Domande Computer Architecture

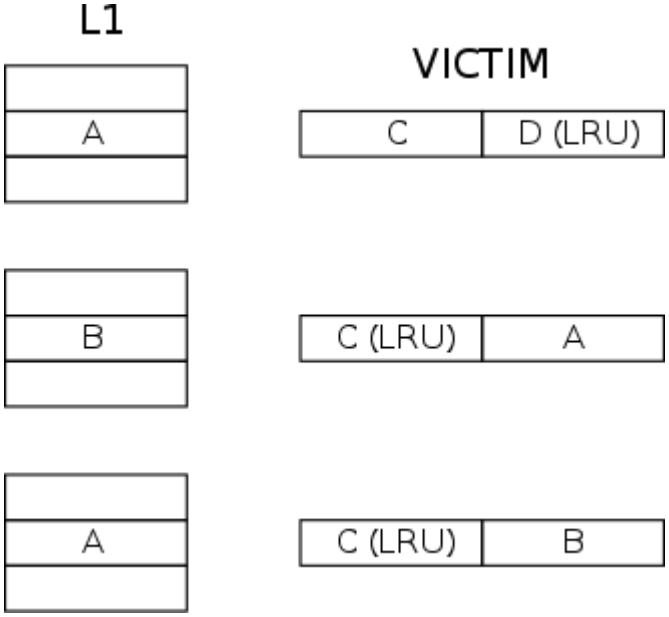
Raccolta di domande fatte durante le lezioni.

1. [presentazione sui droni] il drone lo costruisco per fare delle elaborazioni (esempio ricostruire la zona con tutte le altezze senza andare a fare misure dirette). Immaginiamo di disporre di un'applicazione in grado di costruire tutto il profilo di un palazzo, ha piú senso, dal punto di vista del consumo, utilizzare l'energia per spedire il filmato a terra oppure creare automaticamente le immagini sul drone e inviare le foto, parzialmente elaborate, a terra. (tipo watt per byte o watt per frame)

2. Devo ridurre i conflitti sulla stessa struttura offrendo due copie di tags, una per offrire il tag in caso di miss della cache di livello uno, una per rispondere alle operazioni effettuate sul bus condiviso. Se ho due copie, devo aggiornarle quando voglio cambiare queste due copie. Questo é un problema?
   L'intervallo di tempo in cui devo cambiare i due valori del tag, potrebbe causare un effetto collaterale sulla cache di primo livello o sul bus condiviso.
   Il problema é che l'attivitá tradizionale del primo livello di cache é risolvere le miss del secondo livello e per cercare quale blocco caricare bisogna cercare attraverso il tag.
   Quando devo aggiornare i tag devo eseguire un'azione atomica per evitare incosistenze.
   Questa update puó essere un problema dal punto di vista delle performance del processore?
   Quando devo aggiornare entrambe le copie del tag? In caso di miss nel secondo livello di cache, devo scrivere un nuovo valore nelle copie di tag.
   La soluzione é che la condizione per la quale devo aggiornare entrambi i tag é nel caso di miss nella cache di secondo livello, ma quando lo faccio uso il bus condiviso per chiedere una copia del blocco coinvolto nella miss, quindi il bus é occupato da un'operazione richiesta dalla cache di secondo livello, quindi non c'é un drop delle performance.

3. Che side effect potrebbe avere la inclusion property? Nell'ultimo microprocessore intel la inclusion property non é garantita perché puó portare a problemi dal punto di vista energetico. Il problema é che se i tre livelli di cache devono avere gli stessi blocchi, la dimensione massima della cache é data dall'ultimo livello di cache, quella piú piccola e questo puó peggiorare le performance. La soluzione é l'uso di una victim cache che permette di usare una copia locale senza richiederla attraverso il bus condiviso.
   La inclusion property é usata per ridurre la complessitá del sistema ma questo puó portare a problemi di performance.
   Come si puó risolvere? **Victim cache**, Miss caching places a fully-associative cache between cache and its re-fill path. Misses in the cache that hit in the miss cache have a one cycle penalty, as opposed to a many cycle miss penalty without the miss cache. Victim Caching is an improvement to miss caching that loads the small fully-associative cache with victim of a miss and not the requested cache line.
   A victim cache is a hardware cache designed to decrease conflict misses and improve hit latency for direct-mapped caches. It is employed at the refill path of a Level 1 cache, such that any cache-line which gets evicted from the cache is cached in the victim cache. Thus, the victim cache gets populated only when data is thrown out of Level 1 cache. In case of a miss in Level 1, the missed entry is looked up in the victim cache. If the resulting access is a hit, the

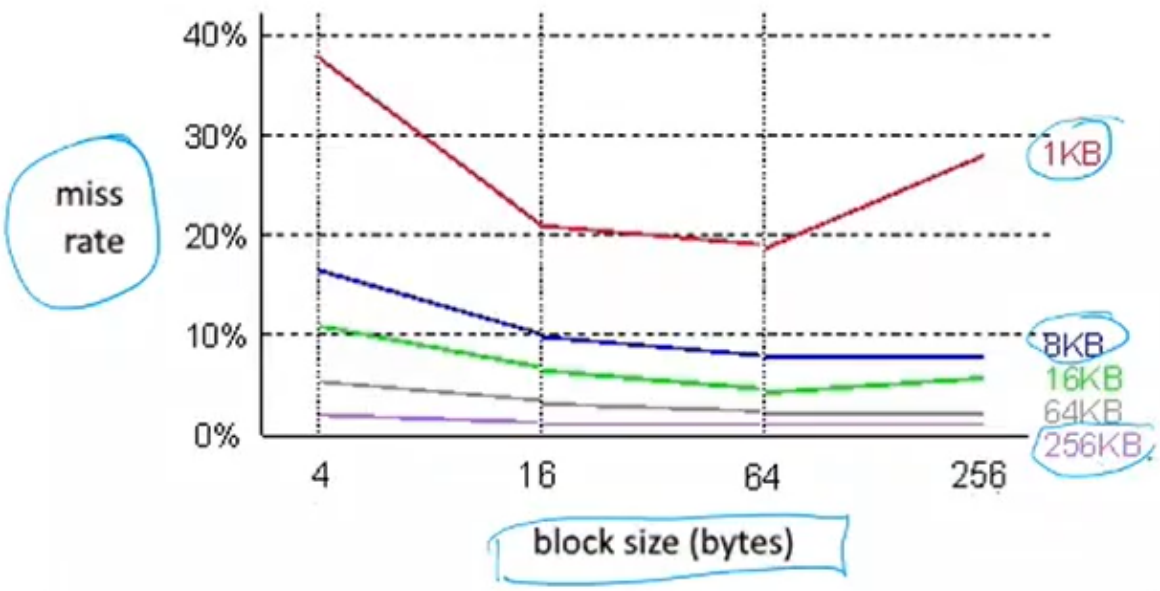
contents of the Level 1 cache-line and the matching victim cache line are swapped.



4. Perché all'interno di un multiprocessore non ho problemi di workload-balancing?

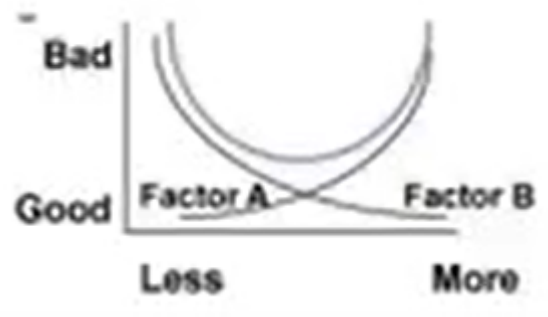
Il workload bilanciato puó essere ottenuto automaticamente tramite lo scheduler che offre un task diverso ad ogni processore. L'unico problema é riuscire a fornire un numero di processi pronti tale da fare operare il sistema alla massima utilizzazione e minimizzare la comunicazione tra processi.

5. Andamento miss rate in base alla grandezza del blocco. Vario la dimensione del blocco in bytes (4-16-64-256) e la dimensione della cache in kB (1-8-16-64-256). Perché ha quell'andamento? (**domanda tipica**)



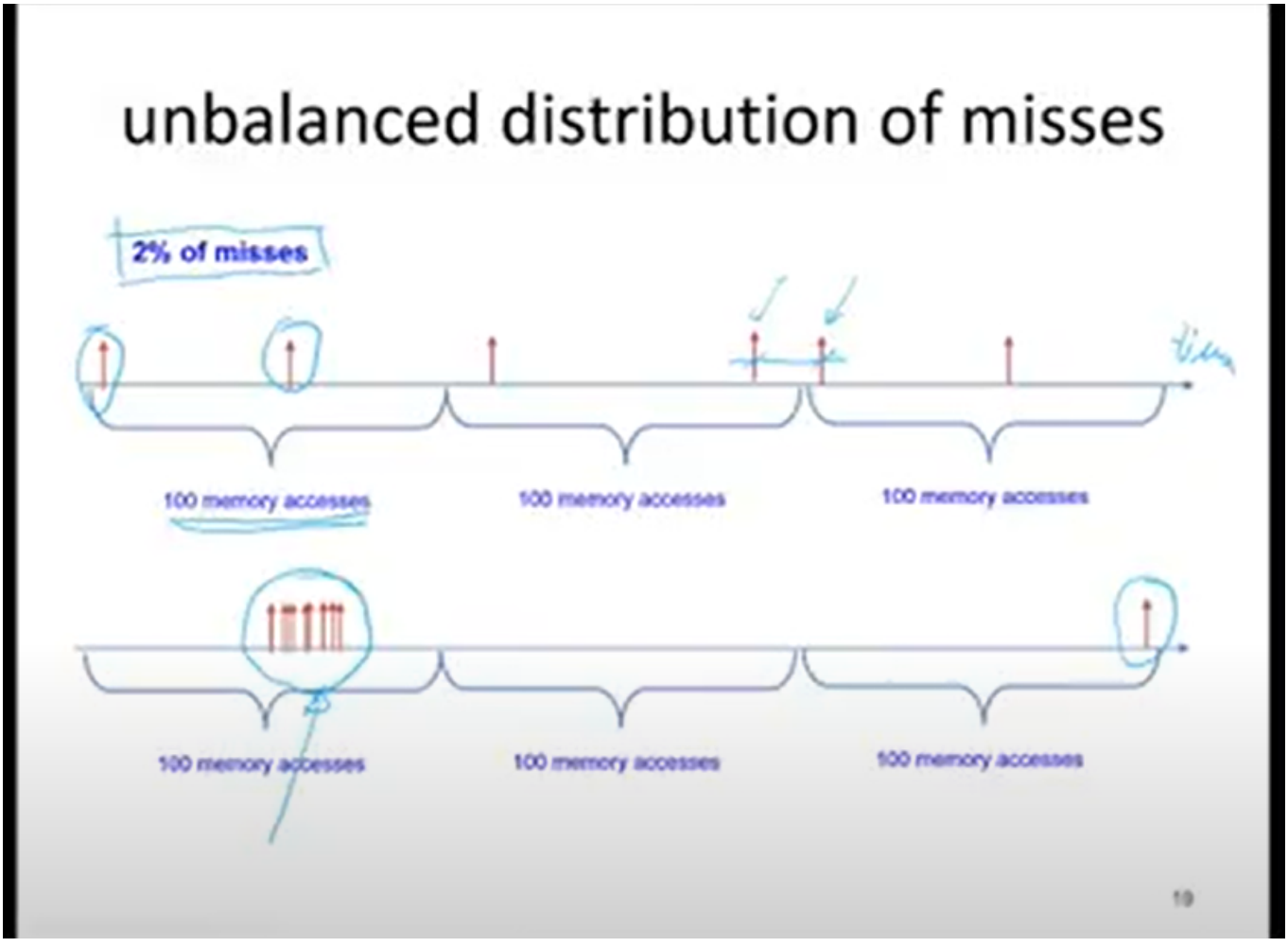
In generale la miss rate va giú quando la dimensione del blocco aumenta. Nel caso della cache da 1 kB invece la miss rate aumenta.
Una soluzione é migliore nel caso cerchiamo *spatial locality*, una nel caso cerchiamo *temporal locality*.
Nel caso di temporal locality voglio avere blocchi piccoli (4 bytes) perché voglio caricarne il piú possibile.

Nel caso di spatial locality vogliamo avere la porzione piú grande di memoria possibile attorno all'indirizzo della miss. (massimizzare la dimensione del blocco)
Tornando al caso della cache da 1 KB, in caso di blocchi da 256 bytes avró solamente 4 blocchi in cache, che é la migliore situazione per la spatial locality, ma decisamente la soluzione non ottimale per la temporal locality.
Nel caso di blocchi da 4 bytes avró 256 blocchi.
Dobbiamo quindi cercare la dimensione ottimale che solitamente é vicina alla dimensione di una word in cache moltiplicata per quattro.
Soluzione = sizeof(word) * 4.



Il valore ottimo peró potrebbe dipendere da applicazione a applicazione, dato che cambia in base al tipo di accessi che fará l'applicazione in cache.

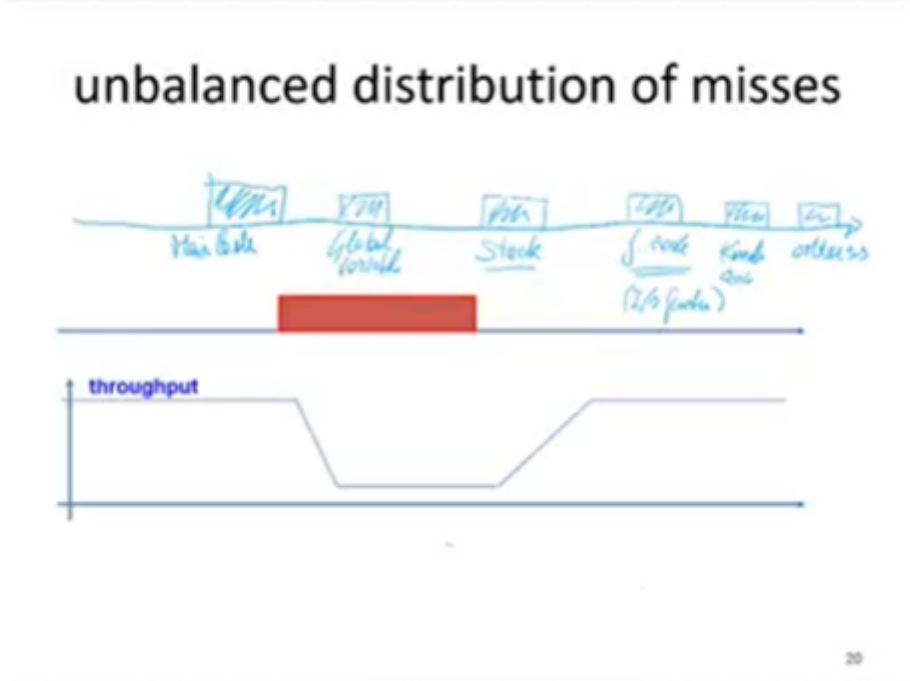
6. Perché potrebbero esserci molte miss in un intervallo molto ristretto di tempo e una molto dopo?



- chiamate di funzioni (call)
- cambio di contesto tra processi
- interrupt (int)

7. Quali sono le possibili aree nelle quali avró gli accessi? (legata a quella precedente)

- main code
- global variables
- stack
- codice di funzioni: codice usato per chiamare le librerie, funzioni I/O
- codice del kernel
- dati del kernel



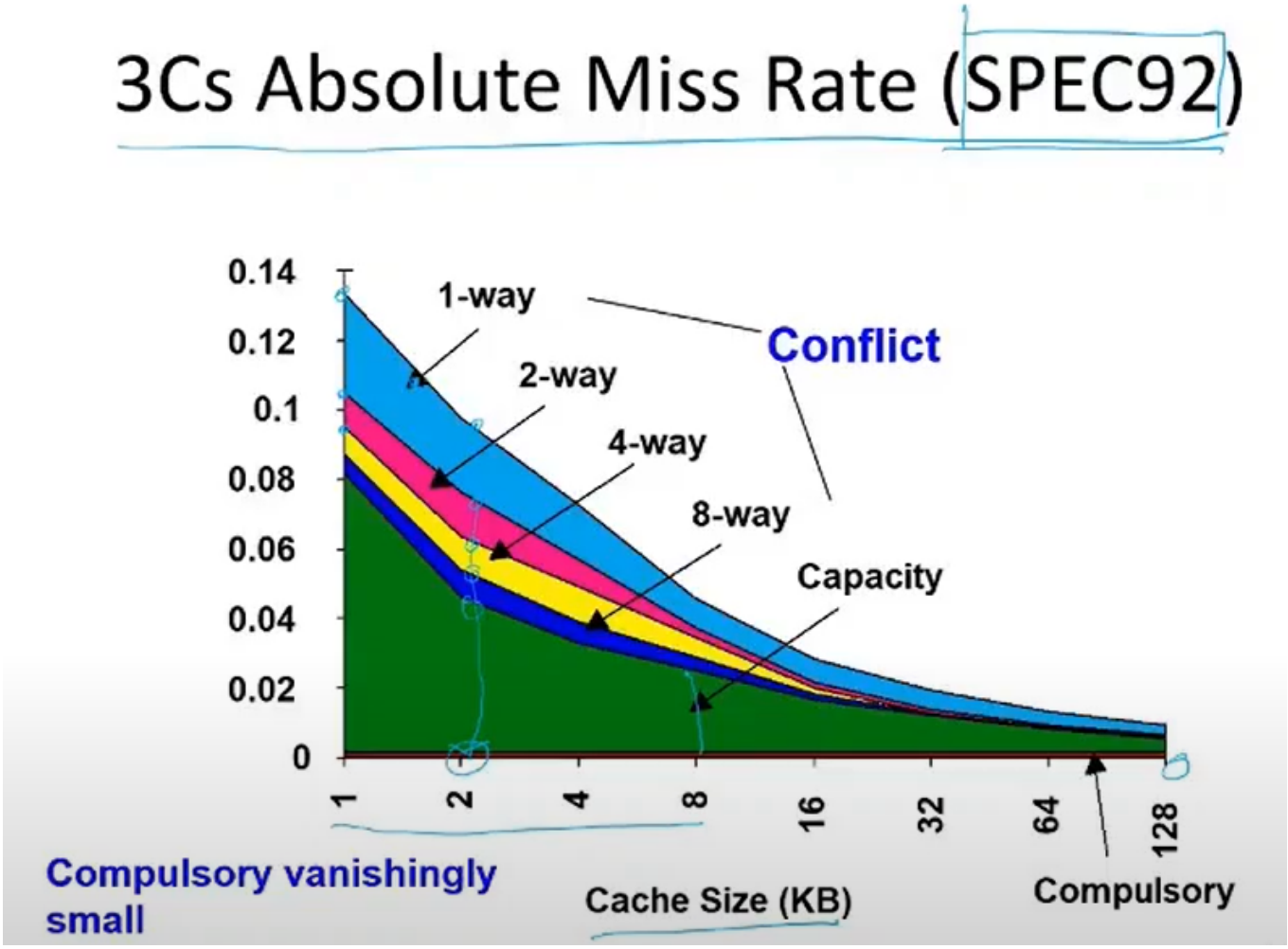
Conclusioni che possiamo trarre?
Come posso ridurre il numero di miss?

Una buona idea é separare le cache per i dati e le cache per il codice.
2 cache, una per il codice e una per i dati.
Da una parte posso leggere i dati, dall'altra fare il fetch di una nuova instruzione. Posso aumentare il parallelismo tra l'accesso ai dati e l'accesso al codice.
Possiamo organizzare questi due livelli di cache in modi diversi, dato che i meccanismi di localitá spaziale e temporale delle cache dei dati e del codice sono molto diversi.
Direct-mapped cache o set-associative?
Set-associative, potrebbe esserci dei conflitti dati dal fatto che potrebbero essere eseguite delle instruzioni contenute nello stesso blocco.

Quanti blocchi per set?

4-way associative cache, perché altrimenti un loop (grande) potrebbe eliminare le copie in cache appartenenti allo stesso loop.

8. Come possiamo valutare la distribuzione delle miss per una cache quando il processore sta eseguendo un particolare programma, riuscendo ad individuare la causa della miss?



Per la causa *compulsory* bisogna considerare il numero di blocchi usati per accessi in sequenza.
Per *capacity* bisogna considerare la diminuzione della capacitá della cache stessa.
Per le *way*, si usa la **2:1 cache rule**.

9. Perché i microprocessori attuali usano una cache ad indirizzi fisici al posto di una ad indirizzi virtuali?

Perché due indirizzi virtuali diversi possono mappare lo stesso indirizzo fisico (alias), che invece é univoco.
(detto meglio) Le pagine fisiche possono essere condivise tramite due indirizzi virtuali che puntano allo stesso indirizzo fisico, questo serve per permettere a programmi diversi di condividere porzioni di dati o di codice.

10. Ad un cambio di contesto posso avere molte miss nello stesso momento, come posso evitarlo?

Il problema si verifica quando due o piú processori hanno un context switch nello stesso momento. Per risolvere quindi basta non permettere di avere cambi di contesto nello stesso momento.

11. [Domanda presentazione per noi] La sicurezza consuma energia, provate a descrivere la sicurezza a livelli. Si puó scegliere il livello di sicurezza che si vuole (l'utente). Vediamo se é

ragionevole o no conoscere il livello di sicurezza necessario e vedere il consumo della parte che potrebbe essere "in piú".

12. Come si puó garantire la mutua esclusione in modo atomico? Introdurre una singola istruzione che fa tutto. Se dobbiamo fare la lock e la unlock su un certo valore, senza un'istruzione singola (read, write, load, store), come possiamo fare? Disabilitare le interruzioni. Conoscete un modo per garantire che la test & set sia atomica in un multiprocessore? Il bus é gestito da un controllore (arbiter) che deve selezionare per ogni operazione il processore che puó usare il bus. Un modo per risolver il problema é bloccare in controllore per due operazioni per garantire l'esecuzione atomica e senza interruzioni.

Domande presentazioni

1. [Gruppo 1] Quali sono le differenze tra implementare una smart home solo con la board (Raspberry) o con un server e quindi connessa ad internet. Offline vs Online. In veritá la domanda era "é meglio un sistema distribuito o un server centrale?" Possibile prossima domanda: Preferisci la soluzione centralizzata o la soluzione distribuita da un punto di vista del fault-tolerance? Puoi rimarcare i principali difetti di un'architettura distribuita da un punto di vista della progettazione? Si puó gestire in single point of failure? Si, tramite repliche.

2. [Gruppo 2] Come determinare la prioritá dei dispositivi per un priority arbiter? Potete suggerire un modo per risolvere la *starvation*? La prioritá va cambiata ad ogni *miss*.

3. [Gruppo 3] Perché la velocitá del clock conta? Perché CoreMark conta? Potete dirmi la situazione del consumo energetico della memoria? Dipende dall'esecuzione del programma o no? E' stabile o aumenta con l'aumentare della frequenza?

4. [Gruppo 4] Parlare del READ/WRITE protocol della memoria. Ci sono dei problemi con il timing? Potete descrivere la soluzione nella quale il bus é pipelined? Veramente il bus pipelined é organizzato con le fasi sovrapposte. Puoi iniziare una read durante la fase di trasferimento di una write. Un'altra soluzione é lo *splitted bus*.

5. [Gruppo 5] *[presentazione sui droni] il drone lo costruisco per fare delle elaborazioni (esempio ricostruire la zona con tutte le altezze senza andare a fare misure dirette). Immaginiamo di disporre di un'applicazione in grado di costruire tutto il profilo di un palazzo, ha piú senso, dal punto di vista del consumo, utilizzare l'energia per spedire il filmato a terra oppure creare automaticamente le immagini sul drone e inviare le foto, parzialmente elaborate, a terra. (tipo watt per byte o watt per frame).

6. [Gruppo 7] Detection e prevenzione degli attacchi.

7. [Gruppo 8] Java processors hardware implementation, energy vs performance

8. [Gruppo 9] Perché l'energia non ha un andamento costante dopo 1.6 GHZ.

9. [Gruppo 10] Come possiamo ottimizzare la power consumption modificando i parametri della cache?

10. [Gruppo 11] Perché l'Intel funziona meglio dell'AMD? Sapete come si ottimizza un'applicazione in base all'architettura?

Domande trovate su cartella anni passati

Cache:
• Why?
• principles of locality
• Cache organization (direct, set, fully associative)
• hit time and miss penalty
• type of misses (3C)
• Replacement policy
• istruction and data cache
• write policy, cache levels
• inclusive and exclusive cache
• victim cache

GPU:
• differences with CPU
• heterogenous architectures,
• memory latency in this archtectures
• SIMT
• CUDA (thread and memory organization,structure of code)

Multiprocessor:
• Multiprocessor vs Multicomputer,
• Consistency vs Coherence,
• definitions of consistency
• UMA Architecture,
• Cache Coherence with Snooping Cache: write-throught, write-back
• (MESI, MOESI, MSI protocols)
• Crossbar interconnection
• omega network
• NUMA architecture
• Cache coherence with directory-based protocol
• Quali sono le possibili miss? Come si possono misurare le percentuali dei vari tipi di miss.

• Differenza tra throughput, latenza e banda. Cos'è più facile ottenere adesso?
[Risposta] Throughput é la quantitá di dati che vengono inviati in un certo intervallo di tempo. Banda é la quantitá massima di dati che possono essere inviati. Latenza é il tempo che trascorre tra l'invio dei dati e l'arrivo. É piú facile ottenere la banda, perché non vuol dire ottenere prestazioni migliori, ma solo migliorarle teoricamente.

• Differenza tra concorrenza e parallelismo.
• Processori superscalari. Cosa sono, cosa significa esecuzione out of order
• Dependability e scalability
• Che cos'è la scalabilità e che cos'è un collo di bottiglia? Che legame c'è tra i due.
• Quali fattori definiscono una word
• Arm Cortex A8, struttura della memoria
• Livelli di parallelismo, architetturale e applicazione
• Come deve struttura il codice il programmatore per sfruttare la parallelizzazione

dell’architettura?
• Struttura cache a 2 vie (con disegno). Least Recently Used e Least Frequently Used

Domande esami anni precedenti:

1. Abbiamo visto come organizzare il lavoro in pipeline. Dal punto di vista delle performance quali sono i fattori che possono incidere sul througput delle ALU?
   [Risposta] Stalli causati da attesa di operandi, la soluzione puó essere usare delle "scorciatoie" a livello hardware. Possiamo ridefinire la fase di decodifica, oltre alla decodifica ho l'accoppiamento degli operandi di input. Normalmente l'input é o nell'indirizzo o in un registro, adesso possiamo riceverlo anche da una ALU precedente.
   Se non usassi questa cosa dovrei inserire uno stallo e aspettare il write-back.
   Dopo ha voluto anche sapere il register renaming. Come ci aiuta la cache in questa situazione? Recupero i dati piú velocemente. La presenza delle cache introduce vari problemi, ma uno di questi é particolarmente interessante e influenza lo sviluppo del sistema operativo in particolare il multicore, quale? Cache coherence, perché la studiamo? Perché é un bottleneck? Perché é molto piú utilizzata e perché il problema fondamentale della cache sono le write. (slide su read e write cache coherence)

2. Perché su un multiprocessore abbiamo introdotto piú linee di pipeline? E perché non troviamo processori con 15 pipeline?
   [Risposta] Ovviamente aumentiamo linee di pipeline perché aumenta throughput, il problema di averne troppi é che devo accelerare programmi sequenziali per quello che riguardare la CPU. Dovrei aumentare il numero di istruzioni parallele in programmi sequenziali, mentre la maggior parte dei programmi che abbiamo sono sequenziali.

3. Modelli di multithreading per i multiprocessori.
   Voleva sapere *coarse-grained* MT, *fine-grained* MT, *SMT*. Soprattutto quali sono i vantaggi dell'avere multithreading.

4. Come si possono ridurre le conflict miss in cache? Poi é passato a chiedere cosa sono le conflict misses.
   [Risposta] Aumentare l'associativitá.

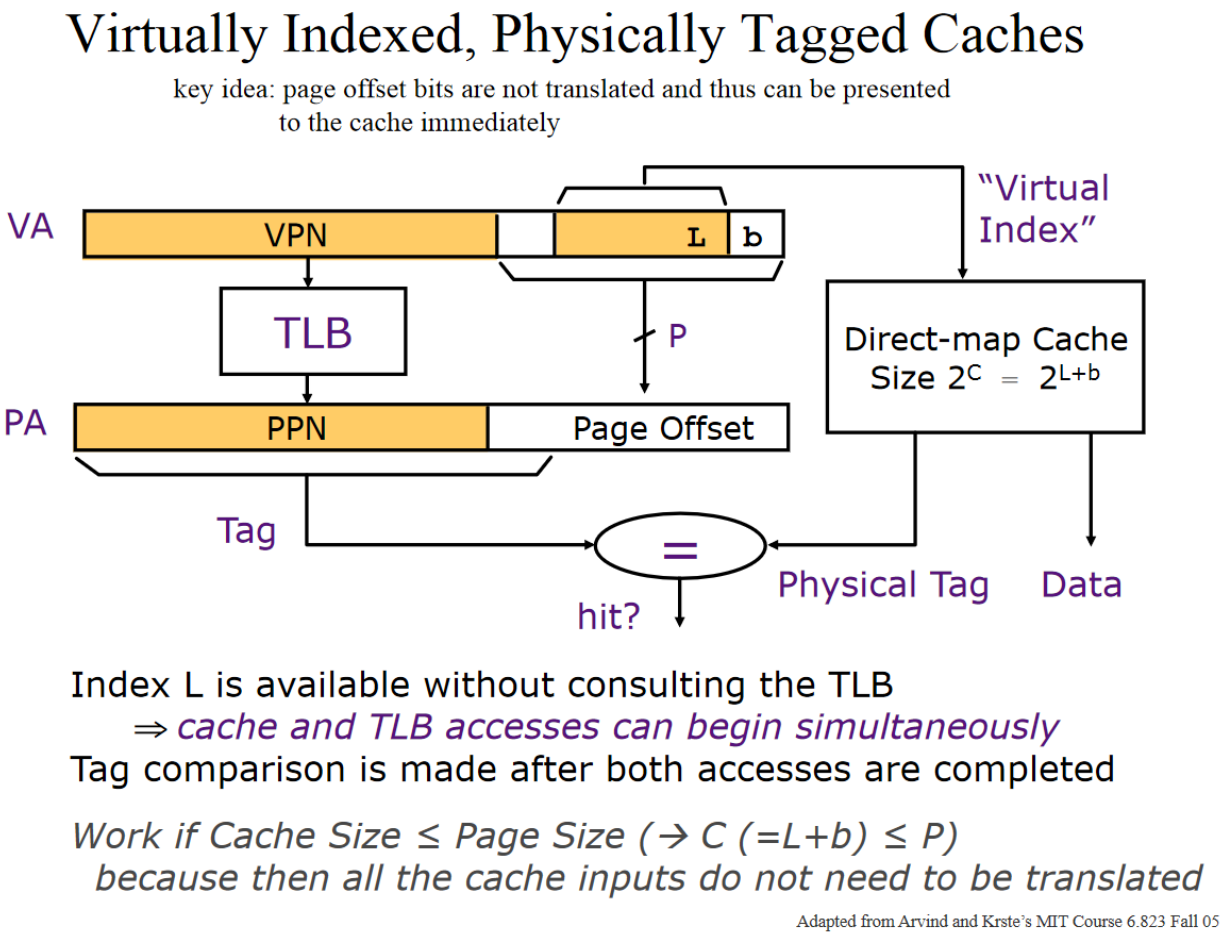
5. Solitamente abbiamo il primo livello di cache piccolo, i successivi grandi, perché?
   [Risposta] Perché vogliamo minimizzare l'access time del primo livello di cache, non ci importa avere un grande numero di hit. Invece i livelli successivi possono avere access time piú alti ma devono minimizzare il numero di miss.

6. Come funziona la cache di primo livello in combinazione con la traduzione degli indirizzi? Il primo livello di cache usa indirizzi fisici o virtuali?
   [Risposta] Ovviamente uso indirizzi fisici, perché per gli indirizzi virtuali ho un problema. Infatti piú di un indirizzo virtuale puó riferire lo stesso indirizzo fisico e per questo motivo vado ad usare direttamente i fisici ed evitare di dover aggiungere hardware per fare controlli. Questo porta ad un problema, infatti ora devo prima tradurre l'indirizzo da virtuale a fisico e poi usarlo nella cache. Come possiamo ottimizzare il tempo di accesso?

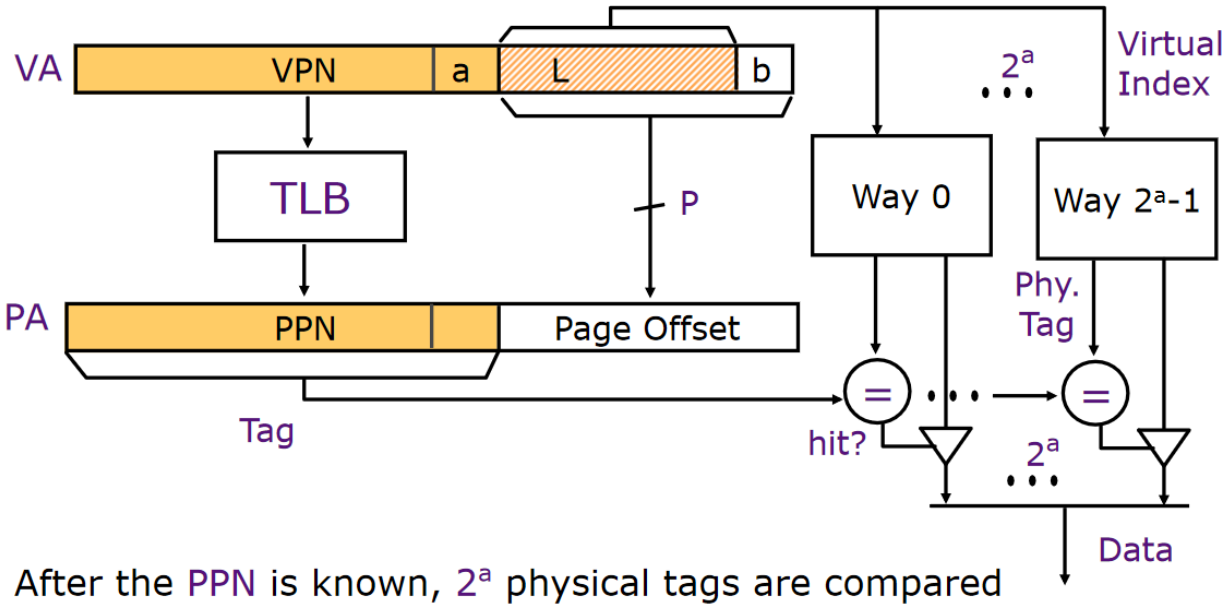
- usare una cache direttamente mappata

- usare un TLB per la traduzione. (Cosa é un TLB) Piccola cache velocissima (full-associative) che contiene le ultime traduzioni degli indirizzi virtuali.
  Come si puó ottimizzare questa traduzione? Sono operazioni eseguite in modo sequenziale, come faccio ad eseguirle insieme?
  [Risposta] Way-prediction non é giusta come risposta, perché non si usa per diminuire il tempo di accesso ma per diminuire l'energia richiesta.

Risposta giusta.



Virtually-Indexed Physically-Tagged Caches:
Using Associativity for Fun and Profit



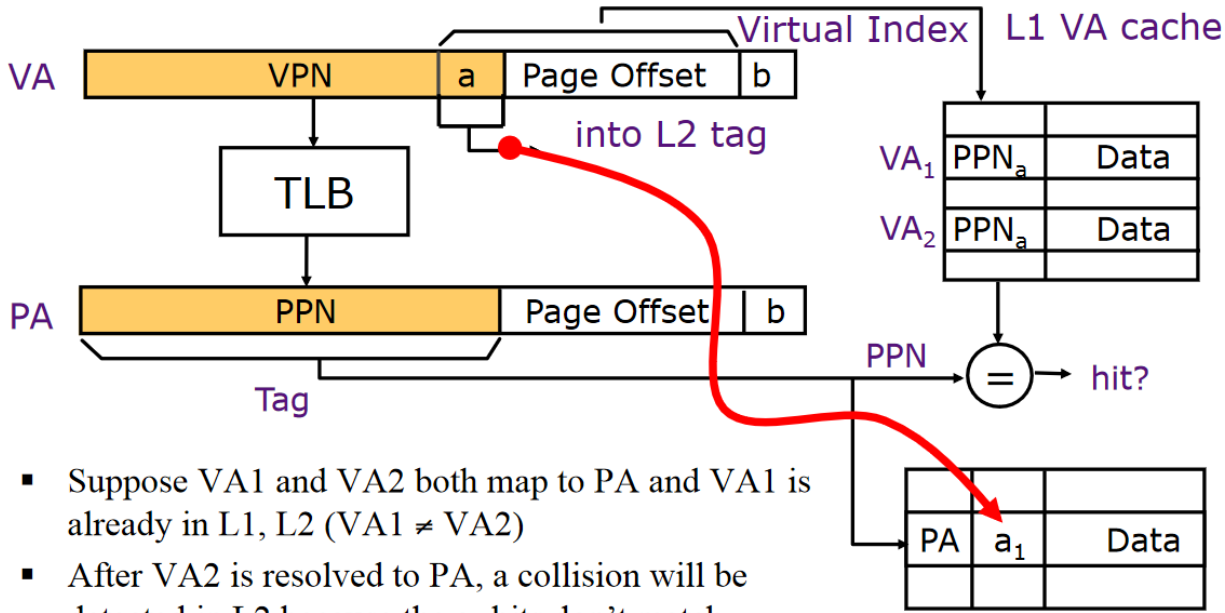
After the PPN is known, 2^a physical tags are compared

Increasing the associativity of the cache reduces the number of address bits needed to index into the cache -

Work if: Cache Size / 2^a ≤ Page Size *(→ C ≤ P + A)*

Anti-Aliasing Using Inclusive Direct Mapped L2: *MIPS R10000-style*

Once again, ensure the invariant that only one copy of physical address is in virtually-addressed L1 cache at any one time. The physically-addressed L2, which includes contents of L1, contains the missing virtual address bits that identify the location of the item in the L1.



- Suppose VA1 and VA2 both map to PA and VA1 is already in L1, L2 (VA1 ≠ VA2)
- After VA2 is resolved to PA, a collision will be detected in L2 because the a_1 bits don't match.
- VA1 will be purged from L1 and L2, and VA2 will be loaded ⇒ *no aliasing !*

Direct-Mapped PA L2

(could be associative too, just need to check more entries)

Adapted from Arvind and Krste's MIT Course 6.823 Fall 05

7. Perché si usano due cache di livello 1, una per le instruzioni e una per i dati?
   [Risposta] Perché nella pipeline in questo modo posso fare il fetch di un'istruzione in contemporanea alla lettura di un dato.

8. Dal punto di vista del consumo energetico é piú importante l'energia o la potenza dissipata? Che importanza ha controllare il picco massimo di potenza di un processore?
   L'energia dissipata é quella che riscalda il nostro sistema. Il picco di potenza da fastidio perché P=VI, se accendo contemporaneamente 4 vie devo fornire corrente e in questo momento ho un picco di potenza dato dalla corrente di 4 vie accese. Si crea un *hotspot*, picco di potenza localizzato e breve che non si puó controllare con la ventola o un altro di sistema di raffreddamento. I sistemi embedded quindi devono mantenere bassa sia la potenza che l'energia.

9. Quali sono le regole piú importanti per individuare il parallelismo all'interno di CUDA.
   [Risposta] Parallelismo piú importante che ho con CUDA é il *data-level* ma anche *multithread*.
   regola 1: *overhead della gestione del parallelismo (distribuzione del carico sui vari thread) é marginale rispetto alla elaborazione del dato* [cosa é l'overhead nel caso della cpu? cambi di contesto, comunicazione tra thread, scrittura]
   regola 2: *massimizzare l'uso dei processori*

10. Andamento del tempo di esecuzione totale al variare della dimensione del blocco con capacitá della cache.
    [Risposta] Voleva sapere localitá temporale e spaziale.

11. Quale é il metodo piú veloce che possiamo avere per capire le prestazioni di un sistema?
    [Risposta] Tool di debug per trovare il bottleneck, trovo la componente piú utilizzata, faccio un benchmark su queste componenti.

12. A partire da una organizzazione funzionale di una CPU siamo arrivati ad una organizzazione a pipeline, perché la pipeline mi permette di aumentare le prestazioni e come faccio a vedere che che effettivamente aumentano?
    [Risposta] Aumento utilizzazione dividendo le fasi di utilizzo della CPU e potendo iniziare ad eseguire un'istruzione prima che l'ultima sia completamente finita. Il parametro che posso guardare é il numero di operazioni committate al secondo (per esempio). Devo mantenere cariche quante piú ALU occupate possibile.

13. Quali sono le soluzioni che si possono utilizzare per un processore vettoriale e quali sono i punti di sovrapposizione tra un processore vettoriale e una GPU? Uso della cache in questo tipo di architettura?

14. Quali sono le cause che possono rallentare una pipeline rispetto all'altra?
    [Risposta] Voleva sapere gli hazard.

15. Nel caso di numero di thread fisso all'aumentare della dimensione del vettore diventa sempre piú conveniente avere piú thread, perché?
    [Risposta] Quando dobbiamo parallelizzare conviene sempre avere una grande dimensione dei dati. Perché posso sfruttare molti piú thread per svolgere le computazioni, con vettori piú

piccoli che possono essere mantenuti anche in cache potrebbe non essere conveniente avere piú thread a causa dell'overhead fisso dato dalla creazione e mantenimento dei thread.

16. Differenza tra protocollo di coerenza *invalidate* e *update*. In che relazione sono coerenza e consistenza?
    [Risposta] La prima é la differenza tra write-back e write-invalidate. La seconda: *coherence* quando (temporalmente) applicare tecniche di consistenza, consistenza come salvare dati in modo consistente. Quale é l'ultimo momento utile per fare l'operazione? Quale é il vantaggio di non fare subito le operazioni e salvarle in un buffer? Nella lock e nella unlock, che anticipano una serie di operazioni sui dati condivisi. Salvo in un buffer per non far interrompere il lavoro al processore.

17. Come possiamo implementare a livello hardware la mutua esclusione?
    [Risposta] La lock ha un parametro, la variabile che vado a gestire in mutua esclusione e che impedisce agli altri di entrare in quella sezione critica. Come faccio ad impedire agli altri di entrare ma far entrare a me che posso? Prima leggo lo stato (lettura) della variabile, vedo lo stato, setto occupato (scrittura) e entro nella sezione critica. Ma un altro processore potrebbe infilarsi tra la lettura e la scrittura, quindi ho bisogno di un meccanismo piú a basso livello. Questo meccanismo é la TEST AND SET o disabilito interruzioni.

18. Quali possono essere i meccanismi implementati nelle pipeline per gestire i thread? Perché creare thread hardware e come gestiscono il parallelismo? Quale é il parametro che ci fa capire che stiamo sfruttando bene l'architettura?
    [Risposta] Posso sfruttare i thread per cambiare contesto durante uno stallo ed evitare attese. Il parametro é il throughput delle unitá di esecuzione (ALU, FPU).
    Come possiamo organizzare i flussi di thread?
    Fine, Coarse e SMT.
    La parte di renaming nasconde due attivitá. Allocazione registro fisico e deallocazione.
    Quando vengono fatte queste due operazioni e quali sono i moduli che si occupano di fare queste due attivitá.
    In quale condizione viene allocato un nuovo registro fisico?
    OGNI REGISTRO DESTINAZIONE alloca un nuovo registro fisico.
    In che condizione viene deallocato? Durante la fase di commit.

19. Definire le conflict miss in the cache. Perché aumentare la dimensione della cache potrebbe essere una soluzione? Perché aumentare l'associativitá é una soluzione?

20. Ruolo del protocollo di consistenza e coerenza. Quali sono i loro obiettivi? Descrivere un protocollo di coerenza in particolare, ad esempio MSI. Quale problema tentano di risolvere.
    [Risposta] Il modello di consistenza stabilisce il modo in cui le operazioni di LOAD e STORE sono viste dalla memoria.

21. Cosa vuol dire che un processore é multi-thread?