Rechnerarchitektur – Teil 2

Fähigkeiten der Registermaschine und Sprachübersetzung

Dipl.-Inform. Meiko Lösch

2006-05-12

Inhalt

Zuweisungen

Arten von Zuweisungen

- Zurücksetzen auf Null
- Setzen auf eine Konstante
- Variablenzuweisung

Variablen auf 0 setzen

Notation und Vorgaben

- `x:=0;`
- x liegt an Speicherstelle 1

logische Struktur

```
while (x<>0) Do
  Begin
    Dec(x);
  End;
```

Implementierung

```
1 TST 1
2 JMP 4
3 HLT
4 DEC 1
5 JMP 1
```

Variablen auf Konstante setzen

Notation und Vorgaben

- `x:=k;`
- x liegt an Speicherstelle 1
- k ist eine (kleine) Konstante

logische Struktur

```
x:=0;
Inc(x);
Inc(x);
[...]
Inc(x);
```

Implementierung für k=3

```
1 TST 1
2 JMP 4
3 JMP 6
4 DEC 1
5 JMP 1
6 INC 1
7 INC 1
8 INC 1
```

Vereinfachte Notation

BONSAI-Assembler

```
1 TST 1
2 JMP 4
3 JMP 6
4 DEC 1
5 JMP 1
6 INC x
7 INC x
8 INC x
```

Vereinfachung 1

```
      TST x
      JMP +2
      JMP Lx0
      DEC x
      JMP -4
Lx0:  INC x
      INC x
      INC x
```

Vereinfachte Notation

Vereinfachung 1

```
      TST x
      JMP +2
      JMP Lx0
      DEC x
      JMP -4
Lx0:  INC x
      INC x
      INC x
```

Vereinfachung 2

```
x:=0;




INC x
INC x
INC x
```

Zuweisung von Variablen

Notation und Vorgaben

- `x:=y;`
- x liegt an Speicherstelle 1
- y liegt an Speicherstelle 2

logische Struktur

```
x:=0;
x:=y, h:=y, y:=0;
y:=h, h:=0;
```

Implementierung

```
x:=0;
h:=0;
TST y
JMP +2
JMP +5
DEC y
INC x
INC h
JMP -6
TST h
JMP +2
HLT
DEC h
INC y
JMP -5
```

Rechnen

Rechenoperationen

- Addition
- Subtraktion
- Multiplikation
- Division
- Potenzen
- Wurzeln

Addition

Notation und Vorgaben

- `x:=x+k`
- x liegt an Speicherstelle 1
- k ist eine Konstante

logische Struktur

```
INC x
INC x
[...]
INC x
```

Implementierung für k=3

```
INC x
INC x
INC x
```

Addition

Notation

- `x:=y+z`

logische Struktur

```
x:=y;
while (z>0) Do Begin
  Dec(z);
  Inc(x);
End;
```

Implementierung

```
x:=y;
TST z
JMP +2
HLT
DEC z
INC x
JMP -5
```

Subtraktion

Notation und Vorgaben

- `x:=y-z`
- $x - y = 0$ für $y > x$

logische Struktur

```
x:=y;
while (x<>0) and (z<>0)
  Do Begin
    Dec(x);
    Dec(z);
  End;
```

Implementierung

```
x:=y;
TST x
JMP +2
HLT
TST z
JMP +2
HLT
DEC x
DEC z
JMP -8
```

Multiplikation

Notation

- `x:=y*z`

logische Struktur

```
x:=0;
while (y<>0) Do
  Begin
    x:=x+z;
    Dec(y);
  End;
```

Problem

`x:=x+z;` zerstört den Inhalt von z.

Lösung

Wert von z in Hilfsvariable kopieren.

Ganzzahl-Division

Notation

- `x:=y DIV z`

logische Struktur

```
while (y<>0) Do
  Begin
    y:=y-z;
    if y<>0 Then Inc(x);
  End;
```

Problem

Das Programm liefert falsche Werte, wenn die Division aufgeht. Lässt man die Abfrage `y<>0` weg, ergeben sich falsche Werte, wenn ein Rest entsteht. $\Rightarrow$ Nachbessern!

Ausblick

Weitere Möglichkeiten

- Berechnungen ohne Nebeneffekte
- weitere Berechnungen
  - Division mit Rest:
    Rest zwischenspeichern.
  - Potenzen:
    wiederholt multiplizieren
  - Wurzeln:
    durch Schleife und Vergleich herantasten

Relationen

einfache Relationen

- Gleich
- Größer
- Kleiner

Idee

- $xRy \Leftrightarrow (x-1)R(y-1)$ für $R \in \{<, >, =\}$
- $0 = 0$
- $0 < i \wedge i > 0$ für $i \neq 0, i \in \mathbb{N}_0$

Vergleiche

logische Struktur

```
while (x<>0) and (y<>0)
  Do Begin
    Dec(x);
    Dec(y);
  End;
if (x=0) AND (y=0) Then
  { = }
if (x=0) AND (y<>0) Then
  { < }
if (x<>0) AND (y=0) Then
  { > }
```

Implementierung

```
TST x
JMP +4
TST y
JMP less
JMP equal
TST y
JMP +2
JMP greater
DEC x
DEC y
JMP -10
```

Schleifen

Schleifentypen

- While-Schleife
- Repeat-Until-Schleife
- For-Schleife

While-Schleife

While Bedingung Do ...

```
Start:
if NOT <Bedingung> Then
  goto Ende
{Schleifenrumpf}
goto Start
Ende:
```

Implementierung

```
Start:
b:=Bedingung;
TST b
JMP +2
JMP Ende
{Rumpf}
JMP Start
```

Repeat-Until-Schleife

Repeat ... Until <Bedingung>

```
Start:
{Schleifenrumpf}
if NOT <Bedingung> Then
  goto Start
```

Implementierung

```
Start:
{Rumpf}
b:=Bedingung;
TST b
JMP +2
JMP Start
```

For-Schleife

For i:=Startwert To Endwert

```
i:=Startwert;
Start:
{Schleifenrumpf}
Inc(i)
if i<=Endwert Then
  goto Start
```

Unterprogramme

Arten von Unterprogrammen

- Unterprogramm als Makro
  - bereits verwendet!
- „echtes“ Unterprogramm $\Rightarrow$ Prozedur/Funktion
  - kann das die Hardware?

Unterprogramme

Makros

- „Schablone“
- komplexere Befehle

Prozeduren/Funktionen

- Sprung an eine feste Adresse
- Übergabe von Parametern/Rückgabewerten
- Rücksprung an aufrufende Stelle

Unterprogramme

Prozeduren/Funktionen

- Sprung an eine feste Adresse
  - JMP
- Übergabe von Parametern/Rückgabewerten
  - Position festlegen
- Rücksprung an aufrufende Stelle
  - Rücksprungadresse kann nicht gespeichert werden

Unterprogramme

Prozeduren/Funktionen

- Rücksprung an aufrufende Stelle
  - Rücksprungadresse kann nicht gespeichert werden

Folgerung

Prozedur- und Funktionsaufrufe werden von der BONSAI-Hardware nicht unterstützt.

Aber

Die Mächtigkeit wird davon nicht beeinflusst, der entsprechende Code kann jeweils als Makro eingefügt werden.

Manuelle Übersetzung

Aufgabe

Übersetzen Sie das folgende Programm in BONSAI-Assembler und protokollieren Sie Ihr Vorgehen.

Pascal

```
PROGRAM Test1;
VAR i:Word;
BEGIN
  Inc(i);
  Dec(i);
END.
```

Assembler

```
1 INC 1
2 DEC 1
3 HLT
```

Manuelle Übersetzung

Pascal

```
BEGIN
  Inc(i);
  Dec(i);
END.
```

Assembler

```
1 INC 1
2 DEC 1
3 HLT
```

Vorgehen

- Befehle durch Pendant in Assembler ersetzen
- Zeilen durchnummerieren $\Rightarrow$ Adressen
- Variablen durchnummerieren $\Rightarrow$ Adressen
- Variablennamen durch Adresse ersetzen

Manuelle Übersetzung

Pascal

```
BEGIN
  Start:
  Inc(i);
  Dec(i);
  Goto Start;
END.
```

Assembler

```
1 INC 1
2 DEC 1
3 JMP 1
4 HLT
```

Vorgehen

- Befehle durch Pendant in Assembler ersetzen
- Zeilen durchnummerieren $\Rightarrow$ Adressen
- Sprungmarken durch Adressen ersetzen
- Variablen durchnummerieren $\Rightarrow$ Adressen
- Variablennamen durch Adressen ersetzen

Manuelle Übersetzung

Komplexere Befehle

- Schablonen/Makros müssen eingesetzt werden
- relative Sprünge machen die Makros ortsunabhängig

Probleme

- Verschachtelung von Makros
- Namensgebung für Hilfsvariablen

Automatisierte Übersetzung

Offene Fragen

1. Ausgangssprache
2. Zielsprache
3. notwendige Schritte

Ausgangssprache

RePas – reduziertes Pascal

- Pascal (Delphi) bereits bekannt
- Sprachelemente
  - so wenig wie möglich
  - so viel wie nötig
  - möglichst exakte Umsetzung von BONSAI-Assemlber

Ausgangssprache

Ergebnis

- Inc(<Var>);
- Dec(<Var>);
- Goto <Label>;
- If (<Var>=0) Then Goto <Label1> Else Goto <Label2>;
- Begin [...] End.
- Program <Name>
- Label <Label1>,<Label2>,[...];
- Var <Var1>:Word,<Var2>:Word,...;

Zielsprache

Möglichkeiten

1. BONSAI-Maschinensprache
   - realistisch
2. BONSAI-Assembler
   - unabhängig vom erweiterten Modus in BONSAI
   - Compiler existiert bereits ;-)

Der Compiler

Aufgabe

Unter `BONSAI\BONCOMP\` finden Sie einen Compiler für BONSAI. Dieser Teilt sich in drei Programme auf: `scanner.exe`, `parser.exe` und `codegen.exe`.

1. Führen Sie diese drei Programme – in dieser Reihenfolge – aus und wenden Sie sie auf ADD5 an.
2. Beobachten Sie die Ausgaben, die die Programme erzeugen. (`*.TOK`, `*.PTK`, `*.BON`)
3. Beschreiben Sie, was die Programme jeweils machen. Überprüfen Sie Ihre Theorie, indem Sie `T1.PAS`, `T2.PAS` und ggf. auch andere Programme übersetzen.

Der Compiler

Scanner

PROGRAM_S	PROGRAM
Bezeichner_S	ADDITION5
SEMIKOLON_S	;
LABEL_S	LABEL
Bezeichner_S	A1
KOMMA_S	,
[...]	
VAR_S	VAR
Bezeichner_S	R1

Parser

PROGRAM_S	PROGRAM
ProgName_S	ADDITION5
SEMIKOLON_S	;
LABEL_S	LABEL
LabelName_S	A1
KOMMA_S	,
[...]	
VAR_S	VAR
VarName_S	R1

Der Compiler

Scanner

- erkennen verbotener Zeichen
- erkennen von Zeichenfolgen

Parser

- Zeichenfolgen in ihrem Kontext sehen
- Struktur überprüfen
- Symboltabelle aufstellen

Codegenerator

- Befehle übersetzen
- Symboltabelle aufstellen und Adressen zuweisen

Ausblick

Weitere Möglichkeiten

- mehr Befehle unterstützen
  - Zuweisungen
  - Grundrechenarten
- mehr Konstrukte unterstützen
  - Schleifen
  - weitere Relationen außer $=0$
- ...

Quellen

Quellen

- http://hsg.region-kl.de/faecher/inf/material/bonsai/
- BONSAI-Dokumentation von Herrn Merkert.