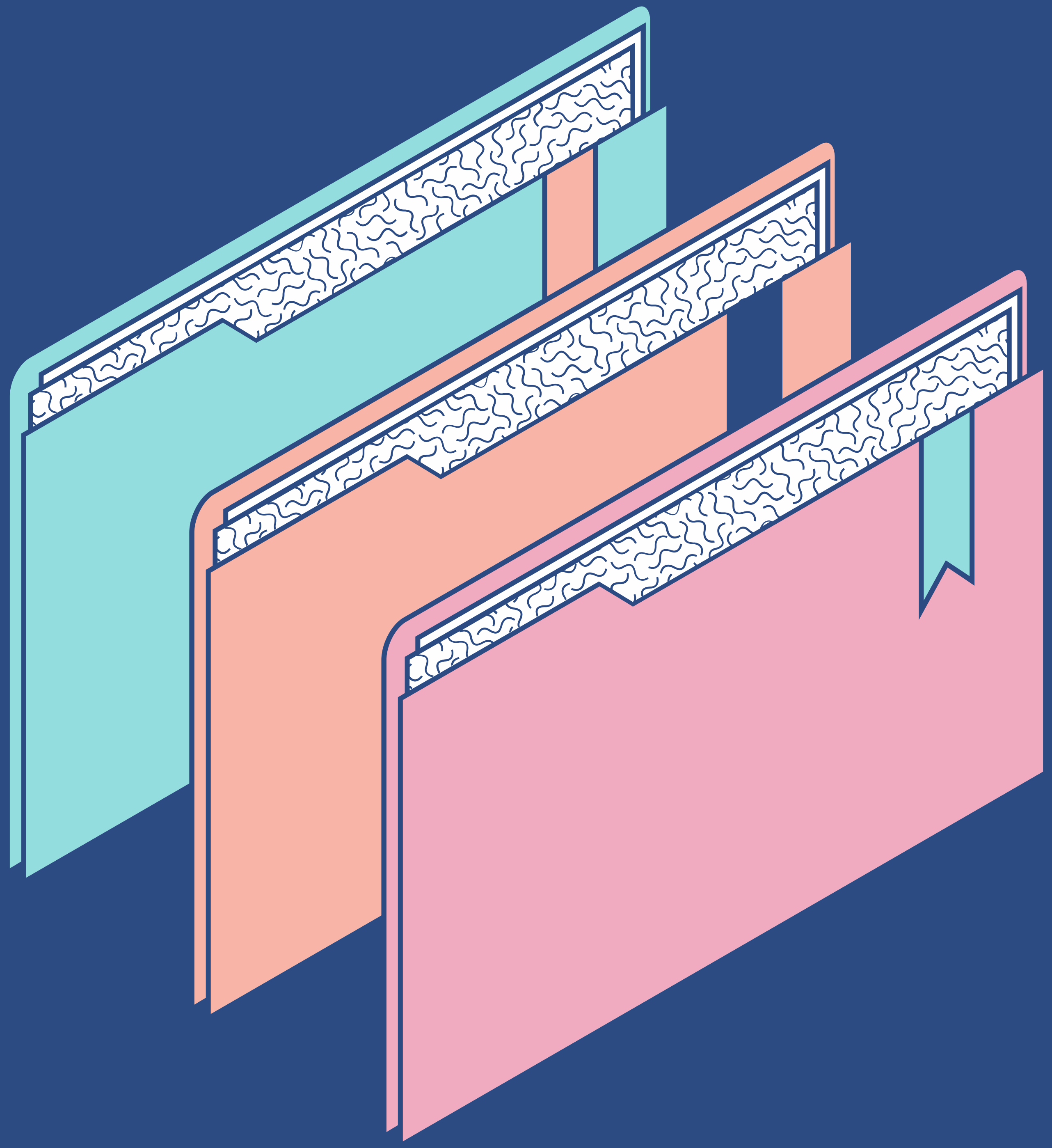
NILE UNIVERSITY

VIRTUAL MEMORY

Agenda

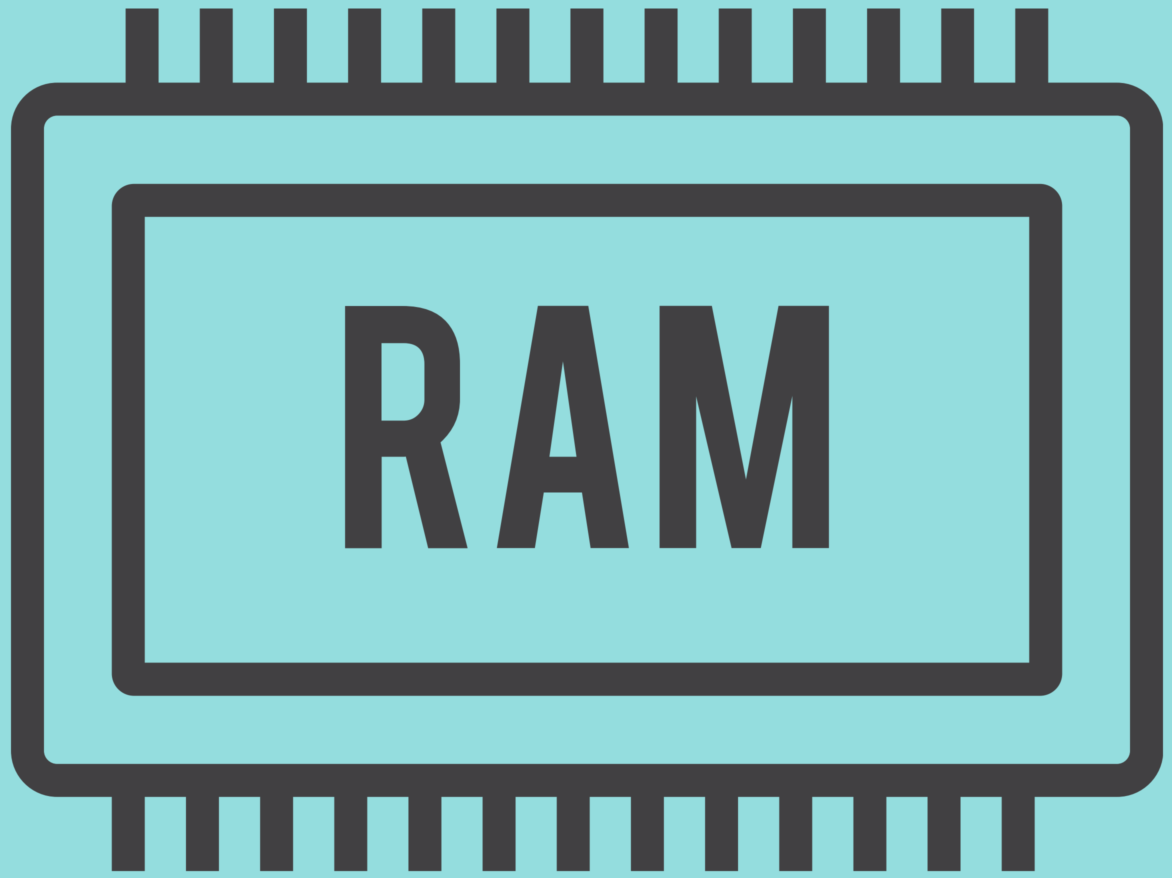
KEY TOPICS DISCUSSED IN THIS PRESENTATION

- INTRODUCTION
- PROJECT GOALS
- VIRTUAL MEMORY DIAGRAM
- IMPLEMENTATION
- RESULTS
- POWER OF VIRTUAL MEMORY
- CONCLUSION



INTRODUCTION

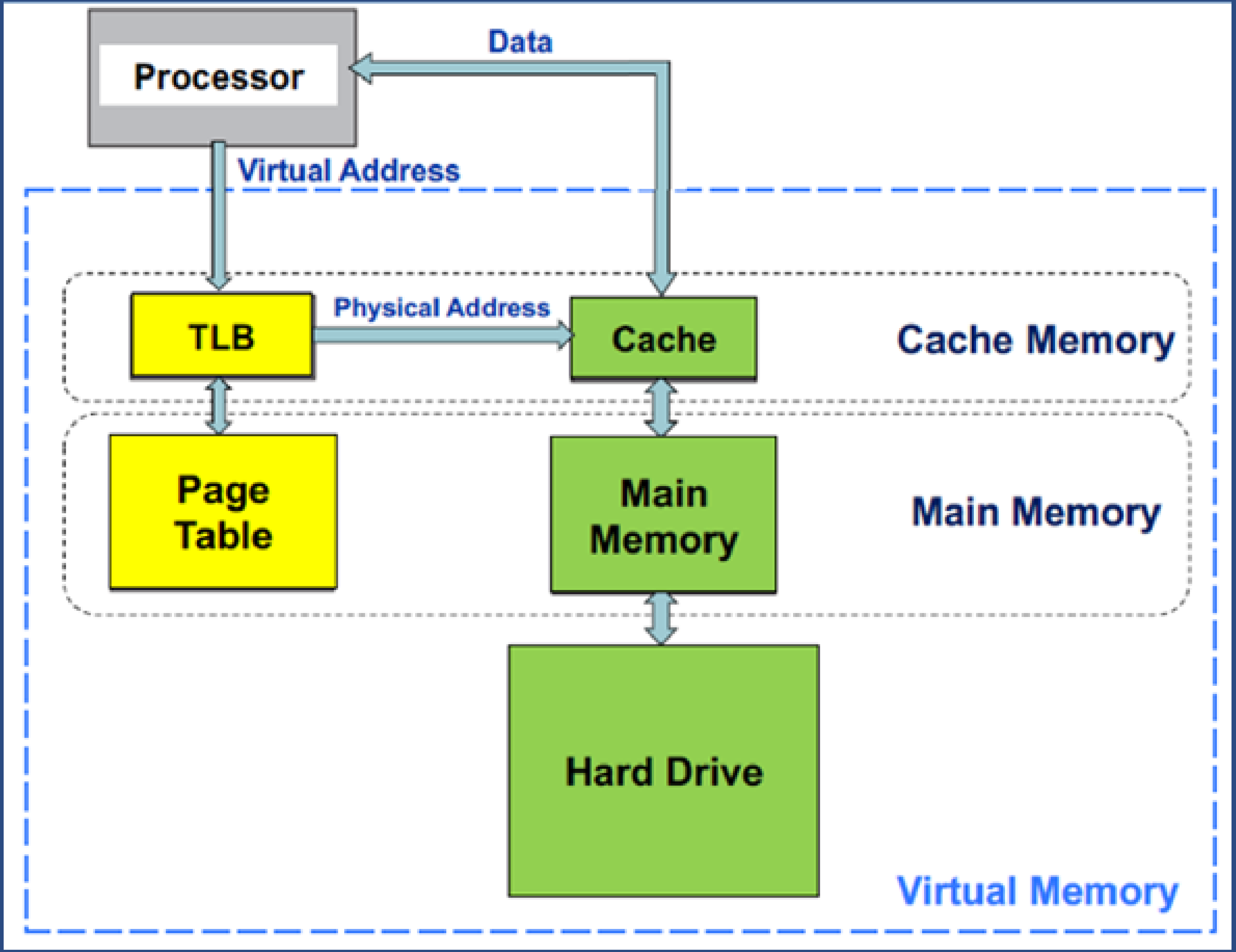
- Virtual memory is a technique used by operating systems.
- It uses a combination of RAM and disk space.
- When RAM is full, data is automatically moved to the disk.
- This frees up space in RAM for other data.
- Virtual memory makes it seem like there is more RAM than there actually is.
- The page table helps translate addresses between RAM and the disk.
- Virtual memory helps programs run even with limited physical memory.



PROJECT GOALS

- Implement a virtual memory system using Vivado Verilog.
- Design and simulate a page table data structure to manage virtual-to-physical address translations.
- Develop a paging mechanism that automatically transfers data between RAM and disk when memory is full, and handle page faults by retrieving data from the disk when needed.

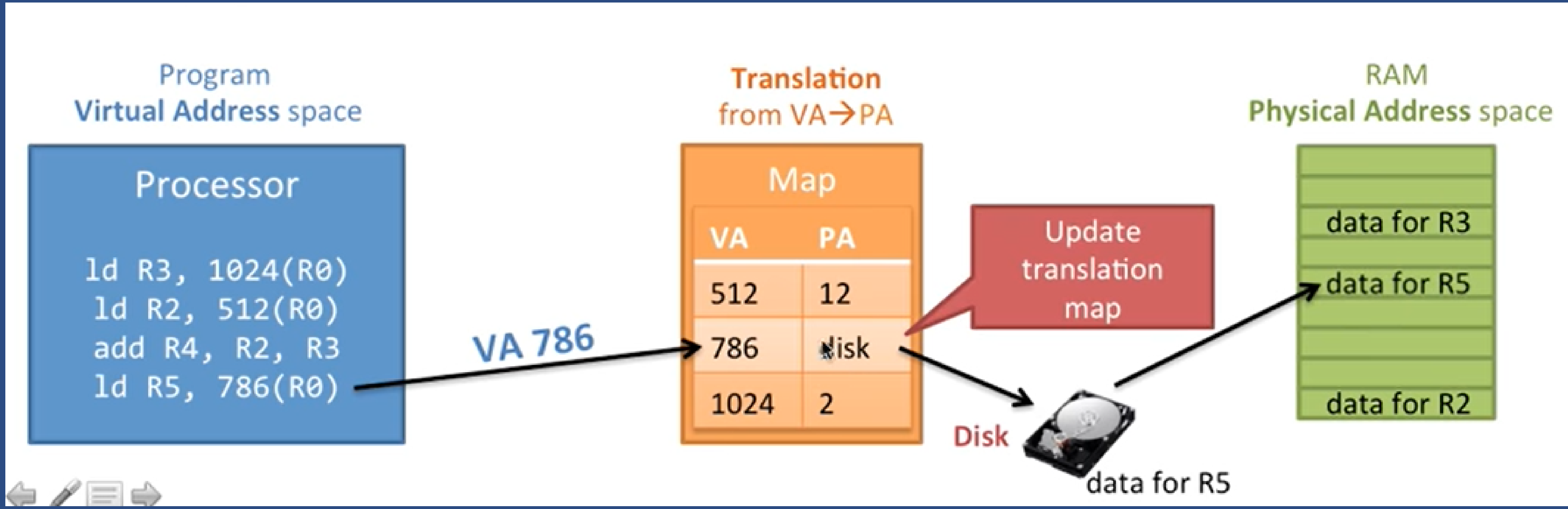
VIRTUAL MEMORY DIAGRAM



VIRTUAL MEMORY IMPLEMENTATION

PAGE TABLE

- THE PAGE TABLE MAPS VIRTUAL ADDRESSES TO PHYSICAL ADDRESSES IN VIRTUAL MEMORY SYSTEMS.
- It enables efficient address translation between virtual and physical memory.
- The page table is dynamically updated to reflect changes in memory mappings due to page faults or memory management operations.

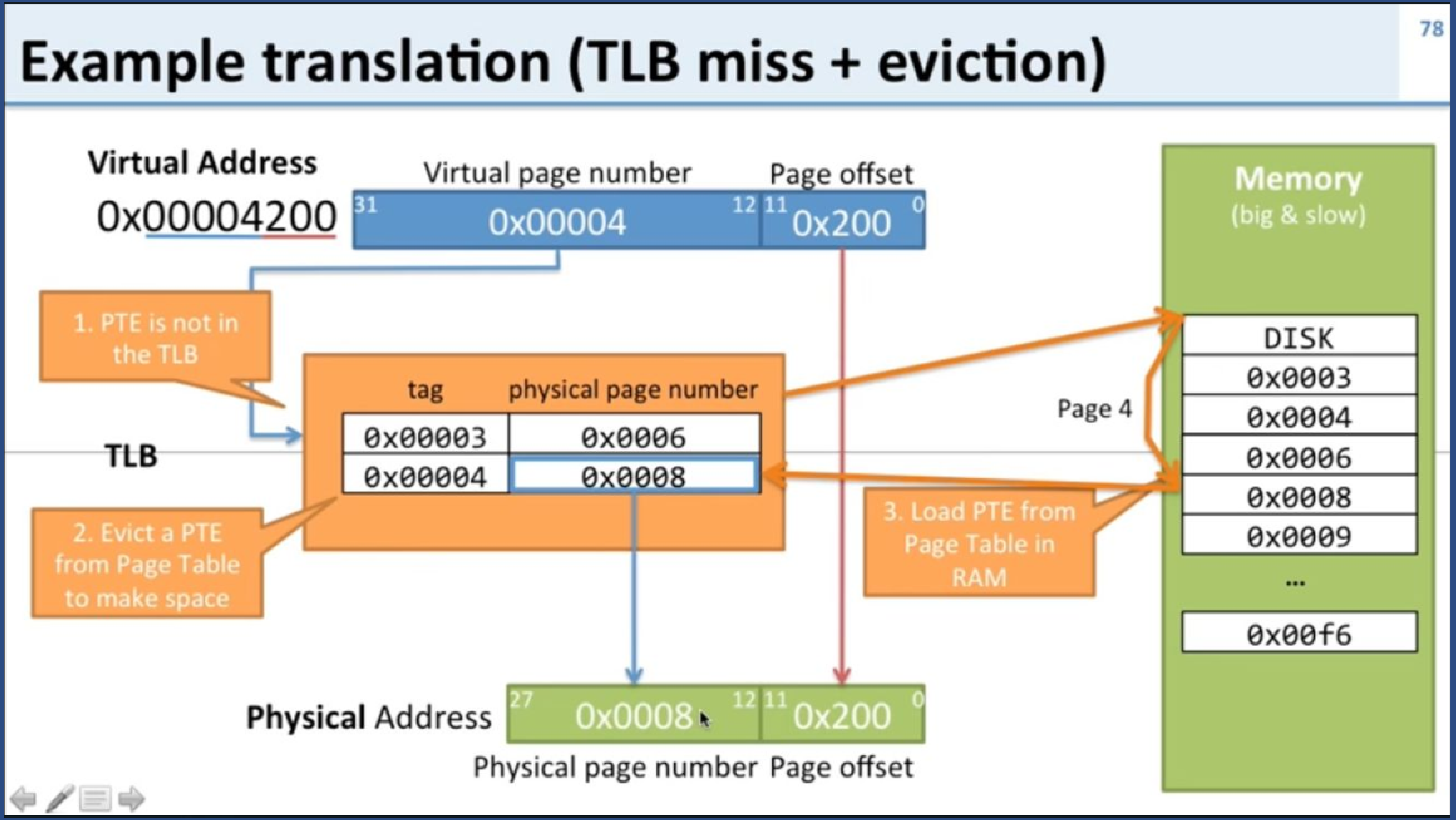


```
ld R3, 1024(R0)
ld R2, 512(R0)
add R4, R2, R3
ld R5, 786(R0)
```

VA	PA
512	12
786	disk
1024	2

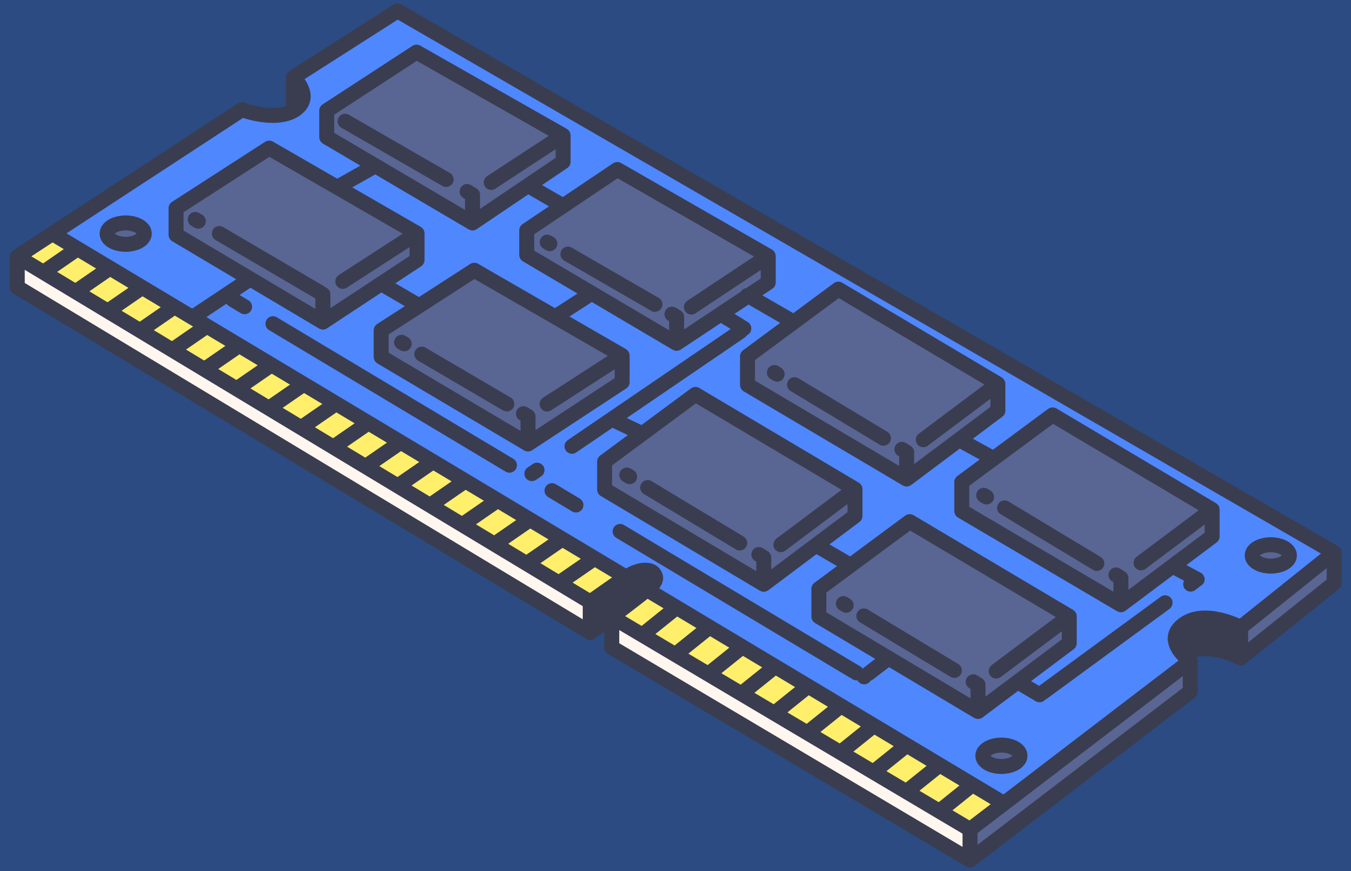
TLB

- THE TLB IS A CACHE THAT STORES RECENTLY USED VIRTUAL-TO-PHYSICAL ADDRESS TRANSLATIONS.
- IT PROVIDES FASTER ADDRESS TRANSLATION COMPARED TO THE PAGE TABLE BY AVOIDING FREQUENT ACCESSES TO THE PAGE TABLE.
- THE TLB HAS A LIMITED SIZE AND CAN ONLY STORE A SUBSET OF TRANSLATIONS, RESULTING IN OCCASIONAL TLB MISSES THAT REQUIRE ACCESSING THE PAGE TABLE.



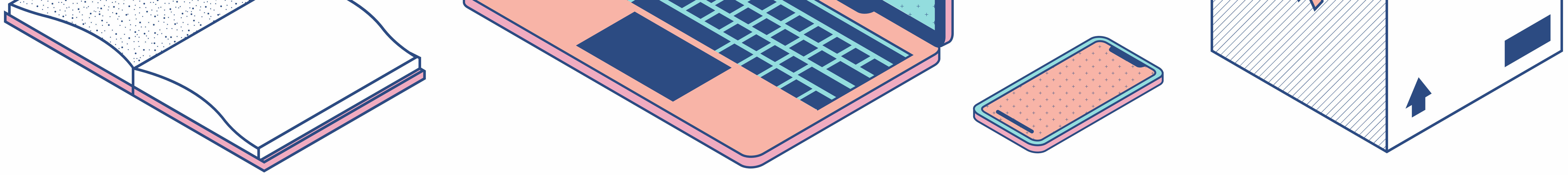
MAIN MEMORY

- THE MAIN_MEMORY MODULE REPRESENTS THE PRIMARY MEMORY (RAM) IN A COMPUTER SYSTEM.
- IT STORES DATA IN A MEMORY ARRAY (MEMORY) WITH 256 BLOCKS, EACH BLOCK CONSISTING OF TWO 32-BIT WORDS
- THE MAIN_MEMORY MODULE PERFORMS WRITE AND READ OPERATIONS FOR MEMORY ACCESS BASED ON SPECIFIED INPUTS.



CACHE

- IT IS RESPONSIBLE FOR HANDLING CACHE OPERATIONS, INCLUDING READ AND WRITE OPERATIONS.
- THE CACHE MODULE USES A TWO-WAY SET-ASSOCIATIVE CACHE DESIGN WITH INDEXING, TAGGING, AND DATA STORAGE ARRAYS.
- THE CACHE MODULE INTERACTS WITH THE MAIN MEMORY MODULE TO FETCH DATA IN CASE OF CACHE MISSES.



TEST FILE

- The testbench simulates various scenarios and operations on the cache.
- It performs read and write operations with different addresses and data.
- After each operation, it displays relevant information, such as the address, read/write status, and memory content.

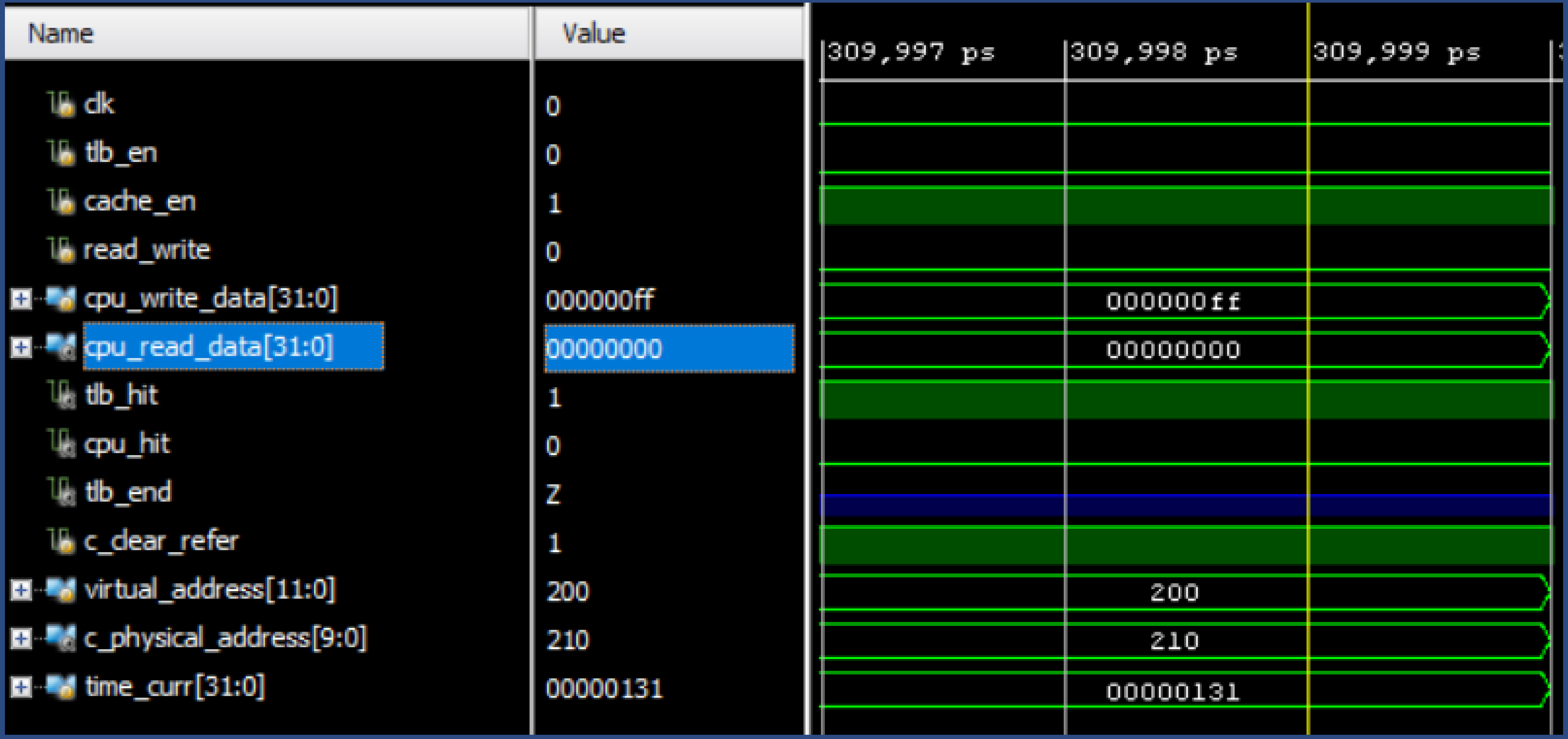
SIM FILE

- The code simulates a memory hierarchy consisting of a Translation Lookaside Buffer (TLB) and a cache.
- It includes a testbench that performs read and write operations, checks cache hits and misses, and displays main memory content after each operation.
- The simulation helps verify the correctness of the memory hierarchy, explores different cache configurations (direct-mapped or two-way associative), and examines various memory access patterns.

RESULTS

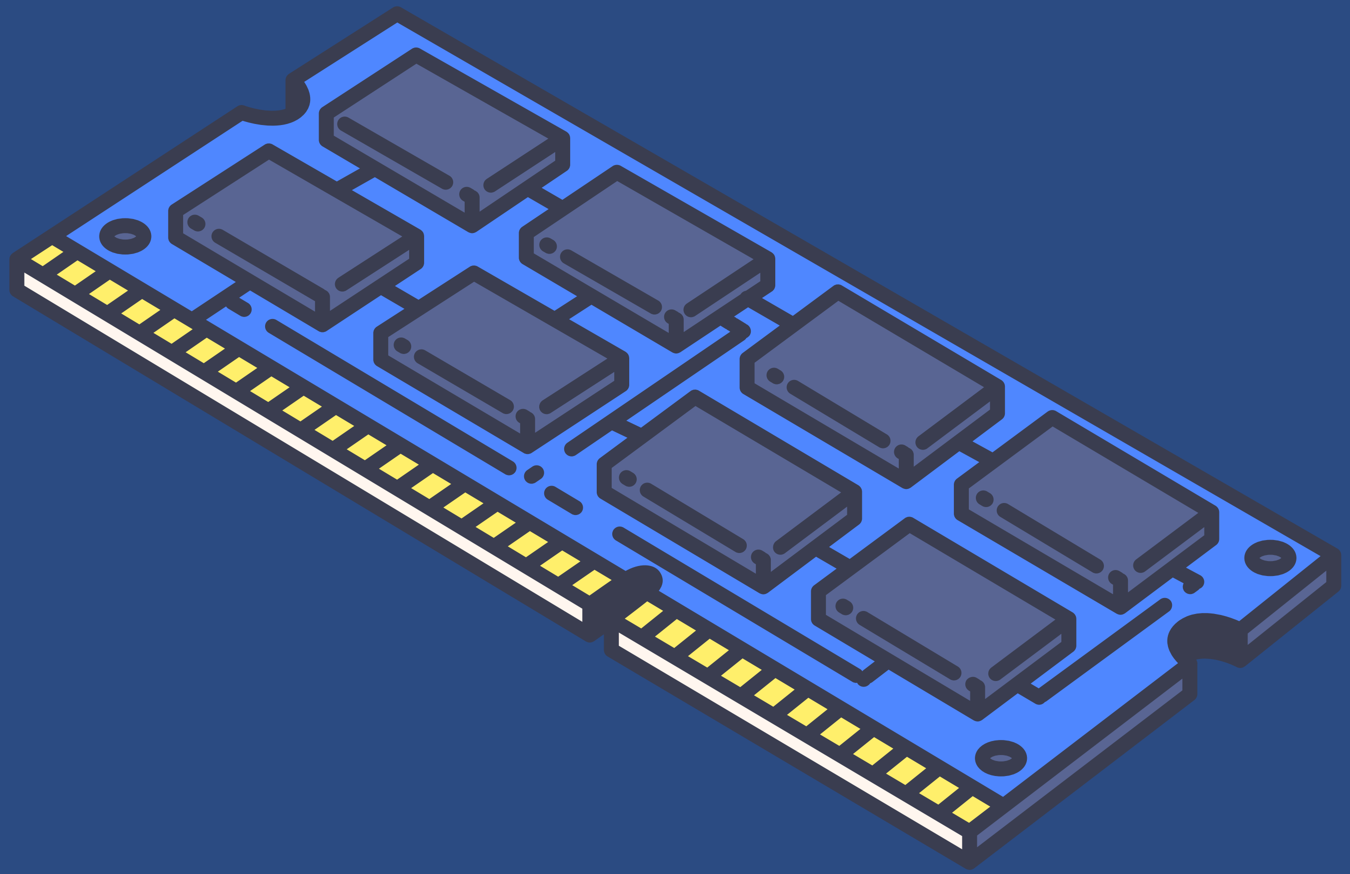
- TEST 1: ADDRESS 0000000000 (READ OPERATION)
- EXPECTED RESULT: CACHE MISS
- ACTUAL RESULT: CACHE MISS
- CONCLUSION: THE CACHE CORRECTLY IDENTIFIED THE ADDRESS AS A CACHE MISS.

- TEST 2: ADDRESS 0000000000 (WRITE OPERATION)
- EXPECTED RESULT: CACHE HIT
- ACTUAL RESULT: CACHE HIT
- CONCLUSION: THE CACHE SUCCESSFULLY IDENTIFIED THE ADDRESS AS A CACHE HIT AND UPDATED THE CACHE WITH THE NEW DATA.



Name	Value
clk	0
tlb_en	0
cache_en	1
read_write	0
cpu_write_data[31:0]	000000ff
cpu_read_data[31:0]	00000000
tlb_hit	1
cpu_hit	0
tlb_end	Z
c_clear_refer	1
virtual_address[11:0]	200
c_physical_address[9:0]	210
time_curr[31:0]	00000131

POWER OF VIRTUAL MEMORY



Limitless Storage Potential

Virtual memory breaks free from physical constraints, storing vast amounts of data effortlessly. It opens doors to new possibilities and ambitious projects. The system exemplifies this power by efficiently storing and accessing extensive data sets.

Exceeding Expectations

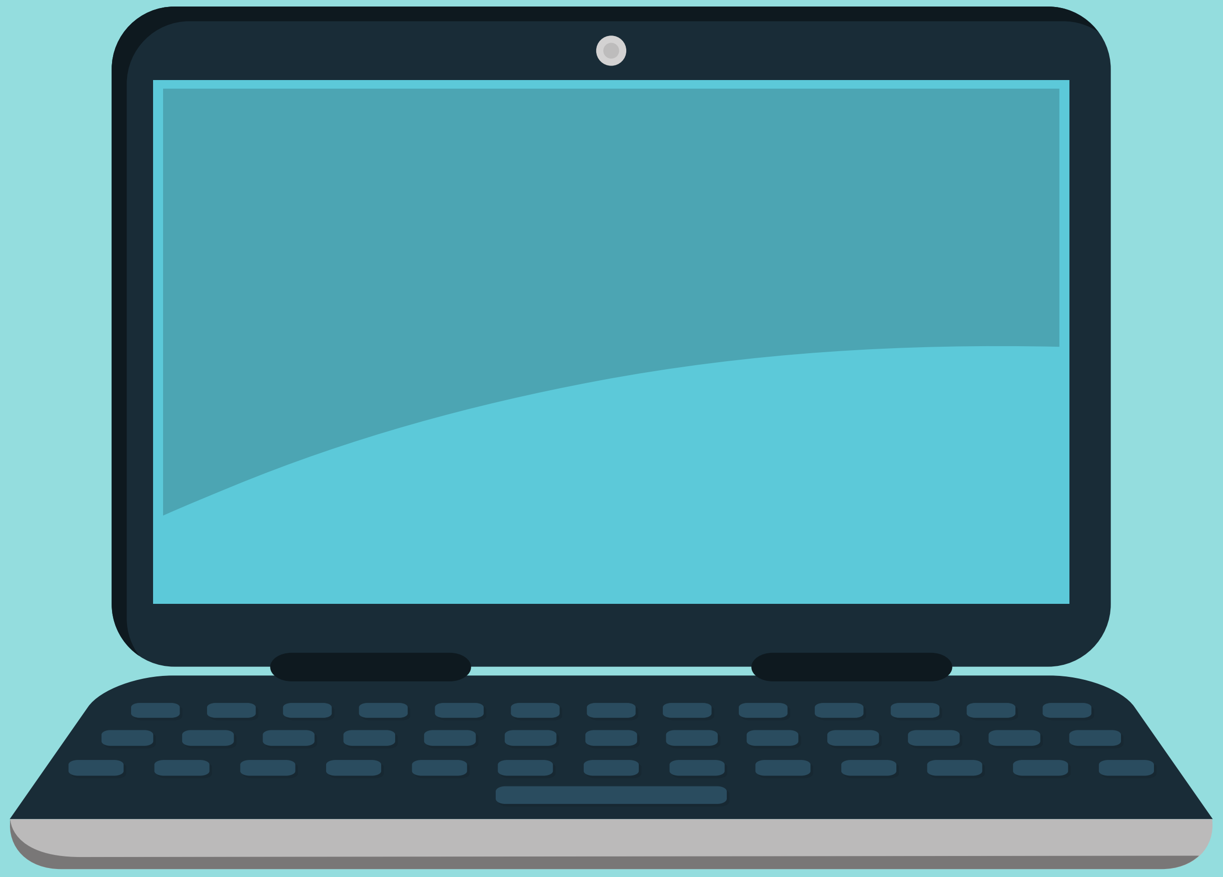
The system's impressive performance, functionality, and seamless user experiences have been validated through simulation, testing, and TLB utilization.

TLB Cache: A Technological Marvel

The TLB cache has revolutionized system efficiency and speed with lightning-fast address translation. This advancement highlights the potential for further optimizations in virtual memory systems.

CONCLUSION

In conclusion, implementing virtual memory using Verilog is a complex but essential process for improving computer performance. It involves multiple files working together to manage memory efficiently. Virtual memory is now a standard feature in modern computers, and understanding its implementation is crucial for computer engineers. As the demand for faster and more efficient systems grows, virtual memory will remain vital in shaping the future of computing.



Do you have any questions?