Virtual Memory

EEL 3713C: Digital Computer Architecture

Quincy Flint

[Ionospheric Radio Lab in NEB]

Outline

1. Memory Problems
   - Not enough memory
   - Holes in address space
   - Programs overwriting
2. What is Virtual Memory?
   - Layer of indirection
   - How does indirection solve above
   - Page tables and translation
3. How do we implement VM?
   - Create and store page tables
   - Fast address translation
4. Virtual Memory and Caches
   - Prevent cache performance degradation when using VM

Multi-Level Page Tables

Page Table Size… again

Page Table Size… again

- Given a 32-bit machine, 4 kB pages…

Page Table Size… again

- Given a 32-bit machine, 4 kB pages…
  - 1 million PTEs
    - 32 bits (machine) – 12 bits (page offset) = 20 bits = 1 million

Page Table Size… again

- Given a 32-bit machine, 4 kB pages…
  - 1 million PTEs
    - 32 bits (machine) – 12 bits (page offset) = 20 bits = 1 million
  - Each PTE is around 4 Bytes wide
    - 20 bits (Physical Page) + permission bits

Page Table Size… again

- Given a 32-bit machine, 4 kB pages…
  - 1 million PTEs
    - 32 bits (machine) – 12 bits (page offset) = 20 bits = 1 million
  - Each PTE is around 4 Bytes wide
    - 20 bits (Physical Page) + permission bits

1M entries x 4 Bytes per entry
= 4 MB page tables

Page Table Size… again

- Given a 32-bit machine, 4 kB pages…
  - 1 million PTEs
    - 32 bits (machine) – 12 bits (page offset) = 20 bits = 1 million
  - Each PTE is around 4 Bytes wide
    - 20 bits (Physical Page) + permission bits

1M entries x 4 Bytes per entry
= 4 MB page tables

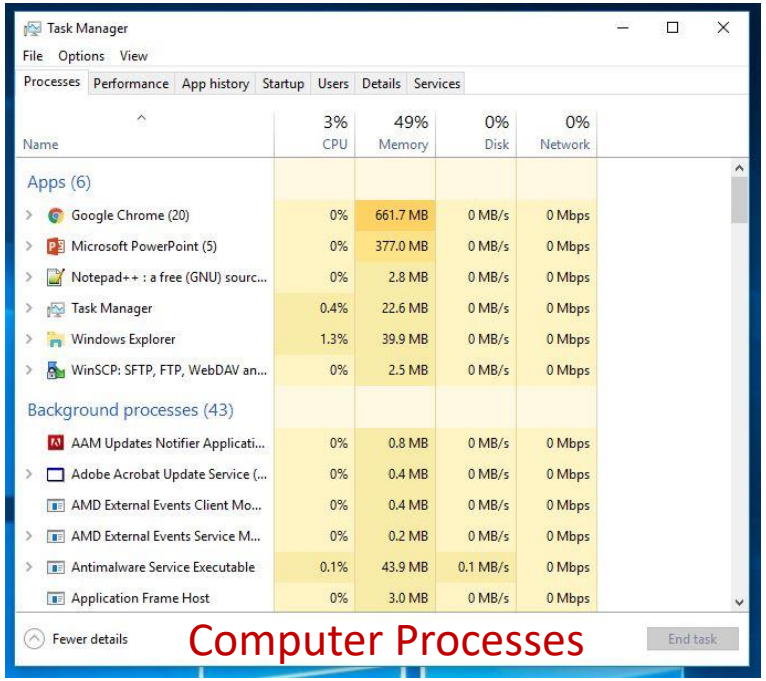
- BUT, each program needs its own page table

Page Table Size… again

- Given a 32-bit machine, 4 kB pages…
  - 1 million PTEs
    - 32 bits (machine) – 12 bits (page offset) = 20 bits = 1 million
  - Each PTE is around 4 Bytes wide
    - 20 bits (Physical Page) + permission bits

1M entries x 4 Bytes per entry
= 4 MB page tables

- BUT, each program needs its own page table



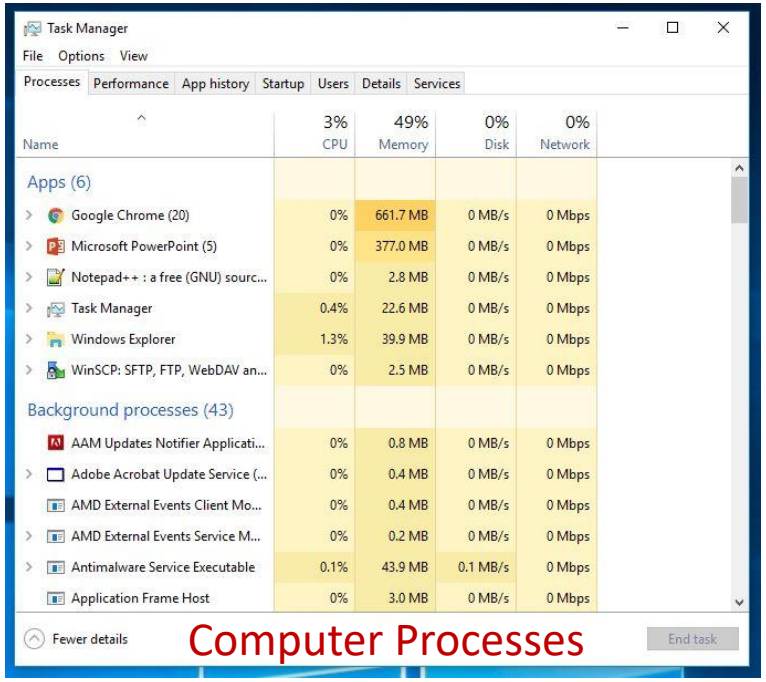
Page Table Size… again

- Given a 32-bit machine, 4 kB pages…
  - 1 million PTEs
    - 32 bits (machine) – 12 bits (page offset) = 20 bits = 1 million
  - Each PTE is around 4 Bytes wide
    - 20 bits (Physical Page) + permission bits

- BUT, each program needs its own page table

1M entries x 4 Bytes per entry
= 4 MB page tables

4 MB page tables x 50 processes
= 200 MB of RAM for PTs



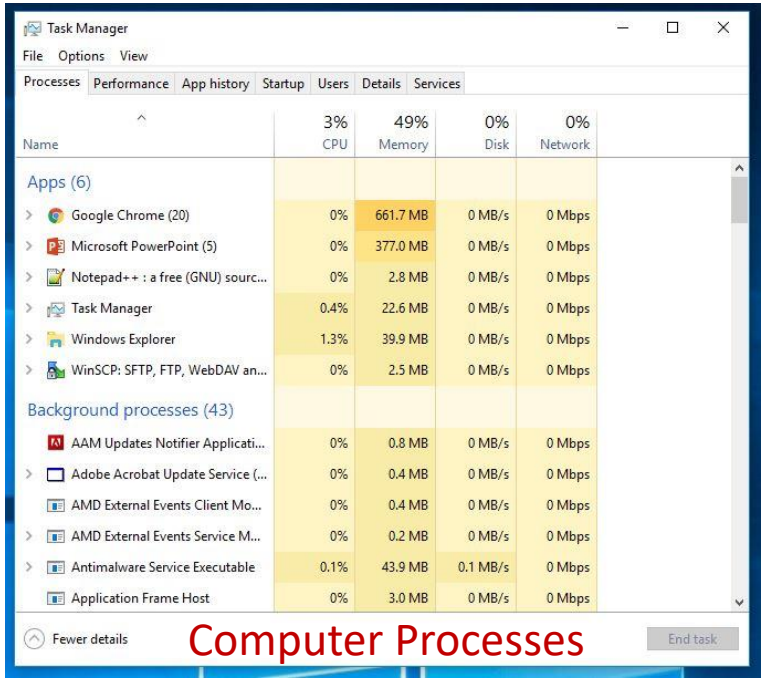
Page Table Size… again

- Given a 32-bit machine, 4 kB pages…
  - 1 million PTEs
    - 32 bits (machine) – 12 bits (page offset) = 20 bits = 1 million
  - Each PTE is around 4 Bytes wide
    - 20 bits (Physical Page) + permission bits

- BUT, each program needs its own page table
  - 100 programs require 400 MB of RAM

1M entries x 4 Bytes per entry
= 4 MB page tables

4 MB page tables x 50 processes
= 200 MB of RAM for PTs



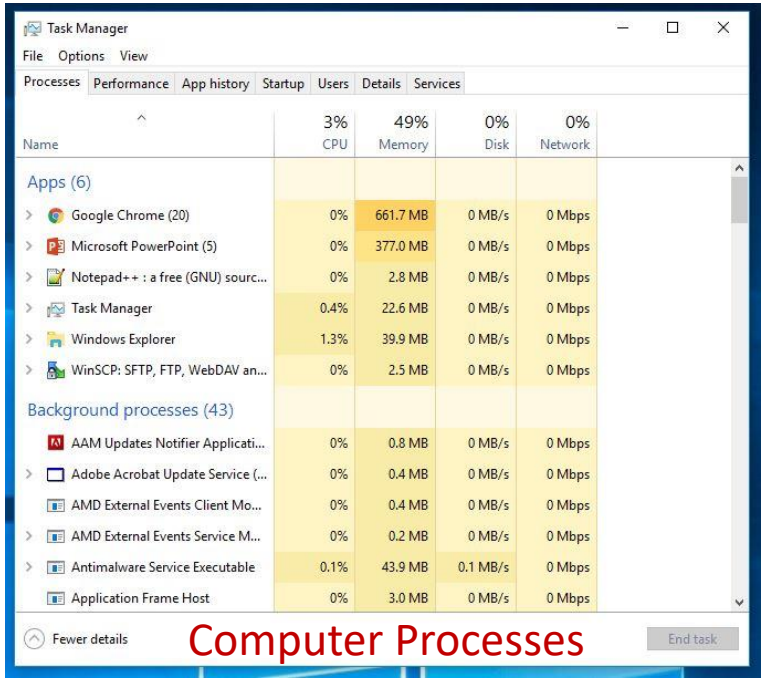
Page Table Size… again

- Given a 32-bit machine, 4 kB pages…
  - 1 million PTEs
    - 32 bits (machine) – 12 bits (page offset) = 20 bits = 1 million
  - Each PTE is around 4 Bytes wide
    - 20 bits (Physical Page) + permission bits

- BUT, each program needs its own page table
  - 100 programs require 400 MB of RAM
  - Cannot swap pages tables to disk (normally)

1M entries x 4 Bytes per entry
= 4 MB page tables

4 MB page tables x 50 processes
= 200 MB of RAM for PTs



Page Table Size… again

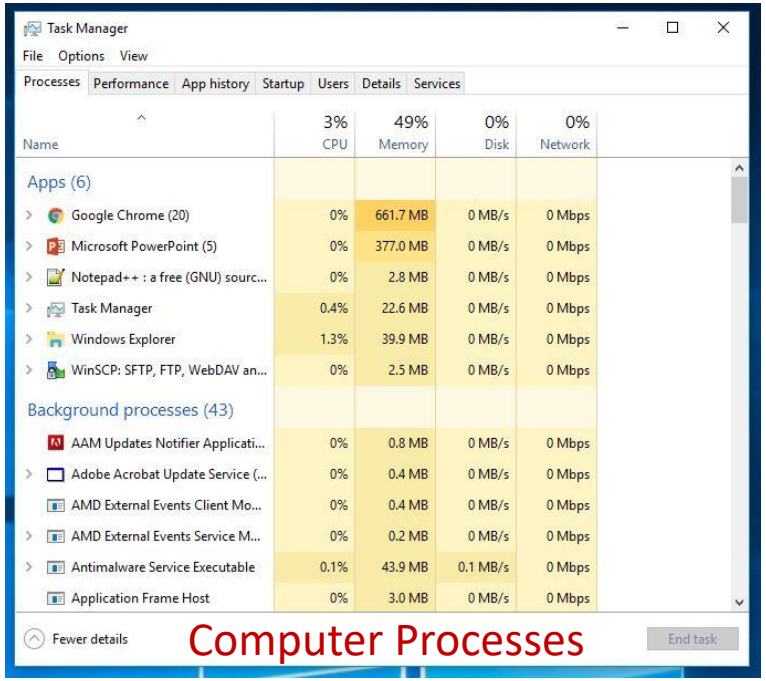
- Given a 32-bit machine, 4 kB pages…
  - 1 million PTEs
    - 32 bits (machine) – 12 bits (page offset) = 20 bits = 1 million
  - Each PTE is around 4 Bytes wide
    - 20 bits (Physical Page) + permission bits

- BUT, each program needs its own page table
  - 100 programs require 400 MB of RAM
  - Cannot swap pages tables to disk (normally)

- How can we fix this?

1M entries x 4 Bytes per entry
= 4 MB page tables

4 MB page tables x 50 processes
= 200 MB of RAM for PTs



Page Table Size… again

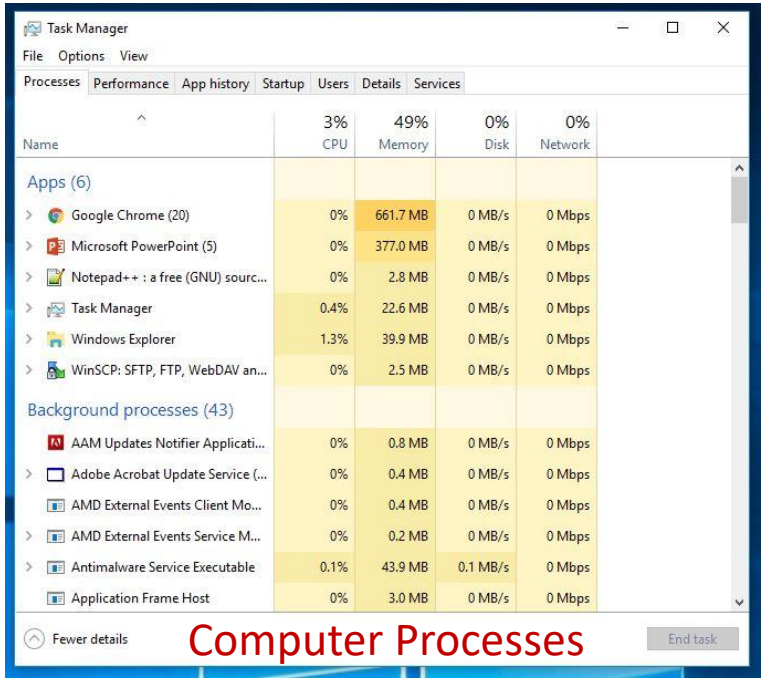
- Given a 32-bit machine, 4 kB pages…
  - 1 million PTEs
    - 32 bits (machine) – 12 bits (page offset) = 20 bits = 1 million
  - Each PTE is around 4 Bytes wide
    - 20 bits (Physical Page) + permission bits

- BUT, each program needs its own page table
  - 100 programs require 400 MB of RAM
  - Cannot swap pages tables to disk (normally)

- How can we fix this? Indirection!

1M entries x 4 Bytes per entry
= 4 MB page tables

4 MB page tables x 50 processes
= 200 MB of RAM for PTs

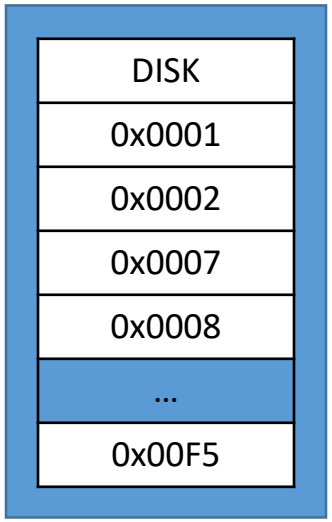


Multi-Level Page Tables

1st Level Page Table

4kB [1,024 entries]

Size of **1** Page



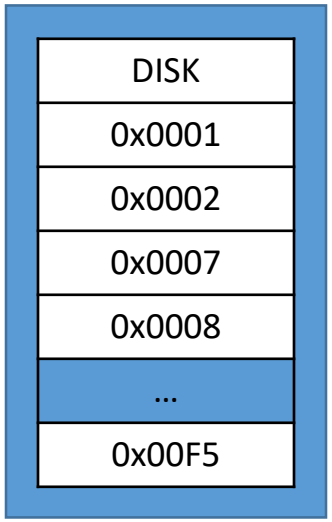
Main Memory

Multi-Level Page Tables

1st Level Page Table
4kB [1,024 entries]

Size of **1** Page

2nd Level Page Table
4kB each [1,024 entries]

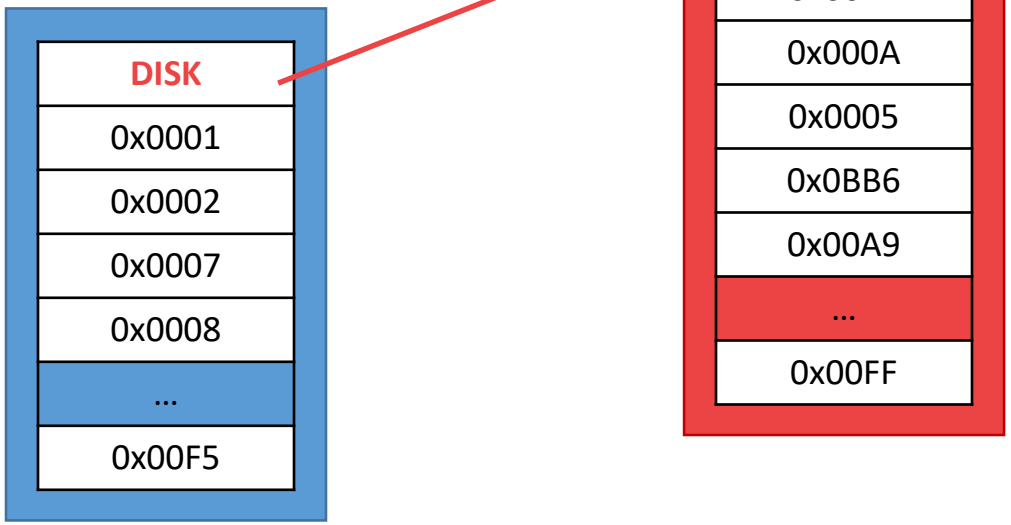


Main Memory

Multi-Level Page Tables

1st Level Page Table
4kB [1,024 entries]

Size of **1** Page



2nd Level Page Table
4kB each [1,024 entries]

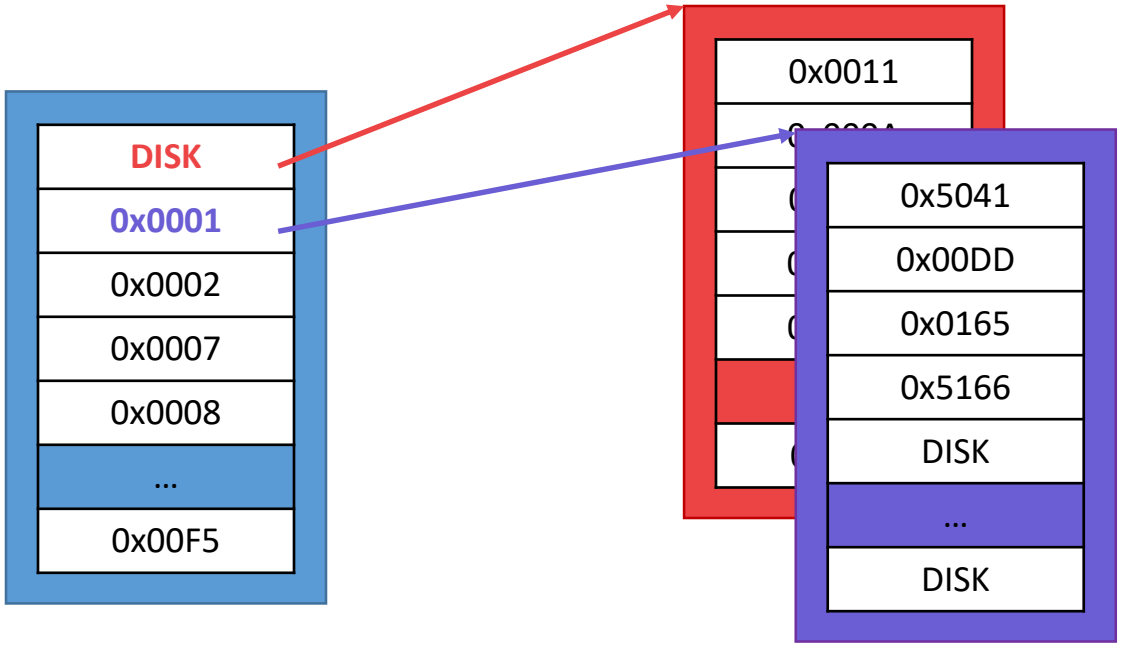
Main Memory

Multi-Level Page Tables

1st Level Page Table
4kB [1,024 entries]

Size of **1** Page

2nd Level Page Table
4kB each [1,024 entries]



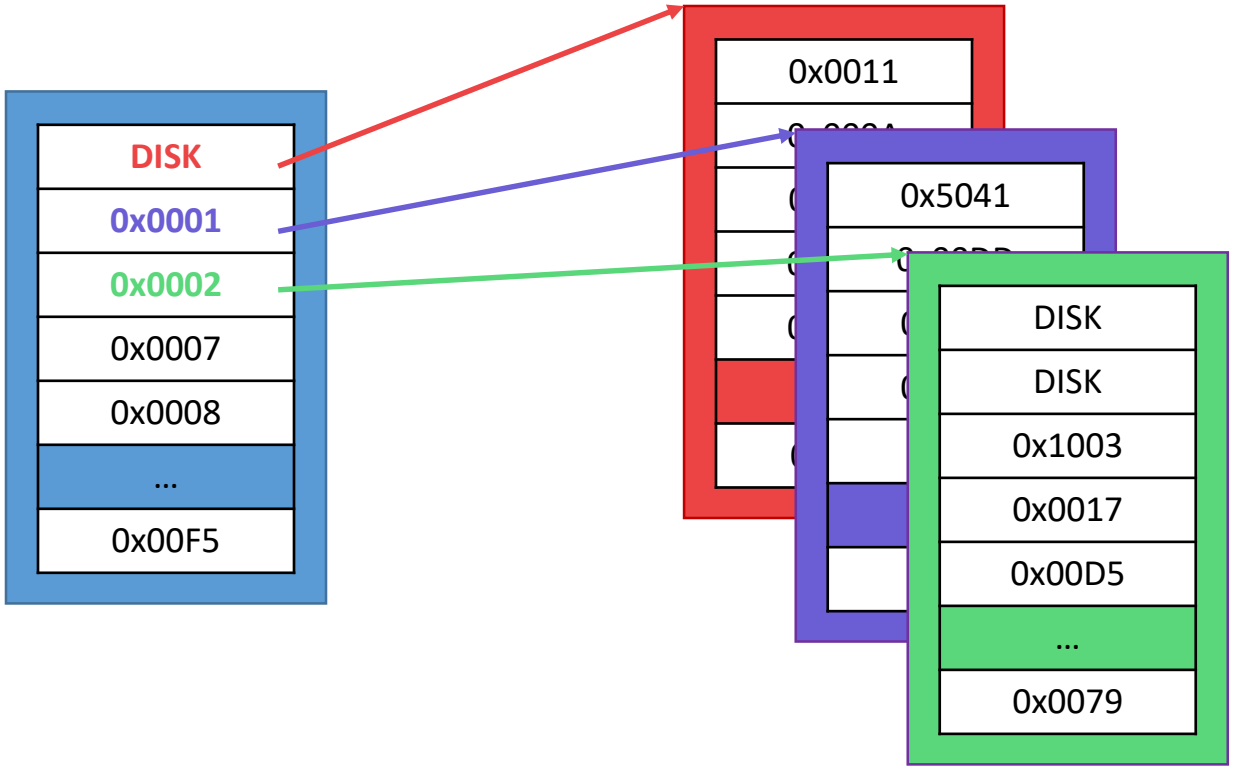
Main Memory

Multi-Level Page Tables

1st Level Page Table
4kB [1,024 entries]

Size of **1** Page

2nd Level Page Table
4kB each [1,024 entries]

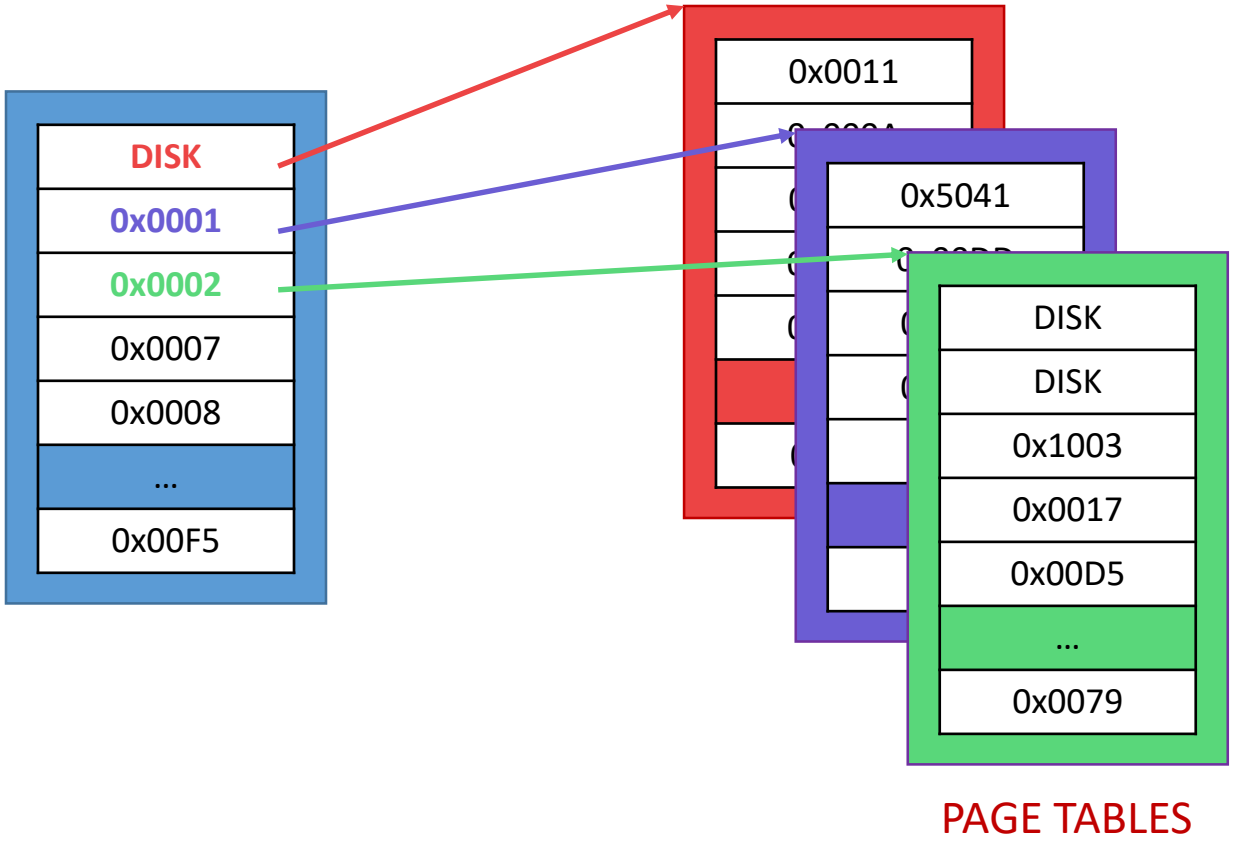


Main Memory

Multi-Level Page Tables

1st Level Page Table
4kB [1,024 entries]

Size of **1** Page



2nd Level Page Table
4kB each [1,024 entries]

PAGE TABLES

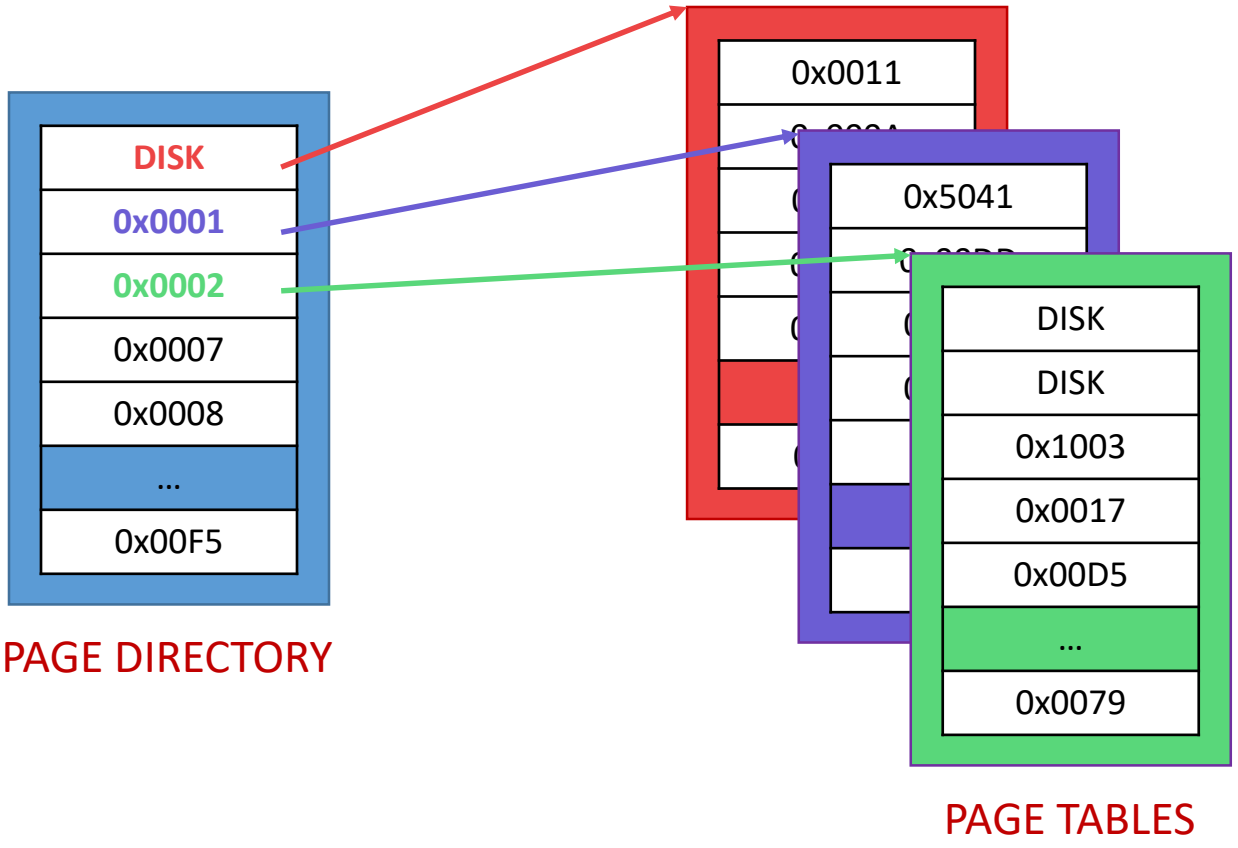
Main Memory

Multi-Level Page Tables

1st Level Page Table
4kB [1,024 entries]

Size of **1** Page

2nd Level Page Table
4kB each [1,024 entries]



PAGE DIRECTORY

PAGE TABLES

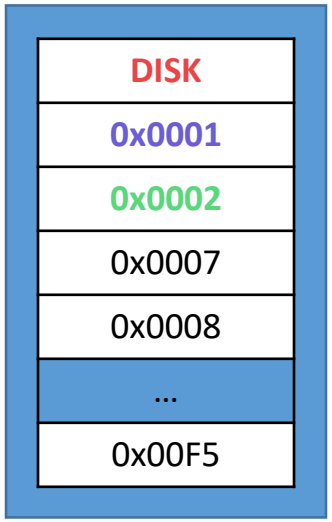
Main Memory

Multi-Level Page Tables

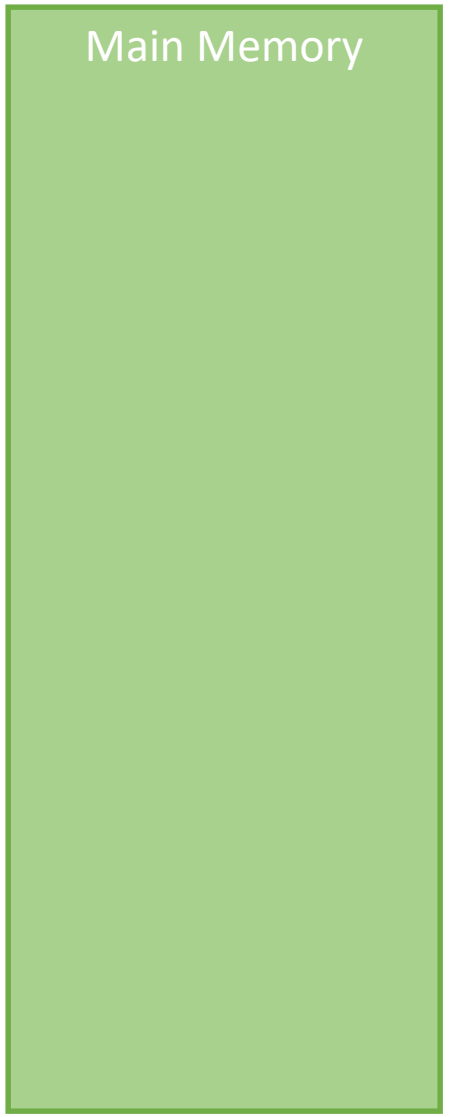
1st Level Page Table
4kB [1,024 entries]

Size of **1** Page

2nd Level Page Table
4kB each [1,024 entries]



PAGE DIRECTORY

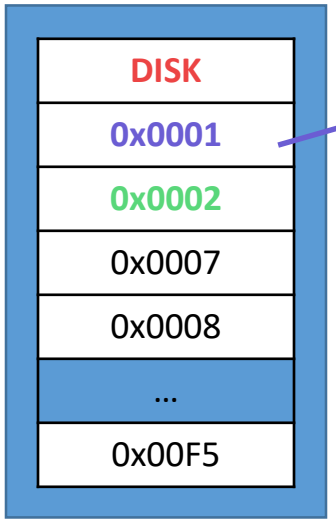


Multi-Level Page Tables

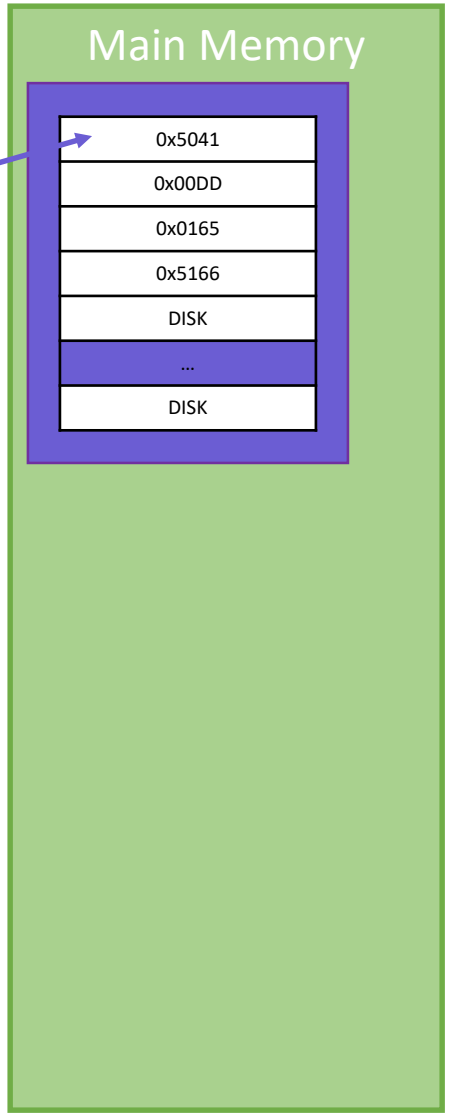
1st Level Page Table
4kB [1,024 entries]

Size of **1** Page

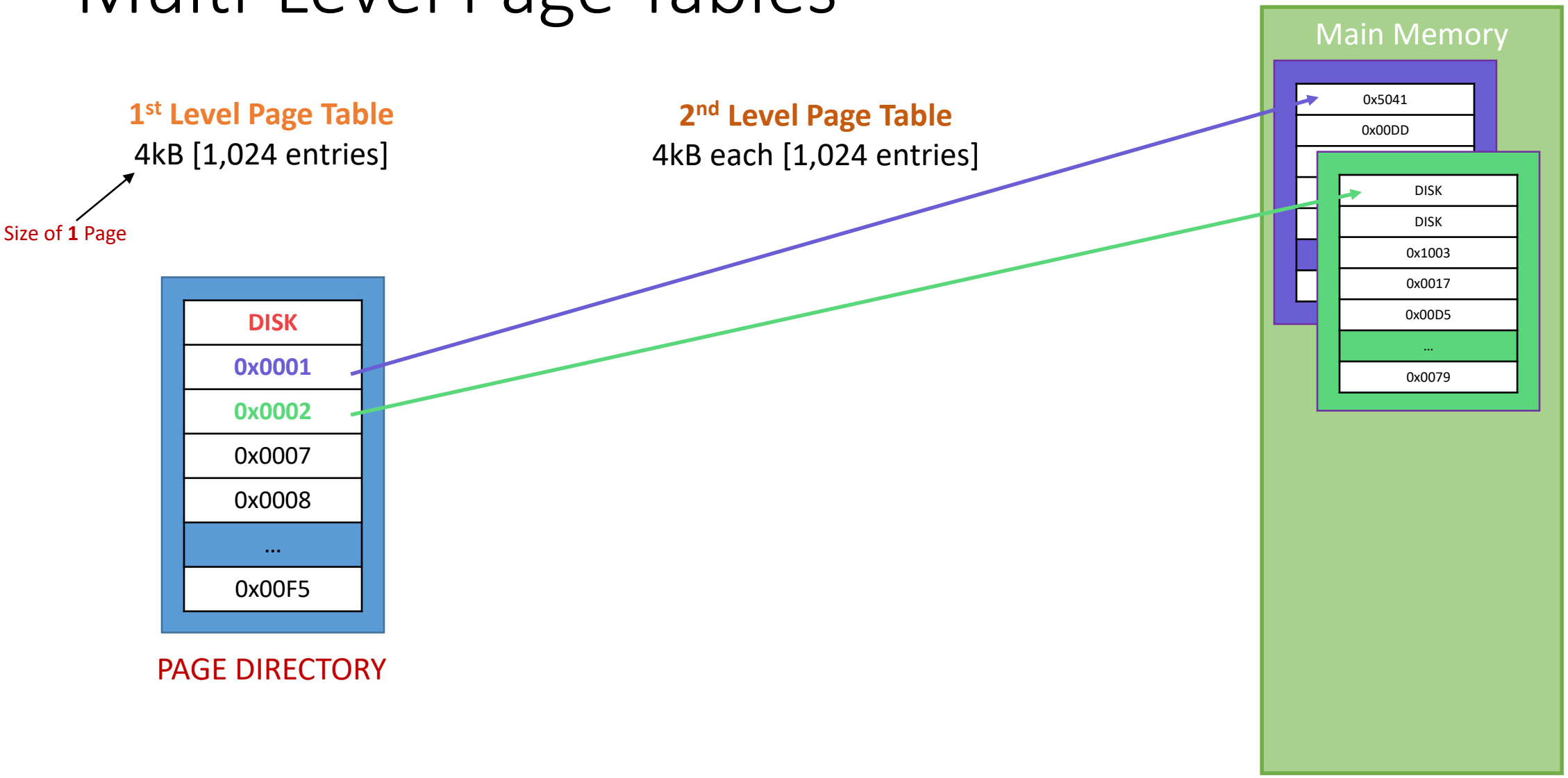
2nd Level Page Table
4kB each [1,024 entries]



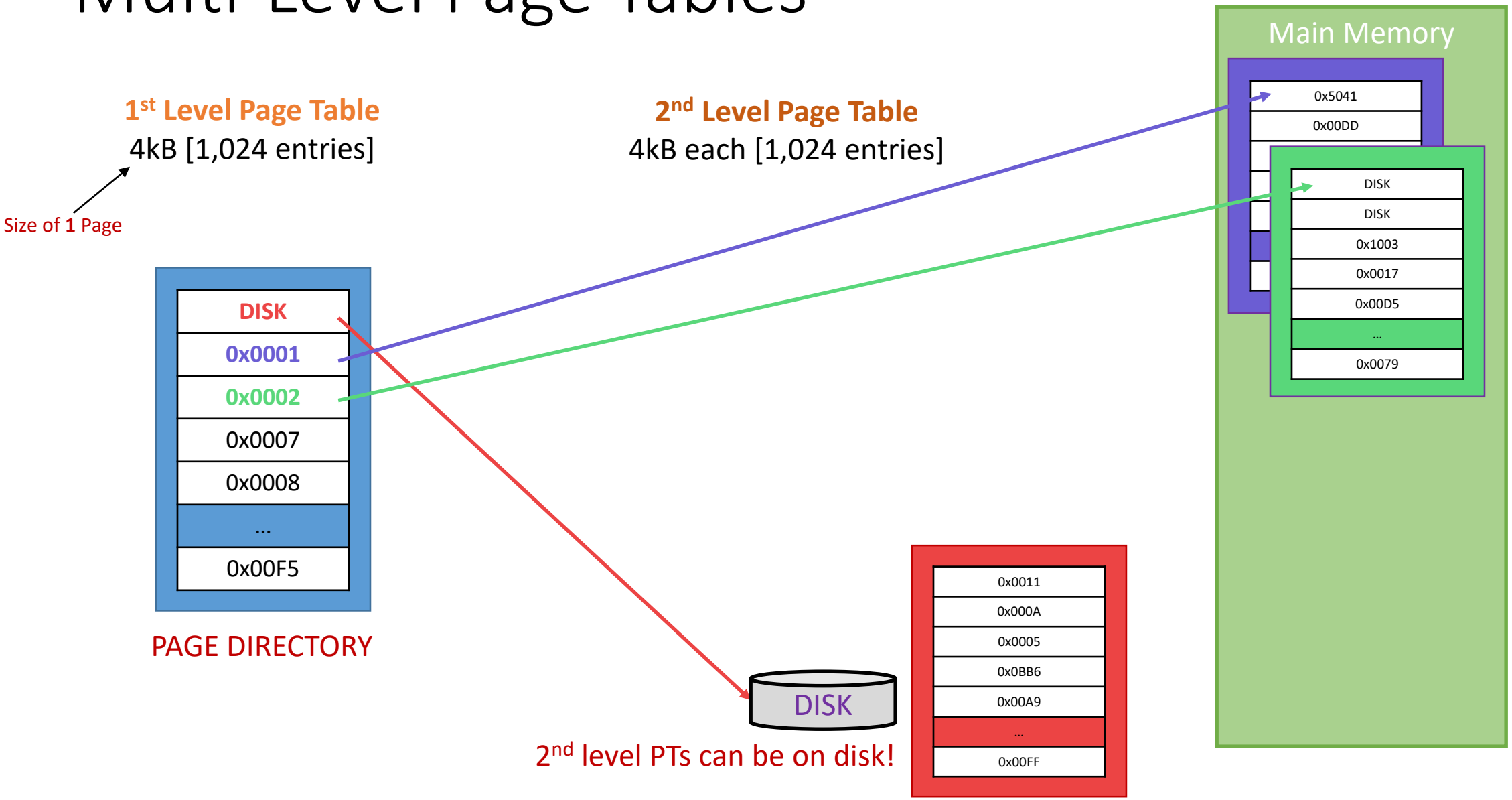
PAGE DIRECTORY



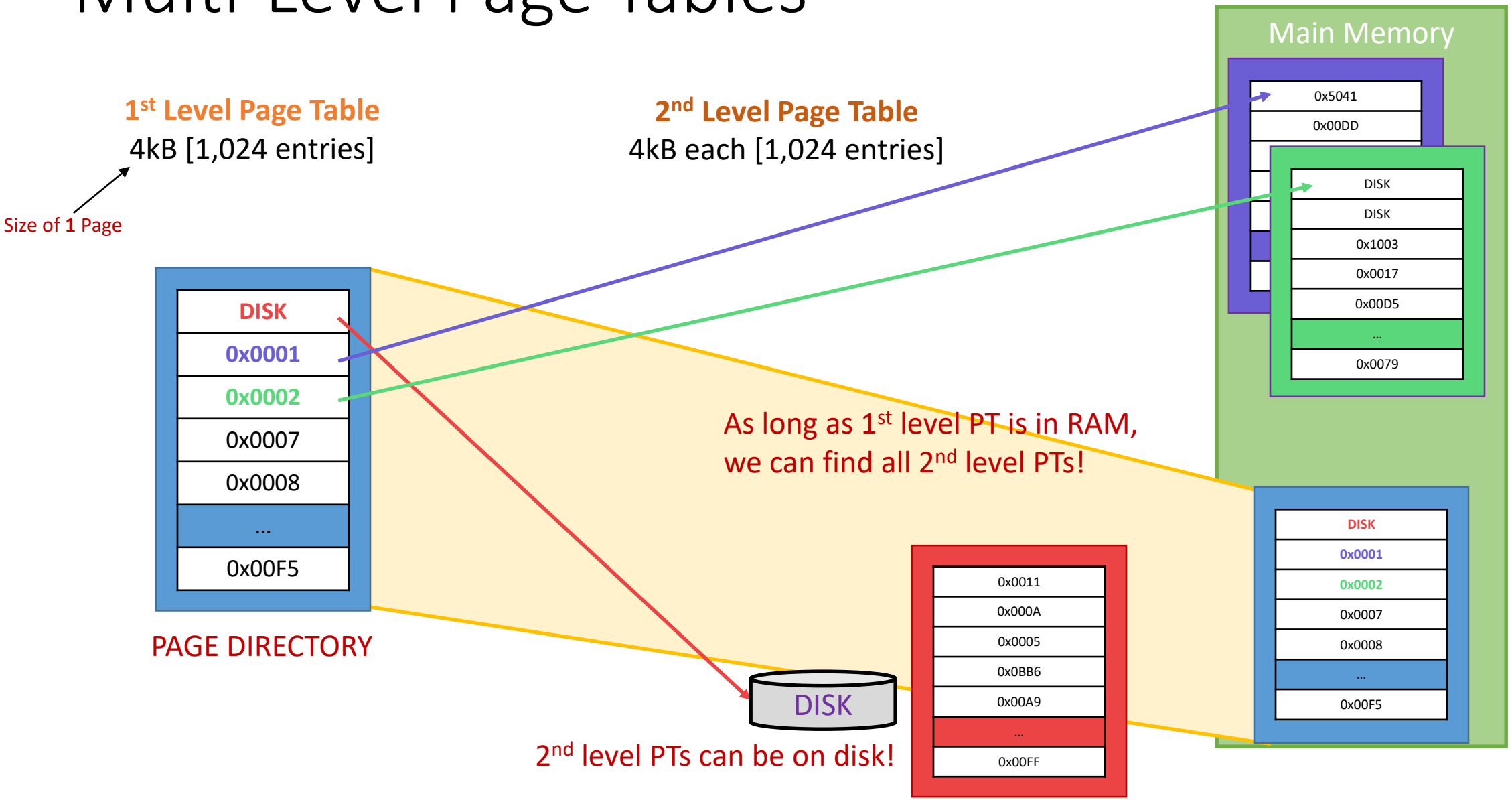
Multi-Level Page Tables



Multi-Level Page Tables



Multi-Level Page Tables



Quiz: Multi-Level Page Tables

Q: With multilevel page tables, what is the smallest amount of page table data that we need in RAM to run a single 32-bit application?

I. 4 kB
II. 8 kB
III. 8 MB
IV. 1 GB
V. 4 MB

Quiz: Multi-Level Page Tables

Q: With multilevel page tables, what is the smallest amount of page table data that we need in RAM to run a single 32-bit application?

I. 4 kB
II. 8 kB
III. 8 MB
IV. 1 GB
V. 4 MB

A:

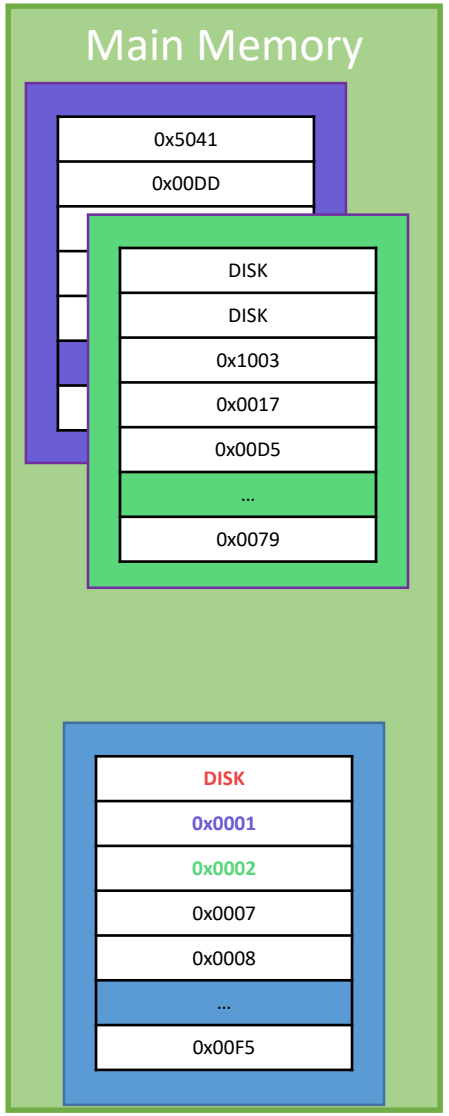
II. 8 kB

We must always keep the 1st level page table in RAM (4 kB) and we need at least the 2nd level page table for the application data.

Multi-Level Page Table Translation

Virtual Address [32 bit]

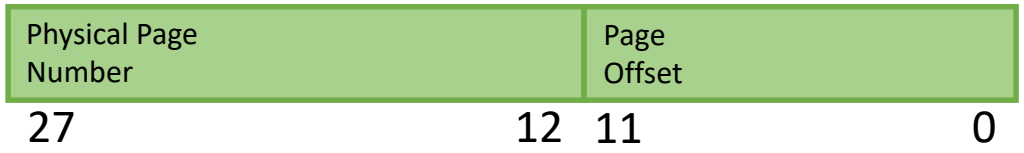
Physical Address [28 bits]

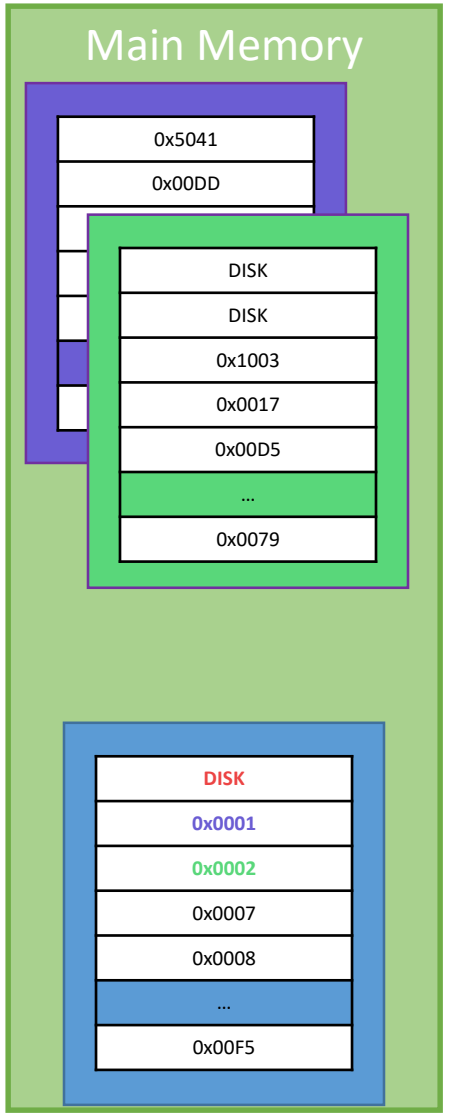


Multi-Level Page Table Translation

Virtual Address [32 bit]

Physical Address [28 bits]



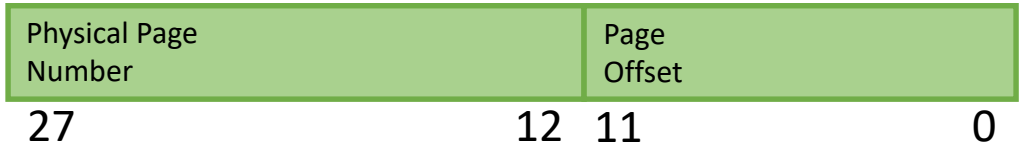


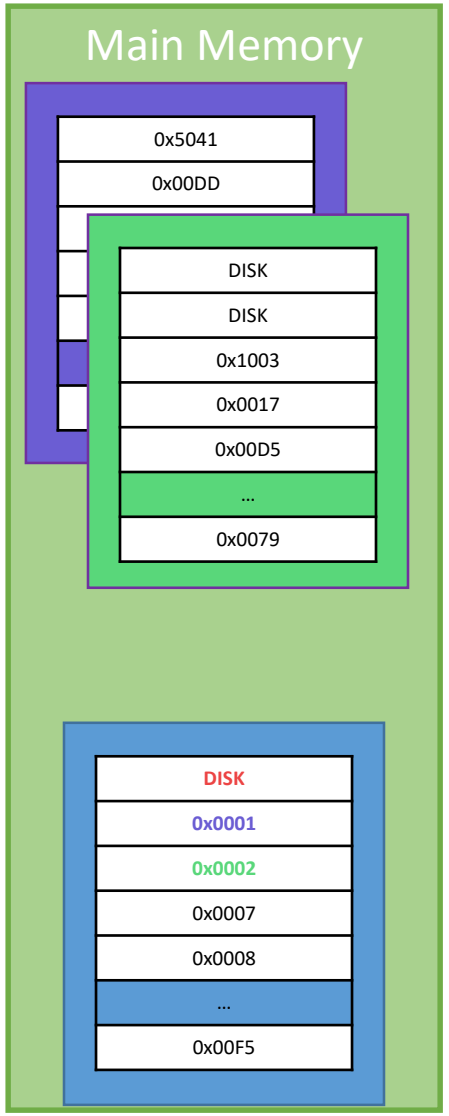
Multi-Level Page Table Translation

VA: 0x00403 713

Virtual Address [32 bit]

Physical Address [28 bits]



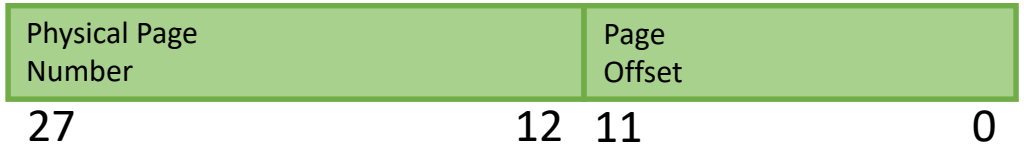


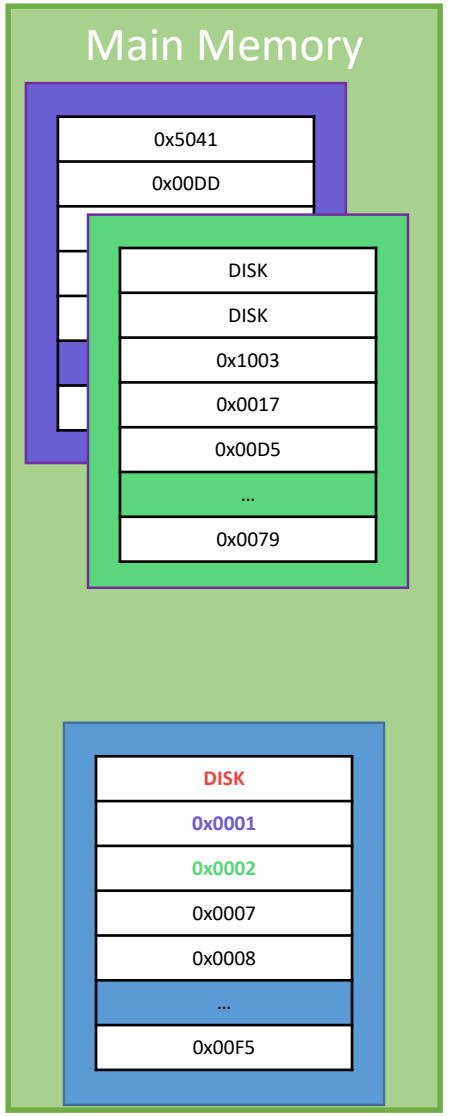
Multi-Level Page Table Translation

VA: 0x00403 713

Virtual Address [32 bit]

Physical Address [28 bits]



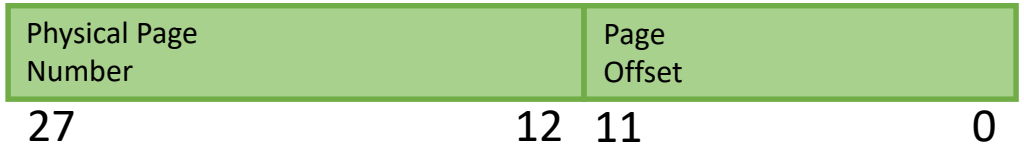


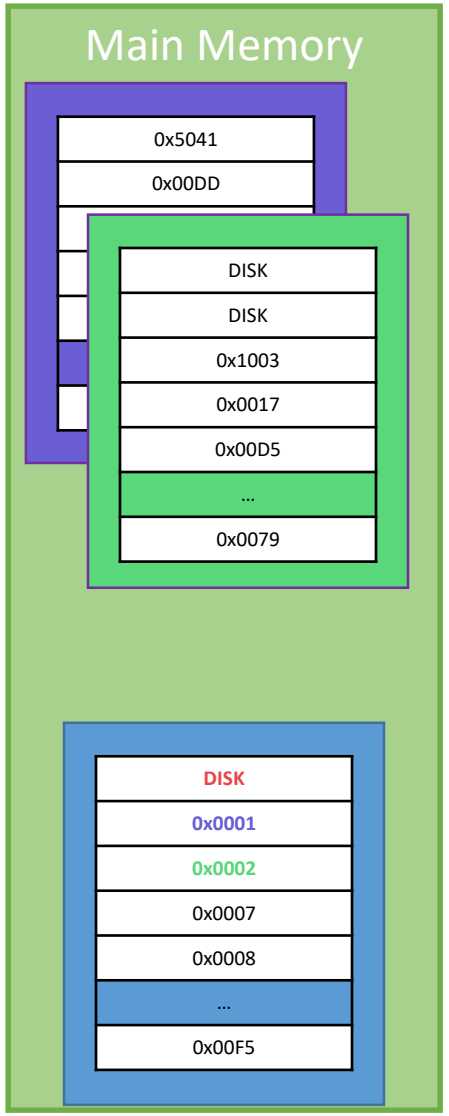
Multi-Level Page Table Translation

VA: 0x00403 713

Virtual Address [32 bit]

Physical Address [28 bits]



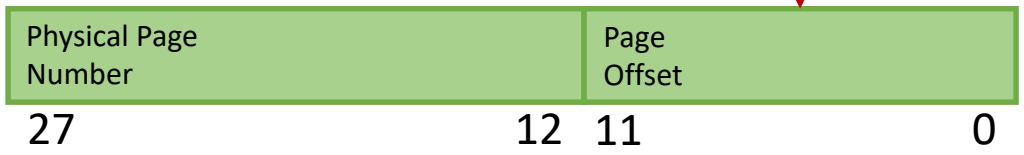


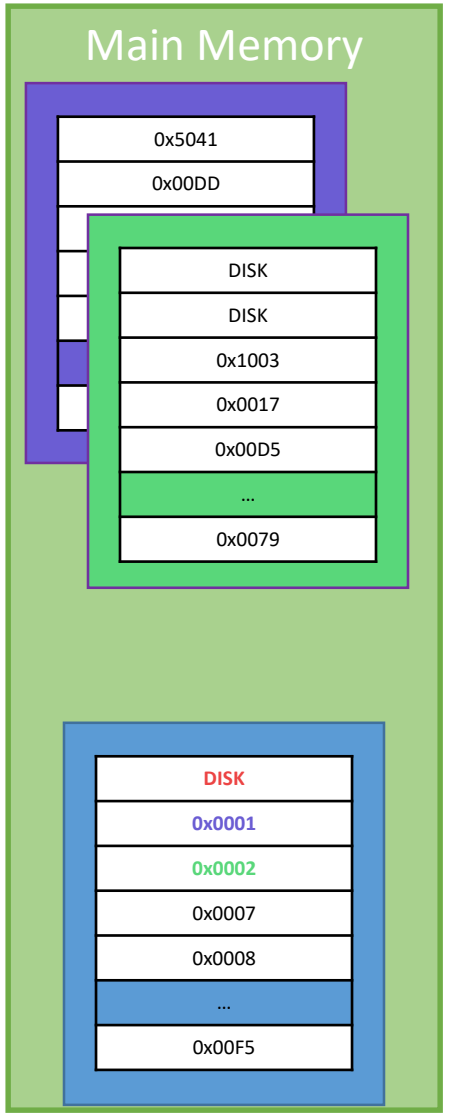
Multi-Level Page Table Translation

VA: 0x00403 713

Virtual Address [32 bit]

Physical Address [28 bits]



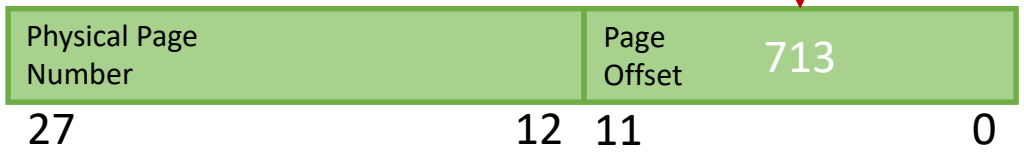


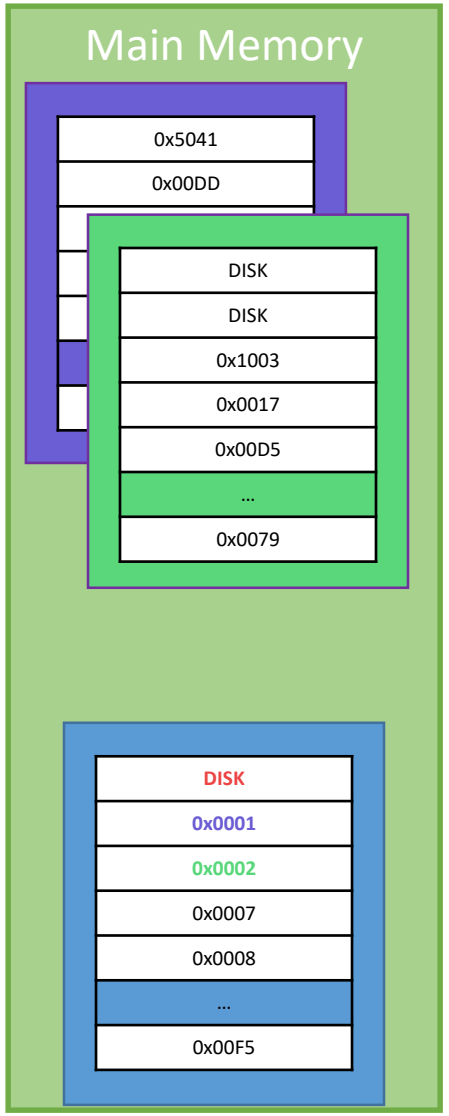
Multi-Level Page Table Translation

VA: 0x00403 713

Virtual Address [32 bit]

Physical Address [28 bits]



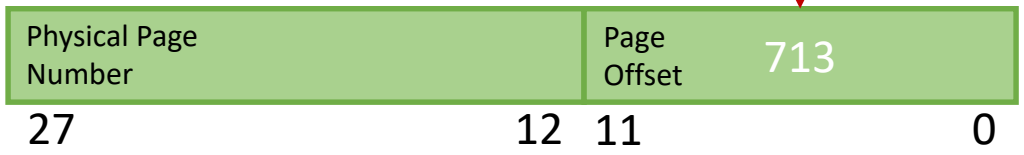


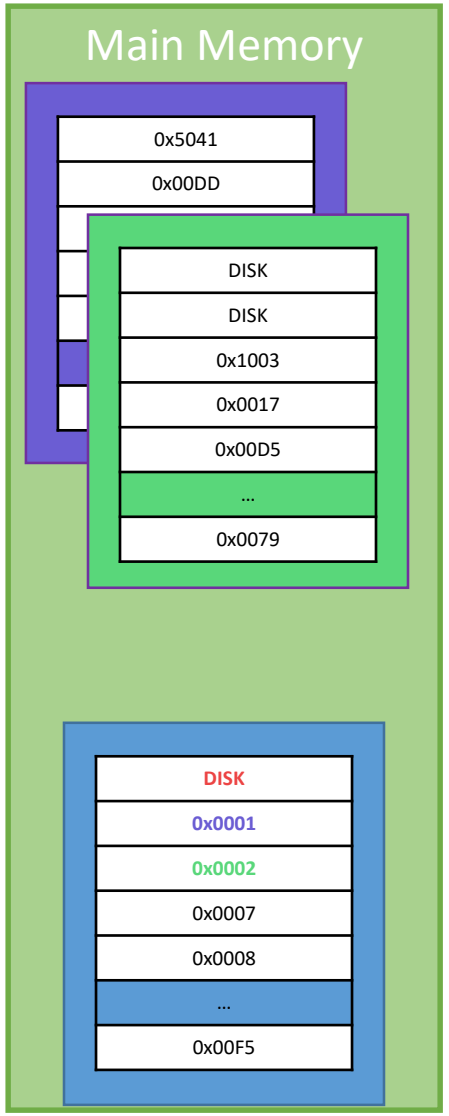
Multi-Level Page Table Translation

VA: 0x00403 713

Virtual Address [32 bit]

Physical Address [28 bits]



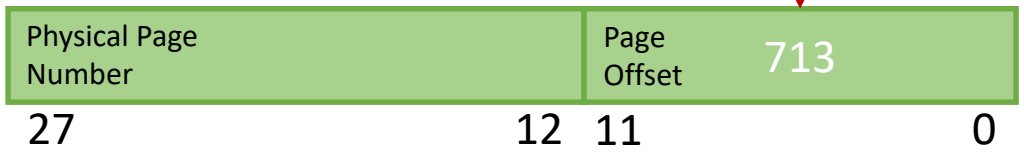


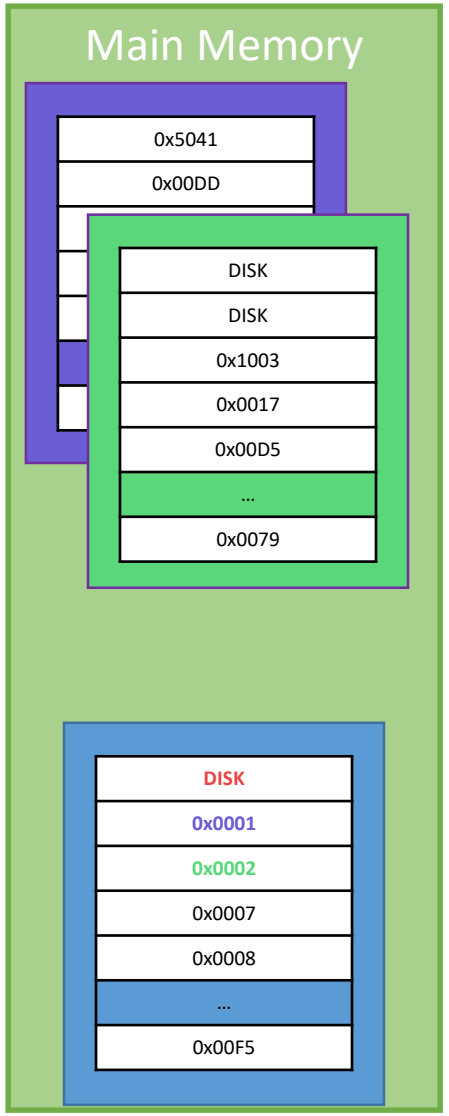
Multi-Level Page Table Translation

VA: 0x00403 713

Virtual Address [32 bit]

Physical Address [28 bits]

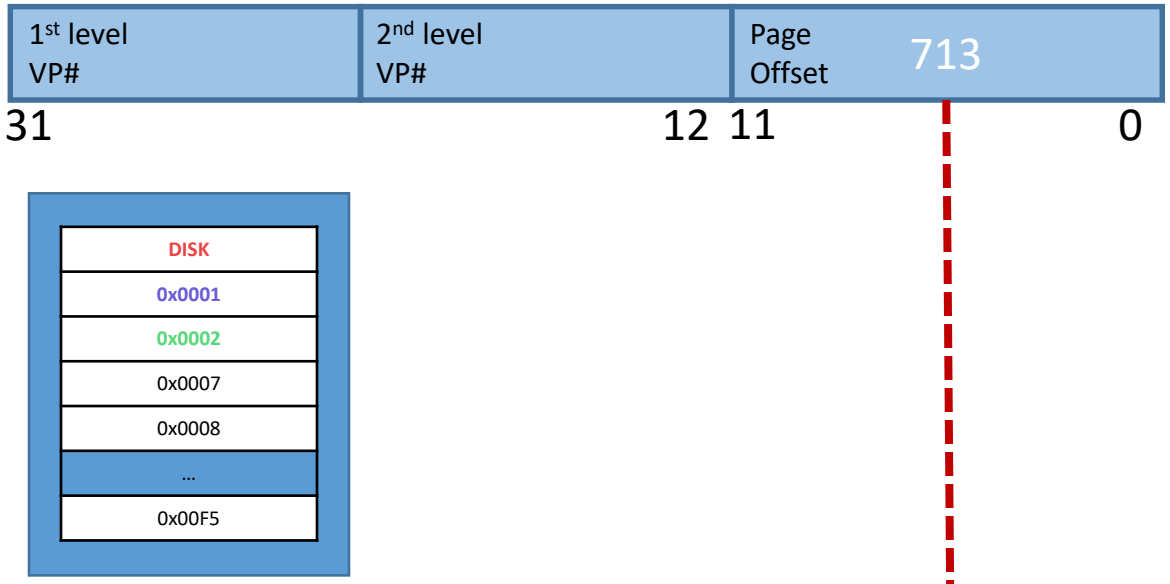




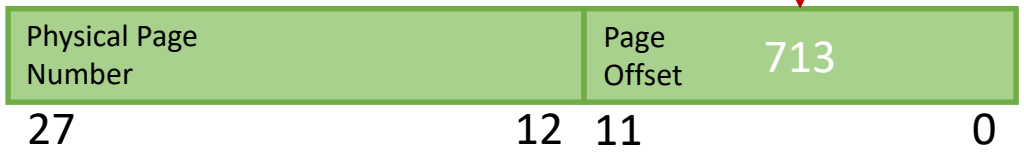
Multi-Level Page Table Translation

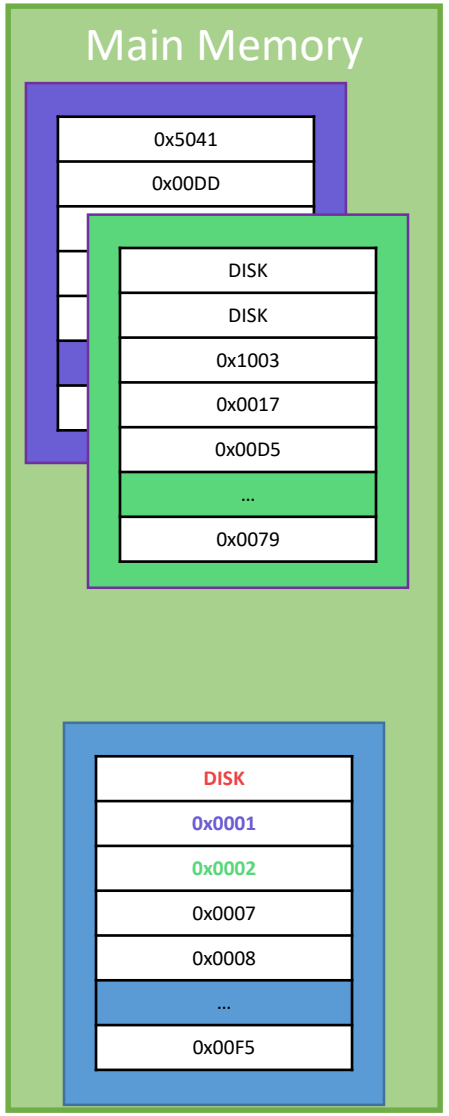
VA: 0x00403 713

Virtual Address [32 bit]



Physical Address [28 bits]

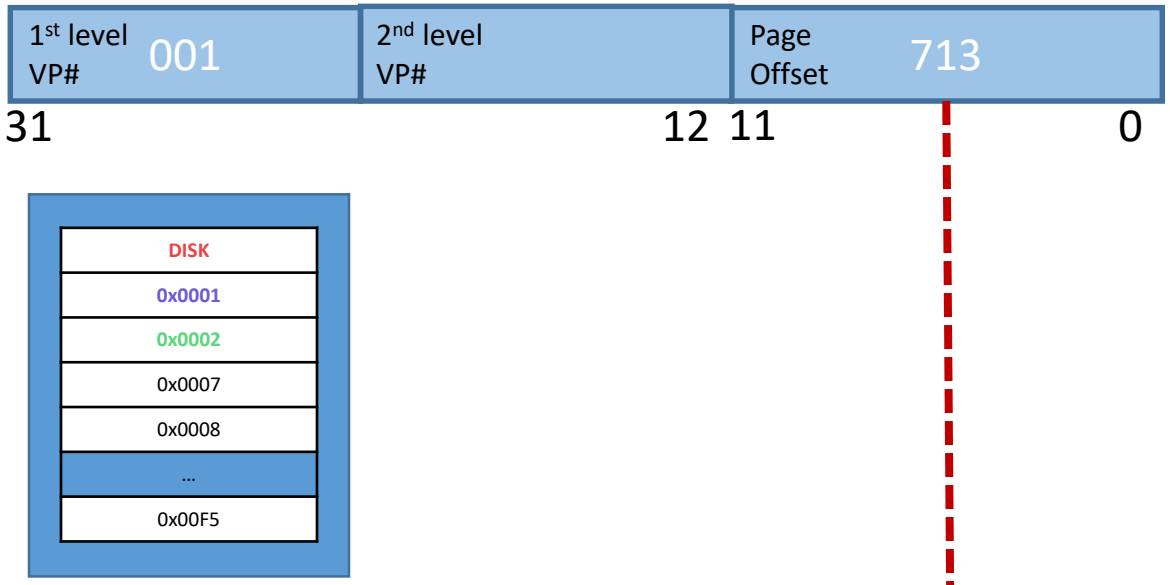




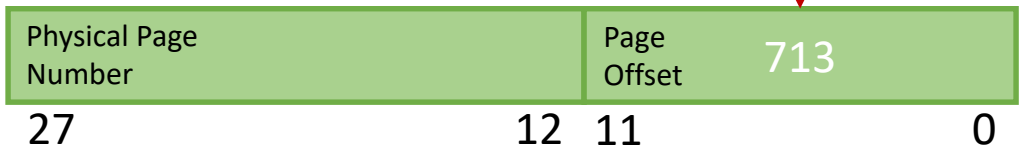
Multi-Level Page Table Translation

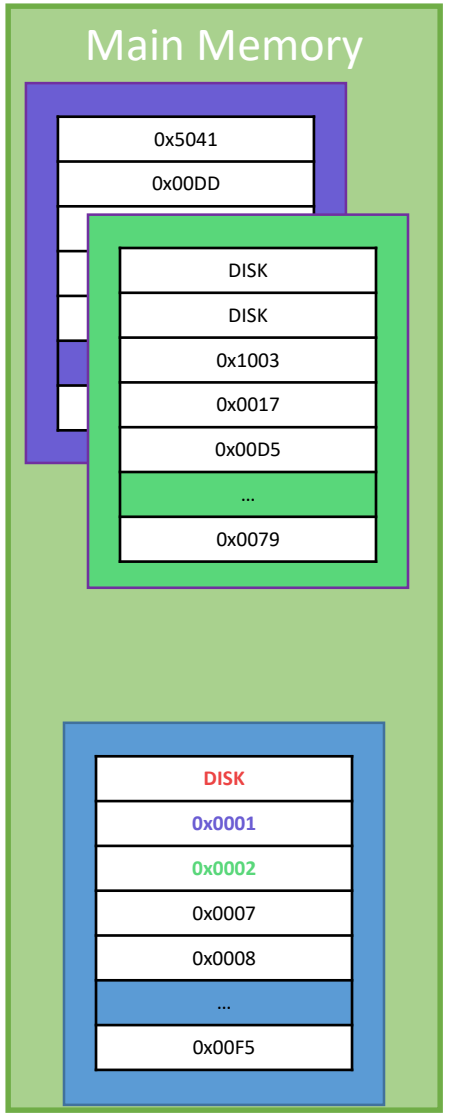
VA: 0x00403 713

Virtual Address [32 bit]



Physical Address [28 bits]

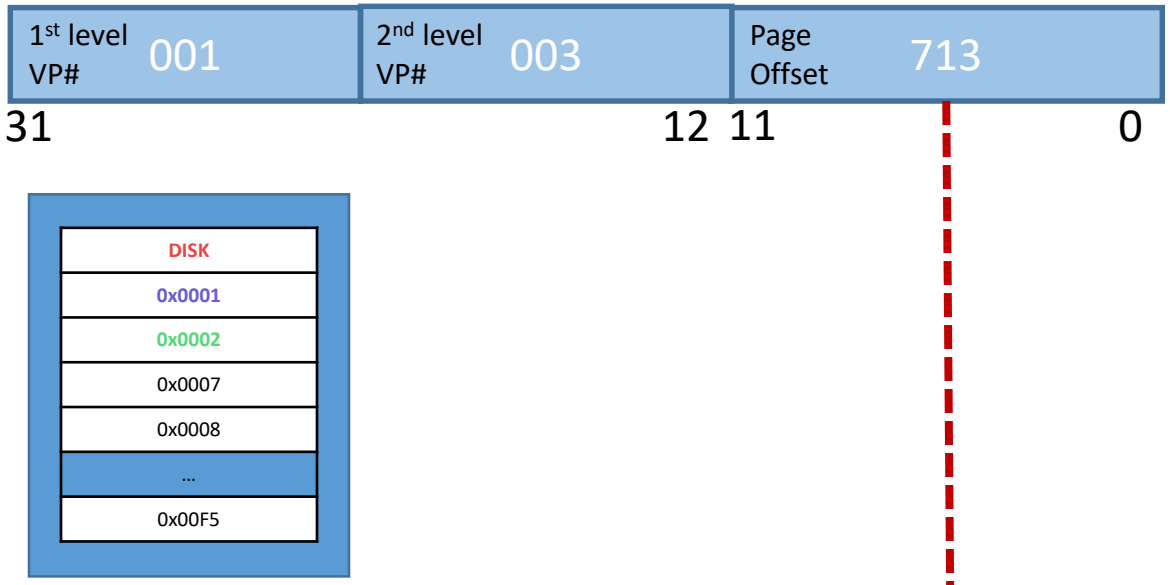




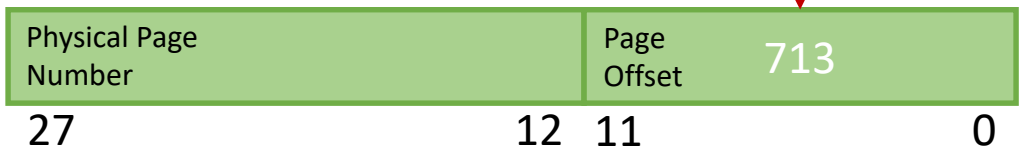
Multi-Level Page Table Translation

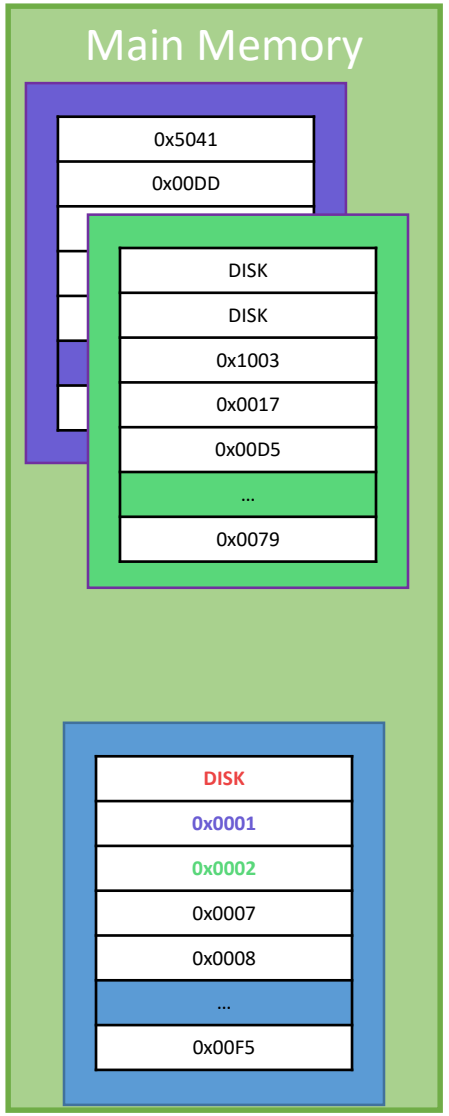
VA: 0x00403 713

Virtual Address [32 bit]



Physical Address [28 bits]

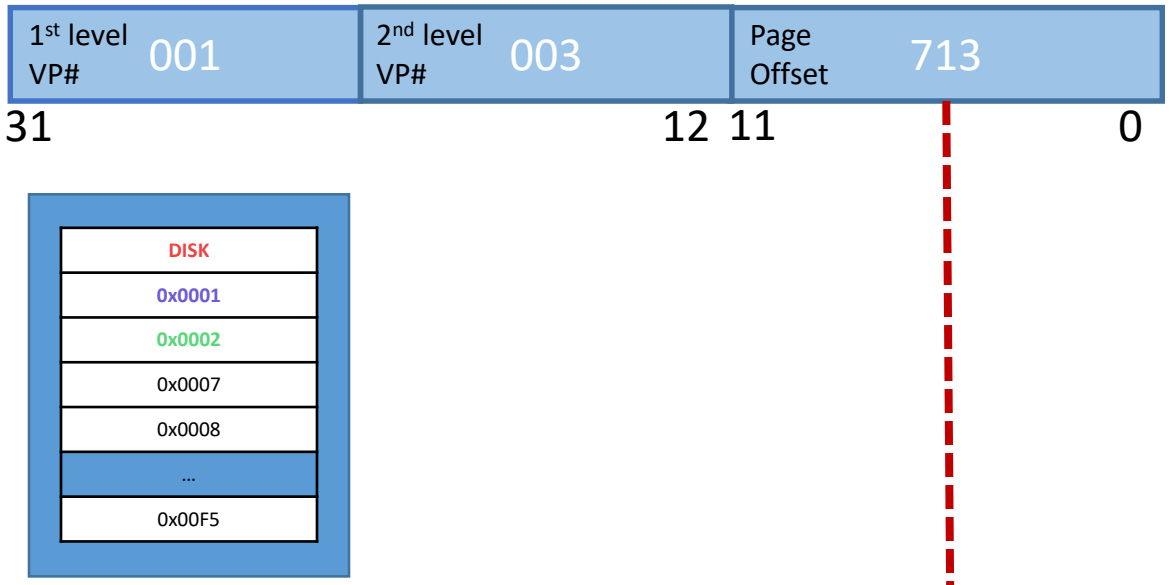




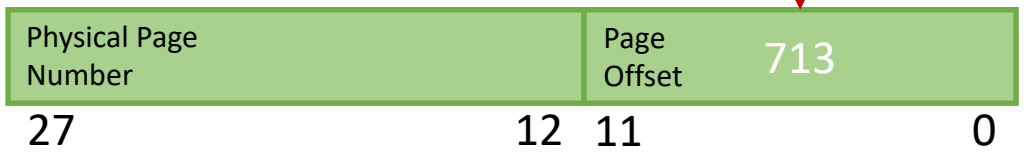
Multi-Level Page Table Translation

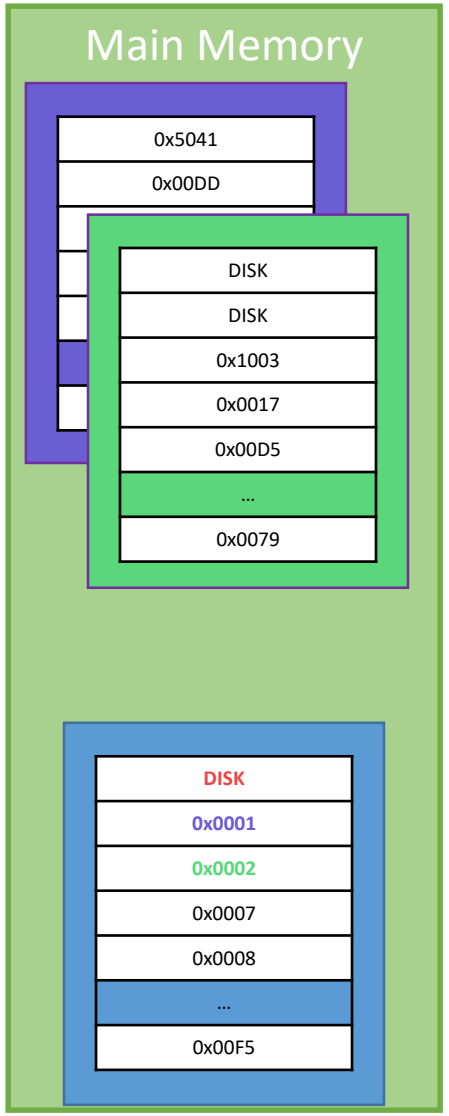
VA: 0x00403 713

Virtual Address [32 bit]



Physical Address [28 bits]

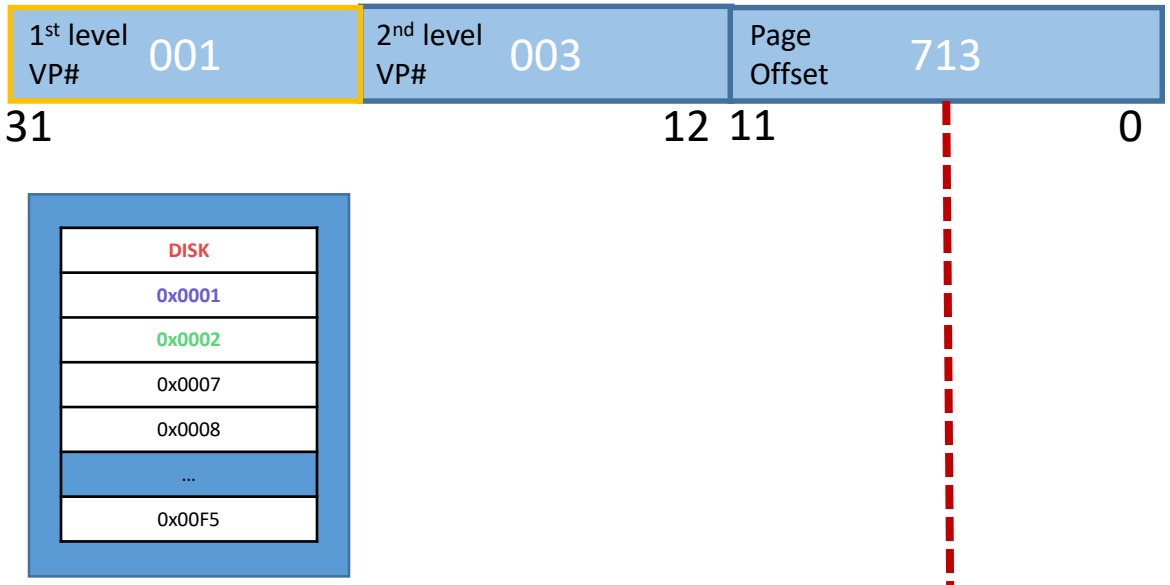




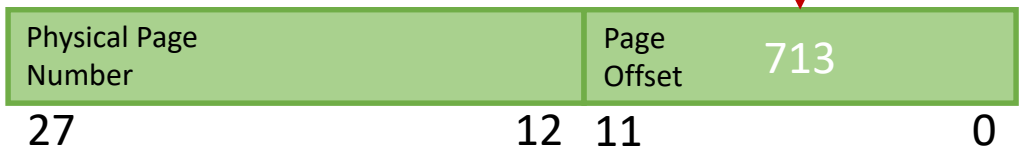
Multi-Level Page Table Translation

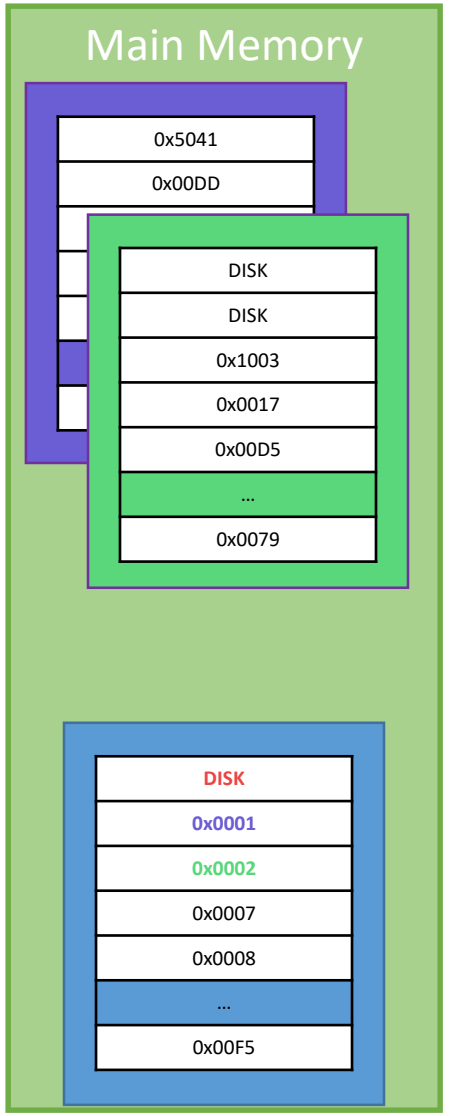
VA: 0x00403 713

Virtual Address [32 bit]



Physical Address [28 bits]

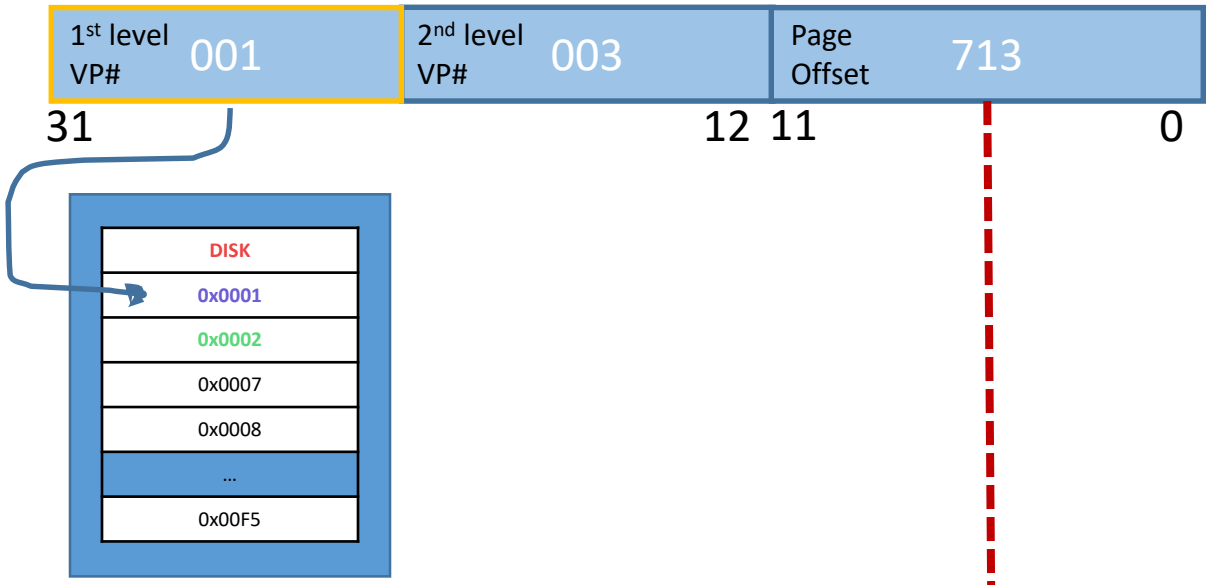




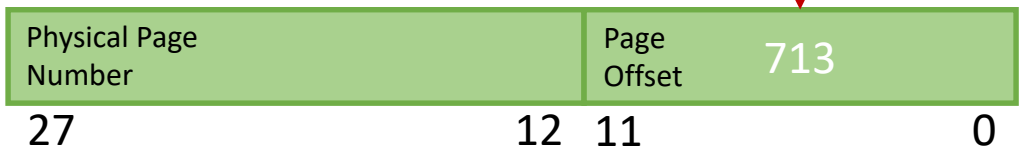
Multi-Level Page Table Translation

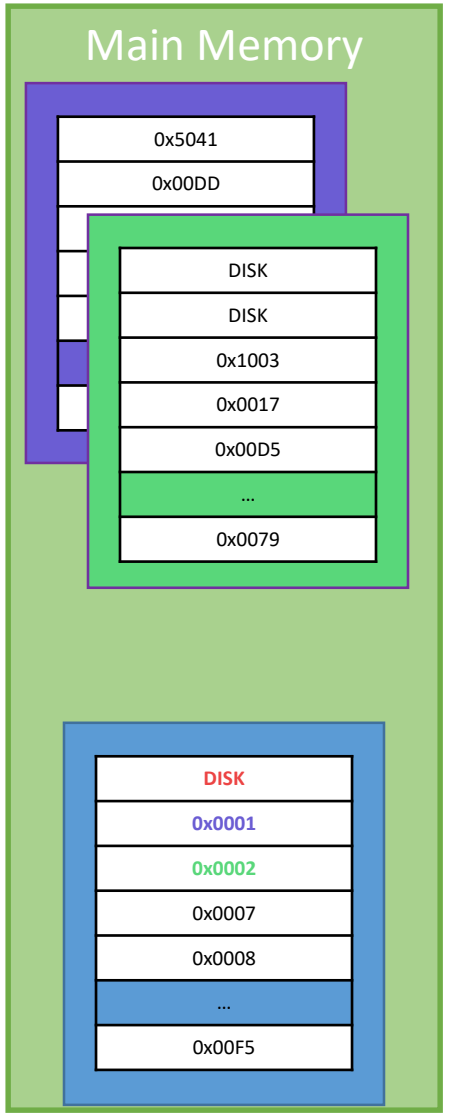
VA: 0x00403 713

Virtual Address [32 bit]

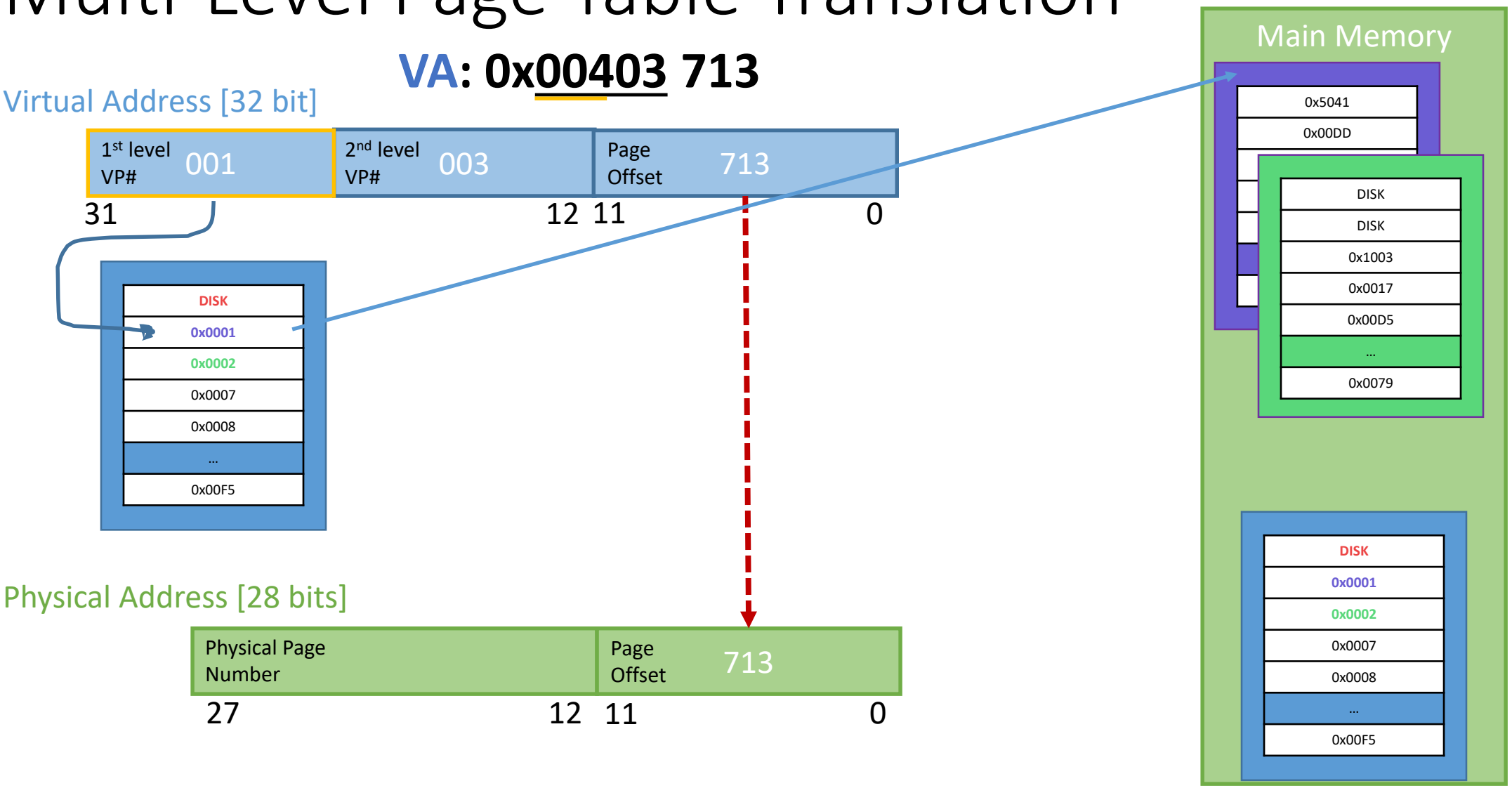


Physical Address [28 bits]

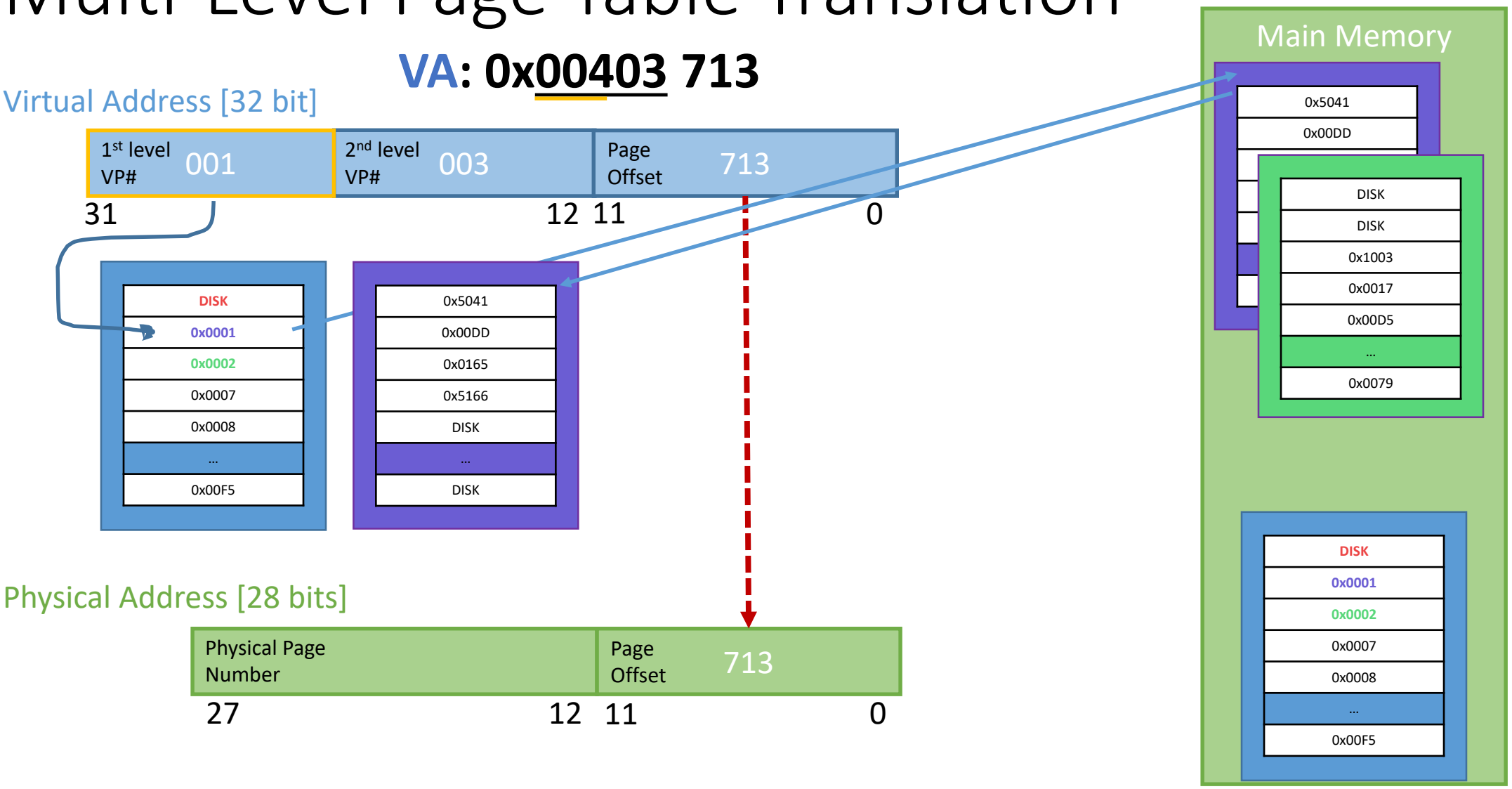




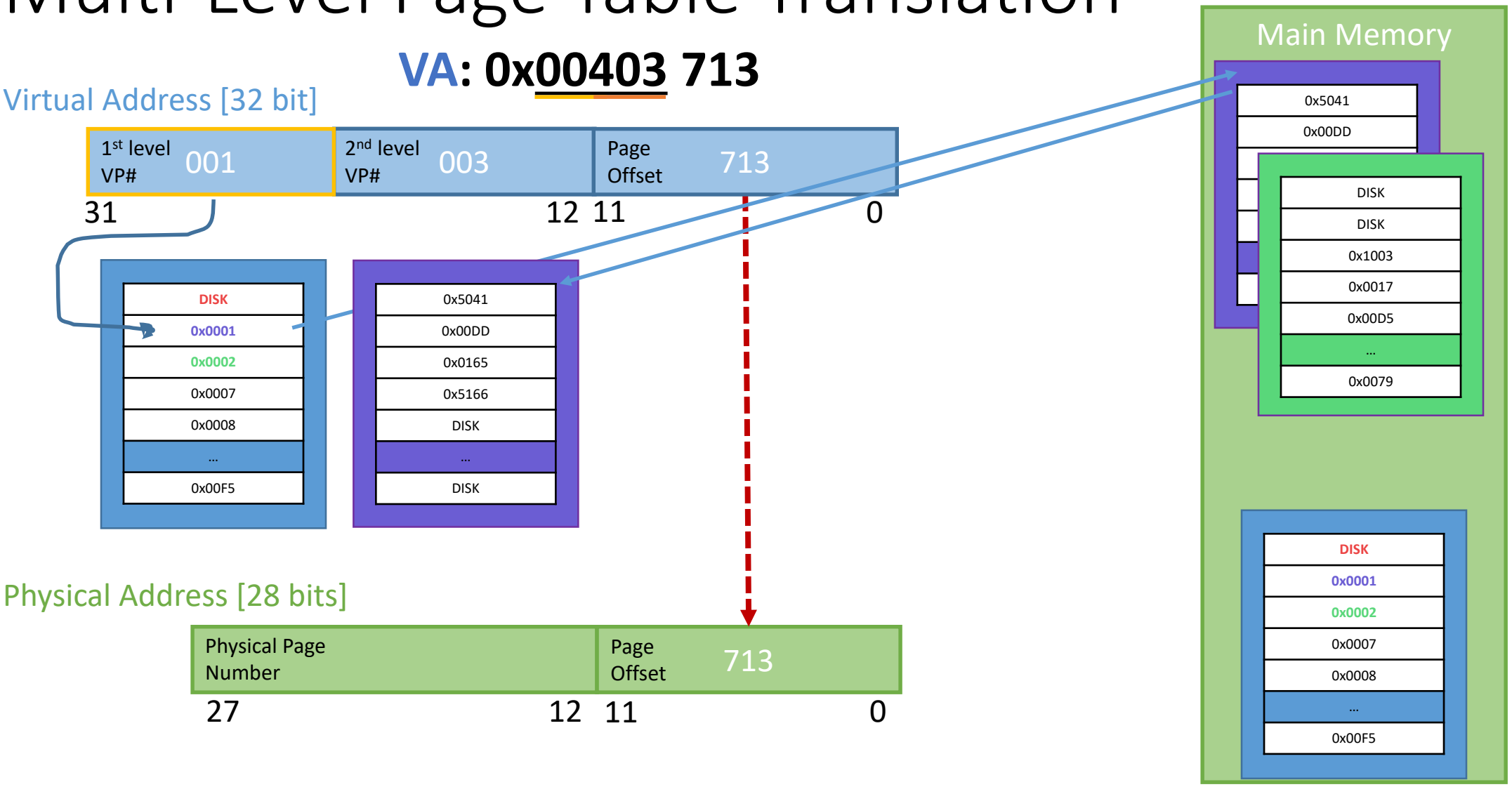
Multi-Level Page Table Translation



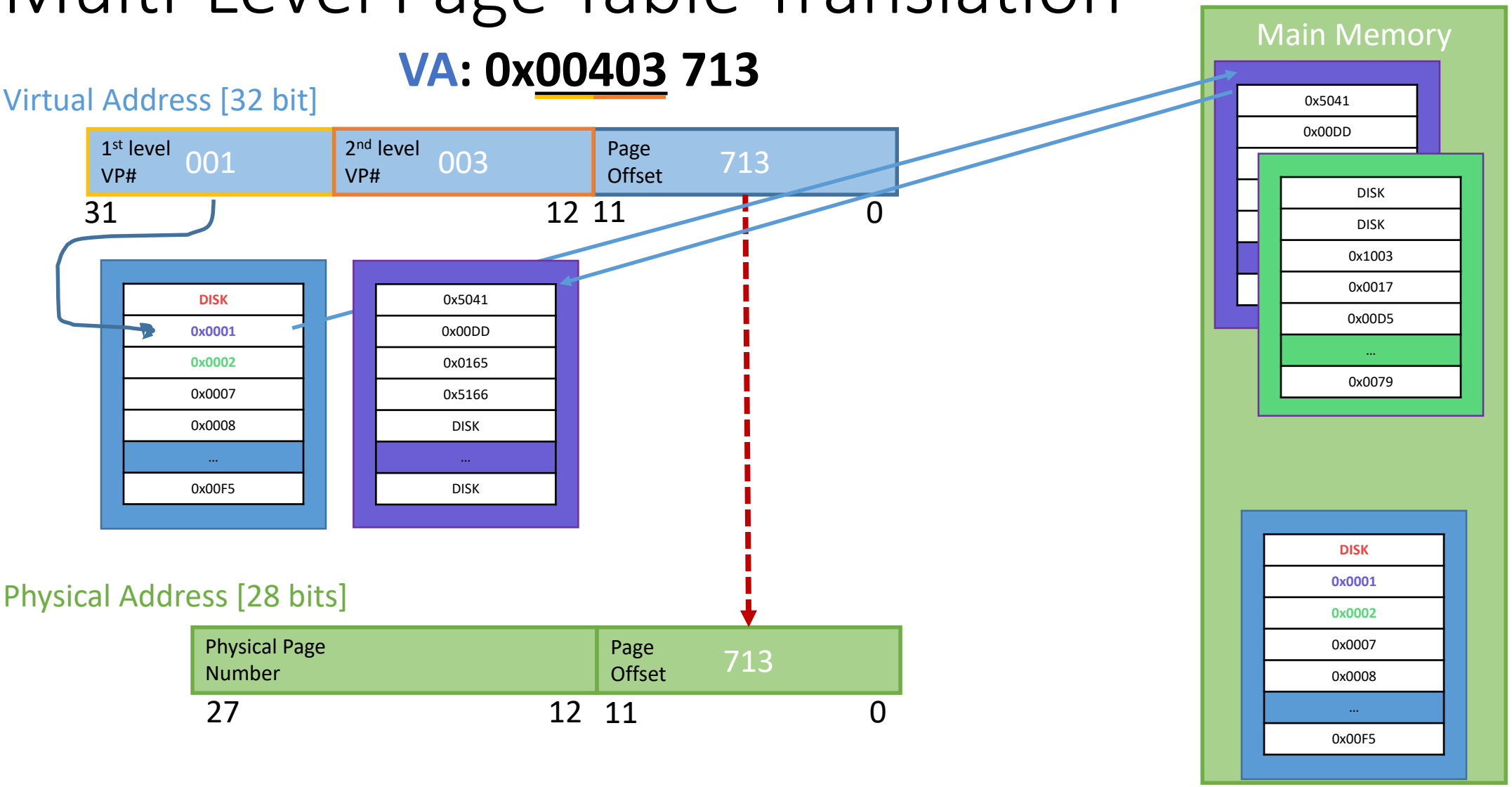
Multi-Level Page Table Translation



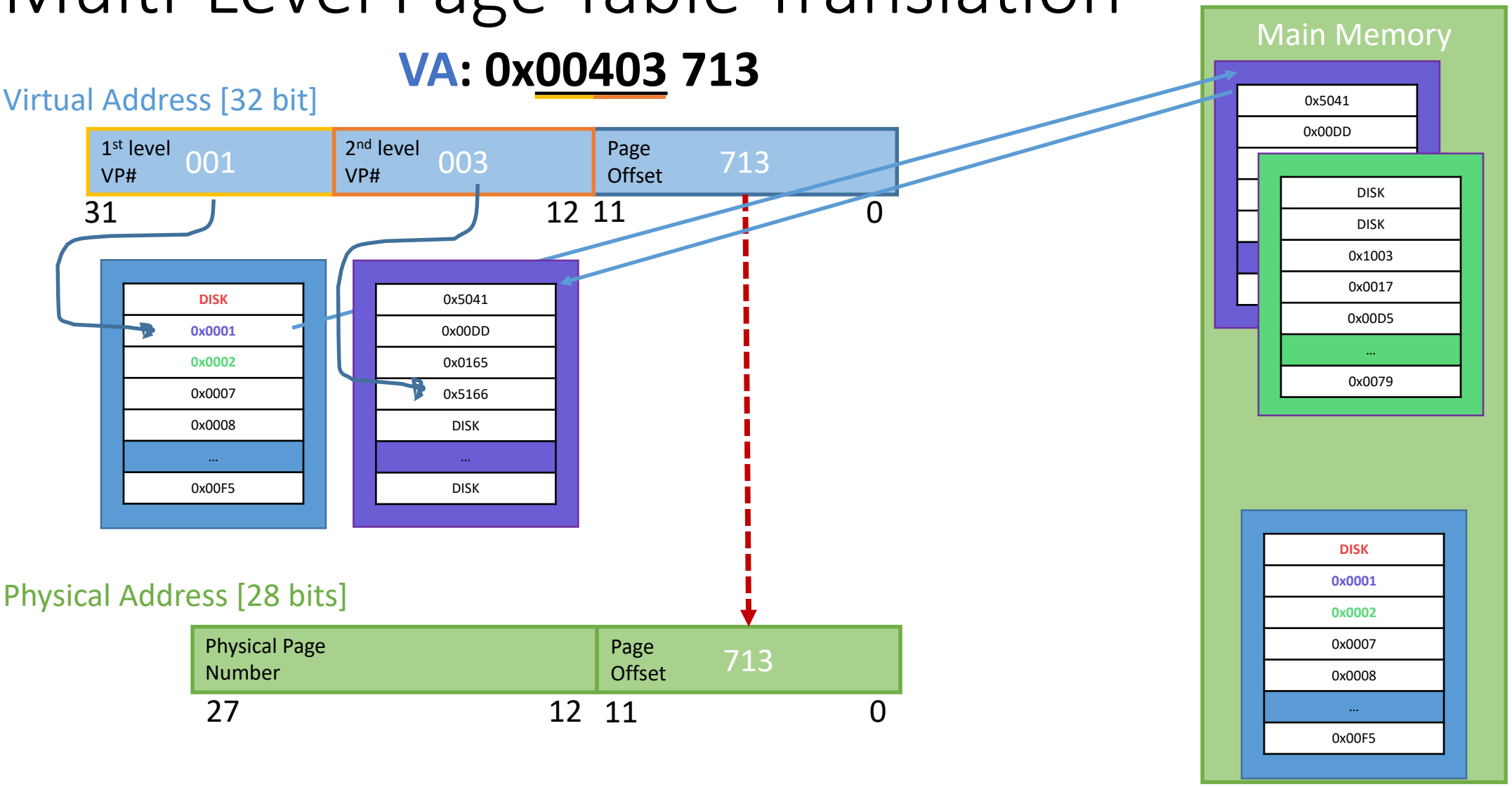
Multi-Level Page Table Translation



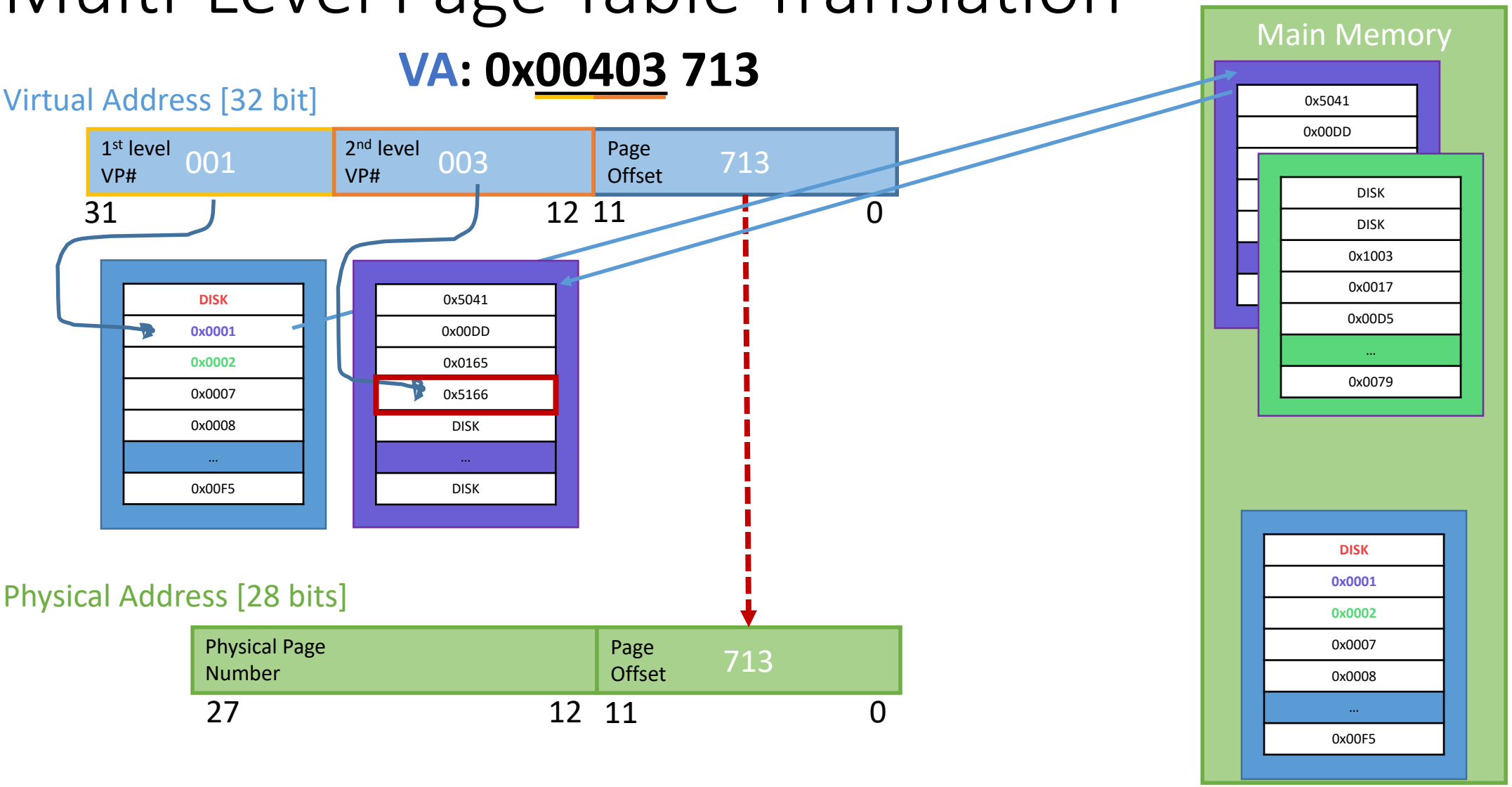
Multi-Level Page Table Translation



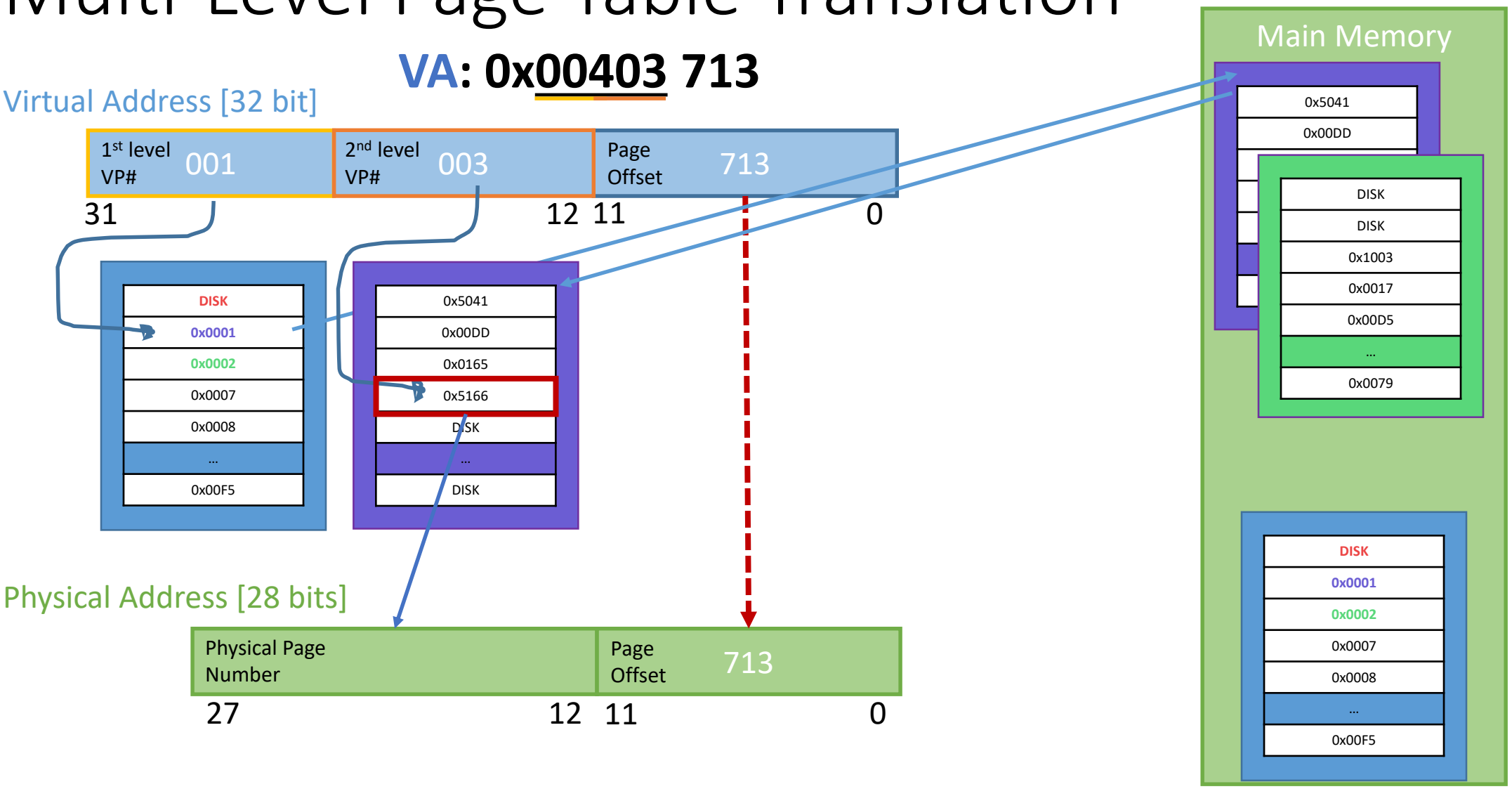
Multi-Level Page Table Translation



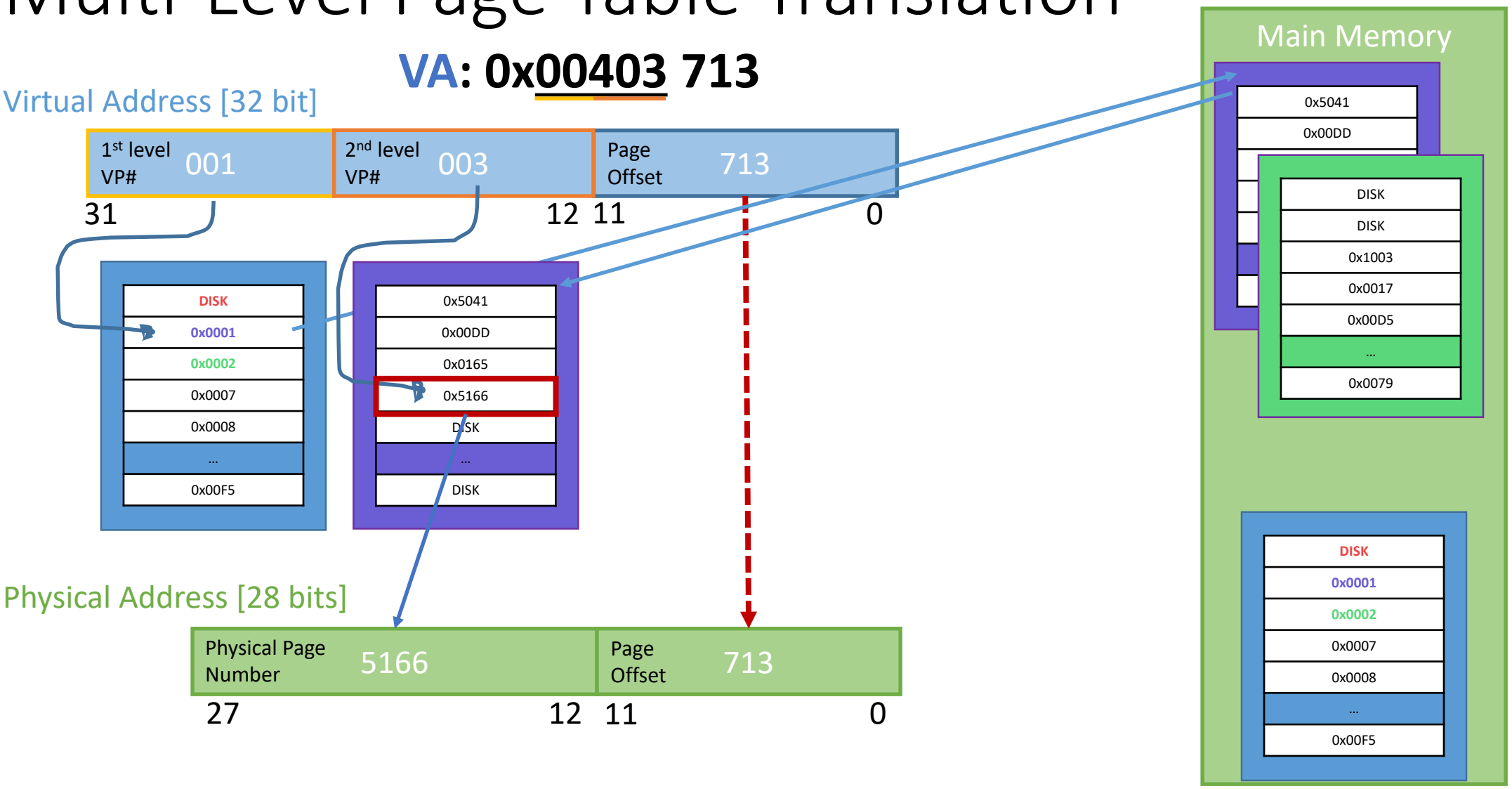
Multi-Level Page Table Translation



Multi-Level Page Table Translation



Multi-Level Page Table Translation



Multi-Level Page Table Translation

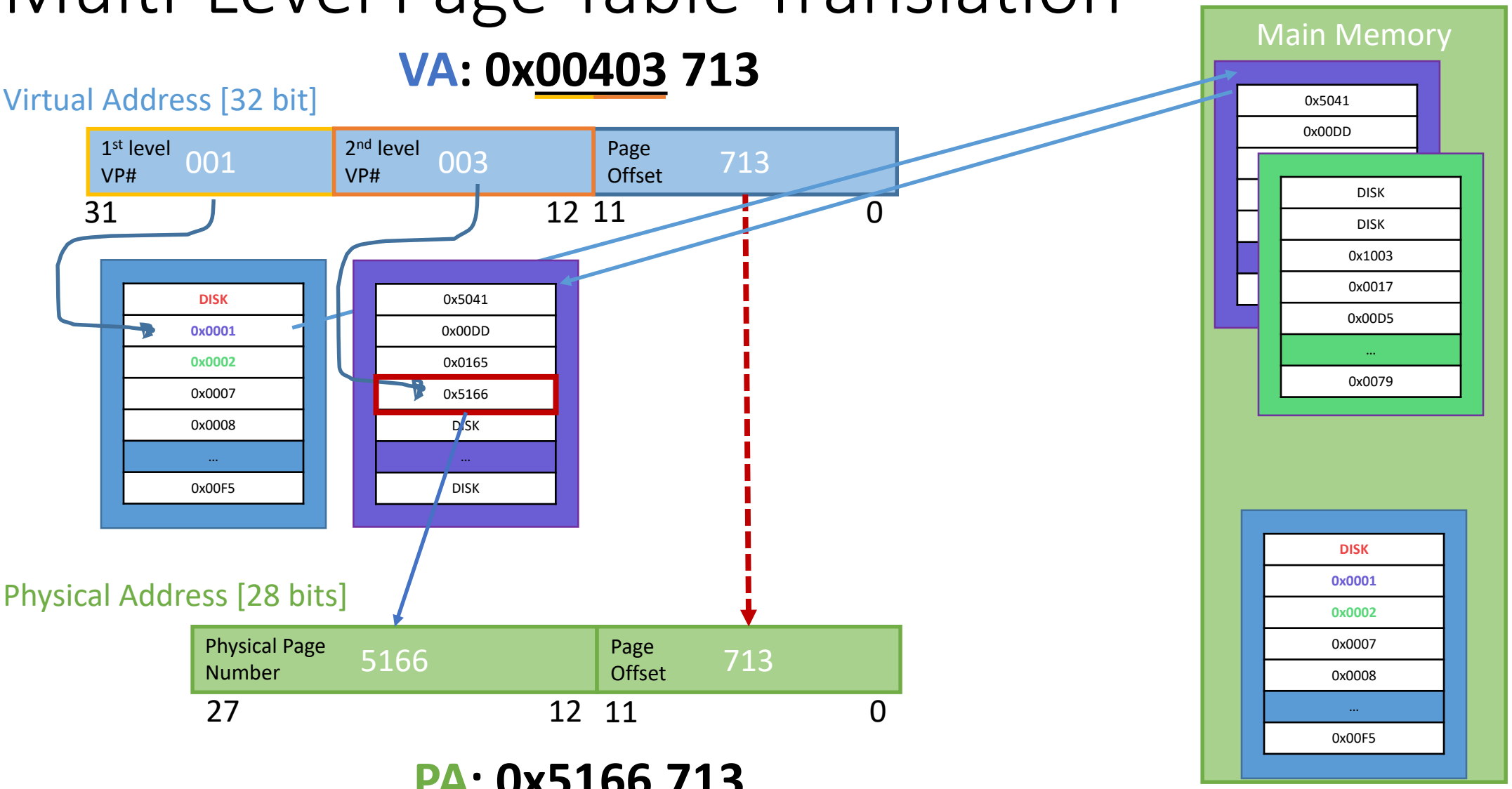
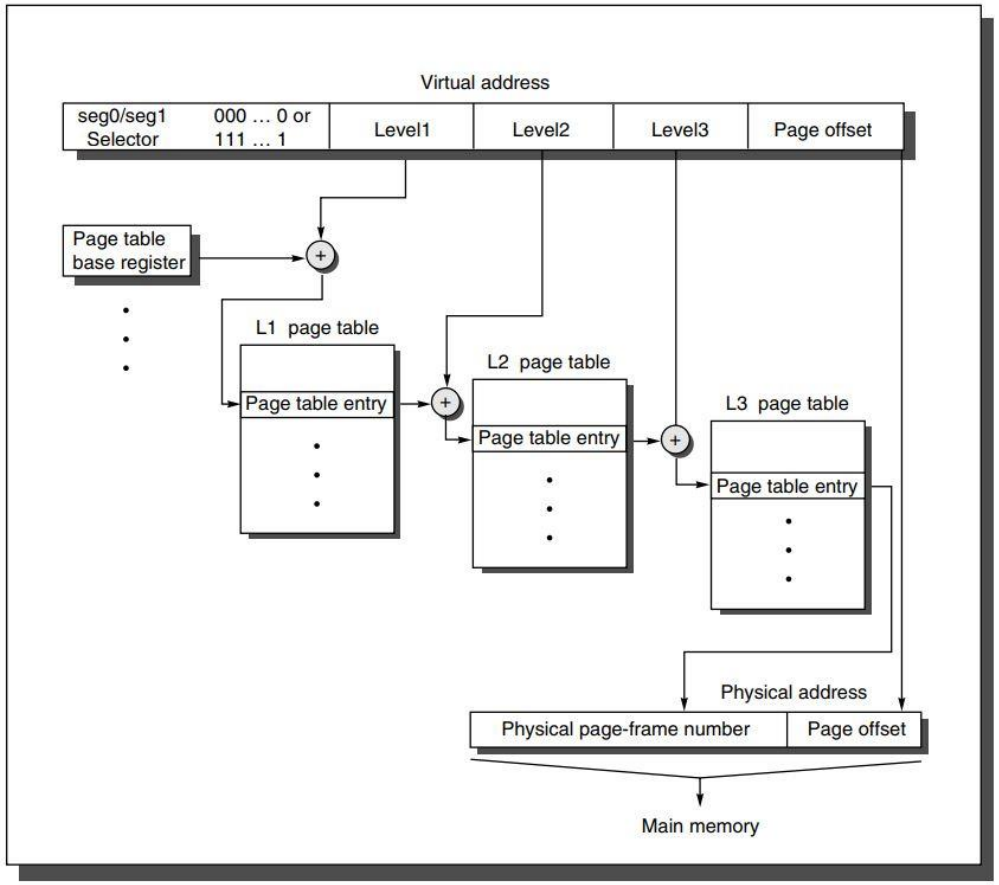


Illustration from the textbook (advanced)



Multi-Level Page Table: Outro

- We always need to keep the 1st level Page Table in memory

Multi-Level Page Table: Outro

- We always need to keep the 1st level Page Table in memory
- We need at least one 2nd level Page Table to do translations

Multi-Level Page Table: Outro

- We always need to keep the 1st level Page Table in memory
- We need at least one 2nd level Page Table to do translations

- 2-Tier translation

Multi-Level Page Table: Outro

- We always need to keep the 1st level Page Table in memory
- We need at least one 2nd level Page Table to do translations

- 2-Tier translation
  - Top-most **10** bits used to index **PT1**

Multi-Level Page Table: Outro

- We always need to keep the 1st level Page Table in memory
- We need at least one 2nd level Page Table to do translations

- 2-Tier translation
  - Top-most **10** bits used to index **PT1**
  - Middle-most **10** bits used to index **PT2**

Multi-Level Page Table: Outro

- We always need to keep the 1st level Page Table in memory
- We need at least one 2nd level Page Table to do translations

- 2-Tier translation
  - Top-most **10** bits used to index **PT1**
  - Middle-most **10** bits used to index **PT2**

- Advantage:
  - We can now keep Page Tables on disk

Multi-Level Page Table: Outro

- We always need to keep the 1st level Page Table in memory
- We need at least one 2nd level Page Table to do translations

- 2-Tier translation
  - Top-most **10** bits used to index **PT1**
  - Middle-most **10** bits used to index **PT2**

- Advantage:
  - We can now keep Page Tables on disk
  - We can still address the same amount of data

Quiz: Multi-Level Page Tables

Q: **<u>With</u>** multilevel page tables, if I am running *100* applications *concurrently*, how much memory do I need in RAM?

I. 4 kB
II. 8 kB
III. 8 MB
IV. 800 kB
V. 400 MB

Quiz: Multi-Level Page Tables

Q: **<u>With</u>** multilevel page tables, if I am running *100* applications *concurrently*, how much memory do I need in RAM?

I. 4 kB
II. 8 kB
III. 8 MB
IV. 800 kB
V. 400 MB

A:

IV. 800 kB

For each program, we must always keep the 1st level page table in RAM (4 kB) and we need at least one 2nd level page table to address data.

Quiz: Multi-Level Page Tables

Q: **<u>Without</u>** multilevel page tables, if I am running *100* applications *concurrently*, how much memory do I need in RAM?

I. 4 kB
II. 8 kB
III. 8 MB
IV. 800 kB
V. 400 MB

Quiz: Multi-Level Page Tables

Q: **<u>Without</u>** multilevel page tables, if I am running *100* applications *concurrently*, how much memory do I need in RAM?

I. 4 kB
II. 8 kB
III. 8 MB
IV. 800 kB
V. 400 MB

A:

IV. 400 MB

For each program, we must keep the entire 4 MB Page Table in RAM at all times.

Inverted Page Tables

References

- David Black-Schaffer: Lecture Series on Virtual Memory
- Patterson, Hennessy: Computer Organization and Design: the Hardware/Software Interface
- Intel Hardware Data-Sheets
- Linux: Anatomy of a Program in Memory