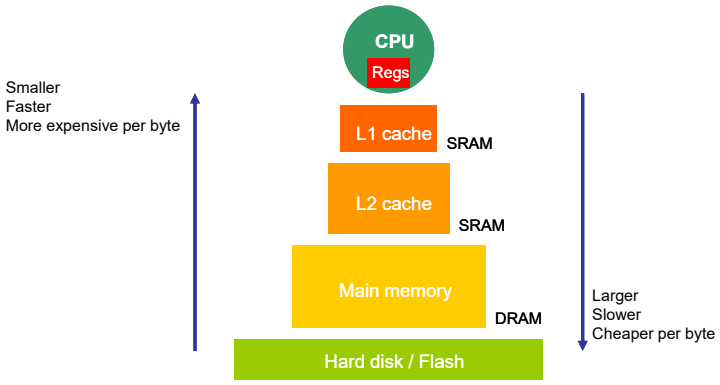
Memory Hierarchy Introduction - §2.1 (summary of Appendix B)



1

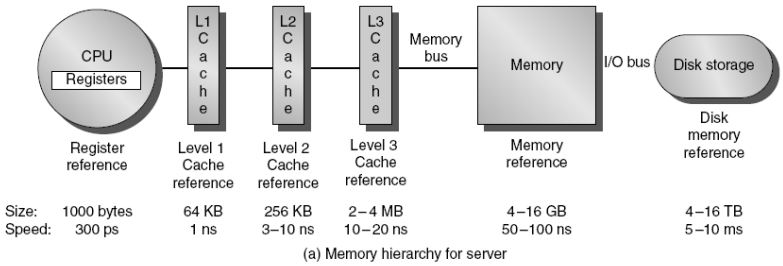
The Principle of Locality

- Two Different Types of Locality:
  - **Temporal Locality** (Locality in Time): If an item is referenced, it will tend to be referenced again soon.
  - **Spatial Locality** (Locality in Space): If an item is referenced, items whose addresses are close by tend to be referenced soon.

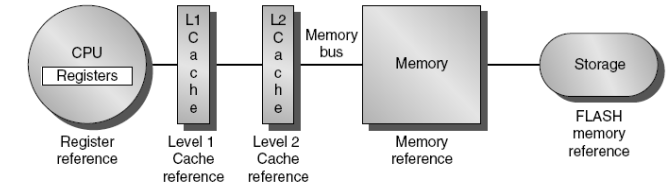
- **Memory** Hierarchy Terminology:
  - block: minimum unit of data transferred between levels
  - hit: data requested is in the upper level
  - miss: data requested is not in the upper level
  - hit rate: percentage of hits (sometimes called hit ratio)
  - miss rate: percentage of miss (sometimes called miss ratio)
  - hit time: the time to satisfy request in case of a hit
  - miss penalty: the time to satisfy a request in case of a miss

2

Memory Hierarchy



(a) Memory hierarchy for server



(b) Memory hierarchy for a personal mobile device

3

Cache performance

Example: Consider a 5-stage (in order) pipeline and assume that

- With perfect cache, CPI = C
- 40% of instructions are load/store (access cache for data)
- Miss penalty = 30 clock cycles
- Miss rate = 2% (for each of the I and D caches)

Number of memory accesses per instruction = 1.4

Actual CPI = C + av. memory stalls per instruction due to cache miss

= C + av. cache misses per instruction * miss penalty

= C + (1.4 * 0.02) * 30 = C + 0.84

NOTE: miss penalty of L1 cache is calculated based on the hit time, miss rate and miss penalty of the L2 cache.

4

4 Questions for Memory Hierarchy:

- Q1: Where can a block be placed in the upper level? *(Block placement)*
- Q2: How is a block found if it is in the upper level? *(Block identification)*
- Q3: Which block should be replaced on a miss? *(Block replacement)*
- Q4: What happens on a write? *(Write strategy)*

Note: a block (cache line) is the unit of traffic between memory and cache. A block is usually from 4 to 8 words.

5

Simplest cache: Direct Mapped

For each item (block) of data in memory, there is exactly one location in the cache where it might be.

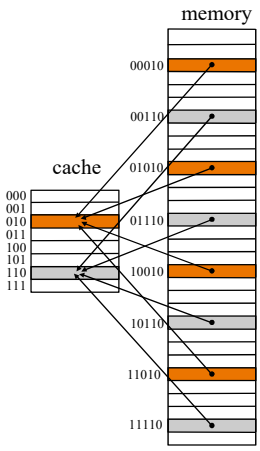
Index = (memory address) **mod** (cache size)

Cache location (block address)

Note: lots of items in memory share locations in cache

- Cache can be accessed using the low-order address bits
- Need to *tag* data using the high order address bits
- A valid bit is used for each block

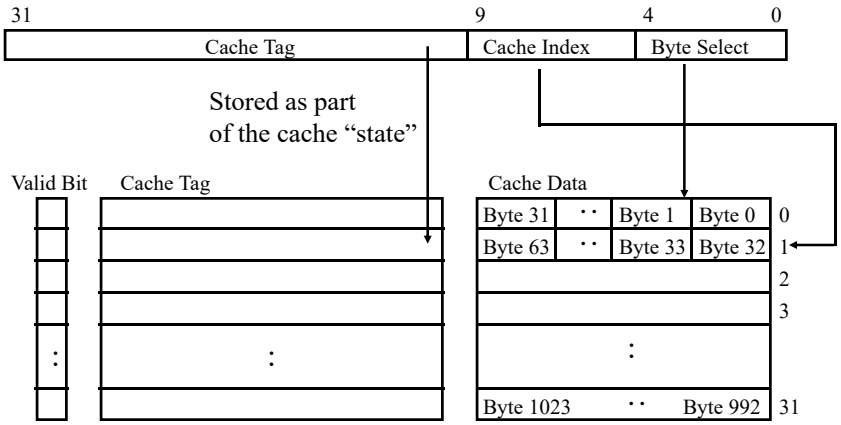
Memory address (byte) = < word address , word offset >
Word address = < block address , block offset >
Block address = < tag , index >



6

1 KB Direct Mapped Cache, 32B blocks

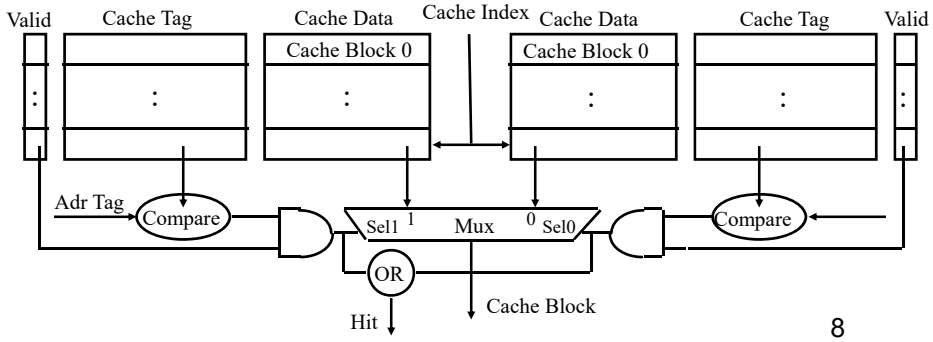
- For a $2^N = 2^{10}$ byte cache and 32-bit memory address:
  - The upper most (32 - N) = 22 bits form the Cache Tag
  - The lowest $M = 5$ bits are the Byte Select (Block Size = 2^M)



7

Set Associative Cache

- N-way set associative: *N* entries for each Cache Index
  - *N* direct mapped caches operates in parallel (*N* typically 2 to 8)
- Example: Two-way set associative cache
  - Cache Index selects a "set" from the cache
  - The two tags in the set are compared in parallel
  - Data is selected based on the tag result



8

Disadvantage of Set Associative Cache:

- N-way Set Associative Cache vs. Direct Mapped Cache:
  - N comparators vs. 1
  - Extra MUX delay for the data
  - Data comes AFTER Hit/Miss
- In a direct mapped cache, Cache Block is available BEFORE Hit/Miss:
  - Possible to assume a hit and continue. Recover later if miss.
- A fully associative cache is one with N=size of the cache (slow ???)

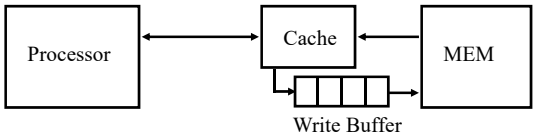
Block replacement:

- Not an issue for Direct Mapped
- Set Associative or Fully Associative:
  - Random
  - LRU (Least Recently Used)
  - FIFO

9

Write policies

- Write through or write back,
- Write back can take advantage of a dirty bit
- A Write Buffer (FIFO) is needed between the Cache and Memory
  - Typical number of entries: 4
  - Write buffer saturation??

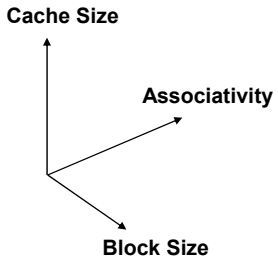


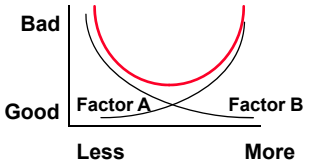
Two options on a write miss :

- Write allocate: read block first to cache (as in a read miss), then write the word
- No-write allocate: write only to memory and not to cache

10

- Several interacting dimensions
  - cache size
  - block size
  - associativity
  - replacement policy
  - write-through vs write-back
  - write allocate
- The optimal choice is a compromise
  - depends on access characteristics
    - » workload
    - » use (I-cache, D-cache, TLB)
  - depends on technology / cost
- Simplicity often wins





11

Average memory access time

Example: Assume that

- Processor speed = 1 GHz (1 n.sec. clock cycle)
- Cache access time = 1 clock cycle
- Miss penalty = 100 n.sec (100 clock cycles)
- I-cache miss rate = 1%, and D-cache miss rate = 3%
- 74% of memory references are for instructions and 26% for data

Effective cache miss rate = 0.01 * 0.74 + 0.03 * 0.26 = 0.0152
Av. (effective) memory access time = 1 + 0.0152 * 100 = 2.52 cycles
= 2.52 n.sec

- Now, assume that you increase the processor speed to 1.5 GHz

Miss penalty = 150 cycles (0.66 n.sec cycles)
Av. (effective) memory access time = 1 + 0.0152 * 150 = 3.28 cycles
= 2.18 n.sec

12

Cache Performance (§B.2)

CPUtime = IC x ($CPI_{execution}$ + Misses per instruction x Miss penalty) x Clock cycle time

IC = Instruction Count

Misses per instruction = Memory accesses per instruction x Miss rate

$$CPUtime = IC \times \left(CPI_{Execution} + \frac{Memory\ \ accesses}{Instruction} \times \textbf{Miss rate} \times Miss\ \ penalty \right) \times Clock\ \ cycle\ \ time$$

Improving Cache Performance:

1. Reduce the miss rate,
2. Reduce the miss penalty,
3. Reduce the time to hit in the cache.

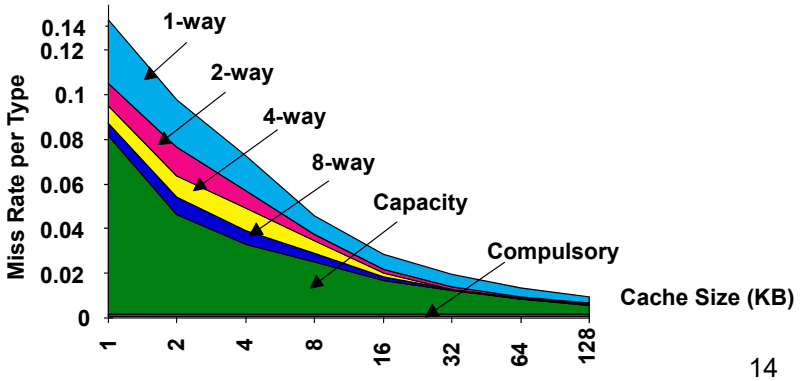
13

Classification of cache misses

– Compulsory Misses: Sad facts of life. Example: cold start misses.

– Capacity Misses: Increase cache size

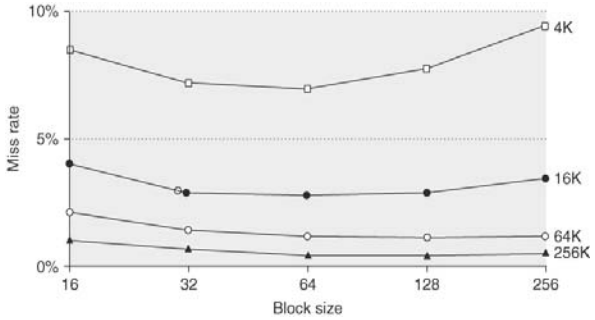
– Conflict Misses: Increase cache size and/or associativity.
Nightmare Scenario: ping pong effect!



14

Basic ways to reduce miss rate (§ B.3)

1. Larger Block Size



- Block size explores spacial locality,
- Large blocks are only useful if penalty to fetch a block of K words is less than K times the penalty to fetch one word
- Why can a large block size help and why can it hurt?
- Which of the three C's are affected by the block size?

15

2. Larger caches

- Which of the three C's are affected by the cache size?
- Why can't we increase the size of the cache arbitrarily?

3. Higher Associativity

- Which of the three C's are affected by associativity?
- 8-way set associativity is as good as full associativity
- 2:1 Cache Rule:
  - Miss Rate DM cache size N = Miss Rate 2-way cache size N/2
- Beware: Execution time is the only final measure!
  - Will hit latency (possibly clock cycle time) increase?
- Effect on power consumption?
- Example: If higher associativity increases the cycle time by 20% but decreases the miss rate from 5% to 2.5%, would you go with higher associativity? - Will your decision depend on other factors?

16

Basic ways to reduce the Miss Penalty

4. Multi-level cache

$AMAT = Hit\ Time_{L1} + Miss\ Rate_{L1} \times (Hit\ Time_{L2} + Miss\ Rate_{L2} \times Miss\ Penalty_{L2})$

Local miss rate— misses in this cache divided by the total number of memory accesses *to this cache* (Miss $rate_{L2}$)

Global miss rate—misses in this cache divided by the total number of memory accesses *generated by the CPU* (Miss $Rate_{L1}$ x Miss $Rate_{L2}$)

- L2 is larger (fewer misses) and slower (larger hit time) than L1
- Since hits are few in L2, may target miss penalty reduction
- In L1, may target hit time reduction.
- May use different organizations and block sizes (easy??)
- Multilevel inclusion is desirable.
- **Danger**: time to DRAM will grow with multiple levels in between
- Out-of-order CPU can hide L1 data cache miss (3–5 clocks), but stall on L2 miss (40–100 clocks)?

17

Example

- Miss rate of L1 = 4%, hit time is one cycle
- If L2 is direct mapped, local miss rate = 25% and hit time = 10 cycles
- If L2 is set-associative, local miss rate = 20% and hit time = 11 cycles
- Miss penalty for L2 is 100 cycle.

AMAT with direct mapped L2 = 1 + 0.04 (10 + 0.25 * 100) = 2.4 cycles
AMAT with set-associative L2 = 1 + 0.04 (11 + 0.20 * 100) = 2.24 cycles

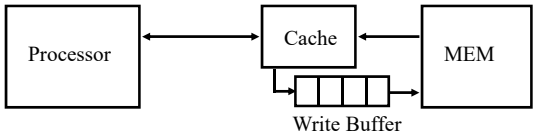
- Compare with a system with larger L1 and no L2, and improved L1 miss rate of 2%

AMAT with large L1 and no L2 = 1 + 0.02 (100) = 3 cycles

- Taking into consideration the effect of writing back replaced dirty blocks complicates slightly the analysis.

18

5. Read Priority over Write on Miss



- If write through, write buffers avoid stalling
- May causes RAW conflicts with reads on cache misses
- Waiting for write buffer to empty will increase read miss penalty
  - Solution: Check write buffer contents before read; if no conflicts, let the memory access continue

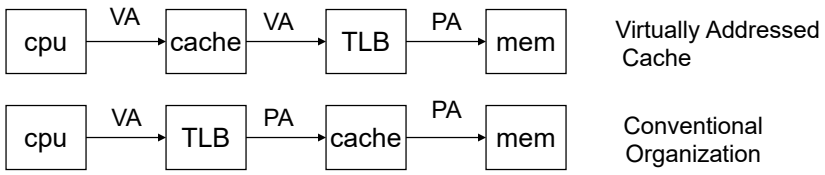
- If write Back (replace dirty block on a miss)
  - Normal: Write dirty block to memory, and then do the read
  - Instead copy the dirty block to a write buffer, then do the read, and then do the write
  - CPU stalls less since restarts as soon as the read completes

19

A basic way to reduce the Hit time

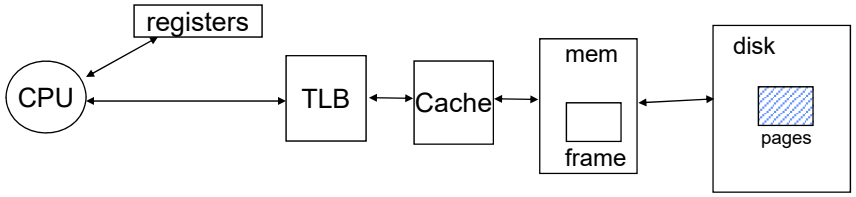
6. Avoiding Address Translation

- Send virtual address to cache? Called *Virtually Addressed Cache* or just *Virtual Cache* -- not *Physical Cache.*
  - Every context switch, must flush the cache; otherwise get false hits. Cost is time to flush + "compulsory" misses from empty cache
  - Does not support *aliases* (*synonyms*); Two different virtual addresses mapped to the same physical address.
- To avoid cache flush, we may add *process identifier tag* to cache blocks.

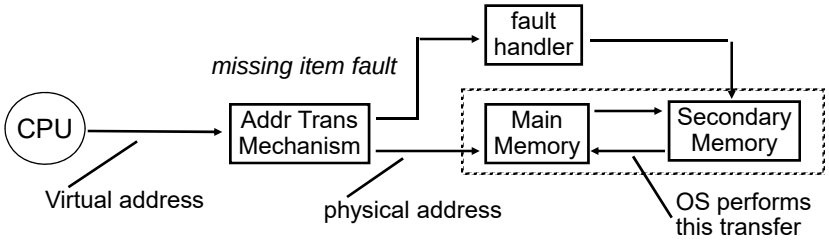


20

Basic Issues in VM System Design (§ B.4)

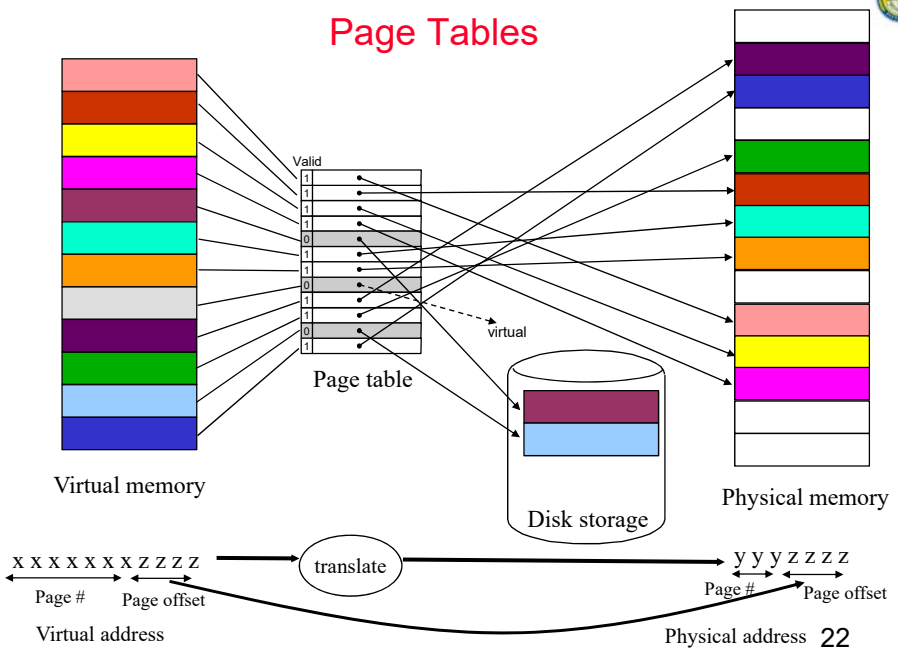


missing item fetched from secondary memory only on the occurrence of a fault --> *load on demand policy*



21

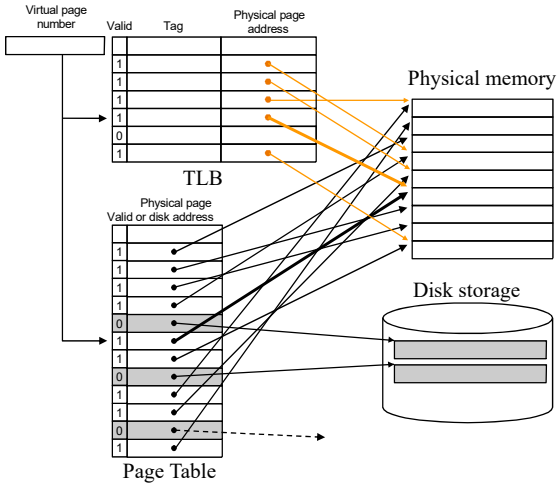
Page Tables



22

Making Address Translation Fast

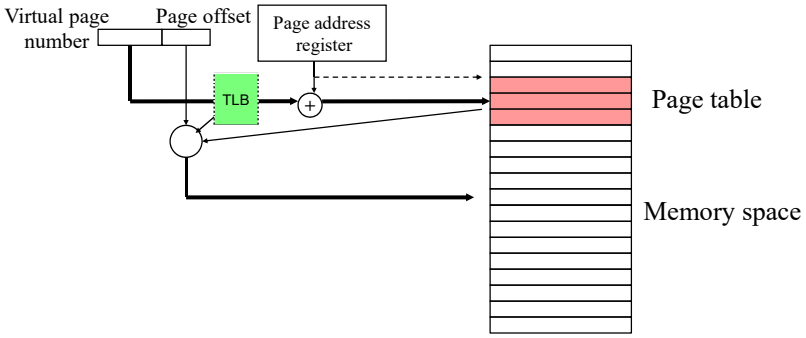
Cache the currently used entries of PT in a *Translation Lookaside Buffer* (TLB).



- What if we get a TLB miss (page table entry is not in TLB)?

23

Page Tables

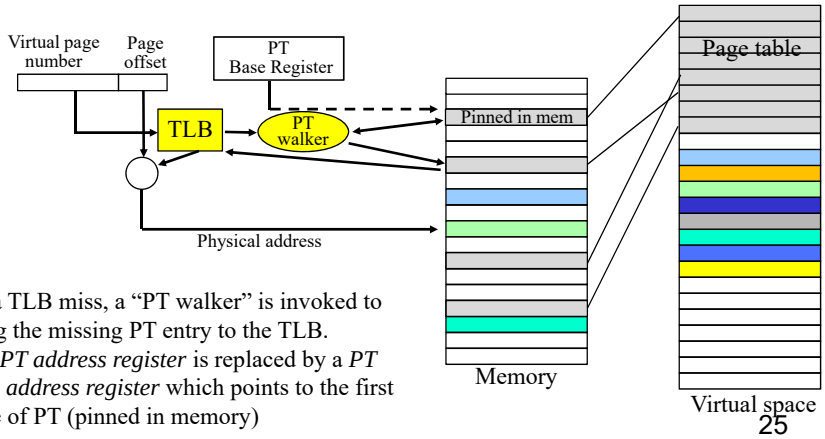


- Each process has its own page table stored in memory starting at a specific address indicated in the *page address register*.
- The page table and page address register are part of the process context
- Each memory reference (hit) requires two memory operations???
- Each page fault requires a memory operation and a disk operation
- The page table can get very large (ex. 32 bit virtual address (4GB), 4KB pages, 4 bytes per PT entry ==> PT size = 4MB = 1024 pages).

24

The Page Table (PT) is very large

- PT's currently used pages are brought to memory, and like any other page in the virtual space, the location of a page in memory is recorded in PT
- Note that (in our example) the 1024 PT entries corresponding to the 1024 pages of the PT can fit in the first page of the PT – That page is pinned in memory.

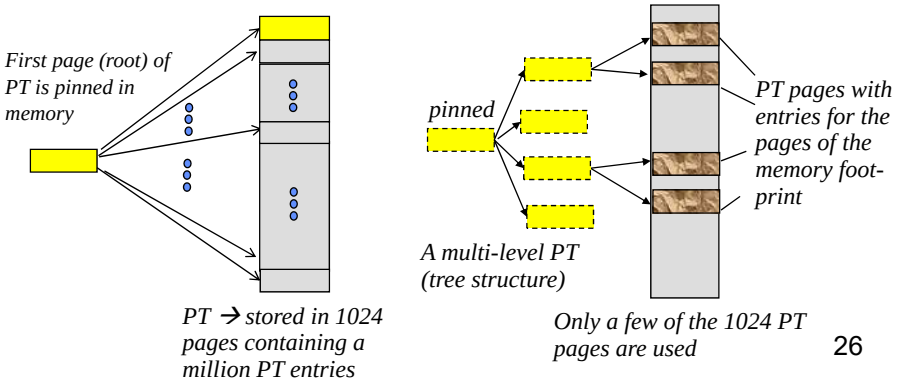


- On a TLB miss, a "PT walker" is invoked to bring the missing PT entry to the TLB.
- The *PT address register* is replaced by a *PT base address register* which points to the first page of PT (pinned in memory)

25

Multi level Page Tables (multi level PT)

- In the example of 4GB VS and 4KB pages, the PT can be stored in 1024 pages
  - Pages of the PT are brought to memory on demand
  - The first page (root) of the PT keeps track of the locations of PT pages in memory.
- This is a "2-level" PT organization – may generalize to a multi-level PT organization
- Memory foot-print = the part of the VS which is actually used (accessed)
  - A large number of pages in the VS are not allocated or used (empty).
  - Hence a large number of entries of the PT are never accessed



26

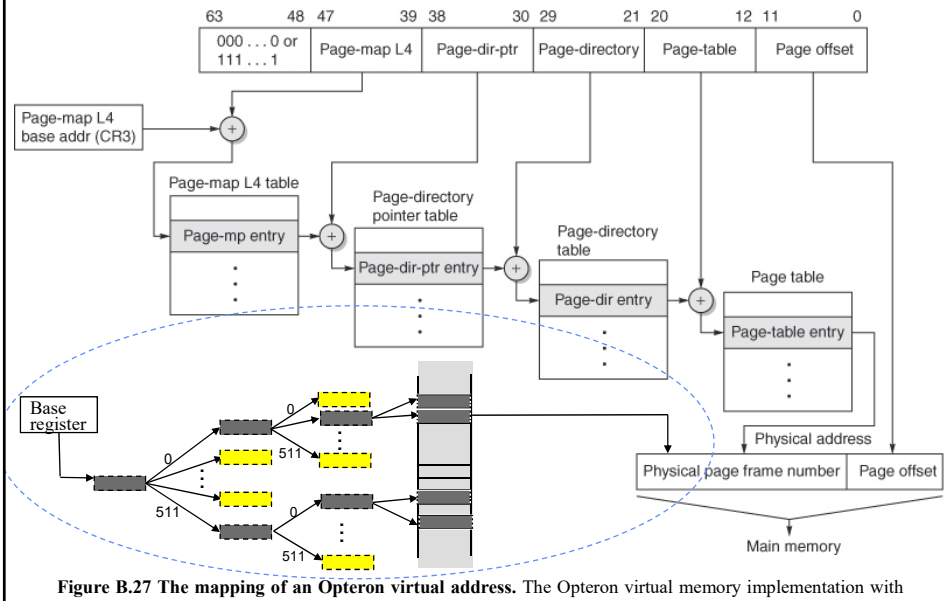
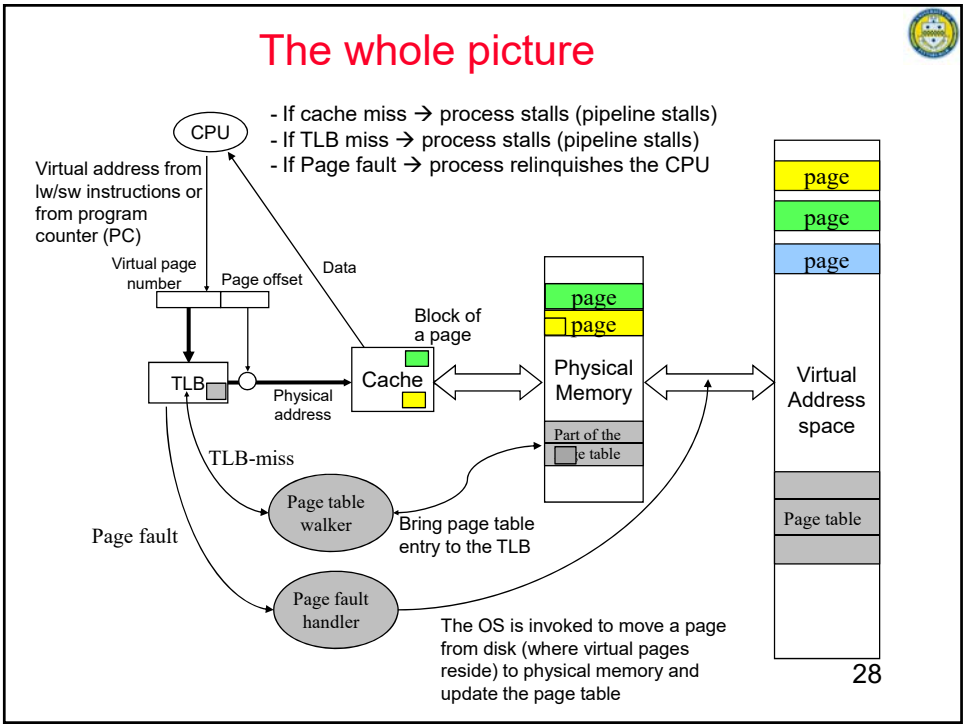


Figure B.27 The mapping of an Opteron virtual address. The Opteron virtual memory implementation with four page table levels supports an effective physical address size of 40 bits. Each page table has 512 entries, so each level field is 9 bits wide. The AMD64 architecture document allows the virtual address size to grow from the current 48 bits to 64 bits, and the physical address size to grow from the current 40 bits to 52 bits.

27



28

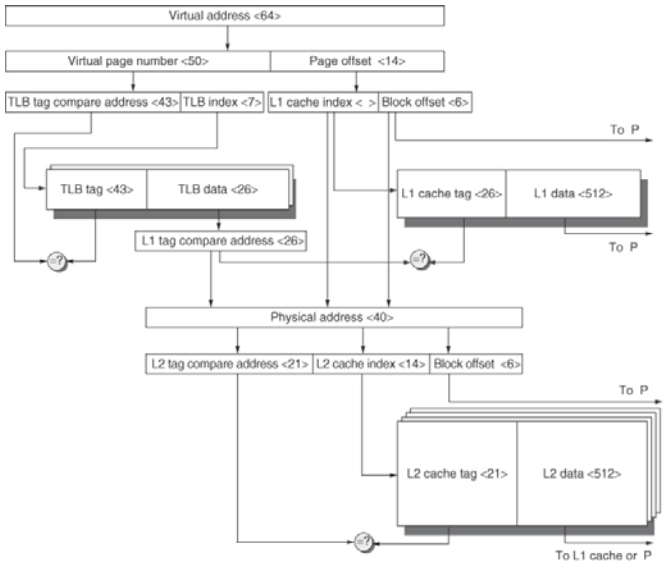


Figure B.17 The overall picture of a hypothetical memory hierarchy going from virtual address to L2 cache access. The page size is 16 KB. The TLB is two-way set associative with 256 entries. The L1 cache is a direct-mapped 16 KB, and the L2 cache is a four-way set associative with a total of 4 MB. Both use 64-byte blocks. The virtual address is 64 bits and the physical address is 40 bits.

29