

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
- BUT, each program needs its own page table

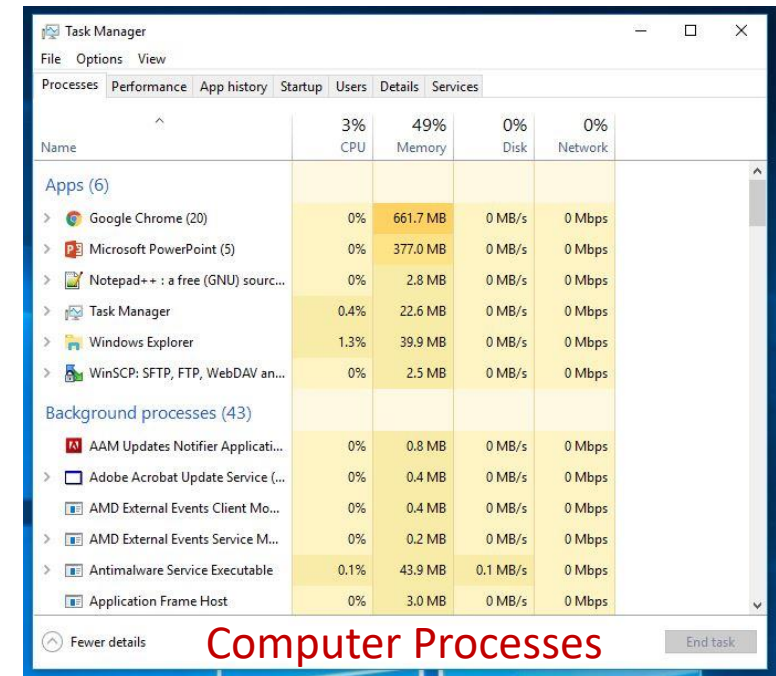
1M entries x 4 Bytes per entry
= **4 MB page tables**

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



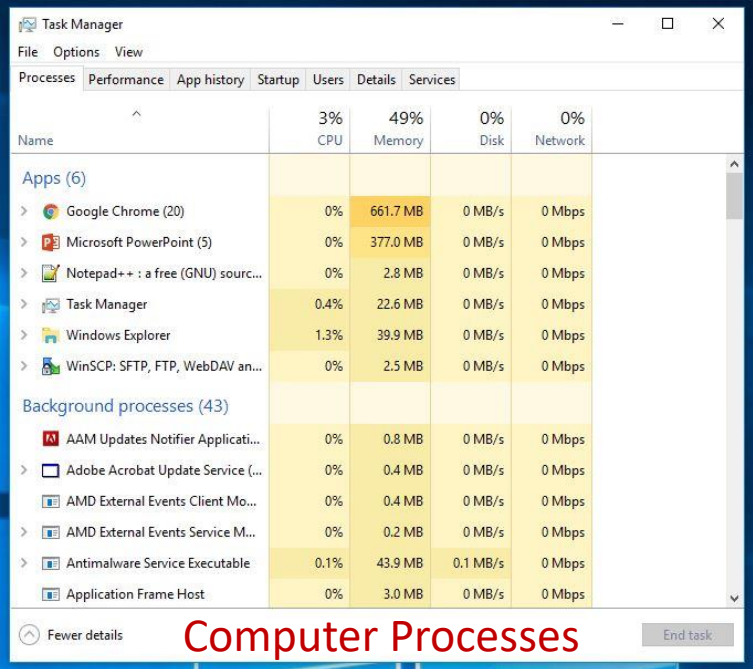
Page Table Size... again

- 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

- 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**



The screenshot shows the Windows Task Manager Performance tab, specifically the 'Processes' section. It displays a table of running processes with columns for Name, CPU usage, Memory usage, Disk usage, and Network usage. The 'Memory' column is highlighted in yellow. The processes are grouped into 'Apps (6)' and 'Background processes (43)'. The total memory usage is shown as 49%.

Name	CPU	Memory	Disk	Network
Apps (6)				
Google Chrome (20)	0%	661.7 MB	0 MB/s	0 Mbps
Microsoft PowerPoint (5)	0%	377.0 MB	0 MB/s	0 Mbps
Notepad++ : a free (GNU) sourc...	0%	2.8 MB	0 MB/s	0 Mbps
Task Manager	0.4%	22.6 MB	0 MB/s	0 Mbps
Windows Explorer	1.3%	39.9 MB	0 MB/s	0 Mbps
WinSCP: SFTP, FTP, WebDAV an...	0%	2.5 MB	0 MB/s	0 Mbps
Background processes (43)				
AAM Updates Notifier Applicati...	0%	0.8 MB	0 MB/s	0 Mbps
Adobe Acrobat Update Service (...)	0%	0.4 MB	0 MB/s	0 Mbps
AMD External Events Client Mo...	0%	0.4 MB	0 MB/s	0 Mbps
AMD External Events Service M...	0%	0.2 MB	0 MB/s	0 Mbps
Antimalware Service Executable	0.1%	43.9 MB	0.1 MB/s	0 Mbps
Application Frame Host	0%	3.0 MB	0 MB/s	0 Mbps

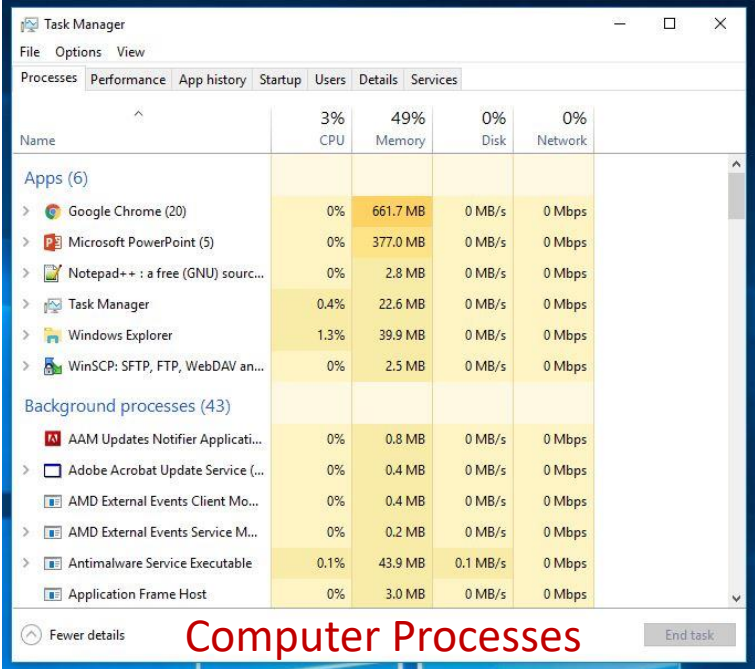
Computer Processes

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**



The screenshot shows the Windows Task Manager Performance tab, specifically the 'Memory' section. It displays a table of processes and their memory usage. The 'Memory' column is highlighted in yellow. The total memory usage is 49%. The processes listed include Google Chrome (20 instances), Microsoft PowerPoint (5 instances), Notepad++, Task Manager, Windows Explorer, WinSCP, and various background services.

Name	CPU	Memory	Disk	Network
Apps (6)				
Google Chrome (20)	0%	661.7 MB	0 MB/s	0 Mbps
Microsoft PowerPoint (5)	0%	377.0 MB	0 MB/s	0 Mbps
Notepad++ : a free (GNU) sourc...	0%	2.8 MB	0 MB/s	0 Mbps
Task Manager	0.4%	22.6 MB	0 MB/s	0 Mbps
Windows Explorer	1.3%	39.9 MB	0 MB/s	0 Mbps
WinSCP: SFTP, FTP, WebDAV an...	0%	2.5 MB	0 MB/s	0 Mbps
Background processes (43)				
AAM Updates Notifier Applicati...	0%	0.8 MB	0 MB/s	0 Mbps
Adobe Acrobat Update Service (...)	0%	0.4 MB	0 MB/s	0 Mbps
AMD External Events Client Mo...	0%	0.4 MB	0 MB/s	0 Mbps
AMD External Events Service M...	0%	0.2 MB	0 MB/s	0 Mbps
Antimalware Service Executable	0.1%	43.9 MB	0.1 MB/s	0 Mbps
Application Frame Host	0%	3.0 MB	0 MB/s	0 Mbps

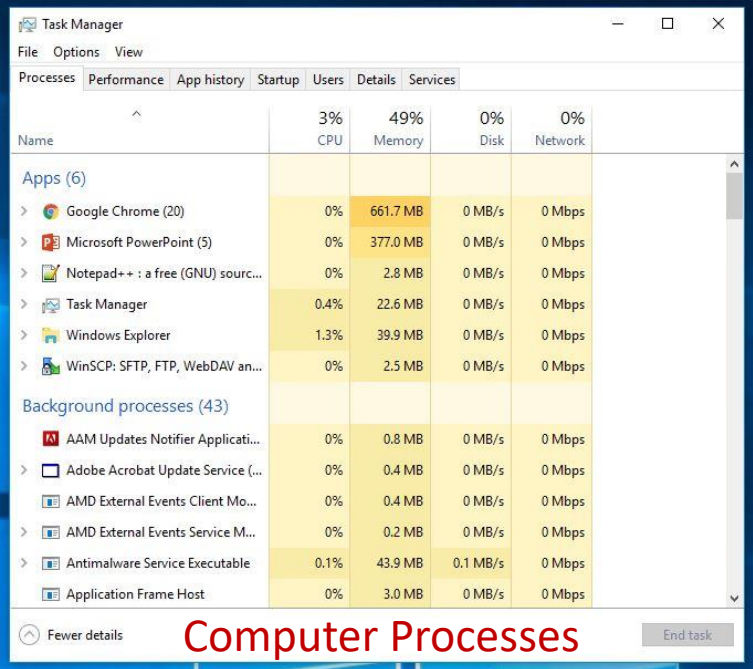
Computer Processes

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**



The screenshot shows the Windows Task Manager Performance tab, specifically the 'Processes' section. It displays a table of running processes with columns for Name, CPU usage, Memory usage, Disk usage, and Network usage. The 'Memory' column is highlighted in yellow. The total system memory usage is shown as 49%.

Name	CPU	Memory	Disk	Network
Apps (6)				
Google Chrome (20)	0%	661.7 MB	0 MB/s	0 Mbps
Microsoft PowerPoint (5)	0%	377.0 MB	0 MB/s	0 Mbps
Notepad++ : a free (GNU) sourc...	0%	2.8 MB	0 MB/s	0 Mbps
Task Manager	0.4%	22.6 MB	0 MB/s	0 Mbps
Windows Explorer	1.3%	39.9 MB	0 MB/s	0 Mbps
WinSCP: SFTP, FTP, WebDAV an...	0%	2.5 MB	0 MB/s	0 Mbps
Background processes (43)				
AAM Updates Notifier Applicati...	0%	0.8 MB	0 MB/s	0 Mbps
Adobe Acrobat Update Service (...)	0%	0.4 MB	0 MB/s	0 Mbps
AMD External Events Client Mo...	0%	0.4 MB	0 MB/s	0 Mbps
AMD External Events Service M...	0%	0.2 MB	0 MB/s	0 Mbps
Antimalware Service Executable	0.1%	43.9 MB	0.1 MB/s	0 Mbps
Application Frame Host	0%	3.0 MB	0 MB/s	0 Mbps

Computer Processes

Page Table Size... again

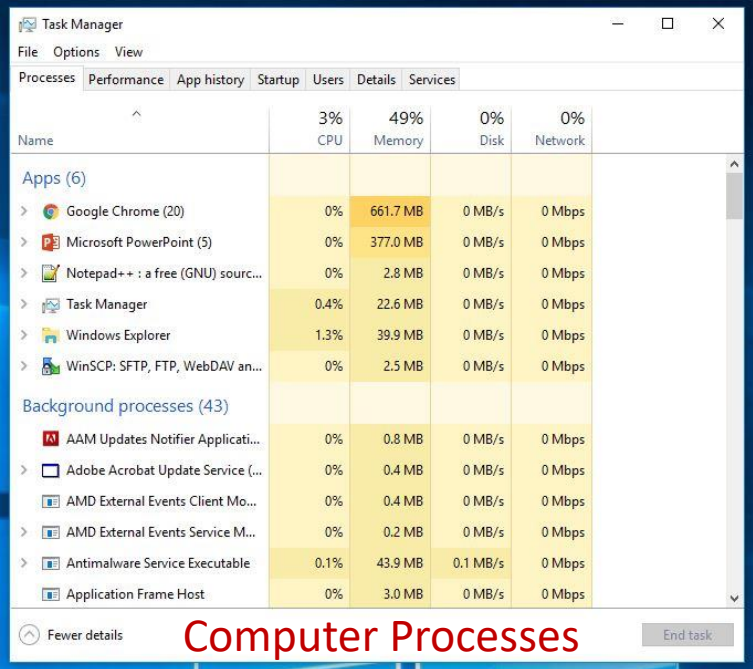
- Given a 32-bit machine, 4 kB pages...
 - 1 million PTEs
 - 32 bits (machine) – 12 bits (page offset) = 20 bits = 1 million
 - Each PTE is around 4 Bytes wide
 - 20 bits (Physical Page) + permission bits

- 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**



The screenshot shows the Windows Task Manager Performance tab, specifically the 'Memory' section. It displays a table of processes and their memory usage. The total system memory usage is 49%. The table lists various applications and background processes, including Google Chrome (20 instances), Microsoft PowerPoint (5 instances), Notepad++, Task Manager, Windows Explorer, and WinSCP. The memory usage for these processes is as follows:

Name	CPU	Memory	Disk	Network
Apps (6)				
Google Chrome (20)	0%	661.7 MB	0 MB/s	0 Mbps
Microsoft PowerPoint (5)	0%	377.0 MB	0 MB/s	0 Mbps
Notepad++ : a free (GNU) sourc...	0%	2.8 MB	0 MB/s	0 Mbps
Task Manager	0.4%	22.6 MB	0 MB/s	0 Mbps
Windows Explorer	1.3%	39.9 MB	0 MB/s	0 Mbps
WinSCP: SFTP, FTP, WebDAV an...	0%	2.5 MB	0 MB/s	0 Mbps
Background processes (43)				
AAM Updates Notifier Applicati...	0%	0.8 MB	0 MB/s	0 Mbps
Adobe Acrobat Update Service (...)	0%	0.4 MB	0 MB/s	0 Mbps
AMD External Events Client Mo...	0%	0.4 MB	0 MB/s	0 Mbps
AMD External Events Service M...	0%	0.2 MB	0 MB/s	0 Mbps
Antimalware Service Executable	0.1%	43.9 MB	0.1 MB/s	0 Mbps
Application Frame Host	0%	3.0 MB	0 MB/s	0 Mbps

Page Table Size... again

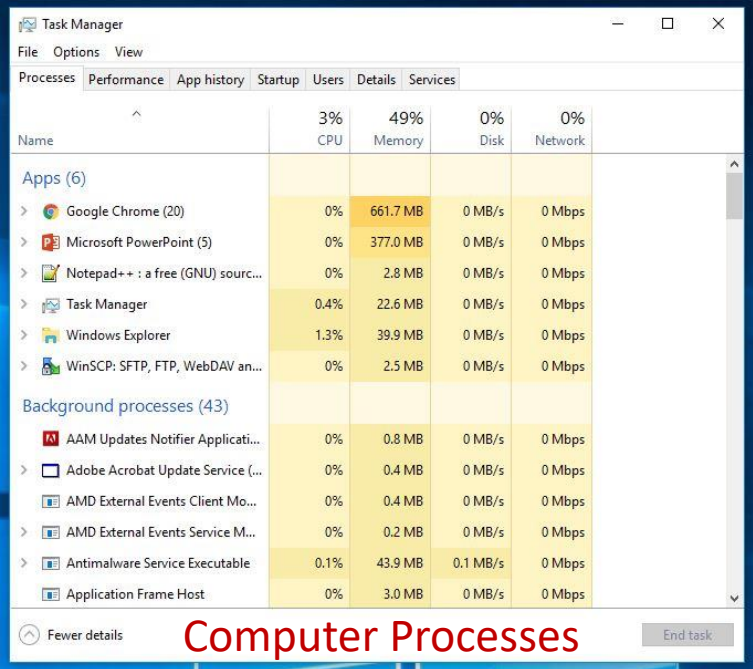
- Given a 32-bit machine, 4 kB pages...
 - 1 million PTEs
 - 32 bits (machine) – 12 bits (page offset) = 20 bits = 1 million
 - Each PTE is around 4 Bytes wide
 - 20 bits (Physical Page) + permission bits

- 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**



The screenshot shows the Windows Task Manager Performance tab, specifically the 'Processes' section. It displays a table of running processes with columns for Name, CPU usage, Memory usage, Disk usage, and Network usage. The 'Memory' column is highlighted in yellow. The processes are grouped into 'Apps (6)' and 'Background processes (43)'. The total memory usage for all processes is 49%.

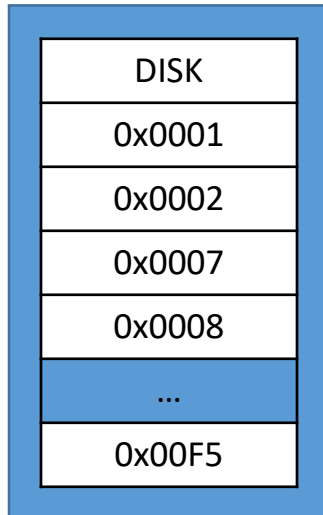
Name	CPU	Memory	Disk	Network
Apps (6)				
Google Chrome (20)	0%	661.7 MB	0 MB/s	0 Mbps
Microsoft PowerPoint (5)	0%	377.0 MB	0 MB/s	0 Mbps
Notepad++ : a free (GNU) sourc...	0%	2.8 MB	0 MB/s	0 Mbps
Task Manager	0.4%	22.6 MB	0 MB/s	0 Mbps
Windows Explorer	1.3%	39.9 MB	0 MB/s	0 Mbps
WinSCP: SFTP, FTP, WebDAV an...	0%	2.5 MB	0 MB/s	0 Mbps
Background processes (43)				
AAM Updates Notifier Applicati...	0%	0.8 MB	0 MB/s	0 Mbps
Adobe Acrobat Update Service (...)	0%	0.4 MB	0 MB/s	0 Mbps
AMD External Events Client Mo...	0%	0.4 MB	0 MB/s	0 Mbps
AMD External Events Service M...	0%	0.2 MB	0 MB/s	0 Mbps
Antimalware Service Executable	0.1%	43.9 MB	0.1 MB/s	0 Mbps
Application Frame Host	0%	3.0 MB	0 MB/s	0 Mbps

Multi-Level Page Tables

1st Level Page Table

4kB [1,024 entries]

Size of 1 Page



Main Memory

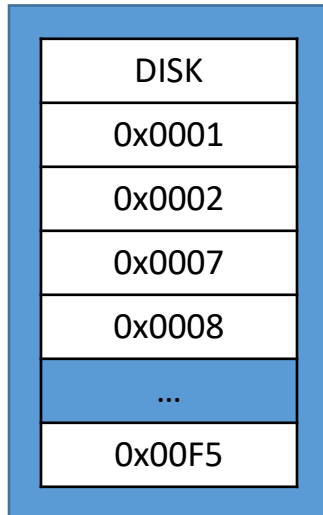
A large, vertical green rectangle representing Main Memory.

Multi-Level Page Tables

1st Level Page Table

4kB [1,024 entries]

Size of 1 Page



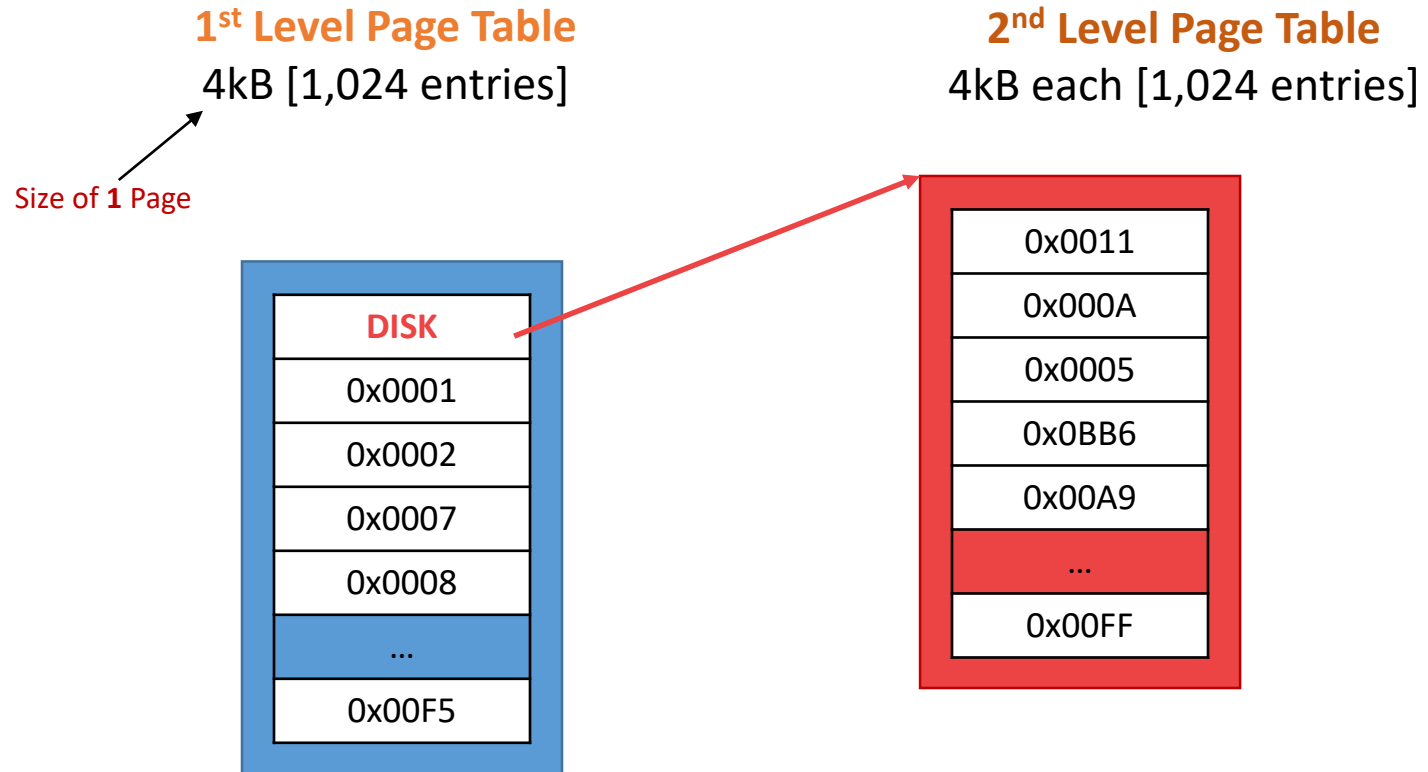
2nd Level Page Table

4kB each [1,024 entries]

Main Memory

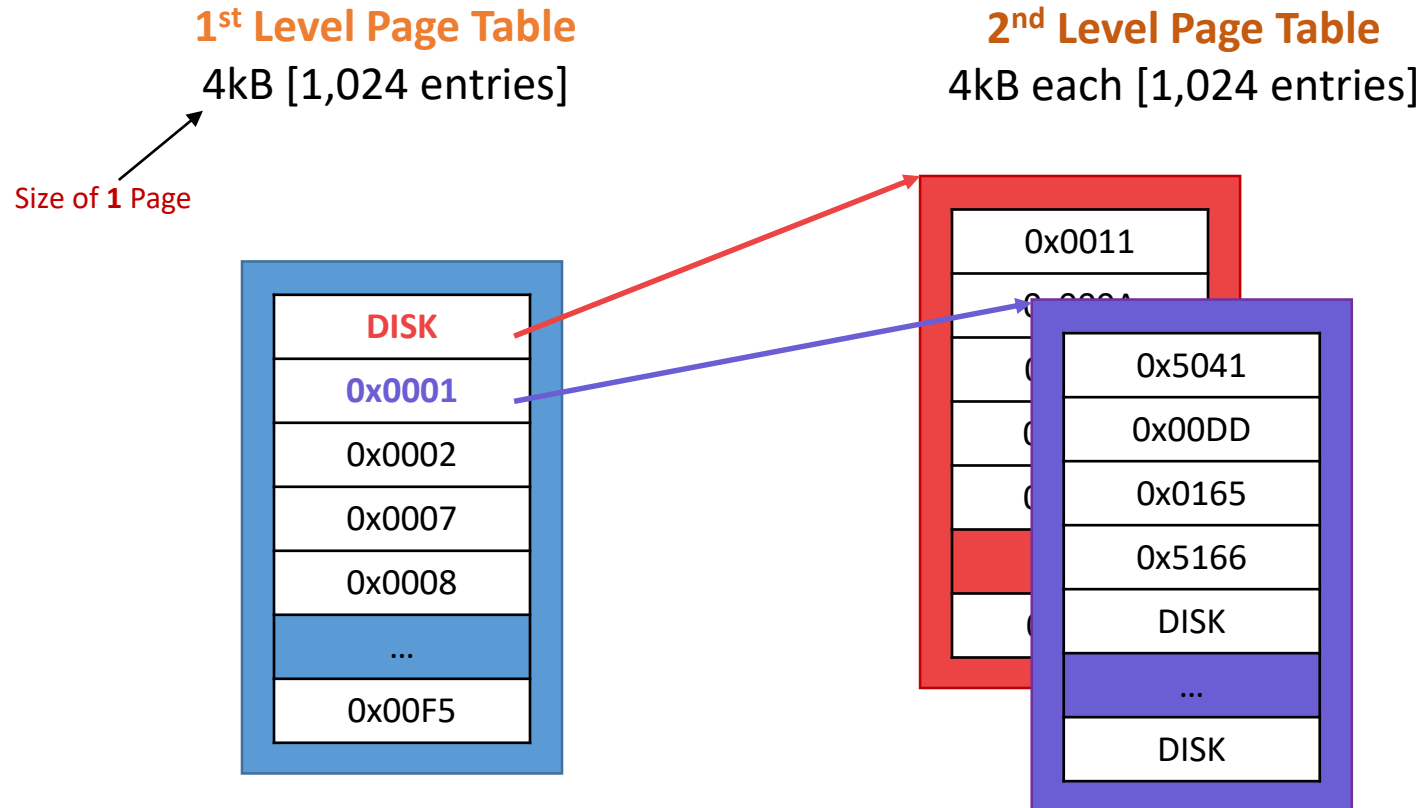
A large, solid green vertical rectangle representing the main memory.

Multi-Level Page Tables

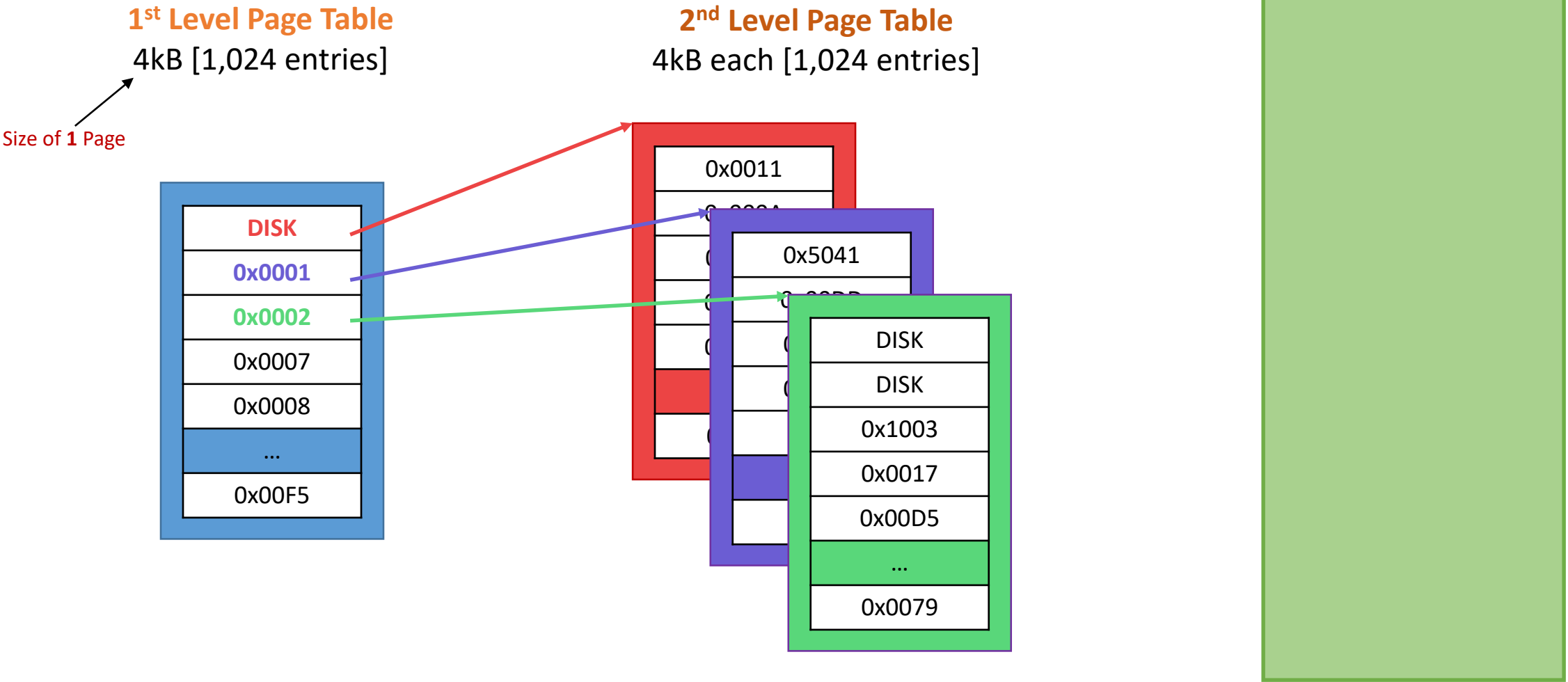


Main Memory

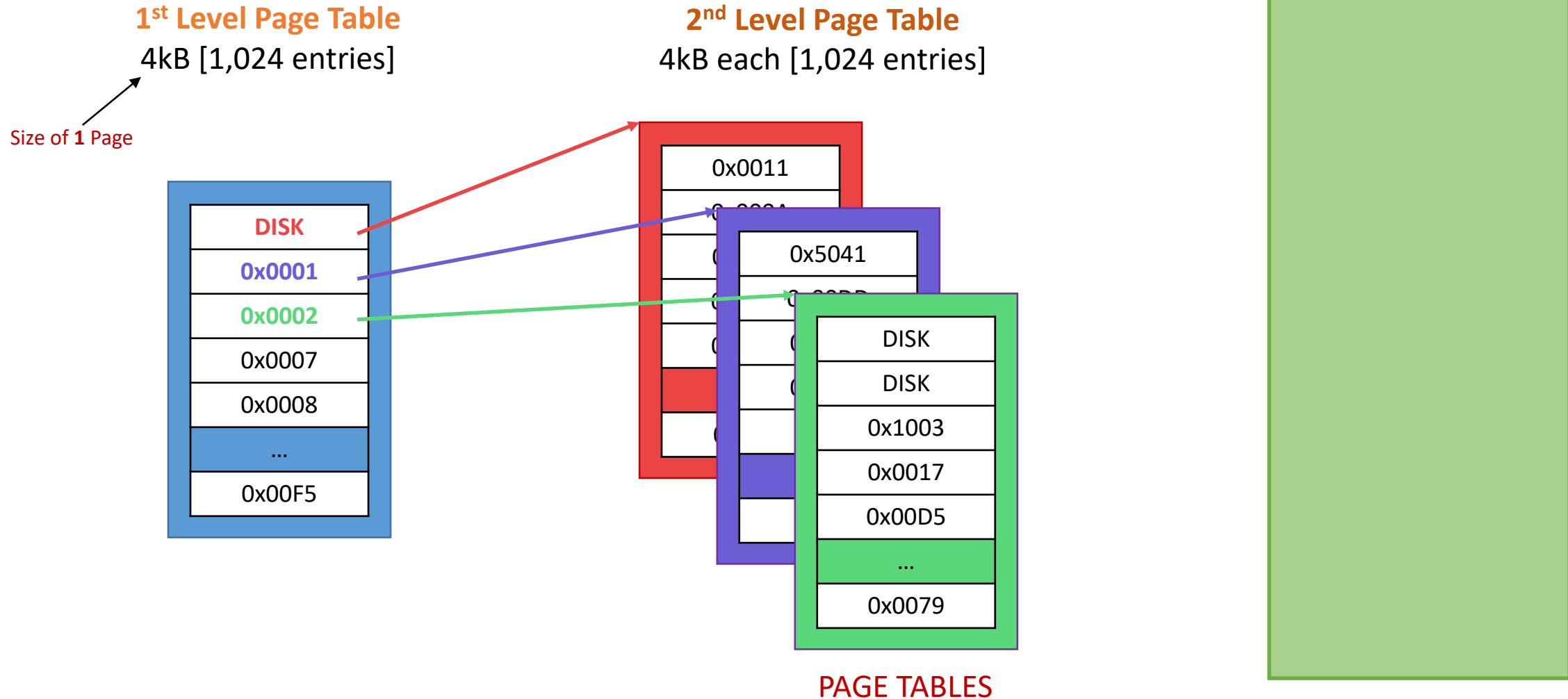
Multi-Level Page Tables



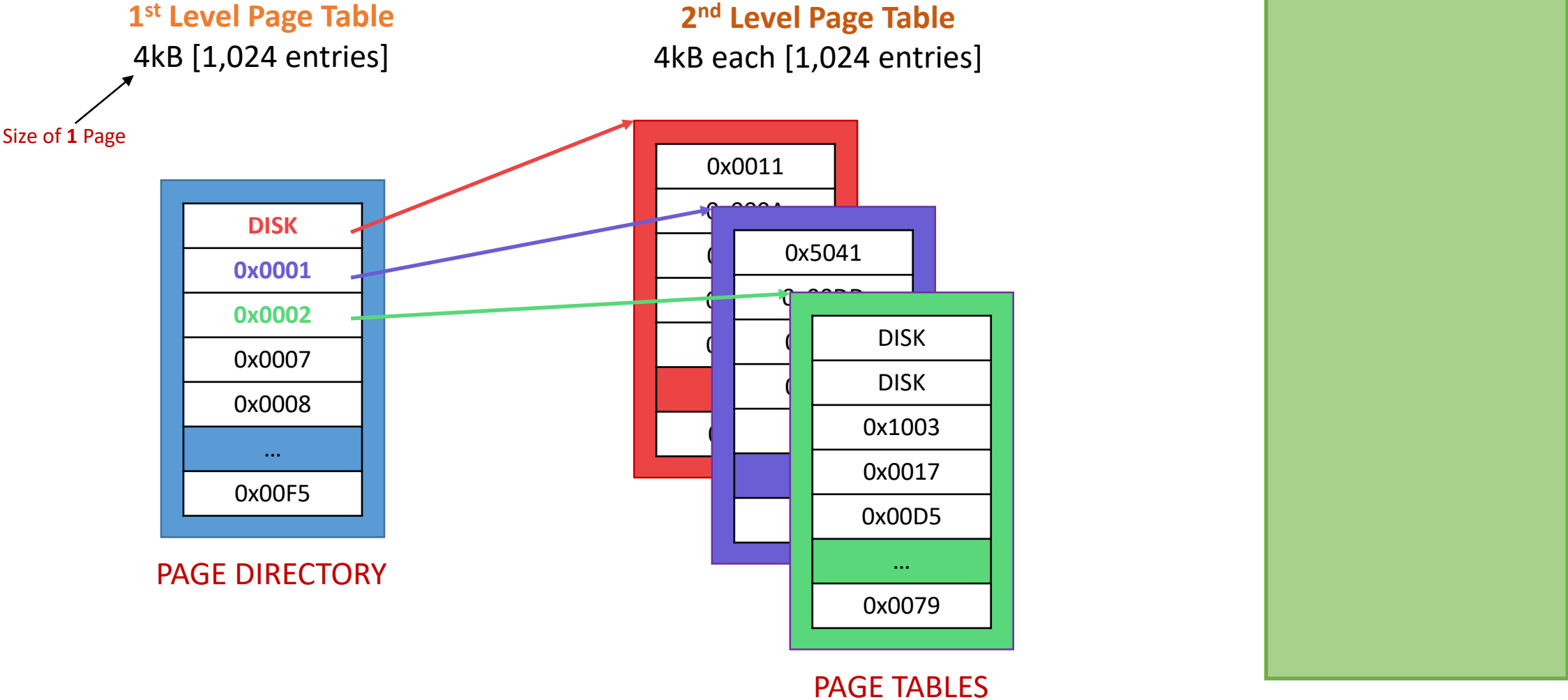
Multi-Level Page Tables



Multi-Level Page Tables



Multi-Level Page Tables

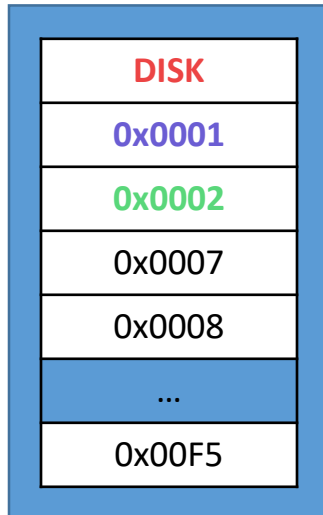


Multi-Level Page Tables

1st Level Page Table

4kB [1,024 entries]

Size of 1 Page

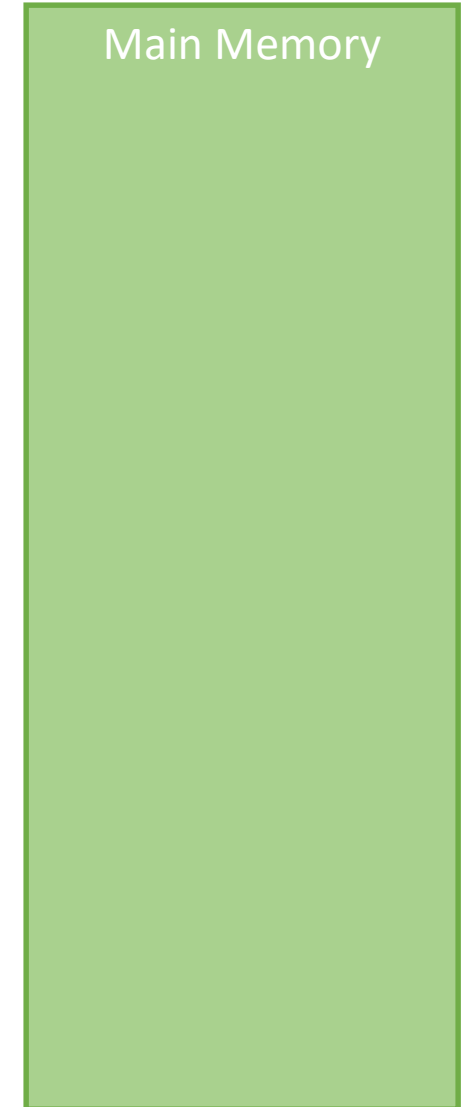


PAGE DIRECTORY

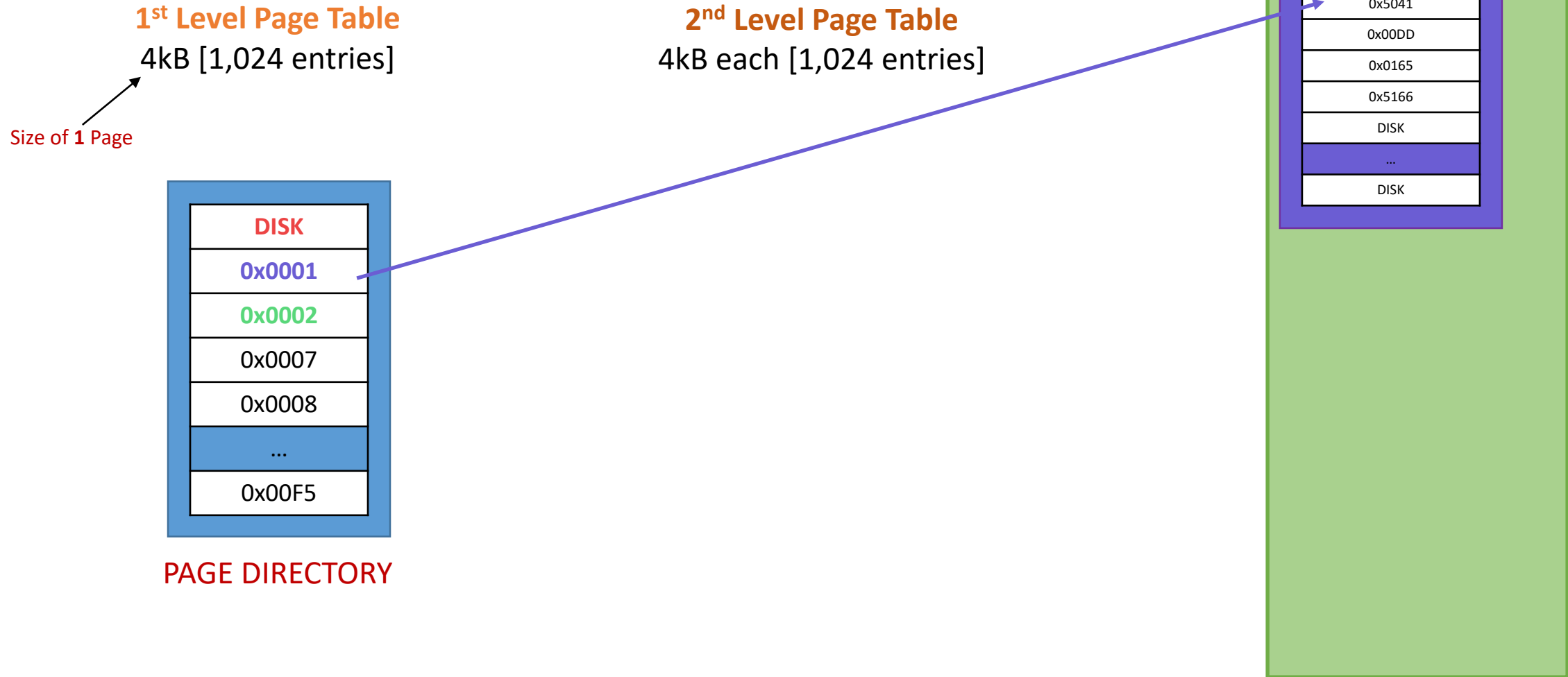
2nd Level Page Table

4kB each [1,024 entries]

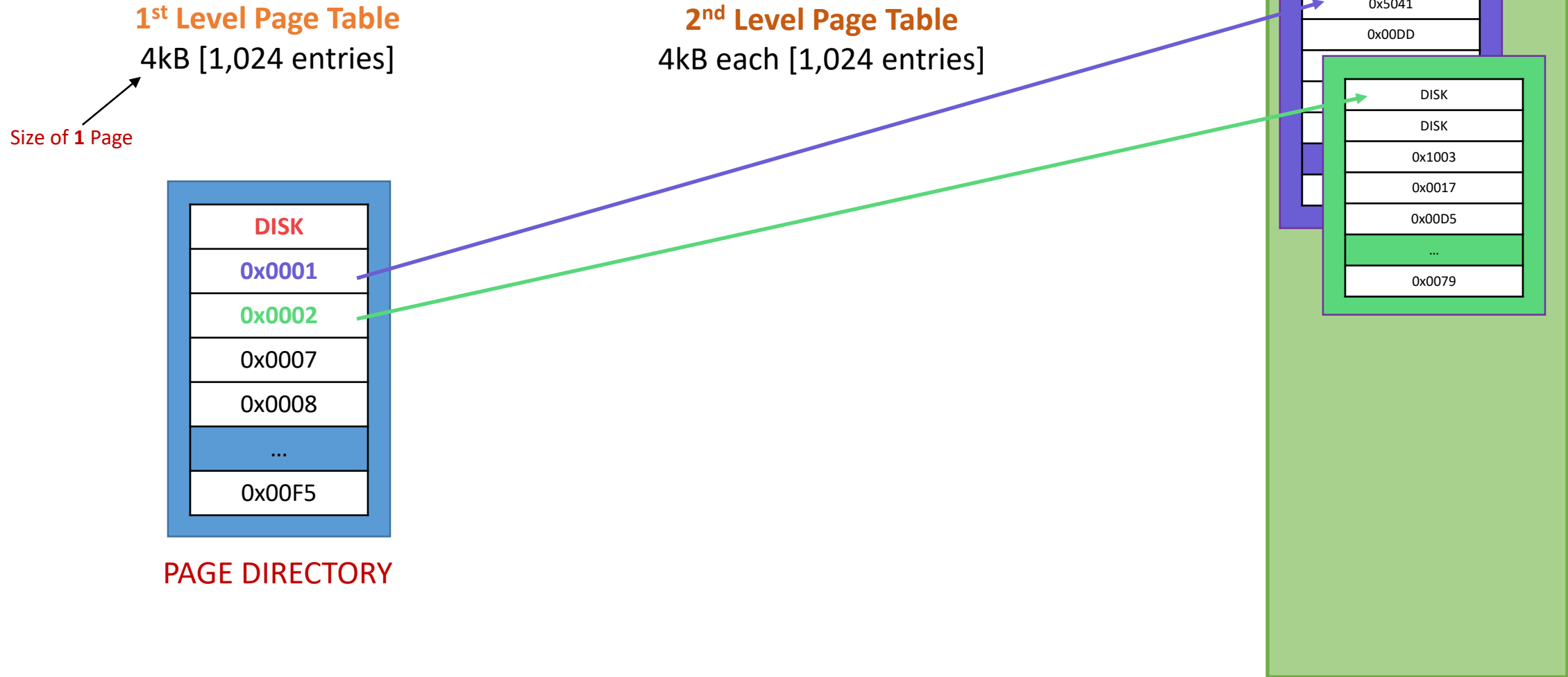
Main Memory



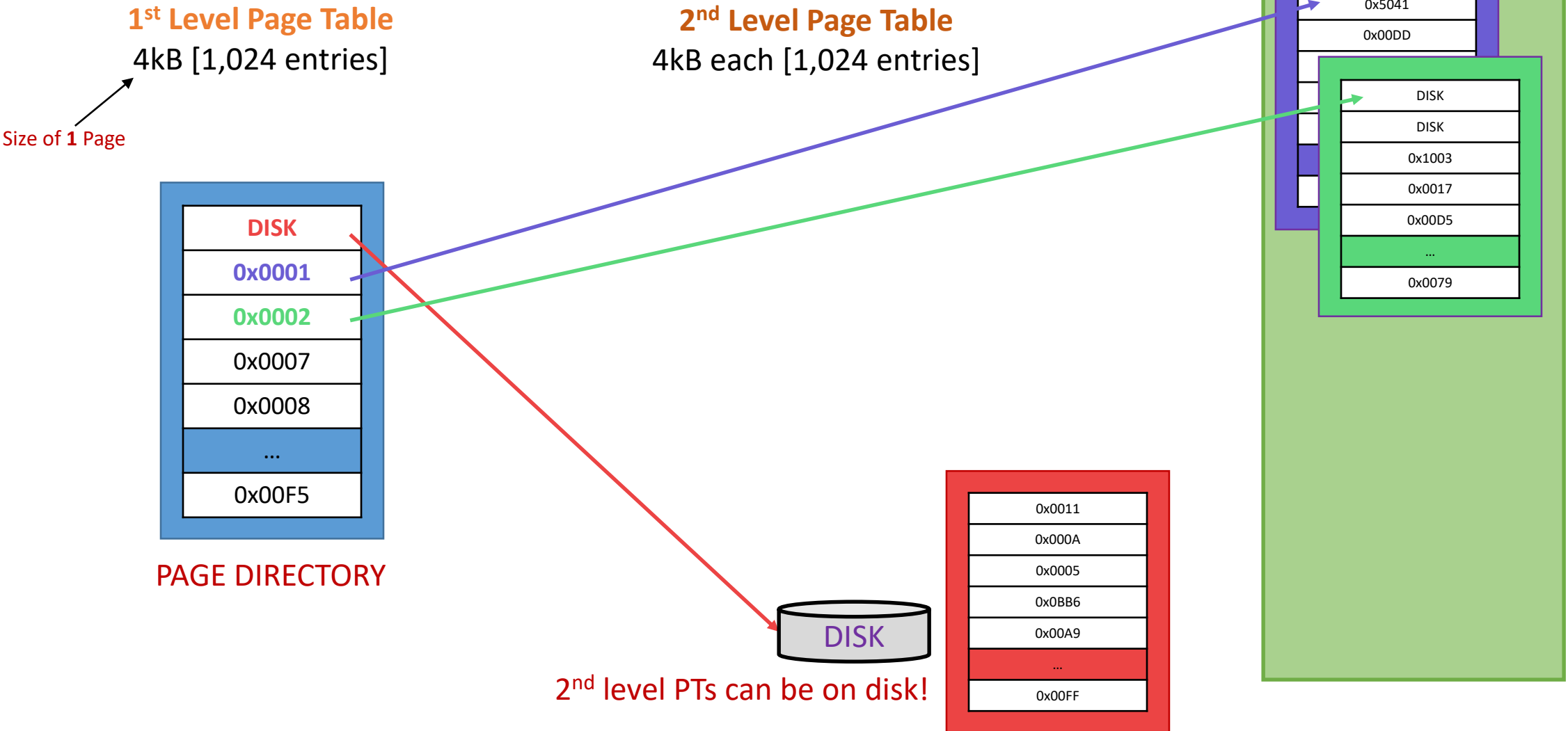
Multi-Level Page Tables



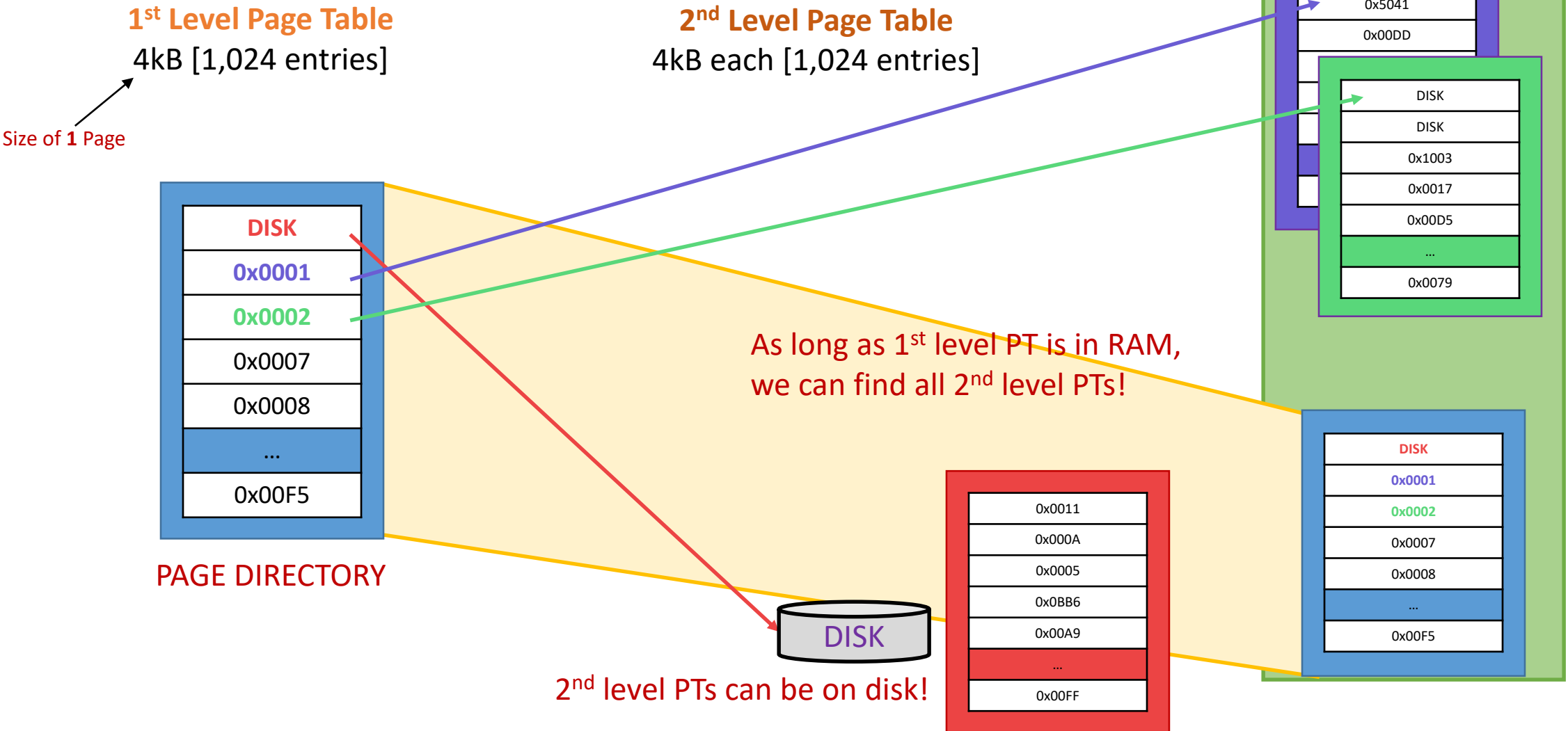
Multi-Level Page Tables



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

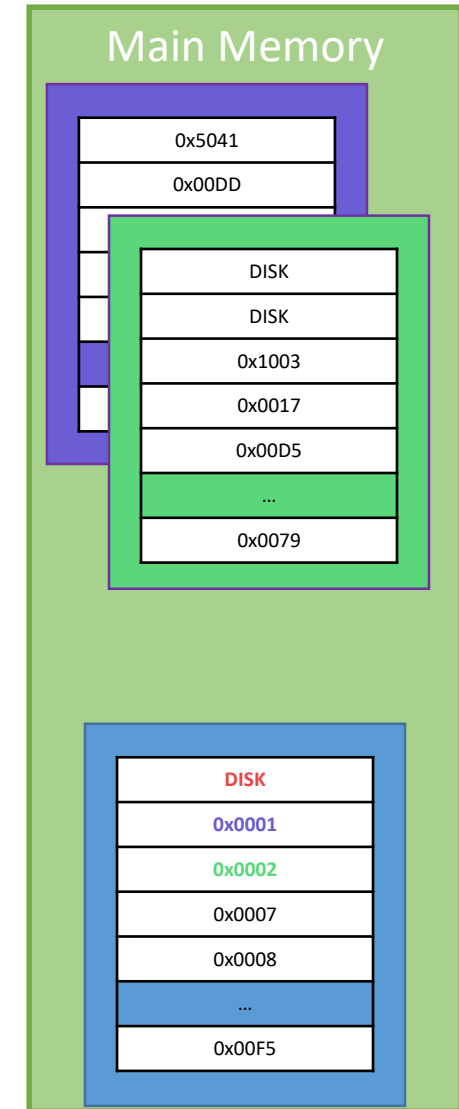
We must always keep the 1st level page table in RAM (4 kB) and we need at least the 2nd level page table for the application data.

Multi-Level Page Table Translation

Virtual Address [32 bit]



Physical Address [28 bits]

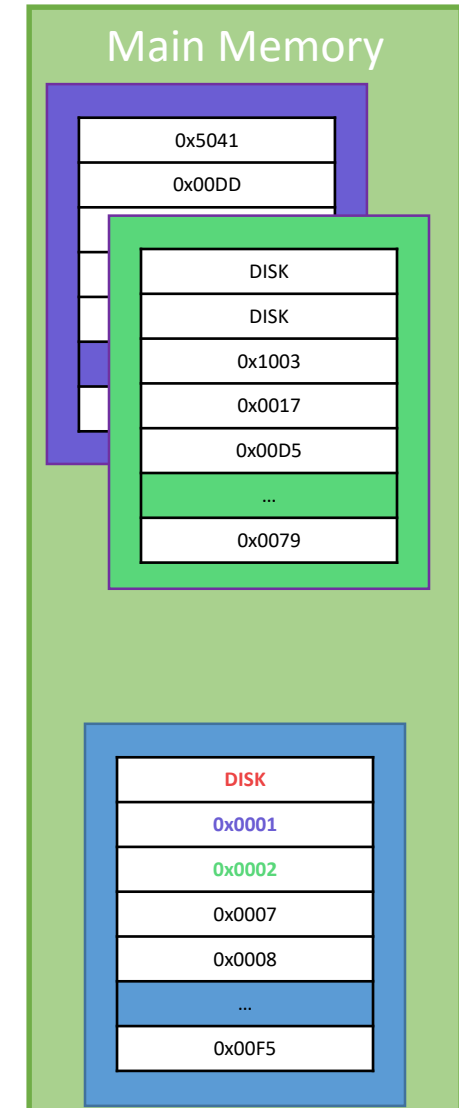


Multi-Level Page Table Translation

Virtual Address [32 bit]



Physical Address [28 bits]



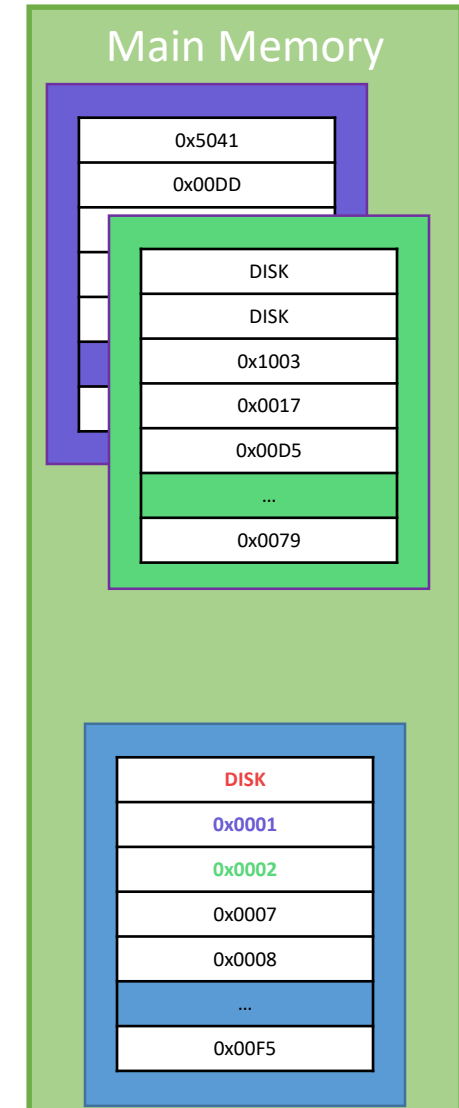
Multi-Level Page Table Translation

VA: 0x00403 713

Virtual Address [32 bit]



Physical Address [28 bits]



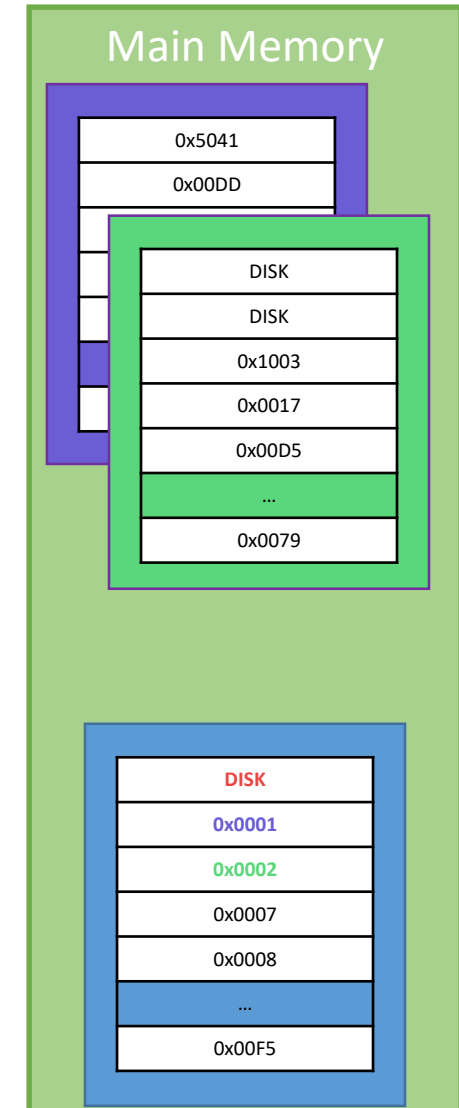
Multi-Level Page Table Translation

VA: 0x00403 713

Virtual Address [32 bit]



Physical Address [28 bits]



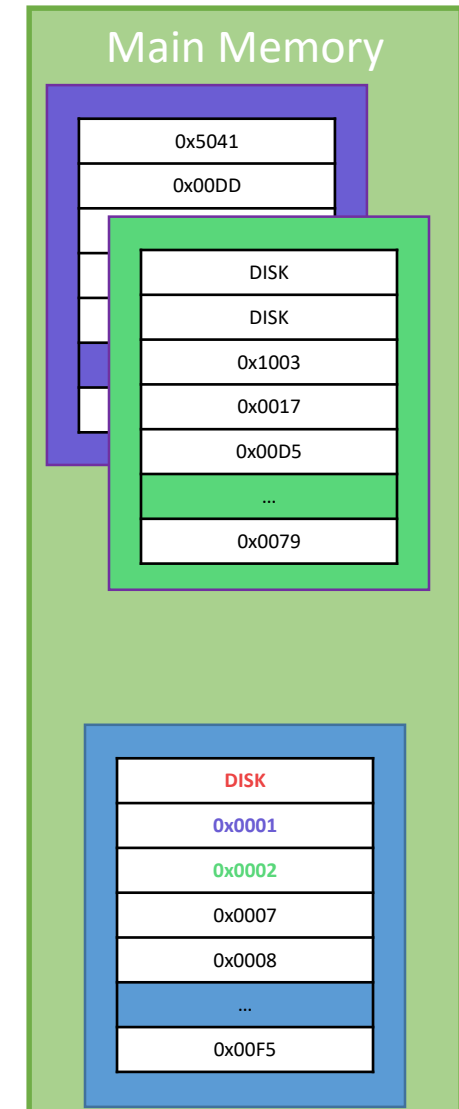
Multi-Level Page Table Translation

VA: 0x00403 713

Virtual Address [32 bit]



Physical Address [28 bits]



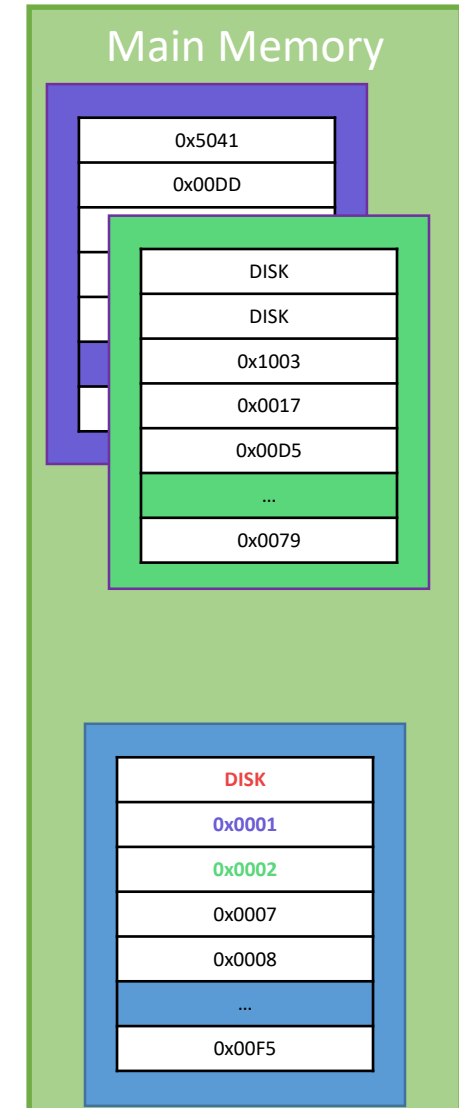
Multi-Level Page Table Translation

VA: 0x00403 713

Virtual Address [32 bit]



Physical Address [28 bits]



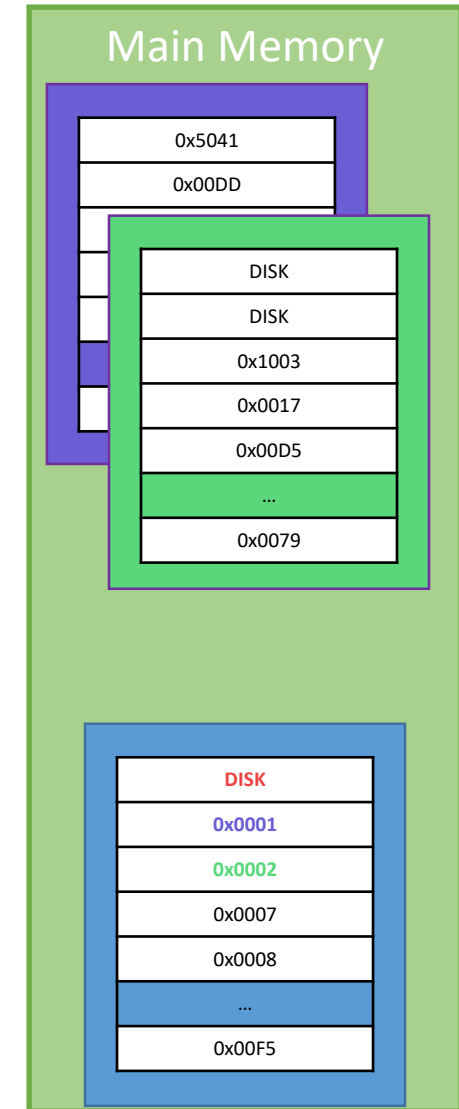
Multi-Level Page Table Translation

VA: 0x00403 713

Virtual Address [32 bit]



Physical Address [28 bits]



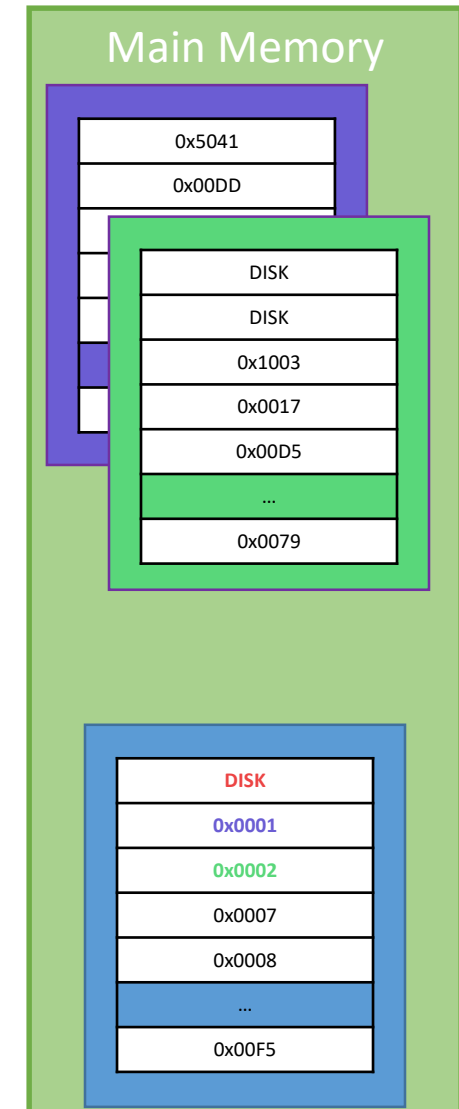
Multi-Level Page Table Translation

VA: 0x00403 713

Virtual Address [32 bit]



Physical Address [28 bits]



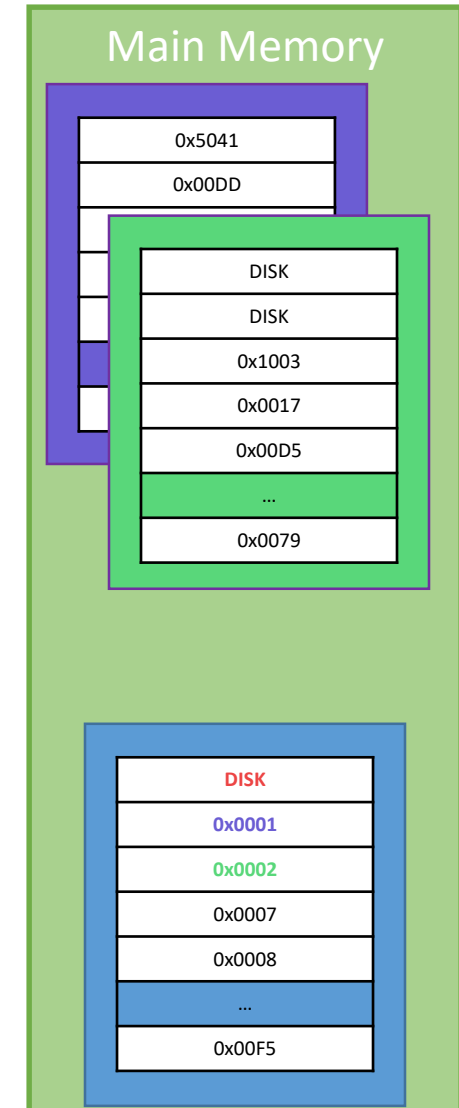
Multi-Level Page Table Translation

VA: 0x00403 713

Virtual Address [32 bit]



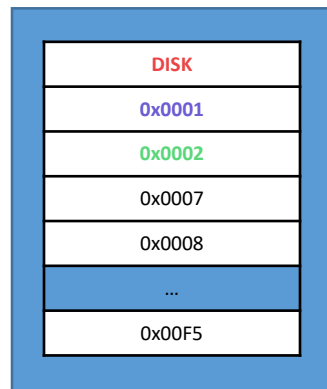
Physical Address [28 bits]



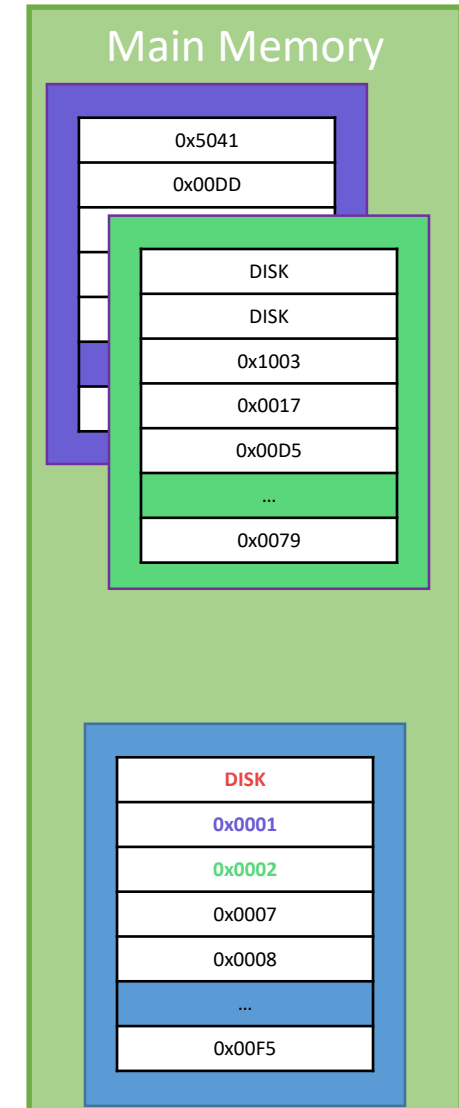
Multi-Level Page Table Translation

VA: 0x00403 713

Virtual Address [32 bit]



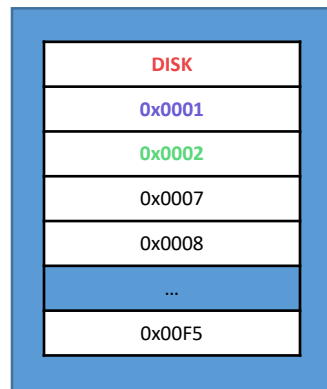
Physical Address [28 bits]



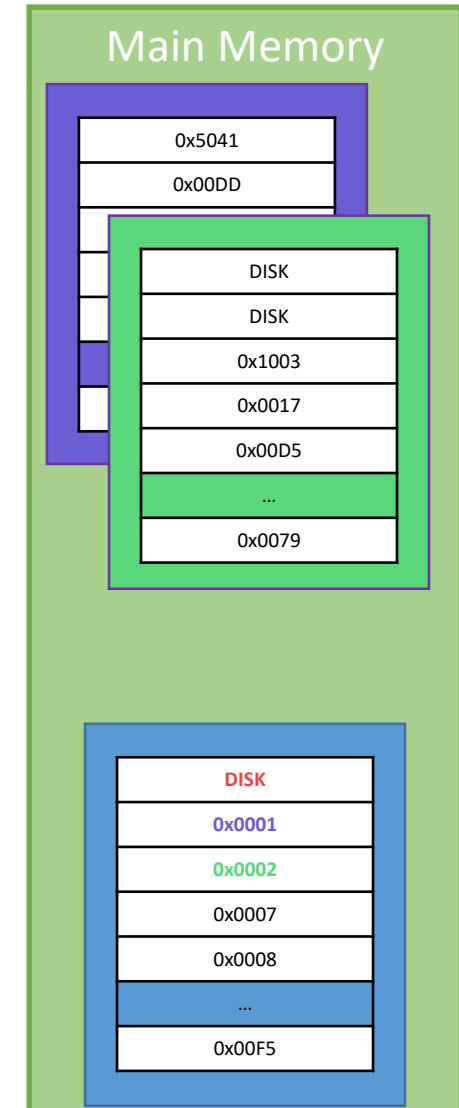
Multi-Level Page Table Translation

VA: 0x00403 713

Virtual Address [32 bit]



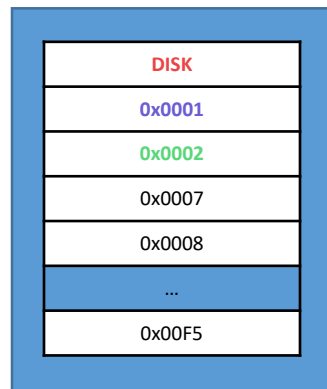
Physical Address [28 bits]



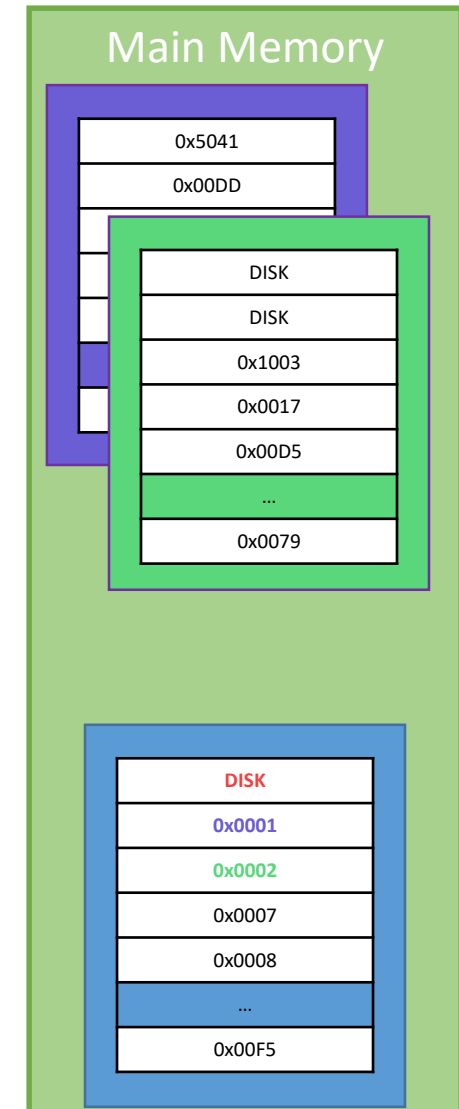
Multi-Level Page Table Translation

VA: 0x00403 713

Virtual Address [32 bit]



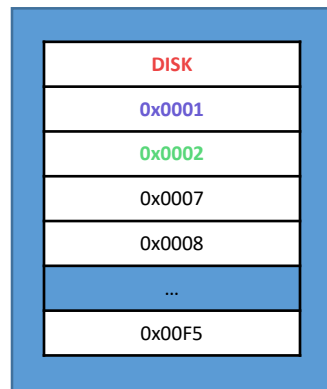
Physical Address [28 bits]



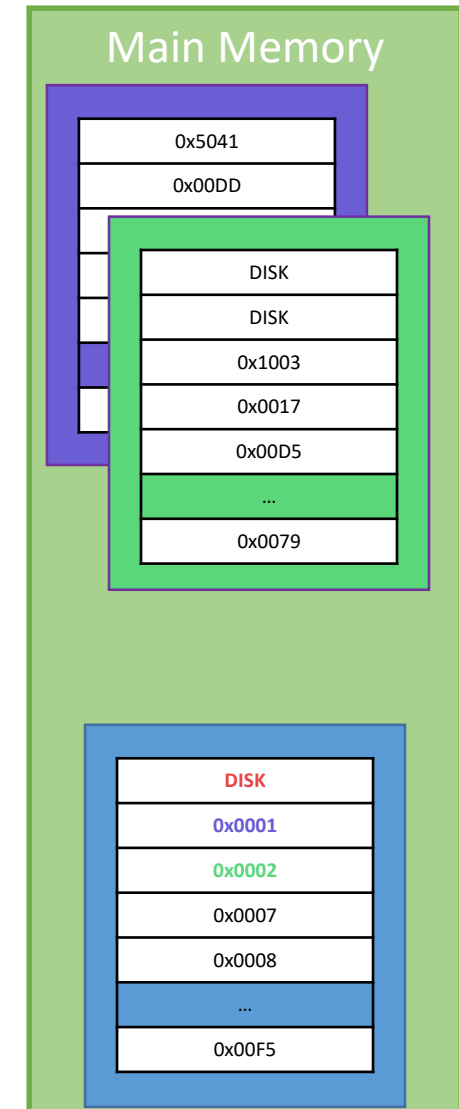
Multi-Level Page Table Translation

VA: 0x00403 713

Virtual Address [32 bit]



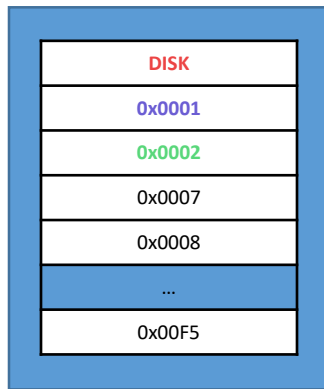
Physical Address [28 bits]



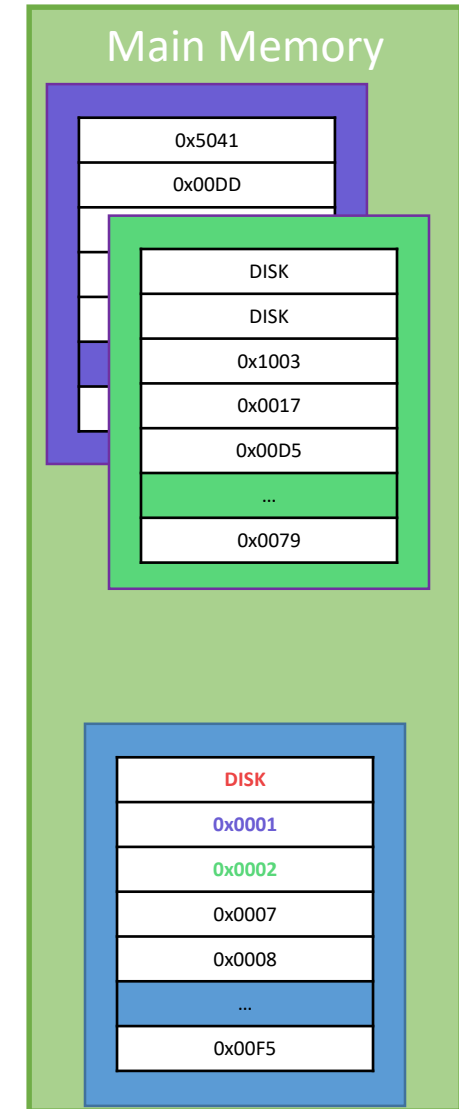
Multi-Level Page Table Translation

VA: 0x00403 713

Virtual Address [32 bit]



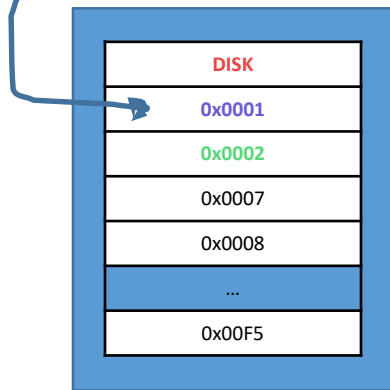
Physical Address [28 bits]



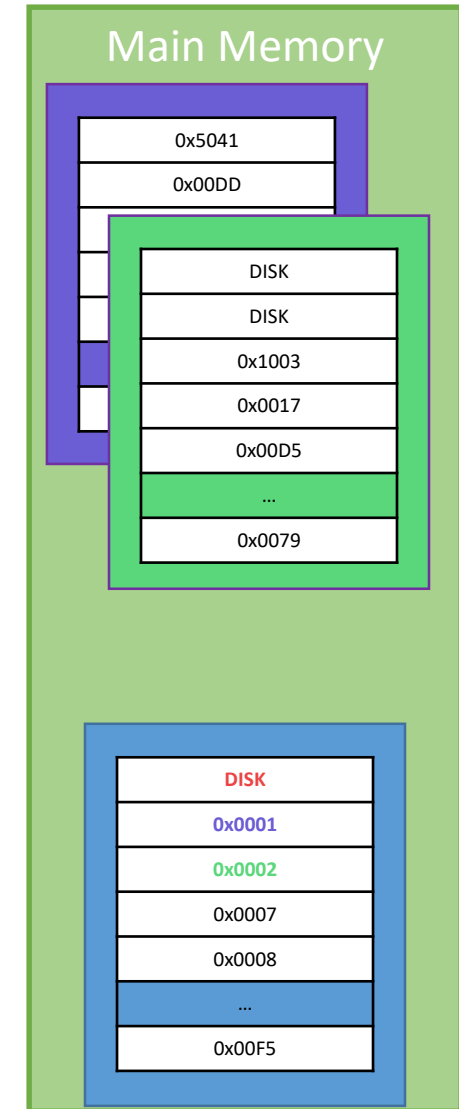
Multi-Level Page Table Translation

VA: 0x00403 713

Virtual Address [32 bit]



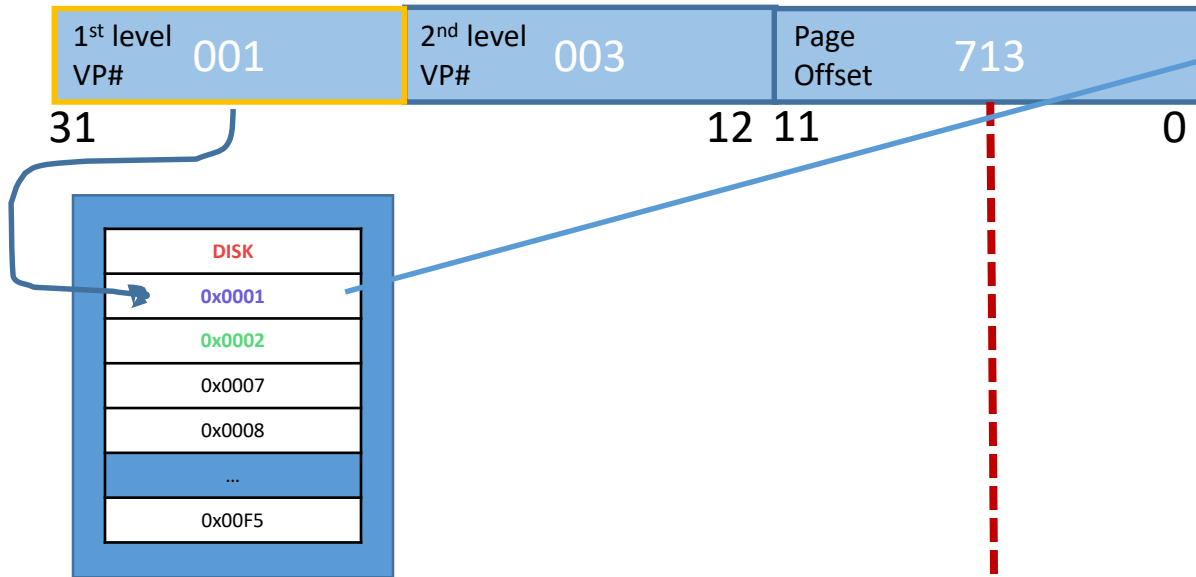
Physical Address [28 bits]



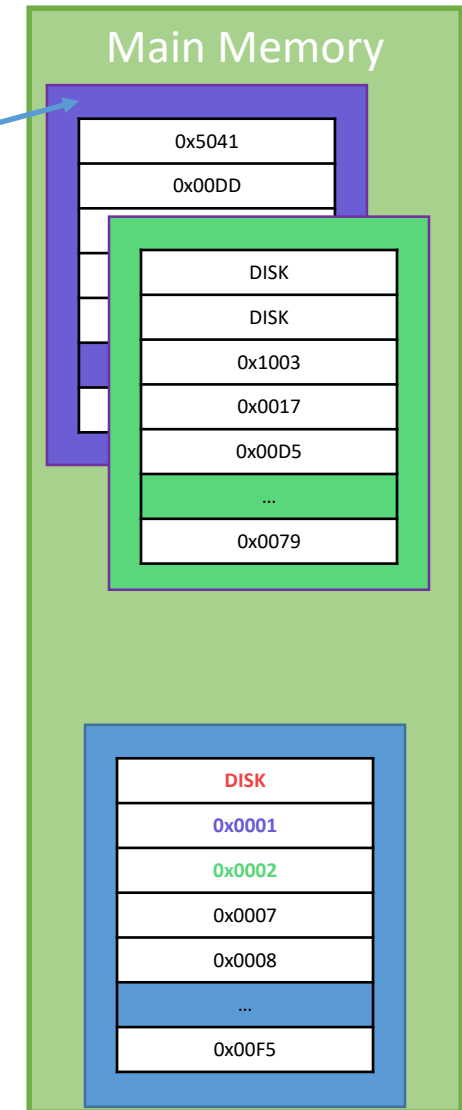
Multi-Level Page Table Translation

VA: 0x00403 713

Virtual Address [32 bit]



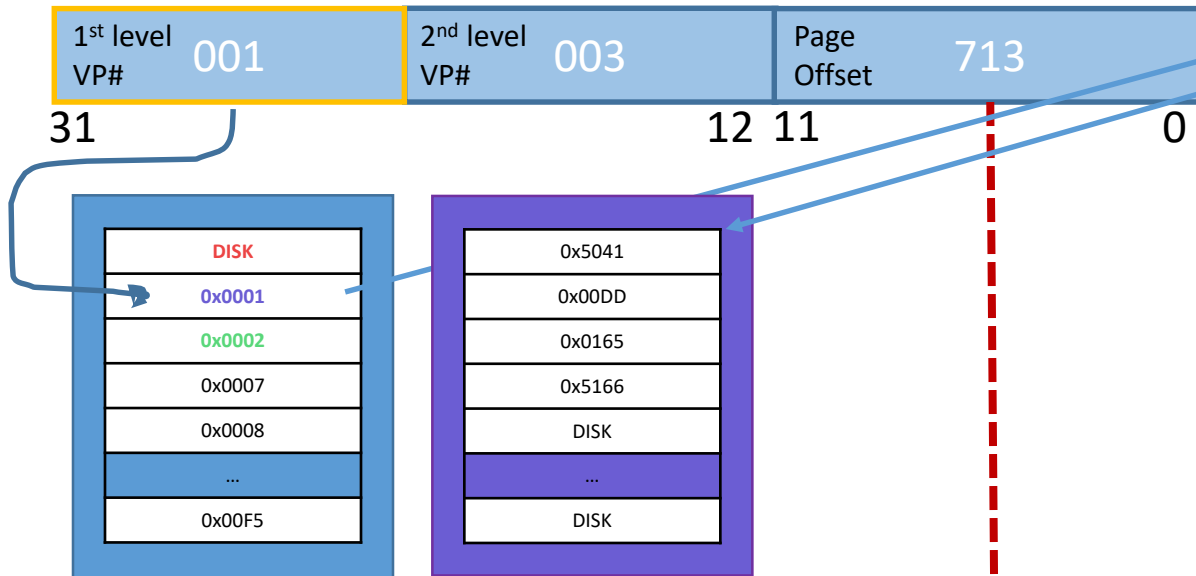
Physical Address [28 bits]



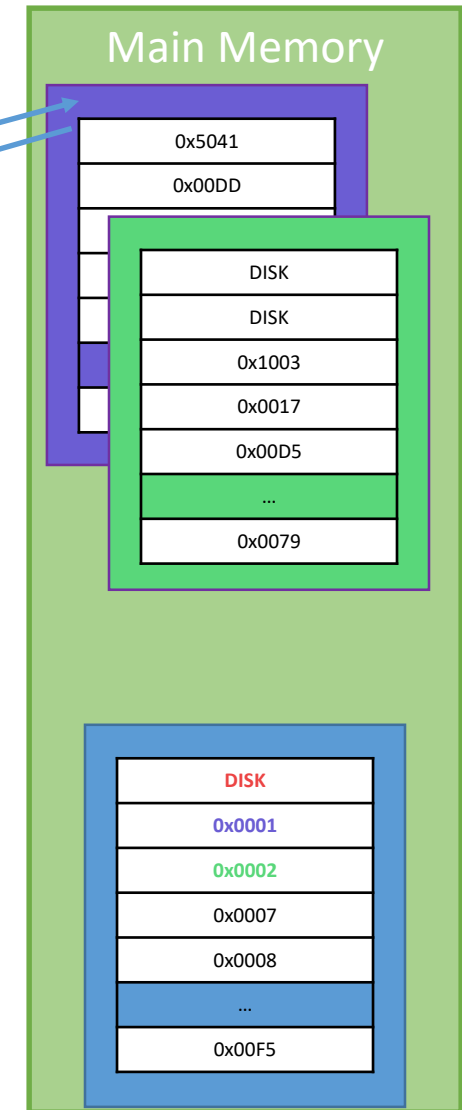
Multi-Level Page Table Translation

VA: 0x00403 713

Virtual Address [32 bit]



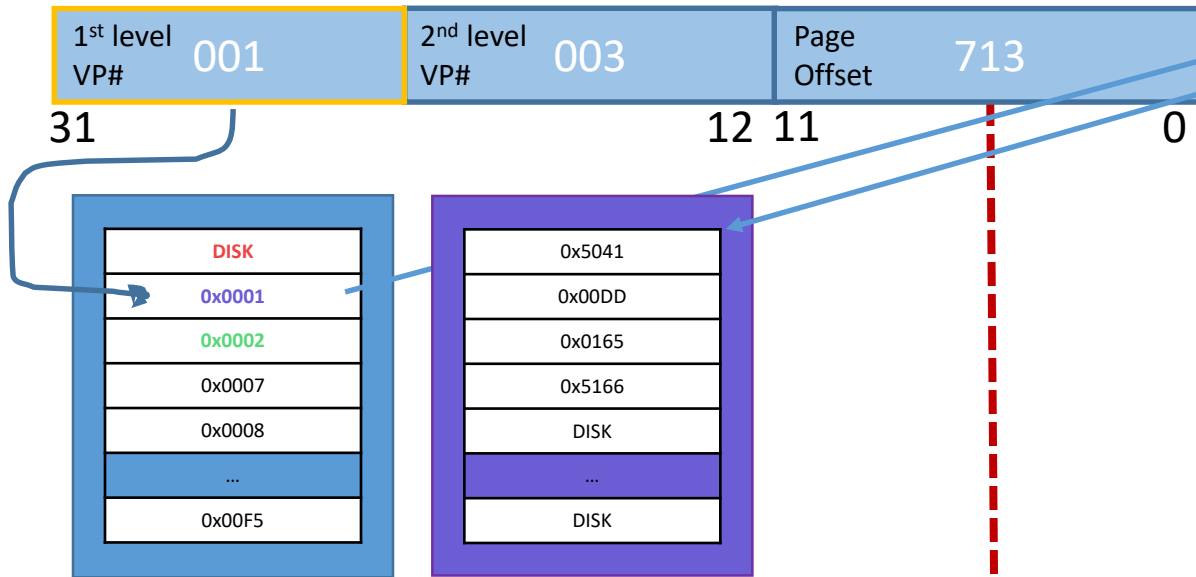
Physical Address [28 bits]



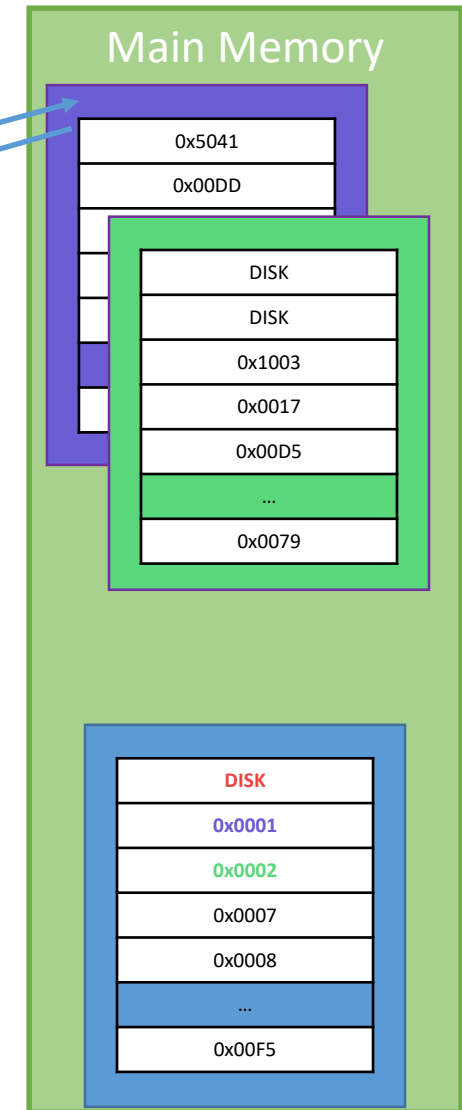
Multi-Level Page Table Translation

VA: 0x00403 713

Virtual Address [32 bit]



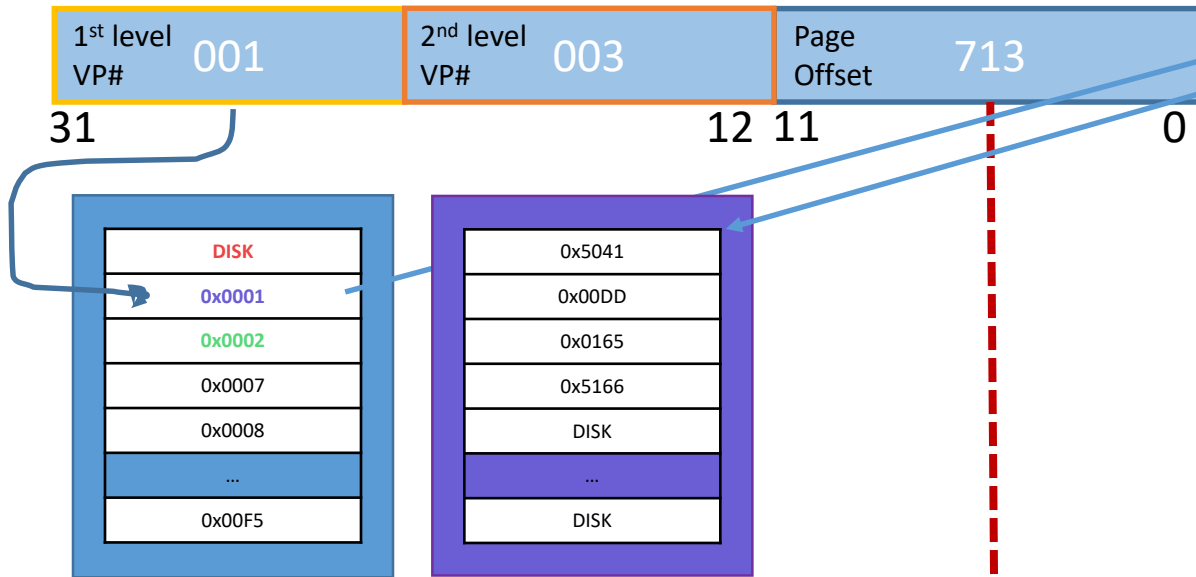
Physical Address [28 bits]



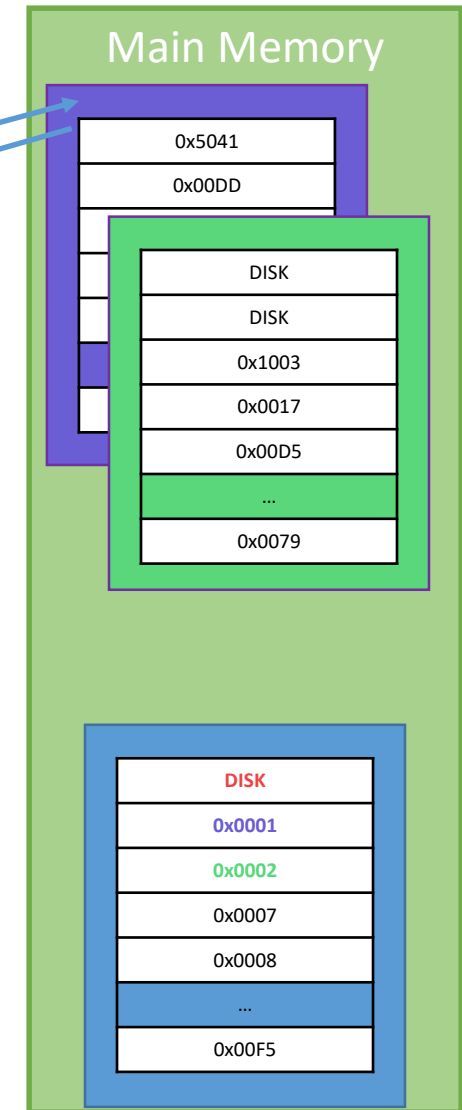
Multi-Level Page Table Translation

VA: 0x00403 713

Virtual Address [32 bit]



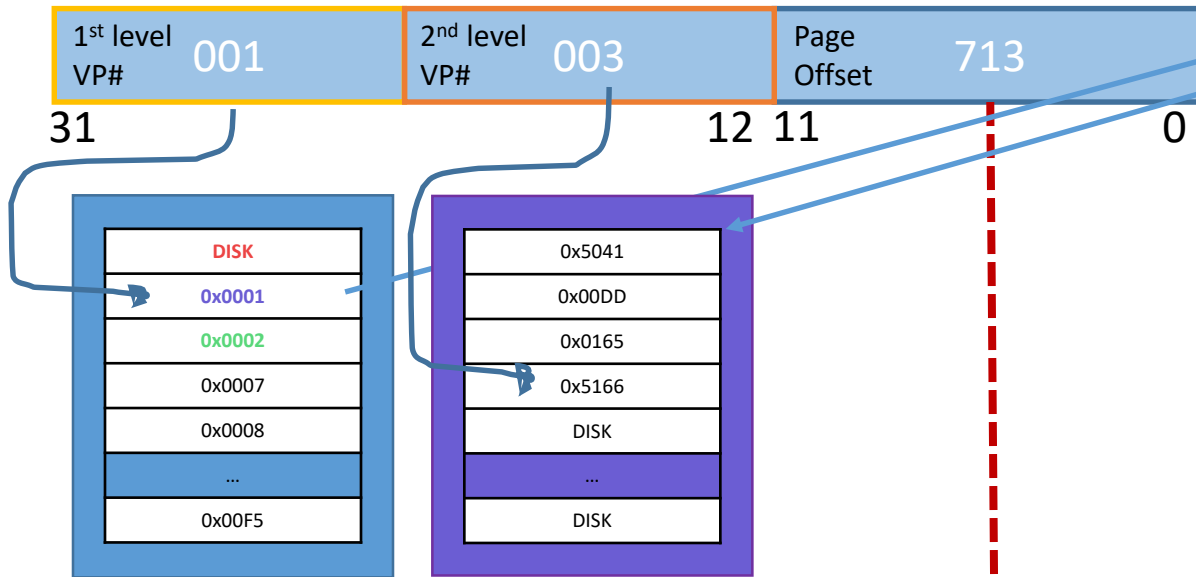
Physical Address [28 bits]



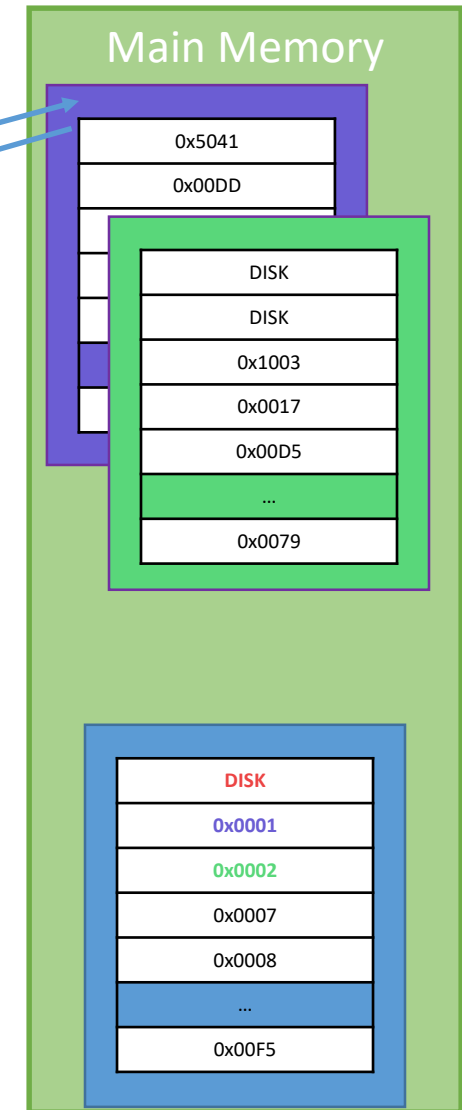
Multi-Level Page Table Translation

VA: 0x00403 713

Virtual Address [32 bit]



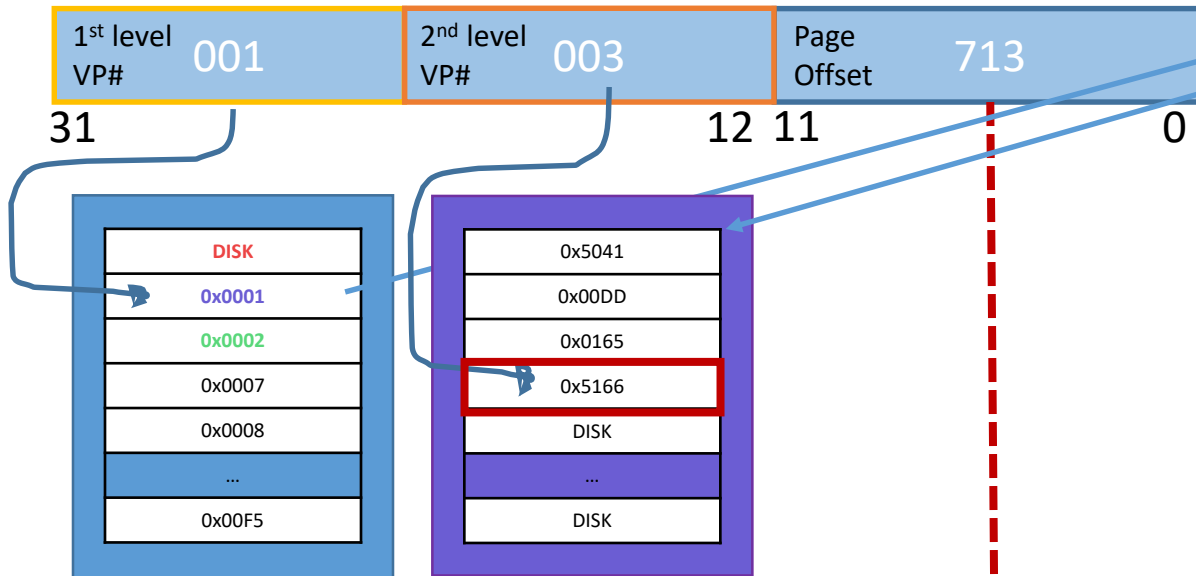
Physical Address [28 bits]



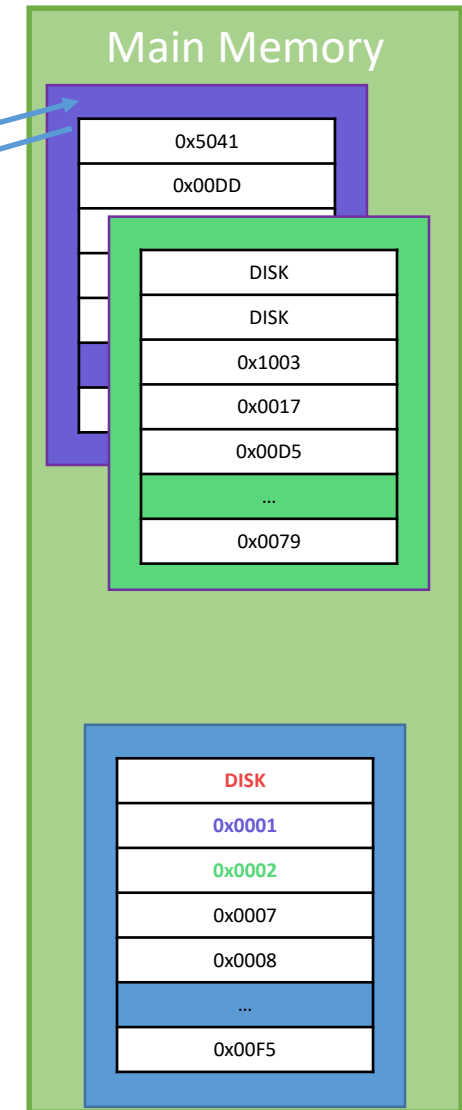
Multi-Level Page Table Translation

VA: 0x00403 713

Virtual Address [32 bit]



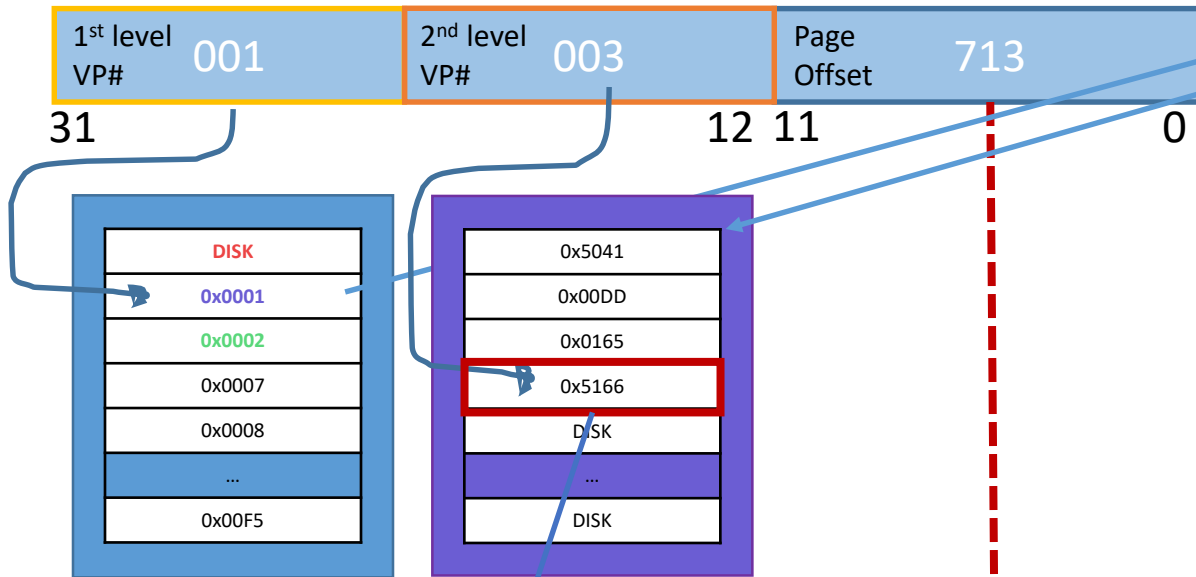
Physical Address [28 bits]



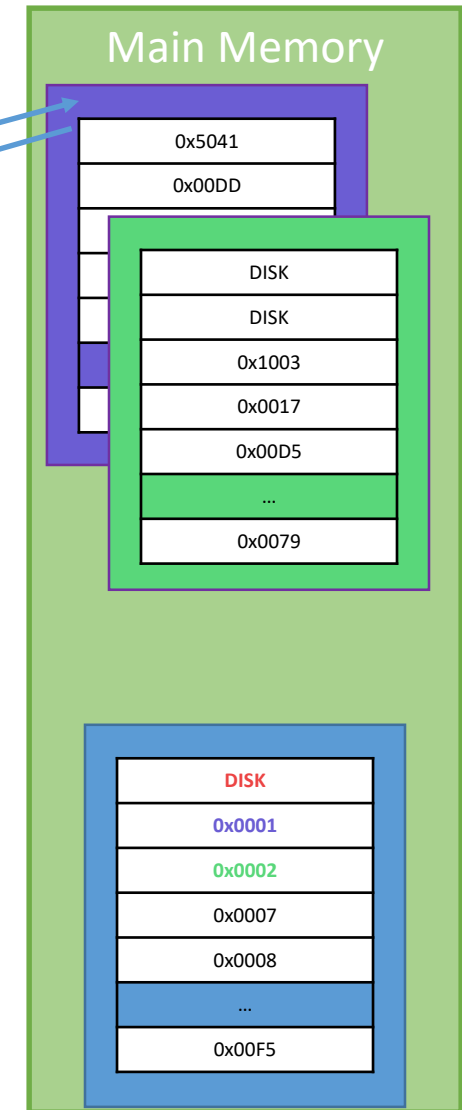
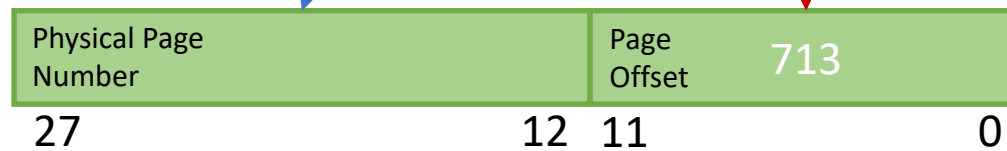
Multi-Level Page Table Translation

VA: 0x00403 713

Virtual Address [32 bit]



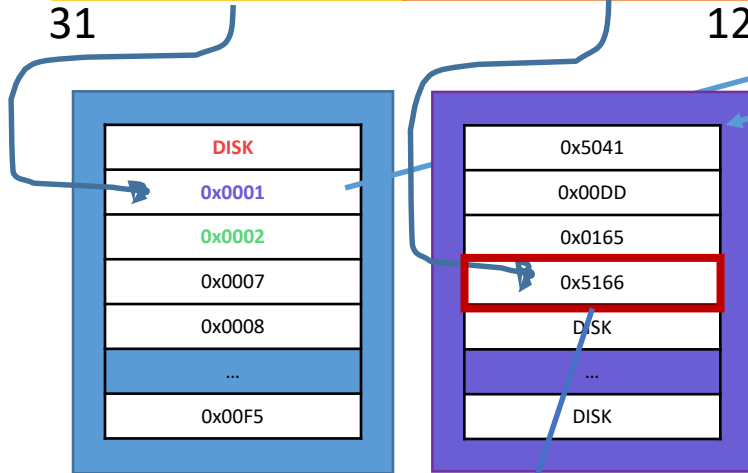
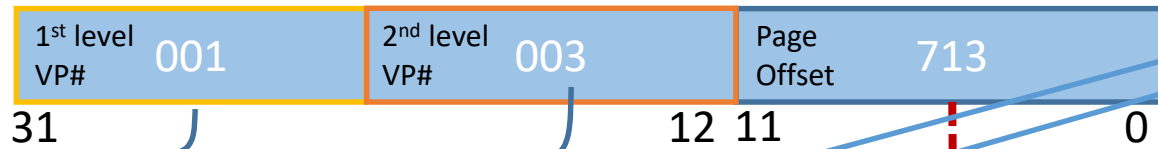
Physical Address [28 bits]



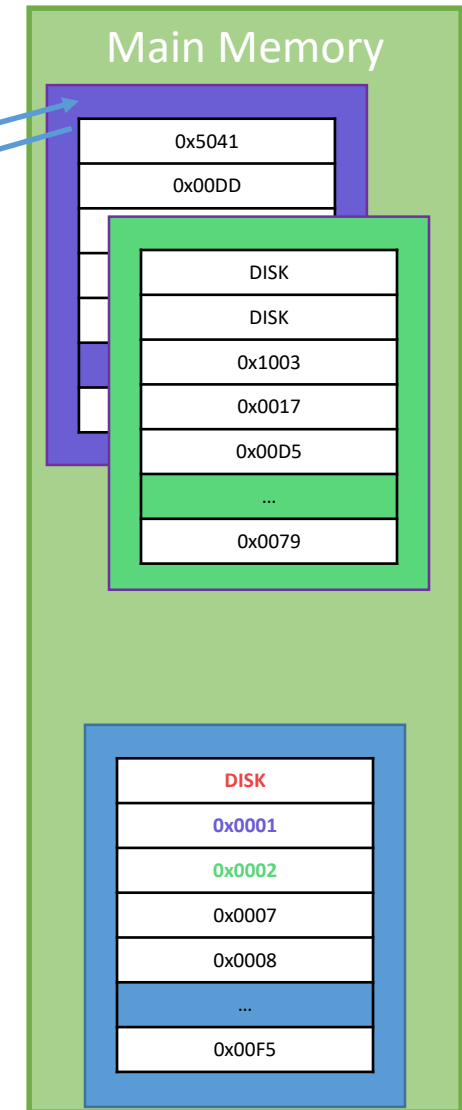
Multi-Level Page Table Translation

VA: 0x00403 713

Virtual Address [32 bit]



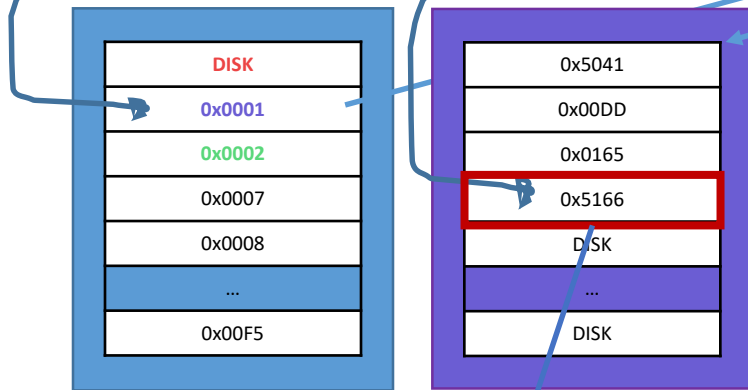
Physical Address [28 bits]



Multi-Level Page Table Translation

VA: 0x00403 713

Virtual Address [32 bit]



Physical Address [28 bits]



PA: 0x5166 713

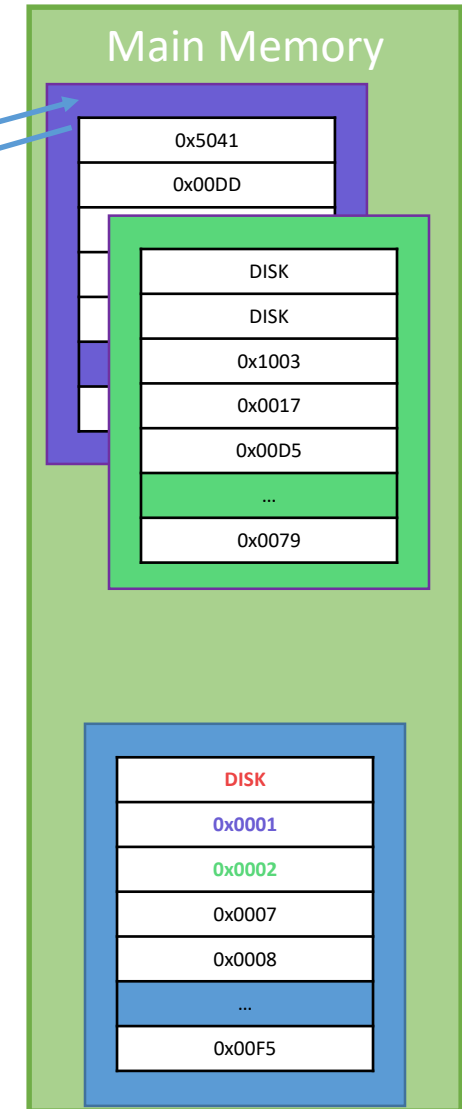
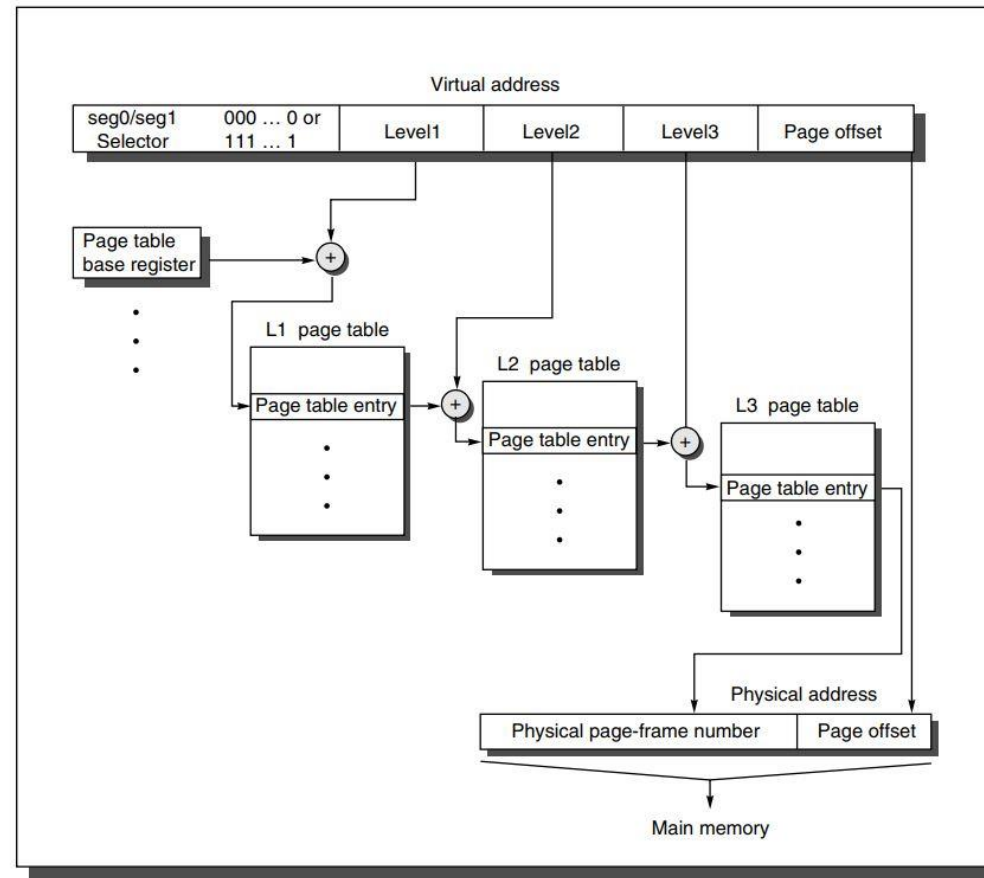


Illustration from the textbook (advanced)



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: With 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: With 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: Without 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: Without 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