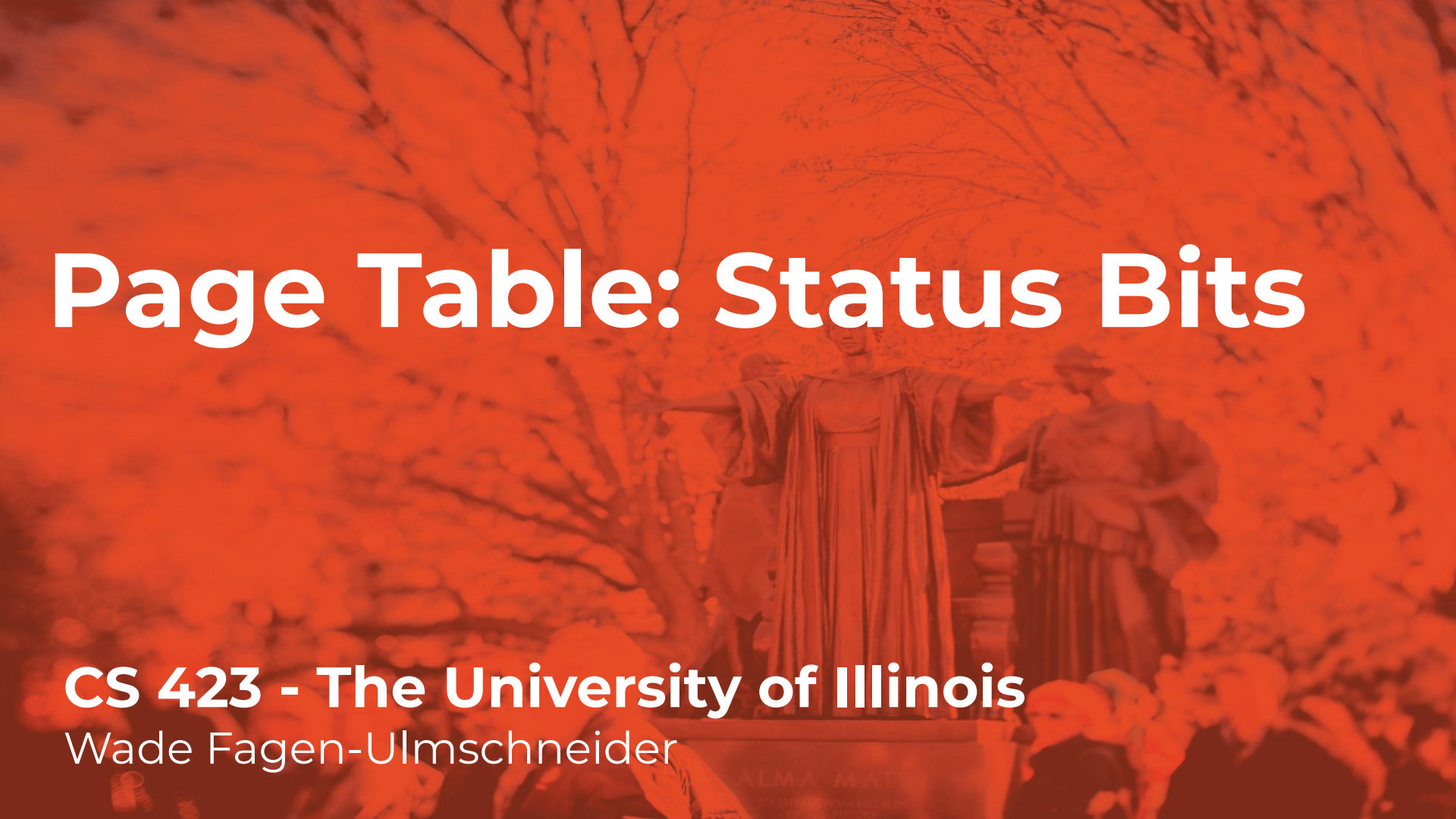
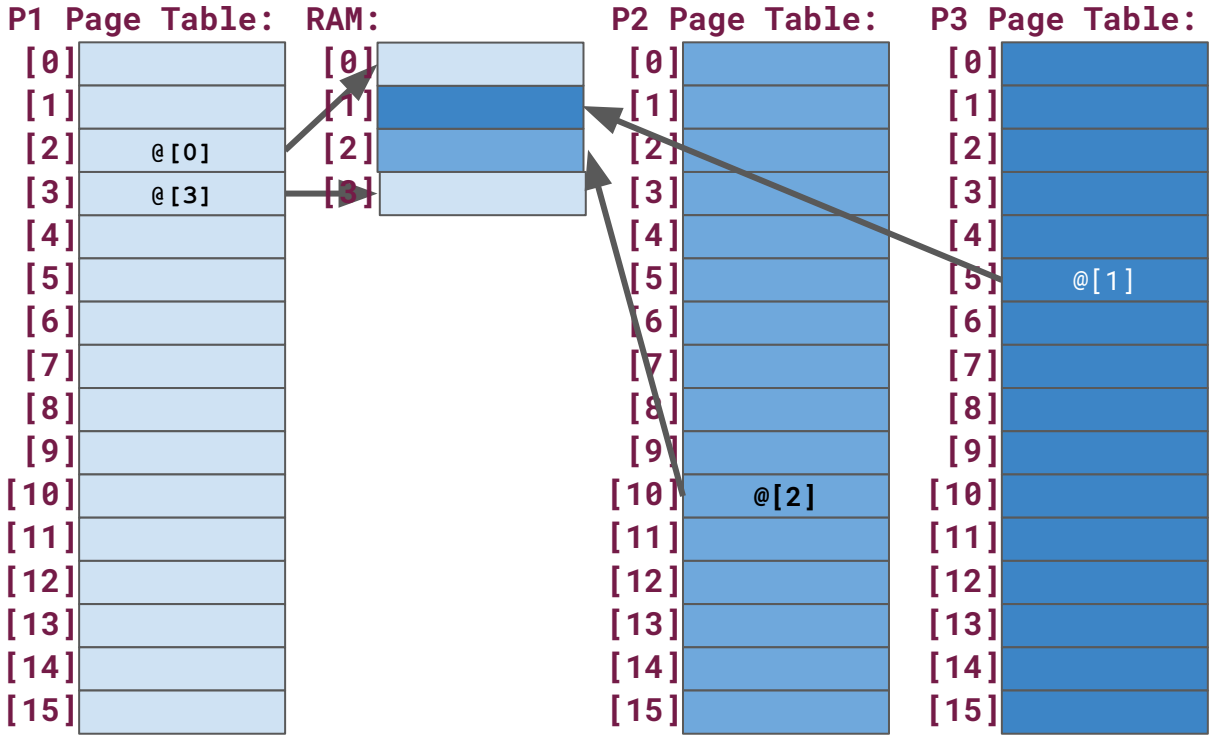


Page Table: Status Bits

CS 423 - The University of Illinois

Wade Fagen-Ulmschneider

Review: Page Tables



Review: Eviction Policies

- ★ Various Eviction Policies:
  - FIFO
  - LRU
  - LFU
  - NRU / “Second Chance”
  - Optimal

Page Table: Status Bits (x64)

★ As part of the Page Table Entry (PTE), all operating systems will maintain “status bits” about the page. On an x64 system:

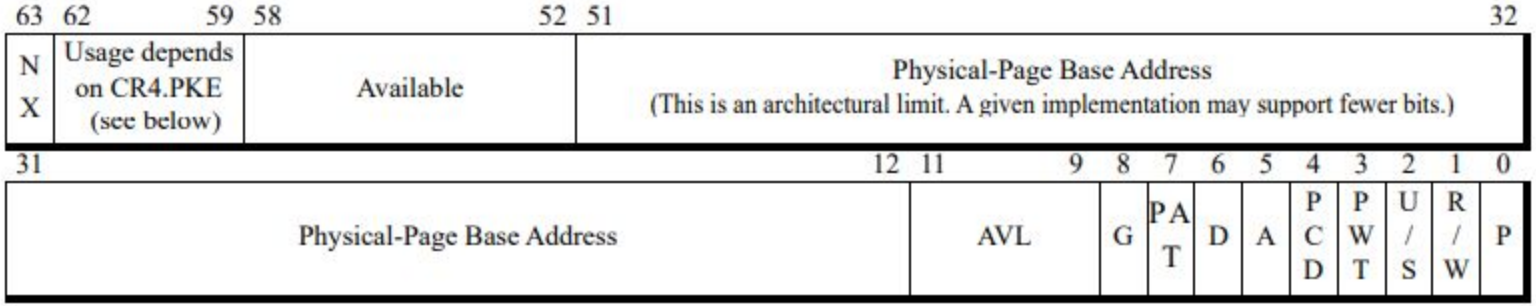
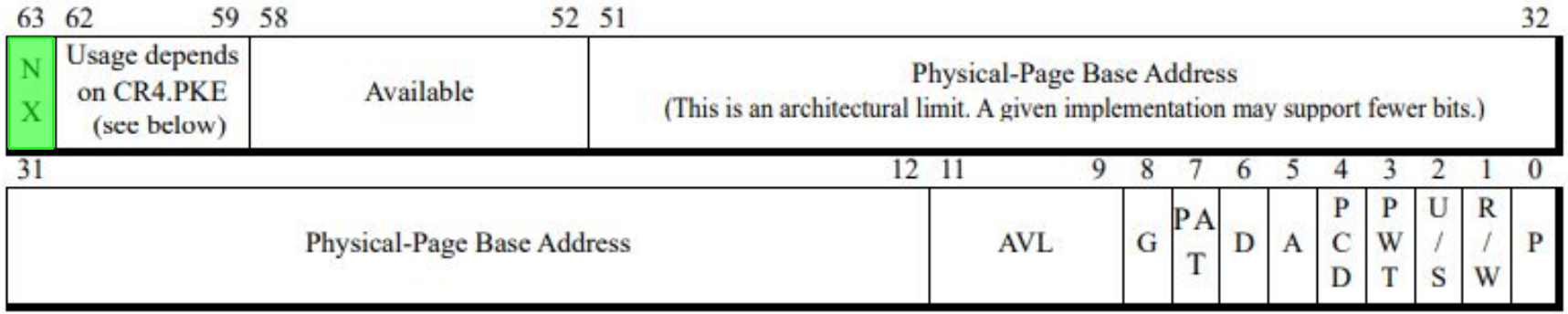


Figure 5-21. 4-Kbyte PTE—Long Mode

Source: AMD64 Architecture Programmer’s Manual, Volume 2: System Programming
https://www.amd.com/system/files/TechDocs/24593.pdf

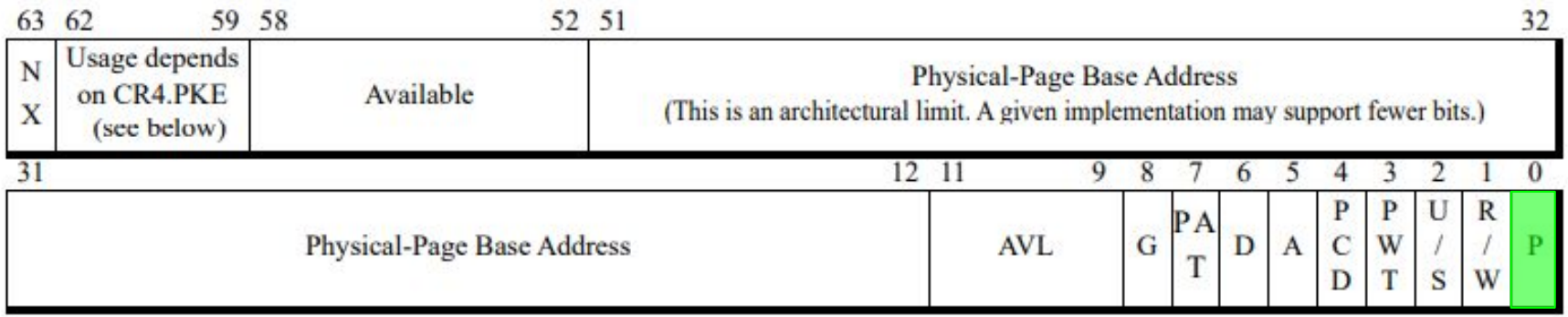


"No Execute Bit" (NX): Should the page's content be executed?

- "When the NX bit is set to 1, code cannot be executed from the mapped physical pages."

- *We don't want to execute stack/heap memory.*
- *The NX bit reduces many types of "buffer overflow" exploits where it was possible to override the return pointer in a function's stack frame and run code placed earlier in the buffer.*

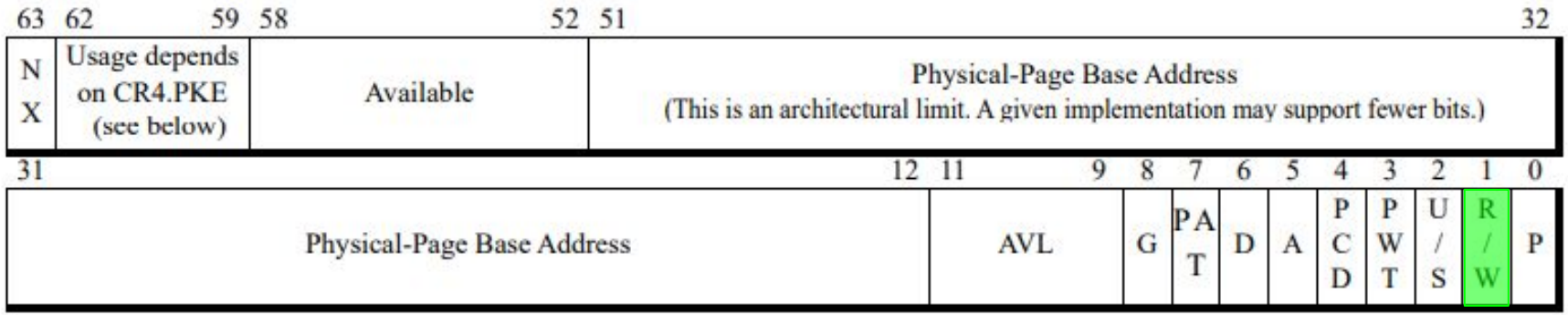
Source: AMD64 Architecture Programmer's Manual, Volume 2: System Programming
https://www.amd.com/system/files/TechDocs/24593.pdf



“Present Bit” (P): Is the page in physical memory?

- “When the P bit is cleared to 0, the table or physical page is not loaded in physical memory. When the P bit is set to 1, the table or physical page is loaded in physical memory. “

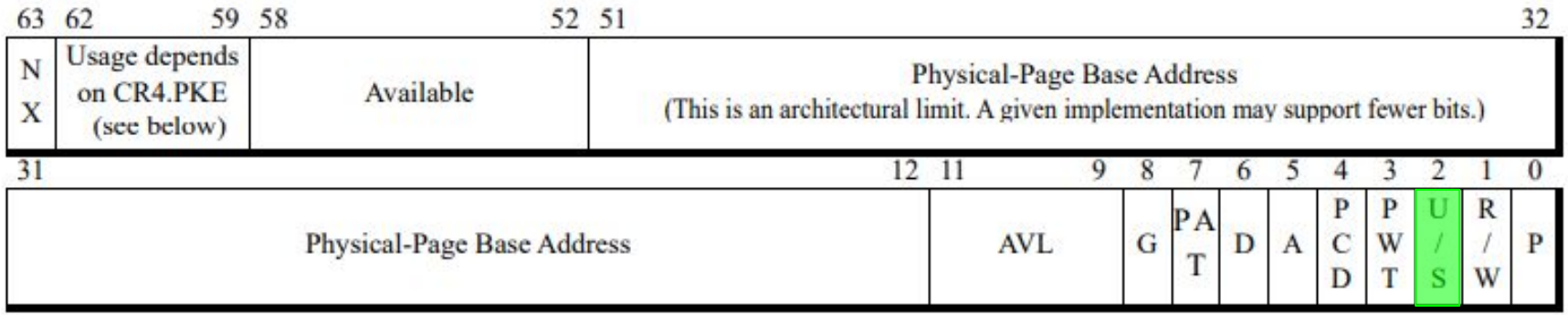
Source: AMD64 Architecture Programmer’s Manual, Volume 2: System Programming
https://www.amd.com/system/files/TechDocs/24593.pdf



“Read/Write Bit” (R/W): Is the page writable?

- “When the R/W bit is cleared to 0, access is restricted to read-only. When the R/W bit is set to 1, both read and write access is allowed.”

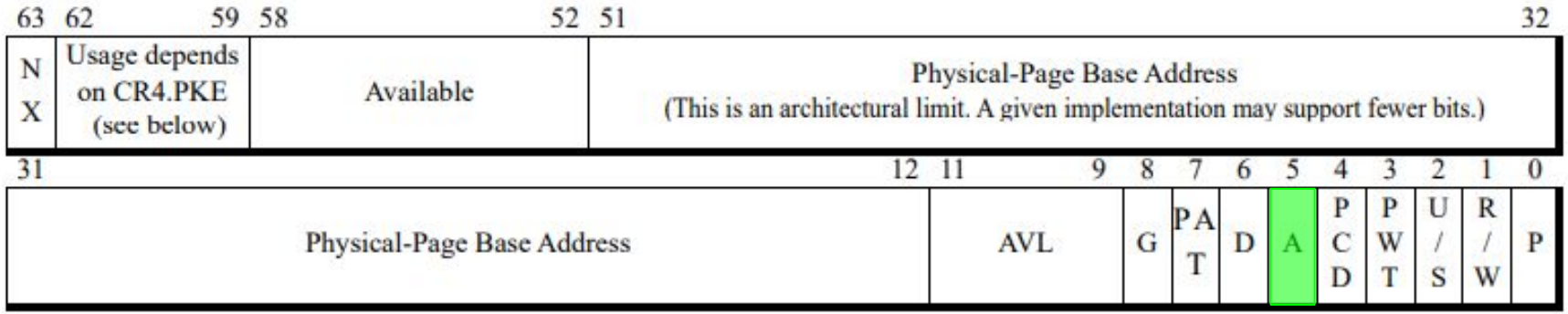
Source: AMD64 Architecture Programmer’s Manual, Volume 2: System Programming
https://www.amd.com/system/files/TechDocs/24593.pdf



“User/Supervisor” Bit (U/S): Is the user-accessible?

- “When the U/S bit is cleared to 0, access is restricted to supervisor level (CPL 0, 1, 2). When the U/S bit is set to 1, both user and supervisor access is allowed.”

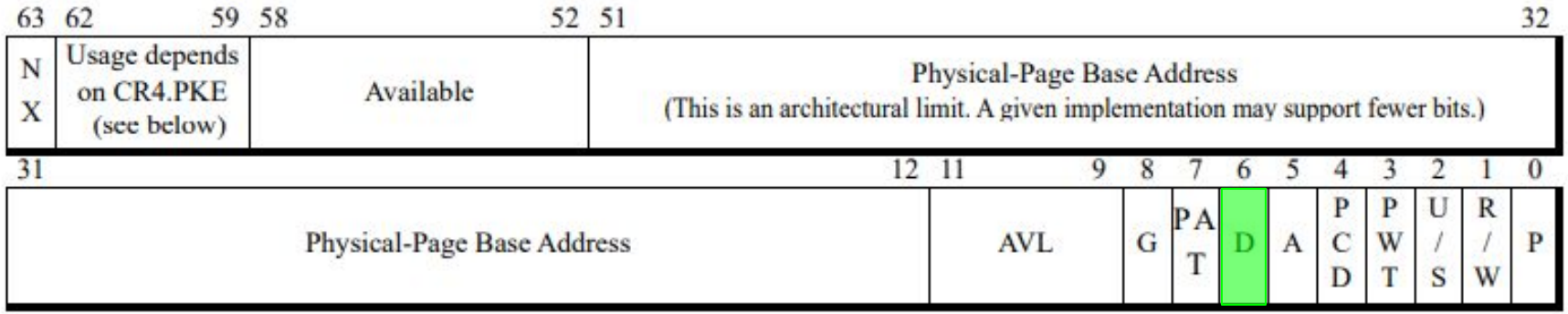
Source: AMD64 Architecture Programmer’s Manual, Volume 2: System Programming
https://www.amd.com/system/files/TechDocs/24593.pdf



“Accessed” Bit (A): Has this page been (recently accessed)?

- “The A bit is set to 1 by the processor the first time the table or physical page is either read from or written to. The A bit is never cleared by the processor. Instead, software must clear this bit to 0 when it needs to track the frequency of table or physical-page accesses.”
- As a Operating System, we decide when to clear the A bit!

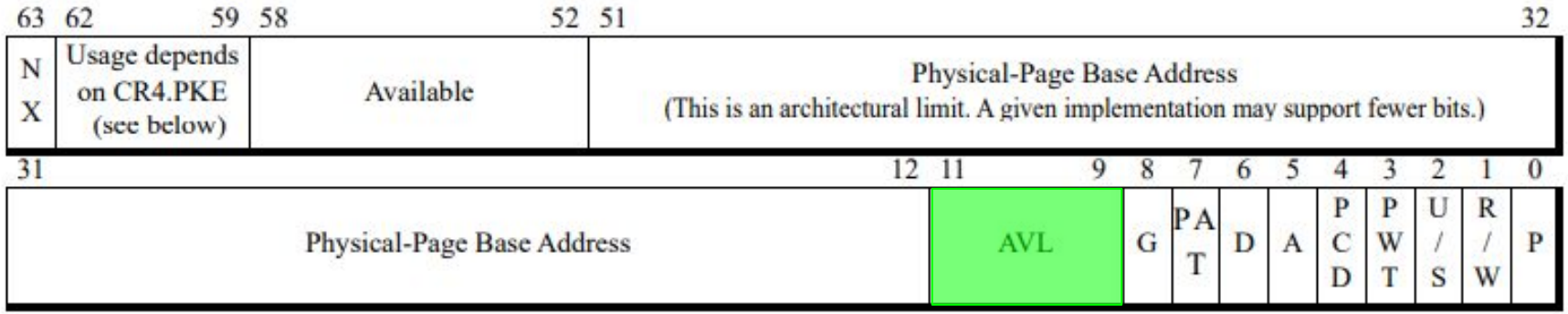
Source: AMD64 Architecture Programmer’s Manual, Volume 2: System Programming
https://www.amd.com/system/files/TechDocs/24593.pdf



“Dirty” Bit (D): Has the contents of this page been modified?

- “The D bit is set to 1 by the processor the first time there is a write to the physical page. The D bit is never cleared by the processor. Instead, software must clear this bit to 0 when it needs to track the frequency of physical page writes.”
- As a Operating System, we decide when to clear the D bit!

Source: AMD64 Architecture Programmer’s Manual, Volume 2: System Programming
https://www.amd.com/system/files/TechDocs/24593.pdf



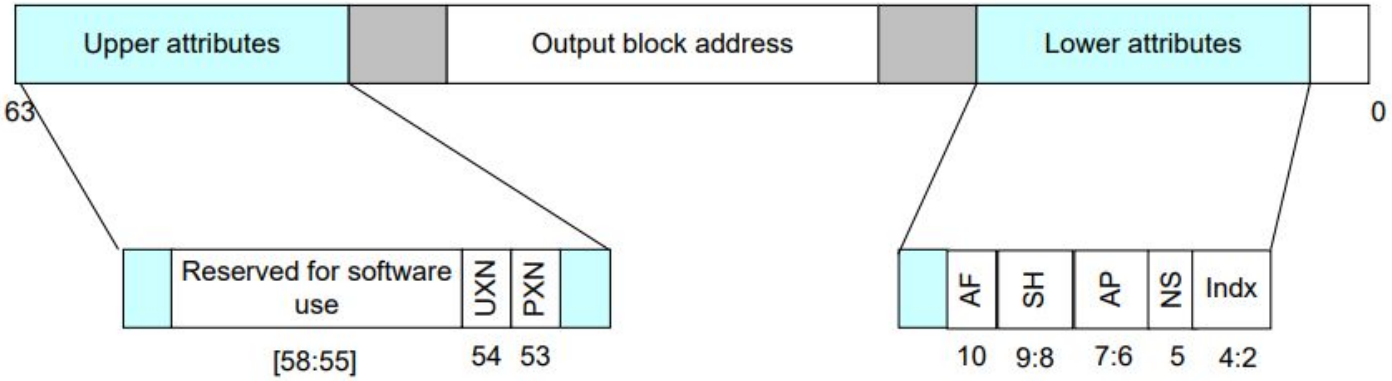
“Available to Software” Bits (AVL): Three bits for the OS to use for any purpose.

- “These bits are not interpreted by the processor and are available for use by system software.”

Source: AMD64 Architecture Programmer’s Manual, Volume 2: System Programming
https://www.amd.com/system/files/TechDocs/24593.pdf

Page Table: Status Bits (ARMv8)

★ An ARM processor has many similar status bits for PTEs:



- Unprivileged eXecute Never (UXN) and Privileged eXecute Never (PXN) are execution permissions.
- AF is the access flag.
- SH is the shareable attribute.
- AP is the access permission.
- NS is the security bit, but only at EL3 and Secure EL1.

Source: ARMv8-A Address translation
https://static.docs.arm.com/100940/0100/armv8_a_address%20translation_100940_0100_en.pdf

Status Bits

- ★ Page Table Entry status bits allow the kernel to know the “state” of a page, and use several bits for implementing page eviction schemes.

- ★ Many aspects of paging is built into hardware for speed, preventing the need for a trap to kernel for every memory access.

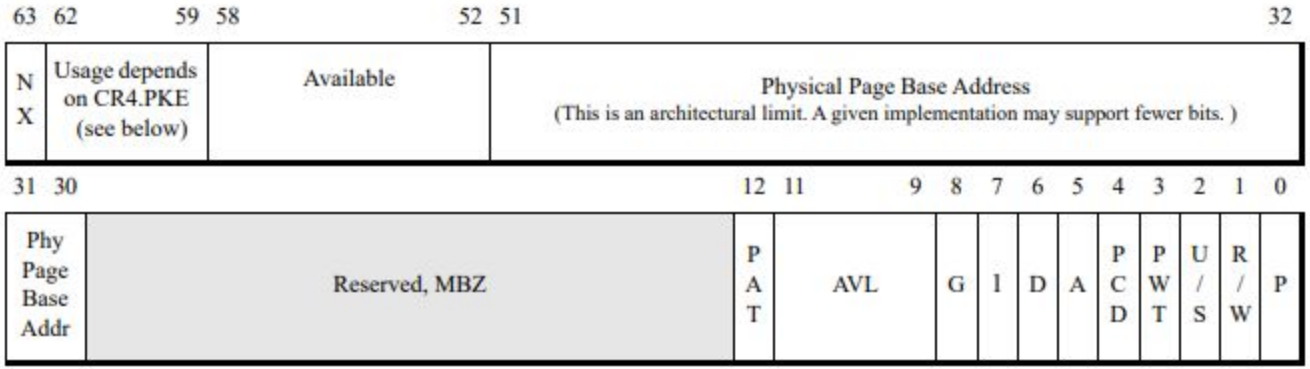
Page Table: Multi-level Page Tables

CS 423 - The University of Illinois

Wade Fagen-Ulmschneider

Page Tables Sizes

★ In the previous lecture, we saw the Page Table Entry for an x64 system:



...each entry uses **64 bits**, or **8B**, of memory!

Source: AMD64 Architecture Programmer's Manual, Volume 2: System Programming
https://www.amd.com/system/files/TechDocs/24593.pdf

Page Tables Sizes

- ★ Known:
  - In many x64 systems (including our VM), each **page is 4 KiB**.
  - On x64 systems, **each PTE is 8 B in size**.

- ★ If we have a system that has 16 GiB of RAM:

Page Tables Sizes

★ Known:
- In many x64 systems (including our VM), each **page is 4 KiB**.
- On x64 systems, **each PTE is 8 B in size**.

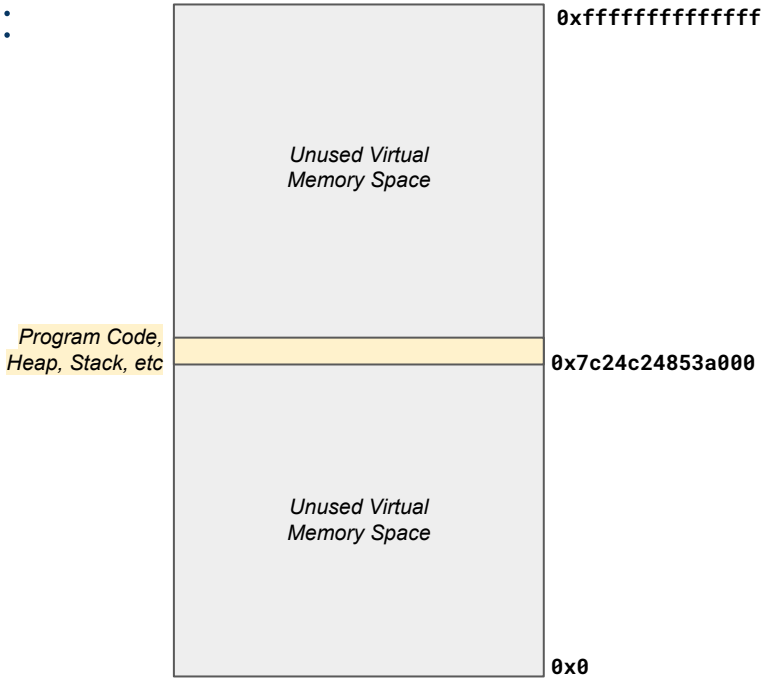
★ If we have a system that has 16 GiB of RAM:
- 16 GiB memory / 4 KiB pages = **4 MiB pages /process**.
- 4 MiB pages * 8 B / PTE =

32 MiB overhead for every page table (!!!!)

Remember each process has its own page table.

Multi-Level Page Table

★ In a given process, most of the allocated memory is in a small region of the full memory space:

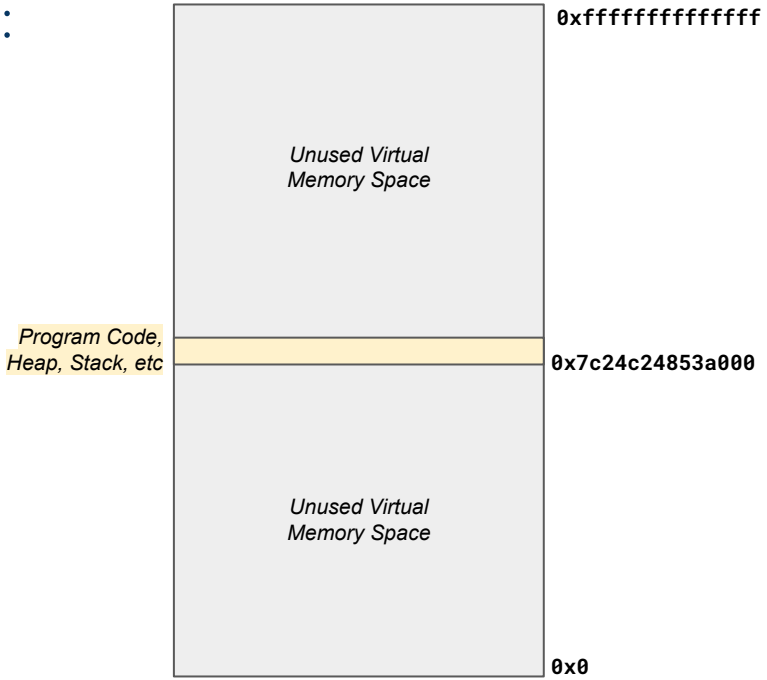


Multi-Level Page Table

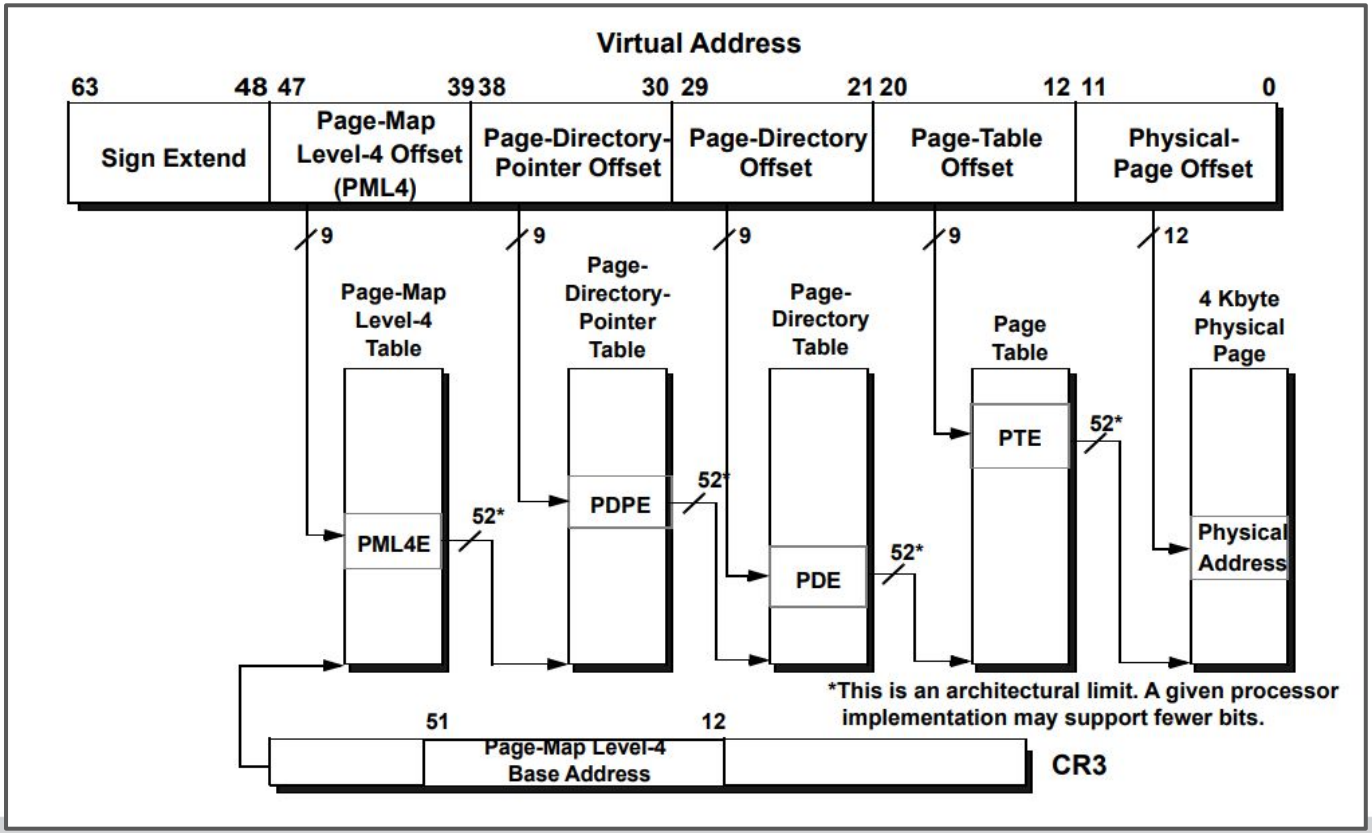
★ In a given process, most of the allocated memory is in a small region of the full memory space:

★ Design Idea:

- Why not use a **tree-like structure** to populate only the needed leaves of “memory tree”?
- Why not have **each “node” in the tree be really small** (maybe fit within a page)?



★ The x64 architecture does uses a multi-level lookup:



Source: AMD64 Architecture Programmer's Manual, Volume 2: System Programming
https://www.amd.com/system/files/TechDocs/24593.pdf

★ Each entry is (effectively) a PTE at each level:

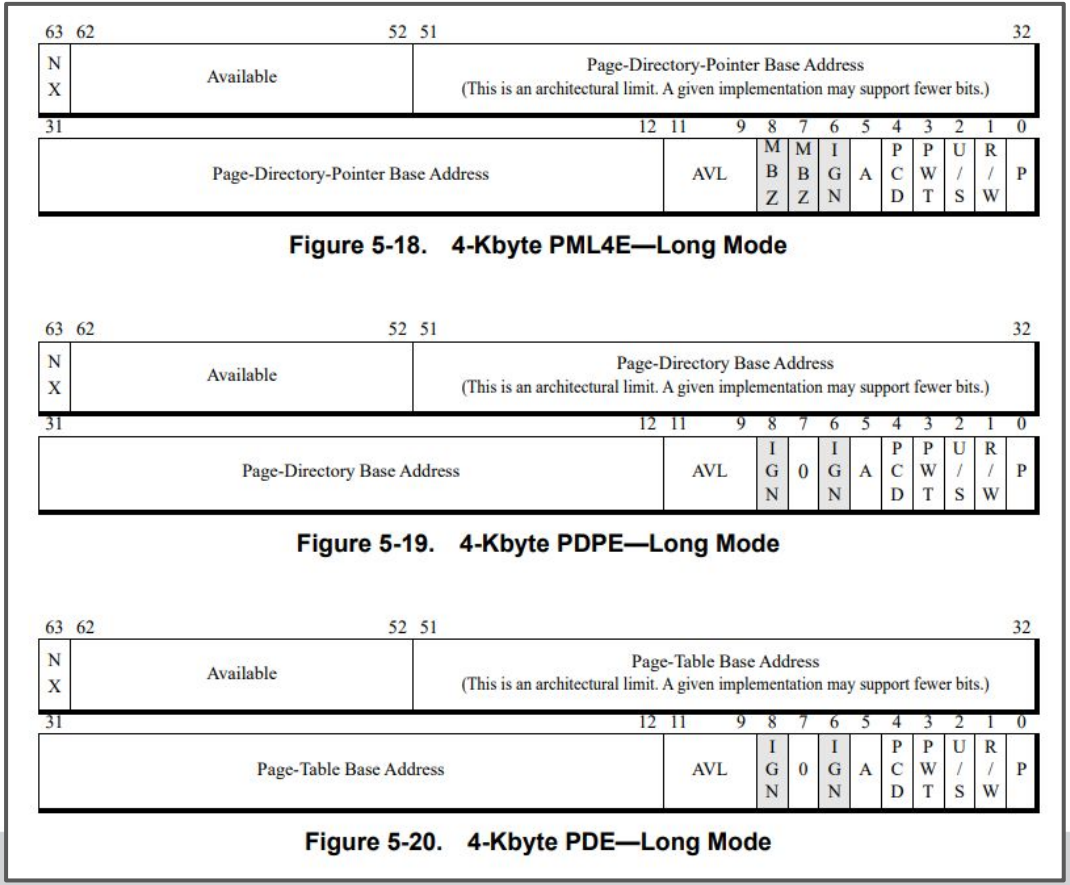


Figure 5-18. 4-Kbyte PML4E—Long Mode

Figure 5-19. 4-Kbyte PDPE—Long Mode

Figure 5-20. 4-Kbyte PDE—Long Mode

Source: AMD64 Architecture Programmer's Manual, Volume 2: System Programming
https://www.amd.com/system/files/TechDocs/24593.pdf

Multi-Level Page Tables

- ★ Since the memory address space for a given process is usually sparse, the “tree” is **very** sparse:
  - The highest-level page table may have only a few entries.
  - All other entries are “NULL” and do not need to be allocated until used.
  - Translation Lookaside Buffers (TLBs) and other hardware optimize the lookup+caching of page tables.

Page Size?

★ What is the ideal page size?

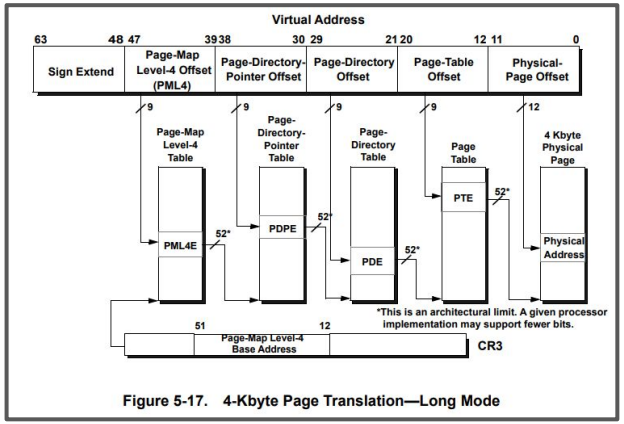


Figure 5-17. 4-Kbyte Page Translation—Long Mode

4 KiB pages ⇒ 4-level lookup

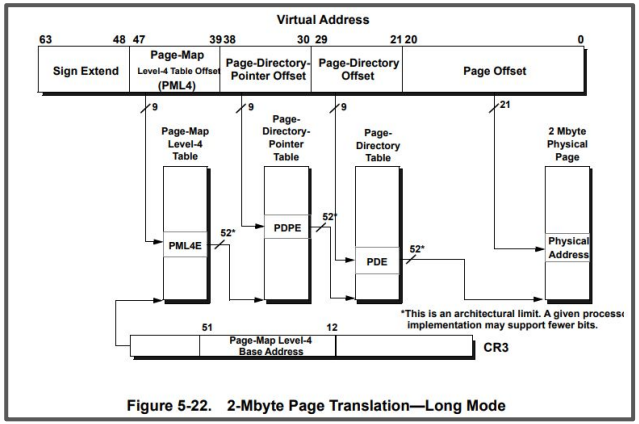


Figure 5-22. 2-Mbyte Page Translation—Long Mode

2 MiB pages ⇒ 3-level lookup

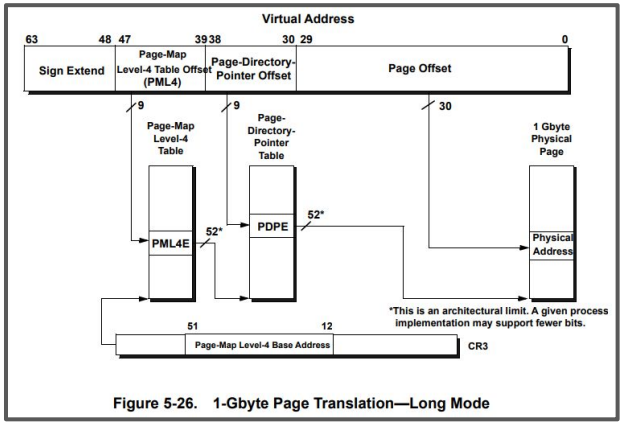


Figure 5-26. 1-Gbyte Page Translation—Long Mode

1 GiB pages ⇒ 2-level lookup

Page Size?

★ What is the ideal page size?

Small Page Size (ex: 4 KiB):

★ “Locality of Reference” tends to be relatively small.

★ Small pages require minimal I/O during a page fault.

★ Very little fragmentation.

★ However, requires many level multi-level page table.

Large Page Size (ex: 1 GiB):

★ Small page table, minimal levels in a multi-level page table.

★ Internal fragmentation (cannot allocate smaller than page size)

★ Large I/O transfers during page faults, impacts system performance.

Page Size?

★ What is the ideal page size?

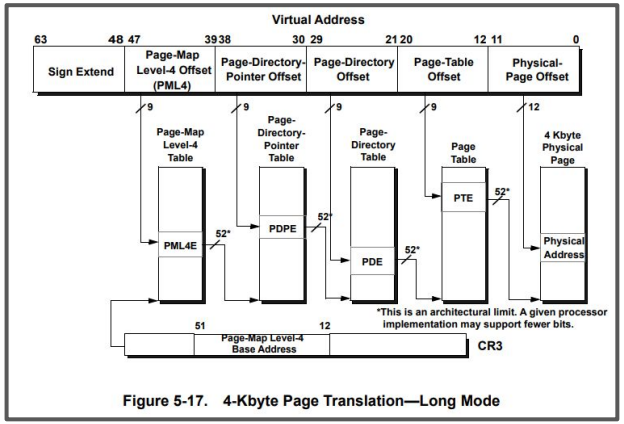


Figure 5-17. 4-Kbyte Page Translation—Long Mode

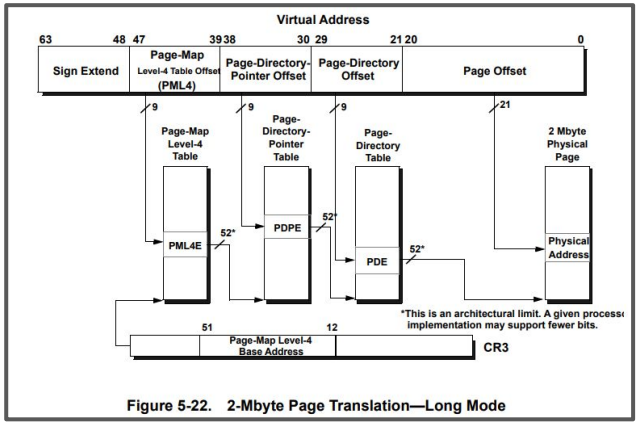


Figure 5-22. 2-Mbyte Page Translation—Long Mode

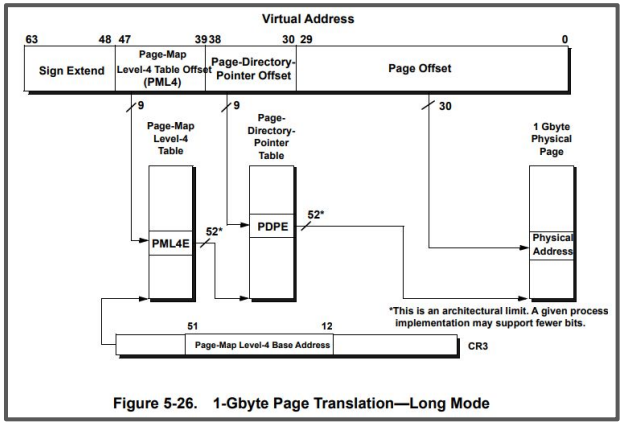


Figure 5-26. 1-Gbyte Page Translation—Long Mode

4 KiB pages ⇒ 4-level lookup

2 MiB pages ⇒ 3-level lookup

1 GiB pages ⇒ 2-level lookup

Likely a very slow migration as RAM sizes increases...

2021

2030?

Memory: Thrashing

CS 423 - The University of Illinois

Wade Fagen-Ulmschneider

Thrashing

- ★ **Page eviction occurs when a system has filled all available memory**.
  - What happens when this happens often?

    *Remember: Each Page Fault is likely a **SLOW** disk I/O operation!*

Thrashing

★ **System is in a page faulting constantly.**

- Pages are being evicted almost as soon as they're paged in.

  ⇒

- Processes are blocked waiting for disk I/O for their pages to be paged in -- CPU begins to be idle waiting for CPU operations just to get paged in!

  ⇒

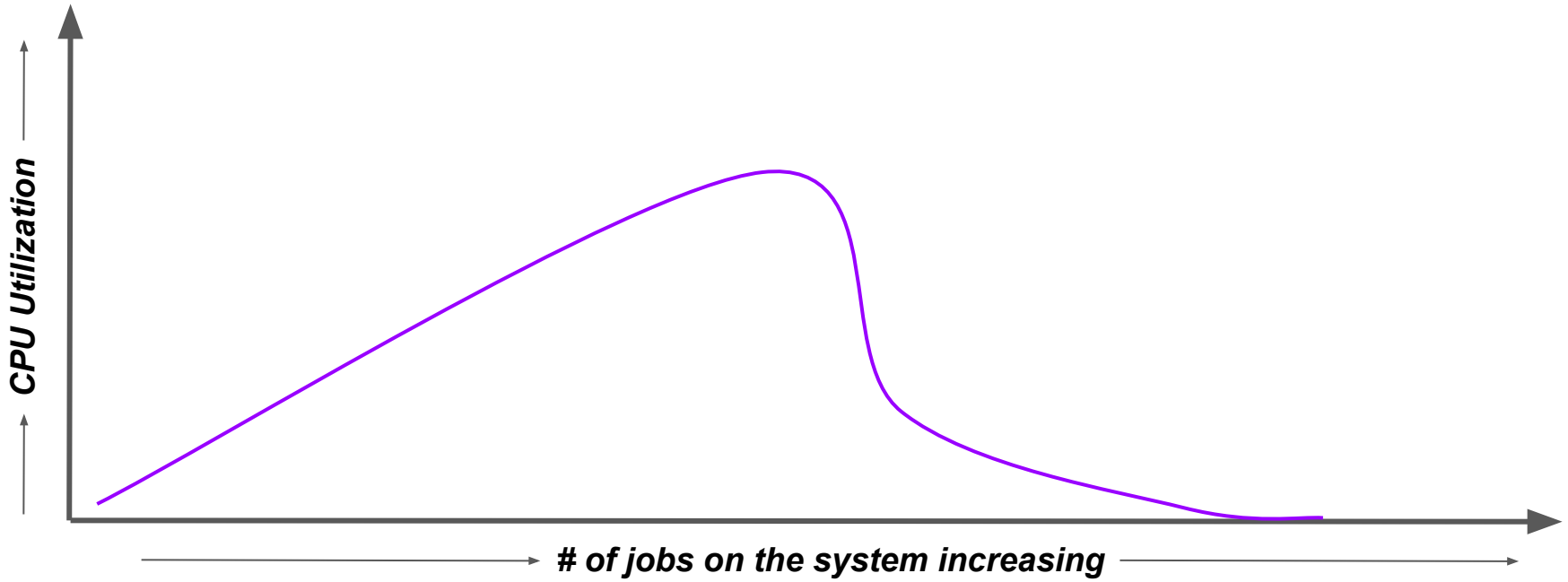
- On server/clusters, new jobs may be launched/assigned due to low CPU utilization.

  ⇒

- More pages are requested and the **cycle gets even worse**!

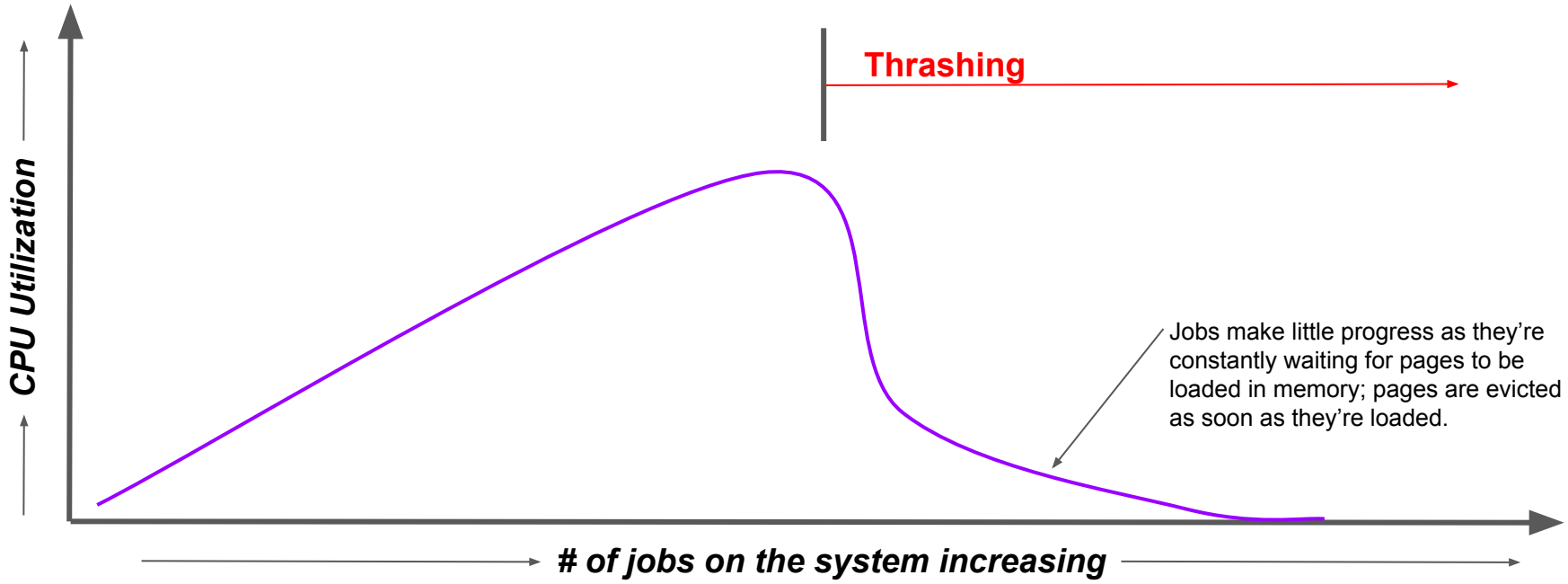
Thrashing

★ **In a memory-constrained system:**



Thrashing

★ **In a memory-constrained system:**

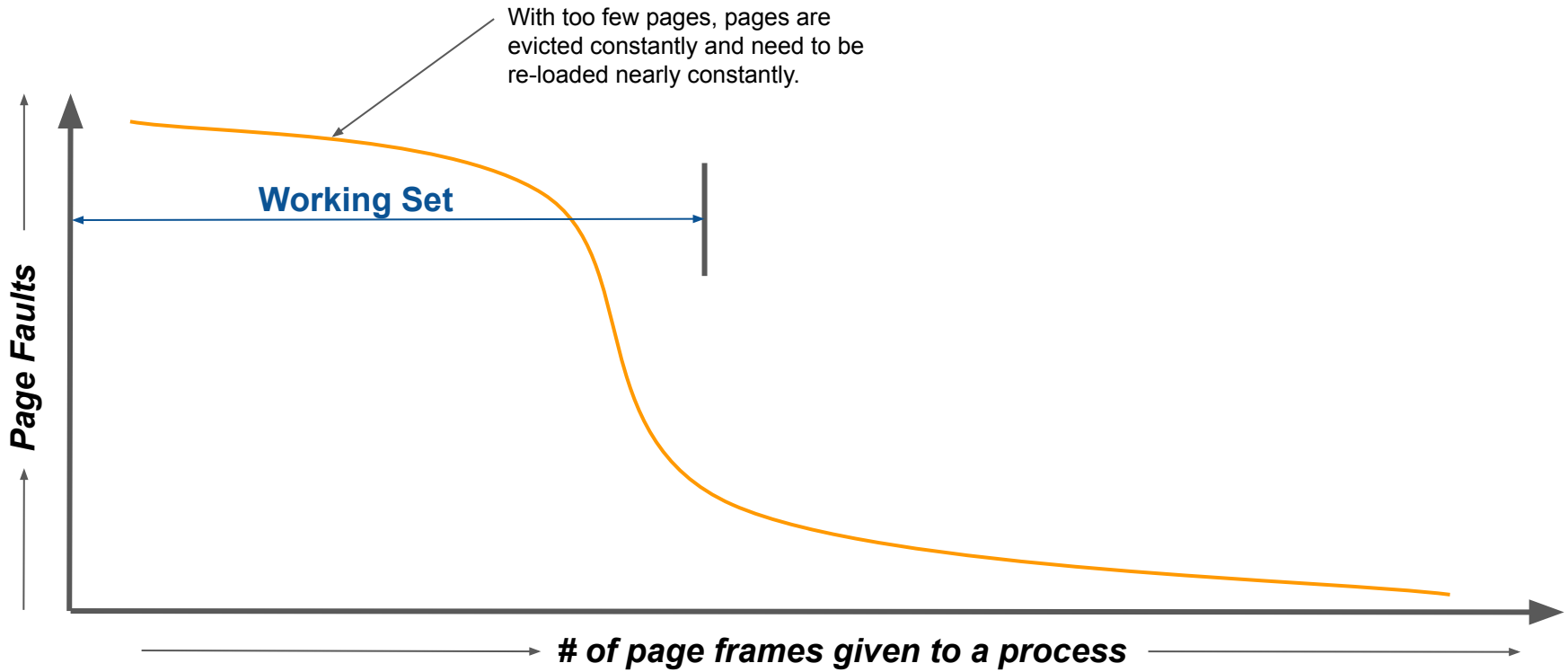


Thrashing

- ★ Thrashing should **always be avoided** and wastes resources (ex: CPU).
  - More total progress can be made over a fixed period of time when thrashing is avoided; total progress is lost with thrashing.
  - Each process has an ideal "**working set**" of pages to minimize the rate of page faults.

Working Set

- ★ The **Working Set** models the number of pages needed for a given process to minimize the page fault rate.
  - *A generalized model that follows the principle of locality.*
  - *Helps to explore the effect of thrashing on a system*



★ ***Idea:*** *As the number of page frames incase above a threshold, the number of page faults drop dramatically!*

Working Set

- ...but is this **always true**?

Memory: Belady's Anomaly

CS 423 - The University of Illinois

Wade Fagen-Ulmschneider

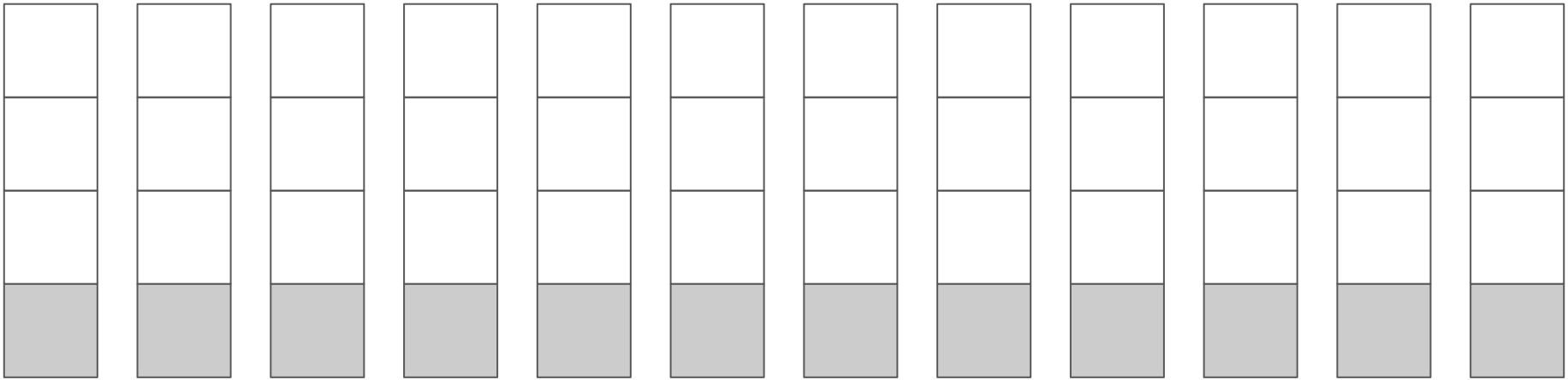
Belady's Anomaly

★ Consider the following access of five pages:

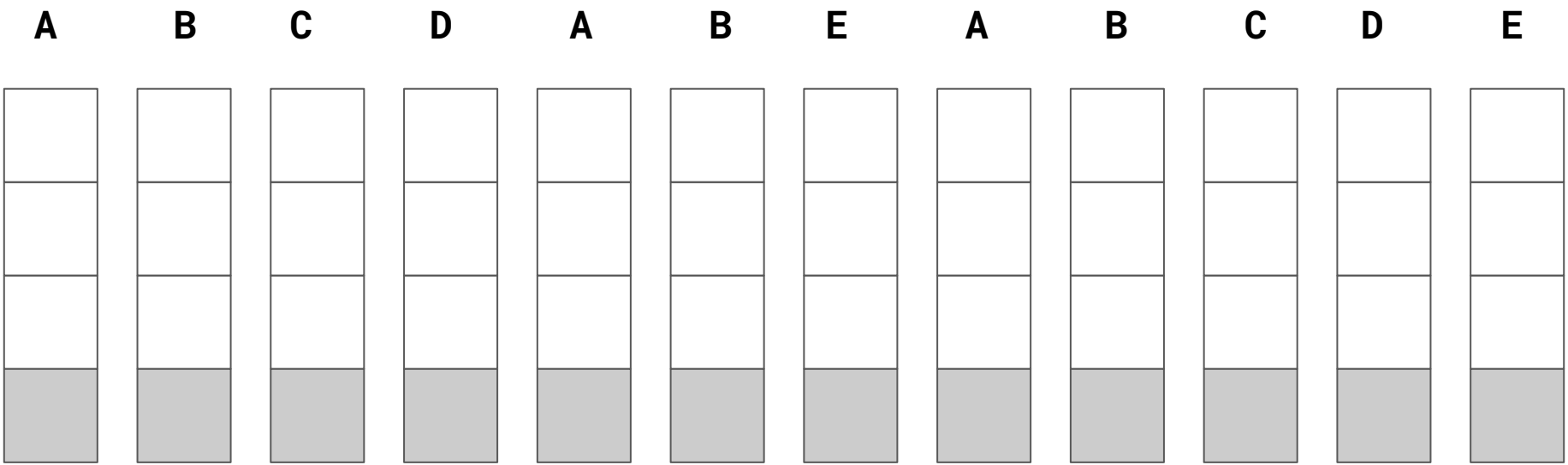
A B C D A B E A B C D E

Optimal w/ 3 RAM pages:

A B C D A B E A B C D E

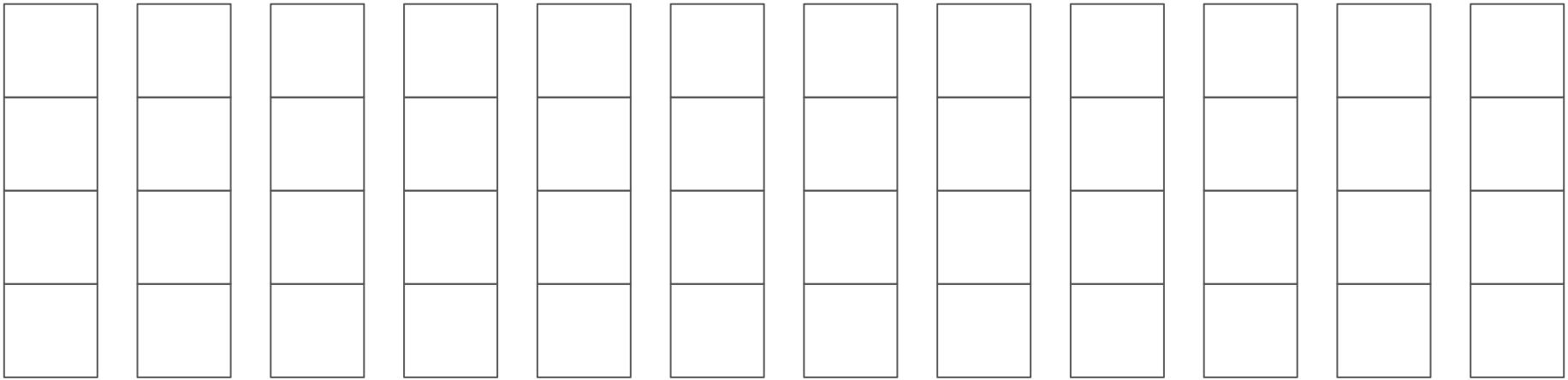


FIFO w/ 3 RAM pages:



FIFO w/ 4 RAM pages:

A B C D A B E A B C D E



Belady's Anomaly

- ★ Increasing the number of page frames affects the order in which items are removed.
  - For certain memory access patterns, this can actually **increase the page fault rate**! **(!!)**

- ★ Belady's Anomaly is reference string dependent; **intuition about increasing page count still holds in general case**.