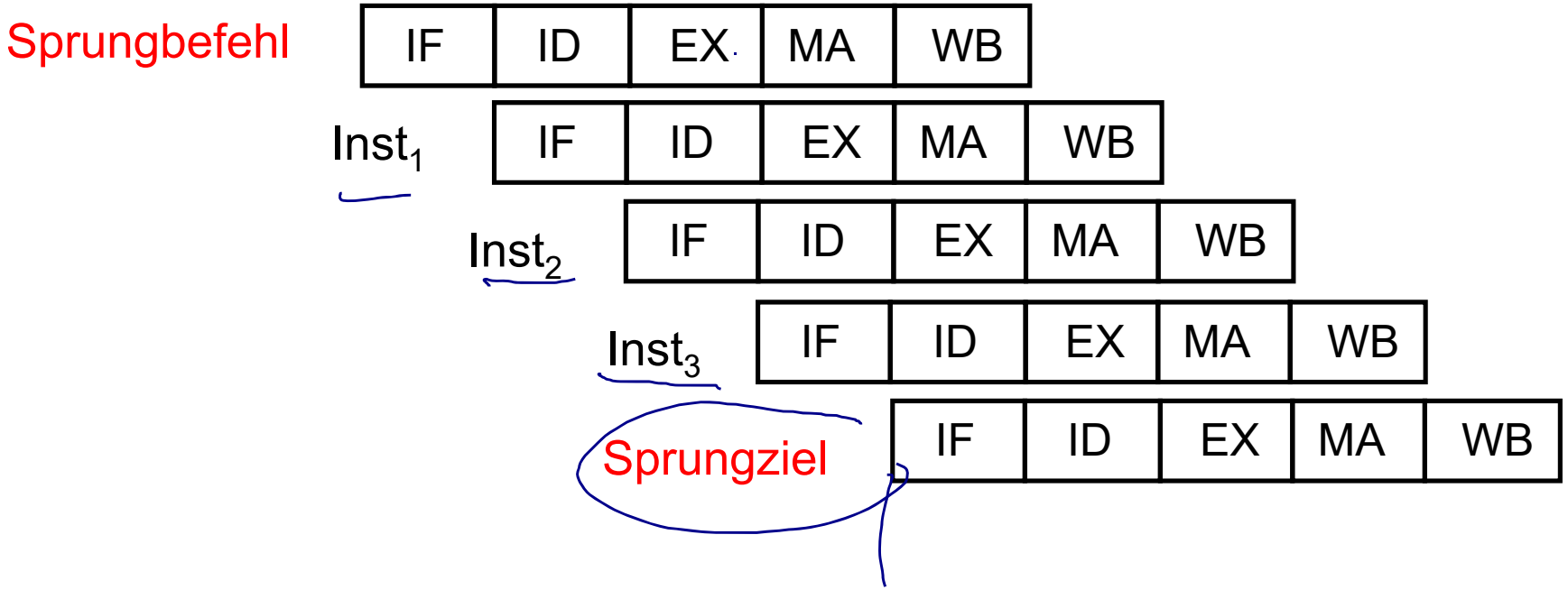
5.3 Pipeline-Konflikte

- **Steuerflusskonflikte**
  - Programmsteuerbefehle:
    - die bedingten und die unbedingten Sprungbefehle,
    - die Unterprogrammaufruf- und -rückkehrbefehle sowie
    - die Unterbrechungsbefehle
  - Steuerflussabhängigkeiten verursachen Steuerflusskonflikte in der DLX-Pipeline, da
    - der Programmsteuerbefehl erst in der ID-Stufe als solcher erkannt und damit bereits ein Befehl des falschen Programmpfades in die Pipeline geladen wurde
    - die Sprungzieladresse von der ALU erst in der EX-Stufe berechnet wird und
    - der PC am Ende der MEM-Stufe ersetzt wird

5.3 Pipeline-Konflikte

Steuerflusskonflikte durch Verzweigung

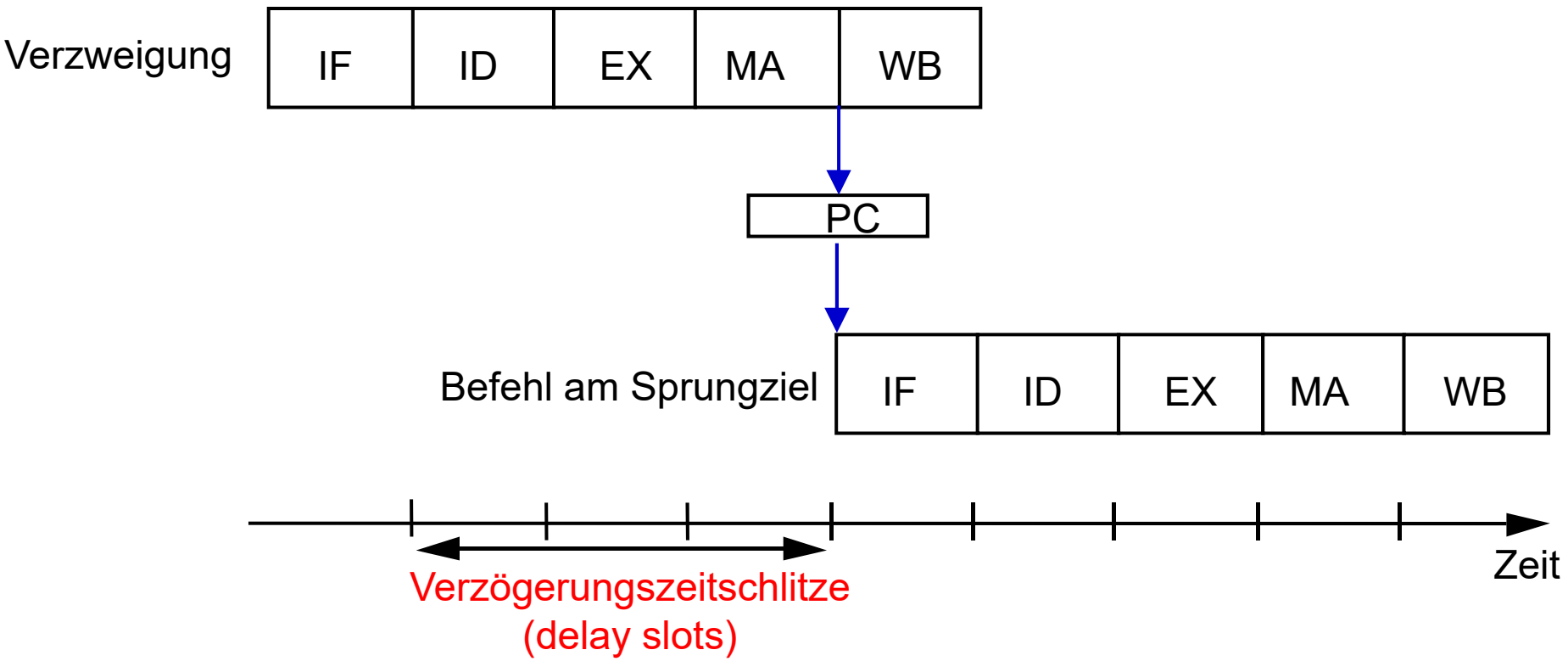
- In der Pipeline befinden sich drei Befehle, die bereits geladen worden sind und wieder gelöscht (annulliert) werden müssen, wenn der Sprung ausgeführt wird
- Bei einer Verzweigung kann erst nach drei Taktzyklen mit der Ausführung der korrekten Befehlsfolge gestartet werden



5.3 Pipeline-Konflikte

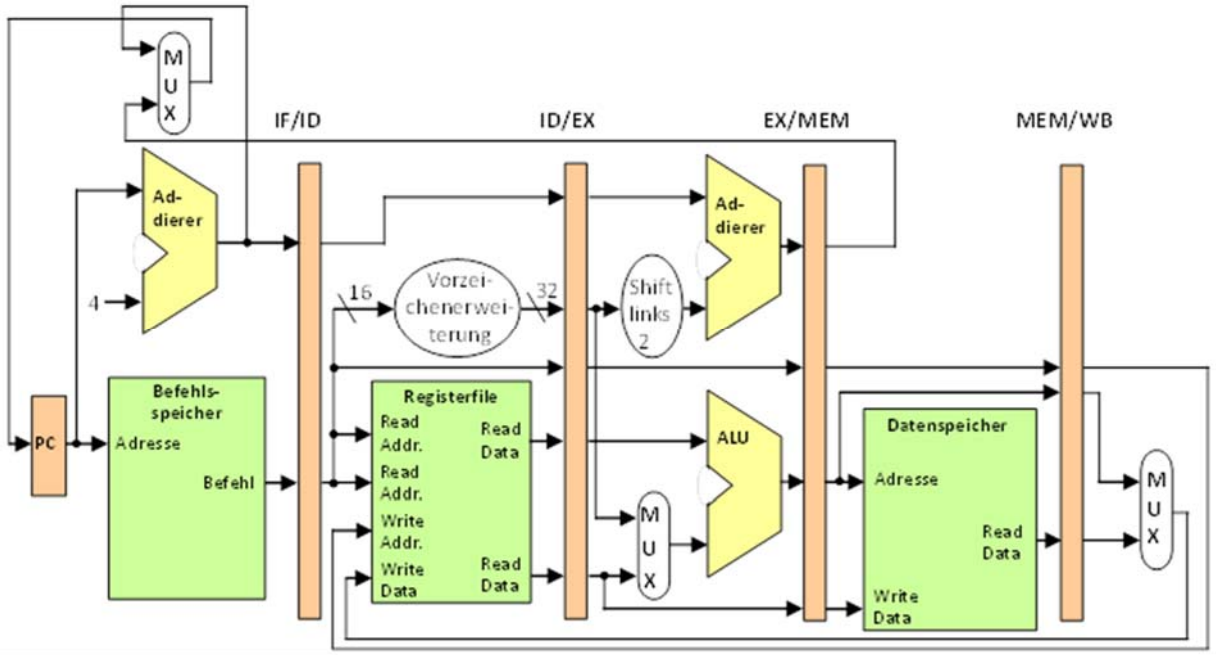
Steuerflusskonflikte durch Verzweigung

- Nach einem genommenen bedingten Sprung ist eine Pipeline-Verzögerung durch Einfügen von Wartezyklen notwendig



5.3 Pipeline-Konflikte

- **Steuerflusskonflikte durch Verzweigung**
  - Verringerung der Anzahl der Wartezyklen: Probleme
    - Probleme bei Berechnung in ID Stufe:
    - Die ALU kann nicht zur Berechnung der Sprung- bedingung benutzt werden, da sie vom vorhergehenden Befehl benötigt wird → Strukturkonflikt
    - Dekodierung, Berechnung der Sprungzieladresse/bedingung und Rückschreibung des PCs in einer Pipeline-Stufe

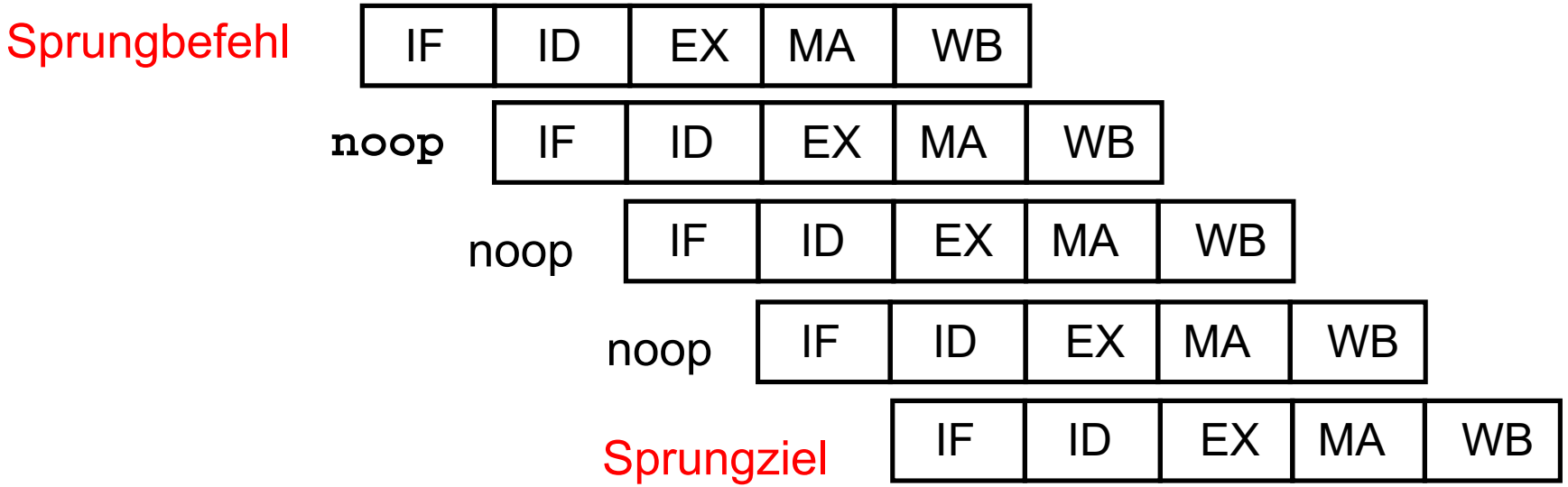


5.3 Pipeline-Konflikte

- **Steuerflusskonflikte durch Verzweigung**
  - Trotz der vorgeschlagenen Pipeline-Reorganisation bleibt ein Wartezyklus (Verzögerungsphase)
  - Bei der DLX-Pipeline werden Steuerflusskonflikte durch die Hardware weder erkannt noch behandelt, d.h. die drei Befehle hinter einem Sprungbefehl werden immer ausgeführt
  - Techniken zur Konfliktauflösung
    - Software-Techniken
    - Hardware-Techniken

Steuerflusskonflikte durch Verzweigung

- **Konfliktauflösung mit Hilfe von Software-Techniken**
  - **Verzögerte Sprungtechnik (delayed branch technique)**
    - Ausfüllen der Verzögerungszeitschlitze (delay slots) mit Leerbefehlen (noop)
    - DLX-Pipeline: Drei Leerbefehle nach jedem Programmsteuerbefehl



Steuerflusskonflikte durch Verzweigung

- **Konfliktauflösung mit Hilfe von Software-Techniken**
  - **Verzögerte Sprungtechnik (delayed branch technique)**
    - Ausfüllen der Verzögerungszeitschlitze (delay slots) mit Leerbefehlen (noop)
    - Der Compiler ordnet Befehle, die in der logischen Programmreihenfolge vor dem Sprungbefehl liegen, anstelle der Leeroperationen
      - Nur dann möglich, wenn die verschobenen Befehle keinen Einfluss auf die Sprungrichtung haben
      - Gibt es keine Befehle, die in die Verzögerungszeitschlitze verschoben werden können, müssen Leerbefehle bleiben
    - **Beispiel (ein Verzögerungsschlitz)**

```
add $s1,$s2,$s3
if $s2=0 then
```


```
if $s2=0 then
```

```
add $s1,$s2,$s3
```

Steuerflusskonflikte durch Verzweigung

- **Konfliktauflösung mit Hilfe von Hardware-Techniken**
  - **Pipeline-Leerlauf**
    - Hardware erkennt Verzweigungsbefehle (in der ID-Stufe) und lädt keine weiteren Befehle in die Pipeline, bis die Zieladresse berechnet und im Falle bedingter Sprungbefehle die Sprungentscheidung getroffen ist
    - Der eine Befehl, der bereits ins Pipeline-Register der IF-Stufe geladen wurde, muss gelöscht werden.
    - Einfachste und ineffizienteste Methode

Steuerflusskonflikte durch Verzweigung

- **Konfliktauflösung mit Hilfe von Hardware-Techniken**
  - **Sprungvorhersage**
    - Beim Auftreten einer Verzweigung: Vorhersage des Sprungziels
    - Füllen der Verzögerungsphasen spekulativ mit Befehlen, die dem Sprung folgen oder die am Sprungziel stehen
    - Nach Auswertung der Sprungbedingung:
    - Fortfahren mit der Ausführung ohne Verzögerung bei korrekter Vorhersage.
    - Verwerfen der geholten Befehle bei falscher Vorhersage

Steuerflusskonflikte durch Verzweigung

- **Konfliktauflösung mit Hilfe von Hardware-Techniken**
  - **Sprungvorhersage (statisch)**
    - Sprungvorhersage ist in Hardware fest verdrahtet
      - Annahme, dass eine Verzweigung nicht genommen wird (branch not taken)
      - Problem: Verzweigungen am Ende einer Schleife wird in den meisten Fällen falsch vorhergesagt

Steuerflusskonflikte durch Verzweigung

- **Konfliktauflösung mit Hilfe von Hardware-Techniken**
  - **Sprungvorhersage (statisch)**
    - Flexiblere Technik nutzt die im Verzweigungsbefehl stehende Information
      - Richtung des Adressoffsets des Verzweigungsbefehls
        - Vorhersageregel bei Rückwärtssprung (negativer Adressoffset, Displacement): branch taken
        - Vorhersage bei Vorwärtssprüngen (positiver Adressoffset): branch not taken
        - Im Idealfall hohe Wahrscheinlichkeit einer korrekten Vorhersage, wenn der Verzweigungsbefehl am Schleifenende steht
        - Vorhersageregel wirkt sich nachteilig aus, wenn Verzweigungsbefehl am Anfang steht

Steuerflusskonflikte durch Verzweigung

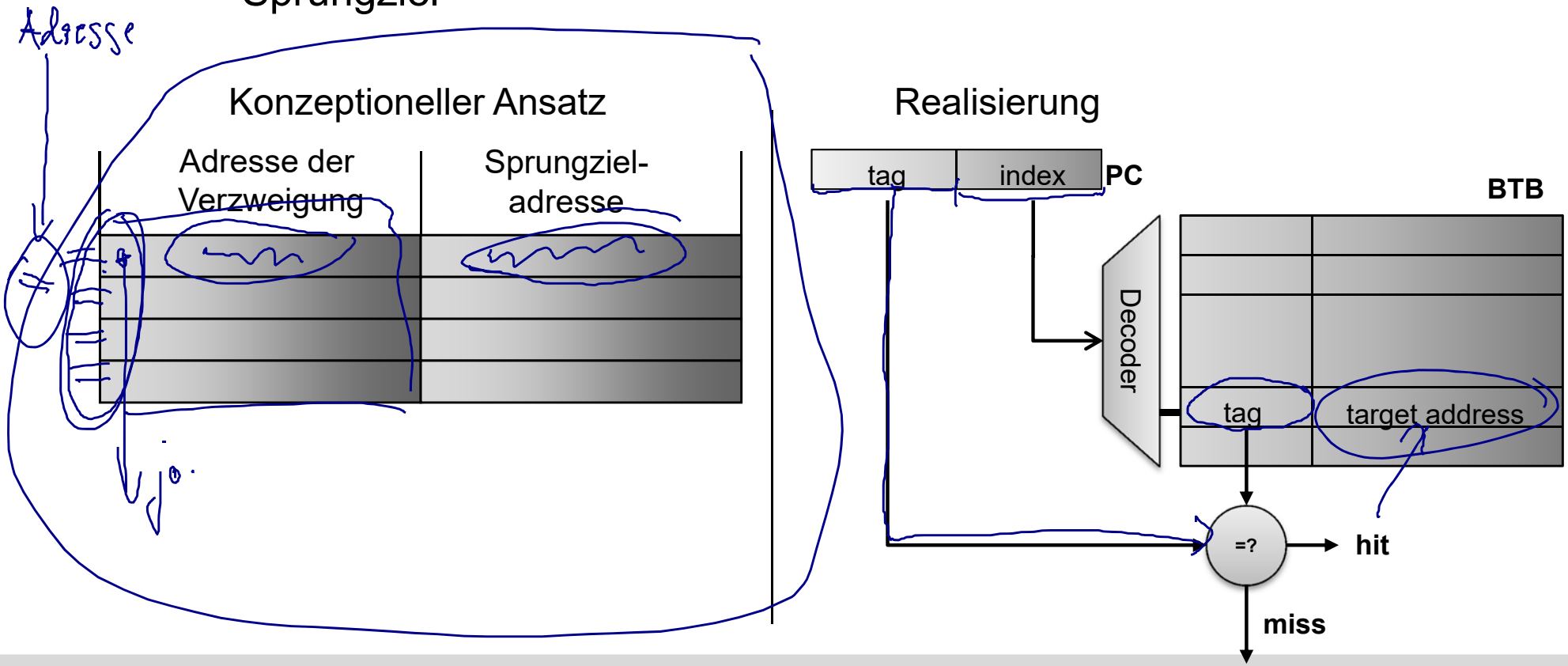
- **Konfliktauflösung mit Hilfe von Hardware-Techniken**
  - **Sprungvorhersage (statisch)**
    - Flexiblere Technik nutzt die im Verzweigungsbefehl stehende Information
      - Richtung des Adressoffsets des Verzweigungsbefehls
      - Mit Unterstützung durch Compiler:
        - Zusatzbit im Verzweigungsbefehl , mit dessen die von der Sprungrichtung abhängige Standardvorhersage negiert werden kann
        - Compiler entscheidet, ob von Standardvorhersage abgewichen werden soll oder nicht
        - Profiling

Steuerflusskonflikte durch Verzweigung

- **Konfliktauflösung mit Hilfe von Hardware-Techniken**
  - **Sprungvorhersage (dynamisch)**
    - Vorhersage von Sprüngen zur Laufzeit mit Hilfe von Laufzeitinformationen
      - Berücksichtigung des Programmverhaltens
      - Die Verzweigungsrichtung hängt von der Vorgeschichte der Verzweigung ab
      - Genauere Vorhersage möglich
      - Hoher Hardware-Aufwand!

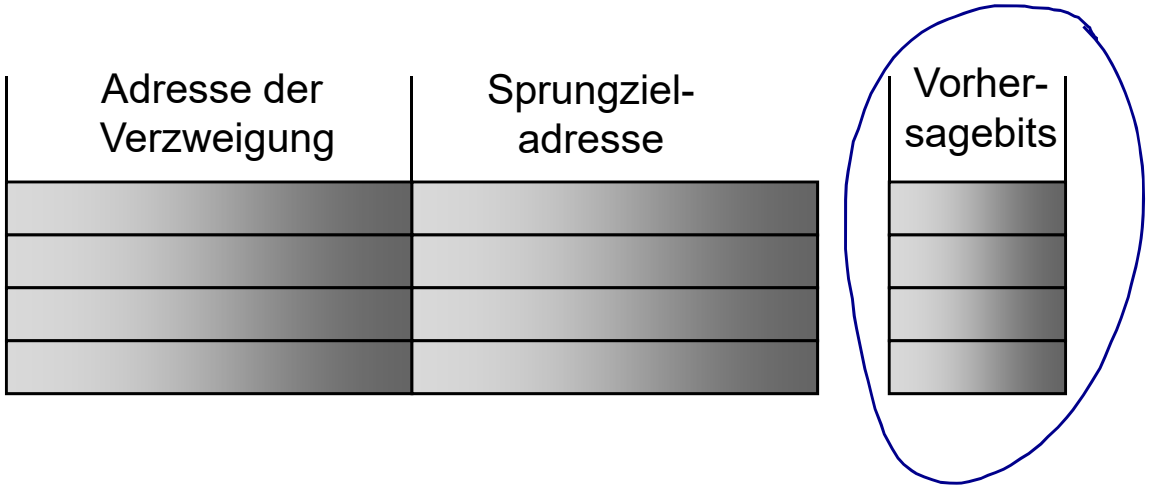
Steuerflusskonflikte durch Verzweigung

- **Konfliktauflösung mit Hilfe von Hardware-Techniken**
  - **Sprungziel-Cache:** Branch Target Address Cache (BTAC), Branch Target Buffer (BTB)
    - Speichert die Adresse der Verzweigung und das entsprechende Sprungziel



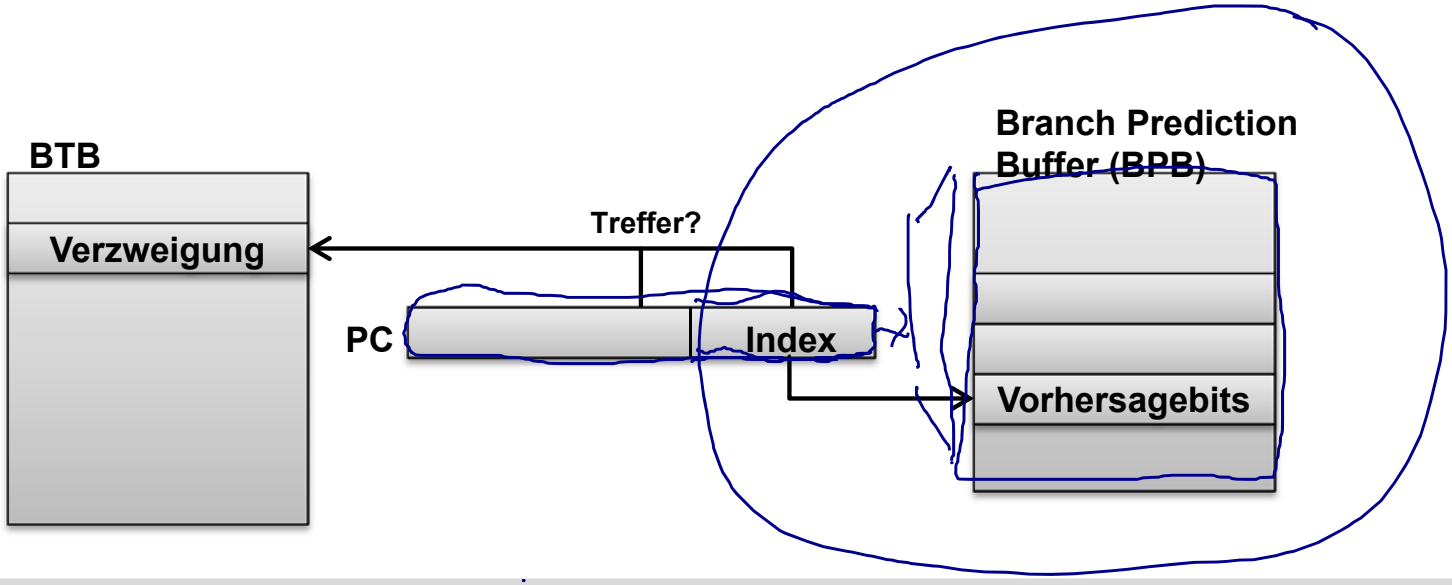
Steuerflusskonflikte durch Verzweigung

- **Konfliktauflösung mit Hilfe von Hardware-Techniken**
  - **Sprungvorhersagepuffer,** Branch Prediction Buffer (BPB)
  - Festhalten des Verhaltens der Sprungbefehle während der Ausführung des Programms: Prädiktoren
  - Vorhersage des Verhaltens eines geholten Sprungbefehls



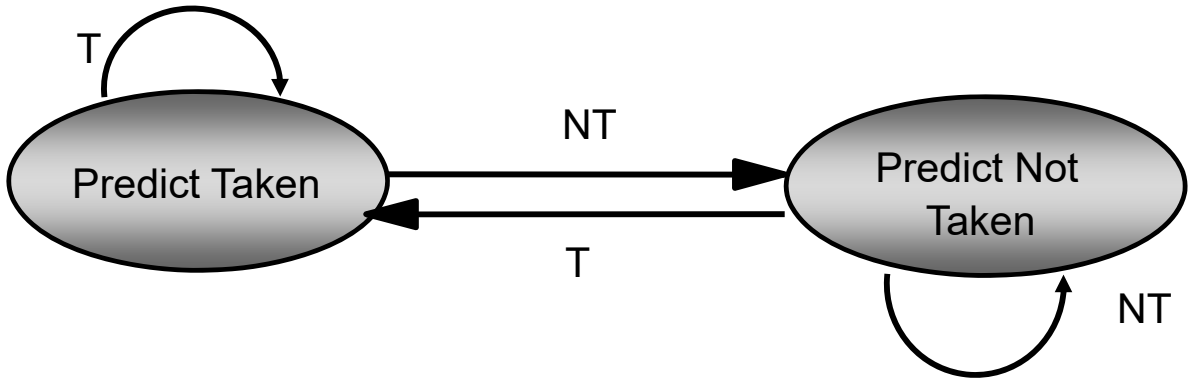
Steuerflusskonflikte durch Verzweigung

- **Konfliktauflösung mit Hilfe von Hardware-Techniken**
  - **Sprungvorhersagepuffer,** Branch Prediction Buffer (BPB)
    - Paralleler Zugriff auf BTB und BPB in der Befehlshofphase
    - Vorhersagebits werden als T (taken) oder NT (not taken) dekodiert
    - Falls die Instruktion keine Verzweigung ist, wird die Vorhersage verworfen
    - Falls die Instruktion eine Verzweigung ist, bestimmt die Vorhersage die nächste zu holende Instruktion
      - Vorhersage NT: Holen der Instruktion an (PC)+4
      - Vorhersage T: Adressrechnung oder BTB



Steuerflusskonflikte durch Verzweigung

- **Konfliktauflösung mit Hilfe von Hardware-Techniken**
  - **Prädiktoren: 1-Bit Prädiktor**
    - Jeder Eintrag in dem BPB ist ein Bit:
    - Vorhersagebit:
      - Wenn das Bit gesetzt ist, wird angenommen, dass der Sprung ausgeführt wird.
      - Wenn das Bit nicht gesetzt ist, wird angenommen, dass der Sprung nicht ausgeführt wird.
    - Bei einer Fehlannahme: Invertieren des Bits



Steuerflusskonflikte durch Verzweigung

- **Konfliktauflösung mit Hilfe von Hardware-Techniken**
  - **Prädiktoren: 1-Bit Prädiktor**
  - Prädiktoren: 1-Bit Prädiktor
    - Beispiel:

```
Loop1:  …
        …
Loop2:  …
        SUB R2,R2,1
        BNEZ R2, Loop2
        …
        BNEZ R3, Loop1
```

Init. Vorhersage NT NT NT

    - R2 wird mit 100 initialisiert
    - Wie hoch ist die Fehlerrate der Vorhersage von `BNEZ R2, Loop2` ?

Steuerflusskonflikte durch Verzweigung

- **Konfliktauflösung mit Hilfe von Hardware-Techniken**
  - **Prädiktoren: 2-Bit Prediktor mit Sättigungszähler**
    - Zwei Bit pro Eintrag für die Kodierung der Vorhersage → vier Zustände:
      - Sicher genommen (stronly taken)
      - Vielleicht genommen (weakly taken)
      - Vielleicht nicht genommen (weakly not taken)
      - Sicher nicht genommen (stronly not taken).