Physikalisch-Technische Lehranstalt Wedel

Staatsexamen für Technische Assistenten SoSe2003
schriftliche Prüfung im Fach Prozesstechnik
gewählter Vorschlag

Dauer : 180 Minuten **keine externen Hilfsmittel**

Aufgabe :

Den einzelnen Programmen in der Prozessdatenverarbeitung werden zumeist Prozessnamen und -nummern zugeordnet. Zur einfachen Übersetzung derartiger Prozessnamen in Prozessnummern können hashartige Datenstrukturen dienen. Entwickeln Sie daher auf Basis des 8086-Prozessors einen simplen Prototypen für Hash-Algorithmen durch möglichst bedeutungstreue Übersetzung des Pascal-Programms *VorschlagEins*.

Zur Realisierung der New-Prozedur und der Dispose-Prozedur werden die DOS-Funktionen "Allocate Memory" (Funktionscode 48h) und "Free Memory" (Funktionscode 49h) verwendet. Die Funktion zum Reservieren von Speicherplatz erwartet im BX-Register die Anzahl der benötigten zusammenhängenden 16-Byte Blöcke und liefert als Ergebnis im AX-Register die Segmentadresse der reservierten Blöcke (als Offsetadresse innerhalb des Segments wird Null impliziert). Der Fall "nicht genügend zur Verfügung stehender Speicher" bleibt unberücksichtigt. Die Funktion zum Freigeben von Speicherplatz erwartet im ES-Register die Segmentadresse der freizugebenden Blöcke, also eine Adresse die von der Funktion "Allocate Memory" zurückgeliefert wurde.

Im Modul *STDIO* steht die Routine *PSTRIN* zur Eingabe von Zeichenketten im Format der Programmiersprache Pascal zur Verfügung. Der Datentransfer (Speicherbereich als Referenzparameter) erfolgt nicht über den Stack sondern über das DS:DX-Registerpaar.

Vergessen Sie nicht die hinreichende Kommentierung Ihres Assembler-Programms z.B. durch Zuordnung der Pascal-Befehle zu den Assembler-Befehlen.

Das PTL-Team wünscht viel Erfolg

```
Program VorschlagEins;

Type tString   = String[127];
     tpContent = ^tContent;
     tContent  = Record
                   Key   : tString;
                   Value : Word;
                   Next  : tpContent
                 End;
     tHash     = Array['A'..'Z'] Of tpContent;

Const EOF : tString = 'EOF*EOF';

Var Hash : tHash;
    P    : tpContent;
    S    : tString;
    W    : Word;

Procedure MakeHash(Var H:tHash);
Var C : Char;
Begin
  For C := 'A' to 'Z' Do H[C] := Nil
End;

Procedure InsertHash(Var H:tHash;P:tpContent);
Var C : Char;
Begin
  C := P^.Key[1];
  P^.Next := H[C];
  H[C] := P
End;

Function FindHash(Var H:tHash;S:tString):tpContent;
Var P : tpContent;
Begin
  FindHash := Nil;
  P := H[S[1]];
  While P <> Nil Do
    If P^.Key = S Then Begin
      FindHash := P;
      P := Nil
    End Else
      P := P^.Next
End;

Procedure DestroyHash(Var H:tHash);
Var C : Char;
    P : tpContent;
Begin
  For C := 'A' to 'Z' Do
    While H[C] <> Nil Do Begin
      P := H[C];
      H[C] := H[C]^.Next;
      Dispose(P)
    End
End;

Function MakeItem(S:tString;W:Word):tpContent;
Var P : tpContent;
Begin
  New(P);
  P^.Key := S;
  P^.Value := W;
  MakeItem := P
End;
```

```
Procedure GetString(Var S:tString);
Begin
  Repeat
    ReadLn(S)
  Until (Byte(S[0]) > 0) And (S[1] >= 'A') And (S[1] <= 'Z')
End;

Procedure ShowWord(W:Word);
Begin
  If W >= 10 Then
    ShowWord(W Div 10);
  Write(Char(W Mod 10 + Byte('0')))
End;

Begin
  MakeHash(Hash);
  W := 1;
  Repeat
    GetString(S);
    If S <> EOF Then Begin
      If FindHash(Hash,S) = Nil Then Begin
        P := MakeItem(S,W);
        InsertHash(Hash,P);
        W := W+1
      End
    End
  Until S = EOF;
  Repeat
    GetString(S);
    If S <> EOF Then Begin
      P := FindHash(Hash,S);
      If P <> Nil Then Begin
        ShowWord(P^.Value);
        Write(#13#10)
      End
    End
  Until S = EOF;
  DestroyHash(Hash)
End.
```