Klausur IA50 (Assembler) am 27.1.94

Dauer : 90 Minuten **Hilfsmittel :** Keine (auch kein Taschenrechner)

Hinweis : Versehen Sie die Assembler-Programme der Aufgaben 1 und 2 mit hinreichenden Kommentaren.

Aufgabe 1 : Schreiben Sie ein vollständiges Assembler-Programm (Ziel : COM-Datei) für folgenden - in Pascal formulierten - Ablauf : *(30 Punkte)*

```
program aufgabe1;
uses crt;  { Bibliothek mit Funktionen und Prozeduren für Bild-
             schirmausgabe und Tastatureingabe }
var c:char;
begin
  repeat
    c := readkey;  { Zeichen von Tastatur ohne Echo einlesen }
    if ord(c) = 0
    then begin
           c := readkey;
           writeln(´<.><?>´)
         end
    else if ord(c) < 32
         then writeln(´<?>´)
         else writeln(´<´,c,´>´)
  until c = ´E´
end.
```

Aufgabe 2 : Schreiben Sie ein vollständiges Assembler-Programm (Ziel : EXE-Datei), welches zunächst eine Zeichenkette einliest und diese anschließend auf dem Bildschirm wieder ausgibt. Benutzen die den DOS-Interrupt 21 mit AH=09h und AH=0Ah.
(20 Punkte)

Aufgabe 3 : Übersetzen Sie das nachfolgende Assembler-Programm in ein COM-Programm : *(30 Punkte)*

```
CSEG     SEGMENT
         ASSUME CS:CSEG,DS:CSEG
         ORG  100h
AUFGABE3 PROC
         MOV  AH,02h
         MOV  BX,0
         MOV  CX,OPB
         MOV  DL,OPA
LOOP1:
         INT  21h
         MOV  TABL[BX],DL
         INC  BX
         INC  DL
         LOOP LOOP1
         MOV  AH,09h
         MOV  DX,OFFSET TABL
         INT  21h
         INT  20h
OPA      DB   ´A´
OPB      DW   32
TABL     DB   32 DUP (?)
DOLLAR   DB   ´$´
AUFGABE3 ENDP
CSEG     ENDS
         END AUFGABE3
```

Welche Aktionen führt das Programm durch ? *(10 Punkte)*

Viel Erfolg

Klausur IA50 (Assembler) am 23.8.94

Dauer : 100 Minuten ***Viel Erfolg*** **Hilfsmittel :** Keine (auch kein Taschenrechner)

Hinweis : Versehen Sie die Assembler-Programme der Aufgaben 1 und 2 mit hinreichenden Kommentaren.

Aufgabe 1 : Schreiben Sie ein vollständiges Assembler-Programm (Ziel : COM-Datei) für folgenden - in Pascal formulierten - Ablauf : *(40 Punkte)*

```
program klausur;
var zekette : string[10];
    zahl    : integer;
    fehler  : boolean;
    i       : integer;
begin
  zekette := '79BD';
  zahl := 0;
  fehler := false;
  for i := 1 to ord(zekette[0]) do
    if zekette[i] in ['0'..'9','A'..'F']
    then if zekette[i] in ['0'..'9']
         then zahl := zahl * 16 + ord(zekette[i]) - ord('0')
         else zahl := zahl * 16 + ord(zekette[i]) - ord('A') + 10
    else fehler := true;
  if fehler
  then write('mißglückte Konvertierung')
  else write('erfolgreiche Konvertierung')
end.
```

Bedenke :
- Multiplikationen mit 2^n lassen sich auch durch n-maliges Linksschieben realisieren.
- Die Standardfunktion ORD liefert für Argumente ordinalen Datentyps die zugehörige Ordnungszahl.
-
```
Zeichen :      !"#$%&´()*+,-./0123456789:;<=>?@ABCDEFGH
ASCII-Code : 33333333444444444555555555566666666777
(dezimal)    23456789012345678901234567890123456789012
```

Aufgabe 2 : Erläutern Sie anhand eines gemeinsamen Beispiels oder zweier getrennter Beispiele die Verknüpfung zwischen Hauptprogramm und externen Unterprogrammen sowie die Parameterübergabe an Unterprogramme via Stack. *(20 Punkte)*

Aufgabe 3 : Übersetzen Sie das nachfolgende Assembler-Programm in ein COM-Programm : *(28 Punkte)*

```
CSEG     SEGMENT PUBLIC
         ASSUME  CS:CSEG,DS:CSEG
         ORG     100h
AUFGABE3 PROC NEAR
         MOV     CX,8
         MOV     SI,1
         MOV     AH,9
LOOP1:
         MOV     BL,STREAM[SI]
         MOV     STREAM[SI],'$'
         MOV     DX,OFFSET STREAM
         INT     21h
         MOV     STREAM[SI],BL
         MOV     DX,OFFSET CRLF
         INT     21h
         INC     SI
         LOOP    LOOP1
         INT     20h
AUFGABE3 ENDP
STREAM   DB      'ABCDEFGH '
CRLF     DB      10,13,'$'
CSEG     ENDS
         END     AUFGABE3
```

Welche Aktionen führt das Programm durch ? *(12 Punkte)*

Klausur IA50 (Assembler) am 14.2.95

Dauer : 90 Minuten **Hilfsmittel :** Keine (auch kein Taschenrechner)

Hinweis : Versehen Sie das Assembler-Programm der Aufgabe 1 mit hinreichenden Kommentaren.

Aufgabe 1 : *(50 Punkte)*

Schreiben Sie ein vollständiges Assembler-Programm (Ziel : COM-Datei) für folgenden - in Pascal formulierten - Ablauf :

```
program aufgabe1;

var zahl    : integer;
    zekette : string[10];
    tabelle : array[1..10] of byte;
    anzahl  : integer;

begin
  zahl := 12345;
  zekette := '';
  if zahl < 0 then begin
    zekette := '-';
    zahl := zahl * (-1)
  end;
  anzahl := 0;
  while zahl <> 0 do begin
    anzahl := anzahl + 1;
    tabelle[anzahl] := zahl mod 10;
    zahl := zahl div 10
  end;
  while anzahl <> 0 do begin
    zekette := zekette + chr( tabelle[anzahl] + ord('0') );
    anzahl := anzahl - 1;
  end;
  writeln(zekette)
end.
```

Bedenke :
- Multiplikationen mit (-1) lassen sich auch durch Zweierkomplementbildung realisieren.
- Die Standardfunktion ORD liefert für Argumente ordinalen Datentyps die zugehörige Ordnungszahl.
- Die Standardfunktion CHR liefert für Argumente des Datentyps Integer das zugehörige Zeichen des Rechnerzeichensatzes (z.B. ASCII-Code).

- DIV-Befehl des 80x86 : Dividend in DX (höherwertige Bits) und AX (niederwertige Bits), Divisor z.B. in BX; Ergebnis in AX, Rest in DX

Aufgabe 2 : *(25 Punkte)*

Erläutern Sie durch Darstellung am Zahlenkreis die Beeinflußung des Carry-, Zero-, Sign- und Overflowflags durch nachfolgende arithmetische Operationen.

```
ADD AL,BL   für AL=02,  BL=04
ADD AL,BL   für AL=7E,  BL=04
ADD AL,BL   für AL=FE,  BL=04
SUB AL,BL   für AL=04,  BL=02
SUB AL,BL   für AL=04,  BL=82
```

Aufgabe 3 : *(15 Punkte)*

Übersetzen Sie die nachfolgenden Assembler-Programmausschnitte (Ziel : COM-Dateien) in Maschinencode.

```
100             MOV   OP,BX
                MOV   TABL[SI],BX
                INC   BX
                INT   20h
```

```
300             JE    MARKE
                ...
330  MARKE:     ...
```

```
2D0  MARKE:     ...
                ...
300             JE    MARKE
```
└ *Offsetadressen*

In allen zugrundeliegenden Assembler-Programmen stehen ab der Offsetadresse 150 jeweils folgende Daten :

```
OP      DW    ?
TABL    DW    10 DUP (?)
```

Hinweis : Alle Offsetadressen sind in hexadezimaler Form angegeben

Viel Erfolg

Klausur IA50 (Assembler) am 10.8.95

Dauer : 90 Minuten **Hilfsmittel :** Keine (auch kein Taschenrechner)

Hinweis : Versehen Sie das Assembler-Programm der Aufgabe 1 mit hinreichenden Kommentaren.

Aufgabe 1 : *(50 Punkte)*

Schreiben Sie ein vollständiges Assembler-Programm (Ziel : COM-Datei) für folgenden - in Pascal formulierten - Ablauf :

```
program aufgabe1;

uses crt;  { Bibliothek mit Funktionen und Prozeduren für Bild-
             schirmausgabe und Tastatureingabe }

const cursorleft  = #75;
      cursorright = #77;
      return      = #13;
      laenge      = 40;

var zekette  : string[laenge];
    position : integer;
    zeichen  : char;

begin
  for position := 1 to laenge do zekette[position] := ' ';
  zekette[0] := chr(laenge);
  position := 1;
  repeat
    zeichen := readkey;  { Zeichen von Tastatur ohne Echo
                           einlesen }
    if ord(zeichen) = 0 then begin
      zeichen := readkey;
      case zeichen of
        cursorleft  : if position > 1
                      then position := position-1;
        cursorright : if position < laenge
                      then position := position+1
      end
    end else
      if (ord(zeichen) >= 32) and (ord(zeichen) <= 127) then begin
        zekette[position] := zeichen;
        if position < laenge then position := position+1
      end
  until zeichen = return;
  writeln(zekette);
end.
```

Bedenke :
- Die Standardfunktion ORD liefert für Argumente ordinalen Datentyps die zugehörige Ordnungszahl.
- Die Standardfunktion CHR liefert für Argumente des Datentyps Integer das zugehörige Zeichen des Rechnerzeichensatzes (z.B. ASCII-Code).

Aufgabe 2 : *(10 Punkte)*

Welche Grundfunktionen bietet das Dienstprogramm Microsoft DEBUG ?

Aufgabe 3 : *(15 Punkte)*

Erläutern Sie anhand eines Beispiels die Segmentierungsebenen (Modul, Segment, Prozedur) eines Assemblerprogramms.

Aufgabe 4 : *(15 Punkte)*

Übersetzen Sie die nachfolgenden Assembler-Programmausschnitte (Ziel : COM-Dateien) in Maschinencode.

```
100             MOV   BX,OP
                MOV   BX,TABL[SI]
                MOV   AX,4C00h
                INT   21h
```

```
300             CALL  UP
                ...
400  UP         PROC
                ...
     UP         ENDP
```

```
200  UP         PROC
                ...
     UP         ENDP
                ...
300             CALL  UP
```

└ *Offsetadressen*

In allen zugrundeliegenden Assembler-Programmen stehen ab der Offsetadresse 150 jeweils folgende Daten :

```
     OP         DW    ?
     TABL       DW    10 DUP (?)
```

Hinweis : Alle Offsetadressen sind in hexadezimaler Form angegeben

Das PTL-Team wünscht viel Erfolg

Klausur IA50 (Assembler) am 6.2.96

Dauer : 100 Minuten **Hilfsmittel :** Keine (auch kein Taschenrechner)

Hinweis : Versehen Sie das Assembler-Programm der Aufgabe 1 mit hinreichenden Kommentaren.

Aufgabe 1 : *(60 Punkte)*

Schreiben Sie ein vollständiges Assembler-Programm (Ziel : EXE-Datei) für folgenden - in Pascal formulierten - Ablauf :

```
program aufgabe1;

var chargenerator : array[char,1..8] of byte absolute $F000:$FA6E;
    s             : string[10];
    i, j, k       : integer;
    mask          : byte;

begin
  s := 'PTL Wedel';
  for i := 1 to ord (s[0]) do
    for j := 1 to 8 do begin
      mask := $80;
      for k := 1 to 8 do begin
        if chargenerator[s[i],j] and mask <> 0 then
          write(chr(178))
        else
          write(' ');
        mask := mask div 2
      end; { for k }
      writeln
    end { for j }
end.
```

Bedenke :

- Divisionen durch 2^n lassen sich auch durch n-maliges Rechtsschieben realisieren.

- Die Standardfunktion ORD liefert für Argumente ordinalen Datentyps die zugehörige Ordnungszahl.
- Die Standardfunktion CHR liefert für Argumente des Datentyps Integer das zugehörige Zeichen des Rechnerzeichensatzes (z.B. ASCII-Code).

- MUL-Befehl des 80x86 : Multiplikand in AX, Multiplikator z.B. in BX; Produkt in DX (höherwertige Bits) und AX (niederwertige Bits)

Aufgabe 2 : *(20 Punkte)*

Geben Sie für jeden bedingten Sprungbefehl an, ob zum angegebenen Sprungziel verzweigt oder nicht verzweigt wird :

```
MOV AL,81h
CMP AL,3h
JL  Marke1
...
MOV AL,81h
CMP AL,3h
JB  Marke2
...
MOV AL,81h
CMP AL,3h
JG  Marke3
...
MOV AL,81h
CMP AL,3h
JA  Marke4
```

Begründen Sie Ihre Aussagen jeweils durch

- kurze "verbale" Erläuterungen und

- "mathematisch exakte" Erläuterungen anhand des Zahlenkreises (Setzen bzw. Nichtsetzen von Flag´s durch arithmetische Befehle) und beiliegender Befehlsübersicht Sprungbefehle (Auswertung gesetzter bzw. nicht gesetzter Flag´s durch bedingte Sprungbefehle).

Aufgabe 3 : *(20 Punkte)*

Erläutern Sie anhand eines Beispiels den Unterschied zwischen NEAR- und FAR-Call. Gehen Sie dabei auch auf die Inhaltsveränderungen der Register CS, IP und SP sowie des Stacks ein.

Das PTL-Team wünscht viel Erfolg

Klausur IA50 (Assembler) am 22.8.96

Dauer : 100 Minuten **Hilfsmittel :** Keine (auch kein Taschenrechner)

Hinweis : Versehen Sie das Assembler-Programm der Aufgabe 1 mit hinreichenden Kommentaren.

Aufgabe 1 : *(60 Punkte)*

Schreiben Sie ein vollständiges Assembler-Programm (Ziel : EXE-Datei) für folgenden - in Pascal formulierten - Ablauf :

```
program aufgabe1;

uses crt;  { Bibliothek mit Funktionen und Prozeduren für Bild-
             schirmausgabe und Tastatureingabe }

const cursorup   = #72;
      cursordown = #80;
      return     = #13;

var chargenerator : array[char,1..8] of byte absolute $F000:$FA6E;
    key           : char;
    i, j          : integer;
    mask          : byte;
    c             : char;

begin
  key := 'A';
  repeat
    for i := 1 to 8 do begin
      mask := $80;
      for j := 1 to 8 do begin
        if chargenerator[key,i] and mask <> 0 then
          write(chr(178))
        else
          write(' ');
        mask := mask div 2
      end; { for j }
      writeln
    end; { for i }
    c := readkey;  { Zeichen von Tastatur ohne Echo einlesen }
    if ord(c) = 0 then begin
      c := readkey;
      case c of
        cursorup   : key := succ(key);
        cursordown : key := pred(key)
      end; { case }
      c := chr(0)
    end { if }
  until c = return
end.
```

Bedenke :
- Die Standardfunktion PRED liefert für Argumente ordinalen Datentyps das vorherige Element dieses Datentyps (z.B. PRED(´B´) -> ´A´).
- Die Standardfunktion SUCC liefert für Argumente ordinalen Datentyps das nachfolgende Element dieses Datentyps (z.B. SUCC(´B´) -> ´C´).

- Die Standardfunktion ORD liefert für Argumente ordinalen Datentyps die zugehörige Ordnungszahl.
- Die Standardfunktion CHR liefert für Argumente des Datentyps Integer das zugehörige Zeichen des Rechnerzeichensatzes (z.B. ASCII-Code).
- MUL-Befehl des 80x86 : Multiplikand in AX, Multiplikator z.B. in BX; Produkt in DX (höherwertige Bits) und AX (niederwertige Bits)

Aufgabe 2 : *(20 Punkte)*

Übersetzen Sie die nachfolgenden Assembler-Programmausschnitte (Ziel : COM-Dateien) in Maschinencode.

```
100             MOV   BX,TABL[SI]
                MOV   TABL[SI],BX
                MOV   BX,OP
                MOV   OP,BX
```

```
300             CALL  UP
                ...
400  UP         PROC
                ...
     UP         ENDP
```

```
200  UP         PROC
                ...
     UP         ENDP
                ...
300             CALL  UP
```

```
300             JE    MARKE
                ...
330  MARKE:     ...
```

```
2D0  MARKE:     ...
                ...
300             JE    MARKE
```

└ *Offsetadressen*

In allen zugrundeliegenden Assembler-Programmen stehen ab der Offsetadresse 150 jeweils folgende Daten :

```
OP      DW    ?
TABL    DW    10 DUP (?)
```

Hinweis : Alle Offsetadressen sind in hexadezimaler Form angegeben

Aufgabe 3 : *(20 Punkte)*

Ermitteln Sie per Schreibtischtest den Inhalt der Register SI und BX vor Ausführung des Befehls INT 20h :

```
CDSEG     SEGMENT
          ASSUME CS:CDSEG, DS:CDSEG
          ORG   100h

AUFGABE3  PROC NEAR
          MOV   SI,0
          MOV   BX,0
          MOV   CX,2
LOOP1:    MOV   DX,CX
          MOV   CX,8
LOOP2:    MOV   AL,QUELLE[SI]
          MOV   ZIEL[SI],AL
          CALL  UP1
          LOOP  LOOP2
          CALL  UP2
          MOV   CX,DX
          LOOP  LOOP1
          MOV   AH,09h
          MOV   DX,OFFSET FERTIG
          INT   21h
          INT   20h
AUFGABE3  ENDP

UP1       PROC NEAR
          INC   SI
          RET
UP1       ENDP

UP2       PROC NEAR
          INC   BX
          CALL  UP1
          RET
UP2       ENDP

QUELLE    DB    1,2,3,4,5,6,7,8,0,9,10,11,12,13,14,15,16,0
ZIEL      DB    18 DUP (?)
FERTIG    DB    'geschafft...$'

CDSEG     ENDS
          END AUFGABE3
```

Das PTL-Team wünscht viel Erfolg

Klausur IA50 (Assembler) am 12.2.97

Dauer : 120 Minuten **Hilfsmittel :** Keine (auch kein Taschenrechner)

Hinweis : Versehen Sie die Assembler-Programme der Aufgaben 1 und 2 mit hinreichenden Kommentaren.

Aufgabe 1 : *(60 Punkte)*

Schreiben Sie ein vollständiges Assembler-Programm (Ziel : EXE-Datei) für folgenden - in Pascal formulierten - Ablauf :

```
Program Aufgabe1;

Uses Crt;

Const Chars : String = 'PTL Wedel';  { Var Chars ... und Chars := ... }
      YPos           = 10;
      XMin           =  1;
      XMax           = 80;
      SpecialKey     =  #0;
      Esc            = #27;
      CursorLeft     = #75;
      CursorRight    = #77;

Var Screen : Array [1..25,1..80,1..2] of Byte absolute $B800:$0000;
    XPos   : Integer;
    Key    : Char;
    Count  : Integer;

Begin
  XPos := XMin;
  Repeat
    For Count := XMin To XMax Do
      Screen[YPos, Count, 1] := Ord (' ');
    For Count := 1 To Ord (Chars[0]) Do
      Screen[YPos, XPos+Count-1, 1] := Ord (Chars[Count]);
    Repeat
      Repeat
        Key := ReadKey  { Zeichen von Tastatur ohne Echo einlesen }
      Until (Key = SpecialKey) Or (Key = Esc);
      If Key = SpecialKey Then Key := ReadKey
    Until (Key = CursorLeft) Or (Key = CursorRight) Or (Key = Esc);
    If Key = CursorLeft Then
      XPos := XPos - 1
    Else If Key = CursorRight Then
      XPos := XPos + 1;
    If XPos < XMin Then
      XPos := XMin;
    If XPos + Ord (Chars[0]) - 1 > XMax Then
      XPos := XMax - Ord (Chars[0]) + 1
  Until Key = Esc
End.
```

Bedenke :
- Die Standardfunktion ORD liefert für Argumente ordinalen Datentyps die zugehörige Ordnungszahl.
- MUL-Befehl des 80x86 : Multiplikand in AX, Multiplikator z.B. in BX; Produkt in DX (höherwertige Bits) und AX (niederwertige Bits)

Aufgabe 2 : *(40 Punkte)*

Entwickeln Sie ein Assembler-Programm (Ziel : EXE-Datei) zur Multiplikation zweier Vektoren. Die Eingabe der Dimension und Elemente der Vektoren sowie die Ausgabe des Ergebnisses sind ebenfalls gefordert.

$$\vec{A} * \vec{B} = (a_1,a_2,...,a_n) * (b_1,b_2,...,b_n)$$

$$= a_1*b_1 + a_2*b_2 + ... + a_n*b_n$$

Für die Ein- und Ausgabe von Integer-Werten liegen in einem *separaten* Modul bereits zwei vorgefertigte Prozeduren vor. Die Codierung dieser Routinen ist daher *nicht* gefordert.

```
CSEG2          SEGMENT
               ASSUME CS:CSEG2

               PUBLIC INT_AUSGABE
               PUBLIC INT_EINGABE

INT_AUSGABE    PROC FAR
; auszugebender Wert wird im AX-Register erwartet
               ...
               RET
INT_AUSGABE    ENDP

INT_EINGABE    PROC FAR
; eingelesener Wert wird im AX-Register bereitgestellt
               ...
               RET
INT_EINGABE    ENDP

CSEG2          ENDS
               END
```

Aufgabe 3 : *(20 Punkte)*

Geben Sie für jeden nachfolgenden Befehl an, ob er "syntaktisch" gültig ist oder nicht. Begründen Sie Ihre Aussagen unter Berücksichtigung von "Anhang A Befehlssatz des 8088" (Bedenke : Ohne Begründung keine Punkte).

a) MOV DX,[AX]
b) MOV DX,[BX]
c) MOV DX,[BP]
d) MOV DX,[IP]
e) MOV DX,[BX+BP]
f) MOV DX,[BX+SI]

ga) JE 27Fh (für JE an Offsetadresse 200h)
gb) JE 280h (dito.)
gc) JE 281h (dito.)
gd) JE 282h (dito.)
ge) JE 283h (dito.)

ha) JE 180h (für JE an Offsetadresse 200h)
hb) JE 181h (dito.)
hc) JE 182h (dito.)
hd) JE 183h (dito.)
he) JE 184h (dito.)

Das PTL-Team wünscht viel Erfolg

Klausur IA50 (Assembler) am 21.8.97

Dauer : 120 Minuten **Hilfsmittel :** Keine (auch kein Taschenrechner)

Hinweis : Versehen Sie die Assembler-Programme der Aufgaben 1 und 2 mit hinreichenden Kommentaren.

Aufgabe 1 : *(60 Punkte)*

Schreiben Sie ein vollständiges Assembler-Programm (Ziel : EXE-Datei) für folgenden - in Pascal formulierten - Ablauf :

```
Program Aufgabe1;

Uses Crt;

Const XMin    =   1;
      XMax    =  80;
      YMin    =   1;
      YMax    =  25;
      Esc     = #27;
      Space   = #32;

Type  TScreen = Array [1..25,1..80,1..2] of Byte;

Var   Screen  : TScreen absolute $B800:$0000;
      BScreen : TScreen;
      SCount  : Integer;
      ZCount  : Integer;
      BByte   : Byte;
      Key     : Char;

Procedure WaitForEscOrSpace;
Begin
  Repeat
    Key := ReadKey  { Zeichen von Tastatur ohne Echo einlesen }
  Until (Key = Esc) Or (Key = Space)
End;

Begin
  BScreen := Screen;
  WaitForEscOrSpace;
  While Key <> Esc Do Begin
    For ZCount := YMin To YMax Do
      For SCount := XMin to (XMax-XMin+1) Div 2 Do Begin
        BByte := Screen[ZCount,Scount,1];
        Screen[ZCount,SCount,1] := Screen[ZCount,XMax-SCount+1,1];
        Screen[Zcount,XMax-SCount+1,1] := BByte
      End;
    WaitForEscOrSpace
  End;
  Screen := BScreen
End.
```

Bedenke :
- MUL-Befehl des 80x86 : Multiplikand in AX, Multiplikator z.B. in BX; Produkt in DX (höherwertige Bits) und AX (niederwertige Bits)
- DIV-Befehl des 80x86 : Dividend in DX (höherwertige Bits) und AX (niederwertige Bits), Divisor z.B. in BX; Ergebnis in AX, Rest in DX

Das Programm der Aufgabe 1 funktioniert nicht für XMin > 1

Ein erster - flüchtiger - Patch funktioniert für XMin > 1 ebenfalls nicht

```
Program Aufgabe1;

...

Begin
  BScreen := Screen;
  WaitForEscOrSpace;
  While Key <> Esc Do Begin
    For ZCount := YMin To YMax Do
      For SCount := XMin to (XMax-XMin+1) Div 2 Do Begin
        BByte := Screen[ZCount,Scount,1];
        Screen[ZCount,SCount,1] := Screen[ZCount,XMax-SCount+XMin,1];
        Screen[Zcount,XMax-SCount+XMin,1] := BByte
      End;
    WaitForEscOrSpace
  End;
  Screen := BScreen
End.
```

Ein zweiter - sorgfältiger - Patch funktioniert auch für XMin > 1

```
Program Aufgabe1;

...

Begin
  BScreen := Screen;
  WaitForEscOrSpace;
  While Key <> Esc Do Begin
    For ZCount := YMin To YMax Do
      For SCount := 1 to (XMax-XMin+1) Div 2 Do Begin
        BByte := Screen[ZCount, Scount+XMin-1, 1];
        Screen[ZCount, SCount+XMin-1, 1] := Screen[ZCount, XMax-SCount+1, 1];
        Screen[Zcount, XMax-SCount+1, 1] := BByte
      End;
    WaitForEscOrSpace
  End;
  Screen := BScreen
End.
```

Klausur IA50 (Assembler) am 21.8.97

Dauer : 120 Minuten **Hilfsmittel :** Keine (auch kein Taschenrechner)

Hinweis : Versehen Sie die Assembler-Programme der Aufgaben 1 und 2 mit hinreichenden Kommentaren.

Aufgabe 1 : *(60 Punkte)*

Schreiben Sie ein vollständiges Assembler-Programm (Ziel : EXE-Datei) für folgenden - in Pascal formulierten - Ablauf :

```
Program Aufgabe1;

Uses Crt;

Const XMin    =   1;
      XMax    =  80;
      YMin    =   1;
      YMax    =  25;
      Esc     = #27;
      Space   = #32;

Type  TScreen = Array [1..25,1..80,1..2] of Byte;

Var   Screen  : TScreen absolute $B800:$0000;
      BScreen : TScreen;
      SCount  : Integer;
      ZCount  : Integer;
      BByte   : Byte;
      Key     : Char;

Procedure WaitForEscOrSpace;
Begin
  Repeat
    Key := ReadKey  { Zeichen von Tastatur ohne Echo einlesen }
  Until (Key = Esc) Or (Key = Space)
End;

Begin
  BScreen := Screen;
  WaitForEscOrSpace;
  While Key <> Esc Do Begin
    For ZCount := YMin To YMax Do
      For SCount := 1 to (XMax-XMin+1) Div 2 Do Begin
        BByte := Screen[ZCount,Scount+XMin-1,1];
        Screen[ZCount,SCount+XMin-1,1] := Screen[ZCount,XMax-SCount+1,1];
        Screen[Zcount,XMax-SCount+1,1] := BByte
      End;
    WaitForEscOrSpace
  End;
  Screen := BScreen
End.
```

Bedenke :
- MUL-Befehl des 80x86 : Multiplikand in AX, Multiplikator z.B. in BX; Produkt in DX (höherwertige Bits) und AX (niederwertige Bits)
- DIV-Befehl des 80x86 : Dividend in DX (höherwertige Bits) und AX (niederwertige Bits), Divisor z.B. in BX; Ergebnis in AX, Rest in DX

Aufgabe 2 : *(60 Punkte)*

Entwickeln Sie ein Assembler-Programm (Ziel : EXE-Datei) zum Einlesen (incl. Plausibilitätsprüfung) der Dimension und Werte einer Matrix sowie zur Berechnung und Ausgabe der Position (Zeile und Spalte) des kleinsten und größten Wertes dieser Matrix. Dabei gelten folgende Einschränkungen : Die Matrix besteht aus maximal hundert Zeilen und hundert Spalten; die Matrix enthält ausschließlich Integer-Werte.

Für die Ein- und Ausgabe von Integer-Werten liegen in einem *separaten* Modul bereits zwei vorgefertigte Prozeduren vor. Die Codierung dieser Routinen ist daher *nicht* gefordert.

```
CSEG2           SEGMENT
                ASSUME CS:CSEG2

                PUBLIC INT_AUSGABE
                PUBLIC INT_EINGABE

INT_AUSGABE     PROC FAR
; auszugebender Wert wird im AX-Register erwartet
                ...
                RET
INT_AUSGABE     ENDP

INT_EINGABE     PROC FAR
; eingelesener Wert wird im AX-Register bereitgestellt
                ...
                RET
INT_EINGABE     ENDP

CSEG2           ENDS
                END
```

Das PTL-Team wünscht viel Erfolg

Klausur IA50 (Assembler) am 12.2.98

Dauer : 120 Minuten **keine externen Hilfsmittel**

Übersetzen Sie das nachfolgende (siehe Rückseite) Pascal-Programm in ein äquivalentes Assemblerprogramm (8086, EXE).

Kommentieren Sie das Assemblerprogramm durch eindeutige Zuordnung der Pascal-Befehle zu den Assembler-Befehlen. Ohne derartige Kommentierung wird die Klausur nicht gewertet !

Falls Sie keine komplette Lösung angeben können, bearbeiten Sie Teilaspekte im Sinne der eindeutigen Zuordnung der Pascal-Befehle zu den Assembler-Befehlen.

Bedenke :
- MUL-Befehl des 80x86 : Multiplikand in AX, Multiplikator z.B. in BX; Produkt in DX (höherwertige Bits) und AX (niederwertige Bits)
- DIV-Befehl des 80x86 : Dividend in DX (höherwertige Bits) und AX (niederwertige Bits), Divisor z.B. in BX; Ergebnis in AX, Rest in DX

Das PTL-Team wünscht viel Erfolg

```
program klausur;

uses crt;

const blank =  32;
      full  = 219;

var pixel :
      array[0..127, 0..7] of byte absolute $F000:$FA6E;
    screen :
      array[0..24, 0..79, 0..1] of byte absolute $B800:$0000;
    linearscreen :
      array[0..3999] of byte absolute $B800:$0000;

    ascii  : integer;
    column : integer;
    mask   : byte;
    key    : char;
    i, j   : integer;

procedure clrscr;

begin
  i := 0;
  while i <= 3999 do begin
    linearscreen[i] := blank;
    i := i+2
  end
end;

begin
  ascii := 65;
  column := 0;
  repeat
    clrscr;
    for i := 0 to 7 do begin
      mask := $80;
      for j := 0 to 7 do begin
        if pixel[ascii,i] and mask <> 0 then
          screen[i,column+j,0] := full
        else
          screen[i,column+j,0] := blank;
        mask := mask div 2
      end
    end;
    repeat
      key := readkey
    until (key = #0) or (key = #27);
    if key = #0 then
      case readkey of
        #72 : if ascii < 127 then ascii := ascii+1;
        #80 : if ascii >   0 then ascii := ascii-1;
        #75 : if column >  0 then column := column-1;
        #77 : if column < 72 then column := column+1
      end
  until key = #27;
  clrscr
end.
```

Klausur IA50 (Assembler) am 10.8.98

Dauer : 120 Minuten **keine externen Hilfsmittel**

Aufgabe 1 : *(60 Punkte)*

Übersetzen Sie das nachfolgende Pascal-Programm in ein äquivalentes Assemblerprogramm (8086, EXE).

Kommentieren Sie das Assemblerprogramm durch eindeutige Zuordnung der Pascal-Befehle zu den Assembler-Befehlen. Ohne derartige Kommentierung wird die Aufgabe nicht gewertet !

```
program aufgabe1;

var s    : array [0..24, 0..79, 0..1] of char
             absolute $B800:$0000;
    l, h : char;
    c    : char;
    i, j : integer;

begin
  l := s[0, 0, 0];
  h := s[0, 0, 0];
  for i := 0 to 24 do
    for j := 0 to 79 do begin
      c := s[i, j, 0];
      if c < l then
        l := c
      else
        if c > h then
          h := c
    end;
  if ord(l) < 32 then
    l := chr(32);
  if ord(h) > 126 then
    h := chr(126);
  writeln;
  write('>', l, ',', h, '<');
  writeln
end.
```

Bedenke :
- Die Standardfunktion ORD liefert für Argumente ordinalen Datentyps die zugehörige Ordnungszahl.
- Die Standardfunktion CHR liefert für Argumente des Datentyps Integer das zugehörige Zeichen des Rechnerzeichensatzes (z.B. ASCII-Code).
- MUL-Befehl des 80x86 : Multiplikand in AX, Multiplikator z.B. in BX; Produkt in DX (höherwertige Bits) und AX (niederwertige Bits)
- DIV-Befehl des 80x86 : Dividend in DX (höherwertige Bits) und AX (niederwertige Bits), Divisor z.B. in BX; Ergebnis in AX, Rest in DX

Aufgabe 2 : *(60 Punkte)*

Welche Zeichen gibt das Programm AUFGABE2 aus ? Belegen Sie Ihre Aussage durch eine Kommentierung (auf dem Aufgabenblatt) des Assemblerprogramms.

```
Turbo Assembler  Version 3.2        07/08/98 12:00:00
aufgabe2.ASM

      1 0000                        CSEG     SEGMENT
      2                                      ASSUME CS:CSEG,DS:SSEG
      3 0000                        AUFGABE2 PROC NEAR
      4 0000  8C D0                          MOV  AX,SS
      5 0002  8E D8                          MOV  DS,AX
      6 0004  E8 0008                        CALL UP1
      7 0007  E8 0032                        CALL UP2
      8 000A  B8 4C00                        MOV  AX,4C00h
      9 000D  CD 21                          INT  21h
     10 000F                        AUFGABE2 ENDP
     11 000F                        UP1      PROC NEAR
     12 000F  50                            PUSH AX
     13 0010  B0 80                 UP11:    MOV  AL,80h
     14 0012  3C 02                          CMP  AL,2h
     15 0014  72 09                          JB   UP12
     16 0016  B8 0001                        MOV  AX,1
     17 0019  E8 002F                        CALL INTOUT
     18 001C  EB 07 90                       JMP  UP13
     19 001F  B8 0002               UP12:    MOV  AX,2
     20 0022  E8 0026                        CALL INTOUT
     21 0025  B0 80                 UP13:    MOV  AL,80h
     22 0027  3C 02                          CMP  AL,2h
     23 0029  7C 09                          JL   UP14
     24 002B  B8 0003                        MOV  AX,3
     25 002E  E8 001A                        CALL INTOUT
     26 0031  EB 07 90                       JMP  UP15
```

```
27 0034  B8 0004                UP14:    MOV  AX,4
28 0037  E8 0011                         CALL INTOUT
29 003A  58                     UP15:    POP  AX
30 003B  C3                              RET
31 003C                         UP1      ENDP
32 003C                         UP2      PROC NEAR
33 003C  50                              PUSH AX
34 003D  53                              PUSH BX
35 003E  8B DC                           MOV  BX,SP
36 0040  83 C3 04                        ADD  BX,4
37 0043  8B 07                           MOV  AX,[BX]
38 0045  E8 0003                         CALL INTOUT
39 0048  5B                              POP  BX
40 0049  58                              POP  AX
41 004A  C3                              RET
42 004B                         UP2      ENDP
43 004B                         INTOUT   PROC NEAR
44 004B  50                              PUSH AX
45 004C  51                              PUSH CX
46 004D  52                              PUSH DX
47 004E  B9 0004                         MOV  CX,4
48 0051  8B D0                  IO1:     MOV  DX,AX
49 0053  83 E2 0F                        AND  DX,000Fh
50 0056  52                              PUSH DX
51 0057  51                              PUSH CX
52 0058  B9 0004                         MOV  CX,4
53 005B  D3 E8                           SHR  AX,CL
54 005D  59                              POP  CX
55 005E  E2 F1                           LOOP IO1
```

```
56 0060  B9 0004                        MOV  CX,4
57 0063  58                     IO2:    POP  AX
58 0064  E8 0006                        CALL FOURBITS
59 0067  E2 FA                          LOOP IO2
60 0069  5A                             POP  DX
61 006A  59                             POP  CX
62 006B  58                             POP  AX
63 006C  C3                             RET
64 006D                         INTOUT  ENDP
65 006D                         FOURBITS PROC NEAR
66 006D  50                             PUSH AX
67 006E  52                             PUSH DX
68 006F  24 0F                          AND  AL,0Fh
69 0071  04 30                          ADD  AL,'0'
70 0073  3C 39                          CMP  AL,'9'
71 0075  76 02                          JBE  FB1
72 0077  04 07                          ADD  AL,7
73 0079  B4 02                  FB1:    MOV  AH,2
74 007B  8A D0                          MOV  DL,AL
75 007D  CD 21                          INT  21h
76 007F  5A                             POP  DX
77 0080  58                             POP  AX
78 0081  C3                             RET
79 0082                         FOURBITS ENDP
80 0082                         CSEG    ENDS
81 0000                         SSEG    SEGMENT STACK
82 0000  20*(????)                      DW   32 DUP (?)
83 0040                         SSEG    ENDS
84                                      END  AUFGABE2
```

```
Symbol Name                     Type   Value

??DATE                          Text   "07/08/98"
??FILENAME                      Text   "aufgabe2"
??TIME                          Text   "12:00:00"
??VERSION                       Number 0314
@CPU                            Text   0101H
@CURSEG                         Text   SSEG
@FILENAME                       Text   AUFGABE2
@WORDSIZE                       Text   2
AUFGABE2                        Near   CSEG:0000
FB1                             Near   CSEG:0079
FOURBITS                        Near   CSEG:006D
INTOUT                          Near   CSEG:004B
IO1                             Near   CSEG:0051
IO2                             Near   CSEG:0063
UP1                             Near   CSEG:000F
UP11                            Near   CSEG:0010
UP12                            Near   CSEG:001F
UP13                            Near   CSEG:0025
UP14                            Near   CSEG:0034
UP15                            Near   CSEG:003A
UP2                             Near   CSEG:003C

Groups & Segments               Bit Size Align  Combine Class

CSEG                            16  0082 Para   none
SSEG                            16  0040 Para   Stack
```

Das PTL-Team wünscht viel Erfolg

Klausur IA10.3/12.0 451 (ex IA50) Assembler am 9.8.99

Dauer : 120 Minuten **keine externen Hilfsmittel**

Übersetzen Sie die beiden nachfolgenden Pascal-Programme in je ein äquivalentes Assemblerprogramm (8086, EXE).

Kommentieren Sie die Assemblerprogramme durch eindeutige Zuordnung der Pascal-Befehle zu den Assembler-Befehlen. Ohne derartige Kommentierung wird die Klausur nicht gewertet !

Im Modul IOINT steht die Routine WRITEDEZ zur Ausgabe von Integer-Zahlen zur Verfügung. Der Datentransfer erfolgt über das DX-Register.

Die beiden Assembler-Programme werden gleichgewichtig gewertet. Falls Sie keine komplette Lösung angeben können, bearbeiten Sie Teilaspekte im Sinne der eindeutigen Zuordnung der Pascal-Befehle zu den Assembler-Befehlen.

Bedenke :
- Die Standardfunktion ORD liefert für Argumente ordinalen Datentyps die zugehörige Ordnungszahl.
- MUL-Befehl des 80x86 : Multiplikand in AX, Multiplikator z.B. in BX; Produkt in DX (höherwertige Bits) und AX (niederwertige Bits)

Das PTL-Team wünscht viel Erfolg

Anlage :

- ASCII-Code
- Befehlssatz des 8088

```
Program Aufgabe1;

Uses Crt;

Const SpecialKey = #0;
      Return     = #13;

Type Zustaende = (Z0,Z1,Z2,Z3);

Var Zustand : Zustaende;
    Zahl    : Integer;
    Negativ : Boolean;
    Zeichen : Char;

Begin
  Zustand := Z0;
  Repeat
    Zeichen := ReadKey;
    If Zeichen = SpecialKey Then Begin
      Zeichen := ReadKey;
      Zeichen := SpecialKey
    End Else
      Case Zustand Of
        Z0    : Case Zeichen Of
                  '+'      : Begin
                               Zustand := Z1;
                               Negativ := False;
                               Write(Zeichen)
                             End;
                  '-'      : Begin
                               Zustand := Z2;
                               Negativ := True;
                               Write(Zeichen)
                             End;
                  '0'..'9' : Begin
                               Zustand := Z3;
                               Zahl := Ord(Zeichen)-Ord('0');
                               Negativ := False;
                               Write(Zeichen)
                             End
                End;
        Z1,Z2 : Case Zeichen Of
                  '0'..'9' : Begin
                               Zustand := Z3;
                               Zahl := Ord(Zeichen)-Ord('0');
                               Write(Zeichen)
                             End
                End;
        Z3    : Case Zeichen Of
                  '0'..'9' : Begin
                               Zahl := Zahl*10+
                                       Ord(Zeichen)-Ord('0');
                               Write(Zeichen)
                             End
                End
      End
  Until (Zeichen = Return) And (Zustand = Z3);
  If Negativ Then Zahl := -1*Zahl;
  Writeln(#13#10,Zahl)
End.
```

```
Program Aufgabe2;

Var Screen     : Array[1..25,1..80,1..2] Of Char
                   Absolute $B800:$0000;
    Zaehler    : Array[Char] Of Integer;
    MaxZaehler : Integer;
    MaxZeichen : Char;
    Zeile      : Integer;
    Spalte     : Integer;
    Zeichen    : Char;

Begin
  For Zeichen := #0 To #255 Do Zaehler[Zeichen] := 0;
  For Zeile := 1 To 25 Do
    For Spalte := 1 To 80 Do
      Inc(Zaehler[Screen[Zeile,Spalte,1]]);
  MaxZaehler := Zaehler[#0];
  MaxZeichen := #0;
  For Zeichen := #1 To #255 Do
    If Zaehler[Zeichen] > MaxZaehler Then Begin
      MaxZaehler := Zaehler[Zeichen];
      MaxZeichen := Zeichen
    End;
  Write('Spitzenreiter ist das Zeichen mit dem ASCII-Code ',
        Ord(MaxZeichen))
End.
```

Klausur IA10.3/12.0 451 (ex IA50) Assembler am 24.01.2000

Dauer : 120 Minuten **keine externen Hilfsmittel**

Übersetzen Sie das nachfolgende (siehe Rückseite) Pascal-Programm in ein äquivalentes Assemblerprogramm (8086, EXE).

Kommentieren Sie das Assemblerprogramm durch eindeutige Zuordnung der Pascal-Befehle zu den Assembler-Befehlen. Ohne derartige Kommentierung wird die Klausur nicht gewertet !

Falls Sie keine komplette Lösung angeben können, bearbeiten Sie Teilaspekte im Sinne der eindeutigen Zuordnung der Pascal-Befehle zu den Assembler-Befehlen.

Bedenke :
- Die Standardfunktion ORD liefert für Argumente ordinalen Datentyps die zugehörige Ordnungszahl.
- Multiplikationen mit (-1) lassen sich auch durch Zweierkomplementbildung realisieren.

- MUL-Befehl des 80x86 : Multiplikand in AX, Multiplikator z.B. in BX; Produkt in DX (höherwertige Bits) und AX (niederwertige Bits)

Das PTL-Team wünscht viel Erfolg

Anlage :

- ASCII-Code
- Befehlssatz des 8088

```
Program Klausur;

Uses Crt;

Const SpecialKey = #0;
      Return     = #13;
      Esc        = #27;

Var AC    : Array[1..25,1..80,1..2] Of Char Absolute $B800:$0000;
    S     : String;
    C     : Char;
    I1,I2 : Integer;

Procedure ClrScr;

Begin
  For I1 := 1 To 25 Do
    For I2 := 1 To 80 Do
      AC[I1,I2,1] := ' '
End;

Begin
  S := '';
  Repeat
    C := ReadKey;
    If C = SpecialKey Then Begin
      C := ReadKey;
      C := SpecialKey
    End Else Begin
      If UpCase(C) In ['A'..'Z'] Then Begin
        S := S+C;
        Write(C)
      End
    End
  Until (C = Return) Or (Ord(S[0]) = 40);
  ClrScr;
  C := ReadKey;
  While C <> Esc Do Begin
    I1 := 80;
    Repeat
      I2 := 1;
      While (I1+I2-1) <= 80 Do Begin
        If I1+I2-1 >= 1 Then Begin
          If I2 <= Ord(S[0]) Then
            C := S[I2]
          Else
            C := ' ';
          AC[10,I1+I2-1,1] := C
        End;
        I2 := I2+1
      End;
      I1 := I1-1;
      C := ReadKey
    Until (I1 = -1*ORD(S[0])) Or (C = Esc)
  End;
  ClrScr
End.
```

Klausur IA12.0 451 Assembler am 07.08.2000

Dauer : 120 Minuten **keine externen Hilfsmittel**

Übersetzen Sie das nachfolgende (siehe Rückseite) Pascal-Programm in ein äquivalentes Assemblerprogramm (8086, EXE).

Gemäß den Konventionen von Borland TurboPascal soll die Parameterübergabe (sowohl für Wert- als auch Referenzparameter) über den Stack, die Ablage lokaler Variablen auf dem Stack und die Rückgabe eines Integer-Funktionswerts über das AX-Register erfolgen.

Kommentieren Sie das Assemblerprogramm durch eindeutige Zuordnung der Pascal-Befehle zu den Assembler-Befehlen. Ohne derartige Kommentierung wird die Klausur nicht gewertet !

Im Modul STDIO steht die Routine DEZOUT zur Ausgabe von Integer-Zahlen zur Verfügung. Der Datentransfer erfolgt über das DX-Register.

Falls Sie keine komplette Lösung angeben können, bearbeiten Sie Teilaspekte im Sinne der eindeutigen Zuordnung der Pascal-Befehle zu den Assembler-Befehlen.

Das PTL-Team wünscht viel Erfolg

Anlage : Befehlssatz des 8088

```
Program Klausur;

Type myString = Array [0..10] of Char;

Const PString : myString =
                (#9 ,'A','s','s','e','m','b','l','e','r',' ');
      CString : myString =
                ('K','l','a','u','s','u','r',#0 ,' ',' ',' ',' ');

Var A,B,C : Integer;

Function PLength(Var S:myString):Integer;

Begin
  PLength := Ord(S[0])
End;

Function CLength(Var S:myString):Integer;

Var I : Integer;

Begin
  I := 0;
  While S[I] <> #0 Do I := I+1;
  CLength := I
End;

Function Mult(X,Y:Integer):Integer;

Begin
  If Y > 1 Then
    Mult := X+Mult(X,Y-1)
  Else
    Mult := X
End;

Begin
  A := PLength(PString);
  B := CLength(CString);
  C := Mult(A,B);
  WriteLn('Produkt beider Stringlaengen : ',C)
End.
```

Klausur IA12.0 451 Assembler am 22.01.2001

Dauer : 120 Minuten **keine externen Hilfsmittel**

Übersetzen Sie das nachfolgende Pascal-Programm *Aufgabe* in ein äquivalentes Assemblerprogramm (8086, EXE).

Gemäß den Konventionen von Borland TurboPascal soll die Parameterübergabe (sowohl für Wert- als auch Referenzparameter) über den Stack, die Ablage lokaler Variablen auf dem Stack und die Rückgabe eines Integer-Funktionswerts über das AX-Register erfolgen.

Kommentieren Sie das Assemblerprogramm durch eindeutige Zuordnung der Pascal-Befehle zu den Assembler-Befehlen. Ohne derartige Kommentierung wird die Klausur nicht gewertet !

Im Modul STDIO steht die Routine DEZOUT zur Ausgabe von Integer-Zahlen zur Verfügung. Der Datentransfer erfolgt über das DX-Register.

Falls Sie keine komplette Lösung angeben können, bearbeiten Sie Teilaspekte im Sinne der eindeutigen Zuordnung der Pascal-Befehle zu den Assembler-Befehlen.

Das PTL-Team wünscht viel Erfolg

Anlage : Befehlssatz des 8088

```
Program Aufgabe;

Type TWochentage    = (Montag,Dienstag,Mittwoch,Donnerstag,Freitag,
                       Sonnabend,Sonntag);
     TArbeitszeiten = Array[TWochentage] Of Byte;

Const Arbeit : TArbeitszeiten = (6,8,11,8,4,1,2);

Var Blautag : TWochentage;
    Keultag : TWochentage;

Procedure SchreibeWochentag(X:TWochentage);

Begin
  Case X Of
    Montag     : Write('Montag');
    Dienstag   : Write('Dienstag');
    Mittwoch   : Write('Mittwoch');
    Donnerstag : Write('Donnerstag');
    Freitag    : Write('Freitag');
    Sonnabend  : Write('Sonnabend');
    Sonntag    : Write('Sonntag')
  End
End;

Function BerechneWochenArbeitszeit(Var X   : TArbeitszeiten;
                                   Var Min : TWochentage;
                                   Var Max : TWochentage) : Integer;

Var Index  : TWochentage;
    Result : Integer;

Begin
  Result := 0;
  Min := Montag;
  Max := Montag;
  For Index := Montag To Sonntag Do Begin
    Result := Result + X[Index];
    If X[Index] < X[Min] Then Min := Index;
    If X[Index] > X[Max] Then Max := Index
  End;
  BerechneWochenarbeitszeit := Result
End;

Begin
  WriteLn('Wochenarbeitszeit : ',
          BerechneWochenArbeitszeit(Arbeit,Blautag,Keultag),
          ' Stunden');
  Write('Ausgeruht am ');
  SchreibeWochentag(Blautag);
  WriteLn;
  Write('Reingehauen am ');
  SchreibeWochentag(Keultag);
  WriteLn
End.
```