JOINT INSTITUTE
交大密西根学院

ADAC LAB

ECE3700J RC 9

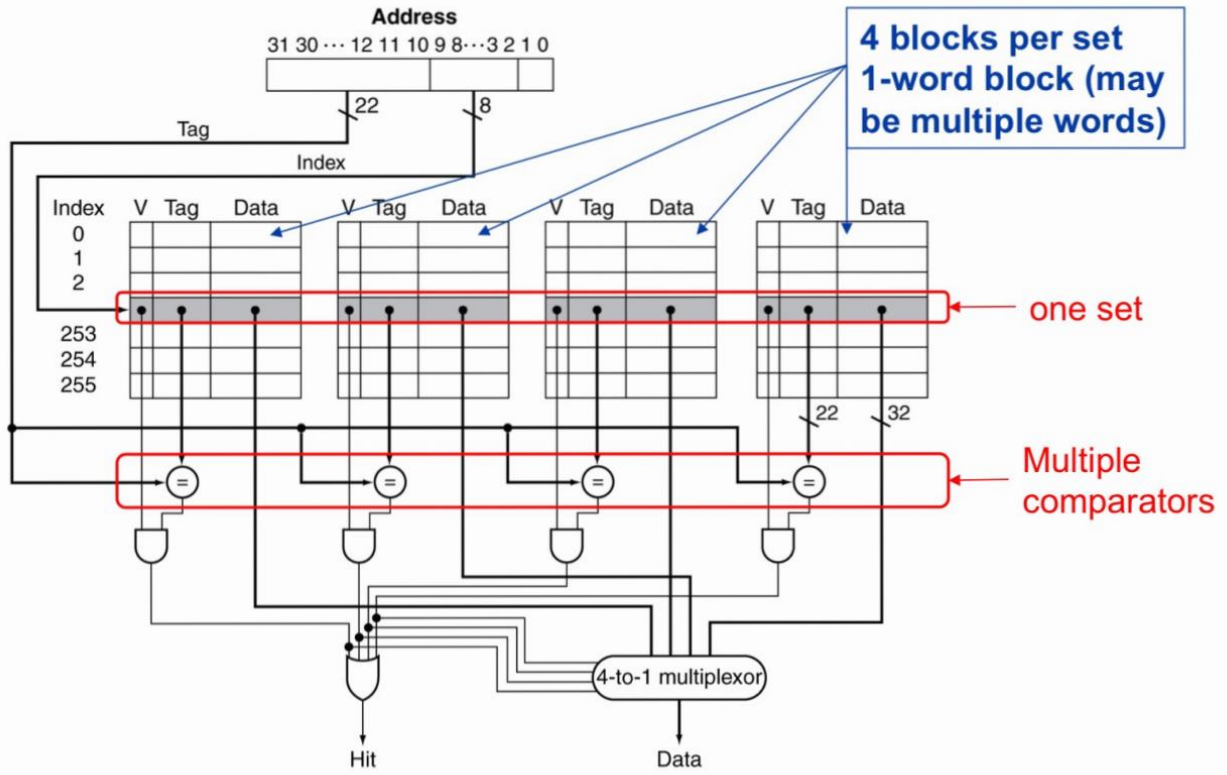
Presenter: Ruan Renjian 阮仁剑

SJTU

Associative

- A set can contain n blocks with the same set index -> Called n-way assocative (special cases: 1-way-direct mapped; all blocks are in 1 set-fully associative)
- Need to compare all tags to locate a specific block

Set index: lower log2(number of set) bits of block address
Tag: bits excluding log2(number of set) bits of block address



Replace Policy

Replacement Policy

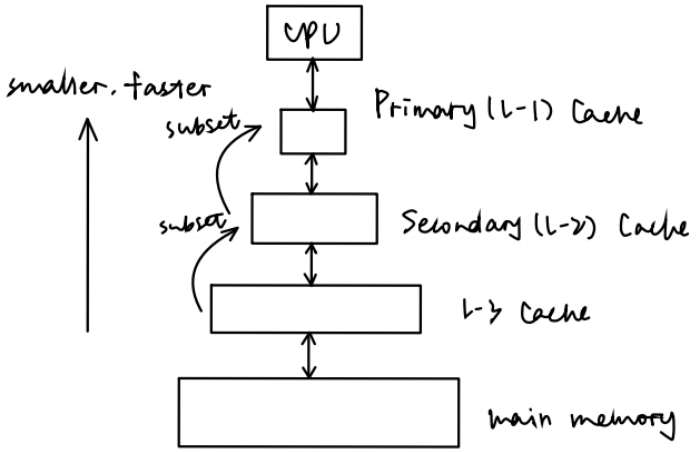
Least Recently Used (LRU)

For set associative scheme, useless for direct mapped

- Choose the one unused for the longest time
- Need a tracking mechanism for usage. But will be more complex for higher level associative

Random replacement is also useful

Multilevel Cache



Virtual Memory

VM Terminology

- In virtual memory context, the data transfer unit is **page** (larger than a block)
- We use **page offset** to locate each byte in a page
- **Page table** is used for translating **virtual page number(VPN)** into **physical page number(PPN)**. And we can use VPN as an index to locate corresponding PPN.
- **Page fault**: The requested page does not exist in the main memory, we need to go to disk to fetch it.

Virtual Address Translation

1. Given a N-bit virtual address
2. Calculate virtual page number for it: higher log2(page size) bits of virtual address
3. Take the virtual page number as index, looking at page table, and fetch PT[index] as the physical page number
4. Physical page offset is just the virtual page offset
5. Combine physical page number of physical page offset and we obtain the physical address

Example

- Given
  - 4KB page size, 16KB physical memory, LRU replacement
  - Virtual address: byte addressable, 20 bits (how many bytes?)
  - Page table for program A stored in page #0 of physical memory, starting at address 0x0100, assume only 2 valid entries in page table:
    - Virtual page number 0 => physical page number 1
    - Virtual page number 1 => physical page number 2
- Show physical memory including page table
- Complete following table

Virtual Address	Virtual page number	Page fault?	Physical Address
0x00F0C			
0x01F0C			
0x20F0C			
0x00100			
0x00200			
0x30000			
0x01FFF			
0x00200			

TLB

Translation Look-aside Buffer

- "Cache" of page table
- 1 translation per entry
- Full associativity
- Arrangement

valid	dirty	reference	tag	physical page number
1 bit	1 bit	1 bit	just VPN	

*Reference bit: Set 1 for every access on that entry (It will be cleared by some module in a certain frequency)
*Dirty bit: Set 1 if the memory data correspond to this physical address is modified

- LRU or random replacement scheme
- Difference between page table: Regard virtual page number as tag but not index to locate an entry

Hit & Miss in TLB

- TLB hit: The requested physical page number is in TLB -> Requested data is in main memory
- TLB miss: The requested physical page number is not in TLB -> Requested data **may** be in main memory, access the page table
  - Located in page table: Load the requested page table entry to TLB
  - Page fault: The page is not in the main memory -> OS handles fetching the page from the disk and updating page table and TLB

TLB

Handle TLB Miss

- If there is still available place in TLB, just fetch an entry from page table
- If TLB is full, replace a TLB entry. Remember to copy back reference bit and dirty bit back to page table.

Usage of TLB

1. Check valid bit field and compare virtual page number with tag field to locate the entry we want in TLB
2. If TLB hit, then we fetch the physical page number in the entry and combine it with the page offset. If TLB miss, change this situation by the steps mentioned above, and then fetch the physical page number to calculate the physical address.