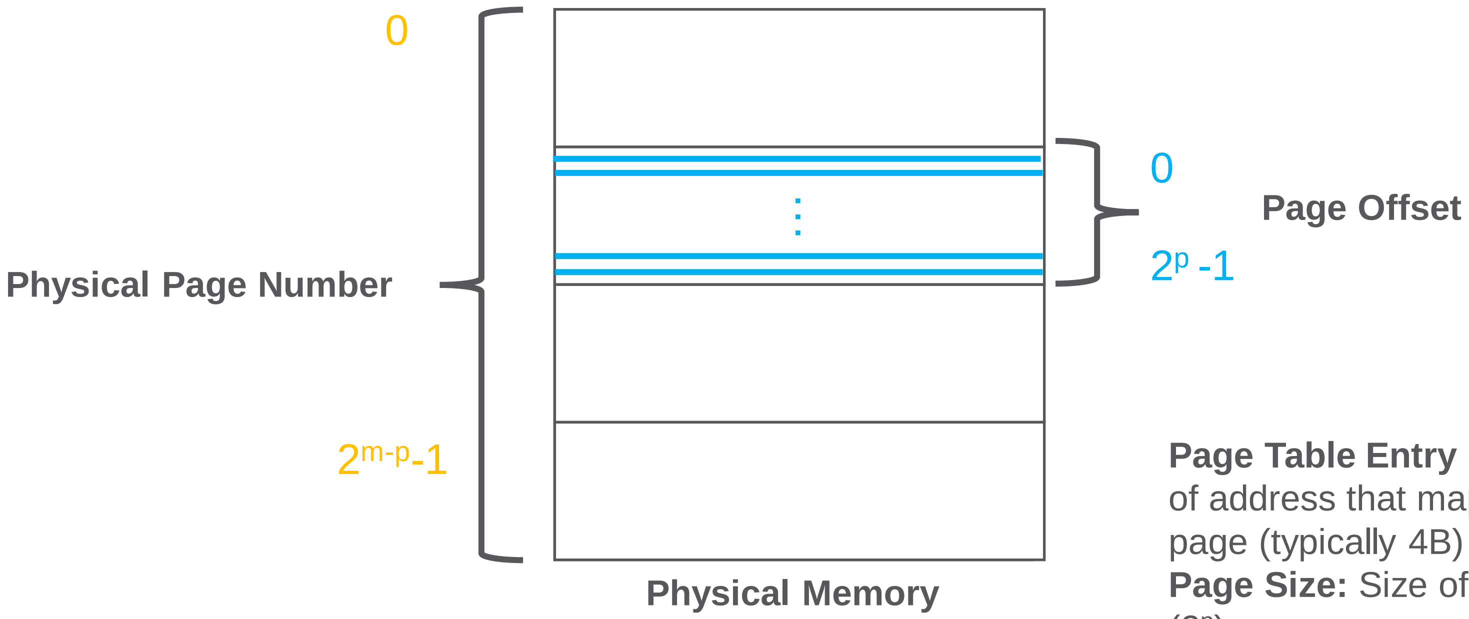
ECE M116C / CS 151B: Week 10 Section

Justin Feng

Page Tables

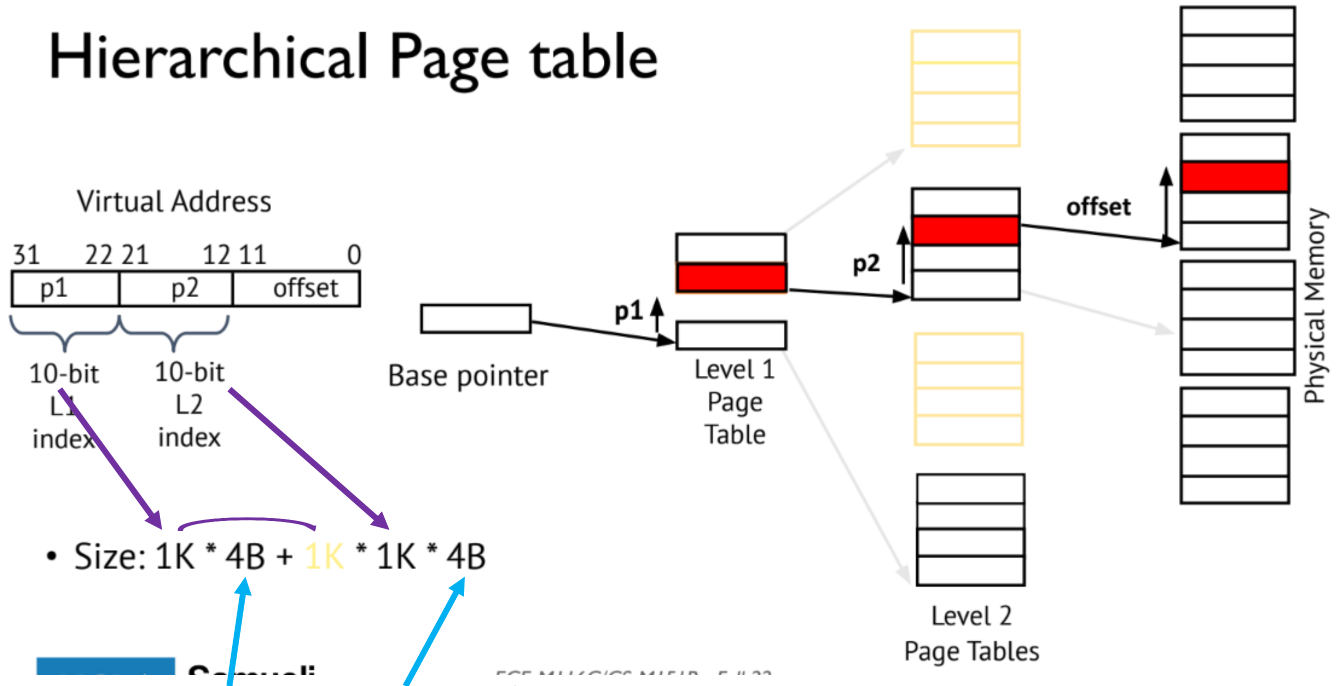


Page Table Entry Size: size of address that maps to the page (typically 4B)
Page Size: Size of the page (2^p)

Hierarchical Page Table

-Virtual addresses are sparse, thus, can make page table storage compact by leveraging this sparsity!

-Issue: more memory accesses



Hierarchical Page table

- Size: 1K * 4B + 1K * 1K * 4B

4B PT Entry Size (**Note: PT Entry size is always same for each level**)

VIPT

-VIPT: Virtually Indexed, Physically Tagged Cache

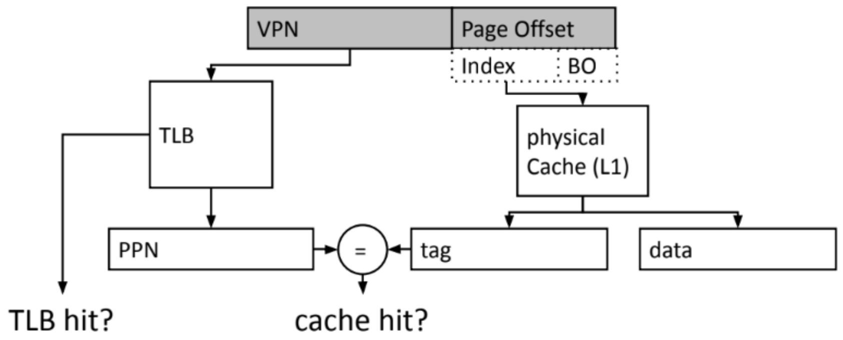
-Goal: access TLB and L1 cache in parallel (can’t access L1 first because of aliasing)

-If we can confirm the PPN = Tag, then we can access the data in the L1 cache

-Hit time: time to access L1 cache

-Limitation: cache is small

-Index bits + BO bits <= Page Offset Bits (If not, aliasing!)



“Weak” Memory Model

- Anything goes: any order of loads, stores, per thread, may be seen
- Need to use fences, barriers, and locks to dictate order of observed execution

Fence vs Barrier vs Lock

- **Fence**: waits for all memory instructions before it to finish (per core)
- **Barrier (external fence)**: blocks execution of threads until the correct number of threads have reached a given point.
- **Lock**: prevents other threads from accessing a shared region of memory

Total Store Order

- Maintain order of stores to stores, and loads to loads, per thread
- Can move loads before other stores, or stores before other loads, as long as there isn't a dependency
  - Not allowed to reorder: load x2, 0(x3)
    store x4, 0(x3)
  - Allowed to reorder: store x2, 0(x3)
    load x4, 0(x5)

UCLA

Sequential Consistency

- The order of loads and stores, per thread, stays the same
- Problem: we lose benefits of multicore. Very slow

HW6, Question 1, a

- Approach:
  - Find possible outcomes of x2, x3, per thread, regardless of memory model
  - Given memory model, find which order of code is allowed/not allowed
  - Find an order of code that results in each possible outcome, if possible given the memory model

HW6, Question 1, b

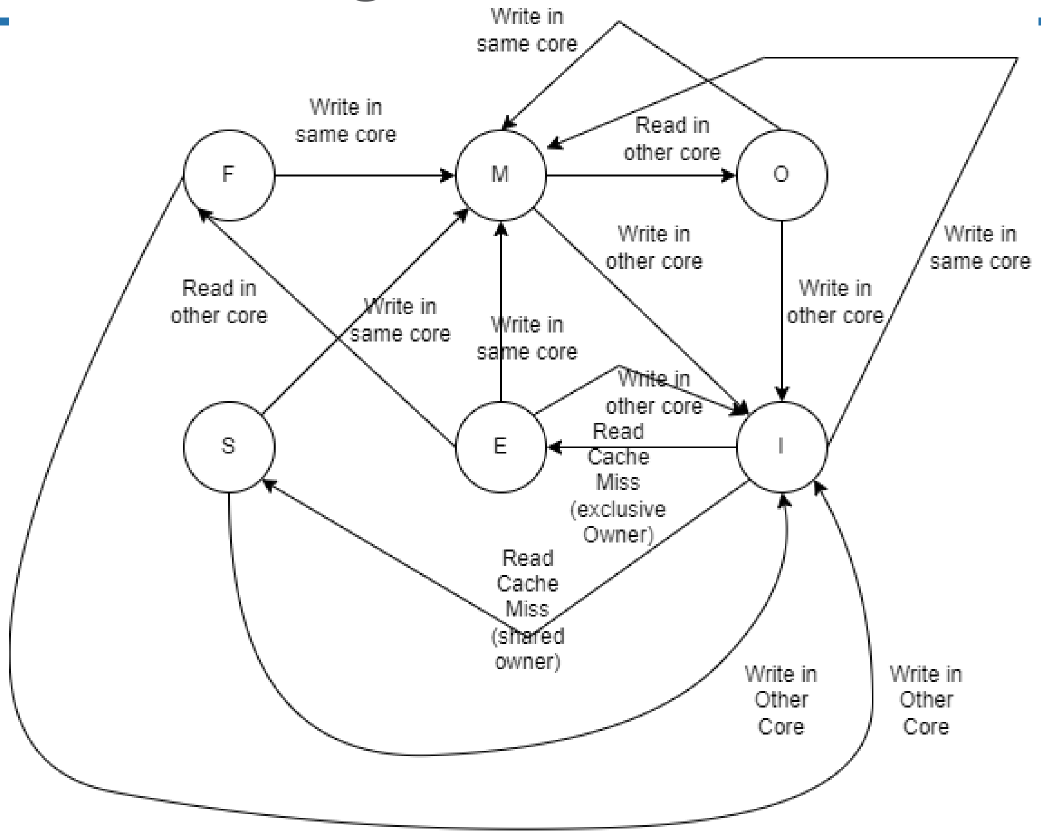
- Approach:
  - Realize that barriers can be used to synchronize between threads (resulting in a desired outcome)
  - Find the values of M[A] and M[B] that are needed in each thread to result in x2=0, x3=0, for each thread
  - Place barriers that make it such that x2 is set to 0, x3 is set to 0, per thread

UCLA

MOESIF: FSM Diagram

- Draw the FSM diagram for MOESIF

MOESIF: FSM Diagram


Write in same core
Write in same core
Read in other core
F
M
O
Write in other core
Write in same core
Read in other core
Write in same core
Write in same core
Write in other core
Write in other core
S
E
I
Read Cache Miss (exclusive Owner)
Read Cache Miss (shared owner)
Write in Other Core
Write in Other Core