Проектирование ЭВМ

Целью проекта является знакомство, освоение и участие в проектировании цифровых устройств уровня ЭВМ на примере синтеза схем для микроэвм mcs51.

Проектирование включает

* разработку теста для контроля функциональности заданных команд,
* разработку функциональных микропрограмм в Borland С для заданной схемы микропрограммного управления,
* тестирование функциональных микропрограмм в IDE Микро51,

--проектирование блочно-структурной схемы для исполнения одной из команд теста и описание этой схемы в AHDL в MaxPIus2

IDE Микро51 разработана для целей образования и представляет собой открытую программу в Borland C++ с визуальным отображением результатов на экране Программной модели. Результат двоичного кодирования микропрограммы формируется в виде файлов загрузки управляющей памяти в проекте MaxPlus2.

Система Микроб 1 настраивается в Borland C++ (или в другой функционально совместимой системе программирования) для кодирования и исполнения микропрограмм заданных команд.

Предполагается знание программирования в C++ и схемотехники на уровне графического и текстового редактора в MaxPlus2 (Quartus2)

Содержание

1. Задание - обзор системы команд,
2. Структура ЭВМ
3. Функциональное тестирование в Ассемблере

тестовая программа функциональная модель в Кейл

1. Микропрограммирование

* Исполнение команд с микропрограммным управлением -Функциональная микропрограмма - исполнение в Си

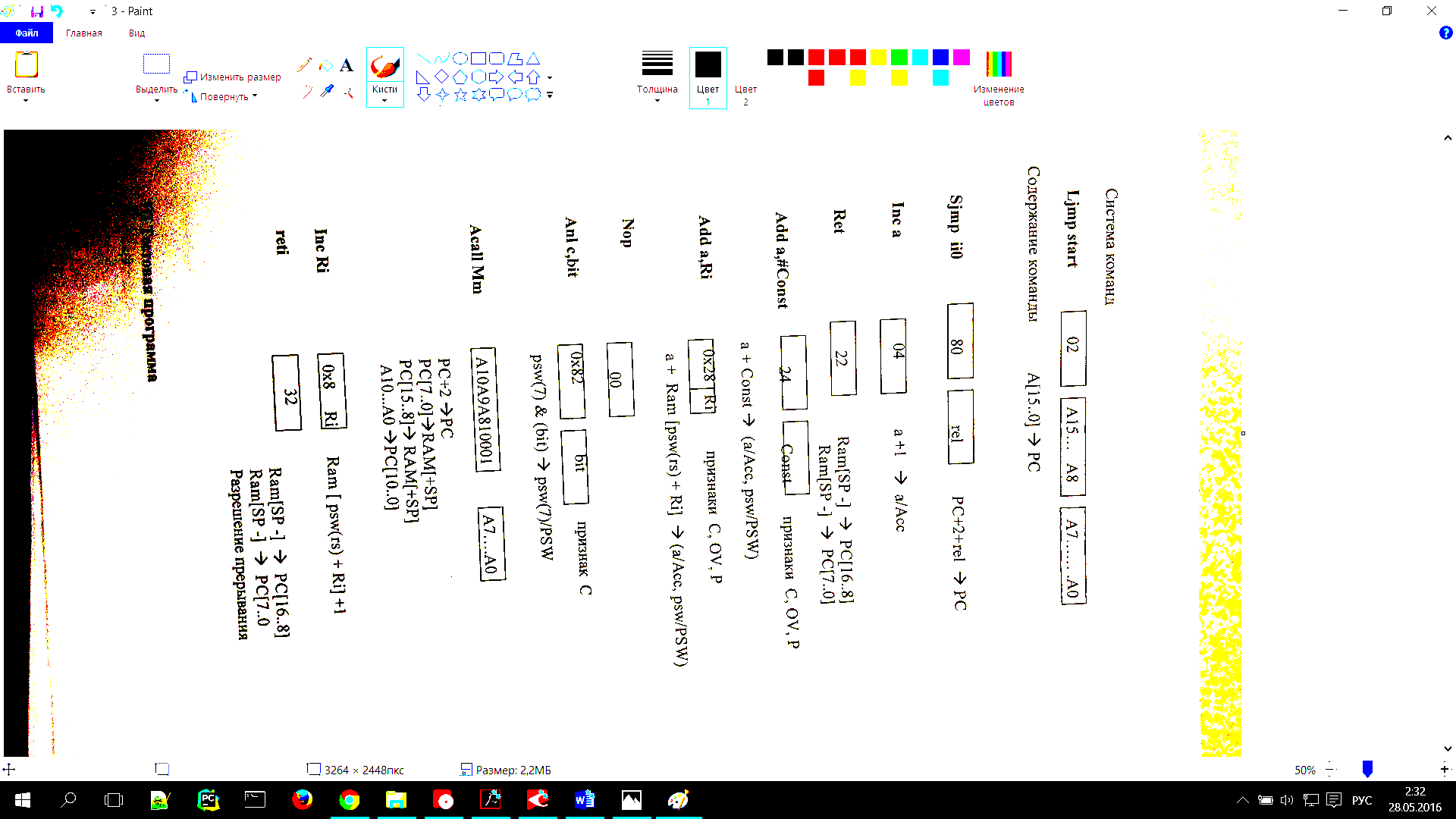
1. Исполнение команды в схемах

* в структурной
* в функциональных (кодирование)

II. Задание включает список команд из системы команд mcs51 Требуется :

1. оформить задание, привести спецификацию(описание) команд задания из ЬЗюшийекрадирование и принцип исполнения
2. разработать тест для заданной системы команд, отладить и компилировать в Keil и выполнить моделирование
3. разработать функциональные микропрограммы для команд в

системе Микро51 на языке Borland C++, используя общую структурную схему ЭВМ



**III Тестовая программа**

Цель тестирования:

- контроль правильности исполнения команд и работы программного управления Сложность и непреодолимые трудности обоснования достоверно полного тестирования ограничивают его функциональным тестированием с визуализацией результатов контрольного исполнения тестовой программы.

Таким образом, с позиций разработчика компьютера проверяем на конечном числе примеров с конкретными исходными данными по исходной спецификации правильное исполнение доступа к данным в различных типах памяти и выполнения операций.

Каждое задание, по умолчанию, предполагает использование команд ljmp start, reti вспомогательных команд, обеспечивающих функциональность основных команд (возврат из подпрограмм, прерываний и установку данных)

Команды

1. **ljmp start** - выборка н декодирование команды из памяти Code, изменение состояния регистра PC
2. **Add a, #Const** - контролируем: доступ к регистру а/Асс, чтение Const из памяти Code, правильное исполнение суммировании и формирования признаков результата в PSW.
3. Add a,Ri - доступ к регистрам общего назначения Ri
4. Аnl с, bit - доступ к битам битового сегмента, правильное исполнение логической операции с битами
5. Асаll Mm - запись в Стек
6. Inc Ri - операция счета
7. Ret - чтение Стека
8. sjmp met -

; АСС=0Х80 RO=OX85 PSW\*0X00 SP=0X07 **Cseg at** 0; сегмент кода - команда старта Ljmp start;

Into: ; вход в прерывание INTO

reti

**Cseg at 0x10**; сегмент кода - подпрограмма Mm:

Inc a ;( a/Acc=0x86, С=0)

Ret ;(PC= )

**Cseg at 0x20** ; сегмент кода – основная программа

Start:

Add а,#0x80 ;(a/Acc=0x80, C=1,P=1,Ov=1)

AnI c, Acc.7 ; (C=0 )

Add a. r0 ; ( a/Acc=0x85 C=1 ,P=0,Ov=0)

Inc r0 ; ( R0=0x86)

Nop ; задержка для формирования прерывания

Acall Mm ; ( PC= Ctek\* ) end

Листинг

-------- ;Cseg at 0

0000 020020 ; Ljmp start;

0003 Intro:

0003 32 reti ;SP=07

------ Cseg at 0x10

0010 Mm:

0010 04 Inc a ; (a/Acc=0x86, C=0)

0011 22 Ret ;(PC=0x29, SP=07)

------ Cseg at 0x20

0020 Start:

0020 2480 Add a, #0x80 ;(a/Acc=0x80, C=1, P=1, Ov = 1)

0022 82E7 Anl c, Acc.7 ;(C=0)

0024 28 Add a, r0 ;(a/Acc=0x85 C=1, P=0, Ov = 0)

0025 08 Inc r0 ;(R0 = 0x86)

0026 00 Nop ; интервал для формирования интервала

0027 1110 Acall Mm ;(PC=0x10, SP=07, Стек = 29 00)

end

Листинг оформляется для размещения в программе Микроб 1

Тест Память программ состояние памяти

//РС=00 SP=07 асс= (г0)= //0: Ijmp start 0x02 00 0x23 Рс=0х23 SP=07

//03: reti 0x32 РС=0х26 sp=07

//13: reti 0x32 Pc=0x28 sp=07

//mсal:

//22: ret 0x22 Pc=0x2a sp=07

//start: acc=0x82 r0=0x21;

//23: add a,#80 0x24 0x80 acc=0x02, PSW=0x81

//25: nop 0x00

//(int0,stack=0026,PC=03) //int1, stack=0026,PC=13)

//26: add a,r0 0x28 acc=0x23, PSW=01

//27: anl c,ACC.7 0x82 0xe7 PSW=01

//29: acall meal 0x11 0x22 PC=0x22, SP=0x09, Ram[sp] = 00 2b

//2b: nop 0 //конец программы

^.Микропрограммирование.

4.1. Блок — схема

Разделение функциональной схемы на блоки формируется в процессе разработки и уточнения функциональной схемы.

Содержание блока определяется исходя из некоторого принятого функционального разделения ЭВМ

1. блок регистрового арифметико-логического устройства (RALU) включает арифметико-логическое устройство (ALU) и регистры оперативного хранения данных (A,B,PA,PB.Wrk) . В библиотеках ПЛИС доступны как функциональные элементы для размещения в RALU умножитель (mul) и делитель (div)
2. блок внешней памяти 16тпст обычно большого объема со схемами управления доступом. В данном случае - память Code|65 кб| и Xdata|65 кб| Адресные регистры PC, Dptr.
3. Блок микропрограммного управления Control включает схему синхронизации, выборки микрокоманд (CU) и память микропрограмм (RomK).
4. Блок управления системной шиной Bus8 включает оперативную память данных Ram[256 б] схемы формирования адреса и управления Стеком, мультиплексор системной шины BUSB[7..0]
5. Блок управления битами ubit и формирования PSW
6. Блок управления прерываниями Interrupt
7. Блок цифровых каналов ввода-вывода Ports включает схемы портов PO­P3