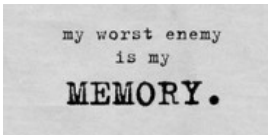
Εθνικο & Καποδιστριακο Πανεπιστημιο Αθηνων
Τμημα Πληροφορικης & Τηλεπικοινωνιων
Μαθημα: **Αρχιτεκτονικη Υπολογιστων II**



Εργασία 2 (*υποχρεωτική*) – Κρυφές Μνήμες

Ακαδημαϊκο Ετος 2016 – 2017

(εκφωνηση) Δευτερα 19 Δεκεμβριου 2016

(παραδοση στο eclass μεχρι) **Τριτη 17 Ιανουαριου 2017**

Επώνυμο	Όνομα	Αριθμός Μητρώου	Email
Αθανασόπουλος	Γεώργιος	1115201300002	sdi1300002@di.uoa.gr
Καρατσενίδης	Κωνσταντίνος	1115201300064	sdi1300064@di.uoa.gr

Πληροφορίες για τις Υποχρεωτικές Εργασίες του μαθήματος

- Οι υποχρεωτικές εργασίες του μαθήματος είναι δύο. Σκοπός τους είναι η πλήρης κατανόηση των εννοιών του μαθήματος με χρήση αρχιτεκτονικών προσομοιωτών (architectural simulators). Η πρώτη υποχρεωτική εργασία αφορούσε τη διοχέτευση (pipelining) και η δεύτερη (αυτή) αφορά τις κρυφές μνήμες (cache memories).
- Η παράδοση των δύο εργασιών του μαθήματος είναι υποχρεωτική για τους φετινούς τριτοετείς φοιτητές καθώς επίσης και για όσους φοιτητές έχουν αριθμό μητρώου 2009 και μεταγενέστερο (δηλαδή πήραν το μάθημα για πρώτη φορά ως τριτοετείς από το εαρινό εξάμηνο του 2012 και μετά). Η βαθμολογία του μαθήματος θα προκύπτει από το γραπτό (60%), την εργασία της διοχέτευσης (20%), και την εργασία των κρυφών μνημών (20%). Καθένας από τους τρεις βαθμούς πρέπει να είναι προβιβάσιμος για να περαστεί προβιβάσιμος βαθμός στη γραμματεία. Οι προαιρετικές ασκήσεις που κατά καιρούς δίνονται δε βαθμολογούνται και δίνονται μόνο για να διευκολυνθεί η κατανόηση του μαθήματος.
- Για τους παλαιότερους φοιτητές (με αριθμό μητρώου 2008 και παλαιότερο) οι δύο εργασίες (pipeline, cache) είναι προαιρετικές. Αν κάποιος παλαιότερος φοιτητής δεν τις παραδώσει θα βαθμολογηθεί με ποσοστό 100% στο γραπτό. Αν τις παραδώσει, θα βαθμολογηθεί με τον παραπάνω τρόπο.
- Κάθε ομάδα μπορεί να αποτελείται **από 1 έως και 3 φοιτητές**. Συμπληρώστε τα στοιχεία όλων των μελών της ομάδας στον παραπάνω πίνακα. Όλα τα μέλη της ομάδας πρέπει να έχουν ισότιμη συμμετοχή και να γνωρίζουν τις λεπτομέρειες της υλοποίησης της ομάδας.
- Για την εξεταστική του Σεπτεμβρίου δε θα δοθούν άλλες εργασίες. Το Σεπτέμβριο θα εξεταστεί μόνο το γραπτό.
- Σε περίπτωση αντιγραφής θα μηδενίζονται όλες οι ομάδες που μετέχουν σε αυτή.
- Η παράδοση της **Εργασίας Κρυφών Μνημών** πρέπει να γίνει μέχρι τα μεσάνυχτα της **Τρίτης 17 Ιανουαρίου 2017** ηλεκτρονικά στο eclass (να ανεβάσετε ένα αρχείο zip ή rar με την τεκμηρίωσή σας σε Word ή PDF και/ή τους κώδικές σας). Μόνο ένα από τα μέλη της ομάδας να ανεβάσει την εργασία στο eclass.

Ζητούμενο

Γράψτε πρόγραμμα assembly MIPS για εκτέλεση στον προσομοιωτή Spim-Cache το οποίο να εκτελεί τον παρακάτω απλό υπολογισμό:

- Επεξεργάζεται έναν τυχαίο τετραγωνικό πίνακα ακεραίων T[i][j] (4 byte κάθε ακέραιος) με διαστάσεις 50x50 (με i,j=0, 1, …, 49) και αποφαίνεται αν ο πίνακας είναι *διαγώνιος*, αν είναι *συμμετρικός* και επίσης βρίσκει τη θέση και την τιμή του μέγιστου στοιχείου του. Στο τέλος εκτυπώνει αποτέλεσμα (π.χ. «Ο πίνακας Τ δεν είναι διαγώνιος, είναι συμμετρικός και το μέγιστο στοιχείο του βρίσκεται στη θέση i=5, j=12 με τιμή 85»).
- *Ο χρόνος εκτέλεσης της εκτύπωσης του αποτελέσματος θα συνυπολογίζεται στο συνολικό χρόνο εκτέλεσης του προγράμματός σας.*

Φυσικά, το πρωταρχικό ζητούμενο είναι το πρόγραμμα να εκτελείται σωστά για οποιοδήποτε περιεχόμενο του πίνακα Τ. Πέρα όμως από την ορθή εκτέλεση πρέπει να προσπαθήσετε το πρόγραμμά σας να εκτελείται καταναλώνοντας τη *μικρότερη δυνατή συνολική ενέργεια* (όχι μέση ενέργεια δηλαδή ισχύ) για τον υπολογισμό. Λάβετε υπ' όψιν σας τα παρακάτω δεδομένα.

- Ο ρυθμός ρολογιού του μικροεπεξεργαστή είναι ίσος με 1GHz.
- Υπάρχει ξεχωριστή κρυφή μνήμη εντολών (instruction cache) και ξεχωριστή κρυφή μνήμη δεδομένων (data cache) δηλαδή Harvard architecture σύμφωνα και με την ορολογία του προσομοιωτή.
- Δεν χρησιμοποιούνται delayed branches ή delayed loads (ρυθμίστε τον προσομοιωτή κατάλληλα).
- Κάθε εντολή που εκτελείται διαρκεί 1 κύκλο ρολογιού αν ευστοχεί (hit) στην κρυφή μνήμη εντολών (instruction cache) και επιβαρύνεται με ποινή 25 επιπλέον κύκλων ρολογιού αν αστοχεί (miss) σε αυτή.

- Εάν η εντολή έχει και αναφορά δεδομένων και ευστοχεί (hit) στη data cache δεν έχει καμία πρόσθετη επιβάρυνση. Εάν όμως η εντολή με αναφορά δεδομένων αστοχήσει (miss) στη data cache επιβαρύνεται με ποινή 25 επιπλέον κύκλων ρολογιού.
- Το μέγεθος των κρυφών μνημών μπορεί να είναι οποιοδήποτε από αυτά που υποστηρίζει ο προσομοιωτής.
- Το μέγεθος του μπλοκ κάθε κρυφής μνήμης μπορεί να είναι οποιοδήποτε από αυτά που υποστηρίζει ο προσομοιωτής.
- Η οργάνωση κάθε κρυφής μνήμης μπορεί να είναι οποιαδήποτε από αυτές που υποστηρίζει ο προσομοιωτής.
- Η πολιτική εγγραφής της κρυφής μνήμης δεδομένων πρέπει να είναι write-back-allocate.
- Ο αλγόριθμος αντικατάστασης πρέπει να είναι LRU και στις δύο κρυφές μνήμες (όταν φυσικά υπάρχει θέμα αντικατάστασης).
- Η τυπική τάση λειτουργίας του μικροεπεξεργαστή είναι 1.2 Volt (ή 1200 mVolt). Έχει όμως τη δυνατότητα (για να καταναλώνει λιγότερη ενέργεια) να λειτουργεί σωστά και σε τάσεις λειτουργίας ίσες με 1200 – k × 10 mVolt με k=1, 2, …, 10. Το μόνο πρόβλημα είναι πως όταν λειτουργεί σε αυτές τις χαμηλότερες τάσεις ο ρυθμός αστοχίας (miss rate) των κρυφών μνημών (τόσο δεδομένων όσο και εντολών) επιβαρύνεται κατά k × 10% σε σχέση με τον ρυθμό αστοχίας όταν η τάση είναι 1200 mVolt. Για παράδειγμα, αν ένα πρόγραμμα έχει 5% ρυθμό αστοχίας στα 1200 mVolt τότε ο ρυθμός αστοχίας του στα 1150 mVolt (k=5) θα είναι 5% × 1.5 = 7.5% ενώ στα 1100 mVolt (k=10) θα είναι 5% × 2 = 10%.

Συμπληρώστε τον παρακάτω πίνακα με τις ρυθμίσεις που επιλέξατε.

Κρυφή μνήμη εντολών: *μέγεθος* *οργάνωση* *μέγεθος μπλοκ*	Κρυφή μνήμη δεδομένων: *μέγεθος* *οργάνωση* *μέγεθος μπλοκ*	Τάση λειτουργίας που επιλέξατε (mVolt) για ελάχιστη κατανάλωση ενέργειας
1024B Fully Assosiative 16B	1024B 2 Ways Set-Assosiative 16B	1200 mVolt

Συμπληρώστε τους παρακάτω πίνακες με τις πληροφορίες του τελικού σας προγράμματος (με βάση τα στατιστικά από τον προσομοιωτή και τα παραπάνω δεδομένα της εργασίας) τόσο για την αρχική τάση λειτουργίας των 1200 mVolt όσο και για αυτή που επιλέξατε για να είναι ελάχιστη η κατανάλωση ενέργειας.

Εκτέλεση στα 1200 mVolt

Συνολικό πλήθος εντολών που εκτελούνται	Συνολικοί κύκλοι ρολογιού για την πλήρη εκτέλεση του προγράμματος	Χρόνος εκτέλεσης του προγράμματος	Ρυθμός αστοχίας κρυφής μνήμης εντολών	Ρυθμός αστοχίας κρυφής μνήμης δεδομένων	Συνολική ενέργεια που καταναλώνεται
8281	25956	25956 ns	0.003985	0.2696	37376.64 J

Εκτέλεση στην τάση λειτουργίας για ελάχιστη κατανάλωση ενέργειας

Συνολικό πλήθος εντολών που εκτελούνται	Συνολικοί κύκλοι ρολογιού για την πλήρη εκτέλεση του προγράμματος	Χρόνος εκτέλεσης του προγράμματος	Ρυθμός αστοχίας κρυφής μνήμης εντολών	Ρυθμός αστοχίας κρυφής μνήμης δεδομένων	Συνολική ενέργεια που καταναλώνεται
8281	25956	25956 ns	0.003985	0.2696	37376.64 J

Τεκμηρίωση

1. Ο αλγόριθμος

Το πρόγραμμα διατρέχει όλο τον πίνακα για να επιλύσει το ζητούμενο της άσκησης.

Κάθε στοιχείο του πίνακα φορτώνεται μόνος μια φορά, για να συγκριθεί με το διαμετρικό του.

Αρχικά το πρόγραμμα διαλέγει το πρώτο στοιχείο του πίνακα ως μέγιστο. Για να εντοπίσουμε την θέση του μεγίστου στοιχείου, επιλέγουμε να κρατάμε τη διεύθυνσή του στη μνήμη καθώς και την τιμή του, σε δύο καταχωρητές, καθ' όλη τη διάρκεια εκτέλεσης του προγράμματος.

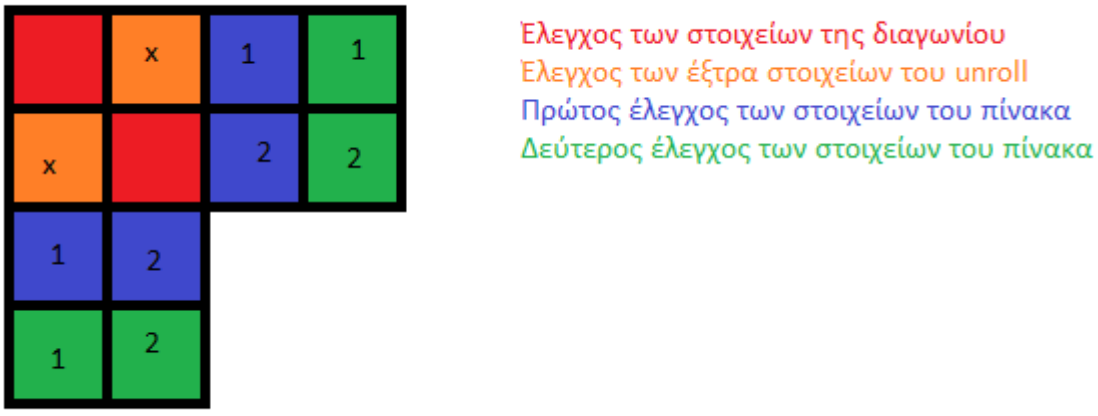
Έπειτα αρχίζει ο έλεγχος για να διαπιστωθεί αν ο πίνακας είναι διαγώνιος. Τα στοιχεία τις διαγωνίου συγκρίνονται με το τρέχον μέγιστο. Τα στοιχεία που είναι μηδέν δεν ελέγχονται με το μέγιστο. Τελικά, αν το μέγιστο παραμείνει αρνητικό, τότε θέτουμε ως μέγιστο στοιχείο το δεύτερο της πρώτης γραμμής.

Σε περίπτωση που διαπιστωθεί ότι ο πίνακας τελικά δεν είναι διαγώνιος, επιτόπου ελέγχεται αν είναι συμμετρικός. Ο έλεγχος αυτός συνεχίζει από το σημείο που έμεινε ο προηγούμενος. Αυτή τη φορά ελέγχεται αν τα διαμετρικά στοιχεία του πίνακα είναι ίσα μεταξύ τους, και έπειτα αν ένα από αυτά τα δύο ξεπερνάει το μέγιστο.

Τελικά, αν ο πίνακας ήταν τυχαίος, δεν ελέγχουμε πλέον για την ισότητα των στοιχείων, αλλά συγκρίνουμε και τα δυο στοιχεία με το μέγιστο. Οπότε και οι τρεις περιπτώσεις κάνουν έναν έλεγχο για τα 50 στοιχεία της διαγωνίου, και δυο ελέγχους για τα υπόλοιπα 1225 ζευγάρια στοιχείων του πίνακα.

Και οι τρεις περιπτώσεις λοιπόν έχουν μια κοινή δομή διπλής επανάληψης, 50 επαναλήψεις εξωτερικά (μια για κάθε στοιχείο της διαγωνίου) και εσωτερικά από το σημείο της διαγωνίου μέχρι το τέλος της γραμμής/στήλης.

Επομένως, επιλέξαμε να κάνουμε unrolls στις επαναλήψεις μας για να μειώσουμε το πλήθος των επαναλήψεων. Αρχικά, σκοπεύαμε να το κάνουμε στην εσωτερική επανάληψη, αλλά υπήρχαν εμπόδια σε αυτό στην στοίχιση των δεδομένων. Οπότε επιλέξαμε να κάνουμε unroll στην εξωτερική επανάληψη με αποτέλεσμα να ελέγχουμε τα κελία του πίνακα όπως φαίνεται στην εικόνα.



Αποφασίσαμε επίσης ότι δεν συμφέρει να κάνουμε unroll και στις δυο επαναλήψεις, αφού τότε ο κώδικας θα αυξανόταν κατά πολύ σε μέγεθος με αποτέλεσμα να μην χωράει όλος στην Instructions Cache.

Στο τέλος του αλγορίθμου ελέγχονται τα τελευταία στοιχεία του πίνακα, δηλαδή τα τελευταία δύο στοιχεία της διαγωνίου και τα δύο γειτονικά τους στοιχεία. Αυτό γίνεται κοινά και για τις τρεις περιπτώσεις.

Επίσης εξετάστηκε η περίπτωση να υπήρχε μόνο μια μορφή επανάληψης και με jr να πηγαίναμε στην κατάλληλη περίπτωση του αλγορίθμου, για να μειωθεί ο αριθμός των Instruction Accesses. Τελικά όμως δεν ήταν καλύτερη αυτή η μορφή αφού εκτελούσαμε 1225 παραπάνω εντολές.

2. H Cache

Το μέγεθος της cache που επιλέξαμε είναι το μέγιστο (1024B) μιας και δεν υπήρχε κάποια ποινή η οποία εξαρτιόταν από αυτό. Έτσι πετύχαμε να χωράνε όσο το δυνατόν περισσότερες εντολές και δεδομένα σ' αυτό.

Το μέγεθος του block είναι και αυτό το μεγαλύτερο δυνατό έτσι ώστε να αποφύγουμε το περισσότερο δυνατόν τα compulsory misses.

Η οργάνωση της μνήμης που χρησιμοποιήσαμε στην cache εντολών είναι Fully Associative, ενώ στη cache δεδομένων 2 Ways Set Associative. Η Fully Associative ωφέλησε γιατί οι εντολές χώρεσαν όλες στην cache και δεν χρειάστηκε να γίνει αντικατάσταση καμίας. Αντίθετα η 2 Ways Set Associative είχε καλύτερη συμπεριφορά στα δεδομένα επειδή τα δεδομένα δεν χωρούσαν όλα στην cache και η αντικαταστάσεις γινόντουσαν πιο σωστά εξαιτίας της σειράς που διαβαζόντουσαν τα δεδομένα.

Στον πρώτο αλγόριθμο που υλοποιήσαμε τα misses ήταν λιγότερα σε σχέση με τον δεύτερο. Όμως στον δεύτερο επικεντρωθήκαμε στο να μειώσουμε όσο το δυνατόν τις εντολές εκτέλεσης και την πρόσβαση στην μνήμη με σκοπό να μειωθούν οι κύκλοι εκτέλεσης. Τελικά ο αλγόριθμος που παραδίδουμε έχει λίγο περισσότερα misses αλλά σημαντική μείωση στους κύκλους εκτέλεσης (1.141 λιγότεροι). Αυτό μας μειώνει τον χρόνο εκτέλεσης και συνεπώς την κατανάλωση ενέργειας.

Τέλος εκτελέσαμε τα πειράματα μειώνοντας την τάση λειτουργείας. Αυτό που παρατηρήσαμε ήταν ότι ενώ η τάση λειτουργίας μειωνόταν η κατανάλωση ενέργειας αυξανόταν. Αυτό οφείλεται στο ότι η αύξηση του ρυθμού αστοχιών ήταν μεγαλύτερη σε σχέση με την μείωση της τάσης. Γι' αυτό ως τάση λειτουργείας επιλέξαμε την 1200 mVolt,

Instruction Cache Accesses:8281 Hits:8248 Hit Rate:0.996015

Console

```
Table T is not diagonal, is symmetric and the max element is in position i=42, j=47 with value 665
```

16	1	1	0	80091	0000000d	00000000	00000001	00000000
17	1	1	0	8008f	00000002	00000000	00000000	00000000
18	1	1	0	8008f	00000001	00000001	00000001	00000000
19	1	0	0	8008f	00000000	00000004	00000000	00000000
20	1	1	0	8008d	00000001	00000000	00000000	00000002
21	1	0	0	80092	00000002	00000004	00000000	00000000

Data Cache Accesses:2500 Hits:1826 Hit Rate:0.730400 Misses: Compulsory:625 Conflict:3 Capacity:46

3. Λεπτομέρεις

Αρκετές φορές κάνουμε προσπέλαση της μνήμης καρφωτά, χωρίς tags, με την σωστή αντικατάσταση των τιμών τους, ώστε να αποφύγουμε μερικές ψευδοεντολές, lui και άλλες τεχνικές λεπτομέρειες. Αντίστοιχα συμβαίνει με τα μηνύματα που εκτυπώνουμε. Επομένως, δεν πρέπει να πειραχτεί σε σημαντικά το data segment του προγράμματος.