


# Übung 5

Dipl.-Inform. Leonard Masing  
Dr.-Ing. Oliver Sander

## Institutsleitung

Prof. Dr.-Ing. Dr. h. c. J. Becker  
Prof. Dr.-Ing. E. Sax  
Prof. Dr. rer. nat. W. Stork

Institut für Technik der Informationsverarbeitung (ITIV)



# Hardware/Software Co-Design

# Agenda

- Wiederholung ausgewählter Themen
- Gruppenarbeit
- Vorstellung der Lösung

## 3.6 Software-Performanz: Metriken

- **MIPS** (million instructions per second)
- **MFLOPS** (million floating-point operations per second)
- **MACS** (million multiply & accumulates per second)
  - wichtig bei DSPs
- **MOPS** (million operations per second)
  - alle Operationen zusammengezählt: ALUs, Adressrechnungen, DMA, ...
  - bei allen:
    - Paralleloperationen berücksichtigt
    - optimale Belegung vorausgesetzt
- **Ausführungszeit**
  - Profiling: Compilation und möglichst viele Testläufe  
⇒ statistische Aussagen
  - Analytische Schätzung: auf Basis des Quell- / Zwischen- / Zielcodes  
⇒ wichtig bei Echtzeitsystemen mit harten Zeitschranken:  
⇒ **worst-case execution time (WCET)**

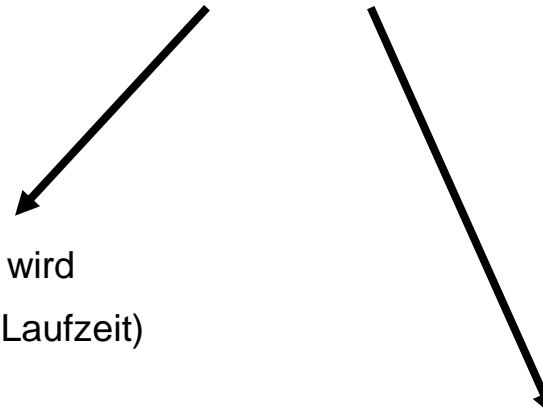
## 3.6 Software-Performanz: Worst-Case Execution Time (WCET)

- kann nicht durch *Profiling* bestimmt werden
- Schätzung mittels **Programmanalysetechniken**

### Programmpfadanalyse

Welche **Reihenfolge** von Instruktionen wird  
im worst-case ausgeführt? (längste Laufzeit)

Problem: die Anzahl der möglichen  
Programmpfade **wächst exponentiell**  
mit der Programmlänge



### Modellierung der Zielarchitektur

Berechnung der geschätzten WCET für ein  
spezifisches Prozessormodell

Probleme: Compileroptimierungen ,  
dynamische Effekte durch Pipelining, Caches  
Vereinfachte Annahmen nötig.

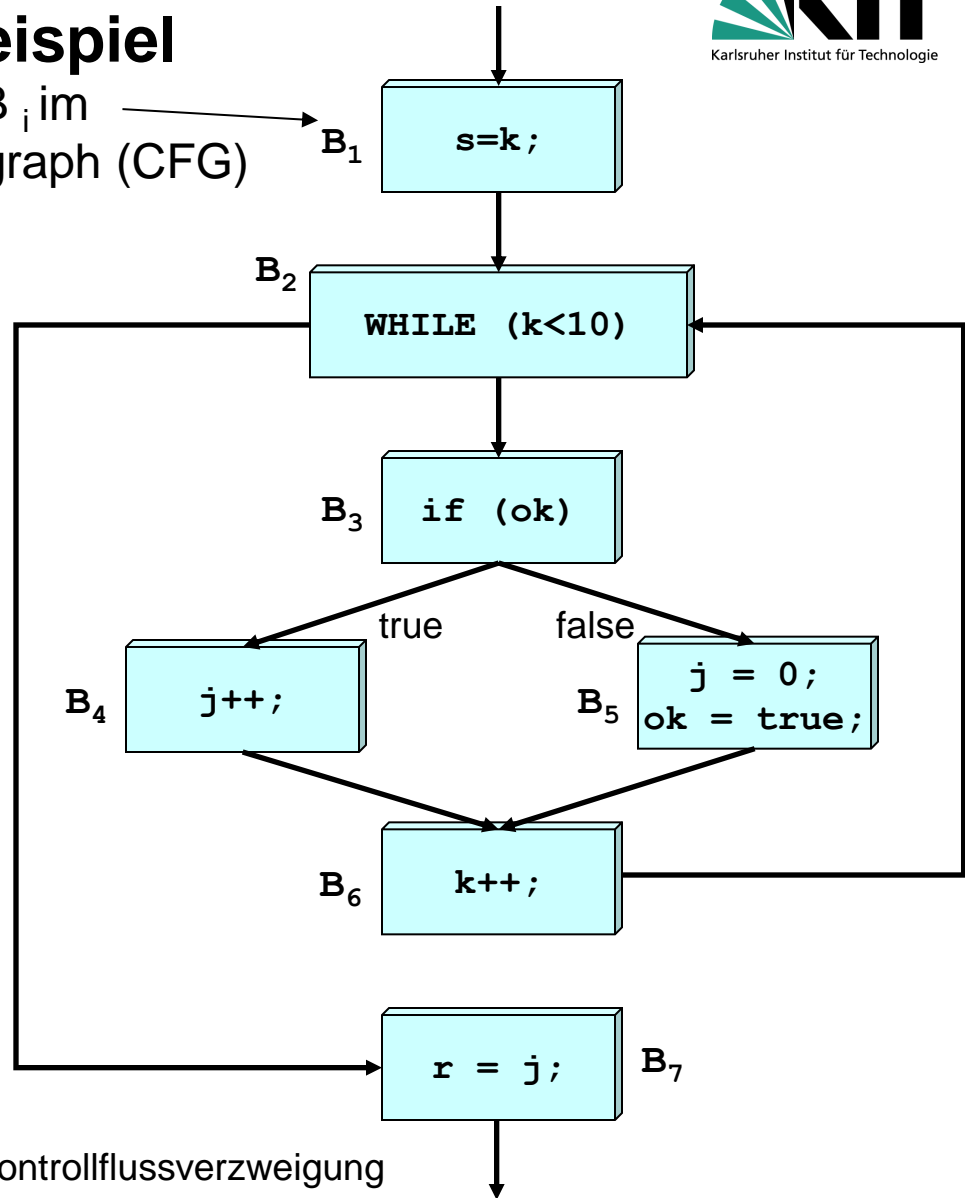
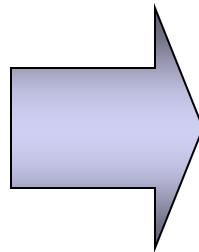
# 3.6 Software-Performanz: Programmpfadanalyse

- Die geschätzte WCET
  - ist immer größer als die tatsächliche WCET,
  - eine gute Schätzung approximiert die tatsächliche WCET aber möglichst nahe.
  
- Prozessormodell
  - ein Prozessor (eine skalare Einheit)
  - keine Interrupts
  - kein Betriebssystem (keine Preemption)
  
- Programmiermodell
  - keine rekursiven Funktionsaufrufe (direkt und indirekt)
  - keine Pointeroperationen
  - Schleifen müssen beschränkt sein

# 3.6 Software-Performanz: Programmpfadanalyse - Beispiel

```
/* k >= 0 */  
s = k;  
WHILE (k < 10) {  
  IF (ok)  
    j++;  
  ELSE {  
    j = 0;  
    ok = true;  
  }  
  k ++;  
}  
r = j;
```

Grundblock  $B_i$  im  
Kontrollflussgraph (CFG)



**Basisblock  $B_i$ :** Sequenz von Instruktionen bis Kontrollflussverzweigung

# 3.6 Software-Performanz: Strukturelle Beschränkungen

## ■ Flussgleichungen:

$$d_1 = d_2 = x_1$$

$$d_2 + d_8 = d_3 + d_9 = x_2$$

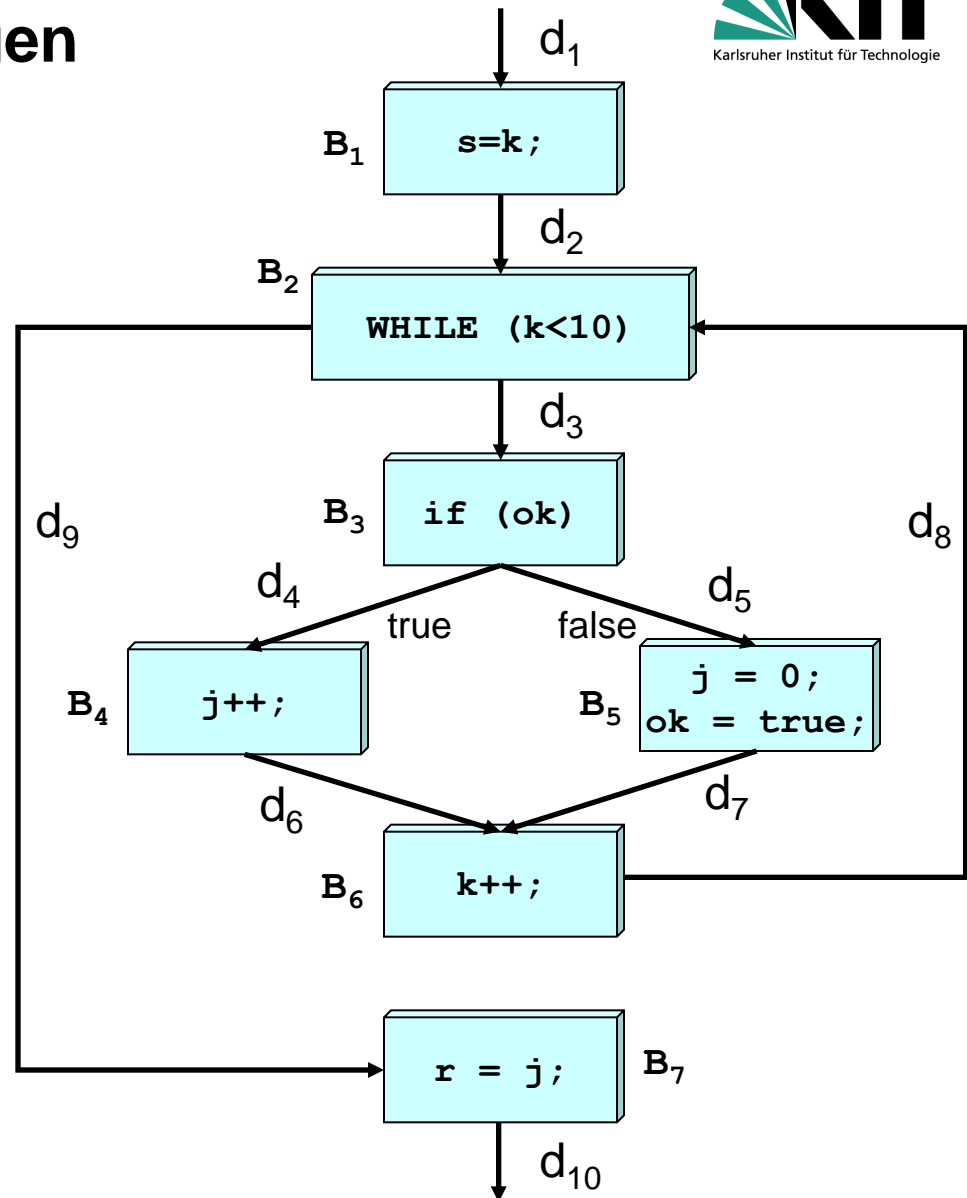
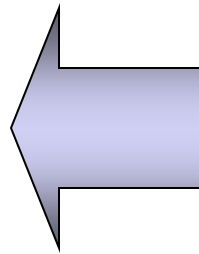
$$d_3 = d_4 + d_5 = x_3$$

$$d_4 = d_6 = x_4$$

$$d_5 = d_7 = x_5$$

$$d_6 + d_7 = d_8 = x_6$$

$$d_9 = d_{10} = x_7$$



## 3.6 Software-Performanz: Berechnung der WCET

### ■ Definition:

- Ein Programm besteht aus  $N$  Grundblöcken, wobei jeder Grundblock  $B_i$  eine Worst-Case Ausführungszeit  $c_i$  hat und genau  $x_i$  mal ausgeführt wird. Dann ist die

$$WCET = \sum_{i=1}^N c_i \cdot x_i$$

- Die  $c_i$  können abgeschätzt werden, da die Sequenz der Instruktionen bekannt ist (Grundblock-Definition).
- Wie berechnet man die  $x_i$  ?
  - **strukturelle Constraints** durch Programmstruktur gegeben
  - **funktionale Constraints** durch Programmierer gegeben (Schleifengrenzen, etc.)
- **Anmerkung:** alles auf vom Compiler erzeugten Assemblercode bezogen; Source Code aber einfacher nachzuvollziehen



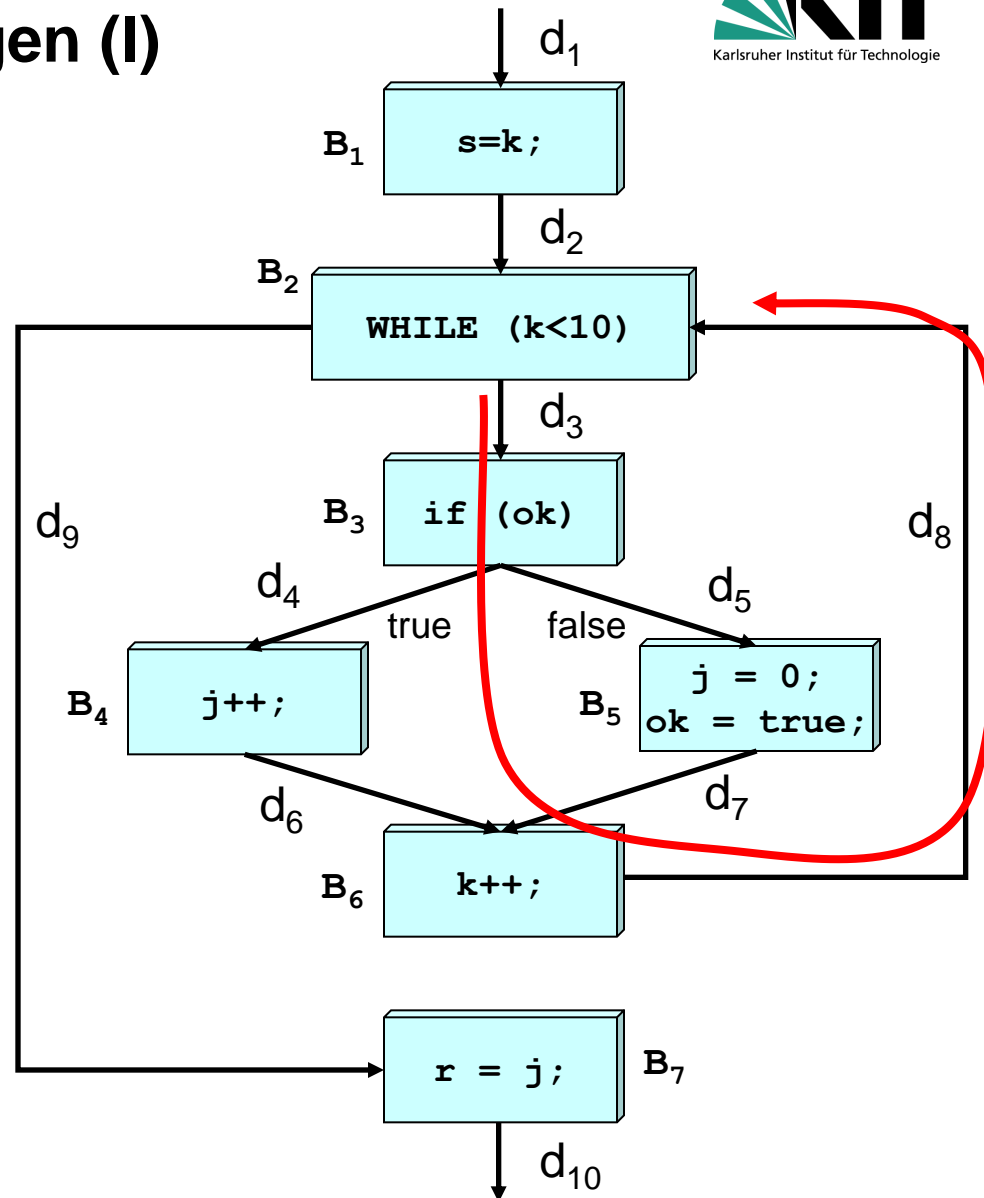
# 3.6 Software-Performanz: Funktionale Beschränkungen (I)

■ Die **While-Schleife** wird maximal **10 mal** durchlaufen:

$$0 \cdot x_1 \leq x_3 \leq 10 \cdot x_1$$

■ **B<sub>5</sub>** wird maximal einmal durchlaufen:

$$x_5 \leq 1 \cdot x_1$$



## 3.6 Software-Performanz: Funktionale Beschränkungen (II)

- Werden durch den Programmierer definiert
  - durch Schranken für Schleifenzähler
  - durch Kenntnis des Programmkontextes

■ Können komplex sein

■ **Beispiel (angenommen!):**

“Wird der ELSE-Zweig in der Schleife ausgeführt, so wird die Schleife genau 5 mal durchlaufen.”

Spaltet die funktionalen Constraints in zwei Sets  $\Rightarrow$  **getrennte ILPs ausführen.**

$$(x_5 = 0) \quad || \quad (x_5 \geq 1) \quad \& \quad (x_3 = 5 \cdot x_1)$$

set#1

set#2

# 3.6 Worst-Case Execution Time (WCET) – ILP Formulierung

- ILP mit strukturellen und funktionalen Beschränkungen:

$$WCET = \max \left\{ \sum_{i=1}^N c_i \cdot x_i \right\}$$

einmaliger Programmaufruf

- Nebenbedingungen:

$$(d_1 = 1) \wedge$$

strukturelle Constraints

$$\left( \sum_{j \in \text{Inputs}(B_i)} d_j = \sum_{k \in \text{Outputs}(B_i)} d_k = x_i, i = 1 \dots N \right) \wedge$$

ein ILP pro Set von Constraints

(funktionale Constraints)

# Arbeitsphase

- Aufgabe 3.04: Worst Case Execution Time
  - ILP Formulierung ohne Cache
  
- Aufgabe 3.05: Erweiterung WCET
  - Allgemeine Fragen
  
- Aufgabe 3.06: Erweiterung WCET
  - ILP Formulierung mit Cache

## Aufgabe 3.04: WCET

■ Gegeben ist folgendes C-Fragment:

```
■ weight=volume*density;  
  
■ if (weight<100) {  
■     max_index=10;  
■ } else {  
■     max_index=5;  
■ }  
  
■ index=1;  
■ additional_weight=0;  
  
■ do {  
■     additional_weight+=20;  
■     index++;  
■ } while (index<=max_index)  
  
■ weight+=additional_weight;
```

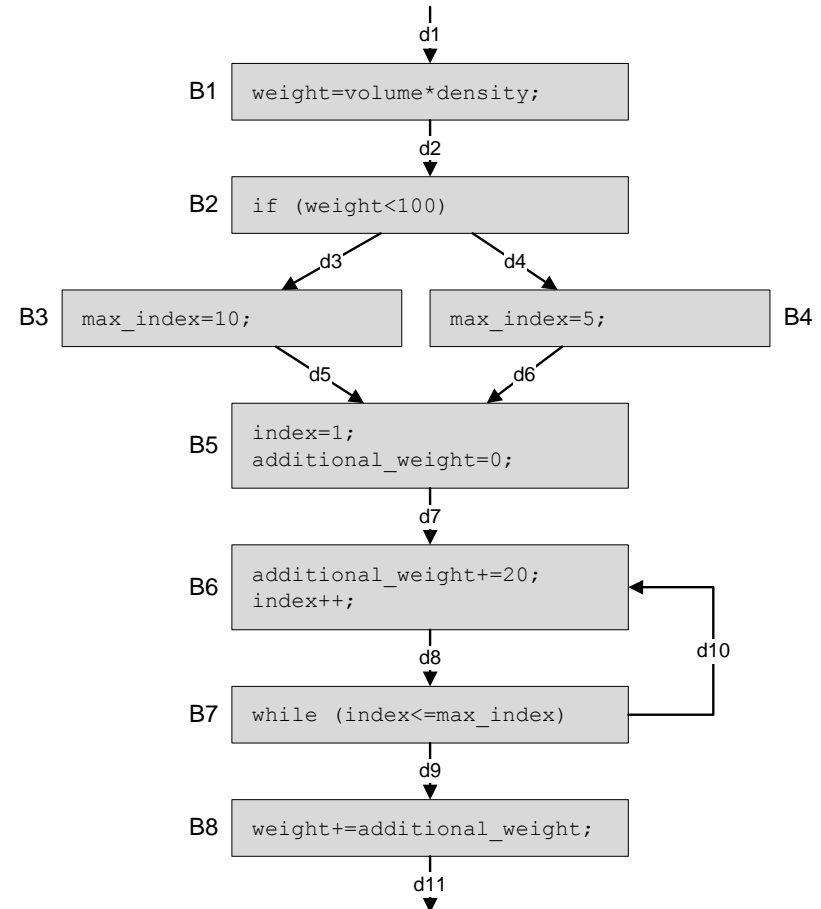
- a) Transformieren Sie das C-Fragment in einen Kontrollflussgraphen (CFG).
- b) Die WCET (ohne Caches) soll mittels ILP bestimmt werden. Stellen Sie hierfür die Flussgleichungen zur Modellierung der strukturellen Nebenbedingungen (Constraints) auf.
- c) Stellen Sie die funktionalen Nebenbedingungen auf.
- d) Geben Sie die Zielfunktion an, die durch ILP maximiert wird.

# Lösung Aufgabe 3.04a: WCET

■ Transformieren Sie das C-Fragment in einen Kontrollflussgraphen (CFG).

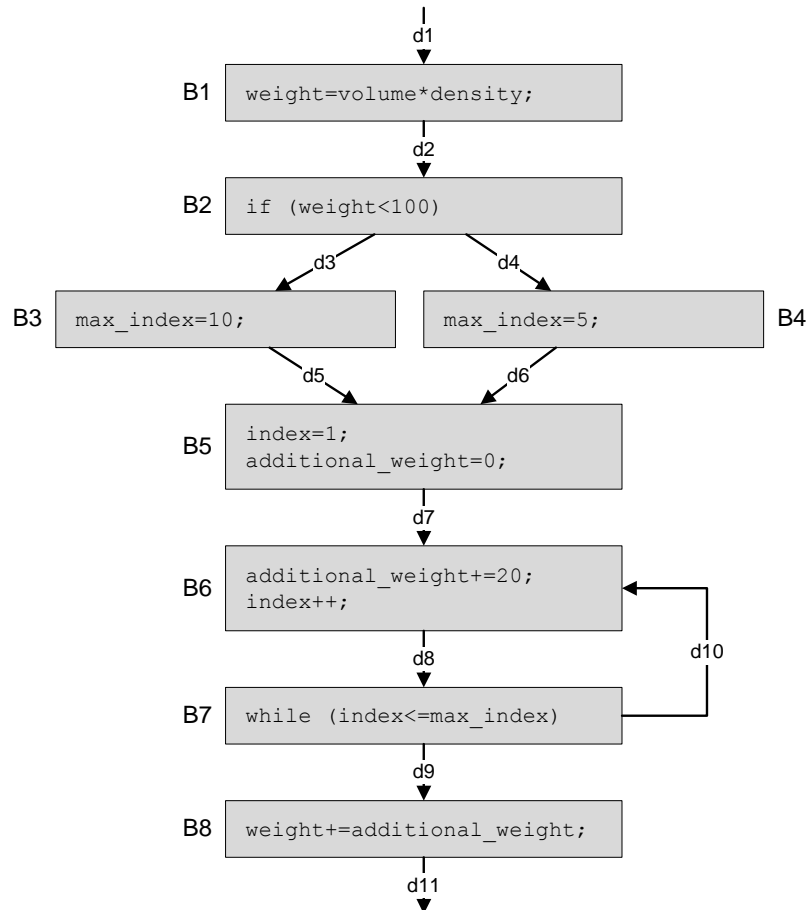
■ Gegeben ist folgendes C-Fragment:

- `weight=volume*density;`
- `if (weight<100) {`
- `max_index=10;`
- `} else {`
- `max_index=5;`
- `}`
- `index=1;`
- `additional_weight=0;`
- `do {`
- `additional_weight+=20;`
- `index++;`
- `} while (index<=max_index)`
- `weight+=additional_weight;`



# Lösung Aufgabe 3.04b: WCET

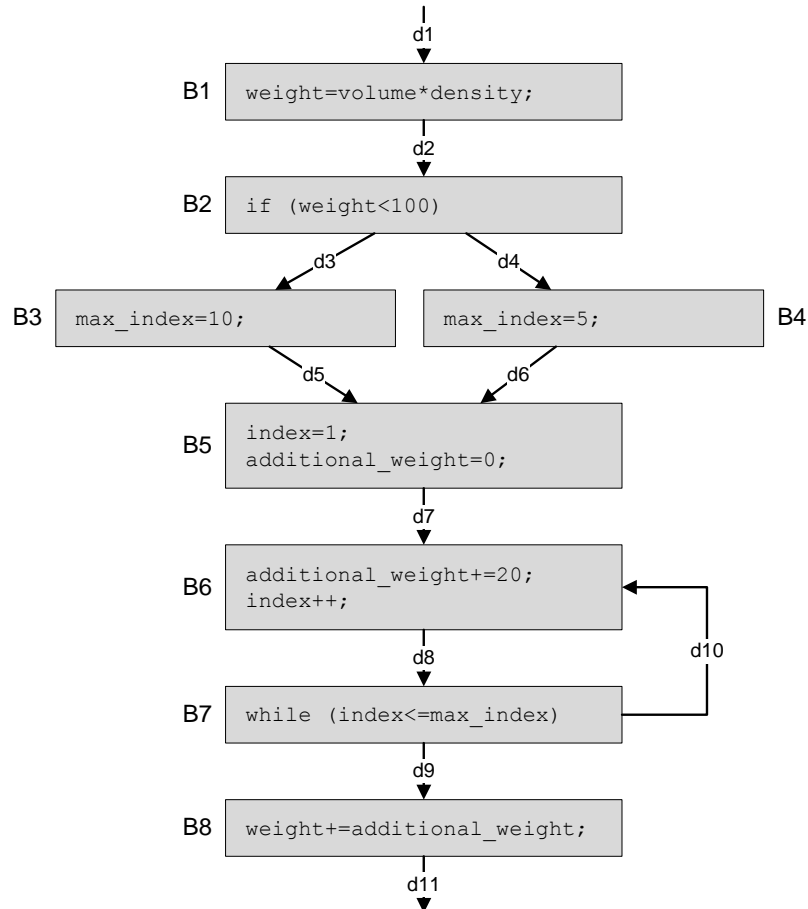
- Die WCET (ohne Caches) soll mittels ILP bestimmt werden. Stellen Sie hierfür die Flussgleichungen zur Modellierung der strukturellen Nebenbedingungen (Constraints) auf.



- $d1=1$  (einmaliger Programmaufruf)
- $x1 = d1 = d2$
- $x2 = d2 = d3 + d4$
- $x3 = d3 = d5$
- $x4 = d4 = d6$
- $x5 = d5 + d6 = d7$
- $x6 = d7 + d9 = d8$
- $x8 = d10 = d11$

# Lösung Aufgabe 3.04c: WCET

■ Stellen Sie die funktionalen Nebenbedingungen auf.



■  $5 \cdot x_5 \leq x_6 \leq 10 \cdot x_5$



# Lösung Aufgabe 3.04d: WCET

- Geben Sie die Zielfunktion an, die durch ILP maximiert wird.

$$WCET = \max \left\{ \sum_{i=1}^N c_i \cdot x_i \right\}$$

- Worst Case Execution Time
  - Wie muss Code für die WCET dargestellt werden?
  - Wie werden die strukturellen Beschränkungen abgeleitet?
  - Wie werden die funktionalen Nebenbedingungen aufgestellt?
  - Welche Zielfunktion wird maximiert?



## Aufgabe 3.05: Erweiterung WCET

- Es wird das erweiterte WCET-Modell zur ILP Formulierung von Caches betrachtet.
  - a) Welche Cache Art kann mit dem Modell modelliert werden?
  - b) Was ist ein L-Block?
  - c) Wie lautet die ILP Zielfunktion der erweiterten WCET Formulierung? Erklären Sie die einzelnen Variablen.
  - d) Was repräsentiert eine Kante des Cachekonfliktgraphen im CFG? Was darf diese nicht beinhalten?

# Lösung Aufgabe 3.05: Erweiterung WCET

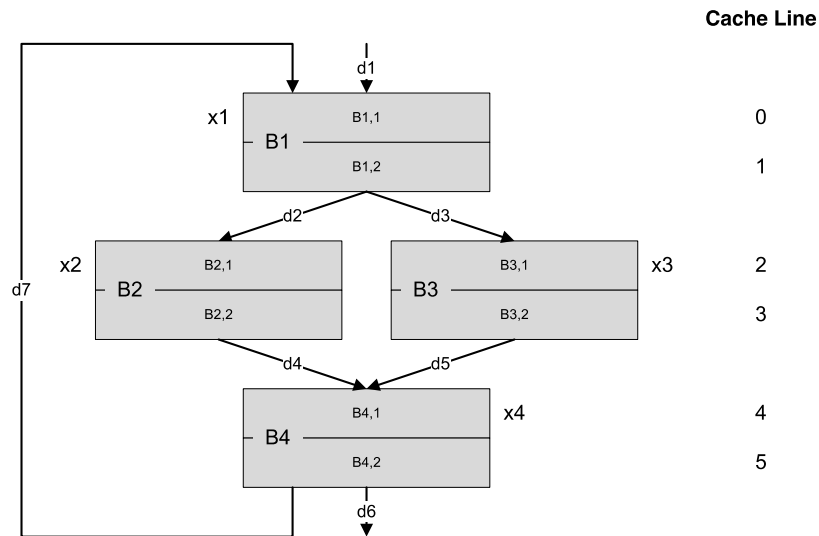
- Es wird das erweiterte WCET-Modell zur ILP Formulierung von Caches betrachtet.
  - a) Welche Cache Art kann mit dem Modell modelliert werden?
    - Direct-Mapped Instruction Cache
  - b) Was ist ein L-Block?
    - Ein L-Block ist die Kurzform für Line-Block und bezeichnet einen Teil eines Basis-Blockes, der zu der gleichen Cache-Zeile gemapped wird. Ein Basis-Block besteht aus ein oder mehreren L-Blöcken.
  - c) Wie lautet die ILP Zielfunktion der erweiterten WCET Formulierung? Erklären Sie die einzelnen Variablen.
    - $$WCET = \max \left\{ \sum_{i=1}^N \sum_{j=1}^{n_i} (c_{i,j}^{hit} \cdot x_{i,j}^{hit} + c_{i,j}^{miss} \cdot x_{i,j}^{miss}) \right\}$$
  - d) Was repräsentiert eine Kante des Cachekonfliktgraphen im CFG? Was darf diese nicht beinhalten?
    - Eine Kante im Cachekonfliktgraphen repräsentiert einen Pfad im Kontrollflussgraphen. Auf diesem Pfad darf kein L-Block liegen, der die gleiche Cache-Zeile verwendet.

- Erweiterung WCET
  - Welche Caches können modelliert werden?
  - Was ist ein L-Block?
  - Wie lautet die Zielfunktion jetzt?
  - Was ist ein CCG?
  - Was repräsentiert eine Kante?



# Aufgabe 3.06: Erweiterung WCET

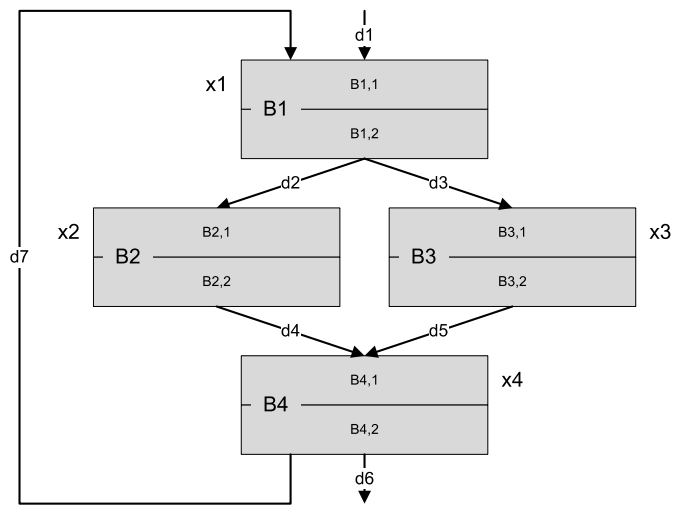
- Gegeben ist ein CFG mit vier Basis-Blöcken bestehend aus jeweils zwei L-Blöcken. Auf der rechten Seite sind die Cache-Zeilen der L-Blöcke angegeben. So verwenden z.B. B2,1 und B3,1 beide die 2te Cache-Zeile und stehen somit im Konflikt.



- Formulieren Sie die strukturellen Constraints aus den Flussgleichungen des CFGs.
- Formulieren Sie die allgemeinen Bedingungen, die für das erweiterte WCET ILP-Modell notwendig sind.
- Zeichnen Sie den Cachekonfliktgraphen für die zweite Cache-Zeile.
- Geben Sie die aus dem CCG folgende strukturelle Constraints der 2ten Cache-Zeile an.
- Geben Sie die strukturellen Constraints für die Cache-Zeilen 0, 1, 4 und 5 an.

# Lösung Aufgabe 3.06a: Erweiterung WCET

- Formulieren Sie die strukturellen Constraints aus den Flussgleichungen des CFGs



Cache Line

0  
1  
2  
3  
4  
5

■  $x1 = d7 + d1 = d2 + d3$

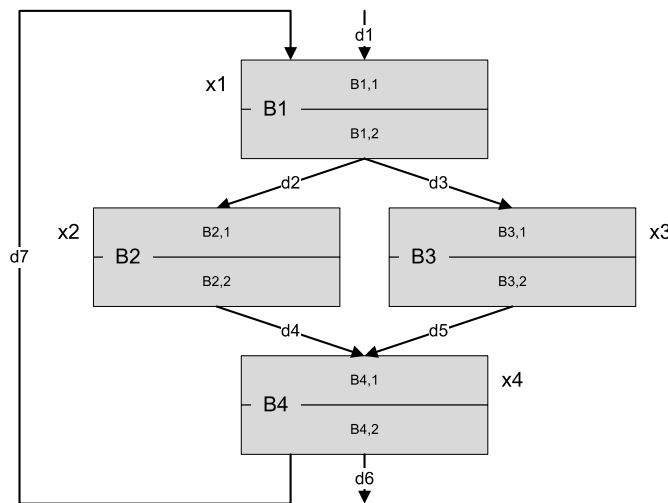
■  $x2 = d2 = d4$

■  $x3 = d3 = d5$

■  $x4 = d4 + d5 = d6 + d7$

# Lösung Aufgabe 3.06b: Erweiterung WCET

- Formulieren Sie die strukturellen Constraints aus den Flussgleichungen des CFGs



Cache Line

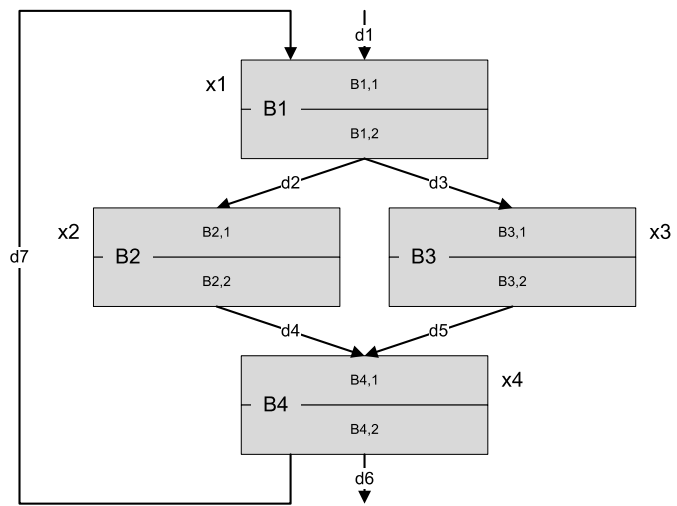
0	$X_1 = X_{1,1} = X_{1,1}^{hit} + X_{1,1}^{miss}$
1	$X_2 = X_{2,1} = X_{2,1}^{hit} + X_{2,1}^{miss}$
2	$X_3 = X_{3,1} = X_{3,1}^{hit} + X_{3,1}^{miss}$
3	$X_4 = X_{4,1} = X_{4,1}^{hit} + X_{4,1}^{miss}$
4	
5	

$X_1 = X_{1,2} = X_{1,2}^{hit} + X_{1,2}^{miss}$
$X_2 = X_{2,2} = X_{2,2}^{hit} + X_{2,2}^{miss}$
$X_3 = X_{3,2} = X_{3,2}^{hit} + X_{3,2}^{miss}$
$X_4 = X_{4,2} = X_{4,2}^{hit} + X_{4,2}^{miss}$



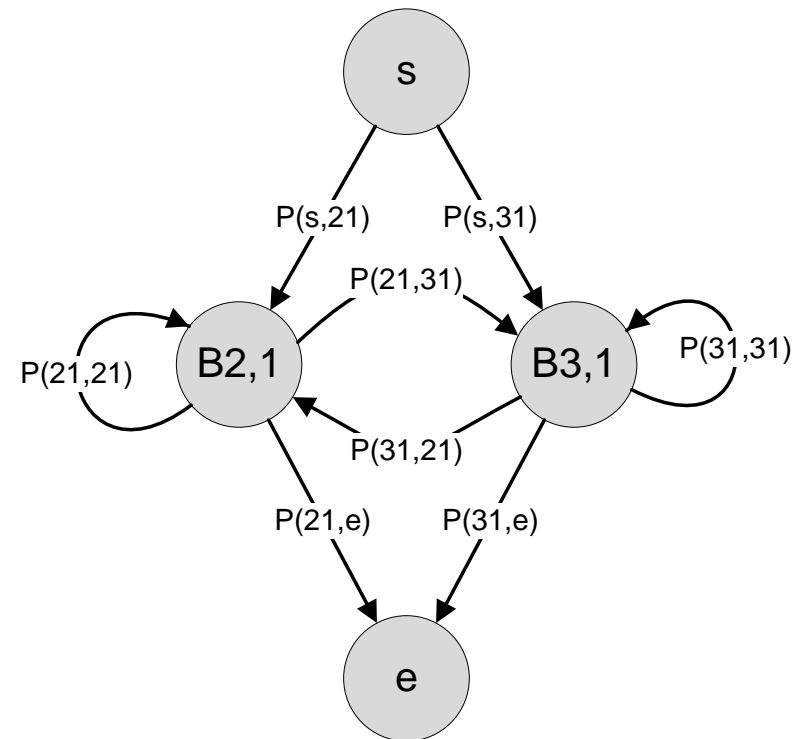
# Lösung Aufgabe 3.06c: Erweiterung WCET

- Zeichnen Sie den Cachekonfliktgraphen für die zweite Cache-Zeile.



Cache Line

0  
1  
2  
3  
4  
5



# Lösung Aufgabe 3.06d: Erweiterung WCET

- Geben Sie die aus dem CCG folgende strukturelle Constraints der 2ten Cache-Zeile an.

## Flussgleichungen

$$x_2 = x_{2,1} = p_{s,21} + p_{31,21} + p_{21,21} = p_{21,31} + p_{21,e} + p_{21,21}$$

$$x_3 = x_{3,1} = p_{s,31} + p_{21,31} + p_{31,31} = p_{31,21} + p_{31,e} + p_{31,31}$$

Cache Line

0  
1  
2  
3  
4  
5

## Cache-Hits

$$x_{2,1}^{hit} = p_{21,21}$$

$$x_{3,1}^{hit} = p_{31,31}$$

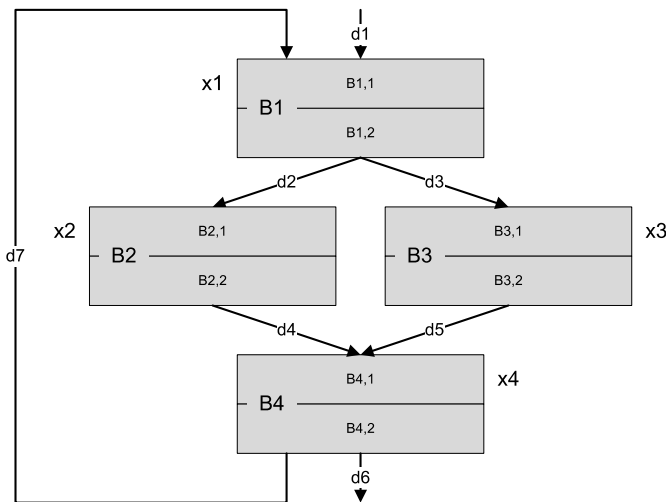
## Cache-Miss

$$x_{2,1}^{miss} = p_{s,21} + p_{31,21}$$

$$x_{3,1}^{hit} = p_{s,31} + p_{21,31}$$

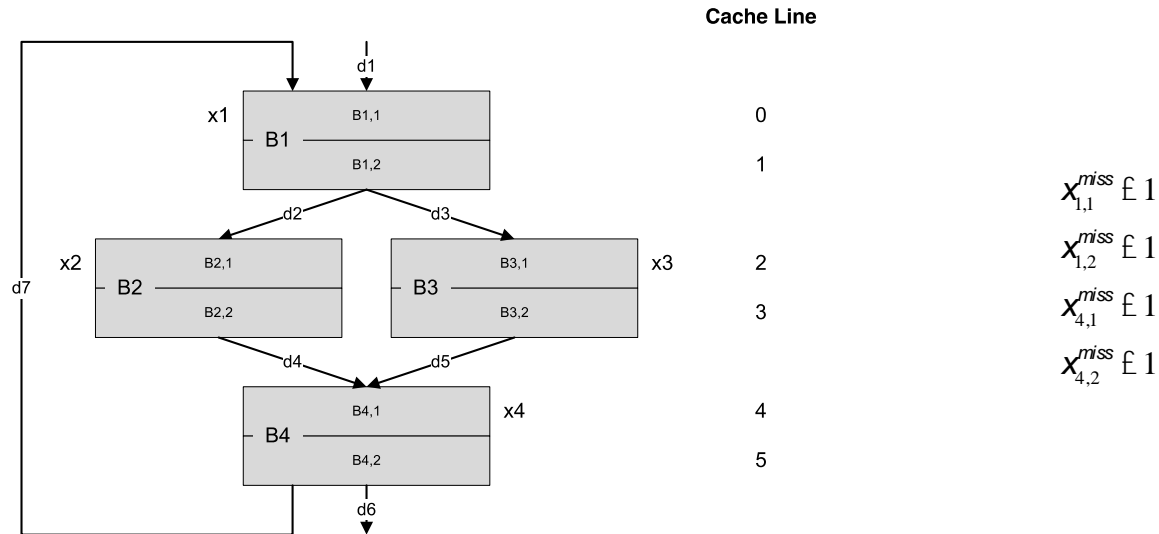
## Einmalige Ausführung

$$p_{s,21} + p_{s,31} + p_{s,e} = 1$$



# Lösung Aufgabe 3.06e: Erweiterung WCET

- Geben Sie die strukturellen Constraints für die Cache-Zeilen 0, 1, 4 und 5 an.



- Erweiterung WCET
  - Wie muss Code für die WCET dargestellt werden?
  - Wie werden die strukturellen Beschränkungen abgeleitet?
  - Wie werden die allgemeinen Bedingungen aufgestellt?
  - Wie wird der CCG abgeleitet?
  - Wie werden die strukturellen Beschränkungen für eine Cache Zeile abgeleitet?

