

# Remap the Kernel

Nach Philipp Oppermann - 05.05.2017



# Gliederung

1. Motivation
2. Wiederholung: Das Paging Modul
  - Inaktive *Tabellen*
  - Temporäres Mapping
3. Wiederholung: Rekursives Mapping
4. Konzept der Implementierung
5. Neubelegung des Kernels



# 1. Motivation

Vorherige Implementierung der  
*unmap*-Funktion führt zu stack overflow



# 1. Motivation

Vorherige Implementierung der  
*unmap*-Funktion führt zu stack overflow



Kernel stack

Page tables



# 1. Motivation

Vorherige Implementierung der *unmap*-Funktion führt zu stack overflow



Kernel stack

The diagram consists of a vertical rectangle divided into two horizontal sections. The top section is labeled 'Kernel stack' and the bottom section is labeled 'Page tables'. Both sections are filled with a light gray color and have a thin black border.

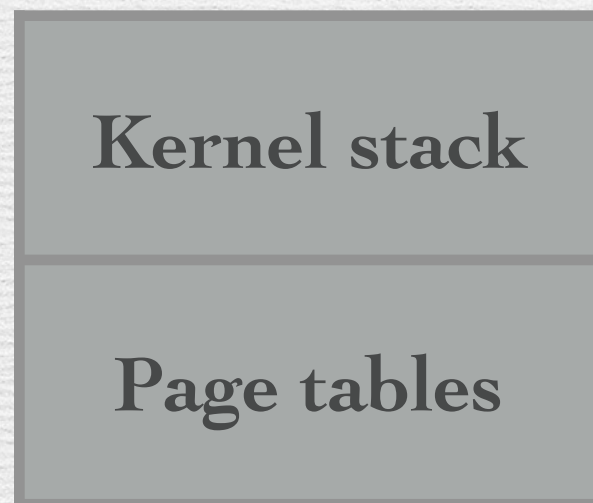
Page tables

Produziert Fehlfunktion,  
schwer zu debuggen!



# 1. Motivation

Vorherige Implementierung der  
*unmap*-Funktion führt zu stack overflow



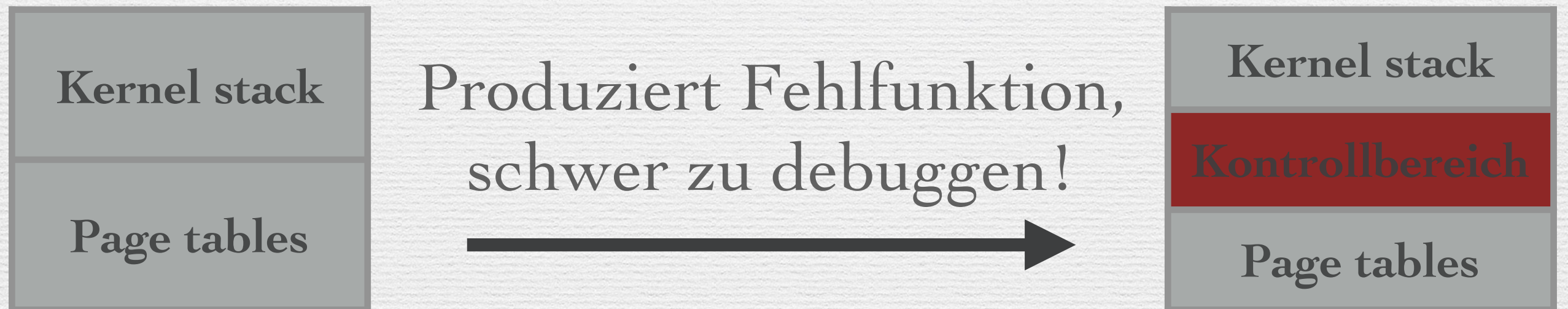
Produziert Fehlfunktion,  
schwer zu debuggen!





# 1. Motivation

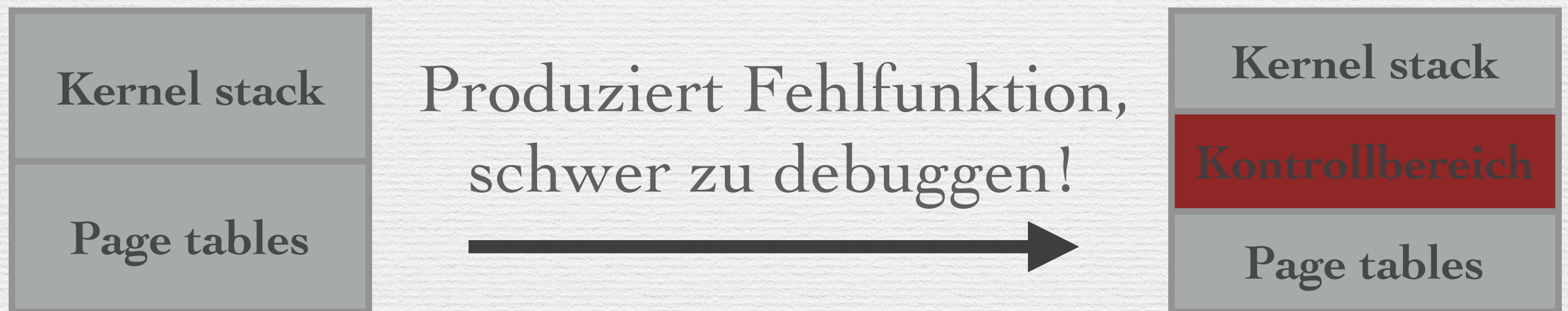
Vorherige Implementierung der *unmap*-Funktion führt zu stack overflow





# 1. Motivation

Vorherige Implementierung der *unmap*-Funktion führt zu stack overflow



Außerdem:

- Individuelles Mapping anstatt blind das erste Gigabyte zu mappen
- Nutzen von Flags zur Verbesserung der Sicherheit (.text und .data nicht änder- und ausführbar)



## 2. Wiederholung: Das Paging Modul

- Bisheriges Modul führt hierarchische Page-Tables mittels rekursivem Mapping ein
- Ziel ist eine Struktur für aktive Tabellen zu verwenden die Kernelbereiche korrekt zuordnet
- Allerdings müssen neue Tabellen erst aktiviert werden, bevor Mapping möglich ist (immediate page fault)
- Weil das nicht möglich ist, wird neue Struktur für inaktive Tabellen benötigt



## 2. Wiederholung: Das Paging Modul

### Inaktive Tabellen

```
pub struct InactivePageTable {  
    p4_frame: Frame,  
}
```



## 2. Wiederholung: Das Paging Modul

### Inaktive Tabellen

```
pub struct InactivePageTable {  
    p4_frame: Frame,  
}
```

- Neue Struktur, die P4 Tabelle verwendet (Nicht von CPU benutzt)
- Zeigt ohne Überschreibung auf zufällige Stelle
- Weil alle unnötigen eins zu eins Mappings entfernt werden, soll temporär auf virtuelle Adressen verwiesen werden



## 2. Wiederholung: Das Paging Modul

### Temporäres Mapping

```
pub struct TemporaryPage {  
    page: Page,  
    allocator: TinyAllocator,  
}
```



## 2. Wiederholung: Das Paging Modul

### Temporäres Mapping

```
pub struct TemporaryPage {  
    page: Page,  
    allocator: TinyAllocator,  
}
```

- Allokator braucht nur drei Tabellen, P4 wird immer gemappt
- Führt zu kleinem Allokator: Drei Slots mit Speicher-Tabellen
- Er ist leer, wenn Tabelle referenziert ist
- Voll, wenn dazugehörige Pagetables unreferenziert sind



## 2. Wiederholung: Das Paging Modul

### **Temporäres Mapping**



## 2. Wiederholung: Das Paging Modul

### Temporäres Mapping





## 2. Wiederholung: Das Paging Modul


### Temporäres Mapping

→ Auf diese Weise können zulässige  
Pagetables erstellt werden, die  
zurückgesetzt und rekursiv  
gemappt sind.



## 2. Wiederholung: Das Paging Modul

### Temporäres Mapping

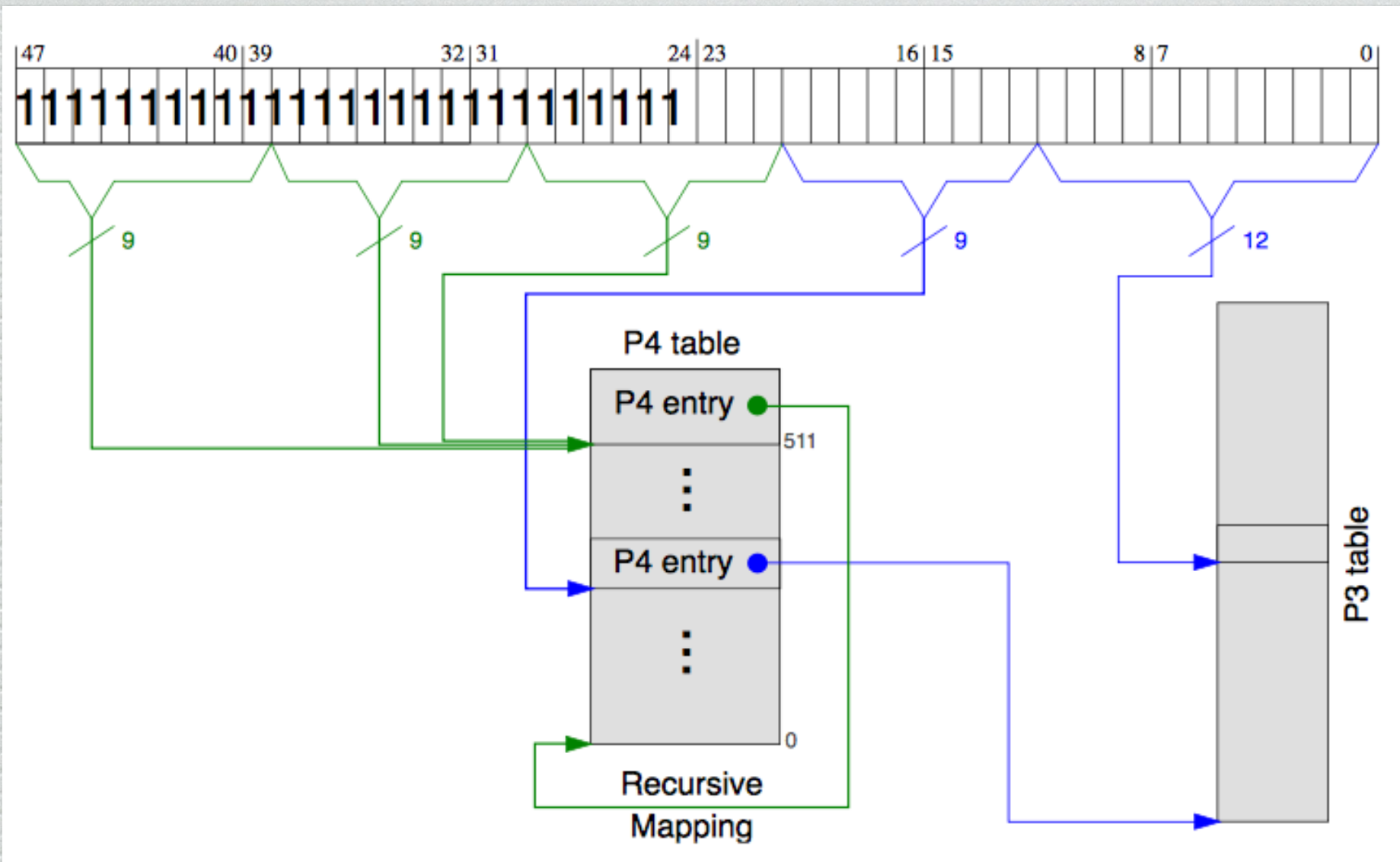
 Auf diese Weise können zulässige  
Pagetables erstellt werden, die  
zurückgesetzt und rekursiv  
gemappt sind.

- Allerdings **können** diese noch **nicht modifiziert werden**:

Dazu wird das rekursive Mapping wiederholt ...

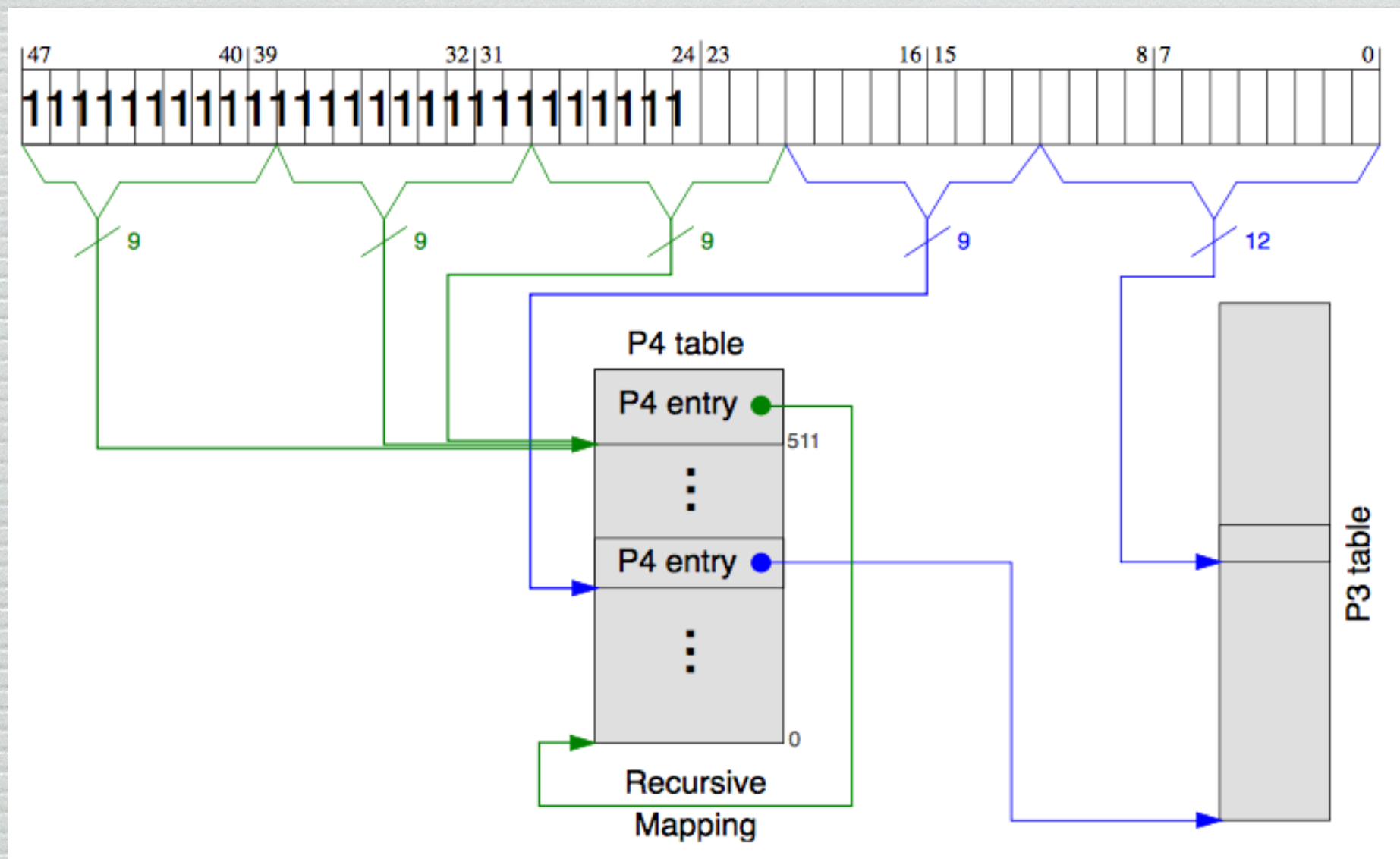


# 3. Wiederholung: Rekursives Mapping





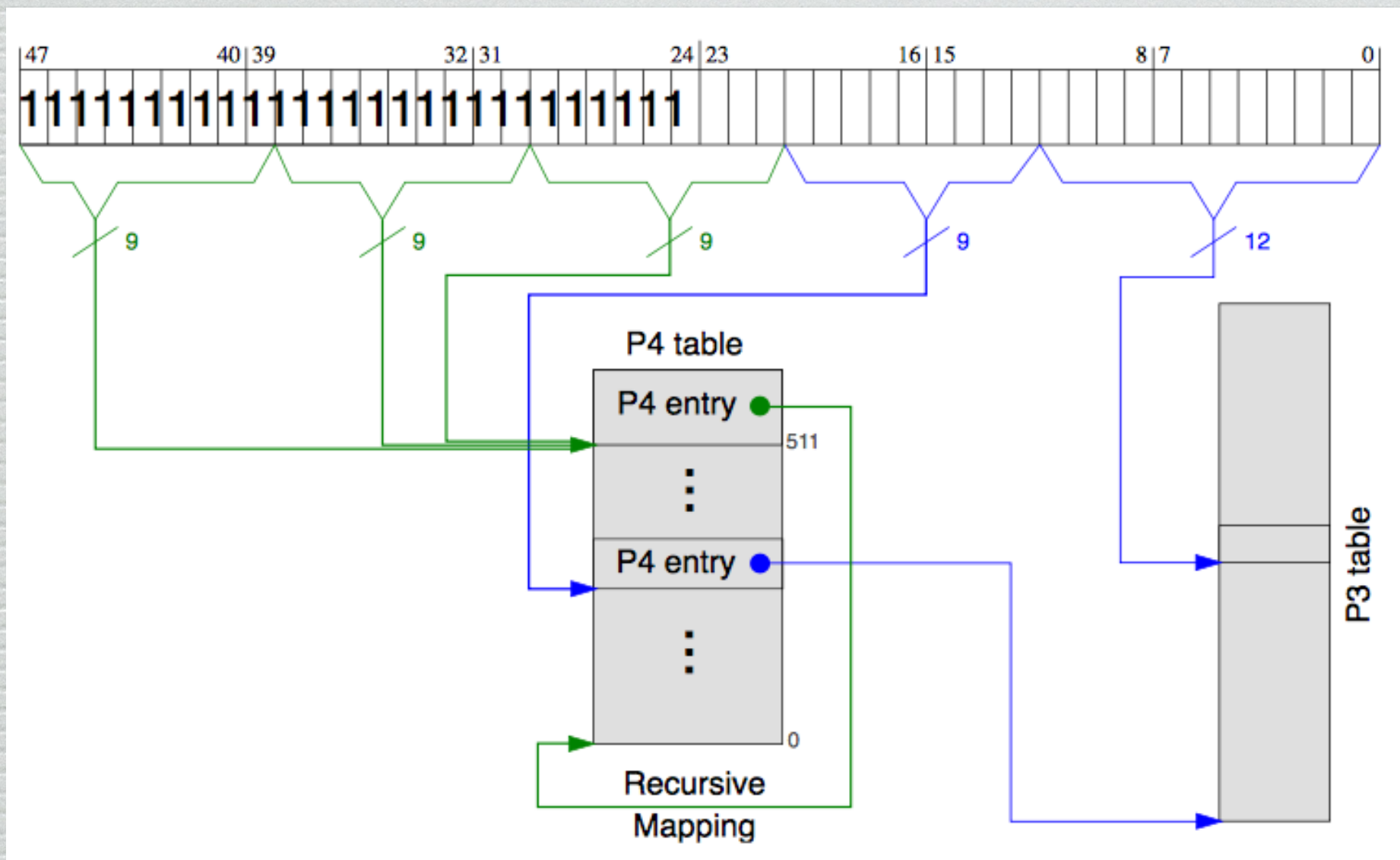
# 3. Wiederholung: Rekursives Mapping



- P4 entry verweist auf sich selbst



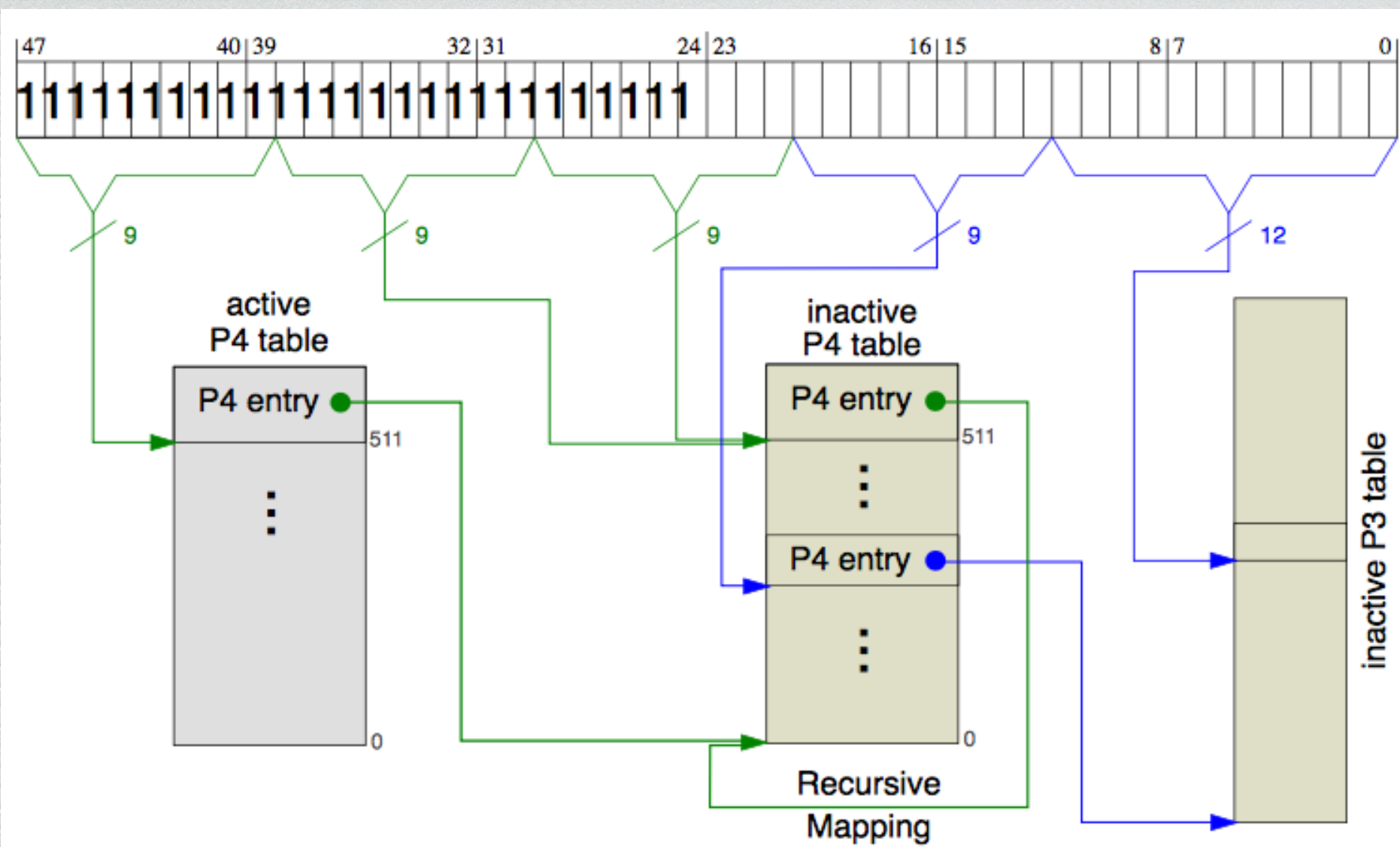
# 3. Wiederholung: Rekursives Mapping



- P4 entry verweist auf sich selbst
- Für Verweis auf P3 werden drei Loops benötigt



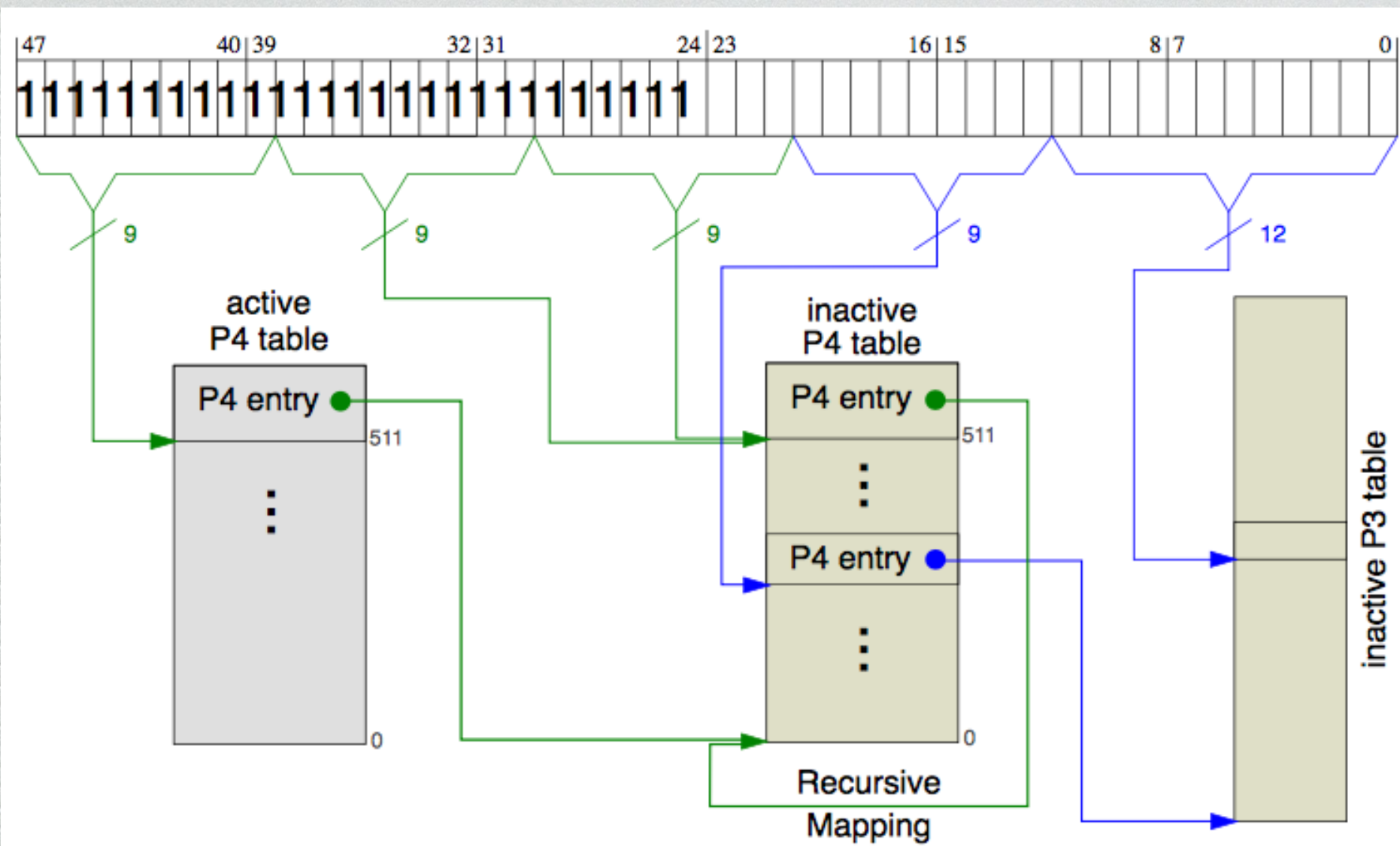
# 3. Wiederholung: Rekursives Mapping





# 3. Wiederholung: Rekursives Mapping

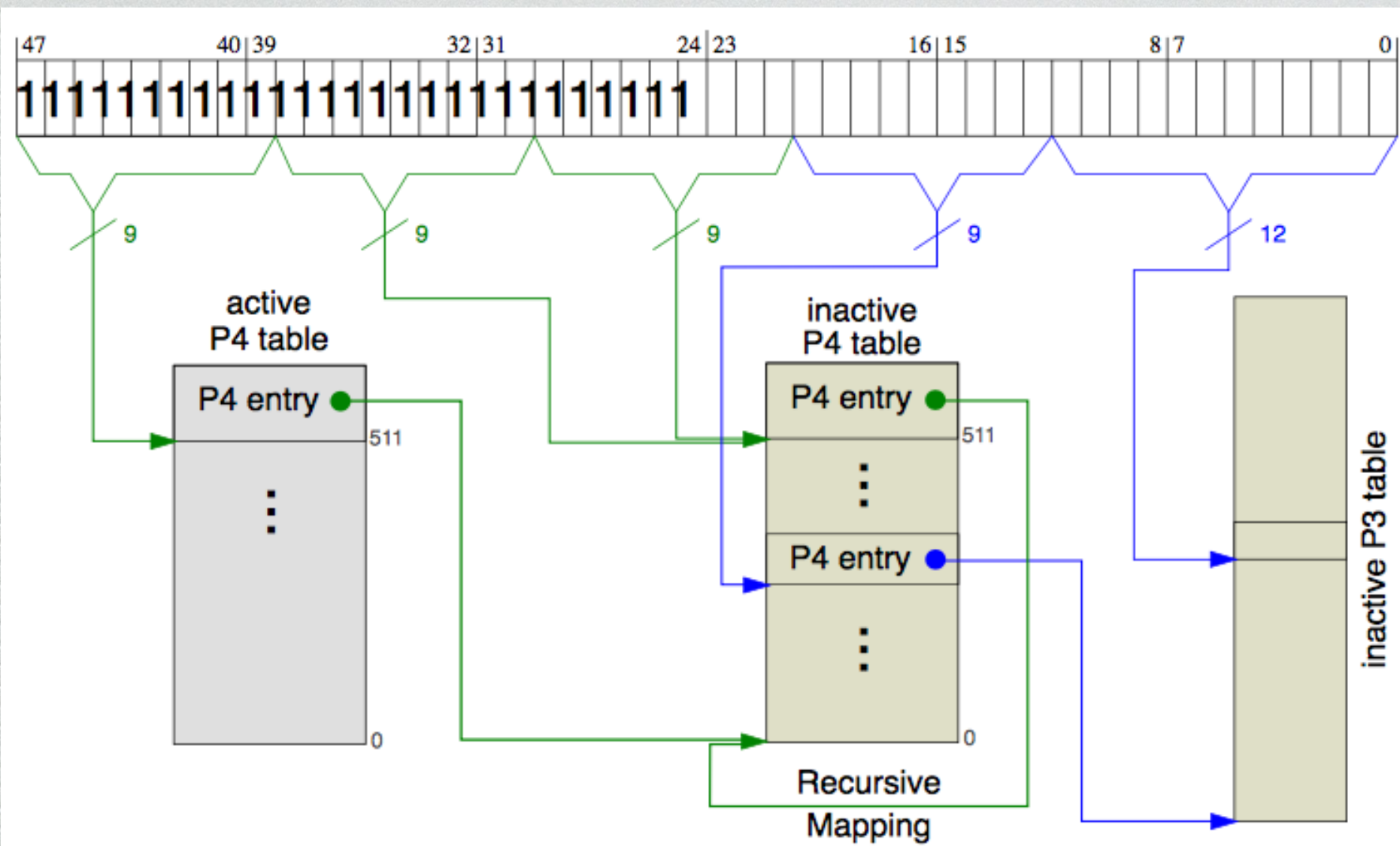
- Rekursives Mapping:  
Anstelle auf  
aktive P4  
Tabelle auf  
inaktive P4  
Tabelle





# 3. Wiederholung: Rekursives Mapping

- Rekursives Mapping:  
Anstelle auf aktive P4 Tabelle auf inaktive P4 Tabelle
- Jetzt können aktive und inaktive Tabelle gleich verarbeitet werden





# 3. Wiederholung: Rekursives Mapping



# 3. Wiederholung: Rekursives Mapping

- Nach dieser Änderung funktioniert alles wie bisher.

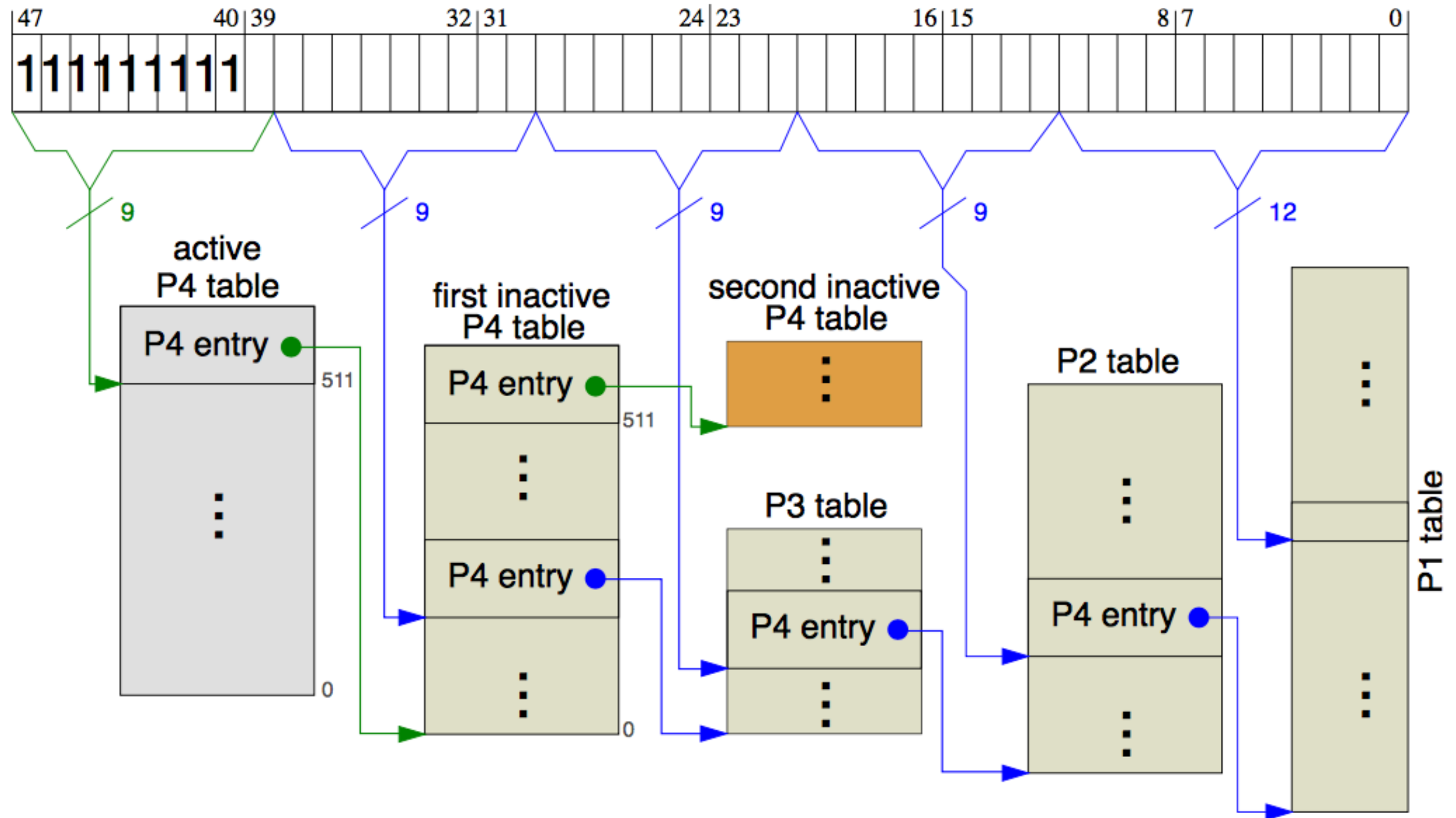
So lange die aktive Tabelle der CPU  
nicht geändert wird



# 4. Konzept der Implementierung



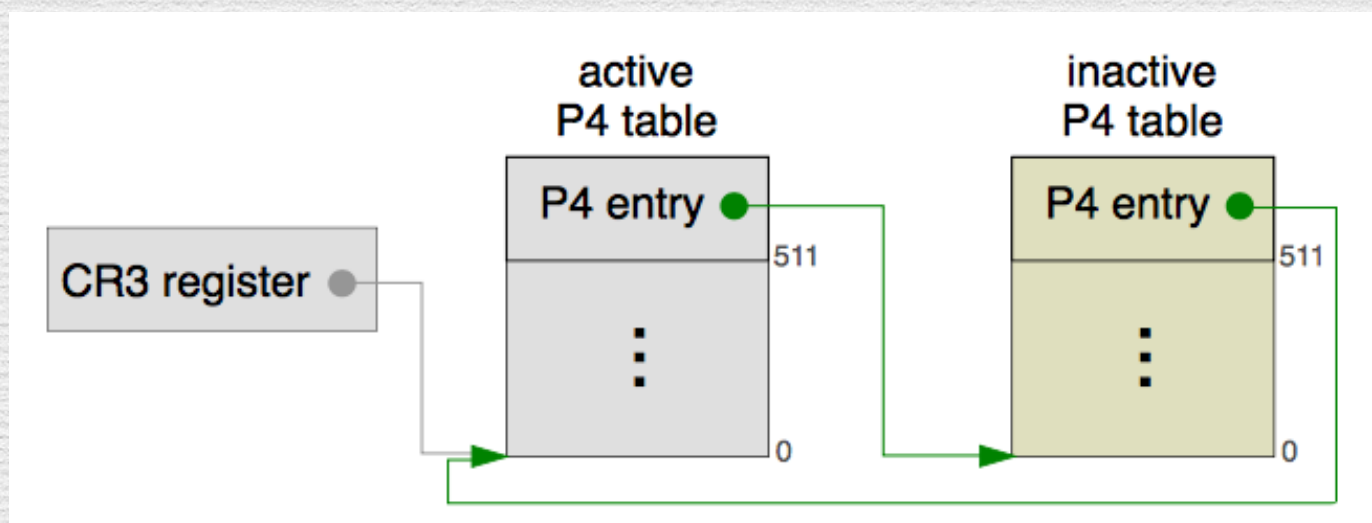
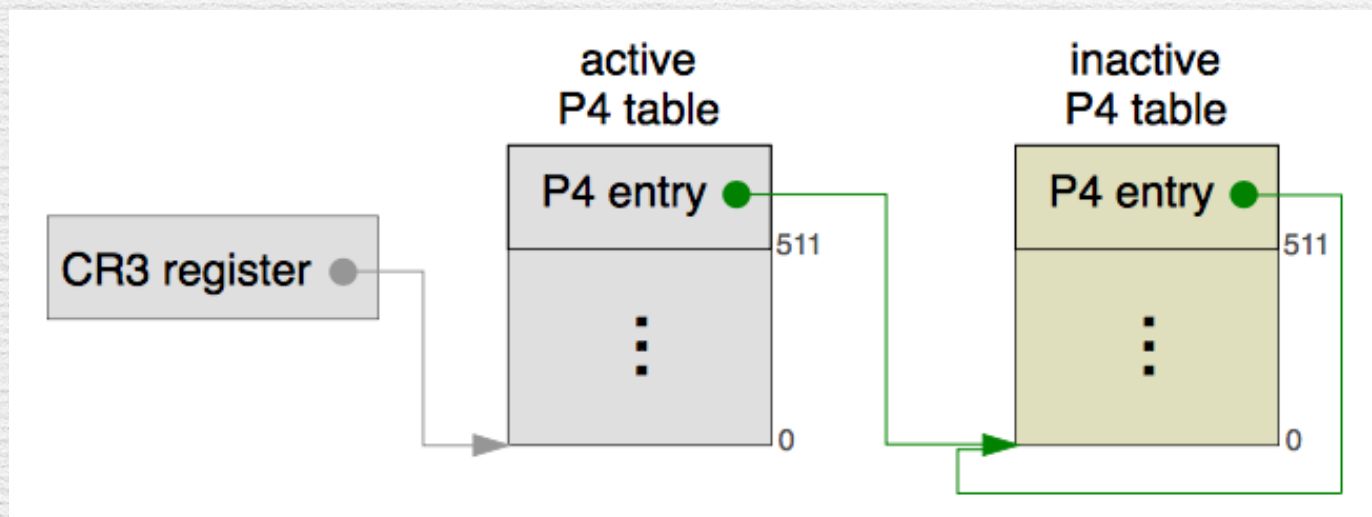
# 4. Konzept der Implementierung





# 4. Konzept der Implementierung

Wiederherstellung der Rekursion:



- Aktuell nicht möglich, da Eintrag auf sich selbst zeigt
- Versuch das rekursive Mapping zu überschreiben um Zugriff auf aktive Tabelle zu erhalten



# 4. Konzept der Implementierung

**Nachteil:** Nachdem inaktive Tabelle nicht rekursiv ist, ist sie nichtmehr zulässig.



# 4. Konzept der Implementierung

**Nachteil:** Nachdem inaktive Tabelle nicht rekursiv ist, ist sie nichtmehr zulässig.





# 4. Konzept der Implementierung

**Nachteil:** Nachdem inaktive Tabelle nicht rekursiv ist, ist sie nichtmehr zulässig.



Daher wird mit temporären Pagetables nachgeholfen



# 4. Konzept der Implementierung

**Nachteil:** Nachdem inaktive Tabelle nicht rekursiv ist, ist sie nichtmehr zulässig.



Daher wird mit temporären Pagetables nachgeholfen

Weil ohnehin eine temporäre Seite benötigt wird können wir den Aufwand vermeiden und direkt die originale P4-Seite mappen



# 5. Neubelegung des Kernels



# 5. Neubelegung des Kernels

1. Schritt: Erstellen einer temporären Seite  
(Muss ungenutzt sein)



# 5. Neubelegung des Kernels

1. Schritt: Erstellen einer temporären Seite  
(Muss ungenutzt sein)
2. Schritt: Ändern der Rekursion -  
Zusammenführung, damit neue Tabelle aktiv ist  
(Ermöglicht Mapping der Sektions ohne Veränderung des aktiven Mappings)



# 5. Neubelegung des Kernels

1. Schritt: Erstellen einer temporären Seite  
(Muss ungenutzt sein)
2. Schritt: Ändern der Rekursion -  
Zusammenführung, damit neue Tabelle aktiv ist  
(Ermöglicht Mapping der Sektions ohne Veränderung des aktiven Mappings)
3. Schritt: Überspringen aller Sektions die nicht in  
den Speicher müssen



# 5. Neubelegung des Kernels

1. Schritt: Erstellen einer temporären Seite  
(Muss ungenutzt sein)
2. Schritt: Ändern der Rekursion -  
Zusammenführung, damit neue Tabelle aktiv ist  
(Ermöglicht Mapping der Sektions ohne Veränderung des aktiven Mappings)
3. Schritt: Überspringen aller Sektions die nicht in  
den Speicher müssen
4. Schritt: Pagealignment der Sections



# 5. Neubelegung des Kernels

1. Schritt: Erstellen einer temporären Seite  
(Muss ungenutzt sein)
2. Schritt: Ändern der Rekursion -  
Zusammenführung, damit neue Tabelle aktiv ist  
(Ermöglicht Mapping der Sektions ohne Veränderung des aktiven Mappings)
3. Schritt: Überspringen aller Sektions die nicht in  
den Speicher müssen
4. Schritt: Pagealignment der Sections
5. Schritt: Erstelle eine Guard Page



Wir haben eine neue Seitentabelle erstellt, um die Kernelabschnitte korrekt abzubilden. Dazu wurde das Paging-Modul erweitert, um auch Änderungen an inaktiven Seitentabellen zu unterstützen.

Dann haben wir unseren Kernel-Stack gesichert, indem wir eine Guard-Seite erstellt haben.







Vielen Dank  
für Ihre  
Aufmerksamkeit



Vielen Dank  
für Ihre  
Aufmerksamkeit

Änderungen im Sinne des Fortschritts vorbehalten ...