Intel 64-bit Architecture: Memory Paging

By Helmut Cespedes and
Michael Ludwikowski

Currently the standard for operating systems popular examples include Windows 11, Linux, and MacOS. We will focus on Linux for this presentation.

The Boot

Once our Linux boots up physical RAM is divided into *page frames,* these are 4 KB chunks. Each of these page frames contain:

- Reference Count: How many processes are using it.
- Flags: Examples Shown Later.
- Location Info: Map to find the page frame itself.

Notice: Page Frame is the physical address and page is virtual address.

The Start

The Memory Management Unit - Makes sense of where thing are, translates **Virtual Memory -> Physical Memory.**

Once a Request is made, a check is sent to the Translation Lookaside buffer (TLB): “Do you know this memory address?”

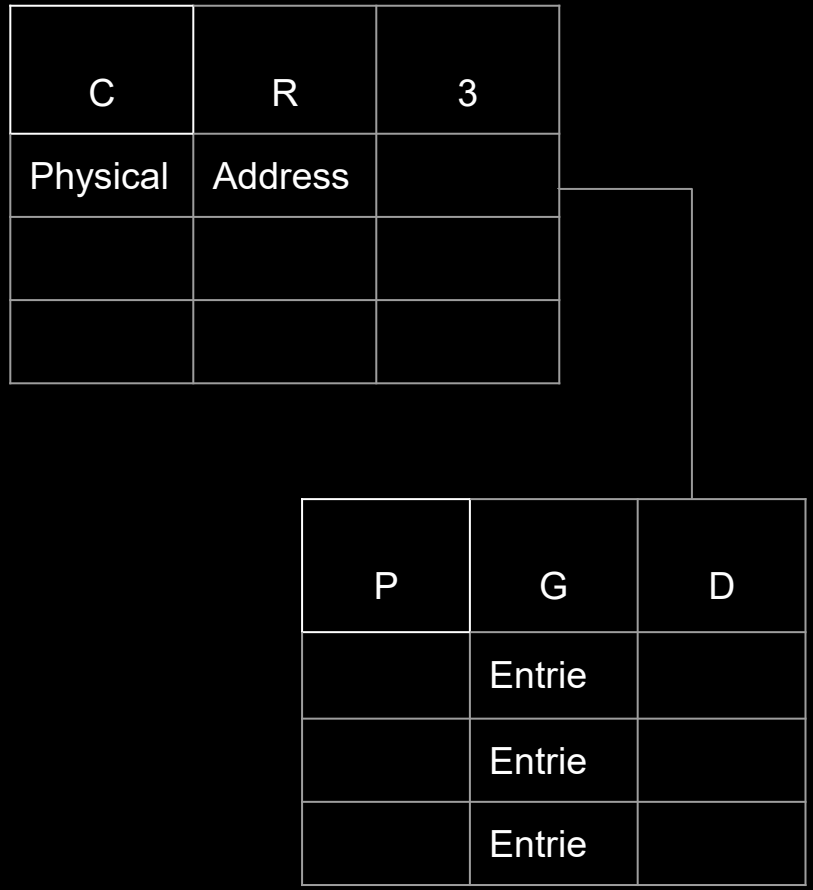
If It does, the MMU receives the physical address very fast.

Else, the MMU must perform a page walk.

Walking

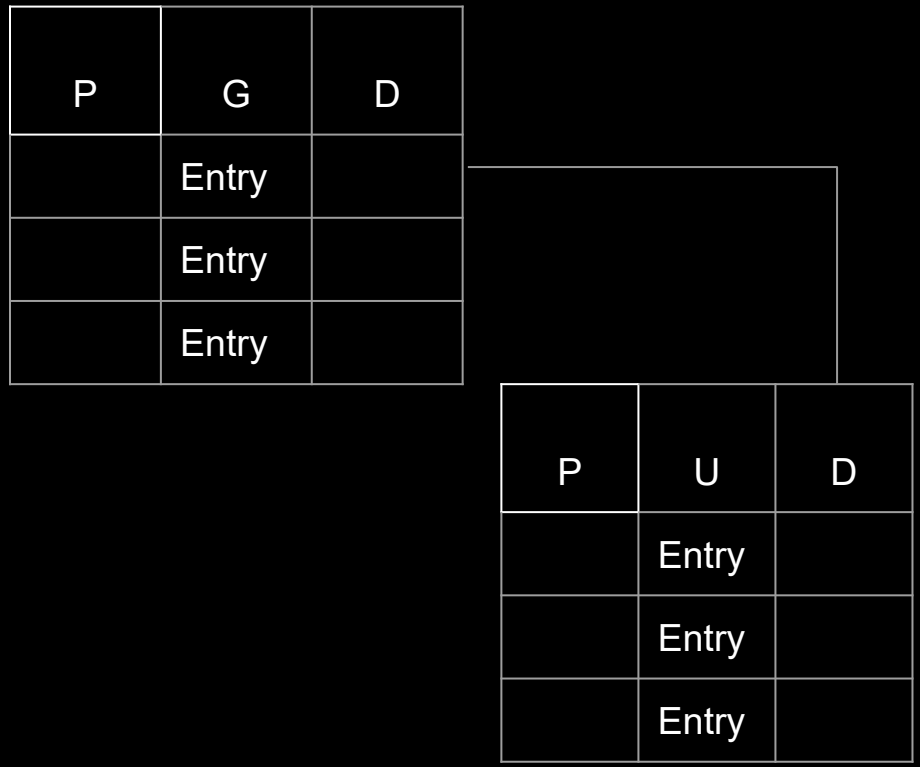
The first step in this walk is to check the CR3 register, this is where processes store their Page Global Director (PGD) addresses, which contain entries.

Only the actual directory page is stored not the entries within this is where we used Bits 47-39 to find our PGD index.



Walking Continued...

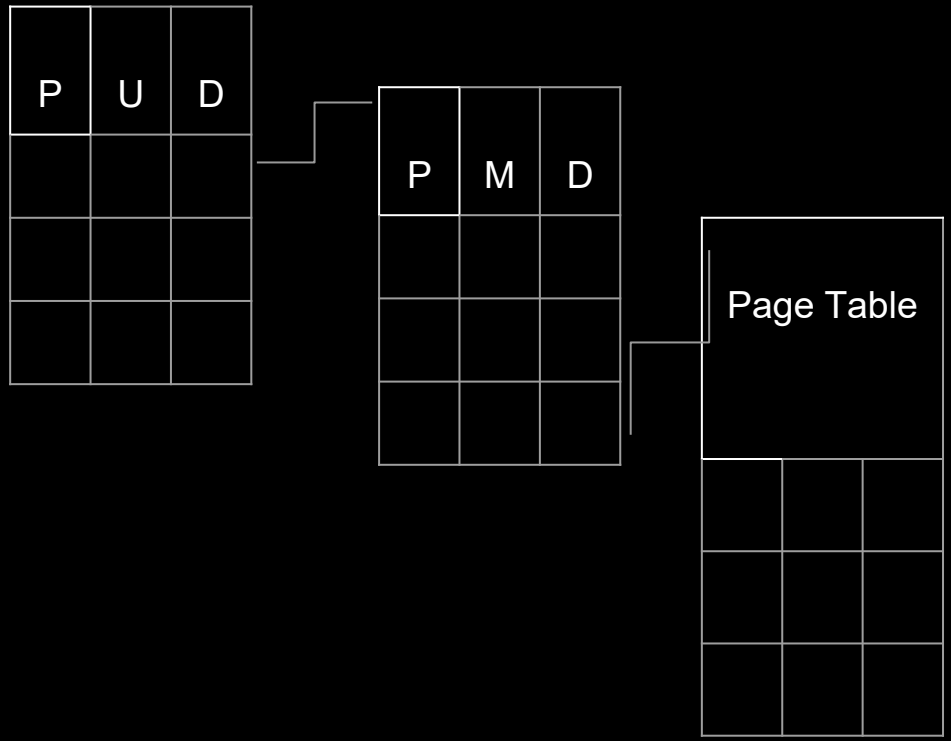
Now were at the second level of the page walk. In which the process repeats. The PGD stores entries where we can get an address to the Page Upper Directory (PUD). Bits 30-38 are used to index the PUD's entries.



Walking Continued...

Now were at the third level of the page walk. In which the process repeats following the same pattern.

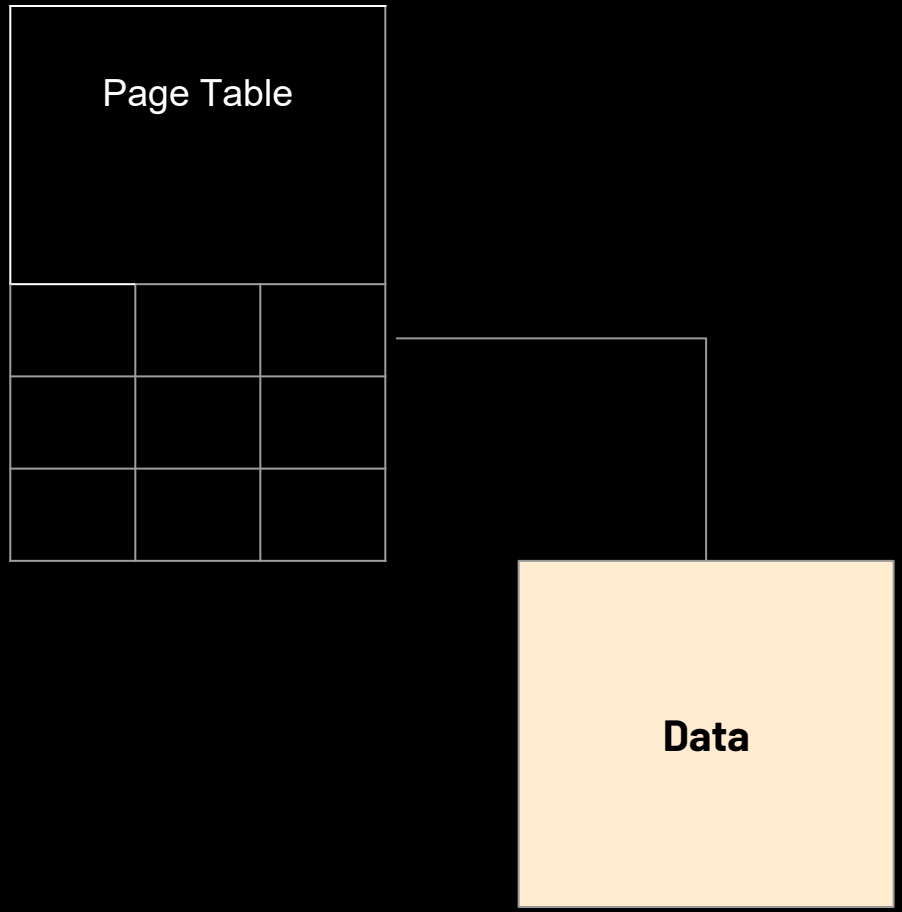
PUD (9 bits) -> (1)PMD (9 bits) -> Page Table



(1) Page Middle Directory

Arrived

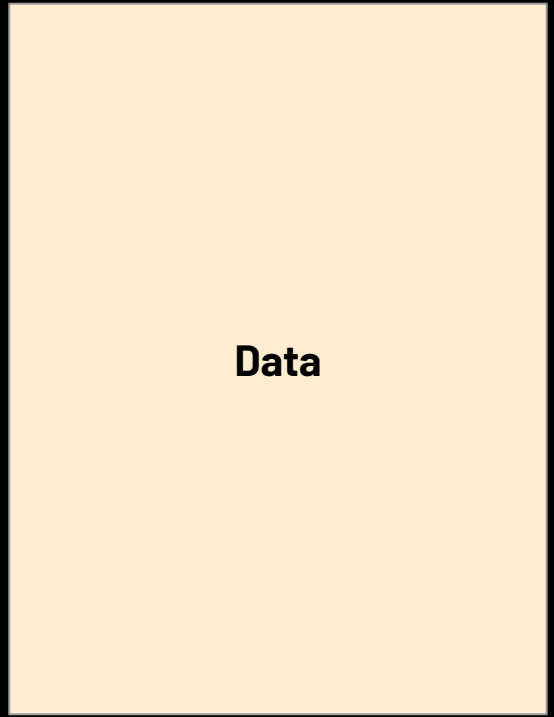
Once we arrive at the page table we repeat a similar process where we get to a specific page (12 bits), in this page is where we find the actual data that we stored.. Now the Page walk is done and this address map may be input into the TLB.



HUGE pages.

While these pages are not limited to just Linux, it has the best support for them.

A regular page has a 4KB size, for a page to have a bigger size there must be a tradeoff. Linux cuts down on page levels in order to offset bits to find data in these bigger pages.



HUGE pages continued...

For a page 2MB we must skip from the PMD to the page itself so 12 + 9 bits are used.

For 1 GB page 12 + 9 + 9 bits.

Entry

Name	No Execute	Protection keys	Software Defined	Ignore	Physical address	Software defined	ignore	dirty	Accesses	Cache ignored	User Mode Access	R/W	Present
													0

Page Fault

If Present = 0, the CPU stops its current process and initiates a Page Fault Exception (Interrupt).

The fault virtual address and error code are stored in a CR2 register so the kernel can see what went wrong.

At this point the Kernel takes over via the Page Fault Handler.

P = 0 is not the only page fault, I simply used it to introduce a new topic.

Page Fault Handler

Now the PFH check why the fault happened, there are several possible faults:

- Missing Page
- Write In Read Only
- Invalid Address

Then it must fix the problem if possible, in the case that its not possible the process crashes. *Ex. Setting a pointer to null then trying to give that pointer a value. (SEG FAULT).*

Once this process is completed, the Kernel Gives access back to the CPU.

Not Mentioned

There is another level in that is not turned on by default which raises the Virtual Address space beyond the normal 48 bits to 57 bits allowing for up to 128 PB of virtual memory as long as the CPU supports it.

Linux uses demand Paging, the kernel does know about the page frames at boot time but it does not immediately load all pages, this is done dynamically on demand, only when a process access them.

Questions?

THANK YOU

COMPANY NAME

Works Cited

Comment *et al.* (2025) *Paging in operating system*, *GeeksforGeeks*. Available at: https://www.geeksforgeeks.org/operating-systems/paging-in-operating-system/ (Accessed: 04 November 2025).

Notes taken and applied from several videos:

https://www.youtube.com/watch?v=LKYKp_ZzIvM
https://www.youtube.com/watch?v=fGP6VHxqkIM&t=386s
https://www.youtube.com/watch?v=A9WLYbE0p-I