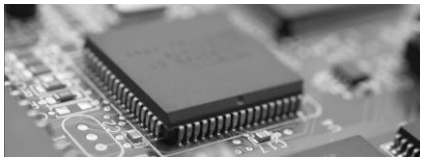
ΕΘΝΙΚΟ & ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ
ΜΑΘΗΜΑ: **ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΥΠΟΛΟΓΙΣΤΩΝ ΙΙ**



Εργασία 2 – Κρυφές Μνήμες

ΑΚΑΔΗΜΑΪΚΟ ΕΤΟΣ 2020 – 2021

Ζητούμενο

Το ζητούμενο της εργασίας είναι να υλοποιήσετε ένα ή περισσότερα προγράμματα σε συμβολική γλώσσα MIPS για τον προσομοιωτή QtMips η εκτέλεση των οποίων να οδηγεί **στην εξαγωγή συμπερασμάτων για την οργάνωση των δύο κρυφών μνημών (data cache και program/instruction cache)**.

Η μόνη διαθέσιμη πληροφορία για την εξαγωγή συμπερασμάτων είναι τα στατιστικά που παρέχει ο QtMips **μετά την πλήρη εκτέλεση** ενός προγράμματος: **Cycles, Hits, Misses, Memory Reads, Memory Writes, Memory Stall Cycles** (μπορείτε να χρησιμοποιήσετε όποια και όσα από αυτά τα στατιστικά στοιχεία χρειάζεστε).

Τα συμπεράσματα (για τις παραμέτρους των κρυφών μνημών) που πρέπει να εξαχθούν είναι:

- **Ποιο είναι το μέγεθος της κάθε κρυφής μνήμης**;
  Πιθανά μεγέθη: 8, 16 και 32 KB.
- **Ποιο είναι το μέγεθος του κάθε μπλοκ της κρυφής μνήμης**;
  Πιθανά μεγέθη: 4 μέχρι και 64 λέξεις ανά μπλοκ.
- **Ποια είναι η συσχετιστικότητα της κάθε κρυφής μνήμης**;
  Πιθανές τιμές συσχετιστικότητας: 1 (direct mapped), 2, 4, και 8.

(Με βάση το μέγεθος κρυφής μνήμης να υπολογίσετε ποιες είναι οι δυνατές τιμές για τις άλλες δύο παραμέτρους – μέγεθος μπλοκ, συσχετιστικότητα – με δεδομένο τον περιορισμό του QtMips για το μέγιστο πλήθος συνόλων.)

Κατά τα λοιπά, η κάθε κρυφή μνήμη πρέπει να υλοποιεί υποχρεωτικά αντικατάσταση LRU (εάν δεν είναι direct mapped) ενώ για την data cache η πολιτική εγγραφής πρέπει να είναι υποχρεωτικά ετερόχρονη (write back). Για την κύρια μνήμη υποθέστε τις default τιμές για read και write access (10 cycles) ενώ για την CPU μπορείτε να επιλέξετε ό,τι θέλετε (με διοχέτευση ή χωρίς, με ανίχνευση κινδύνων ή όχι κ.λπ.).

Η λύση σας μπορεί να αποτελείται από N προγράμματα mb1, … , mbN. Η πλήρης εκτέλεση του καθενός μέχρι το τέλος παρέχει τα στατιστικά του QtMips από τα οποία πρέπει να εξαχθούν τα συμπεράσματα. Για κάθε ζητούμενο συμπέρασμα μπορείτε να χρησιμοποιήσετε αν θέλετε το ίδιο σύνολο προγραμμάτων ή διαφορετικό σύνολο προγραμμάτων. Να εξηγήσετε με ποιον τρόπο αντιλαμβάνεστε από τα στατιστικά εκτέλεσης του κάθε συνδυασμού προγραμμάτων καθεμία από τις ζητούμενες παραμέτρους των κρυφών μνημών του επεξεργαστή.

Οι ζητούμενες παράμετροι είναι έξι (τρεις για κάθε κρυφή μνήμη: μέγεθος, μέγεθος μπλοκ, συσχετιστικότητα). Όποια ομάδα παραδώσει εργασία που εξάγει τουλάχιστον δύο από τις έξι παραμέτρους θα πάρει τουλάχιστον βαθμό 6. Όποιες εργασίες οδηγούν σε εξαγωγή περισσότερων παραμέτρων θα βαθμολογηθούν με μεγαλύτερο βαθμό.

Εκτός από τα προγράμματά σας σε συμβολική γλώσσα, που πρέπει να παραδώσετε, να συμπληρώσετε τον ακόλουθο πίνακα και φυσικά να εξηγήσετε στην τεκμηρίωση που ακολουθεί.

Συμπέρασμα για παράμετρο	Προγράμματα που απαιτούνται (ονόματα των αρχείων σας)	Στατιστικά εκτέλεσης για εξαγωγή συμπεράσματος (ποια και ποιες τιμές)
Μέγεθος data cache	data_cache_size.s	Cache Misses
Μέγεθος μπλοκ data cache	data_block_size.s	Cache Misses
Συσχετιστικότητα data cache	data_assoc.s	Cache Misses
Μέγεθος κρυφής program cache	prog_cache_size.s	Cache Misses
Μέγεθος μπλοκ program cache	prog_block_size.s	Cache Misses
Συσχετιστικότητα program cache	prog_assoc.s	Cache Misses

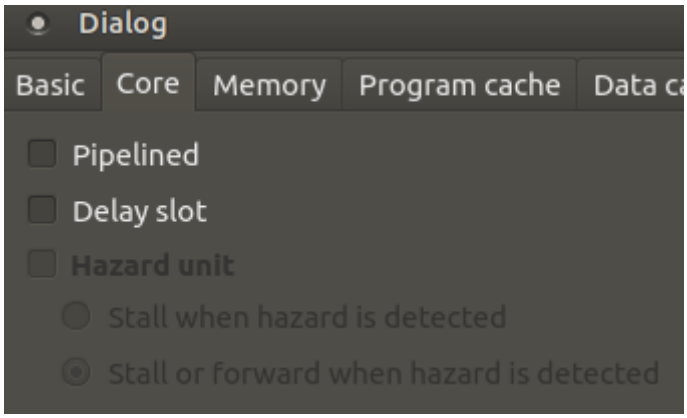
Παραδοτέα αρχεία

***.s :** Προγράμματα assembly MIPS για την εξαγωγή στατιστικών που καθορίζουν διάφορα χαρακτηριστικά των κρυφών μνημών. Η μοναδική μετρική εξαγωγής συμπερασμάτων που χρησιμοποιούμε είναι τα **Cache Misses** .

combinations.txt : Αρχείο κειμένου όπου έχουμε καταγράψει όλους τους δυνατούς συνδυασμούς <Cache Memory Size – Degree of Associativity - Block Size – Number of sets> , που υποστηρίζονται από τον QtMips, με βάση και τους περιορισμούς που αναγράφονται στην εκφώνηση. Για αυτούς τους συνδυασμούς παραμέτρων έχουν ελεγχθεί τα προγράμματά μας και έχουν εξαχθεί τα στατιστικά που παρουσιάζουμε στο κείμενο.

Παράμετροι του επεξεργαστή

Για την εκτέλεση των διαγνωστικών προγραμμάτων μας έχουμε επιλέξει την Single Cycle υλοποίηση του επεξεργαστή χωρίς delay slot, επομένως από το παράθυρο "*Core*" του προσομοιωτή απενεργοποιούμε τις ρυθμίσεις **Pipelined** και **Delay Slot** :



Γενικός αλγόριθμος εύρεσης των ζητούμενων ποσοτήτων

1. Βρίσκουμε το **μέγεθος του block** της cache μέσω του προγράμματος *{data/prog}_block_size.s* .
2. Βρίσκουμε το **μέγεθος της cache** μέσω του προγράμματος *{data/prog}_cache_size.s* .
3. Βρίσκουμε το **βαθμό συσχετιστικότητας** της cache μέσω του προγράμματος *{data/prog}_assoc.s* .

Τρόπος εύρεσης των ζητούμενων ποσοτήτων - Data Cache

Block Size

Εκτελούμε το πρόγραμμα *data_block_size.s* . Το πρόγραμμα αυτό φορτώνει 64 συνεχόμενες λέξεις από τη μνήμη δεδομένων. Συνεπώς, περιμένουμε ότι ο αριθμός από cache misses θα είναι:

Cache Misses = 64/Block Size και άρα Block Size = 64 / Cache Misses

Block Size	Cache Misses
4	16
8	8
16	4
32	2
64	1

Cache Size

Εκτελούμε το πρόγραμμα *data_cache_size.s* . Το πρόγραμμα αυτό αρχικά φορτώνει 4K συνεχόμενα words (16K bytes) 2 φορές, και στη συνέχεια φορτώνει 8K συνεχόμενα words (32K bytes), επίσης 2 φορές. Τα σύνολα των 4K και 8K λέξεων ξεκινούν από την ίδια διεύθυνση μνήμης (και άρα από την ίδια λέξη).

Περιμένουμε ότι εάν η μνήμη μας είναι 32K bytes τότε θα έχει τα λιγότερα δυνατά cache misses, όσα χρειάζονται για να φορτώσει τα 8K συνεχόμενα words (4K από την 1η επανάληψη, και τα υπόλοιπα 4K από την 3η επανάληψη όπου φορτώνει 8K). Εάν η μνήμη μας είναι 16K bytes, τότε θεωρητικά η 2η φόρτωση των 4K words δε θα επηρεάσει τα cache misses, ενώ η 3η και η 4η επανάληψη θα προσθέσουν cache misses (χαρακτηριστικά, στην 4η επανάληψη κάθε καινούργιο block που προσκομίζεται θα έχει cache miss). Τέλος, εάν η μνήμη είναι 8K, τότε κάθε επανάληψη θα επιφέρει το μέγιστο δυνατό αριθμό από cache misses.

Επειδή χρησιμοποιούμε τα cache misses ως μετρική για να βρούμε το cache size, πρέπει να λάβουμε υπόψιν μας το **block size,** που επηρεάζει τον αριθμό των cache misses. Συνεπώς, *για να βρούμε το cache size πρέπει οπωσδήποτε να γνωρίζουμε το block size*.

Πίνακας με Cache Misses για κάθε συνδυασμό Cache Size – Block Size.

Cache sz/Block sz	**4**	**8**	**16**	**32**	**64**
8K	6144	3072	1536	768	384
16K	-	2048	1024	512	256
32K	-	-	512	256	128

Degree of Associativity

Εκτελούμε το πρόγραμμα *data_assoc.s* . Το πρόγραμμα αυτό χρησιμοποιεί 8 λέξεις από τη μνήμη που βρίσκονται σε διευθύνσεις τέτοιες ώστε να έχουν ίδιο αριθμοδείκτη όταν πρόκειται να απεικονιστούν στην cache.

Οι διευθύνσεις των λέξεων είναι οι ακόλουθες: 0x10000000, 0x11000000, … , 0x17000000

Εκμεταλλευόμαστε το γεγονός ότι το μέγιστο πλήθος συνόλων (number of sets) που εμφανίζεται στους πιθανούς συνδυασμούς είναι 64, συνεπώς απαιτούνται το πολύ 6 bit για το πεδίο αριθμοδείκτη στη διεύθυνση της θέσης μνήμης. Το μέγεθος block μπορεί να είναι το πολύ 64 byte, και δουλεύουμε με λέξεις των 4 byte, συνεπώς απαιτούνται το πολύ 6 bit για την επιλογή της λέξης στο block (block offset) και 2 bit για την επιλογή του ζητούμενου byte. Επομένως, καταλήγουμε στο συμπέρασμα ότι, στη χειρότερη περίπτωση (64 σύνολα, 64 words block size), τα χαμηλότερα 8 bit της διεύθυνσης [0:7] χρησιμοποιούνται για τα offset επιπέδου block και λέξης, τα επομένα 6 bit [8:13] χρησιμοποιούνται για τον αριθμοδείκτη και τα υπόλοιπα 18 bit [14:31] χρησιμοποιούνται για την ετικέτα. Συνεπώς, θέτοντας τα bit [2:13] στις διευθύνσεις των λέξεων που φορτώνουμε να είναι 0, έχουμε εξασφαλίσει ότι όλες οι λέξεις θα απεικονιστούν στο ίδιο σύνολο, **ανεξάρτητα** από το μέγεθος block ή το πλήθος συνόλων της cache.

Ο σκοπός του προγράμματος είναι να φορτώσει σε ένα συγκεκριμένο σύνολο 2, 4 και 8 blocks. Οι φορτώσεις των συγκεκριμένων block γίνονται 2 φορές, έτσι ώστε να καθοριστεί μονοσήμαντα ο βαθμός συσχετιστικότητας. Η αντιστοίχιση ανάμεσα στο βαθμό συσχετιστικότητας και τον αριθμό των Cache Misses είναι η ακόλουθη:

Degree of Associativity	**Cache Misses**
1	28
2	24
4	16
8	8

Τρόπος εύρεσης των ζητούμενων ποσοτήτων - Program Cache

Block Size

Εκτελούμε το πρόγραμμα *prog_block_size.s* . Το πρόγραμμα αυτό εκτελεί 64 συνεχόμενες εντολές από τη μνήμη εντολών. Συνεπώς, περιμένουμε ότι ο αριθμός από cache misses θα είναι:

Cache Misses = 64/Block Size και άρα Block Size = 64 / Cache Misses

Block Size	Cache Misses
4	16
8	8
16	4
32	2
64	1

Cache Size

Εκτελούμε το πρόγραμμα prog_*cache_size.s* . Η λογική είναι ίδια με αυτή της data cache. Το πρόγραμμα αρχικά εκτελεί 4K συνεχόμενες εντολές (16K bytes) 2 φορές, και στη συνέχεια εκτελεί ένα διαφορετικό σύνολο από 8K συνεχόμενες εντολές (32K bytes), επίσης 2 φορές.

Εκμεταλλευόμαστε το γεγονός ότι ο QtMips γεμίζει τη μνήμη εντολών με εντολές **nop** και χρησιμοποιούμε τις ετικέτες **.org** για να ορίσουμε ρητά τη διεύθυνση των εντολών στη μνήμη εντολών.

Επειδή χρησιμοποιούμε τα cache misses ως μετρική για να βρούμε το cache size, πρέπει να λάβουμε υπόψιν μας το block size, που επηρεάζει τον αριθμό των cache misses. Συνεπώς, για να βρούμε το cache size πρέπει οπωσδήποτε να γνωρίζουμε το **block size**.

Πίνακας με Cache Misses για κάθε συνδυασμό Cache Size – Block Size.

Cache sz/Block sz	4	8	16	32	64
8K	6149	3077	1541	773	389
16K	-	2564	1284	644	324
32K	-	-	770	386	194

Degree of Associativity

Εκτελούμε το πρόγραμμα *prog_assoc.s* . Η λογική είναι παρόμοια με το αντίστοιχο πρόγραμμα εύρεσης του βαθμού συσχετιστικότητας για την data cache. Το πρόγραμμα αυτό χρησιμοποιεί 8 block (με το πολύ 4 λέξεις ανά block) από τη μνήμη που βρίσκονται σε διευθύνσεις τέτοιες ώστε να έχουν ίδιο αριθμοδείκτη όταν πρόκειται να απεικονιστούν στην cache. Εξηγήσαμε πώς πετυχαίνουμε αυτή τη διευθυνσιοδότηση στο κομμάτι της data cache. Η ίδια λογική ακολουθείται κι εδώ.

Ο σκοπός του προγράμματος είναι να φορτώσει σε ένα συγκεκριμένο σύνολο 1, 2, 4 και 8 blocks. Οι φορτώσεις των συγκεκριμένων "ομάδων" block γίνονται 2 φορές, έτσι ώστε να καθοριστεί μονοσήμαντα ο βαθμός συσχετιστικότητας. Η αντιστοίχιση ανάμεσα στο βαθμό συσχετιστικότητας και τον αριθμό των Cache Misses είναι η ακόλουθη:

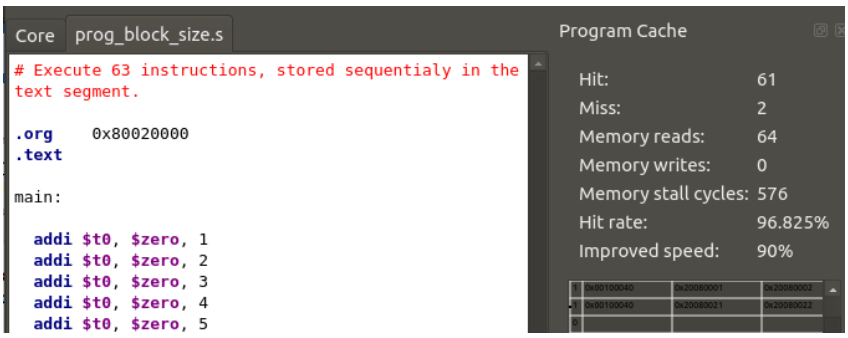
Degree of Associativity	Cache Misses
1	31
2	27
4	19
8	11

Παραδείγματα εκτέλεσης

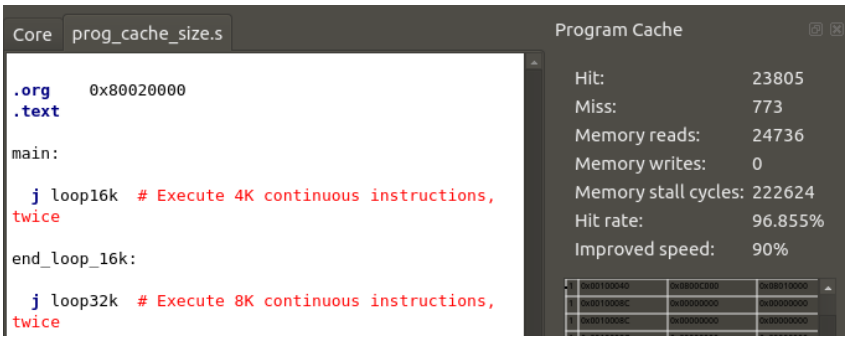
Έχουμε ελέγξει ενδελεχώς όλους τους δυνατούς συνδυασμούς που βρίσκονται στο αρχείο **combinations.txt** ώστε να βεβαιωθούμε για την εγκυρότητα των αποτελεσμάτων μας. Ενδεικτικά, παραθέτουμε από ένα παράδειγμα συνδυασμού για την Program και την Data cache.

Program Cache

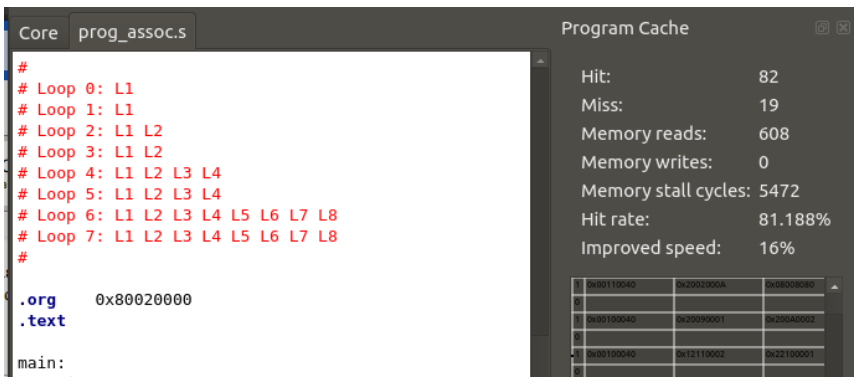
Degree of Associativity : 4 – Block Size: 32 – Number of Sets: 16 (Cache Size: 8K)



Block Size	Cache Misses
32	2



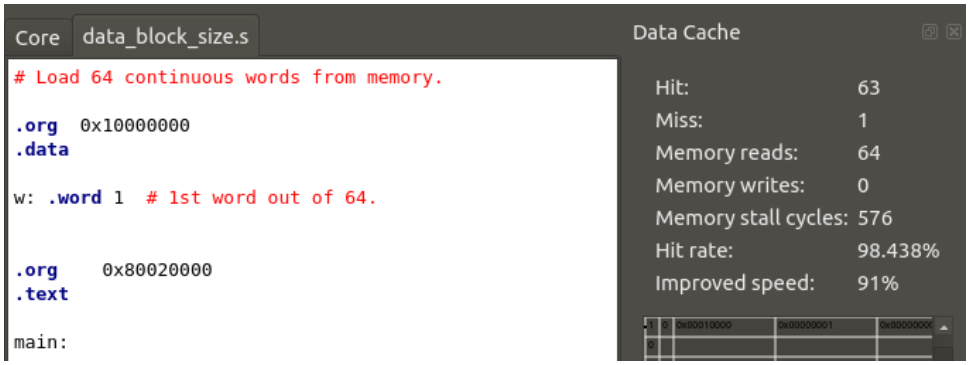
Cache sz/Block sz	32
8K	773



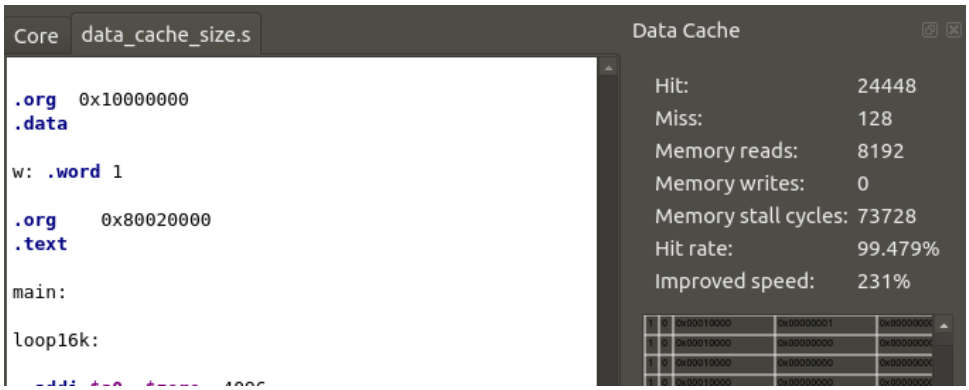
Degree of Associativity	Cache Misses
4	19

Data Cache

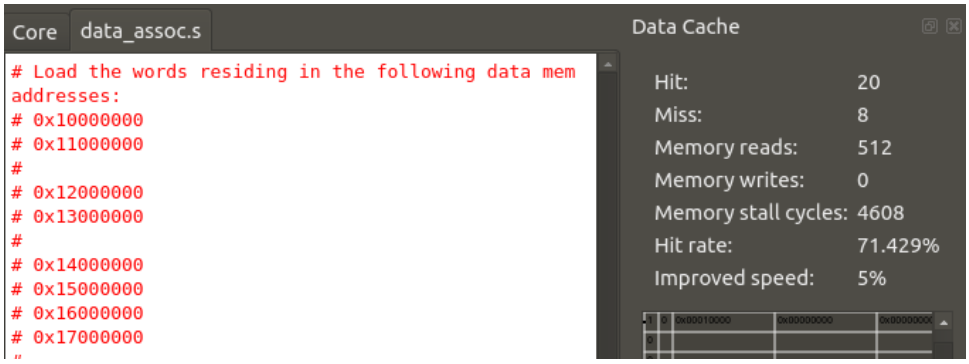
Degree of Associativity: 8 - Block size: 64 - Num of sets: 16 (Cache Size: 32K)



Block Size	Cache Misses
64	1



Cache sz/Block sz	64
32K	128



Degree of Associativity	Cache Misses
8	8