

Wednesday, March 18, 2009

Cache Memory - Fully Associative Mapped Cache

If a Main memory block can be placed in any of the Cache slots, then the cache is said to be mapped in fully associative.

0
tweets
tweet

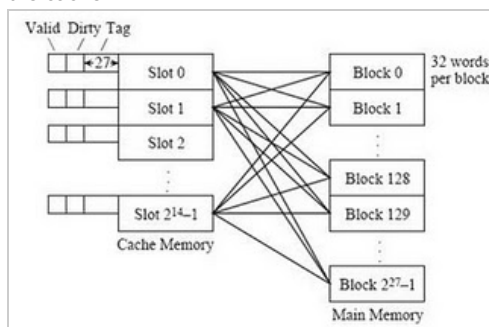
Let us assume we have a Main Memory of size 4GB (2^{32}), with each byte directly addressable by a 32-bit address. We will divide Main memory into blocks of each 32 bytes (2^5). Thus there are 128M (i.e. $2^{32}/2^5 = 2^{27}$) blocks in Main memory.

We have a Cache memory of 512KB (i.e. 2^{19}), divided into blocks of each 32 bytes (2^5). Thus there are 16K (i.e. $2^{19}/2^5 = 2^{14}$) blocks also known as **Cache slots** or **Cache lines** in cache memory. It is clear from above numbers that there are more Main memory blocks than Cache slots.

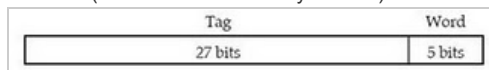
NOTE: The Main memory is not physically partitioned in the given way, but this is the view of Main memory that the cache sees.

NOTE: We are dividing both Main Memory and cache memory into blocks of same size i.e. 32 bytes.

In fully associative mapping any one of the 128M (i.e. 2^{27}) Main memory blocks can be mapped into any of the single Cache slot. To keep track of which one of the 2^{27} possible blocks is in each slot, a 27-bit tag field is added to each slot which holds an identifier in the range from 0 to $2^{27} - 1$. The tag field is the most significant 27 bits of the 32-bit memory address presented to the cache.



In an associative mapped cache, each Main memory block can be mapped to any slot. The mapping from main memory blocks to cache slots is performed by partitioning an address into fields for the tag and the word (also known as the "byte" field) as shown below:



When a reference is made to a Main memory address, the cache hardware intercepts the reference and searches the cache tag memory to see if the requested block is in the cache. For each slot, if the valid bit is 1, then the tag field of the referenced address is compared with the tag field of the slot. All of the tags are searched in parallel, using an associative memory. If any tag in the cache tag memory matches the tag field of the memory reference, then the word is taken from the position in the slot specified by the word field. If the referenced word is not found in the cache, then the main memory block that contains the word is brought into the cache and the referenced word is then taken from the cache. The tag, valid, and dirty fields are updated, and the program resumes execution.

Associative mapped cache has the advantage of placing any main memory block into any

SEARCH THIS BLOG



SUBSCRIBE TO RSS FEED!



FOLLOW ME ON TWITTER!

LABELS

[Announcement](#) [Asterix](#) [Audio](#) [Blah...Blah](#) [Blogger](#) [C](#)
[Comic](#) [Computer](#) [Organization](#) [Cryptography](#) [ctags](#)
[Documentary](#) [Environment](#) [Facts](#) [Firefox](#) [GCC](#) [Google](#)
[Hacking](#) [Hardware](#) [How to](#) [Image](#) [Processing](#)
[India](#) [Laptop](#) [Linux](#) [Memory](#) [Microsoft](#) [Mobile](#)
[Networks](#) [Open Source](#) [PHP](#) [Qemu](#) [RSS](#) [Security](#)
[Signals](#) [Software](#) [Tin](#) [Tin](#) [Video](#) [Windows](#)



BLOG ARCHIVE

- 2010 (8)
- ▼ 2009 (37)
 - December (2)
 - November (2)
 - October (2)
 - September (5)
 - August (3)

cache line. This means that regardless of how irregular the data and program references are, if a slot is available for the block, it can be stored in the cache. This results in considerable hardware overhead needed for **cache bookkeeping**.

Although this mapping scheme is powerful enough to satisfy a wide range of memory access situations, there are two implementation problems that limit performance.

- The process of deciding which slot should be freed when a new block is brought into the cache can be complex. This process requires a significant amount of hardware and introduces delays in memory accesses.
- When the cache is searched, the tag field of the referenced address must be compared with all 2^{14} tag fields in the cache.

LABELS: [COMPUTER ORGANIZATION](#), [MEMORY](#)

1 comments:

Anonymous said...

It is excellent tutorial for fundamentals and basic level people. It can be understood by assuming. {MAHESWARA 9493818580}

December 17, 2010 5:28 PM

[Post a Comment](#)

Links to this post

[Create a Link](#)

[Computer Architecture, 5E](#)

New Hennessy/Patterson's Computer Architecture, 5th Edition
elsevierdirect.com/9780123838728


AdChoices

[Newer Post](#)

[Home](#)

[Older Post](#)

- [July](#) (1)
- [May](#) (2)
- [April](#) (1)
- ▼ [March](#) (6)
 - [Cache Memory - Part2](#)
 - [Principle of Locality](#)
 - [Cache Memory - Set Associative Mapped Cache](#)
 - [Cache Memory - Fully Associative Mapped Cache](#)
 - [Cache Memory - Direct Mapped Cache](#)
 - [Cache Memory - Part1](#)
- [February](#) (9)
- [January](#) (4)
- [2008](#) (39)
- [2007](#) (6)




[Clean your Mac in 5 minutes](#)

Clean your Mac

MacKeeper
will safely
remove gigabytes
of junk from
your Mac

Clean your Mac

[Click here to clean your Mac](#)



POPULAR POSTS

- [Implementation of Singly Linked List](#)
- [Cache Memory - Direct Mapped Cache](#)
- [Cache Memory - Set Associative Mapped Cache](#)
- [TinTin E-Book collection](#)
- [Cache Memory - Fully Associative Mapped Cache](#)
- [iptables - Rate-limit incoming connections](#)

[Singly Linked List in C](#)

[Tin Tin EBook Collection - Part2](#)

[Using GOOGLE as proxy to hide IP-Adress](#)

[Tin Tin EBook Collection - Part3](#)

REALTIME TWITTER UPDATES

VISITORS



- [Live Traffic Feed](#)



See your visitors in RealTime!
Get the Free Live Traffic Feed [Get Feedjit Now!](#)

A visitor from Mestre, Veneto viewed "[Cache Memory - Direct Mapped Cache ~ codingfreak](#)" 1 secs ago

A visitor from Kielce, Swietokrzyskie viewed "[Singly Linked List in C ~ codingfreak](#)" 7 mins ago

A visitor from Durgapur, West Bengal viewed "[Implementation of Singly Linked List ~ codingfreak](#)" 11 mins ago

A visitor from Barcelona, Catalonia viewed "[Calculate Time in Linux ~ codingfreak](#)" 13 mins ago

A visitor from Jaipur, Rajasthan viewed "[Cache Memory - Direct Mapped Cache ~ codingfreak](#)" 15 mins ago

A visitor from Nairobi, Nairobi Area viewed "[Cache Memory - Set Associative Mapped Cache ~ codingfreak](#)" 18 mins ago

A visitor from Mardimago, Veneto viewed "[Cache Memory - Fully Associative Mapped Cache ~ codingfreak](#)" 27 mins ago

A visitor from San Francisco, California viewed "[Implementation of Singly Linked List ~ codingfreak](#)" 28 mins ago

TOTAL PAGEVIEWS

~~1 7 1 0 5 3~~
