4.1.F Datenstrukturen/Zeiger – Ergänzungen und Bilder

4.1.F.1 Beispiel für Zeigervariable (Darstellung A)

Das folgende Programm wird in Abbildung 1 dargestellt.

```
1  int k;
2  int *pm;
3  int *pn;
4  pm = NULL;
5  pm = &k;
6  pn = pm;
7  *pn = 80;
```

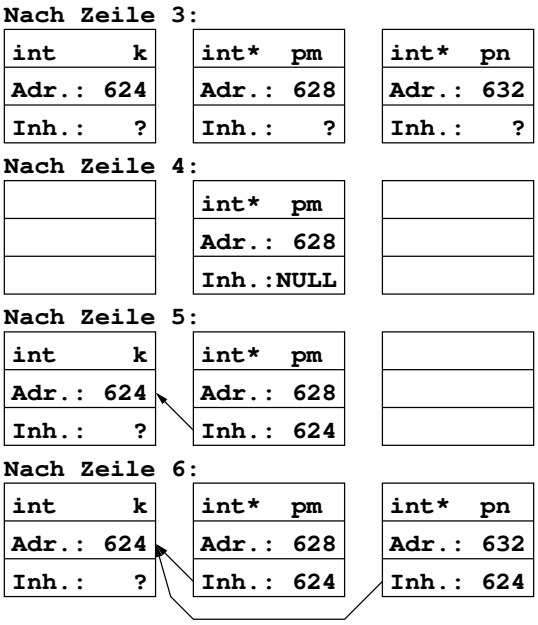

Abbildung 1: Zeigervariablen, graphisch dargestellt

4.1.F.2 Beispiel für Zeigervariable (Darstellung B)

Das gleiche Programm wird in Abbildung 2 in anderer Form dargestellt.

```
1   int k;
2   int *pm;
3   int *pn;
4   pm = NULL;
5   pm = &k;
6   pn = pm;
7   *pn = 80;
```

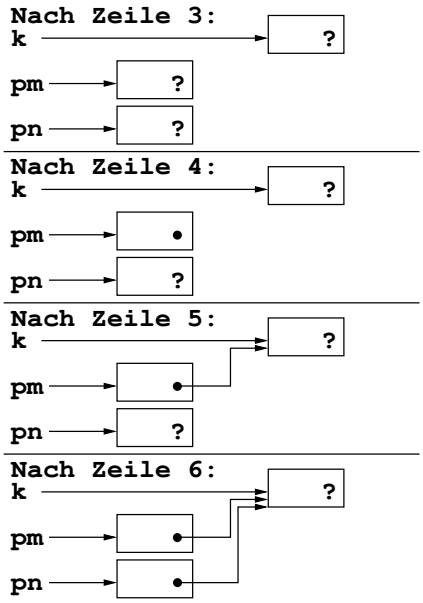

Abbildung 2: Zeigervariablen, anders dargestellt

4.1.F.3 Beispiel für Zeigervariable (Darstellung C)

Abbildung 3 zeigt wieder das gleiche Programm in einer dritten Weise.

```
1   int k;
2   int *pm;
3   int *pn;
4   pm = NULL;
5   pm = &k;
6   pn = pm;
7   *pn = 80;
```

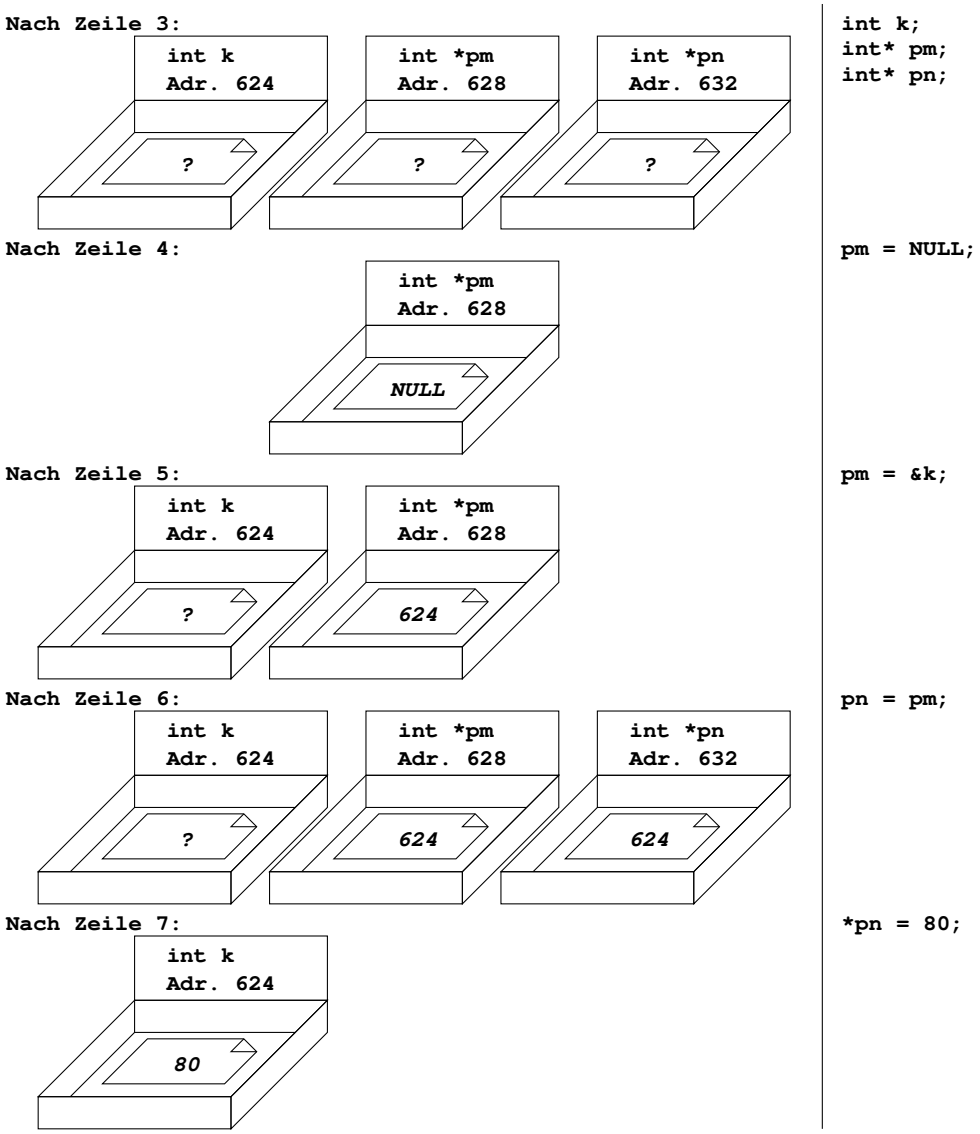

Abbildung 3: Zeigervariablen, wieder anders dargestellt

4.1.F.4 Inhalts-Operator und indirekte Adressierung

a) Absolute Adressierung:
Der Operand steht im Programm (Abbildung 4)

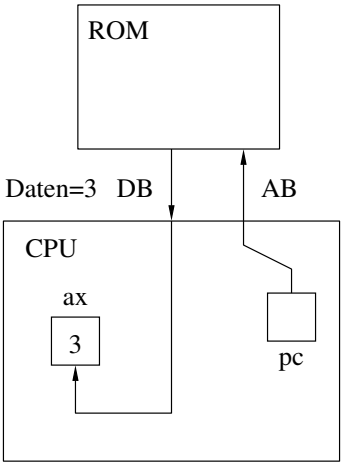

Abbildung 4: Absolute Adressierung

```
a=3;          // lade Konstante 3 in Variable a
mov ax, 3      ; lade Konstante 3 in Register ax
```

b) Register-direkte Adressierung:
Der Operand steht in dem Register, das im Befehl genannt ist (Abbildung 5)

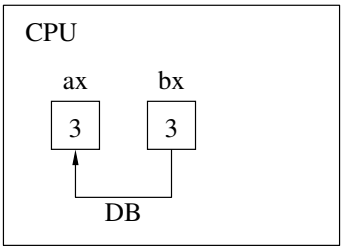

Abbildung 5: Register-direkte Adressierung

```
a=b;          // lade Inhalt von Variable b in Variable a
mov ax, bx     ; lade Inhalt von Register bx in ax
```

c) Register-indirekte Adressierung:
Der Operand steht in dem RAM-Datenwort, dessen Adresse im (im Befehl genannten) Register liegt (Abbildung 6)

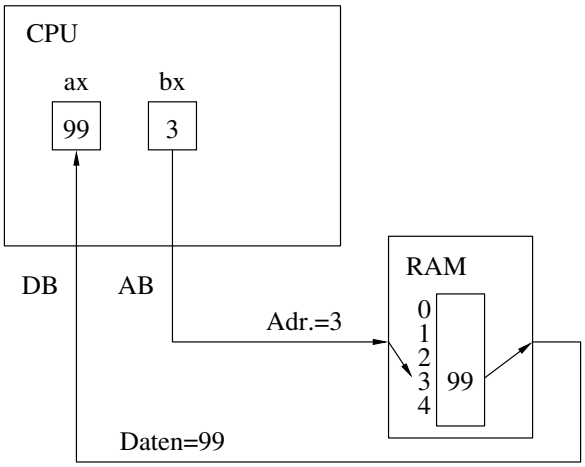

Abbildung 6: Register-indirekte Adressierung

```
a=*b;          // lade Inhalt der Var., deren Adresse in b steht
mov ax, [bx]    ; lade Inhalt des RAM-Datenworts,
                ; dessen Adresse in bx steht
```

d) Speicher-direkte Adressierung: Der Operand steht in dem RAM-Datenwort, dessen Adresse im Befehl angegeben ist (Abbildung 7)

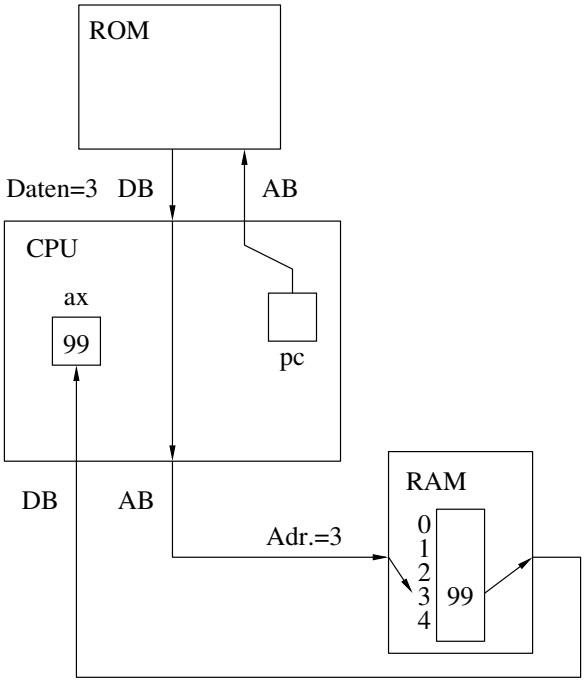

Abbildung 7: Speicher-direkte Adressierung

```
a=b;         // oder a= *3, falls b an Adresse 3 steht
mov ax, [3]  ; lade Inhalt des RAM-Datenworts mit der Adresse 3
```

e) Speicher-indirekte Adressierung: Der Operand steht in dem RAM-Datenwort, dessen Adresse in dem Datenwort steht, dessen Adresse im Befehl gegeben ist (Abbildungen 8 und 9):

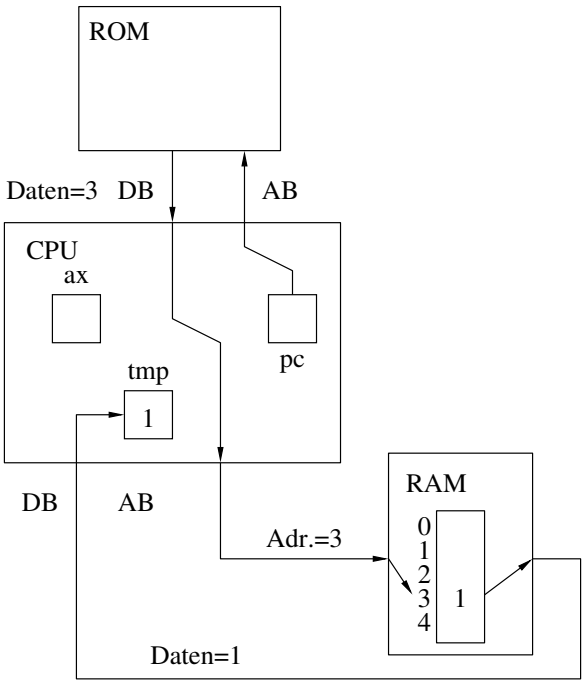

Abbildung 8: Speicher-indirekte Adressierung, Schritt 1

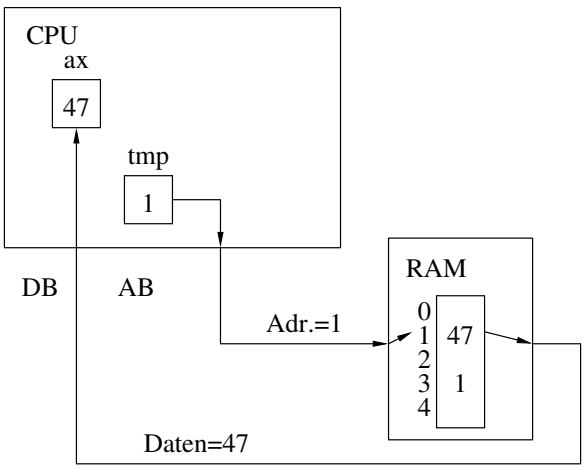

Abbildung 9: Speicher-indirekte Adressierung, Schritt 2

```
a=*b;         // oder a= **3, falls b an Adresse 3 liegt
mov ax, [[3]] ; lade Inhalt des RAM-Datenworts, dessen Adresse im
              ; RAM-Datenwort mit der Adresse 3 liegt.
```