Virtual Memory Management Evolution: From 80386 to Modern x86 Systems

MD. Abdul Momin Sheikh

Department of CSE, PUST

ID:210103

A key component of contemporary operating systems, virtual memory management allows for effective memory use, process isolation, and security. The development of virtual memory management in x86 systems is reviewed in this paper, beginning with the Intel 80386 processor, which was the first x86 CPU to implement paging, and moving on to more recent architectures. We look at important advancements in hardware-assisted virtualization, page table structures, paging algorithms, and security extensions. The study emphasizes how these developments have influenced contemporary computing by making features like robust memory protection, huge memory support, and effective address translation possible.

# I. INTRODUCTION

Processes can function as though they have exclusive access to a vast, continuous address space thanks to virtual memory, which abstracts physical memory. Since the 80386 was first released in 1985, the x86 architecture has experienced substantial modifications in terms of virtual memory management. This essay investigates: [4] The 80386’s fundamental paging system.

improvements in more recent x86 processors (such as the Pentium, AMD64, and Intel Core). 64-bit computing’s effect on address space.

virtualization aided by hardware (Intel VT-x, AMD-V). Extensions for security (NX bit, SMEP, SMAP, MPK) .[5]

## II. VIRTUAL MEMORY IN THE 80386

The Intel 80386 (i386), introduced in 1985, marked a significant milestone in x86 architecture by introducing hardware-supported paging while maintaining backward compatibility with segmentation. This section examines its virtual memory implementation.

*A. Paging Architecture*

The 80386 implemented a two-level hierarchical paging structure:

Page Directory: A single 4KB table containing 1,024 32bit entries (PDEs)

Page Tables: Second-level 4KB tables, each mapping

1,024 4KB pages

Address Translation: Linear address divided into:

10-bit Page Directory index

10-bit Page Table index

12-bit page offset (supporting 4KB granularity)

This structure enabled full 32-bit addressing (4GB virtual address space), a dramatic improvement over previous 16-bit processors.

1. *Memory Protection Features*

The 80386 introduced several protection mechanisms: Privilege Levels: Four protection rings (0-3) Page Attributes:

Read/Write permissions

User/Supervisor privilege separation

Present/Absent status for demand paging

Segmentation Persistence: While paging was optional, segmentation remained mandatory, creating a hybrid protection model.

1. *Performance Optimization*

Key performance features included:

Translation Lookaside Buffer (TLB):

Fully associative cache for page translations

Separate instruction and data TLBs

Automatic invalidation on page table updates

Write-Through Caching: Page directories and tables used a write-through policy for coherency

## III. ENHANCEMENTS IN LATER X86 PROCESSORS

Later x86 processors introduced significant improvements to virtual memory management, addressing limitations of the 80386 while scaling for modern workloads.

1. *Physical Address Extension (PAE) – Pentium Pro (1995)*

Extended physical addressing from 32-bit (4 GiB) → 36bit (64 GiB) while retaining 32-bit virtual addresses Introduced three-level paging:

Page Directory Pointer Table (PDPT)

Page Directory (PD)

Page Table (PT)

Added support for 2 MiB "large pages" (reducing TLB misses) [6].

1. *x86-64 (AMD64 / Intel EM64T) – 2003–2004*

Revolutionary Changes:

64-bit virtual addressing (initially 48-bit → 256 TiB, later extended).

Deprecated segmentation in long mode (simplified memory model).

Four-level paging (PML4 PDPT PD PT), with NX (No-

Execute) bit for security [1].

Added 1 GiB "huge pages" (reducing TLB pressure for large workloads).

Impact:

Enable modern OSes (Windows 64-bit, Linux x8664)

*C. Extended Page Tables (EPT) Virtualization – 2005–2006*

Intel VT-x AMD-V Innovations:

EPT (Intel) / NPT (AMD): Hardware-assisted physical translation (reducing VM exits) [3].

VPID (Virtual Processor ID): TLB tagging to avoid flushes during VM switches.

PCID (Process Context ID): Reduced TLB invalidations for process switches.

Use Case: Critical for cloud computing (AWS, Azure) and nested virtualization.

Note that the equation is centered using a center tab stop. Be sure that the symbols in your equation have been defined before or immediately following the equation. Use “(1)”, not “Eq. (1)” or “equation (1)”, except at the beginning of a sentence: “Equation (1) is...”

*D. Five-Level Paging (LA57) – 2016*

For 57-bit Virtual Addressing (128 PiB):

Added a 5th level (PML5) to the page table hierarchy [3]

Required for memory-intensive applications (e.g., inmemory databases, ML workloads). OS Support:

## IV. ARCHITECTURAL LIMITATIONS

Despite the continuous evolution of x86 virtual memory systems, several architectural limitations persist that impact performance, scalability, and security.

*A. Address Space Fragmentation*

* Fixed page sizes (4 KB, 2 MB, 1 GB) can lead to internal fragmentation.
* Memory allocations may not fully utilize physical memory, causing inefficient usage. [2]

*B. TLB Bottlenecks*

* Translation Lookaside Buffers (TLBs) cache page translations..
* TLB misses require multi-level page table walks, adding latency.
* Small TLB sizes can become performance bottlenecks for memory-intensive workloads.[7]

*C. Limited ASLR Effectiveness on 32-bit Systems*

* Address Space Layout Randomization (ASLR) is constrained by the limited 4 GB address space of 32-bit systems.
* This weakens defenses against memory corruption attacks like buffer overflows. [8]

# V. CONCLUSIONS

x86 virtual memory management offers a balance between legacy support and modern features. However, architectural constraints such as limited TLB size, lack of fine-grained control, and reliance on newer hardware for advanced features continue to pose challenges. Understanding these limitations is key to optimizing system design and planning future memory architectures.

# ACKNOWLEDGMENT

The authors would like to express their sincere gratitude to the following individuals and organizations for their contributions to this research:

Intel Corporation and Advanced Micro Devices (AMD) for making their architecture manuals and technical documentation publicly available, which formed the foundational references for this work.

The Linux Kernel Documentation Team for maintaining detailed records of x86 memory management evolution, which provided critical insights into practical implementations.

Reviewers of this manuscript for their constructive suggestions to improve technical accuracy and clarity.

The open-source community, whose collective work on operating systems (particularly Linux and BSD derivatives) has demonstrated real-world applications of the concepts discussed herein.

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

References are important to the reader; therefore, each citation must be complete and correct. If at all possible, references should be commonly available publications.

## REFERENCES

1. Advanced Micro Devices (AMD). Amd64 architecture programmer’s manual, volume 2: System programming. [https://www.amd.com/ system/files/TechDocs/24593.pdf,](https://www.amd.com/system/files/TechDocs/24593.pdf) 2022. Accessed: 2023-

10-01.

1. Intel Corporation. Intel® memory protection extensions (mpk). [https://wiki.osdev.org/Paging,](https://wiki.osdev.org/Paging) 2015. Accessed: 2023-10-

01.

1. Intel Corporation. 5-level paging and 5-level ept. [https:](https://www.intel.com/content/www/us/en/developer/overview.html#gs.lni8bf)

[//www.intel.com/content/www/us/en/developer/ overview.html#gs.lni8bf,](https://www.intel.com/content/www/us/en/developer/overview.html#gs.lni8bf) 2016. Accessed: 2023-10-01.

1. Intel Corporation. Intel vt-x technology. [https://pdos.csail. mit.edu/6.828/2018/readings/i386/toc.htm,](https://pdos.csail.mit.edu/6.828/2018/readings/i386/toc.htm) 2023. Accessed: 2023-10-01.
2. Intel Corporation. Intel vt-x technology. [https://pdos.csail. mit.edu/6.828/2018/readings/i386/toc.htm,](https://pdos.csail.mit.edu/6.828/2018/readings/i386/toc.htm) 2023. Accessed: 2023-10-01.
3. Intel Corporation. Intel vt-x technology. [https://www. intel.com/content/www/us/en/developer/articles/ technical/intel-sdm.html,](https://www.intel.com/content/www/us/en/developer/articles/technical/intel-sdm.html) 2023. Accessed: 2023-10-01.
4. Intel Corporation. Intel® 64 and ia-32 architectures software developer’s manual (vol. 3). [https://www.intel.com/ content/www/us/en/developer/articles/technical/ intel-sdm.html,](https://www.intel.com/content/www/us/en/developer/articles/technical/intel-sdm.html) 2023. Accessed: 2023-10-01.
5. PaX Team. Address space layout randomization (aslr). [https:// pax.grsecurity.net/docs/aslr.txt,](https://pax.grsecurity.net/docs/aslr.txt) 2015. Accessed: 2023-

10-05.