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

Α. Ευθυμίου

**Παραδοτέο: Τετάρτη 11 Δεκέμβρη 2019, 23:59**

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

Το εργαστήριο αυτό είναι αρκετά διαφορετικό από τα προηγούμενα: υπάρχουν πολλά 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.

Δείτε το παραδοτέο για πληροφορίες σχετικές με το πως θα δώσετε τις απαντήσεις σας.

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.

## 1 Προετοιμασία

Για να ξεκινήσετε ακολουθείστε τον σύνδεσμο https://classroom.github.com/a/X_nda6Ph και κλωνοποιείστε το αποθετήριο που θα δημιουργηθεί.

Θα χρησιμοποιήσετε τον MARS και τα "Data Cache Simulator", "Memory Reference Visualization" που βρίσκονται κάτω από το μενού Tools. Το πρώτο είναι ένας προσομοιωτής κρυφής μνήμης δεδομένων (όχι 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» μια 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".

## 2 Fully Associative Cache

Φορτώστε το πρόγραμμα cache.asm στον MARS και παρατηρείστε τον κώδικά του. Είναι ένα απλό πρόγραμμα που διατρέχει έναν πίνακα και αλλάζει κάθε λέξη του. Υπάρχουν 2 «παράμετροι» που μπορούν να διαφοροποιήσουν το πρόγραμμα:

- Το μέγεθος (size) του πίνακα σε λέξεις (32 bit) - στον καταχωρητή `$s0`. Μπορείτε να δώσετε τιμές από 1 έως το μέγιστο που έχει προβλεφθεί στο τμήμα .data (512 λέξεις).
- Ο αριθμός των επαναλήψεων (repetitions) του εξωτερικού βρόγχου - στον καταχωρητή `$s3`. Οι προσπελάσεις του πίνακα γίνονται στον εσωτερικό βρόγχο του προγράμματος. Μπορείτε να αυξήσετε τον αριθμό των προσπελάσεων μνήμης αλλάζοντας τον αριθμό αυτό.

Αρχικά, το μέγεθος του πίνακα είναι 32 και οι επαναλήψεις 1. Περάστε το πρόγραμμα από τον assembler και ξεκινήστε το Data Cache simulator. Διαλέξτε **προσεκτικά** την οργάνωση: Fully Associative, LRU, Number of blocks 32, cache block size (words) 1. Το συνολικό μέγεθος της κρυφής μνήμης θα είναι 128 bytes και το Set size (blocks) 32. Επιλέξτε το enabled στο Runtime Log για να βλέπετε τη διεύθυνση στην οποία γίνεται προσπέλαση και αν γίνει ευστοχία ή αστοχία. Πατήστε το "Connect to MIPS".

Ξεκινήστε και το Memory Reference Visualizer για να παρατηρείτε πως κάνει προσπελάσεις μνήμης το πρόγραμμα. 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;

**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 θέσεις 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 ποσοστό ευστοχίας (hit rate) του προγράμματος;

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

## 3 Χρονική τοπικότητα αναφορών μνήμης

Αλλάξτε τον αριθμό επαναλήψεων, την τιμή του καταχωρητή $s3, ώστε το πρόγραμμα διατρέχει τον ίδιο πίνακα 2 φορές, κρατώντας τον αρχικό αριθμό γραμμών κρυφής μνήμης, 32. Περάστε το αλλαγμένο πρόγραμμα από τον assembler, και πατήστε το Reset και στα δύο εργαλεία. Βάλτε την ταχύτητα εκτέλεσης (Run speed slider) στο κύριο παράθυρο του Mars στις 15 εντολές ανά δευτερόλεπτο (ή σε όποια τιμή σας επιτρέπει να βλέπετε τις μεταβολές στα 2 εργαλεία με ρυθμό που μπορείτε να ακολουθήσετε). **Μη ξεχνάτε να αρχικοποιείτε (reset) τα εργαλεία σε κάθε αλλαγή του προγράμματος ή της οργάνωσης κρυφής μνήμης!** Τρέξτε ολόκληρο το πρόγραμμα και παρατηρείστε τι γίνεται στη κρυφή μνήμη κατά τη 2η επανάληψη. Μπορείτε να την διακρίνετε από τις τιμές στη μνήμη που θα αλλάζουν σε 2.

**Ερώτηση 6**: Σε τί οφείλεται το μεγαλύτερο ποσοστό ευστοχίας;

**Ερώτηση 7**: Αυξήστε περισσότερο των αριθμό των επαναλήψεων. Γιατί το ποσοστό ευστοχίας αυξάνεται με τον αριθμό επαναλήψεων;

**Ερώτηση 8**: Αν αυξάνατε τον αριθμό γραμμών κρυφής μνήμης σε περισσότερες από 32, κρατώντας όλες τις υπόλοιπες παραμέτρους (μνήμης και προγράμματος), θα πετυχαίνατε μεγαλύτερο ποσοστό ευστοχίας; Εξηγείστε γιατί.

## 4 Αντικατάσταση γραμμών

Βάλτε τον αριθμό επαναλήψεων στο 4 και αλλάξτε τον αριθμό γραμμών κρυφής μνήμης σε 16 (συνολικός χώρος αποθήκευσης 64 bytes). Μηδενίστε (reset) τα εργαλεία και τρέξτε ολόκληρο το πρόγραμμα. Παρατηρήστε τι συμβαίνει και το τελικό ποσοστό ευστοχίας.

**Ερώτηση 9**: Σε τί οφείλεται η μείωση του ποσοστού ευστοχίας; Η απάντηση θα πρέπει να αναφέρεται και σε ιδιότητες του προγράμματος και της κρυφής μνήμης.

**Ερώτηση 10**: Αν ο αριθμός γραμμών της κρυφής μνήμης γινόταν 1, ποιό θα ήταν το ποσοστό ευστοχίας; Εξηγήστε τη σχέση των ποσοστών ευστοχίας σε αυτές τις δύο περιπτώσεις (ερωτήσεις 9, 10).

## 5 Χωρική τοπικότητα αναφορών.

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. Τρέξτε το πρόγραμμα και παρατηρήστε το ποσοστό ευστοχίας.

**Ερώτηση 11**: Γιατί έγινε τόσο υψηλό παρόλο που ο αποθηκευτικός χώρος της κρυφής μνήμης δεν άλλαξε;

**Ερώτηση 12**: Αν αλλάζατε τον αριθμό επαναλήψεων στο 1, θα άλλαζε το ποσοστό ευστοχίας;

## 6 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 αφήστε την κρυφή μνήμη να έχει 4 γραμμές των 4 λέξεων, όπως στο προηγούμενο πείραμα. Χρησιμοποιείστε το cache.asm με αριθμό επαναλήψεων 4.

**Ερώτηση 13**: Συγκρίνετε το ποσοστό ευστοχίας της νέας οργάνωσης, direct mapped, σε σχέση με το αντίστοιχο ποσοστό της fully associative του προηγούμενου πειράματος που είχε την ίδια χωρητικότητα. Εξηγείστε τι συμβαίνει.

Στη συνέχεια θα κάνετε μια μικρή αλλαγή στο πρόγραμμα που αλλάζει τη σειρά διευθύνσεων μνήμης που παράγει. Προσθέστε μια μεταβλητή στην αρχή των δεδομένων, ώστε η αρχή του κώδικα να είναι: Είναι πολύ σημαντικό η νέα μεταβλητή να δηλωθεί πριν το array.

```
   .data
counter:        # ADD THIS LABEL
    .word 0     #  AND THIS DIRECTIVE
array:
.space 2048
```

Επιπλέον αλλάξτε την αρχή του κώδικα, έξω από τους βρόγχους, ώστε η διεύθυνση της νέας μεταβλητής να υπάρχει σε έναν καταχωρητή:

```
    la     $s1, counter  # ADD THIS INSTRUCTION
outerloop:
```

Και μια εντολή μέσα στον εσωτερικό βρόγχο που γράφει κάτι σε αυτή τη μεταβλητή:

```
    addiu  $s5, $s5, 1
    addiu  $s6, $s6, 4
```

```
sw     $s5, 0($s1)     # ADD THIS INSTRUCTION
bne    $s5, $s0, innerloop
```

Βάλτε ένα breakpoint στον εσωτερικό βρόγχο ή μειώστε τη ταχύτητα εκτέλεσης εντολών ώστε να μπορείτε να παρατηρήσετε την επίδραση της νέας sw στην ακολουθία διευθύνσεων του προγράμματος. Βρείτε το μοτίβο προσπελάσεων σε θέσεις (γραμμές-σετ) της κρυφής μνήμης. Θα χρειαστεί να περάσει κάποιος χρόνος για να παρατηρήσετε τι συμβαίνει. Η αλλαγή σε σχέση με το προηγούμενο πείραμα θέλει αρκετές επαναλήψεις μέχρι να κάνει την εμφανισή της.

**Ερώτηση 14**: Ποιό είναι το νέο μοτίβο και ποιό το ποσοστό ευστοχίας; Τι παρατηρείτε;

**Ερώτηση 15**: Ο Data Cache Simulator, δεν έχει επιλογές για πολιτικές εγγραφών (write policies). Από αυτά που παρατηρήσατε στο προηγούμενο πείραμα, μπορείτε να αποφανθείτε ότι είναι write allocate ή no write-allocate;

## 7 Παραδοτέο

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.