

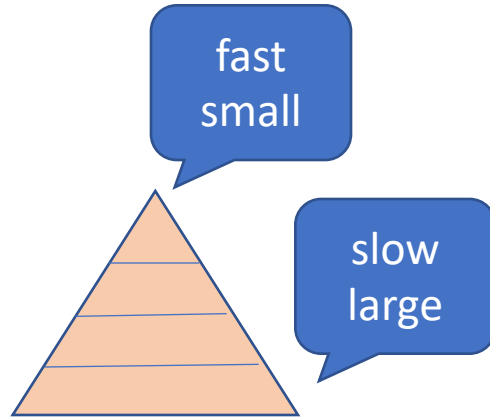
Virtual memory

Jinyang Li

Some slides are from Patterson and Hennessy

What we've learnt last time

- CPU design
 - Single cycle
 - 5 stage pipeline
- Memory hierarchy
- Caching
 - Why it works: principles of locality
 - Direct map vs. fully associative vs. n-way associative



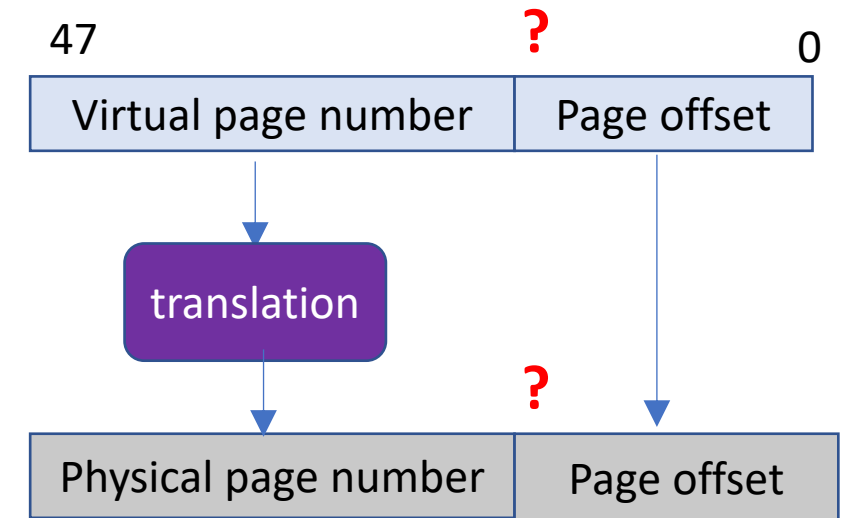
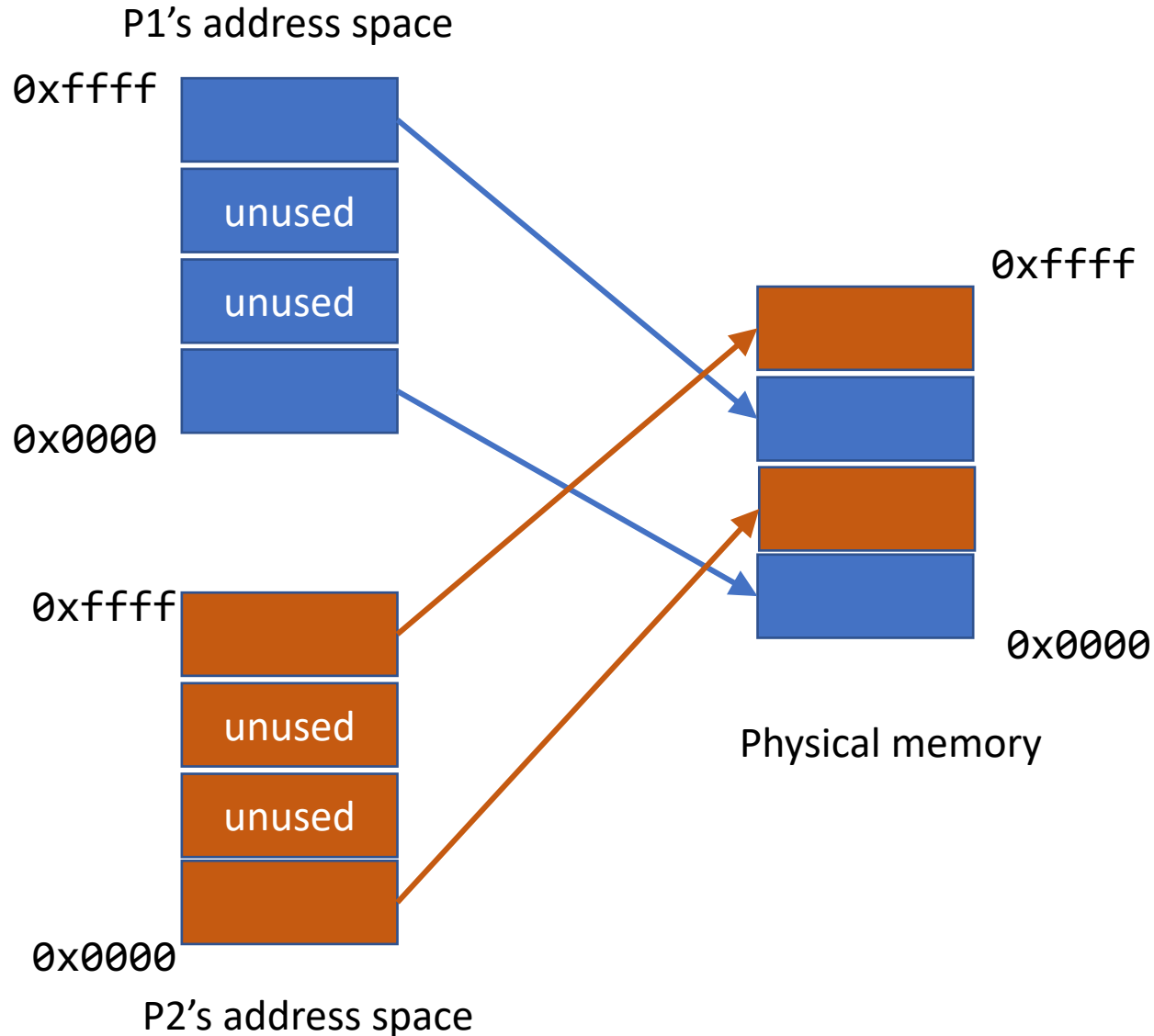
Today's lesson plan

- Virtual memory
 - Why?
 - How? Address translation
 - Make it fast

Virtual Memory: goals

- Goal #1: Use main memory as a “cache” for disk storage
- Goal #2: Let multiple processes share main memory safely
 - Each process gets a private virtual address space, isolated from others
- Solution: CPU and OS translate virtual addresses to physical addresses
 - VM “block” is called a page
 - VM translation “miss” is called a page fault

Sharing memory safely using VM



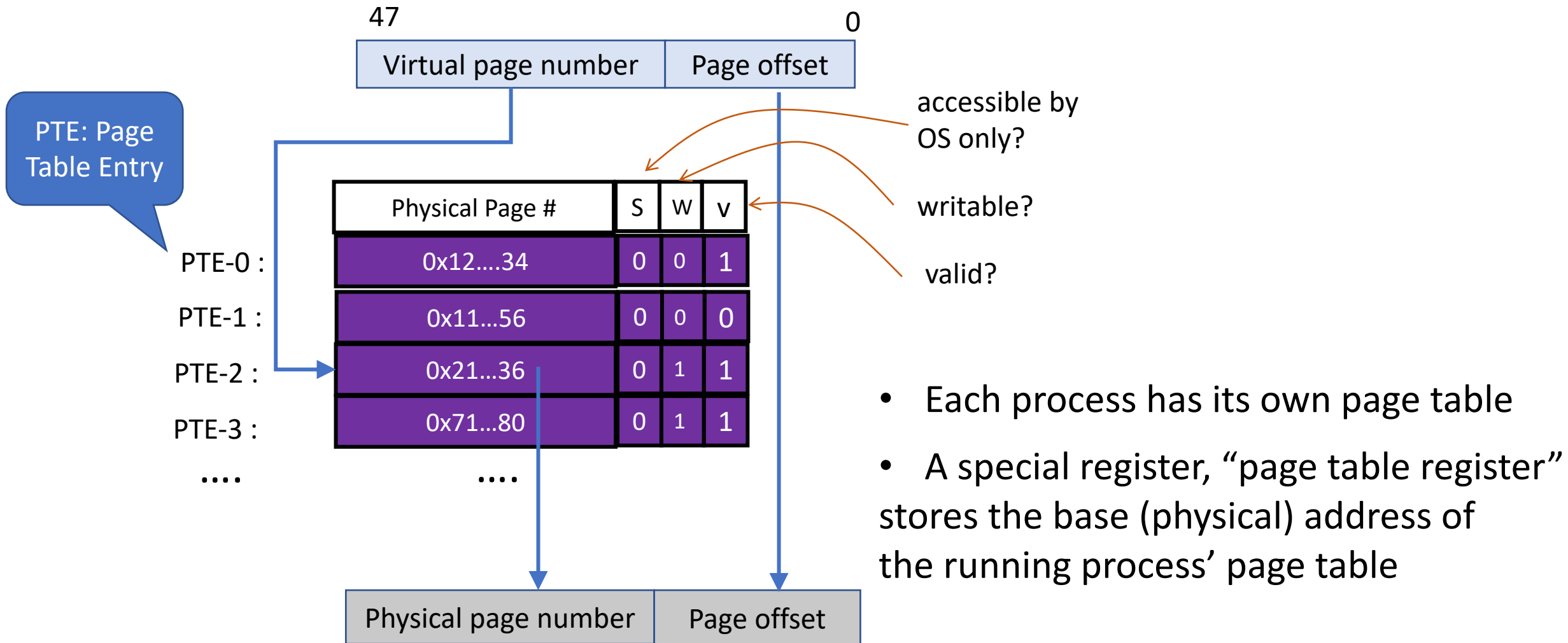
Suppose page size is 4KB,
how many bits in page offset?

$2^{12} = 4\text{KB}$, so 12 bits

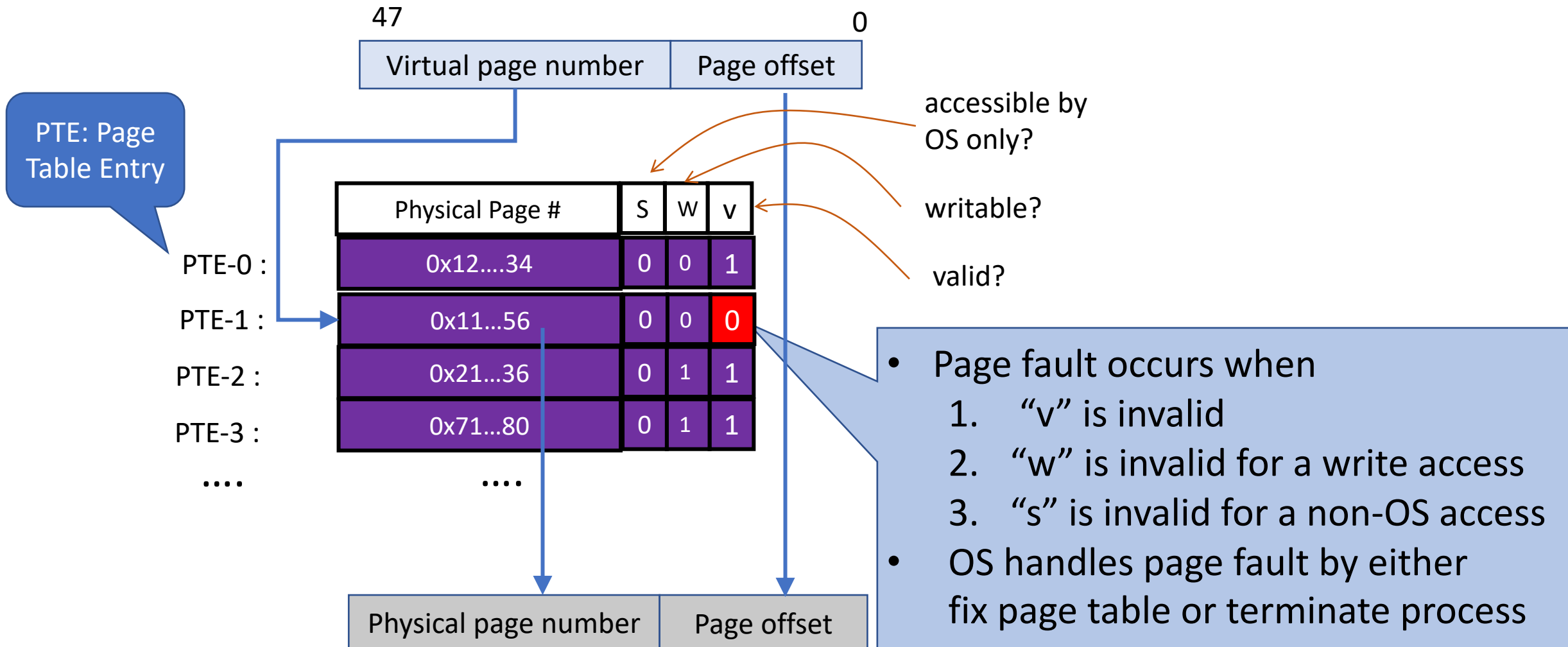
Mapping from Virtual to Physical pages

- “Fully associative”
 - A virtual page can map to any physical page
- Use an index called “page table” to store the mapping
 - No need to check all physical pages for a match
- OS populates the page table for each process
 - Page table is stored in the main memory
- CPU (MMU) performs translation using the page table

Translation using the page table



Page Fault



One level page table is impractical

- How large is the page table for 48-bit virtual address space?
 - Page size 4KB

$$2^{48}/2^{12} = 2^{36} \text{ pages}$$

$$\text{Page Table size} = 2^{36} \text{ pages} * 8 \text{ byte-PTE/page} = 2^{39} \text{ bytes} = 0.5\text{TB!!!}$$

Multi-level page tables

Problem

- how to reduce # of page table entries required?

Solution

- Multi-level page table
 - A tree of “page tables”

2-level Paging Example

- 6-bit virtual and physical address, 4-byte page

2-level Paging Example

- 6-bit virtual and physical address, 4-byte page
- 1-level page table

Suppose a process has only two 2 mapped virtual pages

	PPN	v
2 ⁴ PTEs	1100	1
	0000	0

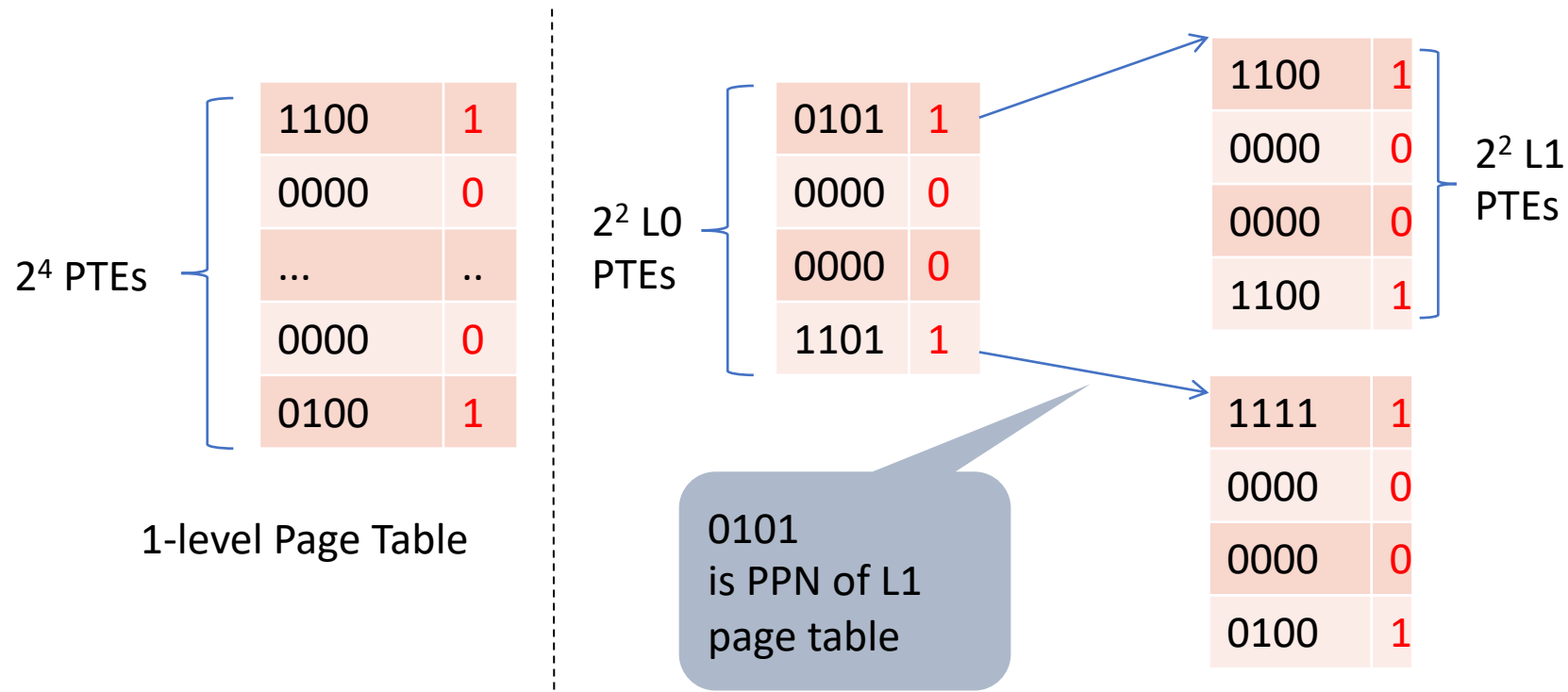
	0000	0
	0100	1

1-level Page Table

2-level Paging Example

- 6-bit virtual and physical address, 4-byte page
- 2-level page table

Suppose a process has only two 2 mapped virtual pages



2-level Paging Example

- how to perform address translation?
 - 1-level page table

1 1 1 1	10
---------	----

Index to
page table

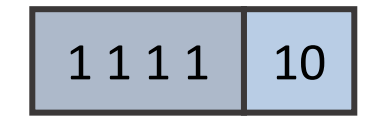
Page
offset

1100	1
0000	0
...	..
0000	0
0100	1

1-level Page Table

2-level Paging Example

- how to perform address translation?
 - 2-level page table



Index to
page table

Page
offset

1100	1
0000	0
...	..
0000	0
0100	1

1-level Page Table



Index to
L0 table

Index to
L1 table

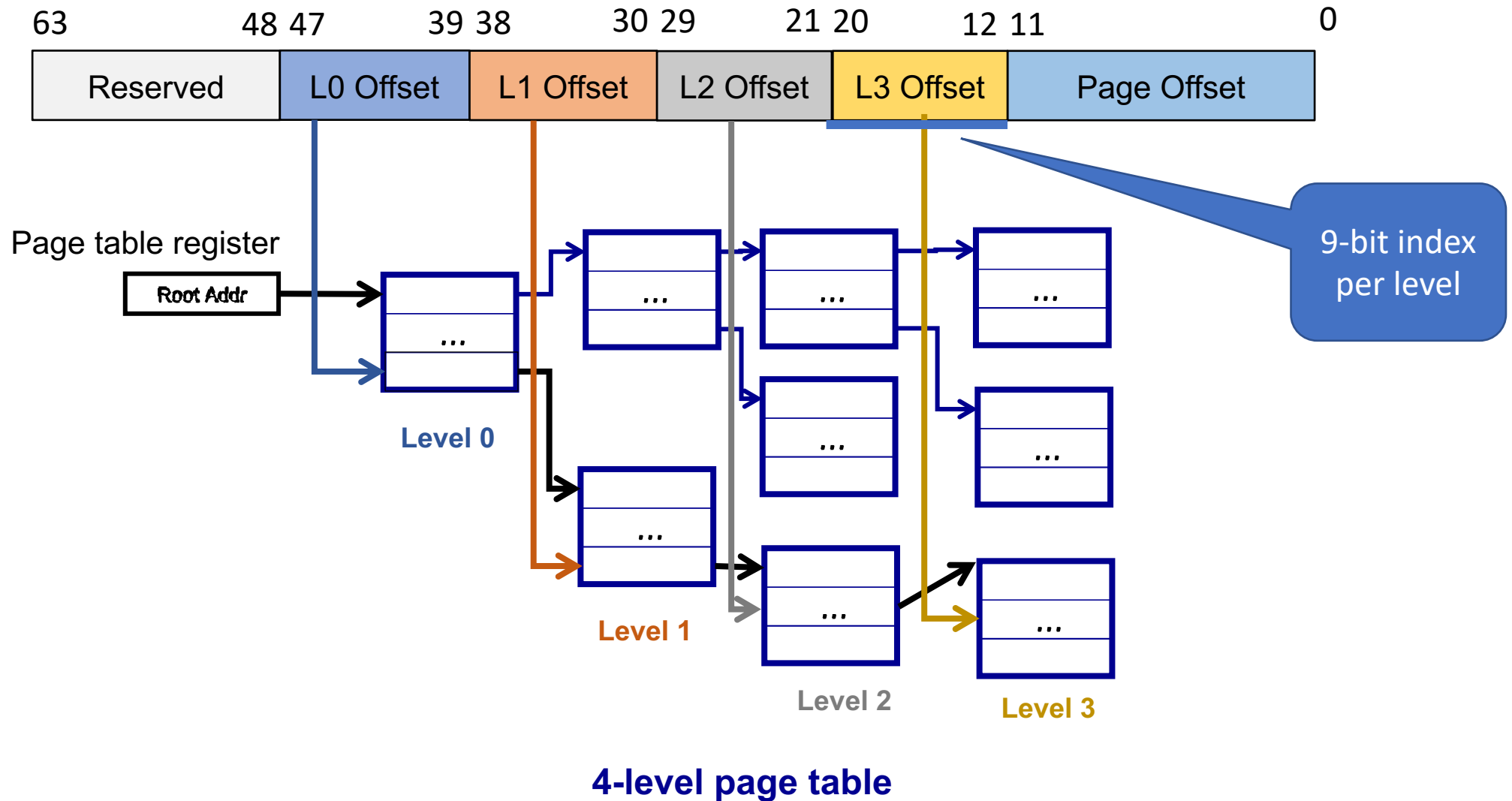
Page
offset

0101	1
0000	0
0000	0
1101	1

1100	1
0000	0
0000	0
1100	1

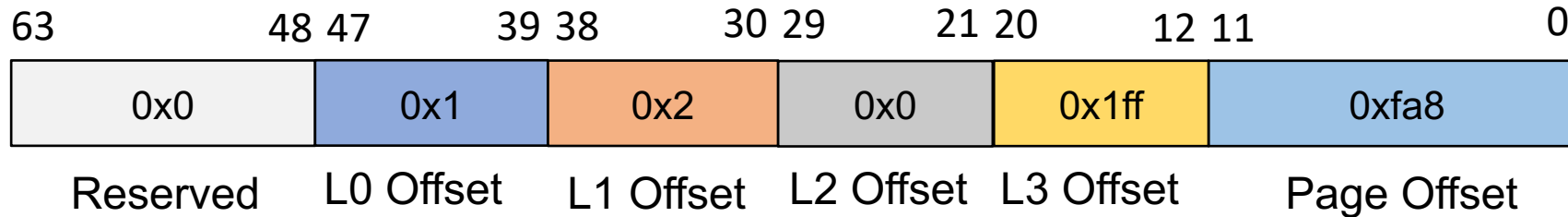
1111	1
0000	0
0000	0
0100	1

X86_64 and RISC-V use 4-level page table

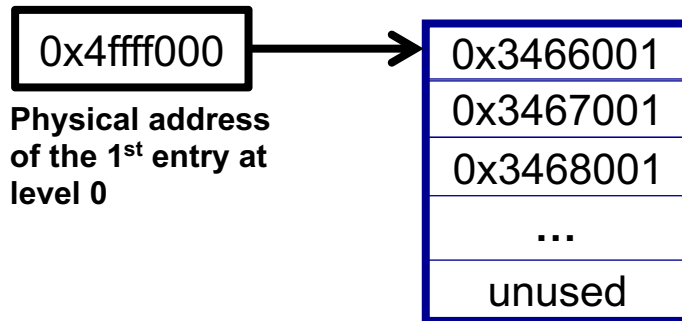


Example translation using a 4-level table

Virtual Address: *0x80801fffa8*



Page table register



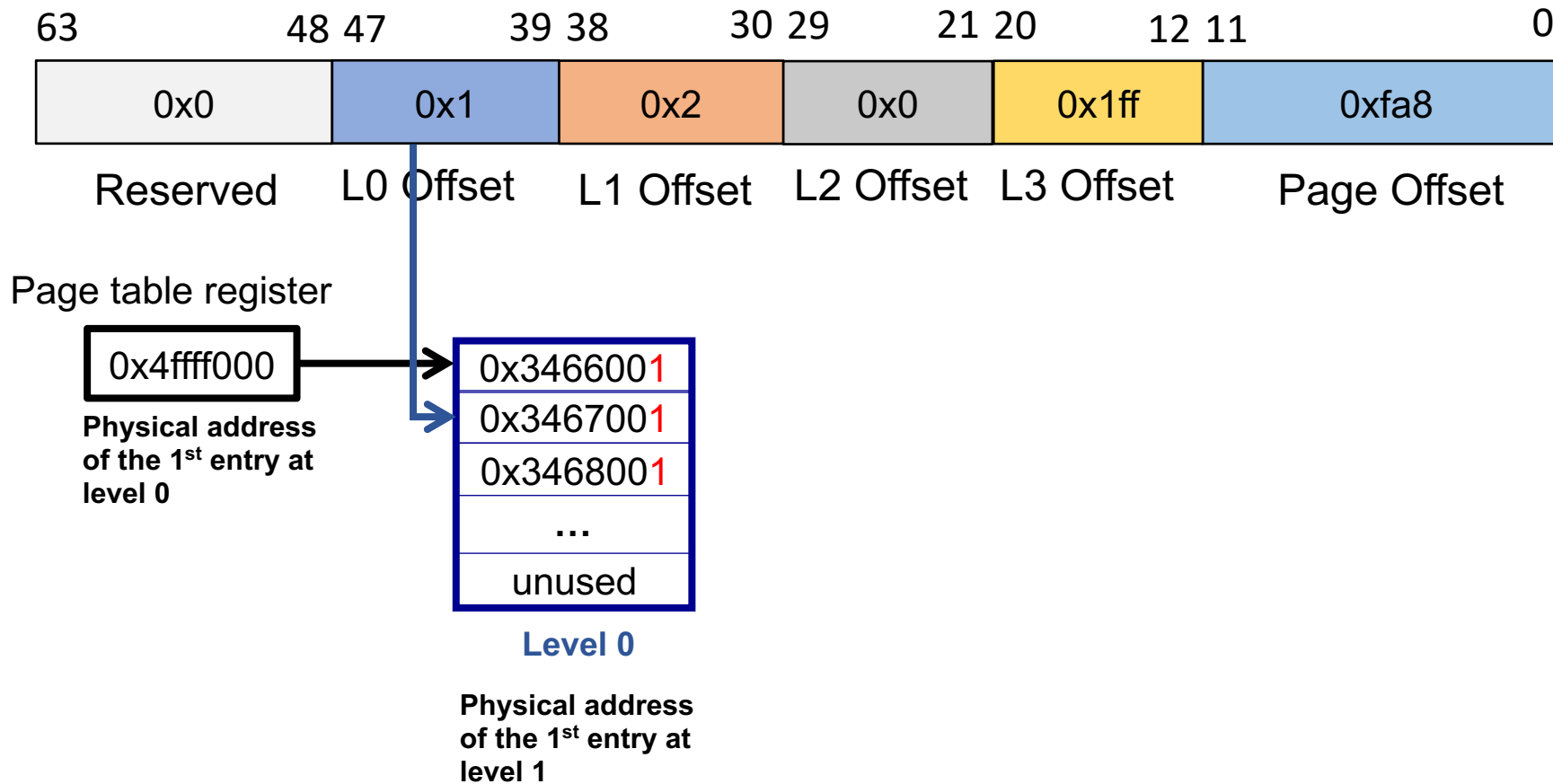
Level 0

Physical address of the 1st entry at level 1

4-level page table

Example translation using a 4-level table

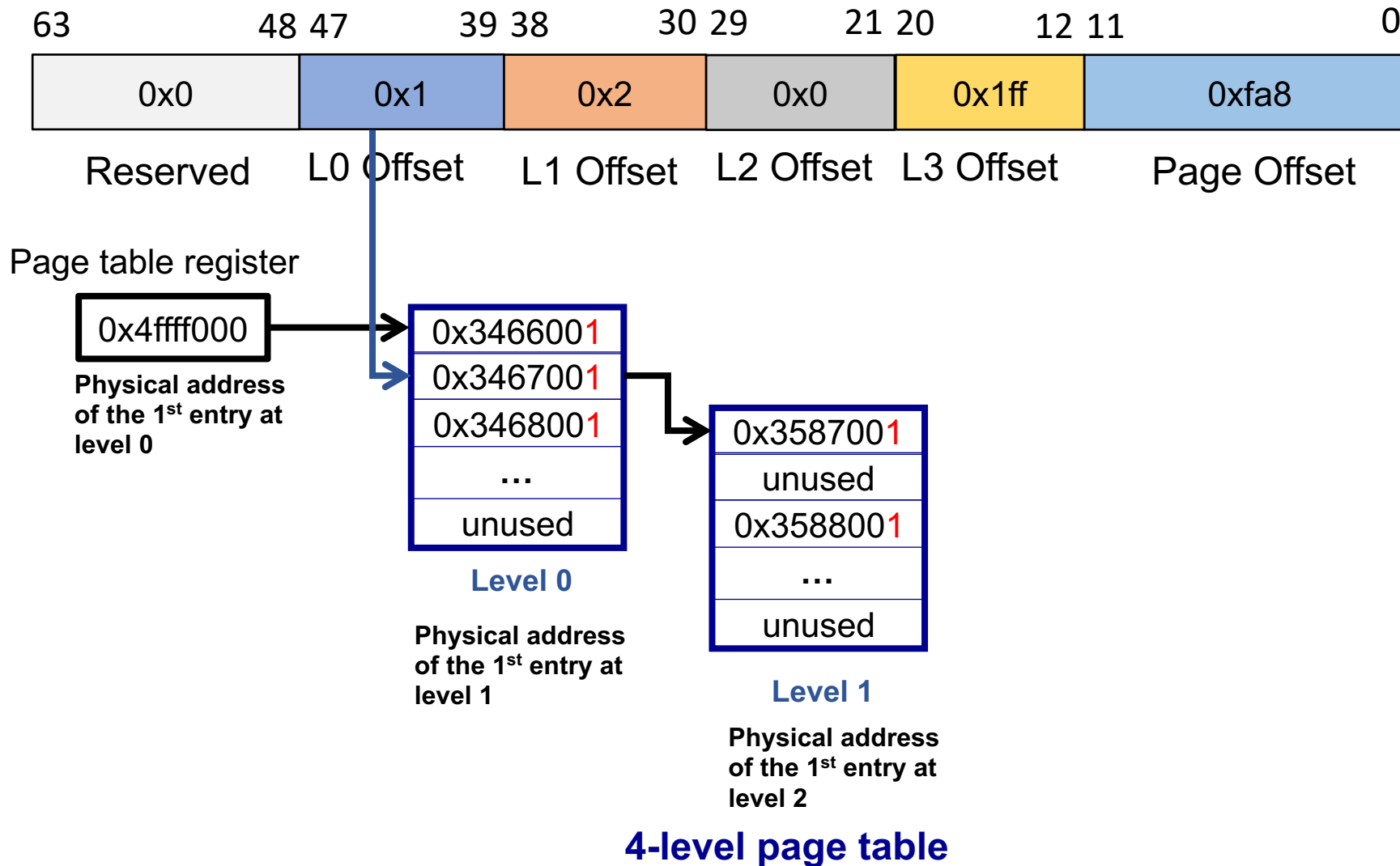
Virtual Address: *0x80801fffa8*



4-level page table

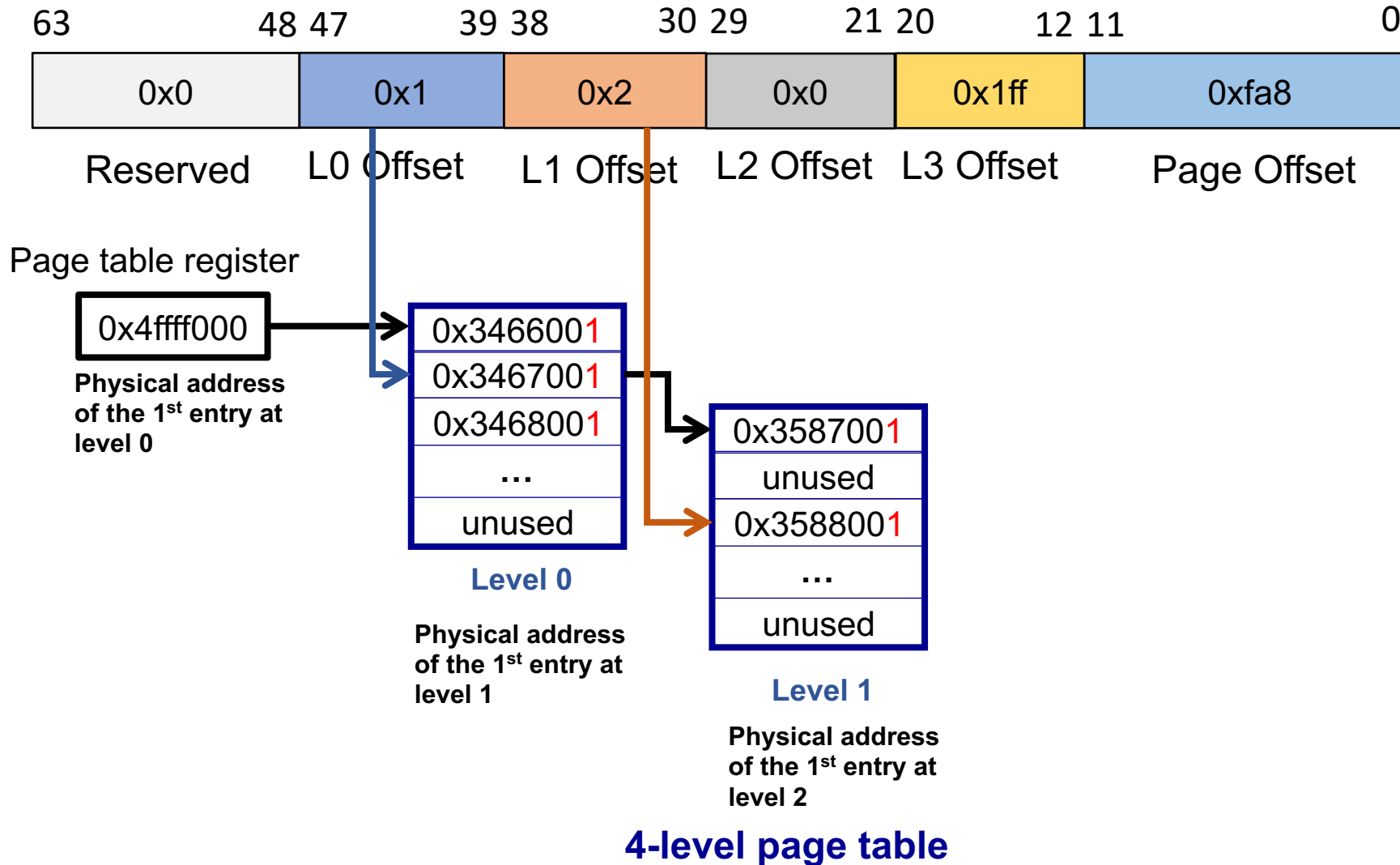
Example translation using a 4-level table

Virtual Address: *0x80801fffa8*



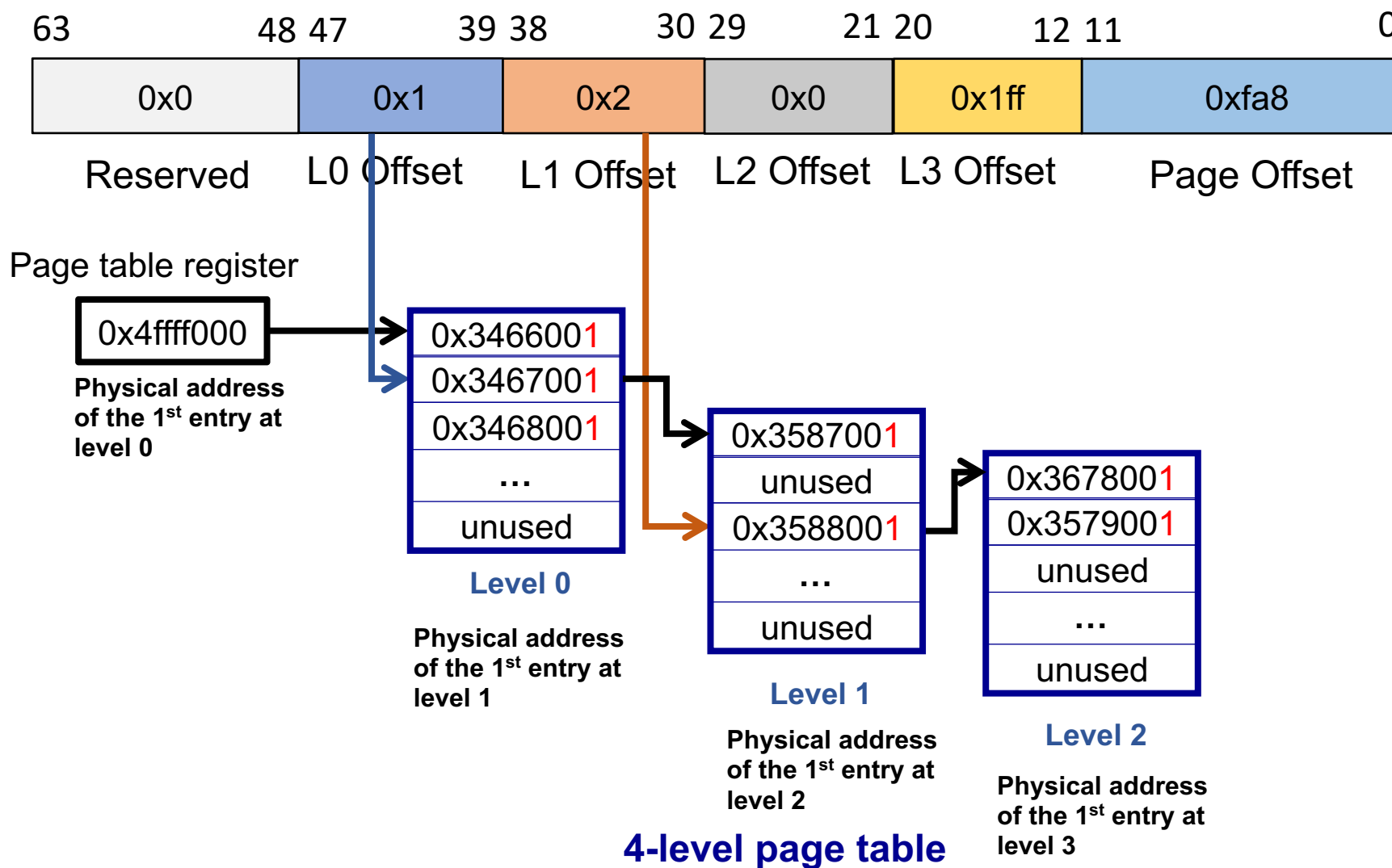
Example translation using a 4-level table

Virtual Address: *0x80801fffa8*



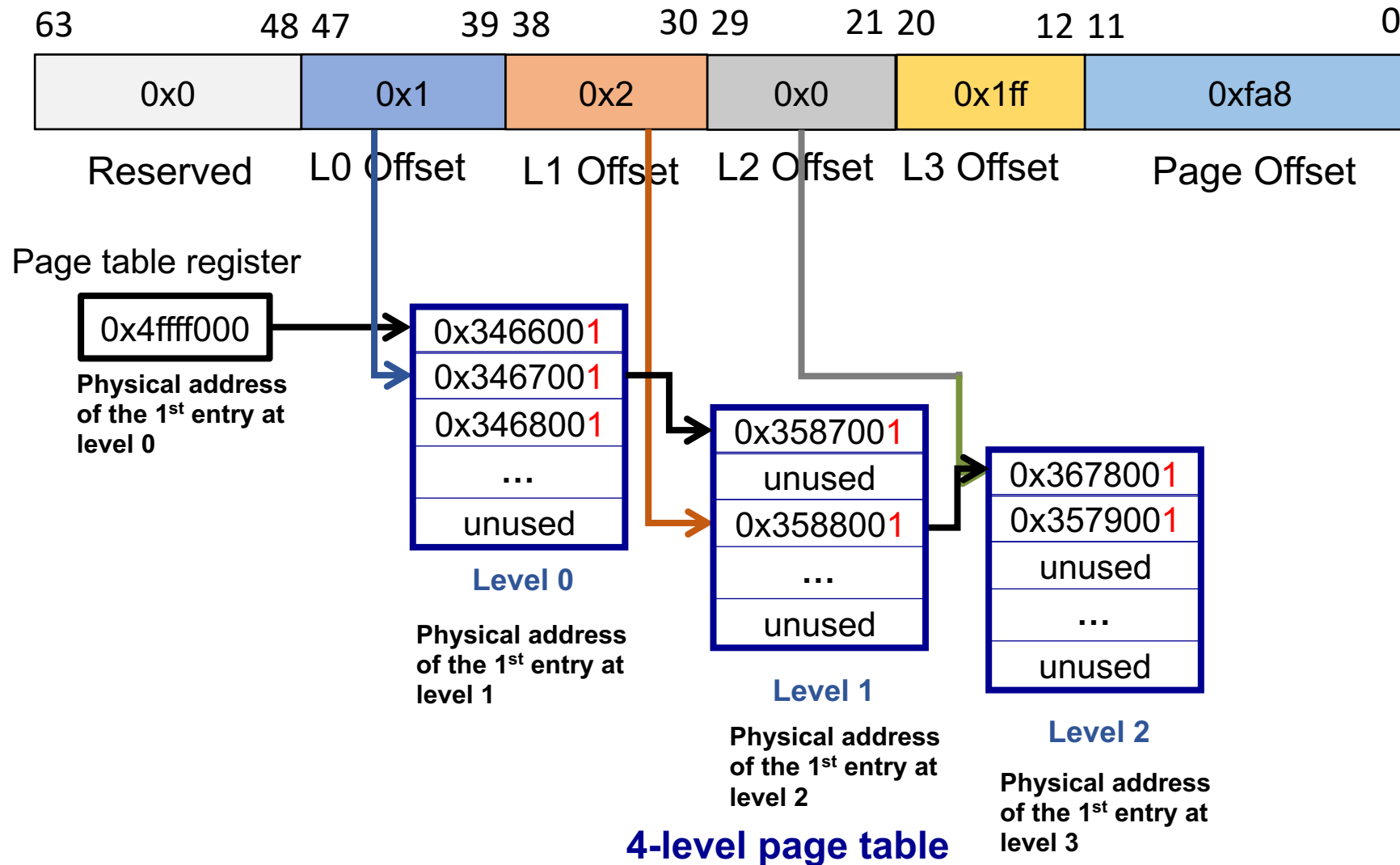
Example translation using a 4-level table

Virtual Address: *0x80801fffa8*



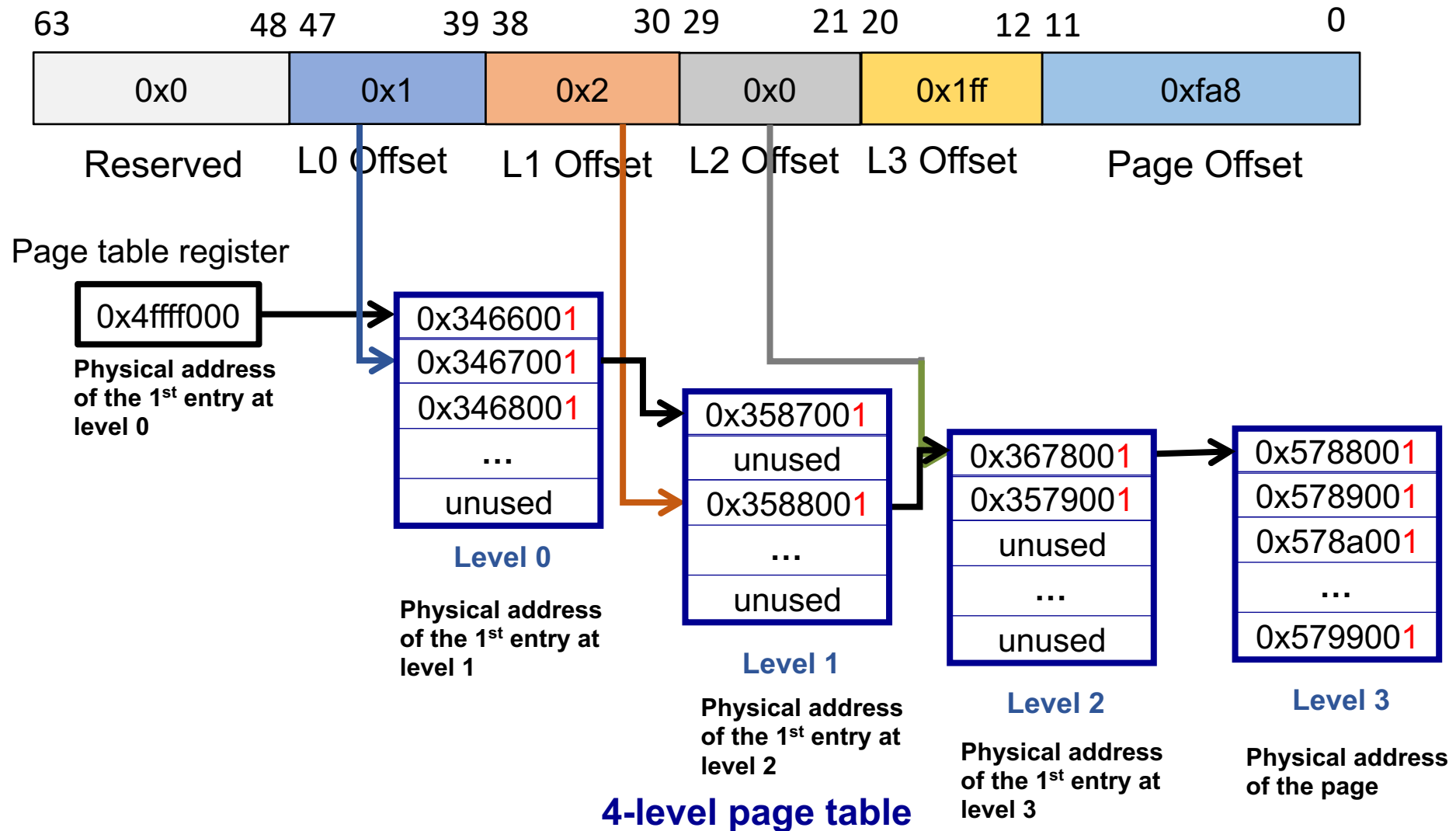
Example translation using a 4-level table

Virtual Address: *0x80801fffa8*



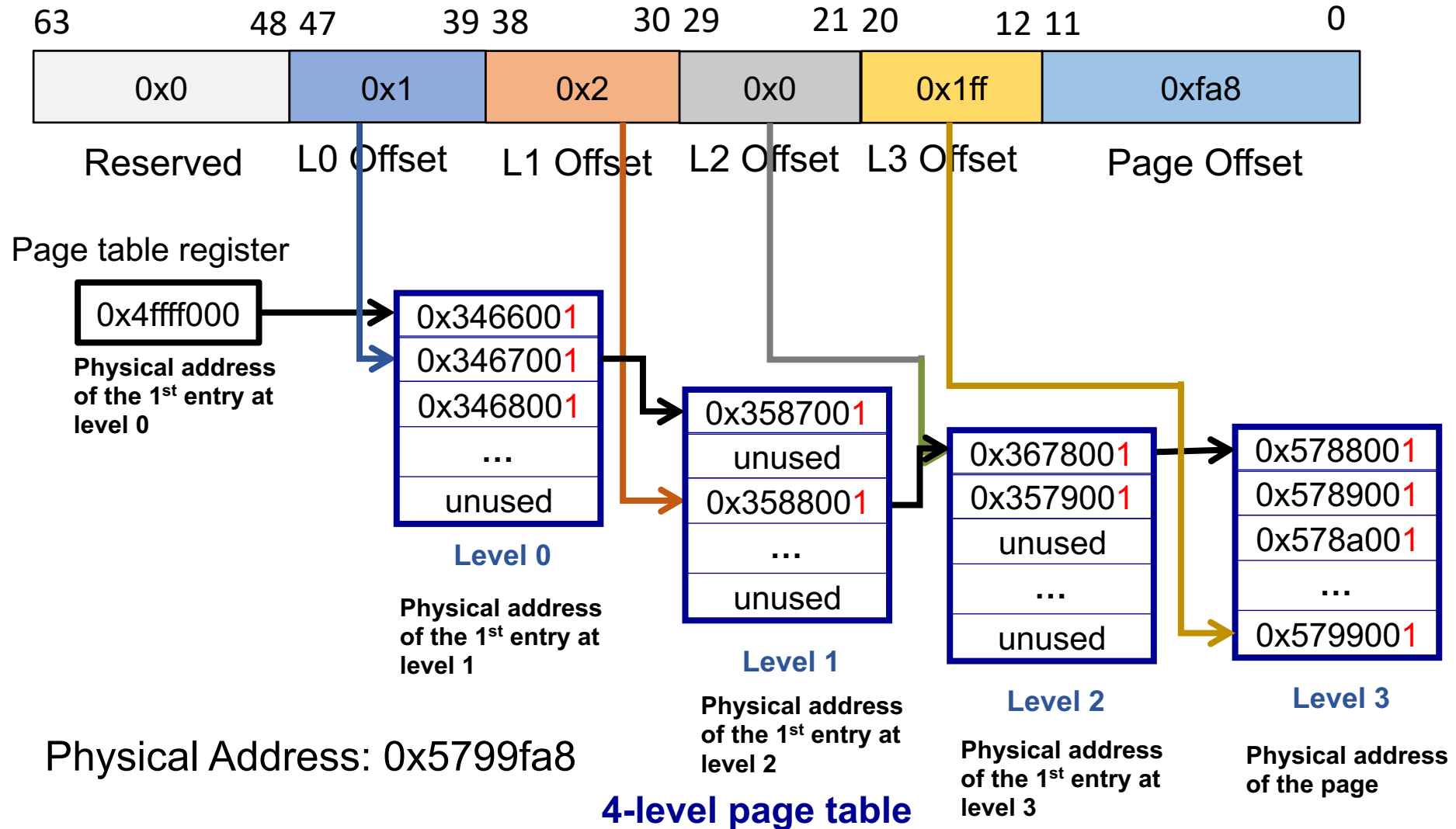
Example translation using a 4-level table

Virtual Address: $0x80801fffa8$



Example translation using a 4-level table

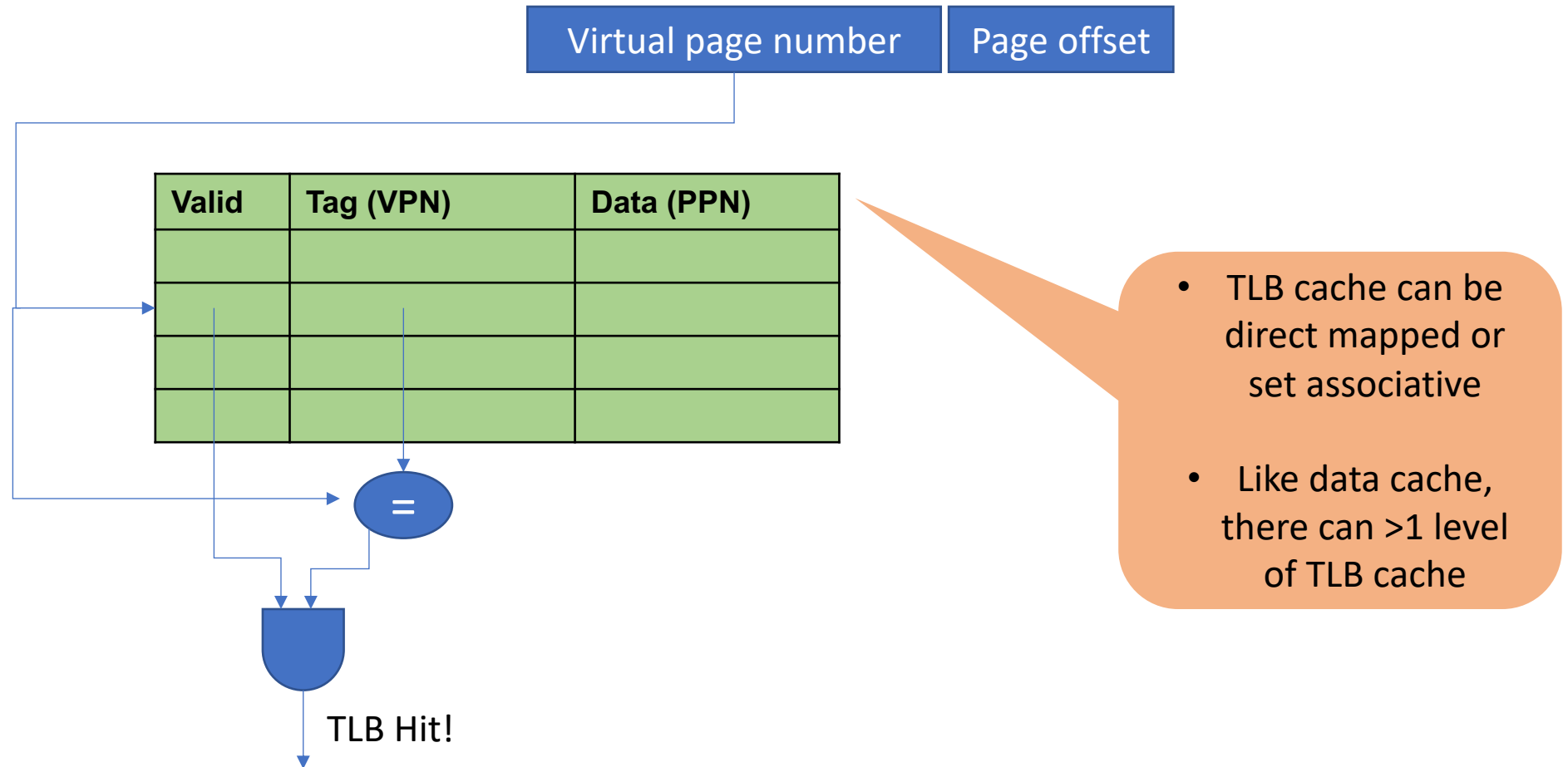
Virtual Address: *0x80801fffa8*



Fast translation using a TLB

- Address translation is costly
 - How many memory accesses per actual data access? (4-level page table)
 - 4 (page table access) + 1 (data access)
- Solution?
 - Cache virtual \rightarrow physical page mappings in a fast small cache
 - This cache is called Translation Lookaside Buffer (TLB)

Address translation with TLB



- If TLB miss, translate using 4-level page table, populate TLB with resulting VPN->PPN

Summary

- Virtual memory allows multiple processes to share memory safely
- Address translation uses multi-level page table
 - Speed up translation using TLB
- Memory hierarchy
 - L1 cache \leftrightarrow L2 cache \leftrightarrow ... \leftrightarrow DRAM memory \leftrightarrow disk
- Memory system design is critical for performance