Betriebssysteme

8. Tutorium - Caches

Peter Bohner

20. Dezember 2024

ITEC - Operating Systems Group

Caches

Caches

How many do you typically have in your computer?

Caches

How many do you typically have in your computer?

Quite a few! Typically L1, L2, L3 <maybe more> and your RAM

Caches

How many do you typically have in your computer?

Quite a few! Typically L1, L2, L3 <maybe more> and your RAM

And why do you have multiple?

Caches

How many do you typically have in your computer?

Quite a few! Typically L1, L2, L3 <maybe more> and your RAM

And why do you have multiple?

- Fast caches are expensive (and size limited) $\Rightarrow$ Small

Caches

How many do you typically have in your computer?

Quite a few! Typically L1, L2, L3 <maybe more> and your RAM

And why do you have multiple?

- Fast caches are expensive (and size limited) $\Rightarrow$ Small
- Multiple levels of slower but larger caches arranged in a *cache hierarchy*

Caches

How many do you typically have in your computer?

Quite a few! Typically L1, L2, L3 <maybe more> and your RAM

And why do you have multiple?

- Fast caches are expensive (and size limited) $\Rightarrow$ Small
- Multiple levels of slower but larger caches arranged in a *cache hierarchy*

Why do caches even work?

Caches

How many do you typically have in your computer?

Quite a few! Typically L1, L2, L3 <maybe more> and your RAM

And why do you have multiple?

- Fast caches are expensive (and size limited) $\Rightarrow$ Small
- Multiple levels of slower but larger caches arranged in a *cache hierarchy*

Why do caches even work?

Two main principles:

- Temporal locality: Memory that was accessed will be accessed again soon

Caches

How many do you typically have in your computer?

Quite a few! Typically L1, L2, L3 <maybe more> and your RAM

And why do you have multiple?

- Fast caches are expensive (and size limited) $\Rightarrow$ Small
- Multiple levels of slower but larger caches arranged in a *cache hierarchy*

Why do caches even work?

Two main principles:

- Temporal locality: Memory that was accessed will be accessed again soon
- Spatial locality: If address X was accessed, $X \pm 1$ will likely be accessed soon
  $\Rightarrow$ Don't just cache single words but entire *lines*. How large?

Caches

How many do you typically have in your computer?

Quite a few! Typically L1, L2, L3 <maybe more> and your RAM

And why do you have multiple?

- Fast caches are expensive (and size limited) $\Rightarrow$ Small
- Multiple levels of slower but larger caches arranged in a *cache hierarchy*

Why do caches even work?

Two main principles:

- Temporal locality: Memory that was accessed will be accessed again soon
- Spatial locality: If address X was accessed, $X \pm 1$ will likely be accessed soon $\Rightarrow$ Don't just cache single words but entire *lines*. How large? Probably ≥ 64 Bytes for *reasons*

Caches

Cache lines

Why is a cache line useful? Can’t we just make N memory accesses?

Caches

Cache lines

Why is a cache line useful? Can't we just make N memory accesses? Reading a chunk from DRAM is *much* more efficient than reading them one after another

Cache Types

What types do you know?

- Fully Associative

Cache Types

What types do you know?

- Fully Associative
- Set-Associative

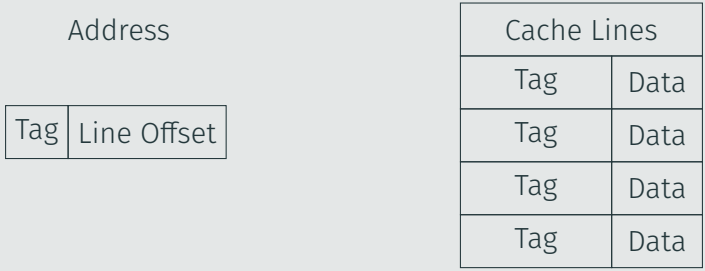
Cache Types

What types do you know?

- Fully Associative
- Set-Associative
- Direct Mapped

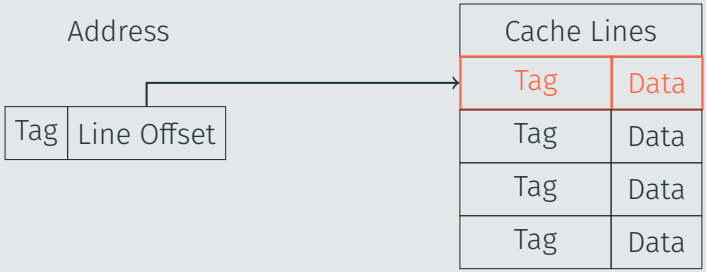
Cache Types - Fully Associative

What is that?



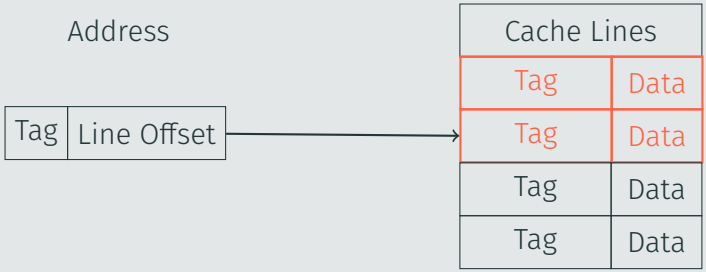
Cache Types - Fully Associative

What is that?



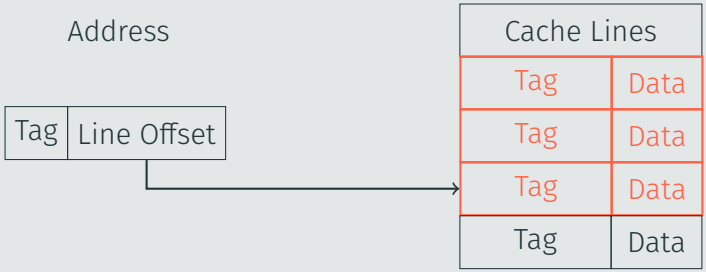
Cache Types - Fully Associative

What is that?



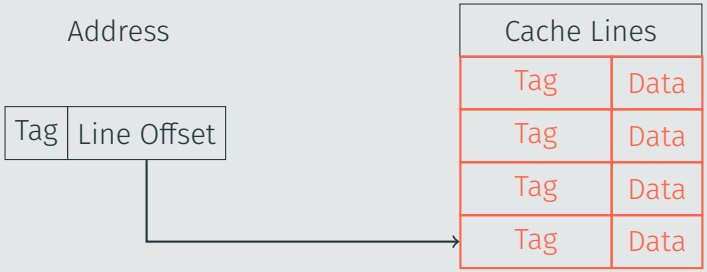
Cache Types - Fully Associative

What is that?



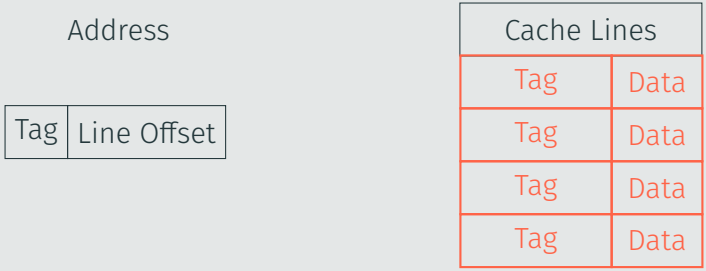
Cache Types - Fully Associative

What is that?



Cache Types - Fully Associative

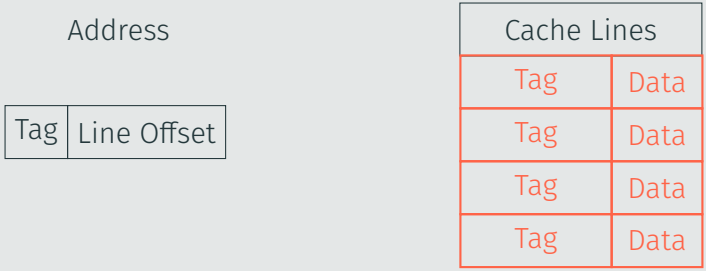
What is that?



When can cache misses happen here?

Cache Types - Fully Associative

What is that?

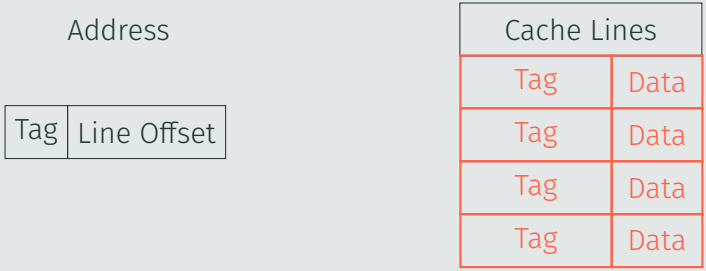


When can cache misses happen here?

- Never accessed that address before. Called a

Cache Types - Fully Associative

What is that?

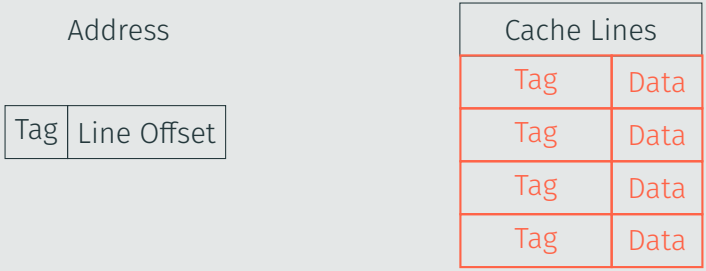


When can cache misses happen here?

- Never accessed that address before. Called a Compulsory Miss

Cache Types - Fully Associative

What is that?

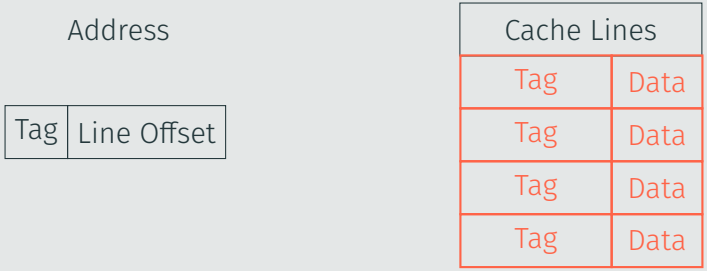


When can cache misses happen here?

- Never accessed that address before. Called a Compulsory Miss
- Cache was full and it was evicted. Called a

Cache Types - Fully Associative

What is that?

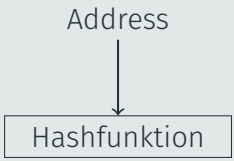


When can cache misses happen here?

- Never accessed that address before. Called a Compulsory Miss
- Cache was full and it was evicted. Called a Capacity Miss

Cache Types - Direct Mapped

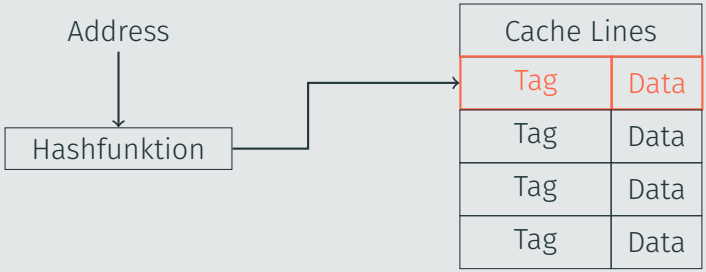
What is that?



Cache Lines	
Tag	Data
Tag	Data
Tag	Data
Tag	Data

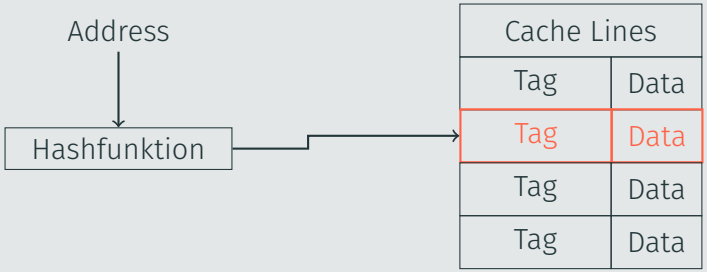
Cache Types - Direct Mapped

What is that?



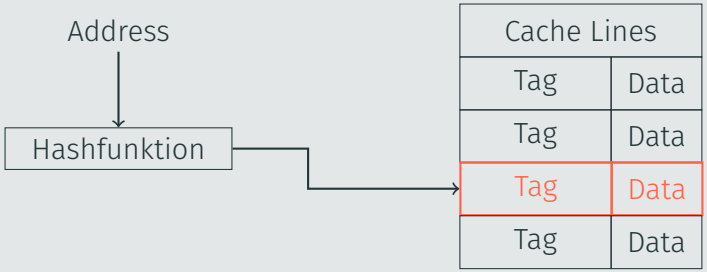
Cache Types - Direct Mapped

What is that?



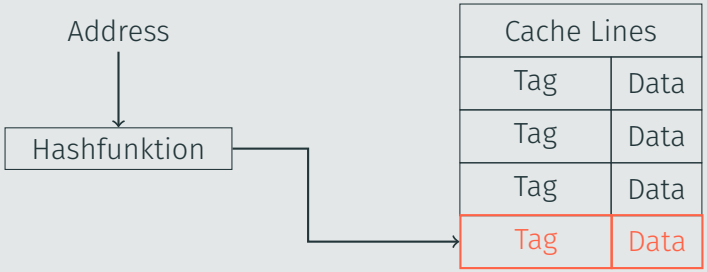
Cache Types - Direct Mapped

What is that?



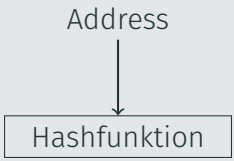
Cache Types - Direct Mapped

What is that?



Cache Types - Direct Mapped

What is that?

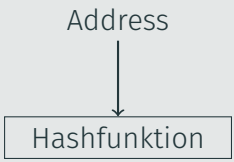


Cache Lines	
Tag	Data
Tag	Data
Tag	Data
Tag	Data

When can cache misses happen here?

Cache Types - Direct Mapped

What is that?



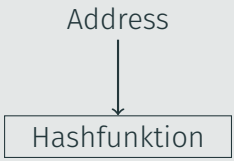
Cache Lines	
Tag	Data
Tag	Data
Tag	Data
Tag	Data

When can cache misses happen here?

- Never accessed that address before. Called a

Cache Types - Direct Mapped

What is that?



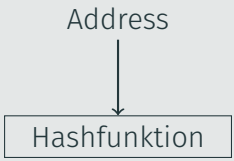
Cache Lines	
Tag	Data
Tag	Data
Tag	Data
Tag	Data

When can cache misses happen here?

- Never accessed that address before. Called a Compulsory Miss

Cache Types - Direct Mapped

What is that?



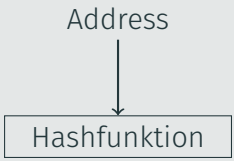
Cache Lines	
Tag	Data
Tag	Data
Tag	Data
Tag	Data

When can cache misses happen here?

- Never accessed that address before. Called a Compulsory Miss
- Cache was full and it was evicted. Called a

Cache Types - Direct Mapped

What is that?



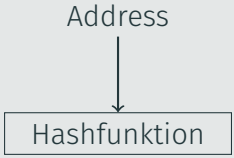
Cache Lines	
Tag	Data
Tag	Data
Tag	Data
Tag	Data

When can cache misses happen here?

- Never accessed that address before. Called a Compulsory Miss
- Cache was full and it was evicted. Called a Capacity Miss

Cache Types - Direct Mapped

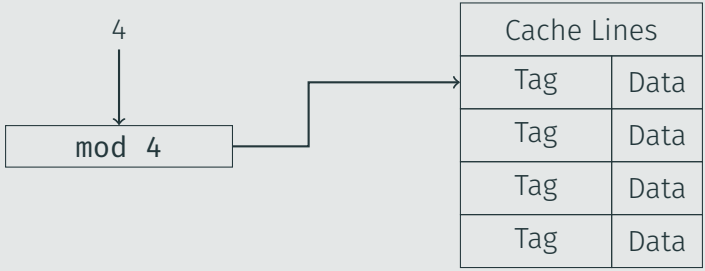
What is that?



Cache Lines	
Tag	Data
Tag	Data
Tag	Data
Tag	Data

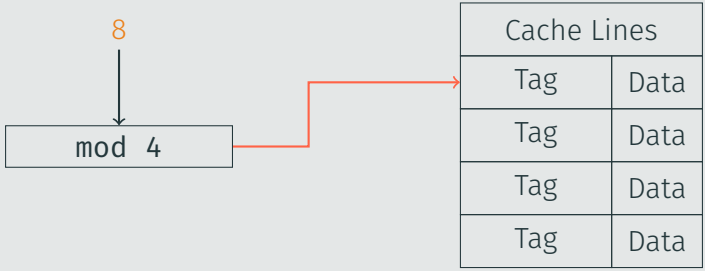
Cache Types - Direct Mapped

What is that?



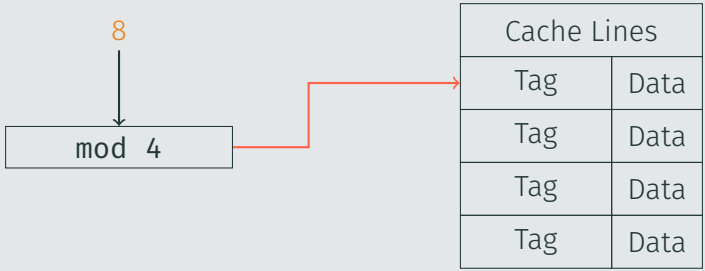
Cache Types - Direct Mapped

What is that?



Cache Types - Direct Mapped

What is that?

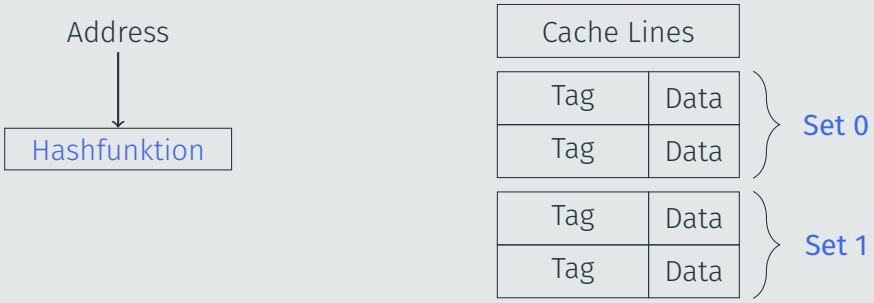


Conflict misses

There was space, but it was mapped to the same slot!

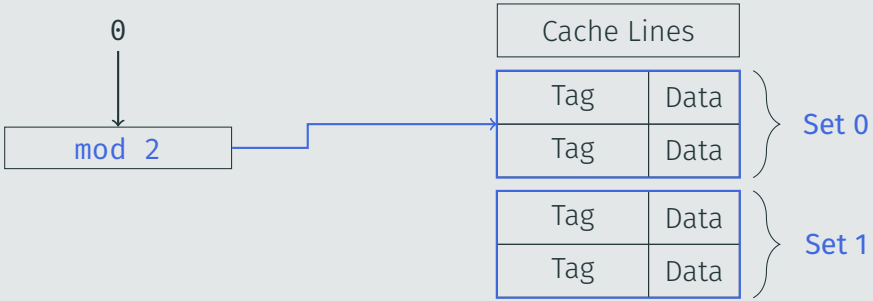
Cache Types - Set associative

What is that?



Cache Types - Set associative

What is that?

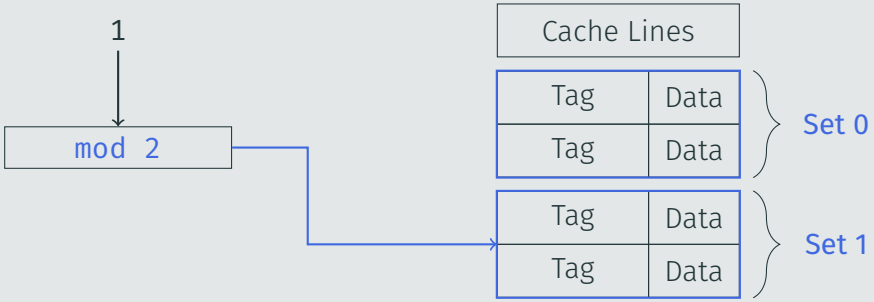


Insertion

Find the correct *set* using the index, then treat each set as a *Fully Associative* cache.

Cache Types - Set associative

What is that?

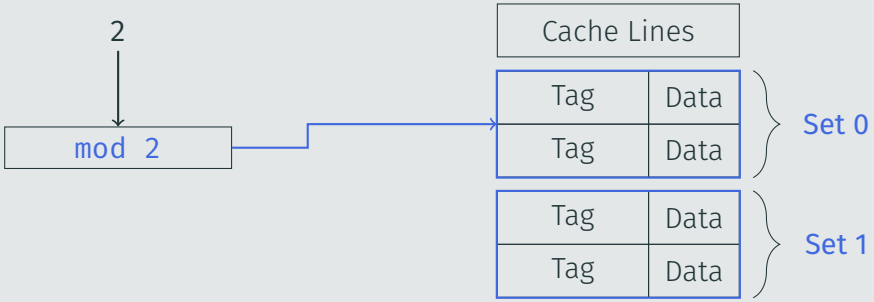


Insertion

Find the correct *set* using the index, then treat each set as a *Fully Associative* cache.

Cache Types - Set associative

What is that?



Insertion

Find the correct *set* using the index, then treat each set as a *Fully Associative* cache.

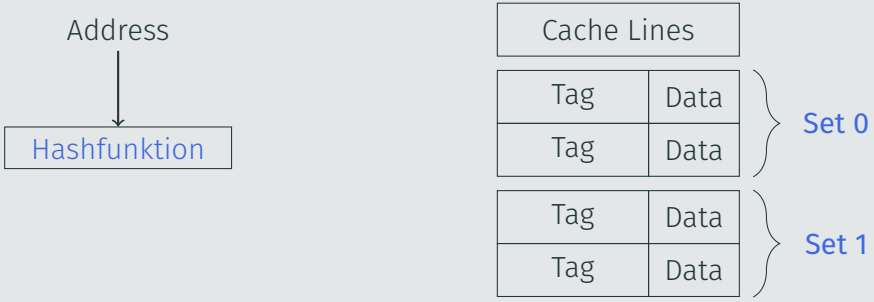
Cache Types - Set associative

What is that?

Conflict misses?

Cache Types - Set associative

What is that?



Conflict misses!

There was space, but it was mapped to the same *set*!

Modifying Caches

You have a Cache and real memory behind

What do you do when you write to ...

... an address in the cache?

Modifying Caches

You have a Cache and real memory behind

What do you do when you write to ...

... an address in the cache?

- Write Through:

Modifying Caches

You have a Cache and real memory behind

What do you do when you write to ...

... an address in the cache?

- Write Through: Update the cache *and the main memory*

Modifying Caches

You have a Cache and real memory behind

What do you do when you write to ...

... an address in the cache?

- Write Through: Update the cache *and the main memory*
- Write Back:

Modifying Caches

You have a Cache and real memory behind

What do you do when you write to ...

... an address in the cache?

- Write Through: Update the cache *and the main memory*
- Write Back: Update only the cache. Update main memory *on eviction*

... an address *not* in the cache?

Modifying Caches

You have a Cache and real memory behind

What do you do when you write to ...

... an address in the cache?

- Write Through: Update the cache *and the main memory*
- Write Back: Update only the cache. Update main memory *on eviction*

... an address *not* in the cache?

- Write-allocate:

Modifying Caches

You have a Cache and real memory behind

What do you do when you write to ...

... an address in the cache?

- Write Through: Update the cache *and the main memory*
- Write Back: Update only the cache. Update main memory *on eviction*

... an address *not* in the cache?

- Write-allocate: First load it into the cache. Then see above

Modifying Caches

You have a Cache and real memory behind

What do you do when you write to ...

... an address in the cache?

- Write Through: Update the cache *and the main memory*
- Write Back: Update only the cache. Update main memory *on eviction*

... an address *not* in the cache?

- Write-allocate: First load it into the cache. Then see above
- Write-to-memory:

Modifying Caches

You have a Cache and real memory behind

What do you do when you write to ...

... an address in the cache?

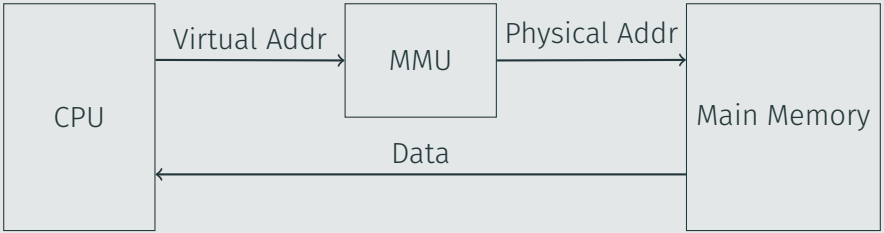
- Write Through: Update the cache *and the main memory*
- Write Back: Update only the cache. Update main memory *on eviction*

... an address *not* in the cache?

- Write-allocate: First load it into the cache. Then see above
- Write-to-memory: Don't load into cache, modify in memory

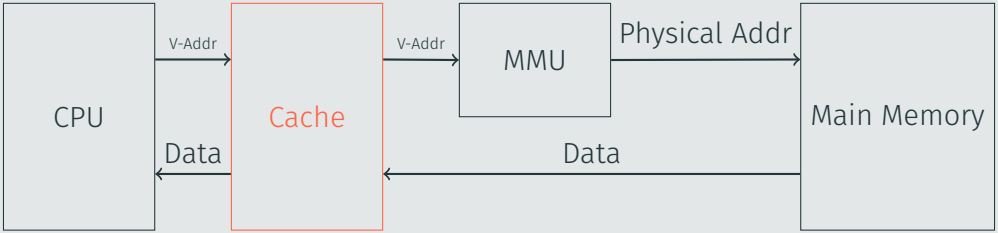
Placing The Cache

At which position do you put the cache?



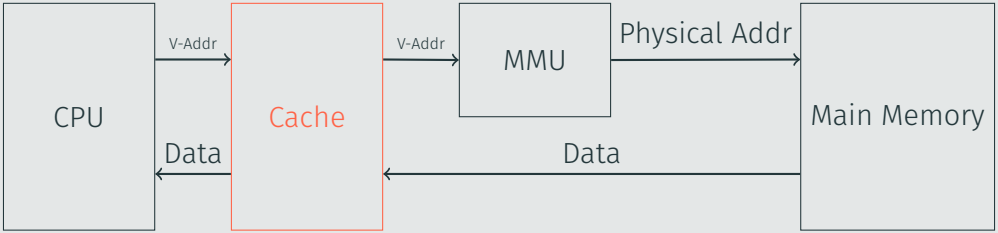
Placing The Cache

At which position do you put the cache?



Placing The Cache

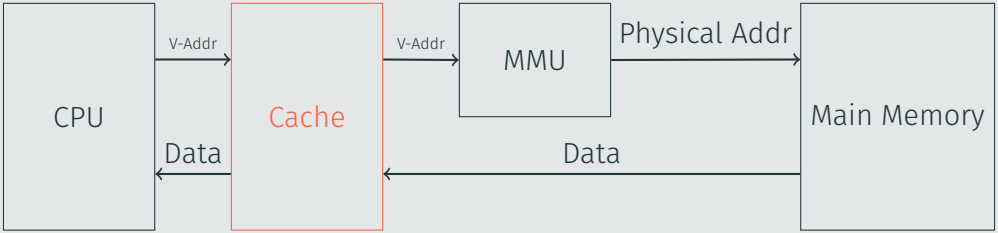
At which position do you put the cache?



What kind of addresses does this use for Index/Tag?

Placing The Cache

At which position do you put the cache?



What kind of addresses does this use for Index/Tag?

Virtually Indexed, Virtually Tagged (VIVT)

Placing The Cache

What kind of addresses does this use for Index/Tag?

Virtually Indexed, Virtually Tagged (VIVT)

Benefits? Drawbacks?

Placing The Cache

What kind of addresses does this use for Index/Tag?

Virtually Indexed, Virtually Tagged (VIVT)

Benefits? Drawbacks?

+ Fast, no address translation

Placing The Cache

What kind of addresses does this use for Index/Tag?

Virtually Indexed, Virtually Tagged (VIVT)

Benefits? Drawbacks?

+ Fast, no address translation
- Ambiguity Problem (The same virtual address might point to different physical addresses over time)

Placing The Cache

What kind of addresses does this use for Index/Tag?

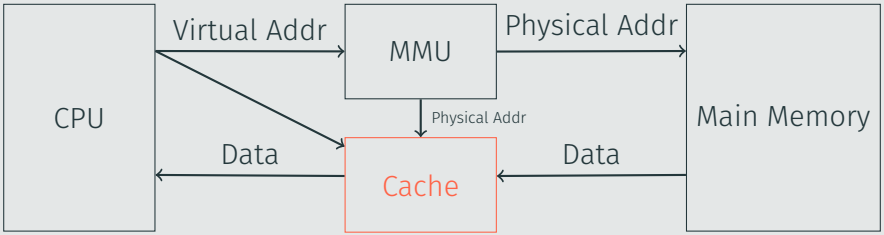
Virtually Indexed, Virtually Tagged (VIVT)

Benefits? Drawbacks?

+ Fast, no address translation
- Ambiguity Problem (The same virtual address might point to different physical addresses over time)
- Alias Problem (Multiple virtual addresses might point to the same physical address)

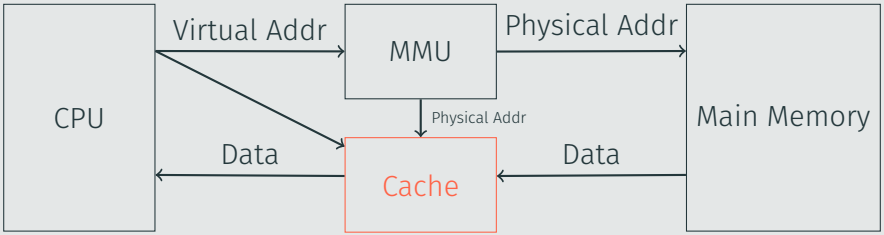
Placing The Cache

At which position do you put the cache?



Placing The Cache

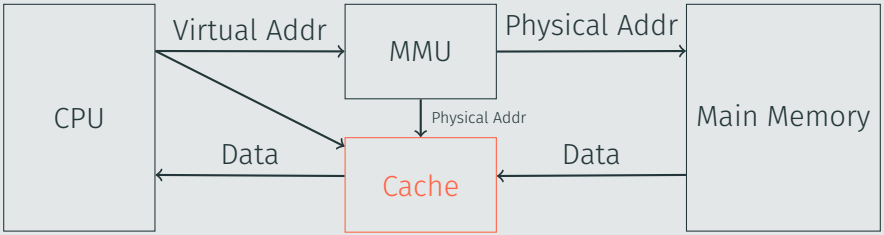
At which position do you put the cache?



What kind of addresses does this use for Index/Tag?

Placing The Cache

At which position do you put the cache?



What kind of addresses does this use for Index/Tag?

Virtually Indexed, Physically Tagged (VIPT)

Placing The Cache

What kind of addresses does this use for Index/Tag?

Virtually Indexed, Physically Tagged (VIPT)

Benefits? Drawbacks?

Placing The Cache

What kind of addresses does this use for Index/Tag?

Virtually Indexed, Physically Tagged (VIPT)

Benefits? Drawbacks?

+ Still relatively fast: Can lookup the index while the MMU works
+ Ambiguity solved, as it is tagged by physical address (iff

Placing The Cache

What kind of addresses does this use for Index/Tag?

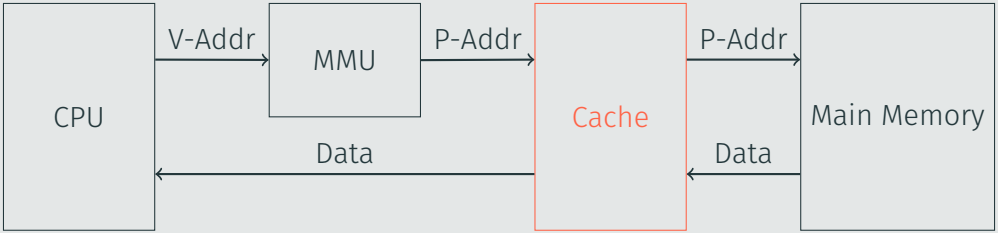
Virtually Indexed, Physically Tagged (VIPT)

Benefits? Drawbacks?

+ Still relatively fast: Can lookup the index while the MMU works
+ Ambiguity solved, as it is tagged by physical address (iff the entire PFN is used as a tag!)
- Alias Problem (sometimes!)

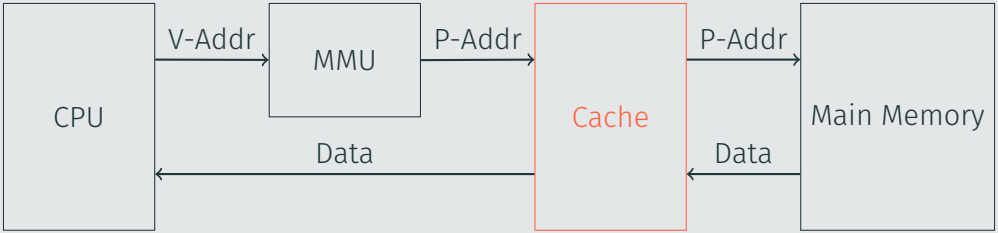
Placing The Cache

At which position do you put the cache?



Placing The Cache

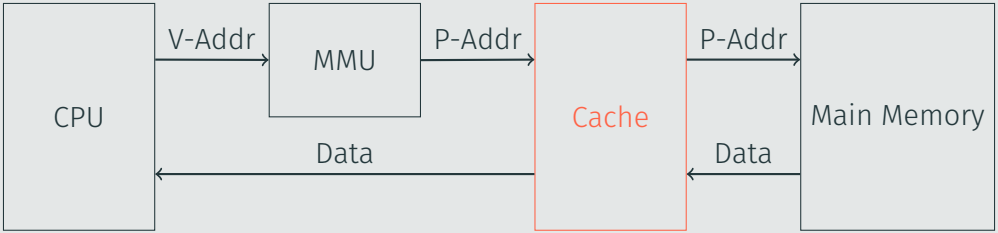
At which position do you put the cache?



What kind of addresses does this use for Index/Tag?

Placing The Cache

At which position do you put the cache?



What kind of addresses does this use for Index/Tag?

Physically Indexed, Physically Tagged (PIPT)

Placing The Cache

What kind of addresses does this use for Index/Tag?

Physically Indexed, Physically Tagged (PIPT)

Benefits? Drawbacks?

Placing The Cache

What kind of addresses does this use for Index/Tag?

Physically Indexed, Physically Tagged (PIPT)

Benefits? Drawbacks?

+ Transparent to CPU: no alias, no ambiguity

Placing The Cache

What kind of addresses does this use for Index/Tag?

Physically Indexed, Physically Tagged (PIPT)

Benefits? Drawbacks?

+ Transparent to CPU: no alias, no ambiguity
+ Coherency can be implemented in hardware

Placing The Cache

What kind of addresses does this use for Index/Tag?

Physically Indexed, Physically Tagged (PIPT)

Benefits? Drawbacks?

+ Transparent to CPU: no alias, no ambiguity
+ Coherency can be implemented in hardware
- Allocation conflicts

Placing The Cache

What kind of addresses does this use for Index/Tag?

Physically Indexed, Physically Tagged (PIPT)

Benefits? Drawbacks?

+ Transparent to CPU: no alias, no ambiguity
+ Coherency can be implemented in hardware
- Allocation conflicts

Allocation conflicts

- How well do you think *contiguous* virtual pages fit into the cache?

Placing The Cache

What kind of addresses does this use for Index/Tag?

Physically Indexed, Physically Tagged (PIPT)

Benefits? Drawbacks?

+ Transparent to CPU: no alias, no ambiguity
+ Coherency can be implemented in hardware
- Allocation conflicts

Allocation conflicts

- How well do you think *contiguous* virtual pages fit into the cache?
- Maybe not so well. The cache operates on *physical* addresses

$\Rightarrow$ Sequential virtual pages might be mapped to *conflicting* physical ones!

Pipes

`ls | less`

Write a C-program for Linux that creates two child processes, *ls* and *less* and uses an ordinary pipe to redirect the standard output of ls to the standard input of less.

Ende

XKCD 361 - Christmas Back Home

F R A G E N?

https://forms.gle/9CwJSKidKibubran9

Bis nächste Woche