3AA

Prof. Dr. Wolfgang P. Kowalk

Universität Oldenburg

WS 2005/2006

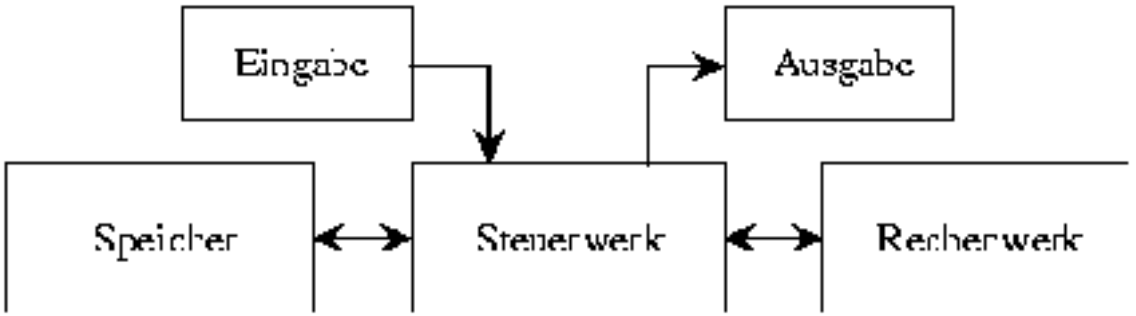
Version vom 24.10.2005

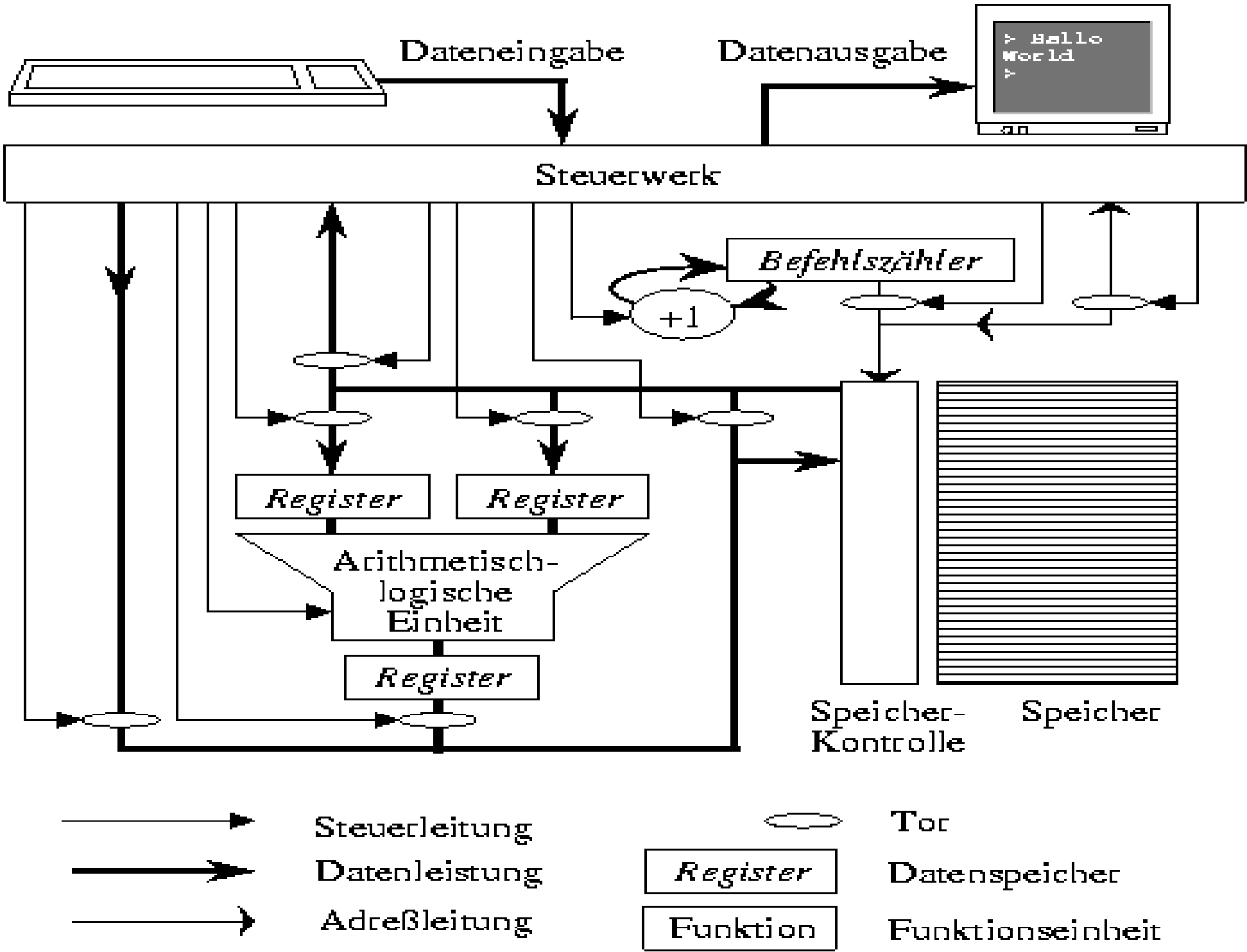
Übersicht

- Einführung in maschinennahe Programmierung
  - Verständnis für grundlegende Vorgänge im Computer
    - Jedes Programm wird durch einzelne Anweisungen ausgeführt
    - Reihenfolge der Programmausführung festgelegt durch
      - Reihenfolge der Notation
      - Explizite Änderung der Reihenfolge durch 'Sprünge' (*goto*)
  - Zeigt
    - alle Probleme mit elementarem System lösbar
    - Grenzen der Strukturierbarkeit

Dreiadress-Assembler (3AA)

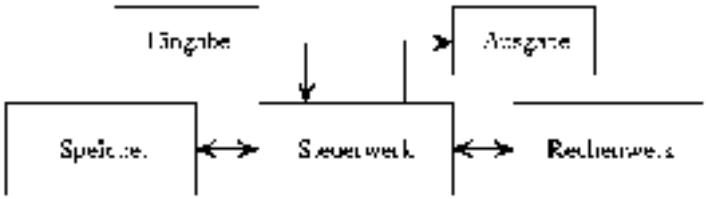
- Grundkonzept eines Computers
  - Meist v. Neumann-Rechner genannt
    - margittai János Lajos Neumann (1903-1957)
    - Bestandteile eines Computers
      - Eingabewerk
      - Ausgabewerk
      - Speicher
      - Steuerwerk
      - Rechenwerk




Dateneingabe
Datenausgabe
> Hallo
World
>
Steuerwerk
Befehlszähler
+1
Register
Register
Arithmetisch-
logische
Einheit
Register
Speicher-
Kontrolle
Speicher
Steuerleitung
Datenleistung
Adreßleitung
Tor
Register
Datenspeicher
Funktion
Funktionseinheit

Dreiadress-Assembler (3AA)

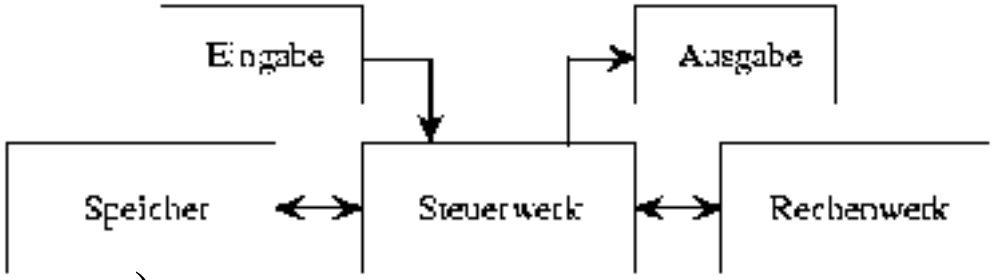
- Grundkonzept eines Computers



  - Speicher
    - in adressierbare (0, 1, 2, ..) Zellen eingeteilt
    - enthält
      - Daten (Zahlenwerte, Zeichen als Zahlen dargestellt)
      - Anweisungen (Befehle, Instruktionen, ...)
    - Speicherinhalt kann verändert werden
      - Daten werden regelmäßig verändert
      - Anweisungen (Programme) werden (i.d.R.) nie verändert!
        (bei v. Neuman durchaus noch angedacht)

Dreiadress-Assembler (3AA)

- Grundkonzept eines Computers (v. Neuman)
  - Prozessor
    - Rechenwerk
      - Kann 'rechnen'
        (negieren, addieren, multiplizieren, ...)
      - Logische Berechnungen (logisch UND, logisch Oder, ...)
      - Organisieren (kopieren, löschen, initialisieren, ...)
    - Steuerwerk
      - Liest Befehle aus Speicher in Steuerwerk
      - Liest Daten aus Speicher in Rechenwerk
      - Lässt Befehle ausführen
      - Schreibt Ergebnisse aus Rechenwerk in Speicher zurück
      - Liest nächsten Befehl, usw.



Dreiadress-Assembler (3AA)

- Dreiadress-Assembler (3AA)
  - maschinennahe Programmierung
    - einfache mathematische Operationen
    - mehrerer Operationen nacheinander ausführen
    - Ausführungsreihenfolge ändern
  - Verständnis der Grundfunktionalität eines Computers
    - Mächtigkeit
    - Komplexität
    - bessere Strukturierung sinnvoll
  - Schwerpunkt liegt auf Funktionalität
    - Dokumentation
    - Tutorium

Dreiadress-Assembler (3AA)

- Dreiadress-Assembler (3AA)
  - Modell eines Assemblers
    - Speicherzellen enthalten
      - Daten oder
      - Anweisungen
    - Daten sind
      - Ganze Zahlen
      - (Zeichen)
      - (Gleitpunktzahlen)
    - Anweisungen
      - Berechnen Werte
      - Speichern Werte
      - Legen Ausführungsreihenfolge fest

Dreiadress-Assembler (3AA)

- Dreiadress-Assembler (3AA)
  - Modell eines Assemblers (2)
    - Beispiel

```
1              // Beispielprogramm
2              nop
3              adr wert1 := con 123
4              adr wert2 := val wert1
5              adr summe := val wert1 + val wert2
6              stop
7 wert1        con 3
8 wert2        con 5
9 summe        con 0
```

Dreiadress-Assembler (3AA)

- Formale Syntaxbeschreibung (1)
  - Ein Programm besteht aus einer Folge von Zeilen
  - Jede Zeile ist leer oder enthält eine Instruktionen
  - Jede Instruktion steht in einer eigenen Zeile
  - Jede Instruktion besteht aus
    – einer optionalen Zeilennummer,
    – einer optionalen Marke und
    – einer Programmanweisung
  - Eine Programmanweisung ist entweder
    – ein Befehl (auch Prozessoranweisung) oder
    – eine Speicheranweisung oder
    – eine Compileranweisung

Dreiadress-Assembler (3AA)

- Formale Syntaxbeschreibung (2)
  - Eine Compileranweisung besteht aus
    - einer Marke,
    - einem Wertzuweisungszeichen und
    - einem Ausdruck
  - compiler instruction :: label ':=' expression
  - wobei der Ausdruck während der Compilezeit ausgewertet werden können muss. Der Wert der Marke wird der Wert dieses Ausdrucks

Dreiadress-Assembler (3AA)

- Formale Syntaxbeschreibung (3)
  - Eine Speicheranweisung besteht aus
    - einer optionalen Marke und
    - einem Ausdruck.
  - memory instruction :: [label] expression
    - wobei der Ausdruck während der Compilezeit ausgewertet werden können muss.
    - Der Wert in dieser Speicherstelle wird der Wert des Ausdrucks.
    - Der Wert des Labels wird der Programmzählerwert dieser Speicherstelle.
  - Eine Marke (label) ist ein Name
  - label :: letter (letter | cipher)*
  - letter :: a, b, c, ..., y, z, A, B, ..., Y, Z
  - cipher :: 0, 1, 2, ... 8, 9

Dreiadress-Assembler (3AA)

- Formale Syntaxbeschreibung (4)
  - Ein Befehl ist eine der folgenden Anweisungen
    - processor instruction :: assignment
    - processor instruction :: goto-instruction
    - processor instruction :: jsr-instruction
    - processor instruction :: proc-instruction
    - processor instruction :: conditional-goto-instruction
    - processor instruction :: repaint-instruction
    - processor instruction :: rewrite-instruction
    - processor instruction :: nop-instruction
    - processor instruction :: step-instruction
    - processor instruction :: stop-instruction
    - processor instruction :: toInt-instruction
    - processor instruction :: toFloat-instruction

Dreiadress-Assembler (3AA)

- Formale Syntaxbeschreibung (5)
  - Eine Zuweisung besteht aus
    - einer Zuweisungsvariablen,
    - einem Zuweisungssymbol und
    - einem Ausdruck.
  - assignment :: assignment-variable := expression
  - Während der Programmausführung wird der Wert des Ausdrucks berechnet und an dem jeweiligen Speicherplatz abgelegt. Danach wird die nächste Prozessoranweisung ausgeführt.

Dreiadress-Assembler (3AA)

- Formale Syntaxbeschreibung (6)
  - Eine Zuweisungsvariable besteht aus einem oder zwei Schlüsselwörtern '**`adr`**' und einem Variablennamen.
  - assignment-variable :: [adr] adr name
  - Dabei bedeutet 'adr name': Speichere den errechneten Wert unter der Adresse der Compilerkonstanten 'name'.
  - Die Bedeutung von 'adr adr name' ist: Speichere den errechneten Wert unter der Adresse, deren Wert in der Compilerkonstanten 'name' steht. Dieses wird als indirekte Adressierung bezeichnet.
    Es ist auch 'adr val name' mit der gleichen Bedeutung erlaubt.

Dreiadress-Assembler (3AA)

- Formale Syntaxbeschreibung (7)
  - Ein Ausdruck besteht aus einem oder zwei Operanden und bei mehreren Operanden aus einem der folgenden Operatoren

    - expression :: operand
    - expression :: operand [+ | - | * | / | % | >> | >>> | << | '|' | & | ^ | ~ | +f | -f | *f | /f] operand

  - mit den üblichen Bedeutungen: addiere, subtrahiere, multipliziere usw.
  - Die Operatoren +f, -f, *f, /f verwenden Gleitpunktzahlen als Operanden.

Dreiadress-Assembler (3AA)

- Formale Syntaxbeschreibung (8)
  - Operanden bestehen entweder aus dem Schlüsselwort 'con' oder aus einem oder zwei Schlüsselwörtern 'val', gefolgt von einem Namen.
  - operand :: (con | val | val val) name
  - 'con name': Das Ergebnis ist der Wert der Kompilerkonstanten 'name'
  - 'val name': Das Ergebnis ist der Wert, der unter der Adresse der Compilerkonstanten 'name' gefunden wird.
  - 'val val name': Das Ergebnis ist der Wert, der unter der Adresse gefunden wird, die im Speicher unter der Kompilerkonstanten 'name' gefunden wird.
  - Der besondere Name PROGRAMCOUNT liefert jeweils den Wert des aktuellen Programmzählers. Er kann beispielsweise für die relative Adressierung verwendet werden.

Dreiadress-Assembler (3AA)

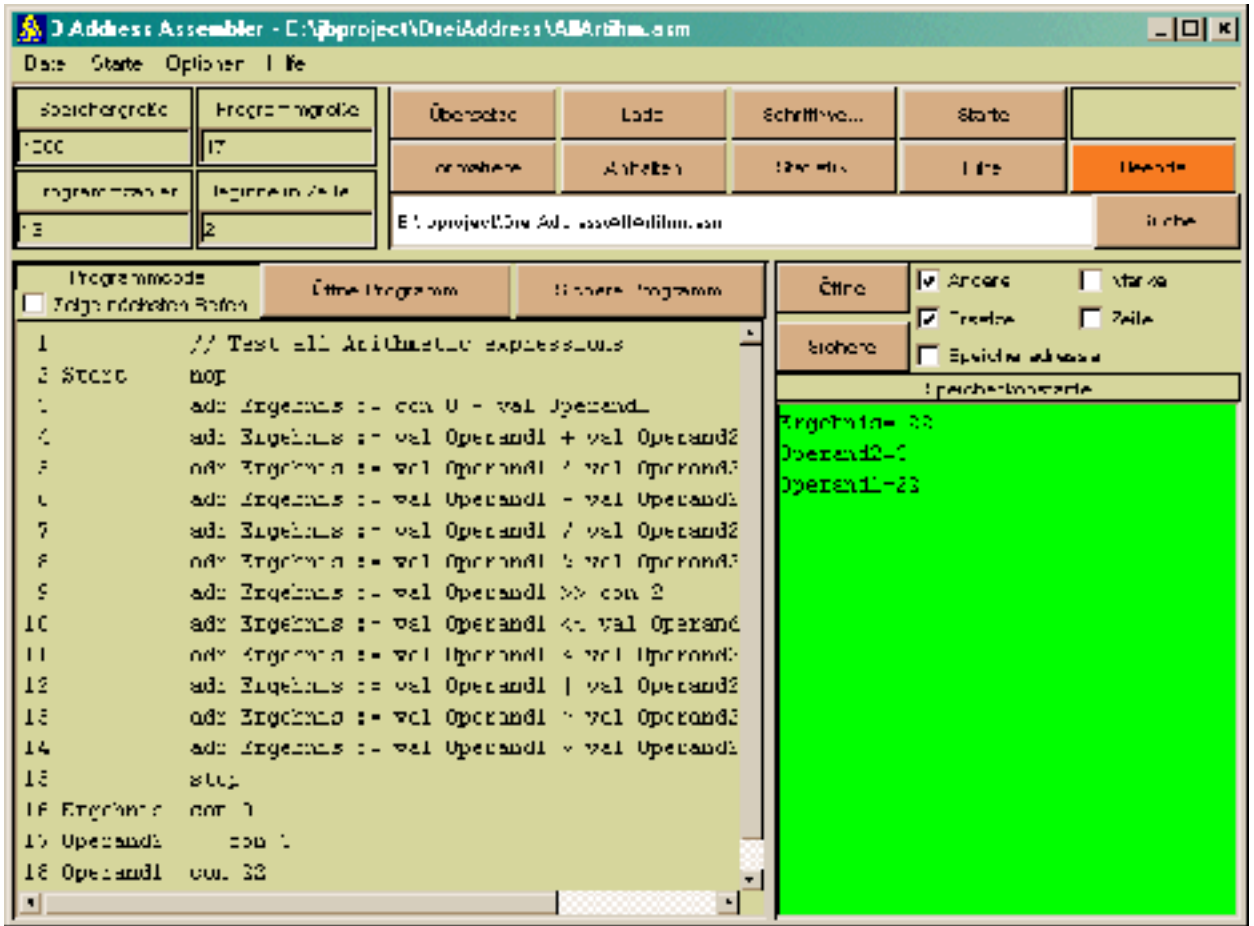
- Formale Syntaxbeschreibung (9)
  - Sprunganweisung besteht aus Schlüsselwort 'goto', gefolgt von einem Operanden.
  - goto-instruction :: goto operand
  - Die Ausführung dieser Anweisung setzt den Programmzähler auf den Wert des Operanden. Die als nächste auszuführende Anweisung steht also an der Speicherstelle: 'operand'.

Dreiadress-Assembler (3AA)

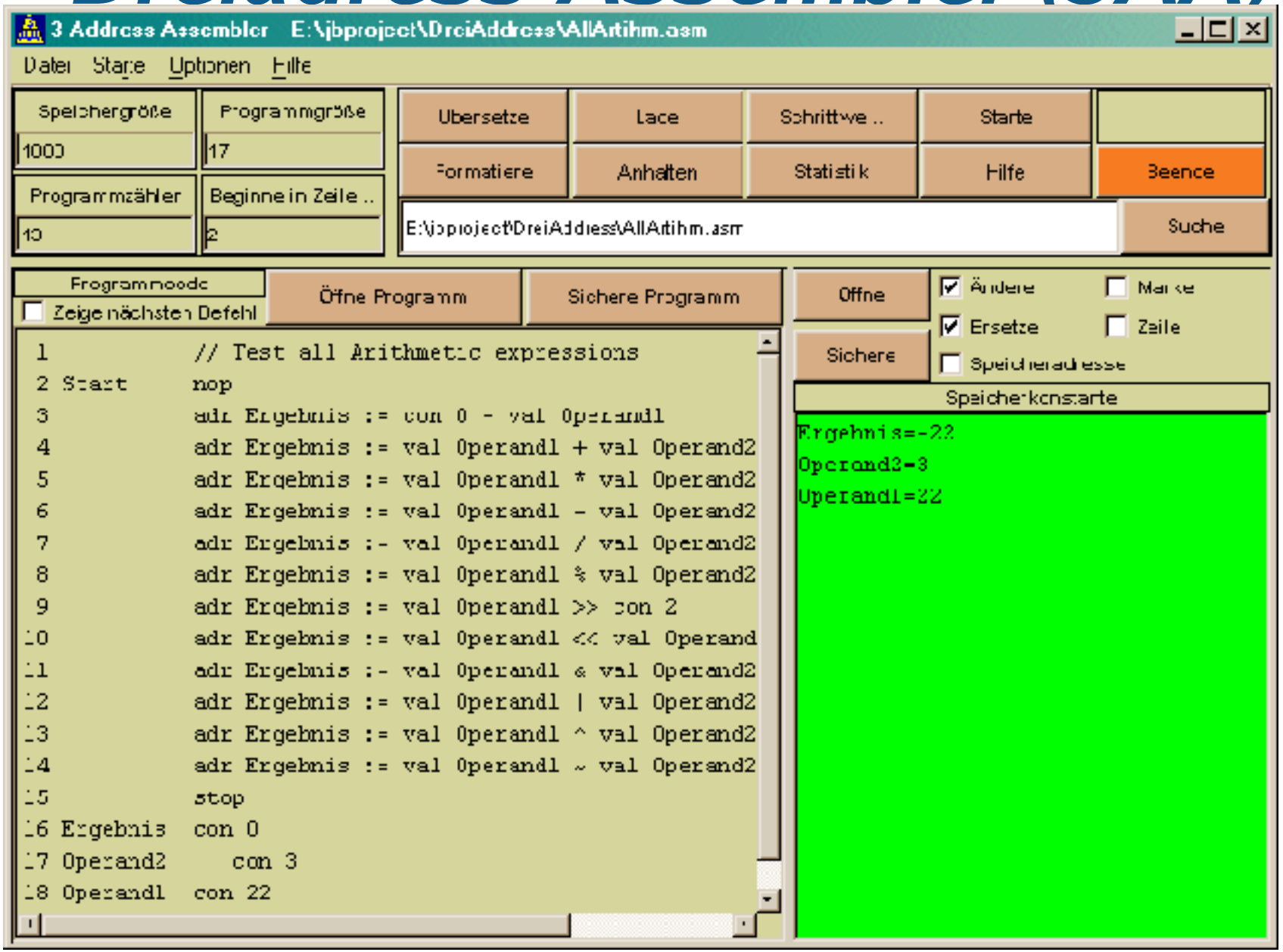
- Formale Syntaxbeschreibung (10)
  - Eine bedingte Sprunganweisung besteht aus dem Schlüsselwort 'if', einem logischen Ausdruck, dem Schlüsselwort 'then goto' und einem Operanden.
  - conditional goto-instruction ::
    if booleanExpression then goto operand
  - Ein logischer Ausdruck (*booleanExpression*) besteht aus einem Operanden, einem Vergleichssymbole und einem weiteren Operanden.
  - booleanExpression ::
    operand (< | <= | > | >= | == | ><) operand
  - Die Operanden haben die übliche Bedeutung. Dass Symbol '==' bedeutet Gleichheit (der Werte).

Dreiadress-Assembler (3AA)

- Aufruf (ARBI)
  - **`> dreiaddress`**



Dreiadress-Assembler (3AA)



Zusammenfassung

- Machinennahe Programmierung
  - grundlegende Vorgänge im Computer
  - alle Probleme mit elementarem System lösbar
    - ➔ soweit Kapazität (Speicher/Zeit) ausreichen
    - ➔ bis auf '*nicht berechenbare*' Probleme
    - ➔ unabhängig von Mächtigkeit des Befehlssatzes!
  - Grenzen der Strukturierbarkeit
  - 3AA ist Modell eines 'Assemblers'
    - Entspricht keiner realen Architektur!