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

Α. Ευθυμίου

Παραδοτέο: Παρασκευή 18 Δεκέμβρη 2018, 23:59

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

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.

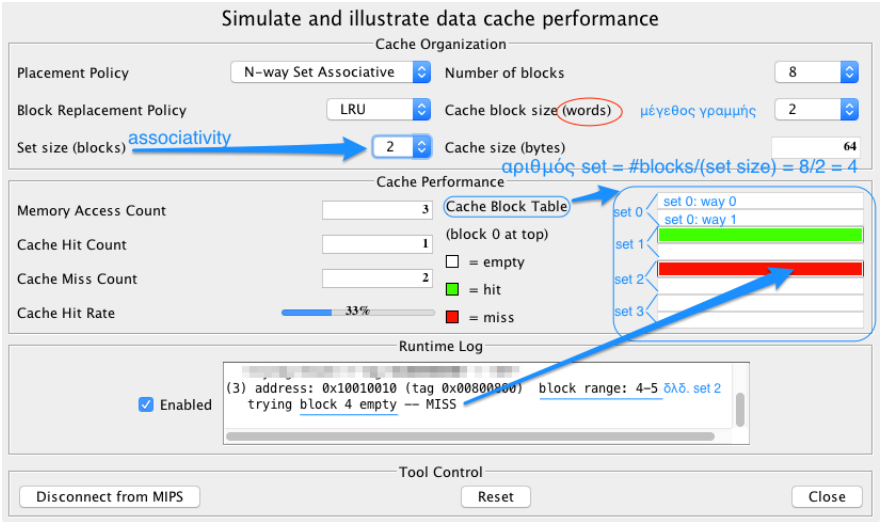
Οι απαντήσεις θα δοθούν σε quiz στο ecourse και δεν θα χρειαστεί να ανεβάσετε αρχεία στο Github. Επομένως, σε αυτή την άσκηση θα χρειαστεί μόνο να πάρετε τον σκελετό της από το Github: https://github.com/UoI-CSE-MYY505/starter_lab06. Δεν θα έχετε το δικό σας ξεχωριστό αποθετήριο.

1 Προσομοιωτής κρυφής μνήμης του Mars

Θα χρησιμοποιήσετε τον MARS και κυρίως το "Data Cache Simulator" (βλ. Σχήμα 1) που βρίσκεται κάτω από το μενού Tools. 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 πως γίνονται οι προσπελάσεις στις γραμμές της καθώς και να πάρει πληροφορίες σχετικά με τον αριθμό και το ποσοστό ευστοχιών.

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 από τον MIPS). Για να καθαριστούν οι προηγούμενες τιμές - πληροφορίες των εργαλείων, υπάρχει το κουμπί "Reset". Αλλιώς, αν δεν αλλάξουν παράμετροι στον προσομοιωτή, συνεχίζει να συγκεντρώνει δεδομένα. Επομένως, **θα χρειαστεί να κάνετε Reset μετά από κάθε "πείραμα"**, σε αυτό το εργαστήριο.

Οι παράμετροι οργάνωσης του cache simulator βρίσκονται στο επάνω μέρος του παραθύρου. Ακο-



Σχήμα 1: Data Cache Simulator.

λουθούν την γνωστή, από τις διαλέξεις, ορολογία εκτός από το "Placement Policy" που στις διαλέξεις αναφέρεται ως *οργάνωση* της κρυφής μνήμης. Προσοχή στην παράμετρο "Set size (blocks)" που είναι ο αριθμός των block ενός set, γνωστός και ως associativity. Αν θέλετε να βρείτε τον αριθμό των set (π.χ. για να υπολογίσετε μετά πόσα bit είναι το index), θα πρέπει να διαιρέσετε το "Number of blocks" με το "Set size". Επίσης προσέξτε ότι το μέγεθος γραμμής, "Cache block size" είναι σε λέξεις όχι bytes. Τέλος, συχνά μπερδεύει κανείς το μέγεθος γραμμής ("Cache block **size**") με τον συνολικό αριθμό γραμμών της cache ("**Number of** blocks").

Οι πληροφορίες στο υπόλοιπο παράθυρο του cache simulator αλλάζουν κατά την εκτέλεση του προγράμματος. Στα δεξιά υπάρχει μια αναπαράσταση της δομής της κρυφής μνήμης, που θα χρησιμοποιήσετε πολύ σε αυτή την άσκηση. Κάθε σειρά αντιστοιχεί σε μία γραμμή κρυφής μνήμης (cache block). Οι λέξεις που περιέχονται στη γραμμή δεν φαίνονται. Το χρώμα δείχνει που έγιναν ευστοχίες ή αστοχίες. Στο σχήμα έχουν επισημανθεί τα set σε μια 2-way set associative οργάνωση, γιατί,δυστυχώς, τα set δεν ξεχωρίζουν στην αναπαράσταση της cache από το εργαλείο. Αν ήταν 4-way, οι τέσσερις πρώτες σειρές/γραμμές θα αντιστοιχούσαν στο set 0, κ.ο.κ. Αν η οργάνωση ήταν direct mapped, κάθε γραμμή θα ήταν και ξεχωριστό set, ενώ αν ήταν fully-associative, θα υπήρχε ένα μόνο set και επομένως όλες οι γραμμές θα ήταν τα ways του set αυτού. Στο κάτω μέρος του παραθύρου υπάρχει το "Runtime Log", που, αν είναι ενεργοποιημένο, δίνει πληροφορίες όπως η διεύθυνση προσπέλασης, το tag της, κ.α.

Μπορεί να σας φανεί χρήσιμο και το "Memory Reference Visualization" tool. 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.

2 Το πρόγραμμα cache.asm

Στην άσκηση θα χρησιμοποιηθεί το cache.asm που εκτελεί τον παρακάτω ψευτοκώδικα.

```
int array[arraySize];  // sizeof(int) == 4 bytes == 1 word
j = repCount;
do {
    // Step through the selected array with the given step size.
    i = 0;
    do {
        if (option == 0)
            // Option 0: One cache access - write
            array[i] = 0;
        else
            // Option 1: Two cache accesses - read AND write
            array[i] = array[i] + 1;
        i += stepSize;
    } while (i < arraySize)
    j--;
} while (j > 0) { // repeat the inner loop repCount times
```

Στις εκτελέσεις του προγράμματος θα γίνονται αλλαγές στις εξής παραμέτρους[1] του.

arraySize - το μέγεθος του πίνακα array σε λέξεις. Αντιστοιχεί στον καταχωρητή s0.

option - Ελέγχει τί θα κάνει ο εσωτερικός βρόγχος: 0 μηδενίζει τα στοιχεία του πίνακα και 1 τα αυξάνει κατά 1. Αντιστοιχεί στον καταχωρητή s1.

stepSize - το βήμα με το οποίο προσπελαύνεται ο πίνακας array. Αντιστοιχεί στον καταχωρητή s2.

repCount - ο αριθμός επαναλήψεων του εσωτερικού βρόγχου. Αντιστοιχεί στον καταχωρητή s3.

[1] Προσοχή να μην συγχέονται με τις παραμέτρους της κρυφής μνήμης!

Στην αρχή του τμήματος δεδομένων (.data), υπάρχει το **label padding** που έχει ως σκοπό να αλλάξει την διεύθυνση από όπου αρχίζει ο πίνακας array. Η οδηγία .space X, που χρησιμοποιείται σε αυτό το label, δεσμεύει X bytes μνήμης. Αρχικά είναι μηδέν, αλλά σε κάποια ερωτήματα θα το αλλάξετε σε 4 ή 8. Με αυτό τον τρόπο το array θα ξεκινά από την διεύθυνση 0x10010000, 0x10010004, 0x10010008, και αυτό θα προκαλεί κάποιες διαφορές στην αντιστοίχιση λέξεων του array σε γραμμές της κρυφής μνήμης.

Το σημαντικό στο cache.asm είναι ο ψευτοκώδικας, οι παράμετροι και το padding. Οι υπόλοιπες λεπτομέρειες του κώδικα δεν θα χρειαστούν στην άσκηση.

2.1 Παρατηρήσεις

Στα παρακάτω πειράματα προσπαθήσετε να κάνετε μια εκτίμηση για τα αποτελέσματα **πριν** τρέξετε την προσομοίωση. Μετά την προσομοίωση, σιγουρευτείτε ότι καταλαβαίνετε **γιατί** βλέπετε ότι βλέπετε.

Αναρωτηθείτε τα παρακάτω σε κάθε πείραμα - αλλαγή παραμέτρων προγράμματος, κρυφής μνήμης:

- Ποιό είναι το μέγεθος γραμμής (cache block);
- Πόσες συνεχόμενες προσπελάσεις "χωράνε" σε μιά γραμμή (παίρνοντας υποψη το stepSize);
- Σε ποιά θέση στην κρυφή μνήμη τοποθετείται μια (ή κάθε) συγκεκριμένη γραμμή;
- Πόσα δεδομένα του προγράμματος χωράνε σε **ολόκληρη** την κρυφή μνήμη;
- Πόσο απέχουν στην μνήμη οι γραμμές που αντιστοιχίζονται στο ίδιο set (και που μπορούν να προκαλέσουν συγκρούσεις);
- Όταν εξετάζετε αν μια προσπέλαση θα ευστοχίσει ή όχι, σκεφτείτε αν έχει ξαναγίνει προσπέλαση σε αυτήν την γραμμή στο παρελθόν. Αν έχει ήδη προσπελαστεί, υπάρχει ακόμα στην κρυφή μνήμη;

3 Μέρος 1: Αντιστοίχιση γραμμών σε θέσεις της cache

> Φορτώστε το cache.asm στον Mars τρέξτε assemble και ανοίξτε το data cache simulator tool. Μή ξεχάσετε το Connect to MIPS. Οι παράμετροι του cache.asm θα πρέπει να είναι: arraySize=32, option=0, stepSize=1, repCount=1 (οι αρχικές τιμές του cache.asm). Το .size στο padding θα πρέπει να είναι 0 (η αρχική τιμή). Στο cache simulator, επιλέξτε direct mapping, αριθμό γραμμών (Number of blocks) 8, μέγεθος γραμμής (cache block size) 2 λέξεις.

Σημαντικό. Αν τρέξετε όλο το πρόγραμμα θα δείτε την τελική κατάσταση της κρυφής μνήμης και τα τελικά αποτελέσματα ευστοχιών - αστοχιών. Είναι καλύτερο να θέσετε breakpoints αμέσως μετά από κάθε εντολή προσπέλασης μνήμης για να βλέπετε τις αλλαγές που προκαλεί στην cache κάθε μία από αυτές. Αυτό γίνεται στο Execute tab του Mars, επιλέγοντας το κουτάκι της στήλης Bkpt στα αριστερά της εντολής που σας ενδιαφέρει. Η εκτέλεση πλέον θα σταματά πρίν εκτελεστεί η "σημαδεμένη" εντολή.

> Εκτελέστε το πρόγραμμα μέχρι την πρώτη προσπέλαση μνήμης και συμπληρώστε τις πληροφορίες που ζητούνται στο quiz (πρώτος πίνακας απαντήσεων) στην γραμμή για padding 0 και την πρώτη προσπέλαση: Αν η κρυφή μνήμη ευστόχησε (Hit ή Miss), το block που προσπελάστηκε (0 το πρώτο μέχρι 7 το τελευταίο) και το tag της γραμμής (φαίνεται στο runtime log του cache simulator). Για το tag χρησιμοποιείστε δεκαεξαδικούς αριθμούς.
>
> Συνεχίστε την εκτέλεση μέχρι και την δεύτερη προσπέλαση μνήμης και συμπληρώστε ξανά τις αντίστοιχες πληροφορίες στο quiz.

Ο σκοπός σας είναι να καταλάβετε σε ποιά γραμμή (cache block) αντιστοιχούν οι λέξεις που προσπελαύνονται, πώς ο προσομοιωτής αποφασίζει σε ποιό block της cache αντιστοιχεί - αποθηκεύεται κάθε

γραμμή της μνήμης, πώς υπολογίζεται το tag από την διεύθυνση. Αν δυσκολεύεστε, χρησιμοποιείστε χαρτί και μολύβι, "σπάστε" την διεύθυνση στα τμήματα tag:index:offset, αφού πρώτα υπολογίσετε πόσα bits θα πρέπει να είναι το καθένα από αυτά. Ο αριθμός που σχηματίζει το index θα πρέπει να είναι ο αριθμός του block στον cache simulator και το tag που βρήκατε θα πρέπει να είναι το ίδιο.

Συνεχίστε μέχρι να αναγνωρίσετε το μοτίβο των προσπελάσεων, αλλά δεν υπάρχουν άλλες ερωτήσεις να συμπληρωθούν στο quiz γι'αυτό το πείραμα.

> Αλλάξτε το padding σε 4, κάντε assemble, reset στον cache simulator, ξαναβάλτε τα breakpoints. Δείτε και καταγράψτε στην δεύτερη γραμμή του πίνακα του quiz ό,τι ζητείται για τις δύο πρώτες προσπελάσεις του προγράμματος.

Οι διαφορές είναι σχετικά μικρές αλλά θα πρέπει να μπορείτε να καταλαβαίνετε γιατί συμβαίνουν.

> Επαναλάβετε με padding 8 αυτή τη φορά και συμπληρώστε το quiz.

> Αλλάξτε την οργάνωση σε Fully Associative, με αλγόριθμο αντικατάστασης (Block Replacement Policy) LRU, και τις υπόλοιπες παραμέτρους ίδιες (αριθμό γραμμών 8, μέγεθος γραμμής 2 λέξεις). Επαναλάβετε τις τρείς παραπάνω παραλλαγές του padding (0, 4, 8). Αυτή τη φορά συμπληρώνετε το δεξί μισό του πίνακα απαντήσεων στο quiz. Μην ξεχνάτε να κάνετε assemble το πρόγραμμα και reset στον cache simulator όταν χρειάζεται.

4 Μέρος 2: Επίδραση μεγέθους γραμμής

> Βάλτε τις παρακάτω τιμές παραμέτρων στο cache.s: arraySize 128, option 1, stepSize 1, repCount 1, και padding 0. Ρυθμίστε τον cache simulator ως εξής: οργάνωση Direct Mapped, αριθμός γραμμών 8, μέγεθος γραμμής 2 λέξεις. Τρέξτε το πρόγραμμα και καταγράψτε το μοτίβο ευστοχιών αστοχιών ως μια ακολουθία από γράμματα M - miss, H - hit. Η απάντησή σας θα πρέπει να είναι το συντομότερο μοτίβο που επαναλαμβάνεται, για παράδειγμα το "HHMHHM" θα πρέπει να γραφεί ως "HHM". Καταγράψτε και το τελικό ποσοστό ευστοχίας του προγράμματος.

Σκεφτείτε πως το μοτίβο μπορεί άμεσα να σας δώσει το τελικό ποστοστό ευστοχίας.

> Αλλάξτε το μέγεθος γραμμής σε 4 λέξεις, αλλά για να διατηρηθεί η συνολική χωρητικότητα της cache ίδια, μειώστε τον αριθμό γραμμών σε 4. Βρείτε και γράψτε το νέο μοτίβο και το ποσοστό ευστοχίας.

> Σκεφτείτε σε ποιό από τα δύο είδη τοπικότητας αναφορών μνήμης οφείλεται κάθε ευστοχία του μοτίβου. Αντιγράψτε το μοτίβο που βρήκατε στο τελευταίο ερώτημα, αλλάζοντας κάθε H (hit) είτε σε T (temporal), για χρονική τοπικότητα αναφοράς, είτε σε S (spatial), για χωρική τοπικότητα. Αφήστε τα M ως είναι.

Σκεφτείτε αν το ποσοστό ευστοχίας θα άλλαζε αν αυξάνατε το repCount με όλες τις υπόλοιπες παραμέτρους προγράμματος και cache αμετάβλητες. Αλλάξτε το cache.s και τρέξτε το για να επιβεβαιώσετε ότι το βρήκατε σωστά. Αν η οργάνωση ήταν Fully Associative και όλοι οι άλλοι παράμετροι έμεναν ίδιοι[2] θα άλλαζε κάτι στο μοτίβο ή στο ποσοστό ευστοχίας; Οι παραπάνω ερωτήσεις δεν χρειάζονται απάντηση στο quiz.

[2] Ο αλγόριθμος αντικατάστασης μπορεί να είναι LRU. Παίζει ρόλο στο συγκεκριμένο πρόγραμμα και cache;

5 Μέρος 3: Επίδραση repCount

Βάλτε τις παρακάτω τιμές παραμέτρων στο cache.s: arraySize 32, option 1, stepSize 4, repCount 1, και padding 0. Ρυθμίστε τον cache simulator ως εξής: οργάνωση Direct Mapped, αριθμός γραμμών 16, μέγεθος γραμμής 2 λέξεις. Καταγράψτε τον αριθμό προσπελάσεων του **εσωτερικού βρόγχου** και το ποσοστό ευστοχίας.

Αλλάξτε το repCount σε 2 και επαναλάβετε το πείραμα. Καταγράψτε το νέο ποσοστό ευστοχίας.

Σκεφτείτε τί θα συμβεί αν συνεχίσετε να αυξάνετε το repCount. Συμπληρώστε το μικρότερο δυνατό μοτίβο που επαναλαμβάνεται, καταγράφοντας και το είδος τοπικότητας αναφορών μνήμης σε κάθε ευστοχία, από την **δεύτερη επανάληψη και έπειτα**. (Όπως παραπάνω, γράψτε το μοτίβο, αλλά στις ευστοχίες, γράψτε S ή T, ενώ τυχόν αστοχίες συμβολίζονται με M.)

Συμπληρώστε στο quiz τον μαθηματικό τύπο που δίνει το ποσοστό αστοχιών σε σχέση με το repCount. Γράψτε το σαν να το γράφατε π.χ. σε Java. Μπορείτε να το επιβεβαιώσετε με μερικά πειράματα, αλλά προσέξτε ότι ο cache simulator στρογγυλοποιεί τα αποτελέσματα που παρουσιάζει στο hit rate και ζητείται το miss rate.

6 Μέρος 4: Επίδραση stepSize και associativity

Βάλτε τις παρακάτω τιμές παραμέτρων στο cache.s: arraySize 128, option 1, stepSize 16, repCount 4, και padding 0. Ρυθμίστε τον cache simulator ως εξής: οργάνωση Direct Mapped, αριθμός γραμμών 16, μέγεθος γραμμής 4 λέξεις. Καταγράψτε τον αριθμό προσπελάσεων όλου του εσωτερικού βρόγχου και το ποσοστό ευστοχίας. Επιπλέον, γράψτε το μοτίβο προσπελάσεων στην μνήμη σε **ολόκληρο τον πρώτο βρόγχο**, αλλά κάθε σε αστοχία θα γράψετε το είδος της αστοχίας που συμβαίνει: Υ-υποχρεωτική, Σ - σύγκρουσης, Χ - Χωρητικότητας και Ε για ευστοχία. Χρησιμοποιούμε Ελληνικούς χαρακτήρες εδώ γιατί στα Αγγλικά όλα τα είδη αστοχιών ξεκινούν με C! **Προσοχή** ενώ προηγουμένως είχε ζητηθεί το συντομότερο δυνατό μοτίβο, εδώ ζητείται πλήρης καταγραφή - χαρακτηρισμός όλων των προσπελάσεων της πρώτης επανάλληψης του εξωτερικού βρόγχου.

Συνεχίστε την προσομοίωση και ολοκληρώστε την **δεύτερη** επανάληψη του εξωτερικού βρόγχου. Σημειώστε το συνολικό προσοστό ευστοχίας μέχρι αυτό το σημείο της εκτέλεσης. Γράψτε το μοτίβο ολόκληρης της **δεύτερης** επανάληψης, όπως και προηγουμένως.

Αν αλλάζατε το repCount σε μεγαλύτερο αριθμό, θα βελτιωνόταν το ποσοστό ευστοχίας;

Αν είχαμε την παράμετρο option του προγράμματος στην τιμή 0, ποιό θα ήταν το ποσοστό ευστοχίας;

Επαναφέρετε τις παραμέτρους του cache.s στις αρχικές του μέρους 3: arraySize 128, option 1, stepSize 16, repCount 4, και padding 0. Ρυθμίστε τον cache simulator ώστε να ακολουθεί την Fully Associative οργάνωση, με όλες τις υπόλοιπες παραμέτρους ίδιες (αριθμός γραμμών 16, μέγεθος γραμμής 4 λέξεις). Τρέξτε τον **πρώτο** βρόγχο και παρατηρείστε πώς τοποθετούνται οι γραμμές δεδομένων στην cache. Καταγράψτε τον αριθμό των θέσεων (γραμμών) της cache που χρησιμοποιούνται, και το ποσοστό ευστοχίας. Το μοτίβο προσπελάσεων (ακολουθία ευστοχιών - αστοχιών και το είδος των αστοχιών) είναι ίδιο με την Direct Mapped οργάνωση για την πρώτη επανάληψη;

Συνεχίστε την προσομοίωση και ολοκληρώστε την **δεύτερη** επανάληψη του εξωτερικού βρόγχου. Σημειώστε το συνολικό προσοστό ευστοχίας μέχρι αυτό το σημείο της εκτέλεσης. Το μοτίβο προσπελάσεων (ακολουθία ευστοχιών αστοχιών και το είδος των αστοχιών) είναι ίδιο με την δεύτερη επανάληψη της Direct Mapped οργάνωσης;

Αν αλλάζατε το repCount σε μεγαλύτερο αριθμό, θα βελτιωνόταν το ποσοστό ευστοχίας;

Βρείτε την μικρότερη associativity (set size στον cache simulator) που να δίνει το ίδιο ποσοστό ευστοχίας με την Fully Associative, χωρίς να αλλάξετε άλλες παραμέτρους ούτε στο πρόγραμμα ούτε στην κρυφή μνήμη.