Θέματα Σχεδίασης Κρυφών Μνημών

[Up - Table of Contents]
[Prev - 14. Processor Performance]
[printer version - PDF]
[16. Virtual Memory - Next]

15.6: Set-associative κρυφή μνήμη

Στο μάθημα είδαμε ότι η κρυφή μνήμη με απ'ευθείας (μονοσήμαντη) απεικόνιση επιφέρει συγκρούσεις μεταξύ διευθύνσεων που τυχαίνει να απεικονίζονται στο ίδιο block της κρυφής μνήμης. Αυτές οι συγκρούσεις είναι ανεπιθύμητες, ειδικά εάν η κρυφή μνήμη έχει χώρο για εισερχόμενα δεδομένα (μία ή περισσότερες γραμμές είναι invalid) αλλά τα εισερχόμενα δεδομένα καταλήγουν να εκδιώξουν (αντικαταστήσουν) άλλα έγκυρα (valid) και πιθανόν χρήσιμα στο πρόγραμμα δεδομένα. Μία πλήρως προσεταιριστική (fully associative) κρυφή μνήμη επιλύει αυτό το πρόβλημα, το κόστος υλοποίησής της όμως είναι πολύ υψηλό για ρεαλιστικά μεγέθη κρυφής μνήμης (πχ. 64K).

Μια ενδιάμεση λύση με χαμηλότερο κόστος από την πλήρως προσεταιριστική κρυφή μνήμη είναι μία "μερικώς προσεταιριστική" κρυφή μνήμη (set associative cache). Σε αυτή την περίπτωση η κρυφή μνήμη οργανώνεται σε περισσότερες από μία στήλες (τυπικά από 2 έως 8 στήλες σε σύγχρονους επεξεργαστές). Οι στήλες αυτές ονομάζονται συχνά και "ways" ή δρόμοι. Η απεικόνιση διευθύνσεων σε γραμμές της κρυφής μνήμης είναι παρόμοια την μονοσήμαντη απεικόνιση. Ωστόσο, κάθε γραμμή της κρυφής μνήμης περιέχει τώρα ένα σύνολο (set) από blocks (τόσα όσες και οι στήλες) και ο επεξεργαστής μπορεί να "επιλέξει" το block (δηλάδη τη στήλη) στο οποίο θα απεικονίσει μία εισερχόμενη διεύθυνση. Για παράδειγμα, εάν το σύνολο έχει άκυρα blocks ένα από αυτά μπορεί να χρησιμοποιηθεί για την απεικόνιση της εισερχόμενης διεύθυνσης χωρίς να εκδιωχθούν άλλα έγκυρα (valid) blocks που υπάρχουν ήδη στο σύνολο. Μπορεί να πει κανείς ισοδύναμα, ότι το σύνολο blocks κάθε γραμμής της κρυφής μνήμης αντιμετωπίζεται σαν μία μικρή πλήρως προσεταιριστική κρυφή μνήμη για να αποφεύγονται ανεπιθύμητες συγκρούσεις. Φυσικά οι συγκρούσεις δεν μπορούν να αποφευχθούν πάντα, εφόσον σε κάθε σύνολο θα απεικονίζονται έτσι ή αλλιώς πολύ περισσότερες διευθύνσεις από το πλήθος των blocks του συνόλου.

Παράδειγμα: Ας υποθέσουμε ότι έχουμε μια κρυφή μνήμη μεγέθους 16 KB και το μέγεθος του block είναι 32 bytes, κατά συνέπεια τα 5 λιγότερο σημαντικά bits της διεύθυνσης δίνουν το block offset. Η μνήμη έχει 512 blocks. Αν η μνήμη χρησιμοποιούσε μονοσήμαντη απεικόνιση, τα επόμενα 9 bits (bit 6 έως bit 14)

της διεύθυνσης θα χρησιμοποιούνταν σαν δείκτης του block. Ας υποθέσουμε ότι η κρυφή μνήμη χρησιμοποιεί μερικώς προσεταιριστική απεικόνιση με 2 σύνολα. Τα blocks χωρίζονται σε 2 στήλες των 256 blocks και χρησιμοποιούνται 8 (αντί για 9 στη μονοσήμαντη απεικόνιση), δηλ. τα bits 6 έως 13, για να δεικτοδοτήσουν ένα σύνολο των 2 blocks. Η ταυτοποίηση της διεύθυνσης γίνεται με έλεγχο των tags και των 2 blocks του συνόλου. Εάν η ταυτοποίηση αποτύχει ο επεξεργαστής επιλέγει ένα από τα 2 blocks του συνόλου για να απεικονίσει τη διεύθυνση, αντικαθιστώντας το block εάν αυτό είναι valid.

Η set associative κρυφή μνήμη τυπικά μειώνει σημαντικά το ποσοστό αστοχιών (miss rate) λόγω αποφυγής συγκρούσεων, χάρη στην περισσότερο ευέλικτη απεικόνιση διευθύνσεων στην κρυφή μνήμη. Πειραματικά, έχει δειχθεί ότι το ποσοστό αστοχίας μίας 2-way associative κρυφής μνήμης είναι ίσο με το ποσοστό αστοχίας μίας κρυφής μνήμης μονοσήμαντης απεικόνισης διπλάσιου μεγέθους. Ωστόσο, η μείωση μόνο του ποσοστού αστοχίας μίας κρυφής δεν σημαίνει απαραίτητα και βελτίωση της επίδοσης του επεξεργαστή! Όπως μάθατε στο μάθημα, η επίδοση μιας ιεραρχίας με κρυφή μνήμη εξαρτάται από 3 παράγοντες: το ποσοστό αστοχίας, την καθυστέρηση της ευστοχίας και την καθυστέρηση της αστοχίας. Στην περίπτωση των set associative κρυφών μνημών, όταν υπάρχει ευστοχία (δηλ. ο επεξεργαστής απεικονίσει τη διεύθυνση στο σύνολο, ταιριάξει την ετικέτα tag της διεύθυνσης με μία από τις αποθηκευμένες ετικέττες του συνόλου, και ελέγξει ότι η αποθηκευμένη διεύθυνση είναι valid), ο επιλογέας της γραμμής επιλέγει ένα σύνολο blocks, κατά συνέπεια απαιτείται επιπλέον υλικό για να διαβαστεί το συγκεκριμένο block που αναζητεί ο επεξεργαστής. Αυτό μπορεί να γίνει με χρήση πολυπλέκτη N:1 όπου N το πλήθος των blocks κάθε συνόλου. Η χρήση του πολυπλέκτη επιφέρει επιπλέον καθυστέρηση στην ανάγνωση του σωστού block, η οποία δεν υπάρχει στην περίπτωση κρυφής μνήμης με μονοσήμαντη απεικόνιση. Μάλιστα, όσο περισσότερα blocks υπάρχουν σε ένα σύνολο, τόσο μεγαλύτερη είναι η καθυστέρηση. Κατά συνέπεια ο χρόνος ευστοχίας σε μία κρυφή μνήμη που είναι set associative είναι μεγαλύτερος από το χρόνο ευστοχίας σε μία κρυφή μνήμη που είναι direct mapped, και αυξάνεται (με ρυθμό περίπου 2% για κάθε στήλη της κρυφής μνήμης). Ο μόνος τρόπος να συμπεράνουμε εάν μία set-associative κρυφή μνήμη επιτυγχάνει καλύτερη επίδοση από μία direct-mapped κρυφή μνήμη είναι να υπολογίσουμε το μέσο χρόνο πρόσβαση στη μνήμη (average memory access time = hit_time + miss_rate * miss_penalty).

15.7: Πολιτικές Αντικατάστασης

Σε κάθε σύστημα μνήμης (κρυφή μνήμη, RAM, ...) στο οποίο υπάρχει επιλογή ως προς την απεικόνιση μιας διεύθυνσης (δηλαδή η απεικόνιση δεν είναι μονοσήμαντη), και όταν όλες οι δυνατές επιλογές για την απεικόνιση απεικονίζουν ήδη νόμιμες (valid) διευθύνσεις, προκύπτει το πρόβλημα της αντικατάστασης μίας νόμιμης διεύθυνσης που είναι ήδη παρούσα στη μνήμη.

Στην περίπτωση των πλήρως προσεταιριστικών κρυφών μνημών για παράδειγμα, εάν μια διεύθυνση του επεξεργαστή δεν απεικονίζεται σε κανένα block και όλα τα blocks είναι νόμιμα (κατάσταση valid) ο επεξεργαστής είναι υποχρεωμένος να αντικαταστήσει ένα από τα νόμιμα blocks. Ο θεωρητικά βέλτιστος αλγόριθμος αντικατάστασης εάν δεν γνωρίζουμε τις μελλοντικές διευθύνσεις που θα παράγει ο επεξεργαστής είναι ο αλγόριθμος Least Recently Used (LRU) ο οποίος αντικαθιστά το block το οποίο προσπελάστηκε λιγότερο πρόσφατα από τον επεξεργαστή και κατά συνέπεια αναμένεται να μην προσπελαστεί σύντομα ή και καθόλου στο μέλλον. Η υλοποίηση της LRU σε περίπτωση επιλογής μεταξύ 2 blocks είναι σχετικά απλή στο υλικό. Μπορεί να γίνει με χρήση ενός bit το οποίο θα δείχνει εάν το πρώτο block (0) ή το δεύτερο block (1) είναι το λιγότερο πρόσφατα χρησιμοποιημένο από τον επεξεργαστή. Δυστυχώς η υλοποίηση του αλγόριθμου LRU για σύνολα 4, 8, ... blocks δεν είναι οικονομική σε υλικό, ενώ ακόμα και σε λογισμικό η υλοποίηση για μεγάλες μνήμες μπορεί να έχει απαγορευτικό κόστος. Πρακτικά, τα συστήματα μνήμης χρησιμοποιούν ικανοποιητικές "προσεγγίσεις" της LRU, όπου ένα η περισσότερα bits αναφοράς (reference bits) μετρούν προσεγγιστικά το αν ένα block προσπελάστηκε στο πρόσφατο παρελθόν και κατά συνέπεια δεν είναι καλός υποψήφιος για αντικατάσταση. Το σύστημα περιοδικά μηδενίζει τα bits αναφοράς (δηλαδή την παλαιότερη ιστορία των προσπελάσεων σε διευθύνσεις μνήμης), ώστε η πληροφορία που καταγράφεται σε αυτά να είναι όσο πιο έγκαιρη γίνεται.

15.8: Πολιτικές Write

Η κρυφή μνήμη περιέχει αντίγραφα δεδομένων που βρίσκονται στην κύρια μνήμη. Εάν ο επεξεργαστής μεταβάλει τα περιεχόμενα μίας διεύθυνσης δεδομένων στην κρυφή μνήμη με μια εντολή store, προκύπτει ασυνέπεια μεταξύ των περιεχομένων του αντιγράφου της διεύθυνσης στην κρυφή μνήμη και των περιεχομένων του αντιγράφου της διεύθυνσης στην κύρια μνήμη. Η ασυνέπεια αυτή επιλύεται με την αντιγραφή στην κύρια μνήμη των περιεχομένων διευθύνσεων που βρίσκονται στην κρυφή μνήμη και τα περιεχόμενά τους έχουν αλλάξει από τη στιγμή που οι διευθύνσεις αυτές απεικονίστηκαν στην κρυφή μνήμη. Οι σύγχρονοι επεξεργαστές χρησιμοποιούν δύο πολιτικές τήρησης της συνέπειας των περιεχομένων των αντιγράφων:

Write through: Στην πολιτική αυτή, όταν ο επεξεργαστής εκτελεί εντολή store και μεταβάλλει τα περιεχόμενα μίας διεύθυνσης στην κρυφή μνήμη, ενημερώνει την ίδια διεύθυνση στην κύρια μνήμη με τα νέα περιεχόμενα. Η πολιτική αυτή προφανώς τηρεί τη συνέπεια σε κάθε εντολή store, αλλά καθυστερεί την εκτέλεση της εντολής store μέχρι να ενημερωθεί η κύρια μνήμη, λειτουργία που μπορεί να απαιτήσεις δεκάδες ή εκατοντάδες κύκλους σε σύγχρονους επεξεργαστές. Με άλλα λόγια η πολιτική write-through μπορεί να αυξήσει το χρόνο ευστοχίας (hit time) στην κρυφή μνήμη όταν εκτελούνται εντολές stores.

Μία λύση σε αυτό το πρόβλημα είναι η ένθεση ενός ενταμιευτή εγγραφών (write buffer) ο οποίος αποθηκεύει τα νέα περιεχόμενα των διευθύνσεων μνήμης στις οποίες γίνονται stores και η ασύγχρονη μεταφορά των δεδομένων του write buffer στην κύρια μνήμη, ενώ ο επεξεργαστής συνεχίζει να εκτελεί εντολές χωρίς να περιμένει να ολοκληρωθούν οι μεταφορές που εκκρεμούν από το write buffer. Θέματα σχεδίασης των write buffers συζητώνται στο μάθημα **HY425 – Αρχιτεκτονική Υπολογιστών**.

Write back: Στην πολιτική αυτή, όταν ο επεξεργαστής εκτελεί εντολή store μεταβάλλει τα περιεχόμενα της διεύθυνσης μόνο στην κρυφή μνήμη. Τα νέα περιεχόμενα της διεύθυνσης προωθούνται στην κύρια μνήμη μόνο όταν η διεύθυνση αυτή αντικατασταθεί από την κρυφή μνήμη, λόγω σύγκρουσης με άλλη διεύθυνσης ή/και της πολιτικής αντικατάστασης που χρησιμοποιεί η κρυφή μνήμη. Σε αυτή την περίπτωση, τα δεδομένα πρέπει να προωθηθούν στην κύρια μνήμη μόνο εάν το αντίγραφο στην κρυφή μνήμη έχει αλλαγμένα περιεχόμενα λόγω ενός ή περισσοτέρων stores. Για να γνωρίζει ο επεξεργαστής ποια αντίγραφα έχουν περιεχόμενα που έχουν μεταβληθεί από τότε που τα αντίγραφα αυτά απεικονίστηκαν στην κρυφή μνήμη, χρησιμοποιεί ένα bit που ονομάζεται dirty bit. Όταν ο επεξεργαστής εκτελεί store και το store ευστοχήσει στην κρυφή μνήμη, θέτει το dirty bit του block στο οποίο έγινε το store. Αν το block αυτό αργότερα αντικατασταθεί, τα περιεχόμενά του γράφονται στην κύρια μνήμη. Αλλιώς η αντικατάσταση του block γίνεται απλώς γράφοντας πάνω από τα παλαιά του περιεχόμενα στην κρυφή μνήμη. Η πολιτική write back δεν επιφέρει την αργοπορία εγγραφής στην κύρια μνήμη σε κάθε store και μία διεύθυνση μπορεί να διαβαστεί ή γραφτεί πολλές φορές στην κρυφή μνήμη, πριν τα περιεχόμενά της προωθηθούν στην κύρια μνήμη. Ωστόσο, η πολιτική write back μπορεί να μεγαλώσει το χρόνο αστοχίας στην κρυφή μνήμη ως εξής: Όταν ο επεξεργαστής αστοχεί και το block της μνήμης το οποίο αντικαθίσταται έχει αλλαγμένα περιεχόμενα (δηλ. έχει το dirty bit ίσο με 1), τότε ο επεξεργαστής καθυστερεί όσο περιμένει να γραφούν τα περιεχόμενα του block που αντικαθίσταται στην κύρια μνήμη.

Write allocate/no-allocate: Όταν ο επεξεργαστής εκτελεί μια εντολή store και αστοχήσει στην κρυφή μνήμη έχει δύο επιλογές: Ο επεξεργαστής μπορεί να γράψει τα νέα περιεχόμενα της διεύθυνσης στην κύρια μνήμη χωρίς να απεικονίσει τη διεύθυνση αυτή στην κρυφή μνήμη. Η πολιτική αυτή ονομάζεται write no-allocate, δηλαδή ο επεξεργαστής δεν δεσμεύει χώρο στην κρυφή μνήμη για διευθύνσεις στις οποίες γράφει και οι οποίες δεν βρίσκονται ήδη στην κοινή μνήμη. Η εναλλακτική επιλογή είναι ο επεξεργαστής να δημιουργήσει αντίγραφο της διεύθυνσης στην κρυφή μνήμη και ονομάζεται πολιτική write allocate. Η write allocate αποδίδει καλύτερα εάν ο επεξεργαστής πρόκειται να διαβάσει σύντομα τα δεδομένα τα οποία έγραψε, οπότε και θα ευστοχήσει διαβάζοντας τα δεδομένα από την κρυφή μνήμη. Η write no allocate αποδίδει καλύτερα εάν ο επεξεργαστής δεν πρόκειται να χρησιμοποιήσει τα δεδομένα τα

οποία έγραψε (π.χ. σε μια εφαρμογή κωδικοποίησης δεδομένων ο επεξεργαστής παράγει σαν έξοδο μια ακολουθία χαρακτήρων κάθε ένας από τους οποίους γράφεται μία φορά). Συνήθως η πολιτική write no allocate συνδυάζεται με την πολιτική write through, ενώ η πολιτική write allocate με την πολιτική write back. <

Up to the Home Page of CS-225