Operating Systems
Lecture 15: Swapping

Nipun Batra
Sep 7, 2018

Reducing Memory Overheads of Paging - PT + PT

	PFN	Valid	…
PFN = 200	10	1	..
	-	0	
	-	0	
	-	0	
PFN = 201	23	1	
	-	0	
	-	0	
PFN = 202	-	0	
	-	0	
	-	0	
	-	0	
	-	0	
PFN = 203	-	0	
	-	0	
	28	1	
	4	1	

Total entries = 16

Reducing Memory Overheads of Paging - PT + PT

	PFN	Valid	…
PFN = 200	10	1	..
	-	0	
	-	0	
	-	0	
PFN = 201	23	1	
	-	0	
	-	0	
PFN = 202	-	0	
	-	0	
	-	0	
	-	0	
	-	0	
PFN = 203	-	0	
	-	0	
	28	1	
	4	1	

Total entries = 16

Reducing Memory Overheads of Paging - PT + PT

	PFN	Valid	…
PFN = 200	10	1	..
	-	0	
	-	0	
	-	0	
PFN = 201	23	1	
	-	0	
	-	0	
PFN = 202	-	0	
	-	0	
	-	0	
	-	0	
	-	0	
PFN = 203	-	0	
	-	0	
	28	1	
	4	1	

Total entries = 16

PFN	Valid	…
10	1	..
-	0	
-	0	
23	1	…

Reducing Memory Overheads of Paging - PT + PT

PFN = 200

PFN	Valid	…
10	1	..
-	0	
-	0	
-	0	

PFN = 201

23	1	
-	0	
-	0	
-	0	

PFN = 202

-	0	
-	0	
-	0	
-	0	

PFN = 203

-	0	
-	0	
28	1	
4	1	

Total entries = 16

PFN	Valid	…
10	1	..
-	0	
-	0	
23	1	…

PFN	Valid	…
-	0	..
-	0	
-	0	
-	0	…

Reducing Memory Overheads of Paging - PT + PT

PFN = 200

PFN	Valid	…
10	1	..
-	0	
-	0	
-	0	

PFN = 201

23	1	
-	0	
-	0	

PFN = 202

-	0	
-	0	
-	0	
-	0	

PFN = 203

-	0	
-	0	
28	1	
4	1	

Total entries = 16

PFN	Valid	…
10	1	..
-	0	
-	0	
23	1	…

PFN	Valid	…
-	0	..
-	0	
-	0	
-	0	…

PFN	Valid	…
-	0	..
-	0	
-	0	
-	0	…

Reducing Memory Overheads of Paging - PT + PT

PFN = 200

PFN	Valid	…
10	1	..
-	0	
-	0	
-	0	

PFN = 201

23	1	
-	0	
-	0	
-	0	

PFN = 202

-	0	
-	0	
-	0	
-	0	

PFN = 203

-	0	
-	0	
28	1	
4	1	

Total entries = 16

PFN	Valid	…
10	1	..
-	0	
-	0	
23	1	…

PFN	Valid	…
-	0	..
-	0	
-	0	
-	0	…

PFN	Valid	…
-	0	..
-	0	
-	0	
-	0	…

PFN	Valid	…
-	0	..
-	0	
28	1	
4	1	…

Reducing Memory Overheads of Paging - PT + PT

PFN = 200

PFN	Valid	…
10	1	..
-	0	
-	0	
-	0	

PFN = 201

23	1	
-	0	
-	0	

PFN = 202

-	0	
-	0	
-	0	
-	0	

PFN = 203

-	0	
-	0	
28	1	
4	1	

Total entries = 16

PFN	Valid	…
201	1	..
202	0	
203	0	
204	1	…

PFN	Valid	…
10	1	..
-	0	
-	0	
23	1	…

PFN	Valid	…
-	0	..
-	0	
-	0	
-	0	…

PFN	Valid	…
-	0	..
-	0	
-	0	
-	0	…

PFN	Valid	…
-	0	..
-	0	
28	1	
4	1	…

Reducing Memory Overheads of Paging - PT + PT

PFN = 200

PFN	Valid	…
10	1	..
-	0	
-	0	
-	0	

PFN = 201

PFN	Valid	…
23	1	
-	0	
-	0	
-	0	

PFN = 202

PFN	Valid	…
-	0	
-	0	
-	0	
-	0	

PFN = 203

PFN	Valid	…
-	0	
-	0	
28	1	
4	1	

Total entries = 16

PFN = 200

PFN	Valid	…
201	1	..
202	0	
203	0	
204	1	…

PFN	Valid	…
10	1	..
-	0	
-	0	
23	1	…

PFN	Valid	…
-	0	..
-	0	
-	0	
-	0	…

PFN	Valid	…
-	0	..
-	0	
-	0	
-	0	…

PFN	Valid	…
-	0	..
-	0	
28	1	
4	1	…

Reducing Memory Overheads of Paging - PT + PT

PFN = 200

PFN	Valid	…
10	1	..
-	0	
-	0	
-	0	

PFN = 201

PFN	Valid	…
23	1	
-	0	
-	0	
-	0	

PFN = 202

PFN	Valid	…
-	0	
-	0	
-	0	
-	0	

PFN = 203

PFN	Valid	…
-	0	
-	0	
28	1	
4	1	

Total entries = 16

PFN = 200

PFN	Valid	…
201	1	..
202	0	
203	0	
204	1	…

PFN = 201

PFN	Valid	…
10	1	..
-	0	
-	0	
23	1	…

PFN	Valid	…
-	0	..
-	0	
-	0	
-	0	…

PFN	Valid	…
-	0	..
-	0	
-	0	
-	0	…

PFN	Valid	…
-	0	..
-	0	
28	1	
4	1	…

Reducing Memory Overheads of Paging - PT + PT

PFN = 200

PFN	Valid	…
10	1	..
-	0	
-	0	
-	0	

PFN = 201

PFN	Valid	…
23	1	
-	0	
-	0	
-	0	

PFN = 202

PFN	Valid	…
-	0	
-	0	
-	0	
-	0	

PFN = 203

PFN	Valid	…
-	0	
-	0	
28	1	
4	1	

Total entries = 16

PFN = 200

PFN	Valid	…
201	1	..
202	0	
203	0	
204	1	…

PFN = 201

PFN	Valid	…
10	1	..
-	0	
-	0	
23	1	…

PFN = 202

PFN	Valid	…
-	0	..
-	0	
-	0	
-	0	…

PFN	Valid	…
-	0	..
-	0	
-	0	
-	0	…

PFN	Valid	…
-	0	..
-	0	
28	1	
4	1	…

Reducing Memory Overheads of Paging - PT + PT

PFN = 200

PFN	Valid	…
10	1	..
-	0	
-	0	
-	0	

PFN = 201

PFN	Valid	…
23	1	
-	0	
-	0	
-	0	

PFN = 202

PFN	Valid	…
-	0	
-	0	
-	0	
-	0	

PFN = 203

PFN	Valid	…
-	0	
-	0	
28	1	
4	1	

Total entries = 16

PFN = 200

PFN	Valid	…
201	1	..
202	0	
203	0	
204	1	…

PFN = 201

PFN	Valid	…
10	1	..
-	0	
-	0	
23	1	…

PFN = 202

PFN	Valid	…
-	0	..
-	0	
-	0	
-	0	…

PFN = 203

PFN	Valid	…
-	0	..
-	0	
-	0	
-	0	…

PFN	Valid	…
-	0	..
-	0	
28	1	
4	1	…

Reducing Memory Overheads of Paging - PT + PT

PFN = 200

PFN	Valid	…
10	1	..
-	0	
-	0	
-	0	

PFN = 201

PFN	Valid	…
23	1	
-	0	
-	0	
-	0	

PFN = 202

PFN	Valid	…
-	0	
-	0	
-	0	
-	0	

PFN = 203

PFN	Valid	…
-	0	
-	0	
28	1	
4	1	

Total entries = 16

PFN = 200

PFN	Valid	…
201	1	..
202	0	
203	0	
204	1	…

PFN = 201

PFN	Valid	…
10	1	..
-	0	
-	0	
23	1	…

PFN = 202

PFN	Valid	…
-	0	..
-	0	
-	0	
-	0	…

PFN = 203

PFN	Valid	…
-	0	..
-	0	
-	0	
-	0	…

PFN = 204

PFN	Valid	…
-	0	..
-	0	
28	1	
4	1	…

Reducing Memory Overheads of Paging - PT + PT

PFN = 200

PFN	Valid	…
10	1	..
-	0	
-	0	
-	0	
23	1	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
28	1	
4	1	

Total entries = 16

PFN = 200

PFN	Valid	…
201	1	..
202	0	
203	0	
204	1	…

PFN = 201

PFN	Valid	…
10	1	..
-	0	
-	0	
23	1	…

PFN = 202

PFN	Valid	…
-	0	..
-	0	
-	0	
-	0	…

PFN = 203

PFN	Valid	…
-	0	..
-	0	
-	0	
-	0	…

PFN = 204

PFN	Valid	…
-	0	..
-	0	
28	1	
4	1	…

Reducing Memory Overheads of Paging - PT + PT

PFN = 200

PFN	Valid	…
10	1	..
-	0	
-	0	
-	0	
23	1	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
28	1	
4	1	

PFN = 201

PFN = 202

PFN = 203

Total entries = 16

PFN = 200

PFN	Valid	…
201	1	..
202	0	
203	0	
204	1	…

PFN = 201

PFN	Valid	…
10	1	..
-	0	
-	0	
23	1	…

PFN = 202

PFN	Valid	…
-	0	..
-	0	
-	0	
-	0	…

PFN = 203

PFN	Valid	…
-	0	..
-	0	
-	0	
-	0	…

PFN = 204

PFN	Valid	…
-	0	..
-	0	
28	1	
4	1	…

Reducing Memory Overheads of Paging - PT + PT

PFN = 200

PFN	Valid	…
10	1	..
-	0	
-	0	
-	0	
23	1	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
28	1	
4	1	

PFN = 201

PFN = 202

PFN = 203

Total entries = 16

PFN = 200

PFN	Valid	…
201	1	..
202	0	
203	0	
204	1	…

PFN = 201

PFN	Valid	…
10	1	..
-	0	
-	0	
23	1	…

PFN = 202

PFN	Valid	…
-	0	..
-	0	
-	0	
-	0	…

PFN = 203

PFN	Valid	…
-	0	..
-	0	
-	0	
-	0	…

PFN = 204

PFN	Valid	…
-	0	..
-	0	
28	1	
4	1	…

Reducing Memory Overheads of Paging - PT + PT

PFN = 200

PFN	Valid	…
10	1	..
-	0	
-	0	
-	0	
23	1	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
28	1	
4	1	

PFN = 201

PFN = 202

PFN = 203

Total entries = 16

PFN = 200

PFN	Valid	…
201	1	..
202	0	
203	0	
204	1	…

PFN = 201

PFN	Valid	…
10	1	..
-	0	
-	0	
23	1	…

PFN = 202

PFN	Valid	…
-	0	..
-	0	
-	0	
-	0	…

PFN = 203

PFN	Valid	…
-	0	..
-	0	
-	0	
-	0	…

PFN = 204

PFN	Valid	…
-	0	..
-	0	
28	1	
4	1	…

Reducing Memory Overheads of Paging - PT + PT

PFN = 200

PFN	Valid	…
10	1	..
-	0	
-	0	
-	0	
23	1	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
28	1	
4	1	

PFN = 201

PFN = 202

PFN = 203

Total entries = 16

PFN = 200

PFN	Valid	…
201	1	..
202	0	
203	0	
204	1	…

Total entries = 16 + 4

PFN = 201

PFN	Valid	…
10	1	..
-	0	
-	0	
23	1	…

PFN = 202

PFN	Valid	…
-	0	..
-	0	
-	0	
-	0	…

PFN = 203

PFN	Valid	…
-	0	..
-	0	
-	0	
-	0	…

PFN = 204

PFN	Valid	…
-	0	..
-	0	
28	1	
4	1	…

Reducing Memory Overheads of Paging - PT + PT

PFN = 200

PFN	Valid	…
10	1	..
-	0	
-	0	
-	0	
23	1	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
28	1	
4	1	

PFN = 201

PFN = 202

PFN = 203

Total entries = 16

PFN = 200

PFN	Valid	…
201	1	..
202	0	
203	0	
204	1	…

Total entries = 16 + 4

PFN = 201

PFN	Valid	…
10	1	..
-	0	
-	0	
23	1	…

PFN = 202

PFN	Valid	…
-	0	..
-	0	
-	0	
-	0	…

PFN = 203

PFN	Valid	…
-	0	..
-	0	
-	0	
-	0	…

PFN = 204

PFN	Valid	…
-	0	..
-	0	
28	1	
4	1	…

Reducing Memory Overheads of Paging - PT + PT

PFN = 200

PFN	Valid	…
10	1	..
-	0	
-	0	
-	0	
23	1	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
28	1	
4	1	

Total entries = 16

PFN = 200

PFN	Valid	…
201	1	..
202	0	
203	0	
204	1	…

Total entries = 16 + 4

PFN = 201

PFN	Valid	…
10	1	..
-	0	
-	0	
23	1	…

PFN = 202

PFN	Valid	…
-	0	..
-	0	
-	0	
-	0	…

PFN = 203

PFN	Valid	…
-	0	..
-	0	
-	0	
-	0	…

PFN = 204

PFN	Valid	…
-	0	..
-	0	
28	1	
4	1	…

Reducing Memory Overheads of Paging - PT + PT

PFN = 200

PFN	Valid	…
10	1	..
-	0	
-	0	
-	0	

PFN = 201

PFN	Valid	…
23	1	
-	0	
-	0	
-	0	

PFN = 202

PFN	Valid	…
-	0	
-	0	
-	0	
-	0	

PFN = 203

PFN	Valid	…
-	0	
-	0	
28	1	
4	1	

Total entries = 16

PFN = 200

PFN	Valid	…
201	1	..
202	0	
203	0	
204	1	…

Total entries = 16 + 4

PFN = 201

PFN	Valid	…
10	1	..
-	0	
-	0	
23	1	…

PFN = 202

PFN = 203

PFN	Valid	…
-	0	..
-	0	
-	0	
-	0	…

PFN = 204

PFN	Valid	…
-	0	..
-	0	
28	1	
4	1	…

Reducing Memory Overheads of Paging - PT + PT

PFN = 200

PFN = 201

PFN = 202

PFN = 203

PFN	Valid	…
10	1	..
-	0	
-	0	
-	0	
23	1	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
28	1	
4	1	

Total entries = 16

PFN = 200

PFN	Valid	…
201	1	..
202	0	
203	0	
204	1	…

Total entries = 16 + 4

PFN = 201

PFN	Valid	…
10	1	..
-	0	
-	0	
23	1	…

PFN = 202

PFN = 203

PFN = 204

PFN	Valid	…
-	0	..
-	0	
28	1	
4	1	…

Reducing Memory Overheads of Paging - PT + PT

PFN = 200

PFN = 201

PFN = 202

PFN = 203

PFN	Valid	…
10	1	..
-	0	
-	0	
-	0	
23	1	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
28	1	
4	1	

Total entries = 16

PFN = 200

PFN	Valid	…
201	1	..
202	0	
203	0	
204	1	…

Total entries = 16 + 4

PFN = 201

PFN	Valid	…
10	1	..
-	0	
-	0	
23	1	…

PFN = 203

PFN = 204

PFN	Valid	…
-	0	..
-	0	
28	1	
4	1	…

Reducing Memory Overheads of Paging - PT + PT

PFN = 200

PFN	Valid	…
10	1	..
-	0	
-	0	
-	0	

PFN = 201

PFN	Valid	…
23	1	
-	0	
-	0	
-	0	

PFN = 202

PFN	Valid	…
-	0	
-	0	
-	0	
-	0	

PFN = 203

PFN	Valid	…
-	0	
-	0	
28	1	
4	1	

Total entries = 16

PFN = 200

PFN	Valid	…
201	1	..
202	0	
203	0	
204	1	…

Total entries = 16 + 4

PFN = 201

PFN	Valid	…
10	1	..
-	0	
-	0	
23	1	…

PFN = 204

PFN	Valid	…
-	0	..
-	0	
28	1	
4	1	…

Reducing Memory Overheads of Paging - PT + PT

PFN = 200

PFN	Valid	…
10	1	..
-	0	
-	0	
-	0	

PFN = 201

PFN	Valid	…
23	1	
-	0	
-	0	
-	0	

PFN = 202

PFN	Valid	…
-	0	
-	0	
-	0	
-	0	

PFN = 203

PFN	Valid	…
-	0	
-	0	
28	1	
4	1	

Total entries = 16

PFN = 200

PFN	Valid	…
201	1	..
202	0	
203	0	
204	1	…

PFN = 201

PFN	Valid	…
10	1	..
-	0	
-	0	
23	1	…

PFN = 204

PFN	Valid	…
-	0	..
-	0	
28	1	
4	1	…

Reducing Memory Overheads of Paging - PT + PT

PFN = 200

PFN	Valid	…
10	1	..
-	0	
-	0	
-	0	
23	1	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
28	1	
4	1	

PFN = 201

PFN = 202

PFN = 203

Total entries = 16

PFN = 200

PFN	Valid	…
201	1	..
202	0	
203	0	
204	1	…

Total entries = 16 + 4 - 8

PFN = 201

PFN	Valid	…
10	1	..
-	0	
-	0	
23	1	…

PFN = 204

PFN	Valid	…
-	0	..
-	0	
28	1	
4	1	…

Reducing Memory Overheads of Paging - PT + PT

PFN = 200

PFN = 201

PFN = 202

PFN = 203

PFN	Valid	…
10	1	..
-	0	
-	0	
-	0	
23	1	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
-	0	
28	1	
4	1	

Total entries = 16

Page Directory Entry (PDE)

PFN = 200

PFN	Valid	…
201	1	..
202	0	
203	0	
204	1	…

Total entries = 16 + 4 - 8

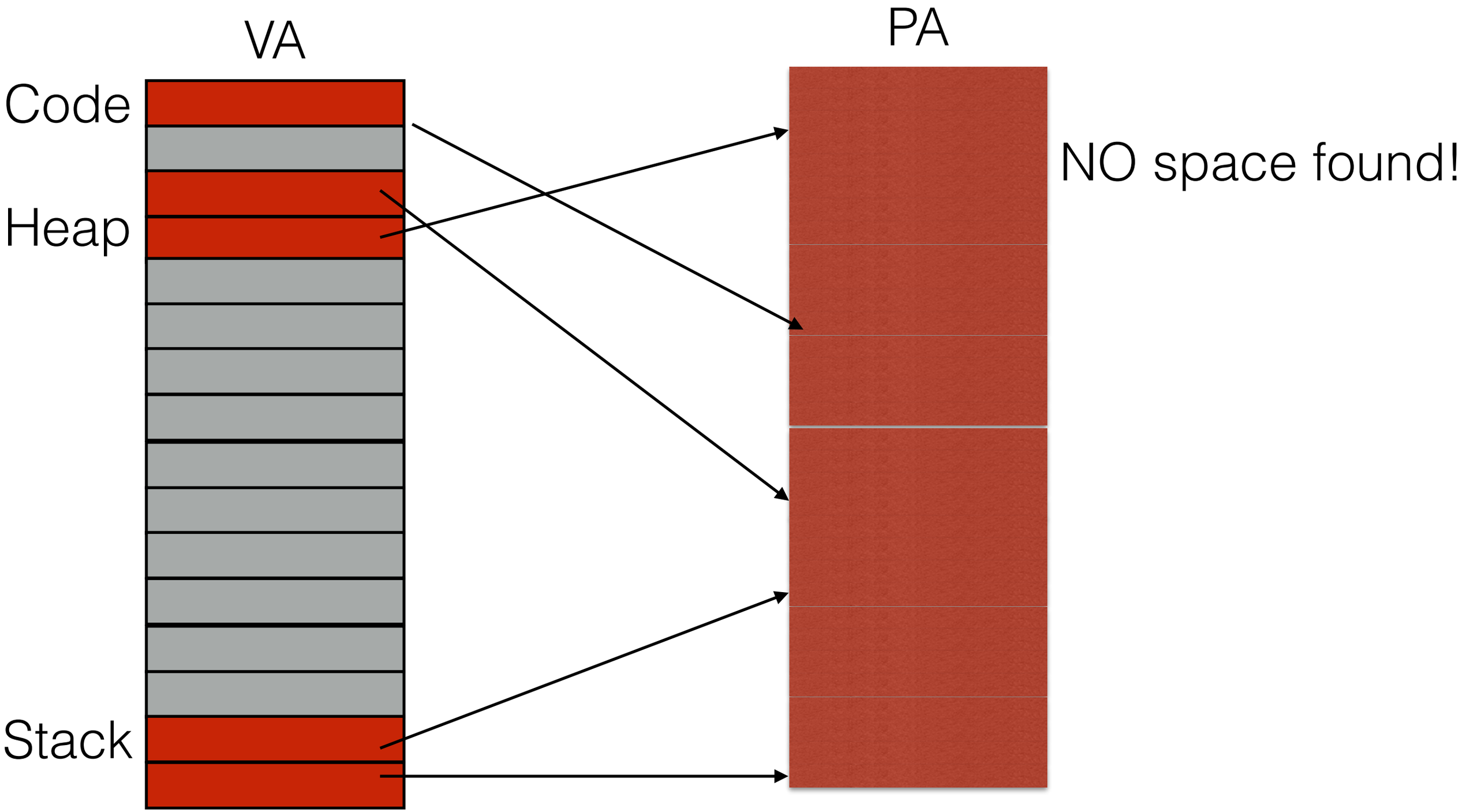
PFN = 201

PFN	Valid	…
10	1	..
-	0	
-	0	
23	1	…

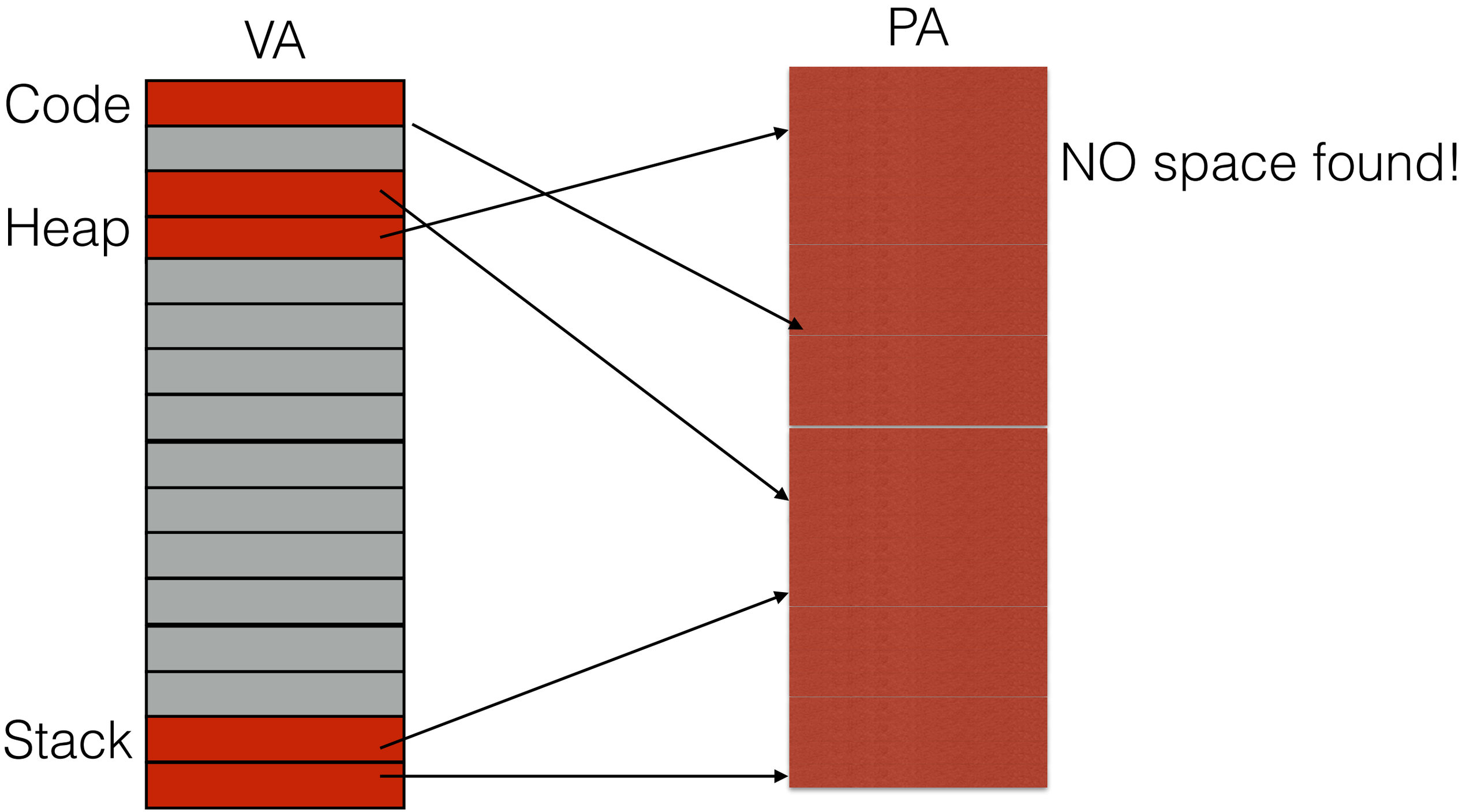
PFN = 204

PFN	Valid	…
-	0	..
-	0	
28	1	
4	1	…

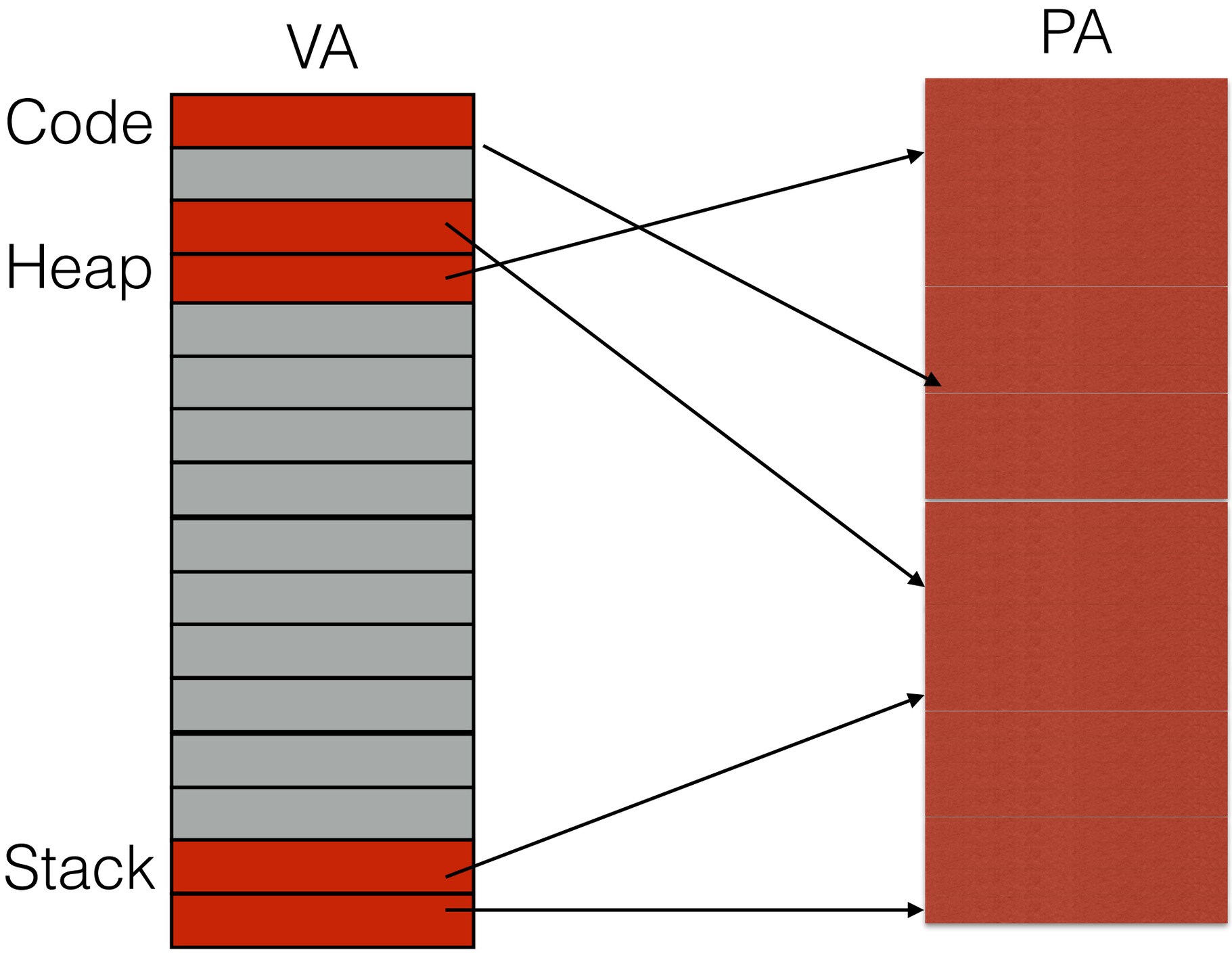
Out of PA space



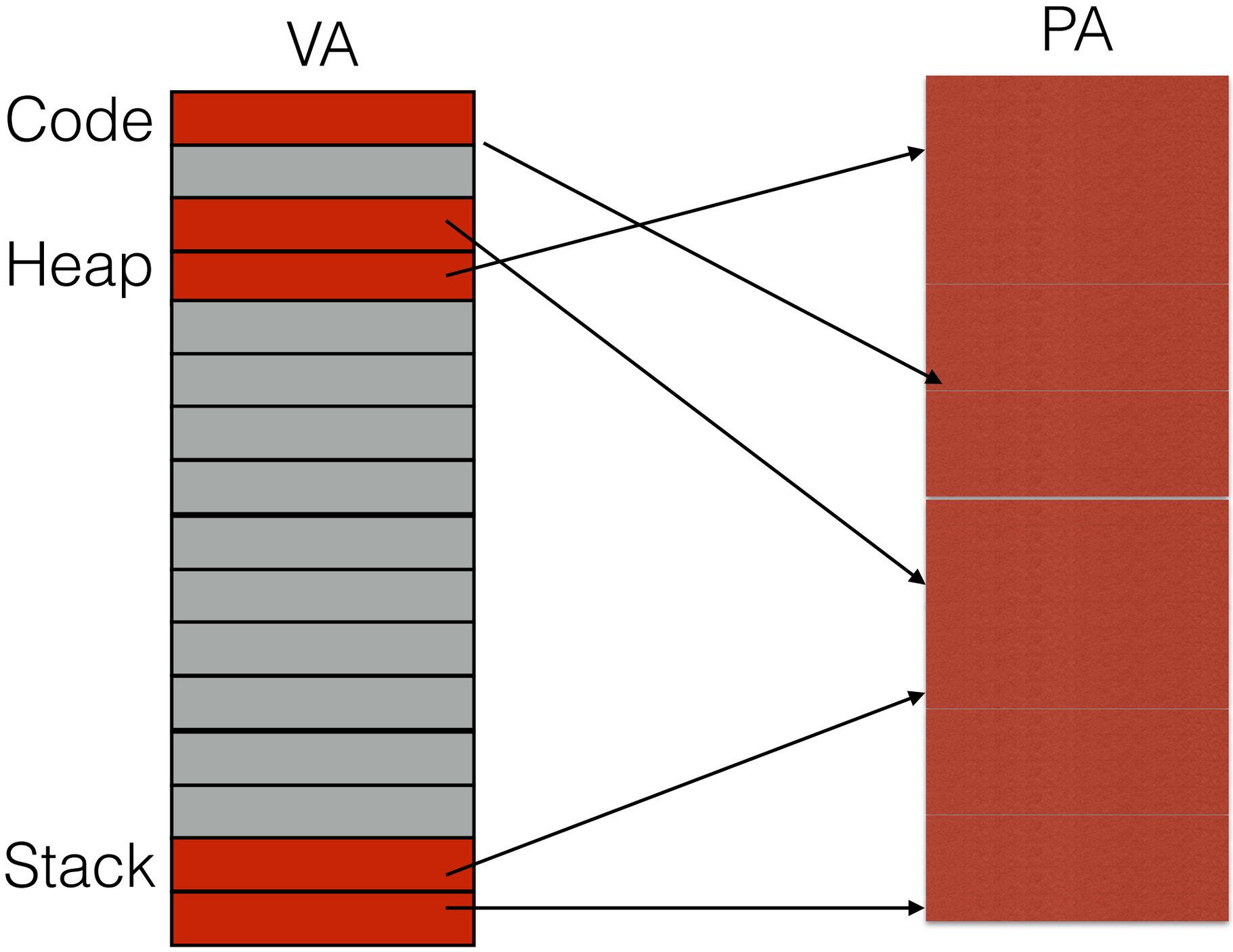
Out of PA space



Out of PA space



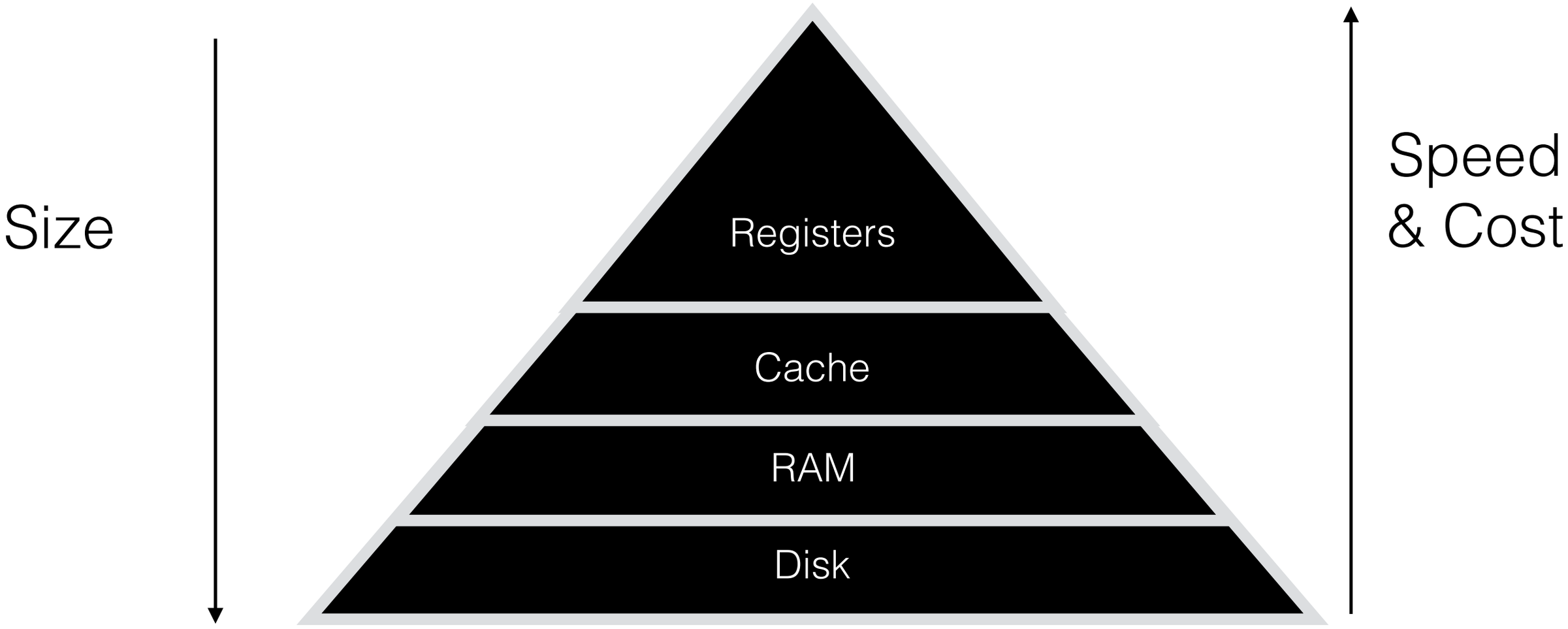
Out of PA space



NO space found!

1. Fail - Kill the process
2. Create more space - Preserving memory contents

Out of PA space - Memory hierarchy



Out of PA space - Virtualisation

Out of PA space - Virtualisation

1. Does the user know whether the PFN came from TLB or memory?

Out of PA space - Virtualisation

1. Does the user know whether the PFN came from TLB or memory?
   - No!

Out of PA space - Virtualisation

1. Does the user know whether the PFN came from TLB or memory?
   - No!
   - Does the user even know if paging is used?

Out of PA space - Virtualisation

1. Does the user know whether the PFN came from TLB or memory?
   - No!
   - Does the user even know if paging is used?
2. Should the user know if the page came from memory or TLB or disk?

Out of PA space - Virtualisation

1. Does the user know whether the PFN came from TLB or memory?
   - No!
   - Does the user even know if paging is used?
2. Should the user know if the page came from memory or TLB or disk?
   - No!

Out of PA space - Virtualisation

1. Does the user know whether the PFN came from TLB or memory?
   - No!
   - Does the user even know if paging is used?
2. Should the user know if the page came from memory or TLB or disk?
   - No!
   - The last time the process used the virtual address, it behaved like memory.

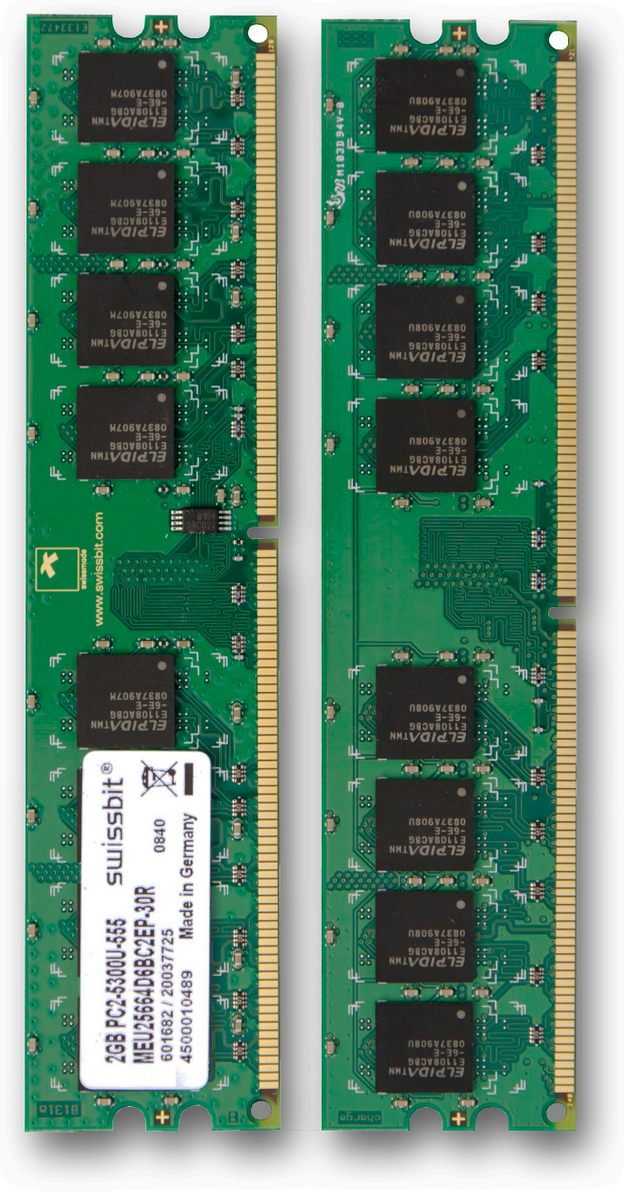
Out of PA space - Virtualisation

1. Does the user know whether the PFN came from TLB or memory?
   - No!
   - Does the user even know if paging is used?
2. Should the user know if the page came from memory or TLB or disk?
   - No!
   - The last time the process used the virtual address, it behaved like memory.
   - The next time the process uses the virtual address, it behaves like memory.

Out of PA space - Virtualisation

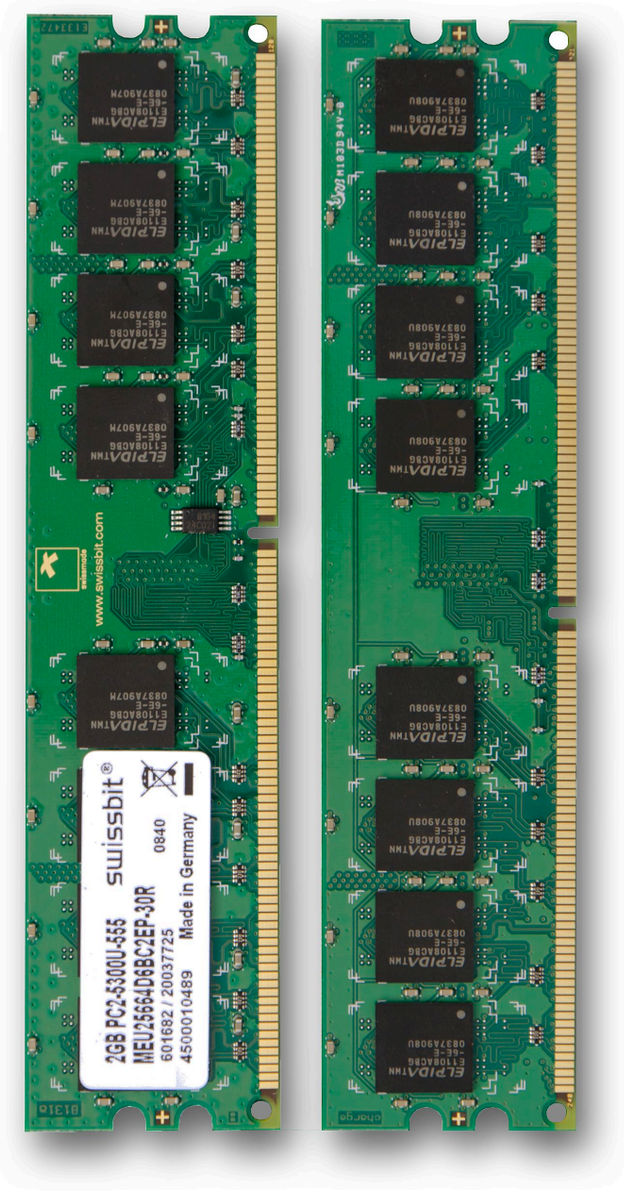
1. Does the user know whether the PFN came from TLB or memory?
   - No!
   - Does the user even know if paging is used?
2. Should the user know if the page came from memory or TLB or disk?
   - No!
   - The last time the process used the virtual address, it behaved like memory.
   - The next time the process uses the virtual address, it behaves like memory.
   - In between, whatever data was stored at that address must be preserved.

Swapping

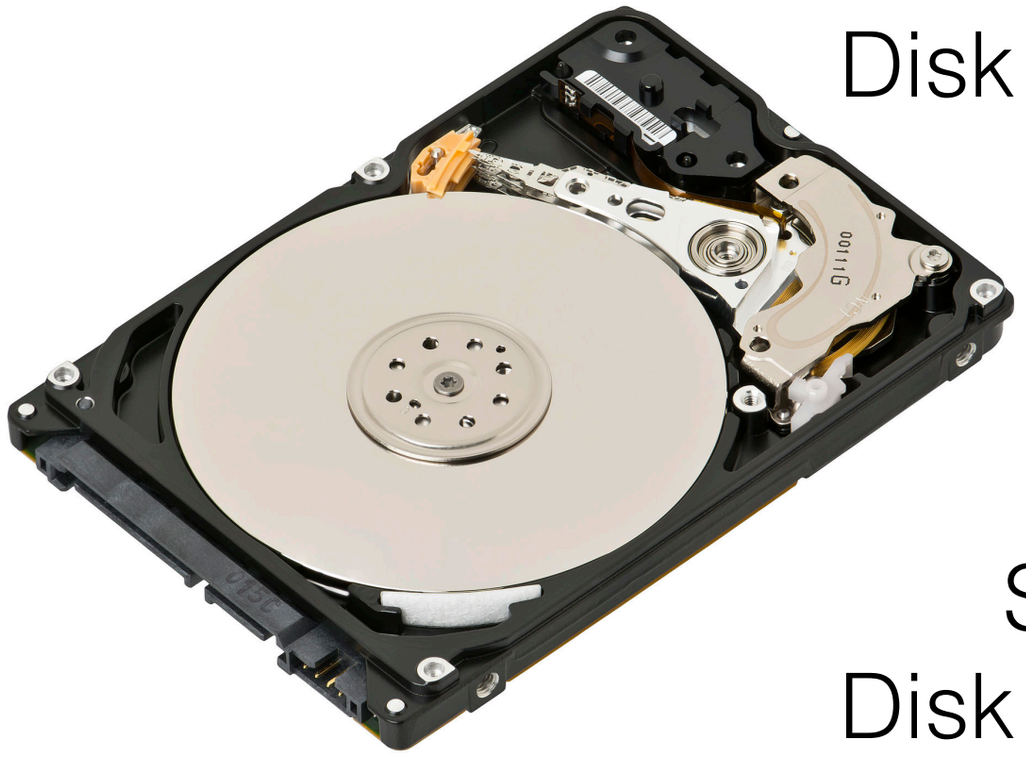

ELPIDATWN
swissbit®
2GB PC2-5300U-555
MEU25664D6BC2EP-30R
0840
Made in Germany

Swapping

Swap in
Disk —> Memory

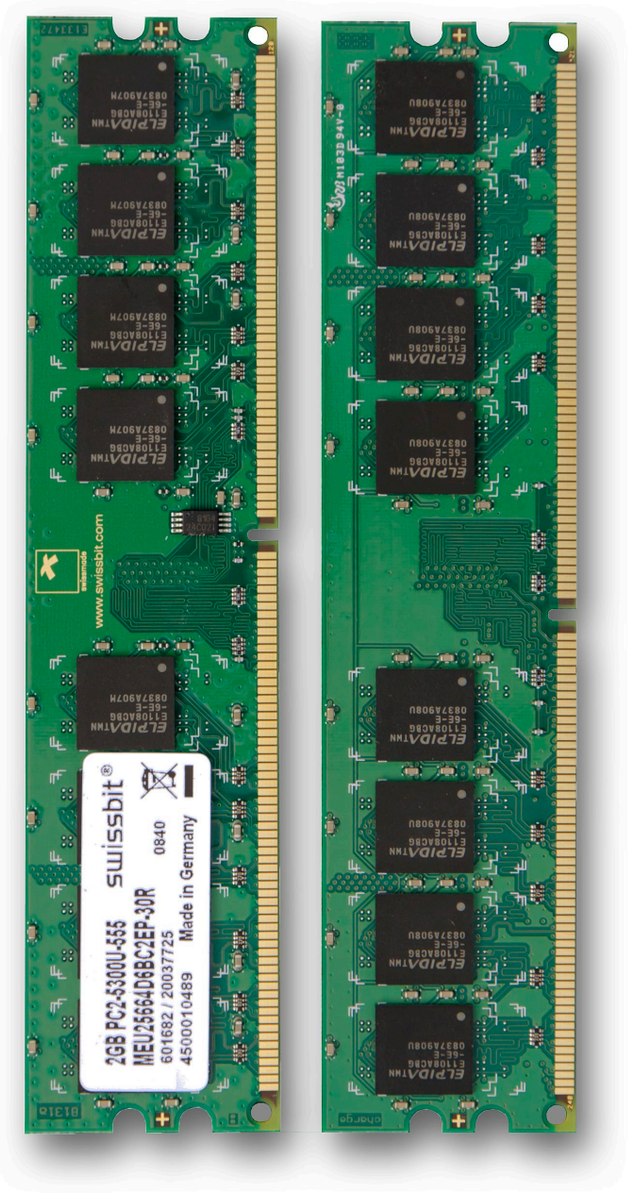


Swapping

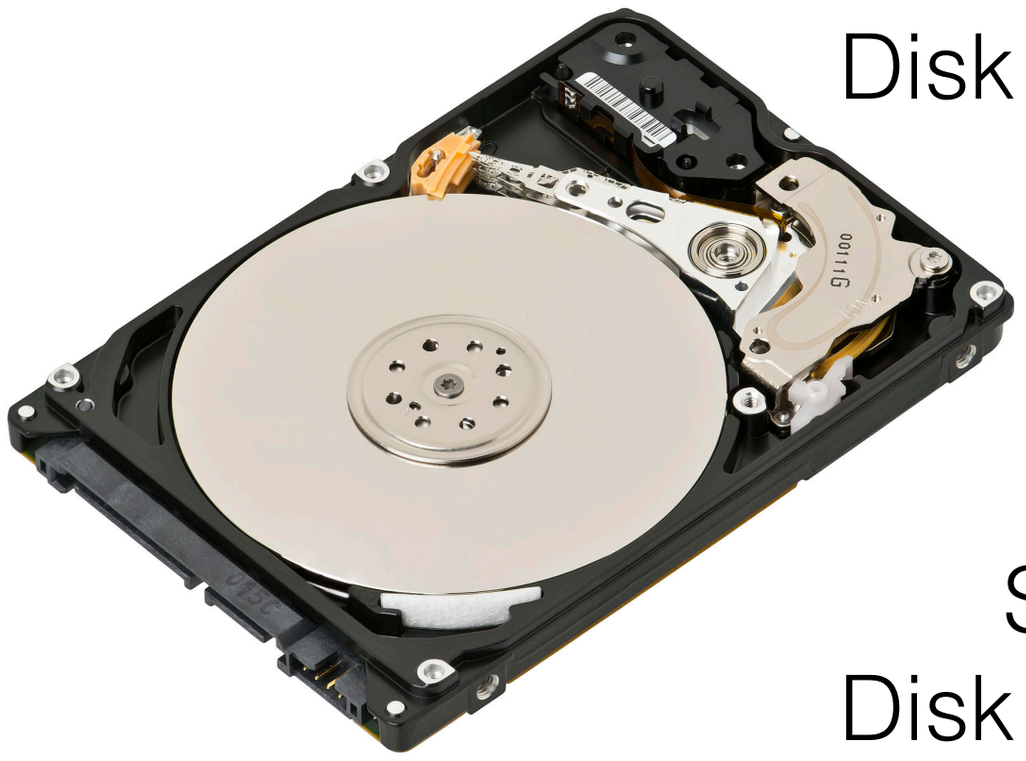


Swap in
Disk —> Memory

Swap out
Disk <— Memory

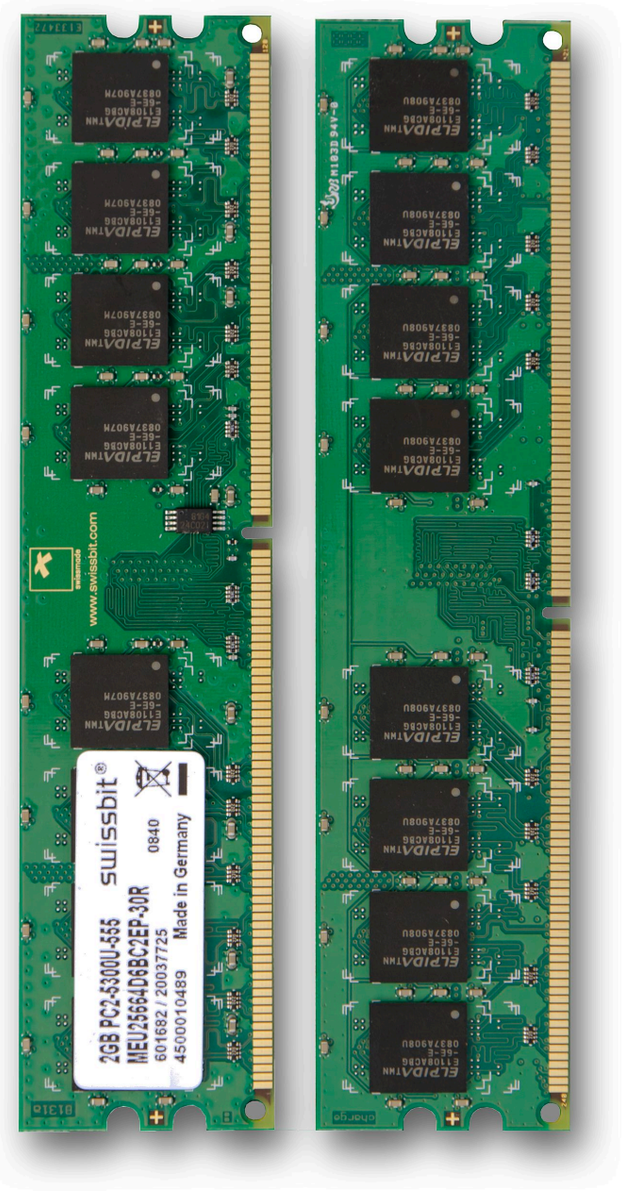


Swapping



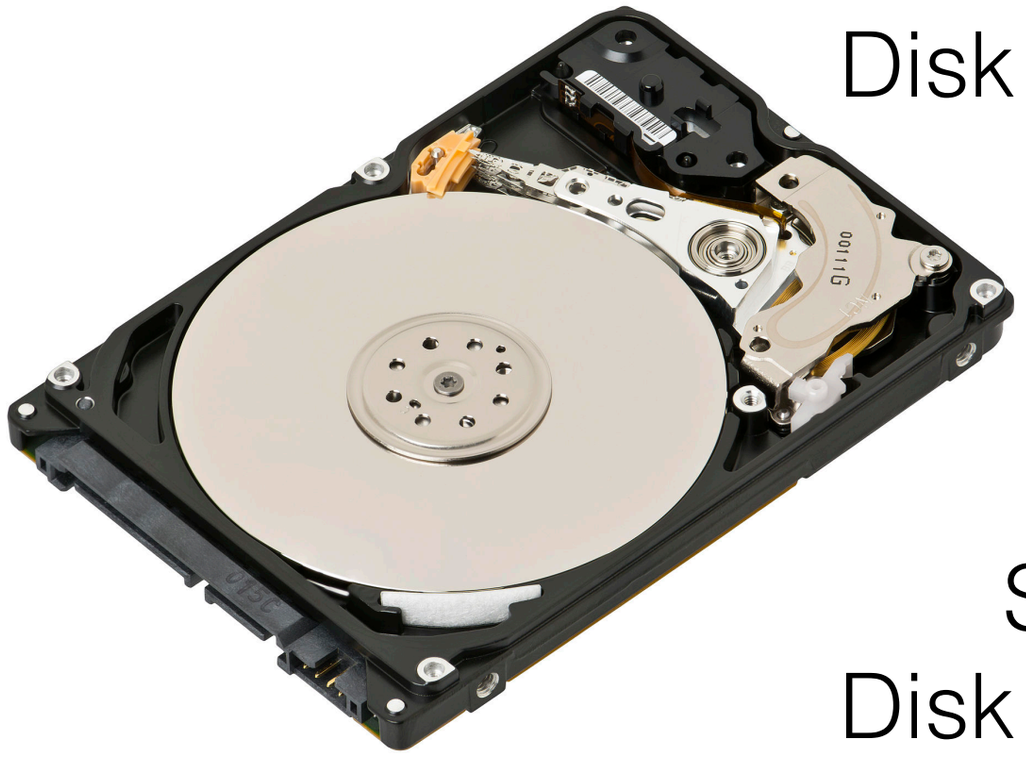
Swap in
Disk —> Memory

Swap out
Disk <— Memory



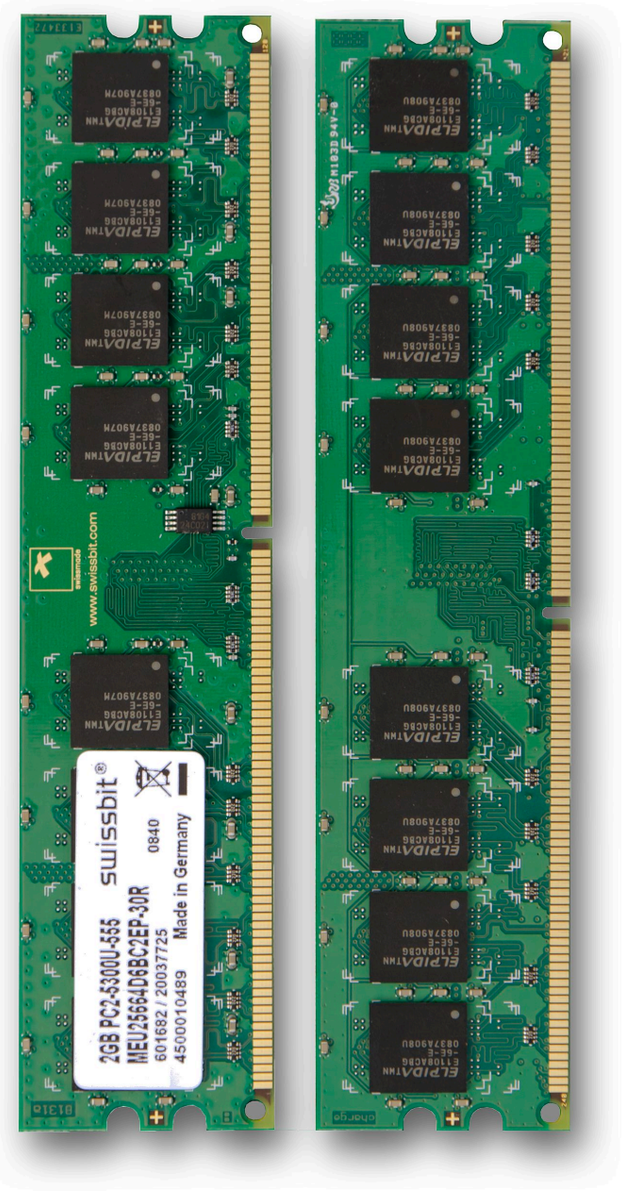
Done well : Memory as large as disk, as fast as RAM

Swapping



Swap in
Disk —> Memory

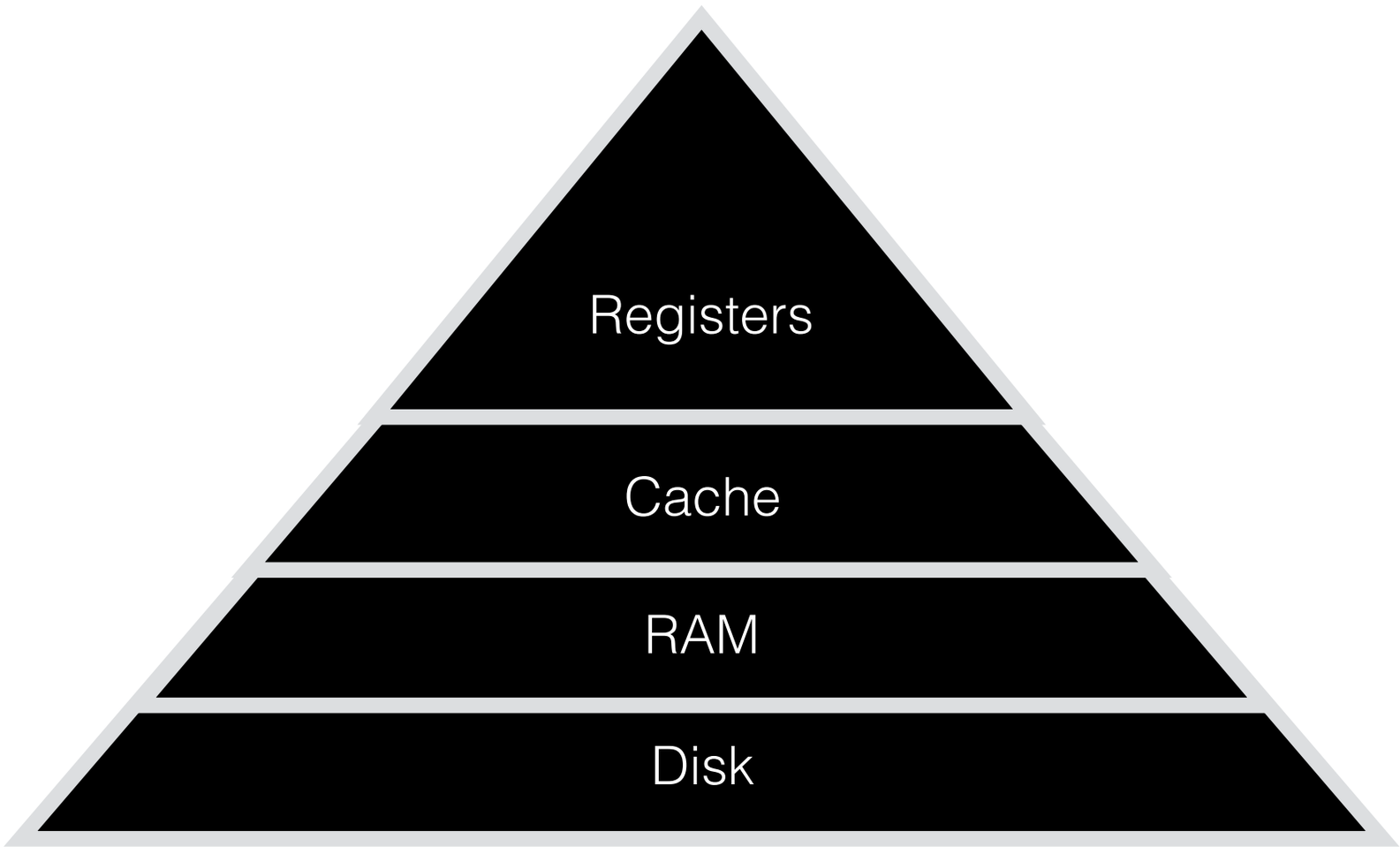
Swap out
Disk <— Memory



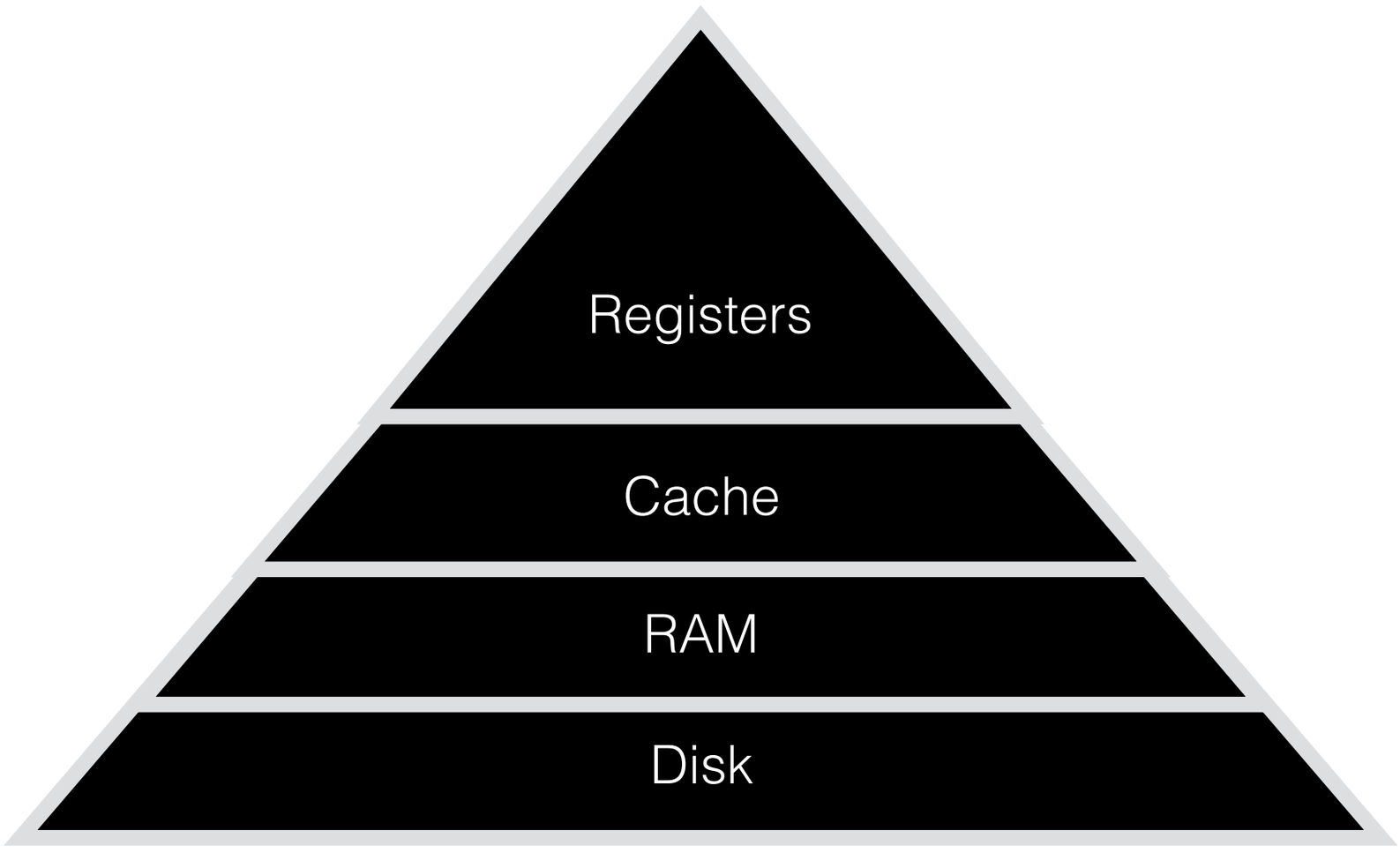
Done well : Memory as large as disk, as fast as RAM

Done bad : Memory as small as RAM, as slow as disk

TLB miss v/s Page Fault

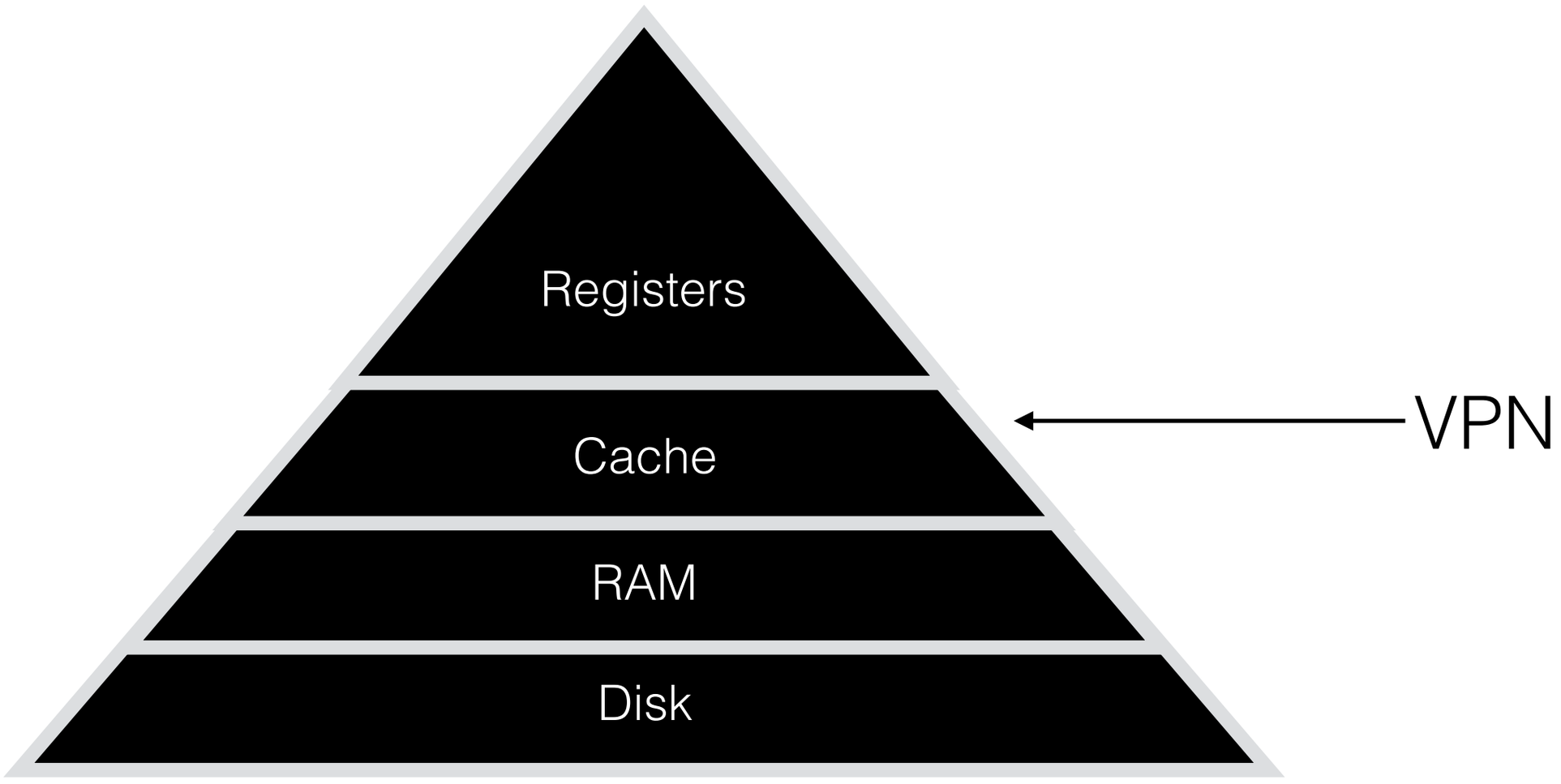


TLB miss v/s Page Fault

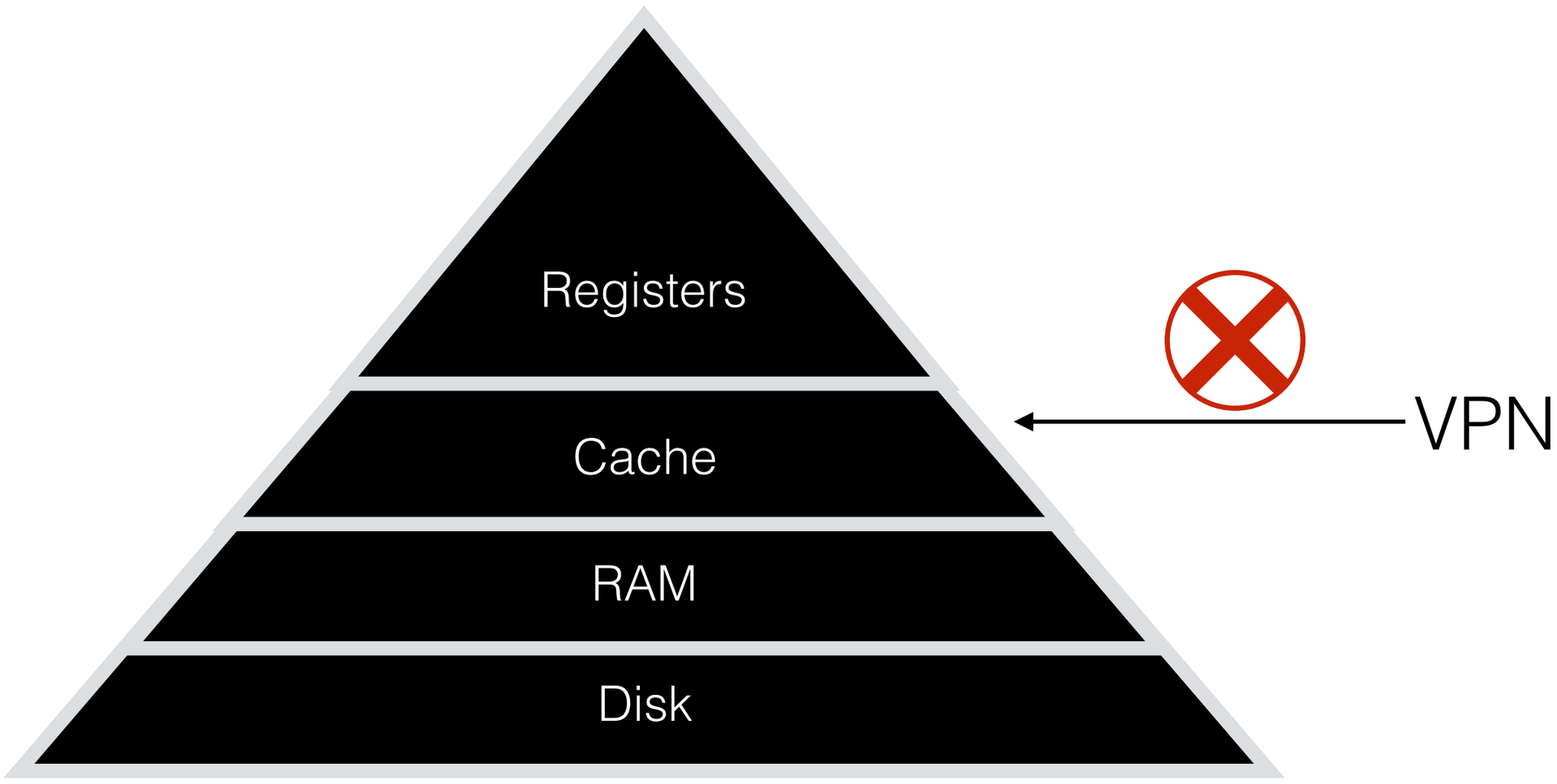


VPN

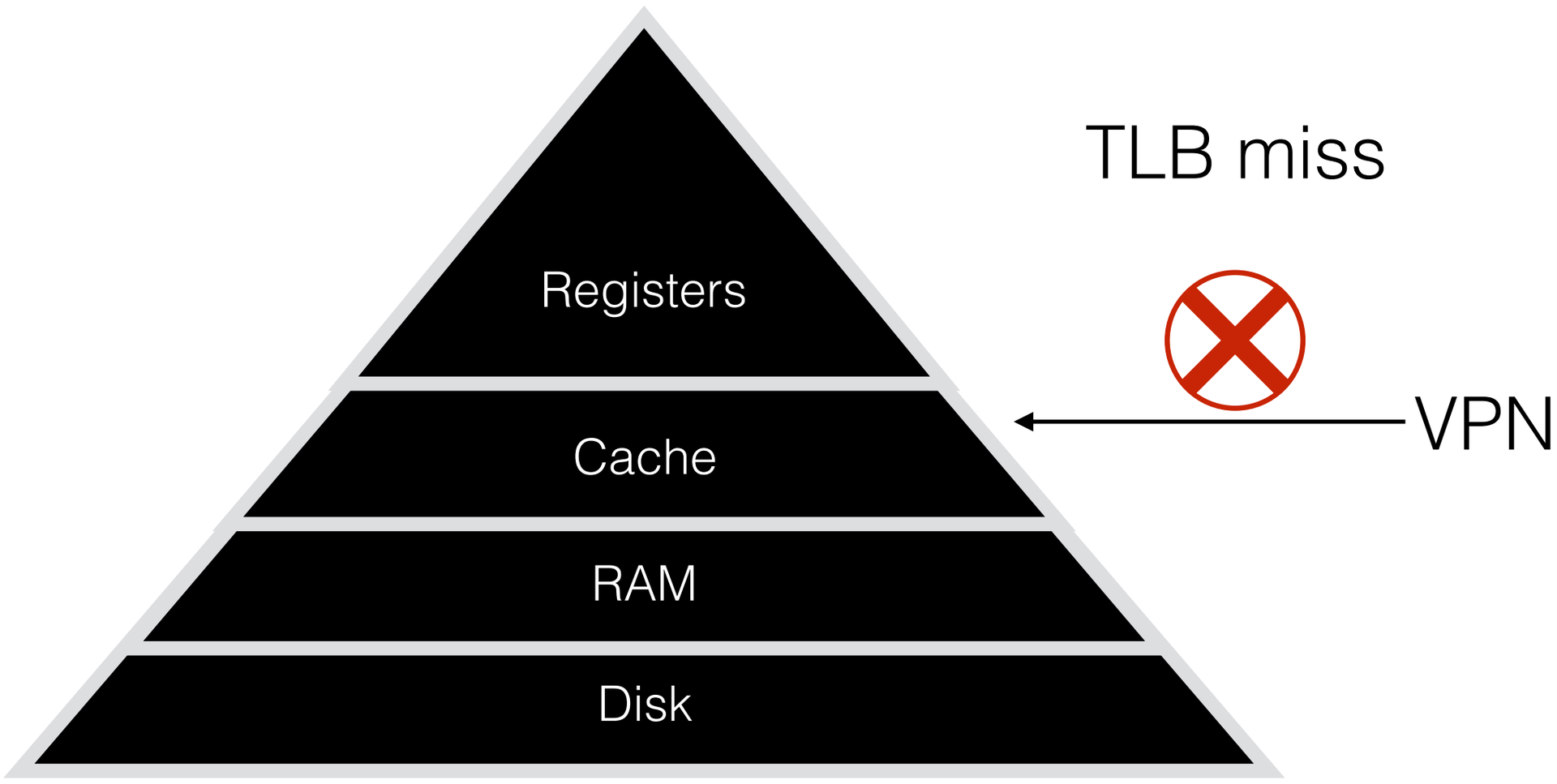
TLB miss v/s Page Fault



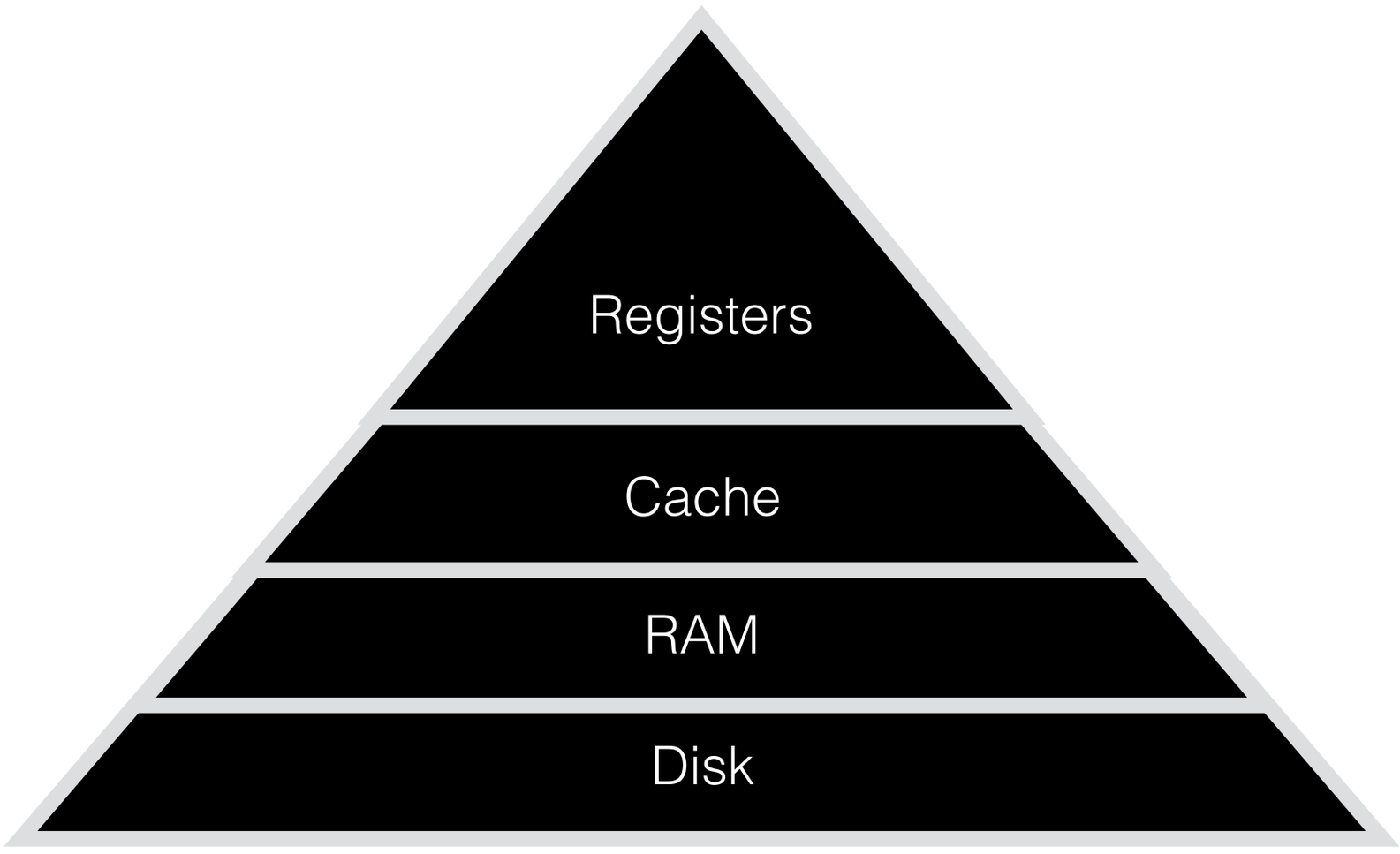
TLB miss v/s Page Fault



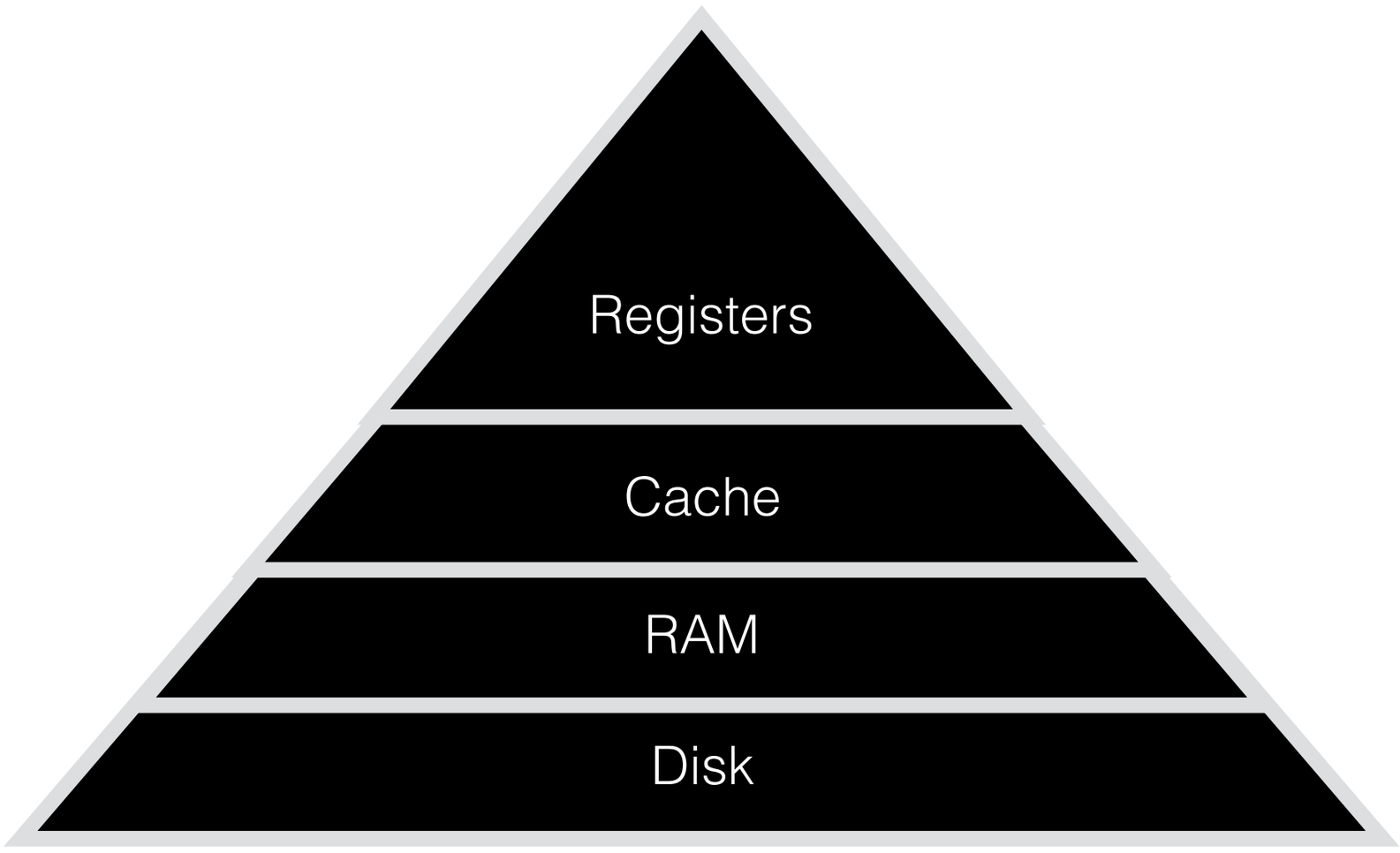
TLB miss v/s Page Fault



TLB miss v/s Page Fault

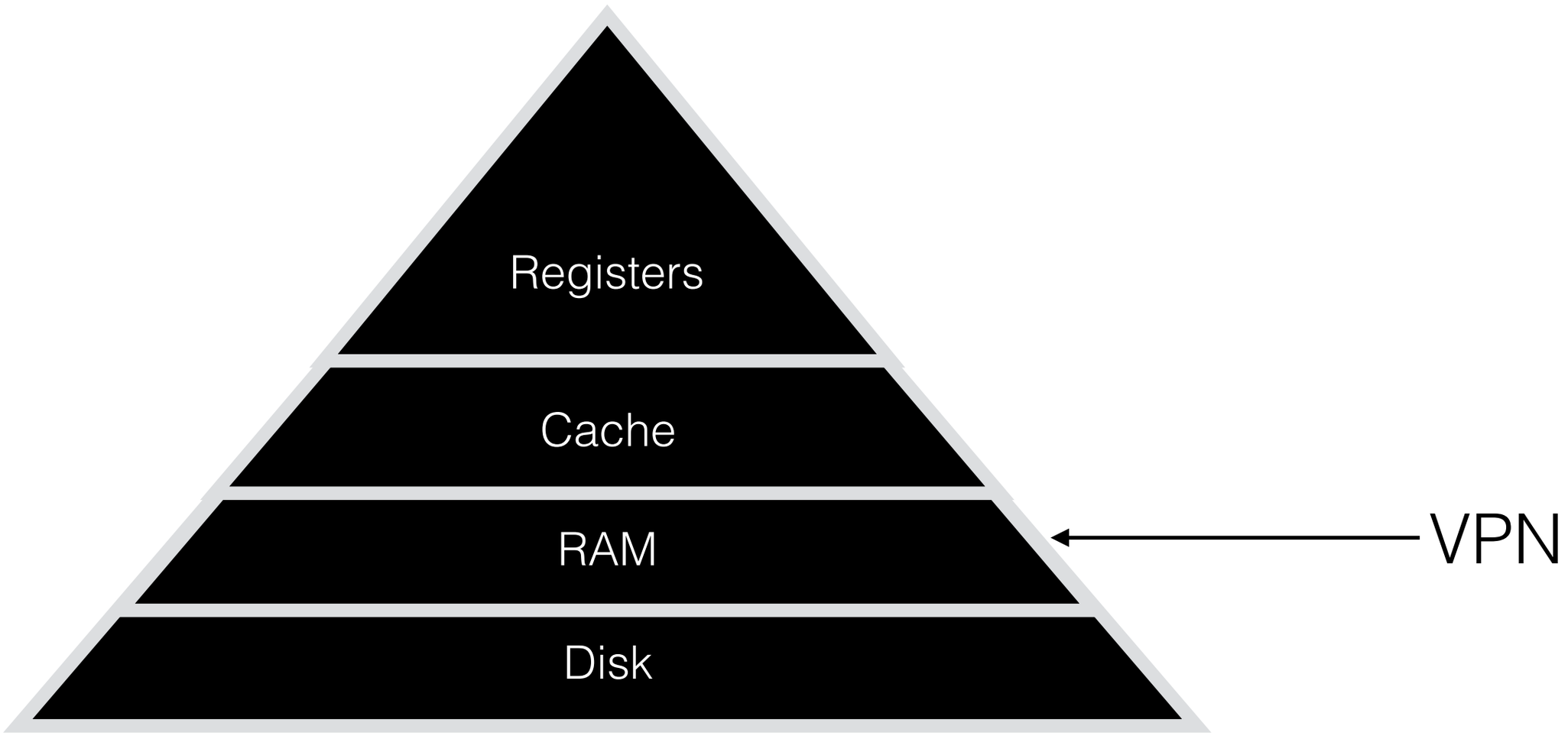


TLB miss v/s Page Fault

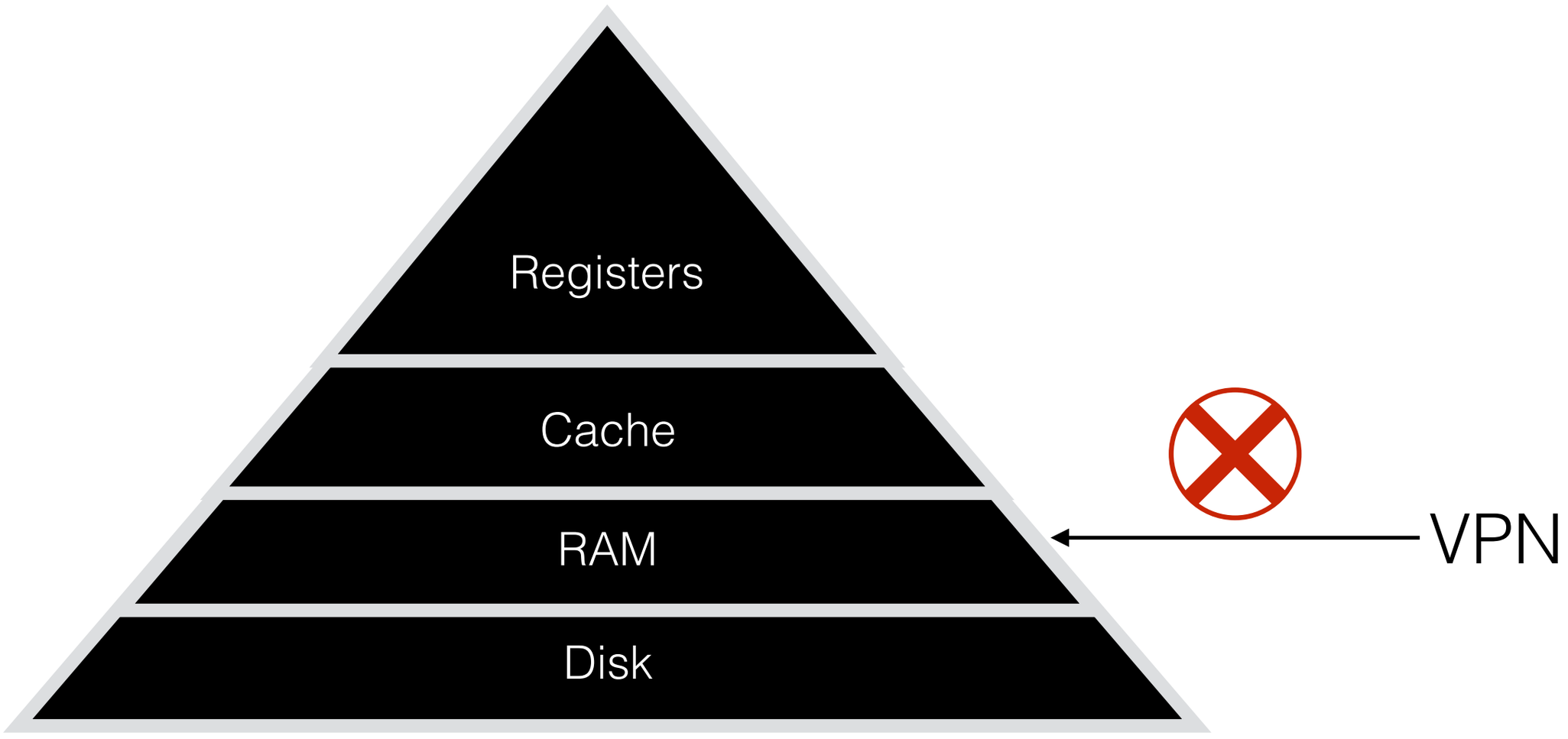


VPN

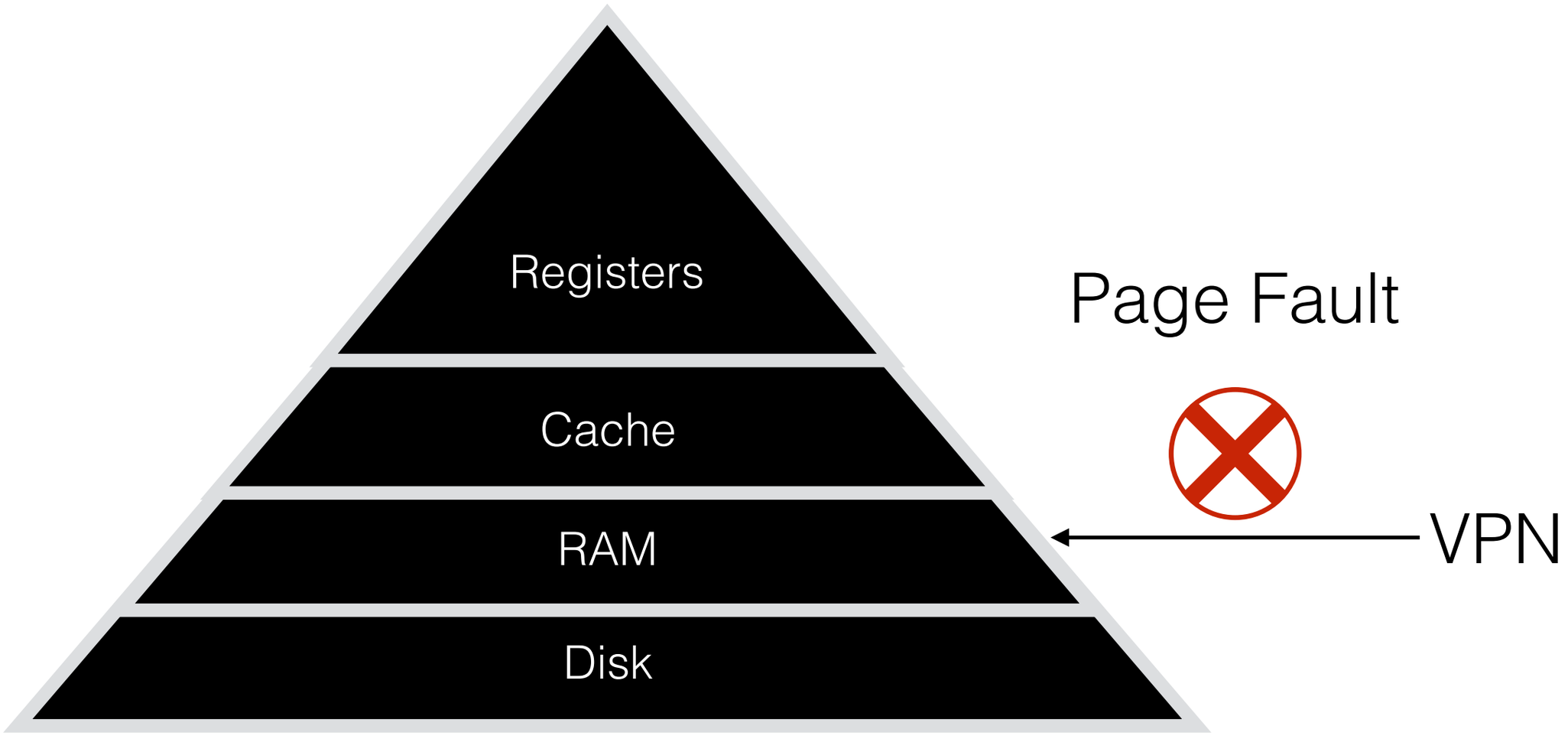
TLB miss v/s Page Fault



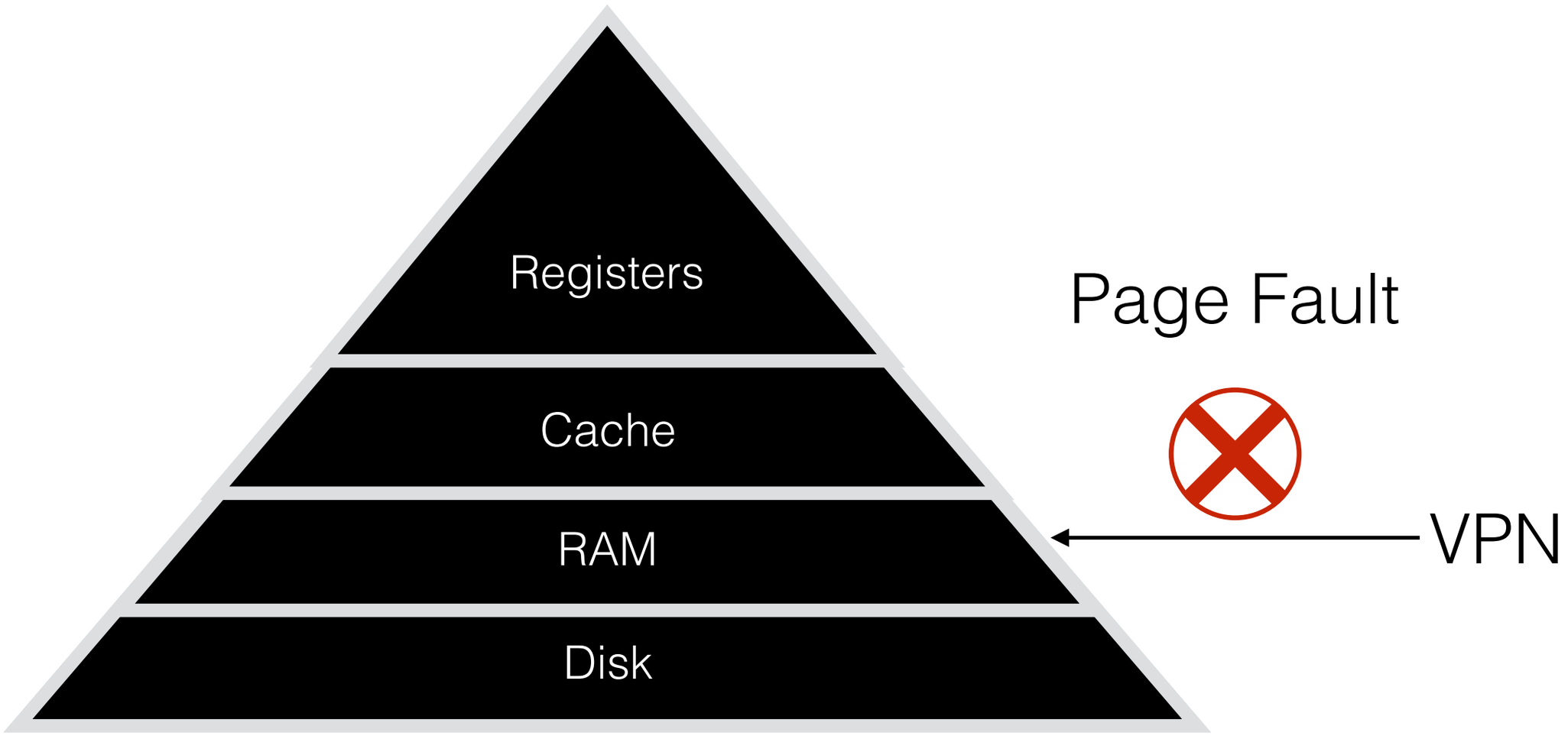
TLB miss v/s Page Fault



TLB miss v/s Page Fault

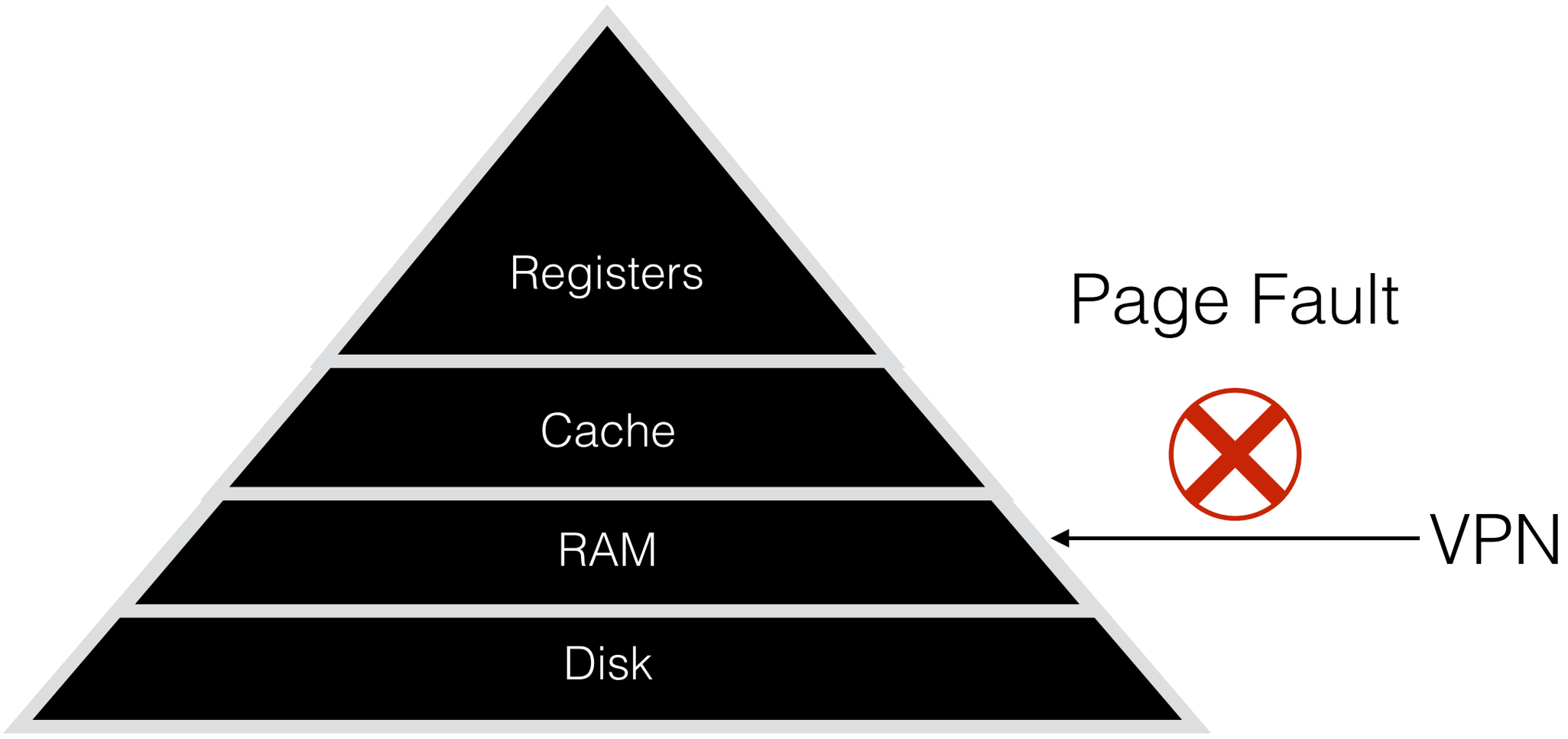


TLB miss v/s Page Fault



- Every page fault preceded by a TLB miss

TLB miss v/s Page Fault



- Every page fault preceded by a TLB miss
- Every TLB miss does not generate a page fault

Swapping Out

Swapping Out

1. Remove the translation from TLB if exists

Swapping Out

1. Remove the translation from TLB if exists
2. Copy the contents of page to disk

Swapping Out

1. Remove the translation from TLB if exists
2. Copy the contents of page to disk
3. Update the PT entry to indicate the page is on disk

Swapping Out

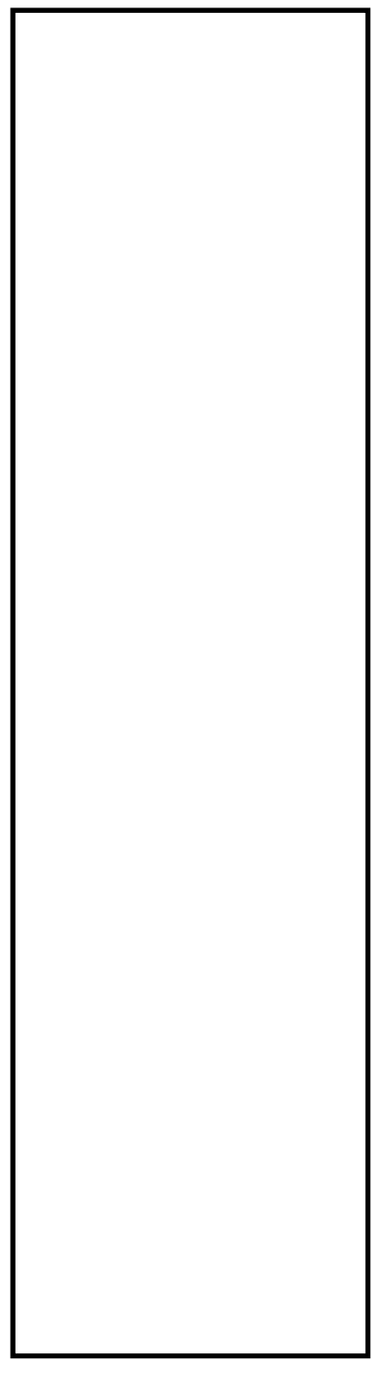
Address space

Disk

TLB

VPN	PFN
10	30
23	40
40	50
50	30

Physical Memory



Swapping Out

Address space

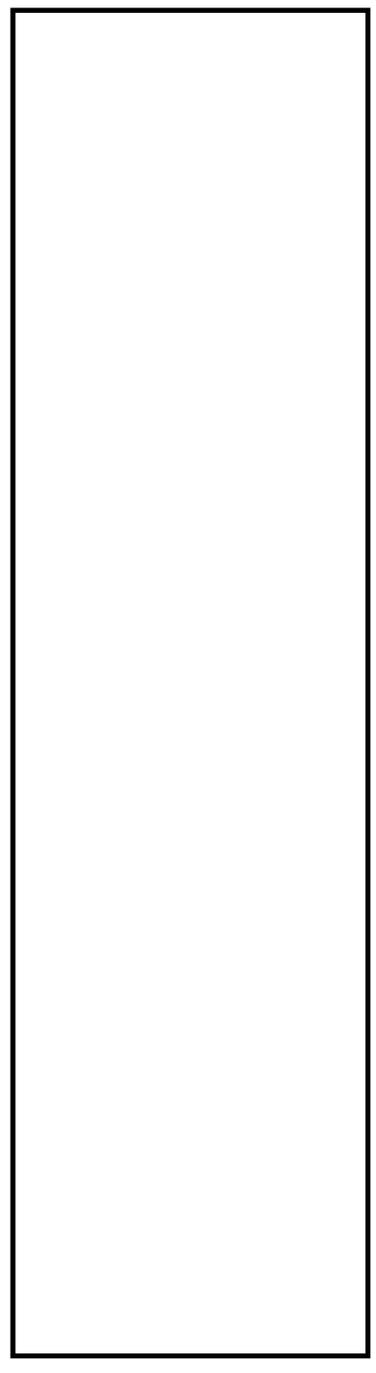
Disk

Swap out VPN =10

TLB

VPN	PFN
10	30
23	40
40	50
50	30

Physical Memory



Swapping Out

Address space

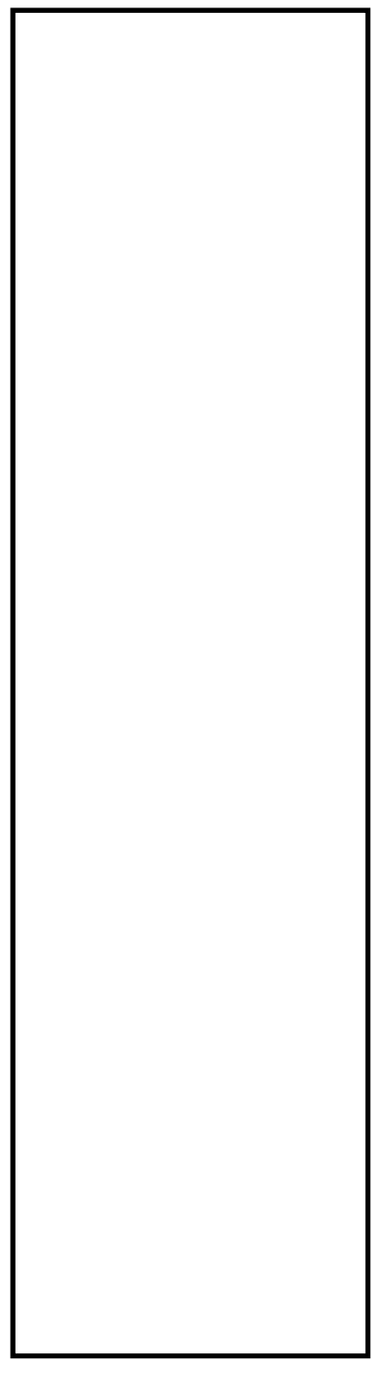
Disk

Swap out VPN =10

TLB

VPN	PFN
10	30
23	40
40	50
50	30

Physical Memory



Swapping Out

Address space

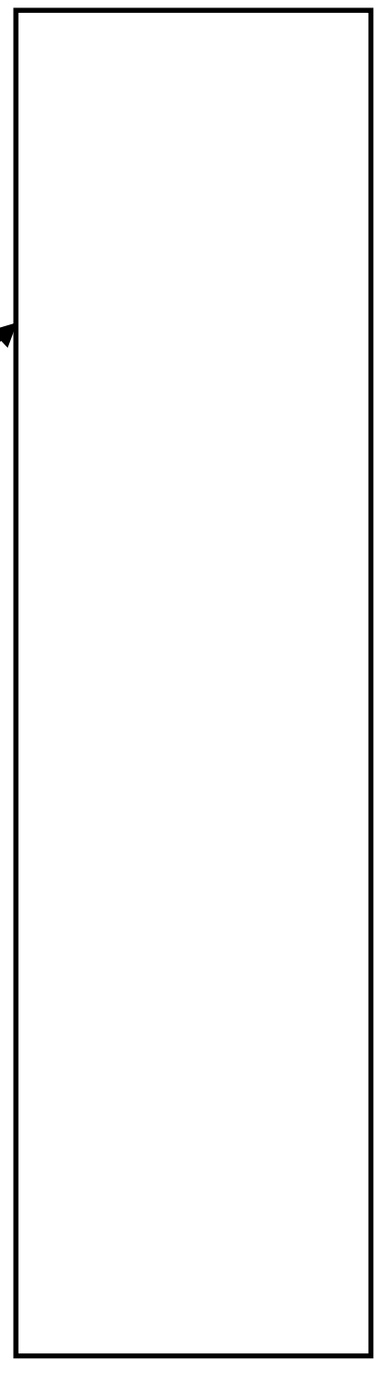
Disk

Swap out VPN =10

TLB

VPN	PFN
10	30
23	40
40	50
50	30

Physical Memory



Swapping Out

Address space

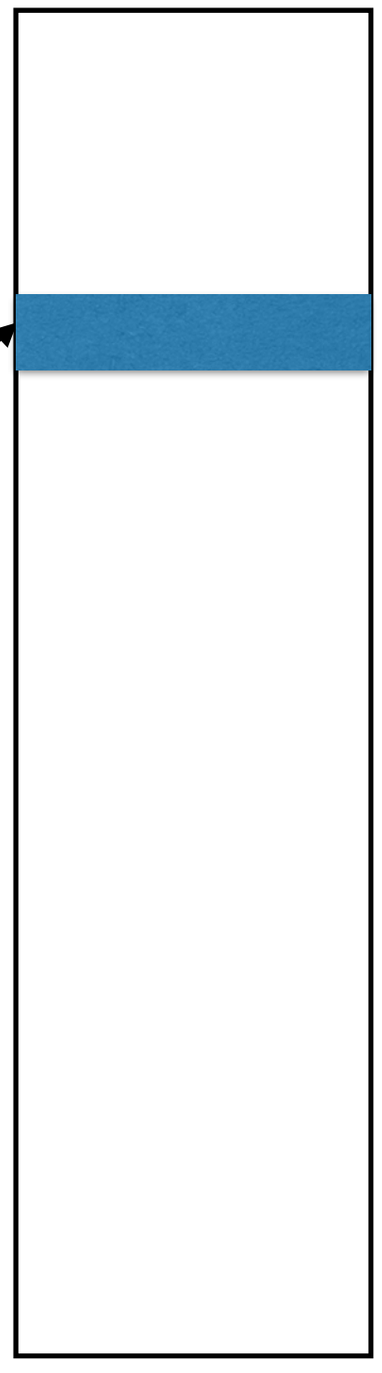
Disk

Swap out VPN =10

TLB

VPN	PFN
10	30
23	40
40	50
50	30

Physical Memory



Swapping Out

Address space

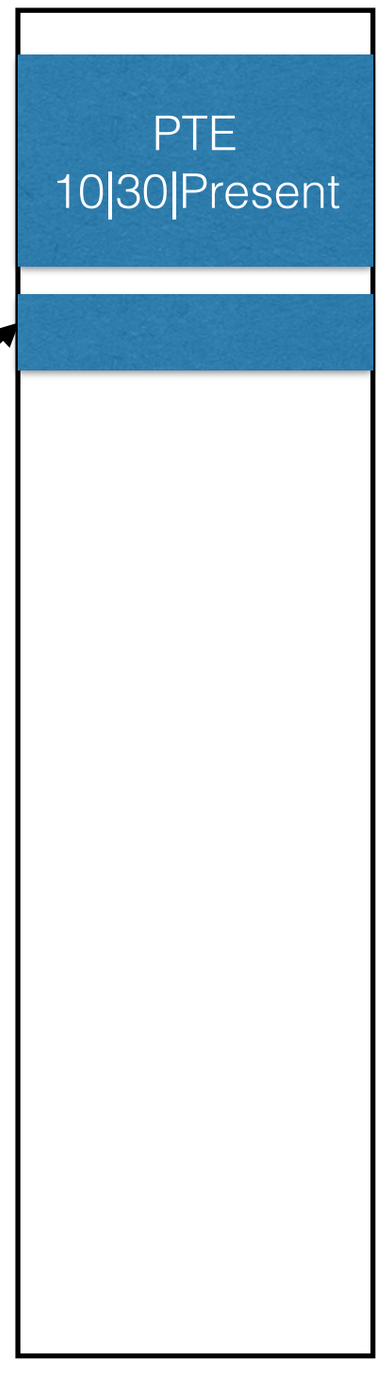
Disk

Swap out VPN =10

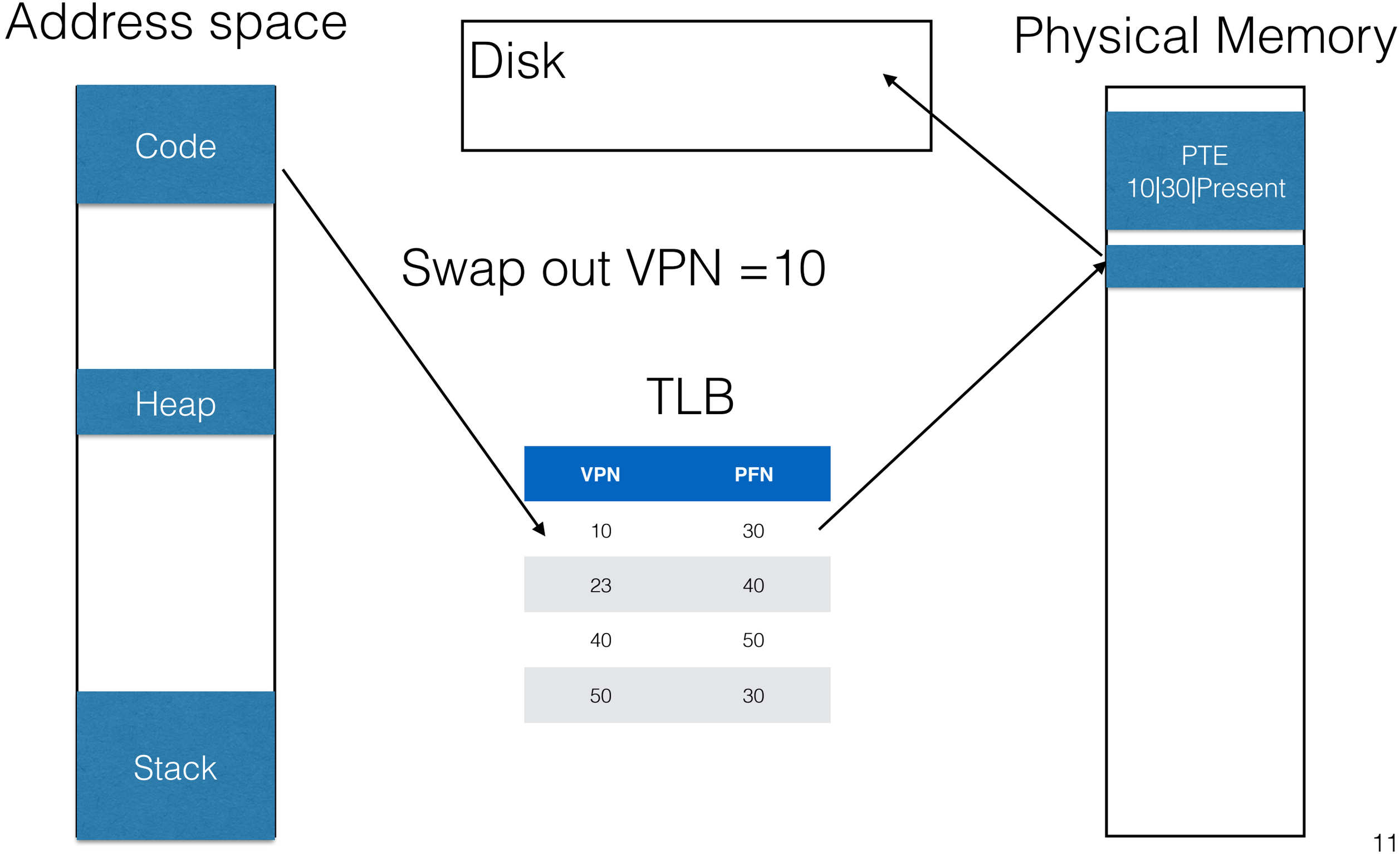
TLB

VPN	PFN
10	30
23	40
40	50
50	30

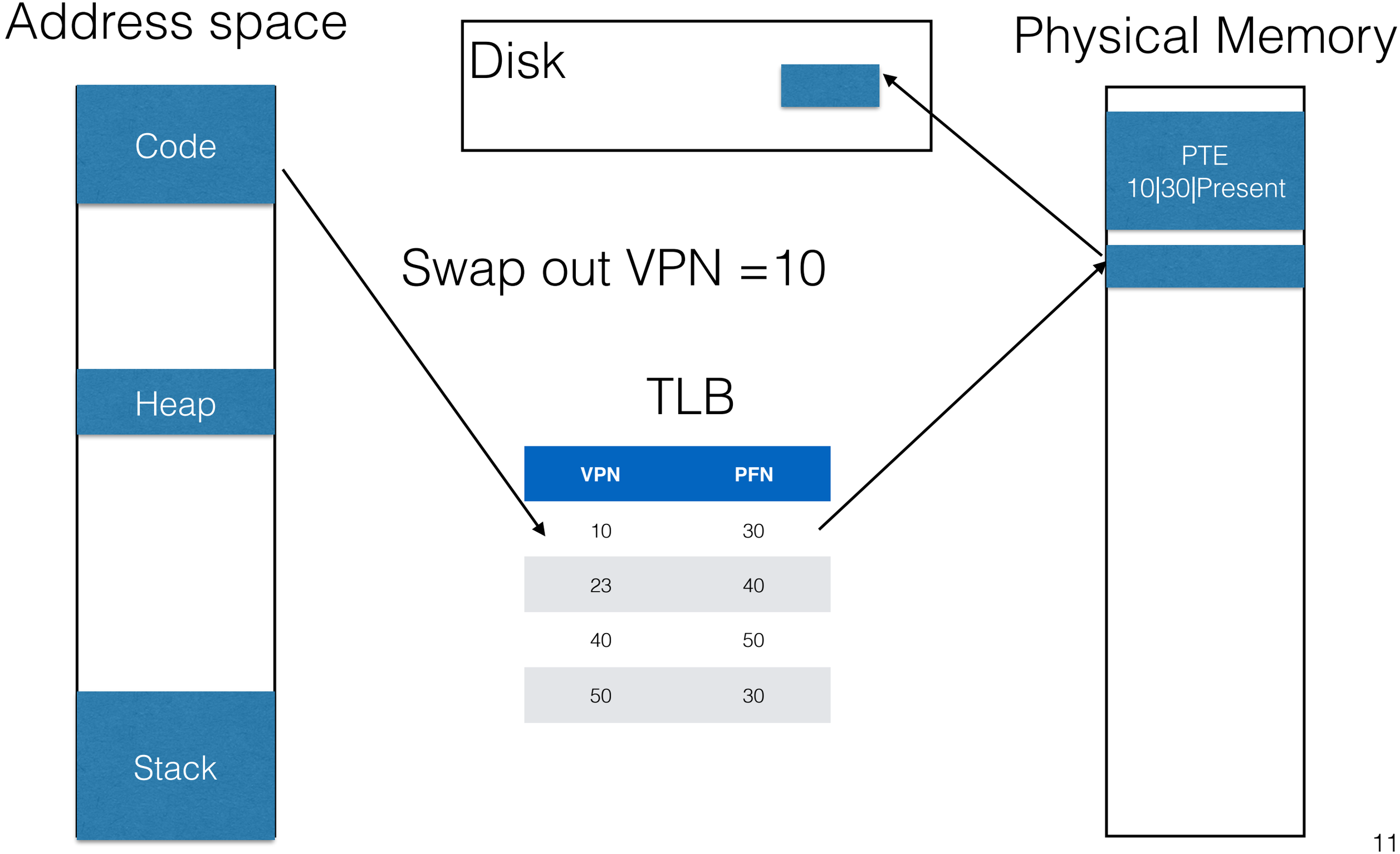
Physical Memory



Swapping Out



Swapping Out



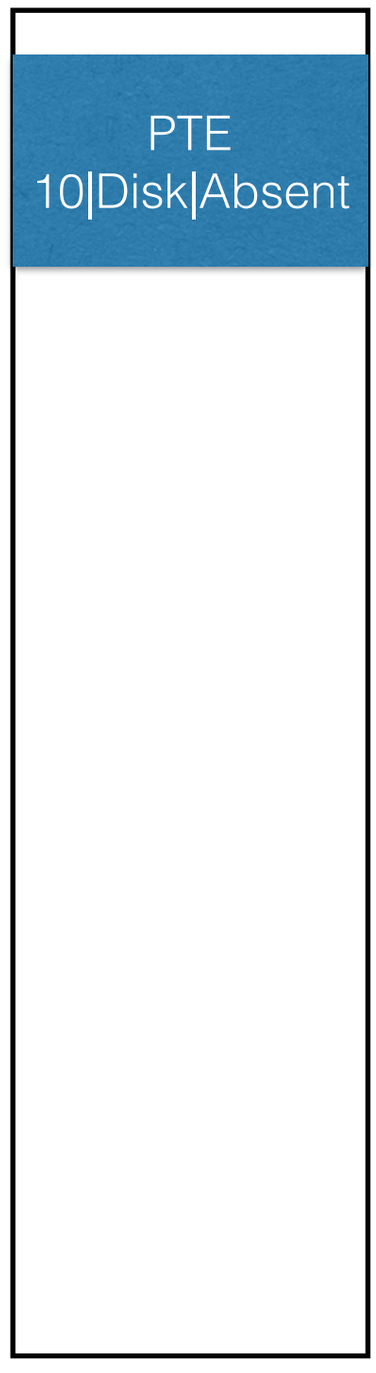
Swapping Out

Address space

TLB

VPN	PFN
23	40
40	50
50	30

Physical Memory



Swapping Out Speed

1. Remove the translation from TLB if exists
2. Copy the contents of page to disk
3. Update the PT entry to indicate the page is on disk

Swapping Out Speed

1. Remove the translation from TLB if exists Fast
2. Copy the contents of page to disk
3. Update the PT entry to indicate the page is on disk

Swapping Out Speed

1. Remove the translation from TLB if exists	Fast
2. Copy the contents of page to disk	Slow
3. Update the PT entry to indicate the page is on disk	

Swapping Out Speed

1. Remove the translation from TLB if exists	Fast
2. Copy the contents of page to disk	Slow
3. Update the PT entry to indicate the page is on disk	Reasonably Fast

Optimizing Swapping Out

1. Remember OS is a responsible program
    1. Optimizes stuff in the background!
2. Each page has dedicated place on disk
3. During idle time, OS writes active memory pages to disk
4. Pages where memory content == disk content —> clean
5. Pages where memory content != disk content —> dirty
    1. Example:
        1. Write to disk
        2. x[i] = x[i] + 5
        3. Dirty! Disk and Memory don’t have same contents

Swapping in

Swapping in

1. Stop the instruction that is trying to translate the address until we can retrieve the contents.

Swapping in

1. Stop the instruction that is trying to translate the address until we can retrieve the contents.
2. Allocate a page in memory to hold the new page contents.

Swapping in

1. Stop the instruction that is trying to translate the address until we can retrieve the contents.
2. Allocate a page in memory to hold the new page contents.
3. Locate the page on disk using the page table entry.

Swapping in

1. Stop the instruction that is trying to translate the address until we can retrieve the contents.
2. Allocate a page in memory to hold the new page contents.
3. Locate the page on disk using the page table entry.
4. Copy the contents of the page from disk.

Swapping in

1. Stop the instruction that is trying to translate the address until we can retrieve the contents.
2. Allocate a page in memory to hold the new page contents.
3. Locate the page on disk using the page table entry.
4. Copy the contents of the page from disk.
5. Update the page table entry to indicate that the page is in memory.

Swapping in

1. Stop the instruction that is trying to translate the address until we can retrieve the contents.
2. Allocate a page in memory to hold the new page contents.
3. Locate the page on disk using the page table entry.
4. Copy the contents of the page from disk.
5. Update the page table entry to indicate that the page is in memory.
6. Load the TLB.

Swapping in

1. Stop the instruction that is trying to translate the address until we can retrieve the contents.
2. Allocate a page in memory to hold the new page contents.
3. Locate the page on disk using the page table entry.
4. Copy the contents of the page from disk.
5. Update the page table entry to indicate that the page is in memory.
6. Load the TLB.
7. Restart the instruction that was addressing the virtual address retrieved.

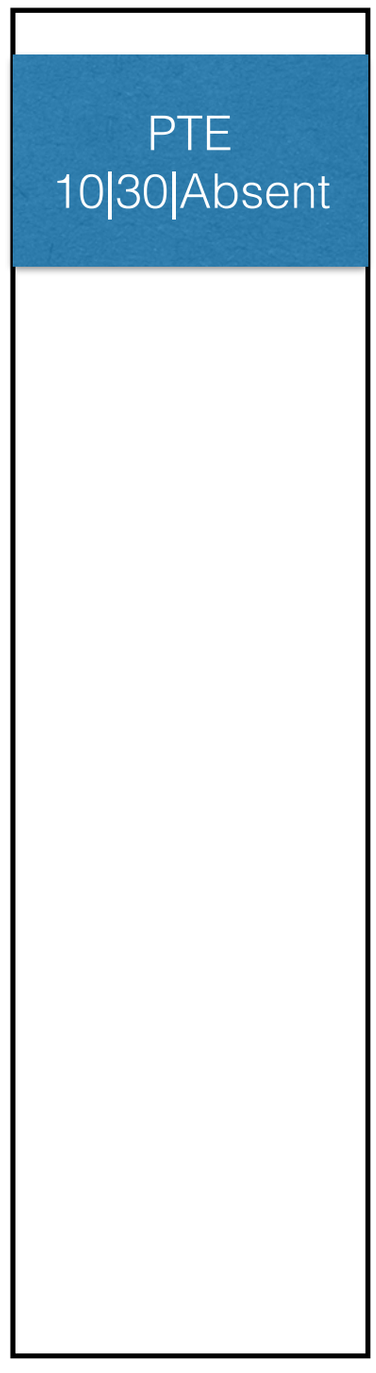
Swapping in

Address space

TLB

VPN	PFN
23	40
40	50
50	30

Physical Memory



Swapping in

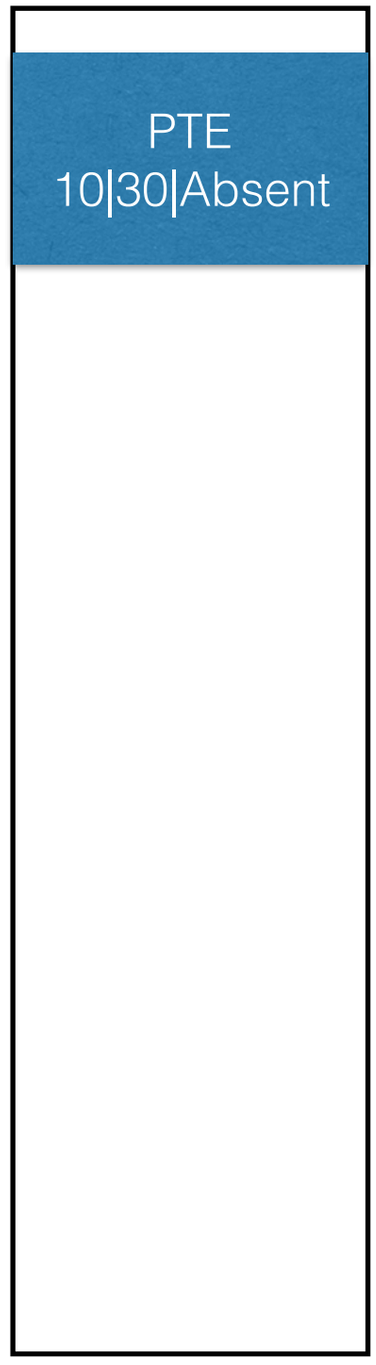
Address space

LOAD VA 10

TLB

VPN	PFN
23	40
40	50
50	30

Physical Memory



Swapping in

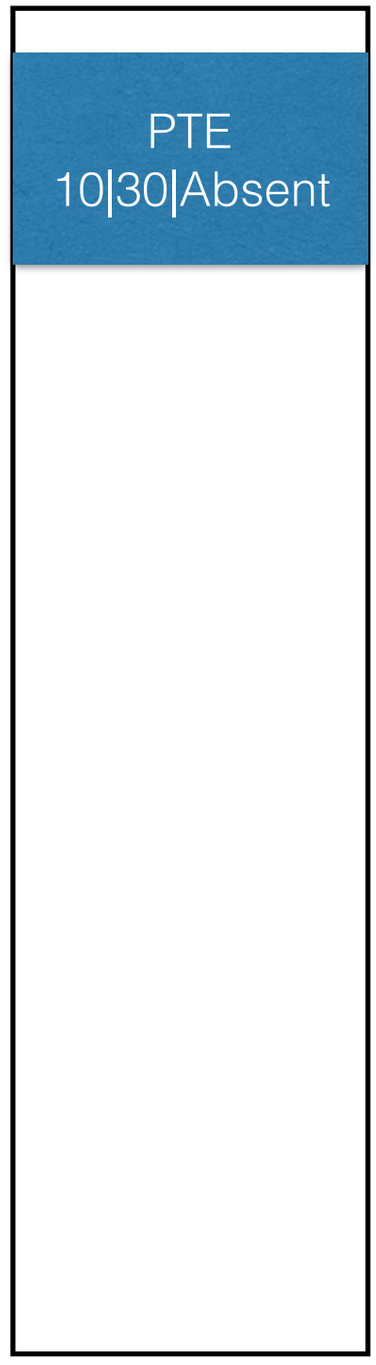
Address space

LOAD VA 10

TLB

VPN	PFN
23	40
40	50
50	30

Physical Memory



Swapping in

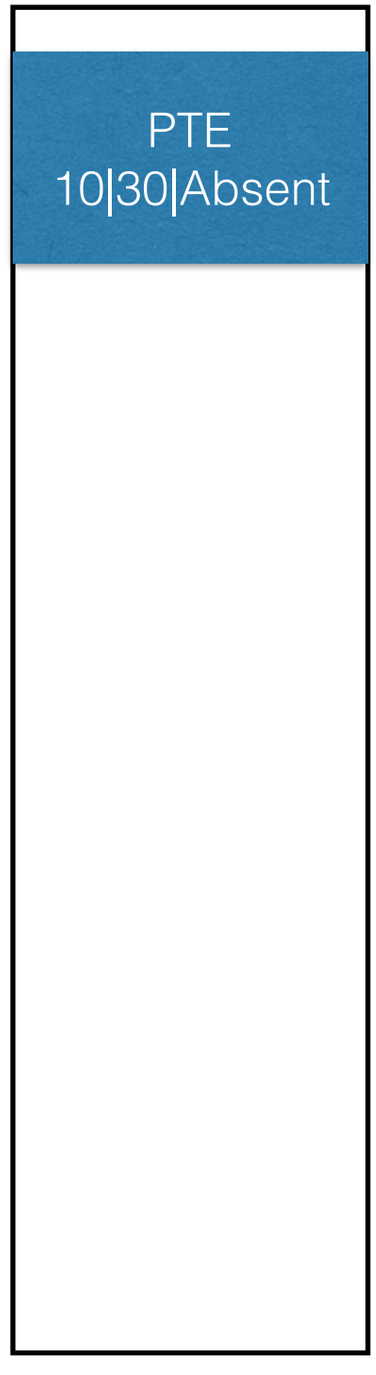
Address space

LOAD VA 10

TLB

VPN	PFN
23	40
40	50
50	30

Physical Memory



Swapping in

Address space

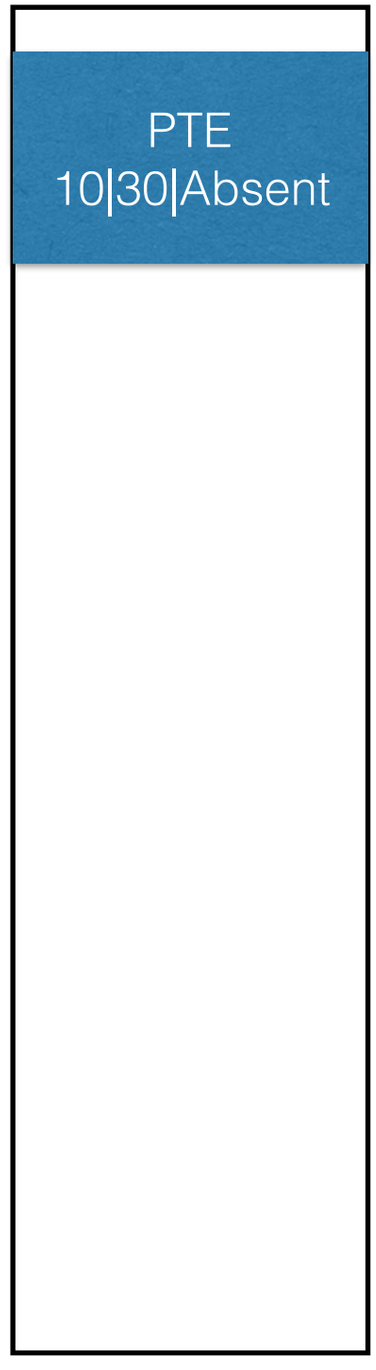
LOAD VA 10

TLB Miss

TLB

VPN	PFN
23	40
40	50
50	30

Physical Memory



Swapping in

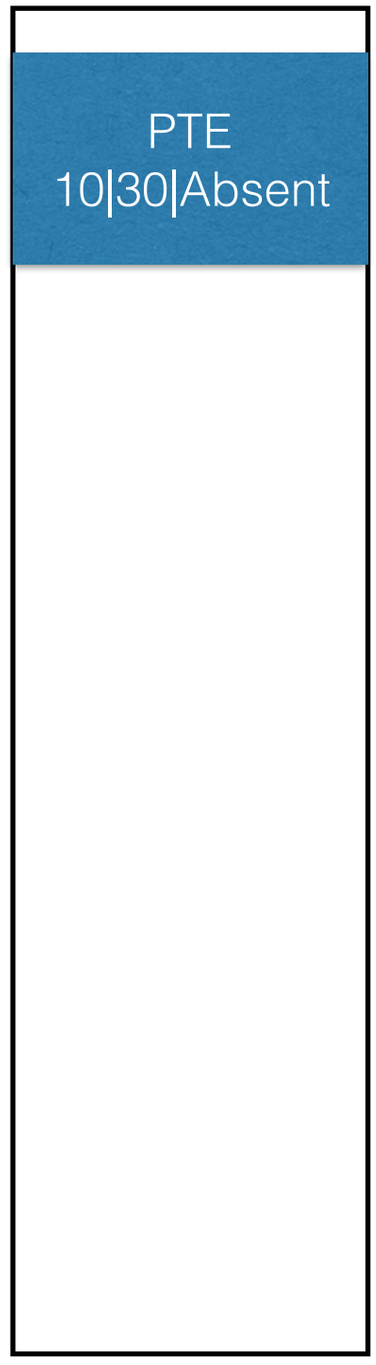
Address space

LOAD VA 10

TLB

VPN	PFN
23	40
40	50
50	30

Physical Memory



Swapping in

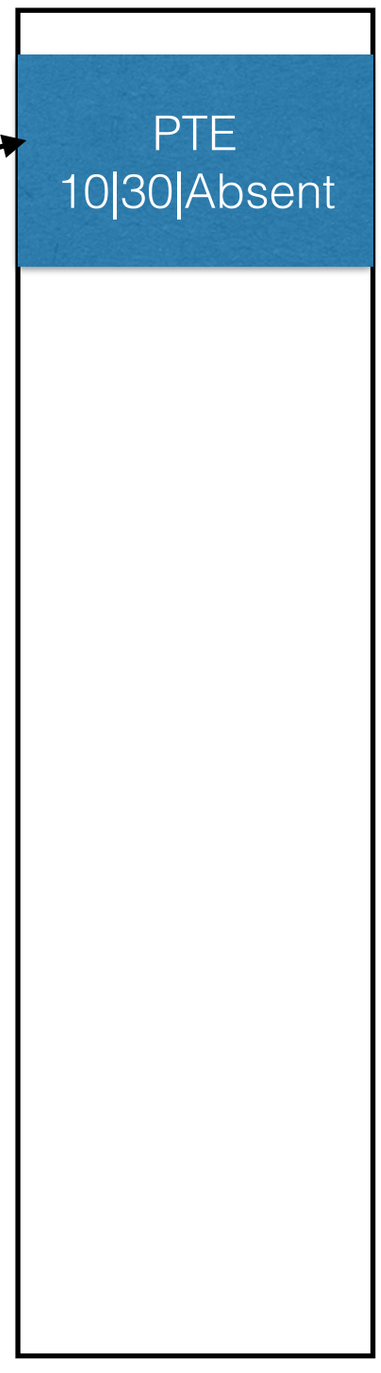
Address space

LOAD VA 10

TLB

VPN	PFN
23	40
40	50
50	30

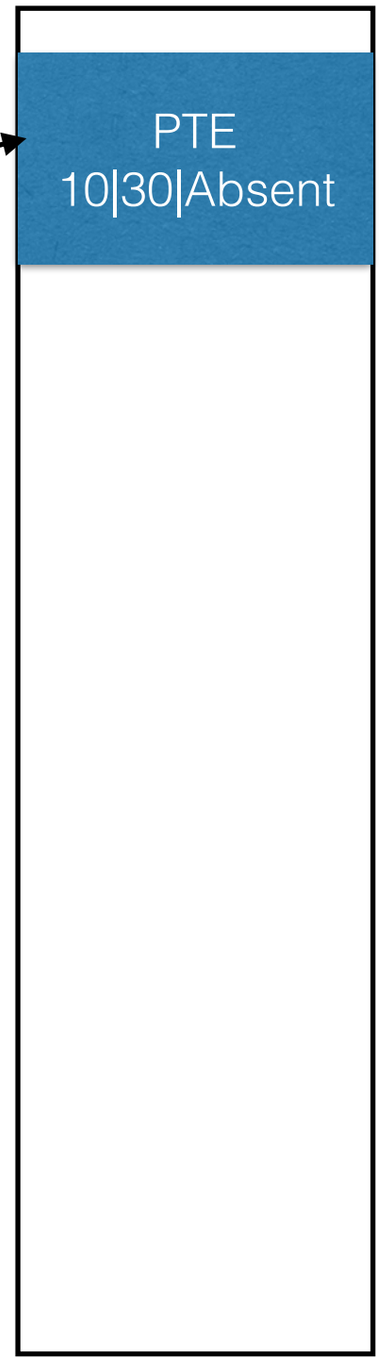
Physical Memory



Swapping in

Address space

Physical Memory



LOAD VA 10

Page fault

TLB

VPN	PFN
23	40
40	50
50	30

Swapping in

Address space

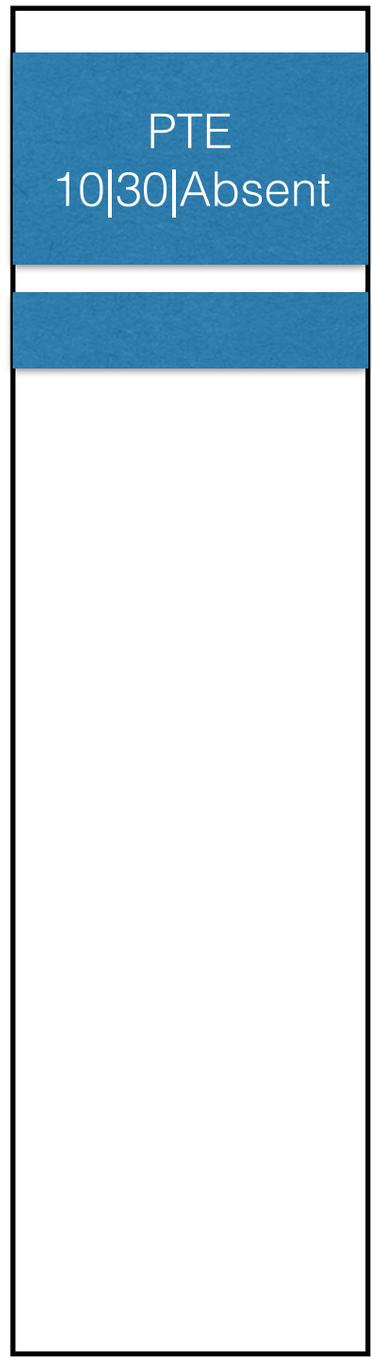
LOAD VA 10

Page fault

TLB

VPN	PFN
23	40
40	50
50	30

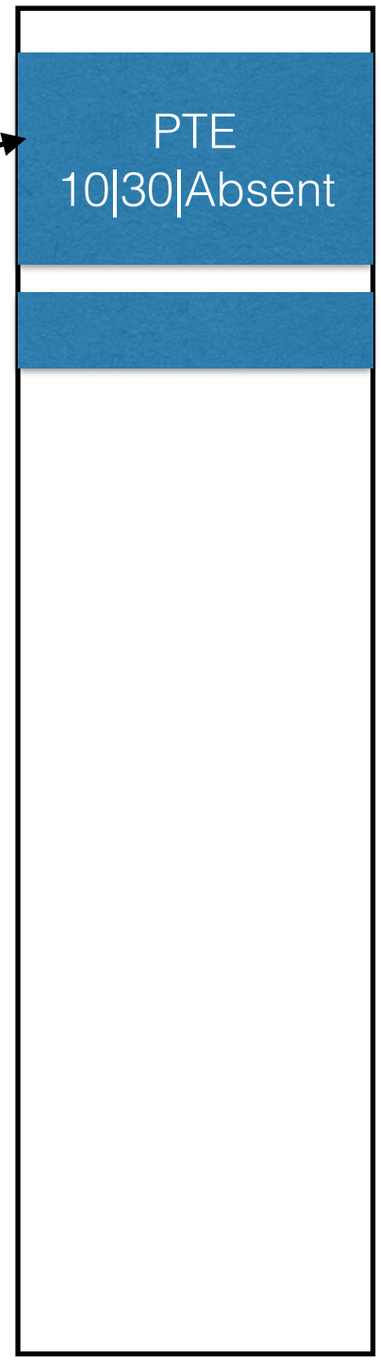
Physical Memory



Swapping in

Address space

Physical Memory



LOAD VA 10

Page fault

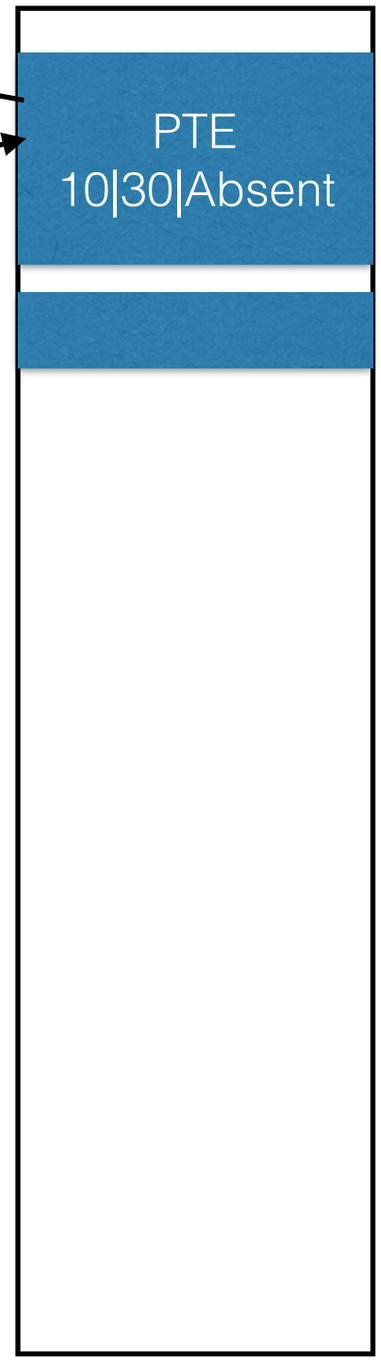
TLB

VPN	PFN
23	40
40	50
50	30

Swapping in

Address space

Physical Memory



LOAD VA 10

Page fault

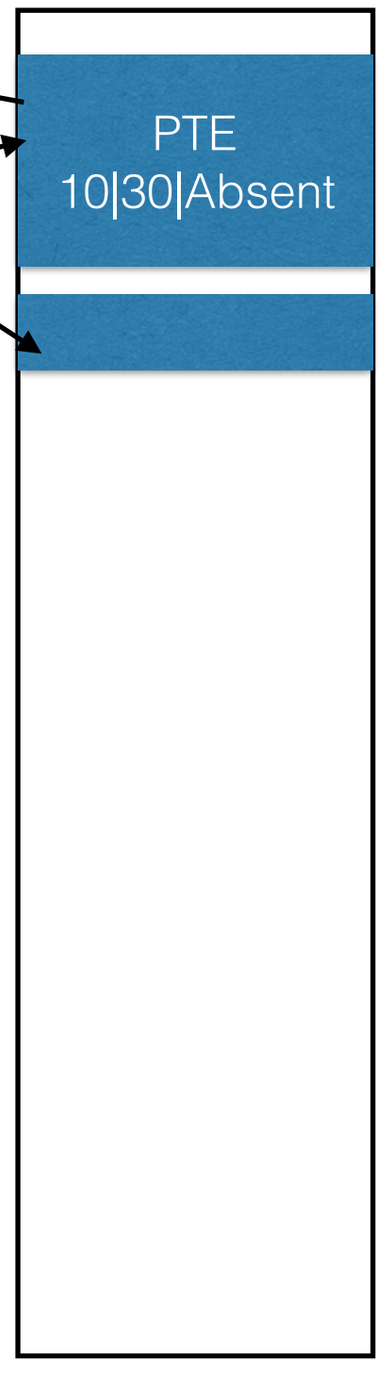
TLB

VPN	PFN
23	40
40	50
50	30

Swapping in

Address space

Physical Memory



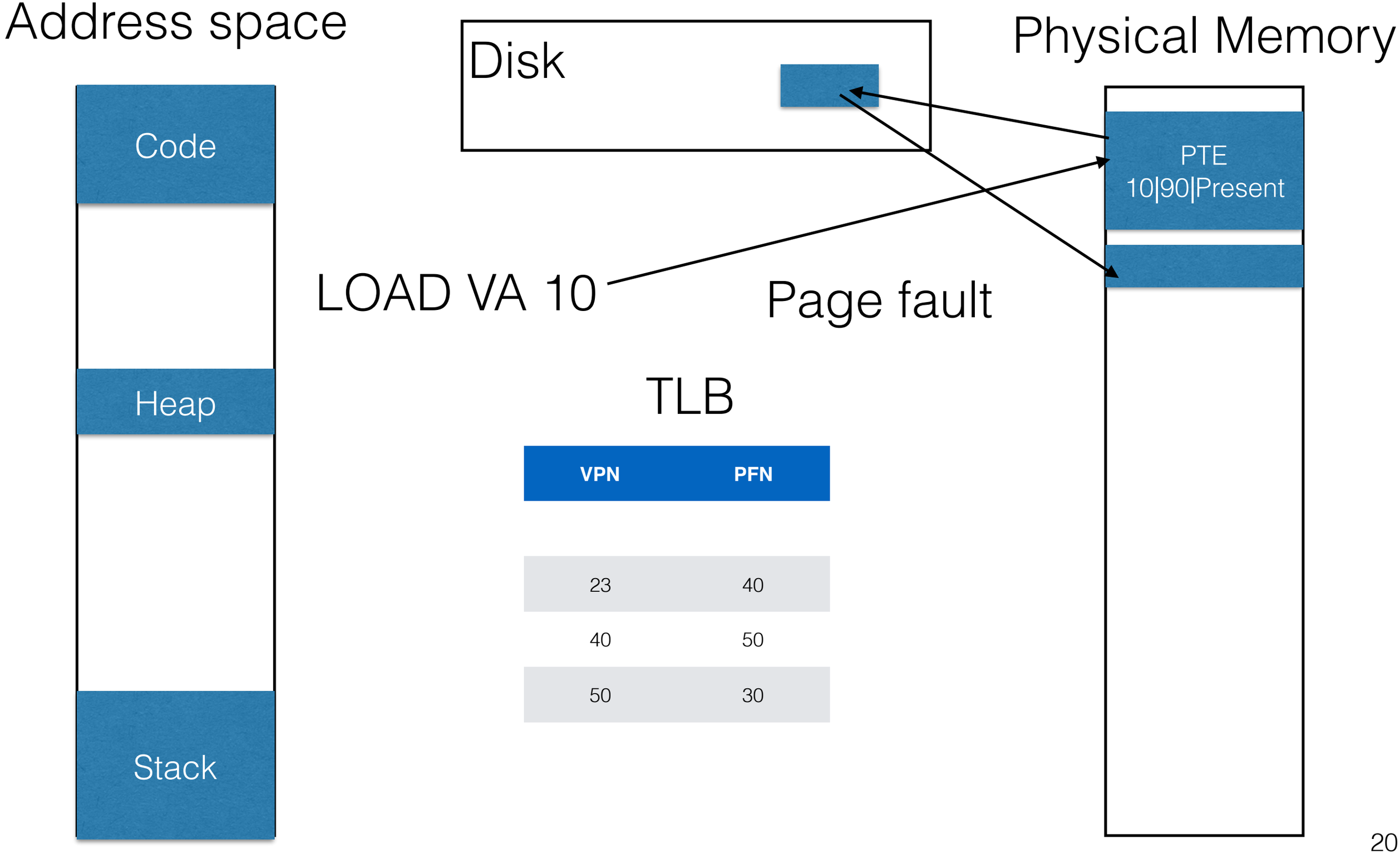
LOAD VA 10

Page fault

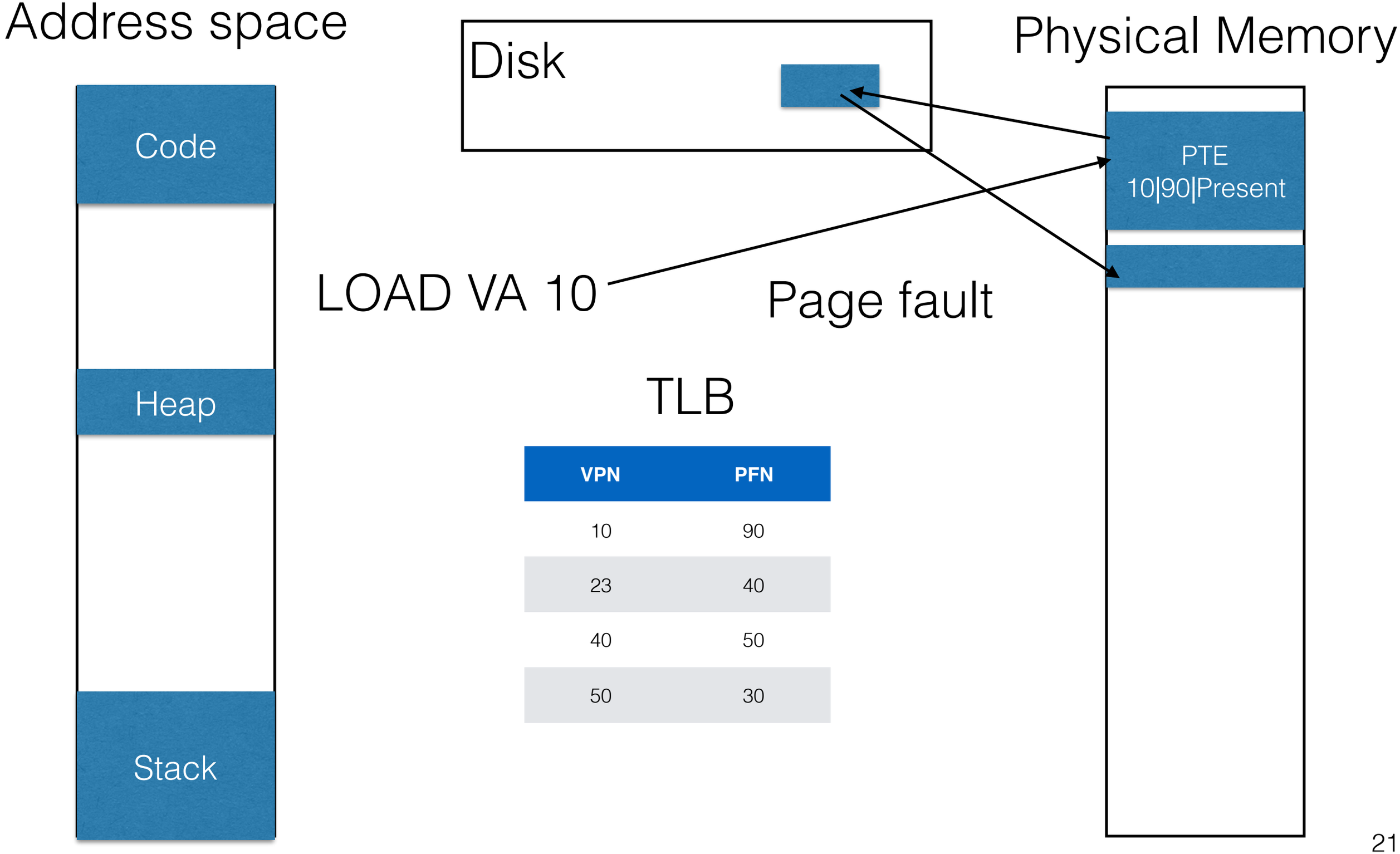
TLB

VPN	PFN
23	40
40	50
50	30

Swapping in


Address space
Code
Heap
Stack
Disk
Physical Memory
PTE
10|90|Present
LOAD VA 10
Page fault
TLB
VPN
PFN
23
40
40
50
50
30

Swapping in



VPN	PFN
10	90
23	40
40	50
50	30

Address Translation

Address Translation

- extract VPN from VA

Address Translation

- extract VPN from VA
- check TLB for VPN

Address Translation

- extract VPN from VA
- check TLB for VPN
- TLB hit:

Address Translation

- extract VPN from VA
- check TLB for VPN
- TLB hit:
  - build PA from PFN and offset

Address Translation

- extract VPN from VA
- check TLB for VPN
- TLB hit:
  - build PA from PFN and offset
  - fetch PA from memory

Address Translation

- extract VPN from VA
- check TLB for VPN
- TLB hit:
  - build PA from PFN and offset
  - fetch PA from memory
- TLB miss:

Address Translation

- extract VPN from VA
- check TLB for VPN
- TLB hit:
  - build PA from PFN and offset
  - fetch PA from memory
- TLB miss:
  - fetch PTE

Address Translation

- extract VPN from VA
- check TLB for VPN
- TLB hit:
  - build PA from PFN and offset
  - fetch PA from memory
- TLB miss:
  - fetch PTE
  - if (!valid): exception [segfault]

Address Translation

- extract VPN from VA
- check TLB for VPN
- TLB hit:
  - build PA from PFN and offset
  - fetch PA from memory
- TLB miss:
  - fetch PTE
  - if (!valid): exception [segfault]
  - else if (!present): exception [page fault, or page miss]

Address Translation

- extract VPN from VA
- check TLB for VPN
- TLB hit:
  - build PA from PFN and offset
  - fetch PA from memory
- TLB miss:
  - fetch PTE
  - if (!valid): exception [segfault]
  - else if (!present): exception [page fault, or page miss]
  - else: extract PFN, insert in TLB, retry

How costly is a Page Fault/Page not found in memory?

How costly is a Page Fault/Page not found in memory?

- Average Memory Access Time (AMAT) =

How costly is a Page Fault/Page not found in memory?

- Average Memory Access Time (AMAT) =
  - Prob. Of hit X Cost of accessing memory +

How costly is a Page Fault/Page not found in memory?

- Average Memory Access Time (AMAT) =
  - Prob. Of hit X Cost of accessing memory +
  - Prob. Of Miss X Cost of accessing memory +

How costly is a Page Fault/Page not found in memory?

- Average Memory Access Time (AMAT) =
  - Prob. Of hit X Cost of accessing memory +
  - Prob. Of Miss X Cost of accessing memory +
  - Prob. Of Miss X Cost of accessing disk =

How costly is a Page Fault/Page not found in memory?

- Average Memory Access Time (AMAT) =
  - Prob. Of hit X Cost of accessing memory +
  - Prob. Of Miss X Cost of accessing memory +
  - Prob. Of Miss X Cost of accessing disk =
  - Cost of accessing memory + Prob. Of Miss X Cost of accessing disk

How costly is a Page Fault/Page not found in memory?

- Average Memory Access Time (AMAT) =
  - Prob. Of hit X Cost of accessing memory +
  - Prob. Of Miss X Cost of accessing memory +
  - Prob. Of Miss X Cost of accessing disk =
  - Cost of accessing memory + Prob. Of Miss X Cost of accessing disk
- Example:

How costly is a Page Fault/Page not found in memory?

- Average Memory Access Time (AMAT) =
  - Prob. Of hit X Cost of accessing memory +
  - Prob. Of Miss X Cost of accessing memory +
  - Prob. Of Miss X Cost of accessing disk =
  - Cost of accessing memory + Prob. Of Miss X Cost of accessing disk
- Example:
  - Prob. Of Miss = 0.1

How costly is a Page Fault/Page not found in memory?

- Average Memory Access Time (AMAT) =
  - Prob. Of hit X Cost of accessing memory +
  - Prob. Of Miss X Cost of accessing memory +
  - Prob. Of Miss X Cost of accessing disk =
  - Cost of accessing memory + Prob. Of Miss X Cost of accessing disk
- Example:
  - Prob. Of Miss = 0.1
  - Memory access = 100 ns

How costly is a Page Fault/Page not found in memory?

- Average Memory Access Time (AMAT) =
  - Prob. Of hit X Cost of accessing memory +
  - Prob. Of Miss X Cost of accessing memory +
  - Prob. Of Miss X Cost of accessing disk =
  - Cost of accessing memory + Prob. Of Miss X Cost of accessing disk
- Example:
  - Prob. Of Miss = 0.1
  - Memory access = 100 ns
  - Disk access = 10 ms

How costly is a Page Fault/Page not found in memory?

- Average Memory Access Time (AMAT) =
  - Prob. Of hit X Cost of accessing memory +
  - Prob. Of Miss X Cost of accessing memory +
  - Prob. Of Miss X Cost of accessing disk =
  - Cost of accessing memory + Prob. Of Miss X Cost of accessing disk
- Example:
  - Prob. Of Miss = 0.1
  - Memory access = 100 ns
  - Disk access = 10 ms
  - AMAT = 100 ns + 1 ms ~ 1ms >> 100 ns

How costly is a Page Fault/Page not found in memory?

- Average Memory Access Time (AMAT) =
  - Prob. Of hit X Cost of accessing memory +
  - Prob. Of Miss X Cost of accessing memory +
  - Prob. Of Miss X Cost of accessing disk =
  - Cost of accessing memory + Prob. Of Miss X Cost of accessing disk
- Example:
  - Prob. Of Miss = 0.1
  - Memory access = 100 ns
  - Disk access = 10 ms
  - AMAT = 100 ns + 1 ms ~ 1ms >> 100 ns
  - AMAT (P (miss) = 0.001) = 100 ns + (0.001)* 10*10^6 ns = 100 ns + 10^4 ns >> 100 ns

Page Replacement Policies - Optimal Replacement

Page Replacement Policies - Optimal Replacement

Oracle!

Page Replacement Policies - Optimal Replacement

Workload (page): 1,2,3,4,1,2,3,4,3,2,1
Cache size/Physical memory size: 3 pages

Page Replacement Policies - Optimal Replacement

Workload (page): 1,2,3,4,1,2,3,4,3,2,1
Cache size/Physical memory size: 3 pages

Optimal strategy: evict pages to be accessed furthest in future
—> Fewest possible cache misses

Page Replacement Policies - Optimal Replacement

Workload (page): 1,2,3,4,1,2,3,4,3,2,1
Cache size/Physical memory size: 3 pages

Access	Hit	State (after)

Page Replacement Policies - Optimal Replacement

Workload (page): 1,2,3,4,1,2,3,4,3,2,1
Cache size/Physical memory size: 3 pages

Access	Hit	State (after)
1	?	1

Page Replacement Policies - Optimal Replacement

Workload (page): 1,2,3,4,1,2,3,4,3,2,1
Cache size/Physical memory size: 3 pages

Access	Hit	State (after)
1	Miss	1
2	Miss	1, 2
3	Miss	1, 2, 3

Cache Full!

Page Replacement Policies - Optimal Replacement

Workload (page): 1,2,3,4,1,2,3,4,3,2,1
Cache size/Physical memory size: 3 pages

Access	Hit	State (after)
1	Miss	1
2	Miss	1, 2
3	Miss	1, 2, 3

Page Replacement Policies - Optimal Replacement

Workload (page): 1,2,3,4,1,2,3,4,3,2,1
Cache size/Physical memory size: 3 pages

Access	Hit	State (after)
1	Miss	1
2	Miss	1, 2
3	Miss	1, 2, 3

Workload rem: 4,1,2,3,4,3,2,1

Page Replacement Policies - Optimal Replacement

Workload (page): 1,2,3,4,1,2,3,4,3,2,1
Cache size/Physical memory size: 3 pages

Access	Hit	State (after)
1	Miss	1
2	Miss	1, 2
3	Miss	1, 2, 3

Workload rem: 4,1,2,3,4,3,2,1

Page Replacement Policies - Optimal Replacement

Workload (page): 1,2,3,4,1,2,3,4,3,2,1
Cache size/Physical memory size: 3 pages

Access	Hit	State (after)
1	Miss	1
2	Miss	1, 2
3	Miss	1, 2, 3

Workload rem: 4,1,2,3,4,3,2,1

Distance (1) = 1

Page Replacement Policies - Optimal Replacement

Workload (page): 1,2,3,4,1,2,3,4,3,2,1
Cache size/Physical memory size: 3 pages

Access	Hit	State (after)
1	Miss	1
2	Miss	1, 2
3	Miss	1, 2, 3

Workload rem: 4,1,2,3,4,3,2,1

Distance (1) = 1
Distance (2) = 2

Page Replacement Policies - Optimal Replacement

Workload (page): 1,2,3,4,1,2,3,4,3,2,1
Cache size/Physical memory size: 3 pages

Access	Hit	State (after)
1	Miss	1
2	Miss	1, 2
3	Miss	1, 2, 3

Workload rem: 4,1,2,3,4,3,2,1

Distance (1) = 1
Distance (2) = 2
Distance (3) = 3

Page Replacement Policies - Optimal Replacement

Workload (page): 1,2,3,4,1,2,3,4,3,2,1
Cache size/Physical memory size: 3 pages

Access	Hit	State (after)
1	Miss	1
2	Miss	1, 2
3	Miss	1, 2, 3

Workload rem: 4,1,2,3,4,3,2,1

Distance (1) = 1
Distance (2) = 2
Distance (3) = 3

Page Replacement Policies - Optimal Replacement

Workload (page): 1,2,3,4,1,2,3,4,3,2,1
Cache size/Physical memory size: 3 pages

Access	Hit	State (after)
1	Miss	1
2	Miss	1, 2
3	Miss	1, 2, 3

Workload rem: 4,1,2,3,4,3,2,1

Distance (1) = 1
Distance (2) = 2
Distance (3) = 3

Evict 3

Page Replacement Policies - Optimal Replacement

Workload (page): 1,2,3,4,1,2,3,4,3,2,1
Cache size/Physical memory size: 3 pages

Access	Hit	State (after)
1	Miss	1
2	Miss	1, 2
3	Miss	1, 2, 3

Workload rem: 4,1,2,3,4,3,2,1

Distance (1) = 1
Distance (2) = 2
Distance (3) = 3

Evict 3

Page Replacement Policies - Optimal Replacement

Workload (page): 1,2,3,4,1,2,3,4,3,2,1
Cache size/Physical memory size: 3 pages

Access	Hit	State (after)
1	Miss	1
2	Miss	1, 2
3	Miss	1, 2, 3
4	Miss	1, 2, 4

Workload rem: 4,1,2,3,4,3,2,1

Distance (1) = 1
Distance (2) = 2
Distance (3) = 3

Evict 3

Page Replacement Policies - Optimal Replacement

Workload (page): 1,2,3,4,1,2,3,4,3,2,1
Cache size/Physical memory size: 3 pages

Access	Hit	State (after)
1	Miss	1
2	Miss	1, 2
3	Miss	1, 2, 3
4	Miss	1, 2, 4
1	Hit	1, 2, 4
2	Hit	1, 2, 4
3	Miss	

Page Replacement Policies - Optimal Replacement

Workload (page): 1,2,3,4,1,2,3,4,3,2,1
Cache size/Physical memory size: 3 pages

Access	Hit	State (after)
1	Miss	1
2	Miss	1, 2
3	Miss	1, 2, 3
4	Miss	1, 2, 4
1	Hit	1, 2, 4
2	Hit	1, 2, 4
3	Miss	

Workload rem: 4, 3, 2, 1

Distance (1) = 4
Distance (2) = 3
Distance (4) = 1

Evict 1

Page Replacement Policies - Optimal Replacement

Workload (page): 1,2,3,4,1,2,3,4,3,2,1
Cache size/Physical memory size: 3 pages

Access	Hit	State (after)
1	Miss	1
2	Miss	1, 2
3	Miss	1, 2, 3
4	Miss	1, 2, 4
1	Hit	1, 2, 4
2	Hit	1, 2, 4
3	Miss	2, 3, 4

Workload rem: 4, 3, 2, 1

Distance (1) = 4
Distance (2) = 3
Distance (4) = 1

Evict 1

Page Replacement Policies - Optimal Replacement

Workload (page): 1,2,3,4,1,2,3,4,3,2,1
Cache size/Physical memory size: 3 pages

Access	Hit	State (after)
1	Miss	1
2	Miss	1, 2
3	Miss	1, 2, 3
4	Miss	1, 2, 4
1	Hit	1, 2, 4
2	Hit	1, 2, 4
3	Miss	2, 3, 4
4	Hit	2, 3, 4
3	Hit	2, 3, 4
2	Hit	2, 3, 4
1	Miss	1, _, _

Page Replacement Policies - Optimal Replacement

Workload (page): 1,2,3,4,1,2,3,4,3,2,1
Cache size/Physical memory size: 3 pages

Access	Hit	State (after)
1	Miss	1
2	Miss	1, 2
3	Miss	1, 2, 3
4	Miss	1, 2, 4
1	Hit	1, 2, 4
2	Hit	1, 2, 4
3	Miss	2, 3, 4
4	Hit	2, 3, 4
3	Hit	2, 3, 4
2	Hit	2, 3, 4
1	Miss	1, _, _

Page Replacement Policies - Optimal Replacement

Workload (page): 1,2,3,4,1,2,3,4,3,2,1
Cache size/Physical memory size: 3 pages

Access	Hit	State (after)
1	Miss	1
2	Miss	1, 2
3	Miss	1, 2, 3
4	Miss	1, 2, 4
1	Hit	1, 2, 4
2	Hit	1, 2, 4
3	Miss	2, 3, 4
4	Hit	2, 3, 4
3	Hit	2, 3, 4
2	Hit	2, 3, 4
1	Miss	1, _, _

Hit rate (%) = 5*100%/11
~45%

Page Replacement Policies - Optimal Replacement

Workload (page): 1,2,3,4,1,2,3,4,3,2,1
Cache size/Physical memory size: 3 pages

Access	Hit	State (after)
1	Miss	1
2	Miss	1, 2
3	Miss	1, 2, 3
4	Miss	1, 2, 4
1	Hit	1, 2, 4
2	Hit	1, 2, 4
3	Miss	2, 3, 4
4	Hit	2, 3, 4
3	Hit	2, 3, 4
2	Hit	2, 3, 4
1	Miss	1, _, _

Hit rate (%) = 5*100%/11
~45%

Compulsory misses:
First miss for a page
= 4 compulsory misses

Page Replacement Policies - Optimal Replacement

Workload (page): 1,2,3,4,1,2,3,4,3,2,1
Cache size/Physical memory size: 3 pages

Access	Hit	State (after)
1	Miss	1
2	Miss	1, 2
3	Miss	1, 2, 3
4	Miss	1, 2, 4
1	Hit	1, 2, 4
2	Hit	1, 2, 4
3	Miss	2, 3, 4
4	Hit	2, 3, 4
3	Hit	2, 3, 4
2	Hit	2, 3, 4
1	Miss	1, _, _

Hit rate (%) = 5*100%/11
~45%

Compulsory misses:
First miss for a page
= 4 compulsory misses

Hit rate (discounting compulsory misses) = 5/(11-4)~71%

Page Replacement Policies - FIFO

Workload (page): 1,2,3,4,1,2,3,4,3,2,1
Cache size/Physical memory size: 4 pages

Page Replacement Policies - FIFO

Workload (page): 1,2,3,4,1,2,3,4,3,2,1
Cache size/Physical memory size: 4 pages

Strategy: evict pages based on FIFO

Page Replacement Policies - FIFO

Workload (page): 1,2,3,4,1,2,3,4,3,2,1
Cache size/Physical memory size: 3 pages

Access	Hit	State (first entry is rightmost)
1	Miss	1
2	Miss	2, 1
3	Miss	3, 2, 1

Page Replacement Policies - FIFO

Workload (page): 1,2,3,4,1,2,3,4,3,2,1
Cache size/Physical memory size: 3 pages

Access	Hit	State (first entry is rightmost)
1	Miss	1
2	Miss	2, 1
3	Miss	3, 2, 1
4	Miss	4, 3, 2

Page Replacement Policies -FIFO

Workload (page): 1,2,3,4,1,2,3,4,3,2,1
Cache size/Physical memory size: 3 pages

Access	Hit	State
1	Miss	1
2	Miss	2, 1
3	Miss	3, 2, 1
4	Miss	4, 3, 2
1	Miss	1, 4, 3
2	Miss	2, 1, 4
3	Miss	3, 2, 1
4	Miss	4, 3, 2
3	Hit	4, 3, 2
2	Hit	4, 3, 2
1	Miss	1, 4, 3

Page Replacement Policies -FIFO

Workload (page): 1,2,3,4,1,2,3,4,3,2,1
Cache size/Physical memory size: 3 pages

Access	Hit	State
1	Miss	1
2	Miss	2, 1
3	Miss	3, 2, 1
4	Miss	4, 3, 2
1	Miss	1, 4, 3
2	Miss	2, 1, 4
3	Miss	3, 2, 1
4	Miss	4, 3, 2
3	Hit	4, 3, 2
2	Hit	4, 3, 2
1	Miss	1, 4, 3

Hit rate (%) = 2*100%/11
~18%

Page Replacement Policies -FIFO

Workload (page): 1,2,3,4,1,2,3,4,3,2,1
Cache size/Physical memory size: 3 pages

Access	Hit	State
1	Miss	1
2	Miss	2, 1
3	Miss	3, 2, 1
4	Miss	4, 3, 2
1	Miss	1, 4, 3
2	Miss	2, 1, 4
3	Miss	3, 2, 1
4	Miss	4, 3, 2
3	Hit	4, 3, 2
2	Hit	4, 3, 2
1	Miss	1, 4, 3

Hit rate (%) = 2*100%/11
~18%

Compulsory misses:
First miss for a page
= 4 compulsory misses

Page Replacement Policies -FIFO

Workload (page): 1,2,3,4,1,2,3,4,3,2,1
Cache size/Physical memory size: 3 pages

Access	Hit	State
1	Miss	1
2	Miss	2, 1
3	Miss	3, 2, 1
4	Miss	4, 3, 2
1	Miss	1, 4, 3
2	Miss	2, 1, 4
3	Miss	3, 2, 1
4	Miss	4, 3, 2
3	Hit	4, 3, 2
2	Hit	4, 3, 2
1	Miss	1, 4, 3

Hit rate (%) = 2*100%/11
~18%

Compulsory misses: First miss for a page = 4 compulsory misses

Hit rate (discounting compulsory misses) = 2/(11-4)~28%

Page Replacement Policies -FIFO

Workload (page): 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5
Cache size/Physical memory size: 3 pages/4 pages

Cache size: 3

Cache size: 4

Page Replacement Policies -FIFO

Workload (page): 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5
Cache size/Physical memory size: 3 pages/4 pages

Cache size: 3

Acces	Hit	State
1	Miss	1
2	Miss	2, 1
3	Miss	3, 2, 1
4	Miss	4, 3, 2
1	Miss	1, 4, 3
2	Miss	2, 1, 4
5	Miss	5, 2, 1
1	**Hit**	5, 2, 1
2	**Hit**	5, 2, 1
3	Miss	3, 5, 2
4	Miss	4, 3, 5
5	**Hit**	4, 3, 5

Cache size: 4

Page Replacement Policies -FIFO

Workload (page): 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5
Cache size/Physical memory size: 3 pages/4 pages

Cache size: 3

Acces	Hit	State
1	Miss	1
2	Miss	2, 1
3	Miss	3, 2, 1
4	Miss	4, 3, 2
1	Miss	1, 4, 3
2	Miss	2, 1, 4
5	Miss	5, 2, 1
1	**Hit**	5, 2, 1
2	**Hit**	5, 2, 1
3	Miss	3, 5, 2
4	Miss	4, 3, 5
5	**Hit**	4, 3, 5

Cache size: 4

Acces	Hit	State
1	Miss	1
2	Miss	2, 1
3	Miss	3, 2, 1
4	Miss	4, 3, 2, 1
1	**Hit**	4, 3, 2, 1
2	**Hit**	4, 3, 2, 1
5	Miss	5, 4, 3, 2
1	Miss	1, 5, 4, 2
2	Miss	2, 1, 5, 4
3	Miss	3, 2, 1, 5
4	Miss	4, 3, 2, 1
5	Miss	5, 4, 3, 2

Page Replacement Policies -FIFO

Belady's anomaly

Workload (page): 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

Cache size/Physical memory size: 3 pages/4 pages

Cache size: 3

Acces	Hit	State
1	Miss	1
2	Miss	2, 1
3	Miss	3, 2, 1
4	Miss	4, 3, 2
1	Miss	1, 4, 3
2	Miss	2, 1, 4
5	Miss	5, 2, 1
1	**Hit**	5, 2, 1
2	**Hit**	5, 2, 1
3	Miss	3, 5, 2
4	Miss	4, 3, 5
5	**Hit**	4, 3, 5

Cache size: 4

Acces	Hit	State
1	Miss	1
2	Miss	2, 1
3	Miss	3, 2, 1
4	Miss	4, 3, 2, 1
1	**Hit**	4, 3, 2, 1
2	**Hit**	4, 3, 2, 1
5	Miss	5, 4, 3, 2
1	Miss	1, 5, 4, 2
2	Miss	2, 1, 5, 4
3	Miss	3, 2, 1, 5
4	Miss	4, 3, 2, 1
5	Miss	5, 4, 3, 2

Page Replacement Policies - Random

Workload (page): 1,2,3,4,1,2,3,4,3,2,1
Cache size/Physical memory size: 4 pages

Page Replacement Policies - Random

Workload (page): 1,2,3,4,1,2,3,4,3,2,1
Cache size/Physical memory size: 4 pages

Random strategy: randomly evict pages

Page Replacement Policies -Random

Workload (page): 1,2,3,4,1,2,3,4,3,2,1
Cache size/Physical memory size: 3 pages

Page Replacement Policies -Random

Workload (page): 1,2,3,4,1,2,3,4,3,2,1
Cache size/Physical memory size: 3 pages

Hit rate (%) =?

Page Replacement Policies -Random

Workload (page): 1,2,3,4,1,2,3,4,3,2,1
Cache size/Physical memory size: 3 pages

Hit rate (%) =?

code/replacement-random.py
code/replacement-random-plot.py

Page Replacement Policies -History based (LRU and LFU)

Workload (page): 1,2,3,4,1,2,3,4,3,2,1
Cache size/Physical memory size: 3 pages

Page Replacement Policies -History based (LRU and LFU)

Workload (page): 1,2,3,4,1,2,3,4,3,2,1
Cache size/Physical memory size: 3 pages

Strategy:

1. Least frequently used - evict least frequently used page
2. Least recently used - evict least recently used page

Page Replacement Policies - LRU

Workload (page): 0, 1, 2, 0, 1, 3, 0, 3, 1, 2, 1
Cache size/Physical memory size: 3 pages

Access	Hit	State (least used entry is rightmost)
0	Miss	0
1	Miss	1, 0
2	Miss	2, 1, 0
0		
1		
3		
0		
3		
1		
2		
1		

Page Replacement Policies - LRU

Workload (page): 0, 1, 2, 0, 1, 3, 0, 3, 1, 2, 1
Cache size/Physical memory size: 3 pages

Access	Hit	State (least used entry is rightmost)
0	Miss	0
1	Miss	1, 0
2	Miss	2, 1, 0
0	Hit	0, 2, 1
1		
3		
0		
3		
1		
2		
1		

Page Replacement Policies - LRU

Workload (page): 0, 1, 2, 0, 1, 3, 0, 3, 1, 2, 1
Cache size/Physical memory size: 3 pages

Access	Hit	State (least used entry is rightmost)
0	Miss	0
1	Miss	1, 0
2	Miss	2, 1, 0
0	Hit	0, 2, 1
1	Hit	1, 0, 2
3		
0		
3		
1		
2		
1		

Page Replacement Policies - LRU

Workload (page): 0, 1, 2, 0, 1, 3, 0, 3, 1, 2, 1
Cache size/Physical memory size: 3 pages

Access	Hit	State (least used entry is rightmost)
0	Miss	0
1	Miss	1, 0
2	Miss	2, 1, 0
0	Hit	0, 2, 1
1	Hit	1, 0, 2
3	Miss	3, 1, 0
0		
3		
1		
2		
1		

Page Replacement Policies - LRU

Workload (page): 0, 1, 2, 0, 1, 3, 0, 3, 1, 2, 1
Cache size/Physical memory size: 3 pages

Access	Hit	State (least used entry is rightmost)
0	Miss	0
1	Miss	1, 0
2	Miss	2, 1, 0
0	Hit	0, 2, 1
1	Hit	1, 0, 2
3	Miss	3, 1, 0
0	Hit	0, 3, 1
3		
1		
2		
1		

Page Replacement Policies - LRU

Workload (page): 0, 1, 2, 0, 1, 3, 0, 3, 1, 2, 1
Cache size/Physical memory size: 3 pages

Access	Hit	State (least used entry is rightmost)
0	Miss	0
1	Miss	1, 0
2	Miss	2, 1, 0
0	Hit	0, 2, 1
1	Hit	1, 0, 2
3	Miss	3, 1, 0
0	Hit	0, 3, 1
3	Hit	3, 0, 1
1		
2		
1		

Page Replacement Policies - LRU

Workload (page): 0, 1, 2, 0, 1, 3, 0, 3, 1, 2, 1
Cache size/Physical memory size: 3 pages

Access	Hit	State (least used entry is rightmost)
0	Miss	0
1	Miss	1, 0
2	Miss	2, 1, 0
0	Hit	0, 2, 1
1	Hit	1, 0, 2
3	Miss	3, 1, 0
0	Hit	0, 3, 1
3	Hit	3, 0, 1
1	Hit	1, 3, 0
2		
1		

Page Replacement Policies - LRU

Workload (page): 0, 1, 2, 0, 1, 3, 0, 3, 1, 2, 1
Cache size/Physical memory size: 3 pages

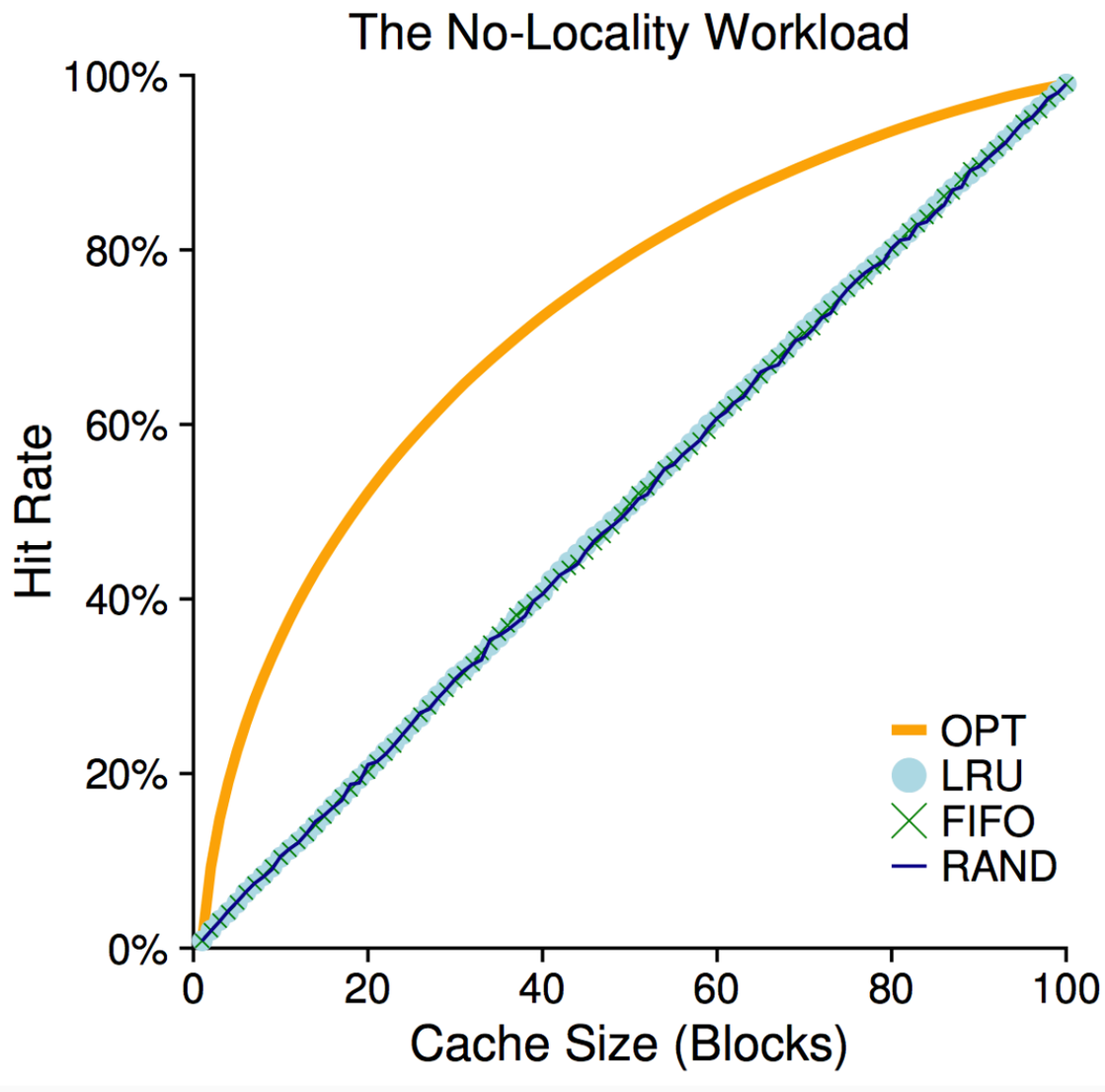
Access	Hit	State (least used entry is rightmost)
0	Miss	0
1	Miss	1, 0
2	Miss	2, 1, 0
0	Hit	0, 2, 1
1	Hit	1, 0, 2
3	Miss	3, 1, 0
0	Hit	0, 3, 1
3	Hit	3, 0, 1
1	Hit	1, 3, 0
2	Miss	2, 1, 3
1	Hit	1, 2, 3

Page Replacement Policies - MRU

Workload (page): 0, 1, 2, 0, 1, 3, 0, 3, 1, 2, 1
Cache size/Physical memory size: 3 pages

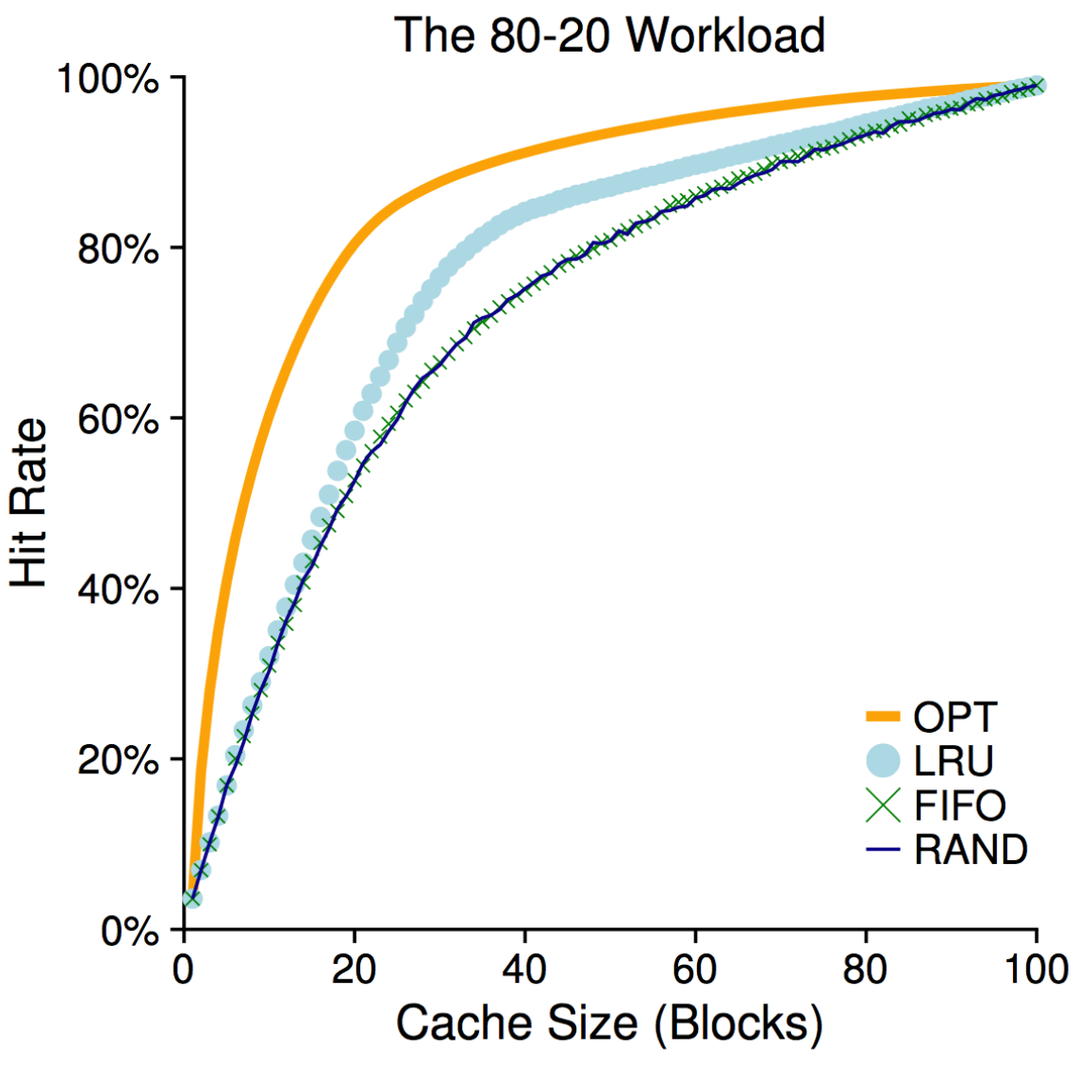
Workload Examples

100 unique pages
10000 page accesses
Randomly chosen page order



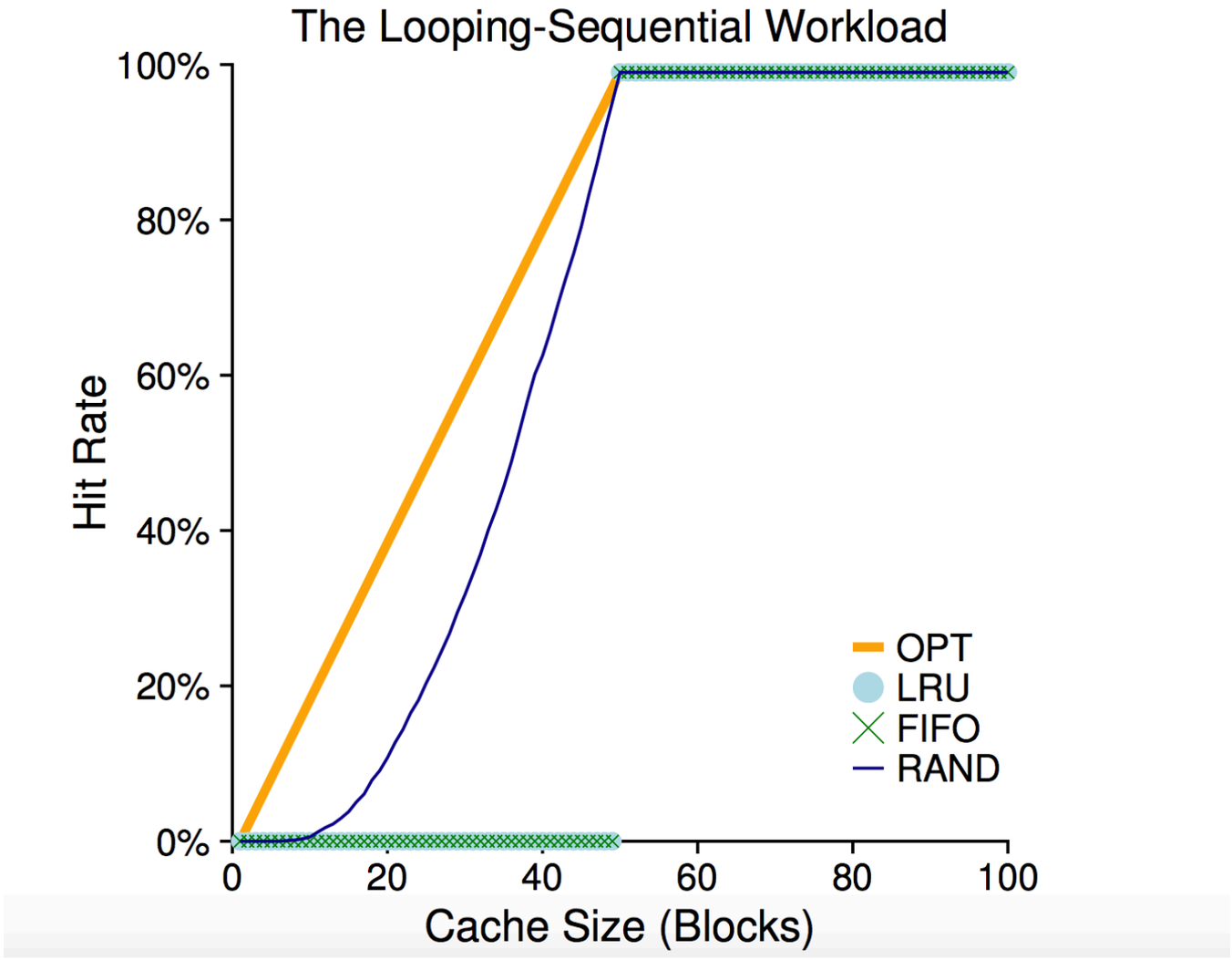
Workload Examples

100 unique pages
10000 page accesses
80/20 load: 80% of accesses to 20% pages



Workload Examples

50 unique pages
10000 page accesses
Accesses in order and looped: 0, 1, 2, …49, 0, 1, ….



LRU implementation

LRU implementation

- On each access, update time of page

LRU implementation

- On each access, update time of page
- When looking for eviction:

LRU implementation

- On each access, update time of page
- When looking for eviction:
  - Search for all candidate sets (millions of pages)

LRU implementation

- On each access, update time of page
- When looking for eviction:
  - Search for all candidate sets (millions of pages)
  - Find least recently used

LRU implementation

- On each access, update time of page
- When looking for eviction:
  - Search for all candidate sets (millions of pages)
  - Find least recently used
- Huge overhead!

LRU implementation (Appx - Clock Hand Algo)

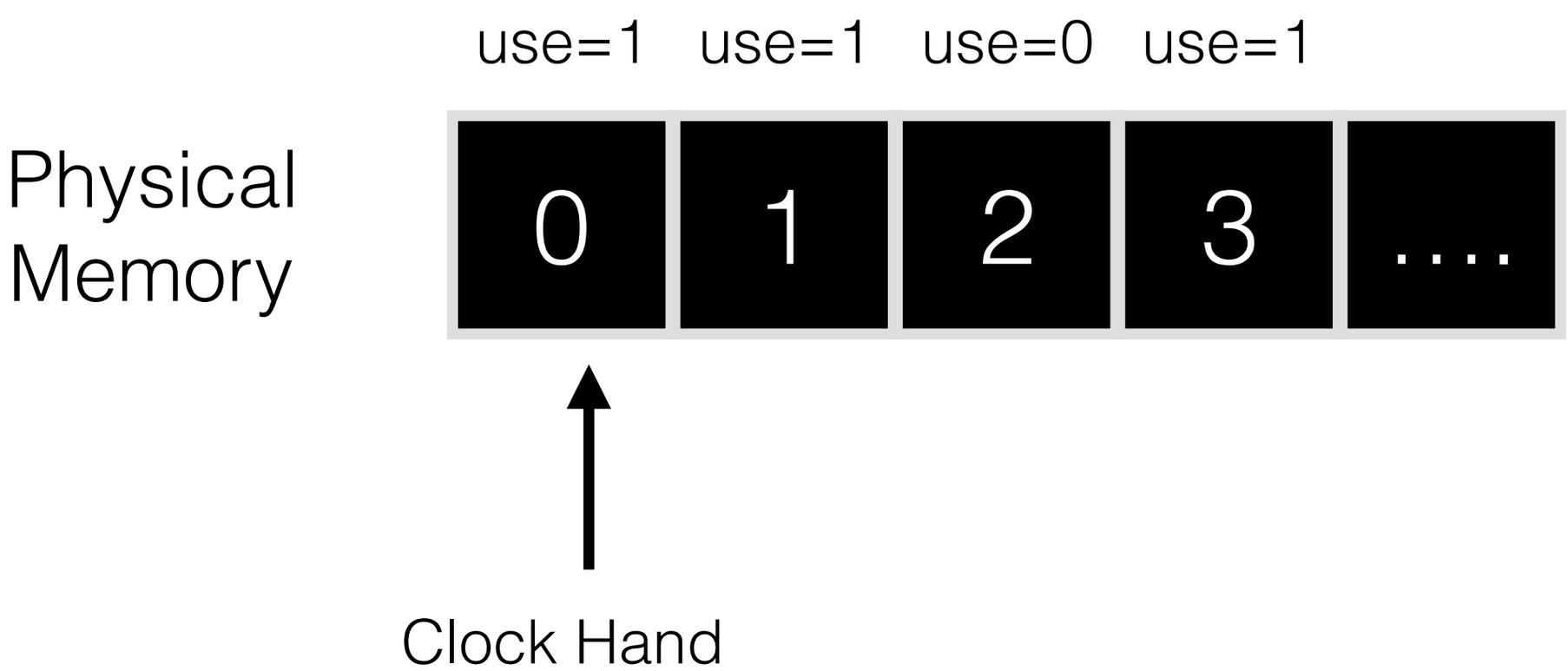
LRU implementation (Appx - Clock Hand Algo)

- On each access, set reference bit for page

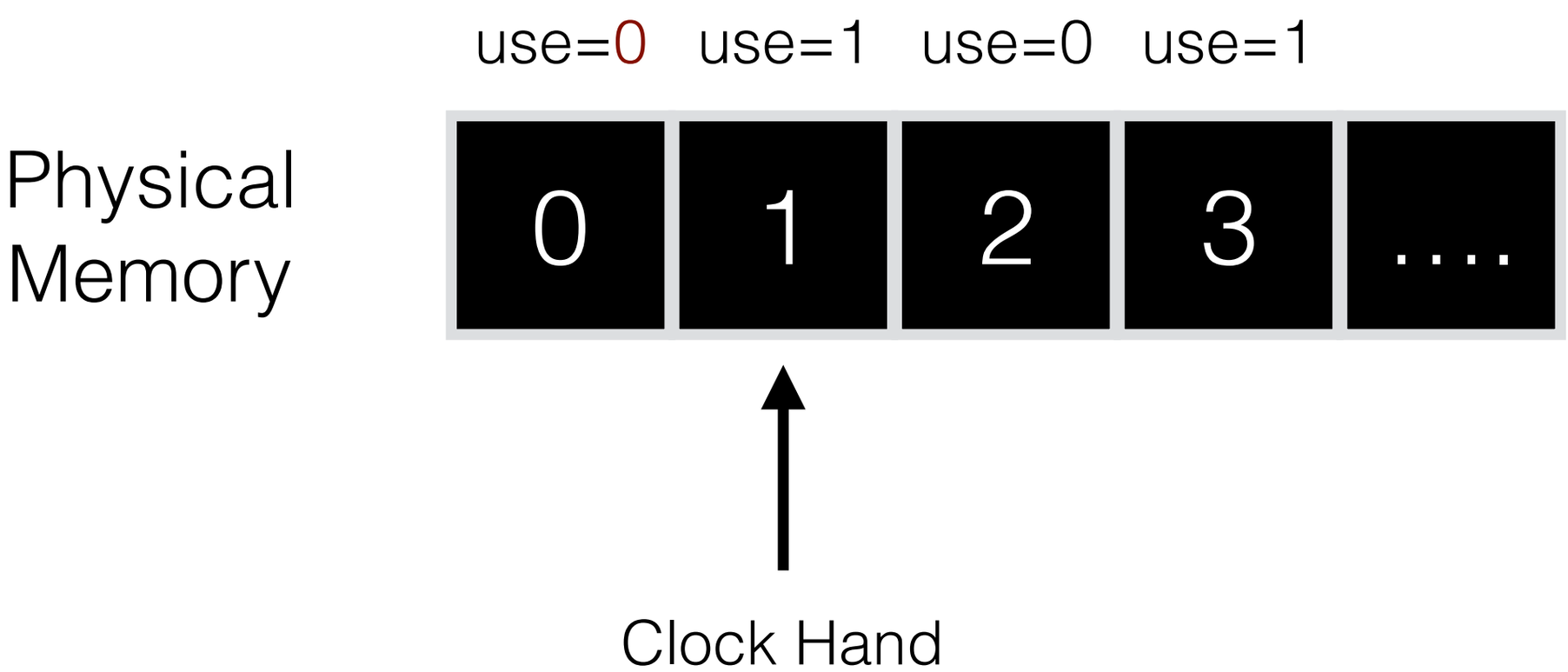
LRU implementation (Appx - Clock Hand Algo)

- On each access, set reference bit for page
- Clock algorithm - look for nearest page without set reference bit

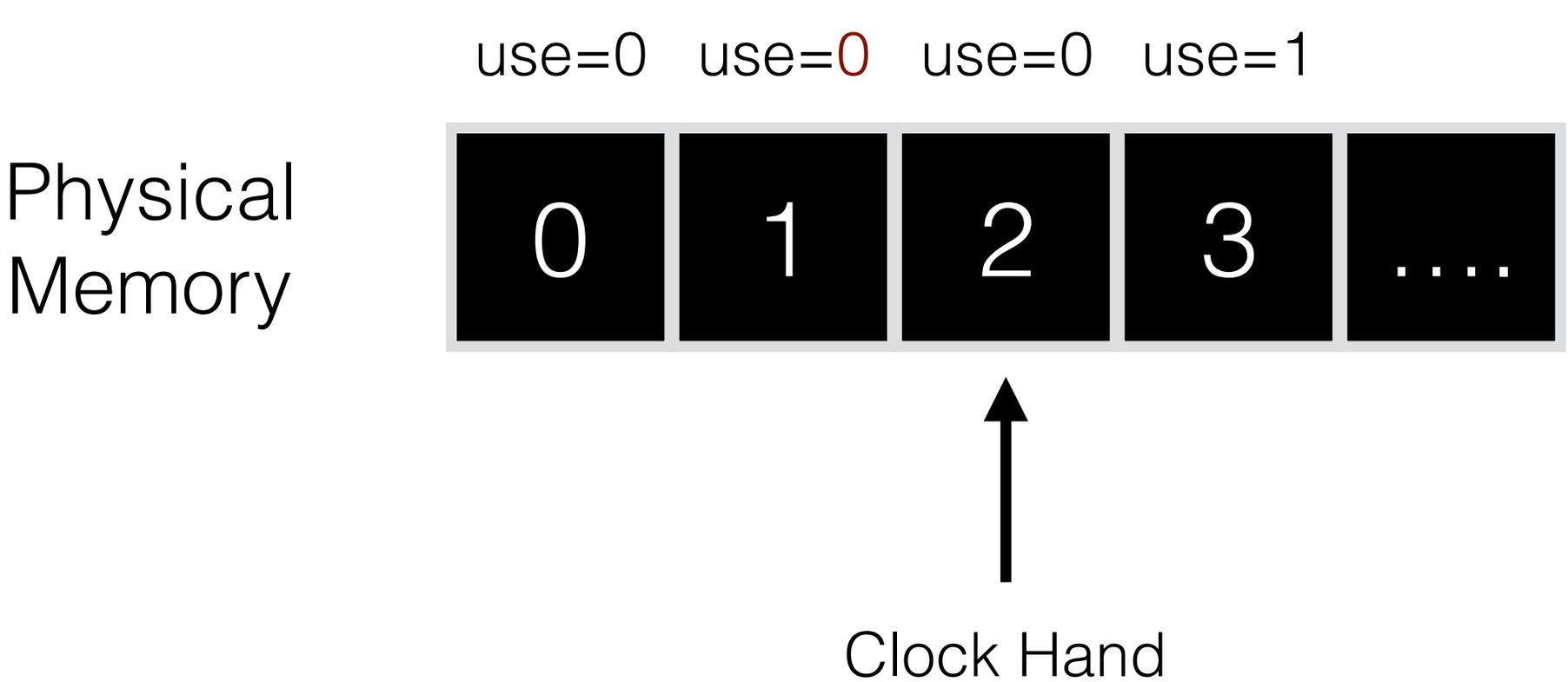
LRU implementation (Appx - Clock Hand Algo)



LRU implementation (Appx - Clock Hand Algo)

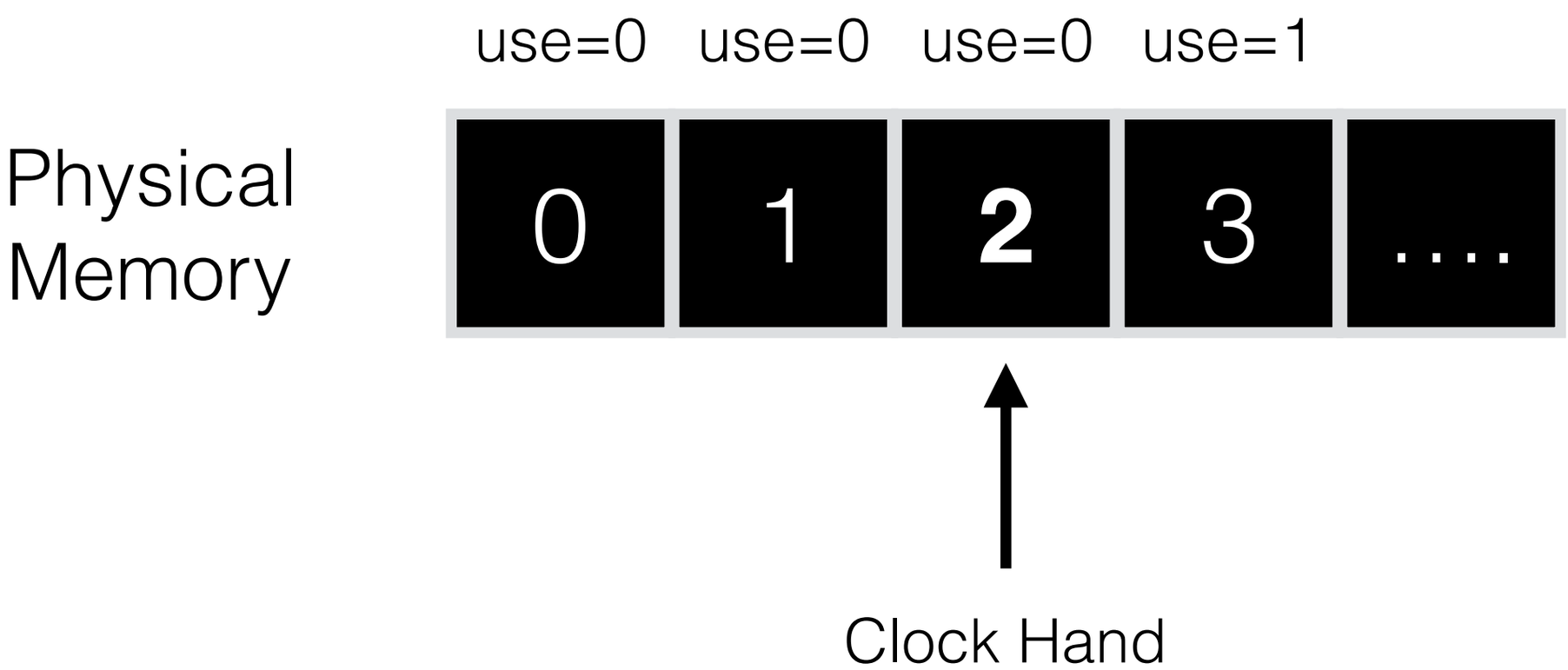


LRU implementation (Appx - Clock Hand Algo)



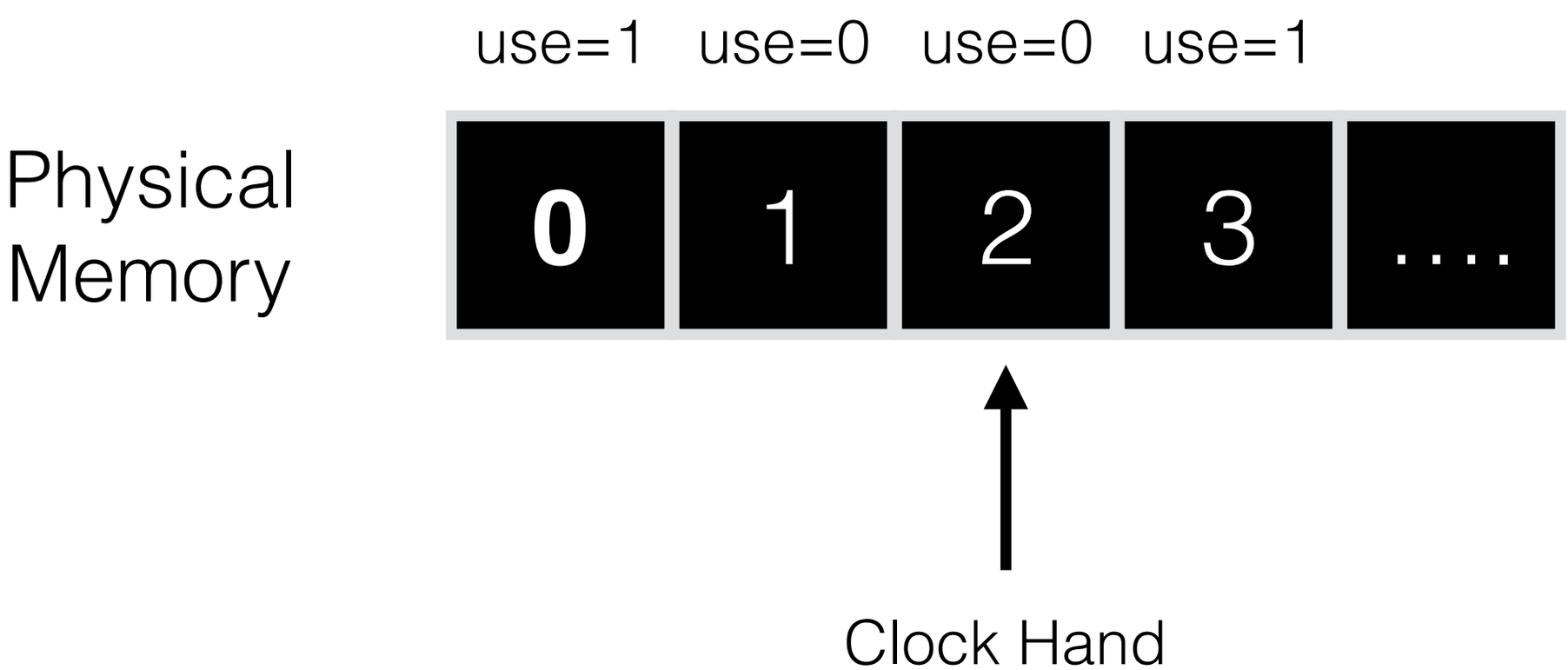
LRU implementation (Appx - Clock Hand Algo)

Evict Page 2: Not recently used

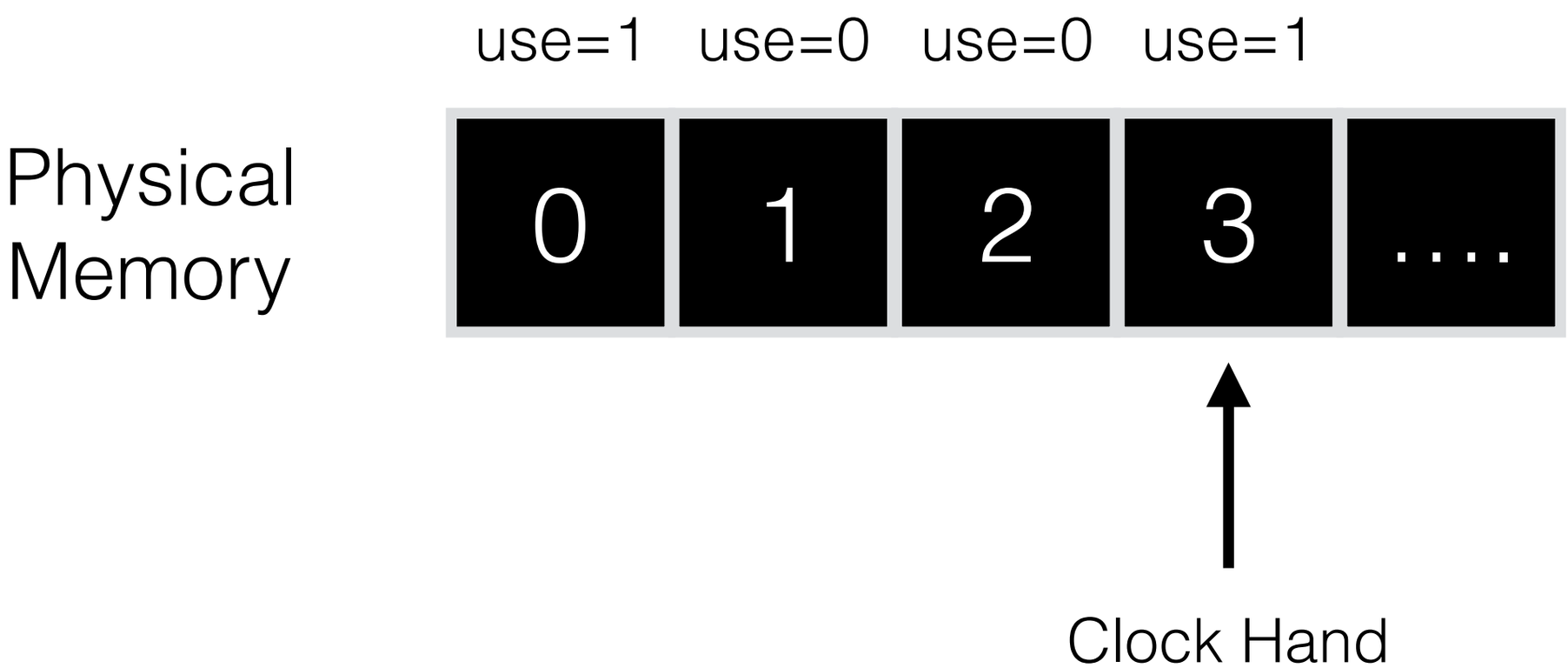


LRU implementation (Appx - Clock Hand Algo)

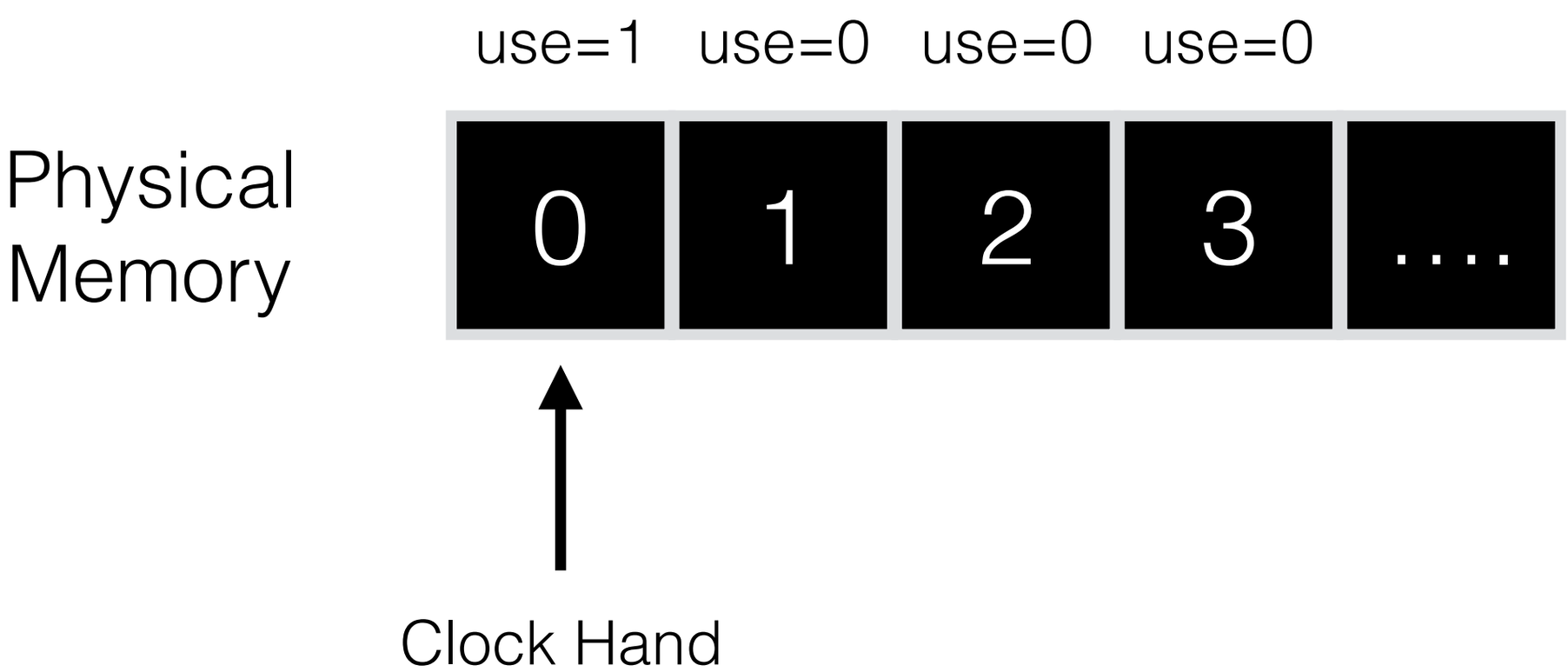
Page 0 is accessed



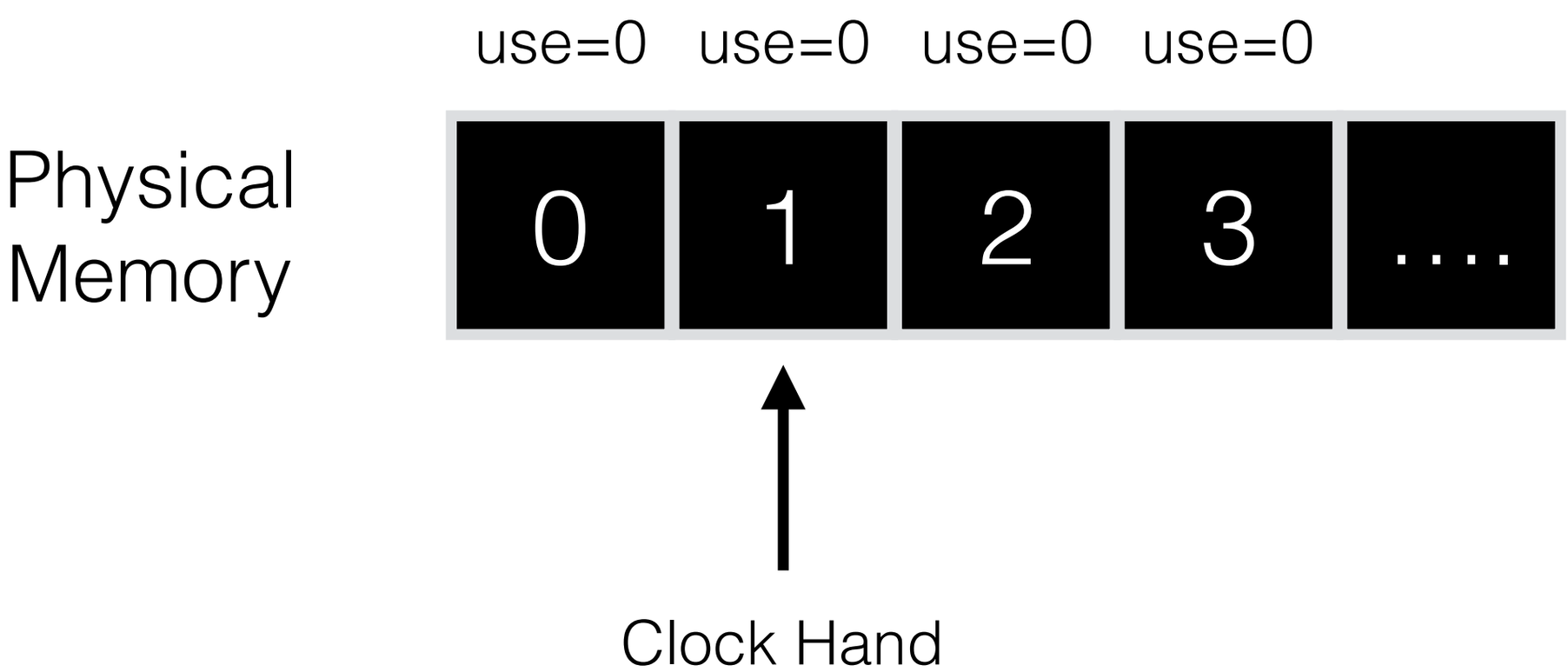
LRU implementation (Appx - Clock Hand Algo)



LRU implementation (Appx - Clock Hand Algo)

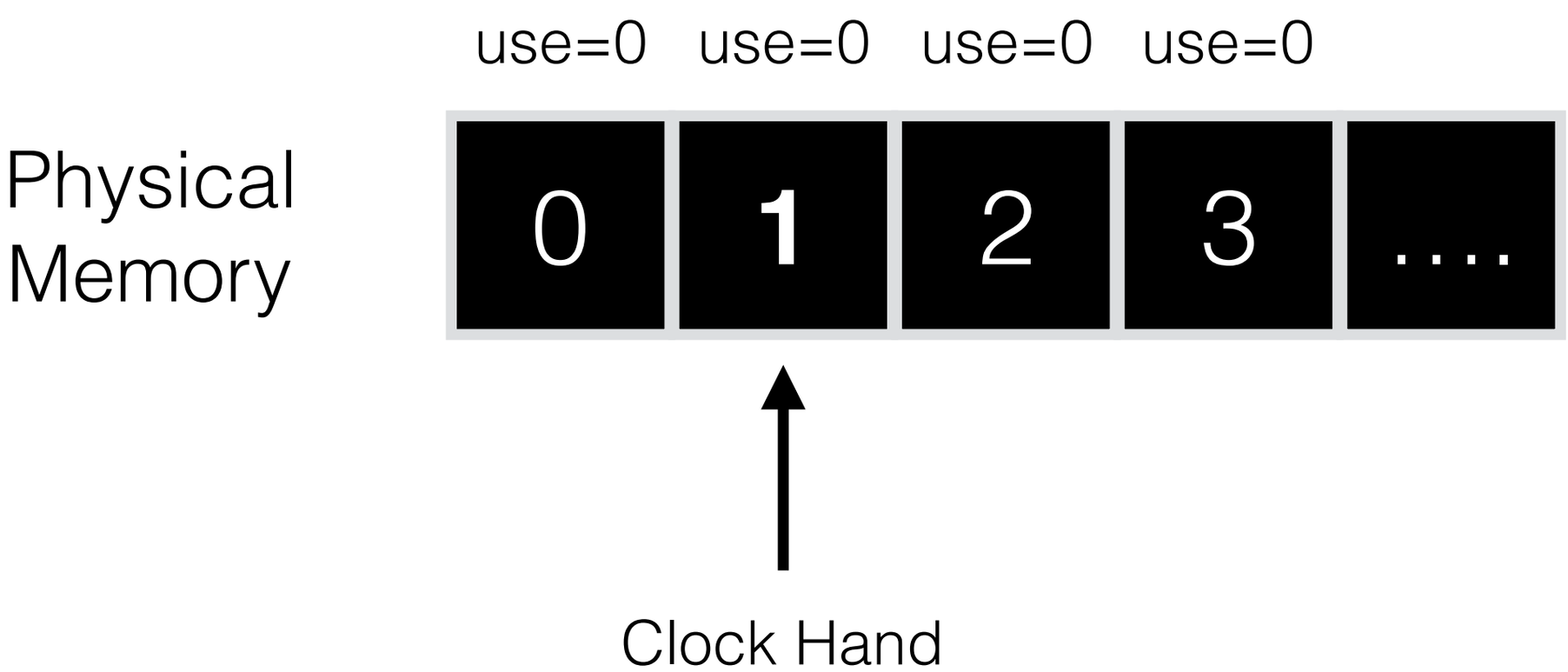


LRU implementation (Appx - Clock Hand Algo)



LRU implementation (Appx - Clock Hand Algo)

Evict Page 1: Not recently used



Other Factors

Other Factors

- Assume page is both on disk and RAM

Other Factors

- Assume page is both on disk and RAM
- Do we have to write the evicted page to disk?

Other Factors

- Assume page is both on disk and RAM
- Do we have to write the evicted page to disk?
  - If page is clean?

Other Factors

- Assume page is both on disk and RAM
- Do we have to write the evicted page to disk?
  - If page is clean?
    - NO!

Other Factors

- Assume page is both on disk and RAM
- Do we have to write the evicted page to disk?
  - If page is clean?
    - NO!
  - If page is dirty?

Other Factors

- Assume page is both on disk and RAM
- Do we have to write the evicted page to disk?
  - If page is clean?
    - NO!
  - If page is dirty?
    - Yes!

Other Factors

Other Factors

- When to swap in?

Other Factors

- When to swap in?
  - Demand paging: swap in when needed

Other Factors

- When to swap in?
  - Demand paging: swap in when needed
  - Prefetching: swap in a page ahead of demand (anticipating demand)

Other Factors

- When to swap in?
  - Demand paging: swap in when needed
  - Prefetching: swap in a page ahead of demand (anticipating demand)
  - When likely?

Other Factors

- When to swap in?
  - Demand paging: swap in when needed
  - Prefetching: swap in a page ahead of demand (anticipating demand)
- When likely?
  - Code page P brought to memory, P+1 also likely

Other Factors

- When to swap in?
  - Demand paging: swap in when needed
  - Prefetching: swap in a page ahead of demand (anticipating demand)
    - When likely?
      - Code page P brought to memory, P+1 also likely
- When to write to disk

Other Factors

- When to swap in?
  - Demand paging: swap in when needed
  - Prefetching: swap in a page ahead of demand (anticipating demand)
    - When likely?
      - Code page P brought to memory, P+1 also likely
- When to write to disk
  - One at a time

Other Factors

- When to swap in?
  - Demand paging: swap in when needed
  - Prefetching: swap in a page ahead of demand (anticipating demand)
    - When likely?
      - Code page P brought to memory, P+1 also likely
- When to write to disk
  - One at a time
  - Clustered writes - preferred - 1 large write quicker than multiple smaller writes