Problem 2: Building a 2-Level Page Table

Assume the following constraints about our Multi-Level Page Table:

- 32-bit Virtual Addresses
- 4 MB of Physical Memory
- 2-Level Page Table
- PTEs store PPNs
- 16KB Pages
- 8 Control Bits (V/D/PLRU/RW)
- Page Tables fit in 1 Page

How many entries are in a 2nd-level page table assuming they are the maximum size possible (1 page)?

16 KB Pages = (2^4 * 2^10B) => 2^14 B pages → log2(2^14 B) = 14 Page Offset bits
4 MB Phys. Mem. = 2^22 B Phys. Mem. → log2(2^22) = 22 Phys. Addr. Bits

22 Phys. Addr. Bits - 14 Page Offset Bits = 8 Phys. Page Num. Bits

8 Phys. Page Num. Bits + 8 Control Bits = 16 bits = 2B per 2nd-level PTE

16 KB per page / 2B per 2nd-level PTE = 2^14 B / 2^1 B = 2^13 entries = 8192 entries

How many virtual address bits are used to index a 2nd-level page table? 1st?

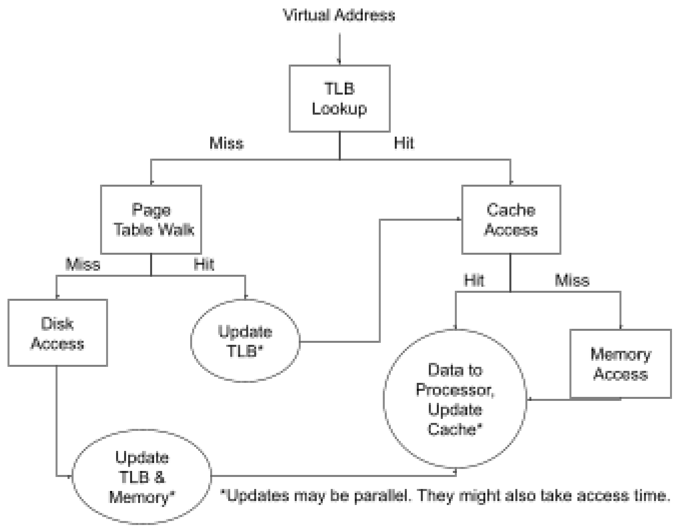
2^13 2nd-level Page Table Entries → log2(2^13) = 13 2nd-level index bits

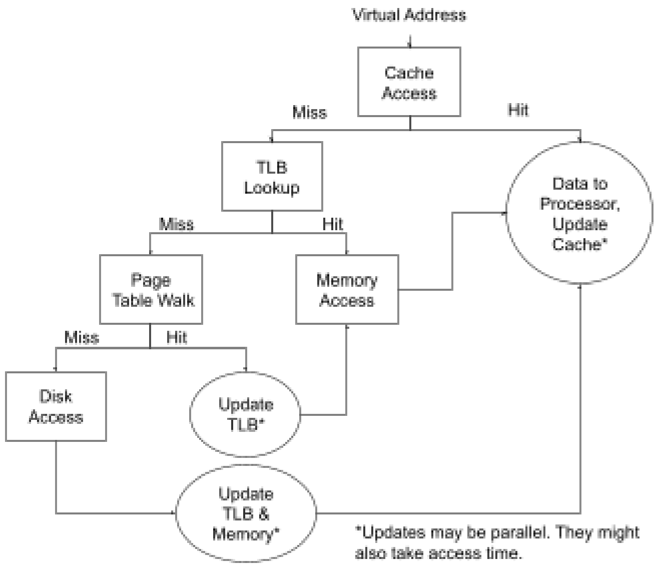
32 virt. addr. bits - 13 2nd-level index bits - 14 page offset bits
= 5 1st-level index bits

How much memory is used to store the page table in the worst case?

Size of 1st level page table = 2B * 2^5 entries = 64 B
In the worst case, we use all second-level page tables
= (Size of 1st-level PT) + (Num. 2nd-level PTs) * (Size of 2nd-level PT)
= 64 B + 2^5 * 16KB
= 64B + 512KB

Physically / Virtually Addressed Cache Reference





Physically Addressed Cache Virtually Addressed Cache

Consider a system with a physically addressed, unified data and instruction cache with a single-level page table. We have the following statistics about memory accesses:

Memory Component	Hit Rate	Access Time
TLB	95%	1 cycle
Cache	90%	10 cycles
Main Memory	99% After Cache, 99.9% Before	100 cycles
Disk	--	100,000 cycles

We also know that:

- TLB accesses and cache accesses are sequential
- TLB hits will never cause page faults
- The page table cannot be cached and is always available in main memory
- Upon retrieval, data is immediately sent to the processor while other updates occur in parallel

Problem 3: Access a Virtually Addressed Cache

a. Following the access tree, what is the mathematical expression for the AMAT [average memory access time] of a virtually addressed cache?

Cache Latency	+	// Cache access
Cache Miss Rate * (TLB Latency	+	// TLB lookup
TLB Hit Rate * Mem Latency	+	// TLB Hit
TLB Miss Rate * (PT Latency	+	// TLB Miss (page walk)
Page Fault Rate * Disk Latency	+	// Disk Access
Page Hit Rate * Mem Latency))		// Memory Hit

b. How long is the average access for our virtually addressed cache?

10 cycles	+	Cache access
0.1 * (1 cycle	+	TLB lookup
0.95 * 100 cycles	+	TLB Hit
0.05 * (100 cycles	+	TLB Miss (page walk)
0.01 * 100000 cycles	+	Disk Access
0.99 * 100 cycles))		Memory Hit

= 25.595 cycles

Problem 4: Access a Physically Addressed Cache

a. Following the access tree, what is the mathematical expression for the AMAT [average memory access time] of a physically addressed cache?

TLB Latency	+	// TLB access
TLB Miss Rate (PT Latency	+	// TLB Miss, Page Walk
Page Fault Rate * Disk Latency)	+	// Page Fault
(TLB Hit Rate	+	
TLB Miss Rate * PT Hit Rate)	*	// %Accesses going to Cache
(Cache Latency	+	// Cache Access
Cache Miss Rate * Mem Latency)		// Cache Miss

b. How long is the average access for our physically addressed cache?

1 cycle	+	TLB access
0.05 (100 cycles	+	TLB Miss, Page Walk
0.001 * 100000 cycles)	+	Page Fault
(0.95	+	
0.05 * 0.999)	*	%Accesses going to Cache
(10 cycles	+	Cache Access
0.1 * 100 cycles)		Cache Miss

= 30.999 cycles