6ο Εργαστήριο Αρχιτεκτονικής Η/Υ: Κρυφές μνήμες (cache)

Α. Ευθυμίου

Παραδοτέο: Τετάρτη 5 Δεκέμβρη 2018, 23:59

Ο σκοπός αυτής της άσκησης είναι η εμβάθυνση της κατανόησης των κρυφών μνημών.

Στο πρώτο μέρος του εργαστηρίου υπάρχουν κάποια βήματα πειραματισμού και ερωτήσεις που πρέπει να απαντήσετε οι οποίες στηρίζονται στις παρατηρήσεις που θα κάνετε όταν τρέχετε τα «πειράματα» - εκτελέσεις προγραμμάτων assembly στον MARS και σε 2 εργαλεία του. Οι ερωτήσεις είναι σχεδιασμένες για να απαντηθούν με τη σειρά που βρίσκονται στο κείμενο γιατί ακολουθούν τη σειρά των «πειραμάτων». Αν δεν μπορείτε να απαντήσετε σε κάποια, συνεχίστε στην επόμενη. Μη δοκιμάσετε όμως να εξετάσετε μια ερώτηση χωρίς τουλάχιστον να προσπαθήσετε την προηγούμενή της.

Στο 2ο μέρος (30%) θα αλλάξετε ένα πρόγραμμα assembly ώστε να πετυχαίνει καλύτερα ποσοστά ευστοχίας στην κρυφή μνήμη.

Θα πρέπει να έχετε μελετήσει τα μαθήματα για το υποσύστημα μνήμης που αντιστοιχούν στις ενότητες 5.1-5.3 και τμήμα του 5.5 του συγγράμματος των Patterson, Hennessy.

Ξεκινήστε ακολουθώντας το σύνδεσμο https://classroom.github.com/a/m7pEf_Tv ώστε να δημιουργηθεί το αποθετήριο της άσκησης στο GitHub. Κλωνοποιήστε το και συνεχίστε στο τοπικό σας αντίγραφο.

1 Πίνακες 2 διαστάσεων

Το πρόγραμμα χρησιμοποιεί ένα πίνακα 2 διαστάσεων και αυξάνει κάθε στοιχείο του κατά 1. Επειδή οι πίνακες αποθηκεύονται στην μνήμη του υπολογιστή που έχει μία μόνο διάσταση, χρησιμοποιείται η λεγόμενη row-major διάταξη (order): πρώτα αποθηκεύονται τα στοιχεία της πρώτης γραμμής από αριστερά προς τα δεξιά, μετά της δεύτερης, κ.ο.κ. Σε αυτή την διάταξη, η διεύθυνση του `A[i][j]` (το i είναι αριθμός γραμμής-σειράς και το j στήλης), γνωρίζοντας ότι κάθε στοιχείο του πίνακα είναι 4 bytes, ότι το πρώτο στοιχείο (`A[0][0]`) βρίσκεται στη διεύθυνση A και ότι ο αριθμός των στηλών είναι c, είναι $A + (i * c + j) * 4$. Δοκιμάστε το: για i=j=0, η διεύθυνση, σύμφωνα με την παραπάνω έκφραση, είναι A, για i=0, j=1, η διεύθυνση είναι $A + 4$, και για i=1, j=0, η διεύθυνση είναι $A + c * 4$.

Παρατηρήστε επίσης ότι στη δήλωση της ετικέτας data του πίνακα χρησιμοποιείται η οδηγία `.word 0 : 256` που αρχικοποιεί τη μνήμη με την τιμή 0, για 256 (=32x16) λέξεις των 32 bit. Επομένως υπάρχει χώρος για ένα πίνακα μεγέθους μέχρι και 16 x 16.

2 Μέρος 1: Πειραματισμός με προσομοιωτή κρυφής μνήμης

Θα χρησιμοποιήσετε τον MARS και κυρίως το "Data Cache Simulator" που βρίσκεται κάτω από το μενού Tools. Χρήσιμο θα σας φανεί και το "Memory Reference Visualization" tool. Το πρώτο είναι ένας προσομοιωτής κρυφής μνήμης δεδομένων (όχι μνήμης εντολών). Μπορεί κανείς να ορίσει τον τρόπο οργάνωσης και τις βασικές παραμέτρους μιας κρυφής μνήμης και να δεί οπτικά πως γίνονται οι προσπελάσεις στις γραμμές της καθώς και να πάρει πληροφορίες σχετικά με τον αριθμό και το ποσοστό ευστοχιών. Το δεύτερο εργαλείο δείχνει ένα «χάρτη» μιας περιοχής μνήμης του MIPS και σε κάθε προσπέλαση αλλάζει το χρώμα της αντίστοιχης λέξης. Έτσι μπορεί να δεί κανείς αν υπάρχει κάποιο μοτίβο στις διευθύνσεις που προσπελαύνονται κάθε φορά.

Σε όλα τα εργαλεία του MARS πρέπει να πατηθεί το κουμπί "Connect to MIPS" για να αρχίσουν τα εργαλεία να «βλέπουν» τις εντολές του MIPS. Μπορεί κανείς να αλλάξει το πρόγραμμα, να φορτώσει άλλο, κλπ. χωρίς να κλείσει τα εργαλεία (ή να τα αποσυνδέσει με τον MIPS). Για να καθαριστούν οι προηγούμενες τιμές - πληροφορίες των εργαλείων, υπάρχει το κουμπί "Reset".

Φορτώστε το πρόγραμμα cache.asm στον MARS και παρατηρείστε τον κώδικά του. Είναι ένα απλό πρόγραμμα που διατρέχει έναν πίνακα δύο διαστάσεων κατά στήλες και αλλάζει κάθε λέξη του. Υπάρχουν 3 «παράμετροι» που μπορούν να διαφοροποιήσουν το πρόγραμμα:

- Ο αριθμός στηλών του πίνακα σε λέξεις (32 bit) - στον καταχωρητή `$a1`. Ο αριθμός θα πρέπει να είναι δύναμη του 2 για να αποφευχθεί η πράξη του πολλαπλασιασμού για τον υπολογισμό των διευθύνσεων.
- Ο λογάριθμος με βάση 2 του αριθμού στηλών - στον καταχωρητή `$a0`. Χρησιμοποιείται για να υλοποιήσει τον πολλαπλασιασμό, που χρειάζεται στον υπολογισμό διευθύνσεων, με ολίσθηση.
- Ο αριθμός γραμμών-σειρών του πίνακα σε λέξεις (32 bit) - στον καταχωρητή `$a2`.

Αρχικά, το μέγεθος του πίνακα είναι 4x4. Περάστε το πρόγραμμα από τον assembler και ξεκινήστε τον Data Cache simulator.

Διαλέξτε προσεκτικά την οργάνωση: **Fully Associative**, **LRU**, **Number of blocks 16**, **cache block size (words) 1**. Το συνολικό μέγεθος της κρυφής μνήμης θα είναι 64 bytes και το Set size (blocks) 16. Επιλέξτε το enabled στο Runtime Log για να βλέπετε τη διεύθυνση στην οποία γίνεται προσπέλαση και αν προκαλεί ευστοχία ή αστοχία. **Πατήστε το "Connect to MIPS"**.

Ξεκινήστε και το Memory Reference Visualizer για να παρατηρείτε πως κάνει προσπελάσεις μνήμης το πρόγραμμα. Διαλέξτε τις τιμές κατάλληλα ώστε να φαίνεται μόνο ο πίνακας. Μη ξεχάσετε να πατήσετε το "Connect to MIPS". Αναδιατάξτε τα παράθυρα ώστε να βλέπετε όσο περισσότερες πληροφορίες μπορείτε. Αν δεν έχετε μεγάλη οθόνη ίσως να χρειαστεί να μετακινείτε τα παράθυρα εργαλείων για να μπορείτε να δείτε τις τιμές των καταχωρητών ή το πρόγραμμα που εκτελείται (text segment στο tab execute).

Βάλτε ένα breakpoint στην εντολή lw που διαβάζει ένα στοιχείο του πίνακα. Πατήστε το Go (F5) που τρέχει το πρόγραμμα μέχρι το τέλος ή το πρώτο breakpoint. Όταν η εκτέλεση φτάσει στο breakpoint τρέξτε την lw με εκτέλεση εντολή προς εντολή (Step - F7). Παρατηρήστε τι συμβαίνει στα 2 εργαλεία. Η lw αστοχεί στην κρυφή μνήμη και θα δείτε κόκκινο χρώμα στην πρώτη γραμμή του Cache block table (το δεξί μέρος στο κέντρο του Data Cache Simulator). Συνεχίστε με εκτέλεση εντολή-προς-εντολή και δείτε τι κάνει η εντολή sw. Η sw ευστοχεί στη κρυφή μνήμη: φαίνεται με πράσινο χρώμα στην ίδια, πρώτη γραμμή του Cache block table. Η ευστοχία οφείλεται στην *χρονική τοπικότητα αναφορών μνήμης* (temporal locality): η sw γράφει στην ίδια διεύθυνση με την lw της ίδιας επανάληψης. Συνεχίστε την εκτέλεση με Go (F5) μέχρι την lw της επόμενης επανάληψης και πολλαπλά Step (F7) μέχρι και την sw που ακολουθεί. Κάντε το μέχρι να καταλάβετε το μοτίβο των προσπελάσεων μνήμης και μετά αφαιρέστε το breakpoint και ολοκληρώστε την εκτέλεση.

Ερώτηση 1: Ποιά είναι η ακολουθία διευθύνσεων που παράγει ο επεξεργαστής; (τις 3-4 πρώτες διευθύνσεις και μετά, αν γίνεται κάποια αλλαγή, αναφερετέ την) Είναι σημαντικό να είστε ακριβείς π.χ. αναφέρετε αν η ίδια διεύθυνση επαναλαμβάνεται και τί είδους προσπέλαση γίνεται (load, store).

Ερώτηση 2: Ποιό είναι το μοτίβο των διευθύνσεων; Δηλαδή τί κοινό φαίνεται να έχουν οι διευθύνσεις που προσπελαύνονται; Π.χ. συνεχόμενες διευθύνσεις, 10 φορές η ίδια διεύθυνση και μετά 10 φορές η διευθυνση+4, κ.ο.κ.

Ερώτηση 3: Ποιό είναι το μοτίβο των προσπελάσεων σε θέσεις της κρυφής μνήμης; Παρόμοια με την προηγούμενη ερώτηση μόνο που τώρα εξετάζετε αν υπάρχει κάτι αντίστοιχο στις γραμμές της κρυφής μνήμης.

Τις παραπάνω 3 ερωτήσεις είναι χρήσιμο να τις σκέφτεστε σε κάθε μετέπειτα αλλαγή της οργάνωσης κρυφής μνήμης ή του προγράμματος. Με λίγη εμπειρία θα μπορείτε να τις απαντάτε κοιτώντας τον κώδικα, χωρίς να χρησιμοποιείτε τον Data Cache Simulator.

Ερώτηση 4: Ποιό είναι το τελικό ποσοστό ευστοχίας (hit rate) του προγράμματος;

Ερώτηση 5: Αλλάζοντας μόνο τον αριθμό γραμμών (cache blocks) της κρυφής μνήμης (που αυξάνει το συνολικό μέγεθός της), μπορεί το ίδιο ακριβώς πρόγραμμα να πετύχει μεγαλύτερο ποσοστό ευστοχιών από αυτό που βλέπετε; Γιατί;

Αλλάξτε την κρυφή μνήμη ώστε να έχει 4 γραμμές των 4 λέξεων: number of blocks = 4, cache block size = 4. Το συνολικό μέγεθος της κρυφής μνήμης παραμένει ίδιο, 64 bytes. Μηδενίστε (reset) τα εργαλεία και τρέξτε ολόκληρο το πρόγραμμα.

Ερώτηση 6: Ποιό είναι το νέο ποσοστό ευστοχίας; Γιατί έγινε τόσο υψηλό παρόλο που ο αποθηκευτικός χώρος της κρυφής μνήμης δεν άλλαξε;

Αλλάξτε το μέγεθος του πίνακα σε 4x8 (βάζοντας 8 αντί για 4 στην addi της a2) κρατώντας την κρυφή μνήμη ίδια.

Ερώτηση 7: Ποιό είναι το νέο μοτίβο των προσπελάσεων;

Ερώτηση 8: Ποιό είναι το νέο ποσοστό ευστοχίας; Σε τί οφείλεται η μείωση του ποσοστού ευστοχίας; Η απάντηση θα πρέπει να αναφέρεται και σε ιδιότητες του προγράμματος και της κρυφής μνήμης.

Ερώτηση 9: Αν ο αριθμός γραμμών της κρυφής μνήμης γινόταν 1 (συνολικό μέγεθος κρυφής μνήμης 16 bytes), ποιό θα ήταν το ποσοστό ευστοχίας;

2.1 Direct mapped cache

Με τα προηγούμενα πειράματα είδατε πως η κρυφή μνήμη εκμεταλεύεται την χρονική και χωρική τοπικότητα των αναφορών ενός προγράμματος. Αλλά η οργάνωση fully associative είναι ακριβή σε υλικό. Εδώ θα εξετάσετε την «οικονομικότερη» σε υλικό, direct mapped, οργάνωση. Σε αυτήν κάθε γραμμή είναι και ξεχωριστό σετ, επομένως ένα τμήμα της διεύθυνσης (το index) καθορίζει σε ποια θέση της κρυφής μνήμης θα τοποθετηθεί μια γραμμή.

Αλλάξτε το Placement Policy στον Data Cache Simulator, σε Direct mapping και κρατήστε στην κρυφή μνήμη τις προηγούμενές παραμέτρους της: 4 γραμμές των 4 λέξεων.

Ερώτηση 10: Συγκρίνετε το ποσοστό ευστοχίας της νέας οργάνωσης, direct mapped, σε σχέση με το αντίστοιχο ποσοστό της fully associative που είχε τον ίδιο αριθμό και μέγεθος γραμμών. Εξηγείστε τι συμβαίνει.

Αλλάξτε το μέγεθος του πίνακα σε 8x4 (βάζοντας 8 αντί για 4 στην addi της a1, 3 σε αυτή του a0 και 4 στην addi της a2) κρατώντας την κρυφή μνήμη ίδια.

Ερώτηση 11: Ποιό είναι το μοτίβο των προσπελάσεων σε θέσεις της κρυφής μνήμης; Περιγράψτε το μοτίβο των αντικαταστάσεων γραμμών της κρυφής μνήμης;

Ερώτηση 12: Ποιό είναι το νέο ποσοστό ευστοχίας;

2.2 Παραδοτέο 1ου μέρους

Το παραδοτέο της άσκησης είναι το συμπληρωμένο με τις απαντήσεις σας απλό αρχείο κειμένου `lab06_answers.txt`. Προσοχή στην κωδικοποίηση των Ελληνικών χαρακτήρων. Θα πρέπει να είναι UTF-8 γιατί κάποιες άλλες κωδικοποιήσεις, κυρίως των Windows, συχνά δεν είναι ορατές. **Απαντήσεις σε Greeklish δεν θα ληφθούν υπόψη.**

Οι απαντήσεις πρέπει να είναι σύντομες, το πολύ 5-6 προτάσεις.

Η παράδοση θα γίνει μέσω GitHub, όπως πάντα.

3 Μέρος 2: Τροποποίηση του προγράμματος

Από τα παραπάνω πειράματα βλέπετε ότι το ποσοστό ευστοχίας του προγράμματος ειδικά σε direct mapped κρυφές μνήμες, δεν είναι καλό. Το κύριο πρόβλημα είναι ότι δεν μπορεί να εκμεταλεύτει τη χωρική τοπικότητα αναφορών.

Ο σκοπός του 2ου μέρους της άσκησης είναι να αλλάξετε το πρόγραμμα ώστε να πετυχαίνει μεγαλύτερα ποσοστά ευστοχίας σε direct mapped κρυφές μνήμες με το ίδιο συνολικό μέγεθος (64bytes) και με μέγεθος γραμμής 4 ή και περισσότερων λέξεων. Δεν μας απασχολεί ο αριθμός εντολών που εκτελούνται παρά μόνο οι προσπελάσεις μνήμης. Επίσης, σε αυτό το μέρος θα χρησιμοποιήσετε μόνο direct mapped κρυφές μνήμες.

Το παραδοτέο του 2ου μέρους είναι το αλλαγμένο πρόγραμμα cache.asm