

Four Memory Hierarchy Questions

- Where can a block be placed in a cache?
- How is a block found if it is in the cache?
- Which block should be replaced on a cache miss?
- What happens on a write?

Where can a block be placed in a cache?

- Direct map: every block has only one place it can appear in the cache.
- Usually: $(\text{Block address}) \text{ MOD } (\text{Number of blocks in cache})$
- Fully associative: a block can be placed everywhere in the cache.
- Set associative: a block can be placed in a restricted set of places in the cache.
- 2-way associative means every block can be placed in 2 places in the cache only.

How is a block found if it is in the cache?

- cache have an address tag on each block that gives the block address.
- the tag of every block that might contain the desired information is checked to see if it matches the block address from the CPU.
- all possible tags are searched in parallel because speed is critical (difficult to be fully associative.)
- A valid bit is used to say whether or not a cache block contains a valid address.
- Three portions in an address.

Three portions

- Block offset – address of the desired data within a block (a block has many words, and CPU only want 1 word at a time.)
- Block address index – the index is used to select the set (for set-associative)
- Block address tag – it is used to compared against it for a hit.

which block should be replaced on a cache miss?

- On a cache miss – cache controller must select a block to be replaced.
- If directed-mapped, there is no choice. Since every block address go to only one cache block!
- If set-associative:
 - random
 - LRU (Least recently used) can be hard to calculate
 - FIFO (first in first out)

What happen on a write?

- In SPEC INT 2000, 10% stores, 26% loads.
- Write is $\frac{10\%}{10\%+26\%+100\%} = 7.4\%$ of overall memory traffic.
- Write is $\frac{10\%}{10\%+26\%}$ of data cache.
- making common case fast means optimizing things for reads:
especially since processors usually wait for reads to complete but need not wait for writes.
- block can be read from cache at the same time that tag is read and compare (if the tag is wrong, just forget what was written.)
- block read begins as soon as block address is available.

Write

- Same trick cannot be used for writes.
- Can't begin modifying a block until tag is checked to see if address is a hit.
- tag checking can't occur in parallel implies writes usually takes longer than read.
- processor specifies size of write – usually between 1 to 8 bytes.
- only a portion of the block can be changed.

Two basic options when writing to the cache

- Write through
- information written to the block in the cache and to the block in the lower-level memory
- Write back
- info. written only to the block in the cache.
- the modified cache block is written to main memory only when it is replaced.

More on Write Back

- Write back
- to reduce the frequencies of writing back blocks on replacement
- dirty bit is commonly used – to indicate whether the block is dirty (if it has been modified while in the cache.) or clean (if it is not modified.)

Write back (Advantages)

- Write back – write occur at speed of cache memory
- multiple writes within a block require only ONE write to lower level memory.
- uses less memory bandwidth and less power (good for embedded applications.)

Write Through (Advantage and disadvantage)

- Write through – easier to implement than write back
- cache always clean – read misses never result in a write to lower level.
- next lower level of memory always has the most current copy of the data – simplifies coherency
- CPU must wait for writes to complete during write through (write stall).
- A common optimization to reduce write stalls is a write buffer.

Write Buffer

- write buffer allows processor to continue as soon as data written to buffer – overlapping processor execution with memory updating.
- write stall can occur even with write buffers.
- Data is not needed on a write – two options on a write miss.
- write allocate – allocate the block, and write hit.
- no write allocate – write to lower level memory directly.

AMD Opteron

- Allow 40-bit addressing (no need for 64-bit yet!)
- 64K bytes of data in 64-byte block, two-way set-associative placement.
- Offset has 6-bit
- index has 9-bit
- write only happens after instruction commits.

AMD Opteron

- On read miss, the cache sends a signal to the processor telling it the data is not available.
- Use LRU
- Use write back, keep 1 dirty bit per block to record if the block was written.
- If modified, its data and address are sent to the “Victim Buffer” (similar to write buffer) (only has 8 victim buffers.)
- separate data and instruction cache to allow better performance.