

# Αρχιτεκτονική Υπολογιστών

## Κεφάλαιο #5 (β)

### Ιεραρχίες Μνήμης – Βελτιστοποίηση Κρυφών Μνημών

Διονύσης Πνευματικάτος

[pnevmati@cslab.ece.ntua.gr](mailto:pnevmati@cslab.ece.ntua.gr)

5ο εξάμηνο ΣΗΜΜΥ – Ακαδημαϊκό Έτος: 2021-22

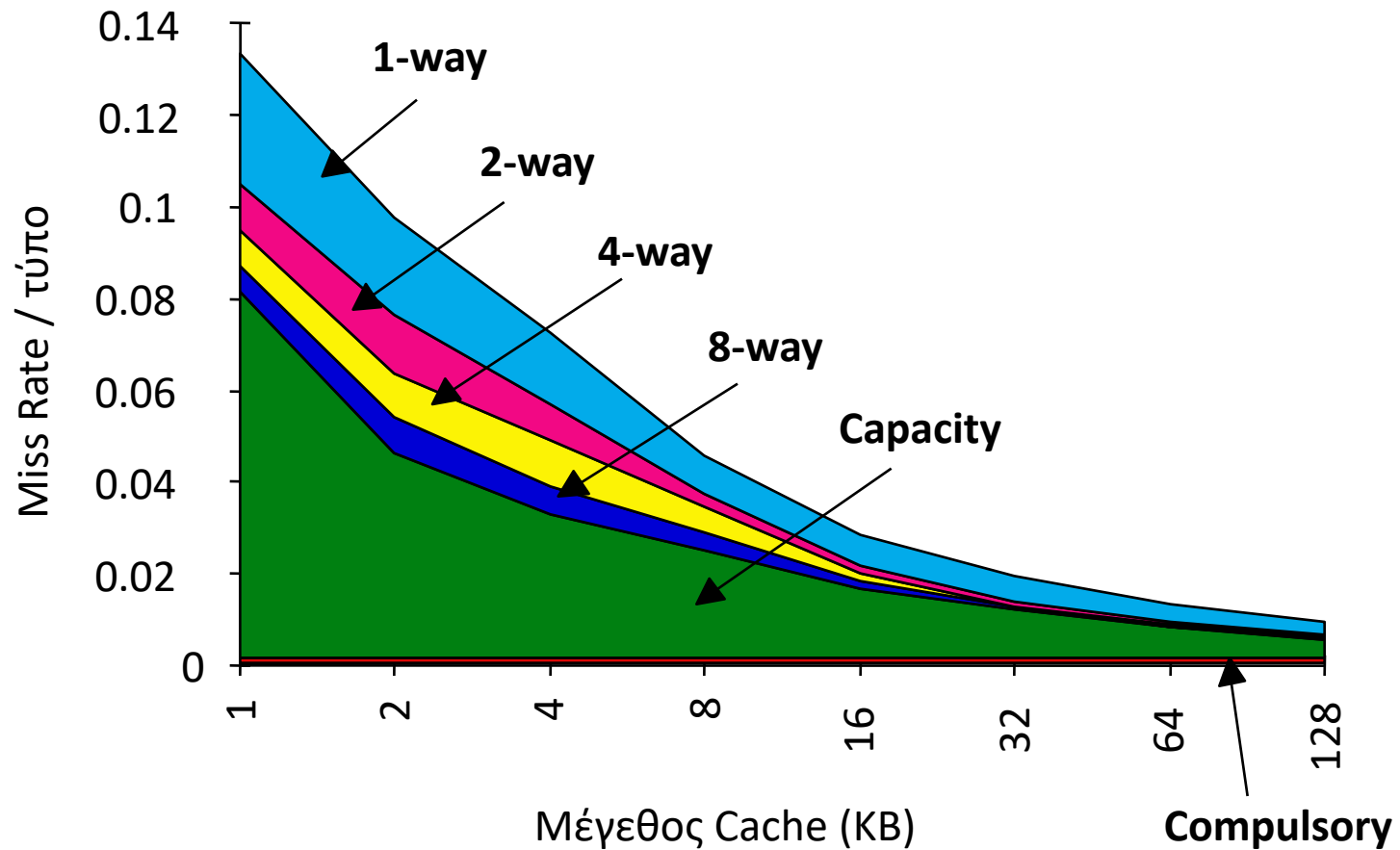
Τμήμα 3 (ΠΑΠΑΔ-Ω)

<http://www.cslab.ece.ntua.gr/courses/comparch/>

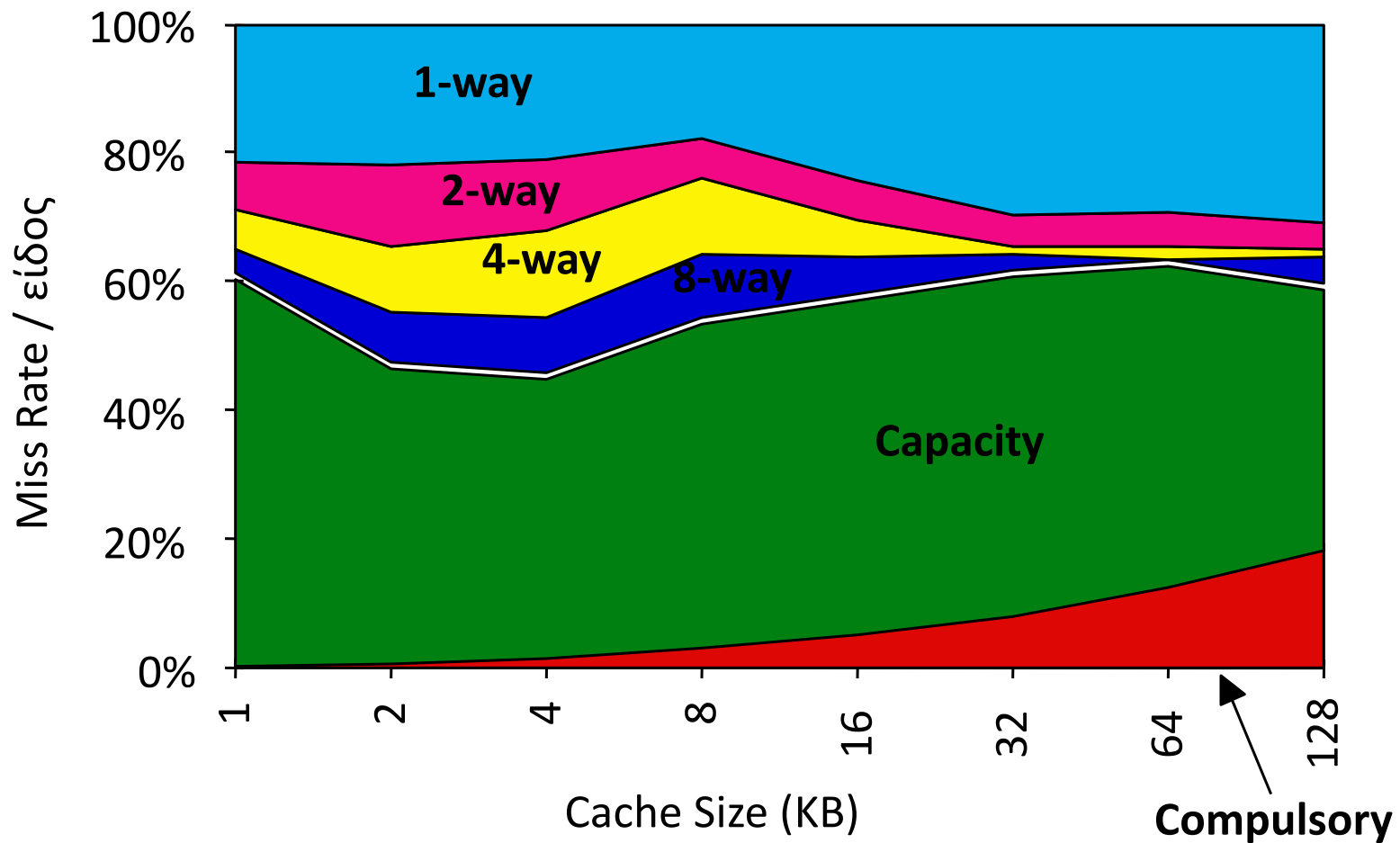
# Είδη των Cache Misses: 3C's

- Compulsory: Συμβαίνουν κατά την πρώτη πρόσβαση σε ένα block. Το block πρέπει να κληθεί από χαμηλότερα επίπεδα μνήμης και να τοποθετηθεί στην cache (αποκαλούνται και cold start misses ή first reference misses)
- Capacity: Τα blocks απομακρύνονται από την cache επειδή δεν χωράνε σε αυτήν όλα όσα απαιτούνται κατά την εκτέλεση ενός προγράμματος (το σύνολο των δεδομένων που χειρίζεται ένα πρόγραμμα είναι πολύ μεγαλύτερο από την χωρητικότητα της cache)
- Conflict: Στην περίπτωση των set associative ή direct mapped caches, conflict misses έχουμε όταν πολλά blocks απεικονίζονται στο ίδιο set (αποκαλούνται και collision misses ή interference misses)

# Τα 3 Cs : Απόλυτα Miss Rates (SPEC92)



# Τα 3 Cs: Σχετικά Miss Rates (SPEC92)



# Βελτιστοποίηση της επίδοσης της Cache

Πώς;

- Περιορισμός του Miss Rate
- Μείωση του Cache Miss Penalty
- Μείωση του χρόνου για Cache Hit

# Βελτιστοποίηση της επίδοσης της Cache

## ■ Τεχνικές μείωσης του Miss Rate:

- Μεγαλύτερο μέγεθος block
- Μεγαλύτερου βαθμού associativity
- Victim caches
- Compiler-controlled prefetching
- Αύξηση της χωρητικότητας της cache
- Pseudo-associative Caches
- Hardware/Software prefetching εντολών-δεδομένων
- Βελτιστοποιήσεις στον Compiler

## ■ Τεχνικές μείωσης του Cache Miss Penalty:

- Cache 2ου επιπέδου (L2)
- Early restart and critical word first
- Προτεραιότητα στα read misses έναντι των writes
- merging write buffers
- Non-blocking caches

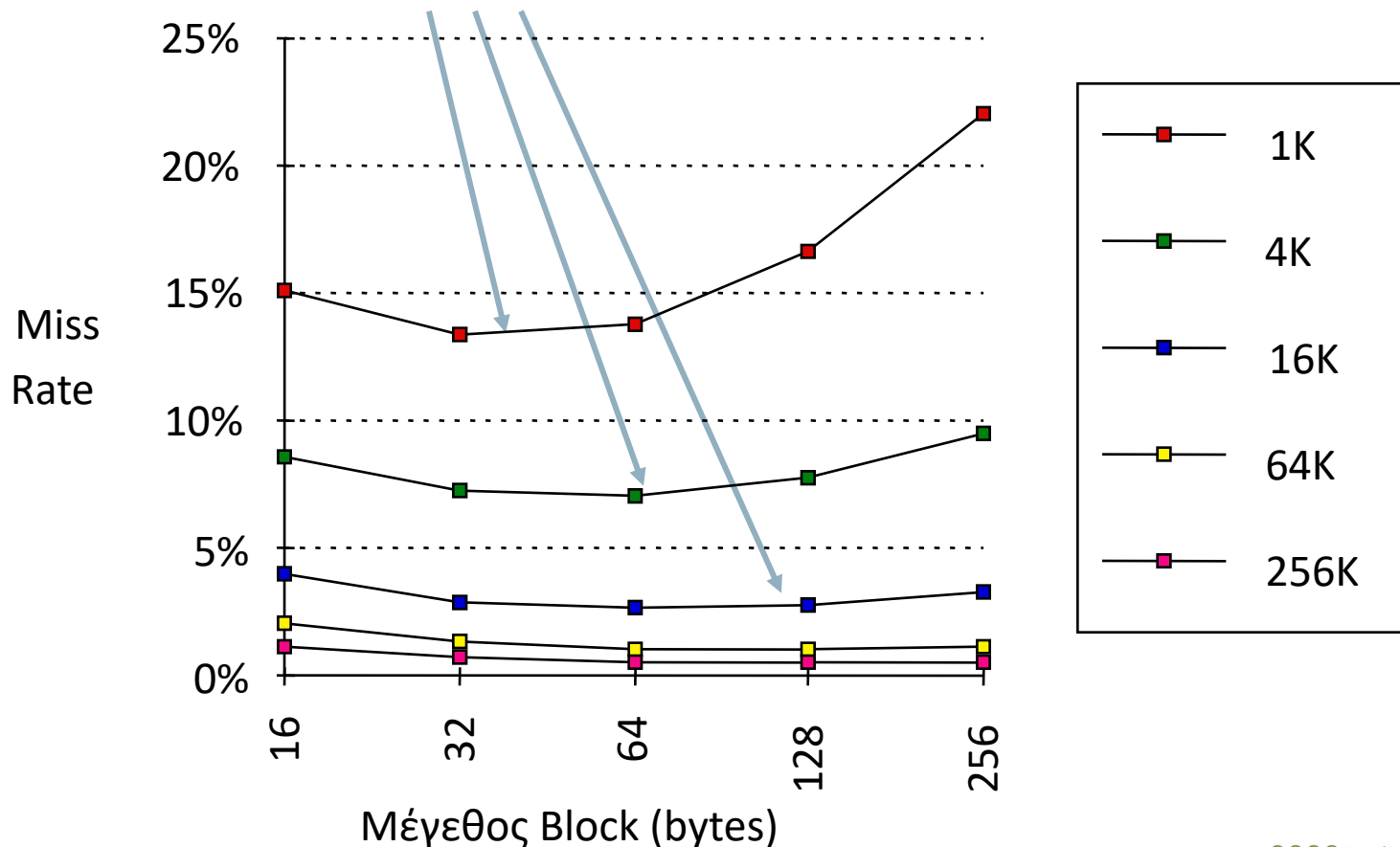
## ■ Τεχνικές μείωσης του Cache Hit Time:

- Μικρές και απλές caches
- Αποφυγή της μετάφρασης των διευθύνσεων κατά τη διάρκεια του indexing
- Pipelining writes για γρήγορα write hits

# Τεχνικές μείωσης του Miss Rate

## Μεγαλύτερο μέγεθος Block

- Το μεγάλο μέγεθος block βελτιώνει την επίδοση της cache επειδή επωφελούμαστε από την spatial locality
- Για δεδομένο μέγεθος cache, μεγαλύτερο μέγεθος block σημαίνει λιγότερα cache block frames
  - Η επίδοση βελτιώνεται μέχρι το σημείο όπου λόγω του μικρού αριθμού των cache block frames αυξάνονται τα conflict misses και επομένως και το συνολικό cache miss rate



# Τεχνικές μείωσης του Miss Rate

## Μεγαλύτερο μέγεθος cache

- Με την αύξηση του μεγέθους της cache προκαλείται:
  - αύξηση του hit time
  - αύξηση του κατασκευαστικού κόστους
- Αυτή η τεχνική δημοφιλής σε off-chip caches
- Σημείωση : οι L2,L3 caches σήμερα έχουν μέγεθος όσο ήταν η Κύρια Μνήμη πριν κάμποσα χρόνια



# Τεχνικές μείωσης του Miss Rate Μεγαλύτερου βαθμού Associativity

Παράδειγμα: Μέσος χρόνος πρόσβασης στη μνήμη vs. Miss Rate

Assume: CCT = 1.10 for 2-way, 1.12 for 4-way, 1.14 for 8-way vs. CCT direct mapped

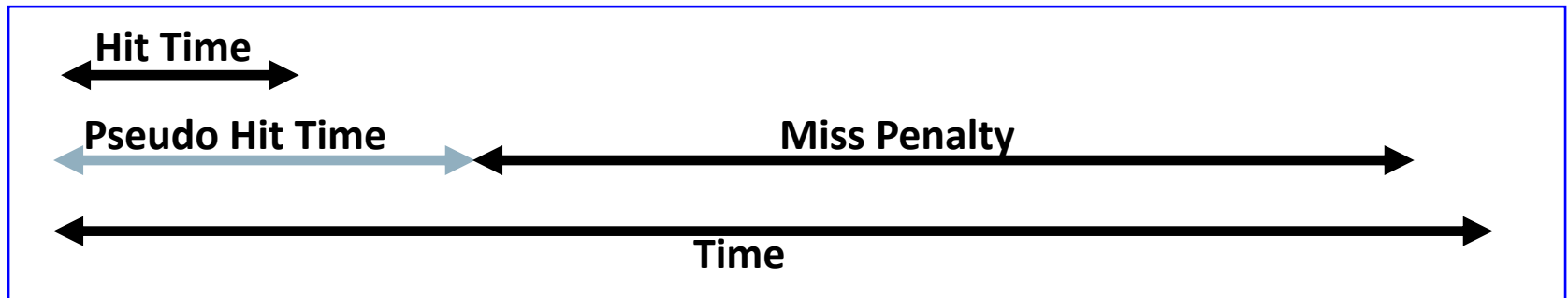
Cache Size (KB)	Associativity			
	1-way	2-way	4-way	8-way
1	2.33	2.15	2.07	2.01
2	1.98	1.86	1.76	1.68
4	1.72	1.67	1.61	1.53
8	1.46	<u>1.48</u>	<u>1.47</u>	1.43
<u>16</u>	<u>1.29</u>	<u>1.32</u>	<u>1.32</u>	<u>1.32</u>
<u>32</u>	<u>1.20</u>	<u>1.24</u>	<u>1.25</u>	<u>1.27</u>
<u>64</u>	<u>1.14</u>	<u>1.20</u>	<u>1.21</u>	<u>1.23</u>
<u>128</u>	<u>1.10</u>	<u>1.17</u>	<u>1.18</u>	<u>1.20</u>

Μπλε σημαίνει ότι ο μέσος χρόνος δεν βελτιώνεται με την αύξηση του associativity

# Τεχνικές μείωσης του Miss Rate

## Pseudo-Associative Cache

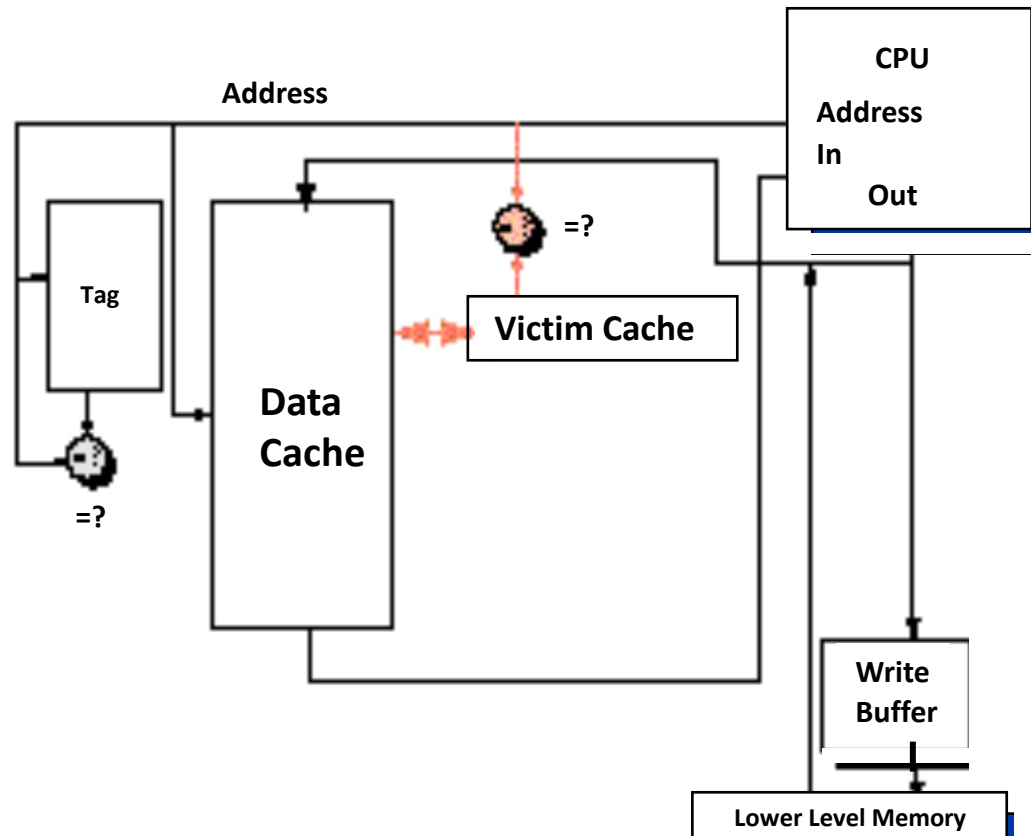
- Συνδυάζει το μικρό χρόνο αναζήτησης (hit time) των Direct Mapped caches και το μικρότερο αριθμό των conflict misses στις 2-way set-associative caches.
- Η cache διαιρείται σε δύο τμήματα: Όταν έχουμε cache miss, ελέγχουμε το άλλο μισό της cache για να δούμε αν τα δεδομένα που αναζητάμε βρίσκονται εκεί. Στην περίπτωση αυτή έχουμε ένα pseudo-hit (slow hit)
- Ο ευκολότερος τρόπος υλοποίησης είναι η αναστροφή του most significant bit στο πεδίο index για να βρίσκουμε το άλλο block στο “pseudo set”



- Μειονέκτημα: είναι δύσκολη η αποδοτική υλοποίηση του CPU pipelining αν το L1 cache hit παίρνει 1 ή 2 κύκλους.
  - Χρησιμοποιείται καλύτερα σε caches που δεν είναι συνδεδεμένες απευθείας με τη CPU (L2 cache)
  - Χρησιμοποιείται στην L2 cache του MIPS R10000, παρόμοια είναι και η L2 του UltraSPARC

# Τεχνικές μείωσης του Miss Rate Victim Caches

- Τα δεδομένα που απομακρύνονται από την cache τοποθετούνται σε έναν μικρό πρόσθετο buffer (victim cache).
- Σε περίπτωση cache miss ελέγχουμε το περιεχόμενο της victim cache πριν τα αναζητήσουμε στην κύρια μνήμη
- Jouppi [1990]: Μία victim cache 4 εισόδων αποτρέπει το 20% έως 95% των conflict misses για μία 4 KB direct mapped cache
- Χρησιμοποιείται σε Alpha, HP PA-RISC μηχανήματα.

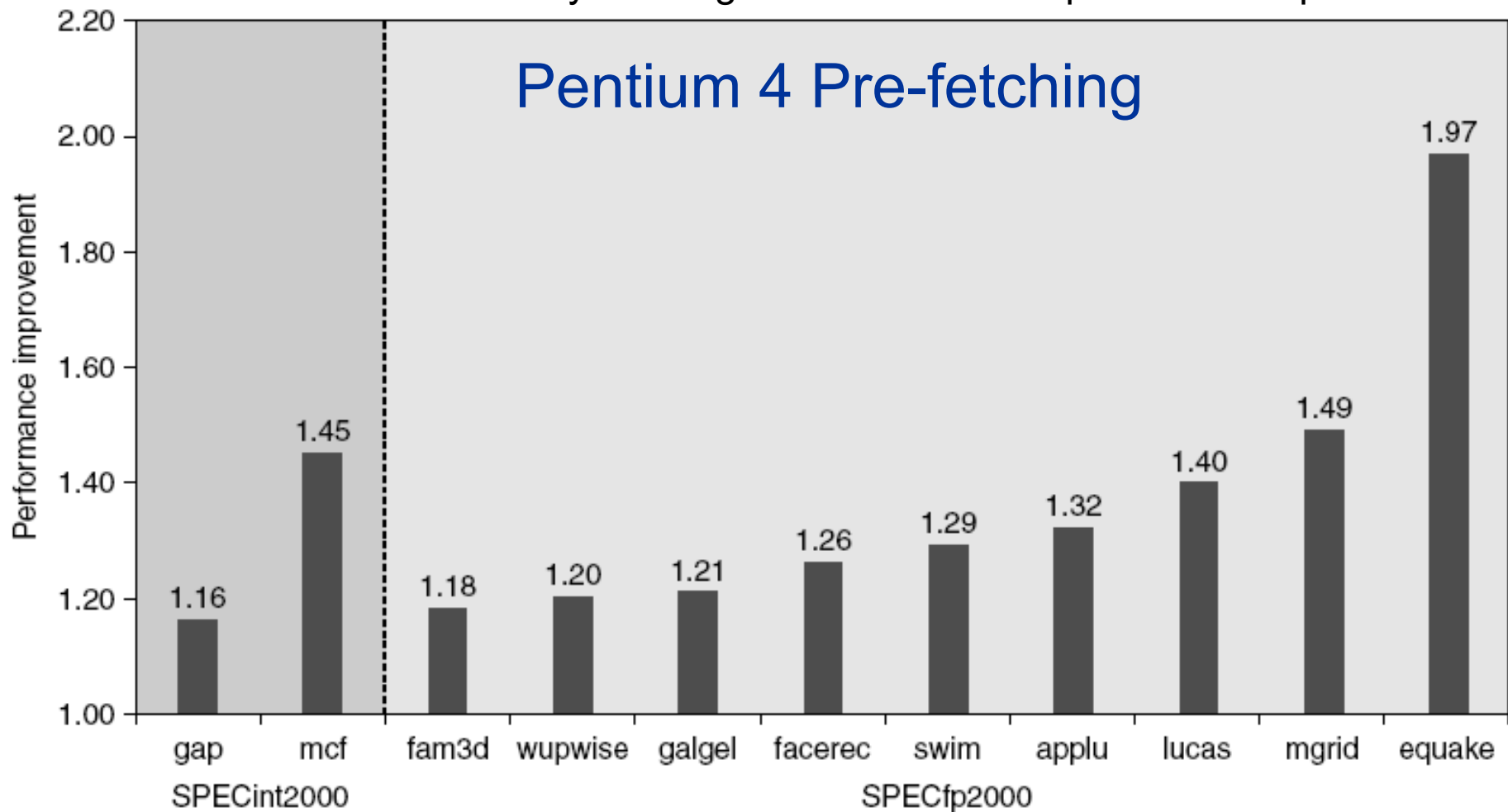


# Hardware/Compiler Prefetching εντολών και δεδομένων

- Φέρνουμε εντολές ή δεδομένα στην cache ή σε έναν εξωτερικό buffer (prefetch) πριν ζητηθούν από τη CPU
- Παράδειγμα: Ο Alpha APX 21064 φέρνει 2 blocks σε κάθε miss: Το ζητούμενο block τοποθετείται στην cache και το αμέσως επόμενο σε έναν stream buffer εντολών
- Η ίδια λογική εφαρμόζεται και στις προσπελάσεις δεδομένων με έναν data buffer
- Μπορεί να επεκταθεί και για πολλαπλούς stream buffers δεδομένων σε διαφορετικές διευθύνσεις (4 streams βελτιώνουν το data hit rate κατά 43%)
- Αποδεικνύεται ότι, σε ορισμένες περιπτώσεις, 8 stream buffers οι οποίοι χειρίζονται δεδομένα ή εντολές, μπορούν να αποτρέψουν το 50-70% των συνολικών misses

# Hardware Prefetching

- Fetch two blocks on miss (include next sequential block): overlapping memory access with execution by fetching data items before processor requests them



# Τεχνικές μείωσης του Miss Rate

## Compiler Optimizations

Βελτιστοποίηση του κώδικα επιτυγχάνοντας τοπικότητα κατά την προσπέλαση δεδομένων:

- *Αναδιοργάνωση συναρτήσεων*: στη μνήμη για τη μείωση των conflict misses
- *Merging Arrays*: Βελτίωση της spatial locality με έναν πίνακα δεδομένων αντί 2 πίνακες
- *Loop Interchange*: Αλλαγή της σειράς φωλιάσματος των βρόχων για να προσπελάζουμε τα δεδομένα με την ίδια σειρά όπως αποθηκεύονται στη μνήμη
- *Loop Fusion*: Συνδυασμός 2 ή περισσότερων ανεξάρτητων βρόχων που περιέχουν τους ίδιους βρόχους και κάποιες κοινές μεταβλητές
- *Blocking*: Βελτίωση της temporal locality προσπελάζοντας ένα τμήμα μόνο των δεδομένων επαναληπτικά αντί να διατρέχουμε ολόκληρες τις γραμμές ή τις στήλες

# Τεχνικές μείωσης του Miss Rate:

## Compiler Optimizations - Array Merging

```
/* Before: 2 sequential arrays */
int val[SIZE];
int key[SIZE];

/* After: 1 array of stuctures */
struct merge {
    int val;
    int key;
};
struct merge merged_array[SIZE];
```

Merging :

- Μειώνονται τα conflicts μεταξύ των στοιχείων των val και key
- Βελτίωση του spatial locality

# Τεχνικές μείωσης Miss Rate: Compiler Optimizations - Loop Interchange

```
/* Before */
for (k = 0; k < 100; k = k+1)
    for (j = 0; j < 100; j = j+1)
        for (i = 0; i < 5000; i = i+1)
            x[i][j] = 2 * x[i][j];

/* After */
for (k = 0; k < 100; k = k+1)
    for (i = 0; i < 5000; i = i+1)
        for (j = 0; j < 100; j = j+1)
            x[i][j] = 2 * x[i][j];
```

Η προσπέλαση δεδομένων που βρίσκονται σε συνεχόμενες θέσεις μνήμης και όχι με απόσταση 100 λέξεων βελτιώνει στο παράδειγμά μας την spatial locality



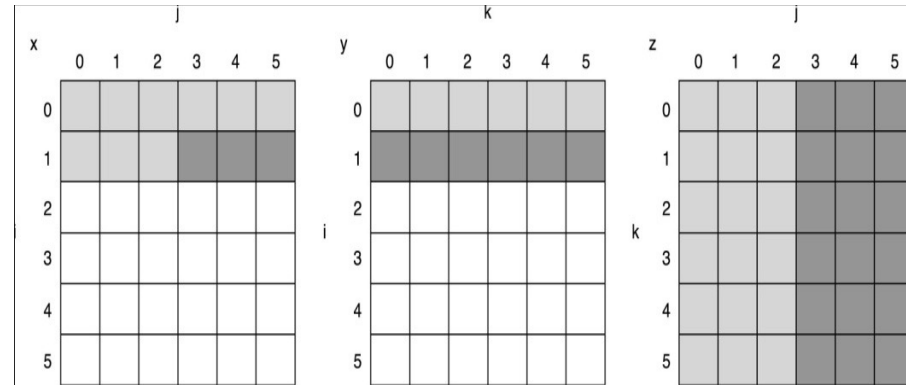
# Τεχνικές μείωσης Miss Rate: Compiler Optimizations - Loop Fusion

```
/* Before */
for (i = 0; i < N; i = i+1)
    for (j = 0; j < N; j = j+1)
        a[i][j] = 1/b[i][j] * c[i][j];
for (i = 0; i < N; i = i+1)
    for (j = 0; j < N; j = j+1)
        d[i][j] = a[i][j] + c[i][j];
/* After */
for (i = 0; i < N; i = i+1)
    for (j = 0; j < N; j = j+1)
    {
        a[i][j] = 1/b[i][j] * c[i][j];
        d[i][j] = a[i][j] + c[i][j];
    }
```

- Αντί 2 misses/access στα a & c τελικά 1 miss/access
- Βελτίωση της spatial locality

# Τεχνικές μείωσης Miss Rate: Compiler Optimizations - Blocking

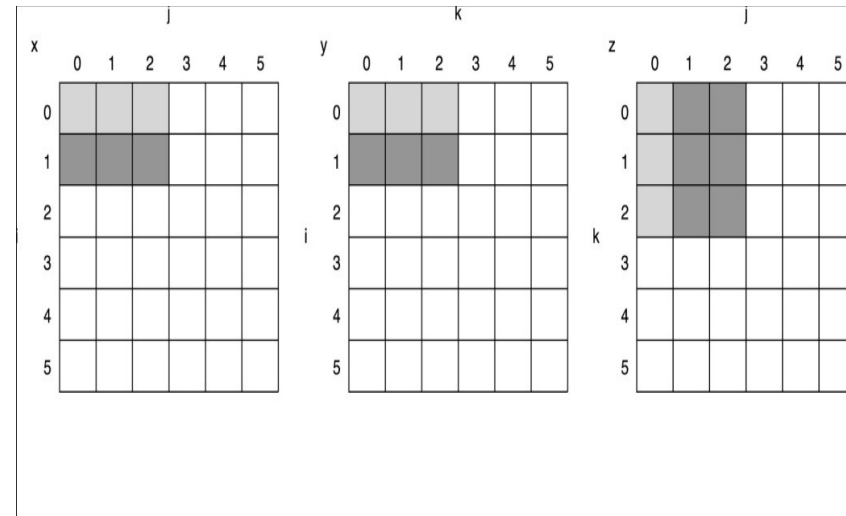
```
/* Before */  
for (i = 0; i < N; i = i+1)  
  for (j = 0; j < N; j = j+1)  
    {r = 0;  
      for (k = 0; k < N; k = k+1) {  
        r = r + y[i][k]*z[k][j];  
      };  
      x[i][j] = r;  
    };
```



- **Οι 2 εσωτερικότεροι βρόχοι:**
  - Προσπελαύνουν όλα τα  $N \times N$  στοιχεία του  $z[ ]$
  - Προσπελαύνουν επαναληπτικά τα  $N$  στοιχεία της 1 γραμμής του  $y[ ]$
  - Εγγραφή των  $N$  στοιχείων της 1 γραμμής του  $x[ ]$
- **Capacity Misses :** είναι συνάρτηση του  $N$  και του μεγέθους της Cache:
  - $3 N \times N \times 4 \leq \text{μεγέθους της Cache} \Rightarrow$  καθόλου capacity misses
- **Βασική ιδέα:** αναζητάμε τον  $B \times B$  υποπίνακα που χωράει στην cache

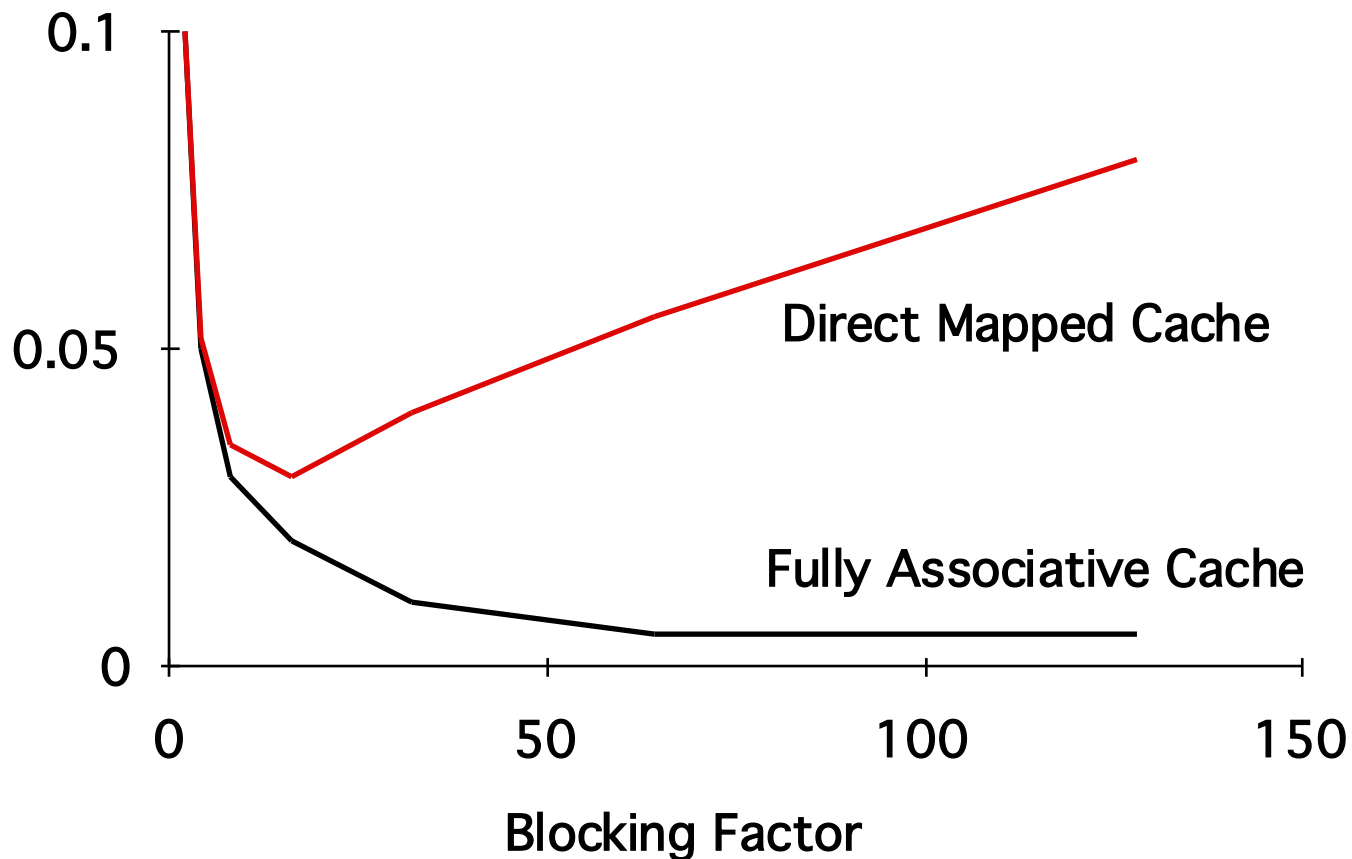
# Τεχνικές μείωσης Miss Rate: Compiler Optimizations - Blocking

```
/* After */
for (jj = 0; jj < N; jj = jj+B)
  for (kk = 0; kk < N; kk = kk+B)
    for (i = 0; i < N; i = i+1)
      for (j=jj; j<min(jj+B,N); j=j+1)
        {r = 0;
         for (k=kk; k<min(kk+B,N); k= +1)
           r = r + y[i][k]*z[k][j];
         x[i][j] = x[i][j] + r;
        };
```



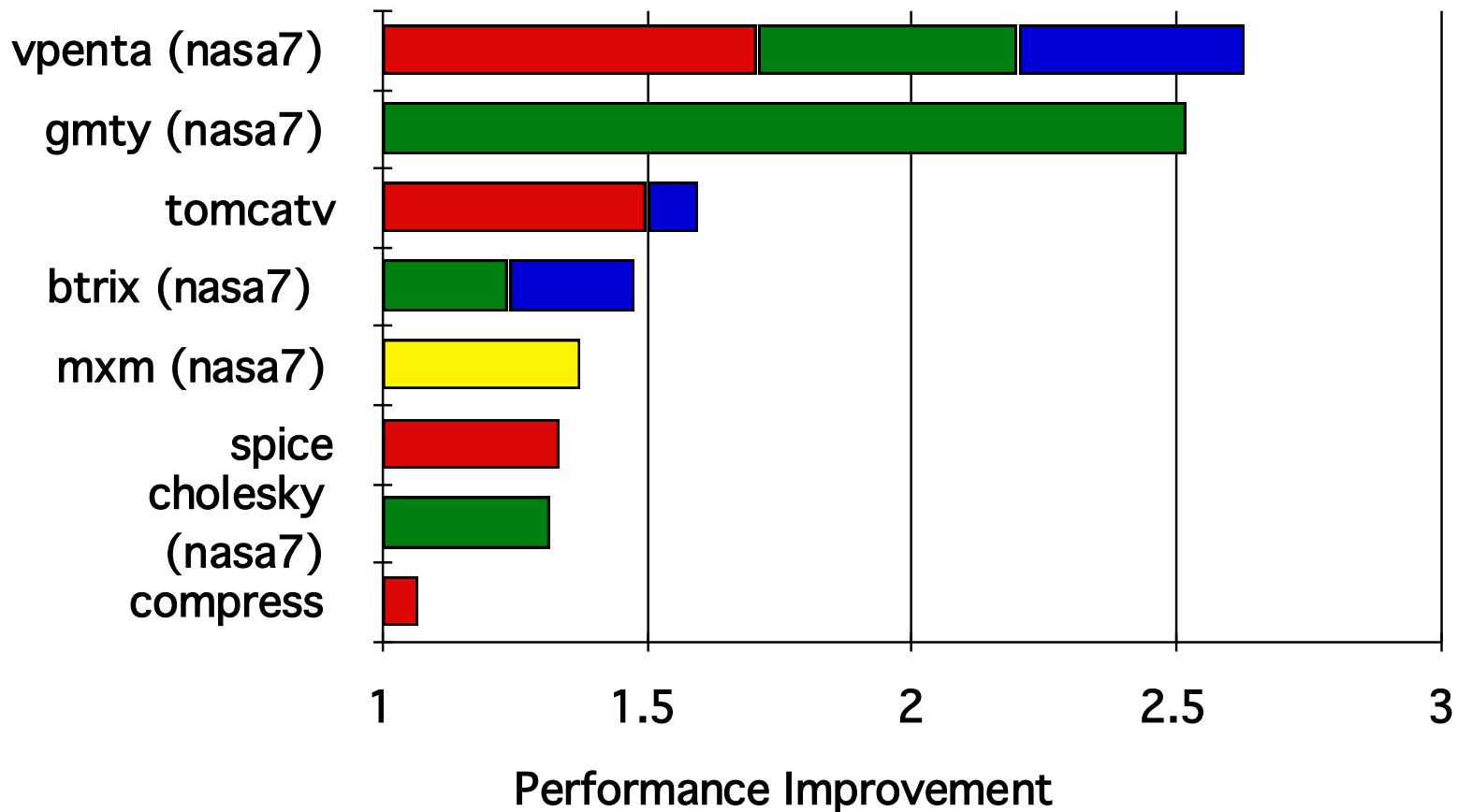
- $B$  : *Blocking Factor*
- Capacity Misses αντί  $2N^3 + N^2 \rightarrow 2N^3/B + N^2$
- Πιθανόν να επηρεάζονται και τα conflict misses

# Τεχνικές μείωσης Miss Rate: Compiler Optimizations - Blocking



- Conflict misses in caches not FA vs. Blocking size
  - Lam et al [1991] a blocking factor of 24 had a fifth the misses vs. 48 despite both fit in cache

# Τεχνικές μείωσης Miss Rate: Compiler Optimizations - Blocking



merged arrays

loop interchange

loop fusion

blocking

# Τεχνικές μείωσης Miss Penalty: Early Restart και Critical Word First

- Δεν περιμένουμε να μεταφερθεί το πλήρες block στην cache πριν την επανεκκίνηση της CPU:
  - **Early restart:** Αμέσως μόλις φορτωθεί η ζητούμενη λέξη του block, αποστέλλεται στη CPU και συνεχίζεται η επεξεργασία των δεδομένων από αυτήν
  - **Critical Word First:** Φορτώνεται πρώτη από όλο το block η ζητούμενη λέξη και αποστέλλεται στη CPU αμέσως μόλις φτάσει
    - Έτσι η CPU συνεχίζει την επεξεργασία ενώ οι υπόλοιπες λέξεις του block μεταφέρονται από την κύρια μνήμη
- Είναι συνήθως χρήσιμες όταν το μέγεθος των cache block είναι μεγάλο
- Τα προγράμματα με καλή spatial locality ζητούν δεδομένα που βρίσκονται σε συνεχόμενες θέσεις μνήμης και δεν επωφελούνται από την τεχνική του early restart

# Τεχνικές μείωσης Miss Penalty

## Προτεραιότητα στα Read Misses

- Στις write-through caches με write buffers παρουσιάζεται πρόβλημα με τις συγκρούσεις RAW κατά την ανάγνωση από την κύρια μνήμη σε περίπτωση που έχουμε cache miss:
  - Ο write buffer κρατά τα προσφάτως τροποποιημένα δεδομένα που χρειάζονται για την ανάγνωση
  - Μία λύση είναι απλά να περιμένουμε μέχρι να αδειάσει ο write buffer, αυξάνοντας έτσι το miss penalty (σε παλιούς MIPS 1000 κατά 50% )
  - Ελέγχουμε τα περιεχόμενα του write buffer πριν την ανάγνωση: αν δεν υπάρχουν εκεί τα ζητούμενα δεδομένα, πρέπει να τα φέρουμε από την κύρια μνήμη
- Σε write-back caches, όταν ένα read miss αντικαθιστά ένα dirty block:
  - Συνήθως: Πρώτα μεταφέρεται το dirty block στη μνήμη και στη συνέχεια πραγματοποιείται η ανάγνωση
  - Διαφορετικά: Αντιγράφεται το dirty block σε έναν write buffer, στη συνέχεια πραγματοποιείται η ανάγνωση, και τέλος η εγγραφή
  - Η CPU καθυστερεί λιγότερο γιατί ξεκινάει την επεξεργασία δεδομένων αμέσως μετά την ανάγνωση

# Τεχνικές μείωσης του Miss Penalty

## Merging write buffers

- Ένα cache block frame διαιρείται σε sub-blocks.
- Υπάρχει ένα valid bit ανά sub-block στα cache block frames.
- Δε χρειάζεται να φορτώσουμε ένα ολόκληρο block στην περίπτωση miss αλλά μόνο το ζητούμενο sub-block.

Διεύθυνση εγγραφής

Write address	V		V		V		V	
100	1	Mem[100]	0		0		0	
108	1	Mem[108]	0		0		0	
116	1	Mem[116]	0		0		0	
124	1	Mem[124]	0		0		0	

Διεύθυνση εγγραφής

Write address	V		V		V		V	
100	1	Mem[100]	1	Mem[108]	1	Mem[116]	1	Mem[124]
	0		0		0		0	
	0		0		0		0	
	0		0		0		0	

κάθε buffer χωράει 4 λέξεις των 64-bit.

Μόνο στο 2ο σχήμα αξιοποιούνται



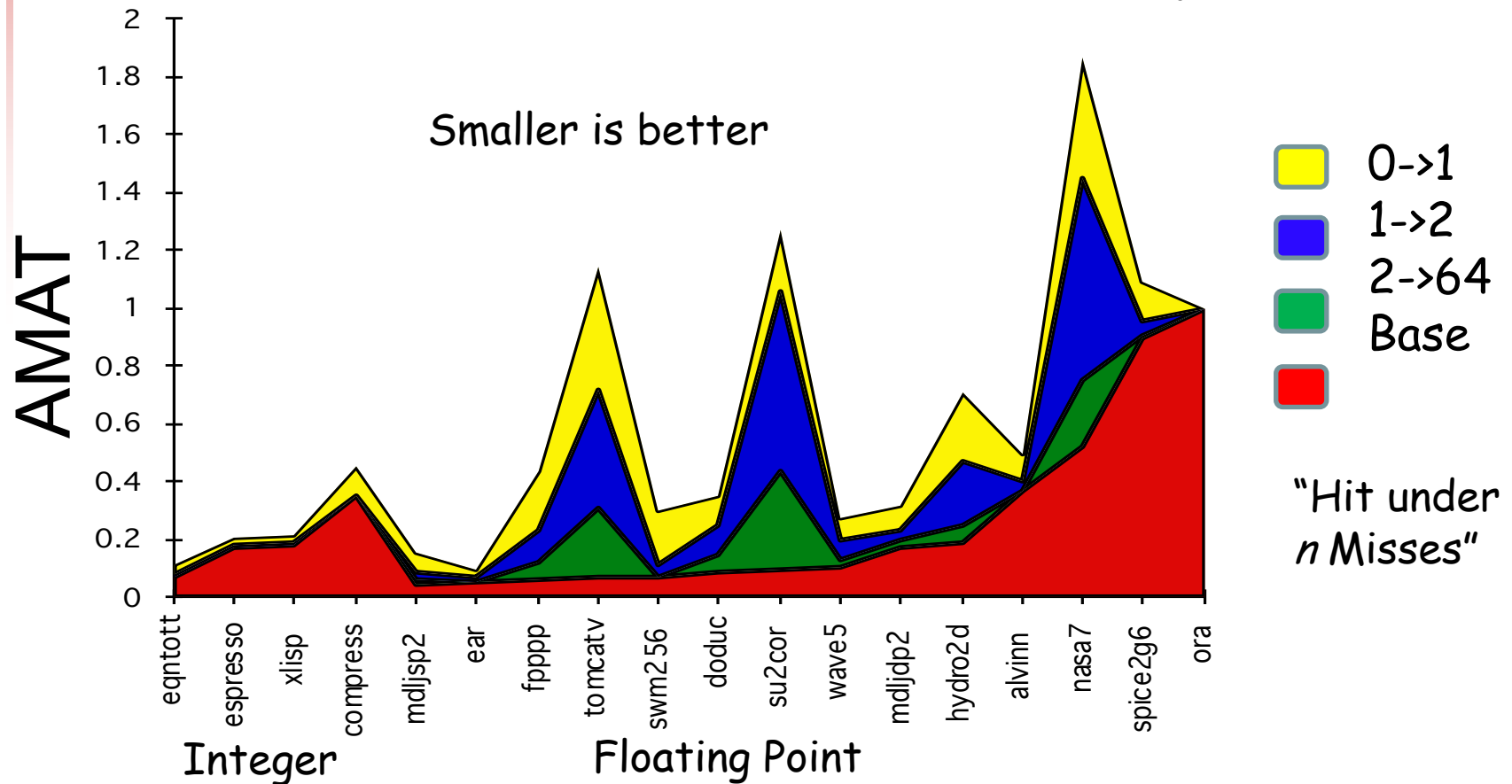
# Τεχνικές μείωσης του Miss Penalty

## Non-Blocking Caches

- Οι **Non-blocking** ή **lockup-free caches** επιτρέπουν στις data caches να αποστέλλουν δεδομένα που περιέχουν (cache hits) όσο διεκπεραιώνεται ένα miss:
  - Απαιτείται out-of-order εκτέλεση των εντολών από τη CPU.
  - “**hit under miss**” : μειώνει το effective miss penalty γιατί συνεχίζεται η επεξεργασία δεδομένων από τη CPU αντί να αγνοούνται οι αιτήσεις για νέα δεδομένα.
  - “**hit under multiple miss**” ή “**miss under miss**” : μπορεί να προσφέρει επιπλέον μείωση του effective miss penalty by επικαλύπτοντας τα πολλαπλά misses.
  - Αυξάνεται σημαντικά η πολυπλοκότητα του cache controller αφού μπορεί να υπάρχουν πολλές μη διεκπεραιωμένες προσπελάσεις στη μνήμη.
  - Απαιτεί πολλαπλά memory banks ώστε να εξυπηρετούνται πολλαπλές προσπελάσεις στη μνήμη.
  - Παράδειγμα: Intel Pentium Pro/III επιτρέπει να εκκρεμούν μέχρι και 4 misses.

# Τεχνικές μείωσης του Miss Penalty Non-Blocking Caches

- FP programs on average: AMAT= 0.68 -> 0.52 -> 0.34 -> 0.26
- Int programs on average: AMAT= 0.24 -> 0.20 -> 0.19 -> 0.19
- 8 KB Data Cache, Direct Mapped, 32B block, 16 cycle miss



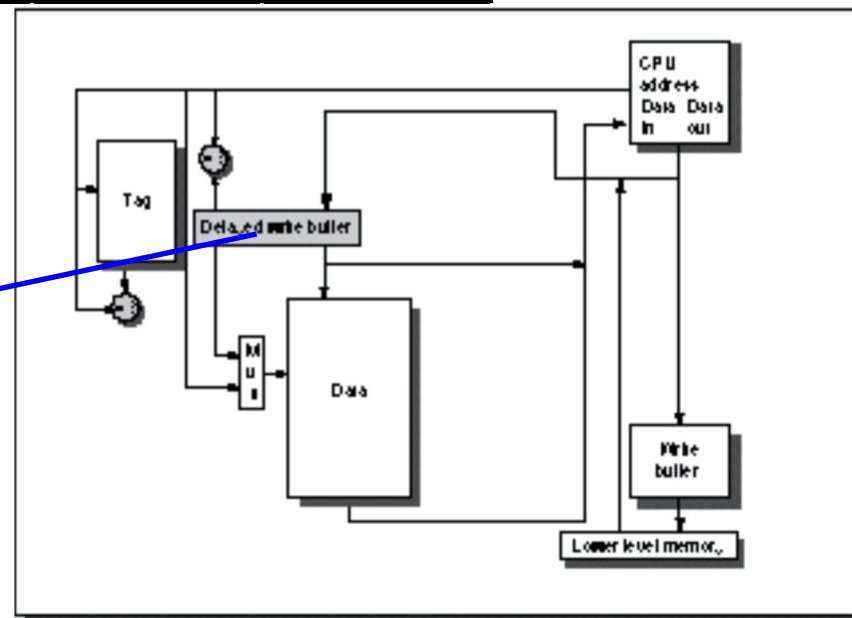
# Τεχνικές μείωσης του Hit Time

## Pipelined Writes

- Ο έλεγχος του tag και η ενημέρωση της cache –από την προηγούμενη εντολή- μπορεί να γίνονται ταυτόχρονα (pipeline) αν υλοποιηθούν ως διαφορετικά στάδια

<b>Instruction</b>	<b>LW</b>	<b>SW</b>	<b>LW???</b>	<b>LW</b>	<b>SW</b>	
<b>Tag Array</b>	Read	Read		Read	Read	
<b>Data Array</b>	Read		Write	Read		Write

<b>Instruction</b>	<i>LW</i>	<i>SW</i>	<i>LW</i>	<i>SW</i>	
<b>Tag Array</b>	Read	Read	Read	Read	
<b>Data Array</b>	Read		Read	Write	Write



- “Delayed Write Buffer”: which must be checked on reads; either complete write or read from buffer

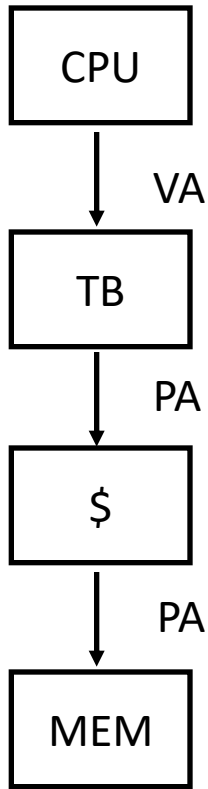
# Τεχνικές μείωσης του Hit Time

## Avoiding Address Translation

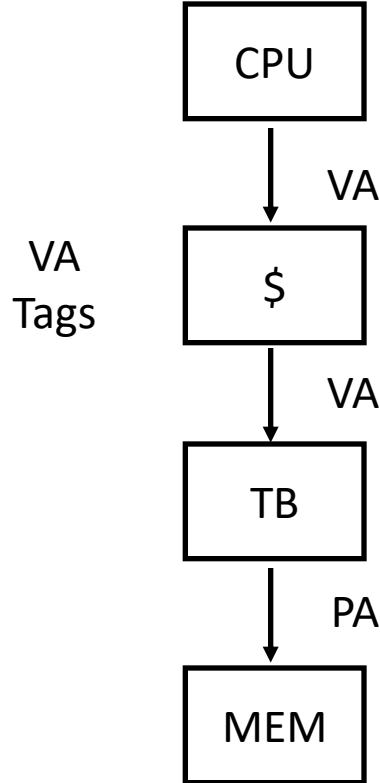
- Αποστολή της virtual address στην cache: Ονομάζεται **Virtually Addressed Cache** ή απλά **Virtual Cache** vs. **Physical Cache**
  - Κάθε φορά που αλλάζουμε διεργασία η cache πρέπει να καθαρίζεται (flushed), διαφορετικά θα επιστρέψει λανθασμένα hits
    - Κόστος : χρόνος flush + “compulsory” misses λόγω του αδειάσματος της cache
  - Χειρισμός των **aliases** (αποκαλούνται και **synonyms**);  
2 διαφορετικές virtual addresses αντιστοιχίζονται στην ίδια physical address
  - I/O πρέπει να επικοινωνεί με την cache, επομένως χρειάζονται οι virtual addresses
- Λύση για τα aliases:
  - Το HW εγγυάται ότι ο συνδυασμός index field & direct mapped είναι μοναδικός : **page coloring**
- Λύση για το cache flush:
  - Προσθέτουμε μία **process identifier tag** η οποία αναγνωρίζει τη διεργασία καθώς και τις διευθύνσεις της διεργασίας: δεν επιστρέφεται hit από λάθος διεργασία

# Τεχνικές μείωσης του Hit Time

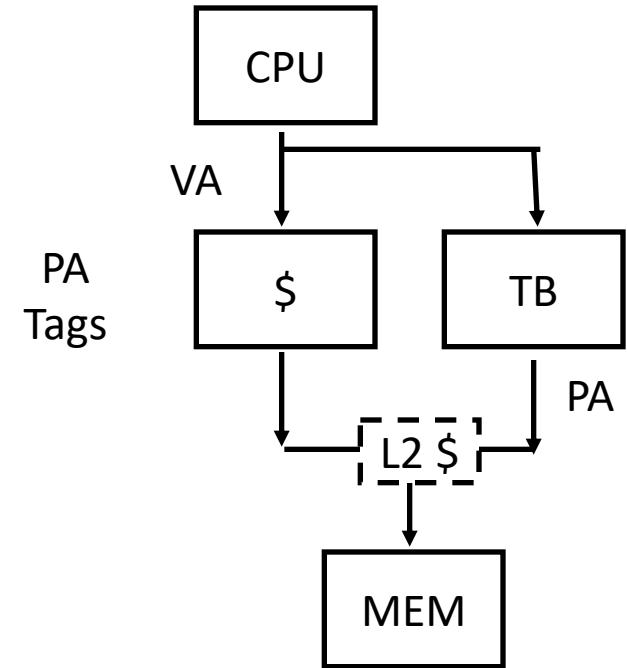
## Virtually Addressed Caches



Συμβατική  
Οργάνωση



Virtually Addressed Cache  
Μετάφραση μόνο σε miss  
Synonym προβλήματα



Επικάλυψη της \$ προσπέλασης με  
VA μετάφραση: Απαιτείται δείκτης  
στην \$ index για να παραμένει  
σταθερό  
κατά τη μετάφραση

# Σύνοψη

	Τεχνική	MR	MP	HT	Complexity
Miss rate	Μεγαλύτερο μέγεθος Block	+	—		0
	Υψηλότερη Associativity	+		—	1
	Victim Caches	+			2
	Pseudo-Associative Caches	+			2
	HW Prefetching of Instr/Data	+			2
	Compiler Controlled Prefetching	+			3
	Compiler Reduce Misses	+			0
Miss Penalty	Προτεραιότητα στα Read Misses		+		1
	Subblock Placement		+	+	1
	Early Restart & Critical Word 1st		+		2
	Non-Blocking Caches		+		3
	Second Level Caches		+		2
Hit time	Small & Simple Caches	—		+	0
	Avoiding Address Translation			+	2
	Pipelining Writes			+	1