Compiler und Codegenerierung

Hw-Sw-Co-Design

Überblick

- Compiler - Aufbau
- Codegenerierung
- Codeoptimierung
- Codegenerierung für Spezialprozessoren
- Retargetable Compiler

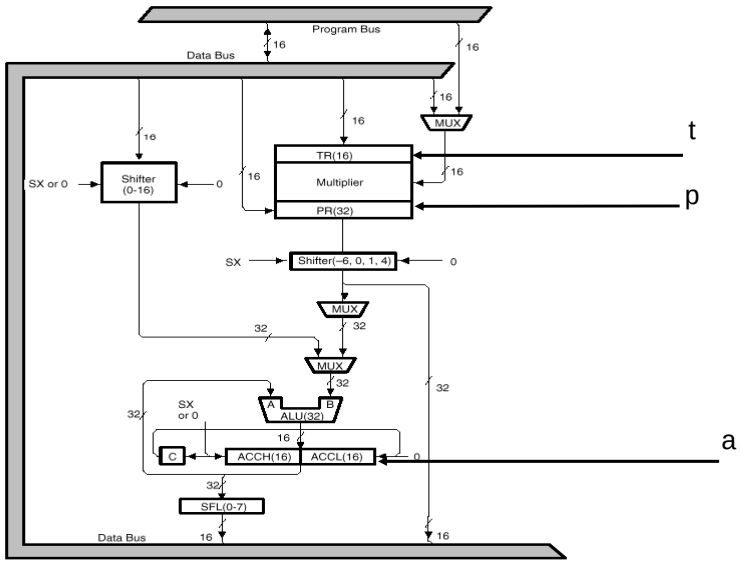
Anforderungen

- Softwareentwurf für eingebettete Systeme
  - Übergang von Assembler zu High-Level Languages (HLLs)
- Hauptanforderungen an den Code
  - korrekt
  - schnell
  - klein
- Weitere Anforderungen an HLL und Compiler
  - hohe Sicherheit: formale Verifikation soll möglich sein
  - Spezifikation von Echtzeitbedingungen
  - Unterstützung von DSP Algorithmen / Architekturen
  - retargetable: schnell an neuen Prozessor anpassbar

Ausgesuchte Problemstellungen

- Nicht-homogene Registersätze, irreguläre Datenpfade
  - enge Kopplung der Phasen Registerbindung, Befehlsauswahl und Ablaufplanung
- Zuweisung von Speicheradressen und Adressregistern
  - effiziente Nutzung von Adressregistern und spezialisierten Adressrechenwerken
- Codekompression
  - Reduktion von Speicher, wichtig bei kostensensitiven Anwendungen

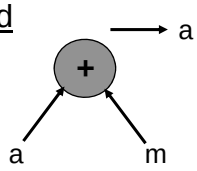
TMS320C25



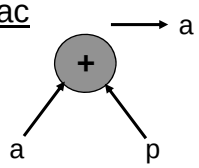
TMS320C25 - Instruktionen (1)

- Addition

add

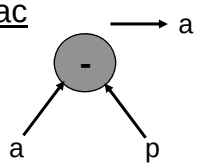


apac



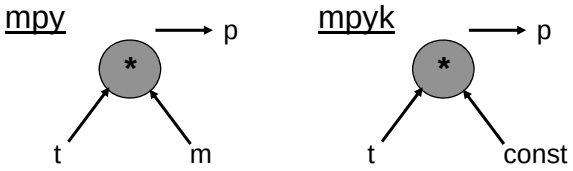
- Subtraktion

spac



TMS320C25 - Instruktionen (2)

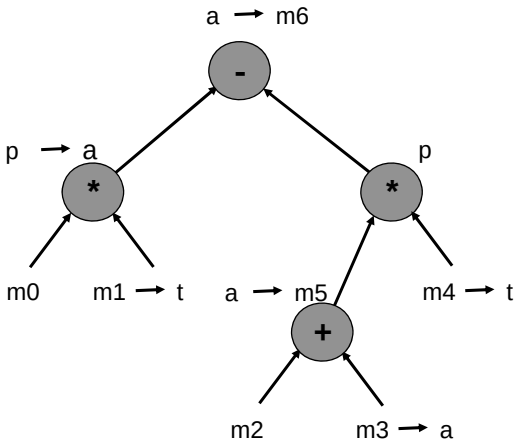
- Multiplikation



- Datentransfer

lack	const	a	lac	m	a
pac	p	a	lt	m	t
sacl	a	m			

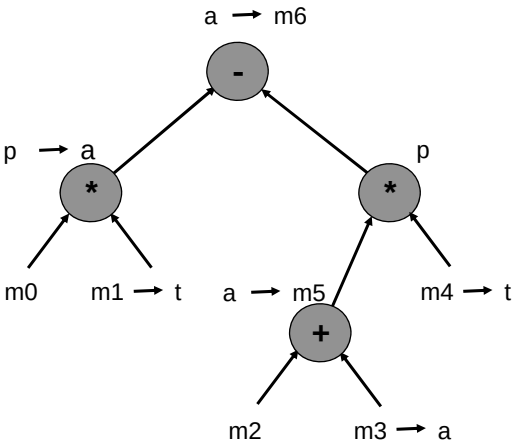
DAG - Beispiel (1)




```
; linker Teilbaum
; zuerst: Kosten 12

lt    m1
mpy   m0
pac
sacl  m7
lac   m3
add   m2
sacl  m5
lt    m4
mpy   m5
lac   m7
spac
sacl  m6
```

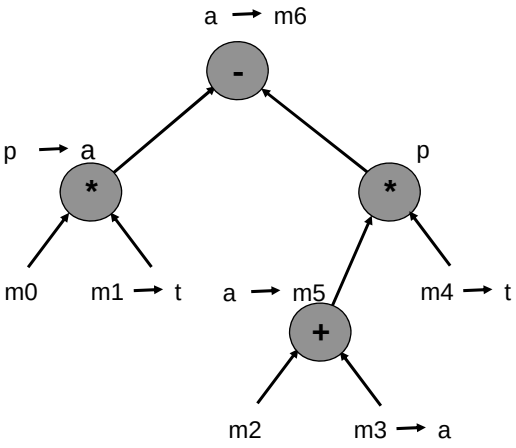
DAG - Beispiel (2)




```
; rechter Teilbaum
; zuerst: Kosten 14

lt    m4
lac   m3
add   m2
sacl  m5
mpy   m5
pac
sacl  m7
lt    m1
mpy   m0
pac
lt    m7
mpyk  1
spac
sacl  m6
```

DAG - Beispiel (3)

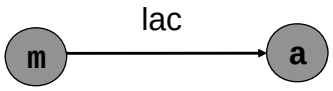



```
; optimaler Code
; Kosten 10

lac   m3
add   m2
sacl  m5
lt    m1
mpy   m0
pac
lt    m4
mpy   m5
spac
sacl  m6
```

Registertransfergraph

- Definition: Der Registertransfergraph eines Prozessors ist ein gerichteter Graph, bei dem jeder Knoten eine Stelle im Datenpfad darstellt, an der Daten gespeichert werden können. Eine Kante zwischen den Knoten r_i und r_j wird mit den Instruktionen beschriftet, die Operanden aus r_i lesen und nach r_j schreiben.



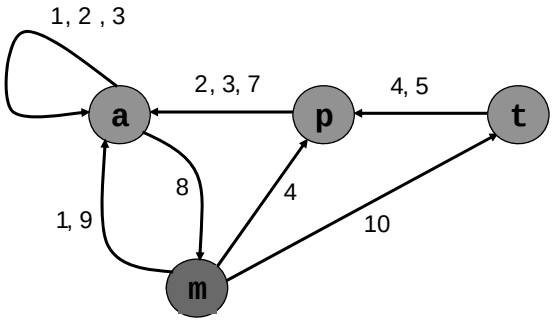
RTG - Kriterium

- Definition: Das RTG-Kriterium ist erfüllt, falls es für alle Knoten r_1 , r_2 und r_3 des RTG, für die
  i) r_3 eingehende Kanten von den Registerknoten r_1 und r_2 mit der gleichen Beschriftung hat,
  und
  ii) es mindestens einen Zyklus zwischen r_1 und r_2 gibt,
  gilt:
  In jedem möglichen Zyklus zwischen r_1 und r_2 existiert ein Speicherknoten.

→ Für Prozessoren, die das RTG-Kriterium erfüllen, kann ein optimaler Ablaufplan in Zeit $O(n)$ generiert werden (n ist die Anzahl der Knoten des DAGs).

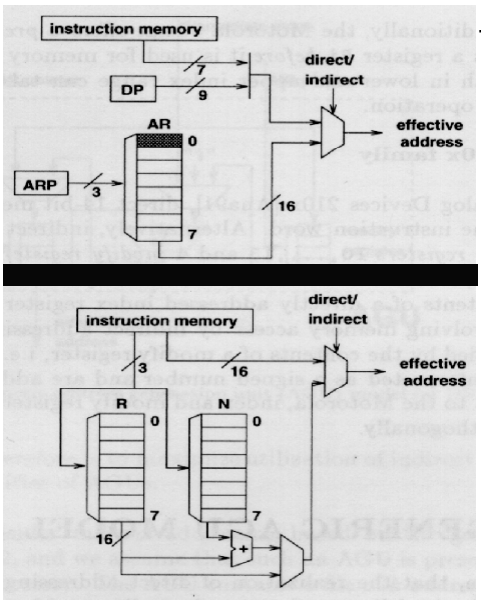
RTG - TMS320C25

1 add: a = a + m
2 apac: a = a + p
3 spac: a = a - p
4 mpy: p = t * m
5 mpyk: p = t * const
6 lack: a = const
7 pac: a = p
8 sacl: m = a
9 lac: a = m
10 lt: t = m



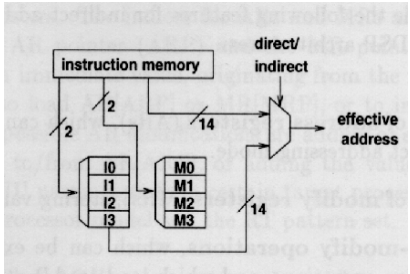
⟶ RTG-Kriterium erfüllt!

Adressrechenwerke (AGUs)



TMS320C2x

ADSP 210x



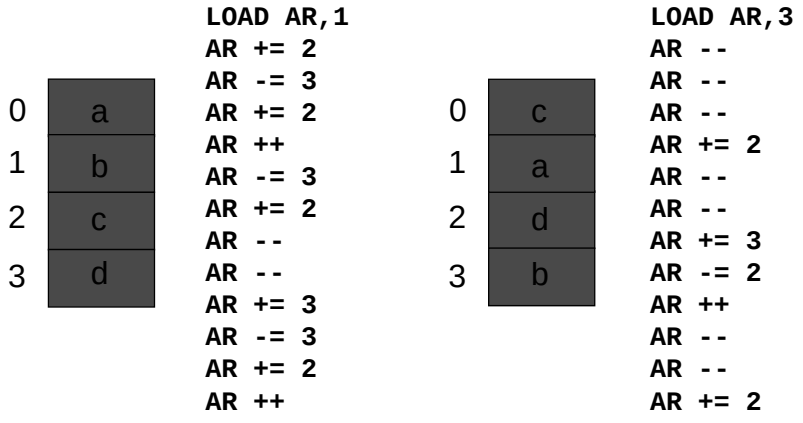
Motorola 56K

AGU - Eigenschaften

- Menge von Adressregistern AR für indirekte Adressierungsarten
- Menge von Modifikationsregistern MR für die Änderungen der AR
- Modifikationsoperationen, die parallel zur Instruktionsabarbeitung durchgeführt werden
  - post-modify: autoinkrement/dekrement um eine Adresse oder den Inhalt eines **MR**

Adressierung von Skalarvariablen

Zugriffssequenz: b, d, a, c, d, a, c, b, a, d, a, c, d



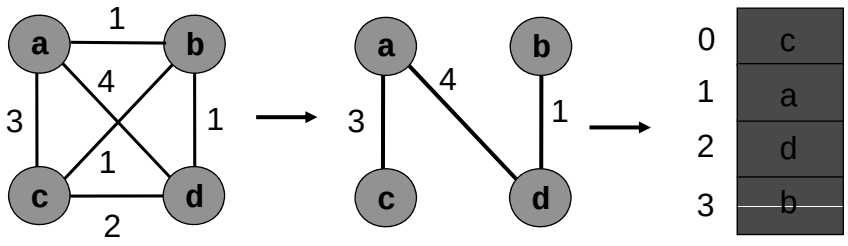
Zugriffsgraph

- Definition: Für eine Menge von Variablen und eine Zugriffssequenz besteht der Zugriffsgraph aus Knoten, die die Variablen darstellen, aus Kanten, die Knoten verbinden, wenn die Variablen benachbart sind und Kantengewichten, die die Anzahl der Transitionen zwischen den Variablen angeben.

→ Finden eines Hamiltonischen Pfades (Pfad, der alle Knoten genau einmal besucht) im Graph, der maximales Gewicht hat.

Zugriffsgraph

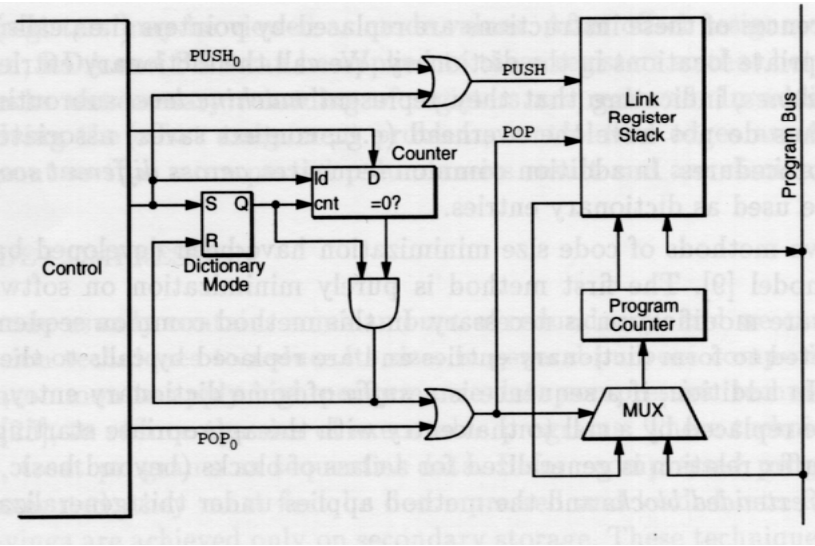
Zugriffssequenz: b, d, a, c, d, a, c, b, a, d, a, c, d



Codekompression

- Zielcode ist redundant und kann komprimiert werden
  - GP-Systeme: Dekompression beim Laden in das RAM
  - bei eingebetteten Systemen ist die Reduktion des Programm-ROMs, RAMs wichtig
- Dekompression im Cache
- External Pointer Macro (EPM) Modell
  - dictionary: enthält oft vorkommende Codesequenzen (mini-subroutines)
  - skeleton: enthält Befehle und pointer zum dictionary
  - Implementierung in SW oder HW

EPM Modell in Hardware



Überblick

- Compiler - Aufbau
- Codegenerierung
- Codeoptimierung
- Codegenerierung für Spezialprozessoren
- Retargetable Compiler

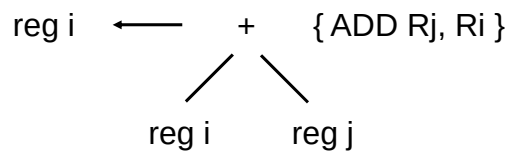
Retargetable Compiler

- maschinenunabhängige Compiler (automatically retargetable)
  - Compiler hat bereits Codegeneratoren für mehrere Prozessoren eingebaut
  - häufig bei parametrisierbaren Architekturen
- Compiler-Compiler (user retargetable)
  - ein Compiler wird aus einer Beschreibung der Zielarchitektur generiert
- portable Compiler (developer retargetable)

Baumübersetzungsschema

- Regeln für die Umwandlung eines Syntax-baumes (DAGs)

Ersetzung ⟵ Muster { Aktion }

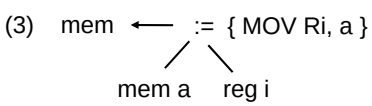


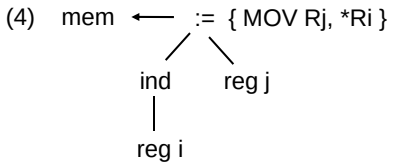
⟶ schrittweises Ersetzen, bis der Baum nur mehr einen Knoten hat

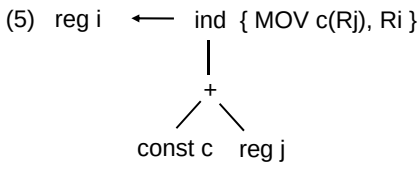
Zielinstruktionen (1)

(1) reg i ⟵ const c { MOV #c, Ri }

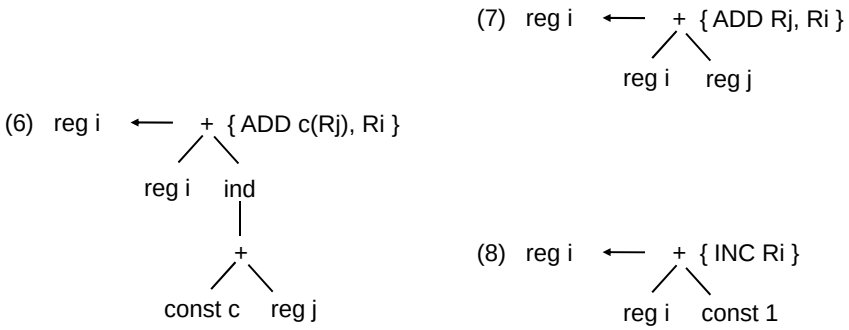
(2) reg i ⟵ mem a { MOV a, Ri }





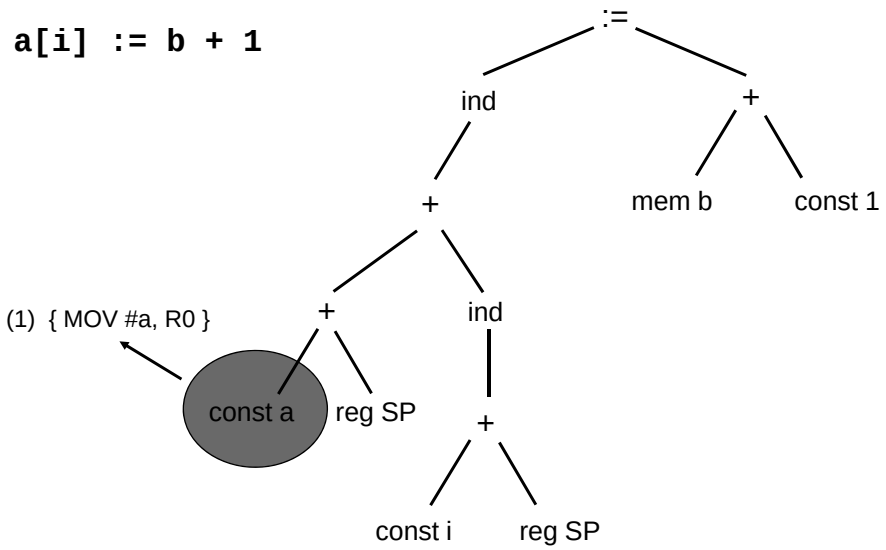


Zielinstruktionen (2)



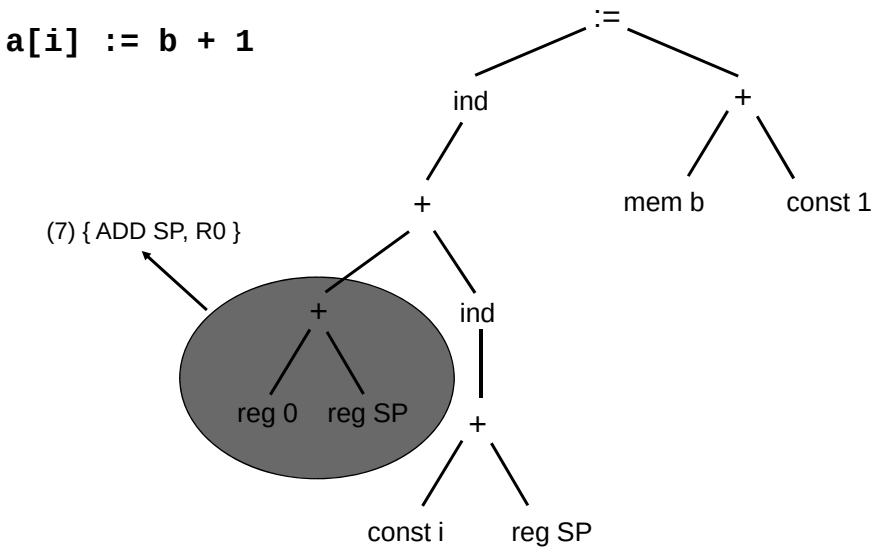
Baumübersetzung - Beispiel (1)

a[i] := b + 1



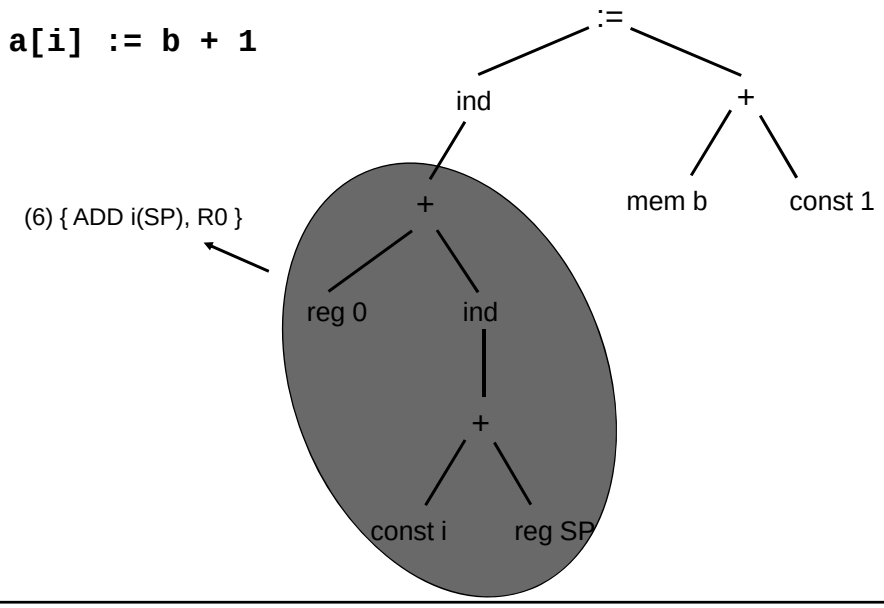
Baumübersetzung - Beispiel (2)

a[i] := b + 1



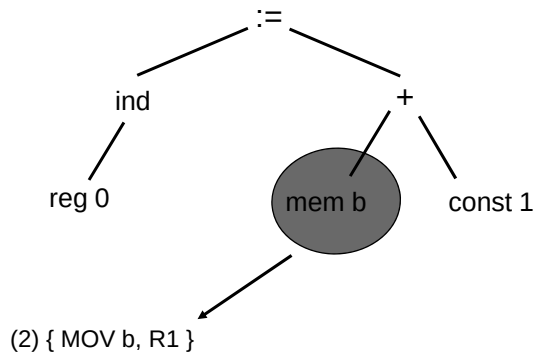
Baumübersetzung - Beispiel (3)

a[i] := b + 1



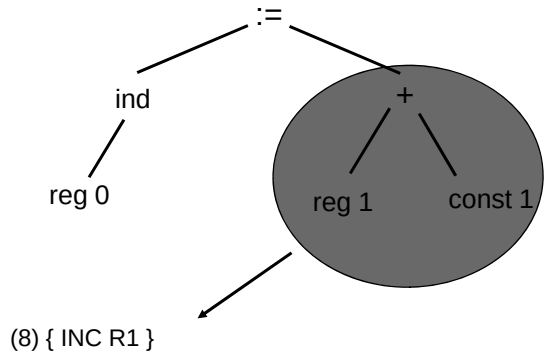
Baumübersetzung - Beispiel (4)

```
a[i] := b + 1
```



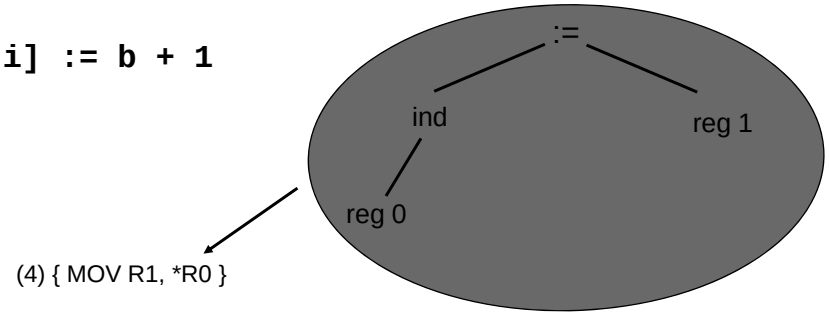
Baumübersetzung - Beispiel (5)

```
a[i] := b + 1
```



Baumübersetzung - Beispiel (6)

```
a[i] := b + 1
```




```
MOV #a, R0
ADD SP, R0
ADD i(SP), R0
MOV b, R1
INC R1
MOV R1, *R0
```

Prozessormodelle

- Verhaltensmodell
  - beschreibt den Instruktionssatz
  - Simulation relativ schnell (100-1000 mal langsamer als die Zielmaschine)
  - ungenau (keine Pipeliningeffekte)
- strukturelles Modell
  - beschreibt den Prozessor auf Registertransferebene
  - genau
  - Simulation ist langsam
  - nicht immer verfügbar
- gemischtes Modell

Fallstudien - Retargetable Compiler

- FlexWare
  - entwickelt bei SGS-Thomson
  - Instruktionssatzsimulator (INSULIN) und Codegenerator (CODESYN)
  - gemischtes Prozessormodell
  - für DSPs, ASIPs
- CHESS
  - entwickelt bei IMEC Leuven
  - Instruktionssatzsimulator und Codegenerator
  - Zielarchitektur in nML beschrieben
  - für DSP-Architekturen