Self-referencing Page Tables for the x86-Architecture *

A simple Paging Implementation for a minimalistic Operating System

Steffen Vogel
Institute for Automation of Complex Power Systems
E.ON Energy Research Center, RWTH Aachen University
Mathieustr. 10, 52074 Aachen, Germany
Academic advisor: Dr. rer nat. Stefan Lankes
steffen.vogel@rwth-aachen.de

ABSTRACT

The adoption of 64 bit architectures went along with an extension of the *virtual address space* (VAS). To cope with this growth, the memory management unit (MMU) had to be extended as well. For paging-based systems like Intel's x86-architecture this was realized by adding more levels of indirection to the *page table walk*.

This walk translates virtual *pages* to physical *page frames* (PF) by performing look-ups in a radix / prefix tree in which every node represents a *page table* (Figure 1a). Since the tables are part of the translation process, they must be referenced by physical *page frame numbers* (PFN, blue line). As the operating system is only eligible to access the VAS, it cannot follow the path of a walk. In order to allow the manipulation of page tables, it must provide:

- Access to the table entries, by mapping the tables themselves to the VAS.
- A mapping between physical references to corresponding locations in the VAS.

Additionally, every level of the page table walk increases the complexity of managing these mappings. They also increase the memory consumption by occupying physical page frames. It is possible to avoid both drawbacks by the technique described in the following.

In my bachelor thesis, I presented an approach, which is compatible with both the 32 bit and 64 bit version of Intel's x86-architecture. This allows for a replacement of two code bases, one for each architecture, by one supporting both. Thus, results in a shorter, easier comprehensible, and maintainable code. As foundation for this implementation our teaching OS called "eduOS" was used[1]. "eduOS" supports only the 32 bit protected mode whereas the 64 bit longmode is only implemented for an internal research kernel.

Thanks to the sophisticated design of Intel's x86 MMU, it is possible to avoid most of the complexity and space requirements by using a little trick. Adding a self-reference in the root table (PML4 resp. PGD) automatically enables access to all page tables from the VAS without the need for manual mappings as described above (Figure 1b). The operating system does not need to manually follow the path of a page table walk, as this task is executed by the MMU for accessing individual tables instead of page frames.

An access to the VAS region covered by a self-reference causes the MMU to look up the root table twice (red line). Effectively, this shifts the whole page table walk by one level. Therefore, it stops with the PFN of page tables instead of page frames that are usually translated by the MMU. Here, both the PML4 and PDPT indexes are used to choose an entry out of the PML4 table. Therefore, it must be guaranteed that PML4 entries can be interpreted as PDPT entries, too. This demands for the following requirements:

- Homogenous coding of paging flags across all paging levels.
- Equal table sizes across all paging levels.

Fortunately, the x86-architecture complies with this prerequisites as shown in Figure 2. Green colored flags are coded consistently across all paging levels. Only *PAT*, *size* and *global* flags have a slightly different meaning for entries in the PGT. My bachelor thesis shows that these deviations still allow maintaining full control caching and memory protection properties of self-mapped tables. This includes for common system calls like `fork()` and `kill()`.

By repeatedly addressing the self-reference, it is also possible to access tables of the upper levels (PGD to PML4). Table 1 shows the resulting virtual addresses of all page tables when using the last (512^{th}) entry of the PML4 table for the self-reference[2]. This grants access to all possible page

*A full version of the thesis and slides are available at:
`http://www.noteblok.net/2014/06/14/bachelor/`

[1] Description and source code at:
`http://www.os.rwth-aachen.de`

[2] This is an arbitrary choice. All other entries are feasible, too.

ASPLOS Student Research Competition '15 Istanbul, Turkey

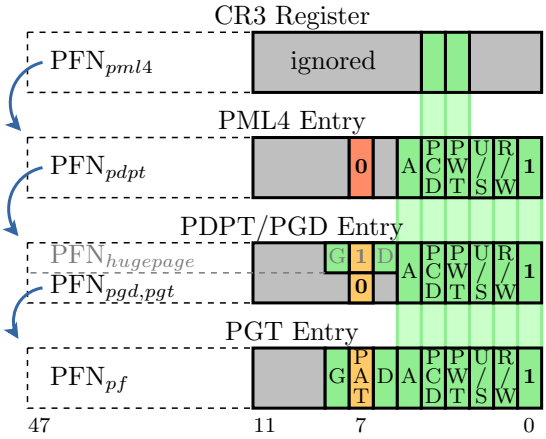


Figure 2: Similar flags across all paging levels.

Table	32 bit Protected-Mode	64 bit Longmode
PGT	`0xFFC00`000	`0xFF8000000`000
PGD	`0xFFFFF`000	`0xFFFFC0000`000
PDPT	–	`0xFFFFFFE00`000
PML4	–	`0xFFFFFFFFF`000

Table 1: Virtual addresses of self-mapped tables.

tables, including those which might not yet exist and may be allocated in the future. Hence, the self-reference reserves a fixed fraction of the VAS for the page tables. The size of this region is equal to $\frac{1}{512} \cdot 256\,\text{TiB} = 512\,\text{GiB}$ for 64 bit (resp. $\frac{1}{1024} \cdot 4\,\text{GiB} = 4\,\text{MiB}$ for 32 bit), which is negligible in comparison to the huge VAS of 2^{48} byte.

For the manipulation of page table entries two approaches are feasible:

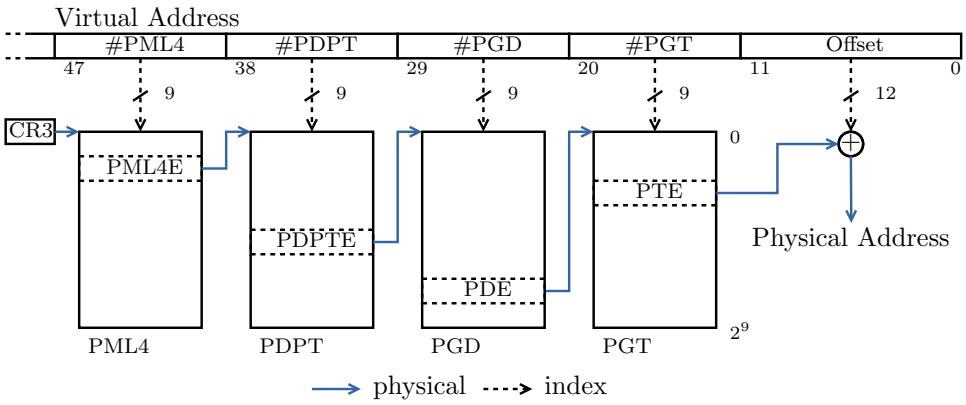
Top-down Use known tree traversals, starting at the root node, which corresponds to the PML4 respectively PGD.

Bottom-up Use the page fault handler to create new tables on-the-fly, when they are not yet present.

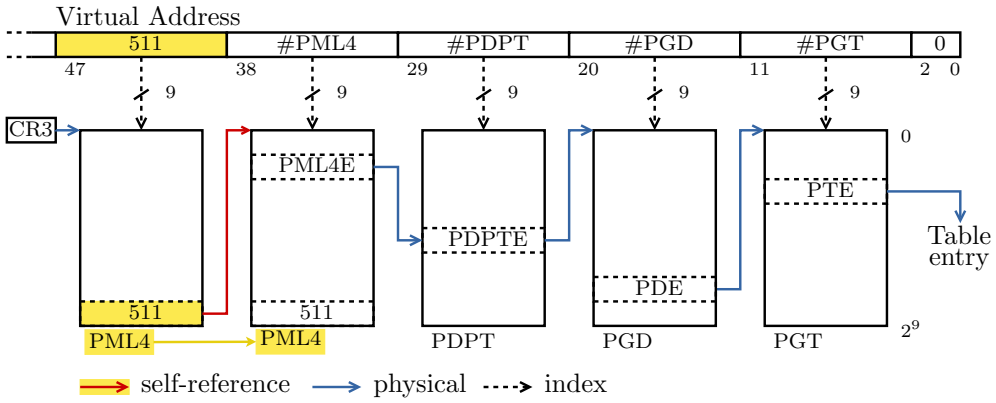
But there are also other architectures which satisfy the prerequisites described above. One of these is the Alpha[3] architecture, which suggests a similar approach in the reference manual. Intel and AMD do not mention the technique in their x86 manuals. In the field of operating systems, support is far more limited. There is only a single reference[4] dated to 2010 indicating that Microsoft might use a similar approach for its NT kernel. Linux cannot profit because its paging implementation must support a broad selection of virtual memory architectures of which not all fulfill the requirements mentioned above.

[3] Compaq Computer Corporation: *Alpha Architecture Reference Manual*. January 2002

[4] Dave Probert, Microsoft: *Windows Kernel Architecture Internals*. April 2010. `http://research.microsoft.com/en-us/um/redmond/events/wincore2010/Dave_Probert_1.pdf`



(a) Traditional, without self-reference.



(b) With self-reference.

Figure 1: Page table walk in the x86_64 longmode.