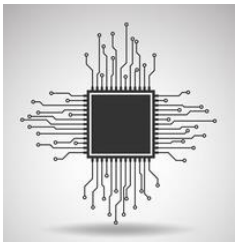
Εθνικο & Καποδιστριακο Πανεπιστημιο Αθηνων
Τμημα Πληροφορικης & Τηλεπικοινωνιων
Μαθημα: **Αρχιτεκτονικη Υπολογιστων ΙΙ**



Εργασία 2 (υποχρεωτική) – Κρυφές Μνήμες

Ακαδημαϊκο Ετος 2023 – 2024
(εκφωνηση) Δευτερα 18 Δεκεμβριου 2023

- Οι υποχρεωτικές εργασίες του μαθήματος είναι **δύο**. Σκοπός τους είναι η κατανόηση των εννοιών του μαθήματος με χρήση αρχιτεκτονικών προσομοιωτών. Η πρώτη υποχρεωτική εργασία αφορά τη διοχέτευση (pipelining) και η δεύτερη (αυτή) αφορά τις κρυφές μνήμες (cache memories).
- Οι δύο εργασίες είναι υποχρεωτικές και η βαθμολογία του μαθήματος θα προκύπτει από το γραπτό (60%), την εργασία της διοχέτευσης (20%), και την εργασία των κρυφών μνημών (20%).
- Κάθε ομάδα μπορεί να αποτελείται **από 1 έως και 3 φοιτητές** (**η υποβολή να γίνει μόνο από έναν φοιτητή εκ μέρους όλης της ομάδας – μην κάνετε υποβολές όλοι**). 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.
- Για την εξεταστική Σεπτεμβρίου δεν θα δοθούν άλλες εργασίες. Το Σεπτέμβριο εξετάζεται μόνο το γραπτό.
- Σε περίπτωση αντιγραφής θα μηδενίζονται όλες οι ομάδες που μετέχουν σε αυτή.
- Η παράδοση της **Εργασίας Κρυφών Μνημών** πρέπει να γίνει **ηλεκτρονικά μέχρι τα μεσάνυχτα της προθεσμίας και μόνο στο eclass** (να ανεβάσετε ένα μόνο αρχείο zip ή rar με την τεκμηρίωσή σας σε PDF και τους κώδικές σας). **Μην περιμένετε μέχρι την τελευταία στιγμή – κάθε εργασία απαιτεί τον χρόνο της**.

Ζητούμενο

Το ζητούμενο της εργασίας είναι η **βέλτιστη κατανάλωση ενέργειας** για συγκεκριμένη υπολογιστική εργασία. Η σχεδίαση του υλικού και του λογισμικού και η αξιολόγηση θα γίνει στον προσομοιωτή QtMips.

Η υπολογιστική εργασία είναι **ο υπολογισμός της πράξης X=X+Y*Z μεταξύ τριών τετραγωνικών πινάκων τάξης n** (δηλαδή διαστάσεων nxn). Οι αρχικοί πίνακες και το αποτέλεσμα θα βρίσκονται στο τμήμα δεδομένων του προγράμματος.

Δεδομένων τριών τετραγωνικών πινάκων ακεραίων αριθμών X, Y, και Z διαστάσεων nxn ο καθένας το πρόγραμμα να υπολογίζει την παραπάνω παράσταση. Να εξετάζεται το ενδεχόμενο υπερχείλισης κατά τη διάρκεια εκτέλεσης των πράξεων και το πρόγραμμα να τερματίζει εάν συμβεί υπερχείλιση.

Η MIPS CPU στην οποία θα εκτελείται το πρόγραμμά σας έχει σταθερή σχεδίαση στον πυρήνα της διοχέτευσης: branch predictor των 2-bit με BHT που προσπελάζεται με 5 bit και επίλυση των διακλαδώσεων στο στάδιο EX και πλήρη μονάδα κινδύνων με ανίχνευση και προώθηση. Η εκτέλεση κάθε εντολής στον επεξεργαστή δαπανά ενέργεια 1 picojoule.

Καλείσθε να ρυθμίσετε τη διαμόρφωση του συστήματος μνήμης ώστε να επιτευχθεί η βέλτιστη κατανάλωση ενέργειας. Ο επεξεργαστής διαθέτει δύο επίπεδα κρυφής μνήμης (ξεχωριστή L1 για εντολές και δεδομένα και ενιαία L2). Η προσπέλαση της κύριας μνήμης διαρκεί 35 κύκλους και η προσπέλαση της L2 διαρκεί 6 κύκλους. Οι κρυφές μνήμες L1 έχουν ίδιο συνολικό μέγεθος (καθεμία μπορεί να έχει μέγεθος s*8KB, όπου s=1 ή 2), ίδιο μέγεθος μπλοκ (που μπορεί να είναι w=2, 4 ή 8 λέξεις) και ίδια συσχετιστικότητα (που μπορεί να είναι a=1, 2 ή 4). Κάθε προσπέλαση της L1 (είτε εντολών είτε δεδομένων) δαπανά ενέργεια (s*w*0.5*a)/4 picojoule. Η κρυφή μνήμη L2 έχει μέγεθος 64KB και έχει ίδιο μέγεθος μπλοκ και συσχετιστικότητα με τις L1 caches. Κάθε προσπέλαση της L2 δαπανά ενέργεια 2*w*a picojoule. Κάθε προσπέλαση στην κύρια μνήμη δαπανά ενέργεια 320 picojoule. Σε όλες τις μνήμες η πολιτική εγγραφής είναι ετερόχρονη (write back) και με κατανομή σε εγγραφή (write allocate) και η αντικατάσταση LRU.

Ο ρυθμός ρολογιού του επεξεργαστή παραμένει σταθερός για οποιαδήποτε από τις παραπάνω διαμορφώσεις.

Να υπολογίσετε τη **συνολική ενέργεια που δαπανά το πρόγραμμά σας** για τιμές των διαστάσεων των πινάκων που αρχίζουν από n=8 και αυξάνονται κατά 2 (8, 10, 12, …) μέχρι τον μεγαλύτερο αριθμό που επιθυμείτε. Προσπαθήστε να συγγράψετε τον κώδικά σας και να ρυθμίσετε τις παραμέτρους των κρυφών μνημών **ώστε η συνολική δαπανώμενη ενέργεια να είναι η μικρότερη δυνατή**. Αν θέλετε, μπορείτε να συγγράψετε διαφορετικό κώδικα για διαφορετικές τιμές του n (στην περίπτωση αυτή να παραδώσετε όλα τα διαφορετικά προγράμματα).

Να συμπληρώσετε τον παρακάτω πίνακα με **τις σχεδιαστικές αποφάσεις σας** για τις κρυφές μνήμες.

Χαρακτηριστικό	**Τιμές**
L1 caches μέγεθος καθεμίας (8KB ή 16KB – τιμή του s)	
L1 caches, L2 cache (μέγεθος μπλοκ – τιμή του w, συσχετιστικότητα – τιμή του a)	

Να συμπληρώσετε τον παρακάτω πίνακα (ή/και να δώσετε σχετικά διαγράμματα) με **τα αποτελέσματά σας για τις διάφορες τιμές του n που δοκιμάσατε**.

n (διάσταση πινάκων)	**Χρόνος εκτέλεσης (κύκλοι)**	**Ενέργεια (picojoules)**
8		
10		
12		
14		
16		
18		
…		
…		

Τεκμηρίωση

[Σύντομη τεκμηρίωση της λύσης σας ενδεικτικά (όχι αυστηρά) μέχρι **10 σελίδες ξεκινώντας από την επόμενη σελίδα** – μην αλλάζετε τη μορφοποίηση του κειμένου (**και παραδώστε την τεκμηρίωση σε αρχείο PDF**). Η τεκμηρίωσή σας πρέπει να περιλαμβάνει παραδείγματα ορθής εκτέλεσης και σχολιασμό για την επίλυση του προβλήματος και την επίτευξη του ζητούμενου. Μπορείτε να χρησιμοποιήσετε εικόνες, διαγράμματα και ό,τι άλλο μπορεί να βοηθήσει στην εξήγηση της δουλειάς σας.]

Επεξήγηση κώδικα

Για την υλοποίηση του υπολογισμού X+=Y*Z βασιστήκαμε στο βοηθητικό υλικό του εργαστηρίου σχετικά με τον πολλαπλασιασμό και πρόσθεση πινάκων, καθώς και στις διαφάνειες που περιέγραφαν τον ψευδοκώδικα για τη ζητούμενη πράξη. Πιο αναλυτικά:

Αρχικά το τμήμα δεδομένων του προγράμματος .data περιέχει 3 πίνακες array_x,array_y,array_z οι οποίοι είναι μεγέθους NxN όπου N=8,10,12,14,16,18,20,22,24 τύπου .word δεσμεύοντας 8 byte . Το πρόγραμμα αρχίζει με τις κατάλληλες φορτώσεις των διευθύνσεων των πινάκων στις s6,s7,s5 για array_x,array_y,array_z αντίστοιχα, ενώ πραγματοποιούνται και οι αρχικοποιήσεις στους μετρητές για τις 3 επαναληπτικές διαδικασίες(loop,outer,inner),το s1 που αποθηκεύει τον αριθμό των στοιχείων που προσπελαύνονται από τον x και οι a1,a2,a3,ra για την inner loop συνθήκη ελέγχου. Στο a3 φορτώνεται μία τιμή που αλλάζει ανά μέγεθος πίνακα και καθορίζει πόσες εξωτερικές επαναλήψεις θα πραγματοποιηθούν-στην περίπτωσή μας για 24x24 576 στοιχεία του array x με μπλοκ 2x2 θα θέλουμε 576%(4*2)=72 outer loops. Η τιμή αυτή αλλάζει ανάλογα το μέγεθος του πίνακα που έχουμε να πραγματοποιήσουμε τη πράξη X+=Y*Z. Ενδεικτικά για 8,10,12,14,16,18,20,22 και 24 οι τιμές του $a3 που επιλέξαμε ήταν 8,12,18,24,32,40,50,60,72 αντίστοιχα.

Ακολουθώντας μεταβαίνουμε στο πεδίο outer όπου αφού ελέγξουμε εάν το a1 είναι ίσο με a3-δηλαδή έχουν προσπελαστεί όλα τα στοιχεία – κάνουμε jump στο πεδίο out.Αλλιώς φορτώνει word από τον array_y στους temporary t3,t4,t5,t6,t7 και πηγαίνουμε στη διαδικασία του πολλαπλασιασμού του μπλοκ. Πολλαπλασιάζει τα στοιχεία του Y με του Z και προσθέτει το αποτέλεσμα στους καταχωρητές $26 και $27.Παρόμοια γίνεται αυτή η διαδικασία 4 φορές για τα στοιχεία του Y που φορτώσαμε νωρίτερα. Έτσι στους $26 και $27 αποθηκεύεται το άθροισμα του μπλοκ που πρόκειται να αποθηκευτεί στο X αφού προστεθεί με το υπάρχον στοιχείο.

Στην inner loop πραγματοποιούνται οι κατάλληλοι έλεγχοι για υπερχείλιση(overflow) τόσο στον πολλαπλασιασμό των στοιχείων του στοιχείου array_z με τον array_y όσο και στη πρόσθεση με το στοιχείο του array_x. Για τον πολλαπλασιασμό η ψευδοεντολή mult αποθηκεύει τα χαμηλότερα 32bit στο mflo και τα υψηλότερα στο mfhi. Για να εξετάσουμε αν ο πολλαπλασιασμός επιτεύχθηκε δίχως overflow εξετάζουμε εάν το mfhi είναι διάφορο του μηδενός, εάν ισχύει τότε μεταβαίνουμε στο πεδίο out και τερματίζει το πρόγραμμα. Το αποτέλεσμα της πράξης αποθηκεύεται στο $30 μέσω της mflo. Στην πρόσθεση πολλαπλασιάζουμε unsigned με addu καθώς η add έχει exception στο overflow επομένως επιλέγουμε την addu για να υλοποιήσουμε το handle της υπερχείλισης. Στο $s2 δίνουμε τη τιμή 1 εάν το αποτέλεσμα της πρόσθεσης είναι μικρότερο της μιας μεταβλητής της πράξης και εάν αυτό είναι αληθές τότε με μία branch μεταβαίνουμε στο πεδίο out και τερματίζεται το πρόγραμμα.

Τέλος στους t8,t9 αποθηκεύουμε το αποτέλεσμα του πολλαπλασιασμού των $26,$27 και προσθέτουμε τα t0,t1 που είναι τα στοιχεία του array_x καθώς τα φορτώσαμε νωρίτερα. Έτσι επιτυγχάνεται η καταχώρηση της πράξης X=X+Y*Z. Γίνονται store στο πίνακα X και ενημερώνονται οι counters του s1 που μετράει πόσα στοιχεία έχουν προσπελασθεί, τα $26 και $27 σε μηδέν, ο inner loop counter και μεταφέρεται στο επόμενο element του X και του Z πριν επανέλθει στο πεδίο inner επαναληπτικά.

Τα πεδία out χωρίζονται σε 2 κατηγορίες: Το out 3 που προχωράει στο επόμενο a1 και μηδενίζει το counter της εσωτερικής loop και το out που καλεί τον τερματισμό του προγράμματος.

Σχεδιαστικές αποφάσεις

Σύμφωνα με τις μετρήσεις που κάναμε καταλήξαμε στις παρακάτω τιμές για s,w,a ως οι αποδοτικότερες:

Χαρακτηριστικό	Τιμές
L1 caches μέγεθος καθεμίας (8KB ή 16KB – τιμή του s)	8KB-s=1
L1 caches, L2 cache (μέγεθος μπλοκ – τιμή του w, συσχετιστικότητα – τιμή του a)	w=8 ,a=1

n (διάσταση πινάκων)	Χρόνος εκτέλεσης (κύκλοι)	Ενέργεια (picojoules)
8	7.721	30.053
10	10.917	39.869
12	15.711	54.593
14	20.505	69.317
16	26.897	88.949
18	33.289	108.581
20	41.279	133.121
22	49.269	157.661
24	58.857	187.109

Αφού ρυθμίσαμε τη διαμόρφωση του συστήματος μνήμης με s=1, το μέγεθος των L1 για εντολές και δεδομένα ήταν 8KB. Ένα μεγαλύτερο μέγεθος θα προσέφερε μείωση των αστοχιών χωρητικότητας, αλλά σε βάρος της απόδοσης και της ενέργειας. Για να διαμορφώσουμε σωστά το σύστημα μνήμης έπρεπε να υπολογίσουμε και τα Number of Sets. Χρησιμοποιήσαμε τον τύπο No.of.sets = cache size/ set size, όπου το set size δίνεται από τον τύπο a(συσχετιστικότητα) * w(μέγεθος μπλοκ) * 4bytes. Για το πρόγραμμα μας οι τιμές w=8 και a=1 οδήγησαν στην μικρότερη δυνατή συνολική δαπανώμενη ενέργεια. Με βάση τη θεωρία αυτό έγινε καθώς η μεγάλη συσχετιστικότητα έχει αρνητική επίπτωση στην απόδοση, αφού αυξάνει το χρόνο προσπέλασης. Σύμφωνα με τις δοκιμές μας παρόλο που η μεγάλη συσχετιστικότητα μείωνε τις αστοχίες διένεξης δεν ήταν αρκετό για να την προτιμήσουμε. Σε αντίθεση χρησιμοποιήσαμε το μεγαλύτερο δυνατό μέγεθος μπλοκ. Η αύξηση της ποινής αστοχίας για την απόδοση του προγράμματος που προκαλεί το μεγάλο w δεν δημιούργησε σημαντικά προβλήματα, καθώς η επίδραση στο ρυθμό αστοχίας που προσφέρει μείωσε τις υποχρεωτικές αστοχίες. Επιστέφοντας στον υπολογισμό του αριθμού των Sets, με βάση τα w και a που αναφέρθηκαν παραπάνω υπολογίσαμε No.SetsL1= 0,25 K(όπου K = 2^10) = 256 και No.SetsL2 (για Cache size=64KB) = 2K = 2048 sets.

Στην συνέχεια υπολογίσαμε την ενέργεια σε picojoules που δαπανά το πρόγραμμα μας για πίνακες 18x18 έως 24x24. Φυσικά δοκιμάσαμε όλους τους πιθανούς συνδυασμούς ρύθμισης του συστήματος μνήμης για να καταλήξουμε στον αποδοτικότερο. Παρακάτω θα δώσουμε ένα παράδειγμα για τον τρόπο που υπολογίσαμε την ενέργεια που ξοδεύει το πρόγραμμα μας για τον πίνακα 8χ8. Με βάση αυτό υπολογίσαμε και τους υπόλοιπους.

Αρχικά από την εκφώνηση γνωρίζουμε πως σε L1 program cache και L1 data cache η ενέργεια που δαπανά ένα hit είναι ίση με (s*w*0.5*a)/4 picojoule, άρα με βάση τις επιλογές μας για s,w,a βγαίνει 1 picojoule. Στην L2 η η ενέργεια που δαπανά ένα hit υπολογίζεται από τον τύπο 2*w*a picojoule, δηλαδή 16 picojoule. Παράλληλα γνωρίζουμε πως κάθε εντολή δαπανά 1 picojoule ενώ τα hit στην RAM 320 picojoule το καθένα. Θεωρήσαμε ότι η δαπάνη ενέργειας των Misses είναι αμελητέα. Για τον υπολογισμό της συνολικής ενέργειας που καταναλώνεται δημιουργήσαμε τον παρακάτω τύπο :

{ (L1_prog_cache_hits + L1_prog_cache_hits) * 1pj } + (L2_cache_hits * 16pj) + (RAM_hits * 320pj) + (instructions * 1pj) = TOTAL ENERGY

Παρατηρήσαμε όμως ότι στην L2 υπήρχαν μόνο αστοχίες. Αυτό συνέβαινε καθώς το πρόγραμμά μας δεν επαναχρησιμοποιεί τιμές ενός array, οπότε αν κάποιο στοιχείο μεταφερθεί στην L2, δεν θα γίνει ποτέ hit αφού δεν ξαναζητείται.

Άρα με βάση τις μετρήσεις στο παρακάτω screenshot καταλήξαμε πως η TOTAL_ENERGY_8x8 = 30.053.