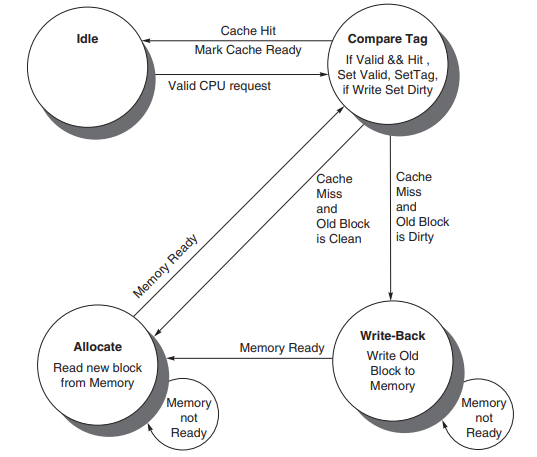
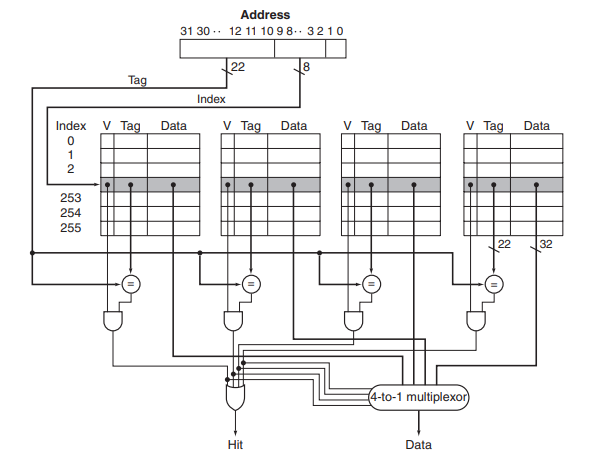
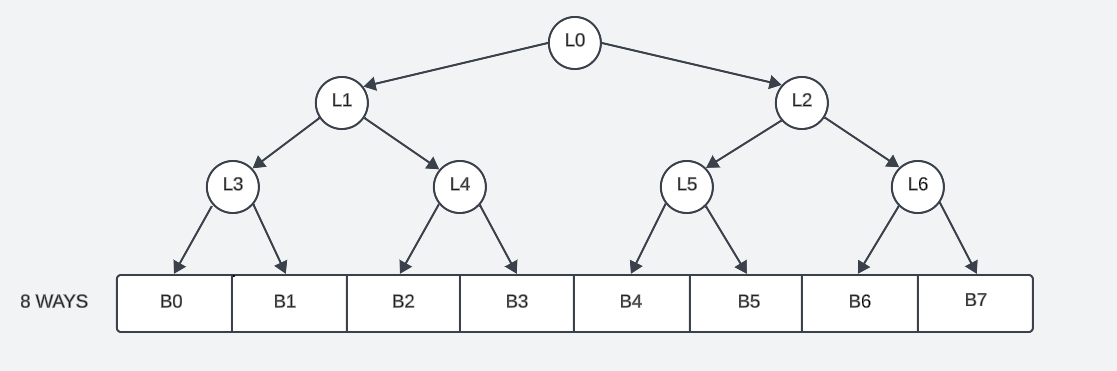
Giản đồ trạng thái của bộ cache controller.



Sơ độ mạch so sánh của bộ set associative cache.

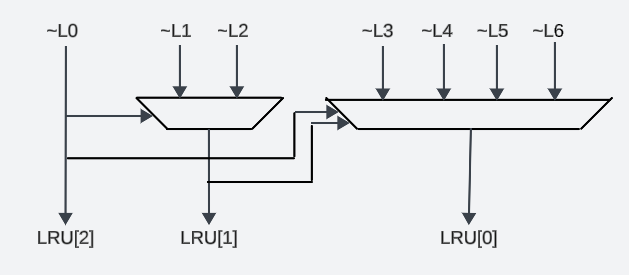


Pseudo LRU 8 ways



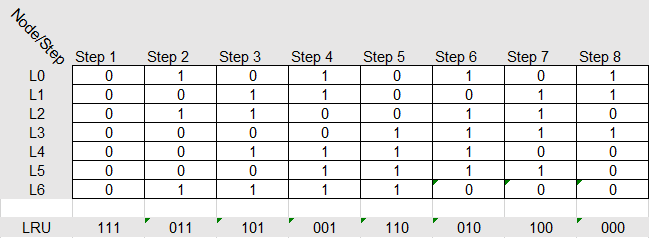
Hình 1.1. Psuedo LRU tree

Trong thiết kế 8-way set associative cache, LRU tree có cấu trúc như hình trên. LRU tree có thể được mô tả như 3 tầng, tầng đầu tiên có L0, tầng thứ hai là L1 và L2, tầng thứ ba chứa L3, L4, L5, L6. Gọi mỗi way là một block (trong block chứa tag và data), ví dụ khi truy cập block B5, giá trị nhị phân là 101, lúc này LRU tree sẽ được cập nhật lại giá trị mỗi lần truy cập, như vậy L0 sẽ bằng 1, L2 bằng 0 và L5 sẽ bằng 1. Như quy tắc trên, mỗi lần truy cập cache, LRU tree sẽ cập nhật lại giá trị và lưu trữ trạng thái ở từng node (L0, L1, L2,…).



Hình 1.2. Psudeo LRU circuit

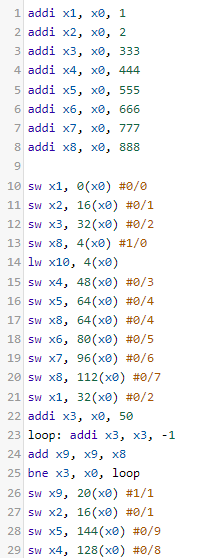
Từ những giá trị đã được lưu trong các node L0, L1, … giá trị LRU sẽ được tính như hình trên, Hình dưới đây sẽ giải thích trực quan hơn các trường hợp cụ thể cách hoạt động của bộ LRU.



Hình 1.3. Giá trị LRU tree ở mỗi lần truy cập

Ở hình trên, giá trị của mỗi node được lưu lại qua từng step (hoặc gọi là lần truy cập), ở step 1 giá trị của các node đều bằng 0, do đó giá trị LRU sẽ có giá trị là 111 như sơ đồ 1.2. Tiếp đến step 2, giá trị LRU ở step 1 sẽ được sử dụng để cập nhật lại LRU tree, ở bước này, giá trị LRU trước đó là 111, do đó L0 được cập nhật bằng 1, vì L0 bằng 1 nên L2 sẽ được cập nhật bằng LRU[1] và bằng 1, và vì L2 bằng 1 nên L6 sẽ được cập nhật bằng LRU[0] và bằng 1. Tương tự với các step sau đó.

Đoạn code assembly như hình dưới sẽ được dùng để kiểm tra hoạt động của bộ 8-way set associative cache.

Ban đầu, các giá trị thanh ghi từ x1 đến x8 được khởi tạo để thuận tiện cho quá trình kiểm tra kết quả.

Ở dòng 10, giá trị thanh ghi x1 sẽ được lưu vào set thứ nhất (mỗi set chứa 8 ways). Tiếp đến x2, x3 cũng được lưu vào set thứ nhất, vì mỗi set chứa 8 block nên lúc này 3 block chứa giá trị của x1, x2, x3 và còn 5 block chưa có giá trị.

Ở dòng 13, giá trị thanh ghi x8 được lưu và set thứ 2. Dòng 14, giá trị ở set thứ 2 sẽ được load vào x10.

Tương ứng các dòng sau đó, x4, x5, x8 (vì x8 được lưu vào địa chỉ 64 đã lưu trước đó nên block có địa chỉ 64 sẽ thay thế x5 bằng x8), x6, x7, x8 lưu vào set thứ nhất. Lúc này set thứ nhất đã lưu đủ 8 giá trị (set thứ nhất đầy).

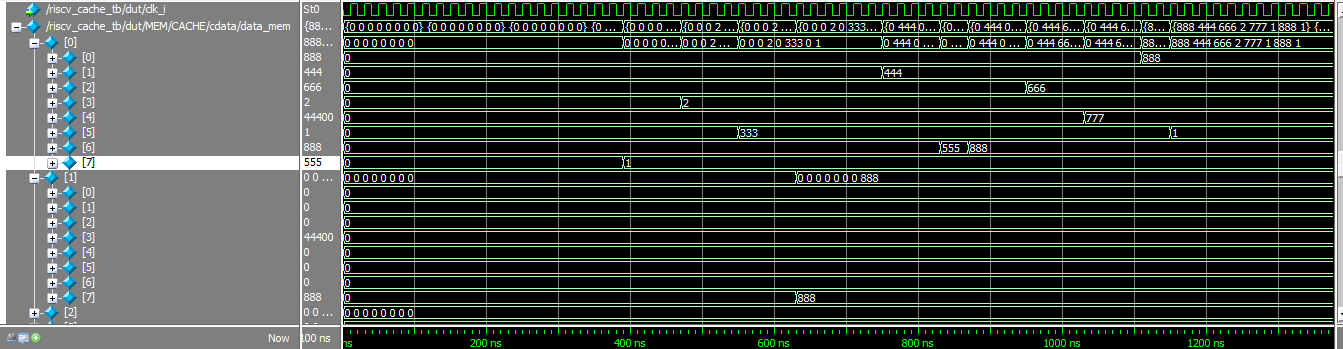
Do đó, lệnh ở dòng 21, ô có địa chỉ 32 chứa giá trị 333 (thanh ghi x3) sẽ được thay thế bằng 1 (thanh ghi x1).

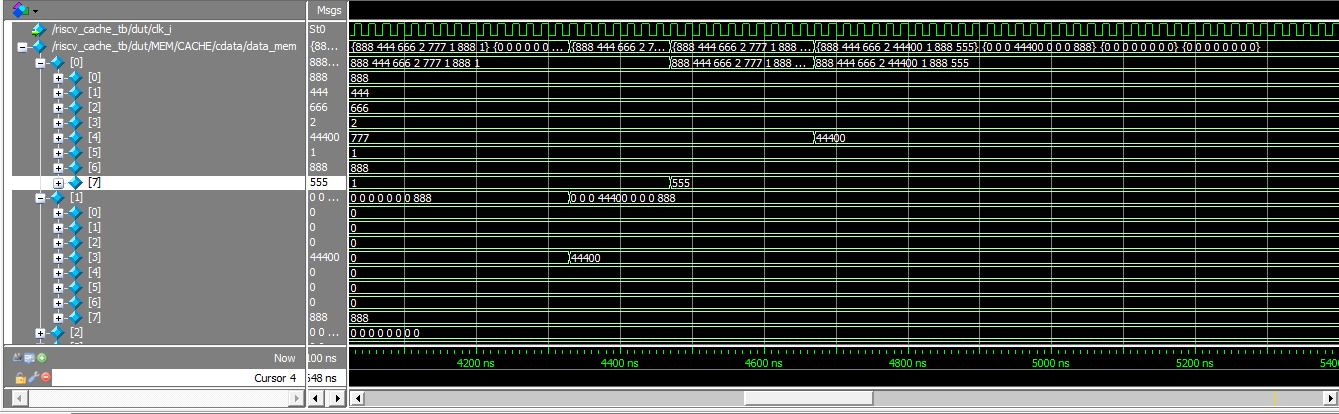
Từ dòng 23 đến dòng 25, giá trị x9 sẽ được tính toán và có giá trị sau cùng là 888 nhân 50 và bằng 44400. Sau đó x9 được lưu vào set thứ 2.

Dòng 27, ô có địa chỉ 16 sẽ cập nhật giá trị của x2.

Ở dòng 28, vì set thứ nhất đã đầy và không có block nào chứa địa chỉ khớp với địa chỉ 144, do đó 1 block nào đó (được quyết định bởi khối LRU) sẽ được lưu lại giá trị vào memory và cập nhật giá trị mới.. Tương tự với dòng lệnh 29.

Dưới đây là waveform mô tả đoạn code assembly phía trên.





Hình 1.4. Waveform cho đoạn test assembly

Như dữ liệu đã tính toán từ hình 1.3, giá trị ban đầu sẽ được lưu vào block 7 tiếp đến lần lượt là block 3, 5, 1, 6, 2, 4, 0. Và mỗi lần truy cập sau đó, LRU tree sẽ được cập nhật và đưa ra địa chỉ của block tiếp theo cần được ghi ở từng set.