ARM Cache 入門

https://blog.csdn.net/sinat_32960911/article/details/127688564

1.1 Cache 概念簡介

1.1.1 Cache Type

Cache 通常按照層級分類，比如常見的 L1 Cache/L2 Cache/L3 Cache ...

通常 L1 Cache 又分為 ICache(instruction cache) 和 DCache(data cache)。ICache 用於快取 CPU 執行的指令，DCache 用於快取 cpu 所使用的資料。

1.1.2 Cache Size

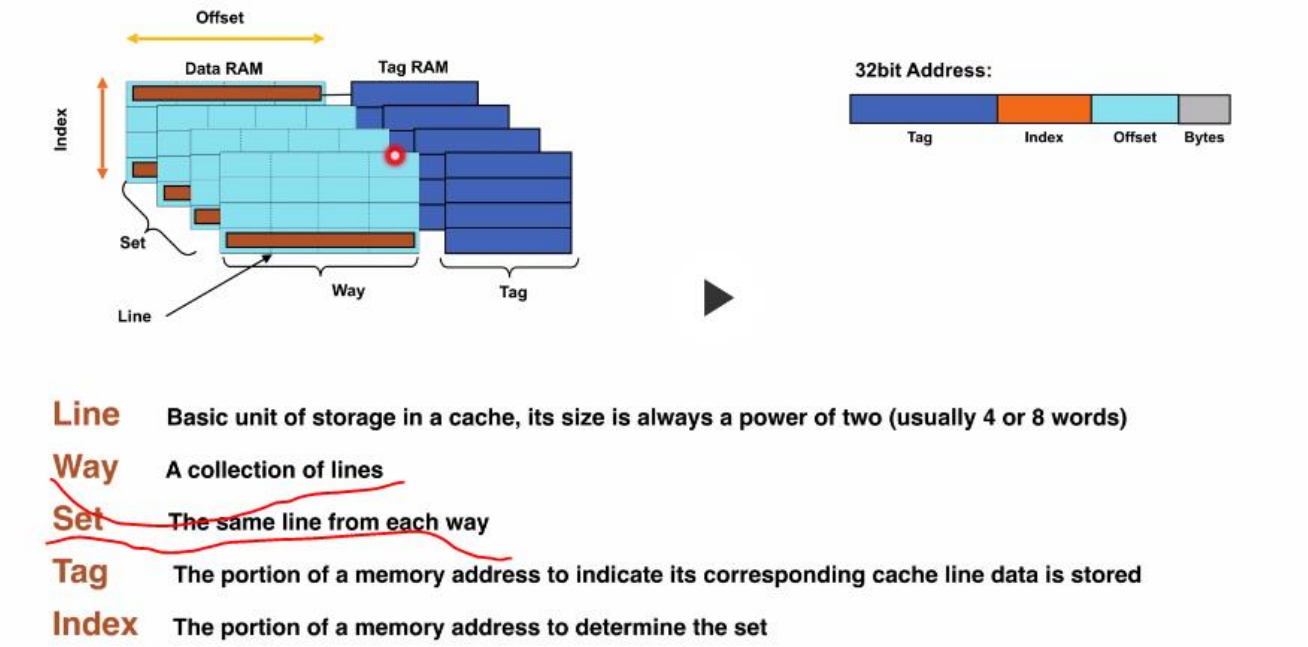
以兩路組相連快取(Two-way set associative cache）為例進行 cache size 介紹，

假設 64 Bytes cache size，cache line size 是 8 Bytes。

那麼什麼是路(way）？

我們將 Cache 平均分成多份，每一份就是一路(Way)。因此，兩路組相連快取就是將 Cache 平均分成 2 份，每份 32 Bytes。cache 被分成 2 路(Way)，每路包含 4 行 cache line。

我們將所有索引一樣的 cache line 組合在一起稱之為組(Set)。



https://aijishu.com/q/1010000000017803

1.1.2 Cache maintenance operations

對 Cache 操作主要為兩種：

invalidate 和 clean

- invalidate 指的是將相應位置的 cache line 狀態置為無效(invalid)，這時候並不需要真的清除相應位置的 cacheline 資料。
- clean cacheline 意味著將 dirty 狀態的 cacheline 寫入主存，同時清除掉 cacheline 的 dirty 位元。通過這種方式可以讓 cacheline 中的資料和主存中的資料一致。

Cache invalid 還可以只針對對應的 cache set、way 或者對應的地址 tag

引用：https://zhuanlan.zhihu.com/p/515450647

具體解釋：

當 cache 外部的儲存器內容發生改變時，需要清除掉相應位置的 cache 狀態。如果 cache 採用的寫策略是 write-back 時，還需要將 cache 中的舊資料刷到外部主存。

對於使用了虛擬記憶體技術的系統，如果 MMU 修改了頁表權限、memory 特性或者 VA2PA 對應關係時也需要對相應的 VA cache 進行 clean 或者 invalidate。

ARM 中通常只使用術語 **clean** 和 **invalidate**，有的地方會使用 **flush(invalidate+clean)**，描述進行 invalidate 和 clean 兩個操作。

invalidate

invalidate 指的是將相應位置的 cache line 狀態置為無效（invalid），這時候並不需要真的清除相應位置的 cacheline 資料。

在一般的系統中，復位必須清除掉所有 cache line 的 valid 狀態，不然的話，這個 cache 的狀態位就能夠讓晶片變成石頭，因為復位後的記憶體訪問可能會 hit，從而拿到錯誤的不可預知的資料。如果 cache 採用了 write-back 寫策略，cacheline 可能包含了 dirty 資料，這個時候直接 invalidate 這個 cache line 也是不對的，可能會丟失本應該寫入到主存中的資料。

clean

clean cacheline 意味著將 dirty 狀態的 cacheline 寫入主存，同時清除掉 cacheline 的 dirty 位元。通過這種方式可以讓 cacheline 中的資料和主存中的資料一致（回想一下，cache 的 write-back 策略帶來了哪些優勢和問題？）。

Cache invalid 可以只針對對應的 cache set、way 或者對應的地址 tag 。

cache invalidate & clean 場景

一個應用 cache invalid 和 clean 的典型場景就是 DMA。

對於 DMA 控製器讀取的地址空間，需要核心 write-back 的 cache 內容對 DMA 控製器可見，所以需要 clean 這個 cache。

當使用 DMA 控製器對外部記憶體寫入資料時，為了讓這個主存的改動對 cache 可見，就需要對受影響的 cache 空間執行 invalid。

1.1.2.1 cache invalidate & clean scenario

一個應用 cache invalid 和 clean 的典型場景就是 DMA。

對於 DMA 控製器讀取的地址空間，需要核心 write-back 的 cache 內容對 DMA 控製器可見，所以需要 clean 這個 cache。

當使用 DMA 控製器對外部記憶體寫入資料時，為了讓這個主存的改動對 cache 可見，就需要對受影響的 cache 空間執行 invalid。

1.1.3 Cache interaction with memory system

- 在 disable 或者 enable 指令 cache 之後，必須執行一條 ISB 指令，用來 flush cpu pipeline。
- 在 reset 之後，必須在做了 cache invalidate 操作之後才能進行 cache enable 操作，因為如果不做 invalidate cache 的在 cache enable 之後有可能發生 cache hit, 這個時候 cache 中的資料是未知，可能會導致致命問題。

- 在 disable cache 之後必須要 clean cache, 這樣才能確保 dirty data 被寫入外部儲存。
- 在使能 data cache 之前，必須先進行 invalidate 整個 data cache, 因為在外部儲存中的資料再關閉 data cache 之後可能被改寫，如果沒有做 invalidate 操作，這時如果 cache hit，那麼 cpu 拿到的資料並不是外部儲存中最新的資料而是舊資料，這樣可能會導致致命問題。
- 在使能 指令 cache 同樣需要先進行整個 指令 cache 的 invalidate 操作。

1.1.4 write back/write allocate

1.1.4.1 CPU 讀 Cache

Read Through: 即直接從記憶體中讀取資料；

Read Allocate: 先把資料讀取到 Cache 中，再從 Cache 中讀資料。

1.1.4.2 CPU 寫 Cache

(1) 若 hit 命中，有兩種處理方式：

cache 的 write-through/write-back policy 是基於 cache 命中的基礎上，如果 cache 沒有命中，是沒有意義的：

- Write-through: 在資料更新時，把資料同時寫入 Cache 和後端儲存。此模式的優點是操作簡單；缺點是因為資料修改需要同時寫入儲存，資料寫入速度較慢。
- Write-back: 在資料更新時唯寫入快取 Cache，並使用 dirty 標誌位記錄 cache 的修改，直到被修改的 cache 塊被替換時，才把修改的內容寫回 main memory。優點是資料寫入速度快，因為不需要寫儲存；缺點是一旦更新後的資料未被寫入儲存時出現系

統掉電的情況，資料將無法找回。

(2) 若 miss，有兩種處理方式：

在寫失效(write miss) 時，即所要寫的地址不在 cache tag 中：

- Write Allocate: 把要寫的地址調入 cache 中，再寫 Cache，後面再通過 flush 方式寫入到記憶體中。
- No-write allocate: 並不將寫入地址讀入 cache，而是直接把要寫的資料寫入到記憶體中。

同理，read allocate policy 就是當要讀的地址不在 cache 中時，把要讀的地址所在的塊先從 main memory 調入 cache 中，然後再從 cache 中讀。

需要注意的是組態成 write allocate 和 no write allocate 差異是很大的

1.2 Cache Coherence(一致性問題)

所謂的 Cache 一致性問題， 主要指的是由於 Cache 存在時，當在有多個 Master(典型的如 MCU 的 Core， DMA 、PCIE、I2C2APB 等)訪問同一塊記憶體時， 由於資料會快取在 Cache 中而沒有更新實際的實體記憶體，導致的問題。在瞭解快取一致性前需要先瞭解記憶體順序模型。

1.2.1 記憶體順序模型簡介(Memory Model)

arm memory 類型分為 normal memory 和 device memory。

1.2.1.1 Normal Memory

Normal memory 就是我們平常所說的記憶體，對該種 memory 訪問時無副作用(side effect)，

即第 n 次訪問與第 n+1 次訪問沒有任何差別。

1.2.1.2 Device Memory

device memory 是外設對應的實體位址空間，對該部分 memory 訪問時，可能存在副作用(side effect)，比如：

- 某些狀態暫存器可能 read clear；
- 某些暫存器有寫入順序（否則寫入不成功）；
- 裝置 fifo 地址固定不變，但是每次訪問，內部的移位暫存器就會將下一個資料移出來[1]，因此訪問同一地址第 n 次訪問與第 n+1 次訪問結果是不同的。

1.2.2 Cache 一致性問題解決方案

- 所有的共享儲存器都定義為共享屬性(Shareability)，共享意味著需要硬體保證一個記憶體位置中的內容對一定範圍內可訪問該位置的多個處理器是一致。
- 通過軟體進行 cache 的維護, 如使用 cache invalidate 和 cache clean 進行維護。

1.2.3 Shareability

Shareability 的由來為了支援資料一致性協議，需要增加硬體很多開銷，會降低系統的性能，同時也會增加系統的功耗。但是，很多時候並不需要系統中的所有模組之間都保持資料一致性，而只需要在系統中的某些模組之間保證資料一致性就行了。因此，需要對系統中的所有模組，根據資料一致性的要求，做出更細粒度的劃分。

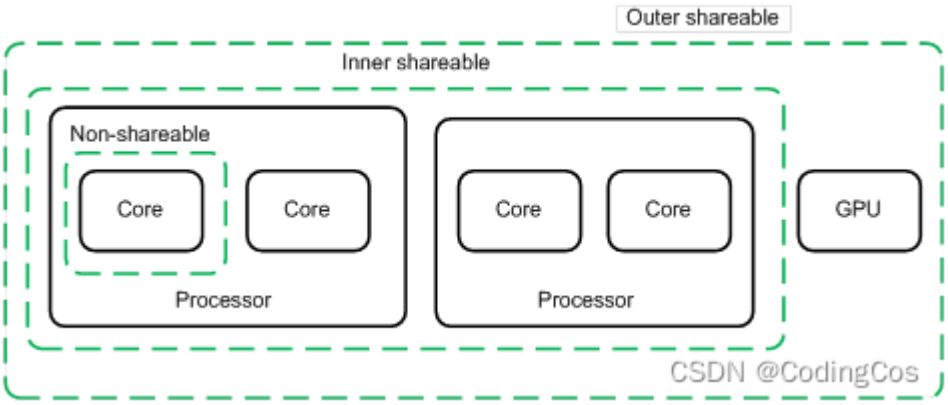
1.2.3.1 Non-Shareable(不共享)

組態為 non-shareable 屬性的記憶體位置一般只能被唯一處理器訪問, 如果還有其他處理器能訪問該位置，需要軟體用快取一致性指令來保證快取一致性。

比如：在單核的場景下，cpu 往某一塊組態為 non-shareable (同時組態了 cacheable)屬性的記憶體寫一段資料，由於這段記憶體只對 cpu 可見，所以如果當使用 DMA 來搬運這塊記憶體中的資料時，需要先進行 cache clean 將快取中的資料刷入 memory 中，否者 dma 搬運的資料可能會有一些 stale data。

1.2.3.2 Inner-Shareable(內部共享)

該記憶體位置可以被 Inner Shareability domain 中的所有處理器訪問，並且硬體保證該位置在這些處理器間的資料一致性，Inner Shareability domain 中的處理器一般被同一個虛擬機器監視器或作業系統控制，如下圖中的兩個 cluseter 都在 inner shareability domain 中。一般不同的 cluseter 會共享 L2 cache。



1.2.3.3 Out-Shareable(外部共享)

能被外部共享的觀察者(cpu, gpu, dma) 觀察到，它適用於內部可共享和外部可共享域。一個 outer shareable domain 可以由一個或多個 inner shareable domain 組成，並且當一個操作影響到 outer shareable domain 時，也會影響到其下所有的 inner shareable domain。

note: 只有組態為 Normal Memory 記憶體屬性的記憶體才能設定 inner 和 outer

shareability，device memory 是不能設定 Shareability。