ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ
www.cslab.ece.ntua.gr

2η ΑΣΚΗΣΗ ΠΡΟΗΓΜΕΝΑ ΘΕΜΑΤΑ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

Ακ. έτος 2024-2025, 8ο Εξάμηνο, Σχολή ΗΜ&ΜΥ

Τελική Ημερομηνία Παράδοσης: **25/05/2025**

1. Εισαγωγή

Στα πλαίσια της παρούσας άσκησης θα χρησιμοποιήσετε το εργαλείο **PIN** (που εγκαταστήσατε και χρησιμοποιήσατε στην προηγούμενη άσκηση) για να μελετήσετε την επίδραση διαφόρων παραμέτρων της ιεραρχίας μνήμης στην απόδοση ενός συνόλου εφαρμογών.

2. PINTOOL: simulator

Στον βοηθητικό κώδικα της άσκησης θα βρείτε το pintool **simulator.cpp**. Για την μεταγλώττισή του, κατευθυνθείτε στο directory του pintool και αφού τροποποιήσετε το PIN_ROOT path στο αρχείο makefile, δώστε:

```
$ cd advcomparch-ex2-helpcode/pintool
$ make clean; make
```

Το simulator pintool δέχεται τα παρακάτω ορίσματα:

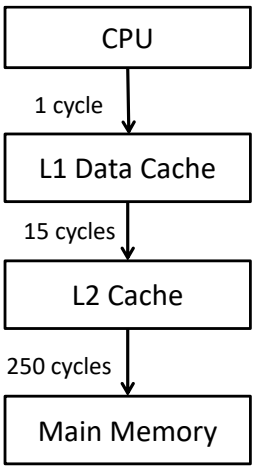
- -o <filename> : το αρχείο εξόδου όπου θα αποθηκευτούν οι παράμετροι και τα στατιστικά της προσομοίωσης
- -L1c : το μέγεθος της L1 (KB)
- -L1a : το associativity της L1
- -L1b : το μέγεθος του block της L1 (bytes)
- -L2c : το μέγεθος της L2 (KB)
- -L2a : το associativity της L2
- -L2b : το μέγεθος του block της L2 (bytes)

Η ιεραρχία κρυφής μνήμης που θα προσομοιώσετε στα πλαίσια αυτής της άσκησης απεικονίζεται στο παρακάτω σχήμα. Πιο συγκεκριμένα, το simulator pintool προσομοιώνει έναν in-order επεξεργαστή με ιεραρχία μνήμης που αποτελείται από δύο επίπεδα inclusive κρυφής μνήμης (L1-Data + L2 caches). Με τις κατάλληλες τροποποιήσεις, το simulator pintool μπορεί να προσομοιώσει διαφορετικές στρατηγικές ως προς το write allocation ή διαφορετικές πολιτικές αντικατάστασης.

Για την εκτίμηση της επίδοσης των εφαρμογών που εκτελούνται στις προσομοιώσεις, χρησιμοποιείται ένα απλό μοντέλο, όπου θεωρούμε ότι κάθε εντολή απαιτεί 1 κύκλο για την εκτέλεσή της (IPC=1). Επιπρόσθετα, οι εντολές που πραγματοποιούν πρόσβαση στη μνήμη (είτε load είτε store) προκαλούν επιπλέον καθυστερήσεις ανάλογα με το που βρίσκουν τα δεδομένα τους.

Συγκεκριμένα, όπως μπορείτε να διακρίνετε και στο παρακάτω σχήμα έχουμε τις εξής περιπτώσεις:

1. L1 hit: 1 cycle
2. L2 hit: 15 cycles
3. Main memory access: 250 cycles



Συνολικά, ο αριθμός των κύκλων υπολογίζεται ως:

Cycles = Instructions + L1_Accesses * L1_hit_cycles + L2_Accesses * L2_hit_cycles + Mem_Accesses * Mem_acc_cycles

3. Μετροπρογράμματα

Για τις προσομοιώσεις της άσκησης θα χρησιμοποιήσετε μετροπρογράμματα (benchmarks) από τη σουίτα SPEC_CPU2006. Πιο συγκεκριμένα θα χρησιμοποιήσετε τα παρακάτω 7 benchmarks:

1. 403.gcc
2. 429.mcf
3. 433.milc
4. 444.namd
5. 450.soplex
6. 459.GemsFDTD
7. 470.lbm

Στον βοηθητικό κώδικα της άσκησης σας δίνεται ο φάκελος **spec_benchmarks**, ο οποίος περιέχει τα εκτελέσιμα μαζί με τα απαραίτητα αρχεία εισόδου την εκτέλεση τους.

4. Πειραματική Αξιολόγηση

4.1 Εκτέλεση πειραμάτων και μετρικές

Για όλες τις προσομοιώσεις θα υποθέσετε μια 4-way, 32KB L1 cache με μέγεθος block size 32B. Σα βασική μετρική επίδοσης μπορείτε να χρησιμοποιήσετε το **Instructions Per Cycle** (IPC), καθώς αν ο κύκλος μηχανής και ο εκτελούμενος αριθμός εντολών παραμένουν σταθεροί κάθε φορά, μεγαλύτερες τιμές στο IPC υποδεικνύουν καλύτερη επίδοση. Για να αναλύσετε καλύτερα τα αποτελέσματα σας, εκτός από τις βασικές μετρικές επίδοσης της ιεραρχίας μνήμης (π.χ. miss rates) σας προτείνουμε να χρησιμοποιήσετε και τα **Misses Per Kilo Instructions** (MPKI).

4.2 Μελέτη χαρακτηριστικών της L2 cache

Μελετήστε την επίδραση που έχουν στην επίδοση τα βασικά χαρακτηριστικά της L2, δηλαδή το μέγεθος, η συσχετιστικότητα (associativity) και το μέγεθος του cache block. Συγκεκριμένα, εκτελέστε τα benchmarks για τις παρακάτω L2 caches και μελετήστε τις μεταβολές στο IPC. Τι συμπεράσματα μπορείτε να εξάγετε για τα μετροπρογράμματα που εκτελέσατε; Ποιες από τις παραμέτρους που εξετάσατε έχουν τη μεγαλύτερη επίδραση στην επίδοση;

Ποια cache θα διαλέγατε ως την βέλτιστη επιλογή για κάθε μια από τις διαφορετικές χωρητικότητες;

L2 size (KB)	L2 associativity	L2 cache block size (B)
256	4, 8	64, 128, 256
512	4, 8	
1024	8, 16	
2048	16	

4.3 Πολιτικές Αντικατάστασης (replacement policies)

Σε όλες τις προηγούμενες προσομοιώσεις, η ιεραρχία μνήμης χρησιμοποιούσε την πολιτική αντικατάστασης Least Recently Used (LRU). Τροποποιήστε/επεκτείνετε κατάλληλα τον κώδικα που σας έχει δοθεί, ώστε οι κρυφές μνήμες να μπορούν να χρησιμοποιούν τις εξής πολιτικές αντικατάστασης:

- **Random**: Η πολιτική αυτή επιλέγει τυχαία σε κάθε set το block που θα αντικατασταθεί. Ως seed για την συνάρτηση τυχαιότητας που θα χρησιμοποιήσετε στον κώδικά σας χρησιμοποιήστε τα 5 τελευταία νούμερα του ΑΜ σας. Για τη συνάρτηση rand() αυτό θα γινόταν με τις δύο εξής εντολές:

```
srand(10242); // Ο ΑΜ μου τελειώνει σε 10242.
int my_random_number = rand();
```

- **LFU**: Η πολιτική **Least Frequently Used** (LFU) [1] καταγράφει πόσες φορές χρησιμοποιείται κάθε block της cache και όταν χρειαστεί να αντικαταστήσει κάποιο, επιλέγει το block του set που έχει χρησιμοποιηθεί λιγότερο.
- **LIP**: Η πολιτική **LRU Insertion Policy** (LIP)[2] αποτελεί παραλλαγή της LRU πολιτικής, με μοναδική διαφορά το χαρακτηρισμό των blocks που εισάγονται κάθε φορά στην cache. Συγκεκριμένα, κάθε block που εισάγεται στο set εξαιτίας κάποιου miss, χαρακτηρίζεται ως το LRU block του set (αντί για το MRU, όπως συμβαίνει στην κλασική LRU πολιτική).
- **SRRIP**: Η πολιτική **Static Re-reference Interval Prediction** (SRRIP)[3] προσπαθεί να αποτρέψει blocks που θα χρησιμοποιηθούν στο μακρινό μέλλον να καταλάβουν πολύτιμο χώρο στην cache. Για να το πετύχει αυτό, εκτιμά πότε θα χρησιμοποιηθεί ξανά κάθε block (θα έχουμε δηλαδή re-reference) και όταν χρειάζεται να αντικαταστήσει κάποιο, επιλέγει το block του set που έχει προβλεφθεί ότι θα χρησιμοποιηθεί στο πιο μακρινό μέλλον (δείτε το section 4.2 του paper και το παράδειγμα που περιγράφεται στο Figure 3). Υποθέστε πως ο μετρητής που χρησιμοποιείται για κάθε block έχει μήκος n bits, όπου n το associativity της cache.

Αφού υλοποιήσετε τις παραπάνω πολιτικές, εκτελέστε τα benchmarks για κάθε μια πολιτική για τις 4 L2 caches που διαλέξατε ως βέλτιστες στο προηγούμενο ερώτημα. Σε κάθε προσομοίωση η L1 και η L2 θα χρησιμοποιούν την ίδια πολιτική αντικατάστασης.

Ποια πολιτική θα διαλέγατε να υλοποιήσετε στο σύστημα σας;

[1] https://en.wikipedia.org/wiki/Least_frequently_used
[2] https://dl.acm.org/citation.cfm?id=1250709
[3] https://people.csail.mit.edu/emer/media/papers/2010.06.isca.rrip.pdf

Παραδοτέο της άσκησης θα είναι ένα ηλεκτρονικό κείμενο (**pdf**, **docx** ή **odt**). *Στο ηλεκτρονικό κείμενο να αναφέρετε στην αρχή τα στοιχεία σας (Όνομα, Επώνυμο, ΑΜ).*

Η άσκηση θα παραδοθεί ηλεκτρονικά στο moodle του μαθήματος:

https://helios.ntua.gr/course/view.php?id=1039

Δουλέψτε ατομικά. Έχει ιδιαίτερη αξία για την κατανόηση του μαθήματος να κάνετε μόνοι σας την εργασία. Μην προσπαθήσετε να την αντιγράψετε από άλλους συμφοιτητές σας.

Μην αφήσετε την εργασία για το τελευταίο Σαββατοκύριακο, απαιτεί αρκετό χρόνο για οργάνωση και την εκτέλεση όλων των προσομοιώσεων, οπότε ξεκινήστε αμέσως!

Καθώς τα ίδια εργαλεία και μετροπρογράμματα είχαν χρησιμοποιηθεί και τις προηγούμενες χρονιές, είναι εξαιρετικά πιθανό αρκετές (αν όχι όλες οι) απορίες που μπορεί να προκύψουν να έχουν ήδη απαντηθεί. Σας συμβουλεύουμε λοιπόν να ψάξετε για απαντήσεις/βοήθεια στα archives της mailing list του μαθήματος, τα οποία βρίσκονται εδώ:

http://lists.cslab.ece.ntua.gr/pipermail/advcomparch/