3AA

Indirekte Adressierung

Prof. Dr. Wolfgang P. Kowalk
Universität Oldenburg

WS 2005/2006

31.10.2005

3AA

- Stand des Wissens über 3AA
  - Kommentare

```
1              // Dieses ist ein Kommentar
2
3              nop  // hier wird nichts getan
4 Start        stop // hier halten wir an
```

  - Speicher
    - Adressen, durchnummeriert: 0, 1, 2, 3, ...
    - Jede Adresse enthält
      - Anweisung oder
      - Wert

3AA

- Stand des Wissens über 3AA
  - Werte (*value*)
    - Immer Zahlenwerte, verschiedene Interpretationen
      - ganze Zahlen: 0, 1, 2, .., -1, -2, es gibt größte/kleinste Zahl
      - Gleitpunktzahlen: 0.0, 0.1, -0.3, 1.2e12 ($=1.2 \cdot 10^{12}$)
      - Zeichen: 'A', ' ', 'a', '%', '☻', 'ω', 'И'. (Unicode)
      - Texte: „Antonio und Berta gehen ins Cinema.“

3AA

- Stand des Wissens über 3AA
  - Werte (*value*)
    - Konstante (*constant*)
      - Wert, der sich nicht ändert
      - Compilerkonstante (Adresse)
      - Programmkonstante (Wert wird von Programmierer explizit festgelegt)

```
1              // Konstante in 3AA
2              nop
3 Start        stop
4 CompilerKonstante con 9      // belegt Speicher, Wert = 9
5 ProgrammKonstante := con 9 //  belegt keinen Speicherplatz!!
6 CompilerKonstante2 con 9     // belegt Speicher, Wert = 9
```

    - Variable (*variable*)
      - Wert, der sich verändert
      - Wert steht (an fester Stelle) im Speicher
      - nur über Speicheradresse (`CompilerKonstante!!`) erreichbar

3AA

- Stand des Wissens
  - Anweisungen (*statement*)
    - Rechnen mit Zahlenwerten, +, +f, –, –f usw.
  - Spezielle Anweisungen
    - **`nop`**
    - **`stop`**
    - **`step`**
    - **`toInt`**
    - **`toFloat`**
    - **`repaint`**
    - **`rewrite`**
    - **`Weitere Anweisung (siehe Gleitpunkt)`**

3AA

- Stand des Wissens
  - Sprünge (**`goto`**) (auch *jump* usw.)
    - unbedingte Sprünge: **`goto`** ...
    - bedingte Sprünge: **`if ... then goto ...`**
  - Schleifen (*loop*)
    - Gleiche Anweisungsfolge wird mehrfach ausgeführt
    - durch Sprünge konstruierbar
  - Adressrechnung: Berechne Adresse eines Datums!

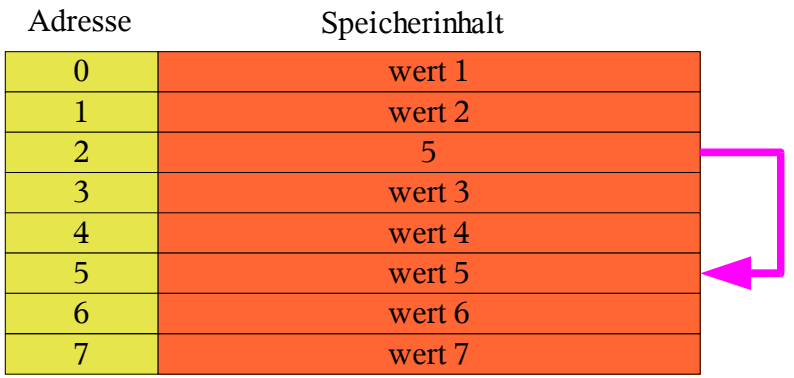
3AA

- Adressrechnung
  - Berechne Adresse eines Datums!
  - Adressen werden wie Daten behandelt:
    - kopieren
    - addieren, subtrahieren usw.
  - erhöht Flexibilität
  - unabdingbar für Mächtigkeit der Programmiersprache
  - Indirekte Adressierung in 3AA
    - `adr adr Ziel := val val Quelle`
    - `adr val Ziel := val val Quelle`

      // Alternative, äquivalent
    - „Wert unter einer Adresse, die sich aus Wert unter einer Adresse bestimmt“

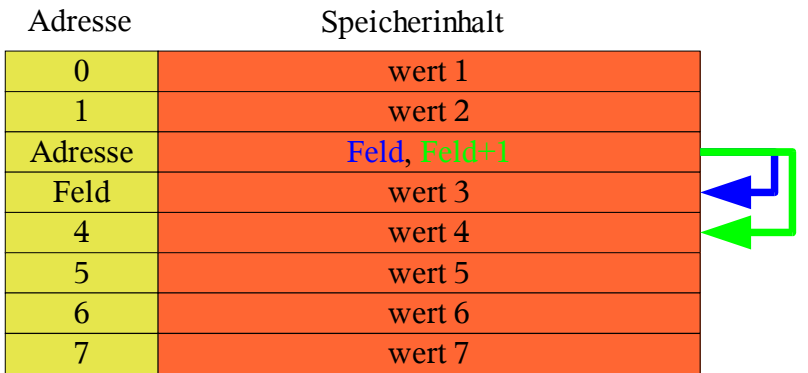
3AA

- Indirekte Adressierung in 3AA
  - `adr adr Ziel := val val Quelle`
  - `adr val Ziel := val val Quelle`
    // Alternative, äquivalent
  - „Wert unter einer Adresse, die sich aus Wert unter einer Adresse bestimmt“



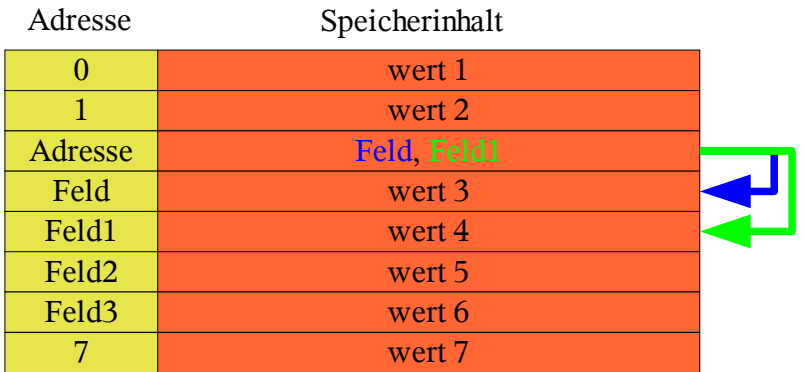
3AA

```
 1                // Indirekte Adressierung, Lesen
 2
 3                nop
 4                adr Adresse := con Feld
 5                adr Wert := val val Adresse
 6                adr Adresse += con 1
 7                adr Wert := val val Adresse
 8                stop
 9
10 Wert
11 Adresse       // enthält Adresse e. Feld-Elemnts
12 Feld          con 1
13               con 2
14               con 3
15               con 4
```



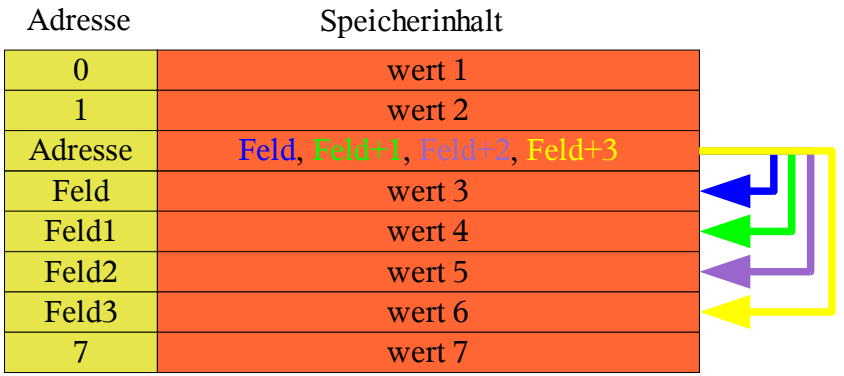
3AA

```
 1              // Indirekte Adressierung, Schreiben
 2
 3              nop
 5              adr val Adresse := con 1
 6              adr Adresse += con 1
 5              adr val Adresse := con 2
 6              stop
 8
 9
10 Wert
11 Adresse     con Feld // Startadresse des Feldes
12 Feld
13 Feld1
14 Feld2
15 Feld3
```



3AA

```
 1              // Indirekte Adressierung, Schleife
 2
 3              nop
 4 Start        adr val Adresse := val Wert
 5              adr Adresse += con 1
 6              adr Wert += con 1
 7              goto con Start
 8
 9 Wert         con 1
10 Adresse      con Feld // Startadresse des Feldes
11 Feld
12 Feld1
13 Feld2
14 Feld3
```



3AA

```
 1              // Indirekte Adressierung, Textlesen
 2
 3              nop
 4 Start        adr Wert := val val Adresse
 5              adr Adresse += con 1
 6    if val Adresse < con TextEnde then goto con Start
 7              stop
 8
 9 Wert         con "%"
10 Adresse      con Text // Startadresse des Textes
11 Text         con "Hallo Welt!"
12 TextEnde
```

3AA

```
 1            // Indirekte Adressierung, UNICODE
 2
 3            nop
 4 Start      adr val Adresse := val Wert
 5            adr Adresse += con 1
 6            adr Wert += con 1
 7       if val Wert < con 128 then goto con Start
 8            stop
 9
10 Wert       con " "
11 Adresse    con Text // Startadresse des Textes
12 Text       con "                                  "
13 TextEnde
```

3AA

```
 1            // Indirekte Adressierung, Textkopieren
 2
 3            nop
 4 Start      adr val nach := val val von
 5            adr nach += con 1
 6            adr von += con 1
 7      if val von < con Nach then goto con Start
 8            stop
 9
10 von        con Text // Startadresse des Textes
11 nach       con Nach // Zieladresse des Textes
12 Text       con "Hänsel und Gretel"
13 Nach       con "xxxxxxxxxxxxxxxxx"
```

3AA

```
 1            // Indirekte Adressierung, Zeichen
 2
 3            nop
 4 Start      adr val nach := val val von
 5            adr nach += con 1
 6            adr von += con 1
 7      if val val von <> con ':' then goto con Start
 8            stop
 9
10 von        con Text // Startadresse des Textes
11 nach       con Nach // Zieladresse des Textes
12 Text       con "Hänsel und Gretel: Die böse Hexe"
13 Nach       con "                              "
```