Betriebssysteme

6. Tutorium - Buddy Allocator, Paging, TLB

Peter Bohner

6. Dezember 2024

ITEC - Operating Systems Group

Buddy Allocator

So far we’ve seen

- Consistent large blocks ⇒ Low external, high internal fragmentation
- Fitted blocks ⇒ High external, low internal fragmentation

Buddy Allocator

So far we've seen

- Consistent large blocks ⇒ Low external, high internal fragmentation
- Fitted blocks ⇒ High external, low internal fragmentation

Can we do better for some applications? Any ideas?

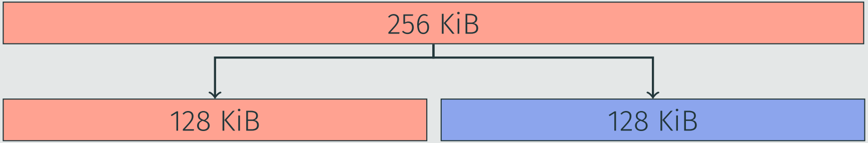
Buddy Allocator

Allocator

256 KiB

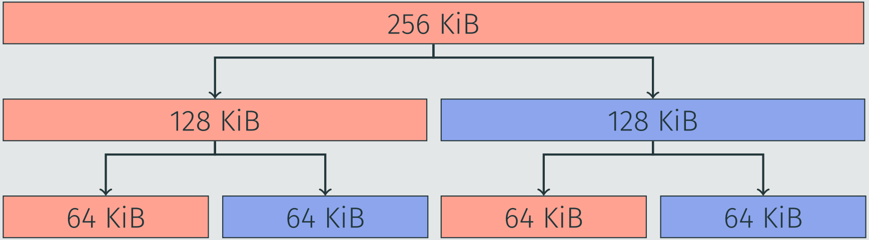
Buddy Allocator

Allocator



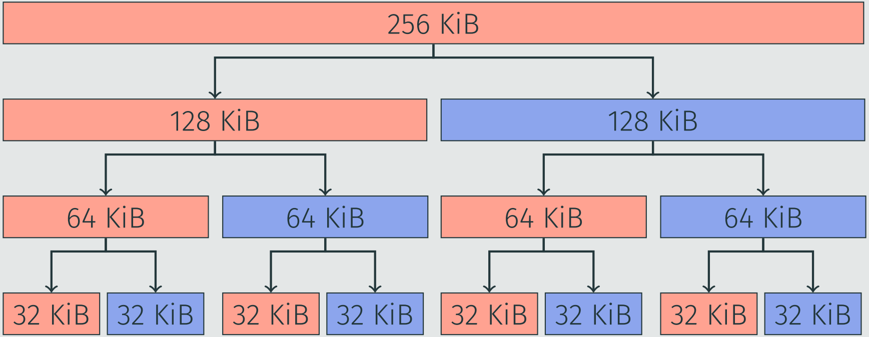
Buddy Allocator

Allocator



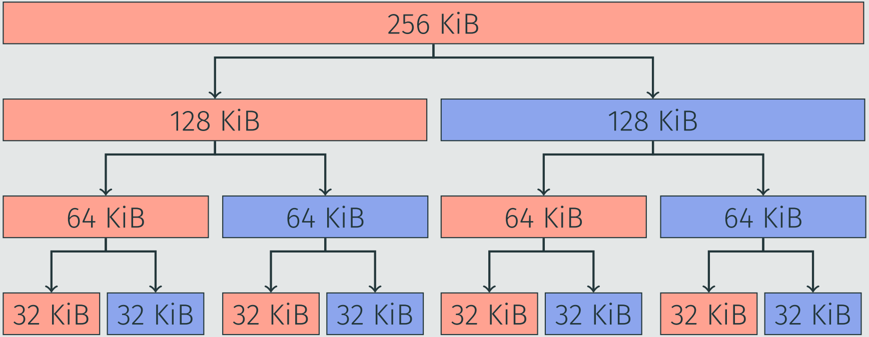
Buddy Allocator

Allocator



Buddy Allocator

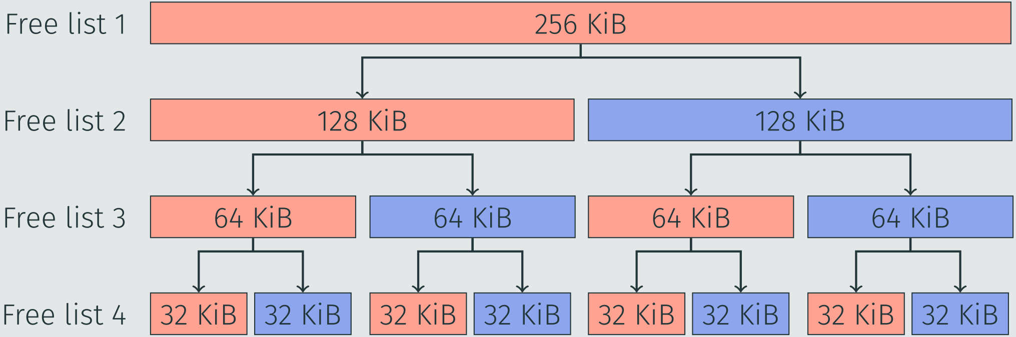
Allocator



How do you find a fitting Element?

Buddy Allocator

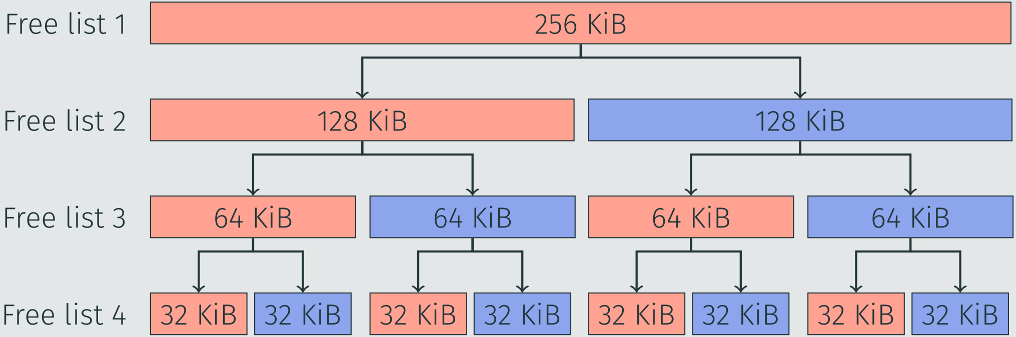
Allocator



How do you find a fitting Element? *Freelist!*

Buddy Allocator

Allocator

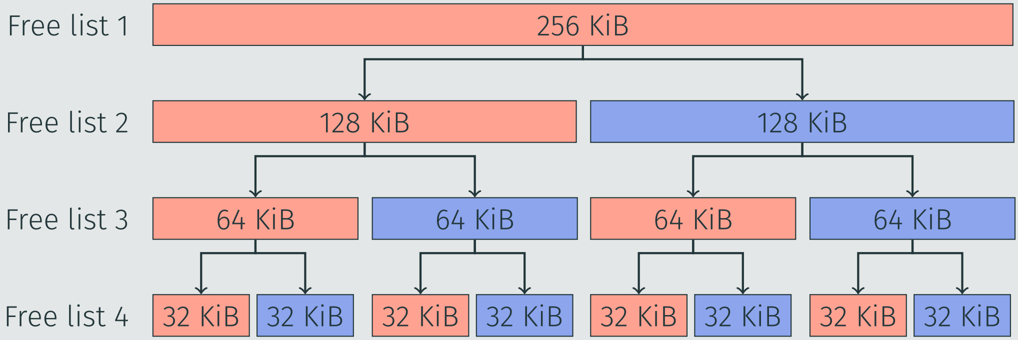


How do you find a fitting Element? *Freelist!*

And if there is no such block?

Buddy Allocator

Allocator

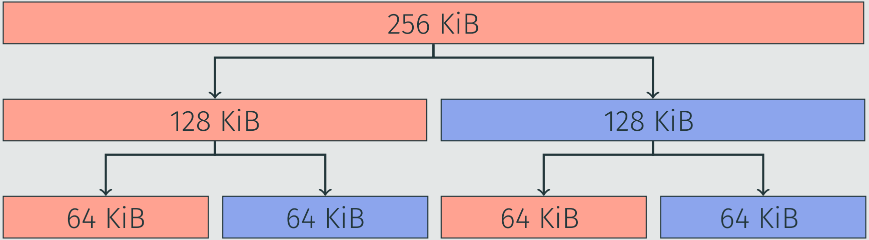


How do you find a fitting Element? *Freelist!*

And if there is no such block? *Recursively split a higher-up block*

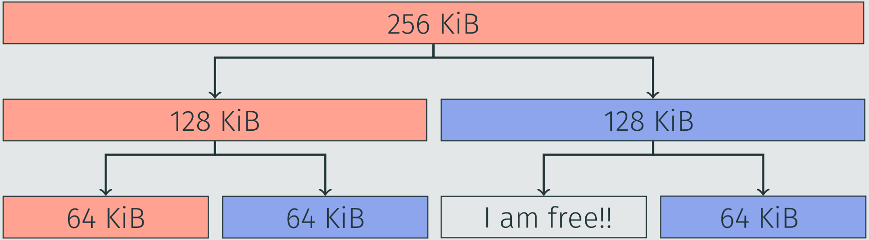
Buddy Allocator - Merging

Merging



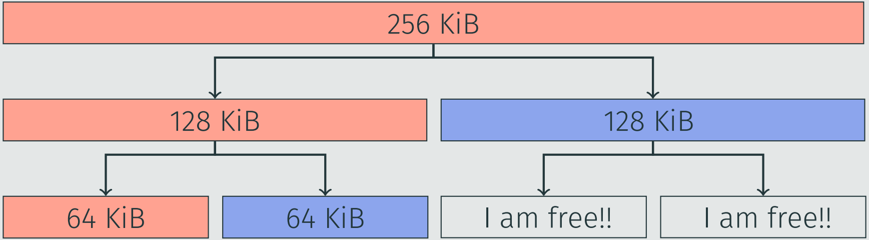
Buddy Allocator - Merging

Merging



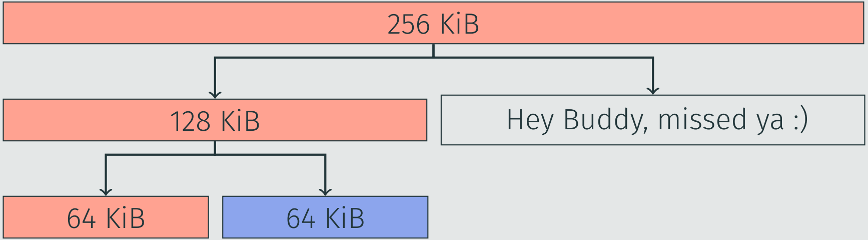
Buddy Allocator - Merging

Merging



Buddy Allocator - Merging

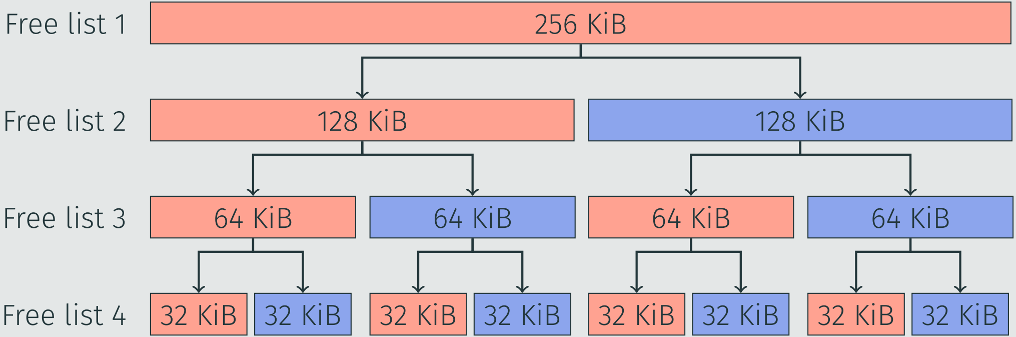
Merging



Buddy Allocator

How small/large can the free list be?

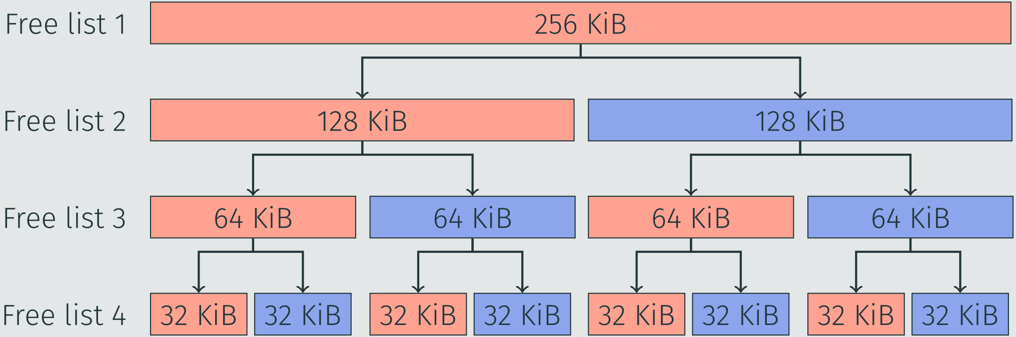
Allocate 2^m chunk of memory in a managed Block of 2^k (here: $k = 18$, as $256\ KiB = 2^{18}$)



Buddy Allocator

How small/large can the free list be?

Allocate 2^m chunk of memory in a managed Block of 2^k (here: $k = 18$, as $256\ KiB = 2^{18}$)



$\Rightarrow$ Max size $\frac{1}{2} \cdot 2^{k-m}$

$\Rightarrow$ Min size 0

What kind of fragmentation can occur?

Internal fragmentation

What kind of fragmentation can occur?

Internal fragmentation

- Power of two blocks

$\Rightarrow$ Request memory of size $2^k + 1, k \in \mathbb{N}_0$

What kind of fragmentation can occur?

Internal fragmentation

- Power of two blocks

$\Rightarrow$ Request memory of size $2^k + 1, k \in \mathbb{N}_0$

External fragmentation

What kind of fragmentation can occur?

Internal fragmentation

- Power of two blocks

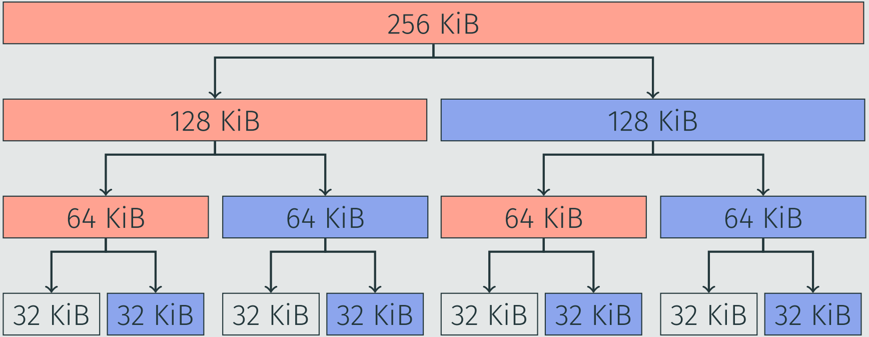
$\Rightarrow$ Request memory of size $2^k + 1, k \in \mathbb{N}_0$

External fragmentation

- Free every other block in a level

Buddy Allocator - Fragmentation

External fragmentation



Buddy Allocator - Fragmentation

But this works alright for larger sizes. So combine it with...

...

Buddy Allocator - Fragmentation

But this works alright for larger sizes. So combine it with...

...the Slab allocator! Allocate large chunks with the buddy allocator and small chunks within them using the slab allocators

What allocator would you make up for this?

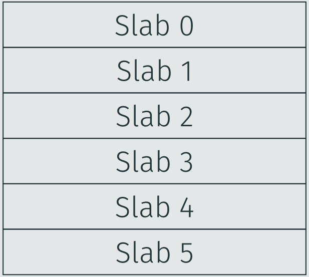
You are a poor kernel and you need lots of inodes

Every inode has the same size, 64 Byte. Can you think of any fast allocation strategy that does not waste a single bit?

What allocator would you make up for this?

You are a poor kernel and you need lots of inodes

Every inode has the same size, 64 Byte. Can you think of any fast allocation strategy that does not waste a single bit?



Slab 0

Slab 1

Slab 2

Slab 3

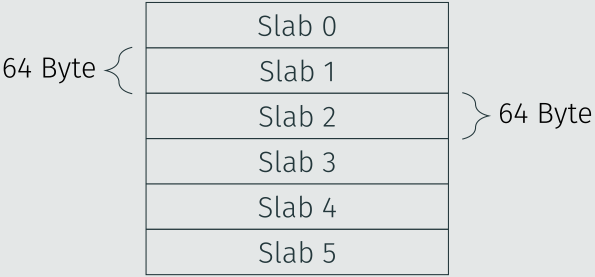
Slab 4

Slab 5

What allocator would you make up for this?

You are a poor kernel and you need lots of inodes

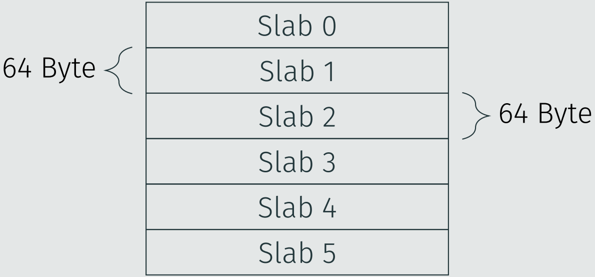
Every inode has the same size, 64 Byte. Can you think of any fast allocation strategy that does not waste a single bit?



What allocator would you make up for this?

You are a poor kernel and you need lots of inodes

Every inode has the same size, 64 Byte. Can you think of any fast allocation strategy that does not waste a single bit?



This is called a *Slab allocator*

Paging

Paging

What is that? What is the difference to Segmentation?

Paging

What is that? What is the difference to Segmentation?

- Virtual memory is broken into fixed-size chunks (pages)

Paging

What is that? What is the difference to Segmentation?

- Virtual memory is broken into fixed-size chunks (pages)
- Physical memory is broken into fixed-size chunks of the same size (frames)

Paging

What is that? What is the difference to Segmentation?

- Virtual memory is broken into fixed-size chunks (pages)
- Physical memory is broken into fixed-size chunks of the same size (frames)
- Virtual pages are mapped to page frames

Paging

What is that? What is the difference to Segmentation?

- Virtual memory is broken into fixed-size chunks (pages)
- Physical memory is broken into fixed-size chunks of the same size (frames)
- Virtual pages are mapped to page frames Always?

Paging

What is that? What is the difference to Segmentation?

- Virtual memory is broken into fixed-size chunks (pages)
- Physical memory is broken into fixed-size chunks of the same size (frames)
- Virtual pages are mapped to page frames Always?
  No! (Paged out, zero pages, ...)

Benefits over Segmentation?

Paging

What is that? What is the difference to Segmentation?

- Virtual memory is broken into fixed-size chunks (pages)
- Physical memory is broken into fixed-size chunks of the same size (frames)
- Virtual pages are mapped to page frames Always?

  No! (Paged out, zero pages, ...)

Benefits over Segmentation?

- Virtual memory does not need to map to *continuous* physical memory

Paging

What is that? What is the difference to Segmentation?

- Virtual memory is broken into fixed-size chunks (pages)
- Physical memory is broken into fixed-size chunks of the same size (frames)
- Virtual pages are mapped to page frames Always?
  No! (Paged out, zero pages, …)

Benefits over Segmentation?

- Virtual memory does not need to map to *continuous* physical memory
- Swapping in/out is easier

Paging

What is that? What is the difference to Segmentation?

- Virtual memory is broken into fixed-size chunks (pages)
- Physical memory is broken into fixed-size chunks of the same size (frames)
- Virtual pages are mapped to page frames Always?
  No! (Paged out, zero pages, ...)

Benefits over Segmentation?

- Virtual memory does not need to map to *continuous* physical memory
- Swapping in/out is easier
- No external fragmentation, little internal

Paging - Single Level Page Table

Segment and Page tables

Segment Number	Base	Limit
0	`0xdead`	`0x00ef`
1	`0xf154`	`0x013a`
2	`0x0000`	`0x0000`
3	`0x0000`	`0x3fff`

Paging - Single Level Page Table

Segment and Page tables

Virtual page number	Base	Limit
0	`0xdead`	`0x00ef`
1	`0xf154`	`0x013a`
2	`0x0000`	`0x0000`
3	`0x0000`	`0x3fff`

Paging - Single Level Page Table

Segment and Page tables

Virtual page number	Frame number	Limit
0	`0xdead`	`0x00ef`
1	`0xf154`	`0x013a`
2	`0x0000`	`0x0000`
3	`0x0000`	`0x3fff`

Paging - Single Level Page Table

Segment and Page tables

Virtual page number	Frame number	~~Limit~~
0	`0xdead`	`0x00ef`
1	`0xf154`	`0x013a`
2	`0x0000`	`0x0000`
3	`0x0000`	`0x3fff`

Paging - Single Level Page Table

Segment and Page tables

```
0x020123
```

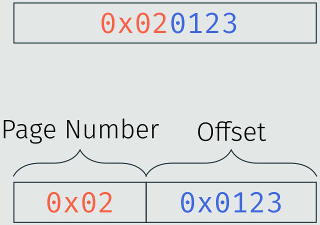
Paging - Single Level Page Table

Segment and Page tables

`0x020123`

Paging - Single Level Page Table

Segment and Page tables



0x020123

Page Number

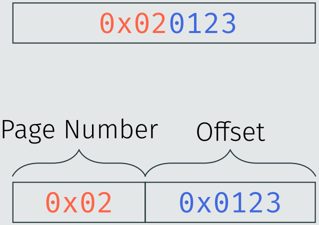
Offset

0x02

0x0123

Paging - Single Level Page Table

Segment and Page tables



0x020123

Page Number

Offset

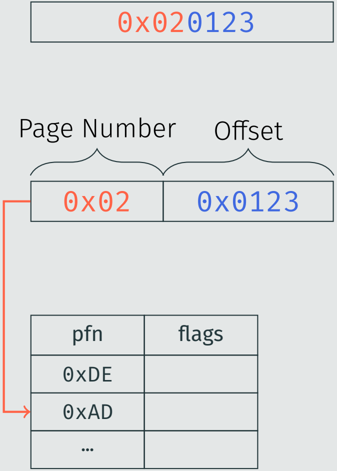
0x02

0x0123

pfn	flags
0xDE	
0xAD	
...	

Paging - Single Level Page Table

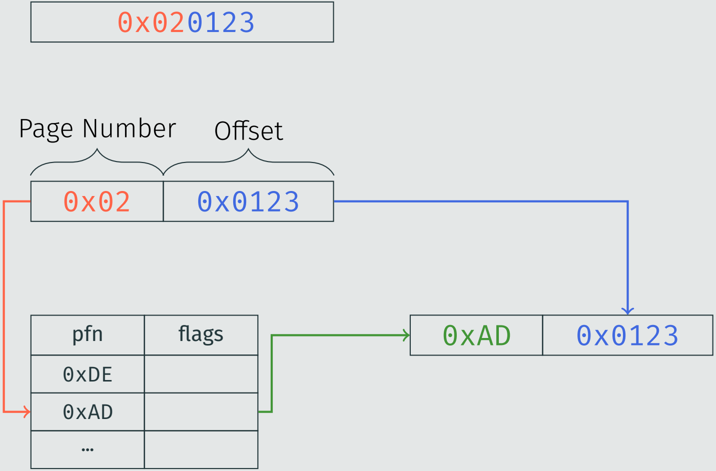
Segment and Page tables



pfn	flags
0xDE	
0xAD	
...	

Paging - Single Level Page Table

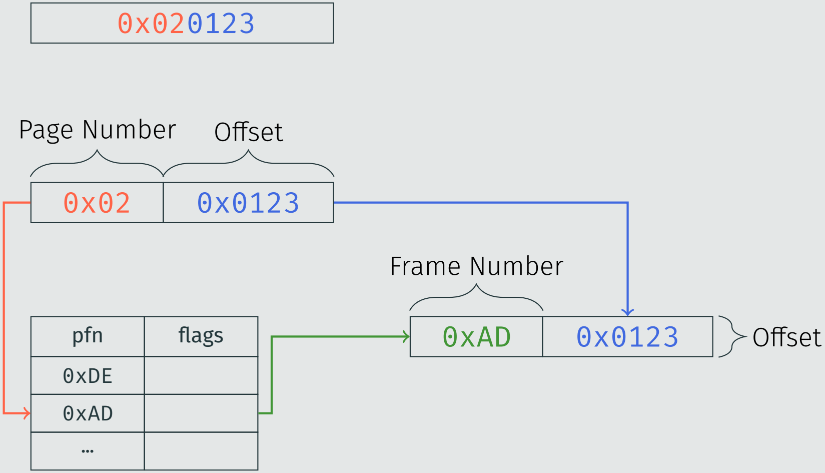
Segment and Page tables



pfn	flags
0xDE	
0xAD	
...	

Paging - Single Level Page Table

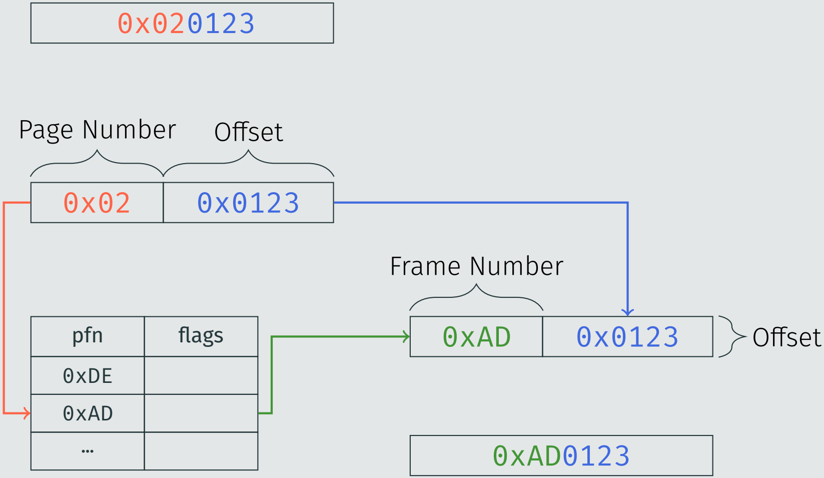
Segment and Page tables



pfn	flags
0xDE	
0xAD	
...	

Paging - Single Level Page Table

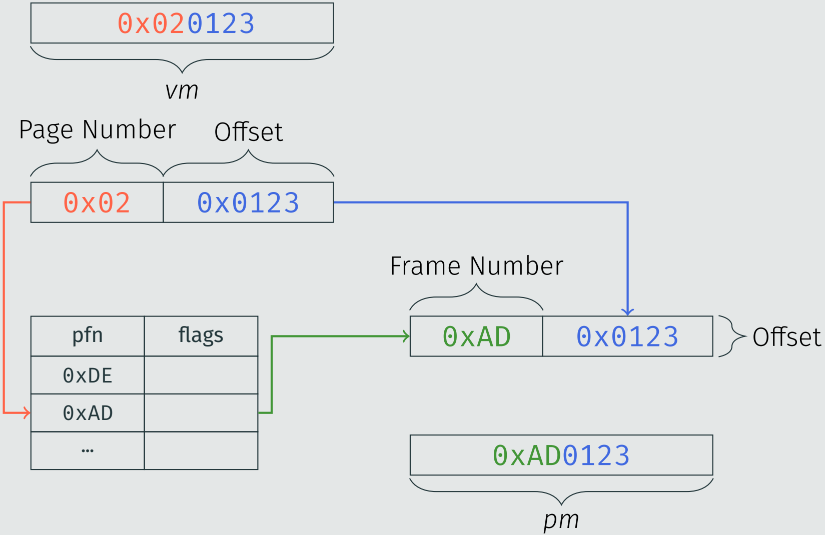
Segment and Page tables



pfn	flags
0xDE	
0xAD	
...	

Paging - Single Level Page Table

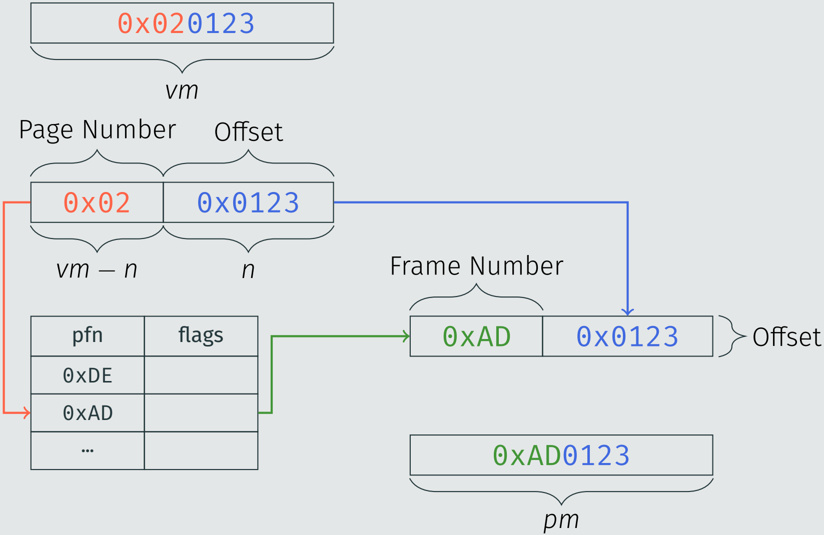
Segment and Page tables



pfn	flags
0xDE	
0xAD	
...	

Paging - Single Level Page Table

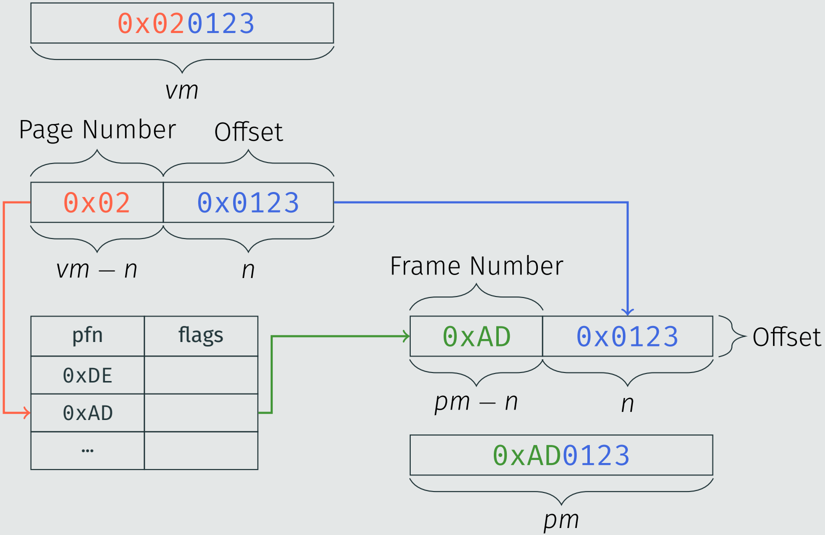
Segment and Page tables



pfn	flags
0xDE	
0xAD	
...	

Paging - Single Level Page Table

Segment and Page tables



pfn	flags
0xDE	
0xAD	
...	

Single Level Page Table - Disadvantages

Why could that be a bit problematic?

Single Level Page Table - Disadvantages

Why could that be a bit problematic?

- Increases with the size of the size of the *virtual* address space

Single Level Page Table - Disadvantages

Why could that be a bit problematic?

- Increases with the size of the size of the *virtual* address space
- 64 Bit AS, 4KiB (2^{12}) pages $\Rightarrow n = 12 \Rightarrow$

Single Level Page Table - Disadvantages

Why could that be a bit problematic?

- Increases with the size of the size of the *virtual* address space
- 64 Bit AS, 4KiB (2^{12}) pages $\Rightarrow n = 12 \Rightarrow 2^{vm-n} = 2^{64-12} = 2^{52}$

Single Level Page Table - Disadvantages

Why could that be a bit problematic?

- Increases with the size of the size of the *virtual* address space
- 64 Bit AS, 4KiB (2^{12}) pages $\Rightarrow n = 12 \Rightarrow 2^{vm-n} = 2^{64-12} = 2^{52}$

$\Rightarrow$ If every entry was 1 Bit we'd need (asking `units`...)

```
You have: 2^52 bit
You want: tebibyte
        * 512
        / 0.001953125
```

- You might *not* have that much memory to spare :)

Single Level Page Table - Disadvantages

Math is fun, let’s do some math

Calculate the space requirements for a single level page table with

- 32-bit virtual addresses, 4KiB pages, 4 bytes per page table entry
- 48-bit virtual addresses, 4KiB pages, 4 bytes per page table entry

Single Level Page Table - Disadvantages

Math is fun, let's do some math

Calculate the space requirements for a single level page table with

- 32-bit virtual addresses, 4KiB pages, 4 bytes per page table entry
- 48-bit virtual addresses, 4KiB pages, 4 bytes per page table entry

32-bit

- vm = 32, 4Kib = $2^{12} \Rightarrow$ n = 12

Single Level Page Table - Disadvantages

Math is fun, let’s do some math

Calculate the space requirements for a single level page table with

- 32-bit virtual addresses, 4KiB pages, 4 bytes per page table entry
- 48-bit virtual addresses, 4KiB pages, 4 bytes per page table entry

32-bit

- vm = 32, 4Kib = 2^{12} $\Rightarrow$ n = 12
- $2^{32-12} = 2^{20}$ entries $\Rightarrow 2^{20} \cdot 2^{2} = 2^{22}$ Byte (4 MiB)

Single Level Page Table - Disadvantages

Math is fun, let’s do some math

Calculate the space requirements for a single level page table with

- 32-bit virtual addresses, 4KiB pages, 4 bytes per page table entry
- 48-bit virtual addresses, 4KiB pages, 4 bytes per page table entry

32-bit

- vm = 32, 4Kib = $2^{12} \Rightarrow$ n = 12
- $2^{32-12} = 2^{20}$ entries $\Rightarrow 2^{20} \cdot 2^{2} = 2^{22}$ Byte (4 MiB)

48-bit

- vm = 48, 4Kib = $2^{12} \Rightarrow$ n = 12
- $2^{48-12} = 2^{36}$ entries $\Rightarrow 2^{36} \cdot 2^{2} = 2^{38}$ Byte (256 GiB)

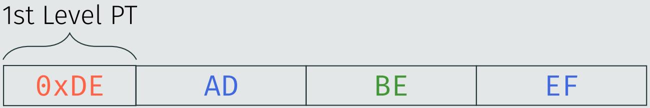
Alternatives to Single Level Page Tables

Alternatives to Single Level Page Tables

Mutli-Level page tables

Alternatives to Single Level Page Tables

Mutli-Level page tables

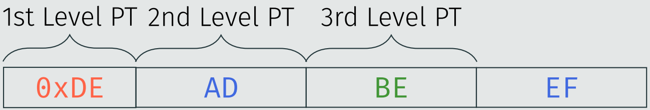


1st Level PT

0xDE AD BE EF

Alternatives to Single Level Page Tables

Mutli-Level page tables

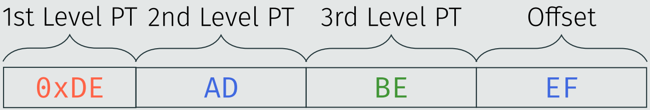


1st Level PT 2nd Level PT 3rd Level PT

0xDE AD BE EF

Alternatives to Single Level Page Tables

Mutli-Level page tables

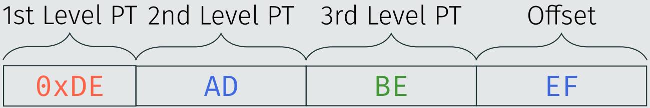


1st Level PT 2nd Level PT 3rd Level PT Offset

0xDE	AD	BE	EF

Alternatives to Single Level Page Tables

Mutli-Level page tables


1st Level PT
2nd Level PT
3rd Level PT
Offset
0xDE
AD
BE
EF

Benefits and Drawbacks?

- Pointer chasing down each level ⇒ More memory accesses

Alternatives to Single Level Page Tables

Mutli-Level page tables

1st Level PT	2nd Level PT	3rd Level PT	Offset
0xDE	AD	BE	EF

Benefits and Drawbacks?

- Pointer chasing down each level ⇒ More memory accesses
+ Address spaces are *sparse* ⇒ Only instantiate page tables you need

Inverted Page Table

What do those do?

Inverted Page Table

What do those do?

- Map *physical* addresses to virtual ones

Inverted Page Table

What do those do?

- Map *physical* addresses to virtual ones
- Why?

Inverted Page Table

What do those do?

- Map *physical* addresses to virtual ones
- Why? Physical address space is much smaller and

Inverted Page Table

What do those do?

- Map *physical* addresses to virtual ones
- Why? Physical address space is much smaller and you don’t need a table per address space

Inverted Page Table

What do those do?

- Map *physical* addresses to virtual ones
- Why? Physical address space is much smaller and you don’t need a table per address space
- If you don’t have a table per address space, how can you still have multiple?

Inverted Page Table

What do those do?

- Map *physical* addresses to virtual ones
- Why? Physical address space is much smaller and you don't need a table per address space
- If you don't have a table per address space, how can you still have multiple?

⇒ e.g. Linked List of entries per physical frame

Inverted Page Table

What do those do?

- Map *physical* addresses to virtual ones
- Why? Physical address space is much smaller and you don't need a table per address space
- If you don't have a table per address space, how can you still have multiple?

⇒ e.g. Linked List of entries per physical frame

Any drawbacks?

Inverted Page Table

What do those do?

- Map *physical* addresses to virtual ones
- Why? Physical address space is much smaller and you don't need a table per address space
- If you don't have a table per address space, how can you still have multiple?

⇒ e.g. Linked List of entries per physical frame

Any drawbacks?

- We mostly care about *the other direction*, i.e. virtual ⇒ physical

Inverted Page Table

What do those do?

- Map *physical* addresses to virtual ones
- Why? Physical address space is much smaller and you don't need a table per address space
- If you don't have a table per address space, how can you still have multiple?

⇒ e.g. Linked List of entries per physical frame

Any drawbacks?

- We mostly care about *the other direction*, i.e. virtual ⇒ physical
- That requires iteration :(

Inverted Page Tables

How can we speed them up?

After having attended *Algorithmen I* we all know:

Inverted Page Tables

How can we speed them up?

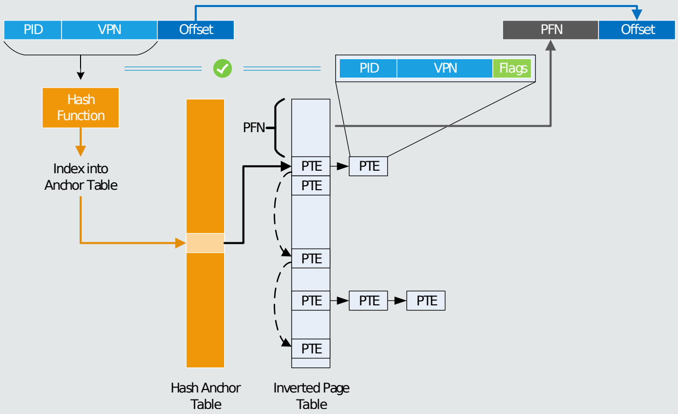
After having attended *Algorithmen I* we all know: Hacktables are O(1)

Inverted Page Tables

How can we speed them up?

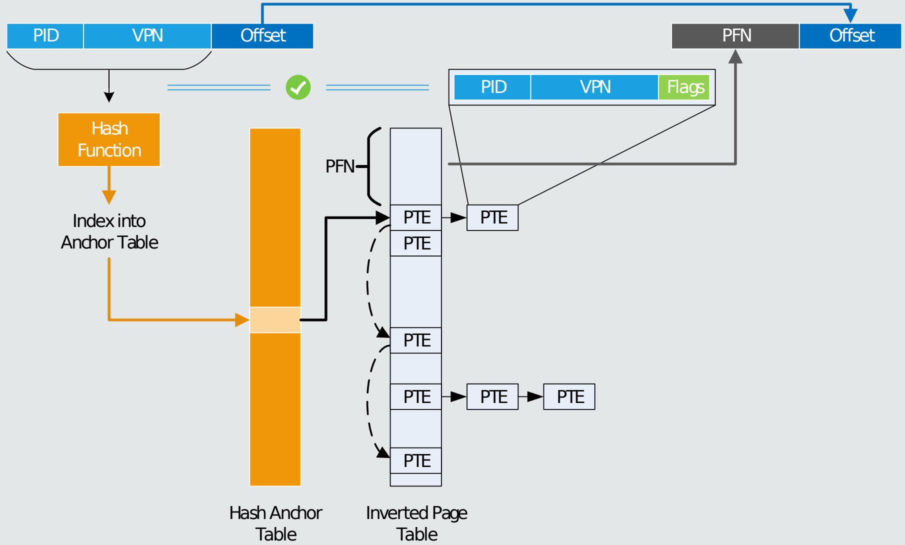
After having attended *Algorithmen I* we all know: Hacktables are O(1)

Hashed inverted page tables



Inverted Page Tables

Hashed inverted page tables



Why do pages have a valid/present bit?

What is it for?

Why do pages have a valid/present bit?

What is it for?

- The page might not be present. Why?

Why do pages have a valid/present bit?

What is it for?

- The page might not be present. Why? Swapped out,

Why do pages have a valid/present bit?

What is it for?

- The page might not be present. Why? Swapped out, not zeroed yet, …

What happens on access if it is not set?

Why do pages have a valid/present bit?

What is it for?

- The page might not be present. Why? Swapped out, not zeroed yet, …

What happens on access if it is not set?

- Page fault!
- Handle it and do sth. sensible (or crash the process…)

TLB

TLB

What does that refer to?

TLB

What does that refer to?

Translation **L**ookaside **B**uffer. What’s that for?

TLB

What does that refer to?

Translation **L**ookaside **B**uffer. What's that for?

- Lookup from virtual to physical address can be *slow*

TLB

What does that refer to?

Translation **L**ookaside **B**uffer. What's that for?

- Lookup from virtual to physical address can be *slow*
- You need to translate addresses *all the time*

TLB

What does that refer to?

Translation **L**ookaside **B**uffer. What's that for?

- Lookup from virtual to physical address can be *slow*
- You need to translate addresses *all the time*

⇒ There is no problem you can't solve with another caching layer (except having too many caching layers) Nearly the Fundamental theorem of software engineering

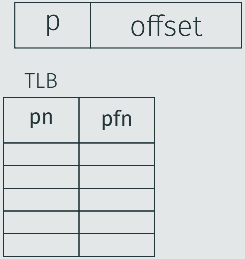
TLB

TLB layout

p	offset

TLB

TLB layout



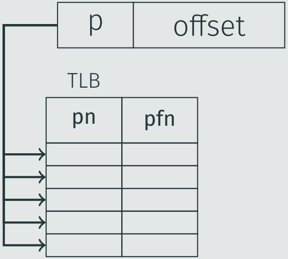
p	offset

TLB

pn	pfn

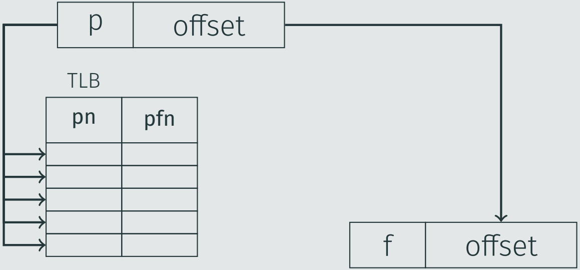
TLB

TLB layout



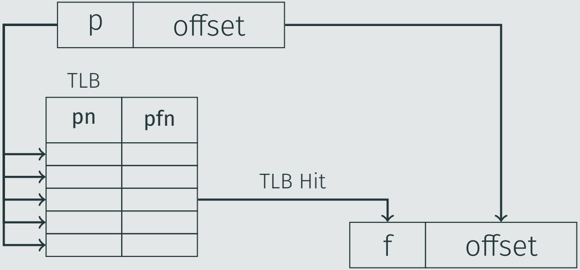
TLB

TLB layout



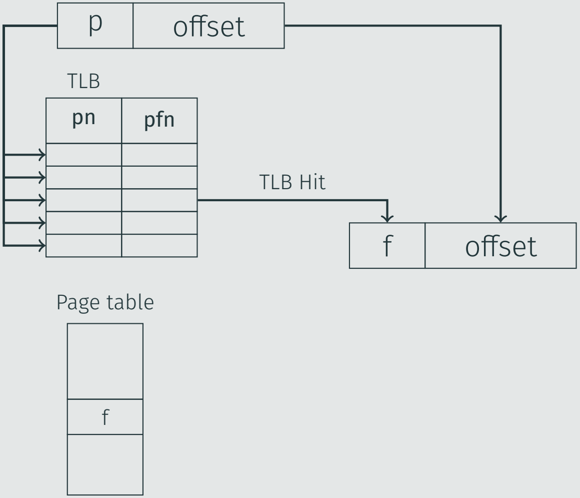
TLB

TLB layout



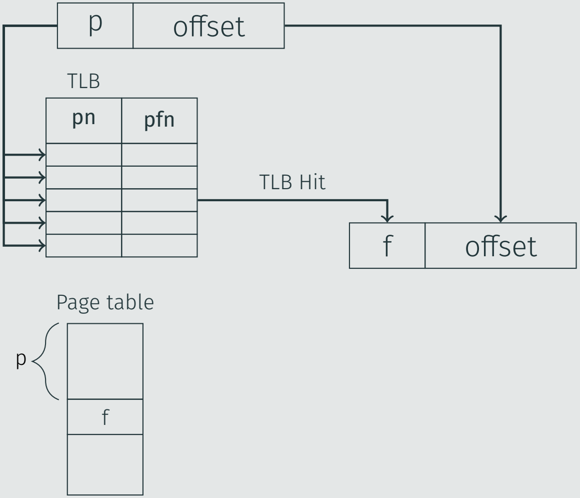
TLB

TLB layout



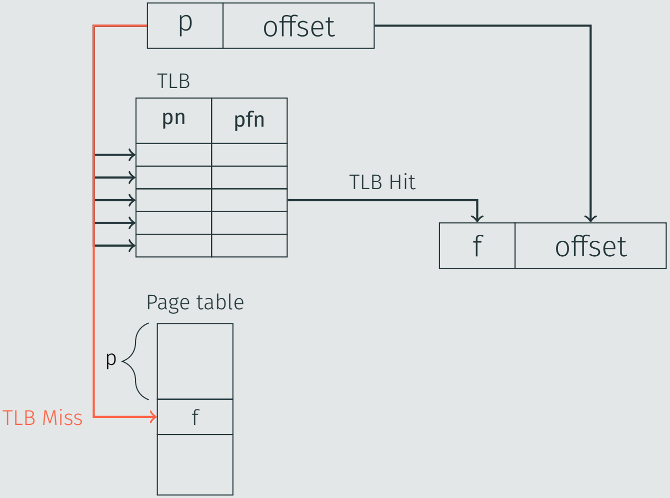
TLB

TLB layout

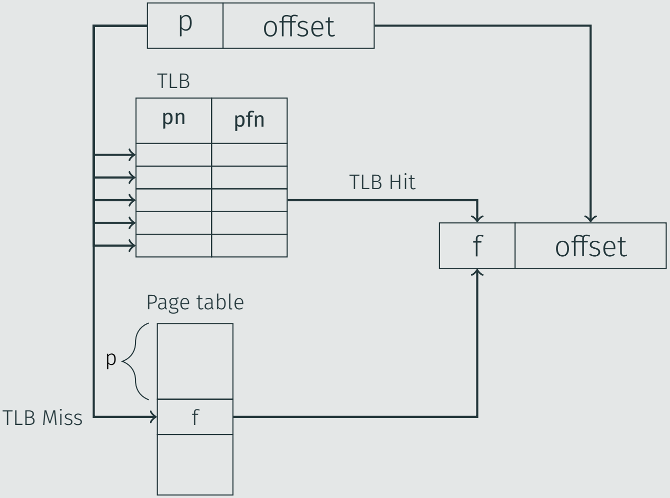


TLB

TLB layout



TLB layout



TLB - Misses

What is the difference between software and hardware walked Page Tables?

TLB - Misses

What is the difference between software and hardware walked Page Tables?

The TLB is *the same*, what differs is the behaviour on a *cache miss*.

TLB - Misses

What is the difference between software and hardware walked Page Tables?

The TLB is *the same*, what differs is the behaviour on a *cache miss*.

Hardware walked

- Hardware looks at your page table
- Hardware *resolves* the physical address using it

TLB - Misses

What is the difference between software and hardware walked Page Tables?

The TLB is *the same*, what differs is the behaviour on a *cache miss*.

Hardware walked

- Hardware looks at your page table
- Hardware *resolves* the physical address using it
- On failure a page fault is raised

⇒ What does that imply?

TLB - Misses

What is the difference between software and hardware walked Page Tables?

The TLB is *the same*, what differs is the behaviour on a *cache miss*.

Hardware walked

- Hardware looks at your page table
- Hardware *resolves* the physical address using it
- On failure a page fault is raised

⇒ What does that imply? Page table layout is fixed

TLB - Misses

What is the difference between software and hardware walked Page Tables?

The TLB is *the same*, what differs is the behaviour on a *cache miss*.

Hardware walked

- Hardware looks at your page table
- Hardware *resolves* the physical address using it
- On failure a page fault is raised

⇒ What does that imply? Page table layout is fixed

Software walked

- Transfers control to TLB miss handler
- Kernel routine walks the page table and tries to find a mapping

TLB - Misses

What is the difference between software and hardware walked Page Tables?

The TLB is *the same*, what differs is the behaviour on a *cache miss*.

Hardware walked

- Hardware looks at your page table
- Hardware *resolves* the physical address using it
- On failure a page fault is raised

⇒ What does that imply? Page table layout is fixed

Software walked

- Transfers control to TLB miss handler
- Kernel routine walks the page table and tries to find a mapping
- Loads that mapping into the TLB and can choose which entry to evict!
- If there is none ⇒ Jump to page fault handler

TLB - Software or hardware walked

What are benefits / drawbacks of software walked TLBs?

TLB - Software or hardware walked

What are benefits / drawbacks of software walked TLBs?

+ Free to choose page table layout (fit your algorithm)

TLB - Software or hardware walked

What are benefits / drawbacks of software walked TLBs?

+ Free to choose page table layout (fit your algorithm)
+ Free to choose which TLB entries to evict ⇒ Use optimized strategy

TLB - Software or hardware walked

What are benefits / drawbacks of software walked TLBs?

+ Free to choose page table layout (fit your algorithm)
+ Free to choose which TLB entries to evict ⇒ Use optimized strategy
- Greater overhead

TLB - Contents

What information does the TLB store?

TLB - Contents

What information does the TLB store?

- Physical Frame Number, Virtual Page Number

TLB - Contents

What information does the TLB store?

- Physical Frame Number, Virtual Page Number
- Valid / Present Bit

TLB - Contents

What information does the TLB store?

- Physical Frame Number, Virtual Page Number
- Valid / Present Bit
- Modified bit, permissions, ...

TLB - Page faults

When is a TLB miss or page fault raised?

Due to the TLB:

- Software walked: No entry found in TLB ⇒ TLB miss exception

TLB - Page faults

When is a TLB miss or page fault raised?

Due to the TLB:

- Software walked: No entry found in TLB ⇒ TLB miss exception
- Hardware walked: No mapping in page table ⇒ Page fault

And other problems:

TLB - Page faults

When is a TLB miss or page fault raised?

Due to the TLB:

- Software walked: No entry found in TLB ⇒ TLB miss exception
- Hardware walked: No mapping in page table ⇒ Page fault

And other problems: *Invalid access to a mapped page*

TLB - Page faults

When is a TLB miss or page fault raised?

Due to the TLB:

- Software walked: No entry found in TLB ⇒ TLB miss exception
- Hardware walked: No mapping in page table ⇒ Page fault

And other problems: *Invalid access to a mapped page*

- Software walked: Raise some kind of TLB fault if the page is e.g. read-only

TLB - Page faults

When is a TLB miss or page fault raised?

Due to the TLB:

- Software walked: No entry found in TLB ⇒ TLB miss exception
- Hardware walked: No mapping in page table ⇒ Page fault

And other problems: *Invalid access to a mapped page*

- Software walked: Raise some kind of TLB fault if the page is e.g. read-only
- Hardware walked: Page fault raised, page fault handler has to find out what happend

Page Fault Handling

Page Fault Handling

What is Demand-Paging and Pre-Paging?

Demand-Paging:

Page Fault Handling

What is Demand-Paging and Pre-Paging?

Demand-Paging:

- Load pages on-demand, right when they are needed

Page Fault Handling

What is Demand-Paging and Pre-Paging?

Demand-Paging:

- Load pages on-demand, right when they are needed

Pre-Paging:

- Loaded Pages speculatively in batches, even *before* you need them

Page Fault Handling

What is Demand-Paging and Pre-Paging?

Demand-Paging:

- Load pages on-demand, right when they are needed

Pre-Paging:

- Loaded Pages speculatively in batches, even *before* you need them

Why would you (not?) use Demand-Paging?

Page Fault Handling

What is Demand-Paging and Pre-Paging?

Demand-Paging:

- Load pages on-demand, right when they are needed

Pre-Paging:

- Loaded Pages speculatively in batches, even *before* you need them

Why would you (not?) use Demand-Paging?

+ Only loads needed data ⇒ Less memory wasted

Page Fault Handling

What is Demand-Paging and Pre-Paging?

Demand-Paging:

- Load pages on-demand, right when they are needed

Pre-Paging:

- Loaded Pages speculatively in batches, even *before* you need them

Why would you (not?) use Demand-Paging?

+ Only loads needed data ⇒ Less memory wasted
- Generates lots of page faults before working set is in memory

Page Fault Handling

Why would you (not?) use Pre-Paging?

Page Fault Handling

Why would you (not?) use Pre-Paging?

- + Might reduce number of page faults

Page Fault Handling

Why would you (not?) use Pre-Paging?

+ Might reduce number of page faults
- Loads more than needed ⇒ Wasteful

Page Fault Handling

Why would you (not?) use Pre-Paging?

- + Might reduce number of page faults
- – Loads more than needed ⇒ Wasteful
- – More I/O ⇒ Slower?

Page Fault Handling

Why would you (not?) use Pre-Paging?

- + Might reduce number of page faults
- – Loads more than needed ⇒ Wasteful
- – More I/O ⇒ Slower?
- + HDDs a lot faster when reading chunks

On-Demand Paging

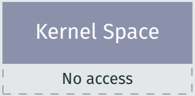
Different kind of page faults

Not all pages are created equal. Do you have any idea what types of page faults typically exist?

On-Demand Paging

Different kind of page faults

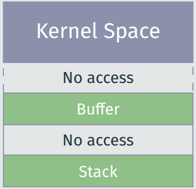
Not all pages are created equal. Do you have any idea what types of page faults typically exist?



On-Demand Paging

Different kind of page faults

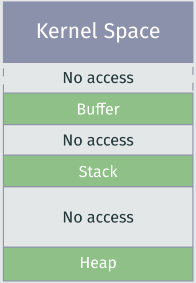
Not all pages are created equal. Do you have any idea what types of page faults typically exist?



On-Demand Paging

Different kind of page faults

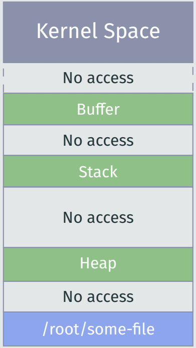
Not all pages are created equal. Do you have any idea what types of page faults typically exist?



On-Demand Paging

Different kind of page faults

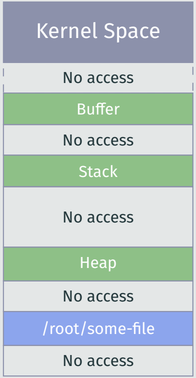
Not all pages are created equal. Do you have any idea what types of page faults typically exist?



On-Demand Paging

Different kind of page faults

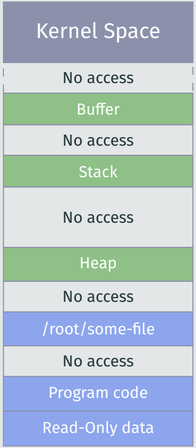
Not all pages are created equal. Do you have any idea what types of page faults typically exist?



On-Demand Paging

Different kind of page faults

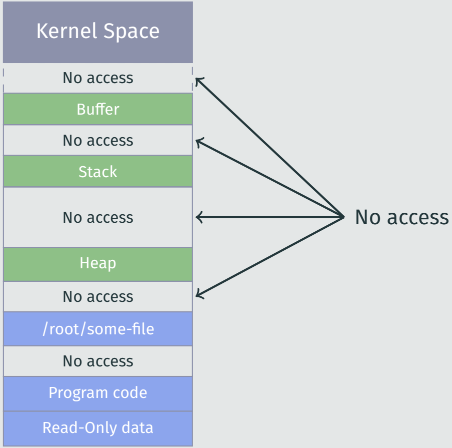
Not all pages are created equal. Do you have any idea what types of page faults typically exist?



On-Demand Paging

Different kind of page faults

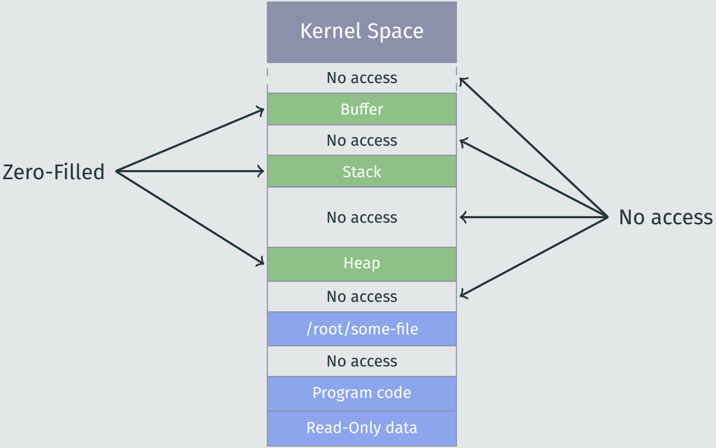
Not all pages are created equal. Do you have any idea what types of page faults typically exist?



On-Demand Paging

Different kind of page faults

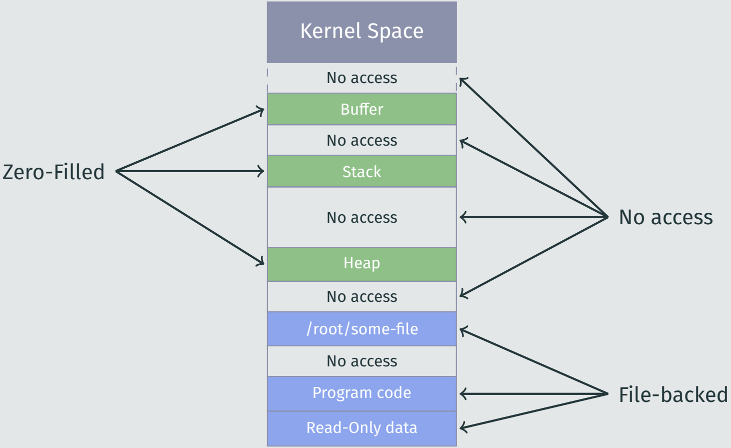
Not all pages are created equal. Do you have any idea what types of page faults typically exist?



On-Demand Paging

Different kind of page faults

Not all pages are created equal. Do you have any idea what types of page faults typically exist?



On Demand Paging

Different kind of page faults

Why are pages to generic memory zero filled?

On Demand Paging

Different kind of page faults

Why are pages to generic memory zero filled? It could leak other processes' memory otherwise (called a Covert Channel).

On Demand Paging

Different kind of page faults

Why are pages to generic memory zero filled? It could leak other processes' memory otherwise (called a Covert Channel).

And there is one other kind of page fault...

On Demand Paging

Different kind of page faults

Why are pages to generic memory zero filled? It could leak other processes' memory otherwise (called a Covert Channel).

And there is one other kind of page fault…

…The page was stolen by the OS and swapped out!

On Demand Paging

Different kind of page faults

Why are pages to generic memory zero filled? It could leak other processes' memory otherwise (called a Covert Channel).

And there is one other kind of page fault...

...The page was stolen by the OS and swapped out!

Also supported on some systems: *Purgable memory*. Stolen from Apple and also implemented in SerenityOS in this video.

Page faults

What kind of information does the page fault handler need?

Page faults

What kind of information does the page fault handler need?

- Access flags: Can the user perform the operation on this page?

Page faults

What kind of information does the page fault handler need?

- Access flags: Can the user perform the operation on this page?
- Where to find the most recent version (different for zero filled, file backed, etc.)

CoW

How could you implement Copy-on-Write memory?

CoW

How could you implement Copy-on-Write memory?

- Mark memory as read-only on fork
- Add an additional `CoW` flag: When a page fault is raised check it, copy the page and clear the `CoW` and `ro` flag

Ende

"THIS IS THE EMERGENCY OVERRIDE SYSTEM, WHICH CAN BE USED TO REGAIN CONTROL OF THE AIRCRAFT.
COMPLETE INSTRUCTIONS FOR ACTIVATING THIS SYSTEM ARE AVAILABLE AS A GNU INFO PAGE."

XKCD 912 - Manual Override

FRAGEN?

Bis nächste Woche :)