НИУ ИТМО

Курсовая работа

Проектирование ЭВМ

Организация ЭВМ и систем

Айдар Шайхиев

08.05.2014

Группа 3121

2013

Список команд которые нужно реализовать в эмуляторе, и написать к ним тестовую программу

- subb a, {@rj, #d}
- rr a
- xch a, {ri, ad}
- acall ad11

Этап 1

SUBB A, #data

C	AC	F0	RS1	RS0	OV		P

Bytes 2

Cycles 1

Encoding 1001 0100 #data //код команды

Operation
```
SUBB
A = A - C - immediate
```

Example
```
SUBB A, #01h
```

SUBB A, @Ri

C	AC	F0	RS1	RS0	OV		P

Bytes 1

Cycles 1

Encoding 1001 011i //код команды

Operation
```
SUBB
A = A - C - (Ri)
```

Example
```
SUBB A, @R1
```

Команда «вычитание с заемом» вычитает указанную переменную вместе с флагом переноса из содержимого аккумулятора, засылая результат в аккумулятор. Эта команда устанавливает флаг переноса (заема), если при

вычитании для бита 7 необходим заем, в противном случае флаг переноса сбрасывается. Если флаг переноса установлен перед выполнением этой команды, то это указывает на то, что заем необходим при вычитании с увеличенной точностью на предыдущем шаге, поэтому флаг переноса вычитается из содержимого аккумулятора вместе с операндом источника. (AC) устанавливается, если заем необходим для бита 3 и сбрасывается в противном случае. Флаг переполнения (OV) устанавливается, если заем необходим для бита 6, но его нет для бита 7, или есть для бита 7, но нет для бита 6.

При вычитании целых чисел со знаком (OV) указывает на отрицательное число, которое получается при вычитании отрицательной величины из положительной, или положительное число, которое получается при вычитании положительного числа из отрицательного.

RR A

C	AC	F0	RS1	RS0	OV		P

Bytes	1
Cycles	1
Encoding	0000 0011
Operation	RR $A_n = A_{n+1}$ where n = 0 to 6 $A_7 = A_0$
Example	RR A

Команда «сдвиг содержимого аккумулятора вправо» сдвигает вправо на один бит все восемь бит аккумулятора. Содержимое бита 0 помещается на место бита 7. На флаги эта команда не влияет.

XCH A, ad

C	AC	F0	RS1	RS0	OV		P

Bytes	2
Cycles	1
Encoding	1100 0101 direct adress
Operation	XCH A swap (direct)

Example	`XCH A, 45h`

XCH A, Rn

C	AC	F0	RS1	RS0	OV		P

Bytes	1
Cycles	1
Encoding	1100 1nnn
Operation	`XCH` `A swap Rn`
Example	`XCH A, R6`

Команда «обмен содержимого аккумулятора с переменной-байтом» осуществляет обмен содержимого аккумулятора с содержимым источника, указанным в команде.

ACALL addr11

C	AC	F0	RS1	RS0	OV		P

Bytes	2
Cycles	2
Encoding	A10A9A81 0001 A7A6A5A4 A3A2A1A0

Operation	`PC = PC + 2`
	`SP = SP + 1`
	`(SP) = PC7-0`
	`SP = SP + 1`
	`(SP) = PC15-8`
	`PC10-0 = A10-0`

Example	`ACALL LABEL`

Команда «абсолютный вызов подпрограммы» вызывает безусловно подпрограмму, размещенную по указанному адресу. При этом счетчик команд увеличивается на 2 для получения адреса следующей команды,

после чего полученное 16-битовое значение PC помещается в стек (сначала следует младший байт), и содержимое указателя стека также увеличивается на два. Адрес перехода получается с помощью конкатенации старших бит увеличенного содержимого счетчика команд, битов старшего байта команды и младшего байта команды.

Этап 2 4

```
#include <vcl.h>
#pragma hdrstop
#include <stdio.h>
#include "modelir.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{}
//---------------------------------------------------------------------
typedef unsigned char uchar;
typedef unsigned int uint;
uchar ROMM[1000];
//регистровая память
uchar IR,   //регистр команд
      Wrk1, //рабочий регистр
      Wrk,  //рабочий регистр
      ACC,  //аккумулятор с прямым доступом
      SP,   //указатель стека
      RA,   //рабочий в RALU
      RB,
      PSW,  //слово состояния
      B,    //рабочий
      Ram[256],
      Xdata[256],
      P3,   //регистр порта P3
      TCON, //tf1 tr1 tf0 tr0 ie1 it1 ie0 it0
      IE,  //EA . .  es et1 ex1 et0 ex0
      IE1, //буфер регистра IE
      int0, //бит запроса прерывания 0
      int1, //бит запроса прерывания 1
      icod; //регистр вектора прерывания;

uint PC;   //PCH=PC>>8,PCL=PC
uint DPTR; //DPH=DPTR>>8,DPL=DPTR

//управление файлами
DWORD bri,ii;
DWORD bro,bufd;
OVERLAPPED ovr,ovrd;
HANDLE hCom,hdisk,hdisk1,htempl=0;
DCB dcb;
char coma;

uchar *stro="      ";
uint RAMM; //регистр адресов микрокоманд

//адреса теневых регистров в SFR
const uchar  Sp = 0x81;
const uchar  Acc = 0xe0;
const uchar  Dph = 0x83;
const uchar  Dpl = 0x82;
const uchar  b = 0xf0;
const uchar  Psw = 0xd0;
```

```
const uchar  p3 = 0x90;

DWORD codDCM; //32-бит код микрокоманды
DWORD DCM[16]; //ПЗУ микрокода

uchar Code[100];//тест в кодах команд + таблица векторов и начало
uchar ADC[0x100]; //память декодирования кода операции в адрес микрокоманды

              // Ifo,    BasA,      BasB=,    =BasB,     BasC=,    =BasC,    AdSFR,   inc
//имя в проекте ifo[3],selbasa[3],selbasb[4],wrbasb[4],selbasc[4],wrbasc[4],AdSFR[3],inc[3] 28 бит
//структура кода микрокоманды
struct cod_mc{
       uchar ifo;     //3
       uchar basa;  // 3
       uchar basb;  // 4
       uchar wrb;   // 4
       uchar basc;  // 4
       uchar acx;   //2
       uchar alu;   //3
       uchar wrc;   // 3
       uchar adsfr;    // 3
       uchar unicod;      // 3
       uchar cin;     //2
       uchar bicc;    //2
       uchar consta;   //3
       uchar pswf;     //2
       uchar ramm;     //2
       } mema0;             //43 bit

void  __fastcall TForm1::clear_mema(void)
{
       mema0.ifo=0;     //3
        mema0.basa=0;  // 3
        mema0.basb=0;  // 4
        mema0.wrb=0;   // 4
        mema0.basc=0;  // 4
        mema0.acx=0;   //2
        mema0.alu=0;   //3
        mema0.wrc=0;   // 3
        mema0.adsfr=0;    // 3
        mema0.unicod=0;      // 3
        mema0.cin=0;     //2
        mema0.bicc=0;    //2
        mema0.consta=0;   //3
        mema0.pswf=0;     //2
        mema0.ramm=0;     //2
}

//таблица кодирования микроопераций по столбцам - полям
//код       1  2   3   4  5
char *Ifo="B0,W7,~W7,~Cc,Cc ";   //конец строки - пробел, первая буква - заглавная
char *Basb="L,Ram,Xram,Code,Acc,Dd,Pcl,Dpl,H,Bitpsw,Ralu,B ";
char *Basc="P1,Acc,B,Wrk,Wrk1,Psw,Pch,Dph,L,Acall,Ralu,Sp,H,Bita,Alu1 ";
char *Basa="Wrk,Ar,Adbit,Asfr,Sp,Pc ";
char *Acx="Pc,Wrkwrk1,Intr,Dptr ";
char *Wrc="Ram,Acc,Xram,Pch,Dph,Rb,Ir ";
char *Wrb="Sp,B,Wrk,Wrk1,Dph,Ra,Psw,Ir,Dpl,Pcl ";
char *Alu="Res,Suba,Subb,Add,Or,Xor,And,Set ";
char *Const="Zero,Mone,Moct,Abi,Sp0 ";
char *bicc="C8,Sign,Acc0 ";
char *Cin="H,L,Cc,Psw7 ";
char *Adsfr="Sp,Acc,Dph,Dpl,P1,Psw ";
char *Unicod="Pc_,_Sp,Sp_,Dptr_,Wrpsw ";
char *Pswf="add,and,andc ";
char *Ramm="Ramm_,Ramm0 ";

uchar __fastcall TForm1::CodMop(char *st,char *sss) //(поле, микрооперация)
```

```
{
     uchar x;
     uchar * memo="       ";
     uchar * memo1;
     uchar   cod=0;
         memo1=memo;
      for(char ii=0; ; ii++)
     {       x=*st++;
         if(((x==',')||(x==' '))&&(*memo1==*sss))
                                 {cod++; break;}
         if (x==',') {cod++; memo=memo1;
                          for(char i=0;i<7;i++)
                                   *memo++=' ';
                       memo=memo1;
                   }
         else *memo++=x;
       }
   return cod;
}

void __fastcall TForm1::TakeCode(char * stpole, char * stmo)
{     //формирование кода микрооперации stmo в поле stpole
      // - имя поля микрооперации --> в AnsiString S
      // - имена полей из таблицы полей микрокоманды --> в Q
      // - вход в столбец таблицы при совпадении имен
      // - выбор кода из столбца=порядковому номеру и
           //формирование структуры кодирования

    uchar x;
     AnsiString S,Q;

        S.printf("%s",stpole);
        Q.printf("Ifo    ");  //символов 7
       if(S==Q) mema0.ifo=CodMop(Ifo,stmo);
       Q.printf("Basa   ");
       if(S==Q) mema0.basa=CodMop(Basa,stmo);
       Q.printf("Basb   ");
       if(S==Q)  mema0.basb=CodMop(Basb,stmo);
       Q.printf("Wrb    ");
       if(S==Q)    mema0.wrb=CodMop(Wrb,stmo);
       Q.printf("Basc   ");
       if(S==Q)     mema0.basc=CodMop(Basc,stmo);
       Q.printf("Wrc    ");
       if(S==Q)  mema0.wrc=CodMop(Wrc,stmo);
       Q.printf("Adsfr  ");
       if(S==Q)  mema0.adsfr=CodMop(Adsfr,stmo);
       Q.printf("Unicod ");
       if(S==Q)   mema0.unicod=CodMop(Unicod,stmo);
       Q.printf("Acx    ");
       if(S==Q)  mema0.acx=CodMop(Acx,stmo);
}

void __fastcall TForm1::Codmema(void)
    //формирование кода микрокоманды в ROM DCM[i]
    //если код существует, то его адрес в памяти микропрораммы ROMM[]
     //если новый, то сохранение в DCM[i]
{
        uchar i;

codDCM=((((((((((((mema0.ifo<<3)+mema0.basa)<<4)+mema0.basb)<<4)+mema0.wrb)<<4)+mema0.basc)<<4)+mem
a0.acx)<<2)+mema0.alu)<<3)+mema0.wrc;

codDCM=((((((((((codDCM<<3)+mema0.adsfr)<<3)+mema0.unicod)<<2)+mema0.cin)<<2)+mema0.bicc)<<2)+mema0
.consta)<<3)+mema0.pswf;
     for(i=1; i<16; i++) //i-адрес, mcod-декод в таблице кодов
     {if(DCM[i]==codDCM) break;
      if(DCM[i]==0)  {DCM[i]=codDCM;  //i mk
```

```
                        break;
                        }
      }
    if (ROMM[RAMM]);
    else  ROMM[RAMM]=i; //минимальный код микрокоманды в микропрограмме
      clear_mema();
}

void __fastcall TForm1::MicroCodMem(char *ss) //функция кодирования микрокоманды
{             //*ss -символьная строка микрокоманды
    uchar *namepole="       "; //имя поля микрооперации
    uchar *namemo="       ";  //имя микрооперации
    uchar x,xx=',';             //признаки лексического разбора
    uchar *namepole1="       ";
    uchar *namemo1="       ";  //указатели, сохраняющие имена
    namepole1=namepole;     //сохранение указателей
    namemo1=namemo;
if((CheckBox1->State==cbChecked)&&(ROMM[RAMM]==0))//разрешение кодирования
    {for(char ii=0; ; ii++)
     {       x=*ss++;
         if(xx=='=')  //правая часть -имя микрооперации
            if((x==',')||(x==' ')) //конец микрооперации
                  {TakeCode(namepole1,namemo1);
                          namepole=namepole1; namemo= namemo1;
                          for(char i=0;i<7;i++)
                                   *namemo++=*namepole++=' ';
                     namepole=namepole1; namemo= namemo1;
                         xx=x;
                    if(x==' ') goto finn; //конец микрокоманды
                    }
            else  *namemo++=x;  //выборка имени микрооперации
          else   //xx=',' выборка имени поля микрооперации
             if(x=='=') xx='='; //переключение на выборку имени микрооперации
             else *namepole++=x;
       }
       finn: Codmema(); //завершение разбора структурной микрокоманды и переход
    }                   //к кодированию
}

char ss[10];
void __fastcall TForm1::StateMCU(void)
{ //состояние
 //Edit6->Text=ACC;
            Edit14->Text=itoa(SP,stro,16);
            Edit20->Text=itoa(Ram[SP],stro,16);
            Acu->Text=itoa(ACC,stro,16);
            Work->Text=itoa(Wrk,stro,16);
            Work1->Text=itoa(Wrk1,stro,16);
            ProgCnt->Text=itoa(PC,stro,16);
            Edit1->Text=itoa(PSW,stro,2);
            Edit6->Text=itoa(B,stro,16);
            Edit2->Text=itoa(P3,stro,2);
            Edit10->Text=itoa(ADC[IR],stro,16);
            Edit20->Text=itoa(Ram[SP],stro,16);
}
void __fastcall TForm1::Button1Click(TObject *Sender)
{  //функция клавиши  Сброс
 uchar i;
 int k;
 //формирование декодера команд - преобразование кода в порядковый номер
 //-----------------------------------------
 for(k=0;k<0x100;k++)    //сброс декодера команд
              ADC[k]=0;

 ADC[0x5a]=0x7f; //несуществующий код операции = микропрограмме входа в прерывание
 uchar j=0;
 //начальный адрес j-ой микропрограммы  RAMM=(j<<3).000, j=ADC[IR]
```

```
 ADC[0]=j++; //команда NOP-->0    j=0
 ADC[02]=j++; // команда ljmp ad   j=1
 ADC[0x95]=j++; // subb a,#d  j=2
 ADC[0x22]=j++; // ret j=3

 for(i=0x36;i<0x3D;i++) ADC[i]=j; j++;  //j=4 команда subb a,@r0

for(uchar i=0x11;i<0xF1;i=i+0x10) ADC[i]=j; j++;  // j=5 команды acall met

ADC[0x03]=j++;  // j=6 команда rr a
ADC[0xC9]=j++;   // j=7 xch a,r1
ADC[0xC5]=j++; //j=8 xch a,ad

//сброс микропрограммной памяти
//-----------------------------------------
 for(k=0;k<1000;k++) ROMM[k]=0;
//сброс декодера микркоманд
//-----------------------------------------
 for(i=0;i<16;i++) DCM[i]=0;
//сброс программной памяти
//-----------------------------------------
  for(i=0;i<100;i++) Code[i]=0;

//начальное состояние регистров-------------------
 Ram[Sp]=SP=07;
 Ram[0]=0x11; //значение R0
 Ram[1]=0x35; //R1
 Ram[Acc]=ACC=0;
 Ram[Psw]=PSW=0x81;
 Ram[0x11]=0x11;
 Ram[0x99]=0x77;
 RAMM=0;  //пустая микрокоанда, например, if(0);
 P3=0xff; //начальное состояние порта

 // тест-программа в памяти программ--------------
 //0:  ljmp start     0x02 00  0x23
         //0x3-0x3f команды в таблице векторов
 //22: ret           0x22
 //23: subb a,#10     0x96 0x10
 //25: subb a,@r0     0x36
 //26: rr     0x03
 //28: acall 0x22    0x11 0x22
 //29: xch a,r1   0xC9
 //30: xch a,0x99  0xC5 0x99
 //2a: nop           0  //конец программы
PC=0;
Code[PC++]=0x02; Code[PC++]=0; Code[PC++]=0x23; //ljmp 23
//Code[0x03]=0x32;  //reti  0
//Code[0x13]=0x32;  //reti 1

PC=0x22;
Code[PC++]=0x22;  //ret
Code[PC++]=0x95; Code[PC++]=0x10; //subb a,#10
Code[PC++]=0x36;  //subb a,@r0
Code[PC++]=0x03; //Code[PC++]=0xe7; //rr a
Code[PC++]=0x11; Code[PC++]=0x22; //acall 0x22
Code[PC++]=0xC9;
Code[PC++]=0xC5; Code[PC++]=0x99;
Code[PC++]=0;
PC=0; //начало программы
 StateMCU(); //начальное состояние MCU
 Instr->Clear();
}

char odd(void) //формирование признака четности
{
   char yy;
```

```
   char x0=ACC&1;
   char x1=ACC&2;
   char x2=ACC&4;
   char x3=ACC&8;
   char x4=ACC&0x10;
   char x5=ACC&0x20;
   char x6=ACC&0x40;
   char x7=ACC&0x80;
   yy=(x0!=0)^(x1!=0)^(x2!=0)^(x3!=0)^(x4!=0)^(x5!=0)^(x6!=0)^(x7!=0);
   return yy;
}

bool W7,Cc;
chartobool(void) //формирование булевских переменных
{    W7= (Wrk&0x80)? -1:0;  //Wrk[7]
     Cc= (PSW&0x80)? -1:0;  //PSW[7]=C
}

void __fastcall TForm1::PSWC(char *simv)
{   int AB=0;
    AnsiString Q,S;

    S.printf("%s",simv);
    Q.printf("subb  ");  //символов 7
    if(S==Q)
        AB=RA - RB - ((PSW&0x80)>>7); PSW= (AB<0)? PSW|0x80 : PSW&0x7f;

    Q.printf("andc ");  //символов 7
    if(S==Q)
        PSW= (Wrk1&(1<<(Wrk&0x7)))? PSW|0x80 : PSW&0x7f;
     (odd())? PSW|=1: PSW&=0xfe;

     char x,y,z;  //битовые переменные
}
//---------------------------------------------------------------------------
//Функция клавиши ШАГ - исполнение тест-программы
void __fastcall TForm1::Button2Click(TObject *Sender)
{   //вход в прерывание
  if(!(((CheckBox2->State==cbChecked)&&int1)||((CheckBox3->State==cbChecked)&&int0)))
     IR=Code[PC++];  //ROMM[0]  микрокоманда чтения команды
  else
     if (CheckBox4->State==cbChecked)
      {        IE1=IE; IE=0;//сохранение бита разрешения прерывания-восстанавливается в reti
              SP++;  //преинкремент SP,
              IR=0x5a; //IR=микропрограмма входа (0x5a-неиспользуемый код команды)
        if((CheckBox3->State==cbChecked)&&int0) //маска int1
          {
               CheckBox3->State=cbUnchecked; int0=0;
                     icod=0x13;//вектор прерывания
                 Instr->Text=" int0";
                     }
        if ((CheckBox2->State==cbChecked)&&int1) //маска int0
           {
                CheckBox2->State=cbUnchecked; int1=0;
                     icod=0x03; //вектор прерывания
                 Instr->Text=" int1";
                     }
      }

        Coda->Text=itoa(IR,stro,16); //код в окне Coda
        //начальная микрокоманда всех микропрограмм
        if(IR==0)
             {Instr->Text="конец программы"; PC--; goto finish; }
        MicroCodMem("Acx=Pc,Basb=Code,Wrc=Ir,Unicod=Pc_ ");
        //формирование кода микрокоманды по описанию структурной микрокоманды
        RAMM=ADC[IR]<<3; //адрес начала микропрограммы в RAMM
    switch(ADC[IR])          //декодирование команды
```

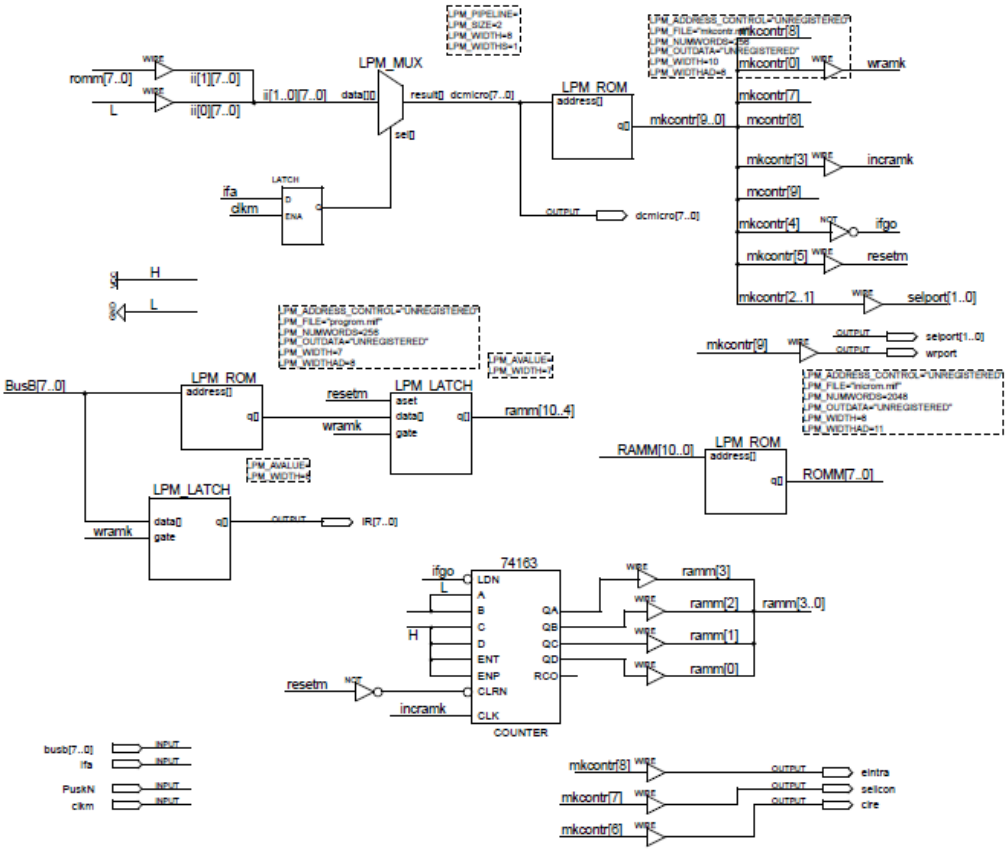
```
{
  case 1: //ljmp adr
        // 1 микрокоманда   -чтение старшего байта адреса в Wrk
        { Wrk=Code[PC++]; RAMM++;
          MicroCodMem("Acx=Pc,Basb=Code,Wrb=Wrk,Unicod=Pc_,Ramm=Ramm_ ");
        }
        // 2 микрокоманда - чтение младшего байта адреса в PCL
        {
        PC=Code[PC]|(Wrk<<8); RAMM=0;
           MicroCodMem("Acx=Pc,Basb=Code,Wrb=Pcl,BasC=Wrk;Wrc=Pch,Ramm=Ramm0 ");
           itoa(PC,&ss[0],16); //формирование мнемокода
      char stroka[10]="ljmp ";
      Instr->Text=StrCat(stroka,ss);
        }
          break;
  case 3: //ret
         // 1 микрокоманда
         {  PC=Ram[SP--]<<8; RAMM++;     //PCH
            MicroCodMem("BasA=SP,Basb=Ram,Wrb=Pcl,Unicod=_Sp,Ramm=Ramm_ ");
         }
         // 2 микрокоманда
         {  PC=PC|Ram[SP]; RAMM++; //PCL
            MicroCodMem("BasA=SP,Basb=Ram,Wrb=Pch,Unicod=_Sp,Ramm=Ramm_ ");
          }
         // 3 микрокоманда
         {  Ram[Sp]=SP; RAMM=0;
            MicroCodMem("BasA=SP,Basb=Ram,Wrb=Pch,Unicod=SP_,Ramm=Ramm0 ");
            Instr->Text="ret";
          }
            break;
  case 4:       //микропрограмма subb a,@ri
      // 1 микрокоманда
     //чтение операндов из ACC и памяти Data по адресу ri из банка PSW(rs1.rs0)
        {RA=ACC, RB=Ram[Ram[(PSW&0x18)|(IR&0x3)]];RAMM++;
          ss[0]=(IR&0x1)|0x30; ss[1]=0; //формирование и вывод мнемокода команды
          char stroka[10]="subb a,@r";
          Instr->Text=StrCat(stroka,ss);
            //кодирование структурной микрокоманды
      MicroCodMem("Basc=Acc,Basb=Ram,Basa=Ir,Wrb=Ra,WrC=Rb,Ramm=Ramm_ ");
          }
      // 2 микрокоманда - операция в АЛУ и формирование признаков результата
        {
         ACC=RA-RB-((PSW&0x80)>>7), RAMM++;
                   char tt[]="subb  "; //формирование мнемокода
                      PSWC(&tt[0]);
                 //кодирование структурной микрокоманды
         char str1[]="Basb=Ralu,Wrb=Acc,Alu=Subb,Pswf=subb,Unicod=Wrpsw,Ramm=Ramm_ ";
         MicroCodMem(&str1[0]);
          }
      // 3 микрокоманда -сохранение ACC в SFR
        {Ram[Acc]=ACC;  RAMM++;
          char str1[]="Basc=Aaa,Wrc=Ram,Basa=Acc,Ramm=Ramm_ ";
          MicroCodMem(&str1[0]);
          }
      // 4 микрокоманда - сохранение PSW в SFR и завершение микропрограммы
        {Ram[Psw]=PSW; RAMM=0;
        char str1[]="Basc=Psw,Wrc=Sfr,Basa=Psw,Ramm=Ramm0 ";
        MicroCodMem(&str1[0]);
          }
             break;
  case 2:      //микропрограмма subb a,#d
     // 1 -чтение операндов из ACC и памяти Code
     {RA=ACC, RB=Code[PC++]; RAMM++;
MicroCodMem("Basa=Pc,Basb=Code,Basc=Acc,Wrb=Ra,Wrc=Rb,Unicod=Pc_,Ramm=Ramm_ ");
        itoa(RB,&ss[0],16); //формирование мнемокода
      char stroka[10]="subb a,#";
      Instr->Text=StrCat(stroka,ss);
```

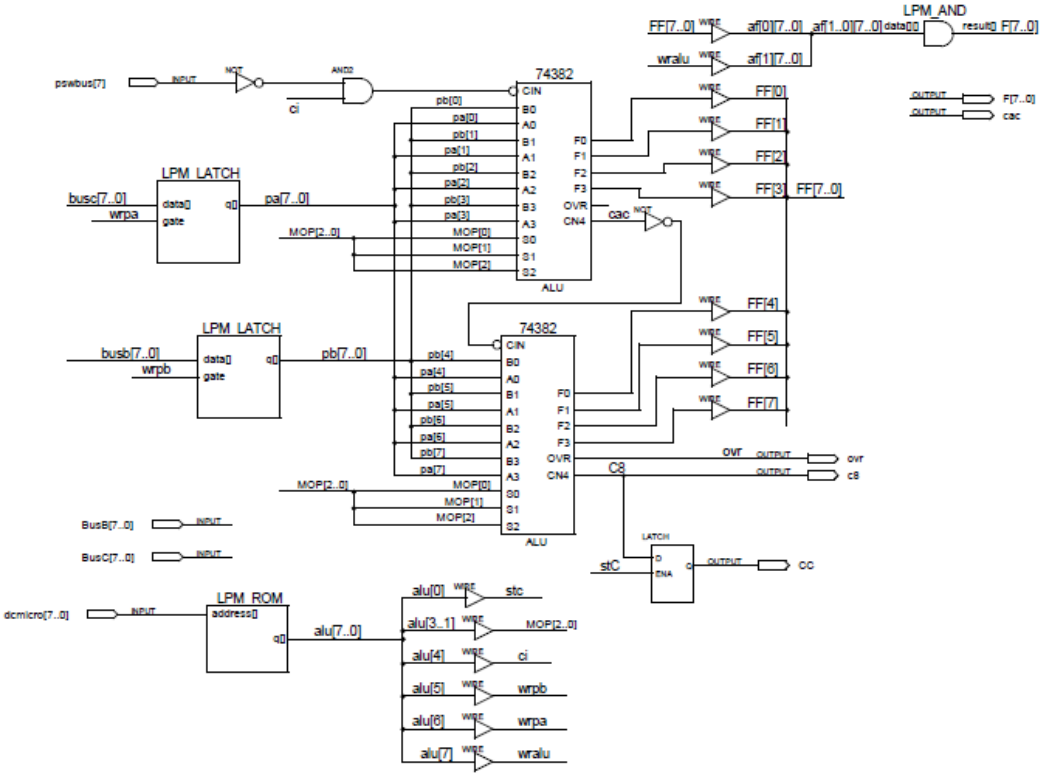
```
      }
    //2 - - операция в АЛУ и формирование признаков результата
    {
        ACC=RA-RB - ((PSW&0x80)>>7), RAMM++;
        char tt[6]="subb  "; PSWC(&tt[0]);
  MicroCodMem("Basb=Alu,Wrb=Acc,Alu=Add,Pswf=subb,Unicod=Wrpsw,Ramm=Ramm_ ");
      }
    //3- микрокоманда -сохранение ACC в SFR
     {Ram[Acc]=ACC;  RAMM++;
         MicroCodMem("Basc=Acc,Wrc=Ram,Ramm=Ramm_,Basa=Acc ");
        }
    //4-  микрокоманда - сохранение PSW в SFR и завершение микропрограммы
     {Ram[Psw]=PSW; RAMM=0;
     MicroCodMem("Basc=Psw,BasA=Psw,Ramm=Ramm0,Wrc=Ram ");
        }
          break;
case 5:         // микропрограмма acall  met
    //1 - чтение второго байта команды и преинкремент указателя
      {
    Wrk=Code[PC++]; SP++; RAMM++;
    MicroCodMem("Acx=Pc,Basb=Code,Wrb=Wrk,Unicod=Pc_,Sp=Sp_,Ramm=Ramm_ ");
    itoa(Wrk,&ss[0],16);
    char stroka[10]="acall ";
    Instr->Text=StrCat(stroka,ss);
        }
     //2- запись в Стек PC(7-0) и постинкремент указателя
      {
    Ram[SP++]=PC;RAMM++;
    char str1[]="Wrc=Ram,Basa=Sp,Basc=Pcl,Sp=Sp_,Ramm=Ramm_ ";
    MicroCodMem(&str1[0]);
        }
    //3- PC(15-8) --> Стек
      {
    Ram[SP]=(PC>>8); RAMM++;
    char str1[]="Wrc=Ram,Basa=Sp,Basc=Pch,Ramm=Ramm_ ";
    MicroCodMem(&str1[0]);
         }
    //4-формирование новогоPC
    {
    PC=((PC&0xf800)|Wrk)|((IR&0xE0)>>5)<<8;RAMM++;
    char str1[]="Basb=Acall,Basc=Wrk,Wrb=Pch,Wrc=Pcl,Ramm=Ramm_ ";
    MicroCodMem(&str1[0]);
                }
    //5 - сохранение продвинутого указателя в SFR и завершение микропрограммы
     {
    Ram[Sp]=SP;  RAMM=0;
    char str1[]="Basc=Sp,Wrc=Ram,Basa=Sp,Ramm=Ramm0 ";
    MicroCodMem(&str1[0]);
       }   break;
case 6: //микропрограмма rr a
    // 1- чтение адреса бита из второго байта команды
       {
       ACC=ACC>>1; RAMM=0;
       MicroCodMem("Acx=Pc,Basb=Code,Wrb=Wrk,Ramm=Ramm_ ");
                      //формирование мнемокода
       char stroka[12]="rr a";
       Instr->Text=stroka;
         }break;
  case 0x7f: //вход в прерывание
    { //микрокманда
      Ram[SP++]=PC; RAMM++;
      char str1[]="Basa=SP,BasC=Pcl,Wrc=Ram,SP_,Ramm=Ramm_ ";
      MicroCodMem(&str1[0]);
      //микрокоманда
      Ram[SP]=PC>>8; PC=icod; IE1=IE;RAMM=0;
      char str2[]="Basa=SP,BasC=Pch,Wrc=Ram,SP_,Ramm=Ramm_Ramm=Ramm0 ";
      MicroCodMem(&str2[0]);
```

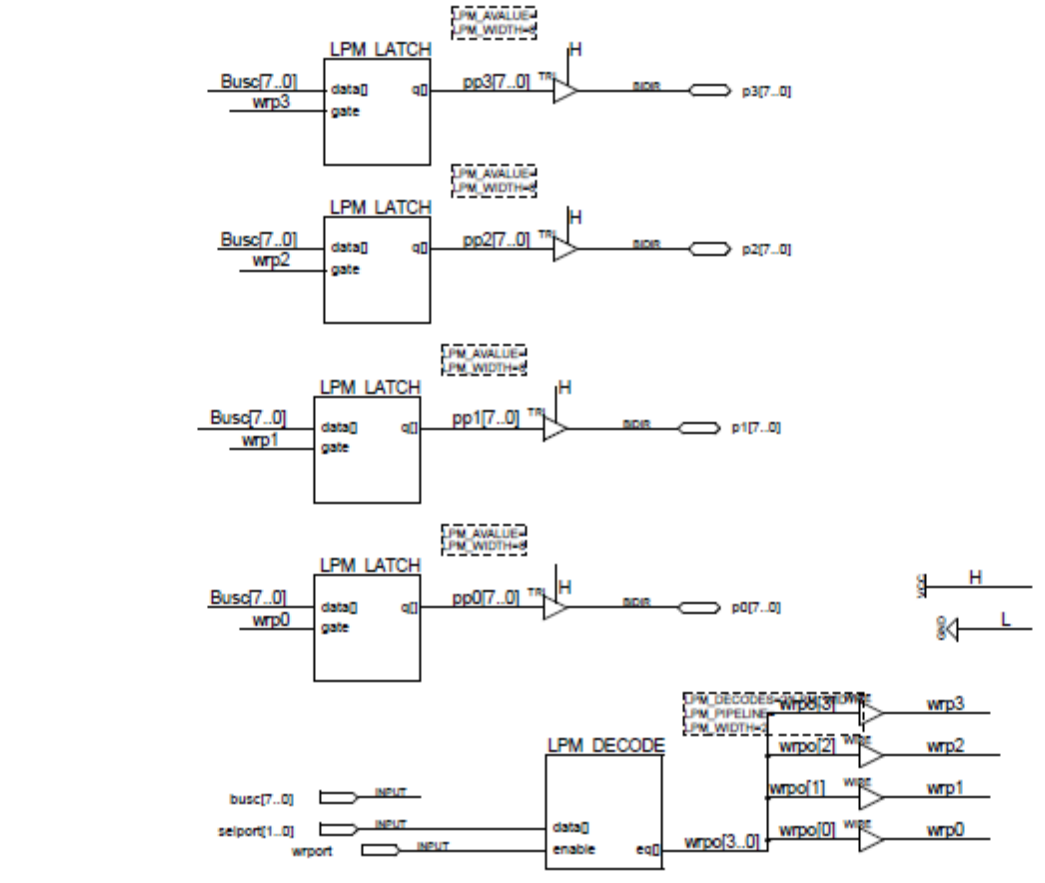
```
            }  break;
            case 7: //xch a, rn
                 {
                     RA=ACC;
                     RB=Ram[(PSW&0x18)|(IR&0x3)];RAMM++;
                     if (RA!=RB)
                     {
                     ss[0]=(IR&0x3)|0x30; ss[1]=0; //формирование и вывод мнемокода команды
                     char stroka[10]="xch a,r";
                     Instr->Text=StrCat(stroka,ss);
                     RA^=RB;
                     RB^=RA;
                     RA^=RB;
                     ACC=RA;
                     Ram[Acc]=ACC;
                     Ram[(PSW&0x18)|(IR&0x3)]=RB;
                     }
                  }  break;
              case 8:  //xch a, ad
                      { Wrk=Code[PC++]; RA=ACC;
                      RB=Ram[Wrk];
                      itoa(Wrk,&ss[0],16);
                      char stroka[10]="xch a, ";
                      Instr->Text=StrCat(stroka,ss);
                      if (RA!=RB)
                      {
                      RA^=RB;
                      RB^=RA;
                      RA^=RB;
                      ACC=RA;
                      Ram[Acc]=ACC;
                      Ram[Wrk]=RB;
                      }
                      }  break;
           default: break;
          }
           //Вывод состояния регистров в HEX-коде
      finish: StateMCU();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::CheckBox1Click(TObject *Sender)
{
   CheckBox1->State=cbChecked;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button7Click(TObject *Sender)
{       char x;
        AnsiString S;
        hdisk=CreateFile("ADC.txt",
        GENERIC_READ|GENERIC_WRITE,
        FILE_SHARE_READ|FILE_SHARE_WRITE,NULL,
        OPEN_ALWAYS,
        FILE_ATTRIBUTE_NORMAL,NULL);
     for(char i=0; i<5;i++)
         { x=i+0x30;
          WriteFile(hdisk,&x,1,&bufd,0);
          }
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button3Click(TObject *Sender)
{
      P3^=0x8; if(P3&0x8); else int1=-1;
       Edit2->Text=itoa(P3,stro,2);
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button4Click(TObject *Sender)
{
```

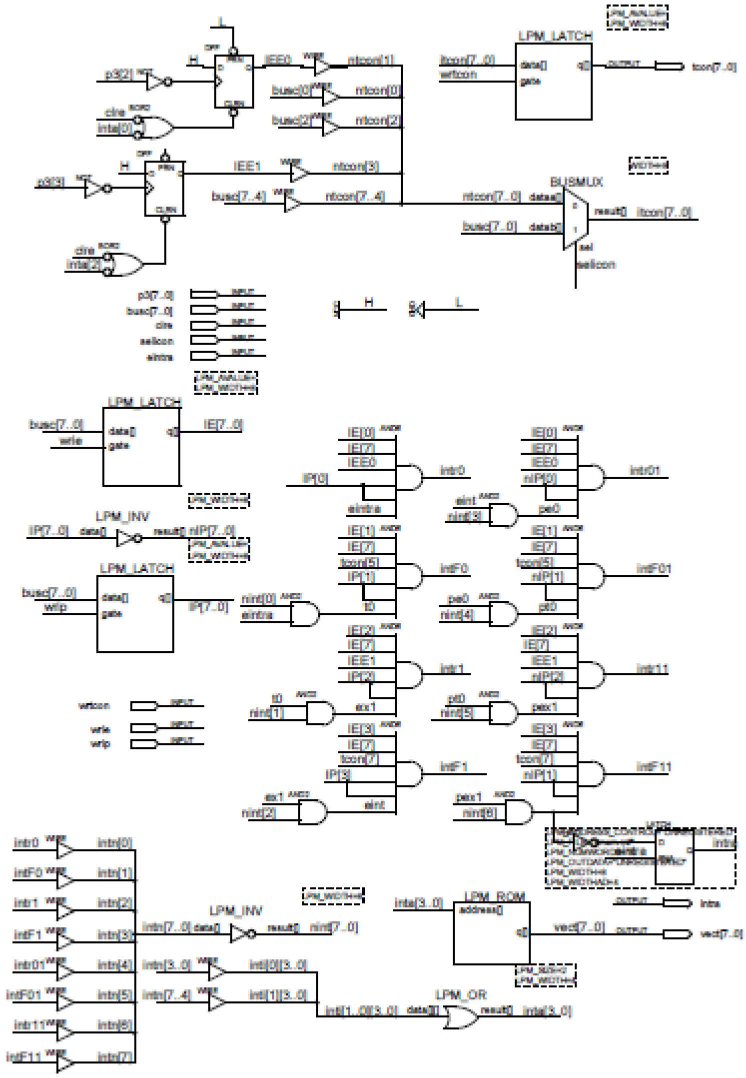
```
    P3^=0x4; if(P3&0x4); else int0=-1;
     Edit2->Text=itoa(P3,stro,2);
}
```

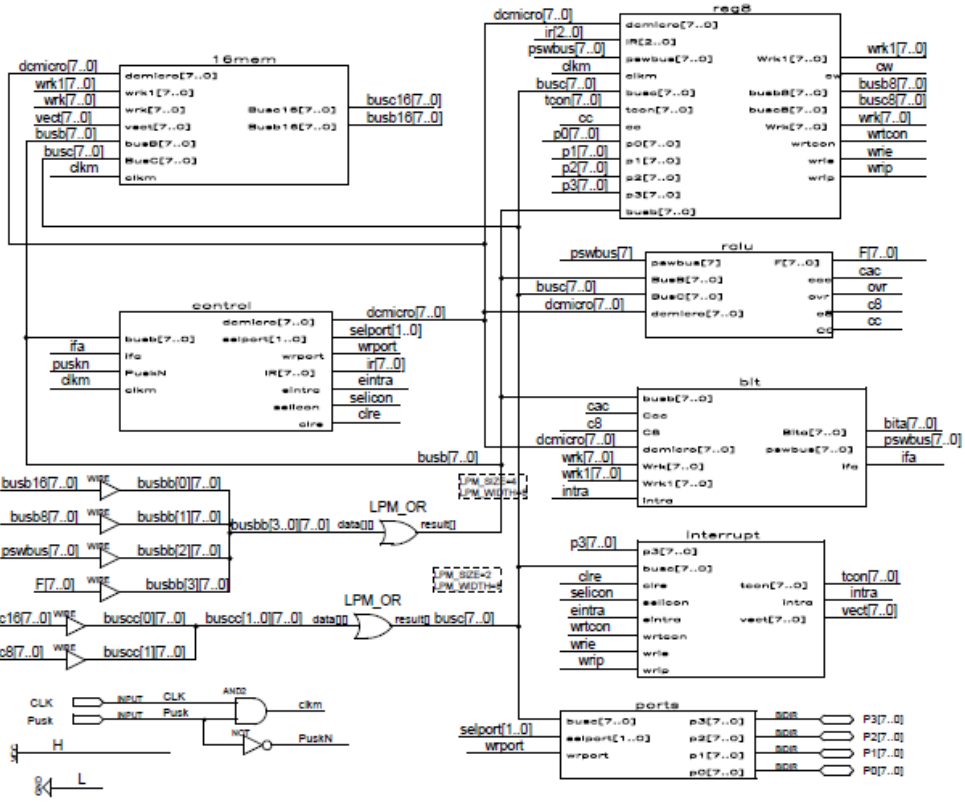
Этап 5




LPM_AND
FF[7..0]
af[0][7..0]
af[1..0][7..0]
data[][]
result[] F[7..0]
wralu
af[1][7..0]
pswbus[7]
INPUT
NOT
AND2
ci
74382
CIN
B0
A0
B1
A1
B2
A2
B3
A3
S0
S1
S2
F0
F1
F2
F3
OVR
CN4
ALU
FF[0]
FF[1]
FF[2]
FF[3]
FF[7..0]
OUTPUT
F[7..0]
cac
LPM_LATCH
busc[7..0]
wrpa
data[]
gate
q[]
pa[7..0]
MOP[2..0]
MOP[0]
MOP[1]
MOP[2]
pb[0]
pa[0]
pb[1]
pa[1]
pb[2]
pa[2]
pb[3]
pa[3]
FF[4]
FF[5]
FF[6]
FF[7]
busb[7..0]
wrpb
pb[7..0]
pb[4]
pa[4]
pb[5]
pa[5]
pb[6]
pa[6]
pb[7]
pa[7]
OVR
C8
ovr
c8
BusB[7..0]
BusC[7..0]
LATCH
D
ENA
stC
CC
LPM_ROM
dcmicro[7..0]
address[]
alu[7..0]
alu[0]
stc
alu[3..1]
alu[4]
ci
alu[5]
wrpb
alu[6]
wrpa
alu[7]
wralu


LPM_AVALUE
LPM_WIDTH=8
LPM_LATCH
Busc[7..0]
wrp3
data[]
gate
q[]
pp3[7..0]
TRI
H
p3[7..0]
wrp2
pp2[7..0]
p2[7..0]
wrp1
pp1[7..0]
p1[7..0]
wrp0
pp0[7..0]
p0[7..0]
L
LPM_DECODE
busc[7..0]
selport[1..0]
wrport
INPUT
data[]
enable
eq[]
wrpo[3..0]
LPM_DECODES=4
LPM_PIPELINE=0
LPM_WIDTH=2
wrpo[3]
wrpo[2]
wrpo[1]
wrpo[0]
wrp3
wrp2
wrp1
wrp0


LPM_LATCH
BUSMUX
LPM_INV
LPM_ROM
LPM_OR
itcon[7..0]
wrtcon
tcon[7..0]
busc[7..0]
IE[7..0]
IP[7..0]
intr0
intr01
intF0
intF01
intr1
intr11
intF1
intF11
inta[3..0]
vect[7..0]
intra


16mem
dcmicro[7..0]
wrk1[7..0]
wrk[7..0]
vect[7..0]
busb[7..0]
busc[7..0]
clkm
busc16[7..0]
busb16[7..0]
reg8
dcmicro[7..0]
ir[2..0]
pswbus[7..0]
clkm
busc[7..0]
tcon[7..0]
cc
p0[7..0]
p1[7..0]
p2[7..0]
p3[7..0]
wrk1[7..0]
cw
busb8[7..0]
busc8[7..0]
wrk[7..0]
wrtcon
wrie
wrip
relu
pswbus[7]
busc[7..0]
dcmicro[7..0]
F[7..0]
cac
ovr
c8
cc
control
dcmicro[7..0]
selport[1..0]
wrport
ir[7..0]
eintra
selicon
clre
ifa
puskn
clkm
bit
cac
c8
dcmicro[7..0]
wrk[7..0]
wrk1[7..0]
intra
busb[7..0]
bita[7..0]
pswbus[7..0]
ifa
busb16[7..0]
busb8[7..0]
pswbus[7..0]
F[7..0]
busbb[0][7..0]
busbb[1][7..0]
busbb[2][7..0]
busbb[3][7..0]
busbb[3..0][7..0]
LPM_OR
busc16[7..0]
busc8[7..0]
buscc[0][7..0]
buscc[1..0][7..0]
LPM_OR
busc[7..0]
interrupt
p3[7..0]
clre
selicon
eintra
wrtcon
wrie
wrip
tcon[7..0]
intra
vect[7..0]
CLK
Pusk
clkm
PuskN
H
L
ports
selport[1..0]
wrport
busc[7..0]
p3[7..0]
p2[7..0]
p1[7..0]
p0[7..0]
P3[7..0]
P2[7..0]
P1[7..0]
P0[7..0]