ΗΜΥ 312
ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
Εαρινό Εξάμηνο 2006

ΔΙΑΛΕΞΕΙΣ 04 - 05: ΜΝΗΜΗ CACHE

ΘΕΟΧΑΡΗΣ ΘΕΟΧΑΡΙΔΗΣ (charisth@ucy.ac.cy)

http://www.eng.ucy.ac.cy/theocharides/Courses/ECE312

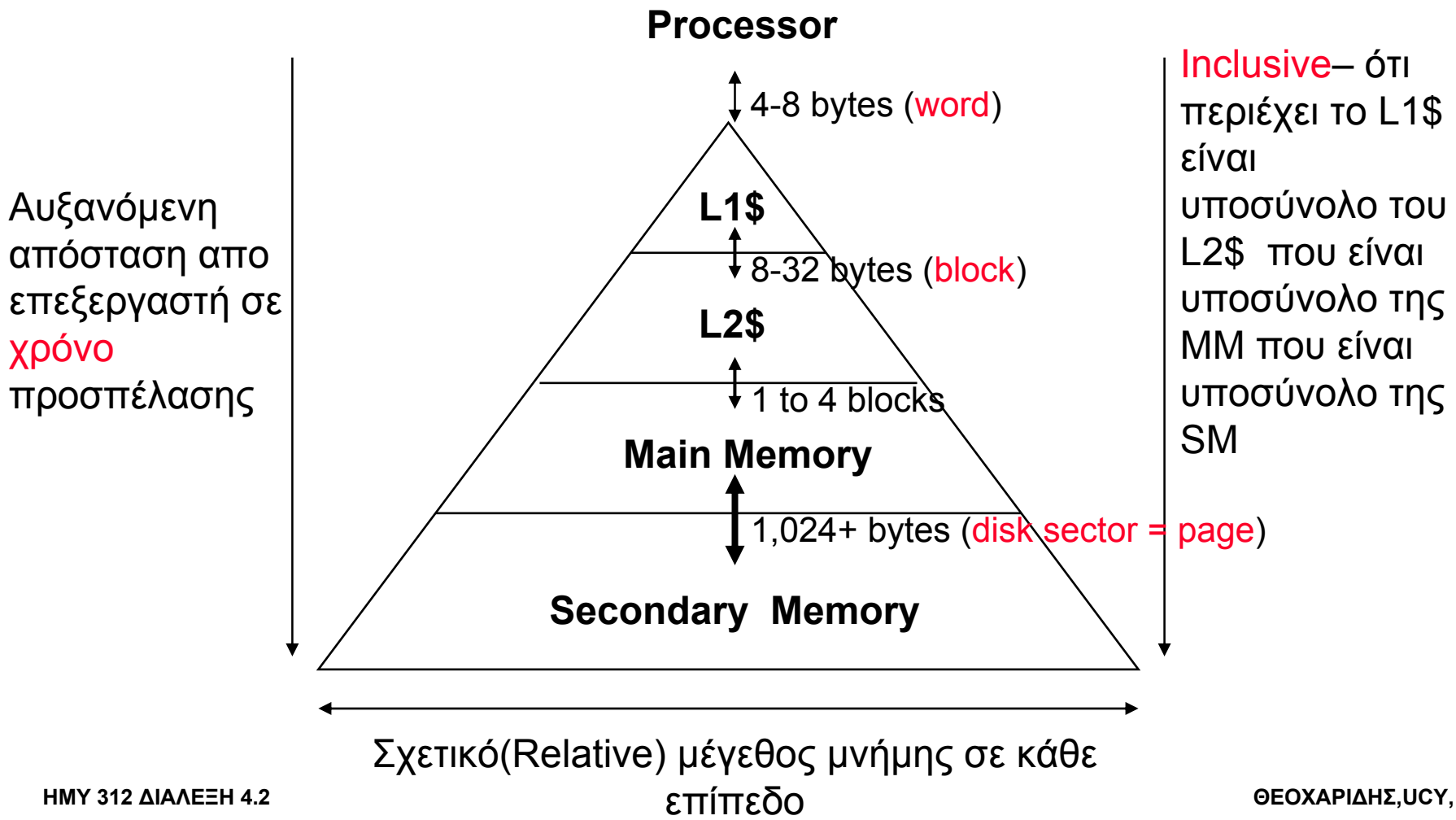
Patterson / Hennesy – Chapter 5

[Προσαρμογή από *Computer Architecture*,

Patterson & Hennessy, © 2005, UCB]

(ΕΠ) ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΙΕΡΑΡΧΙΑΣ ΜΝΗΜΗΣ

Χρησημοποίηση πλεονεκτίματος τοπικότητας(locality). Να παρουσιάσουμε στον χρήστη όση μνήμη μπορούμε στην πιο φθηνή τεχνολογία, στην ταχύτητα που προσφέρει η πιο γρήγορη τεχνολογία



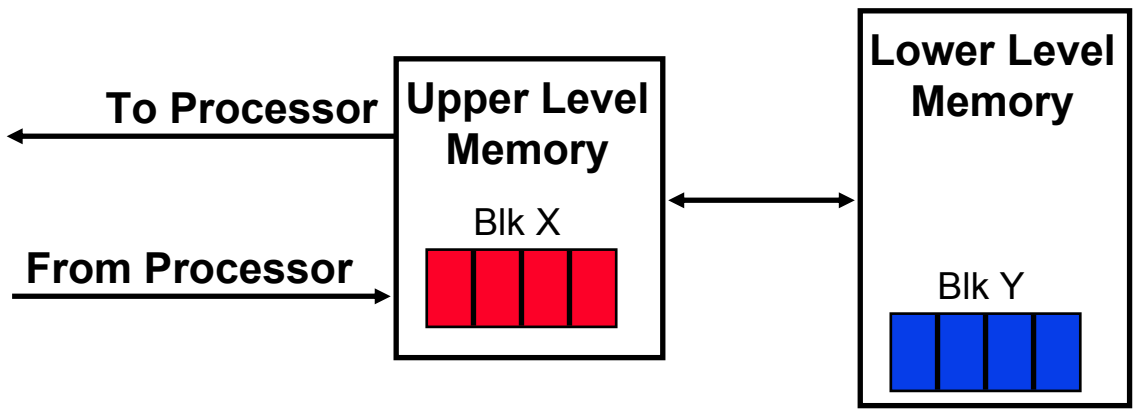
ΙΕΡΑΡΧΙΑ ΜΝΗΜΗΣ – ΠΩΣ

- Temporal Locality (Locality in Time) Τοπικότητα στον χρόνο:

  ⇒ Κρατάμε τα πιο πρόσφατα προσπελασμένα δεδομένα όσο πιο κοντά στον επεξεργαστή. (most recently accessed)

- Spatial Locality (Locality in Space)Τοπικότητα στον χώρο:

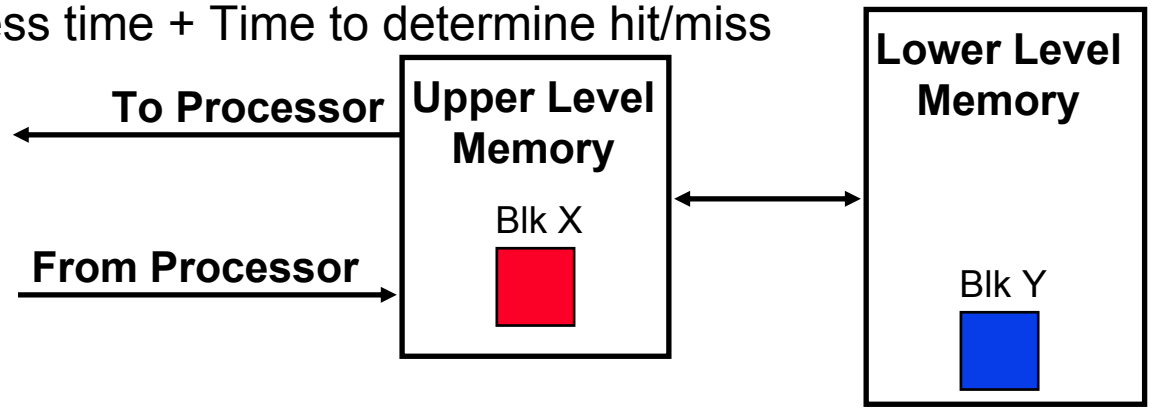
  ⇒ Μεταφέρουμε blocks που αποτελούνται απο συνεχή δεδομένα στα πιο ψηλά επίπεδα (contiguous words).



ΙΕΡΑΡΧΙΑ ΜΝΗΜΗΣ - ΟΡΟΛΟΓΙΑ

- Hit: Τα δεδομένα είναι σε ένα block στο πιο ψηλό επίπεδο (Blk X)
  - Hit Rate: το κλάσμα δεδομένων που βρίσκονται στο πιο ψηλό επίπεδο
  - Hit Time: Η ώρα προσπέλασης του ψηλού επιπέδου ως

    RAM access time + Time to determine hit/miss



- Miss: Τα δεδομένα δεν βρίσκονται στο πιο ψηλό επίπεδο άρα χρειάζετε επαναφορά απο πιο χαμηλό επίπεδο. (Blk Y)
  - Miss Rate = 1 - (Hit Rate)
  - Miss Penalty: Time to replace a block in the upper level
    + Time to deliver the block the processor
  - Hit Time << Miss Penalty

ΠΩΣ ΕΛΕΓΧΕΤΑΙ Η ΙΕΡΑΡΧΙΑ?

- registers ↔ memory
  - Από μεταφραστή (compiler+ programmer?)
- cache ↔ main memory
  - Από το σύστημα ελέγχου CACHE (cache controller hardware)
- main memory ↔ disks
  - Από το λειτουργικό σύστημα (operating system (virtual memory))
  - virtual to physical address mapping assisted by the hardware (TLB)
  - by the programmer (files)

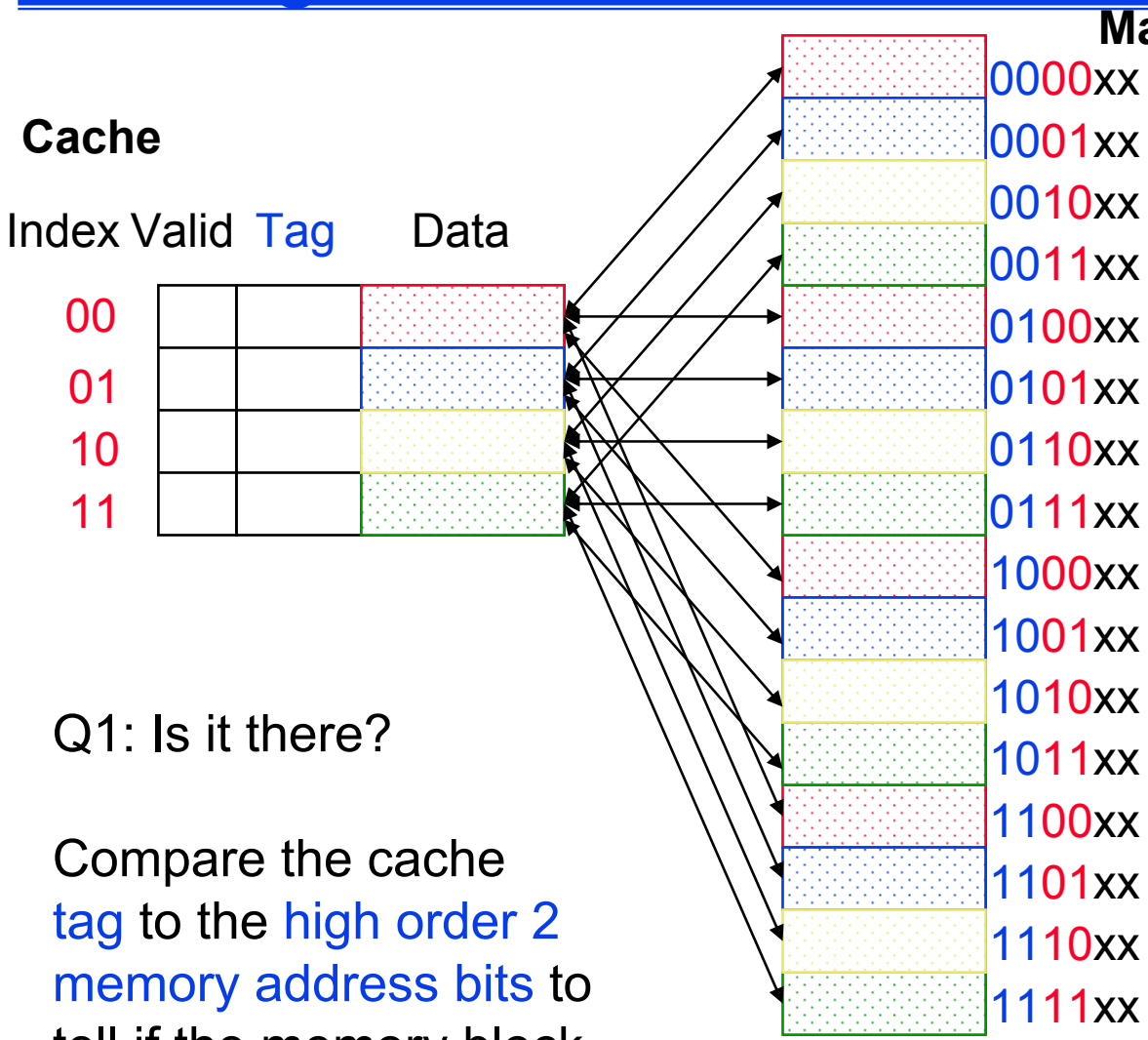
Cache

- Δύο Ερωτήσεις (in hardware):
  - Q1: Πως ξέρουμε ότι τα δεδομένα είναι στο cache?
  - Q2: Αν είναι εκεί, πως τα βρίσκουμε?

- Απευθείας (Direct mapped)
  - Για κάθε δεδομένο που είναι στο χαμηλότερο επίπεδο μνήμης, υπάρχει μια τοποθεσία στο Cache που βρίσκετε. Άρα πολλά δεδομένα στο χαμηλότερο επίπεδο μοιράζονται θέσεις στο ψηλότερο επίπεδο.
    - For each item of data at the lower level, there is exactly one location in the cache where it might be - so lots of items at the lower level must share locations in the upper level
  - Αντιστοιχεία Διεύθηνσης (Address mapping):

    (block address) modulo (# of blocks in the cache)
  - Ας υποθέσουμε πρώτα block sizes of one word

Caching: ΕΝΑ ΑΠΛΟ ΠΑΡΑΔΕΙΓΜΑ



Two low order bits define the byte in the word (32b words)

Q1: Is it there?

Compare the cache tag to the high order 2 memory address bits to tell if the memory block is in the cache

Q2: How do we find it?

Use next 2 low order memory address bits – the index – to determine which cache block (i.e., modulo the number of blocks in the cache)

(block address) modulo (# of blocks in the cache)

Direct Mapped Cache (Απ'ευθείας)

- Consider the main memory word reference string

Start with an empty cache - all blocks initially marked as not valid

0 1 2 3 4 3 4 15

0 miss

00	Mem(0)

1 miss

00	Mem(0)
00	Mem(1)

2 miss

00	Mem(0)
00	Mem(1)
00	Mem(2)

3 miss

00	Mem(0)
00	Mem(1)
00	Mem(2)
00	Mem(3)

4 miss

01 → 4

~~00~~	Mem(~~0~~)
00	Mem(1)
00	Mem(2)
00	Mem(3)

3 hit

01	Mem(4)
00	Mem(1)
00	Mem(2)
00	Mem(3)

4 hit

01	Mem(4)
00	Mem(1)
00	Mem(2)
00	Mem(3)

15 miss

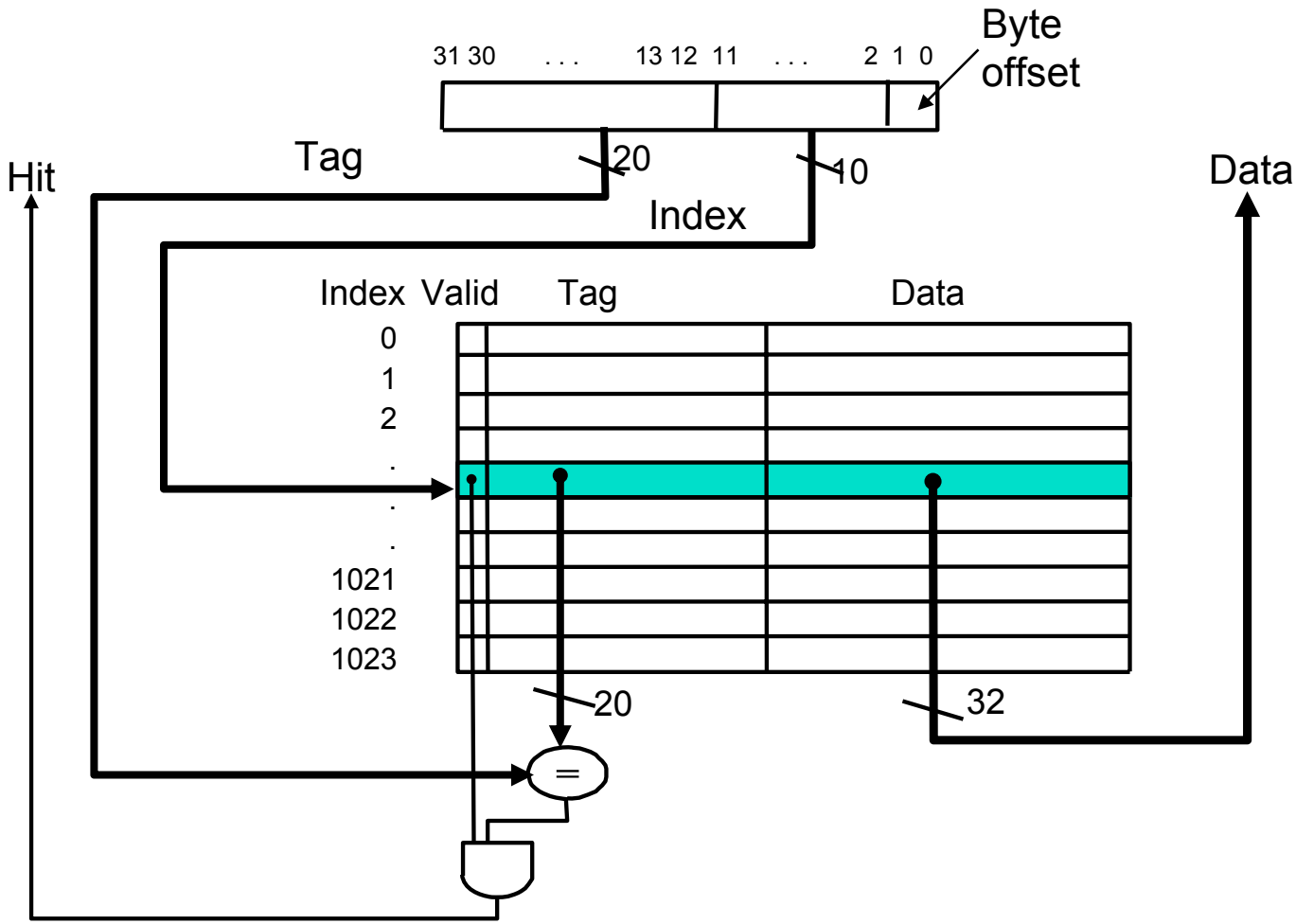
11 → 15

01	Mem(4)
00	Mem(1)
00	Mem(2)
~~00~~	Mem(~~3~~)

- 8 requests, 6 misses

MIPS Direct Mapped Cache Example

- One word/block, cache size = 1K words



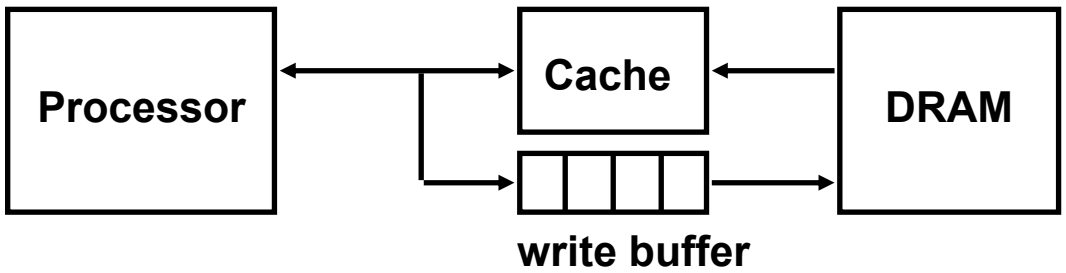
What kind of locality are we taking advantage of?

ΠΩΣ ΧΕΙΡΙΖΟΜΑΣΤΕ Cache Hits

- Read hits (I$ and D$)
  - Αυτά θέλουμε!!!

- Write hits (D$ only)
  - Είτε αφήνουμε το cache και την μνήμη να είναι αποσυντονισμένα (inconsistent)
    - γράφουμε τα δεδομένα στο cache block *(write-back the cache contents to the next level in the memory hierarchy when that cache block is "evicted")*
    - Χρειαζόμαστε το λεγόμενο "λερωμένο" (dirty) bit για κάθε data cache block ώστε να γνωρίζει πια δεδομένα θα γράψει πίσω στη μνήμη όταν το block "διωχτεί"
  - Είτε επιβάλουμε στην μνήμη και στο Cache να είναι συνεπής (consistent)
    - γράφουμε πάντα στο cache block και στο επόμενο επίπεδο στην μνήμη (write-through) και δεν χρειαζόμαστε dirty bit.
    - το γράψημο δουλεύει στην ταχύτητα του επόμενου επιπέδου μνήμης (πολύ πιο σιγά δηλ.) ή μπορεί να χρησημοποιηθεί ένα write buffer, ώστε να γράφουμε μόνο όταν το write buffer είναι γεμάτο (μειώνοντας την συχνότητα που πληρώνουμε το τίμημα να γράψουμε στην πιο αργή μνήμη.

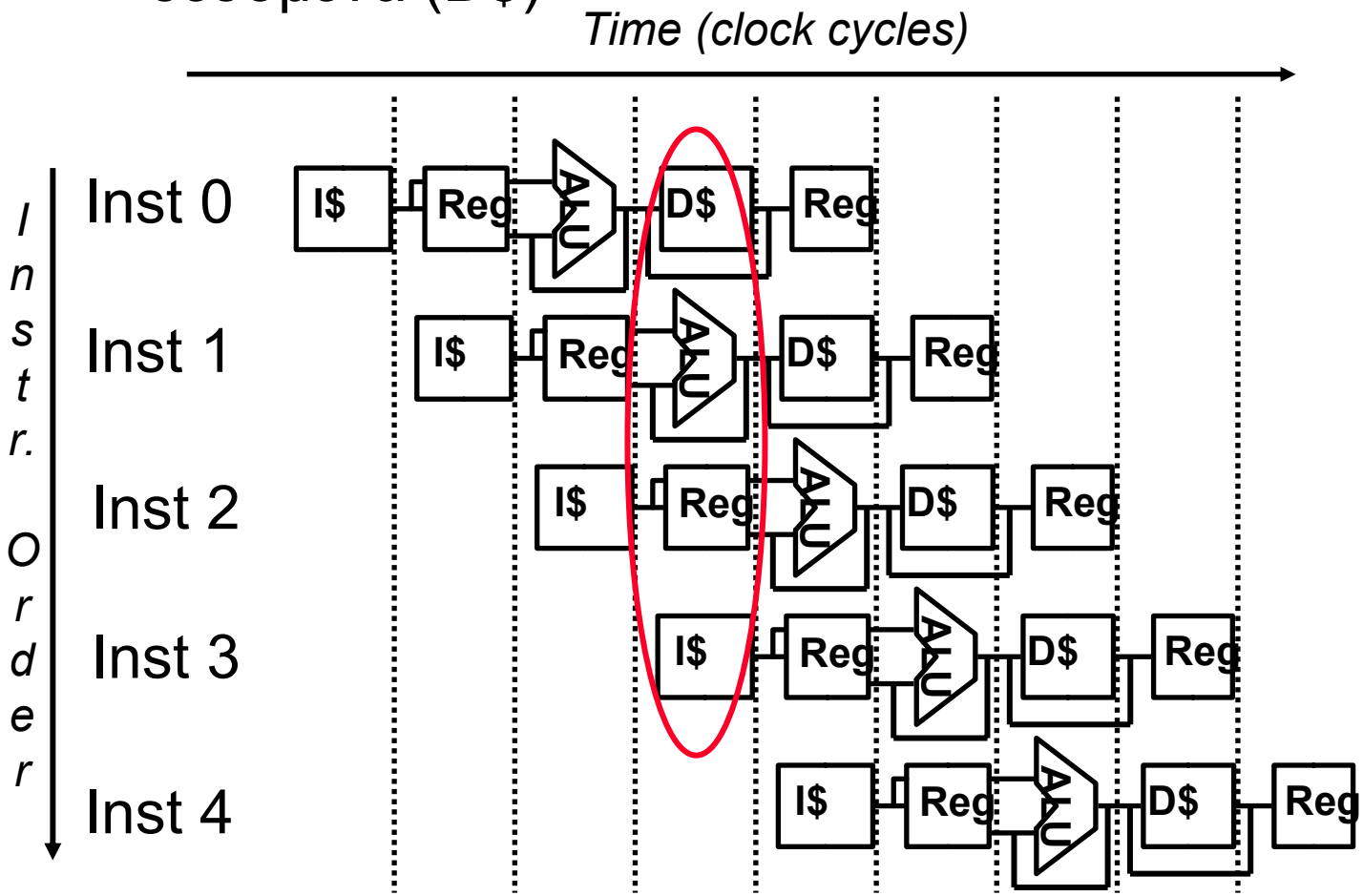
Write Buffer για Write-Through Caching



- Βάζουμε ένα write buffer μεταξύ cache και κύριας μνήμης
  - Επεξεργαστής: γράφει δεδομένα στο cache και στο write buffer
  - Memory Controller: γράφει δεδομένα από write buffer στο memory
- Το write buffer είναι απλά ένα FIFO (First-In, First-Out)
  - Typical number of entries: 4
  - Works fine if store frequency (w.r.t. time) << 1 / DRAM write cycle
- Memory system designer's nightmare
  - When the store frequency (w.r.t. time) → 1 / DRAM write cycle leading to write buffer saturation
    - One solution is to use a write-back cache; another is to use an L2 cache (next lecture)

(Επ): ΓΙΑΤΙ ΔΙΑΣΩΛΗΝΟΝΟΥΜΕ? ΓΙΑ ΠΑΡΑΓΩΓΗ!

- Για αποφυγή ενός structural hazard χρειαζόμαστε δυο caches on-chip: ένα για εντολές (I$) και ένα για δεδομένα (D$)



To keep the pipeline running at its maximum rate both I$ and D$ need to satisfy a request from the datapath every cycle.

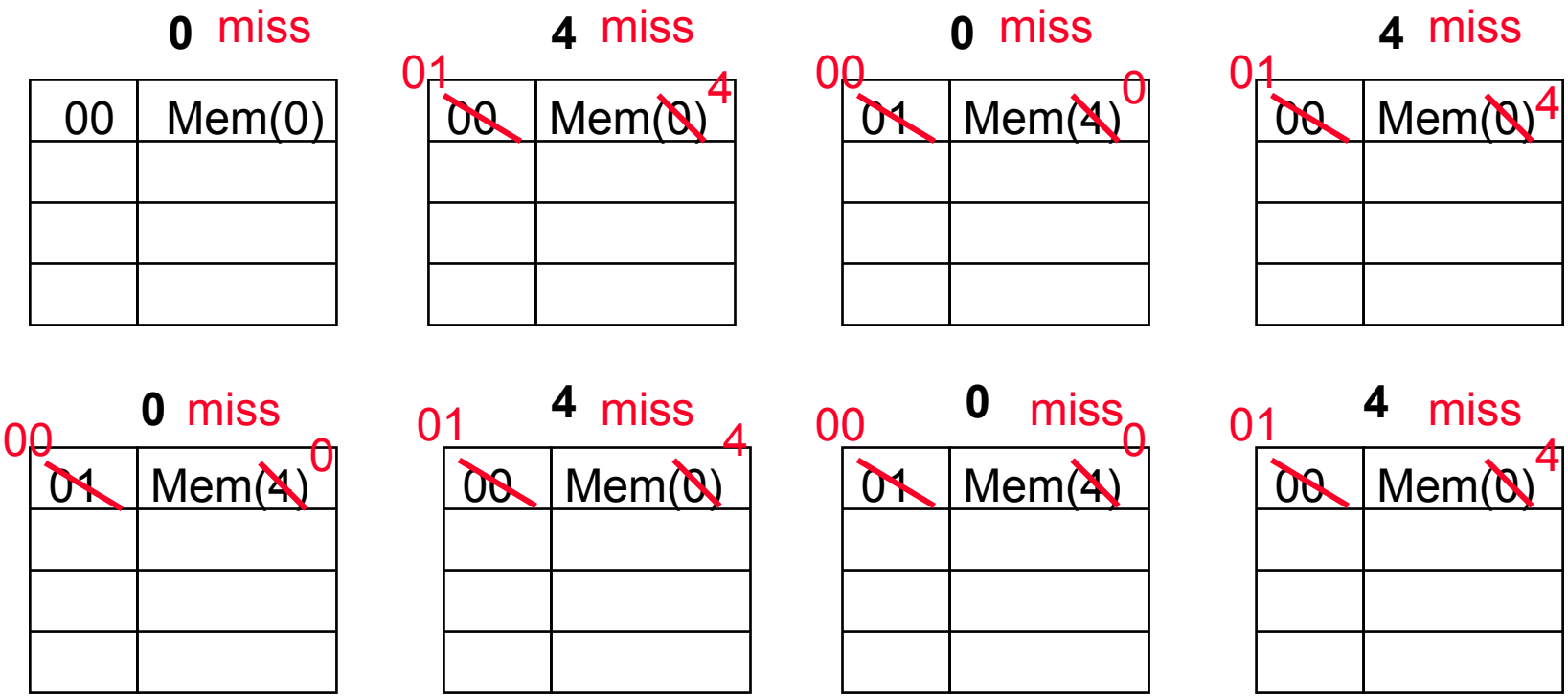
What happens when they can't do that?

Ακόμα ένα Reference String Mapping

- Consider the main memory word reference string

Start with an empty cache - all blocks initially marked as not valid

0 4 0 4 0 4 0 4



- 8 requests, 8 misses

- Ping pong effect due to conflict misses - two memory locations that map into the same cache block

Πηγές που συνδράμουν σε Cache Misses

- Compulsory /Αποδεσμευτικά (‘οταν έχει αλλαγή διεργασίας (process migration),κατά τις πρώτες αναφορές(first reference):
  - Την πρώτη φορά που ζητούμε ένα block (“cold” fact of life, not a whole lot you can do about it)
  - μπροστά σε εκατομμύρια εντολές, τα compulsory misses είναι τελείως ασήμαντα
- Conflict /Σύγκρουση (collision):
  - Πολλαπλές διευθήνσεις δεδομένων στην κύρια μνήμη αντιστοιχούν σε διευθήνσεις στην ίδια τοποθεσία στο cache
  - Λύση 1: Αυξάνουμε το μέγεθος του cache block
  - Λύση 2: Αυξάνουμε το associativity ()
- Capacity / Χωρητικότητα:
  - Cache δέν μπορεί να χωρέσει όλα τα δεδομένα ενός προγράμματος
  - Λύση: Αυξάνουμε το μέγεθος του cache

One More! Invalidation Misses

Χειρισμός Cache Misses

- Read misses (I$ και D$)
  - σταματάμε τη διασωλήνωση (stall), φέρνουμε το block από το επόμενο επίπεδο της ιεραρχίας μνήμης, το τοποθετούμε στο cache και στέλλουμε τα απαιτούμενα δεδομένα στον επεξεργαστή, και μετά συνεχίζουμε την διασωλήνωση
- Write misses (D$ μόνο)
  1. σταματάμε τη διασωλήνωση (stall), φέρνουμε το block από το επόμενο επίπεδο της ιεραρχίας μνήμης, το τοποθετούμε στο cache(εξυπακούει ότι μπορεί να χρειαστέι να βγάλουμε block αν είναι write-back cache), γράφουμε τη λέξη από τον επεξεργαστή,και συνεχίζουμε

  ή (συνήθως χρησημοποιήται σε write-back caches)

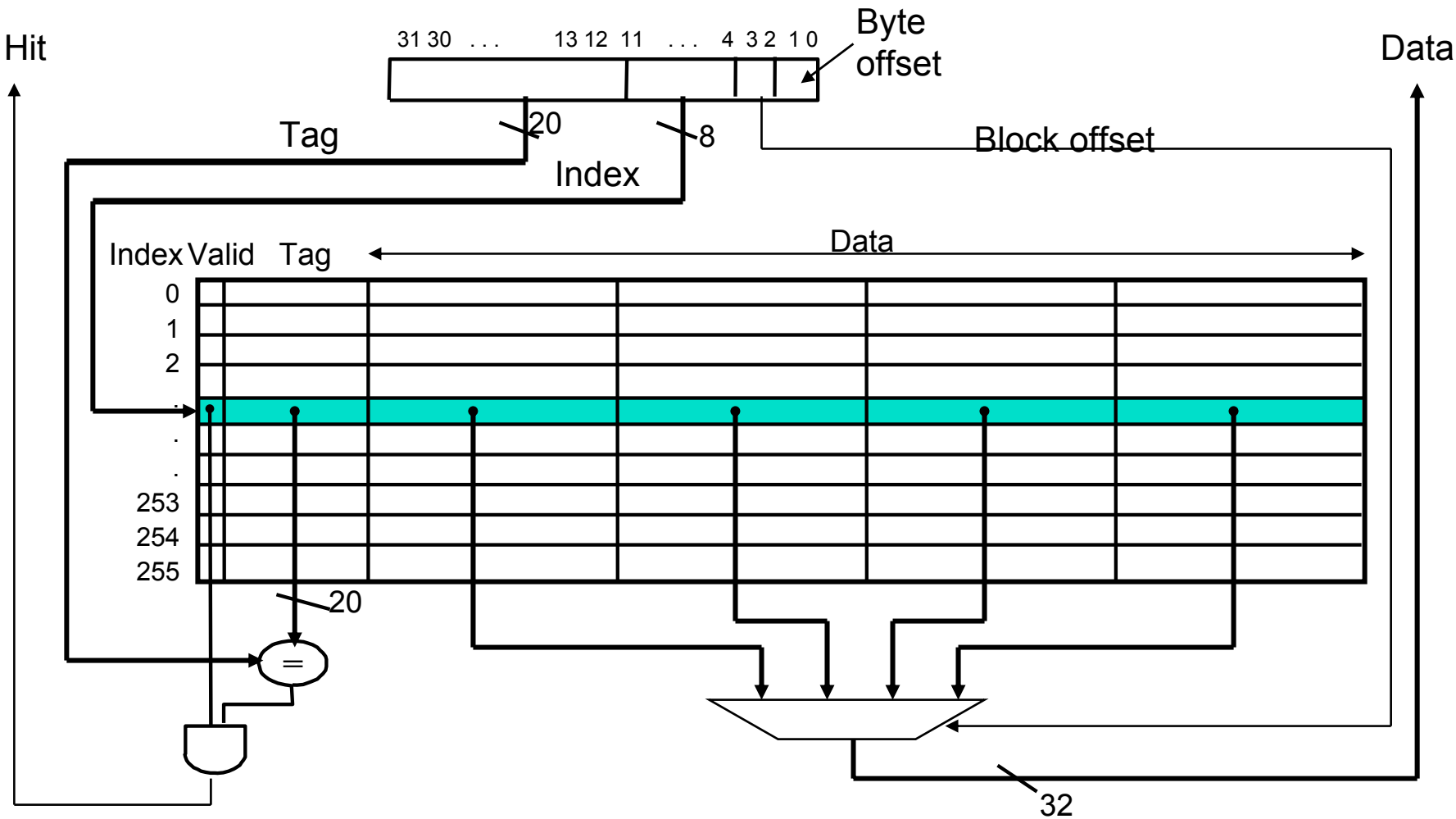
  2. Write allocate (– απλά γράφουμε τη λέξη στο cache, γράφοντας και το tag και τα δεδομένα, δέν χρειάζεται να κοιτάξουμε για cache hit, δεν χρειάζεται να σταματήσουμε)

  ή (συνηθήζεται σε write-through caches με write buffer)

  3. No-write allocate – δέν γράφουμε στο cache, απλά γράφουμε τη λέξη στο write buffer (και τελικά στο επόμενο επίπεδο μνήμης), δεν χρειάζεται να σταματήσουμε μέχρι να γεμίσει το write buffer / πρέπει να ακυρώσουμε (invalidate) το cache block αφού θα περιέχει ασυνέπεια στα δεδομένα (inconsistent)

Multiword Block Direct Mapped Cache

- Four words/block, cache size = 1K words



What kind of locality are we taking advantage of?

Τοπικότητα χώρου (Spatial Locality)

- Let cache block hold more than one word

Start with an empty cache - all blocks initially marked as not valid

0 1 2 3 4 3 4 15

0 miss

00	Mem(1)	Mem(0)

1 hit

00	Mem(1)	Mem(0)

2 miss

00	Mem(1)	Mem(0)
00	Mem(3)	Mem(2)

3 hit

00	Mem(1)	Mem(0)
00	Mem(3)	Mem(2)

4 miss

~~00~~ 01	~~Mem(1)~~ 5	~~Mem(0)~~ 4
00	Mem(3)	Mem(2)

3 hit

01	Mem(5)	Mem(4)
00	Mem(3)	Mem(2)

4 hit

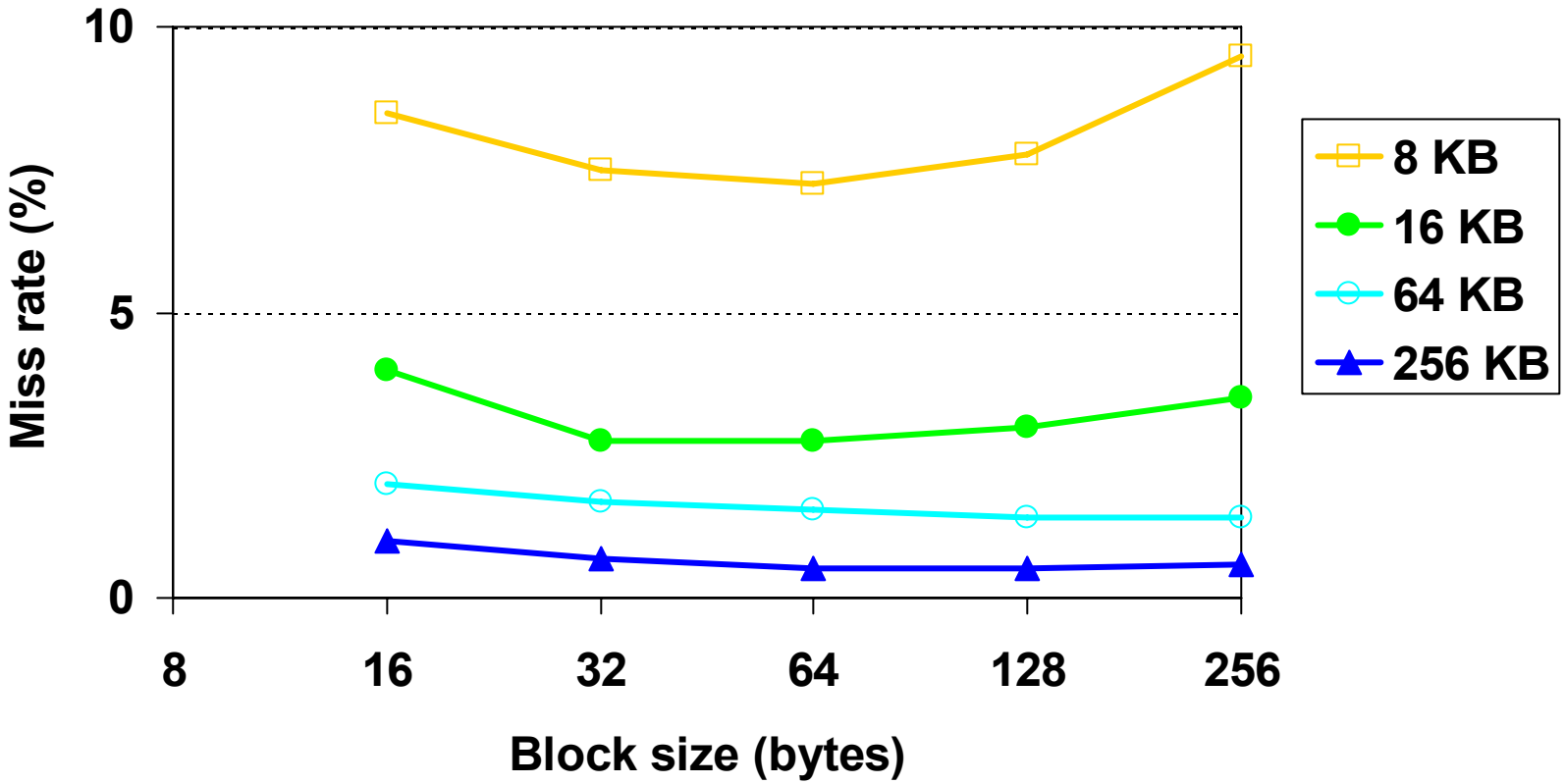
01	Mem(5)	Mem(4)
00	Mem(3)	Mem(2)

15 miss

11 ~~01~~	~~Mem(5)~~	~~Mem(4)~~
~~00~~	~~Mem(3)~~ 15	~~Mem(2)~~ 14

- 8 requests, 4 misses

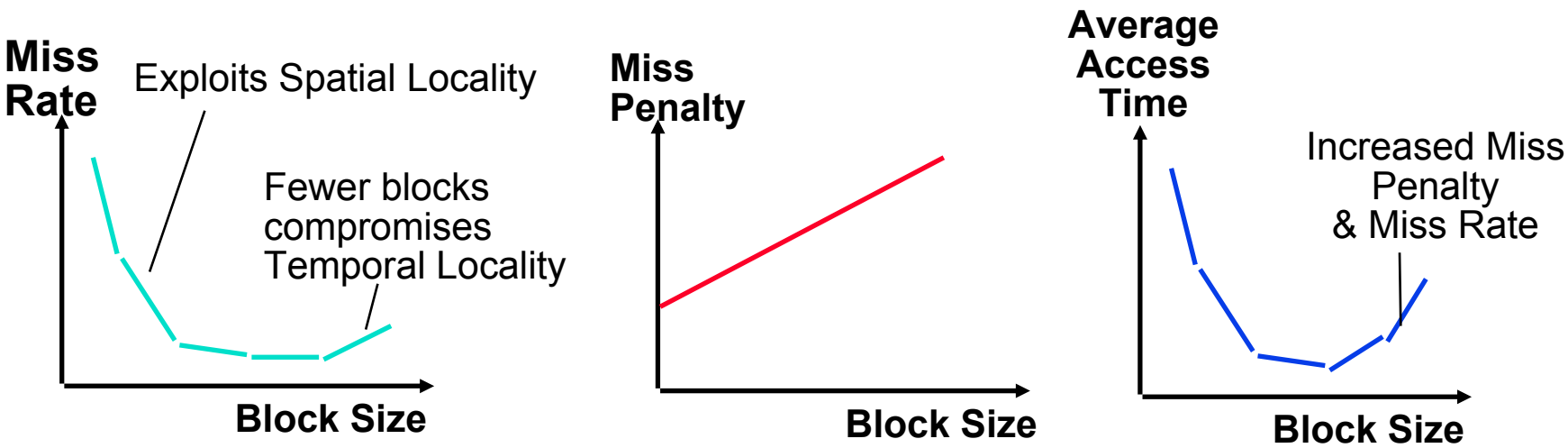
Miss Rate vs Block Size vs Cache Size



- Το Miss rate αυξάνεται άμα το block size γίνεται ένα σημαντικό ποσοστό του cache size επειδή ο αριθμός των blocks που χωράνε σε ένα cache ιδίου μεγέθους είναι πιό μικρά (increasing capacity misses)

Block Size Tradeoff

- Μεγαλύτερα blocks εκμεταλλεύονται το spatial locality αλλά
  - Άμα το block size είναι μεγάλο σε σχέση με το cache size, θα αυξηθεί το miss rate
  - Πιο μεγάλο block size εξυπακούει πιο μεγάλο miss penalty
    - Latency to first word in block + transfer time for remaining words



- In general, Average Memory Access Time

  = Hit Time + Miss Penalty x Miss Rate

Multiword Block Considerations (Πολλαπλές Λέξεις σε ένα Block)

- Read misses (I$ and D$)
  - Ακριβώς όπως ένα single word block – ένα miss επιστρέφει ένα ολόκληρο block από τη μνήμη
  - Το Miss penalty αυξάνεται όσο το block size αυξάνεται
    - Early restart – datapath resumes execution as soon as the requested word of the block is returned
    - Requested word first – requested word is transferred from the memory to the cache (and datapath) first
  - Nonblocking cache – Αφήνει το datapath να συνεχίσει να διαβάζει το cache καθώς το cache εργάζεται σε ένα προηγούμενο miss
- Write misses (D$)
  - Δεν μπορούμε να “κρατήσουμε” χώρο για γραφή γιατί αλλιώς θα καταλήξουμε σε “garbled” block στο cache (e.g., for 4 word blocks, καινούριο tag, μία λέξη δεδομένων απο το νέο block, και τρείς λέξεις δεδομένων από το προηγόυμενο block), άρα πρέπει να φέρουμε το block από το επόμενο επίπεδο μνήμης και να πληρώσουμε το τίμημα για το σταμάτημα της διασωλήνωσης.

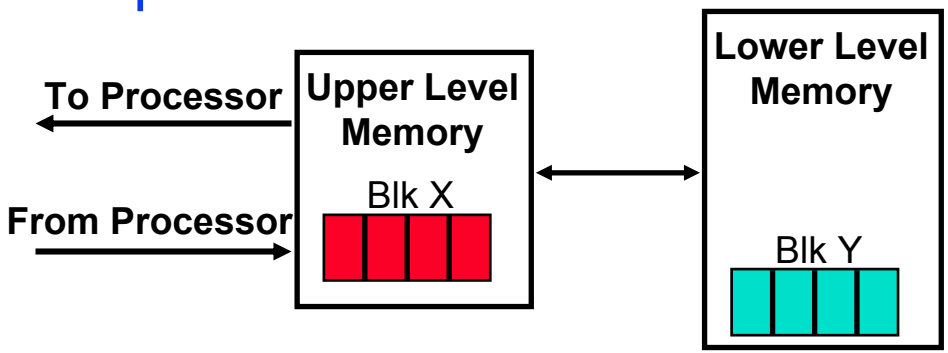
Τί μάθαμε μέχρι τώρα για το Cache

- Η αρχή της τοπικότητας (Principle of Locality):
  - Program likely to access a relatively small portion of the address space at any instant of time
    - Temporal Locality: Locality in Time
    - Spatial Locality: Locality in Space
- Three major categories of cache misses:
  - Compulsory misses: sad facts of life. Example: cold start misses
  - Conflict misses: increase cache size and/or associativity (Nightmare Scenario: ping pong effect!)
  - Capacity misses: increase cache size
- Cache design space
  - total size, block size, associativity (replacement policy)
  - write-hit policy (write-through, write-back)
  - write-miss policy (write allocate, write buffers)

Επανάληψη : Τοπικότητα

- Temporal Locality – Τοπικότητα Χρόνου
  - Κρατάμε τα πιο πρόσφατα δεδομένα κοντά στον επεξεργαστή

- Spatial Locality – Τοπικότητα Χώρου
  - Κινούμε blocks που αποτελούνται από συνεχείς λέξεις κοντά στον επεξεργαστή.



- Hit Time << Miss Penalty
  - Hit: data appears in some block in the upper level (Blk X)
    - Hit Rate: the fraction of accesses found in the upper level
    - Hit Time: RAM access time + Time to determine hit/miss
  - Miss: data needs to be retrieve from a lower level block (Blk Y)
    - Miss Rate = 1 - (Hit Rate)
    - Miss Penalty: Time to replace a block in the upper level with a block from the lower level + Time to deliver this block's word to the processor
    - Miss Types: Compulsory, Conflict, Capacity

Μετρώντας την απόδοση του Cache

- Αν υποθέσουμε ότι το κόστος ενός Cache hit υπολογίζεται σαν μέρος του χρόνου του επεξεργαστή (CPU execution cycle)

$$\underbrace{\text{CPU time} = \text{IC} \times \text{CPI} \times \text{CC}}_{\text{CPI}_{\text{stall}}}$$

CC?

$$= \text{IC} \times (\text{CPI}_{\text{ideal}} + \text{Memory-stall cycles}) \times \text{CC}$$

- Οι κύκλοι που το CPU περιμένει την μνήμη προέρχονται από cache misses (a sum of read-stalls and write-stalls)

$$\text{Read-stall cycles} = \text{reads/program} \times \text{read miss rate} \times \text{read miss penalty}$$

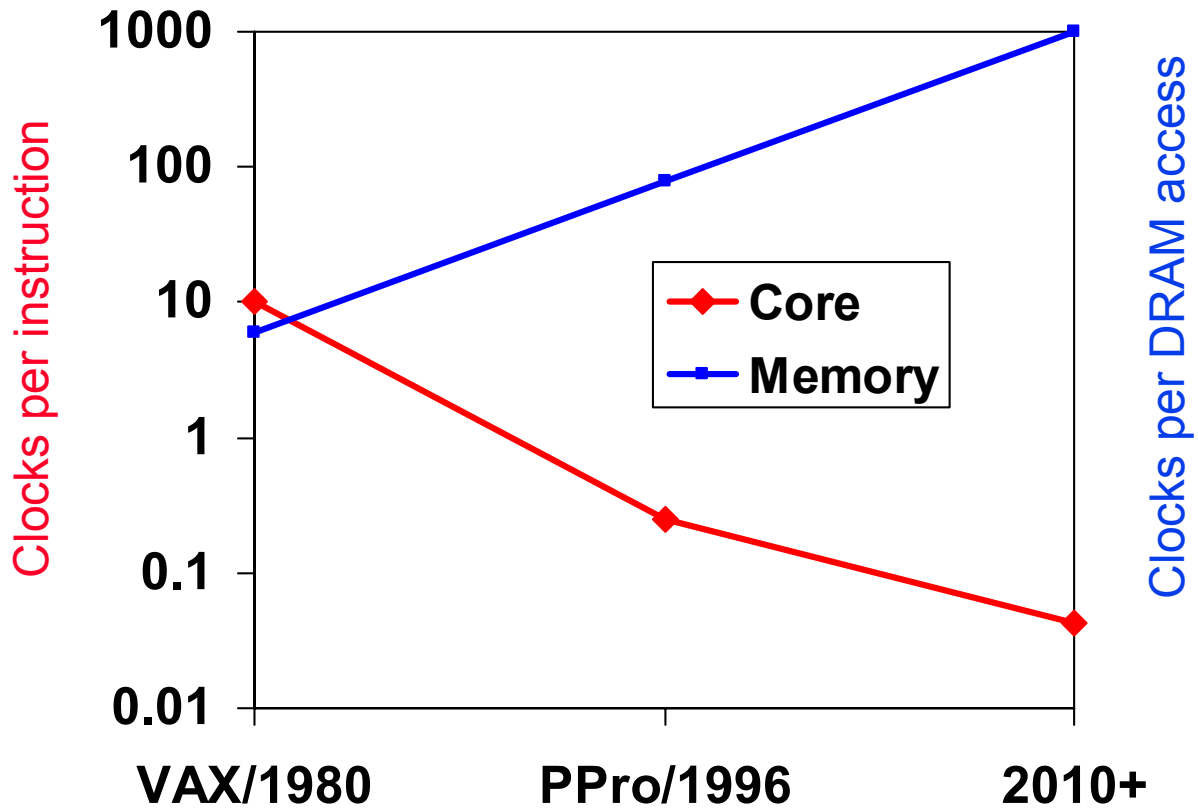
$$\text{Write-stall cycles} = (\text{writes/program} \times \text{write miss rate} \times \text{write miss penalty}) + \text{write buffer stalls}$$

- Για write-through caches, απλοποιούμε την εξίσωση:

$$\text{Memory-stall cycles} = \text{miss rate} \times \text{miss penalty}$$

ΑΣ ΘΥΜΗΘΟΥΜΕ: The “Memory Wall”

- Logic vs DRAM speed gap continues to grow



Επίδραση στην απόδοση του Cache

- Σχετικά, το penalty για το Cache αυξάνεται(faster clock rate and/or lower CPI)
  - Η ταχύτητα της μνήμης δέν βελτιώνεται τόσο σε σχέση με την ταχύτητα του CPU. Όταν υπολογίζουμε το CPI_{stall}, το cache miss penalty μετριέται σε *processor* clock cycles που χρειάζονται γιανα αντιμετωπίσουν ένα miss
  - Όσο πιο χαμηλό [CPI_{ideal}, τόσο πιο πολύ το πέναλτυ για stalls
- Ένας επεξεργαστής με CPI_{ideal} 2, με 100 cycle miss penalty, 36% load/store instr's, και 2% I$ και 4% D$ miss rates

$$\text{Memory-stall cycles} = 2\% \times 100 + 36\% \times 4\% \times 100 = 3.44$$

$$\text{So} \quad CPI_{stalls} = 2 + 3.44 = 5.44$$

- Τι γίνεται αν το CPI_{ideal} μειώνεται σε 1? 0.5? 0.25?
- Τι γίνεται αν το processor clock rate διπλασιαστεί (doubling the miss penalty)?

ΜΕΙΩΣΗ Cache Miss Rates (#1)

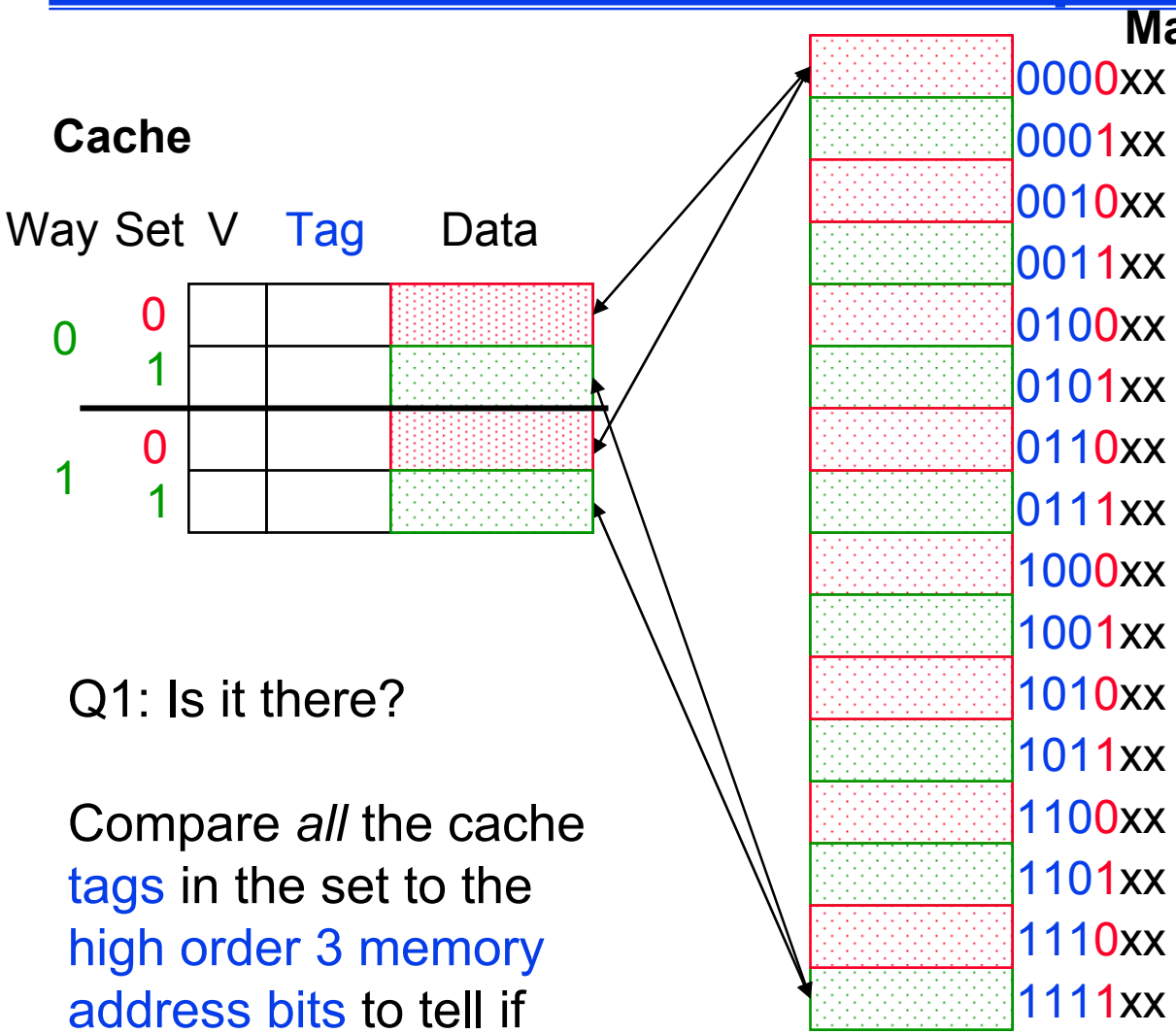
1. Πιο ευέλικτη τοποθέτηση blocks στο Cache

- Σε ένα απ'ευθείας ταξινομιμένο (direct mapped) cache ένα block μνήμης αντιστοιχεί σε ένα cache block
- Αντίθετα, μπορούμε να αφήσουμε ένα memory block να αντιστοιχεί σε οποιοδήποτε cache block – fully associative cache

- Ένας συμβιβασμός είναι η υποδιέρεση του cache σε sets,το καθένα εκ των οποίων αποτελείται από *n* τρόπους (n "ways" (n-way set associative)). Ένα block μνήμης αντιστοιχεί σε ένα μοναδικό set (καθορίζεται από το index field) και μπορεί να τοποθετηθεί με οποιοδήποτε τρόπο σε αυτό το set (άρα έχουμε *n* επιλογές)

(block address) modulo (# sets in the cache)

Set Associative Cache Example



Two low order bits define the byte in the word (32-b words) One word blocks

Q1: Is it there?

Compare *all* the cache tags in the set to the high order 3 memory address bits to tell if the memory block is in the cache

Q2: How do we find it?

Use next 1 low order memory address bit to determine which cache set (i.e., modulo the number of sets in the cache)

Another Reference String Mapping

- Consider the main memory word reference string

Start with an empty cache - all blocks initially marked as not valid

0 4 0 4 0 4 0 4

0 miss

000	Mem(0)

4 miss

000	Mem(0)
010	Mem(4)

0 hit

000	Mem(0)
010	Mem(4)

4 hit

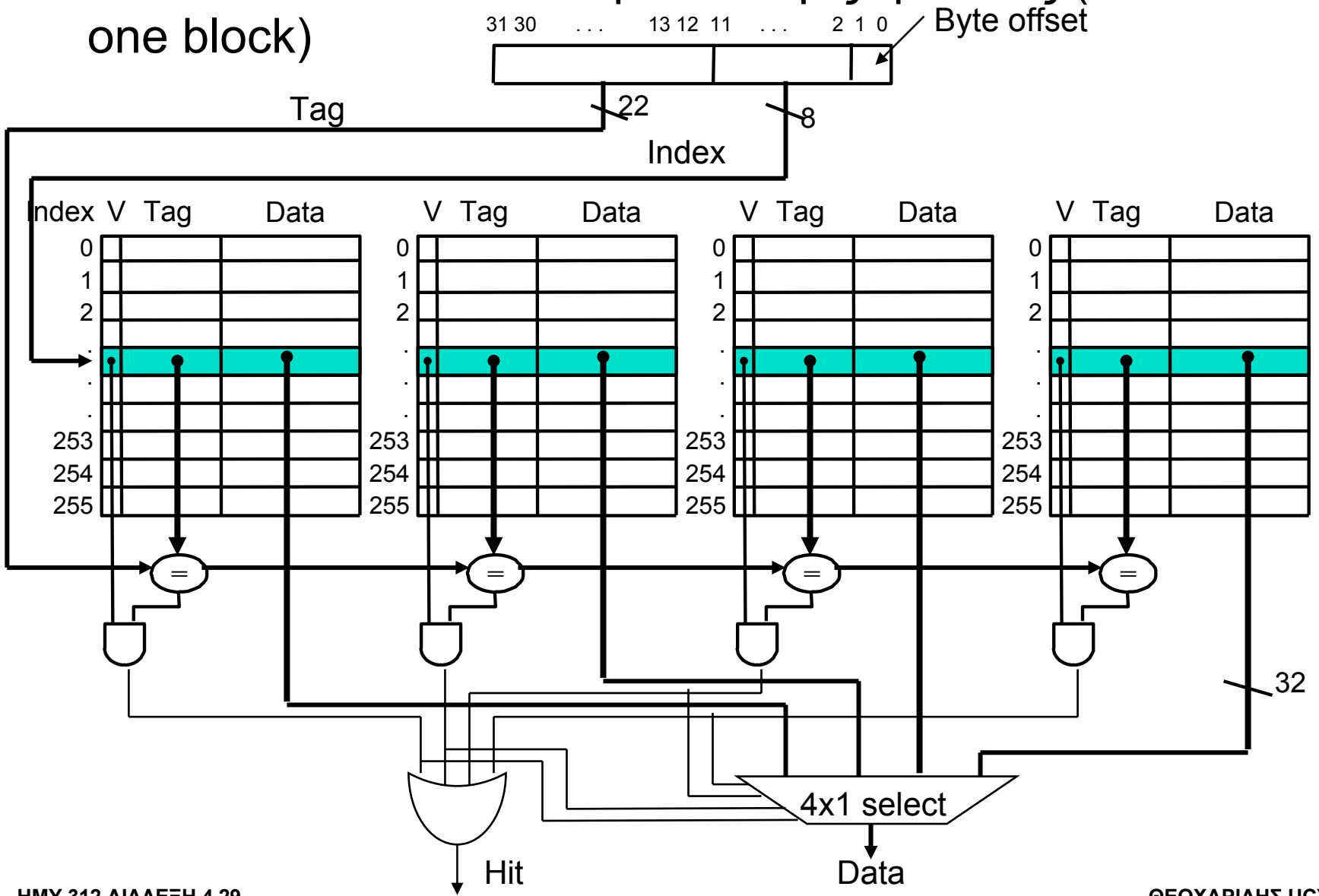
000	Mem(0)
010	Mem(4)

- 8 requests, 2 misses

- Μας λύνει το ping pong effect σε ένα direct mapped cache που οφείλεται στα conflict misses επειδή δύο (και περισσότερες(θέσεις μνήμης που αντιστοιχούν στο ίδιο cache set μπορούν να συνυπάρξουν!

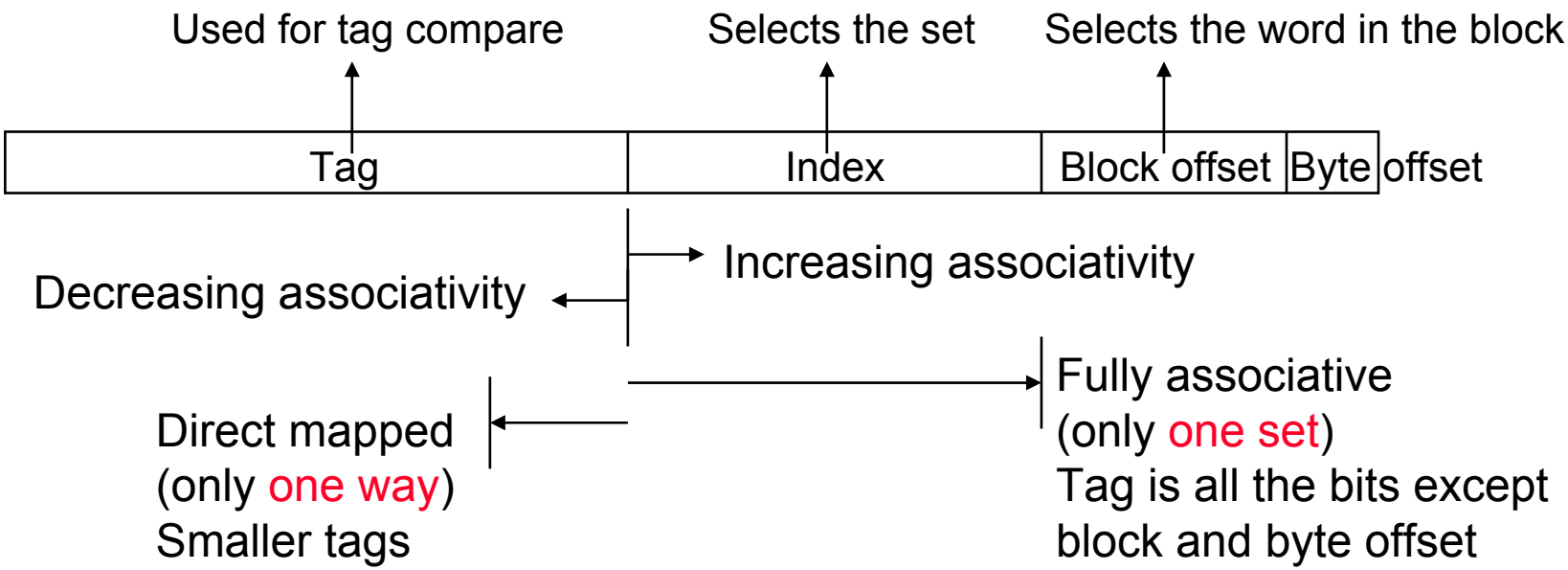
(Τεταπλό Set) Four-Way Set Associative Cache

- 2^8 = 256 sets το καθένα με τέσσερις τρόπους (each with one block)



Range of Set Associative Caches

- Για ένα συγκεκριμένο μέγεθος Cache, κάθε διπλασιασμός στην σύζευξη (assosiativity) διπλασιάζει τον αριθμό blocks ανα σετ (τον αριθμό τρόπων – ways) και μειώνει δια δύο των αριθμό sets – μειώνει το index κατά ένα bit και αυξάνει το tag κατά ένα bit.

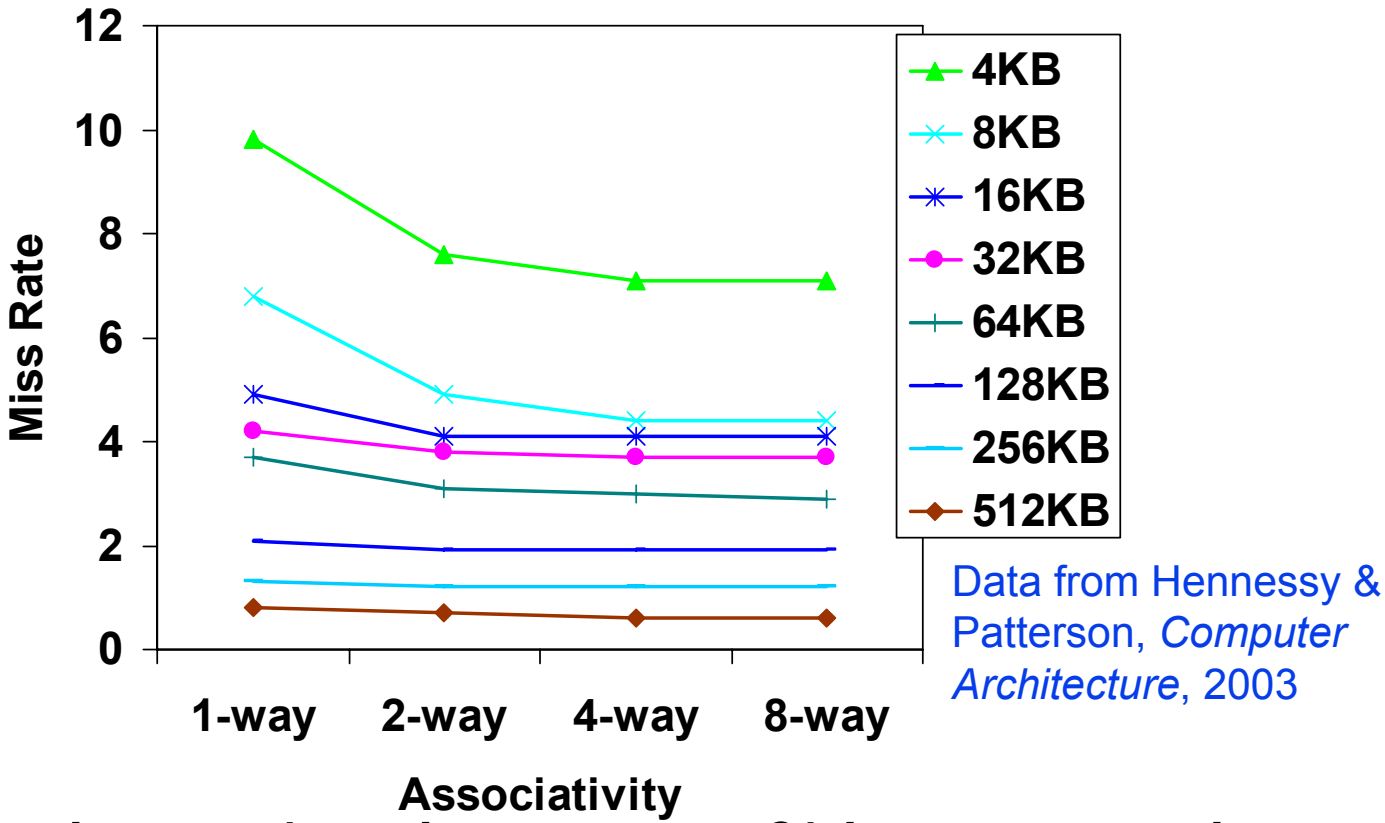


ΚΟΣΤΟΣ για Set Associative Caches

- Όταν έχουμε miss, πιανού τρόπου (way's) το block διαλέγουμε για αλλαγή?
  - Το block που χρησημοποιήθηκε για τελεύταία φορα (Least Recently Used (LRU): το block που αλλάζουμε είναι εκείνο που δεν έχει χρησημοποιηθεί για την περισσότερη ώρα.
    - Πρέπει να έχουμε hardware για να εντοπίζουμε πιανού τρόπου (way's) το block εκδιώξαμε και πότε
    - Για 2-way set associative, θέλουμε ένα bit ανά set → set the bit when a block is referenced (and reset the other way's bit)
- N-way set associative cache costs
  - N comparators (delay και area)
  - MUX delay (set selection) πριν τα δεδομένα να είναι διαθέσιμα
  - Τα δεδομένα είναι διαθέσιμα μετα από την επιλογή του set (απόφαση για Hit/Miss). Σε ένα απευθείας (direct mapped) cache, το cache block είναι διαθέσιμο πριν την απόφαση για Hit/Miss
    - Άρα δεν είναι δυνατό να υποθέσουμε ότι είχε hit και να συνεχίσουμε επανακτώντας μετά αν είταν miss

Πλεονεκτήματα για Set Associative Caches

- Η επιλογή ενός direct mapped ή ενός set associative cache εξαρτάται από το κόστος ενός miss έναντι του κόστους του Cache (implementation cost).



Data from Hennessy & Patterson, *Computer Architecture*, 2003

- Τα περισσότερα πλεονέκτηματα τα βλέπουμε πηγαίνοντας από direct mapped σε 2-way (20%+ reduction in miss rate)

Μειώνοντας το Cache Miss Rate (#2)

2. Χρήση πολλαπλών επιπέδων caches

- Με τις εξελίξεις στην τεχνολογία έχουμε τη δυνατότητα για να βάλουμε μεγαλύτερα L1 caches *ή* για δεύτερο επίπεδο caches – συνήθως ενός unified L2 cache (i.e., κρατάει και εντολές και δεδομένα) και σε μερικές περιπτώσεις ακόμα και μέχρι ενός unified L3 cache
- Για το παράδειγμα μας, CPI_{ideal} με 2, 100 cycle miss penalty (για κυρίως μνήμη), 36% load/stores, με 2% (4%) L1I$ (D$) miss rate, προσθέτουμε ένα UL2$ που έχει ένα 25 cycle miss penalty και ένα 0.5% miss rate

$$CPI_{stalls} = 2 + .02\times25 + .36\times.04\times25 + .005\times100 + .36\times.005\times100 = 3.54$$

(σε σύγκριση με το 5.44 χωρίς L2$)

Σχεδιάζοντας Πολλαπλά Επίπεδα Cache (Multilevel Cache)

- Οι βάσεις για σχεδιασμό ενός L1 και ενός L2 caches είναι πολύ διαφορετικές
  - Το κυρίως cache πρέπει να μπορεί να μειώνει οσο πιο πολύ το hit time έναντι πιο γρήγορου ωρολογιακού κύκλου
    - Primary cache should focus on minimizing hit time in support of a shorter clock cycle
    - Smaller with smaller block sizes
  - Το δεύτερο(α) cache(s) πρέπει να μειώνουν το miss rate για να μειώνουν την ποινή μιας μακράς προσπέλασης μνήμης.
    - reducing miss rate to reduce the penalty of long main memory access times
    - Larger with larger block sizes
- Το miss penalty του L1 cache μειώνετε αισθητά με την παρουσία L2 cache – άρα μπορεί να είναι μικρότερο (i.e., γρηγορότερο) αλλά θα έχει ψηλότερο miss rate
- Για το L2 cache, το hit time είναι λιγότερο σημαντικό από το miss rate
  - The L2$ hit time determines L1$'s miss penalty
  - L2$ local miss rate >> than the global miss rate

Σημαντικές Παραμέτροι ενός Cache

	L1 typical	L2 typical
Total size (blocks)	250 to 2000	4000 to 250,000
Total size (KB)	16 to 64	500 to 8000
Block size (B)	32 to 64	32 to 128
Miss penalty (clocks)	10 to 25	100 to 1000
Miss rates (global for L2)	2% to 5%	0.1% to 2%

Παραμέτροι Cache Δύο Δημοφιλών Επεξεργαστών

	Intel P4	AMD Opteron
L1 organization	Split I$ and D$	Split I$ and D$
L1 cache size	8KB for D$, 96KB for trace cache (~I$)	64KB for each of I$ and D$
L1 block size	64 bytes	64 bytes
L1 associativity	4-way set assoc.	2-way set assoc.
L1 replacement	~ LRU	LRU
L1 write policy	write-through	write-back
L2 organization	Unified	Unified
L2 cache size	512KB	1024KB (1MB)
L2 block size	128 bytes	64 bytes
L2 associativity	8-way set assoc.	16-way set assoc.
L2 replacement	~LRU	~LRU
L2 write policy	write-back	write-back

4 Ερωτήσεις για την ιεραρχία Μνήμης

- Q1: Πού μπορεί να τοποθετηθεί ένα block στο ψηλότερο επιπεδο? *(Block placement)*
- Q2: Πώς βρίσκουμε ένα block στο ψηλότερο επίπεδο? *(Block identification)*
- Q3: Πιο block αλλάζουμε σε περίπτωση ενός miss? *(Block replacement)*
- Q4: Τι γίνεται όταν γράφουμε δεδομένα? *(Write strategy)*

Q1&Q2: Πού μπορεί ένα block να τοποθετηθεί/βρεθεί?

	# of sets	Blocks per set
Direct mapped	# of blocks in cache	1
Set associative	(# of blocks in cache)/ associativity	Associativity (typically 2 to 16)
Fully associative	1	# of blocks in cache

	Location method	# of comparisons
Direct mapped	Index	1
Set associative	Index the set; compare set's tags	Degree of associativity
Fully associative	Compare all blocks tags	# of blocks

Q3: Πιο block πρέπει να ανταλλαχτεί κατα ένα miss?

- Εύκολο για direct mapped – μόνο μια επιλογή!
- Set associative ή fully associative
  - Τυχαία / Random
  - Αυτό που δεν χρησημοποιήθηκε για την περισσότερη ώρα LRU (Least Recently Used)
- Για ένα 2-way set associative cache, τυχαία ανταλλαγή έχει miss rate about 1.1 φορές ψηλότερη απο το LRU.
- Το LRU στοιχίζει αρκετά για να εφαρμοστεί σε πιο ψηλά επίπεδα συζευξηκότητας (associativity) (> 4-way) επειδή για να ακολουθούμε τα δεδομένα στο Cache έχει αρκετό κόστος στον χρόνο.

Q4: Τι γίνεται κατά ένα write?

- *Write-through* – Τα δεδομένα γράφονται και στο block στο cache και στο block στο αμέσως χαμηλότερο επίπεδο ιεραρχίας μνήμης
  - Write-through is always combined with a write buffer so write waits to lower level memory can be eliminated (as long as the write buffer doesn't fill)
- *Write-back* – Τα δεδομένα γράφονται μόνο στο block στο cache. Το διεγραμμένο/αλλαγμένο cache block μεταφέρεται στην κύρια μνήμη μόνο όταν ανταλλαγεί
  - Χρειαζόμαστε ένα dirty bit για να εντοπίζουμε αν το block έχει μετατροπεί ή όχι.
- Πλεονεκτήματα και μειονεκτήματα?
  - Write-through: Τα read misses δεν προκαλούν writes (so are simpler and cheaper)
  - Write-back: Επαναλαμβανόμενα writes απαιτούν μόνο ένα write αμέσως χαμηλότερο επίπεδο.

Βελτιώνοντας την απόδοση του Cache

0. Μειώνουμε την ώρα για ένα hit στο cache
   - smaller cache
   - direct mapped cache
   - smaller blocks
   - for writes
     - no write allocate – no “hit” on cache, just write to write buffer
     - write allocate – to avoid two cycles (first check for hit, then write) pipeline writes via a delayed write buffer to cache

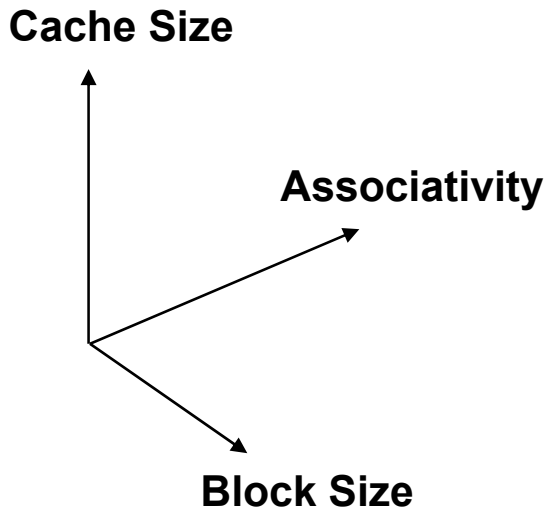
1. Μειώνουμε το miss rate
   - bigger cache
   - more flexible placement (increase associativity)
   - larger blocks (16 to 64 bytes typical)
   - victim cache – small buffer holding most recently discarded blocks

Βελτιώνοντας την απόδοση του Cache

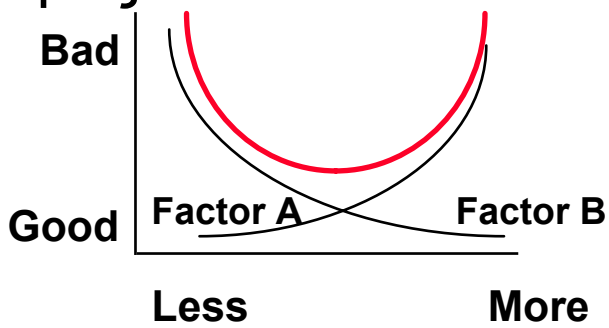
2. Μειώνουμε το miss penalty
   - smaller blocks
   - use a write buffer to hold dirty blocks being replaced so don't have to wait for the write to complete before reading
   - check write buffer (and/or victim cache) on read miss – may get lucky
   - for large blocks fetch critical word first
   - use multiple cache levels – L2 cache not tied to CPU clock rate
   - faster backing store/improved memory bandwidth
     - wider buses
     - memory interleaving, page mode DRAMs

ΠΕΡΙΛΗΨΗ : ΒΑΣΙΚΕΣ ΑΡΧΕΣ Cache Design

- Πάρα πολλές συγκρουόμενες διαστάσεις
  - cache size
  - block size
  - associativity
  - replacement policy
  - write-through vs write-back
  - write allocation



- Η πιο σωστή λύση είναι ο συμβιβασμός
  - depends on access characteristics
    - workload
    - use (I-cache, D-cache, TLB)
  - depends on technology / cost



- Simplicity often wins