```
//---------------------------------------------------------------------------
#include <vcl\vcl.h>
#pragma hdrstop
#include "pc20--1.h"
//---------------------------------------------------------------------------
#pragma resource "*.dfm"
TForm1 *Form1;

//---------------------------------------------------------------------------
  fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
}
//---------------------------------------------------------------------------
class Tabel
{
protected:
int aantal,pos[1024],waarde[1024];
public:
Tabel()
  {
  aantal=0;
  }
//*****************************add
public:
virtual add(int adres)
  {
  if (find(adres)<0 )    waarde[aantal++]=adres;
  };

//*****************************Find
int find(int adres)
  {
  for (int n=0;n<aantal;n++)
    {
    if (adres==waarde[n] ){return(n);};
    }
    return(-1);

  };

public:
int get(int nr)
  {
  if (nr>=aantal)  return  (-1);
  return (waarde[pos[nr]]);
  }

public:
void Sort()
{
  int min;
  for (int n=0;n<aantal;n++)
  {
    min=0;
    for (int m=0;m<aantal;m++)
    {
      if (waarde[m]<waarde[n]) min++;
    }
    pos[min]=n;
  }
}

};

class SubTabel: public Tabel
{
public:
//*****************************Show
void Show()
```

```
        PrintFooter(recta,lineH);
        Printer()->EndDoc();

        }

}
void TForm1::PrintFooter(TRect& recta,int lineH)
{
char buff[10];
sprintf(buff,"Page %d",Printer()->PageNumber);
recta.Top=Printer()->PageHeight-(lineH*2);
recta.Bottom=recta.Top+lineH;

DrawText(Printer()->Handle,buff,-1,(RECT*)&recta,DT_CENTER);

Printer()->Canvas->MoveTo(0,recta.Top-2);
Printer()->Canvas->LineTo(recta.Right,recta.Top-2);

}
//---------------------------------------------------------------------------
void   fastcall TForm1::ClearClick(TObject *Sender)
{
MemoEpromA->Clear();
MemoEpromB->Clear();
MemoResult->Clear();
}
//---------------------------------------------------------------------------
void   fastcall TForm1::OpenEpromAClick(TObject *Sender)
{
MemoEpromA->WordWrap=true;
Label1->Caption = "Loading EpromA";
if (OpenDialog1->Execute() )
    {
    MemoEpromA->Lines->LoadFromFile(OpenDialog1->FileName);
    }
Label1->Caption = "Idle";
}
//---------------------------------------------------------------------------
void   fastcall TForm1::OpenEpromBClick(TObject *Sender)
{
MemoEpromB->WordWrap=true;
Label1->Caption = "Loading EpromB";
if (OpenDialog1->Execute() )
    {
    MemoEpromB->Lines->LoadFromFile(OpenDialog1->FileName);
    }
Label1->Caption = "Idle";
}

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

void   fastcall TForm1::SaveToFileClick(TObject *Sender)
{
if (SaveDialog1->Execute() ) MemoResult->Lines->SaveToFile(SaveDialog1->FileName)
}
//---------------------------------------------------------------------------
void   fastcall TForm1::disasClick(TObject *Sender)
{
int ih,il,s,line,word,wordc,bit,sma,data,adres,opcode;
char *ah;
char *al,*sopc;
char bh[120],bl[120],c[120],sop[120];
int nr,type,v1,v2;
char naam[100];
Label1->Caption = "Decoding Eproms";

for (  s=0;s<(MemoEpromA->Lines->Count)-1;s++)
{
ah=MemoEpromA->Lines->Strings[s].c_str();
strcpy(bh,ah+9);
```

```
        al=MemoEpromB->Lines->Strings[s].c_str();
        strcpy(bl,al+9);

        for (int m=0 ;m<32;m+=2)
            {   line=s*16+(m)/2;
                sscanf(bh+m ,"%2x",&ih); sscanf(bl+m ,"%2x",&il);
                word=(ih*256)+il;
                adres=word &2047;wordc=word;
                data=adres;data=data>>7;
                sma=wordc &511;wordc=wordc>>9;
                bit=wordc &3;  wordc=wordc>>2;
                opcode= wordc &31;
                sopc=Controller->Lines->Strings[opcode].c_str();

                int slen= Controller->Lines->Strings[opcode].Length();
                strncpy(sop,sopc,slen);

                sscanf(sop,"%d %d %s %d %d",&nr,&type,&naam,&v1,&v2);

                switch (type)
                {
                case 0:sprintf(c,"%04d %04X %-8s ",
                line,word,naam);break;
                case 1:sprintf(c,"%04d %04X %-8s %03d.%1d",
                       line,word,naam,sma,bit);


                break;
                case 2:sprintf(c,"%04d %04X %-8s %04d ",
                       line,word,naam,sma+bit*1000);


                break;
                case 3:sprintf(c,"%04d %04X %-8s %d",
                    line,word,naam,data);break;
                case 4:sprintf(c,"%04d %04X %-8s %04d  to %04d",
                    line,word,naam,adres,line-adres);
                    Jumplijst.add(line-adres);
                    break;
                case 5:sprintf(c,"%04d %04X %-8s %04d  to %04d",
                    line,word,naam,adres,line+adres);
                     Jumplijst.add(line+adres);
                    break;
                case 6:sprintf(c,"%04d %04X %-8s %04d",
                    line,word,naam,adres);
                    Sublijst.add(adres);
                    break;
                default:sprintf(c,"%04d %04X ERROR TYPE %s %d %d",
                    line,word,naam,adres,sma);
                }


                MemoResult->Lines->Add(c);

            }
        }
         Label1->Caption = "Sorting Jumps - Subs";
         Sublijst.Sort();
         Jumplijst.Sort();

         Sublijst.Show();
         Jumplijst.Show();
         Label1->Caption = "Idle";
        }

        //---------------------------------------------------------------------------
        void   fastcall TForm1::MakePclTextClick(TObject *Sender)
        {
        int ih,il,s,line,word,wordc,bit,sma,data,adres,opcode,label;
        char *ah;
```

```
char *al,*sopc;
char bh[120],bl[120],c[120],d[120],sop[120];
int nr,type,v1,v2;
int subpo,jumppo;
char naam[100];
MemoResult->Clear();
subpo=0;jumppo=0;
Label1->Caption = "Making PCL text";
for (  s=0;s<(MemoEpromA->Lines->Count)-1;s++)
  {
  ah=MemoEpromA->Lines->Strings[s].c_str();
  strcpy(bh,ah+9);
  al=MemoEpromB->Lines->Strings[s].c_str();
  strcpy(bl,al+9);

  for (int m=0 ;m<32;m+=2)
    {   line=s*16+(m)/2;
        sscanf(bh+m ,"%2x",&ih); sscanf(bl+m ,"%2x",&il);
        word=(ih*256)+il;
        adres=word &2047;wordc=word;
        data=adres;data=data>>7;
        sma=wordc &511;wordc=wordc>>9;
        bit=wordc &3;  wordc=wordc>>2;
        opcode= wordc &31;
        sopc=Controller->Lines->Strings[opcode].c_str();

        int slen= Controller->Lines->Strings[opcode].Length();
        strncpy(sop,sopc,slen);

        sscanf(sop,"%d %d %s %d %d",&nr,&type,&naam,&v1,&v2);
        strcpy (d,"    ");
        if (line==Sublijst.get(subpo))
          {
          sprintf(d,"S%03d",subpo++);
          }
         if (line==Jumplijst.get(jumppo))
          {
          sprintf(d,"K%03d",jumppo++);
          }
        switch (type)
        {
        case 0:sprintf(c,"%s  %-8s ",d,naam);break;
        case 1:sprintf(c,"%s  %-8s %03d.%1d",d,naam,sma,bit);break;
        case 2:sprintf(c,"%s  %-8s %04d",d,naam,sma+bit*1000);break;
        case 3:sprintf(c,"%s  %-8s %d",d,naam,data);break;
        //  4 relatief achteruit ,5 relatief vooruit,6 absoluut
        case 4: label=Jumplijst.find(line-adres);
                sprintf(c,"%s  JUMP    K%03d",d,label);break;
        case 5: label=Jumplijst.find(line+adres);
                sprintf(c,"%s  JUMP    K%03d",d,label); break;
        case 6: label=Sublijst.find(adres);
                sprintf(c,"%s  JSAT    S%03d",d,label); break;

        default:sprintf(c," %04X ERROR TYPE %s %d %d",
            line,word,naam,adres,sma);
        }
      MemoResult->Lines->Add(c);
    }
  }
Label1->Caption = "Idle";
}


//---------------------------------------------------------------------------
void   fastcall TForm1::FormCreate(TObject *Sender)
{
Label1->Caption = "Idle";
}
//---------------------------------------------------------------------------
```

0	0	NOP	0	0
1	1	TRIG	511	3
2	1	EQL	511	3
3	1	EQLNT	511	3
4	2	SHFTL	511	3
5	2	SHFTR	511	3
6	2	CNTD	2047	0
7	2	CNTU	2047	0
8	1	SET0	511	3
9	1	SET1	511	3
10	1	STRB	511	3
11	1	FTCHB	511	3
12	3	FTCHC	15	0
13	2	FTCHD	2047	0
14	2	STRD	2047	0
15	2	COMP	2047	0
16	1	AND	511	3
17	1	ANDNT	511	3
18	1	OR	511	3
19	1	ORNT	511	3
20	2	ADD	2047	0
21	2	SUBTR	2047	0
22	2	MULT	2047	0
23	2	DIV	2047	0
24	6	JSAF	2047	0
25	6	JSAT	2047	0
26	0	RET	2047	0
27	2	END	2047	0
28	0	NO_use	0	0
29	4	JBRF	2047	0
30	5	JFRF	2047	0
31	2	LSTIO	2047	0

type			
0 no data			
1 bit			max 511.3
2 sma +bit*1000 (page)			
3 constant		max 15	
4 relative backwards	JBRF	max 2047	
5 relative forwards	JFRF	max 2047	
6 absolute	JSAT	max 2047	