Национальный исследовательский университет информационных технологий,
механики и оптики
Кафедра вычислительной техники
Организация ЭВМ

Курсовая работа
«Проектирование микроЭВМ»

Работу выполнил студент группы P3315
Халанский Дмитрий

2016

1. Система команд

1.1. `dec @rj`

C	AC	F0	RS1	RS0	OV		P

Команда записывает в ячейку памяти, адрес которой указан в регистре R_i, значение, на единицу меньшее текущего.

Размер 1 байт

Число циклов 1

Кодирование

0	0	0	0	0	1	1	i

Механизм $(R_i) \leftarrow (R_i) - 1$

Пример `DEC @R0`

1.2. `dec a`

C	AC	F0	RS1	RS0	OV		**P**

Команда записывает в аккумулятор a значение, на единицу меньшее текущего.

Размер 1 байт

Число циклов 1

Кодирование

0	0	0	1	0	1	0	0

Механизм $A \leftarrow A - 1$

Пример `DEC A`

1.3. `orl c, bit`

C	AC	F0	RS1	RS0	OV		P

Команда считывает бит, адрес которого указан в операнде `bit`, и записывает в C результат логического сложения C с этим битом.

Размер 2 байт

Число циклов 2

Кодирование

0	1	1	1	0	0	1	0	bit

Механизм $C \leftarrow C \vee b$

Пример `ORL C, 22h`

1.4. `orl c, /bit`

C	AC	F0	RS1	RS0	OV		P

Команда считывает бит, адрес которого указан в операнде `bit`, и записывает в C результат логического сложения C с битом, инверсным данному.

Размер 2 байт

Число циклов 2

Кодирование

1	0	1	0	0	0	0	0	bit

Механизм $C \leftarrow C \vee \neg b$

Пример `ORL C, /22h`

1.5. `mov a, @rj`

C	AC	F0	RS1	RS0	OV		**P**

Команда записывает в аккумулятор a содержимое ячейки памяти, адрес которой указан в регистре R_i.

Размер 1 байт

Число циклов 1

Кодирование

1	1	1	0	0	1	1	i

Механизм $A \leftarrow (R_i)$

Пример `MOV A, @R1`

1.6. `mov a, ad`

C	AC	F0	RS1	RS0	OV		**P**

Команда записывает в аккумулятор a содержимое ячейки памяти по адресу `ad`.

Размер 2 байт

Число циклов 1

Кодирование

1 1 1 0 0 1 0 1	ad

Механизм $A \leftarrow (\text{ad})$

Пример `MOV A, P0`

1.7. `jb bit, rel`

C	AC	F0	RS1	RS0	OV		P

Команда считывает бит `bit` и, если он установлен, переходит к адресу, указанному во втором операнде.

Размер 3 байт

Число циклов 2

Кодирование

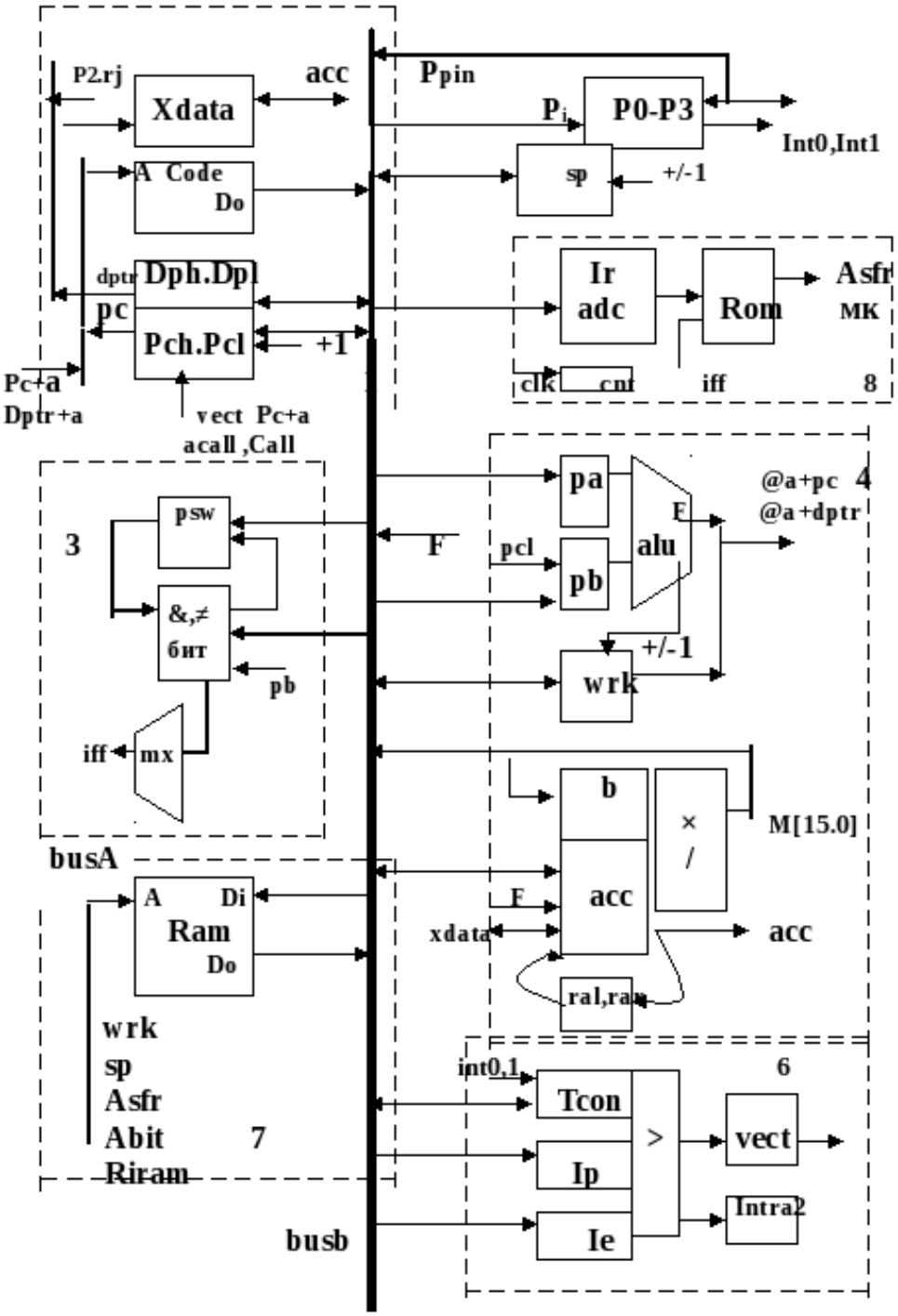
0 0 1 0 0 0 0 0	bit	rel

Механизм $\text{PC} \leftarrow \text{PC} + 3$
IF $(\text{bit}) = 1, \text{PC} \leftarrow \text{PC} + \text{rel}$

Пример `JB P1.2 LABEL`

---

2. Структура ЭВМ



3. Функциональный тест на A51

3.1. Код программы

```
cseg at 0x0
  ljmp start

cseg at 0x3
  reti

cseg at 0x4
  ret_test:
    dec a // test "dec a" (a <- 0x3C, P <- 0)
    ret   // test "ret"

  start:
    // the initial state
    mov ie,   #0x81
    mov 0x40, #0xDE
    mov 0x41, #0x3E
    mov 0x20, #0x0F // Set bits 0-3 to 1, 4-7 to 0.
    mov r1,   #0x41
    anl c,     0x04 // c <- 0

    dec @R1      // test "dec @ri"      (0x41 <- 0x3D)
    mov a, 0x40 // test "mov a, ad"    (a <- 0xDE, P <- 0)
    mov a, @R1  // test "mov a, @ri"   (a <- 0x3D, P <- 1)
    orl c, 0x1  // test "orl c, bit"   (c <- 1)
    anl c, 0x4
    orl c, /0x4 // test "orl c, /bit" (c <- 1)

    acall ret_test

    jb 0x1, pas // test "jb TRUE, rel"
  err:
    ljmp err
  pas:
    jb 0x4, err // test "jb FALSE, rel"
  suc:
    ljmp suc

end
```

4. Микропрограммирование

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop
#include <stdio.h>
#include <string.h>
#include "modelir.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------
DWORD bufd;
char str_hex[40];

typedef unsigned char uchar;
typedef unsigned int uint;
typedef unsigned long ulong;
//  ----------------------------
DWORD bri,ii;
OVERLAPPED ovr,ovrd;
HANDLE hCom,hdisk,hdisk1,htempl=0;
DCB dcb;
char coma;
char priznak; //!=0
char cod;

//  mcs51=====================================
// ---------------------------

uchar IR,    //
      Wrk,   //
      ACC,   //
      SP,    //
      PA,    //  RALU
      PB,    //  RALU
      PSW,   //
      B,     //
      Ram[256],
      Xdata[2048],
      DPH,
      DPL,
      P0,
      P1,
      P2,
      P3=0xff,    //  3
      TCON, //tf1 tr1 tf0 tr0 ie1 it1 ie0 it0
      IE,   //EA . .  es et1 ex1 et0 ex0
      IP,
      IEE, //  IE
      vect; //  ;

//    SFR-------------------
const  Sp = 0x81;       //wrsp - 1
const  Acc = 0xe0;      //wracc - 2
const  Dph = 0x83;      //wrdph  -3
const  Dpl = 0x82;      //wrdpl  -4
const  b = 0xf0;        //wrb  - 5
const  Psw = 0xd0;      //wrpsw - 6
const  p3 = 0xb0;       //wrp3 - 7
const  p0 = 0x80;       //wrp0 - 8
const  p1 = 0x90;       //wrp1 - 9
const  p2 = 0xa0;       //wrp2 - 10
const  Ie =  0xa8;      //wrie - 11
const  Ip = 0xb8;       //wrip - 12
const  Tcon = 0x88;     //wrtcon -13

char adress(char *ss)
{ char numm;
if(strcmp(ss, "Acc") == 0) return numm=Acc;
     if(strcmp(ss, "Sp") == 0)  return numm=Sp;
     if(strcmp(ss, "Dph") == 0) return numm=Dph;
     if(strcmp(ss, "Dpl") == 0) return numm=Dpl;
     if(strcmp(ss, "Psw") == 0) return numm=Psw;
     if(strcmp(ss, "p3") == 0) return numm=p3;
     if(strcmp(ss, "p0") == 0) return numm=p0;
     if(strcmp(ss, "p1") == 0) return numm=p1;
     if(strcmp(ss, "p2") == 0) return numm=p2;
     if(strcmp(ss, "Tcon") == 0) return numm=Tcon;
```

```
      if(strcmp(ss, "Ie") == 0) return numm=Ie;
      if(strcmp(ss, "Ip") == 0) return numm=Ip;
      if(strcmp(ss, "b") == 0) return numm=b;

      return 0;
 }

uint RAMK; //
uint PC;    //PCH=PC>>8,PCL=PC
uint DPTR; //DPH=DPTR>>8,DPL=DPTR
uint ROMM[128];    //  - ( <10) + 3 <128

uchar WRSFR[128];

//  ()----------------------
bool EA,EX0,EX1, //
      IE0,IE1,     //
      intr0,intr1, //
         intra,intra2, // -
      Cc,   //    RALU
      W7;   // wrk[7]

//ulong codDCM; //32-
uchar CODE[2048];//     +

uchar ADC[0x100];   //

//============================

//============================
uchar *stro="      ";   //
uchar prizn;    //

char CodMo(char *ss,char *sss)
{  //    *sss    *ss
      char locs[5];
      char x;
      char *namemo="      ";
      char *namemo1="        ";
          namemo1=namemo;
        cod=1;
       for(char ii=0; ; ii++)
      {         x=*ss; ss++;
          if(((x==',')||(x=='  '))&&((strcmp(namemo1,sss)==0)))
                {priznak=1; return cod;}
          if (x==',')
                {         cod++;
                  namemo= namemo1;

                }
          if(x=='  ')  goto exita;
                //StrCat(itoa(PC,&locs[0],16),ss);

          if(x!=',')
          {*namemo++=x; *namemo=0;}
        }
     exita:  cod=0; //
     return cod;
}

//

uint DCM1[64];//MCROM[0x100];
uchar DCM8[64],DCM2[64],DCM3[64];
unsigned long DCM4[64],DCM7[64];

//     i.code
uint CodDCM1; //  7  16mem
uint CodDCM2; //  6  reg8
uint CodDCM3; //  7 bita
ulong CodDCM4; //  19  ralu
uint CodDCM6; //  7  interrupt
ulong CodDCM7; //  17 bus8
uint CodDCM8; //  3 control
//  4 contr

uchar i1; // ,   DCM[i]

//    ============================

//1  16mem===(DCM 1)=============================================================

struct mik16{//     16mem
        uchar selpc;  // 3
        uchar unicod16;  //6
        uchar selaxa;  //1
        }mk16 ;               //7 bit
//
```

```
//   -  ,    -
//char *QQQ ="    ";
char *Selpc ="Acall,Pc+a,Vect,Call ";
 //              0      1     2     3
char  *Unicod16 = "Wrpc,Wrxda,Incdptr,Incpc,Wrreg,Rdreg ";
//     1      2      3      4
// 0 -
char *Selaxa = "Dptr,P2wrk ";
            //     0      1

void Newmk16(void)
  { mk16.selpc=0;mk16.unicod16=0;}

void TakeCode1(char *stpole,char *stmo)
{ //     *stpole     *stmo
        if(strcmp(stpole, "Selpc") == 0)
                mk16.selpc=CodMo(Selpc,stmo)-1;
        else
        if(strcmp(stpole, "Selaxa") == 0)
                mk16.selaxa=CodMo(Selaxa,stmo)-1;
        else
        if(strcmp(stpole, "Unicod16") == 0)
          mk16.unicod16 |= 1<<(CodMo(Unicod16,stmo)-1);
}

//3   BIT=====(DCM 3)=============================================================
   char bita;
   char *Unibit="Bitor,Bitand,Nebit,Newbit,Setb,Wlocpsw,Selpsw,Clrb,Movcb ";
//                  1      2      3      4      5       6       7      8      9     bit

void TakeCode3(char *stpole,char *stmo)
{     if(strcmp(stpole, "Unibit") == 0)
           bita |= 1<<(CodMo(Unibit,stmo)-1);

}
//4   RALU=====(DCM 4)=============================================================
  struct ralu {
          uchar mop; //3
          uchar selacc; //3
          uchar selb; //2
          uchar uniralu;   //10
          uchar selpa;     //1
          uchar selpb;   //2
        } alu;          //20

char *Mop="L,Suba,Subb,Add,Or,Xor,And,H ";
   //       0   1     2     3   4   5     6   7        bit 3
char *Selacc="Busb,M[7.0],Quot,Lac,Rac,F,Xdata ";
   //            0       1     2    3     4   5     6          bit 3
char *SelB="Busb,M[15.8],Remain,Wrk ";
   //           0       1       2     3                bit 2
char *Uniralu ="Wrbloc,Wrpb,Wrpa,Ci,Mul,Div,Wrwrk,Incwrk,Count,Selpa ";
   //               1      2     3    4    5    6     7      8      9      10        bit 10
char  * Selpb =  "L,Pcl,Dapb";
   //              0    1      2           bit 2
 char  *Selpa = "Pa,Acc";
           //      0    1


void Newalu(void)
{ alu.mop=0; alu.uniralu=0;alu.selb=0;alu.selacc=0;
  alu.selpa=0; alu.selpb=0;
  }

void TakeCode4(char *stpole,char *stmo)
{ //     *stpole     *stmo

        if(strcmp(stpole, "Mop") == 0)
             alu.mop =CodMo(Mop,stmo);
        else
        if(strcmp(stpole, "Selacc") == 0)
             alu.selacc =CodMo(Selacc,stmo)-1;
        else
        if(strcmp(stpole, "SelB") == 0)
             alu.selb =CodMo(SelB,stmo)-1;
        else
        if(strcmp(stpole, "Uniralu") == 0)
          alu.uniralu = alu.uniralu |(1<<(CodMo(Uniralu,stmo)-1));
        else
        if(strcmp(stpole, "Selpa") == 0)
        alu.selpa =CodMo(Selpa,stmo)-1;
        else
         if(strcmp(stpole, "Selpb") == 0)
          alu.selpb =CodMo(Selpb,stmo)-1;
}

//4   Ports======(DCM 2 )=============================================================
```

```
struct mk8{//ports
       uchar adpi;   // 2
      uchar adpin;   //2
       uchar uniport;   //5
       }mkport;                 //9 bit

char *Uniport="Wrpi,Xadr,Altf2,Altf3,Xdata ";
      //          1     2     3     4     5          bit 5
char      * Adpi ="Rp0,Rp1,Rp2,Rp3";
      //           0     1     2     3          bit 2
char  *Adpin = "Pp0,Pp1,Pp2,Pp3";
      //          0    1     2     3          bit 2

void TakeCode2(char *stpole,char *stmo)
  //
{
      if(strcmp(stpole, "Uniport") == 0)
         mkport.uniport |= 1<<(CodMo(Uniport,stmo)-1);
      else
      if(strcmp(stpole, "Adpin") == 0)
         mkport.adpin |= 1<<(CodMo(Adpin,stmo)-1);
      else
          if(strcmp(stpole, "Adpi") == 0)
         mkport.adpi |= 1<<(CodMo(Adpi,stmo)-1);
 }


//5  Interrupt=============================================================
 //  Contr

 //6    bus8   (dcm7 )
//=========================================================================
 //          SFR
struct BUSS8{ //     ralu
          uchar selbusb; //4
          uchar selbusa;  //3
          uchar asfr;   //7
          uchar unibus8; //4
        } mkbus8;        //  18
char *SelbusB="Com,Wrk,Pch,Pcl,Pinpi,Rdpi,Nwtcon,Acc,Ram,Ff,Psw ";
//                0   1    2    3     4     5      6     7   8   9   10    4
char *SelbusA="Riram,Wrk,Asfr,Sp,Abitwrk ";
//                 0     1    2    3     4        3
char *Unibus8="Ari,Wram,Selsp,Incsp ";
//                 1     2     3     4            4
//adsfr -
char  *Asfr ="Acc,Sp,B,Psw,Dpl,Dph,P0,P1,P2,P3,Tcon,Ip,Ie "   ;

void Newbus8(void)
{ mkbus8.selbusb=0;
  mkbus8.selbusa=0;    mkbus8.asfr=0;   mkbus8.unibus8=0;
}

void TakeCode7(char *stpole,char *stmo)
{ //    *stpole     *stmo

      if(strcmp(stpole, "SelbusA") == 0)
           mkbus8.selbusa =CodMo(SelbusA,stmo)-1;
      else
      if(strcmp(stpole, "SelbusB") == 0)
           mkbus8.selbusb=CodMo(SelbusB,stmo)-1;
      else
      if(strcmp(stpole, "Asfr") == 0)
                  { mkbus8.asfr=*stmo&0x7f; priznak=1;}
      else
      if(strcmp(stpole, "Unibus8") == 0)
          mkbus8.unibus8|=1<<(CodMo(Unibus8,stmo)-1);
  }
 //7   Control  (dcm8.mif )  + interrupt
//=========================================================================
 //          SFR
uchar unicontr;
char *Unicon = "Clramk,Ramk1,Wrir,Eintra,Clreq,Clrinta ";
//                    1       2      3    4      5       6
void TakeCode8(char *stpole,char *stmo)
{    if(strcmp(stpole, "Unicont") == 0)
         unicontr =1<<(CodMo(Unicon,stmo)-1);
 }

 //   ROMM  (DCM9.mif )
//=========================================================================
 //          SFR
uint ifromm; //Selif,Neiff,dcrom)
char *Selif ="L,Cc,Zacc,Bitwrk,Wrk7,PswC,Intra";
//              0 1     2       3     4    5      6            3

void Takeromm(char *stpole,char *stmo)
{ //    *stpole     *stmo
```

```
        if(strcmp(stpole, "Selif") == 0)
                ifromm |=CodMo(Selif,stmo);
        if (strcmp(stmo, "Neiff"))      //
              ifromm|=0x80;
     }

//==================
void __fastcall TForm1::TakeCode(char *stpole,char *stmo)
{ //    *stpole      *stmo
        priznak=0;
        TakeCode1(stpole,stmo); //mem16
        TakeCode2(stpole,stmo);  //ports
         TakeCode3(stpole,stmo);  //bita
         TakeCode4(stpole,stmo);  //ralu
         TakeCode7(stpole,stmo);  //bus8
         TakeCode8(stpole,stmo);  //Control
         Takeromm(stpole,stmo);  //ROMM
         if(priznak) return; //
         Edit8->Text="␣";//StrCat("PC=",ss);
}
void EqualCode(void)
{      char priznak; //
     for(char i=0;i<i1; i++)
      { priznak=0;
      if(DCM1[i]!=CodDCM1){priznak=1; continue;}   // ;
      if (DCM2[i]!=CodDCM2){priznak=1; continue;};
      if (DCM3[i]!=CodDCM3){priznak=1; continue;};
      if (DCM4[i]!=CodDCM4){priznak=1; continue;};
      if (DCM7[i]!=CodDCM7){priznak=1; continue;};
      if (DCM8[i]!=CodDCM8){priznak=1; continue;};
      if(priznak==0) ROMM[RAMK]=i|(ifromm<<6); break;
      }
      if(priznak==1) //
           {DCM1[i1]=CodDCM1;     //mem16
           DCM2[i1]=CodDCM2;      //reg8
           DCM3[i1]=CodDCM3;      //bita
           DCM4[i1]=CodDCM4;      //ralu
           DCM7[i1]=CodDCM7;      //bus8
           DCM8[i1]=CodDCM8;      //Control
            //
             i1++;
            }
}

void CompareCode(void) //
{
    CodDCM1=(((mk16.selpc<<1)|mk16.selaxa)<<4)|mk16.unicod16;

    //CodDCM2=mkporta;

    //CodDCM3=(bita.selpsw<<6)|bita.unibit;

    //CodDCM4= (((((((((alu.mop <<3)|alu.selacc)<<2)\
      //                        |alu.selb)<<1)|alu.selpa)<<1)|\
        //                          alu.selpb)<<9)|alu.uniralu;

    CodDCM7 =(((((mkbus8.selbusb<<3)|mkbus8.selbusa)<<7)\
                                  |mkbus8.asfr)<<3)|mkbus8.unibus8;

    CodDCM8 = unicontr;
   //
   //
   //0- =0
    ROMM[RAMK]=ifromm<<6; //+dcmicro

    EqualCode();
}

char ss[10];
void __fastcall TForm1::MicroCodMem(char *ss) //
{    //*ss -
   if(CheckBox1->State==cbChecked)//
   { uchar *namepole="␣␣␣␣␣␣␣␣␣␣"; //
    uchar *namemo="␣␣␣␣␣␣␣␣␣␣";  //
    uchar x,xx=',';  //
    uchar *namepole1="␣␣␣␣␣␣␣␣␣␣";
    uchar *namemo1="␣␣␣␣␣␣␣␣␣␣";  //,
    namepole1=namepole;      //
    namemo1=namemo;
    //
    //Newmk16();  Newalu();  Newbus8(); Newbita();
    //ifromm=0; unicontr=0;  mkporta=0;

    {for(char ii=0; ; ii++)
     {       x=*ss; ss++;
          if(xx=='=')   //  -
             if((x==',')||(x=='␣')) //
```

```
                    {TakeCode(namepole1,namemo1);
                             namepole=namepole1; namemo= namemo1;
                             for(char i=0;i<10;i++)  //7
                                      *namemo++=*namepole++=0;//' ';
                       namepole=namepole1; namemo= namemo1;
                           xx=x;
                      if(x==' ') goto finn; //
                      }
              else  {*namemo++=x; *namemo=0;}  //
            else   //xx=','
               if(x=='=') xx='='; //
               else {*namepole++=x;
                              *namepole=0;}
        }
        finn: CompareCode(); //
                        //
     }                    //
    }
}

//=================================

void __fastcall TForm1::GoToInt(void) //     \
                                                  //+ 0-
{
        //if(CheckBox9->State==cbUnchecked)

            if(!intra2)
              { if(IE0) //IE0 -    TCON
                 {intr0= EX0&((TCON&0x2)!=0)&EA;//
                 CheckBox5->State=cbChecked;  P3^=0x4;}
                 else
                   if(IE1)
                    {intr1= EX1&((TCON&0x8)!=0)&EA&!intr0;
                   CheckBox6->State=cbChecked;P3^=0x8;}
              intra=intr0 | intr1;
              if(intra) intra2=1;   //
              if(intr0) {vect=0x3; intr0=0; IE0=0;
                          CheckBox5->State=cbUnchecked;        }
              if(intr1) {vect=0x13; intr1=0; IE1=0;
                          CheckBox6->State=cbUnchecked;         }
              if(intra)   //
                 CheckBox9->State=cbChecked; //
            }
}


void __fastcall TForm1::ClearInt(void)
{
/* if(CheckBox5->State!=cbUnchecked)
       {IE0=0; TCON&=0xFD; Ram[Tcon]=TCON;
       intr0=0; //
       CheckBox5->State=cbUnchecked;
       P3|=0x4;
   //   Int0   Tcon
     }
  if(CheckBox6->State!=cbUnchecked)
 {//   Int1   Tcon
    IE1=0; TCON&=0xF7; Ram[Tcon]=TCON;
      intr1=0; //
        CheckBox6->State=cbUnchecked;
        P3|=0x8;
        }
        */
}

//---------------------

char odd(void) //   P
{
   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;
}

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

     S.printf("%s",simv);
```

```
    Q.printf("add  ");  // 7
    if(S==Q)
       { AB=PA + PB; PSW= (AB&0x100)? PSW|0x80 : PSW&0x7f;  //C
        PSW=((~(PA^PB)&0x80)&(PA^AB))? PSW|0x04 : PSW&0xfb; //OV
        }

    Q.printf("subb  ");
    if(S==Q)
       { AB=PA - PB - (PSW>>7); PSW= (AB&0x100)? PSW&0x7f : PSW|0x80; //=C
        PSW=((~(PA^(~PB))&0x80)&(PA^AB))? PSW|0x04 : PSW&0xfb; //OV
        }


    Q.printf("andc ");  // 7
    if(S==Q)
        PSW= (PB&(1<<(Wrk&0x7)))? PSW&0xff : PSW&0x7f; //c=c&bit

// *ROUANTH

    Q.printf("orlc ");  // simvolov, yopta
    if (S == Q)
        PSW |= (PB&(1 << (Wrk & 0x7))) ? 0x80 : 0x00;

// ROUANTH*

     (odd())? PSW|=1: PSW&=0xfe; //P

}

//=============================
void __fastcall TForm1::Reg(char i) //
   { char Q[6];
     Q[0]='R'; Q[1]=0x30+i; Q[2]='='; Q[3]=0;
     if(Ram[i])
           ComboBox2->Items->Add(StrCat(Q,itoa(Ram[i],stro,16)));
    }
void __fastcall TForm1::Stack(int i)
   { char Q[8];
     for(char j=0; j<8; j++) Q[j]=0;
     Q[0]='D'; Q[1]='[';
     Q[2]= (i<=9)? 0x30+i : 0x57+i;
     Q[3]=']'; Q[4]='='; Q[5]=0;
           ComboBox6->Items->Add(StrCat(Q,itoa(Ram[i],stro,16)));
    }
char Q[9];
void __fastcall TForm1::bitreg(unsigned char var)
{
     Q[8]=0;
     uchar j=0x80;
      for(char i=0;i<8;i++)
               { Q[i]=(var&j)? '1' : '0';
                 j=j>>1;
                 }
 }

 void __fastcall TForm1::StateMCU(void)
{ //

            //uchar i,j=0x80;
            Acu->Text=itoa(ACC,stro,16);
            Work->Text=itoa(Wrk,stro,16);
            ProgCnt->Text=itoa(PC,stro,16);
            bitreg(PSW);  //
            Edit1->Text=&Q[0];  //   PSW
            bitreg(P3);
            Edit2->Text=&Q[0];
            W7= (Wrk&0x80)? 1:0;  //Wrk[7]
         }

uchar vector[0x10];
void __fastcall TForm1::Button1Click(TObject *Sender)
{  //
 uchar i;
 int k;
 //   ============================
   vector[0x1]=0x03; //int0
   vector[0x2]=0x0b; //tf0
   vector[0x4]=0x13; //int1
   vector[0x8]=0x1b; //tf1
   vector[0x10]=0x23; //T1 v R1
 //       Sfr

 //-----------------------------------------------
 for(k=0;k<0x100;k++)     //
                ADC[k]=0;
 uchar j=0;  // NOP
 //  j-   RAMM=(j<<3).000, j=ADC[IR]
 ADC[0]=j++; // NOP-->0
```

```
 ADC[1]=j++;
 ADC[02]=j++; //   ljmp  ad     j=2
 ADC[0x24]=j++; // add  a,#d   j=3
 ADC[0x22]=j++; // ret  j=4

 for(i=0x28;i<0x2f;i++) ADC[i]=j; j++;   //j=5   add  a,ri

for(uchar  i=0x11;i<0xF1;i=i+0x10) ADC[i]=j; j++;   // j=6   acall  met

ADC[0x82]=j++;   // j=7   anl  c,  bit
ADC[0x32]=j++; // reti    j=8;

// *ROUANTH

     // dec  @ri  :  j  =  9;
     for(uchar  i  =  0x16;  i  <=  0x17;  ++i)
          ADC[i]  =  j;
     ++j;

     // dec  a  :  j  =  10;
     ADC[0x14]  =  j++;

     // orl  c,  bit  :  j  =  11;
     ADC[0x72]  =  j++;

     // orl  c,  /bit  :  j  =  12;
     ADC[0xA0]  =  j++;

     // mov  a,  @rj  :  j  =  13;
     for(uchar  i  =  0xE6;  i  <=  0xE7;  ++i)
          ADC[i]  =  j;
     ++j;

     // mov  ri,  #d  :  j  =  14;
     for(uchar  i  =  0x78;  i  <=  0x7F;  ++i)
          ADC[i]  =  j;
     ++j;

     // mov  a,  ad  :  j  =  15;
     ADC[0xE5]  =  j++;

     // jb  bit,  rel  :  j  =  16;
     ADC[0x20]  =  j++;

     // mov  ad,  #d  :  j  =  17;
     ADC[0x75]  =  j++;

// ROUANTH*

//
//-----------------------------------------------
 for(k=0;k<128;k++)
 ROMM[k]=0;
//===============
 Ram[Sp]=SP=07;
 Ram[0]=0x11; // R0
 Ram[Acc]=ACC=0x82;
 Ram[Psw]=PSW=0x80;
 Ram[Tcon]=TCON=0;
 P0=P1=P2=P3=0xff; //
//
//-----------------------------------------------
 for(i=0;i<16;i++) DCM1[i]=DCM2[i]=DCM3[i]=DCM4[i]=\
  DCM7[i]=DCM8[i]=0;

// *ROUANTH
/*
// ROUANTH*
//-----------------------------------------------
 // -    ----- ---------

 //0:  ljmp  start     0x02  00   0x23      Pc=0x23    SP=07
 //03: nop             0x00
 //04: reti            0x32                PC=0x26    sp=07
 //13: nop             0x00
 //14: reti            0x32                Pc=0x28    sp=07

 //mcal:
 //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 mcal      0x11  0x22          PC=0x22, SP=0x09, Ram[sp]=00  2b
 //
```

```
 //2b: nop              0   //
 //=================================
//if(CheckBox8->State==cbUnchecked)
{   for(i=0;i<100;i++) CODE[i]=0;//
 PC=0;
CODE[PC++]=0x02; CODE[PC++]=0;
CODE[PC++]=0x23; //ljmp 23
CODE[0x03]= 0;
CODE[0x04]=0x32;   //reti   0
CODE[0x13]= 0;
CODE[0x14]=0x32;   //reti 1

PC=0x22;
CODE[PC++]=0x22;   //ret
CODE[PC++]=0x24; CODE[PC++]=0x80; //add a,#10
CODE[PC++]=0; //Nop
CODE[PC++]=0x28;   //add a,r0
CODE[PC++]=0x82; CODE[PC++]=0xe7; //anl ACC.7
CODE[PC++]=0x11; CODE[PC++]=0x22; //acall 0x22
CODE[PC++]=0;      //
 }

// *ROUANTH
*/

{   for(i=0;i<100;i++) CODE[i]=0;//
 PC=0;
// 00: LJMP 0x06
CODE[PC++] = 0x02; CODE[PC++] = 0x00; CODE[PC++] = 0x06;
// 03: RETI
CODE[PC++] = 0x32;
// 04: DEC A
CODE[PC++] = 0x14;
// 05: RET
CODE[PC++] = 0x22;
// 06: MOV 0x40, #0xDE
CODE[PC++] = 0x75; CODE[PC++] = 0x40; CODE[PC++] = 0xDE;
// 09: MOV 0x41, #0x3E
CODE[PC++] = 0x75; CODE[PC++] = 0x41; CODE[PC++] = 0x3E;
// 0C: MOV 0x20, #0x0F
CODE[PC++] = 0x75; CODE[PC++] = 0x20; CODE[PC++] = 0x0F;
// 0F: MOV R1, #0x41
CODE[PC++] = 0x79; CODE[PC++] = 0x41;
// 11: ANL C, 0x20.4
CODE[PC++] = 0x82; CODE[PC++] = 0x04;
// 13: DEC @R1
CODE[PC++] = 0x17;
// 14: MOV A, 0x40
CODE[PC++] = 0xE5; CODE[PC++] = 0x40;
// 16: MOV A, @R1
CODE[PC++] = 0xE7;
// 17: ORL C, 0x20.1
CODE[PC++] = 0x72; CODE[PC++] = 0x01;
// 19: ANL C, 0x20.4
CODE[PC++] = 0x82; CODE[PC++] = 0x04;
// 1B: ORL C,/0x20.4
CODE[PC++] = 0xA0; CODE[PC++] = 0x04;
// 1D: ACALL RET_TEST
CODE[PC++] = 0x11; CODE[PC++] = 0x04;
// 1F: JB 0x20.1, PAS
CODE[PC++] = 0x20; CODE[PC++] = 0x01; CODE[PC++] = 0x03;
// 22: LJMP ERR
CODE[PC++] = 0x02; CODE[PC++] = 0x00; CODE[PC++] = 0x22;
// 25: JB 0x20.4, ERR
CODE[PC++] = 0x20; CODE[PC++] = 0x04; CODE[PC++] = 0xFA;
// 28: LJMP SUC
CODE[PC++] = 0x02; CODE[PC++] = 0x00; CODE[PC++] = 0x28;

 }
// ROUANTH*

 Instr->Clear();
  IE=0;        //
  EX1=EX0=EA=0;   //
  //
  CheckBox1->State=cbUnchecked;     //
  CheckBox2->State=cbUnchecked;     //EX1=0
  CheckBox3->State=cbUnchecked;     //EX0=0
  CheckBox4->State=cbUnchecked;      //EA=0

   PC=0; //
   ComboBox2->Clear();     //
   ComboBox6->Clear();     //
  for(char i=0; i<8; i++)
                  Reg(i); //      !=0
  StateMCU(); //
  i1=1; //      DCMi
  //CheckBox8->State=cbUnchecked;
```

```
  CheckBox9->State=cbUnchecked;
}

//------   ---------------------------------------


//    -  -
void __fastcall TForm1::Button2Click(TObject *Sender)
{
     //0.0 mk
     GoToInt();   //      -
        MicroCodMem("Unicontr=Eintra, Unicontr=Wrtcon ");
     //
      if (!intra)  {RAMK=0x05; goto decoder;} //
         intra=0; RAMK++;

     //0.1 mk
      SP++;RAMK++;
     MicroCodMem("Unireg8=Cntsp,Unireg8=Incsp ");
     //0.2 mk
     Ram[SP++]=PC;RAMK++;
     MicroCodMem("SelbusB=Pcl,SelbusA=Sp,Unireg8=Cntsp,\
                                     Unireg8=Incsp,Unibus8=WramNsfr ");
     //0.3 mk
     Ram[SP]=PC>>8; PC=vect;   RAMK++;   //ClearInt();
     MicroCodMem("SelbusB=Pch,SelbusA=Sp,Unibus8=WramNsfr,Selpc=Vect");
     //0.4 mk       PC=vect
     Ram[Sp]=SP; RAMK=0;   Ram[Tcon]=TCON^Ram[Tcon]; P3^=TCON;
     MicroCodMem("SelbusB=Sp,SelbusA=Asfr,Adsfr=Sp,Unicontr=Clramk,Unibus8=Wram,\
             Unicontr=Wrtcon ");

     //-----------------------------------------
     decoder: //0.5 mk
        {IR=CODE[PC++];RAMK=(ADC[IR]<<3)+1;
        MicroCodMem("SelbusB=Code,Unicontr =Wrir,Unicod16=Incpc ");}
     switch(ADC[IR])           //
         { case 0:  //1.0   mk  Nop-
           {Instr->Text="Nop"; RAMK=0;
                   goto finish;}

           case 2: //ljmp adr
                 // 2.1      -       Wrk
                 { Wrk=CODE[PC++]; RAMK++;
           MicroCodMem("SelbusB=Code,Uniralu=Wrwrk,Unicod16=Incpc ");
                   }
                 // 2.2  -         PCL
                 {   PC=CODE[PC]|(Wrk<<8); RAMK=0;
           MicroCodMem("Selpc=Call,SelbusB=Code,Unicod16=Wrpc,Unicontr=Ramk1 ");
                    itoa(PC,&ss[0],16); //
                char stroka[10]="ljmp ";
                Instr->Text=StrCat(stroka,ss);
                   }
                   goto finish;
           case 4: //ret
                  // 4.1
                  {   Wrk=Ram[SP--]; RAMK++;       //PCH
           MicroCodMem("SelbusA=Sp,SelbusB=Ram,Uniralu=Wrwrk,Uniport=Cntsp ");
                     }
                  // 4.2
                  {   PC=Wrk|Ram[SP--]; RAMK++; //PCL
           MicroCodMem("SelbusA=Sp,SelbusB=Ram,Selpc=Call,\
                                            Unireg8=Cntsp,Unicod16=Wrpc ");
                     }
                  // 4.3
                  {   Ram[Sp]=SP; RAMK=0;
           MicroCodMem("SelbusB=Sp,Adsfr=Sp,Unibus8=WramNsfr,Unicontr=Ramk8 ");
                     Instr->Text="ret";
                     }
                     goto finish;
           case 5:        // add a,ri
           // 5.1
                 { PB= Ram[(PSW&0x18)|(IR&0x3)];RAMK++;
                   ss[0]=(IR&0x3)|0x30; ss[1]=0;
                                       //
                   char stroka[10]="add a,r";
                   Instr->Text=StrCat(stroka,ss);
           MicroCodMem("SelbusB=Ram,Unibus8=Ari,Uniralu=Wrpb ");
                   }
         // 5.2  -
                 {
                  ACC=ACC+PB, Ram[Acc]=ACC; RAMK++;
                  char tt[]="add  "; //
                  PSWC(&tt[0]); //
           MicroCodMem("SelbusB=F,Mop=Add,Selpsw=Bitsw,Unibit=Wlocpsw,\
                SelbusA=Asfr,Adsfr=Acc,Unibus8=Wram,Selpa=Acc,Selpb=Pb  ");
                   }
               // 5.3  - PsW  SFR
```

```
                {   Ram[Psw]=PSW;  RAMK=0;
         MicroCodMem("SelbusB=Psw,SelbusA=Asfr,Adsfr=Psw,\
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣Unibus8=Wram,Unicontr=Ramk1␣");
                  }
                      goto finish;
         case 3:         // add a,#d
              // 3.1 -        Code
         {PB=CODE[PC++]; RAMK++;
       MicroCodMem("SelbusB=Code,Uniralu=WrpbUnicod16=Incpc␣");
                itoa(PB,&ss[0],16); //
             char stroka[10]="add␣a,#";
             Instr->Text=StrCat(stroka,ss);
              }
              //3.2 - -
         {ACC=ACC+PB, RAMK++; Ram[Acc]=ACC;PSWC("add");
                char tt[5]="add␣␣";
           MicroCodMem("SelbusB=F,Mop=Add,Selpsw=Bitsw,Unibit=Wlocpsw,\
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣SelbusA=Asfr,Adsfr=Acc,Unibus8=Wram␣");
                }
              //3.3
              {Ram[Psw]=PSW;   RAMK=0;
           MicroCodMem("SelbusB=Psw,SelbusA=Asfr,Adsfr=Psw,Unibus8=Wram,\
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣Unicontr=Ramk1␣");
                  }
                    goto finish;
         case 6:            //   acall   met
             //6.1 -
                {Wrk=CODE[PC++]; SP++; RAMK++;
             MicroCodMem("SelbusB=Code,Uniralu=Wrwrk,Unicod16=Incpc,\
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣Uniport=Cntsp,Uniport=Incsp␣");
             itoa(Wrk,&ss[0],16);
             char stroka[10]="acall␣";
             Instr->Text=StrCat(stroka,ss);
                  }
              //6.2-      PC(7-0)
                {Ram[SP++]=PC;RAMK++;
             MicroCodMem("SelbusB=Pcl,SelbusA=Sp,Unibus8=WramNsfr,\
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣Uniport=Cntsp,Uniport=Incsp␣");
                  }
             //6.3         PC(15-8) --> , PC
                {Ram[SP]=PC>>8; RAMK++;
                 PC=((PC&0xf800)|Wrk)|((IR&0xE0)<<3);
         MicroCodMem("SelbusB=Pch,SelbusA=Sp,Unibus8=WramNsfr,Selpc=Acall␣");
                   }
       //6.4       SFR
               {Ram[Sp]=SP;   RAMK=0;
        MicroCodMem("SelbusB=Sp,SelbusA=Asfr,Adsfr=Sp,Unibus8=Wram,\
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣Unicontr=Ramk8␣");
                }
                goto finish ;
         case 7: // anl c,bit
             // 7.1-
                {Wrk=CODE[PC++]; RAMK++;
         MicroCodMem("SelbusB=Code,Uniralu=Wrwrk,Unicod16=Incpc␣");
                ss[0]=(Wrk&0x7)+0x30; ss[1]=0; //
                char stroka[12]="anl␣c,Acc.";
                Instr->Text=StrCat(stroka,ss);
                   }
              // 7.2 -     SFR,         SFR
                {
                if(Wrk&0x80)PB=Ram[Wrk&0xf8];
                else   PB=Ram[0x20|((Wrk&0x78)>>3)]; RAMK++;
         MicroCodMem("Selif=Wrk7,SelbusA=Abitwrk,SelbusB=Ram,Uniralu=Wrpb␣");
                   }
              // 7.3 -        PSW     PB
              //  PSW  SFR    Psw
                { char tt[]="andc␣"; PSWC(&tt[0]);
         MicroCodMem("Unibit=Bitand,Unibit=Wlocpsw,Selpsw=Bitsw␣");
                }
              // 7.4 -
                   Ram[Psw]=PSW;   RAMK=0;
            MicroCodMem("SelbusB=Psw,SelbusA=Asfr,Adsfr=Psw,\
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣Unibus8=WramNsfr,Unicontr=Ramk1␣");
                    goto finish;

             case 8: //reti
                // 8.1
                {   Wrk=Ram[SP--]<<8; RAMK++; intra2=0; //
         MicroCodMem("SelbusA=Sp,SelbusB=Ram,Uniralu=Wrwrk,Unireg8=Cntsp␣");
                   }
                // 8.2
                {   PC=Wrk|Ram[SP--]; RAMK++; //PCH.PCL
         MicroCodMem("SelbusA=Sp,SelbusB=Ram,Selpc=Call,Unireg8=Cntsp␣");
                   }
                // 8.3
                {   Ram[Sp]=SP;   RAMK=0; intra2=0;
         MicroCodMem("SelbusB=Sp,Adsfr=Sp,Unibus8=WramNsfr,\
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣Unicontr=Ramk1,Unicontr=Clrinta␣");
```

```
                    Instr->Text="reti";
                    CheckBox9->State=cbUnchecked;

// *ROUANTH
                    goto finish;
                }
            case 9: // dec @ri
            {
                { // 9.1. Read Ri (can only be R0 or R1)
                    Wrk = Ram[(PSW&0x18)|(IR&0x1)];
                    ss[0]=(IR&0x1)|0x30;
                    ss[1]='\0';
                    char code[10]="dec @r";
                    Instr->Text=StrCat(code,ss);
                }
                { // 9.2. Read @Ri
                    PB = Ram[Wrk] - 1;
                }
                { // 9.3. Write [@Ri - 1] to @Ri
                    Ram[Wrk] = PB;
                }
                goto finish;
            }
            case 10: // dec a
            {
                { // 10.1. Modifying A
                    Wrk = Ram[Acc] - 1;
                }
                { // 10.2. Writing A
                    Ram[Acc] = ACC = Wrk;
                    char code[12]="dec a";
                    ss[0] = '\0';
                    Instr->Text=StrCat(code,ss);
                }
                { // 10.3. Updating the status register
                    ss[0] = 'a'; ss[1] = 'b'; ss[2] = '\0';
                    PSWC(ss);
                    Ram[Psw]=PSW;
                }
                goto finish;
            }
            case 11: // orl c, bit
            {
                { // Reading the bit address
                    Wrk=CODE[PC++];
                    RAMK++;
                    MicroCodMem("SelbusB=Code,"
                        "Uniralu=Wrwrk,"
                        "Unicod16=Incpc ");
                    ss[0]=(Wrk&0x80)|0x30;
                    ss[1]='\0';
                    char code[12]="orl c,Acc."; // actually a lie
                    Instr->Text=StrCat(code,ss);
                }
                { // 7.2. Reading from SFR or RAM
                    if (Wrk&0x80)
                        PB=Ram[Wrk&0xf8];
                    else
                        PB=Ram[0x20|((Wrk&0x78)>>3)];
                    RAMK++;
                    MicroCodMem("Selif=Wrk7,"
                        "SelbusA=Abitwrk,"
                        "SelbusB=Ram,"
                        "Uniralu=Wrpb ");
                }
                { // 7.3.
                    PSWC("orlc ");
                    RAMK++;
                    MicroCodMem("Unibit=Bitand,"
                    "Unibit=Wlocpsw,"
                    "Selpsw=Bitsw ");
                }
                {
                    Ram[Psw]=PSW;
                    RAMK=0;
                    MicroCodMem("SelbusB=Psw,"
                        "SelbusA=Asfr,"
                        "Adsfr=Psw,"
                        "Unibus8=WramNsfr,"
                        "Unicontr=Ramk1 ");
                }
                goto finish;
            }
            case 12: // orl c, /bit
            {
                { // Reading the bit address
                    Wrk=CODE[PC++];
                    ss[0]=(Wrk&0x7)|0x30;
```

```
            ss[1]='\0';
            char code[12]="orl c,/Acc."; // actually a lie
            Instr->Text=StrCat(code,ss);
            RAMK++;
            MicroCodMem("SelbusB=Code,"
                 "Uniralu=Wrwrk,"
                 "Unicod16=Incpc ");
        }
        { // 7.2. Reading from SFR or RAM
            if (Wrk&0x80)
                 PB=Ram[Wrk&0xf8];
            else
                 PB=Ram[0x20|((Wrk&0x78)>>3)];
            PB = ~PB;
            RAMK++;
            MicroCodMem("Selif=Wrk7,"
                 "SelbusA=Abitwrk,"
                 "SelbusB=Ram,"
                 "Uniralu=Wrpb ");
        }
        { // 7.3.
            PSWC("orlc ");
            RAMK++;
            MicroCodMem("Unibit=Bitand,"
            "Unibit=Wlocpsw,"
            "Selpsw=Bitsw ");
        }
        {
            Ram[Psw]=PSW;
            RAMK=0;
            MicroCodMem("SelbusB=Psw,"
                 "SelbusA=Asfr,"
                 "Adsfr=Psw,"
                 "Unibus8=WramNsfr,"
                 "Unicontr=Ramkl ");
        }
        goto finish;
    }
    case 13: // mov a, @rj
    {
        {
            Wrk = Ram[(PSW&0x18)|(IR&0x1)];
            RAMK++;
            ss[0]=(IR&0x1)|0x30;
            ss[1]=0;
            char code[10]="mov a,@r";
            Instr->Text=StrCat(code,ss);
            MicroCodMem("SelbusB=Ram,"
                 "Unibus8=Ari,"
                 "Uniralu=Wrpb ");
        }
        {
            Ram[Acc] = ACC = Ram[Wrk];
            RAMK++;
            PSWC(ss); //
            MicroCodMem("SelbusB=F,"
                 "Mop=Add,"
                 "Selpsw=Bitsw,"
                 "Unibit=Wlocpsw,"
                 "SelbusA=Asfr,"
                 "Adsfr=Acc,"
                 "Unibus8=Wram,"
                 "Selpa=Acc,"
                 "Selpb=Pb  ");
        }
        {
            Ram[Psw]=PSW;
            RAMK=0;
            MicroCodMem("SelbusB=Psw,"
                 "SelbusA=Asfr,"
                 "Adsfr=Psw,"
                 "Unibus8=Wram,"
                 "Unicontr=Ramkl ");
          }

        goto finish;
    }
    case 14: // mov ri, #d
    {
        { // 14.1. Read Ri (can only be R0 or R1)
            Ram[(PSW&0x18)|(IR&0x3)] = CODE[PC++];
            ss[0]=(IR&0x3)|0x30;
            ss[1]='\0';
            char code[10]="mov @r";
            Instr->Text=StrCat(code,ss);
        }
        goto finish;
    }
```

```
case 15: // mov a, ad
{
    {
        Wrk = CODE[PC++];
        RAMK++;
        itoa(Wrk,ss,16);
        char code[10]="mov a,";
        Instr->Text=StrCat(code,ss);
        MicroCodMem("SelbusB=Ram,"
            "Unibus8=Ari,"
            "Uniralu=Wrpb,");
    }
    {
        ACC = Ram[Wrk];
        RAMK++;
        PSWC(ss); //
        MicroCodMem("SelbusB=F,"
            "Mop=Add,"
            "Selpsw=Bitsw,"
            "Unibit=Wlocpsw,"
            "SelbusA=Asfr,"
            "Adsfr=Acc,"
            "Unibus8=Wram,"
            "Selpa=Acc,"
            "Selpb=Pb,,");
    }
    {
        Ram[Acc] = ACC;
    }
    {
        Ram[Psw]=PSW;
        RAMK=0;
        MicroCodMem("SelbusB=Psw,"
            "SelbusA=Asfr,"
            "Adsfr=Psw,"
            "Unibus8=Wram,"
            "Unicontr=Ramk1,");
    }

    goto finish;
}
case 16: // jb bit, rel
{
    { // Reading the bit address
        Wrk = CODE[PC++];
    }
    {
        PA  = CODE[PC++];
        RAMK++;
        MicroCodMem("SelbusB=Code,"
            "Uniralu=Wrwrk,"
            "Unicod16=Incpc,");
        ss[0]=(Wrk&0x7)|0x30;
        ss[1]='\0';
        char code[12]="jb Acc."; // actually a lie
        Instr->Text=StrCat(code,ss);
    }
    { // 7.2. Reading from SFR or RAM
        if (Wrk&0x80)
            PB=Ram[Wrk&0xf8];
        else
            PB=Ram[0x20|((Wrk&0x78)>>3)];
        RAMK++;
        MicroCodMem("Selif=Wrk7,"
            "SelbusA=Abitwrk,"
            "SelbusB=Ram,"
            "Uniralu=Wrpb,");
    }
    {
        if (PB & (1 << (Wrk & 0x7)))
            PC += PA;
    }
    goto finish;
}
case 17: // mov ad, #d
{
    {
        Wrk=CODE[PC++];
        RAMK++;
        MicroCodMem("SelbusB=Code,"
            "Uniralu=Wrpb,"
            "Unicod16=Incpc,");
        itoa(Wrk,&ss[0],16);
        char code[10]="mov ad,#";
        Instr->Text=StrCat(code,ss);
    }
    {
        Ram[Wrk] = CODE[PC++];
```

```
                        RAMK++;
                        MicroCodMem("SelbusB=F,"
                              "Mop=Add,"
                              "Selpsw=Bitsw,"
                              "Unibit=Wlocpsw,"
                              "SelbusA=Asfr,"
                              "Adsfr=Acc,"
                              "Unibus8=Wram␣");
                    }
                    goto finish;
// ROUANTH*
                        }

             //     HEX-
     finish: //     RAMK=-
                  StateMCU();
                        //---------------------------------------
          //    Nop -
     if (CheckBox7->State==cbChecked)
               {   for(char i=8; i<(SP+5); i++) Stack(i);
                     //    RET,RETI
               Instr->Text="␣"; PC--;
                  CheckBox7->State==cbUnchecked;
                                  }

   }  //switch


}


//---------------------------------------------------------------------------

char Chi;
void __fastcall TForm1::CheckBox1Click(TObject *Sender)
{     if(Chi==0)
        {CheckBox1->State=cbChecked; Chi=1;}
       else  {CheckBox1->State=cbUnchecked; Chi=0;}

}
//---------------------------------------------------------------------------

void Creatwrsfr(void)    //  SFR  wrsfr
{    WRSFR[Sp&0x7f]= 2;
     WRSFR[Acc&0x7f]= 1;
     WRSFR[Dph&0x7f]= 3;
     WRSFR[Dpl&0x7f]= 4;
     WRSFR[b&0x7f]=   3;
     WRSFR[Psw&0x7f]=  4;
     WRSFR[p3&0x7f]=    10;

     WRSFR[p0&0x7f]=  7;
     WRSFR[p1&0x7f]=  5;
     WRSFR[p2&0x7f]=  9;
     WRSFR[Ie&0x7f]=  13;
     WRSFR[Ip&0x7f]=  12;
     WRSFR[Tcon&0x7f]= 3;
}

uchar VECTOR[16];
void Creatvector(void)    //
{
     for(uchar i=0;i<16;i++)
                 VECTOR[i]=0;
     VECTOR[01]=0x03;
     VECTOR[02]=0x0B;
     VECTOR[04]=0x13;
     VECTOR[0x8]=0x1B;
}

void ToFile(AnsiString S, char *ss )
{        //  .mfi    maxplus
         char *std="␣␣␣␣␣";
         char *std1="␣␣␣␣␣";
         char *std2="␣␣␣␣␣";
         char *std3="␣␣␣␣␣";
         std2=std; std3=std1;
        FILE *tmpFILE=fopen( S.c_str(),"w");
        if(tmpFILE!=NULL)
                {
                 if(strcmp(ss,"CODE")==0)
                 {fprintf(tmpFILE,"␣%s␣\n", "DEPTH␣=␣2048;");
                 fprintf(tmpFILE,"␣%s␣\n", "WIDTH␣=␣8;");
                   goto mms;}
                 if(strcmp(ss,"Xdata")==0)
                 {fprintf(tmpFILE,"␣%s␣\n", "DEPTH␣=␣2048;");
                 fprintf(tmpFILE,"␣%s␣\n", "WIDTH␣=␣8;");
                 goto mms;}
```

```
        if(strcmp(ss,"ADC")==0)
        {fprintf(tmpFILE,"␣%s␣\n", "DEPTH␣=␣256;");
        fprintf(tmpFILE,"␣%s␣\n", "WIDTH␣=␣7;");
        goto mms;}
        if(strcmp(ss,"ROMM")==0)
        {fprintf(tmpFILE,"␣%s␣\n", "DEPTH␣=␣128;");
        fprintf(tmpFILE,"␣%s␣\n", "WIDTH␣=␣10;");
        goto mms;}
        if(strcmp(ss,"WRSFR")==0)
        {fprintf(tmpFILE,"␣%s␣\n", "DEPTH␣=␣128;");
        fprintf(tmpFILE,"␣%s␣\n", "WIDTH␣=␣4;");
        goto mms;}
        if(strcmp(ss,"VECTOR")==0)
        {fprintf(tmpFILE,"␣%s␣\n", "DEPTH␣=␣16;");
        fprintf(tmpFILE,"␣%s␣\n", "WIDTH␣=␣6;");
        goto mms;}

        fprintf(tmpFILE,"␣%s␣\n", "DEPTH␣=␣64;");
        if(strcmp(ss,"DCM1")==0)
        fprintf(tmpFILE,"␣%s␣\n", "WIDTH␣=␣7;");
        else
        if(strcmp(ss,"DCM2")==0)
        fprintf(tmpFILE,"␣%s␣\n", "WIDTH␣=␣6;");
        else
        if(strcmp(ss,"DCM3")==0)
        fprintf(tmpFILE,"␣%s␣\n", "WIDTH␣=␣7;");
        else
        if(strcmp(ss,"DCM4")==0)
        fprintf(tmpFILE,"␣%s␣\n", "WIDTH␣=␣19;");
        else
        if(strcmp(ss,"DCM6")==0)
        fprintf(tmpFILE,"␣%s␣\n", "WIDTH␣=␣5;");
        else
        if(strcmp(ss,"DCM7")==0)
        fprintf(tmpFILE,"␣%s␣\n", "WIDTH␣=␣17;");
        else
        if(strcmp(ss,"DCM8")==0)
        fprintf(tmpFILE,"␣%s␣\n", "WIDTH␣=␣3;");
    mms:
        fprintf(tmpFILE,"␣%s␣\n", "ADDRESS_RADIX␣=␣HEX;");
        fprintf(tmpFILE,"␣%s␣\n", "DATA_RADIX␣=␣HEX;");
        fprintf(tmpFILE,"␣%s␣\n", "CONTENT");
        fprintf(tmpFILE,"␣%s␣\n", "BEGIN");
        }

if(strcmp(ss,"ADC")==0)
 { for(uint i=0; i<256; i++)
   fprintf(tmpFILE,"␣%s␣%s␣%s␣␣\n", itoa(i,std,16),"␣␣:␣␣", \
                                        itoa(ADC[i],std1,16), "␣;");
   goto mms1;
  }
  if(strcmp(ss,"CODE")==0)
  {for(uint i=0; i<2048; i++)
   fprintf(tmpFILE,"␣%s␣%s␣%s␣␣\n", itoa(i,std,16),"␣␣:␣␣", itoa(CODE[i],std1,16), "␣;");
   goto mms1;}
  if(strcmp(ss,"VECTOR")==0)
    {for(uchar i=0; i<16; i++)
      fprintf(tmpFILE,"␣%s␣%s␣%s␣␣\n", itoa(i,std,16),"␣␣:␣␣", \
                                        itoa(VECTOR[i],std1,16), "␣;");
     goto mms1;}

   if(strcmp(ss,"ROMM")==0)
   {for(uint i=0; i<128; i++)
   fprintf(tmpFILE,"␣%s␣%s␣%s␣␣\n", itoa(i,std,16),"␣␣:␣␣", \
                                       itoa(ROMM[i],std1,16), "␣;");
     goto mms1;}
   if(strcmp(ss,"WRSFR")==0)
   {for(uint i=0; i<128; i++)
  fprintf(tmpFILE,"␣%s␣%s␣%s␣␣\n", itoa(i,std,16),"␣␣:␣␣", \
                                        itoa(WRSFR[i],std1,16), "␣;");
     goto mms1;}
 for(uchar i=0; i<64; i++)
   {
   if(strcmp(ss,"DCM1")==0)
   fprintf(tmpFILE,"␣%s␣%s␣%s␣␣\n", itoa(i,std,16),"␣␣:␣␣", \
                                       itoa(DCM1[i],std1,16), "␣;");
   else
   if(strcmp(ss,"DCM2")==0)
   fprintf(tmpFILE,"␣%s␣%s␣%s␣␣\n", itoa(i,std,16),"␣␣:␣␣", \
                                        itoa(DCM2[i],std1,16), "␣;");
   else
   if(strcmp(ss,"DCM3")==0)
   fprintf(tmpFILE,"␣%s␣%s␣%s␣␣\n", itoa(i,std,16),"␣␣:␣␣", \
                                       itoa(DCM3[i],std1,16), "␣;");
   else
   if(strcmp(ss,"DCM4")==0)
   fprintf(tmpFILE,"␣%s␣%s␣%s␣␣\n", itoa(i,std,16),"␣␣:␣␣", \
                                       itoa(DCM4[i],std1,16), "␣;");
```

```
            else
             if(strcmp(ss,"DCM7")==0)
               fprintf(tmpFILE,"␣%s␣%s␣%s␣␣\n", itoa(i,std,16),"␣␣:␣␣", \
                                                 itoa(DCM7[i],std1,16), "␣;");
            else
             if(strcmp(ss,"DCM8")==0)
               fprintf(tmpFILE,"␣%s␣%s␣%s␣%s␣␣\n", itoa(i,std,16),"␣␣:␣␣", \
                                                 itoa(DCM8[i],std1,16), "␣;");

             else break;
               std=std2; std1=std3;
              }

          mms1:
            fprintf(tmpFILE,"␣%s␣\n", "END;");
            fclose(tmpFILE);
}



void __fastcall TForm1::Button7Click(TObject *Sender)
{

          ToFile("DCM1.mif","DCM1");   //16mem
          ToFile("DCM2.mif","DCM2");   //port
          ToFile("DCM3.mif","DCM3");   //bit
          ToFile("DCM4.mif","DCM4");   //ralu
          ToFile("DCM6.mif","DCM6");   //intrupt
          ToFile("DCM7.mif","DCM7");   //bus8
          ToFile("DCM8.mif","DCM8");   //control
          ToFile("ADC.mif","ADC");     //-->
          ToFile("ROMM.mif","ROMM");   //
          ToFile("CODE.mif","CODE");   //
          Creatwrsfr();
          ToFile("WR8.mif","WR8");     // SFR
          Creatvector();
          ToFile("VECTOR.mif","VECTOR"); //vect
          Edit4->Text="␣";
}
//---------------------------------------------------------------------------


void __fastcall TForm1::Button3Click(TObject *Sender)
{     //   H/L INT1   IT1=1
            IE1=1;
           TCON=TCON|0x8; Ram[Tcon]=TCON; CheckBox6->State=cbChecked;P3^=0x8;

}
//---------------------------------------------------------------------------



void __fastcall TForm1::CheckBox2Click(TObject *Sender)
{       //    INT1

     EX1^=1;
     if (EX1)
        {IE=IE|0x4;//     IE
             CheckBox2->State= cbChecked;}
     else
        { IE=IE&0xfb; //
             CheckBox2->State= cbUnchecked;}
         Ram[Ie]=IE;
}
//---------------------------------------------------------------------------

void __fastcall TForm1::CheckBox3Click(TObject *Sender)
{       //    INT0

      EX0^=1;
     if (EX0)
        {IE=IE|0x1; //     IE
            CheckBox3->State= cbChecked;}
     else
        { IE=IE&0xfe; //     IE
            CheckBox3->State= cbUnchecked;}
         Ram[Ie]=IE;
}
//---------------------------------------------------------------------------

void __fastcall TForm1::CheckBox4Click(TObject *Sender)
{       //

       EA^=1;
     if (EA)
        {IE=IE|0x80; //     IE
             CheckBox4->State= cbChecked;}
```

```
      else
         { IE=IE&0x7f; //     IE
              CheckBox4->State= cbUnchecked;}
         Ram[Ie]=IE;

}
//---------------------------------------------------------------------------



void __fastcall TForm1::Edit7DblClick(TObject *Sender)
{    char s;
     char ss[3];
     int L;
     AnsiString Q=Edit7->Text;
     if(Q[1]=='@') {s=Q[3];
                    Edit7->Text=itoa(Ram[Ram[s&1]],ss,16);
                    }
     else   Edit7->Text=itoa(Ram[StrToInt(Q.c_str())],ss,16);
}
//---------------------------------------------------------------------------



void __fastcall TForm1::Edit6DblClick(TObject *Sender)
{
       uint numb=0xffff;
      AnsiString S=Edit6->Text;
      char *ss;
      ss=S.c_str();
      if(strcmp(ss, "ACC") == 0) numb=ACC;
      if(strcmp(ss, "SP") == 0)  numb=SP;
      if(strcmp(ss, "DPH") == 0) numb=DPH;
      if(strcmp(ss, "DPL") == 0) numb=DPL;
      if(strcmp(ss, "PSW") == 0) numb=PSW;
      if(strcmp(ss, "P3") == 0) numb=P3;
      if(strcmp(ss, "P0") == 0) numb=P0;
      if(strcmp(ss, "P1") == 0) numb=P1;
      if(strcmp(ss, "P2") == 0) numb=P2;
      if(strcmp(ss, "TCON") == 0) numb=TCON;
      if(strcmp(ss, "IE") == 0) numb=IE;
      if(strcmp(ss, "IP") == 0) numb=IP;
     if(numb!=0xffff)
       Edit6->Text=itoa(numb,ss,16);
     else
       Edit6->Text="";
}
//---------------------------------------------------------------------------



void __fastcall TForm1::Edit9DblClick(TObject *Sender)
{
char s;
     char ss[3];
     int L;
     AnsiString Q=Edit9->Text;
     if(Q[1]=='@') {s=Q[3];
                    Edit9->Text=itoa(Xdata[(P2<<8)+Ram[s&1]],ss,16);
                    }
     else   Edit9->Text=itoa(Xdata[StrToInt(Q.c_str())],ss,16);
}
//---------------------------------------------------------------------------









//---------------------------------------------------------------------------
```

```
void __fastcall TForm1::CheckBox6Click(TObject *Sender)
{
   /* IE0^=1;
     if (~IE0)
        {TCON=TCON|2; CheckBox5->State= cbChecked;
             }
      else
        { TCON=TCON&0xfd;
                       CheckBox5->State= cbUnchecked;
                       }
            Ram[Tcon]=TCON;
         */
}
//---------------------------------------------------------------------------




void __fastcall TForm1::Button10Click(TObject *Sender)
{
   //   H/L INT0  IT0=1
      IE0=1;
           TCON=TCON|0x2; Ram[Tcon]=TCON; CheckBox5->State=cbChecked; P3^=0x4;
}
//---------------------------------------------------------------------------







void __fastcall TForm1::Button13Click(TObject *Sender)
{      AnsiString S;
       char j=0; char numb=0; int adr=0; int y; char x;
       if (OpenDialog1->Execute())
         { S=OpenDialog1->FileName;  //S=

         hdisk=CreateFile(S.c_str(),
         GENERIC_READ|GENERIC_WRITE,
         FILE_SHARE_READ|FILE_SHARE_WRITE,NULL,
         OPEN_ALWAYS,
         FILE_ATTRIBUTE_NORMAL,NULL);
           }
        if (hdisk)  //
     {  j=0;
     do
      {  numb=0; adr=0;
        do
         { ReadFile(hdisk,&x,1,&bufd,0);
            x= (x<0x3a)? x&0xf: x-'A'+10;
            if((j>=1)&&(j<3)) numb=numb*16+x;
            if((j>=3)&&(j<7))
             {if(numb==0) goto exit;
               adr=adr*16+x;}
            if(j>=9)
             if(j&1) y= x;
             else {y=y*16+x;
                  CODE[adr++]=y;
                  numb--;
              if(numb==0)
                        goto brk;
                 }
             j++;
          }
         while(1);
       brk:
       do {
            ReadFile(hdisk,&x,1,&bufd,0);
           }
          while(x!=':');
          j=1;
         }
        while(1);
       exit:
         Edit3->Text="File open";
       }
       else  Edit3->Text="No open";
       CloseHandle(hdisk);
       CheckBox8->State=cbChecked;
}
```

```
//---------------------------------------------------------------------------
```

```
void __fastcall TForm1::Edit2DblClick(TObject *Sender)
{
            char i;
            int ss;
            char qq[18];
          AnsiString S;
          AnsiString Q=Edit2->Text;
          ss=atoi(Q.c_str());
          itoa(ss,&qq[0],2);
          qq[17]=0;
          for(i=0;i<8;i++)
                {qq[16-i]=qq[7-i]; //qq[7-i]=' ';
                                    }
        Edit2->Text=&qq[0];

}
//---------------------------------------------------------------------------
```