

CHAPTER 4

TRANSITIONING TO LONG MODE

OVERVIEW

4.1 PAGING

4.2 SETTING UP A GDT

4.3 JUMPING HEADLONG INTO
LONG MODE

PAGING

PAGING

- Was ist *Paging*? : *Paging* ist eine Methode den Speicher zu verwalten.
- Es gilt zwei Fragen zu beantworten:
 - 1. Wieviel physikalischer Speicher ist verfügbar?
 - 8 GB = 8.5 Milliarden Bytes
 - 16 GB = 16.9 Milliarden Bytes
 - 2. Wieviel Speicherzellen können adressiert werden?
(Wieviele Speicheradressen können verwaltet werden)
 - Speicherbereich: 2^{32} oder 2^{64} sind 4.294.967.296 bzw. 18.446.744.073.709.551.616 Bytes

PAGING

PHYSICAL AND VIRTUAL ADDRESSES

- Wie kann Ungleichheit zwischen möglichen Adressen und realem Speicher behoben werden?
 - physical = real value of location (hardware)
 - virtual = address anywhere inside of space (software)
- Wegen Ineffizienz wird Speicher aufgeteilt in *chunks* oder auch *pages*
 - Umgesetzt in MMU: Memory management unit
(Übersetzt virtuelle in physikalische Adresse, Handling durch *page tables*)
 - Das Betriebssystem lädt *page table* in spezieller Datenstruktur und befiehlt der CPU die Aktivierung von Paging.

PAGING

MAPPING

- Unterschiedliche Strategien denkbar.
 - komplex \longleftrightarrow vs \longleftrightarrow simple
- Heute, einfache Methode: *Identity mapping*
- Jede *virtuelle* Adresse wird auf *physikalische* Adresse gemappt.

Virtuelle Adressen

Realer, physikalischer Speicher



PAGING MAPPING

- Unterschiedliche *page tables* möglich:
die Page-Map Level-4 Table (PML4),
die Page-Directory Pointer Table (PDP),
die Page-Directory Table (PD),
und the Page Table (PT).
- *Erstellen einer page table* (boot.asm - unten):

PAGING MAPPING

- Unterschiedliche *page tables* möglich:
die Page-Map Level-4 Table (PML4),
die Page-Directory Pointer Table (PDP),
die Page-Directory Table (PD),
und the Page Table (PT).
- *Erstellen einer page table* (boot.asm - unten):

```
section .bss

align 4096

p4_table:
    resb 4096
p3_table:
    resb 4096
p2_table:
    resb 4096
```


PAGING MAPPING

- Unterschiedliche *page tables* möglich:
die Page-Map Level-4 Table (PML4),
die Page-Directory Pointer Table (PDP),
die Page-Directory Table (PD),
und the Page Table (PT).
- *Erstellen einer page table* (boot.asm - unten):

block started by symbol
(Einträge werden vom Linker mit 0 beschrieben)

```
section .bss

align 4096

p4_table:
    resb 4096
p3_table:
    resb 4096
p2_table:
    resb 4096
```

PAGING MAPPING

- Unterschiedliche *page tables* möglich:
die Page-Map Level-4 Table (PML4),
die Page-Directory Pointer Table (PDP),
die Page-Directory Table (PD),
und the Page Table (PT).
- *Erstellen einer page table* (boot.asm - unten):

block started by symbol
(Einträge werden vom Linker mit 0 beschrieben)

```
section .bss  
  
align 4096  
  
p4_table:  
    resb 4096  
p3_table:  
    resb 4096  
p2_table:  
    resb 4096
```

PAGING MAPPING

- Unterschiedliche *page tables* möglich:
die Page-Map Level-4 Table (PML4),
die Page-Directory Pointer Table (PDP),
die Page-Directory Table (PD),
und the Page Table (PT).
- *Erstellen einer page table* (boot.asm - unten):

```
section .bss  
  
align 4096  
  
p4_table:  
    resb 4096  
p3_table:  
    resb 4096  
p2_table:  
    resb 4096
```

block started by symbol
(Einträge werden vom Linker mit 0 beschrieben)

Stellt sicher, dass Adressen korrekt ausgerichtet sind

PAGING MAPPING

- Unterschiedliche *page tables* möglich:
die Page-Map Level-4 Table (PML4),
die Page-Directory Pointer Table (PDP),
die Page-Directory Table (PD),
und the Page Table (PT).
- *Erstellen einer page table* (boot.asm - unten):

```
section .bss
```

```
align 4096
```

```
p4_table:
```

```
    resb 4096
```

```
p3_table:
```

```
    resb 4096
```

```
p2_table:
```

```
    resb 4096
```

block started by symbol

(Einträge werden vom Linker mit 0 beschrieben)

Stellt sicher, dass Adressen korrekt ausgerichtet sind

PAGING MAPPING

- Unterschiedliche *page tables* möglich:
die Page-Map Level-4 Table (PML4),
die Page-Directory Pointer Table (PDP),
die Page-Directory Table (PD),
und the Page Table (PT).
- *Erstellen einer page table* (boot.asm - unten):

```
section .bss
```

```
align 4096
```

```
p4_table:  
    resb 4096
```

```
p3_table:  
    resb 4096
```

```
p2_table:  
    resb 4096
```

block started by symbol

(Einträge werden vom Linker mit 0 beschrieben)

Stellt sicher, dass Adressen korrekt ausgerichtet sind

Reserviert Bytes für jeden Eintrag

PAGING MAPPING

- Unterschiedliche *page tables* möglich:
die Page-Map Level-4 Table (PML4),
die Page-Directory Pointer Table (PDP),
die Page-Directory Table (PD),
und the Page Table (PT).
- *Erstellen einer page table* (boot.asm - unten):

```
section .bss
```

```
align 4096
```

```
p4_table:
```

```
    resb 4096
```

```
p3_table:
```

```
    resb 4096
```

```
p2_table:
```

```
    resb 4096
```

block started by symbol

(Einträge werden vom Linker mit 0 beschrieben)

Stellt sicher, dass Adressen korrekt ausgerichtet sind

Reserviert Bytes für jeden Eintrag

PAGING MAPPING

- Page 4 beinhaltet nur 0-en, daher sind keine sinnvollen pages vorhanden. → Weitere Anpassungen nötig

(boot.asm - oben)

PAGING

MAPPING

- Page 4 beinhaltet nur 0-en, daher sind keine sinnvollen pages vorhanden. → Weitere Anpassungen nötig

(boot.asm - oben)

```
global start

section .text
bits 32
start:
    ; Point the first entry of the level 4 page table to the first entry in the
    ; p3 table
    mov eax, p3_table
    or eax, 0b11
    mov dword [p4_table + 0], eax
```


PAGING MAPPING

- Page 4 beinhaltet nur 0-en, daher sind keine sinnvollen pages vorhanden. → Weitere Anpassungen nötig

(boot.asm - oben)

Kopiere Einträge von Level-3 Tabelle in eax Register

```
global start

section .text
bits 32
start:
    ; Point the first entry of the level 4 page table to the first entry in the
    ; p3 table
    mov eax, p3_table
    or eax, 0b11
    mov dword [p4_table + 0], eax
```

PAGING MAPPING

- Page 4 beinhaltet nur 0-en, daher sind keine sinnvollen pages vorhanden. → Weitere Anpassungen nötig

(boot.asm - oben)

Kopiere Einträge von Level-3 Tabelle in eax Register

```
global start

section .text
bits 32
start:
    ; Point the first entry of the level 4 page table to the first entry in the
    ; p3 table
    mov eax, p3_table
    or eax, 0b11
    mov dword [p4_table + 0], eax
```

PAGING MAPPING

- Page 4 beinhaltet nur 0-en, daher sind keine sinnvollen pages vorhanden. → Weitere Anpassungen nötig

(boot.asm - oben)

```
global start

section .text
bits 32
start:
    ; Point the first entry of the level 4 page table to the first entry in the
    ; p3 table
    mov eax, p3_table
    or eax, 0b11
    mov dword [p4_table + 0], eax
```

Kopiere Einträge von Level-3 Tabelle in eax Register

OR mit Eintrag in eax Register und 0b11
Die Ersten zwei Bits auf 1, Rest auf 0
Erstes: Aktuell im Speicher
Zweites: Beschreibbar

PAGING MAPPING

- Page 4 beinhaltet nur 0-en, daher sind keine sinnvollen pages vorhanden. → Weitere Anpassungen nötig

(boot.asm - oben)

```
global start

section .text
bits 32
start:
    ; Point the first entry of the level 4 page table to the first entry in the
    ; p3 table
    mov eax, p3_table
    or eax, 0b11
    mov dword [p4_table + 0], eax
```

Kopiere Einträge von Level-3 Tabelle in eax Register

OR mit Eintrag in eax Register und 0b11
Die Ersten zwei Bits auf 1, Rest auf 0
Erstes: Aktuell im Speicher
Zweites: Beschreibbar

PAGING

MAPPING

- Page 4 beinhaltet nur 0-en, daher sind keine sinnvollen pages vorhanden. → Weitere Anpassungen nötig

(boot.asm - oben)

```
global start

section .text
bits 32
start:
    ; Point the first entry of the level 4 page table to the first entry in the
    ; p3 table
    mov eax, p3_table
    or eax, 0b11
    mov dword [p4_table + 0], eax
```

Kopiere Einträge von Level-3 Tabelle in eax Register

OR mit Eintrag in eax Register und 0b11

Die Ersten zwei Bits auf 1, Rest auf 0

Erstes: Aktuell im Speicher
Zweites: Beschreibbar

```
    ; Point the first entry of the level 3 page table to the first entry in the
    ; p2 table
    mov eax, p2_table
    or eax, 0b11
    mov dword [p3_table + 0], eax

    ; point each page table level two entry to a page
    mov ecx, 0          ; counter variable
.map_p2_table:
    mov eax, 0x200000    ; 2MiB
    mul ecx
    or eax, 0b10000011
    mov [p2_table + ecx * 8], eax

    inc ecx
    cmp ecx, 512
    jne .map_p2_table
```

PAGING MAPPING

- Page 4 beinhaltet nur 0-en, daher sind keine sinnvollen pages vorhanden. → Weitere Anpassungen nötig

(boot.asm - oben)

```
global start

section .text
bits 32
start:
    ; Point the first entry of the level 4 page table to the first entry in the
    ; p3 table
    mov eax, p3_table
    or eax, 0b11
    mov dword [p4_table + 0], eax
```

Kopiere Einträge von Level-3 Tabelle in eax Register

OR mit Eintrag in eax Register und 0b11

Die Ersten zwei Bits auf 1, Rest auf 0

Erstes: Aktuell im Speicher
Zweites: Beschreibbar

```
    ; Point the first entry of the level 3 page table to the first entry in the
    ; p2 table
    mov eax, p2_table
    or eax, 0b11
    mov dword [p3_table + 0], eax

    ; point each page table level two entry to a page
    mov ecx, 0          ; counter variable
.map_p2_table:
    mov eax, 0x200000    ; 2MiB
    mul ecx
    or eax, 0b10000011
    mov [p2_table + ecx * 8], eax

    inc ecx
    cmp ecx, 512
    jne .map_p2_table
```

Selbe Aktion wie zuvor

PAGING MAPPING

- Page 4 beinhaltet nur 0-en, daher sind keine sinnvollen pages vorhanden. → Weitere Anpassungen nötig

(boot.asm - oben)

```
global start

section .text
bits 32
start:
    ; Point the first entry of the level 4 page table to the first entry in the
    ; p3 table
    mov eax, p3_table
    or eax, 0b11
    mov dword [p4_table + 0], eax
```

Kopiere Einträge von Level-3 Tabelle in eax Register

OR mit Eintrag in eax Register und 0b11

Die Ersten zwei Bits auf 1, Rest auf 0

Erstes: Aktuell im Speicher
Zweites: Beschreibbar

```
    ; Point the first entry of the level 3 page table to the first entry in the
    ; p2 table
    mov eax, p2_table
    or eax, 0b11
    mov dword [p3_table + 0], eax

    ; point each page table level two entry to a page
    mov ecx, 0          ; counter variable
.map_p2_table:
    mov eax, 0x200000    ; 2MiB
    mul ecx
    or eax, 0b10000011
    mov [p2_table + ecx * 8], eax

    inc ecx
    cmp ecx, 512
    jne .map_p2_table
```

Selbe Aktion wie zuvor

PAGING MAPPING

- Page 4 beinhaltet nur 0-en, daher sind keine sinnvollen pages vorhanden. → Weitere Anpassungen nötig

(boot.asm - oben)

```
global start

section .text
bits 32
start:
    ; Point the first entry of the level 4 page table to the first entry in the
    ; p3 table
    mov eax, p3_table
    or eax, 0b11
    mov dword [p4_table + 0], eax
```

Kopiere Einträge von Level-3 Tabelle in eax Register

OR mit Eintrag in eax Register und 0b11
Die Ersten zwei Bits auf 1, Rest auf 0
Erstes: Aktuell im Speicher
Zweites: Beschreibbar

```
    ; Point the first entry of the level 3 page table to the first entry in the
    ; p2 table
    mov eax, p2_table
    or eax, 0b11
    mov dword [p3_table + 0], eax

    ; point each page table level two entry to a page
    mov ecx, 0          ; counter variable
.map_p2_table:
    mov eax, 0x200000    ; 2MiB
    mul ecx
    or eax, 0b10000011
    mov [p2_table + ecx * 8], eax

    inc ecx
    cmp ecx, 512
    jne .map_p2_table
```

Selbe Aktion wie zuvor
Start: Schleife in Assembler

PAGING MAPPING

- Page 4 beinhaltet nur 0-en, daher sind keine sinnvollen pages vorhanden. → Weitere Anpassungen nötig

(boot.asm - oben)

```
global start

section .text
bits 32
start:
    ; Point the first entry of the level 4 page table to the first entry in the
    ; p3 table
    mov eax, p3_table
    or eax, 0b11
    mov dword [p4_table + 0], eax
```

Kopiere Einträge von Level-3 Tabelle in eax Register

OR mit Eintrag in eax Register und 0b11

Die Ersten zwei Bits auf 1, Rest auf 0

Erstes: Aktuell im Speicher
Zweites: Beschreibbar

```
    ; Point the first entry of the level 3 page table to the first entry in the
    ; p2 table
    mov eax, p2_table
    or eax, 0b11
    mov dword [p3_table + 0], eax

    ; point each page table level two entry to a page
    mov ecx, 0          ; counter variable
.map_p2_table:
    mov eax, 0x200000    ; 2MiB
    mul ecx
    or eax, 0b10000011
    mov [p2_table + ecx * 8], eax

    inc ecx
    cmp ecx, 512
    jne .map_p2_table
```

Selbe Aktion wie zuvor

Start: Schleife in Assembler

PAGING MAPPING

- Page 4 beinhaltet nur 0-en, daher sind keine sinnvollen pages vorhanden. → Weitere Anpassungen nötig

(boot.asm - oben)

```
global start

section .text
bits 32
start:
    ; Point the first entry of the level 4 page table to the first entry in the
    ; p3 table
    mov eax, p3_table
    or eax, 0b11
    mov dword [p4_table + 0], eax
```

Kopiere Einträge von Level-3 Tabelle in eax Register

OR mit Eintrag in eax Register und 0b11

Die Ersten zwei Bits auf 1, Rest auf 0

Erstes: Aktuell im Speicher
Zweites: Beschreibbar

```
    ; Point the first entry of the level 3 page table to the first entry in the
    ; p2 table
    mov eax, p2_table
    or eax, 0b11
    mov dword [p3_table + 0], eax

    ; point each page table level two entry to a page
    mov ecx, 0          ; counter variable
.map_p2_table:
    mov eax, 0x200000    ; 2MiB
    mul ecx
    or eax, 0b10000011
    mov [p2_table + ecx * 8], eax

    inc ecx
    cmp ecx, 512
    jne .map_p2_table
```

Selbe Aktion wie zuvor

Start: Schleife in Assembler

2 MiB als Zahl in eax Register

PAGING MAPPING

- Page 4 beinhaltet nur 0-en, daher sind keine sinnvollen pages vorhanden. → Weitere Anpassungen nötig

(boot.asm - oben)

```
global start

section .text
bits 32
start:
    ; Point the first entry of the level 4 page table to the first entry in the
    ; p3 table
    mov eax, p3_table
    or eax, 0b11
    mov dword [p4_table + 0], eax
```

Kopiere Einträge von Level-3 Tabelle in eax Register

OR mit Eintrag in eax Register und 0b11

Die Ersten zwei Bits auf 1, Rest auf 0

Erstes: Aktuell im Speicher
Zweites: Beschreibbar

```
    ; Point the first entry of the level 3 page table to the first entry in the
    ; p2 table
    mov eax, p2_table
    or eax, 0b11
    mov dword [p3_table + 0], eax

    ; point each page table level two entry to a page
    mov ecx, 0          ; counter variable
.map_p2_table:
    mov eax, 0x200000    ; 2MiB
    mul ecx
    or eax, 0b10000011
    mov [p2_table + ecx * 8], eax

    inc ecx
    cmp ecx, 512
    jne .map_p2_table
```

Selbe Aktion wie zuvor

Start: Schleife in Assembler

2 MiB als Zahl in eax Register

PAGING MAPPING

- Page 4 beinhaltet nur 0-en, daher sind keine sinnvollen pages vorhanden. → Weitere Anpassungen nötig

(boot.asm - oben)

```
global start

section .text
bits 32
start:
    ; Point the first entry of the level 4 page table to the first entry in the
    ; p3 table
    mov eax, p3_table
    or eax, 0b11
    mov dword [p4_table + 0], eax
```

Kopiere Einträge von Level-3 Tabelle in eax Register

OR mit Eintrag in eax Register und 0b11

Die Ersten zwei Bits auf 1, Rest auf 0

Erstes: Aktuell im Speicher
Zweites: Beschreibbar

```
    ; Point the first entry of the level 3 page table to the first entry in the
    ; p2 table
    mov eax, p2_table
    or eax, 0b11
    mov dword [p3_table + 0], eax

    ; point each page table level two entry to a page
    mov ecx, 0          ; counter variable
.map_p2_table:
    mov eax, 0x200000    ; 2MiB
    mul ecx
    or eax, 0b10000011
    mov [p2_table + ecx * 8], eax

    inc ecx
    cmp ecx, 512
    jne .map_p2_table
```

Selbe Aktion wie zuvor

Start: Schleife in Assembler

2 MiB als Zahl in eax Register

Zusätzliches Bit um huge page zu aktivieren (2 MiB statt 4 KiB)

PAGING MAPPING

- Page 4 beinhaltet nur 0-en, daher sind keine sinnvollen pages vorhanden. → Weitere Anpassungen nötig

(boot.asm - oben)

```
global start

section .text
bits 32
start:
    ; Point the first entry of the level 4 page table to the first entry in the
    ; p3 table
    mov eax, p3_table
    or eax, 0b11
    mov dword [p4_table + 0], eax
```

Kopiere Einträge von Level-3 Tabelle in eax Register

OR mit Eintrag in eax Register und 0b11

Die Ersten zwei Bits auf 1, Rest auf 0

Erstes: Aktuell im Speicher
Zweites: Beschreibbar

```
    ; Point the first entry of the level 3 page table to the first entry in the
    ; p2 table
    mov eax, p2_table
    or eax, 0b11
    mov dword [p3_table + 0], eax

    ; point each page table level two entry to a page
    mov ecx, 0          ; counter variable
.map_p2_table:
    mov eax, 0x200000    ; 2MiB
    mul ecx
    or eax, 0b10000011
    mov [p2_table + ecx * 8], eax

    inc ecx
    cmp ecx, 512
    jne .map_p2_table
```

Selbe Aktion wie zuvor

Start: Schleife in Assembler

2 MiB als Zahl in eax Register

Zusätzliches Bit um huge page zu aktivieren (2 MiB statt 4 KiB)

PAGING MAPPING

- Page 4 beinhaltet nur 0-en, daher sind keine sinnvollen pages vorhanden. → Weitere Anpassungen nötig

(boot.asm - oben)

```
global start

section .text
bits 32
start:
    ; Point the first entry of the level 4 page table to the first entry in the
    ; p3 table
    mov eax, p3_table
    or eax, 0b11
    mov dword [p4_table + 0], eax
```

Kopiere Einträge von Level-3 Tabelle in eax Register

OR mit Eintrag in eax Register und 0b11

Die Ersten zwei Bits auf 1, Rest auf 0

Erstes: Aktuell im Speicher
Zweites: Beschreibbar

```
    ; Point the first entry of the level 3 page table to the first entry in the
    ; p2 table
    mov eax, p2_table
    or eax, 0b11
    mov dword [p3_table + 0], eax

    ; point each page table level two entry to a page
    mov ecx, 0          ; counter variable
.map_p2_table:
    mov eax, 0x200000    ; 2MiB
    mul ecx
    or eax, 0b10000011
    mov [p2_table + ecx * 8], eax

    inc ecx
    cmp ecx, 512
    jne .map_p2_table
```

Selbe Aktion wie zuvor

Start: Schleife in Assembler

2 MiB als Zahl in eax Register

Zusätzliches Bit um huge page zu aktivieren (2 MiB statt 4 KiB)

Multiplikation von 2 MiB mit Counter wird in fortlaufende Adresse geschrieben

PAGING MAPPING

- Page 4 beinhaltet nur 0-en, daher sind keine sinnvollen pages vorhanden. → Weitere Anpassungen nötig

(boot.asm - oben)

```
global start

section .text
bits 32
start:
    ; Point the first entry of the level 4 page table to the first entry in the
    ; p3 table
    mov eax, p3_table
    or eax, 0b11
    mov dword [p4_table + 0], eax
```

Kopiere Einträge von Level-3 Tabelle in eax Register

OR mit Eintrag in eax Register und 0b11

Die Ersten zwei Bits auf 1, Rest auf 0

Erstes: Aktuell im Speicher
Zweites: Beschreibbar

```
    ; Point the first entry of the level 3 page table to the first entry in the
    ; p2 table
    mov eax, p2_table
    or eax, 0b11
    mov dword [p3_table + 0], eax

    ; point each page table level two entry to a page
    mov ecx, 0          ; counter variable
.map_p2_table:
    mov eax, 0x200000    ; 2MiB
    mul ecx
    or eax, 0b10000011
    mov [p2_table + ecx * 8], eax

    inc ecx
    cmp ecx, 512
    jne .map_p2_table
```

Selbe Aktion wie zuvor

Start: Schleife in Assembler

2 MiB als Zahl in eax Register

Zusätzliches Bit um huge page zu aktivieren (2 MiB statt 4 KiB)

Multiplikation von 2 MiB mit Counter wird in fortlaufende Adresse geschrieben

PAGING MAPPING

- Page 4 beinhaltet nur 0-en, daher sind keine sinnvollen pages vorhanden. → Weitere Anpassungen nötig

(boot.asm - oben)

```
global start

section .text
bits 32
start:
    ; Point the first entry of the level 4 page table to the first entry in the
    ; p3 table
    mov eax, p3_table
    or eax, 0b11
    mov dword [p4_table + 0], eax
```

Kopiere Einträge von Level-3 Tabelle in eax Register

OR mit Eintrag in eax Register und 0b11

Die Ersten zwei Bits auf 1, Rest auf 0

Erstes: Aktuell im Speicher
Zweites: Beschreibbar

```
    ; Point the first entry of the level 3 page table to the first entry in the
    ; p2 table
    mov eax, p2_table
    or eax, 0b11
    mov dword [p3_table + 0], eax

    ; point each page table level two entry to a page
    mov ecx, 0          ; counter variable
.map_p2_table:
    mov eax, 0x200000    ; 2MiB
    mul ecx
    or eax, 0b10000011
    mov [p2_table + ecx * 8], eax

    inc ecx
    cmp ecx, 512
    jne .map_p2_table
```

Selbe Aktion wie zuvor

Start: Schleife in Assembler

2 MiB als Zahl in eax Register

Zusätzliches Bit um huge page zu aktivieren (2 MiB statt 4 KiB)

Multiplikation von 2 MiB mit Counter wird in fortlaufende Adresse geschrieben

Vergleich für Wiederholung der Schleife

PAGING MAPPING

- Page 4 beinhaltet nur 0-en, daher sind keine sinnvollen pages vorhanden. → Weitere Anpassungen nötig

(boot.asm - oben)

```
global start

section .text
bits 32
start:
; Point the first entry of the level 4 page table to the first entry in the
; p3 table
mov eax, p3_table
or eax, 0b11
mov dword [p4_table + 0], eax
```

Kopiere Einträge von Level-3 Tabelle in eax Register

OR mit Eintrag in eax Register und 0b11

Die Ersten zwei Bits auf 1, Rest auf 0

Erstes: Aktuell im Speicher
Zweites: Beschreibbar

```
; Point the first entry of the level 3 page table to the first entry in the
; p2 table
mov eax, p2_table
or eax, 0b11
mov dword [p3_table + 0], eax

; point each page table level two entry to a page
mov ecx, 0 ; counter variable
.map_p2_table:
mov eax, 0x200000 ; 2MiB
mul ecx
or eax, 0b10000011
mov [p2_table + ecx * 8], eax

inc ecx
cmp ecx, 512
jne .map_p2_table
```

Selbe Aktion wie zuvor

Start: Schleife in Assembler

2 MiB als Zahl in eax Register

Zusätzliches Bit um huge page zu aktivieren (2 MiB statt 4 KiB)

Multiplikation von 2 MiB mit Counter wird in fortlaufende Adresse geschrieben

Vergleich für Wiederholung der Schleife

PAGING

ENABLE PAGING

Wir haben nun eine gültige Tabelle und müssen diese dem Betriebssystem bekannt machen.

- Adresse der Level-3 Page in spezielles Register
- Aktiviere physikalisch Adresserweiterung (PAE)
- Setze das LONG-MODE-BIT
- Aktiviere Paging

PAGING

ENABLE PAGING

Wir haben nun eine gültige Tabelle und müssen diese dem Betriebssystem bekannt machen.

- Adresse der Level-3 Page in spezielles Register
- Aktiviere physikalisch Adresserweiterung (PAE)
- Setze das LONG-MODE-BIT
- Aktiviere Paging

```
; move page table address to cr3
mov eax, p4_table
mov cr3, eax

; enable PAE
mov eax, cr4
or eax, 1 << 5
mov cr4, eax

; set the long mode bit
mov ecx, 0xC0000080
rdmsr
or eax, 1 << 8
wrmsr

; enable paging
mov eax, cr0
or eax, 1 << 31
or eax, 1 << 16
mov cr0, eax
```

PAGING

ENABLE PAGING

Wir haben nun eine gültige Tabelle und müssen diese dem Betriebssystem bekannt machen.

Register benötigt Ort der Page Table

- Adresse der Level-3 Page in spezielles Register
- Aktiviere physikalisch Adresserweiterung (PAE)
- Setze das LONG-MODE-BIT
- Aktiviere Paging

```
; move page table address to cr3
mov eax, p4_table
mov cr3, eax

; enable PAE
mov eax, cr4
or eax, 1 << 5
mov cr4, eax

; set the long mode bit
mov ecx, 0xC0000080
rdmsr
or eax, 1 << 8
wrmsr

; enable paging
mov eax, cr0
or eax, 1 << 31
or eax, 1 << 16
mov cr0, eax
```



PAGING

ENABLE PAGING

Wir haben nun eine gültige Tabelle und müssen diese dem Betriebssystem bekannt machen.

- Adresse der Level-3 Page in spezielles Register
- Aktiviere physikalisch Adresserweiterung (PAE)
- Setze das LONG-MODE-BIT
- Aktiviere Paging

Register benötigt Ort der Page Table



```
; move page table address to cr3
mov eax, p4_table
mov cr3, eax

; enable PAE
mov eax, cr4
or eax, 1 << 5
mov cr4, eax

; set the long mode bit
mov ecx, 0xC0000080
rdmsr
or eax, 1 << 8
wrmsr


; enable paging
mov eax, cr0
or eax, 1 << 31
or eax, 1 << 16
mov cr0, eax
```

PAGING

ENABLE PAGING

Wir haben nun eine gültige Tabelle und müssen diese dem Betriebssystem bekannt machen.

Register benötigt Ort der Page Table



- Adresse der Level-3 Page in spezielles Register
- Aktiviere physikalisch Adresserweiterung (PAE)
- Setze das LONG-MODE-BIT
- Aktiviere Paging

```
; move page table address to cr3
mov eax, p4_table
mov cr3, eax

; enable PAE
mov eax, cr4
or eax, 1 << 5
mov cr4, eax

; set the long mode bit
mov ecx, 0xC0000080
rdmsr
or eax, 1 << 8
wrmsr

; enable paging
mov eax, cr0
or eax, 1 << 31
or eax, 1 << 16
mov cr0, eax
```

Modifiziere cr4 Wert
mit *left shift* um alternativ 100000 zu schreiben.
Mit OR den Wert des eax Registers verändern!

PAGING

ENABLE PAGING

Wir haben nun eine gültige Tabelle und müssen diese dem Betriebssystem bekannt machen.

- Adresse der Level-3 Page in spezielles Register
- Aktiviere physikalisch Adresserweiterung (PAE)
- Setze das LONG-MODE-BIT
- Aktiviere Paging

Register benötigt Ort der Page Table

```
; move page table address to cr3
mov eax, p4_table
mov cr3, eax

; enable PAE
mov eax, cr4
or eax, 1 << 5
mov cr4, eax

; set the long mode bit
mov ecx, 0xC0000080
rdmsr
or eax, 1 << 8
wrmsr

; enable paging
mov eax, cr0
or eax, 1 << 31
or eax, 1 << 16
mov cr0, eax
```

Modifiziere cr4 Wert
mit *left shift* um alternativ 100000 zu schreiben.
Mit OR den Wert des eax Registers verändern!

PAGING

ENABLE PAGING

Wir haben nun eine gültige Tabelle und müssen diese dem Betriebssystem bekannt machen.

- Adresse der Level-3 Page in spezielles Register
- Aktiviere physikalisch Adresserweiterung (PAE)
- Setze das LONG-MODE-BIT
- Aktiviere Paging

Register benötigt Ort der Page Table

```
; move page table address to cr3
mov eax, p4_table
mov cr3, eax

; enable PAE
mov eax, cr4
or eax, 1 << 5
mov cr4, eax

; set the long mode bit
mov ecx, 0xC0000080
rdmsr
or eax, 1 << 8
wrmsr

; enable paging
mov eax, cr0
or eax, 1 << 31
or eax, 1 << 16
mov cr0, eax
```

Modifiziere cr4 Wert
mit *left shift* um alternativ 100000 zu schreiben.
Mit OR den Wert des eax Registers verändern!

Lesen und schreiben eines *model specific register*

PAGING

ENABLE PAGING

Wir haben nun eine gültige Tabelle und müssen diese dem Betriebssystem bekannt machen.

- Adresse der Level-3 Page in spezielles Register
- Aktiviere physikalisch Adresserweiterung (PAE)
- Setze das LONG-MODE-BIT
- Aktiviere Paging

Register benötigt Ort der Page Table

```
; move page table address to cr3
mov eax, p4_table
mov cr3, eax

; enable PAE
mov eax, cr4
or eax, 1 << 5
mov cr4, eax

; set the long mode bit
mov ecx, 0xC0000080
rdmsr
or eax, 1 << 8
wrmsr

; enable paging
mov eax, cr0
or eax, 1 << 31
or eax, 1 << 16
mov cr0, eax
```

Modifiziere cr4 Wert
mit *left shift* um alternativ 100000 zu schreiben.
Mit OR den Wert des eax Registers verändern!

Lesen und schreiben eines *model specific register*

PAGING

ENABLE PAGING

Wir haben nun eine gültige Tabelle und müssen diese dem Betriebssystem bekannt machen.

- Adresse der Level-3 Page in spezielles Register
- Aktiviere physikalisch Adresserweiterung (PAE)
- Setze das LONG-MODE-BIT
- Aktiviere Paging

Register benötigt Ort der Page Table

```
; move page table address to cr3
mov eax, p4_table
mov cr3, eax

; enable PAE
mov eax, cr4
or eax, 1 << 5
mov cr4, eax

; set the long mode bit
mov ecx, 0xC0000080
rdmsr
or eax, 1 << 8
wrmsr

; enable paging
mov eax, cr0
or eax, 1 << 31
or eax, 1 << 16
mov cr0, eax
```

Modifiziere cr4 Wert
mit *left shift* um alternativ 100000 zu schreiben.
Mit OR den Wert des eax Registers verändern!

Lesen und schreiben eines *model specific register*

PAGING

ENABLE PAGING

Wir haben nun eine gültige Tabelle und müssen diese dem Betriebssystem bekannt machen.

- Adresse der Level-3 Page in spezielles Register
- Aktiviere physikalisch Adresserweiterung (PAE)
- Setze das LONG-MODE-BIT
- Aktiviere Paging

Register benötigt Ort der Page Table

```
; move page table address to cr3
mov eax, p4_table
mov cr3, eax

; enable PAE
mov eax, cr4
or eax, 1 << 5
mov cr4, eax

; set the long mode bit
mov ecx, 0xC0000080
rdmsr
or eax, 1 << 8
wrmsr

; enable paging
mov eax, cr0
or eax, 1 << 31
or eax, 1 << 16
mov cr0, eax
```

Modifiziere cr4 Wert
mit *left shift* um alternativ 100000 zu schreiben.
Mit OR den Wert des eax Registers verändern!

Lesen und schreiben eines *model specific register*

Setzte Bit 16 und 31

PAGING

ENABLE PAGING

Wir haben nun eine gültige Tabelle und müssen diese dem Betriebssystem bekannt machen.

- Adresse der Level-3 Page in spezielles Register
- Aktiviere physikalisch Adresserweiterung (PAE)
- Setze das LONG-MODE-BIT
- Aktiviere Paging

Register benötigt Ort der Page Table

```
; move page table address to cr3
mov eax, p4_table
mov cr3, eax

; enable PAE
mov eax, cr4
or eax, 1 << 5
mov cr4, eax

; set the long mode bit
mov ecx, 0xC0000080
rdmsr
or eax, 1 << 8
wrmsr

; enable paging
mov eax, cr0
or eax, 1 << 31
or eax, 1 << 16
mov cr0, eax
```

Modifiziere cr4 Wert
mit *left shift* um alternativ 100000 zu schreiben.
Mit OR den Wert des eax Registers verändern!

Lesen und schreiben eines *model specific register*

Setzte Bit 16 und 31

“

TECHNICALLY WE ARE IN
LONG-MODE

— *But not in real long mode*

”

SETTING UP A GDT

JUMPING HEADLONG
INTO LONG MODE