже (табл. 4.3).

Таблица 4.3. Результаты работы программы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № теста | Массив | Константа | Ответ теор. | Ответ практ. |
| 1 | 10, 1A, 0A, 0A, 1F | 0A | 2 | 2 |
| 2 | 26, F2, 13, F2, F2 | F2 | 3 | 3 |

# Выводы

В ходе данной работы мы ознакомились с программно-аппаратным комплексом поддержки проектирования микроконтроллерных систем на базе МК SAB80C515. На примере тестовой программы (пункт 4.1) был рассмотрен цикл создания прикладного ПО, включающий следующие этапы:

1. Разработка функциональной спецификации
2. Разработка алгоритмов
3. Программирование (написание текста программы)
4. Трансляция и исправление ошибок
5. Отладка программы и исправление логических ошибок
6. Прошивка ППЗУ

Выполняя тестовую программу, мы убедились, графический пользовательский интерфейс программного комплекса SHELL51 значительно упрощает и ускоряет разработку прикладного ПО на большинстве этапов цикла. К факторам, упрощения разработки можно отнести следующие:

1. Ввод команд программы происходит в окне текстового редактора, таким образом ввод программы становится быстрее, а её чтение нагляднее;
2. Существует возможность сохранить текст программы в файл, что позволяет быстро восстановить программу и продолжить её написание, не вводя с самого начала;
3. Наглядное отображение ошибок хотя и не является достаточно «умным», чтобы всегда сообщить, в чем именно вызвана ошибка, но автоматическое определение места программы значительно ускоряет процесс отладки;
4. Благодаря встроенному симулятору исполнения программы, в ходе покомандного выполнения кода, можно гораздо более наглядно отслеживать возникновение логических ошибок.

В следующих двух программах (пункты 4.2 и 4.3) было продемонстрировано выполнение команд арифметических и логических операций, а также использование команд битового процессора. Стоит отметить, что использование команд битового процессора при вычислении логических выражений значительно экономит память (была задействована всего одна ячейка памяти).

В отличие от предыдущих программ, в последних двух (пункты 4.4 и 4.5) была продемонстрирована работа с внешней памятью. В первой программе мы осуществили запись значений из определенного диапазона во внешнюю память, а во второй программе прочли значения массива и выполнили необходимые вычисления. Таким образом было продемонстрировано, что, оперируя с внешней памятью, можно выполнять любые необходимые команды и использовать ее, если возникает такая необходимость. Также это открывает возможности для «саморедактирования» программы в момент выполнения, что может быть и нетривиальным решением, и ошибкой, допущенной по невнимательности (которой легко можно избежать благодаря этапу отладки в симуляторе).