Testfragen und Aufgaben zur Speicherbenutzung

1. Überlegen Sie ob die folgenden Befehle korrekt sind:

```
.DATA
Zaehler1 DB ?
Zaehler2 DB 0
Endechar DB ?
Startchar DB 'A'
Pixelx DW ?
Pixely DW 01FFh
Schluessel DD 1200h
.CODE
mov Zaehler1, 100h   ; Konstante zu gross für 8-Bit-Variable
mov Zaehler2, ax     ; Fehlermeldung - zaehler2: 8 Bit, AX: 16 Bit

mov ah,2
mov dx, Startchar    ; Fehlermeldung - Startchar: 8 Bit, DX: 16 Bit
int 21h

movzx Endechar,'Q'   ; movzx wird nicht gebraucht, zwei 8-Bit-Operanden
mov edx,Startchar    ; statt mov muss movzx benutzt werden
xchg Pixely,cx       ; o.k.

mov schluessel,ebp   ; o.k.
mov Pixelx,Pixely    ; geht nicht,zwei Speicheroperanden in einem Befehl
```

2. Überlegen Sie welche der folgenden Befehle zu Fehlermeldungen, Warnungen oder Laufzeitfehlern führen:

```
.DATA
Nummer DB 25 DUP (0)
zahl DW 0
.CODE
.386
mov [Nummer+cx],al     ; mit cx kann nicht adressiert werden

mov [Nummer+ecx],al    ; o.k., da beliebige 32-Bit Register erlaubt

mov al,[Nummer+bl]     ; mit 8-Bit-Registern kann nie adressiert werden

mov [bx+bp+10],0       ; nicht erlaubt, zwei Basisregister

mov [si+di+1],10h      ; nicht erlaubt, zwei Indexregister

mov bx, offset zahl
mov cl, [Nummer+bx]    ; Laufzeitfehler, Adresse liegt
                       ; hinter dem Feld Nummer

mov cl, Nummer         ; o.k., überträgt das erste Byte von Nummer

inc [bx]
```

3. Wie ist der Inhalt der Register ax,cx,edx,esi nach der Ausführung der folgenden Befehle?

```
.DATA
Bytevar       DB 66h
wfeld         DW 711h,822h,933h
dwort         DD 12001300h
.CODE
mov bx,offset bytevar
mov ax,[bx]
mov di,2
mov cx,[bx+di]
inc di
mov edx,[bx+di]
add di,3
mov esi,[bx+di]
```

Speicher: 66 11 07 22 08 33 09 00 13 00 12

ax=1166h, cx=2207h, edx=09330822h, esi=00130009h

4. Die vier Speichervariablen a,b,c,d im folgenden Codeabschnitt sind vom Typ int, irgendeine Form eines Zeigers auf int. Welchen Typ haben die vier Variablen? gehört der Codeabschnitt zu einer 16-Bit oder 32-Bit-Umgebung?

```
mov eax,a
shl eax,2
mov ebx,b
mov [ebx],eax
mov edi,c
mov edi,[edi]
mov [edi],edx
mov d,ebx
```

int a; int *b, *d; int **c;