ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Πρόγραμμα Μεταπτυχιακών Σπουδών Επιστήμη και Τεχνολογία της Πληροφορικής και των Υπολογιστών

Ειδίκευση Υλικού και Υπολογιστικών Συστημάτων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Υλοποίηση ενός εκπαιδευτικού προσομοιωτή μνήμης cache για κινητά Android

Γεώργιος Τσιαμπάς
Α.Μ. 19029

Εισηγητής: Δημήτριος Κεχαγιάς, Ομότιμος Καθηγητής

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Υλοποίηση ενός εκπαιδευτικού προσομοιωτή μνήμης cache για κινητά Android

Γεώργιος Τσιαμπάς
Α.Μ. 19029

Εισηγητής:

Δημήτριος Κεχαγιάς, Ομότιμος Καθηγητής

Εξεταστική Επιτροπή:

ΚΕΧΑΓΙΑΣ ΔΗΜΗΤΡΙΟΣ
ΜΠΟΓΡΗΣ ΑΝΤΩΝΗΣ
ΜΑΜΑΛΗΣ ΒΑΣΙΛΕΙΟΣ

Ημερομηνία εξέτασης 20/11/2023

DIMITRIOS KECHAGIAS Digitally signed by DIMITRIOS KECHAGIAS Date: 2023.12.22 10:48:17 +02'00'

Antonios Bogris Digitally signed by Antonios Bogris Date: 2023.12.22 12:51:01 +02'00'

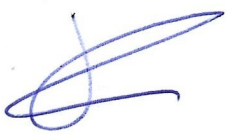
Vasileios Mamalis Digitally signed by Vasileios Mamalis Date: 2023.12.22 12:12:24 +02'00'

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΜΕΤΑΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

Ο κάτωθι υπογεγραμμένος Γεώργιος Τσιαμπάς του Αθανασίου με αριθμό μητρώου 19029 φοιτητής του Προγράμματος Μεταπτυχιακών Σπουδών Επιστήμης και Τεχνολογίας της Πληροφορικής και των Υπολογιστών του Τμήματος Μηχανικών Πληροφορικής και Υπολογιστών της Σχολής Μηχανικών του Πανεπιστημίου Δυτικής Αττικής, δηλώνω ότι:

«Βεβαιώνω ότι είμαι συγγραφέας αυτής της Διπλωματικής εργασίας και κάθε βοήθεια την οποία είχα για την προετοιμασία της, είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος.
Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του πτυχίου μου».

Ο Δηλών



ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα πρώτα να ευχαριστήσω τους ανθρώπους που με βοήθησαν και με στήριξαν σε αυτό το μεταπτυχιακό πρόγραμμα, χάρη στους οποίους όχι μόνο ολοκλήρωσα το μεταπτυχιακό αυτό πρόγραμμα, αλλά και έμαθα αρκετά τόσο εντός όσο και εκτός του μεταπτυχιακού προγράμματος. Επίσης να συγχαρώ όσους συμμετείχαν στο μεταπτυχιακό πρόγραμμα και εύχομαι να συνεχίσουνε να μεταδίδουν γνώσεις και σε άλλους ανθρώπους γιατί η γνώση είναι αρετή και δύναμη. Επίσης, να ευχαριστήσω και τον επιβλέπων καθηγητή μου, που πέρα από τη συμβολή του εντός του μεταπτυχιακού προγράμματος, βοήθησε και στη συγγραφή της διπλωματικής αυτής εργασίας.

ΠΕΡΙΛΗΨΗ

Το χάσμα μεταξύ των ταχυτήτων της μνήμης και του μικροεπεξεργαστή έχει ενθαρρύνει τους αρχιτέκτονες μικροεπεξεργαστών να παρέχουν μηχανισμούς στους οποίους θα “κρύβουν” τις μεγάλες καθυστερήσεις μνήμης. Οι κρυφές μνήμες (μνήμες cache) έχουν γίνει ο βασικός μηχανισμός που χρησιμοποιούν οι επεξεργαστές για την απόκρυψη αυτών των καθυστερήσεων και τη μείωση του μέσου χρόνου πρόσβασης στα δεδομένα.

Οι κρυφές μνήμες αποτελούν ένα από τα βασικά θέματα σε όλα τα μαθήματα οργάνωσης και αρχιτεκτονικής υπολογιστών. Κατά συνέπεια, υπάρχουν αρκετές εκπαιδευτικές προτάσεις που αφορούν στην κατανόηση αυτού του θέματος.

Αντικείμενο της παρούσας διπλωματικής εργασίας είναι να υλοποιήσουμε έναν προσομοιωτή κρυφής μνήμης που θα παρέχει ένα ελκυστικό και εύκολα κατανοητό εργαλείο στους φοιτητές στη μελέτη εννοιών που σχετίζονται με την κρυφή μνήμη. Για το σκοπό αυτό χρησιμοποιήσαμε τεχνολογίες όπως το android studio με το οποίο μπορούμε να δημιουργήσουμε μια εφαρμογή για κινητά που χρησιμοποιούν android.

ABSTRACT

The gap between the memory and the microprocessor speeds has encouraged microprocessor architects to provide mechanisms in which to “hide” the long memory latencies. Cache memories have become the basic mechanism employed by processors to hide these latencies and reduce the average data access time.

Cache memories are one of the key topics in all computer organization and architecture courses. Consequently, there are several educational proposals concerning the understanding of this topic.

The objective of this thesis is to implement a cache simulator that will provide an attractive and easy-to-understand tool for students in the study of cache-related concepts. For this purpose we have used technologies like android studio with which we can create a mobile application using android.

ΕΠΙΣΤΗΜΟΝΙΚΗ ΠΕΡΙΟΧΗ: Αρχιτεκτονική Υπολογιστών
ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: επεξεργαστής, κρυφή μνήμη, προσομοιωτής κρυφής μνήμης

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

ΚΕΦΑΛΑΙΟ 1

ΕΙΣΑΓΩΓΗ

Για την εξέλιξη των υπολογιστών στη σημερινή τους μορφή συνετέλεσαν τόσο η ανάπτυξη της τεχνολογίας κατασκευής των, όσο και η επινόηση καινοτομιών στη σχεδίασή τους.

Η ανάπτυξη της τεχνολογίας με την εμφάνιση των μικροπεξεργαστών ήταν ένα τεράστιο άλμα στην ανάπτυξη των υπολογιστών. Ακόμα και σήμερα πολυεπεξεργαστές, υπερυπολογιστές αλλά και ενσωματωμένοι υπολογιστές για την ανάγκη ειδικών εφαρμογών, κατασκευάζονται από μικροεπεξεργαστές. Επίσης, η εμφάνιση των μικροεπεξεργαστών μείωσαν το κόστος των υπολογιστών με συνέπεια το ευρύ κοινό να μπορεί εύκολα να τους αποκτήσει.

Το επόμενο βήμα στην ανάπτυξη των υπολογιστών πραγματοποιήθηκε μέσω της σχεδίασής των με την αρχιτεκτονική RISC, η οποία βελτίωσε την απόδοσή τους. Οι σχεδιαστές των υπολογιστών RISC εστίασαν σε δυο καινοτομίες που αύξησαν την απόδοση των υπολογιστών. Η μια καινοτομία είναι ο παραλληλισμός στην εκτέλεση των εντολών μέσω της διασωλήνωσης και της πολλαπλής έκδοσης εντολών. Η άλλη καινοτομία είναι η χρήση κρυφών μνημών, οι οποίες και εξελίχθηκαν με την πάροδο του χρόνου. Δεν είναι τυχαίο ότι κάποιες αρχιτεκτονικές που δεν ακολούθησαν την αρχιτεκτονική RISC εξαφανίστηκαν και κάποιες διαφοροποιήθηκαν ώστε να υιοθετήσουν διάφορα στοιχεία της αρχιτεκτονικής RISC με συνέπεια να δημιουργήσουν μια υβριδική μορφή αρχιτεκτονικών.

Από ένα σημείο (περίπου το 2004) και μετά, τα περιθώρια στην ανάπτυξη της τεχνολογίας των υπολογιστών καθώς και στη βελτίωση των υπαρχόντων καινοτομιών περιορίστηκαν. Αυτό είχε ως συνέπεια η απόδοση των υπολογιστών να παρουσιάζει μια στασιμότητα με την πάροδο του χρόνου. Για αυτό τον λόγο οι αρχιτέκτονες υπολογιστών αναζήτησαν μια καινούρια καινοτομία που θα αύξανε την απόδοση των υπολογιστών. Η καινοτομία που εμφανίστηκε ήταν η δημιουργία πολυεπεξεργαστών. Οι πολυεπεξεργαστές επέτρεπαν την παράλληλη εκτέλεση του κώδικα με συνέπεια την ταχύτερη εκτέλεση του και άρα την αύξηση της απόδοσης των υπολογιστών. Αρκετοί ήταν οι κατασκευαστές υπολογιστών που επικεντρώθηκαν στην δημιουργία πολλαπλών επεξεργαστών από ότι στην δημιουργία ταχύτερων μονών επεξεργαστών.

Η πρόοδος κάποιων τεχνολογιών των υπολογιστών είναι τόσο μεγάλη σε μερικές περιπτώσεις που οι σχεδιαστές συχνά σχεδιάζουν υπολογιστές με γνώμονα την επόμενη

τεχνολογία. Κάποιες τεχνολογίες που αλλάζουν με ταχύτατο ρυθμό είναι η πυκνότητα των τρανζίστορ σε ένα ολοκληρωμένο κύκλωμα, η χωρητικότητα της DRAM και του μαγνητικού δίσκου και η απόδοση των δικτύων.

Παρά την πολύ μεγάλη ανάπτυξη των υπολογιστών, η τιμή τους μειωνόταν με την πάροδο του χρόνου σε βαθμό που πλέον οι υπολογιστές είναι διαθέσιμοι για το ευρύ κοινό όχι μόνο για εφαρμογές γενικού σκοπού αλλά και ειδικού σκοπού.

Για την μείωση του κόστους των υπολογιστών βοήθησαν τρεις παράγοντες. Ένας παράγοντας είναι το κόστος των μικροεπεξεργαστών. Να αναφέρουμε ότι όσο πιο μικρό είναι το εμβαδόν των μικροεπεξεργαστών τόσο μικρότερο είναι το κόστος τους.

Άλλος παράγοντας που βοήθησε στη μείωση του κόστους των μικροεπεξεργαστών είναι ο όγκος της παραγωγής. Όσο πιο μεγάλος είναι ο όγκος παραγωγής τόσο πιο μικρό είναι το κόστος των μικροεπεξεργαστών αλλά και το κέρδος.

Ένας άλλος παράγοντας είναι ο ανταγωνισμός μεταξύ των προμηθευτών. Ο ανταγωνισμός αυτός έχει γεφυρώσει το χάσμα ανάμεσα στο κόστος και στην τιμή πώλησης. Επίσης μειώνει το κόστος γιατί αυξάνεται ο όγκος παράγωγης.

Το αντικείμενο της παρούσης διπλωματικής εργασίας είναι η κρυφή μνήμη, μια καινοτομία στην ιεραρχία της μνήμης που αυξάνει την απόδοση των υπολογιστών. Συγκεκριμένα αναπτύξαμε έναν προσομοιωτή κρυφής μνήμης μέσω του οποίου κάποιος μπορεί εύκολα να κατανοήσει τον τρόπο λειτουργίας της κρυφής μνήμης.

Το χάσμα μεταξύ των ταχυτήτων της μνήμης και του μικροεπεξεργαστή έχει ενθαρρύνει τους αρχιτέκτονες μικροεπεξεργαστών να παρέχουν μηχανισμούς στους οποίους θα κρύβουν τις μεγάλες καθυστερήσεις μνήμης. Οι κρυφές μνήμες έχουν γίνει ο βασικός μηχανισμός που χρησιμοποιούν οι επεξεργαστές για την απόκρυψη αυτών των καθυστερήσεων και τη μείωση του μέσου χρόνου πρόσβασης στα δεδομένα.

Η κρυφή μνήμη αποτελεί διάφορα επίπεδα μιας ιεραρχίας μνήμης, η οποία είναι μια δομή που χρησιμοποιεί πολλά επίπεδα μνημών. Όσο αυξάνεται η απόσταση από τη CPU, τόσο αυξάνεται το μέγεθος και ο χρόνος προσπέλασης των μνημών.

Η ιεραρχία μνήμης βασίζεται στην αρχή της τοπικότητας (locality), η οποία αναφέρει ότι τα περισσότερα προγράμματα δεν προσπελάζουν το σύνολο του κώδικα ή των δεδομένων ομοιόμορφα. Η τοπικότητα συμβαίνει στον χρόνο και στο χώρο. Κατά τη χρονική τοπικότητα (temporal locality) εάν γίνει αναφορά σε μια θέση δεδομένων, τότε είναι πιθανόν θα

ξαναγίνει αναφορά στην ίδια θέση σύντομα. Κατά τη χωρική τοπικότητα (spatial locality) εάν γίνει αναφορά σε μια θέση δεδομένων, τότε είναι πιθανόν να γίνει σύντομα αναφορά και σε θέσεις γειτονικών δεδομένων.

Οι επεξεργαστές περιλαμβάνουν διάφορα επίπεδα κρυφής μνήμης. Για παράδειγμα ο επεξεργαστής Intel Core i7 περιλαμβάνει 3 επίπεδα (L1, L2, L3) κρυφής μνήμης.

Υπάρχουν 3 οργανώσεις κρυφής μνήμης: (α) άμεσης απεικόνισης (direct mapped), όπου ένα block της κύριας μνήμης μπορεί να τοποθετηθεί ακριβώς σε μια συγκεκριμένη θέση της κρυφής μνήμης, (β) πλήρως συσχετιστικής (fully associative), όπου ένα block μπορεί να τοποθετηθεί σε οποιαδήποτε θέση της κρυφής μνήμης και (γ) συσχετιστικής συνόλου (set-associative), όπου ένα block μπορεί να τοποθετηθεί σε έναν καθορισμένο αριθμό (τουλάχιστον δύο) θέσεων της κρυφής μνήμης.

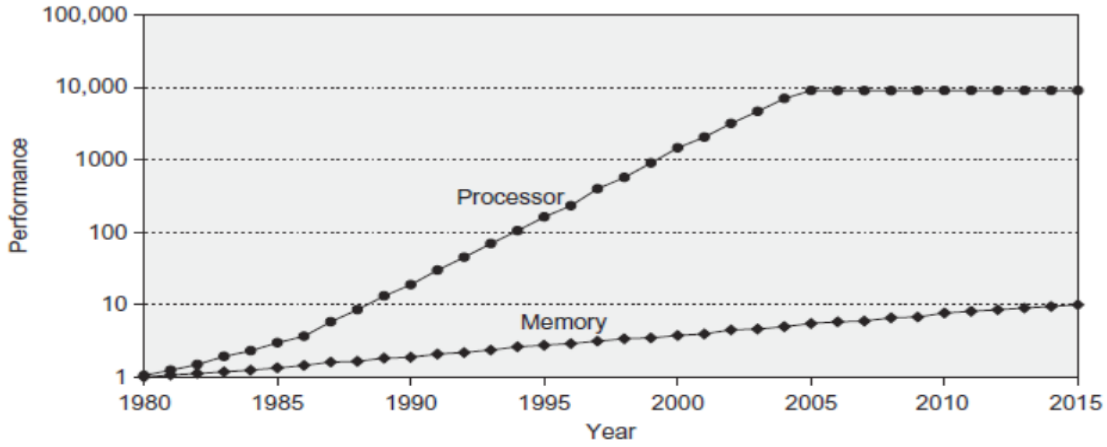
Η διπλωματική δομείται από πέντε κεφάλαια.

Στο πρώτο κεφάλαιο παρουσιάζεται η έννοια της ιεραρχίας της μνήμης, στο δεύτερο αναφέρονται οι διάφορες οργανώσεις της κρυφής μνήμης, στο τρίτο κεφάλαιο αναλύεται η σχεδίαση και ανάπτυξη του προσομοιωτή κρυφής μνήμης, στο τέταρτο κεφάλαιο παρουσιάζονται παραδείγματα χρήσης του προσομοιωτή και στο πέμπτο κεφάλαιο ολοκληρώνουμε τη διπλωματική με τα συμπεράσματά μας.

ΚΕΦΑΛΑΙΟ 2

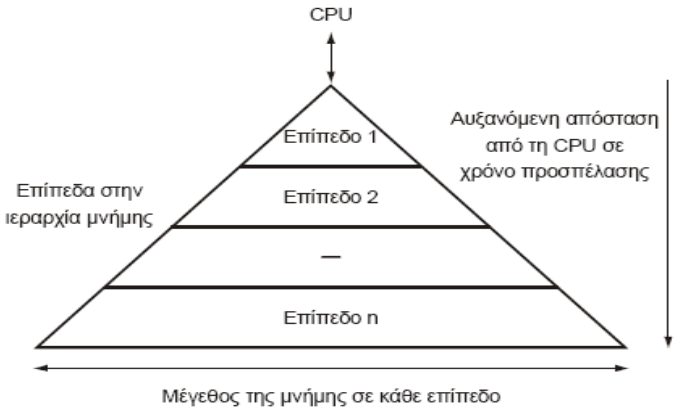
ΙΕΡΑΡΧΙΑ ΜΝΗΜΗΣ

Στο σχήμα 2.1 φαίνεται διαχρονικά η απόδοση επεξεργαστή έναντι μνήμης. Η απόδοση αυτή έως περίπου το 2004 είναι της τάξεως του 55% ανά έτος και στη συνέχεια βαίνει μειούμενη. Αντίστοιχα η απόδοση της μνήμης είναι περίπου της τάξεως 7% ανά έτος. Συμπερασματικά, υπάρχει ένα χάσμα στην απόδοση μεταξύ επεξεργαστή και μνήμης.



Σχήμα 2.1[2]: Απόδοση επεξεργαστή έναντι μνήμης

Η λύση που αποφασίστηκε και υλοποιήθηκε από τους σχεδιαστές υπολογιστικών συστημάτων ήταν η ιεραρχία μνήμης, μια δομή που χρησιμοποιεί πολλά επίπεδα μνημών. Όσο αυξάνεται η απόσταση από τη CPU, τόσο αυξάνεται το μέγεθος και ο χρόνος προσπέλασης των μνημών (Σχήμα 2.2). Αυτό οφείλεται στην κατασκευή του κάθε επιπέδου μνήμης.



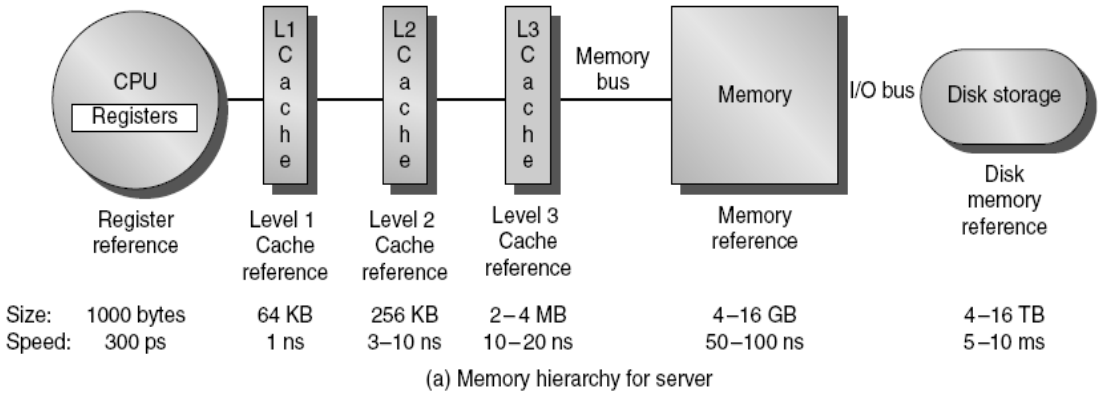
Σχήμα 2.2[2]: Ιεραρχία μνήμης

Με αυτή τη δομή μνήμης ο χρήστης έχει την ψευδαίσθηση ότι διαθέτει πολύ μεγάλη και πολύ γρήγορη μνήμη.

Στο σχήμα 2.3 φαίνονται διάφορα επίπεδα μνήμης, με την ονομασία τους, τυπικό μέγεθός τους, τεχνολογία υλοποίησης κλπ., ενώ στο σχήμα 2.4 παρουσιάζεται μια συγκεκριμένη ιεραρχία μνήμης ενός εξυπηρετητή.

Level	1	2	3	4
Name	Registers	Cache	Main memory	Disk storage
Typical size	<4 KiB	32 KiB to 8 MiB	<1 TB	>1 TB
Implementation technology	Custom memory with multiple ports, CMOS	On-chip CMOS SRAM	CMOS DRAM	Magnetic disk or FLASH
Access time (ns)	0.1–0.2	0.5–10	30–150	5,000,000
Bandwidth (MiB/sec)	1,000,000–10,000,000	20,000–50,000	10,000–30,000	100–1000
Managed by	Compiler	Hardware	Operating system	Operating system
Backed by	Cache	Main memory	Disk or FLASH	Other disks and DVD

Σχήμα 2.3 [2]: Τυπικά επίπεδα ιεραρχίας μνήμης



(a) Memory hierarchy for server

Σχήμα 2.4 [2]: Ιεραρχία μνήμης εξυπηρετητή

Η ιεραρχία μνήμης στηρίζεται στην αρχή της τοπικότητας: τα περισσότερα προγράμματα δεν προσπελάζουν το σύνολο του κώδικα ή των δεδομένων ομοιόμορφα. Η τοπικότητα συμβαίνει στον χρόνο και στο χώρο. Κατά την χρονική τοπικότητα εάν γίνει αναφορά σε μια θέση δεδομένων, τότε είναι πιθανόν θα ξαναγίνει αναφορά στην ίδια θέση σύντομα. Κατά την χωρική τοπικότητα εάν γίνει αναφορά σε μια θέση δεδομένων, τότε είναι πιθανόν να γίνει σύντομα αναφορά και σε θέσεις γειτονικών δεδομένων.

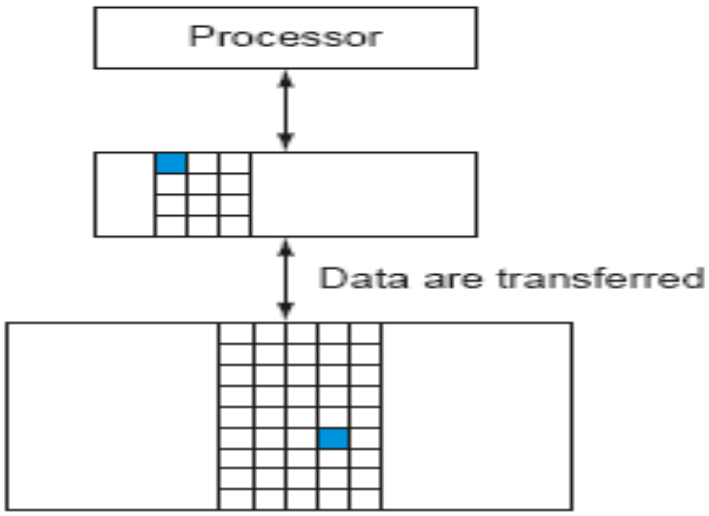
Στην ιεραρχικά δομημένη μνήμη τα δεδομένα αναζητούνται πρώτα από το πρώτο επίπεδο της μνήμης και το οποίο είναι και το ταχύτερο. Εάν τα δεδομένα είναι εντός του πρώτου επιπέδου τότε έχουμε μέγιστη ταχύτητα προσπέλασης μνήμης. Με αυτόν τον τρόπο το ‘αργό’ κατώτατο επίπεδο που υπάρχει στο σύστημα μνήμης δεν επηρεάζει την απόδοση του επεξεργαστή. Στην περίπτωση που δεν υπάρχει το ζητούμενο δεδομένο στο πρώτο επίπεδο τότε αυτό αναζητείται στα κατώτερα επίπεδα του συστήματος μνήμης. Αυτό έχει σαν

συνέπεια να αυξάνεται ο χρόνος που το δεδομένο θα είναι έτοιμο για τον επεξεργαστή όσο κατεβαίνουμε επίπεδο στην ιεραρχία μνήμης, πράγμα που μειώνει και την απόδοση του επεξεργαστή.

Για να αποφεύγεται η προσπέλαση των κατώτερων επιπέδων του συστήματος μνήμης το δεδομένο πριν διαβαστεί από τον επεξεργαστή μεταφέρεται στο ανώτερο επίπεδο. Επίσης μαζί με το ζητούμενο δεδομένο μεταφέρονται και τα 'διπλανά' από άποψη διευθυνσιοδότησης δεδομένα στο ανώτερο επίπεδο. Αυτές οι λειτουργίες γίνονται γιατί υπάρχει μεγάλη πιθανότητα το δεδομένο που ζητήθηκε αλλά και τα πλησίον δεδομένα από άποψη διευθυνσιοδότησης να ξανά αναζητηθούν από τον επεξεργαστή. Με αυτόν τον τρόπο την επόμενη φορά που θα αναζητηθούν τα δεδομένα από τον επεξεργαστή αυτά θα βρίσκονται στο πρώτο επίπεδο του συστήματος μνήμης και έτσι θα έχουμε μέγιστη ταχύτητα προσπέλασης. Ένα παράδειγμα τη παρατήρησης αυτής είναι η εκτέλεση ενός βρόχου στο πρόγραμμα. Σε αυτήν την περίπτωση εκτελούνται συνέχεια οι ίδιες εντολές και αναζητούνται τα ίδια δεδομένα. Με τον τρόπο αυτό αξιοποιείται η τοπικότητα που παρουσιάζουν τα δεδομένα στην εκτέλεση του προγράμματος.

Στη συνέχεια θα αναφέρουμε κάποιες βασικές έννοιες της ιεραρχίας μνήμης.

Block (σχήμα 2.5): είναι η ελάχιστη μονάδα πληροφορίας που μπορεί να είναι παρούσα ή όχι στην ιεραρχία δύο επιπέδων. Κάθε μπλοκ περιέχει συνήθως πολλά byte δεδομένων (το μέγεθος του μπλοκ είναι δύναμη 2). Κατά τη μεταφορά δεδομένων μεταξύ επιπέδων διαβάζεται ή γράφεται ένα ολόκληρο μπλοκ.



Σχήμα 2.5 [2]: Μπλοκ στην ιεραρχία μνήμης

Εάν τα δεδομένα που ζητούνται από τον επεξεργαστή υπάρχουν σε κάποιο block στο υψηλότερο επίπεδο, έχουμε **ευστοχία (hit)** ενώ εάν δεν υπάρχουν σε κάποιο block στο υψηλότερο επίπεδο, έχουμε **αστοχία (miss).**

Ρυθμός ευστοχίας (hit rate): το ποσοστό των προσπελάσεων μνήμης που είναι εύστοχες: ευστοχίες / συνολικές προσπελάσεις.

Ρυθμός αστοχίας (miss rate): το ποσοστό των προσπελάσεων μνήμης που είναι άστοχες, ή : 1 – ρυθμός ευστοχίας.

Χρόνος ευστοχίας (hit time): Ο χρόνος για την προσπέλαση του υψηλότερου επιπέδου στην ιεραρχία, συν το χρόνο που χρειάζεται για να προσδιοριστεί εάν η προσπέλαση είναι ευστοχία ή αστοχία.

Ποινή αστοχίας (miss penalty): Ο χρόνος για την προσκόμιση ενός block σε ένα επίπεδο της ιεραρχίας της μνήμης από ένα χαμηλότερο επίπεδο, συν το χρόνο για την προσπέλαση του block.

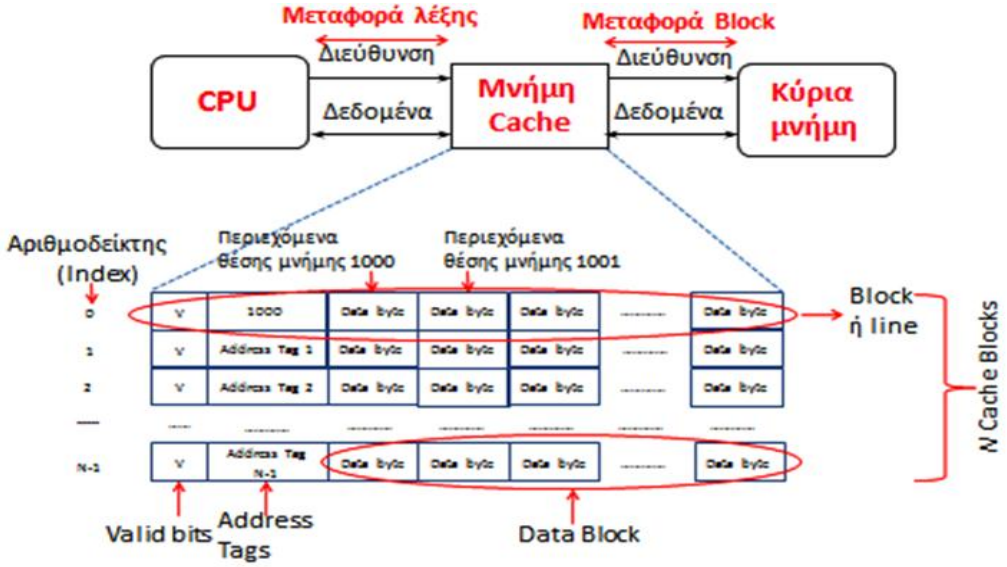
ΚΕΦΑΛΑΙΟ 3

ΟΡΓΑΝΩΣΕΙΣ ΤΗΣ ΜΝΗΜΗΣ CACHE

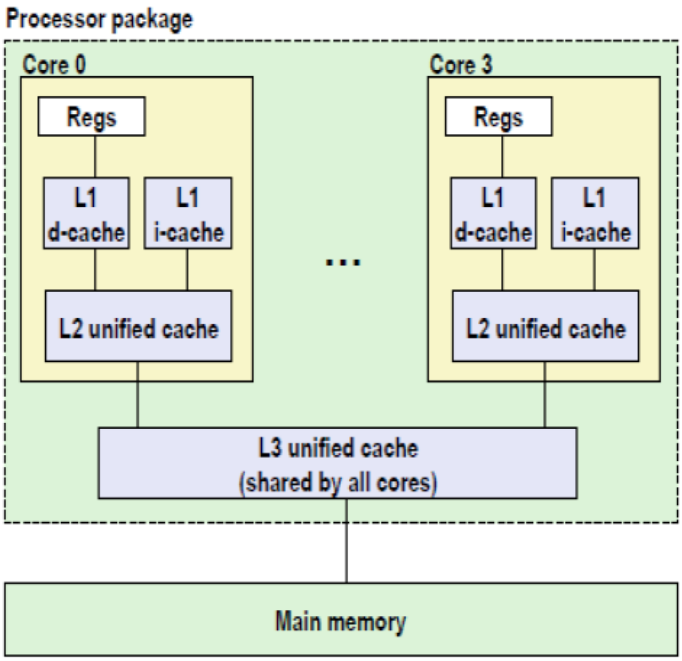
Η μνήμη cache διακρίνεται σε τρεις οργανώσεις: την άμεσης απεικόνισης, την πλήρως συσχετιστική και τη συσχετιστική συνόλου. Σε αυτό το κεφάλαιο θα αναφέρουμε τις τρεις αυτές οργανώσεις της cache, θα παρουσιάσουμε τη δομή μιας φυσικής διεύθυνσης μέσω της οποίας αναζητείται ένα δεδομένο στην cache, θα εξηγήσουμε τον τρόπο με τον οποίο η cache χειρίζεται τις εγγραφές και θα δούμε τον τρόπο που η cache μπορεί να βελτιστοποιηθεί με σκοπό να μειωθούν οι αστοχίες της.

3.1 Ετικέτες – Έγκυρα bits – Δομή διεύθυνσης μνήμης

Η δομή της κρυφής μνήμης παρουσιάζεται στο σχήμα 3.1. Βρίσκεται μέσα στο chip της CPU (σχήμα 3.2) και όταν ο επεξεργαστής θέλει να αναζητήσει κάποιο δεδομένο στέλνει τη διεύθυνσή του στην cache. Εάν βρεθεί το δεδομένο το διαβάζει, ενώ όταν δε βρεθεί η διεύθυνσή του στέλνεται στην κύρια μνήμη, όπου και μεταφέρεται στην cache μαζί με άλλα δεδομένα που συγκροτούν ένα μπλοκ. Στη συνέχεια από την cache διαβάζεται το επιθυμητό δεδομένο. Δηλαδή μεταφέρονται λέξεις μεταξύ cache και επεξεργαστή και μπλοκ μεταξύ κύριας μνήμης και cache. Εσωτερικά η cache αποτελείται από έναν αριθμό από μπλοκ. Κάθε μπλοκ χαρακτηρίζεται από μία μοναδική ετικέτα και ένα έγκυρο bit (valid bit), η τιμή του οποίου δηλώνει εάν το συγκεκριμένο μπλοκ περιέχει έγκυρα ή μη δεδομένα.



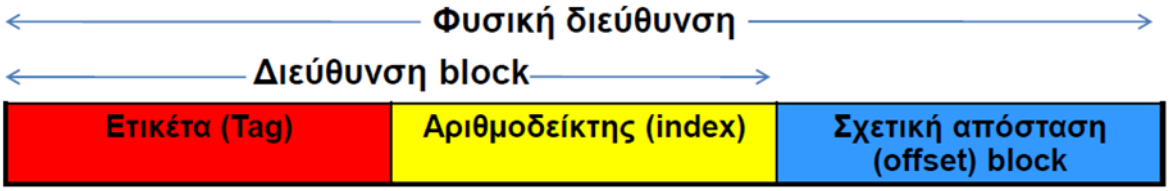
Σχήμα 3.1 [1]: Δομή της κρυφής μνήμης



Σχήμα **3.2 [3]:** Ιεραρχία της μνήμης cache στον επεξεργαστή Intel Core i7

Κατά συνέπεια, ο επεξεργαστής όταν εκτελεί ένα πρόγραμμα αναζητάει τα δεδομένα και τις εντολές του προγράμματος αρχικά στην κρυφή μνήμη και όταν δεν τα εντοπίσει, στην συνέχεια τα αναζητάει στην κύρια μνήμη. Η αναζήτηση στη μνήμη είτε πρόκειται για ανάγνωση είτε για εγγραφή γίνεται μέσω της φυσικής διεύθυνσης του ζητούμενου δεδομένου ή εντολής.

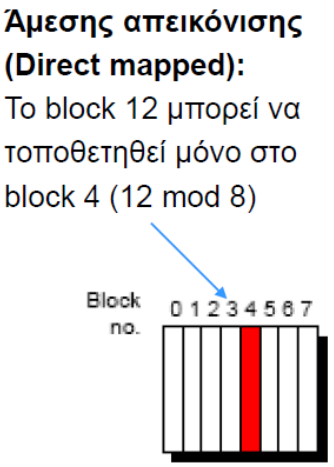
Η φυσική διεύθυνση χωρίζεται σε τρία πεδία, την ετικέτα (tag), τον δείκτη (index) και την σχετική απόσταση μπλοκ (offset) (σχήμα 3.3). Τα πεδία αυτά χρησιμοποιούνται για την αναζήτηση ενός δεδομένου στην κρυφή μνήμη είτε πρόκειται για ανάγνωση είτε για εγγραφή. Συγκεκριμένα, ο αριθμοδείκτης δηλώνει τη θέση του μπλοκ στην cache και η ετικέτα τη μοναδικότητα του μπλοκ. Επομένως, η αναζήτηση ενός δεδομένου στην cache γίνεται με το να εντοπιστεί αρχικά η θέση του μπλοκ στην cache, στη συνέχεια ελέγχεται η ετικέτα και τέλος αναζητείται το δεδομένο στο συγκεκριμένο μπλοκ.



Σχήμα 3.3 [1]: Δομή φυσικής διεύθυνσης

3.2 Cache άμεσης απεικόνισης

Στις κρυφές μνήμες άμεσης απεικόνισης (direct mapped) (σχήμα 3.4) ένα μπλοκ που μεταφέρεται από την κεντρική μνήμη μπορεί να τοποθετηθεί σε μια συγκεκριμένη θέση (μπλοκ) στην cache.



Σχήμα 3.4 [2]: Cache άμεσης απεικόνισης

Το πεδίο του δείκτη στη φυσική διεύθυνση αριθμεί τα μπλοκ της κρυφής μνήμης και αποτελείται από κ-bits της φυσικής διεύθυνσης. Ο αριθμός των bits του δείκτη υπολογίζονται από την εξίσωση (1):

$$n = log2(a) = log2(\frac{b}{c}) \qquad (1)$$

όπου n είναι ο αριθμός των bits του δείκτη, a είναι ο αριθμός των μπλοκ της κρυφής μνήμης, b είναι το μέγεθος της κρυφής μνήμης και c είναι το μέγεθος του μπλοκ.

Ένα μπλοκ μπορεί να περιέχει ένα ή περισσότερα πεδία δεδομένων ώστε να μπορούν να αποθηκευτούν πολλά διαφορετικά δεδομένα ή εντολές. Τα δεδομένα ή οι εντολές είναι αποθηκευμένα σε διαφορετική διεύθυνση στην κύρια μνήμη. Τα δεδομένα ή οι εντολές είναι γειτονικά μεταξύ τους ώστε να υποστηρίζεται η χωρική τοπικότητα που εμφανίζουν τα δεδομένα ή οι εντολές όταν τρέχει το πρόγραμμα,. Για να δείξει ο επεξεργαστής ποια από τις εντολές ή δεδομένα που περιέχει το μπλοκ χρειάζεται, χρησιμοποιείται το πεδίο της σχετικής απόστασης στη φυσική διεύθυνση. Το πεδίο της σχετικής απόστασης αριθμεί τις εντολές ή τα δεδομένα που υπάρχουν σε ένα μπλοκ. Το πεδίο της σχετικής απόστασης προκύπτει από τα τελευταία κ- bits της φυσικής διεύθυνσης τα οποία υπολογίζονται από την εξίσωση (2):

$$n = log2(a) \qquad (2)$$

όπου n είναι ο αριθμός των bits της σχετικής απόστασης και a είναι το μέγεθος του μπλοκ. Τέλος το πεδίο της ετικέτας δεν αριθμεί κάτι. Χρησιμοποιείται για την ταυτοποίηση του μπλοκ και προκύπτει από τα πρώτα κ- bits της φυσικής διεύθυνσης. Πιο συγκεκριμένα ο υπολογισμός των bits της ετικέτας ισούστε με

bits_ετικέτας = bits_φυσικής_διεύθυνσις – (bits_δείκτη + bits_σχετικής_απόστασης)

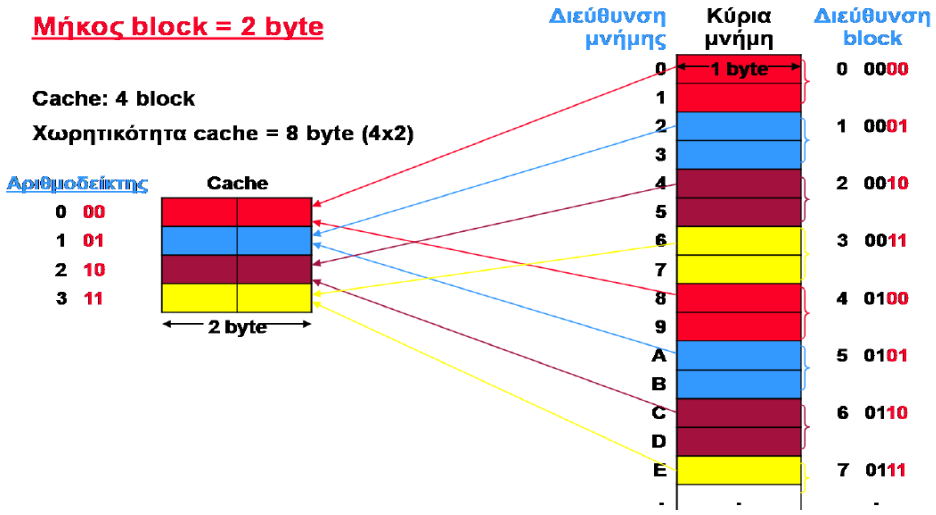
Όταν ένα δεδομένο ή εντολή που αναζητείται δεν υπάρχει στην κρυφή μνήμη (εμφάνιση αστοχίας στην κρυφή μνήμη) τότε αυτό θα αντιγραφεί ή θα αντικατασταθεί (αναλόγως ποια τεχνική ακολουθείται (πολυεπίπεδη συμπερίληψη ή πολυεπίπεδος αποκλεισμός)) στην κρυφή μνήμη. Στην κρυφή μνήμη άμεσης απεικόνισης το δεδομένο ή η εντολή τοποθετείται σε συγκεκριμένο μπλοκ που ορίζεται από την φυσική διεύθυνση. Πιο συγκεκριμένα στην περίπτωση που θέλουμε να μεταφέρουμε ένα δεδομένο από την κύρια μνήμη προς την κρυφή μνήμη τότε αυτό θα τοποθετηθεί στο μπλοκ που ορίζεται από την εξής εξίσωση (3):

$$a1 = a2 mod b \quad (3)$$

οπού α1 είναι η διεύθυνση του μπλοκ της κρυφής μνήμης, α2 είναι η διεύθυνση της κύριας μνήμης και b είναι ο αριθμός των μπλοκ της κρυφής μνήμης άμεσης απεικόνισης.

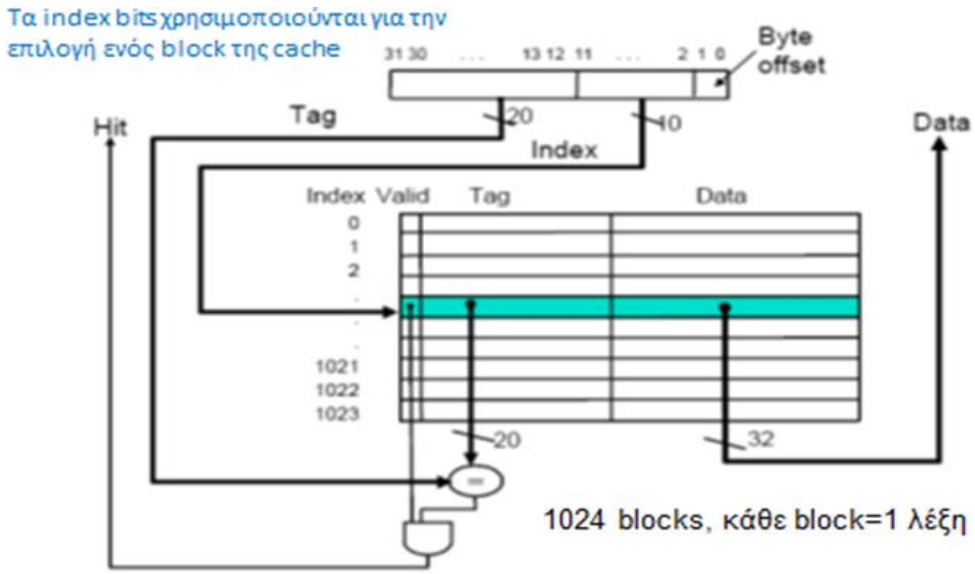
Οι διευθύνσεις του παραπάνω τύπου είναι σε δεκαδική μορφή. Μπορούμε όμως να βρούμε το μπλοκ της κρυφής μνήμης που θα μεταφερθεί το δεδομένο από την κύρια μνήμη και μέσω της δυαδικής διεύθυνσης, χωρίς πράξεις, ως εξής (σχήμα 3.5): Βρίσκουμε το πεδίο δείκτη που αντιστοιχεί στην κρυφή μνήμη από την διεύθυνση της κύριας μνήμης με τον τρόπο που εξηγήσαμε παραπάνω. Βρίσκουμε δηλαδή τον αριθμό των μπιτ που αποτελείται ο δείκτης μέσω της φυσικής διεύθυνσης. Έπειτα απομονώνουμε τον δείκτη από την φυσική διεύθυνση. Ο δείκτης αυτός δείχνει το μπλοκ της κρυφής μνήμης άμεσης απεικόνισης που θα τοποθετηθεί το δεδομένο από την κύρια μνήμη.

Η ίδια λογική ισχύει και εάν μεταφέραμε ένα δεδομένο ή εντολή από την κρυφή μνήμη δεύτερου επίπεδου στην κρυφή μνήμη πρώτου επιπέδου. Η μόνη διαφορά είναι ότι θα χρησιμοποιούσαμε την διευθυνσιοδότηση της κρυφής μνήμης δεύτερου επίπεδου.



Σχήμα 3.5 [1]: Μνήμη cache άμεσης απεικόνισης

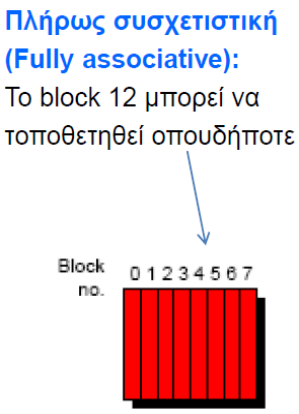
Για την αναζήτηση ενός μπλοκ της κρυφής μνήμης άμεσης απεικόνισης γίνεται η εξής λειτουργία. Για αρχή ο δείκτης της φυσικής διεύθυνσης δείχνει σε ένα συγκεκριμένο μπλοκ της κρυφής μνήμης. Στο επόμενο βήμα συγκρίνεται η ετικέτα τη φυσικής διεύθυνσης με αυτή που περιέχει το μπλοκ στο οποίο δείχνει ο δείκτης. Εάν είναι ίδιες τότε το ο επεξεργαστής λαμβάνει το δεδομένο που 'δείχνει' η σχετική απόσταση. Στο σχήμα 3.6 φαίνεται η υλοποίηση μιας μνήμης cache άμεσης απεικόνισης.



Σχήμα 3.6 [2]: Υλοποίηση μνήμης cache άμεσης απεικόνισης

3.3 Πλήρως συσχετιστική Cache

Στις πλήρως συσχετιστικές κρυφές μνήμες (fully associative) (σχήμα 3.7) ένα μπλοκ που μεταφέρεται από την κύρια μνήμη μπορεί να τοποθετηθεί σε οποιαδήποτε θέση (μπλοκ) στην cache.



Σχήμα **3.7 [2]**: Πλήρως συσχετιστική cache

Σε αυτή την περίπτωση δεν απαιτείται να έχουμε πεδίο δείκτη για τον προσδιορισμό της θέσης που θα μεταφερθεί ένα μπλοκ. Έχουμε όμως πεδίο σχετικής απόστασης και υπολογίζεται με τον ίδιο τρόπο που έχουμε δείξει την κρυφή μνήμη άμεσης απεικόνισης.

Το πεδίο της ετικέτας χρησιμοποιείται για την ταυτοποίηση των μπλοκ της πλήρως συσχετικής κρυφής μνήμης και υπολογίζεται ως εξής:

bits_ετικέτας = bits_φυσικής διεύθυνσης – bits_σχετικής_απόστασης (4)

Όταν ένα δεδομένο ή εντολή που αναζητείται δεν υπάρχει στην κρυφή μνήμη (εμφάνιση αστοχίας στην κρυφή μνήμη) τότε αυτό θα αντιγραφεί ή θα αντικατασταθεί (αναλόγως ποια τεχνική ακολουθείται (πολυεπίπεδη συμπερίληψη ή πολυεπίπεδος αποκλεισμός)) στην κρυφή μνήμη. Στην πλήρως συσχετική κρυφή μνήμη το δεδομένο ή η εντολή τοποθετείται οπουδήποτε στην κρυφή μνήμη, ανεξαρτήτως της διεύθυνσής του, όπως γινόταν πχ στην κρυφή μνήμη άμεσης απεικόνισης. Στην περίπτωση που υπάρχει ‘άδειο’ μπλοκ τότε το δεδομένο ή η εντολή μεταφέρεται στο ‘άδειο’ μπλοκ. Εάν δεν υπάρχει τότε αυτό θα αντικαταστήσει ένα μπλοκ με βάση την τεχνική αντικατάστασης του μπλοκ.

Εάν ακολουθηθεί η τυχαία αντικατάσταση τότε αντικαταστείται ένα τυχαίο μπλοκ. Εάν ακολουθηθεί η τεχνική FIFO (First In First Out) τότε αντικαταστείται το μπλοκ που τοποθετήθηκε πρώτο (ανάμεσα στα μπλοκ που υπάρχουν στην κρυφή μνήμη) στην κρυφή μνήμη. Τέλος έχουμε την τεχνική LRU (Last Recently Used). Εδώ αντικαταστείται το μπλοκ που χρησιμοποιήθηκε τελευταίο. Η διαφορά με την τεχνική FIFO είναι ότι η LRU ελέγχει τα

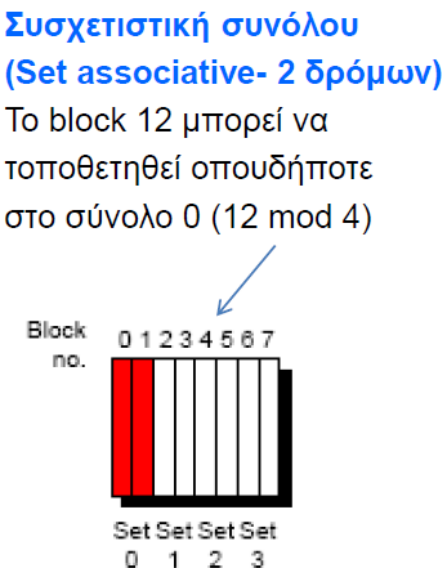
ζητούμενα μπλοκ και εφόσον αυτά υπάρχουν ήδη στην κρυφή μνήμη και όχι μόνο όταν ‘εισέρχονται’. Η LRU εκμεταλλεύεται την τοπικότητα που μπορεί να παρουσιάζουν τα δεδομένα ή οι εντολές, αλλά είναι πιο πολύπλοκος ο σχεδιασμός της.

Για την αναζήτηση ενός μπλοκ στη πλήρως συσχετική κρυφή μνήμη γίνεται η εξής λειτουργία. Συγκρίνεται η ετικέτα της φυσικής διεύθυνσης με αυτή που περιέχει κάθε μπλοκ της κρυφής μνήμης. Εάν είναι ίδιες τότε ο επεξεργαστής λαμβάνει το δεδομένο που ‘δείχνει’ η σχετική απόσταση.

Πέρα από το πεδίο της ετικέτας, ένα μπλοκ μπορεί να περιεχέι και το bit εγκυρότητας. Η λειτουργία του bit εγκυρότητας εμφανίζεται στις περιπτώσεις των πολυεπεξεργαστών. Σε έναν πολυεπεξεργαστή ένα δεδομένο μπορεί να υπάρχει σε περισσότερους από έναν επεξεργαστή (τότε το δεδομένο είναι διαμοιραζόμενο), πράγμα που σημαίνει ότι το δεδομένο βρίσκεται σε περισσότερες της μιας κρυφής μνήμης. Ο κάθε επεξεργαστής ‘βλέπει’ μόνο την δικιά του κρυφή μνήμη. Στην περίπτωση που ένας άλλος επεξεργαστής έχει τροποποιήσει την τιμή ενός διαμοιραζομένου δεδομένου, πρέπει τότε να αποτραπεί από τους υπόλοιπους επεξεργαστές να διαβάσουν αυτό το διαμοιραζόμενο δεδομένο από την κρυφή μνήμη τους, γιατί αυτό περιέχει λανθασμένη τιμή (παλιά τιμή). Αυτό γίνεται μέσω του bit εγκυρότητας. Εάν bit εγκυρότητας = 0 τότε εμφανίζεται αστοχία στην ανάγνωση του μπλοκ αυτού ακόμα και εάν ο δείκτης και η ετικέτα του μπλοκ ταιριάζουν με αυτές της φυσικής διεύθυνσης.

3.4 Συσχετιστική συνόλου cache

Μια άλλη οργάνωση κρυφής μνήμης είναι η συσχετιστική κρυφή μνήμη συνόλου (set associative) (σχήμα 3.8). Στις συσχετικές κρυφές μνήμες τα μπλοκ χωρίζονται σε σύνολα. Οι εντολές ή τα δεδομένα μπορούν να τοποθετηθούν σε ένα συγκεκριμένο σύνολο, αλλά σε οποιοδήποτε μπλοκ εντός του συνόλου αυτού. Αν υπάρχουν **n** μπλοκς σε ένα σύνολο, τότε έχουμε μια συσχετιστική μνήμη cache **n δρόμων** (n-way).



Σχήμα 3.8: [2]: Συσχετιστική συνόλου cache

Το πεδίο του δείκτη αριθμεί τα σύνολα της κρυφής μνήμης και αποτελείται από κ-bits της φυσικής διεύθυνσης. Ο αριθμός των bits του δείκτη υπολογίζονται από την εξίσωση (5):

$$n = log2(a) = log2(\frac{b}{c}) \quad (5)$$

όπου n είναι ο αριθμός των bits του δείκτη, a είναι ο αριθμός των συνόλων της κρυφής μνήμης, b είναι ο συνολικός αριθμός των μπλοκ της κρυφής μνήμης και c είναι ο αριθμός των μπλοκ κάθε σύνολου της κρυφής μνήμης.

Το πεδίο της ετικέτας και της σχετικής απόστασης αντιπροσωπεύουν και υπολογίζονται με τον ίδιο τρόπο όπως στην κρυφή μνήμη άμεσης απεικόνισης.

Όταν ένα δεδομένο ή εντολή που αναζητείται δεν υπάρχει στην κρυφή μνήμη (εμφάνιση αστοχίας στην κρυφή μνήμη) τότε αυτό θα αντιγραφεί ή θα αντικατασταθεί (αναλόγως ποια τεχνική ακολουθείται (πολυεπίπεδη συμπερίληψη ή πολυεπίπεδος αποκλεισμός)) στην κρυφή μνήμη. Στην συσχετική κρυφή μνήμη το δεδομένο ή η εντολή τοποθετείται σε συγκεκριμένο σύνολο που ορίζεται από την διεύθυνση. Πιο συγκεκριμένα στην περίπτωση που θέλουμε να μεταφέρουμε ένα δεδομένο από την κύρια μνήμη προς την κρυφή μνήμη τότε αυτό θα τοποθετηθεί στο σύνολο που ορίζεται από τον εξής εξίσωση (6):

$$a1 = a2 mod b \quad (6)$$

όπου α1 είναι η διεύθυνση του μπλοκ της κρυφής μνήμης, α2 είναι η διεύθυνση της κύριας μνήμης και b είναι ο αριθμός των συνόλων της συσχετικής κρυφής μνήμης.

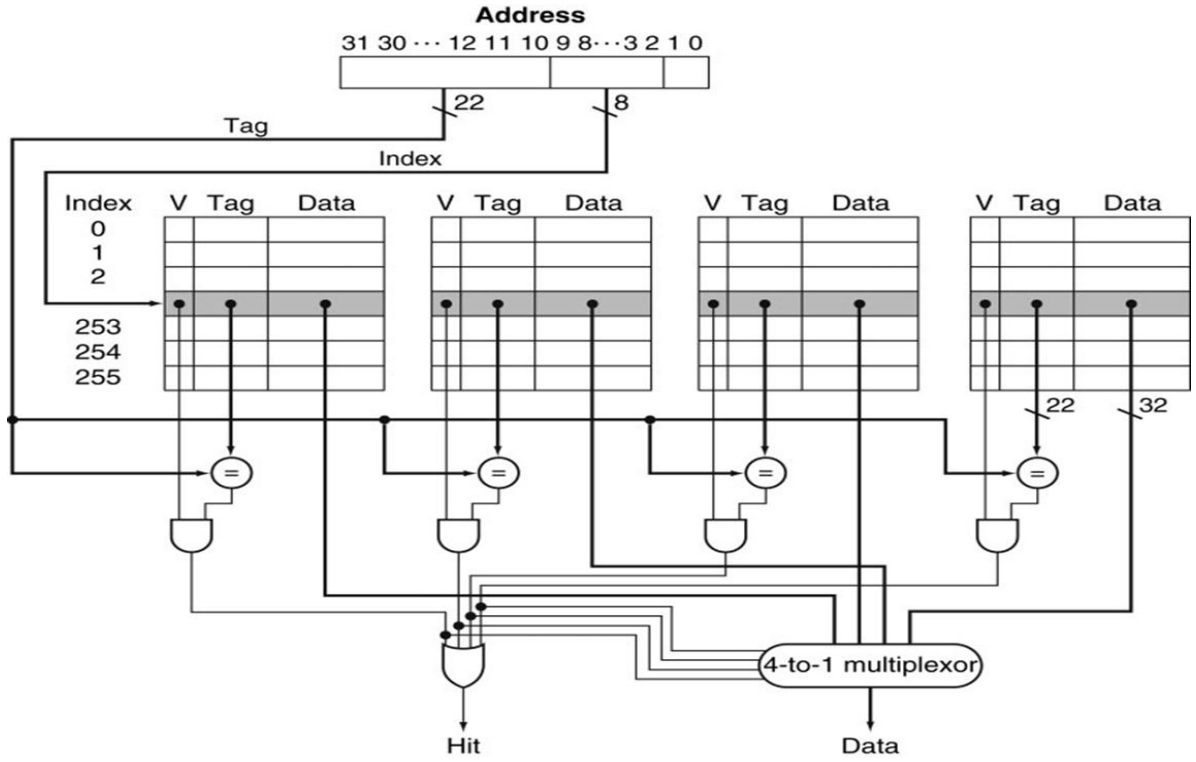
Οι διευθύνσεις του παραπάνω τύπου είναι σε δεκαδική μορφή. Μπορούμε όμως να βρούμε το σύνολο της κρυφής μνήμης που θα μεταφερθεί το δεδομένο από την κύρια μνήμη και μέσω της δυαδικής διεύθυνσης, χωρίς πράξεις, ως εξής. Βρίσκουμε το πεδίο δείκτη που αντιστοιχεί στην κρυφή μνήμη από την διεύθυνση της κυρίας μνήμης με τον τρόπο που εξηγήσαμε παραπάνω. Βρίσκουμε δηλαδή τον αριθμό των bits που αποτελείται ο δείκτης μέσω της φυσικής διεύθυνσης. Έπειτα απομονώνουμε τον δείκτη από την φυσική διεύθυνση. Ο δείκτης αυτός δείχνει το σύνολο της συσχετικής κρυφής μνήμης που θα τοποθετηθεί το δεδομένο από την κύρια μνήμη.

Η ίδια λογική ισχύει και εάν μεταφέραμε ένα δεδομένο ή εντολή από την κρυφή μνήμη δευτέρου επιπέδου στην κρυφή μνήμη πρώτου επιπέδου. Η μόνη διαφορά είναι ότι θα χρησιμοποιούσαμε την διευθυνσιοδότηση της κρυφής μνήμης δευτέρου επίπεδου.

Στην περίπτωση όμως της συσχετικής κρυφής μνήμης πρέπει να οριστεί και μια μεθοδολογία αντικατάστασης των μπλοκς ενός συνόλου. Ενώ πριν αναφέρθηκε ένας τρόπος για να επιλεγεί το σύνολο που θα μεταφερθεί το δεδομένο ή η εντολή από την κύρια μνήμη στην κρυφή μνήμη, δεν αναφέρθηκε πως επιλέγεται το μπλοκ εντός του συνόλου που θα μεταφερθεί το δεδομένο ή η εντολή. Στην περίπτωση που υπάρχει ‘άδειο’ μπλοκ τότε το δεδομένο ή η εντολή μεταφέρεται στο ‘άδειο’ μπλοκ. Εάν δεν υπάρχει τότε αυτό θα αντικαταστήσει ένα μπλοκ με βάση την τεχνική αντικατάστασης του μπλοκ (τυχαία, FIFO, ή LRU).

Για την αναζήτηση ενός μπλοκ στην συσχετική κρυφή μνήμη γίνεται η εξής λειτουργία . Για αρχή ο δείκτης της φυσικής διεύθυνσης δείχνει σε ένα συγκεκριμένο σύνολο της κρυφής μνήμης. Στο επόμενο βήμα συγκρίνεται η ετικέτα τη φυσικής διεύθυνσης με αυτή που περιέχει κάθε μπλοκ του συνόλου στο οποίο δείχνει ο δείκτης. Εάν είναι ίδιες τότε ο επεξεργαστής λαμβάνει το δεδομένο που ‘δείχνει’ η σχετική απόσταση.

Στο σχήμα 3.9 παρουσιάζεται η υλοποίηση μιας συσχετιστικής κρυφής μνήμης 4 δρόμων.



Σχήμα 3.9[2]: Υλοποίηση συσχετιστικής κρυφής μνήμης 4 δρόμων.

3.5 Χειρισμός εγγραφών στην cache

Υπάρχουν πολλά αντίγραφα δεδομένων (μπλοκ) στα διάφορα επίπεδα της κρυφής μνήμης L1, L2, L3 ακόμη και στην κεντρική μνήμη. Στην περίπτωση που ο επεξεργαστής θέλει να τροποποιήσει κάποια θέση, δηλαδή να κάνει μια εγγραφεί, θα πρέπει να δούμε η θέση που θα τροποποιηθεί που βρίσκεται. Μπορεί να βρίσκεται στην κρυφή μνήμη ή στην κύρια μνήμη.

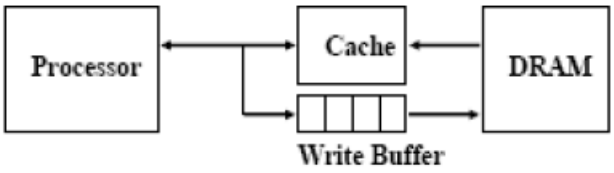
Όταν η θέση που θα εγγραφή βρίσκεται στην κρυφή μνήμη υπάρχουν δύο προσεγγίσεις για την εγγραφή της: ταυτόχρονη εγγραφή (write through) και ετερόχρονη εγγραφή (write back).

Στην ταυτόχρονη εγγραφή οι εγγραφές ενημερώνουν πάντοτε τόσο την κρυφή μνήμη όσο και την κεντρική μνήμη. Με αυτόν τον τρόπο η κρυφή μνήμη και η κύρια μνήμη είναι συνεπείς (consistent), όμως από την άλλη παρατηρείται επιβάρυνση της απόδοσης.

Στην ετερόχρονη εγγραφή οι εγγραφές ενημερώνουν μόνο την κρυφή μνήμη (το υψηλότερο επίπεδο) και το τροποποιημένο μπλοκ γράφεται στο χαμηλότερο επίπεδο της ιεραρχίας μόνο όταν αντικαθίσταται. Για τον σκοπό αυτό χρησιμοποιείται ένα dirty bit για τα τροποποιημένα μπλοκ. Η κρυφή και η κύρια μνήμη είναι ασυνεπείς και ο τρόπος υλοποίησης είναι πιο πολύπλοκος από αυτόν της ταυτόχρονης εγγραφής. Τα πλεονεκτήματα της ετερόχρονης εγγραφής είναι η μεγαλύτερη ταχύτητα εγγραφής λόγω ότι προσπελάζεται μόνο η κρυφή μνήμη, η μείωση του εύρους ζώνης του μέσου διάδοσης λόγω ότι μειώνεται η ανάγκη να

προσπελαστεί η κύρια μνήμη από πολλούς επεξεργαστές και η λιγότερη κατανάλωση ισχύος λόγω ότι η εγγραφή χρειάζεται λιγότερους κύκλους.

Και στις δυο περιπτώσεις χρησιμοποιείται ένας buffer (απομονωτής εγγραφής - write buffer) (σχήμα 3.10) στον οποίο μπορούν να αποθηκευτούν αρκετά μπλοκ και ο οποίος είναι υπεύθυνος για την αποθήκευση των μπλοκ που περιέχει προς στην κύρια μνήμη. Η λειτουργία του buffer γίνεται παράλληλα με τις λειτουργίες της κρυφής μνήμης και ο σκοπός του είναι να μην βρίσκεται σε αναμονή η κρυφή μνήμη γιατί τότε μπορεί να βρεθεί σε αναμονή και ο επεξεργαστής. Βέβαια όταν ο buffer είναι γεμάτος τότε η κρυφή μνήμη βρίσκεται σε αναμονή.



Σχήμα 3.10 [2]: Προσωρινή μνήμη εγγραφής

Όταν η θέση που πρόκειται να εγγραφεί δεν βρίσκεται στην κρυφή μνήμη υπάρχουν δύο προσεγγίσεις για την εγγραφή της: no-write-allocate, όπου το μπλοκ ενημερώνεται στη μνήμη και δεν μεταφέρεται στην κρυφή μνήμη και write-allocate, όπου το μπλοκ ενημερώνεται στη μνήμη και μετά μεταφέρεται στην κρυφή μνήμη. Με αυτόν τον τρόπο μπορεί να ξαναχρησιμοποιήσουμε αυτό το μπλοκ σύντομα, αλλά εάν δεν ξαναχρησιμοποιηθεί δημιουργείται καθυστέρηση. Η τάση είναι ότι σε όλα τα επίπεδα χρησιμοποιείται ετερόχρονη εγγραφή και write-allocate.

Μπορεί στα μονοεπεξεργαστικά συστήματα να μην έχει νόημα τόσο η επιλογή του τρόπου εγγραφής των δεδομένων από την κρυφή μνήμη προς την κύρια μνήμη αφού για να αποφευχθεί η σπατάλη χρόνου από τον επεξεργαστή θα επιλεγόταν η ετερόχρονη εγγραφή, αλλά η διαδικασία αυτή παίζει σημαντικό ρόλο στην σχεδίαση πολυεπεξεργαστών διαμοιραζόμενης μνήμης.

3.6 Αστοχίες στην κρυφή μνήμη και βελτιστοποίηση

Σε αυτήν την ενότητα θα εξηγήσουμε τα είδη αστοχιών και τους τρόπους μείωσης των αστοχιών. Η ενότητα αυτή δίνει μια εξήγηση στη σημαντικότητα της παραμετροποίησης τους.

Οι λόγοι που μπορεί να εμφανιστεί αστοχία είναι διαφορετικοί και οφείλονται σε λόγους σχεδιασμού. Όπως όμως θα εξηγήσουμε και παρακάτω οι αστοχίες εξαρτώνται από πολλούς παράγοντες. Αυτό έχει ως αποτέλεσμα όταν επηρεάζεται ένας παράγοντας, ενώ από την μια υπάρχει βελτίωση σε ένα είδος αστοχίας, από την άλλη χειροτερεύει ένα άλλο είδος αστοχίας.

Οι αστοχίες που εμφανίζονται στην κρυφή μνήμη χωρίζονται σε διάφορες κατηγορίες αναλόγως την αιτία εμφάνισης τους. Ένα είδος αστοχίας είναι οι υποχρεωτικές αστοχίες. Η υποχρεωτική αστοχία είναι η αστοχία που συμβαίνει όταν προσπελάζεται ένα μπλοκ για πρώτη φορά και θα πρέπει να μεταφερθεί από τα χαμηλότερα επίπεδα της ιεραρχίας. Οι αστοχίες αυτές είναι αναγκαστικές και δεν σχετίζονται από την σχεδίαση της κρυφής μνήμης.

Ένα άλλο είδος αστοχίας είναι οι αστοχίες χωρητικότητας. Οι αστοχίες χωρητικότητας οφείλονται λόγω ότι η κρυφή μνήμη δεν μπορεί να αποθηκεύσει όλα τα απαιτούμενα δεδομένα ή εντολές και άρα πρέπει να αντικατασταθούν τα ήδη αποθηκευμένα μπλοκ στην κρυφή μνήμη. Αυτό έχει ως συνέπεια να αντικατασταθούν μπλοκ που μπορεί να αναζητηθούν αργότερα. Οι αστοχίες χωρητικότητας μειώνονται όσο αυξάνουμε την χωρητικότητα της κρυφής μνήμης.

Άλλο είδος αστοχίας είναι οι αστοχίες σύγκρουσης ή διαμάχης. Οι αστοχίες αυτές οφείλονται στο γεγονός ότι λόγω λειτουργίας της κρυφής μνήμης θα αντικατασταθούν συγκεκριμένα μπλοκ αντί να τοποθετηθούν σε άδεια μπλοκ. Αυτό έχει ως συνέπεια να αντικατασταθούν μπλοκ που μπορεί να αναζητηθούν αργότερα. Οι αστοχίες σύγκρουσης μειώνονται όσο αυξάνουμε τη σχετικότητα της κρυφής μνήμης.

Άλλο ένα είδος αστοχίας με το οποίο δεν θα ασχοληθούμε είναι οι αστοχίες συνοχής και εμφανίζονται σε πολυεπεξεργαστές.

Τώρα θα δούμε με ποιους τρόπους μπορούμε να βελτιστοποιήσουμε την κρυφή μνήμη, επηρεάζοντας όμως αρνητικά σε άλλα σημεία την κρυφή μνήμη. Για αρχή θα δούμε πως μπορούμε να μειώσουμε τον ρυθμό αστοχίας της κρυφής μνήμης:

- Ένας τρόπος για να μειώσουμε τον ρυθμό αστοχίας είναι αυξάνοντας τη χωρητικότητα του μπλοκ σε μέγεθος. Αυτό έχει ως συνέπεια όταν τοποθετείται ένα δεδομένο ή εντολή στην κρυφή μνήμη τότε να τοποθετούνται και ‘διπλανά’ (από άποψη διεύθυνσης) δεδομένα ή εντολές στην κρυφή μνήμη ώστε να ΄γεμίσουν’ όλα τα πεδία δεδομένων του μπλοκ της κρυφής μνήμης. Αυτό έχει ως αποτέλεσμα να

είναι διαθέσιμα στην κρυφή μνήμη δεδομένα ή εντολές που μπορεί να χρειαστούν αργότερα. Αξιοποιούμε έτσι την χωρική τοπικότητα. Η αύξηση όμως του μπλοκ σε μέγεθος έχει ως αποτέλεσμα την μείωση των αριθμό των μπλοκ στην κρυφή μνήμη (η αύξηση του μεγέθους του μπλοκ δεν σημαίνει αύξηση στο μέγεθος της κρυφής μνήμης). Αυτό έχει ως συνέπεια να αυξάνονται οι αστοχίες σύγκρουσης.

- Άλλος τρόπος μείωσης του ρυθμού αστοχίας είναι να μεγαλώσουμε τη συσχετικότητα της κρυφής μνήμης. Η αύξηση της συσχετικότητας μειώνει τις αστοχίες σύγκρουσης. Η αύξηση όμως της συσχετικότητας αυξάνει τον χρόνο ευστοχίας και την κατανάλωση ισχύος γιατί απαιτούνται περισσότεροι έλεγχοι για την αναζήτηση του μπλοκ καθώς και πολυπλοκότερα κυκλώματα ελέγχου. Για παράδειγμα η αύξηση της συσχετικότητας (λόγω αύξησης των μπλοκς) απαιτεί έναν πολυπλοκότερο πολυπλέκτη γιατί αυτός πρέπει να επιλέξει ανάμεσα σε περισσότερα μπλοκς σε ένα σύνολο με βάση την ετικέτα που αναζητείται και φυσικά την εμφάνιση ευστοχίας/αστοχίας. Αυτό έχει ως συνέπεια να πρέπει να επιμηκυνθεί ο χρόνος κύκλου του ρολογιού ώστε να ‘βγει’ το αποτέλεσμα του πολυπλέκτη εντός του κύκλου του ρολογιού. Επίσης αυξάνεται και η κατανάλωση ισχύος λόγω των περισσότερων λογικών στοιχείων που απαιτούνται για τον έλεγχο των μπλοκς τη κρυφής μνήμης.
- Παρόμοια άλλος τρόπος μείωσης του ρυθμού αστοχίας είναι η αύξηση του μεγέθους της κρυφής μνήμης, με αποτέλεσμα τη μείωση των αστοχιών χωρητικότητας. Η αύξηση όμως της κρυφής μνήμης σε μέγεθος αυξάνει τους απαιτούμενους ελέγχους και τα στοιχεία των κυκλωμάτων έλεγχου με αποτέλεσμα να αυξάνεται ο χρόνος ευστοχίας και η κατανάλωση ισχύος.

Τώρα θα παρουσιάσουμε τρόπους βελτιστοποίησης της κρυφής μνήμης μειώνοντας την ποινή αστοχίας:

- Μπορούμε να μειώσουμε την ποινή αστοχίας χρησιμοποιώντας πολυεπίπεδη κρυφή μνήμη. Η λογική είναι σε περίπτωση αστοχίας της κρυφής μνήμης πρώτου επίπεδου, να αναζητηθεί το δεδομένο ή η εντολή στην κρυφή μνήμη δεύτερου επιπέδου η οποία είναι πιο γρήγορη από άποψη προσπέλασης από ότι η κύρια μνήμη. Με αυτόν τον τρόπο μειώνεται η ποινή αστοχίας.
- Ένας άλλος τρόπος μείωσης της ποινής αστοχίας είναι η αποφυγή της αναμονής μιας εντολής ανάγνωσης μέχρι να ολοκληρωθεί μια εντολή εγγραφής. Η λύση στο

πρόβλημα αυτό είναι ο απομονωτής εγγραφής που εξηγήσαμε παραπάνω. Σε αυτήν την περίπτωση την εγγραφή την αναλαμβάνει ο απομονωτής εγγραφής και συνεχίζεται η ανάγνωση. Το πρόβλημα που μπορεί να υπάρξει είναι εάν η εντολή ανάγνωσης απαιτεί το δεδομένο που υπάρχει στον απομονωτή εγγραφής. Ένας τρόπος αντιμετώπισης του προβλήματος αυτού είναι να περιμένουμε να αδειάσει ο απομονωτής εγγραφής, αλλά έτσι δεν κερδίζουμε χρόνο. Άλλος τρόπος είναι να ελέγχεται εάν το δεδομένο που απαιτείται για ανάγνωση υπάρχει στον απομονωτή εγγραφής. Εάν δεν υπάρχει τότε συνεχίζεται η ανάγνωση από την κύρια μνήμη. Εάν υπάρχει τότε αναμένεται το άδειασμα του απομονωτή εγγραφής.

Τρόποι βελτιστοποίησης της κρυφής μνήμης μειώνοντας τον χρόνο ευστοχίας:

Ένας λόγος αύξησης του χρόνου ευστοχίας είναι ότι απαιτείται η μετάφραση των ιδεατών διευθύνσεων σε φυσικές διευθύνσεις. Η χρήση των ιδεατών κρυφών μνημών είναι αναγκαία, οπότε είναι δύσκολη η μη χρησιμοποίηση τους ώστε να μειωθεί ο χρόνος ευστοχίας. Ένας άλλος τρόπος είναι η χρήση μόνο ιδεατών κρυφών μνημών αποφεύγοντας έτσι την μετάφραση των ιδεατών διευθύνσεων σε φυσικές διευθύνσεις. Υπάρχουν όμως κάποιοι λόγοι που αποτρέπουν την τελευταία περίπτωση. Ένας λόγος είναι ότι η μετάφραση της ιδεατής διεύθυνσης είναι κομμάτι της προστασίας των σελίδων. Μια λύση είναι η προσθήκη ενός πεδίου στην ιδεατή κρυφή μνήμη, το οποίο να περιέχει κάποιες πληροφορίες προστασίας και το πεδίο αυτό να ελέγχεται στην προσπέλαση τη ιδεατής κρυφής μνήμης. Άλλος λόγος είναι ότι μπορεί να χρησιμοποιούνται διαφορετικές ιδεατές διευθύνσεις για την ίδια φυσική διεύθυνση. Αυτές οι δυο ιδεατές διευθύνσεις ονομάζονται ψευδώνυμα (aliasis). Με χρήση φυσικής κρυφής μνήμης δεν υπάρχει πρόβλημα γιατί οι ιδεατές διευθύνσεις οδηγούν στην ίδια φυσική διεύθυνση. Εάν όμως δεν έχουμε φυσική κρυφή μνήμη τότε στην περίπτωση που τροποποιηθεί ένα δεδομένο στην μια ιδεατή διεύθυνση τότε η άλλη ιδεατή διεύθυνση θα περιέχει λανθασμένη τιμή γιατί δεν είναι ίδια με αυτή της άλλης ιδεατής διεύθυνσης. Μια λύση σε επίπεδο υλικού που χρησιμοποιείται, εξασφαλίζει ότι κάθε μπλοκ της κρυφής μνήμης θα διαθέτει μοναδική φυσική διεύθυνση.

ΚΕΦΑΛΑΙΟ 4

Ο ΠΡΟΣΟΜΟΙΩΤΗΣ CACHE

Στο κεφάλαιο αυτό θα παρουσιάσουμε την εφαρμογή που αναπτύξαμε, επεξηγώντας τον τρόπο λειτουργίας της, το περιβάλλον της, τον τρόπο ορισμού των διάφορων παραμέτρων και θα τρέξουμε διάφορα παραδείγματα.

4.1 Περιγραφή λειτουργίας του προσομοιωτή

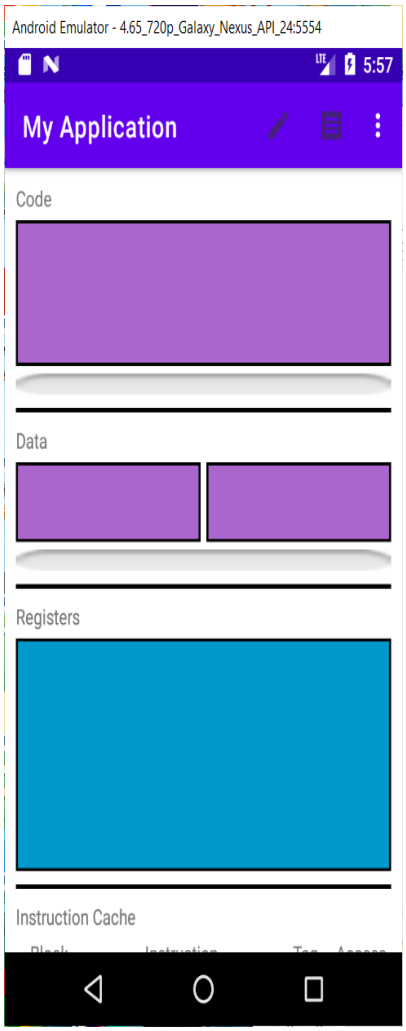
Η εφαρμογή προσομοίωσης της κρυφής μνήμης που έχει δημιουργηθεί, έχει σκοπό να παρουσιάσει τον τρόπο λειτουργίας της κρυφής μνήμης υπογραμίζοντας κάποια στοιχεία της.

Έχοντας γράψει τον κώδικα που θέλουμε να τρέξουμε στην εφαρμογή, θα μπορούμε να δούμε κάθε εντολή τον τρόπο που επηρεάζει τις τιμές των καταχωρητών, τα περιεχόμενα της κρυφής μνήμης και της κύριας μνήμης, καθώς και τον αριθμό των ευστοχιών/αστοχιών κατά τη διάρκεια που τρέχει το πρόγραμμα. Ο προσομοιωτής παρέχει πληροφορίες που χρησιμοποιούνται για την ταυτοποίηση του ζητούμενου μπλοκ αλλά και πληροφορίες στην περίπτωση που δεν υπάρχει ταυτοποίηση.

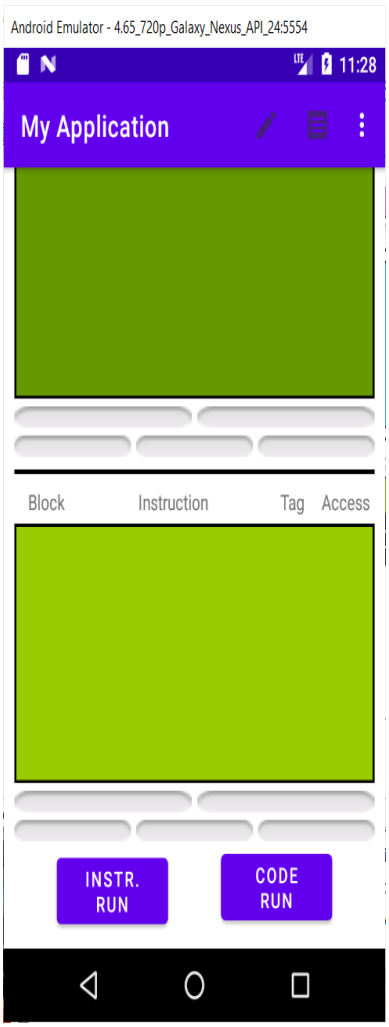
Τα παραπάνω επηρεάζονται από πολλούς παράγοντες, όπως την αρχιτεκτονική της κρυφής μνήμης, το μέγεθος της, το μέγεθος του μπλοκ, την μέθοδο αντικατάστασης του μπλοκ και για αυτό ο προσομοιωτής δίνει την δυνατότητα να αλλάξουμε τις παραμέτρους αυτές ώστε να μελετηθούν τα αποτελέσματα κάθε αλλαγής παραμέτρου.

4.2 Το περιβάλλον του προσομοιωτή

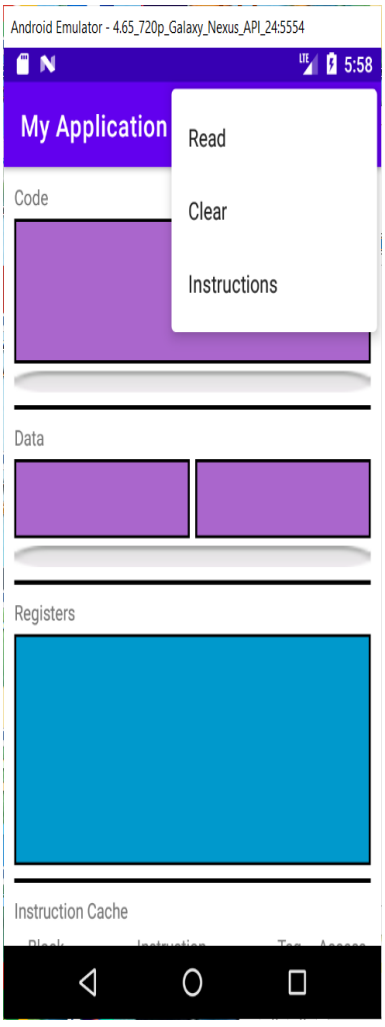
Οι Εικόνες 4.1 και 4.2 παρουσιάζουν την εφαρμογή καθώς ξεκινάει. Η Εικόνα 4.3 δείχνει και τις υπόλοιπες επιλογές του μενού της εφαρμογής που εμφανίζονται όταν πατήσουμε τις τελείες πάνω δεξιά.



Εικόνα 4.1



Εικόνα 4.2



Εικόνα 4.3

Εικόνες 4.1, 4.2, 4.3: Το περιβάλλον της εφαρμογής

Στο πρώτο παράθυρο της Εικόνας 4.1 (Code) θα δούμε τον κώδικα όταν τον φορτώσουμε πατώντας το κουμπί Read του μενού της εφαρμογής. Κάτω από το παράθυρο του κώδικα υπάρχει ένα πεδίο όπου εμφανίζεται ο PC.

Στο αμέσως επόμενο αριστερό παράθυρο (Data) εμφανίζονται τα δεδομένα που βρίσκονται στην μνήμη καθώς και οι διευθύνσεις που είναι αποθηκευμένα. Στην εφαρμογή μας θεωρούμε ότι τα δεδομένα (αριθμοί ακέραιοι ή υποδιαστολής) καταλαμβάνουν **4 bytes** στην μνήμη και για αυτό θα παρατηρούμε στα παραδείγματα ότι τα δεδομένα ανεξαρτήτως τύπου απέχουν μεταξύ τους 4 bytes. Κάτω από το παράθυρο των δεδομένων υπάρχει ένα πεδίο όπου εμφανίζεται η διεύθυνση δεδομένων.

Στο δεξί παράθυρο εμφανίζονται οι σταθερές που περιέχουν διευθύνσεις για καλύτερη παρακολούθηση των εντολών διευθύνσεων.

Στη συνέχεια υπάρχει το παράθυρο των καταχωρητών (32 κατωρητές ακεραίων (R0-R31) και 32 καταχωρητές υποδιαστολής (F0-F31)). Σε αυτό το παράθυρο βλέπουμε τις τιμές που έχει κάθε καταχωρητής. Θα πρέπει να χρησιμοποιήσουμε τους ίδιους καταχωρητές που υπάρχουν στο παράθυρο ώστε να είναι εμφανής η τιμή που περιέχουν.

Στην Εικόνα 4.2 φαίνεται η κρυφή μνήμη εντολών. Κάθε μπλοκ τη κρυφής μνήμης εντολών αριθμείται. Εάν χρησιμοποιήσουμε συσχετική κρυφή μνήμη τότε πέρα από την αρίθμηση του μπλοκ φαίνεται και το σύνολο στο οποίο ανήκει το μπλοκ (πχ set 0, num 2). Σε κάθε μπλοκ της κρυφής μνήμης εντολών εμφανίζεται η ετικέτα, η εντολή και η ευστοχία/αστοχία του μπλοκ που αναζητείται (Block, Instruction, Tag, Access).

Κάτω από το παράθυρο της κρυφής μνήμης εντολών έχουμε πέντε πεδία. Στο πρώτο πεδίο (πάνω αριστερά) εμφανίζεται ο δείκτης για την κρυφή μνήμη εντολών, στο δεύτερο πεδίο (πάνω δεξιά) εμφανίζεται η ετικέτα για την κρυφή μνήμη εντολών, στο τρίτο πεδίο (κάτω αριστερά) εμφανίζονται οι προσπελάσεις της κρυφής μνήμης εντολών, στο τέταρτο πεδίο (κάτω μέση) εμφανίζονται οι ευστοχίες της κρυφής μνήμης εντολών και στο πέμπτο πεδίο (κάτω δεξιά) εμφανίζεται το ποσοστό ευστοχίας της κρυφής μνήμης εντολών. Η παρατήρησή τους βοηθάει να δούμε πως λειτουργεί η κρυφή μνήμη.

Αντίστοιχα στη συνέχεια έχουμε τα ίδια πράγματα για την κρυφή μνήμη δεδομένων. Η διαφορά είναι ότι στην κρυφή μνήμη δεδομένων εμφανίζονται οι τιμές των δεδομένων σε κάθε μπλοκ αντί για την εντολή.

Από κάτω έχουμε δυο κουμπιά. Το αριστερό κουμπί τρέχει τον κώδικα εντολή – εντολή (INSTR RUN), ενώ το δεύτερο κουμπί (CODE RUN) τρέχει όλο τον κώδικα.

Το μενού της εφαρμογής (Εικόνες 4.1, 4.2 και 4.3) περιέχει τις εξής επιλογές. Το πρώτο εικονίδιο περιέχει την επιλογή που μας μεταφέρει στο παράθυρο για να παραμετροποιήσουμε τις κρυφές μνήμες εντολών και δεδομένων. Το δεύτερο εικονίδιο περιέχει την επιλογή που μας μεταφέρει στο παράθυρο που γράφουμε τον κώδικα που θέλουμε να τρέξουμε. Η επιλογή Read χρησιμοποιείται για να φορτώσουμε τον κώδικα που θέλουμε να τρέξουμε στην εφαρμογή. Η επιλογή Clear χρησιμοποιείται για να επαναφέρουμε την εφαρμογή στην αρχική της κατάσταση ώστε να μπορούμε να τρέξουμε την εφαρμογή από την αρχή. Η επιλογή Instructions χρησιμοποιείται για να μεταφερθούμε στο παράθυρο που βλέπουμε τις εντολές που μπορούμε να χρησιμοποιήσουμε για να γράψουμε τον κώδικα.

Πριν αναλύσουμε τις επιλογές αυτές, να πούμε ότι για να τρέξουμε τον κώδικα πρέπει να

κάνουμε τα εξής βήματα. Για αρχή πρέπει να γράψουμε τον κώδικα που θέλουμε να τρέχει η εφαρμογή μας και να παραμετροποιήσουμε τις κρυφές μνήμες εντολών και δεδομένων. Μετά πριν τρέξουμε τον κώδικα πρέπει να φορτώσουμε τον κώδικα στην εφαρμογή. Ο κώδικας μπορεί να τρέξει εντολή-εντολή ή όλος μαζί ταυτόχρονα.

4.3 Ορισμός παραμέτρων

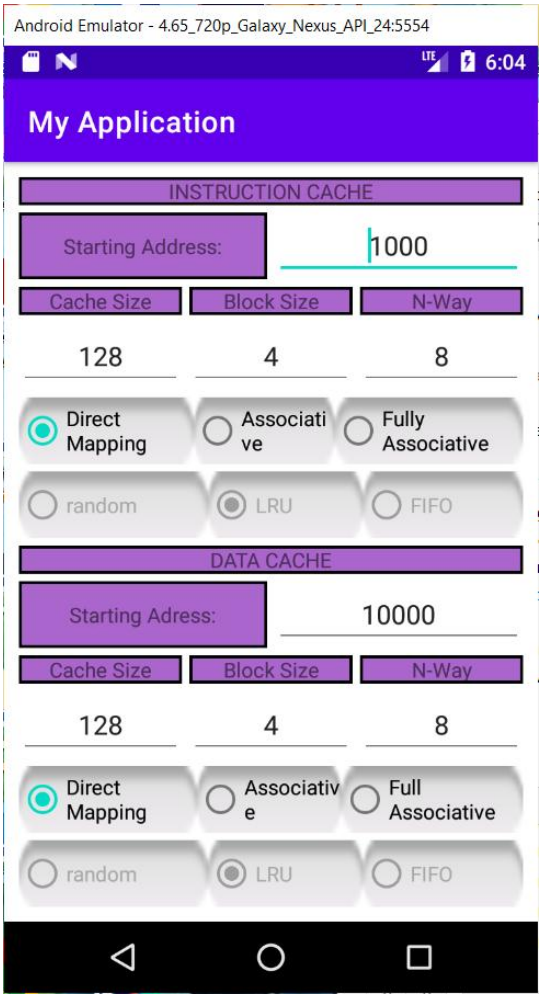
Για να παραμετροποιήσουμε τις κρυφές μνήμες εντολών και δεδομένων πατάμε στο πρώτο εικονίδιο του μενού της εφαρμογής. Ξεκινώντας από πάνω έχουμε τις εξής επιλογές. Πρώτα μπορούμε να επιλέξουμε την αρχική διεύθυνση (σε δεκαδική μορφή) που θα αποθηκευτεί ο κώδικας. Έπειτα ορίζουμε το μέγεθος της κρυφής μνήμης εντολών, το μέγεθος του μπλοκ και τα μπλοκ από τα οποία αποτελείται κάθε σύνολο (εάν δεν έχουμε επιλέξει συσχετική κρυφή μνήμη δεν έχει σημασία). Αυτό που πρέπει να προσέξουμε είναι το μέγεθος της κρυφής μνήμης εντολών πρέπει να διαιρείται ακριβώς με το μέγεθος του μπλοκ. Επίσης τα μπλοκ της κρυφής μνήμης πρέπει να διαιρούνται ακριβώς με τα μπλοκ κάθε συνόλου.

Στη συνέχεια μπορούμε να επιλέξουμε την αρχιτεκτονική της κρυφής μνήμης εντολών (άμεσης απεικόνισης, συσχετική ή πλήρως συσχετική). Μετά μπορούμε να επιλέξουμε τον τρόπο αντικατάστασής του μπλοκ σε περίπτωση που δεν υπάρχει ελεύθερο μπλοκ στην κρυφή μνήμη εντολών (αυτό δεν ισχύει εννοείται για κρυφή μνήμη άμεσης απεικόνισης). Μπορούμε να επιλέξουμε είτε τυχαία αντικατάσταση (random) είτε με βάση ποιο αναζητήθηκε πρώτο (LRU) είτε με βάση ποιο αποθηκεύτηκε πρώτο στην κρυφή μνήμη εντολών (FIFO).

Εάν η κρυφή μνήμη είναι άμεσης απεικόνισης τότε οι επιλογές αντικατάστασης του μπλοκ (random, LRU, FIFO) είναι κλειδωμένες. Επίσης, εάν επιλέξουμε να ρυθμίσουμε των αριθμό των μπλοκ που θα περιέχει κάθε σύνολο τότε αυτομάτως επιλέγεται η συσχετική κρυφή μνήμη.

Αντίστοιχα, τις ίδιες ρυθμίσεις μπορούμε να κάνουμε και για την κρυφή μνήμη δεδομένων.

Στην Εικόνα 4.4 φαίνεται το παράθυρο παραμετροποίησης των κρυφών μνημών .



Εικόνα 4.4: Παραμετροποίηση κρυφής μνήμης

4.4 Συγγραφή κώδικα

Για να γράψουμε τον κώδικα πατάμε το δεύτερο εικονίδιο του μενού της εφαρμογής μας. Αφού γράψουμε τον κώδικα που θέλουμε να τρέξουμε πατάμε save ώστε να αποθηκευτεί ο κώδικας και να μπορούμε να τον τρέξουμε και σε επόμενες φορές.

Ο κώδικας που γράφουμε χωρίζεται σε δυο τμήματα. Στο πρώτο τμήμα που επισημαίνεται με την επισήμανση data: αναθέτουμε τιμές σε μεταβλητές.

Το δεύτερο τμήμα του κώδικα που επισημαίνεται με την επισήμανση main: γράφουμε τον κώδικα. Έστω το επόμενο παράδειγμα:

data:

A=10

B=8

C=0

main:

LOAD R3,0(A)

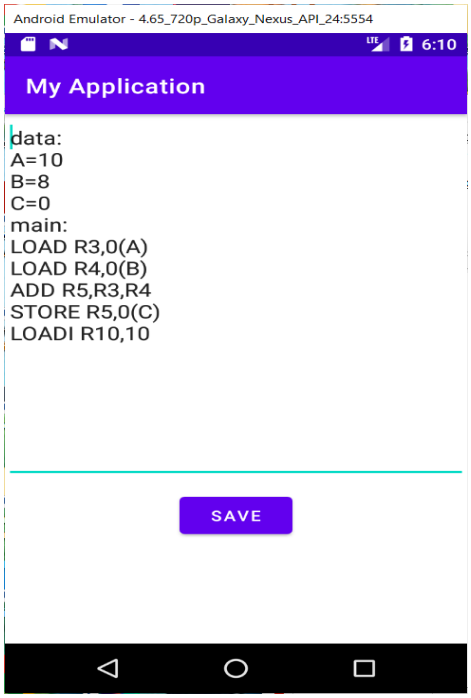
LOAD R4,0(B)

ADD R5,R3,R4

STORE R5,0(C)

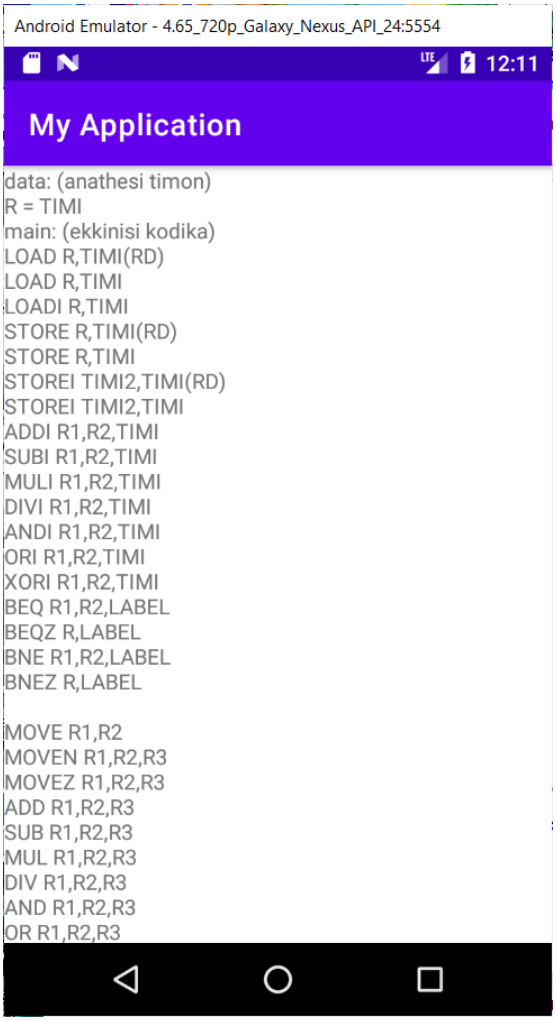
LOADI R10,10

Η Εικόνα 4.5 δείχνει το παράθυρο όπου γράφουμε τον κώδικα καθώς και τον κώδικα που έχουμε γράψει.



Εικόνα 4.5: Το παράθυρο του κώδικα.

Μπορούμε επίσης να δούμε τις εντολές που έχουμε στη διάθεσή μας για τη συγγραφή κώδικα πατώντας το κουμπί Instructions στο μενού της εφαρμογής. Στην Εικόνα 4.6 φαίνεται το παράθυρο με τις εντολές.



Εικόνα 4.6: Παράθυρο με τις διαθέσιμες εντολές

Ακολουθεί η επεξήγηση των διαθέσιμων εντολών.

LOAD R,TIMI(RD)

Αποθηκεύει την τιμή που βρίσκεται στην διεύθυνση TIMI+RD στον καταχωρητή R

LOAD R,TIMI

Αποθηκεύει την τιμή που βρίσκεται στην διεύθυνση TIMI στον καταχωρητή R

LOADI R,TIMI

Αποθηκεύει την τιμή "TIMI" στον καταχωρητή R

STORE R,TIMI(RD)

Αποθηκεύει το περιεχόμενο του καταχωρητή R στην διεύθυνση TIMI+RD. Η εφαρμογή υποστηρίζει μόνο την **ταυτόχρονη εγγραφή**, οπότε όποτε χρησιμοποιείται αυτή η εντολή θα παρατηρείται αλλαγή και στο μπλοκ της κρυφή μνήμης και στο μπλοκ της κύριας μνήμης. Εάν είχαμε ετερόχρονη εγγραφή τότε το δεδομένο θα εγγραφόταν στην κρυφή μνήμη και όταν θα χρειαζόταν να αντικατασταθεί το συγκεκριμένο μπλοκ τότε πριν την αντικατάστασή

του, θα εγγραφόταν το προηγούμενο μπλοκ στην μνήμη. Επίσης κάθε μπλοκ θα είχε ένα bit που θα επισήμαινε εάν το μπλοκ χρειάζεται (bit=1) ή όχι (bit=0) να εγγραφεί στην μνήμη.

STORE R,TIMI

Αποθηκεύει το περιεχόμενο του καταχωρητή R στην διεύθυνση TIMI. Η εφαρμογή υποστηρίζει μόνο την **ταυτόχρονη εγγραφή**, οπότε όποτε χρησιμοποιείται αυτή η εντολή θα παρατηρείται αλλαγή και στο μπλοκ της κρυφή μνήμης και αυτό της κύριας μνήμης.

STOREI TIMH2,TIMI(RD)

Αποθηκεύει την τιμή "TIMH2" στην διεύθυνση TIMI+RD. Η εφαρμογή υποστηρίζει μόνο την **ταυτόχρονη εγγραφή**, οπότε όποτε χρησιμοποιείται αυτή η εντολή θα παρατηρείται αλλαγή και στο μπλοκ της κρυφή μνήμης και αυτό της κύριας μνήμης.

STOREI TIMH2,TIMI

Αποθηκεύει την τιμή "TIMH2" στην διεύθυνση TIMI. Η εφαρμογή υποστηρίζει μόνο την **ταυτόχρονη εγγραφή**, οπότε όποτε χρησιμοποιείται αυτή η εντολή θα παρατηρείται αλλαγή και στο μπλοκ της κρυφή μνήμης και αυτό της κύριας μνήμης.

ADDI R1,R2,TIMI

Αποθηκεύει την τιμή "R2+TIMI" στον καταχωρητή R1. Οποιαδήποτε από τις τιμές R2 ή TIMI μπορεί να είναι ακέραια ή υποδιαστολής. Εάν κάποια τιμή έχει υποδιαστολή τότε και το αποτέλεσμα έχει υποδιαστολή.

SUBI R1,R2,TIMI

Αποθηκεύει την τιμή "R2-TIMI" στον καταχωρητή R1. Οποιαδήποτε από τις τιμές R2 ή TIMI μπορεί να είναι ακέραια ή υποδιαστολής. Εάν κάποια τιμή έχει υποδιαστολή τότε και το αποτέλεσμα έχει υποδιαστολή.

MULI R1,R2,TIMI

Αποθηκεύει την τιμή "R2*TIMI" στον καταχωρητή R1. Οποιαδήποτε από τις τιμές R2 ή TIMI μπορεί να είναι ακέραια ή υποδιαστολής. Εάν κάποια τιμή έχει υποδιαστολή τότε και το αποτέλεσμα έχει υποδιαστολή.

DIVI R1,R2,TIMI

Αποθηκεύει την τιμή "R2/TIMI" στον καταχωρητή R1. Οποιαδήποτε από τις τιμές R2 ή TIMI μπορεί να είναι ακέραια ή υποδιαστολής. Εάν κάποια τιμή έχει υποδιαστολή τότε και το αποτέλεσμα είναι έχει υποδιαστολή.

ANDI R1,R2,TIMI

Εκτελεί την λογική πράξη AND στις τιμές R2 και TIMI και το αποτέλεσμα το αποθηκεύει στον καταχωρητή R1. Οι τιμές που δίνονται και εμφανίζονται είναι σε δεκαδική μορφή.

ORI R1,R2,TIMI

Εκτελεί την λογική πράξη OR στις τιμές R2 και TIMI και το αποτέλεσμα το αποθηκεύει στον καταχωρητή R1. Οι τιμές που δίνονται και εμφανίζονται είναι σε δεκαδική μορφή.

XORI R1,R2,TIMI

Εκτελεί την λογική πράξη XOR στις τιμές R2 και TIMI και το αποτέλεσμα το αποθηκεύει στον καταχωρητή R1. Οι τιμές που δίνονται και εμφανίζονται είναι σε δεκαδική μορφή.

BEQ R1,R2,LABEL

Ελέγχει εάν R1 = R2 και εάν ναι τότε πηγαίνει στην εντολή με ετικέτα LABEL

BEQZ R,LABEL

Ελέγχει εάν R = 0 και εάν ναι τότε πηγαίνει στην εντολή με ετικέτα LABEL

BNE R1,R2,LABEL

Ελέγχει εάν R1 != R2 και εάν ναι τότε πηγαίνει στην εντολή με ετικέτα LABEL

BNEZ R,LABEL

Ελέγχει εάν R != 0 και εάν ναι τότε πηγαίνει στην εντολή με ετικέτα LABEL

MOVE R1,R2

Αντιγράφει την τιμή του καταχωρητή R2, στον καταχωρητή R1

MOVEN R1,R2,R3

Εάν R3<0 τότε αντιγράφει την τιμή του καταχωρητή R2, στον καταχωρητή R1

MOVEZ R1,R2,R3

Εάν R3=0 τότε αντιγράφει την τιμή του καταχωρητή R2, στον καταχωρητή R1

ADD R1,R2,R3

Αποθηκεύει την τιμή "R2+R3" στον καταχωρητή R1. Οποιαδήποτε από τις τιμές R2 ή R3 μπορεί να είναι ακέραια ή υποδιαστολής. Εάν κάποια τιμή έχει υποδιαστολή τότε και το αποτέλεσμα έχει υποδιαστολή.

SUB R1,R2,R3

Αποθηκεύει την τιμή "R2-R3" στον καταχωρητή R1. Οποιαδήποτε από τις τιμές R2 ή R3 μπορεί να είναι ακέραια ή υποδιαστολής. Εάν κάποια τιμή έχει υποδιαστολή τότε και το αποτέλεσμα έχει υποδιαστολή.

MUL R1,R2,R3

Αποθηκεύει την τιμή "R2*R3" στον καταχωρητή R1. Οποιαδήποτε από τις τιμές R2 ή R3 μπορεί να είναι ακέραια ή υποδιαστολής. Εάν κάποια τιμή έχει υποδιαστολή τότε και το αποτέλεσμα έχει υποδιαστολή.

DIV R1,R2,R3

Αποθηκεύει την τιμή "R2/R3" στον καταχωρητή R1. Οποιαδήποτε από τις τιμές R2 ή R3 μπορεί να είναι ακέραια ή υποδιαστολής. Εάν κάποια τιμή έχει υποδιαστολή τότε και το

αποτέλεσμα έχει υποδιαστολή.

AND R1,R2,R3

Εκτελεί την λογική πράξη AND στις τιμές R2 και R3 και το αποτέλεσμα το αποθηκεύει στον καταχωρητή R1. Οι τιμές που δίνονται και εμφανίζονται είναι σε δεκαδική μορφή.

OR R1,R2,R3

Εκτελεί την λογική πράξη OR στις τιμές R2 και R3 και το αποτέλεσμα το αποθηκεύει στον καταχωρητή R1. Οι τιμές που δίνονται και εμφανίζονται είναι σε δεκαδική μορφή.

XOR R1,R2,R3

Εκτελεί την λογική πράξη XOR στις τιμές R2 και R3 και το αποτέλεσμα το αποθηκεύει στον καταχωρητή R1. Οι τιμές που δίνονται και εμφανίζονται είναι σε δεκαδική μορφή.

SLIDEL R1,R2,TIMI

Εκτελεί αριστερή ολίσθηση της τιμής του καταχωρητή R2 κατά τόσα bits όσα ορίζει η TIMI και αποθηκεύει το αποτέλεσμα στον καταχωρητή R1. Οι τιμές που δίνονται και εμφανίζονται είναι σε δεκαδική μορφή. Η ολίσθηση είναι κυκλική προς τα αριστερά μη κυκλική προς τα δεξιά και γίνεται σε έναν προσημασμένο 32-bit καταχωρητή, όπου το πρόσημο αντιπροσωπεύεται από το υψηλότερο σημαντικό bit του καταχωρητή.

SLIDER R1,R2,TIMI

Εκτελεί δεξιά ολίσθηση της τιμής του καταχωρητή R2 κατά τόσα bits όσα ορίζει η TIMI και αποθηκεύει το αποτέλεσμα στον καταχωρητή R1. Οι τιμές που δίνονται και εμφανίζονται είναι σε δεκαδική μορφή. Η ολίσθηση είναι κυκλική προς τα αριστερά μη κυκλική προς τα δεξιά και γίνεται σε έναν προσημασμένο 32-bit καταχωρητή, όπου το πρόσημο αντιπροσωπεύεται από το υψηλότερο σημαντικό bit του καταχωρητή.

SMALLER R1,R2,R3

Συγκρίνει τα περιεχόμενα των καταχωρητών R2 και R3 και εάν R2 < R3 τότε R1 = 1 ,αλλιώς R1 = 0.

BIGGER R1,R2,R3

Συγκρίνει τα περιεχόμενα των καταχωρητών R2 και R3 και εάν R2 > R3 τότε R1 = 1 ,αλλιώς R1 = 0.

J LABEL

Κάνει άλμα στην εντολή με ετικέτα LABEL

JAL LABEL

Αποθηκεύει την διεύθυνση επιστροφής στον καταχωρητή R31 και κάνει άλμα στην εντολή με ετικέτα LABEL

ENDF

Διακλαδώνει στην διεύθυνση που περιέχει ο καταχωρητής R31.

END

Επισημαίνει το τέλος του κύριου κώδικα. Δεν απαιτείται στην περίπτωση που έχουμε γράψει μόνο τον κύριο κώδικα.

4.5 Εκτέλεση κώδικα

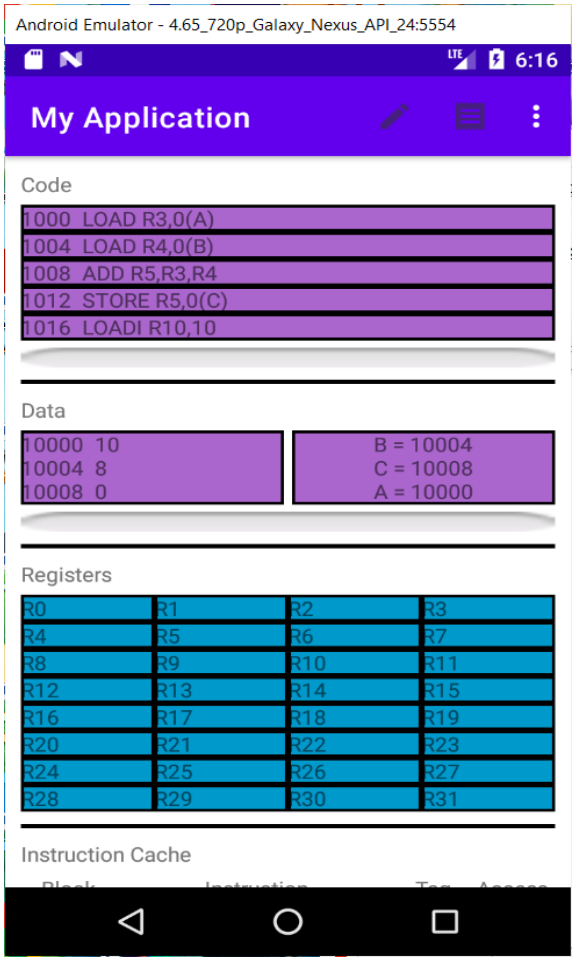
Πριν τρέξουμε τον κώδικα πρέπει να τον φορτώσουμε στην εφαρμογή. Αυτό το πετυχαίνουμε πατώντας το κουμπί Read από το μενού της εφαρμογής .

Στην Εικόνα 4.7 φαίνεται η εφαρμογή όταν έχει φορτώσει τον κώδικα. Παρατηρούμε ότι στο παράθυρο που φαίνεται ο κώδικας (το πρώτο από πάνω παράθυρο) εμφανίζεται ο κώδικας που θα τρέξει, καθώς και σε ποια θέση μνήμης είναι αποθηκευμένη κάθε εντολή.

Επίσης, παρατηρούμε στο παράθυρο που φαίνονται τα δεδομένα (το δεύτερο παράθυρο), να εμφανίζονται τα δεδομένα που χρησιμοποιεί ο κώδικας στην αρχή όταν φορτώνεται καθώς και σε ποια θέση μνήμης είναι αποθηκευμένα. Αυτά τα δεδομένα είναι τα δεδομένα που έχουμε αναθέσει όταν γράφαμε τον κώδικα.

Στο παράθυρο των καταχωρητών βλέπουμε τους καταχωρητές που τους έχει ανατεθεί μια τιμή από την αρχή του κώδικα, με τη διαφορά ότι η ανάθεση τιμών σε αυτούς γίνεται μέσω της διεύθυνσης που βρίσκονται τα δεδομένα. Το ποια δεδομένα περιέχει κάθε διεύθυνση φαίνεται στο παράθυρο δεδομένων. Αυτό το κάνουμε για να μπορούμε να χρησιμοποιήσουμε τους καταχωρητές αυτούς σε εντολές διευθυνσιοδότησης (διευθυνσιοδότηση μετατόπισης (LOAD R,(num(R)).

Τέλος παρατηρούμε ότι έχουν αριθμηθεί και τα μπλοκ των κρυφών μνημών.



Εικόνα 4.7: Παράθυρο με τον κώδικα, τα δεδομένα και τους καταχωρητές

ΚΕΦΑΛΑΙΟ 5

ΠΑΡΑΔΕΙΓΜΑΤΑ ΧΡΗΣΗΣ ΤΟΥ ΠΡΟΣΟΜΟΙΩΤΗ

5.1 Παράδειγμα cache άμεσης απεικόνισης

Σε αυτήν την ενότητα θα τρέξουμε τον κώδικα που αναφέραμε στο κεφάλαιο 4, εντολή-εντολή, σε κρυφή μνήμη άμεσης απεικόνισης και θα δείξουμε τα αποτελέσματα κάθε βήματος σε εικόνα.

Πιο συγκεκριμένα θα παρατηρήσουμε πως ο δείκτης και η ετικέτα (που υπολογίζονται από την φυσική διεύθυνση) βοηθάνε στην ταυτοποίηση του μπλοκ, ώστε να ανακτηθεί η πληροφορία του ζητούμενου μπλοκ. Σε περίπτωση που δεν βρεθεί η ζητούμενη πληροφορία στην κρυφή μνήμη άμεσης απεικόνισης, τότε η πληροφορία που αναζητάει ο επεξεργαστής θα ανακτηθεί από την κύρια μνήμη και θα αποθηκευτεί στο μπλοκ που δείχνει ο δείκτης μαζί με την ετικέτα. Στη συνέχεια, θα παρατηρήσουμε τον τρόπο που επηρεάζει τα περιεχόμενα των καταχωρητών ή της μνήμης η εκτέλεση της εντολής, περιεχόμενα τα οποία μπορεί να χρησιμοποιηθούν αργότερα.

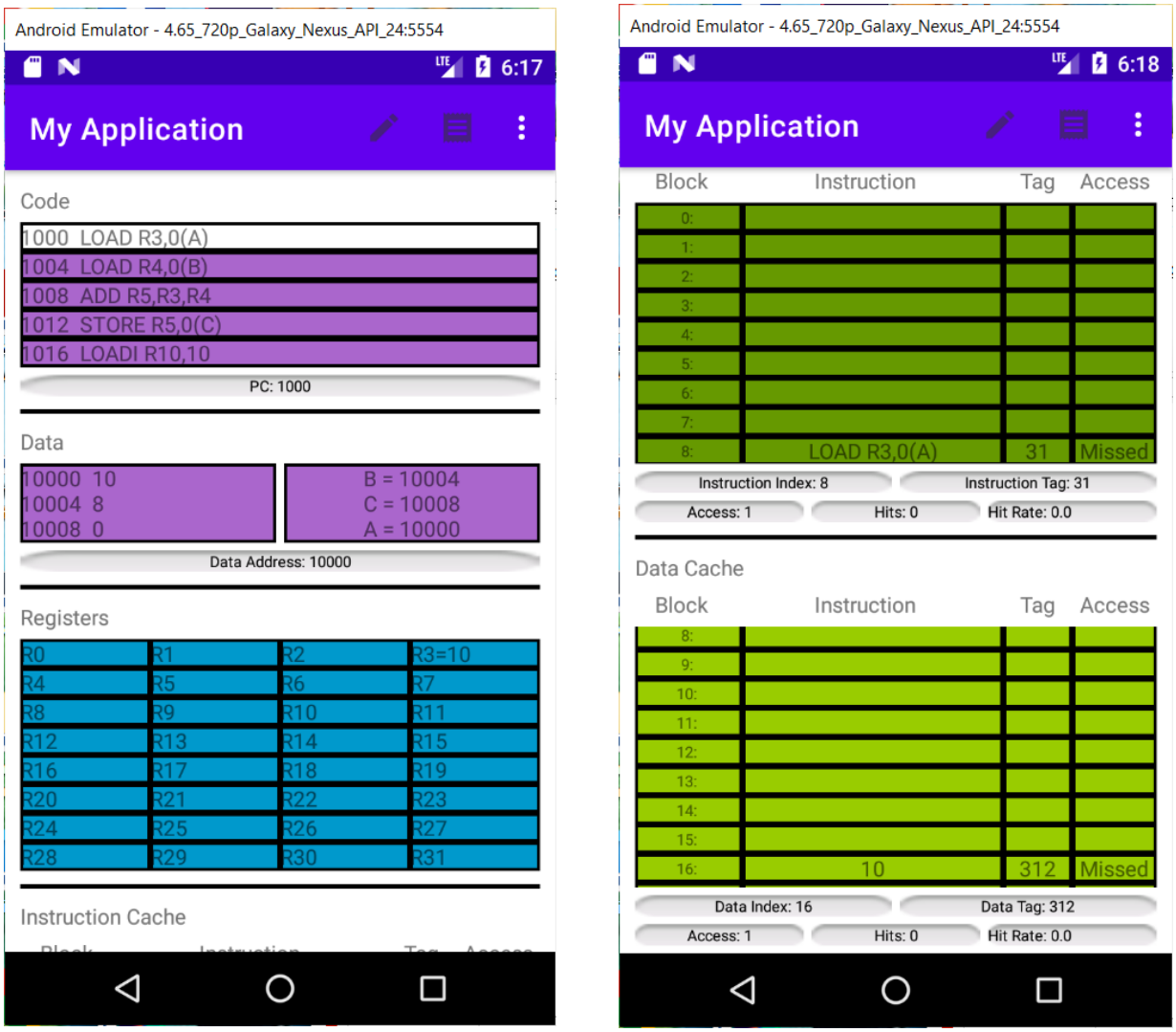
Τέλος μπορούμε να παρατηρήσουμε το ποσοστό ευστοχίας που παρουσιάζει η κρυφή μνήμη στην εκτέλεση αυτού του κώδικα, ποσοστό ευστοχίας το οποίο εξαρτάται από την παραμετροποίηση της κρυφής μνήμης,

Στην Εικόνα 5.1 φαίνεται η εκτέλεση της εντολής LOAD R3,0(A). Παρατηρούμε τον PC να δείχνει στην επόμενη εντολή (1000).

Παρατηρούμε επίσης ότι στον καταχωρητή R3 έχει αποθηκευτεί η τιμή (10) που υπάρχει στην διεύθυνση που υποδεικνύει η μεταβλητή A (10000).

Η εντολή αποθηκεύτηκε στην κρυφή μνήμη άμεσης απεικόνισης εντολών στη θέση που υποδεικνύει ο δείκτης εντολών (8). Στο μπλοκ της κρυφής μνήμης αποθηκεύτηκε η ετικέτα εντολών (31), η οποία χρησιμοποιείται για την ταυτοποίηση του μπλοκ και η ίδια η εντολή (LOAD R3,0(A)) ώστε να ανακτηθεί όταν αυτή ζητηθεί. Επίσης, βλέπουμε ότι εμφανίστηκε αστοχία στην αναζήτηση του μπλοκ αυτού.

Κάτω από το παράθυρο της κρυφή μνήμης εντολών βλέπουμε τις προσπελάσεις της κρυφής μνήμης εντολών (Access Cache = 1), τις ευστοχίες (Hits Cache = 0) και το ποσοστό



ευστοχίας (Hit Rate = 0).

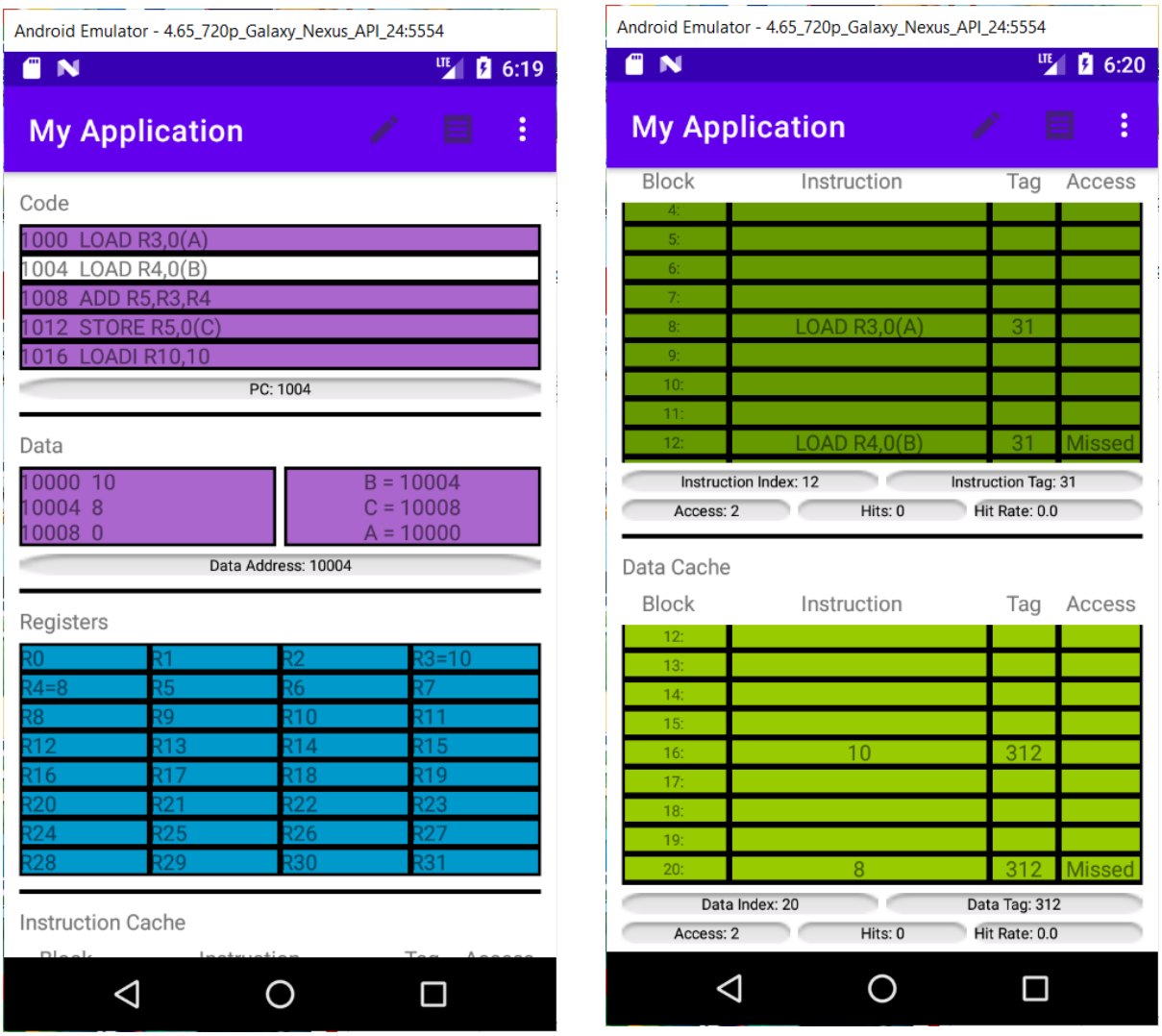
Εικόνα 5.1: Εκτέλεση της εντολής LOAD R3,0(A)

Αντίστοιχα το δεδομένο αποθηκεύτηκε στην κρυφή μνήμη άμεσης απεικόνισης δεδομένων στη θέση που υποδεικνύει ο δείκτης δεδομένων (16). Στο μπλοκ της κρυφής μνήμης αποθηκεύτηκε η ετικέτα δεδομένων (312), η οποία χρησιμοποιείται για την ταυτοποίηση του μπλοκ και το δεδομένο της διεύθυνσης μνήμης 10000 (10) ώστε να αναζητηθεί όταν αυτό ζητηθεί. Επίσης βλέπουμε ότι εμφανίστηκε αστοχία στην αναζήτηση του μπλοκ αυτού.

Κάτω από το παράθυρο της κρυφή μνήμης δεδομένων βλέπουμε τις προσπελάσεις της κρυφής μνήμης δεδομένων (Access Cache = 1), τις ευστοχίες (Hits Cache = 0) και το ποσοστό ευστοχίας (Hit Rate=0).

Στην Εικόνα 5.2 φαίνεται η εκτέλεση της εντολής LOAD R4,0(B). Παρατηρούμε τον PC να

δείχνει στην επόμενη εντολή (1004).



Εικόνα 5.2: Εκτέλεση της εντολής LOAD R4,0(B)

Παρατηρούμε ότι στον καταχωρητή R4 έχει αποθηκευτεί η τιμή (8) που υπάρχει στην διεύθυνση που υποδεικνύει η μεταβλητή B (10004).

Η εντολή αποθηκεύτηκε στην κρυφή μνήμη άμεσης απεικόνισης εντολών στη θέση που υποδεικνύει ο δείκτης εντολών (12). Στο μπλοκ της κρυφής μνήμης αποθηκεύτηκε η ετικέτα εντολών (31), η οποία χρησιμοποιείται για την ταυτοποίηση του μπλοκ και η ίδια η εντολή (LOAD R4,0(B)) ώστε να ανακτηθεί όταν αυτή ζητηθεί. Επίσης, βλέπουμε ότι εμφανίστηκε αστοχία στην αναζήτηση του μπλοκ αυτού.

Κάτω από το παράθυρο της κρυφή μνήμης εντολών βλέπουμε τις προσπελάσεις της κρυφής μνήμης εντολών (Access Cache = 2), τις ευστοχίες (Hits Cache = 0) και το ποσοστό ευστοχίας (Hit Rate = 0).

Αντίστοιχα το δεδομένο αποθηκεύτηκε στην κρυφή μνήμη άμεσης απεικόνισης δεδομένων στη θέση που υποδεικνύει ο δείκτης δεδομένων (20). Στο μπλοκ της κρυφής μνήμης αποθηκεύτηκε η ετικέτα δεδομένων (312), η οποία χρησιμοποιείται για την ταυτοποίηση του μπλοκ και το δεδομένο της διεύθυνσης μνήμης 10004 (8) ώστε να ανακτηθεί όταν αυτό ζητηθεί. Επίσης, βλέπουμε ότι εμφανίστηκε αστοχία στην αναζήτηση του μπλοκ αυτού.

Κάτω από το παράθυρο της κρυφή μνήμης δεδομένων βλέπουμε τις προσπελάσεις της κρυφής μνήμης δεδομένων (Access Cache = 2), τις ευστοχίες (Hits Cache = 0) και το ποσοστό ευστοχίας (Hit Rate=0).

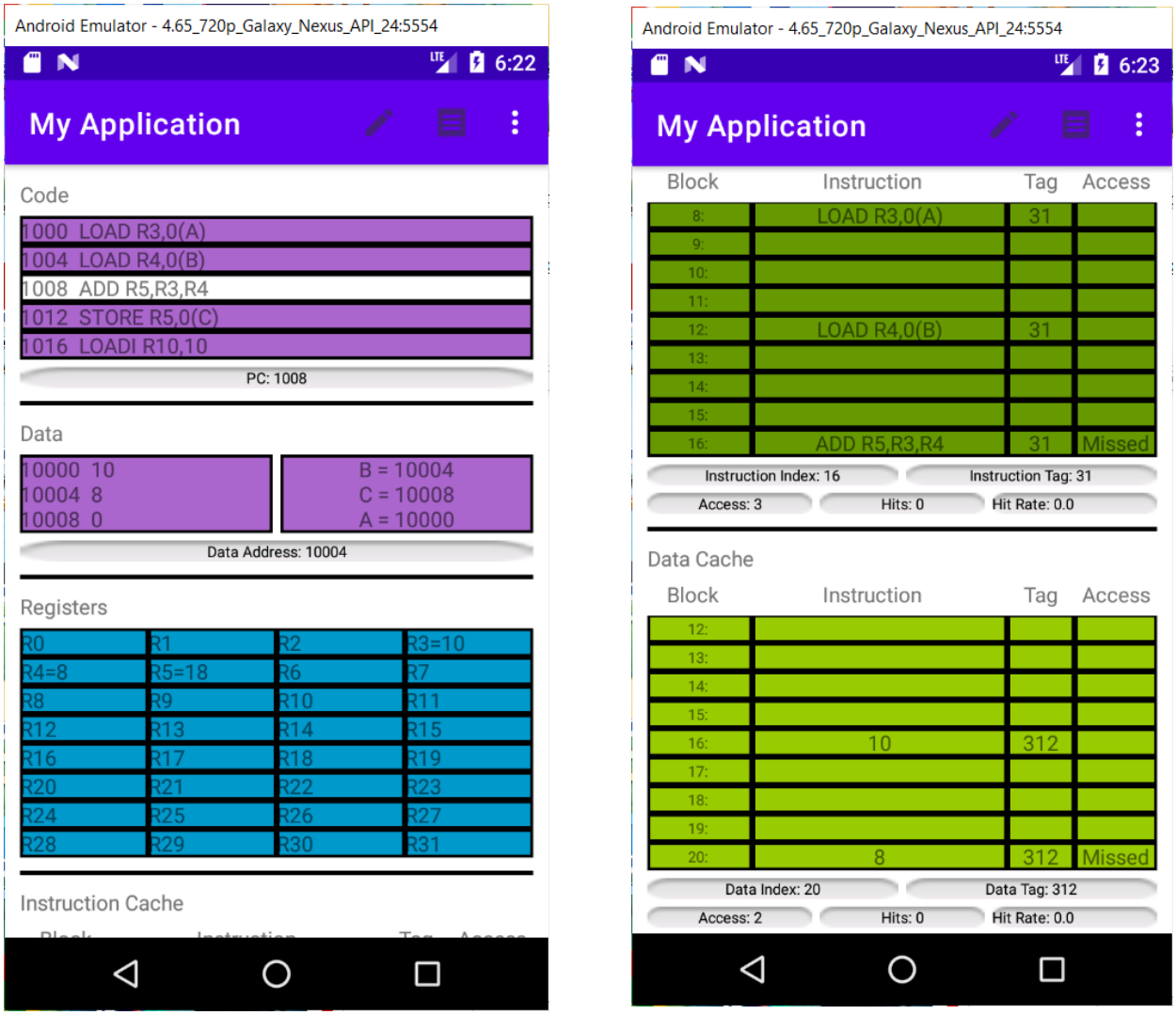
Στην Εικόνα 5.3 φαίνεται η εκτέλεση της εντολής ADD R5,R3,R4. Παρατηρούμε τον PC να δείχνει στην επόμενη εντολή (1008).

Παρατηρούμε επίσης ότι στον καταχωρητή R5 έχει αποθηκευτεί η τιμή (18) που προκύπτει από την πρόσθεση των τιμών που περιέχουν οι καταχωρητές R3 και R4.

Η εντολή αποθηκεύτηκε στην κρυφή μνήμη άμεσης απεικόνισης εντολών στη θέση που υποδεικνύει ο δείκτης εντολών (16). Στο μπλοκ της κρυφής μνήμης αποθηκεύτηκε η ετικέτα εντολών (31), η οποία χρησιμοποιείται για την ταυτοποίηση του μπλοκ και η ίδια η εντολή (ADD R5,R3,R4) ώστε να ανακτηθεί όταν αυτή ζητηθεί. Επίσης βλέπουμε ότι εμφανίστηκε αστοχία στην αναζήτηση του μπλοκ αυτού.

Κάτω από το παράθυρο της κρυφή μνήμης εντολών βλέπουμε τις προσπελάσεις της κρυφής μνήμης εντολών (Access Cache = 3), τις ευστοχίες (Hits Cache = 0) και το ποσοστό ευστοχίας (Hit Rate = 0).

Σε αυτήν την εντολή δεν έχουμε αναζήτηση από την κύρια μνήμη, οπότε δεν έχουμε κάποια αλλαγή στην κρυφή μνήμη δεδομένων, στον δείκτη και την ετικέτα δεδομένων, στις προσπελάσεις, στις ευστοχίες και στο ποσοστό ευστοχίας της κρυφής μνήμης δεδομένων.

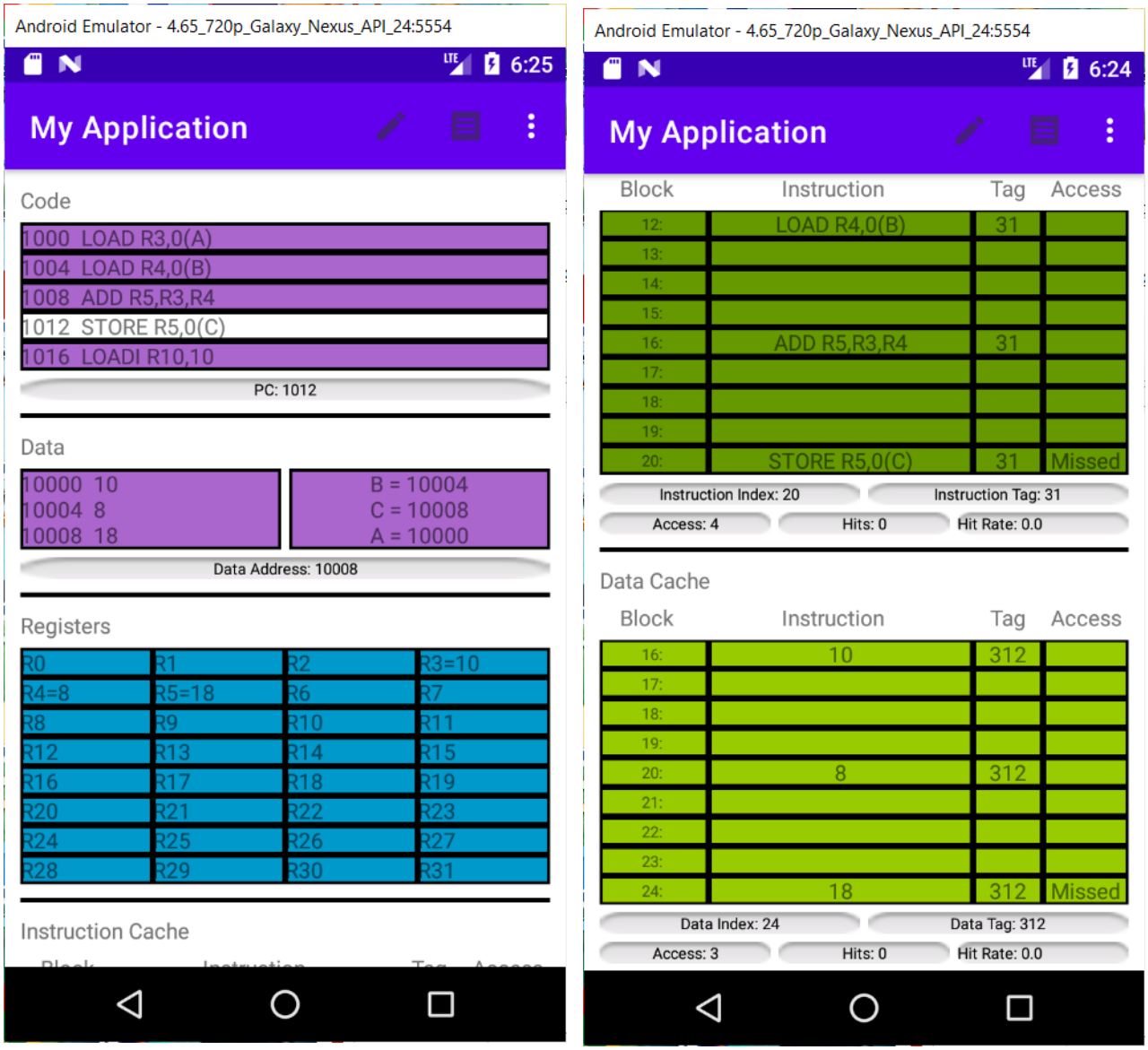


Εικόνα 5.3: Εκτέλεση της εντολής ADD R5,R3,R4

Στην Εικόνα 5.4 φαίνεται η εκτέλεση της εντολής. Παρατηρούμε τον PC να δείχνει στην επόμενη εντολή (1012).

Το δεδομένο μας (18) που υποδεικνύει ο καταχωρητής R5 αποθηκεύτηκε στην διεύθυνση που υποδεικνύει η μεταβλητή C (10008).

Η εντολή αποθηκεύτηκε στην κρυφή μνήμη άμεσης απεικόνισης εντολών στη θέση που υποδεικνύει ο δείκτης εντολών (20). Στο μπλοκ της κρυφής μνήμης αποθηκεύτηκε η ετικέτα εντολών (31), η οποία χρησιμοποιείται για την ταυτοποίηση του μπλοκ και η ίδια η εντολή (STORE R5,0(C)) ώστε να ανακτηθεί όταν αυτό ζητηθεί. Επίσης, βλέπουμε ότι εμφανίστηκε αστοχία στην αναζήτηση του μπλοκ αυτού.



Εικόνα 5.4: Εκτέλεση της εντολής STORE R5,0(C)

Κάτω από το παράθυρο της κρυφή μνήμης εντολών βλέπουμε τις προσπελάσεις της κρυφής μνήμης εντολών (Access Cache = 4), τις ευστοχίες (Hits Cache = 0) και το ποσοστό ευστοχίας (Hit Rate = 0).

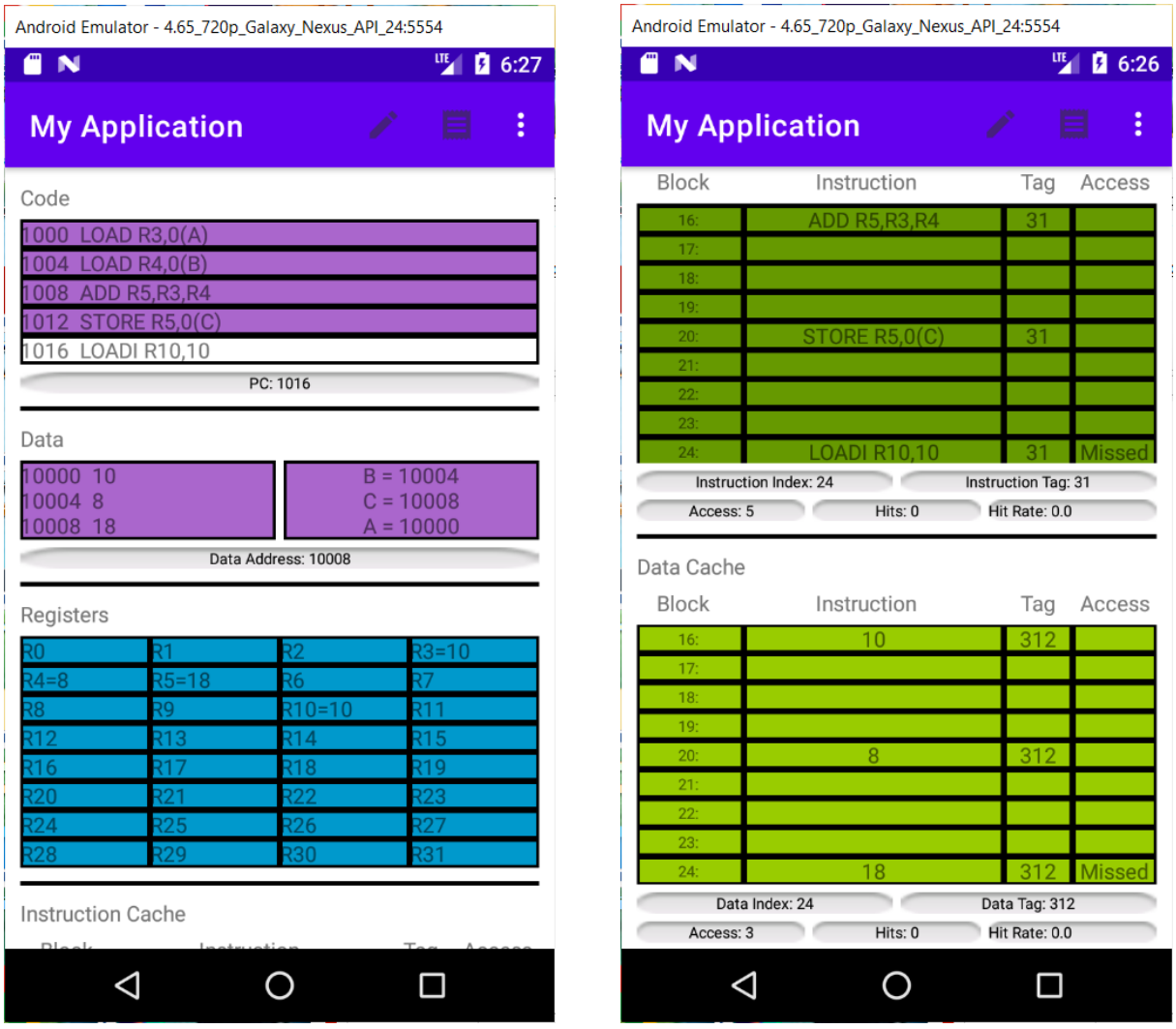
Αντίστοιχα το δεδομένο αποθηκεύτηκε στην κρυφή μνήμη άμεσης απεικόνισης δεδομένων στη θέση που υποδεικνύει ο δείκτης δεδομένων (24). Στο μπλοκ της κρυφής μνήμης αποθηκεύτηκε η ετικέτα δεδομένων (312), η οποία χρησιμοποιείται για την ταυτοποίηση του μπλοκ και το δεδομένο (18) ώστε να ανακτηθεί όταν αυτό ζητηθεί. Επίσης, βλέπουμε ότι εμφανίστηκε αστοχία στην αναζήτηση του μπλοκ αυτού.

Κάτω από το παράθυρο της κρυφή μνήμης δεδομένων βλέπουμε τις προσπελάσεις της κρυφής μνήμης δεδομένων (Access Cache = 3), τις ευστοχίες (Hits Cache = 0) και το

ποσοστό ευστοχίας (Hit Rate=0).

Παρατηρούμε ότι επειδή η εφαρμογή μας υποστηρίζει την ταυτόχρονη εγγραφή, έχουμε εγγραφή του δεδομένου και στην κρυφή μνήμη δεδομένων και στην κύρια μνήμη.

Στην Εικόνα 5.5 φαίνεται η εκτέλεση της εντολής LOADI R10,10. Παρατηρούμε τον PC να δείχνει στην επόμενη εντολή (1016).



Εικόνα 5.5: Εκτέλεση της εντολής LOADI R10,10

Παρατηρούμε ότι στον καταχωρητή R10 έχει αποθηκεύτηκε η τιμή 10.

Η εντολή αποθηκεύτηκε στην κρυφή μνήμη άμεσης απεικόνισης εντολών στη θέση που υποδεικνύει ο δείκτης εντολών (24). Στο μπλοκ της κρυφής μνήμης αποθηκεύτηκε η ετικέτα εντολών (31), η οποία χρησιμοποιείται για την ταυτοποίηση του μπλοκ και η ίδια η εντολή

(LOADI R10,10) ώστε να ανακτηθεί όταν αυτή ζητηθεί. Επίσης, βλέπουμε ότι εμφανίστηκε αστοχία στην αναζήτηση του μπλοκ αυτού.

Κάτω από το παράθυρο της κρυφή μνήμης εντολών βλέπουμε τις προσπελάσεις της κρυφής μνήμης εντολών (Access Cache = 5), τις ευστοχίες (Hits Cache = 0) και το ποσοστό ευστοχίας (Hit Rate = 0).

Σε αυτήν την εντολή δεν έχουμε αναζήτηση από την μνήμη, οπότε δεν έχουμε κάποια αλλαγή στην κρυφή μνήμη δεδομένων, στον δείκτη και την ετικέτα δεδομένων, στις προσπελάσεις, στις ευστοχίες και στο ποσοστό ευστοχίας της κρυφής μνήμης δεδομένων.

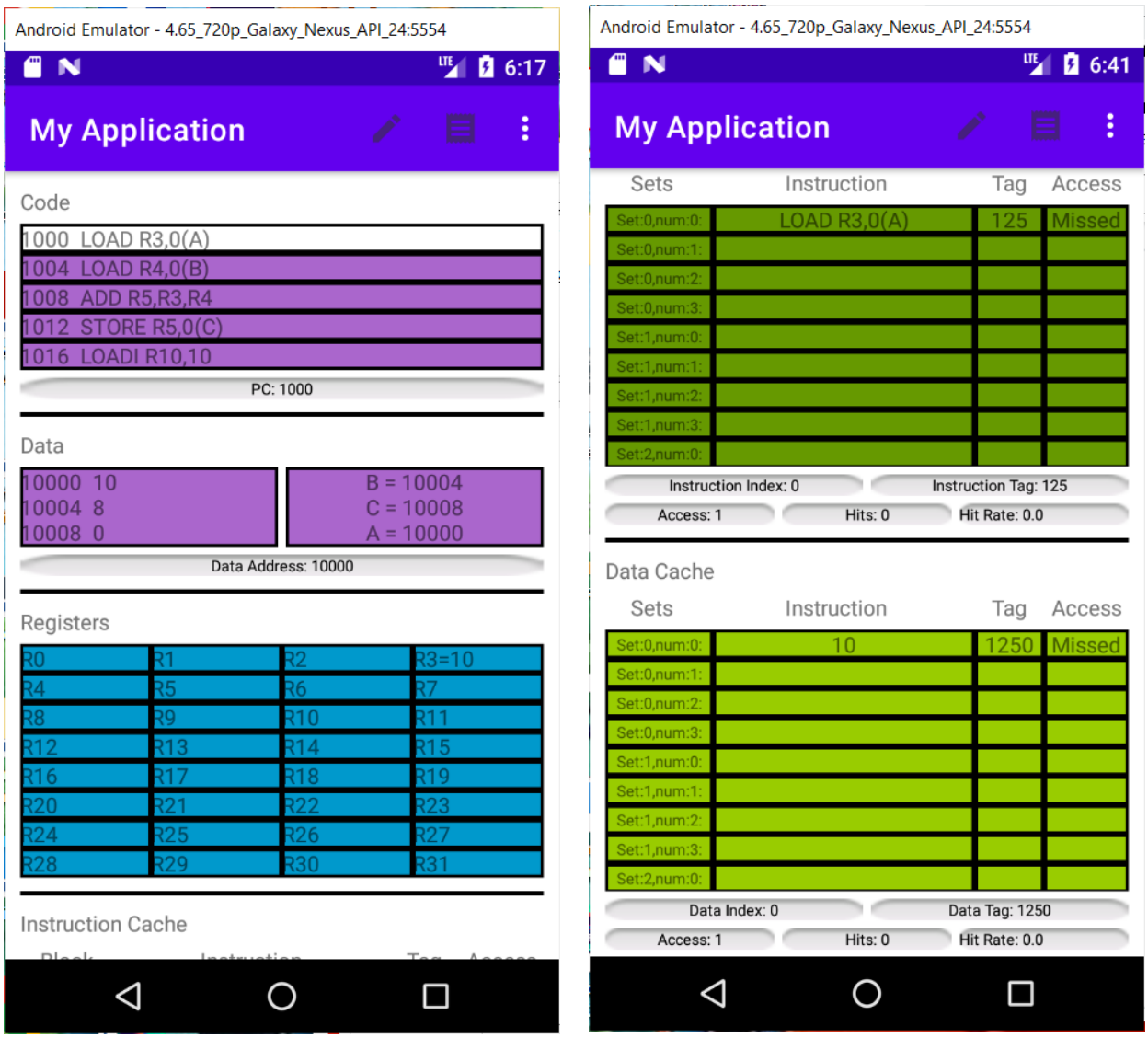
5.2 Παράδειγμα cache συσχετιστικής συνόλου

Σε αυτήν την ενότητα θα τρέξουμε τον κώδικα που αναφέραμε στο κεφάλαιο 4 εντολή-εντολή σε συσχετιστική κρυφή μνήμη συνόλου 4 δρόμων και θα δείχνουμε τα αποτελέσματα κάθε βήματος σε εικόνα.

Πιο συγκεκριμένα θα παρατηρήσουμε τον τρόπο που ο δείκτης και η ετικέτα (που υπολογίζονται από την διεύθυνση) βοηθάνε στην ταυτοποίηση του μπλοκ, ώστε να ανακτηθεί η πληροφορία του ζητούμενου μπλοκ. Σε περίπτωση που δεν βρεθεί η ζητούμενη πληροφορία στην συσχετιστική κρυφή μνήμη, τότε η πληροφορία που αναζητάει ο επεξεργαστής θα ανακτηθεί από την κύρια μνήμη και θα αποθηκευτεί στο σύνολο που δείχνει ο δείκτης και στο επόμενο ελεύθερο μπλοκ που υπάρχει στο σύνολο, μαζί με την ετικέτα. Στη συνέχεια, θα παρατηρήσουμε τον τρόπο που επηρεάζει η εκτέλεση της εντολής τα περιεχόμενα των καταχωρητών ή της μνήμης, περιεχόμενα τα οποία μπορεί να χρησιμοποιηθούν αργότερα.

Τέλος μπορούμε να παρατηρήσουμε το ποσοστό ευστοχίας που παρουσιάζει η κρυφή μνήμη κατά την εκτέλεση του κώδικα αυτού, ποσοστό ευστοχίας το οποίο εξαρτάται από την παραμετροποίηση της κρυφής μνήμης,

Στην Εικόνα 5.6 φαίνεται η εκτέλεση της εντολής LOAD R3,0(A). Παρατηρούμε τον PC να δείχνει στην επόμενη εντολή (1000).



Εικόνα 5.6: Εκτέλεση της εντολής LOAD R3,0(A)

Παρατηρούμε ότι στον καταχωρητή R3 έχει αποθηκευτεί η τιμή (10) που υπάρχει στην διεύθυνση που υποδεικνύει η μεταβλητή A (10000).

Η εντολή αποθηκεύτηκε στο επόμενο ελεύθερο μπλοκ που βρέθηκε, στο σύνολο της συσχετιστικής κρυφής μνήμης εντολών που υποδεικνύει ο δείκτης εντολών (0). Στο μπλοκ της κρυφής μνήμης αποθηκεύτηκε η ετικέτα εντολών (125), η οποία χρησιμοποιείται για την ταυτοποίηση του μπλοκ και η ίδια η εντολή (LOAD R3,0(A)) ώστε να ανακτηθεί όταν αυτή ζητηθεί. Επίσης βλέπουμε ότι εμφανίστηκε αστοχία στην αναζήτηση αυτού του μπλοκ.

Κάτω από το παράθυρο της κρυφή μνήμης εντολών βλέπουμε τις προσπελάσεις της κρυφής μνήμης εντολών (Access Cache = 1), τις ευστοχίες (Hits Cache = 0) και το ποσοστό ευστοχίας (Hit Rate = 0).

Αντίστοιχα το δεδομένο αποθηκεύτηκε στο επόμενο ελεύθερο μπλοκ που βρέθηκε, στο σύνολο της συσχετιστικής κρυφής μνήμης δεδομένων που υποδεικνύει ο δείκτης δεδομένων (0). Στο μπλοκ της κρυφής μνήμης αποθηκεύτηκε η ετικέτα δεδομένων (1250), η οποία χρησιμοποιείται για την ταυτοποίηση του μπλοκ και το δεδομένο της διεύθυνσης μνήμης 10000 (10) ώστε να ανακτηθεί όταν αυτό ζητηθεί. Επίσης, βλέπουμε ότι εμφανίστηκε αστοχία στην αναζήτηση αυτού του μπλοκ.

Κάτω από το παράθυρο της κρυφή μνήμης δεδομένων βλέπουμε τις προσπελάσεις της κρυφής μνήμης δεδομένων (Access Cache = 1), τις ευστοχίες (Hits Cache = 0) και το ποσοστό ευστοχίας (Hit Rate=0).

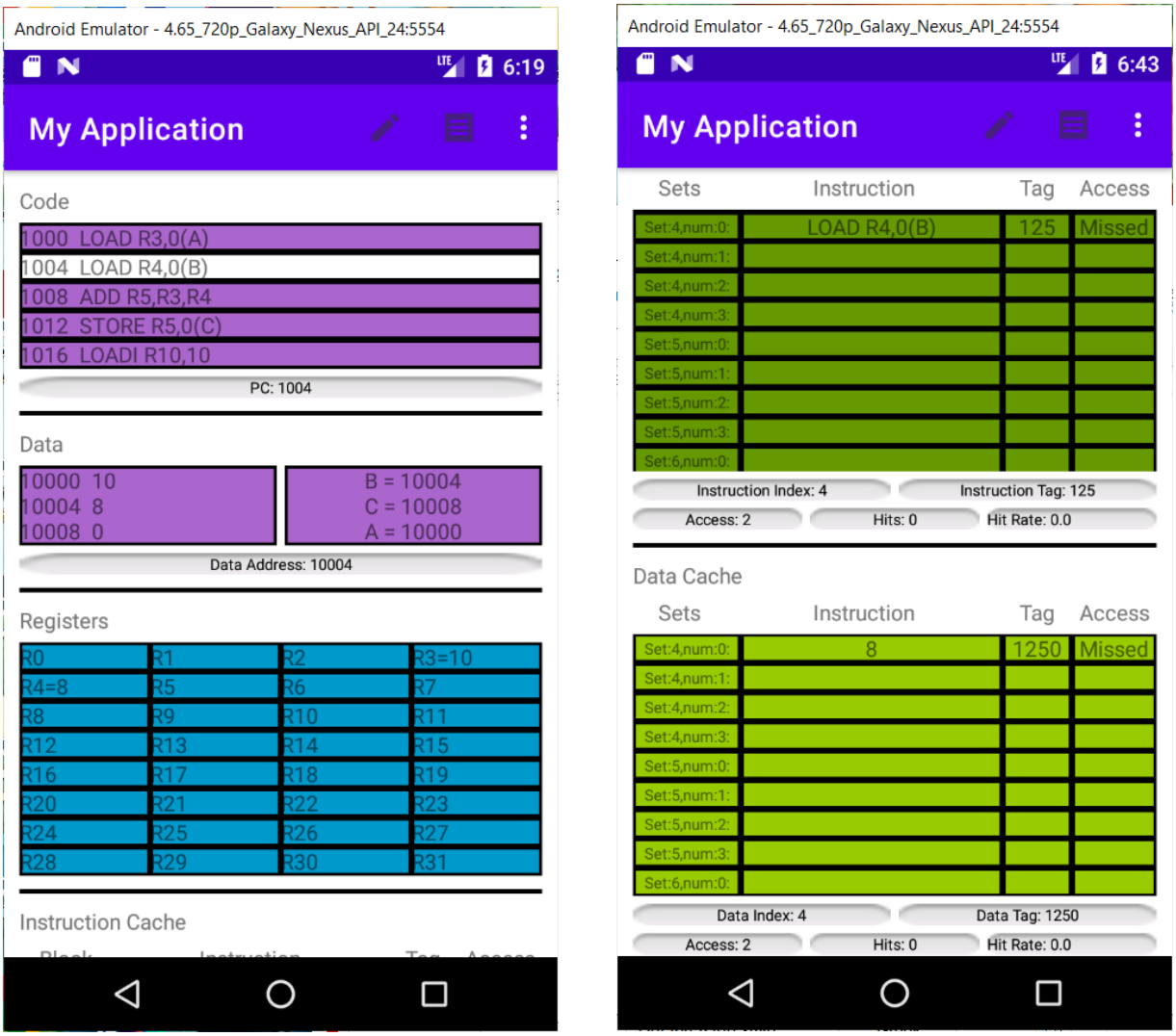
Στην Εικόνα 5.7 φαίνεται η εκτέλεση της εντολής LOAD R4,0(B). Παρατηρούμε τον PC να δείχνει στην επόμενη εντολή (1004).

Παρατηρούμε επίσης ότι στον καταχωρητή R4 έχει αποθηκευτεί η τιμή (8) που υπάρχει στην διεύθυνση που υποδεικνύει η μεταβλητή B (10004).

Η εντολή αποθηκεύτηκε στο επόμενο ελεύθερο μπλοκ που βρέθηκε, στο σύνολο της συσχετιστικής κρυφής μνήμης εντολών που υποδεικνύει ο δείκτης εντολών (4). Στο μπλοκ της κρυφής μνήμης αποθηκεύτηκε η ετικέτα εντολών (125), η οποία χρησιμοποιείται για την ταυτοποίηση του μπλοκ και η ίδια η εντολή (LOAD R4,0(B)) ώστε να ανακτηθεί όταν αυτή ζητηθεί. Επίσης, βλέπουμε ότι εμφανίστηκε αστοχία στην αναζήτηση του μπλοκ αυτού.

Κάτω από το παράθυρο της κρυφή μνήμης εντολών βλέπουμε τις προσπελάσεις της κρυφής μνήμης εντολών (Access Cache = 2), τις ευστοχίες (Hits Cache = 0) και το ποσοστό ευστοχίας (Hit Rate = 0).

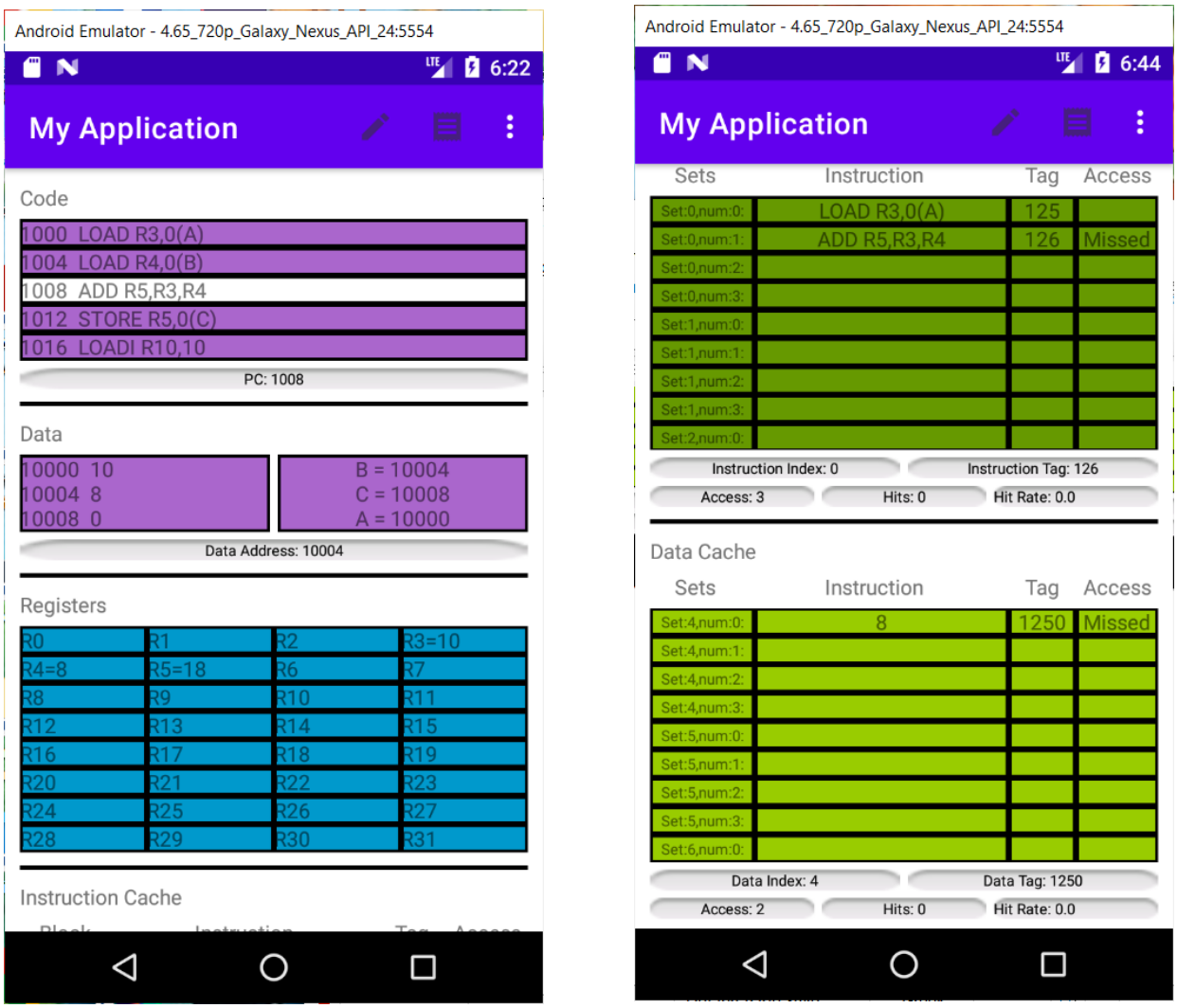
Αντίστοιχα, το δεδομένο αποθηκεύτηκε στο επόμενο ελεύθερο μπλοκ που βρέθηκε, στο σύνολο της συσχετιστικής κρυφής μνήμης δεδομένων που υποδεικνύει ο δείκτης δεδομένων (4). Στο μπλοκ της κρυφής μνήμης αποθηκεύτηκε η ετικέτα δεδομένων (1250), η οποία χρησιμοποιείται για την ταυτοποίηση του μπλοκ και το δεδομένο της διεύθυνσης μνήμης 10004 (8) ώστε να ανακτηθεί όταν αυτό ζητηθεί. Επίσης βλέπουμε ότι εμφανίστηκε αστοχία στην αναζήτηση του μπλοκ αυτού.



Εικόνα 5.7: Εκτέλεση της εντολής LOAD R4,0(B)

Κάτω από το παράθυρο της κρυφή μνήμης δεδομένων βλέπουμε τις προσπελάσεις της κρυφής μνήμης δεδομένων (Access Cache = 2), τις ευστοχίες (Hits Cache = 0) και το ποσοστό ευστοχίας (Hit Rate=0).

Στην Εικόνα 5.8 φαίνεται η εκτέλεση της εντολής ADD R5,R3,R4. Παρατηρούμε τον PC να δείχνει στην επόμενη εντολή (1008).



Εικόνα 5.8: Εκτέλεση της εντολής ADD R5,R3,R4

Παρατηρούμε ότι στον καταχωρητή R5 έχει αποθηκευτεί η τιμή (18) που προκύπτει από την πρόσθεση των τιμών που περιέχουν οι καταχωρητές R3 και R4.

Η εντολή αποθηκεύτηκε στο επόμενο ελεύθερο μπλοκ που βρέθηκε, στο σύνολο της συσχετιστικής κρυφής μνήμης εντολών που υποδεικνύει ο δείκτης εντολών (0). Στο μπλοκ της κρυφής μνήμης αποθηκεύτηκε η ετικέτα εντολών (126), η οποία χρησιμοποιείται για την ταυτοποίηση του μπλοκ και η ίδια η εντολή (ADD R5,R3,R4) ώστε να ανακτηθεί όταν αυτή ζητηθεί. Επίσης, βλέπουμε ότι εμφανίστηκε αστοχία στην αναζήτηση του μπλοκ αυτού.

Κάτω από το παράθυρο της κρυφή μνήμης εντολών βλέπουμε τις προσπελάσεις της κρυφής μνήμης εντολών (Access Cache = 3), τις ευστοχίες (Hits Cache = 0) και το ποσοστό ευστοχίας (Hit Rate = 0).

Σε αυτήν την εντολή δεν έχουμε αναζήτηση από την μνήμη, οπότε δεν έχουμε κάποια αλλαγή

στην κρυφή μνήμη δεδομένων, στον δείκτη και την ετικέτα δεδομένων, στις προσπελάσεις, στις ευστοχίες και στο ποσοστό ευστοχίας της κρυφής μνήμης δεδομένων.

Στην Εικόνα 5.9 φαίνεται η εκτέλεση της εντολής STORE R5,0(C). Παρατηρούμε τον PC να δείχνει στην επόμενη εντολή (1012).

Το δεδομένο μας (18) που υποδεικνύει ο καταχωρητής R5 αποθηκεύτηκε στην διεύθυνση που υποδεικνύει η μεταβλητή C (10008).

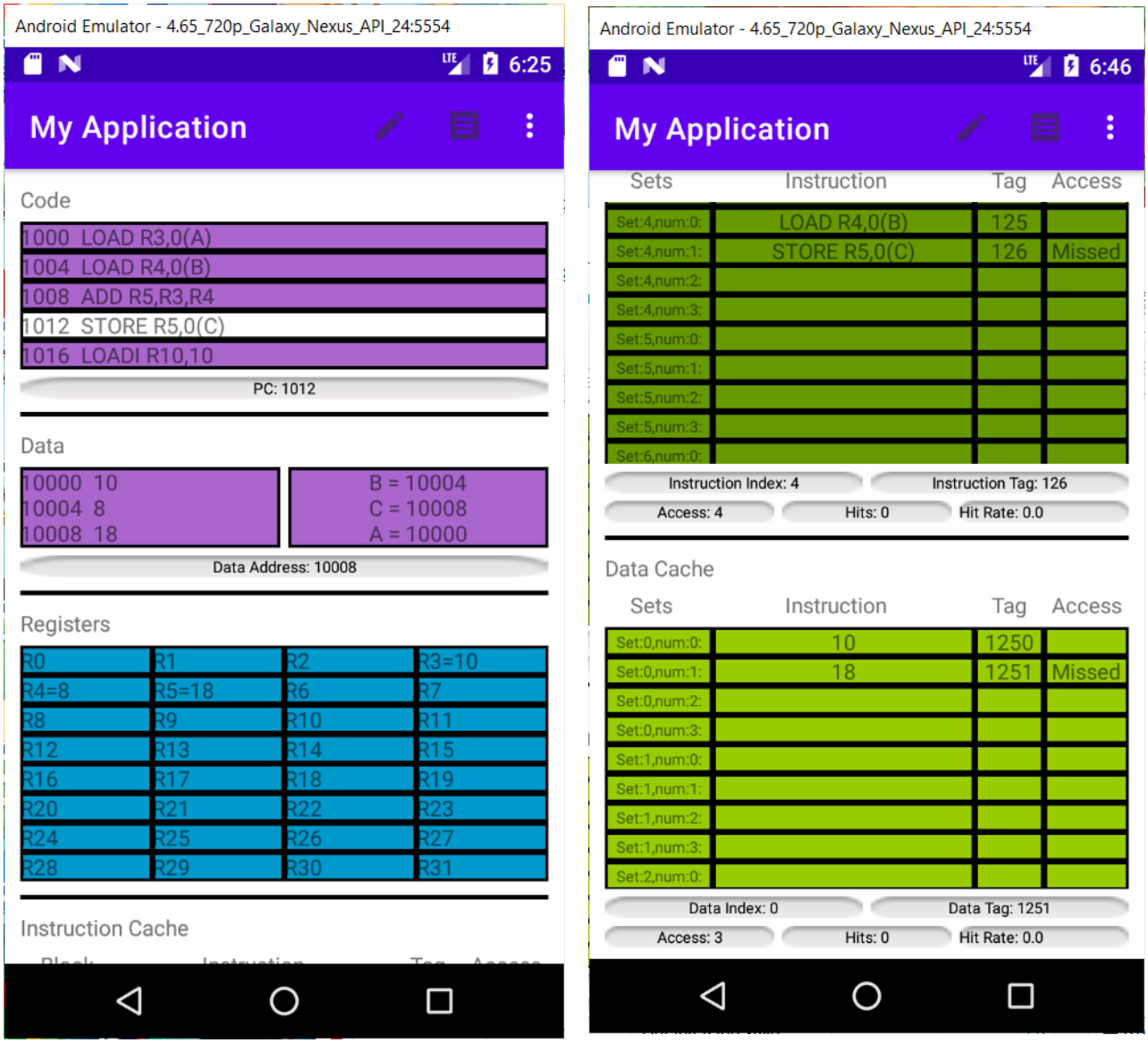
Η εντολή αποθηκεύτηκε στο επόμενο ελεύθερο μπλοκ που βρέθηκε, στο σύνολο της συσχετιστικής κρυφής μνήμης εντολών που υποδεικνύει ο δείκτης εντολών (4). Στο μπλοκ της κρυφής μνήμης αποθηκεύτηκε η ετικέτα εντολών (126), η οποία χρησιμοποιείται για την ταυτοποίηση του μπλοκ και η ίδια η εντολή (STORE R5,0(C)) ώστε να ανακτηθεί όταν αυτή ζητηθεί. Επίσης, βλέπουμε ότι εμφανίστηκε αστοχία στην αναζήτηση του μπλοκ αυτού.

Κάτω από το παράθυρο της κρυφή μνήμης εντολών βλέπουμε τις προσπελάσεις της κρυφής μνήμης εντολών (Access Cache = 4), τις ευστοχίες (Hits Cache = 0) και το ποσοστό ευστοχίας (Hit Rate = 0).

Αντίστοιχα το δεδομένο αποθηκεύτηκε στο επόμενο ελεύθερο μπλοκ που βρέθηκε, στο σύνολο της συσχετιστικής κρυφής μνήμης δεδομένων που υποδεικνύει ο δείκτης δεδομένων (0). Στο μπλοκ της κρυφής μνήμης αποθηκεύτηκε η ετικέτα δεδομένων (1251), η οποία χρησιμοποιείται για την ταυτοποίηση του μπλοκ και το δεδομένο (18) ώστε να ανακτηθεί όταν αυτό ζητηθεί. Επίσης, βλέπουμε ότι εμφανίστηκε αστοχία στην αναζήτηση του μπλοκ αυτού.

Κάτω από το παράθυρο της κρυφή μνήμης δεδομένων βλέπουμε τις προσπελάσεις της κρυφής μνήμης δεδομένων (Access Cache = 3), τις ευστοχίες (Hits Cache = 0) και το ποσοστό ευστοχίας (Hit Rate=0).

Παρατηρούμε ότι επειδή η εφαρμογή μας υποστηρίζει την ταυτόχρονη εγγραφή, έχουμε εγγραφή του δεδομένου και στην κρυφή μνήμη δεδομένων και στην κύρια μνήμη.

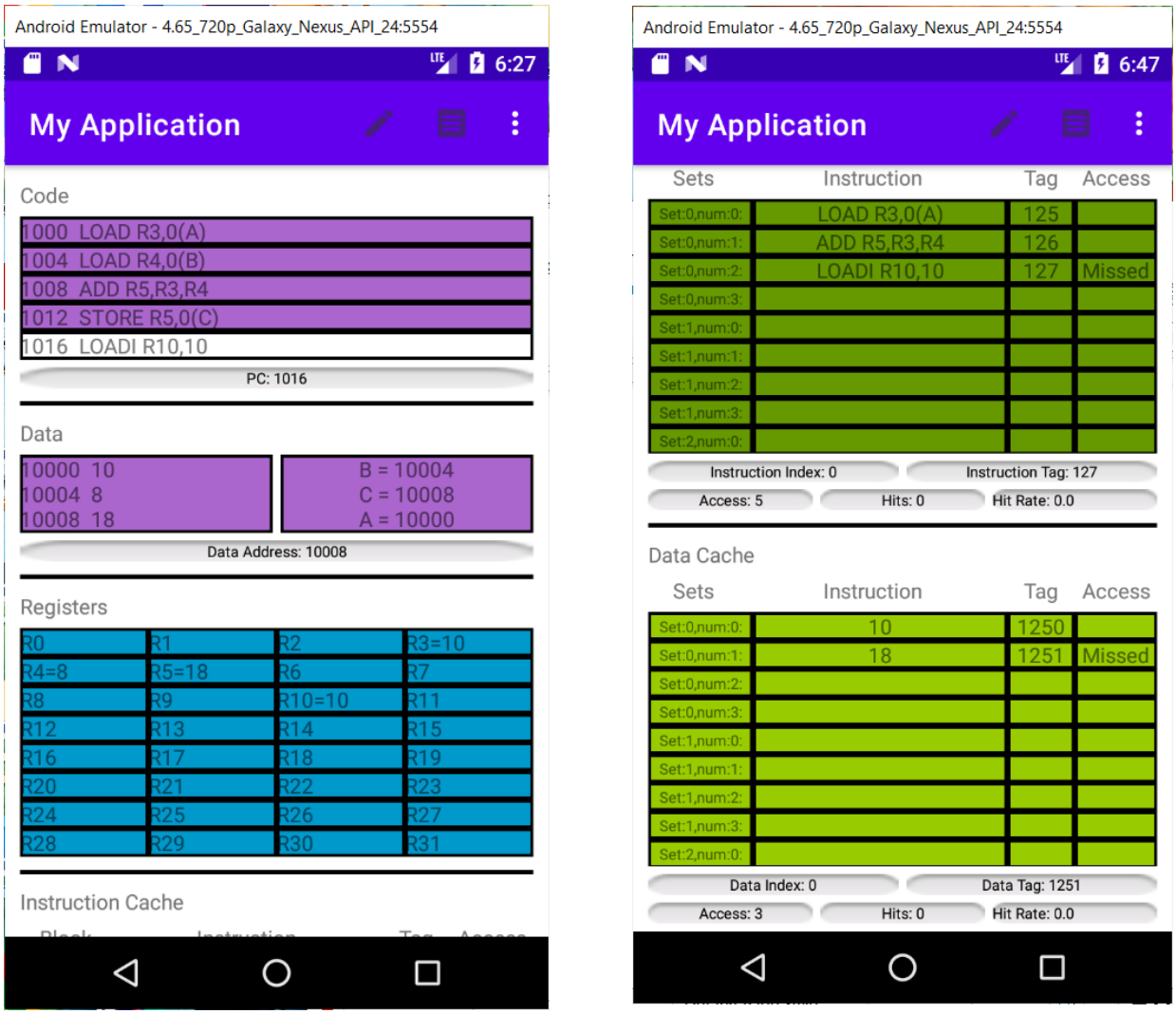


Εικόνα 5.9: Εκτέλεση της εντολής STORE R5,0(C)

Στην Εικόνα 5.10 φαίνεται η εκτέλεση της εντολής LOADI R10,10. Παρατηρούμε τον PC να δείχνει στην επόμενη εντολή (1016).

Παρατηρούμε ότι στον καταχωρητή R10 έχει αποθηκευτεί η τιμή 10.

Η εντολή αποθηκεύτηκε στο επόμενο ελεύθερο μπλοκ που βρέθηκε, στο σύνολο της συσχετιστικής κρυφής μνήμης εντολών που υποδεικνύει ο δείκτης εντολών (0). Στο μπλοκ της κρυφής μνήμης αποθηκεύτηκε η ετικέτα εντολών (127), η οποία χρησιμοποιείται για την ταυτοποίηση του μπλοκ και η ίδια η εντολή (LOADI R10,10) ώστε να ανακτηθεί όταν αυτή ζητηθεί. Επίσης, βλέπουμε ότι εμφανίστηκε αστοχία στην αναζήτηση του μπλοκ αυτού.



Εικόνα 5.10: Εκτέλεση της εντολής LOADI R10,10

Κάτω από το παράθυρο της κρυφή μνήμης εντολών βλέπουμε τις προσπελάσεις της κρυφής μνήμης εντολών (Access Cache = 5), τις ευστοχίες (Hits Cache = 0) και το ποσοστό ευστοχίας (Hit Rate = 0).

Σε αυτήν την εντολή δεν έχουμε αναζήτηση από την μνήμη, οπότε δεν έχουμε κάποια αλλαγή στην κρυφή μνήμη δεδομένων, στον δείκτη και την ετικέτα δεδομένων, στις προσπελάσεις, στις ευστοχίες και στο ποσοστό ευστοχίας της κρυφής μνήμης δεδομένων.

ΚΕΦΑΛΑΙΟ 6

ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΠΡΟΟΠΤΙΚΕΣ

6.1 Σύνοψη της διπλωματικής εργασίας

Σε ένα υπολογιστικό σύστημα ο χρόνος που δαπανά ο επεξεργαστής για την εκτέλεση και ολοκλήρωση μιας εργασίας εξαρτάται και από την ταχύτητα της μνήμης με την οποία συνεργάζεται για την εκτέλεση της εργασίας. Διαχρονικά η απόδοση του επεξεργαστή σε σχέση με την απόδοση της μνήμης ήταν αποκλίνουσα με αποτέλεσμα την επιβράδυνση εκτέλεσης των εργασιών από τον επεξεργαστή.

Η λύση που επινοήθηκε από τους μηχανικούς κατασκευής υπολογιστικών συστημάτων ήταν να ιεραρχήσουν τη μνήμη σε επίπεδα με το πλησιέστερο επίπεδο στον επεξεργαστή να είναι η κρυφή μνήμη.

Η λειτουργία των διαφόρων ειδών των κρυφών μνημών βοηθάει στην κατανόηση των ειδών των αστοχιών που εμφανίζονται στην μνήμη και συντελούν στην επιβράδυνση εκτέλεσης εργασιών.

Η αντιμετώπιση των αστοχιών δεν είναι πάντα μονόδρομη αφού διάφορες λύσεις μείωσης των αστοχιών οδηγούν στην εμφάνιση νέων προβλημάτων. Μια λύση για την μείωση της ποινής αστοχίας είναι η χρήση πολυεπίπεδων κρυφών μνημών, μια ιδέα που προέκυψε ως απάντηση στο ερώτημα αν πρέπει η κρυφή μνήμη να είναι γρήγορη ή πρέπει να είναι μεγάλη.

Για την λειτουργία και την μέτρηση της απόδοσης του επεξεργαστή και του συστήματος μνήμης υπάρχουν διάφορα μετροπρογράμματα, κάθε ένα κατάλληλο για συγκεκριμένες εφαρμογές. Κάθε υπολογιστικό σύστημα μπορεί να παρουσιάζει διαφορετικές μετρήσεις για διαφορετικά είδη εφαρμογών. Με αυτόν τον τρόπο οι μηχανικοί μπορούν να έχουν μια εικόνα του υπολογιστικού συστήματος και με αυτό τον τρόπο να μεταβούν στις κατάλληλες σχεδιάσεις του αναλόγως και τις απαιτήσεις που έχουν τεθεί.

Για την κατανόηση της λειτουργίας καθώς και των παραγόντων που επηρεάζουν την απόδοση του υπολογιστικού συστήματος έχουν δημιουργηθεί και διάφοροι προσομοιωτές επεξεργαστών και μνήμης. Με την χρήση ενός προσομοιωτή κρυφής μνήμης μπορούμε να δούμε πως η παραμετροποίηση της κρυφής μνήμης επηρεάζει την απόδοση της.

Για να γίνουν εμφανείς οι επιπτώσεις που έχουν οι παραμετροποιήσεις μια κρυφής μνήμης, αναφέρουμε στη συνέχεια διάφορες μετρήσεις μετροπρογραμμάτων. Βέβαια οι μετρήσεις αυτές διαφέρουν από υπολογιστικό σύστημα σε υπολογιστικό σύστημα.

Όσον αφορά την μέθοδο αντικατάστασης του μπλοκ η χρήση μετροπρογραμμάτων έχει δείξει ότι η μέθοδος αντικατάστασης LRU εμφανίζει λιγότερες αστοχίες σε μικρή κρυφή μνήμη ανεξαρτήτου συσχετιστικότητας. Όσο όμως αυξάνεται το μέγεθος της κρυφής μνήμης οι αστοχίες είναι περίπου ίδιες για όλες τις μεθόδους αντικατάστασης μπλοκ.

Επίσης η χρήση μετροπρογραμμάτων έχει δείξει ότι η αύξηση της συσχετιστικότητας όπως και η αύξηση του μεγέθους της μνήμης μειώνει τον ρυθμό αστοχίας. Βέβαια η μείωση του ρυθμού αστοχίας είναι κυρίως μηδαμινή από συσχετιστική μνήμη τεσσάρων δρόμων σε συσχετιστική μνήμη οχτώ δρόμων ανεξαρτήτου μεγέθους της κρυφής μνήμης. Επίσης, με βάση τις μετρήσεις παρατηρείται πολύ μεγαλύτερη μείωση του ρυθμού αστοχίας όταν αυξάνεται το μέγεθος από ότι όταν αυξάνεται η συσχετιστικότητα.

Αυτό όμως δε σημαίνει ότι η μεγάλη συσχετιστικότητα είναι και η καλύτερη λύση πάντοτε. Αυτό δείχνει και η χρήση μετροπρογραμμάτων για την μέτρηση του μέσου χρόνου προσπέλασης της μνήμης για κρυφές μνήμες διαφόρων μεγεθών και συσχετιστικότητας. Οι μετρήσεις δείχνουν ότι μειώνεται ο μέσος χρόνος προσπέλασης όσο αυξάνεται το μέγεθος της κρυφής μνήμης αλλά αυξάνεται όταν μεγαλώνει η συσχετιστικότητα λόγω της αύξησης του χρόνου ευστοχίας.

Τέλος να αναφέρουμε ότι η χρήση μετροπρογραμμάτων δείχνει ανεβοκατεβάσματα στον ρυθμό αστοχίας για κρυφές μνήμες με διαφορετικό μέγεθος μπλοκ και διαφορετικό μέγεθος κρυφής μνήμης για κάθε μέγεθος μπλοκ. Η αύξηση του μεγέθους της κρυφής μνήμης ανεξαρτήτου επιλογής του μεγέθους του μπλοκ μειώνει τον ρυθμό αστοχίας. Από την άλλη η αύξηση του μεγέθους του μπλοκ δεν μειώνει πάντα τον ρυθμό αστοχίας.

Από τις μετρήσεις των μετροπρογραμμάτων φαίνεται ότι το μέγεθος της κρυφής μνήμης αποτελεί σημαντικό παράγοντα στην εμφάνιση αστοχιών και άρα στον μέσο χρόνο προσπέλασης της μνήμης. Αλλά όπως εξηγήσαμε το μέγεθος της μνήμης έχει και μειονεκτήματα του.

6.2 Προοπτικές

Αντικείμενο της παρούσας διπλωματικής εργασίας είναι η υλοποίηση ενός προσομοιωτή

κρυφής μνήμης που θα παρέχει ένα ελκυστικό και εύκολα κατανοητό εργαλείο στους φοιτητές στη μελέτη εννοιών που σχετίζονται με την κρυφή μνήμη. Ο βαθμός που αυτό επιτεύχθηκε θα φανεί με τη χρήση του προσομοιωτή από φοιτητές μαθημάτων που περιλαμβάνουν την έννοια της κρυφής μνήμης.

ΠΑΡΑΡΤΗΜΑ

Στο παράρτημα αυτό παρατίθεται ο κώδικας ανάπτυξης της παρούσας εφαρμογής.

Κλάση MainActivity.java

```
package com.example.myapplication;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.GridLayoutManager;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import android.content.Intent;
import android.os.Bundle;
import android.text.method.ScrollingMovementMethod;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Scanner;


public class MainActivity extends AppCompatActivity
{
   List<entoli> listEntolon = new ArrayList<entoli>();
   List<String> listKataxoriton = new ArrayList<String>();
   List<BlockKmEntolon> listKMEntolon = new ArrayList<BlockKmEntolon>();
   List<BlockKmDedomenon> listKMDedomenon = new ArrayList<BlockKmDedomenon>();
   private RecyclerView recyclerViewEntolon;
   private EntolesAdapter mEntolesAdapter;
   private RecyclerView recyclerViewKataxoriton;
   private KataxoritesAdapter mKataxoritesAdapter;
   private RecyclerView recyclerViewKMEntolon;
   private KMEntolonAdapter mKMEntolonAdapter;
   private RecyclerView recyclerViewKMDedomenon;
   private KMDedomenonAdapter mKMDedomenonAdapter;
   private static final int REQUEST_CODE_CONF = 0;
```

```
    private static final String TAG = "values";
    private Button instrRun;
    private Button codeRun;
    private TextView dedomenaStinMnimi;
    private TextView metavlitesDieuthinsisTxtView;
    private TextView dieutEntolon;
    private TextView dieutDedomenon;
    private TextView intDeiktisEntolon;
    private TextView intEtiketaEntolon;
    private TextView intDeiktisDedomenon;
    private TextView intEtiketaDedomenon;
    private TextView accessKMEntolon;
    private TextView hitsKMEntolon;
    private TextView hitRateKMEntolon;
    private TextView accessKMDedomenon;
    private TextView hitsKMDedomenon;
    private TextView hitRateKMDedomenon;
    private TextView instructionNumTextView;
    private TextView dataNumTextView;
    static String dieutDedString;
    static String metavlitesDieuthinsis;
    static int blockLineInstruction;
    static int blockLineData;
    static boolean code_is_running = false;


//=================================================================================
===========//

    static int cacheSizeEnt = 128;
    static int blockSIzeEnt = 4;
    static int arithmosBlocksEnt = cacheSizeEnt/blockSIzeEnt;
    static int sinolaCacheEnt;
    static int blocksSinolouCacheEnt = 8;
    static String[][] sisxetikiCacheEntolon;
    static String[][] sisxetikiCacheEntolon2;
    static int[][] LRUpinEnt;
    static int[] LRUdeiktisEnt;
    static int[][] FIFOpinEnt;
    static int[] FIFOdeiktisEnt;
    static String[] plirosSisxetikiCacheEntolon;
    static String[] plirosSisxetikiCacheEntolon2;
    static int[] LRUpinEntPS;
    static int LRUdeiktisEntPS;
    static int[] FIFOpinEntPS;
    static int FIFOdeiktisEntPS;
    static String cacheMappingEnt = "Direct Mapping";
    static String writingPolicyEnt = "LRU";
    static double hitsEntolon = 0;
    static double accessCacheEntolon = 0;

    static int cacheSizeDed = 128;
    static int blockSIzeDed = 4;
    static int arithmosBlocksDed = cacheSizeDed/blockSIzeDed;
    static int sinolaCacheDed;
    static int blocksSinolouCacheDed = 8;
```

```
    static String[][] sisxetikiCacheDedomenon;
    static String[][] sisxetikiCacheDedomenon2;
    static int[][] LRUpinDed;
    static int[] LRUdeiktisDed;
    static int[][] FIFOpinDed;
    static int[] FIFOdeiktisDed;
    static String[] plirosSisxetikiCacheDedomenon;
    static String[] plirosSisxetikiCacheDedomenon2;
    static int[] LRUpinDedPS;
    static int LRUdeiktisDedPS;
    static int[] FIFOpinDedPS;
    static int FIFOdeiktisDedPS;
    static String cacheMappingDed = "Direct Mapping";
    static String writingPolicyDed = "LRU";;
    static double hitsDedomenon = 0;
    static double accessCacheDedomenon = 0;

    static int bitsDeiktiEntolon;
    static char[] charDeiktisEntBin;
    static int bitsDeiktiDedomenon;
    static char[] charDeiktisDedBin;
    static int intDeiktisEnt;
    static int intEtiketaEnt;
    static int intDeiktisDed;
    static int intEtiketaDed;

    static HashMap<Integer,Integer> antisDeiktisEtiketaEntolon = new HashMap<Integer,Integer>();
    static HashMap<Integer,String> antisDeiktisEtiketaEntolon2 = new HashMap<Integer,String>();
    static HashMap<Integer,Integer> antisEtiketaEntolon = new HashMap<Integer,Integer>();
    static HashMap<Integer,String> antisEtiketaEntolon2 = new HashMap<Integer,String>();
    static HashMap<Integer,Integer> antisDeiktisEtiketaDedomenon = new HashMap<Integer,Integer>();
    static HashMap<Integer,String> antisDeiktisEtiketaDedomenon2 = new HashMap<Integer,String>();
    static HashMap<Integer,Integer> antisEtiketaDedomenon = new HashMap<Integer,Integer>();
    static HashMap<Integer,String> antisEtiketaDedomenon2 = new HashMap<Integer,String>();


//================================================================
============================//

    static void setupDeiktisCache()
    {
       if (cacheMappingEnt.equals("Direct Mapping"))
       {
          bitsDeiktiEntolon = (int)(Math.log(arithmosBlocksEnt) / Math.log(2));
          charDeiktisEntBin = new char [bitsDeiktiEntolon];
       }
       else if(cacheMappingEnt.equals("Associative"))
       {
            sinolaCacheEnt = arithmosBlocksEnt/blocksSinolouCacheEnt;
            bitsDeiktiEntolon = (int)(Math.log(sinolaCacheEnt) / Math.log(2));
            charDeiktisEntBin = new char [bitsDeiktiEntolon];
            sisxetikiCacheEntolon = new String[sinolaCacheEnt][blocksSinolouCacheEnt];
            sisxetikiCacheEntolon2 = new String[sinolaCacheEnt][blocksSinolouCacheEnt];
            LRUpinEnt = new int[sinolaCacheEnt][blocksSinolouCacheEnt];
            LRUdeiktisEnt = new int[sinolaCacheEnt];
            FIFOpinEnt = new int[sinolaCacheEnt][blocksSinolouCacheEnt];
```

```
            FIFOdeiktisEnt = new int[sinolaCacheEnt];

            for(int i=0;i<sinolaCacheEnt;i++)
            {
                LRUdeiktisEnt[i] = 0;
                FIFOdeiktisEnt[i] = 0;
            }
        }
        else if(cacheMappingEnt.equals("Full Associative"))
        {
            intDeiktisEnt = 0;

            LRUpinEntPS = new int[arithmosBlocksEnt];
            FIFOpinEntPS = new int[arithmosBlocksEnt];

            plirosSisxetikiCacheEntolon = new String[arithmosBlocksEnt];
            plirosSisxetikiCacheEntolon2 = new String[arithmosBlocksEnt];
        }

        if (cacheMappingDed.equals("Direct Mapping"))
        {
            bitsDeiktiDedomenon = (int)(Math.log(arithmosBlocksDed) / Math.log(2));
            charDeiktisDedBin = new char [bitsDeiktiDedomenon];
        }
        else if(cacheMappingDed.equals("Associative"))
        {
            sinolaCacheDed  = arithmosBlocksDed/blocksSinolouCacheDed;
            bitsDeiktiDedomenon = (int)(Math.log(sinolaCacheDed) / Math.log(2));
            charDeiktisDedBin = new char [bitsDeiktiDedomenon];
            sisxetikiCacheDedomenon = new String[sinolaCacheDed][blocksSinolouCacheDed];
            sisxetikiCacheDedomenon2 = new String[sinolaCacheDed[blocksSinolouCacheDed];
            LRUpinDed = new int[sinolaCacheDed][blocksSinolouCacheDed];
            LRUdeiktisDed = new int[sinolaCacheDed];
            FIFOpinDed = new int[sinolaCacheDed][blocksSinolouCacheDed];
            FIFOdeiktisDed = new int[sinolaCacheDed];

            for(int i=0;i<sinolaCacheDed;i++)
            {
                LRUdeiktisDed[i] = 0;
                FIFOdeiktisDed[i] = 0;
            }
        }
        else if(cacheMappingDed.equals("Full Associative"))
        {
            intDeiktisDed = 0;

            LRUpinDedPS = new int[arithmosBlocksDed];
            FIFOpinDedPS = new int[arithmosBlocksDed];

            plirosSisxetikiCacheDedomenon = new String[arithmosBlocksDed];
            plirosSisxetikiCacheDedomenon2 = new String[arithmosBlocksDed];
        }
    }

    private void cacheEntolon(int dieuthEnt,String mOlokliriEntoli)
    {
```

```
int bitsEtiketaEntolon;
char[] charEtiketaEntBin;
intDeiktisEnt = dieuthEnt;
intEtiketaEnt = dieuthEnt;

accessCacheEntolon = accessCacheEntolon + 1;

String dieuthEntBin = Integer.toBinaryString(dieuthEnt);


char[] charDieuthEntBin;
if(dieuthEntBin.length() <= bitsDeiktiEntolon)
{
   String zero_bit = "0";
   for(int i=0;i<(bitsDeiktiEntolon-dieuthEntBin.length());i++)
   {
      zero_bit = zero_bit.concat("0");
   }

   dieuthEntBin = zero_bit.concat(dieuthEntBin);

   charDieuthEntBin = new char[dieuthEntBin.length()];
   dieuthEntBin.getChars(0, dieuthEntBin.length(), charDieuthEntBin,0);
}
else
{
   charDieuthEntBin = new char[dieuthEntBin.length()];
   dieuthEntBin.getChars(0, dieuthEntBin.length(), charDieuthEntBin, 0);
}



if (cacheMappingEnt.equals("Direct Mapping"))
{
   bitsEtiketaEntolon = dieuthEntBin.length() - bitsDeiktiEntolon;
   charEtiketaEntBin = new char [bitsEtiketaEntolon];

   int j = 0;
   for(int i=0;i<charDieuthEntBin.length;i++)
   {
      if(i<bitsEtiketaEntolon)
      {
         charEtiketaEntBin[i] = charDieuthEntBin[i];
      }
      else
      {
         charDeiktisEntBin[j] = charDieuthEntBin[i];
         j = j + 1;
      }
   }

   intDeiktisEnt = Integer.parseInt(String.valueOf(charDeiktisEntBin),2);
   intEtiketaEnt = Integer.parseInt(String.valueOf(charEtiketaEntBin),2);
}
else if(cacheMappingEnt.equals("Associative"))
{
   bitsEtiketaEntolon = dieuthEntBin.length() - bitsDeiktiEntolon;
```

```
        charEtiketaEntBin = new char [bitsEtiketaEntolon];

        int j = 0;
        for(int i=0;i<charDieuthEntBin.length;i++)
        {
           if(i<bitsEtiketaEntolon)
           {
              charEtiketaEntBin[i] = charDieuthEntBin[i];
           }
           else
           {
              charDeiktisEntBin[j] = charDieuthEntBin[i];
              j = j + 1;
           }
        }

        intDeiktisEnt = Integer.parseInt(String.valueOf(charDeiktisEntBin),2);
        intEtiketaEnt = Integer.parseInt(String.valueOf(charEtiketaEntBin),2);
     }
     else if(cacheMappingEnt.equals("Full Associative"))
     {
        intEtiketaEnt = dieuthEnt;
     }


//=================================================================
=================//

     if (cacheMappingEnt.equals("Direct Mapping"))
     {
        blockLineInstruction = intDeiktisEnt;

        if(antisDeiktisEtiketaEntolon.get(intDeiktisEnt) != null)
        {
           if(antisDeiktisEtiketaEntolon.get(intDeiktisEnt) == intEtiketaEnt)
           {
              hitsEntolon = hitsEntolon + 1;
              listKMEntolon.get(intDeiktisEnt).access = true;

              antisDeiktisEtiketaEntolon.put(intDeiktisEnt,intEtiketaEnt);
              antisDeiktisEtiketaEntolon2.put(intDeiktisEnt,mOlokliriEntoli);
           }
           else
           {
              antisDeiktisEtiketaEntolon.put(intDeiktisEnt,intEtiketaEnt);
              antisDeiktisEtiketaEntolon2.put(intDeiktisEnt,mOlokliriEntoli);
           }
        }
        else
        {
           antisDeiktisEtiketaEntolon.put(intDeiktisEnt,intEtiketaEnt);
           antisDeiktisEtiketaEntolon2.put(intDeiktisEnt,mOlokliriEntoli);
        }
        for (int i : antisDeiktisEtiketaEntolon.keySet())
        {
           listKMEntolon.get(i).olokEntoli = antisDeiktisEtiketaEntolon2.get(i);
```

```
            listKMEntolon.get(i).etiketa = Integer.toString(antisDeiktisEtiketaEntolon.get(i));
            mKMEntolonAdapter.setList(listKMEntolon,blockLineInstruction);
            mKMEntolonAdapter.notifyDataSetChanged();
        }
    }
    else if(cacheMappingEnt.equals("Associative"))
    {
        boolean hitTick = false;
        int keniThesi = -1;
        for(int j=0;j<blocksSinolouCacheEnt;j++)
        {
            if(sisxetikiCacheEntolon[intDeiktisEnt][j]!=null)
            {
                if (Integer.parseInt(sisxetikiCacheEntolon[intDeiktisEnt][j]) == intEtiketaEnt)
                {
                    blockLineInstruction = (intDeiktisEnt * blocksSinolouCacheEnt) + j;

                    hitsEntolon = hitsEntolon + 1;
                    listKMEntolon.get((intDeiktisEnt * blocksSinolouCacheEnt) + j).access = true;
                    hitTick = true;
                    if (writingPolicyEnt.equals("LRU"))
                    {
                        if (LRUdeiktisEnt[intDeiktisEnt] < blocksSinolouCacheEnt)
                        {
                            LRUpinEnt[intDeiktisEnt][j] = LRUdeiktisEnt[intDeiktisEnt];

                            LRUdeiktisEnt[intDeiktisEnt] = LRUdeiktisEnt[intDeiktisEnt] + 1;
                        }
                        else
                        {
                            for (int k = 0; k < blocksSinolouCacheEnt; k++)
                            {
                                LRUpinEnt[intDeiktisEnt][k] = LRUpinEnt[intDeiktisEnt][k] - 1;
                            }
                            LRUpinEnt[intDeiktisEnt][j] = LRUdeiktisEnt[intDeiktisEnt];
                        }
                    }
                }
            }
            if (sisxetikiCacheEntolon[intDeiktisEnt][j] == null
                    || sisxetikiCacheEntolon[intDeiktisEnt][j].length() == 0)
            {
                if (keniThesi < 0)
                {
                    keniThesi = j;
                }
            }
        }

        if(!hitTick && keniThesi>=0)
        {
            blockLineInstruction = (intDeiktisEnt*blocksSinolouCacheEnt)+keniThesi;

            sisxetikiCacheEntolon[intDeiktisEnt][keniThesi] = Integer.toString(intEtiketaEnt);
            sisxetikiCacheEntolon2[intDeiktisEnt][keniThesi] = mOlokliriEntoli;
```

```
        if(writingPolicyEnt.equals("LRU"))
        {
            if(LRUdeiktisEnt[intDeiktisEnt]<blocksSinolouCacheEnt)
            {
                LRUpinEnt[intDeiktisEnt][keniThesi] = LRUdeiktisEnt[intDeiktisEnt];

                LRUdeiktisEnt[intDeiktisEnt] = LRUdeiktisEnt[intDeiktisEnt] + 1;
            }
            else
            {
                for(int k=0;k<blocksSinolouCacheEnt;k++)
                {
                    LRUpinEnt[intDeiktisEnt][k] = LRUpinEnt[intDeiktisEnt][k] - 1;
                }
                LRUpinEnt[intDeiktisEnt][keniThesi] = LRUdeiktisEnt[intDeiktisEnt];
            }
        }
        else if(writingPolicyEnt.equals("FIFO"))
        {
            if(FIFOdeiktisEnt[intDeiktisEnt]<blocksSinolouCacheEnt)
            {
                FIFOpinEnt[intDeiktisEnt][keniThesi] = FIFOdeiktisEnt[intDeiktisEnt];

                FIFOdeiktisEnt[intDeiktisEnt] = FIFOdeiktisEnt[intDeiktisEnt] + 1;
            }
            else
            {
                for(int k=0;k<blocksSinolouCacheEnt;k++)
                {
                    FIFOpinEnt[intDeiktisEnt][k] = FIFOpinEnt[intDeiktisEnt][k] - 1;
                }
                FIFOpinEnt[intDeiktisEnt][keniThesi] = FIFOdeiktisEnt[intDeiktisEnt];
            }
        }
    }
    if(!hitTick && keniThesi<0)
    {
        int thesi = 0;
        if(writingPolicyEnt.equals("random"))
        {
            thesi = (int)(Math.random()*((blocksSinolouCacheEnt-1)-0+1)+0);

            sisxetikiCacheEntolon[intDeiktisEnt][thesi] = Integer.toString(intEtiketaEnt);
            sisxetikiCacheEntolon2[intDeiktisEnt][thesi] = mOlokliriEntoli;
        }
        else if(writingPolicyEnt.equals("LRU"))
        {
            thesi = 0;
            for(int j=1;j<blocksSinolouCacheEnt;j++)
            {
                if(LRUpinEnt[intDeiktisEnt][thesi]<LRUpinEnt[intDeiktisEnt][j])
                {
                    thesi = thesi;
                }
                else
                {
```

```
                thesi = j;
            }
        }
        sisxetikiCacheEntolon[intDeiktisEnt][thesi] = Integer.toString(intEtiketaEnt);
        sisxetikiCacheEntolon2[intDeiktisEnt][thesi] = mOlokliriEntoli;

        if(LRUdeiktisEnt[intDeiktisEnt]<blocksSinolouCacheEnt)
        {
            LRUpinEnt[intDeiktisEnt][thesi] = LRUdeiktisEnt[intDeiktisEnt];

            LRUdeiktisEnt[intDeiktisEnt] = LRUdeiktisEnt[intDeiktisEnt] + 1;
        }
        else
        {
            for(int k=0;k<blocksSinolouCacheEnt;k++)
            {
                LRUpinEnt[intDeiktisEnt][k] = LRUpinEnt[intDeiktisEnt][k] - 1;
            }
            LRUpinEnt[intDeiktisEnt][thesi] = LRUdeiktisEnt[intDeiktisEnt];
        }
    }
    else if(writingPolicyEnt.equals("FIFO"))
    {
        thesi = 0;
        for(int j=1;j<blocksSinolouCacheEnt;j++)
        {
            if(FIFOpinEnt[intDeiktisEnt][thesi]<FIFOpinEnt[intDeiktisEnt][j])
            {
                thesi = thesi;
            }
            else
            {
                thesi = j;
            }
        }
        sisxetikiCacheEntolon[intDeiktisEnt][thesi] = Integer.toString(intEtiketaEnt);
        sisxetikiCacheEntolon2[intDeiktisEnt][thesi] = mOlokliriEntoli;

        if(FIFOdeiktisEnt[intDeiktisEnt]<blocksSinolouCacheEnt)
        {
            FIFOpinEnt[intDeiktisEnt][thesi] = FIFOdeiktisEnt[intDeiktisEnt];

            FIFOdeiktisEnt[intDeiktisEnt] = FIFOdeiktisEnt[intDeiktisEnt] + 1;
        }
        else
        {
            for(int k=0;k<blocksSinolouCacheEnt;k++)
            {
                FIFOpinEnt[intDeiktisEnt][k] = FIFOpinEnt[intDeiktisEnt][k] - 1;
            }
            FIFOpinEnt[intDeiktisEnt][thesi] = FIFOdeiktisEnt[intDeiktisEnt];
        }
    }

    blockLineInstruction = (intDeiktisEnt*blocksSinolouCacheEnt)+thesi;
}
```

```
        for(int i=0;i<blocksSinolouCacheEnt;i++)
        {
            listKMEntolon.get((intDeiktisEnt*blocksSinolouCacheEnt)+i).olokEntoli =
sisxetikiCacheEntolon2[intDeiktisEnt][i];
            listKMEntolon.get((intDeiktisEnt*blocksSinolouCacheEnt)+i).etiketa =
sisxetikiCacheEntolon[intDeiktisEnt][i];

            mKMEntolonAdapter.setList(listKMEntolon,blockLineInstruction);
            mKMEntolonAdapter.notifyDataSetChanged();
        }
    }
    else if(cacheMappingEnt.equals("Full Associative"))
    {
        boolean hitTick = false;
        int keniThesi = -1;
        for(int j=0;j<arithmosBlocksEnt;j++)
        {
            if(plirosSisxetikiCacheEntolon[j]!=null)
            {
                if (Integer.parseInt(plirosSisxetikiCacheEntolon[j]) == intEtiketaEnt)
                {
                    blockLineInstruction = j;

                    hitsEntolon = hitsEntolon + 1;
                    listKMEntolon.get(j).access = true;
                    hitTick = true;
                    if (writingPolicyEnt.equals("LRU"))
                    {
                        if (LRUdeiktisEntPS < arithmosBlocksEnt)
                        {
                            LRUpinEntPS[j] = LRUdeiktisEntPS;

                            LRUdeiktisEntPS = LRUdeiktisEntPS + 1;
                        }
                        else
                        {
                            for (int k = 0; k < arithmosBlocksEnt; k++)
                            {
                                LRUpinEntPS[k] = LRUpinEntPS[k] - 1;
                            }
                            LRUpinEntPS[j] = LRUdeiktisEntPS;
                        }
                    }
                }
            }
            if (plirosSisxetikiCacheEntolon[j] == null
                    || plirosSisxetikiCacheEntolon[j].length() == 0)
            {
                if (keniThesi < 0)
                {
                    keniThesi = j;
                }
            }
        }

        if(!hitTick && keniThesi>=0)
```

```
{
   blockLineInstruction = keniThesi;

   plirosSisxetikiCacheEntolon[keniThesi] = Integer.toString(intEtiketaEnt);
   plirosSisxetikiCacheEntolon2[keniThesi] = mOlokliriEntoli;
   antisEtiketaEntolon.put(intEtiketaEnt,intEtiketaEnt);
   antisEtiketaEntolon2.put(intEtiketaEnt,mOlokliriEntoli);

   if(writingPolicyEnt.equals("LRU"))
   {
      if(LRUdeiktisEntPS < arithmosBlocksEnt)
      {
         LRUpinEntPS[keniThesi] = LRUdeiktisEntPS;

         LRUdeiktisEntPS = LRUdeiktisEntPS + 1;
      }
      else
      {
         for(int k=0;k<arithmosBlocksEnt;k++)
         {
            LRUpinEntPS[k] = LRUpinEntPS[k] - 1;
         }
         LRUpinEntPS[keniThesi] = LRUdeiktisEntPS;
      }
   }
   else if(writingPolicyEnt.equals("FIFO"))
   {
      if(FIFOdeiktisEntPS < arithmosBlocksEnt)
      {
         FIFOpinEntPS[keniThesi] = FIFOdeiktisEntPS;

         FIFOdeiktisEntPS = FIFOdeiktisEntPS + 1;
      }
      else
      {
         for(int k=0;k<arithmosBlocksEnt;k++)
         {
            FIFOpinEntPS[k] = FIFOpinEntPS[k] - 1;
         }
         FIFOpinEntPS[keniThesi] = FIFOdeiktisEntPS;
      }
   }
}
if(!hitTick && keniThesi<0)
{
   antisEtiketaEntolon.put(intEtiketaEnt,intEtiketaEnt);
   antisEtiketaEntolon2.put(intEtiketaEnt,mOlokliriEntoli);

   int thesi = 0;
   if(writingPolicyEnt.equals("random"))
   {
      thesi = (int)(Math.random()*((arithmosBlocksEnt-1)-0+1)+0);

      plirosSisxetikiCacheEntolon[thesi] = Integer.toString(intEtiketaEnt);
      plirosSisxetikiCacheEntolon2[thesi] = mOlokliriEntoli;
   }
```

```
else if(writingPolicyEnt.equals("LRU"))
{
   thesi = 0;
   for(int j=1;j<arithmosBlocksEnt;j++)
   {
      if(LRUpinEntPS[thesi]<LRUpinEntPS[j])
      {
         thesi = thesi;
      }
      else
      {
         thesi = j;
      }
   }
   plirosSisxetikiCacheEntolon[thesi] = Integer.toString(intEtiketaEnt);
   plirosSisxetikiCacheEntolon2[thesi] = mOlokliriEntoli;

   if(LRUdeiktisEntPS < arithmosBlocksEnt)
   {
      LRUpinEntPS[thesi] = LRUdeiktisEntPS;

      LRUdeiktisEntPS = LRUdeiktisEntPS + 1;
   }
   else
   {
      for(int k=0;k<arithmosBlocksEnt;k++)
      {
         LRUpinEntPS[k] = LRUpinEntPS[k] - 1;
      }
      LRUpinEntPS[thesi] = LRUdeiktisEntPS;
   }
}
else if(writingPolicyEnt.equals("FIFO"))
{
   thesi = 0;
   for(int j=1;j<arithmosBlocksEnt;j++)
   {
      if(FIFOpinEntPS[thesi]<FIFOpinEntPS[j])
      {
         thesi = thesi;
      }
      else
      {
         thesi = j;
      }
   }
   plirosSisxetikiCacheEntolon[thesi] = Integer.toString(intEtiketaEnt);
   plirosSisxetikiCacheEntolon2[thesi] = mOlokliriEntoli;

   if(FIFOdeiktisEntPS < arithmosBlocksEnt)
   {
      FIFOpinEntPS[thesi] = FIFOdeiktisEntPS;

      FIFOdeiktisEntPS = FIFOdeiktisEntPS + 1;
   }
   else
```

```
                {
                    for(int k=0;k<arithmosBlocksEnt;k++)
                    {
                        FIFOpinEntPS[k] = FIFOpinEntPS[k] - 1;
                    }
                    FIFOpinEntPS[thesi] = FIFOdeiktisEntPS;
                }
            }

            blockLineInstruction = thesi;
        }

        for (int i=0;i<arithmosBlocksEnt;i++)
        {
            listKMEntolon.get(i).olokEntoli = plirosSisxetikiCacheEntolon2[i];
            listKMEntolon.get(i).etiketa = plirosSisxetikiCacheEntolon[i];

            mKMEntolonAdapter.setList(listKMEntolon,blockLineInstruction);
            mKMEntolonAdapter.notifyDataSetChanged();
        }
    }

    dieutEntolon.setText("PC: "+dieuthEnt);
    if(cacheMappingEnt.equals("Direct Mapping") || cacheMappingEnt.equals("Associative"))
    {
        intDeiktisEntolon.setText("Instruction Index: "+intDeiktisEnt);
        intEtiketaEntolon.setText("Instruction Tag: "+intEtiketaEnt);
    }
    else if(cacheMappingEnt.equals("Full Associative"))
    {
        intEtiketaEntolon.setText("Instruction Tag: "+intEtiketaEnt);
    }
    accessKMEntolon.setText("Access: " + (int)accessCacheEntolon);
    hitsKMEntolon.setText("Hits: " + (int)hitsEntolon);
    hitRateKMEntolon.setText("Hit Rate: "+(hitsEntolon/accessCacheEntolon));
}
private void cacheDedomenon(int dieuthDed,String mTimi)
{
    int bitsEtiketaDedomenon;
    char[] charEtiketaDedBin;

    intDeiktisDed = dieuthDed;
    intEtiketaDed = dieuthDed;

    accessCacheDedomenon = accessCacheDedomenon + 1;

    String dieuthDedBin = Integer.toBinaryString(dieuthDed);


    char[] charDieuthDedBin;
    if(dieuthDedBin.length() <= bitsDeiktiDedomenon)
    {
        String zero_bit = "0";
        for(int i=0;i<(bitsDeiktiDedomenon-dieuthDedBin.length());i++)
        {
            zero_bit = zero_bit.concat("0");
        }
```

```
    dieuthDedBin = zero_bit.concat(dieuthDedBin);

    charDieuthDedBin = new char[dieuthDedBin.length()];
    dieuthDedBin.getChars(0, dieuthDedBin.length(), charDieuthDedBin,0);
}
else
{
    charDieuthDedBin = new char[dieuthDedBin.length()];
    dieuthDedBin.getChars(0, dieuthDedBin.length(), charDieuthDedBin, 0);
}

if (cacheMappingDed.equals("Direct Mapping"))
{
    bitsEtiketaDedomenon = dieuthDedBin.length() - bitsDeiktiDedomenon;
    charEtiketaDedBin = new char [bitsEtiketaDedomenon];

    int j = 0;
    for(int i=0;i<dieuthDedBin.length();i++)
    {
        if(i<bitsEtiketaDedomenon)
        {
            charEtiketaDedBin[i] = charDieuthDedBin[i];
        }
        else
        {
            charDeiktisDedBin[j] = charDieuthDedBin[i];
            j = j + 1;
        }
    }

    intDeiktisDed = Integer.parseInt(String.valueOf(charDeiktisDedBin),2);
    intEtiketaDed = Integer.parseInt(String.valueOf(charEtiketaDedBin),2);
}
else if(cacheMappingDed.equals("Associative"))
{
    bitsEtiketaDedomenon = dieuthDedBin.length() - bitsDeiktiDedomenon;
    charEtiketaDedBin = new char [bitsEtiketaDedomenon];

    int j = 0;
    for(int i=0;i<dieuthDedBin.length();i++)
    {
        if(i<bitsEtiketaDedomenon)
        {
            charEtiketaDedBin[i] = charDieuthDedBin[i];
        }
        else
        {
            charDeiktisDedBin[j] = charDieuthDedBin[i];
            j = j + 1;
        }
    }

    intDeiktisDed = Integer.parseInt(String.valueOf(charDeiktisDedBin),2);
    intEtiketaDed = Integer.parseInt(String.valueOf(charEtiketaDedBin),2);
}
```

```
    else if(cacheMappingDed.equals("Full Associative"))
    {
       intEtiketaDed = dieuthDed;
    }


//=================================================================================//

    String pliroforiesKMDedomenon = "";
    if (cacheMappingDed.equals("Direct Mapping"))
    {
       blockLineData = intDeiktisDed;

       if(antisDeiktisEtiketaDedomenon.get(intDeiktisDed) != null)
       {
          if(antisDeiktisEtiketaDedomenon.get(intDeiktisDed) == intEtiketaDed)
          {
             hitsDedomenon = hitsDedomenon + 1;
             listKMDedomenon.get(intDeiktisDed).access = true;

             antisDeiktisEtiketaDedomenon.put(intDeiktisDed,intEtiketaDed);
             antisDeiktisEtiketaDedomenon2.put(intDeiktisDed,mTimi);
          }
          else
          {
             antisDeiktisEtiketaDedomenon.put(intDeiktisDed,intEtiketaDed);
             antisDeiktisEtiketaDedomenon2.put(intDeiktisDed,mTimi);
          }
       }
       else
       {
          antisDeiktisEtiketaDedomenon.put(intDeiktisDed,intEtiketaDed);
          antisDeiktisEtiketaDedomenon2.put(intDeiktisDed,mTimi);
       }
       for (int i : antisDeiktisEtiketaDedomenon.keySet())
       {
          listKMDedomenon.get(i).olokEntoli = antisDeiktisEtiketaDedomenon2.get(i);
          listKMDedomenon.get(i).etiketa = Integer.toString(antisDeiktisEtiketaDedomenon.get(i));

          mKMDedomenonAdapter.setList(listKMDedomenon,blockLineData);
          mKMDedomenonAdapter.notifyDataSetChanged();
       }
    }
    else if(cacheMappingDed.equals("Associative"))
    {
       boolean hitTick = false;
       int keniThesi = -1;
       for(int j=0;j<blocksSinolouCacheDed;j++)
       {
          if(sisxetikiCacheDedomenon[intDeiktisDed][j]!=null)
          {
             if (Integer.parseInt(sisxetikiCacheDedomenon[intDeiktisDed][j]) == intEtiketaDed)
             {
                blockLineData = (intDeiktisDed * blocksSinolouCacheDed) + j;
```

```
            hitsDedomenon = hitsDedomenon + 1;
            listKMDedomenon.get(intDeiktisDed).access = true;
            hitTick = true;
            if (writingPolicyEnt.equals("LRU"))
            {
               if (LRUdeiktisDed[intDeiktisDed] < blocksSinolouCacheDed)
               {
                  LRUpinDed[intDeiktisDed][j] = LRUdeiktisDed[intDeiktisDed];

                  LRUdeiktisDed[intDeiktisDed] = LRUdeiktisDed[intDeiktisDed] + 1;
               } else {
                  for (int k = 0; k < blocksSinolouCacheDed; k++) {
                     LRUpinDed[intDeiktisDed][k] = LRUpinDed[intDeiktisDed][k] - 1;
                  }
                  LRUpinDed[intDeiktisDed][j] = LRUdeiktisDed[intDeiktisDed];
               }
            }
         }
      }

      if(sisxetikiCacheDedomenon[intDeiktisDed][j] == null
            || sisxetikiCacheDedomenon[intDeiktisDed][j].length() == 0)
      {
         if (keniThesi < 0)
         {
            keniThesi = j;
         }
      }
   }

   if(!hitTick && keniThesi>=0)
   {
      blockLineData = (intDeiktisDed*blocksSinolouCacheDed)+keniThesi;

      sisxetikiCacheDedomenon[intDeiktisDed][keniThesi] = Integer.toString(intEtiketaDed);
      sisxetikiCacheDedomenon2[intDeiktisDed][keniThesi] = mTimi;

      if(writingPolicyEnt.equals("LRU"))
      {
         if(LRUdeiktisDed[intDeiktisDed]<blocksSinolouCacheDed)
         {
            LRUpinDed[intDeiktisDed][keniThesi] = LRUdeiktisDed[intDeiktisDed];

            LRUdeiktisDed[intDeiktisDed] = LRUdeiktisDed[intDeiktisDed] + 1;
         }
         else
         {
            for(int k=0;k<blocksSinolouCacheDed;k++)
            {
               LRUpinDed[intDeiktisDed][k] = LRUpinDed[intDeiktisDed][k] - 1;
            }
            LRUpinDed[intDeiktisDed][keniThesi] = LRUdeiktisDed[intDeiktisDed];
         }
      }
      else if(writingPolicyDed.equals("FIFO"))
      {
```

```
            if(FIFOdeiktisDed[intDeiktisDed]<blocksSinolouCacheDed)
            {
               FIFOpinDed[intDeiktisDed][keniThesi] = FIFOdeiktisDed[intDeiktisDed];

               FIFOdeiktisDed[intDeiktisDed] = FIFOdeiktisDed[intDeiktisDed] + 1;
            }
            else
            {
               for(int k=0;k<blocksSinolouCacheDed;k++)
               {
                  FIFOpinDed[intDeiktisDed][k] = FIFOpinDed[intDeiktisDed][k] - 1;
               }
               FIFOpinDed[intDeiktisDed][keniThesi] = FIFOdeiktisDed[intDeiktisDed];
            }
         }
      }
      if(!hitTick && keniThesi<0)
      {
         int thesi = 0;
         if(writingPolicyDed.equals("random"))
         {
            thesi = (int)(Math.random()*((blocksSinolouCacheEnt-1)-0+1)+0);
            sisxetikiCacheDedomenon[intDeiktisDed][thesi] = Integer.toString(intEtiketaDed);
            sisxetikiCacheDedomenon2[intDeiktisDed][thesi] = mTimi;
         }
         else if(writingPolicyDed.equals("LRU"))
         {
            thesi = 0;
            for(int j=1;j<blocksSinolouCacheDed;j++)
            {
               if(LRUpinDed[intDeiktisDed][thesi]<LRUpinDed[intDeiktisDed][j])
               {
                  thesi = thesi;
               }
               else
               {
                  thesi = j;
               }
            }
            sisxetikiCacheDedomenon[intDeiktisDed][thesi] = Integer.toString(intEtiketaDed);
            sisxetikiCacheDedomenon2[intDeiktisDed][thesi] = mTimi;
            if(LRUdeiktisDed[intDeiktisDed]<blocksSinolouCacheDed)
            {
               LRUpinDed[intDeiktisDed][thesi] = LRUdeiktisDed[intDeiktisDed];

               LRUdeiktisDed[intDeiktisDed] = LRUdeiktisDed[intDeiktisDed] + 1;
            }
            else
            {
               for(int k=0;k<blocksSinolouCacheDed;k++)
               {
                  LRUpinDed[intDeiktisDed][k] = LRUpinDed[intDeiktisDed][k] - 1;
               }
               LRUpinDed[intDeiktisDed][thesi] = LRUdeiktisDed[intDeiktisDed];
            }
         }
```

```
            else if(writingPolicyDed.equals("FIFO"))
            {
               thesi = 0;
               for(int j=1;j<blocksSinolouCacheDed;j++)
               {
                  if(FIFOpinDed[intDeiktisDed][thesi]<FIFOpinDed[intDeiktisDed][j])
                  {
                     thesi = thesi;
                  }
                  else
                  {
                     thesi = j;
                  }
               }
               sisxetikiCacheDedomenon[intDeiktisDed][thesi] = Integer.toString(intEtiketaDed);
               sisxetikiCacheDedomenon2[intDeiktisDed][thesi] = mTimi;
               if(FIFOdeiktisDed[intDeiktisDed]<blocksSinolouCacheDed)
               {
                  FIFOpinDed[intDeiktisDed][thesi] = FIFOdeiktisDed[intDeiktisDed];

                  FIFOdeiktisDed[intDeiktisDed] = FIFOdeiktisDed[intDeiktisDed] + 1;
               }
               else
               {
                  for(int k=0;k<blocksSinolouCacheDed;k++)
                  {
                     FIFOpinDed[intDeiktisDed][k] = FIFOpinDed[intDeiktisDed][k] - 1;
                  }
                  FIFOpinDed[intDeiktisDed][thesi] = FIFOdeiktisDed[intDeiktisDed];
               }
            }

            blockLineData = (intDeiktisDed*blocksSinolouCacheDed)+thesi;
         }
         for(int i=0;i<blocksSinolouCacheDed;i++)
         {
            listKMDedomenon.get((intDeiktisDed*blocksSinolouCacheDed)+i).olokEntoli =
sisxetikiCacheDedomenon2[intDeiktisDed][i];
            listKMDedomenon.get((intDeiktisDed*blocksSinolouCacheDed)+i).etiketa =
sisxetikiCacheDedomenon[intDeiktisDed][i];

            mKMDedomenonAdapter.setList(listKMDedomenon,blockLineData);
            mKMDedomenonAdapter.notifyDataSetChanged();
         }
      }
      else if(cacheMappingDed.equals("Full Associative"))
      {
         boolean hitTick = false;
         int keniThesi = -1;
         for(int j=0;j<arithmosBlocksDed;j++)
         {
            if(plirosSisxetikiCacheDedomenon[j]!=null)
            {
               if (Integer.parseInt(plirosSisxetikiCacheDedomenon[j]) == intEtiketaEnt)
               {
                  blockLineData = j;
```

```
            hitsDedomenon = hitsDedomenon + 1;
            listKMDedomenon.get(j).access = true;
            hitTick = true;
            if (writingPolicyEnt.equals("LRU"))
            {
               if (LRUdeiktisDedPS < arithmosBlocksEnt)
               {
                  LRUpinDedPS[j] = LRUdeiktisDedPS;

                  LRUdeiktisDedPS = LRUdeiktisDedPS + 1;
               }
               else
               {
                  for (int k = 0; k < arithmosBlocksDed; k++)
                  {
                     LRUpinDedPS[k] = LRUpinDedPS[k] - 1;
                  }
                  LRUpinDedPS[j] = LRUdeiktisDedPS;
               }
            }
         }
      }
      if (plirosSisxetikiCacheDedomenon[j] == null
            || plirosSisxetikiCacheDedomenon[j].length() == 0)
      {
         if (keniThesi < 0)
         {
            keniThesi = j;
         }
      }
   }

   if(!hitTick && keniThesi>=0)
   {
      blockLineData = keniThesi;

      plirosSisxetikiCacheDedomenon[keniThesi] = Integer.toString(intEtiketaEnt);
      plirosSisxetikiCacheDedomenon2[keniThesi] = mTimi;
      antisEtiketaDedomenon.put(intEtiketaDed,intEtiketaDed);
      antisEtiketaDedomenon2.put(intEtiketaDed,mTimi);

      if(writingPolicyDed.equals("LRU"))
      {
         if(LRUdeiktisDedPS < arithmosBlocksDed)
         {
            LRUpinDedPS[keniThesi] = LRUdeiktisDedPS;

            LRUdeiktisDedPS = LRUdeiktisDedPS + 1;
         }
         else
         {
            for(int k=0;k<arithmosBlocksDed;k++)
            {
               LRUpinDedPS[k] = LRUpinDedPS[k] - 1;
            }
```

```
                LRUpinDedPS[keniThesi] = LRUdeiktisDedPS;
            }
        }
        else if(writingPolicyDed.equals("FIFO"))
        {
            if(FIFOdeiktisDedPS < arithmosBlocksDed)
            {
                FIFOpinDedPS[keniThesi] = FIFOdeiktisDedPS;

                FIFOdeiktisDedPS = FIFOdeiktisDedPS + 1;
            }
            else
            {
                for(int k=0;k<arithmosBlocksDed;k++)
                {
                    FIFOpinDedPS[k] = FIFOpinDedPS[k] - 1;
                }
                FIFOpinDedPS[keniThesi] = FIFOdeiktisDedPS;
            }
        }
    }
    if(!hitTick && keniThesi<0)
    {
        antisEtiketaDedomenon.put(intEtiketaDed,intEtiketaDed);
        antisEtiketaDedomenon2.put(intEtiketaDed,mTimi);

        int thesi = 0;
        if(writingPolicyDed.equals("random"))
        {
            thesi = (int)(Math.random()*((arithmosBlocksDed-1)-0+1)+0);

            plirosSisxetikiCacheDedomenon[thesi] = Integer.toString(intEtiketaDed);
            plirosSisxetikiCacheDedomenon2[thesi] = mTimi;
        }
        else if(writingPolicyDed.equals("LRU"))
        {
            thesi = 0;
            for(int j=1;j<arithmosBlocksDed;j++)
            {
                if(LRUpinDedPS[thesi]<LRUpinDedPS[j])
                {
                    thesi = thesi;
                }
                else
                {
                    thesi = j;
                }
            }
            plirosSisxetikiCacheDedomenon[thesi] = Integer.toString(intEtiketaDed);
            plirosSisxetikiCacheDedomenon2[thesi] = mTimi;

            if(LRUdeiktisDedPS < arithmosBlocksDed)
            {
                LRUpinDedPS[thesi] = LRUdeiktisDedPS;

                LRUdeiktisDedPS = LRUdeiktisDedPS + 1;
```

```
            }
            else
            {
               for(int k=0;k<arithmosBlocksDed;k++)
               {
                  LRUpinDedPS[k] = LRUpinDedPS[k] - 1;
               }
               LRUpinDedPS[thesi] = LRUdeiktisDedPS;
            }
         }
         else if(writingPolicyDed.equals("FIFO"))
         {
            thesi = 0;
            for(int j=1;j<arithmosBlocksDed;j++)
            {
               if(FIFOpinDedPS[thesi]<FIFOpinDedPS[j])
               {
                  thesi = thesi;
               }
               else
               {
                  thesi = j;
               }
            }
            plirosSisxetikiCacheDedomenon[thesi] = Integer.toString(intEtiketaDed);
            plirosSisxetikiCacheDedomenon2[thesi] = mTimi;

            if(FIFOdeiktisDedPS < arithmosBlocksDed)
            {
               FIFOpinDedPS[thesi] = FIFOdeiktisDedPS;

               FIFOdeiktisDedPS = FIFOdeiktisDedPS + 1;
            }
            else
            {
               for(int k=0;k<arithmosBlocksDed;k++)
               {
                  FIFOpinDedPS[k] = FIFOpinDedPS[k] - 1;
               }
               FIFOpinDedPS[thesi] = FIFOdeiktisDedPS;
            }
         }
         blockLineData = thesi;
      }
      for (int i=0;i<arithmosBlocksDed;i++)
      {
         listKMDedomenon.get(i).olokEntoli = plirosSisxetikiCacheDedomenon2[i];
         listKMDedomenon.get(i).etiketa = plirosSisxetikiCacheDedomenon[i];

         mKMDedomenonAdapter.setList(listKMDedomenon,blockLineData);
         mKMDedomenonAdapter.notifyDataSetChanged();
      }
   }

   dieutDedomenon.setText("Data Address: " + dieuthDed);
   if(cacheMappingDed.equals("Direct Mapping") || cacheMappingDed.equals("Associative"))
```

```
        {
            intDeiktisDedomenon.setText("Data Index: "+intDeiktisDed);
            intEtiketaDedomenon.setText("Data Tag: "+intEtiketaDed);
        }
        else if(cacheMappingDed.equals("Full Associative"))
        {
            intEtiketaDedomenon.setText("Data Tag: "+intEtiketaDed);
        }
        accessKMDedomenon.setText("Access: " + (int)accessCacheDedomenon);
        hitsKMDedomenon.setText("Hits: " + (int)hitsDedomenon);
        hitRateKMDedomenon.setText("Hit Rate: "+(hitsDedomenon/accessCacheDedomenon));
    }


//=================================================================
==================//

    public class entoli
    {
        String olokEntoli;
        String instructionAddress;
        String leitourgia;
        String katax1;
        String katax2;
        String katax3;
        String jump;
        String label;
        String stathera;
        String stathera2;
    }

    public class BlockKmEntolon
    {
        String olokEntoli;
        String number;
        String etiketa;
        Boolean access = false;
    }
    public class BlockKmDedomenon
    {
        String olokEntoli;
        String number;
        String etiketa;
        Boolean access = false;
    }


//=================================================================
==================//

    private void readText()
    {

//=================================================================
===============//
        Scanner myReader = null;
```

```
try
{
   FileInputStream fileInputStream = openFileInput("codefile.txt");
   myReader = new Scanner(fileInputStream);
   String[] arrOfStr;
   String[] arrOfStr2;
   String[] arrOfStr3;
   String[] arrOfStr4;
   String[] arrOfStr5;
   boolean dataput = false;
   int nextAddress = 0;

   arxikopoihsiOtanEkteleitaiISinartisiRead();

   while (myReader.hasNextLine())
   {
      MainActivity.entoli ent = new MainActivity.entoli();

      String data = myReader.nextLine();
      ent.olokEntoli = data;

      arrOfStr = data.split(" ", 2);
      ent.leitourgia = arrOfStr[0].trim();
      arrOfStr2 = arrOfStr[arrOfStr.length-1].trim().split(",",0);

      if(ent.leitourgia.equals("STOREI") || ent.leitourgia.equals("STORED"))
      {
         ent.stathera2 = arrOfStr2[0].trim();
      }
      else
      {
         ent.katax1 = arrOfStr2[0].trim();
      }
      if (arrOfStr2[arrOfStr2.length-1].contains("("))
      {
         arrOfStr3 = arrOfStr2[arrOfStr2.length-1].trim().split("\\(",2);

         for(int i = 0; i < arrOfStr3.length; i++)
         {
            if(i==0)
            {
               if (arrOfStr3[i].trim().contains(":"))
               {
                  arrOfStr4 = arrOfStr3[i].trim().split(":",2);
                  ent.stathera = arrOfStr4[0].trim();
                  ent.label = arrOfStr4[1].trim();
               }
               else
               {
                  ent.stathera = arrOfStr3[i].trim();
               }
            }
            else if(i==1)
            {
               ent.katax2 = arrOfStr3[i].replace(')', ' ').trim();
```

```
            if (ent.katax2.contains(":"))
            {
               arrOfStr4 = ent.katax2.trim().split(":",2);
               ent.katax2 = arrOfStr4[0].trim();
               ent.label = arrOfStr4[1].trim();
            }
         }
      }
   }
   else if(arrOfStr2.length<1)
   {
      if(ent.leitourgia.equals("J") || ent.leitourgia.equals("JAL"))
      {
         ent.jump = arrOfStr2[arrOfStr2.length-1].trim();

         if (ent.jump.contains(":"))
         {
            arrOfStr4 = ent.jump.trim().split(":",2);
            ent.jump = arrOfStr4[0].trim();
            ent.label = arrOfStr4[1].trim();
         }
      }
      else if(ent.leitourgia.equals("JR") || ent.leitourgia.equals("JALR"))
      {
         ent.katax1 = arrOfStr2[arrOfStr2.length-1].trim();

         if (ent.katax1.contains(":"))
         {
            arrOfStr4 = ent.jump.trim().split(":",2);
            ent.katax1 = arrOfStr4[0].trim();
            ent.label = arrOfStr4[1].trim();
         }
      }
   }
   else
   {
      for(int i = 0; i < arrOfStr2.length; i++)
      {
         if(i==1)
         {
            if(ent.leitourgia.equals("BEQZ") || ent.leitourgia.equals("BNEZ"))
            {
               ent.jump = arrOfStr2[i].trim();

               if (ent.jump.contains(":"))
               {
                  arrOfStr4 = ent.jump.trim().split(":",2);
                  ent.jump = arrOfStr4[0].trim();
                  ent.label = arrOfStr4[1].trim();
               }
            }
            else if(ent.leitourgia.equals("LOAD")
                  || ent.leitourgia.equals("LOADD") || ent.leitourgia.equals("LOADI")
                  || ent.leitourgia.equals("STORE") || ent.leitourgia.equals("LUI"))
            {
               if (arrOfStr2[i].trim().contains(":"))
```

```
            {
                arrOfStr4 = arrOfStr2[i].trim().split(":",2);
                ent.stathera = arrOfStr4[0].trim();
                ent.label = arrOfStr4[1].trim();
            }
            else
            {
                ent.stathera = arrOfStr2[i].trim();
            }
        }
        else
        {
            ent.katax2 = arrOfStr2[i].trim();

            if (ent.katax2.contains(":"))
            {
                arrOfStr4 = ent.katax2.trim().split(":",2);
                ent.katax2 = arrOfStr4[0].trim();
                ent.label = arrOfStr4[1].trim();
            }
        }
    }
    else if(i==2)
    {
        if(ent.leitourgia.equals("BEQ") || ent.leitourgia.equals("BNE"))
        {
            ent.jump = arrOfStr2[i].trim();

            if (ent.jump.contains(":"))
            {
                arrOfStr4 = ent.jump.trim().split(":",2);
                ent.jump = arrOfStr4[0].trim();
                ent.label = arrOfStr4[1].trim();
            }
        }
        else if(ent.leitourgia.equals("SLIDEL") || ent.leitourgia.equals("SLIDER")
                || ent.leitourgia.equals("ORI") || ent.leitourgia.equals("ANDI")
                || ent.leitourgia.equals("XORI")
                || ent.leitourgia.equals("ADDI") || ent.leitourgia.equals("ADDD")
                || ent.leitourgia.equals("SUBI") || ent.leitourgia.equals("SUBD")
                || ent.leitourgia.equals("DIVI") || ent.leitourgia.equals("DIVD")
                || ent.leitourgia.equals("MULI") || ent.leitourgia.equals("MULD"))
        {
            if (arrOfStr2[i].contains(":"))
            {
                arrOfStr4 = arrOfStr2[i].trim().split(":",2);
                ent.stathera = arrOfStr4[0].trim();
                ent.label = arrOfStr4[1].trim();
            }
            else
            {
                ent.stathera = arrOfStr2[i].trim();
            }
        }
        else
        {
```

```
                ent.katax3 = arrOfStr2[i].trim();

                if (ent.katax3.contains(":"))
                {
                   arrOfStr4 = ent.katax3.trim().split(":",2);
                   ent.katax3 = arrOfStr4[0].trim();
                   ent.label = arrOfStr4[1].trim();
                }
             }
          }
       }
    }

    if (arrOfStr[0].startsWith("main:"))
    {
       dataput = false;
    }
    if(dataput)
    {
       arrOfStr5 = data.trim().split("=",2);

       ent.leitourgia = "LOADA";
       ent.katax1 = arrOfStr5[0].trim();
       ent.stathera = arrOfStr5[arrOfStr5.length-1].trim();

       antisKatTimi.put(ent.katax1,ent.stathera);
       antisDieutTimi.put(dataAddress,ent.stathera);
       antisKatDieut.put(ent.katax1,Integer.toString(dataAddress));

       if(ent.stathera.contains("."))
       {
          doubleYes = true;
       }

       dieutDedString = "";
       for(int j : antisDieutTimi.keySet())
       {
          dieutDedString = dieutDedString + j + "  " + antisDieutTimi.get(j) + "\n";
       }
       dedomenaStinMnimi.setText(dieutDedString);
       metavlitesDieuthinsis = "";
       for(String j : antisKatDieut.keySet())
       {
          metavlitesDieuthinsis = metavlitesDieuthinsis + j + " = " + antisKatDieut.get(j) + "\n";
       }
       metavlitesDieuthinsisTxtView.setText(metavlitesDieuthinsis);

       dataAddress = dataAddress + 4;
    }

    if (arrOfStr[0].startsWith("data:"))
    {
       dataput = true;
    }

    if(!ent.leitourgia.equals("data:") && !ent.leitourgia.equals("main:") && !dataput)
```

```
            {
                ent.instructionAddress = Integer.toString(instructionAddress + (nextAddress*4));
                nextAddress = nextAddress + 1;
                listEntolon.add(ent);
            }
        }
        myReader.close();
        mEntolesAdapter.setList(listEntolon,line);
        mEntolesAdapter.notifyDataSetChanged();
    }
    catch (FileNotFoundException e)
    {
        System.out.println("An error occurred.");
        e.printStackTrace();
    }
}


//=================================================================================//

private void arxikopoihsiOtanEkteleitaiISinartisiRead()
{
    for(int i=0;i<64;i++)
    {
        if(i<32)
        {
            listKataxoriton.add("R"+i);
        }
        else
        {
            listKataxoriton.add("F"+(i-32));
        }
    }
    mKataxoritesAdapter.setList(listKataxoriton);
    mKataxoritesAdapter.notifyDataSetChanged();


//=================================================================================//

    if(cacheMappingEnt.equals("Direct Mapping"))
    {
        for(int i=0;i<arithmosBlocksEnt;i++)
        {
            MainActivity.BlockKmEntolon block = new MainActivity.BlockKmEntolon();
            block.number = i+": ";
            listKMEntolon.add(block);
        }
    }
    else if(cacheMappingEnt.equals("Associative"))
    {
        for(int i=0;i<sinolaCacheEnt;i++)
        {
            for(int j=0;j<blocksSinolouCacheEnt;j++)
            {
```

```
                MainActivity.BlockKmEntolon block = new MainActivity.BlockKmEntolon();
                block.number = "Set:"+i+",num:"+j+":";
                listKMEntolon.add(block);
            }
        }

        instructionNumTextView.setText("Sets");
    }
    else if(cacheMappingEnt.equals("Full Associative"))
    {
        for(int i=0;i<arithmosBlocksEnt;i++)
        {
            MainActivity.BlockKmEntolon block = new MainActivity.BlockKmEntolon();
            block.number = i+": ";
            listKMEntolon.add(block);
        }
    }
    mKMEntolonAdapter.setList(listKMEntolon,blockLineInstruction);
    mKMEntolonAdapter.notifyDataSetChanged();


//=================================================================================//

    if(cacheMappingDed.equals("Direct Mapping"))
    {
        for(int i=0;i<arithmosBlocksDed;i++)
        {
            MainActivity.BlockKmDedomenon block = new MainActivity.BlockKmDedomenon();
            block.number = i+": ";
            listKMDedomenon.add(block);
        }
    }
    else if(cacheMappingDed.equals("Associative"))
    {
        for(int i=0;i<sinolaCacheDed;i++)
        {
            for(int j=0;j<blocksSinolouCacheDed;j++)
            {
                MainActivity.BlockKmDedomenon block = new MainActivity.BlockKmDedomenon();
                block.number = "Set:"+i+",num:"+j+":";
                listKMDedomenon.add(block);
            }
        }

        dataNumTextView.setText("Sets");
    }
    else if(cacheMappingDed.equals("Full Associative"))
    {
        for(int i=0;i<arithmosBlocksDed;i++)
        {
            MainActivity.BlockKmDedomenon block = new MainActivity.BlockKmDedomenon();
            block.number = i+": ";
            listKMDedomenon.add(block);
        }
    }
```

```
        mKMDedomenonAdapter.setList(listKMDedomenon,blockLineData);
        mKMDedomenonAdapter.notifyDataSetChanged();
    }


//=====================================================================
====================//
    HashMap<String,String> antisKatTimi = new HashMap<String,String>();
    HashMap<String,String> antisKatDieut = new HashMap<String,String>();
    HashMap<Integer,String> antisDieutTimi = new HashMap<Integer,String>();
    int dataAddress=10000;
    int startingDataAddress=10000;
    int instructionAddress = 1000;

    String olokEntoli;
    String leitourgia;
    String katax1;
    String katax2;
    String katax3;
    String jump;
    String label;
    String stathera;
    String stathera2;
    boolean doubleYes;

    int dieuthinsiDedomenon;
    int timi1;
    int timi2;
    double timi1D;
    double timi2D;

    public int runCode(int i)
    {
        int timiEpistrofis = -1;

        olokEntoli = listEntolon.get(i).olokEntoli;
        leitourgia = listEntolon.get(i).leitourgia;
        katax1 = listEntolon.get(i).katax1;
        katax2 = listEntolon.get(i).katax2;
        katax3 = listEntolon.get(i).katax3;
        stathera = listEntolon.get(i).stathera;
        stathera2 = listEntolon.get(i).stathera2;
        jump = listEntolon.get(i).jump;
        label = listEntolon.get(i).label;

        doubleYes = false;

        switch(leitourgia)
        {
            case "LOAD":
                if(katax2 == null)
                {
                    dieuthinsiDedomenon = Integer.parseInt(stathera);
                }
                else
                {
```

```
                dieuthinsiDedomenon = Integer.parseInt(stathera) +
Integer.parseInt(antisKatDieut.get(katax2));
            }
            antisKatTimi.put(katax1,antisDieutTimi.get(dieuthinsiDedomenon));

            if(antisKatTimi.get(katax1).contains("."))
            {
                doubleYes = true;
            }
            writeRegister(katax1,antisKatTimi.get(katax1),doubleYes);

            cacheEntolon(instructionAddress + (i*4),olokEntoli);
            cacheDedomenon(dieuthinsiDedomenon,antisDieutTimi.get(dieuthinsiDedomenon));

            return -1;
        case "LOADI":
            antisKatTimi.put(katax1,stathera);

            if(stathera.contains("."))
            {
                doubleYes = true;
            }
            writeRegister(katax1,antisKatTimi.get(katax1),doubleYes);

            cacheEntolon(instructionAddress + (i*4),olokEntoli);

            return -1;
        case "STORE":
            if(katax2 == null)
            {
                dieuthinsiDedomenon = Integer.parseInt(stathera);
            }
            else
            {
                dieuthinsiDedomenon = Integer.parseInt(stathera) +
Integer.parseInt(antisKatDieut.get(katax2));
            }
            antisDieutTimi.put(dieuthinsiDedomenon,antisKatTimi.get(katax1));

            dieutDedString = "";
            for(int j : antisDieutTimi.keySet())
            {
                dieutDedString = dieutDedString+j+"  "+antisDieutTimi.get(j)+"\n";
            }
            dedomenaStinMnimi.setText(dieutDedString);

            cacheEntolon(instructionAddress + (i*4),olokEntoli);
            cacheDedomenon(dieuthinsiDedomenon,antisDieutTimi.get(dieuthinsiDedomenon));

            return -1;
        case "STOREI":
            if(katax2 == null)
            {
                dieuthinsiDedomenon = Integer.parseInt(stathera);
            }
            else
```

```
            {
               dieuthinsiDedomenon = Integer.parseInt(stathera) +
Integer.parseInt(antisKatDieut.get(katax2));
            }
            antisDieutTimi.put(dieuthinsiDedomenon,stathera2);

            dieutDedString = "";
            for(int j : antisDieutTimi.keySet())
            {
               dieutDedString = dieutDedString+j+"  "+antisDieutTimi.get(j)+"\n";
            }
            dedomenaStinMnimi.setText(dieutDedString);

            cacheEntolon(instructionAddress + (i*4),olokEntoli);
            cacheDedomenon(dieuthinsiDedomenon,antisDieutTimi.get(dieuthinsiDedomenon));

            return -1;
         case "MOVE":
            antisKatTimi.put(katax1,antisKatTimi.get(katax2));

            writeRegister(katax1,antisKatTimi.get(katax1),doubleYes);

            cacheEntolon(instructionAddress + (i*4),olokEntoli);

            return -1;
         case "MOVEZ":
            if(Integer.parseInt(antisKatTimi.get(katax3)) == 0)
            {
               antisKatTimi.put(katax1,antisKatTimi.get(katax2));
            }

            writeRegister(katax1,antisKatTimi.get(katax1),doubleYes);

            cacheEntolon(instructionAddress + (i*4),olokEntoli);

            return -1;
         case "MOVEN":
            if(Integer.parseInt(antisKatTimi.get(katax3)) < 0)
            {
               antisKatTimi.put(katax1,antisKatTimi.get(katax2));
            }

            writeRegister(katax1,antisKatTimi.get(katax1),doubleYes);

            cacheEntolon(instructionAddress + (i*4),olokEntoli);

            return -1;
         case "ADD":
            if(antisKatTimi.get(katax2).contains(".") || antisKatTimi.get(katax3).contains("."))
            {
               timi1D = Double.parseDouble(antisKatTimi.get(katax2));
               timi2D = Double.parseDouble(antisKatTimi.get(katax3));
               doubleYes = true;
            }
            else
            {
```

```
            timi1 = Integer.parseInt(antisKatTimi.get(katax2));
            timi2 = Integer.parseInt(antisKatTimi.get(katax3));
        }
        if(doubleYes)
        {
            antisKatTimi.put(katax1,Double.toString(timi1D + timi2D));
        }
        else
        {
            antisKatTimi.put(katax1,Integer.toString(timi1 + timi2));
        }

        writeRegister(katax1,antisKatTimi.get(katax1),doubleYes);

        cacheEntolon(instructionAddress + (i*4),olokEntoli);

        return -1;
    case "ADDI":
        if(stathera.contains(".") || antisKatTimi.get(katax2).contains("."))
        {
            timi1D = Double.parseDouble(antisKatTimi.get(katax2));
            doubleYes = true;
        }
        else
        {
            timi1 = Integer.parseInt(antisKatTimi.get(katax2));
        }
        if(doubleYes)
        {
            antisKatTimi.put(katax1,Double.toString(timi1D + Double.parseDouble(stathera)));
        }
        else
        {
            antisKatTimi.put(katax1,Integer.toString(timi1 + Integer.parseInt(stathera)));
        }

        writeRegister(katax1,antisKatTimi.get(katax1),doubleYes);

        cacheEntolon(instructionAddress + (i*4),olokEntoli);

        return -1;
    case "SUB":
        if(antisKatTimi.get(katax2).contains(".") || antisKatTimi.get(katax3).contains("."))
        {
            timi1D = Double.parseDouble(antisKatTimi.get(katax2));
            timi2D = Double.parseDouble(antisKatTimi.get(katax3));
            doubleYes = true;
        }
        else
        {
            timi1 = Integer.parseInt(antisKatTimi.get(katax2));
            timi2 = Integer.parseInt(antisKatTimi.get(katax3));
        }
        if(doubleYes)
        {
            antisKatTimi.put(katax1,Double.toString(timi1D - timi2D));
```

```
        }
        else
        {
            antisKatTimi.put(katax1,Integer.toString(timi1 - timi2));
        }

        writeRegister(katax1,antisKatTimi.get(katax1),doubleYes);

        cacheEntolon(instructionAddress + (i*4),olokEntoli);

        return -1;
    case "SUBI":
        if(stathera.contains(".") || antisKatTimi.get(katax2).contains("."))
        {
            timi1D = Double.parseDouble(antisKatTimi.get(katax2));
            doubleYes = true;
        }
        else
        {
            timi1 = Integer.parseInt(antisKatTimi.get(katax2));
        }
        if(doubleYes)
        {
            antisKatTimi.put(katax1,Double.toString(timi1D - Double.parseDouble(stathera)));
        }
        else
        {
            antisKatTimi.put(katax1,Integer.toString(timi1 - Integer.parseInt(stathera)));
        }

        writeRegister(katax1,antisKatTimi.get(katax1),doubleYes);

        cacheEntolon(instructionAddress + (i*4),olokEntoli);

        return -1;
    case "MUL":
        if(antisKatTimi.get(katax2).contains(".") || antisKatTimi.get(katax3).contains("."))
        {
            timi1D = Double.parseDouble(antisKatTimi.get(katax2));
            timi2D = Double.parseDouble(antisKatTimi.get(katax3));
            doubleYes = true;
        }
        else
        {
            timi1 = Integer.parseInt(antisKatTimi.get(katax2));
            timi2 = Integer.parseInt(antisKatTimi.get(katax3));
        }
        if(doubleYes)
        {
            antisKatTimi.put(katax1,Double.toString(timi1D * timi2D));
        }
        else
        {
            antisKatTimi.put(katax1,Integer.toString(timi1 * timi2));
        }
```

```
            writeRegister(katax1,antisKatTimi.get(katax1),doubleYes);

            cacheEntolon(instructionAddress + (i*4),olokEntoli);

            return -1;
        case "MULI":
            if(stathera.contains(".") || antisKatTimi.get(katax2).contains("."))
            {
                timi1D = Double.parseDouble(antisKatTimi.get(katax2));
                doubleYes = true;
            }
            else
            {
                timi1 = Integer.parseInt(antisKatTimi.get(katax2));
            }
            if(doubleYes)
            {
                antisKatTimi.put(katax1,Double.toString(timi1D * Double.parseDouble(stathera)));
            }
            else
            {
                antisKatTimi.put(katax1,Integer.toString(timi1 * Integer.parseInt(stathera)));
            }

            writeRegister(katax1,antisKatTimi.get(katax1),doubleYes);

            cacheEntolon(instructionAddress + (i*4),olokEntoli);

            return -1;
        case "DIV":
            if(antisKatTimi.get(katax2).contains(".") || antisKatTimi.get(katax3).contains("."))
            {
                timi1D = Double.parseDouble(antisKatTimi.get(katax2));
                timi2D = Double.parseDouble(antisKatTimi.get(katax3));
                doubleYes = true;
            }
            else
            {
                timi1 = Integer.parseInt(antisKatTimi.get(katax2));
                timi2 = Integer.parseInt(antisKatTimi.get(katax3));
            }
            if(doubleYes)
            {
                antisKatTimi.put(katax1,Double.toString(timi1D / timi2D));
            }
            else
            {
                antisKatTimi.put(katax1,Integer.toString(timi1 / timi2));
            }

            writeRegister(katax1,antisKatTimi.get(katax1),doubleYes);

            cacheEntolon(instructionAddress + (i*4),olokEntoli);

            return -1;
        case "DIVI":
```

```
            if(stathera.contains(".") || antisKatTimi.get(katax2).contains("."))
            {
               timi1D = Double.parseDouble(antisKatTimi.get(katax2));
               doubleYes = true;
            }
            else
            {
               timi1 = Integer.parseInt(antisKatTimi.get(katax2));
            }
            if(doubleYes)
            {
               antisKatTimi.put(katax1,Double.toString(timi1D / Double.parseDouble(stathera)));
            }
            else
            {
               antisKatTimi.put(katax1,Integer.toString(timi1 / Integer.parseInt(stathera)));
            }

            writeRegister(katax1,antisKatTimi.get(katax1),doubleYes);

            cacheEntolon(instructionAddress + (i*4),olokEntoli);

            return -1;
         case "AND":
            timi1 = Integer.parseInt(antisKatTimi.get(katax2));
            timi2 = Integer.parseInt(antisKatTimi.get(katax3));
            antisKatTimi.put(katax1,Integer.toString(timi1 & timi2));

            writeRegister(katax1,antisKatTimi.get(katax1),doubleYes);

            cacheEntolon(instructionAddress + (i*4),olokEntoli);

            return -1;
         case "ANDI":
            timi1 = Integer.parseInt(antisKatTimi.get(katax2));
            antisKatTimi.put(katax1,Integer.toString(timi1 & Integer.parseInt(stathera)));

            writeRegister(katax1,antisKatTimi.get(katax1),doubleYes);

            cacheEntolon(instructionAddress + (i*4),olokEntoli);

            return -1;
         case "OR":
            timi1 = Integer.parseInt(antisKatTimi.get(katax2));
            timi2 = Integer.parseInt(antisKatTimi.get(katax3));
            antisKatTimi.put(katax1,Integer.toString(timi1 | timi2));

            writeRegister(katax1,antisKatTimi.get(katax1),doubleYes);

            cacheEntolon(instructionAddress + (i*4),olokEntoli);

            return -1;
         case "ORI":
            timi1 = Integer.parseInt(antisKatTimi.get(katax2));
            antisKatTimi.put(katax1,Integer.toString(timi1 | Integer.parseInt(stathera)));
```

```
            writeRegister(katax1,antisKatTimi.get(katax1),doubleYes);

            cacheEntolon(instructionAddress + (i*4),olokEntoli);

            return -1;
        case "XOR":
            timi1 = Integer.parseInt(antisKatTimi.get(katax2));
            timi2 = Integer.parseInt(antisKatTimi.get(katax3));
            antisKatTimi.put(katax1,Integer.toString(timi1 ^ timi2));

            writeRegister(katax1,antisKatTimi.get(katax1),doubleYes);

            cacheEntolon(instructionAddress + (i*4),olokEntoli);

            return -1;
        case "XORI":
            timi1 = Integer.parseInt(antisKatTimi.get(katax2));
            antisKatTimi.put(katax1,Integer.toString(timi1 ^ Integer.parseInt(stathera)));

            writeRegister(katax1,antisKatTimi.get(katax1),doubleYes);

            cacheEntolon(instructionAddress + (i*4),olokEntoli);

            return -1;
        case "LUI":
            String statheraBin = Integer.toBinaryString(Integer.parseInt(stathera));
            char[] charStatheraBin = new char[statheraBin.length()];
            statheraBin.getChars(0, statheraBin.length(), charStatheraBin, 0);
            char[] epektasiPros = new char[64];
            int j = 0;
            for(int k=0;k<64;k++)
            {
                if(k<32+(16-statheraBin.length()))
                {
                    epektasiPros[k] = 0;
                }
                else if(k>=32+(16-statheraBin.length()))
                {
                    epektasiPros[k] = charStatheraBin[j];
                    j = j + 1;
                }
                else if(k>48)
                {
                    epektasiPros[k] = 0;
                }
            }
            antisKatTimi.put(katax1,Integer.toString(Integer.parseInt(String.valueOf(epektasiPros),2)));

            writeRegister(katax1,antisKatTimi.get(katax1),doubleYes);

            cacheEntolon(instructionAddress + (i*4),olokEntoli);

            return -1;
        case "SLIDEL":
            timi1 = Integer.parseInt(antisKatTimi.get(katax2));
            antisKatTimi.put(katax1,Integer.toString(timi1 << Integer.parseInt(stathera)));
```

```
            writeRegister(katax1,antisKatTimi.get(katax1),doubleYes);

            cacheEntolon(instructionAddress + (i*4),olokEntoli);

            return -1;
        case "SLIDER":
            timi1 = Integer.parseInt(antisKatTimi.get(katax2));
            antisKatTimi.put(katax1,Integer.toString(timi1 >> Integer.parseInt(stathera)));

            writeRegister(katax1,antisKatTimi.get(katax1),doubleYes);

            cacheEntolon(instructionAddress + (i*4),olokEntoli);

            return -1;
        case "SMALLER":
            timi1 = Integer.parseInt(antisKatTimi.get(katax2));
            timi2 = Integer.parseInt(antisKatTimi.get(katax3));
            if(timi1 < timi2)
            {
                antisKatTimi.put(katax1,"1");
            }
            else
            {
                antisKatTimi.put(katax1,"0");
            }

            writeRegister(katax1,antisKatTimi.get(katax1),doubleYes);

            cacheEntolon(instructionAddress + (i*4),olokEntoli);

            return -1;
        case "BIGGER":
            timi1 = Integer.parseInt(antisKatTimi.get(katax2));
            timi2 = Integer.parseInt(antisKatTimi.get(katax3));
            if(timi1 < timi2)
            {
                antisKatTimi.put(katax1,"0");
            }
            else
            {
                antisKatTimi.put(katax1,"1");
            }

            writeRegister(katax1,antisKatTimi.get(katax1),doubleYes);

            cacheEntolon(instructionAddress + (i*4),olokEntoli);

            return -1;
        case "BEQZ":
            cacheEntolon(instructionAddress + (i*4),olokEntoli);

            timi1 = Integer.parseInt(antisKatTimi.get(katax1));
            if(timi1==0)
            {
                for(int k=0;k<listEntolon.size();k++)
```

```
            {
                if(listEntolon.get(k).label.equals(jump))
                {
                    timiEpistrofis = k-1;
                }
            }
        }

        return timiEpistrofis;
    case "BNEZ":
        cacheEntolon(instructionAddress + (i*4),olokEntoli);

        timi1 = Integer.parseInt(antisKatTimi.get(katax1));
        if(timi1!=0)
        {
            for(int k=0;k<listEntolon.size();k++)
            {
                if(listEntolon.get(k).label != null)
                {
                    if(listEntolon.get(k).label.equals(jump))
                    {
                        timiEpistrofis = k-1;
                    }
                }
            }
        }

        return timiEpistrofis;
    case "BEQ":
        cacheEntolon(instructionAddress + (i*4),olokEntoli);

        timi1 = Integer.parseInt(antisKatTimi.get(katax2));
        timi2 = Integer.parseInt(antisKatTimi.get(katax3));
        if(timi1==timi2)
        {
            for(int k=0;k<listEntolon.size();k++)
            {
                if(listEntolon.get(k).label.equals(jump))
                {
                    timiEpistrofis = k-1;
                }
            }
        }

        return timiEpistrofis;
    case "BNE":
        cacheEntolon(instructionAddress + (i*4),olokEntoli);

        timi1 = Integer.parseInt(antisKatTimi.get(katax2));
        timi2 = Integer.parseInt(antisKatTimi.get(katax3));
        if(timi1!=timi2)
        {
            for(int k=0;k<listEntolon.size();k++)
            {
                if(listEntolon.get(k).label.equals(jump))
                {
```

```
                    timiEpistrofis = k-1;
                }
            }
        }

        return timiEpistrofis;
    case "J":
        cacheEntolon(instructionAddress + (i*4),olokEntoli);

        for(int k=0;k<listEntolon.size();k++)
        {
            if(listEntolon.get(k).label.equals(jump))
            {
                timiEpistrofis = k-1;
            }
        }

        return timiEpistrofis;
    case "JAL":
        cacheEntolon(instructionAddress + (i*4),olokEntoli);
        writeRegister("R31",Integer.toString(instructionAddress + ((i+1)*4)),doubleYes);
        for(int k=0;k<listEntolon.size();k++)
        {
            if(listEntolon.get(k).label.equals(jump))
            {
                timiEpistrofis = k-1;
            }
        }

        return timiEpistrofis;
    case "ENDF":
        timiEpistrofis = ((Integer.parseInt(antisKatTimi.get("R31"))-instructionAddress)/4)-1;

        return timiEpistrofis;
    case "END":
        timiEpistrofis = listEntolon.size();

        return timiEpistrofis;
    default:
        Toast.makeText(getApplicationContext(), "den iparxei auti i entoli: "+olokEntoli,
Toast.LENGTH_SHORT).show();

        return timiEpistrofis = listEntolon.size();
    }
}

public void writeRegister(String katax1,String timi,boolean doubleYes)
{
    String[] kataxList;
    if(doubleYes)
    {
        for (int j=32;j<64;j++)
        {
            if (listKataxoriton.get(j).contains("="))
            {
                kataxList = listKataxoriton.get(j).split("=", 2);
```

```
                if (kataxList[0].equals(katax1))
                {
                    listKataxoriton.set(j, "F" + (j - 32) + "=" + timi);
                }
            }
            else
            {
                if (listKataxoriton.get(j).equals(katax1))
                {
                    listKataxoriton.set(j, "F" + (j - 32) + "=" + timi);
                }
            }
        }
    }
    else
    {
        for (int j=0;j<32;j++)
        {
            if(listKataxoriton.get(j).contains("="))
            {
                kataxList = listKataxoriton.get(j).split("=",2);
                if(kataxList[0].equals(katax1))
                {
                    listKataxoriton.set(j,"R"+j+"="+timi);
                }
            }
            else
            {
                if(listKataxoriton.get(j).equals(katax1))
                {
                    listKataxoriton.set(j,"R"+j+"="+timi);
                }
            }
        }
    }

    mKataxoritesAdapter.setList(listKataxoriton);
    mKataxoritesAdapter.notifyDataSetChanged();
}


//=================================================================
==================//

  private static int line = 0;
  private void clear()
  {
    listEntolon.clear();
    mEntolesAdapter.setList(listEntolon,line);
    mEntolesAdapter.notifyDataSetChanged();

    dieutEntolon.setText("");

//=================================================================
====================//
    dieutDedString = "";
```

```
        dedomenaStinMnimi.setText("");
        metavlitesDieuthinsis = "";
        metavlitesDieuthinsisTxtView.setText(metavlitesDieuthinsis);
        dataAddress = startingDataAddress;
        dieutDedomenon.setText("");

//=============================================================================
=====================//
        listKataxoriton.clear();
        mKataxoritesAdapter.setList(listKataxoriton);
        mKataxoritesAdapter.notifyDataSetChanged();

//=============================================================================
=====================//
        if(cacheMappingEnt.equals("Direct Mapping"))
        {
           antisDeiktisEtiketaEntolon.clear();
           antisDeiktisEtiketaEntolon2.clear();
        }
        else if (cacheMappingEnt.equals("Associative"))
        {
           for(int i=0;i<(int)Math.pow(2,bitsDeiktiEntolon);i++)
           {
              for(int j=0;j<blocksSinolouCacheEnt;j++)
              {
                 sisxetikiCacheEntolon[i][j] = "";
                 sisxetikiCacheEntolon2[i][j] = "";
                 if (writingPolicyEnt.equals("LRU"))
                 {
                    LRUpinEnt[i][j] = 0;
                    LRUdeiktisEnt[i] = 0;
                 }
                 if (writingPolicyEnt.equals("FIFO"))
                 {
                    FIFOpinEnt[i][j] = 0;
                    FIFOdeiktisEnt[i] = 0;
                 }
              }
           }
        }
        else if (cacheMappingEnt.equals("Full Associative"))
        {
           antisEtiketaEntolon.clear();
           antisEtiketaEntolon2.clear();
           for(int i=0;i<arithmosBlocksEnt;i++)
           {
              plirosSisxetikiCacheEntolon[i] = "";
              plirosSisxetikiCacheEntolon2[i] = "";
              if (writingPolicyEnt.equals("LRU"))
              {
                 LRUpinEntPS[i] = 0;
                 LRUdeiktisEntPS = 0;
              }
              if (writingPolicyEnt.equals("FIFO"))
              {
                 FIFOpinEntPS[i] = 0;
```

```
                FIFOdeiktisEntPS = 0;
            }
        }
    }
    if(cacheMappingDed.equals("Direct Mapping"))
    {
        antisDeiktisEtiketaDedomenon.clear();
        antisDeiktisEtiketaDedomenon2.clear();
    }
    else if (cacheMappingDed.equals("Associative"))
    {
        for(int i=0;i<(int)Math.pow(2,bitsDeiktiDedomenon);i++)
        {
            for(int j=0;j<blocksSinolouCacheDed;j++)
            {
                sisxetikiCacheDedomenon[i][j] = "";
                sisxetikiCacheDedomenon2[i][j] = "";
                if (writingPolicyEnt.equals("LRU"))
                {
                    LRUpinDed[i][j] = 0;
                    LRUdeiktisDed[i] = 0;
                }
                if (writingPolicyEnt.equals("FIFO"))
                {
                    FIFOpinDed[i][j] = 0;
                    FIFOdeiktisDed[i] = 0;
                }
            }
        }
    }
    else if (cacheMappingDed.equals("Full Associative"))
    {
        antisEtiketaDedomenon.clear();
        antisEtiketaDedomenon2.clear();
        for(int i=0;i<arithmosBlocksDed;i++)
        {
            plirosSisxetikiCacheDedomenon[i] = "";
            plirosSisxetikiCacheDedomenon2[i] = "";
            if (writingPolicyDed.equals("LRU"))
            {
                LRUpinDedPS[i] = 0;
                LRUdeiktisDedPS = 0;
            }
            if (writingPolicyDed.equals("FIFO"))
            {
                FIFOpinDedPS[i] = 0;
                FIFOdeiktisDedPS = 0;
            }
        }
    }
    listKMEntolon.clear();
    mKMEntolonAdapter.setList(listKMEntolon,blockLineInstruction);
    mKMEntolonAdapter.notifyDataSetChanged();
    listKMDedomenon.clear();
    mKMDedomenonAdapter.setList(listKMDedomenon,blockLineData);
    mKMDedomenonAdapter.notifyDataSetChanged();
```

```
        intDeiktisEnt = 0;
        intEtiketaEnt = 0;
        accessCacheEntolon = 0;
        hitsEntolon = 0;
        intDeiktisEntolon.setText("");
        intEtiketaEntolon.setText("");
        accessKMEntolon.setText("");
        hitsKMEntolon.setText("");
        hitRateKMEntolon.setText("");

        intDeiktisDed = 0;
        intEtiketaDed = 0;
        accessCacheDedomenon = 0;
        hitsDedomenon = 0;
        intDeiktisDedomenon.setText("");
        intEtiketaDedomenon.setText("");
        accessKMDedomenon.setText("");
        hitsKMDedomenon.setText("");
        hitRateKMDedomenon.setText("");

//=================================================================================//
        line = 0;
        code_is_running = false;
        antisKatTimi.clear();
        antisKatDieut.clear();
        antisDieutTimi.clear();

//=================================================================================//
        setupDeiktisCache();
    }


//=================================================================================//

    private void updateRecyclerViewEntolon()
    {
        mEntolesAdapter.setList(listEntolon,line);
        mEntolesAdapter.notifyDataSetChanged();
    }


//=================================================================================//

    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        antisKatTimi.put("R0","0");
        setupDeiktisCache();
```

```
instrRun = (Button)findViewById(R.id.instruction_running);
instrRun.setOnClickListener(new View.OnClickListener()
{
   @Override
   public void onClick(View v)
   {
      code_is_running = true;
      if(listEntolon.size()>=1)
      {
         if(line<listEntolon.size())
         {
            updateRecyclerViewEntolon();

            int jumpLine;
            jumpLine = runCode(line);

            if(jumpLine>-1)
            {
               line = jumpLine+1;
            }
            else
            {
               line = line + 1;
            }
         }
      }
   }
});
codeRun = (Button)findViewById(R.id.code_running);
codeRun.setOnClickListener(new View.OnClickListener()
{
   @Override
   public void onClick(View v)
   {
      code_is_running = true;
      if(listEntolon.size()>=1)
      {
         int jumpLine;
         for(int i=line;i<listEntolon.size();i++)
         {
            line = i;

            updateRecyclerViewEntolon();

            jumpLine = runCode(i);
            if(jumpLine>-1)
            {
               i = jumpLine;
               line = i;
            }
         }
      }
   }
});
```

```
//=====================================================================
======================//
        recyclerViewEntolon = findViewById(R.id.recyclerViewEntolon);
        recyclerViewEntolon.setLayoutManager(new LinearLayoutManager(getApplicationContext()));
        mEntolesAdapter = new EntolesAdapter(listEntolon);
        recyclerViewEntolon.setAdapter(mEntolesAdapter);

//=====================================================================
======================//
        recyclerViewKataxoriton = findViewById(R.id.recyclerViewKataxoriton);
        GridLayoutManager gridLayoutManager = new
GridLayoutManager(getApplicationContext(),4,GridLayoutManager.VERTICAL,false)
        {
          public boolean canScrollVertically()
          {
            return true;
          }
          public boolean canScrollHorizontally()
          {
            return true;
          }
        };
        recyclerViewKataxoriton.setLayoutManager(gridLayoutManager);
        mKataxoritesAdapter = new KataxoritesAdapter(listKataxoriton);
        recyclerViewKataxoriton.setAdapter(mKataxoritesAdapter);

//=====================================================================
======================//
        recyclerViewKMEntolon = findViewById(R.id.recyclerViewKMEnt);
        if(cacheMappingEnt.equals("Direct Mapping"))
        {
          recyclerViewKMEntolon.setLayoutManager(new LinearLayoutManager(getApplicationContext()));
        }
        else if(cacheMappingEnt.equals("Associative"))
        {
          recyclerViewKMEntolon.setLayoutManager(new LinearLayoutManager(getApplicationContext()));
        }
        else if(cacheMappingEnt.equals("Full Associative"))
        {
          recyclerViewKMEntolon.setLayoutManager(new LinearLayoutManager(getApplicationContext()));
        }
        mKMEntolonAdapter = new KMEntolonAdapter(listKMEntolon);
        recyclerViewKMEntolon.setAdapter(mKMEntolonAdapter);

//=====================================================================
======================//
        recyclerViewKMDedomenon = findViewById(R.id.recyclerViewKMDed);
        if(cacheMappingDed.equals("Direct Mapping"))
        {
          recyclerViewKMDedomenon.setLayoutManager(new
LinearLayoutManager(getApplicationContext()));
        }
        else if(cacheMappingDed.equals("Associative"))
        {
          recyclerViewKMDedomenon.setLayoutManager(new
LinearLayoutManager(getApplicationContext()));
```

```
        }
        else if(cacheMappingDed.equals("Full Associative"))
        {
            recyclerViewKMDedomenon.setLayoutManager(new
LinearLayoutManager(getApplicationContext()));
        }
        mKMDedomenonAdapter = new KMDedomenonAdapter(listKMDedomenon);
        recyclerViewKMDedomenon.setAdapter(mKMDedomenonAdapter);

//================================================================
======================//
        dedomenaStinMnimi = (TextView) findViewById(R.id.dieuthinseisDed);
        dedomenaStinMnimi.setMovementMethod(new ScrollingMovementMethod());
        metavlitesDieuthinsisTxtView = (TextView) findViewById(R.id.metavlites_dieuthinsis);
        metavlitesDieuthinsisTxtView.setMovementMethod(new ScrollingMovementMethod());
        dieutEntolon = (TextView) findViewById(R.id.dieuthEnt);
        dieutDedomenon = (TextView) findViewById(R.id.dieuthDed);
        intDeiktisEntolon = (TextView) findViewById(R.id.deiktisEnt);
        intEtiketaEntolon = (TextView) findViewById(R.id.etiketaEnt);
        intDeiktisDedomenon = (TextView) findViewById(R.id.deiktisDed);
        intEtiketaDedomenon = (TextView) findViewById(R.id.etiketaDed);
        accessKMEntolon = (TextView) findViewById(R.id.accessEnt);
        hitsKMEntolon = (TextView) findViewById(R.id.hitsEnt);
        hitRateKMEntolon = (TextView) findViewById(R.id.hitRateEnt);
        accessKMDedomenon = (TextView) findViewById(R.id.accessDed);
        hitsKMDedomenon = (TextView) findViewById(R.id.hitsDed);
        hitRateKMDedomenon = (TextView) findViewById(R.id.hitRateDed);
        instructionNumTextView = (TextView) findViewById(R.id.instruction_num_textView);
        dataNumTextView = (TextView) findViewById(R.id.data_num_textView);
    }


//================================================================
==========================//

    private class EntolesHolder extends RecyclerView.ViewHolder
    {
        private final TextView textView;
        public EntolesHolder(LayoutInflater inflater,ViewGroup parent)
        {
            super(inflater.inflate(R.layout.entoles,parent,false));
            textView = itemView.findViewById(R.id.entoli);
        }
        private void bind (String entoli,int position,int lineInRecyclerView)
        {
            textView.setText(entoli);
            if(position==lineInRecyclerView && code_is_running)
            {
                textView.setBackground(getResources().getDrawable(R.drawable.running_instruction));
            }
            else
            {
                textView.setBackground(getResources().getDrawable(R.drawable.border));
            }
        }
    }
```

```
  private class EntolesAdapter extends RecyclerView.Adapter<EntolesHolder>
  {
    private List<entoli> list;
    private int lineInRecyclerView;
    public EntolesAdapter(List<entoli> listt)
    {
      list = listt;
    }
    @NonNull
    @Override
    public EntolesHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType)
    {
      LayoutInflater layoutInflater = LayoutInflater.from(getApplicationContext());
      return new EntolesHolder(layoutInflater, parent);
    }
    @Override
    public void onBindViewHolder(EntolesHolder holder,int position)
    {
      String entoli = list.get(position).instructionAddress+"  "+list.get(position).olokEntoli;
      holder.bind(entoli,position,lineInRecyclerView);
    }
    @Override
    public int getItemCount()
    {
      return list.size();
    }
    public void setList(List<entoli> listt,int lineInRecyclerView2)
    {
      list = listt;
      lineInRecyclerView = lineInRecyclerView2;
    }
  }

//=================================================================
============================
  private class KataxoritesHolder extends RecyclerView.ViewHolder
  {
    private final TextView textView1;
    public KataxoritesHolder(LayoutInflater inflater,ViewGroup parent)
    {
      super(inflater.inflate(R.layout.kataxorites,parent,false));
      textView1 = itemView.findViewById(R.id.timi_kataxoriti);
    }
    private void bind (String katax)
    {
      textView1.setText(katax);
    }
  }
  private class KataxoritesAdapter extends RecyclerView.Adapter<KataxoritesHolder>
  {
    private List<String> list;
    public KataxoritesAdapter(List<String> listt)
    {
      list = listt;
    }
    @NonNull
```

```
    @Override
    public KataxoritesHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType)
    {
        LayoutInflater layoutInflater = LayoutInflater.from(getApplicationContext());
        return new KataxoritesHolder(layoutInflater, parent);
    }
    @Override
    public void onBindViewHolder(KataxoritesHolder holder,int position)
    {
        String katax = list.get(position);
        holder.bind(katax);
    }
    @Override
    public int getItemCount()
    {
        return list.size();
    }
    public void setList(List<String> listt)
    {
        list = listt;
    }
}

//=================================================================================
=============================
  private class KMEntolonHolder extends RecyclerView.ViewHolder
  {
    private final TextView textView1;
    private final TextView textView2;
    private final TextView textView3;
    private final TextView textView4;

    public KMEntolonHolder(LayoutInflater inflater,ViewGroup parent)
    {
        super(inflater.inflate(R.layout.km_entolon,parent,false));
        textView1 = (TextView) itemView.findViewById(R.id.plirofories_km_entolon);
        textView2 = (TextView) itemView.findViewById(R.id.plirofories_km_entolon_2);
        textView3 = (TextView) itemView.findViewById(R.id.plirofories_km_entolon_3);
        textView4 = (TextView) itemView.findViewById(R.id.plirofories_km_entolon_4);
    }
    private void bind (BlockKmEntolon block,int position,int blockLineInRecyclerView)
    {
        textView1.setText(block.number);
        textView2.setText(block.olokEntoli);
        textView3.setText(block.etiketa);
        if(position==blockLineInRecyclerView && code_is_running)
        {
            if(block.access)
            {
                textView4.setText("Hit");
            }
            else
            {
                textView4.setText("Missed");
            }
        }
```

```
            else
            {
               textView4.setText("");
            }
         }
      }
      private class KMEntolonAdapter extends RecyclerView.Adapter<KMEntolonHolder>
      {
         private List<BlockKmEntolon> list;
         private int blockLineInRecyclerView;
         public KMEntolonAdapter(List<BlockKmEntolon> listt)
         {
            list = listt;
         }
         @NonNull
         @Override
         public KMEntolonHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType)
         {
            LayoutInflater layoutInflater = LayoutInflater.from(getApplicationContext());
            return new KMEntolonHolder(layoutInflater, parent);
         }
         @Override
         public void onBindViewHolder(KMEntolonHolder holder,int position)
         {
            BlockKmEntolon block = list.get(position);
            holder.bind(block,position,blockLineInRecyclerView);
         }
         @Override
         public int getItemCount()
         {
            return list.size();
         }
         public void setList(List<BlockKmEntolon> listt,int blockLineInRecyclerView2)
         {
            list = listt;
            blockLineInRecyclerView = blockLineInRecyclerView2;
         }
      }

//=================================================================================
============================
      private class KMDedomenonHolder extends RecyclerView.ViewHolder
      {
         private final TextView textView1;
         private final TextView textView2;
         private final TextView textView3;
         private final TextView textView4;
         public KMDedomenonHolder(LayoutInflater inflater,ViewGroup parent)
         {
            super(inflater.inflate(R.layout.km_dedomenon,parent,false));
            textView1 = (TextView) itemView.findViewById(R.id.plirofories_km_dedomenon);
            textView2 = (TextView) itemView.findViewById(R.id.plirofories_km_dedomenon_2);
            textView3 = (TextView) itemView.findViewById(R.id.plirofories_km_dedomenon_3);
            textView4 = (TextView) itemView.findViewById(R.id.plirofories_km_dedomenon_4);
         }
         private void bind (BlockKmDedomenon block,int position,int blockLineInRecyclerView)
```

```
        {
            textView1.setText(block.number);
            textView2.setText(block.olokEntoli);
            textView3.setText(block.etiketa);
            if(position==blockLineInRecyclerView && code_is_running)
            {
                if(block.access)
                {
                    textView4.setText("Hit");
                }
                else
                {
                    textView4.setText("Missed");
                }
            }
            else
            {
                textView4.setText("");
            }
        }
    }
    private class KMDedomenonAdapter extends RecyclerView.Adapter<KMDedomenonHolder>
    {
        private List<BlockKmDedomenon> list;
        private int blockLineInRecyclerView;
        public KMDedomenonAdapter(List<BlockKmDedomenon> listt)
        {
            list = listt;
        }
        @NonNull
        @Override
        public KMDedomenonHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType)
        {
            LayoutInflater layoutInflater = LayoutInflater.from(getApplicationContext());
            return new KMDedomenonHolder(layoutInflater, parent);
        }
        @Override
        public void onBindViewHolder(KMDedomenonHolder holder,int position)
        {
            BlockKmDedomenon block = list.get(position);
            holder.bind(block,position,blockLineInRecyclerView);
        }
        @Override
        public int getItemCount()
        {
            return list.size();
        }
        public void setList(List<BlockKmDedomenon> listt,int blockLineInRecyclerView2)
        {
            list = listt;
            blockLineInRecyclerView = blockLineInRecyclerView2;
        }
    }

//================================================================
```

```
//==========================

    @Override
    public boolean onCreateOptionsMenu(Menu menu)
    {
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.menu_one, menu);
        return true;
    }
    @Override
    public boolean onOptionsItemSelected(MenuItem item)
    {
        switch(item.getItemId())
        {
            case R.id.configuration:
                Intent intent =
Configuration.newIntent(this,cacheMappingEnt,cacheMappingDed,writingPolicyEnt,writingPolicyDed);
                startActivityForResult(intent,REQUEST_CODE_CONF);
                return true;
            case R.id.write:
                Intent intent2 = WriteText.newIntent(this);
                startActivityForResult(intent2,REQUEST_CODE_CONF);
                return true;
            case R.id.read:
                readText();
                return true;
            case R.id.clear:
                clear();
                return true;
            case R.id.instructions:
                Intent intent3 = ReadInstructions.newIntent(this);
                startActivity(intent3);
                return true;
            default:
                return super.onOptionsItemSelected(item);
        }
    }

//=========================================================================
==========================//
    @Override
    protected void onActivityResult(int requestCode,int resultCode,Intent intent)
    {
        super.onActivityResult(requestCode, resultCode, intent);
        if(requestCode==REQUEST_CODE_CONF)
        {
            if(intent==null)
            {
                return;
            }
            Log.d(TAG,Configuration.InstructionStartingAddress(intent));
            Log.d(TAG,Configuration.SizeKMEntolon(intent));
            Log.d(TAG,Configuration.SizeBlockEntolon(intent));
            Log.d(TAG,Configuration.BlocksSinolouCacheEnt(intent));
            Log.d(TAG,Configuration.ArxKMEntolon(intent));
            Log.d(TAG,Configuration.EgrafiKMEntolon(intent));
```

```
        Log.d(TAG,Configuration.DataStartingAddress(intent));
        Log.d(TAG,Configuration.SizeKMDedomenon(intent));
        Log.d(TAG,Configuration.SizeBlockDedomenon(intent));
        Log.d(TAG,Configuration.BlocksSinolouCacheDed(intent));
        Log.d(TAG,Configuration.ArxKMDedomenon(intent));
        Log.d(TAG,Configuration.EgrafiKMDedomenon(intent));


        instructionAddress = Integer.parseInt(Configuration.InstructionStartingAddress(intent));
        cacheSizeEnt = Integer.parseInt(Configuration.SizeKMEntolon(intent));
        blockSIzeEnt = Integer.parseInt(Configuration.SizeBlockEntolon(intent));
        if(Integer.parseInt(Configuration.BlocksSinolouCacheEnt(intent)) != 0)
        {
            blocksSinolouCacheEnt = Integer.parseInt(Configuration.BlocksSinolouCacheEnt(intent));
        }
        cacheMappingEnt = Configuration.ArxKMEntolon(intent);
        writingPolicyEnt = Configuration.EgrafiKMEntolon(intent);

        dataAddress = Integer.parseInt(Configuration.DataStartingAddress(intent));
        startingDataAddress = Integer.parseInt(Configuration.DataStartingAddress(intent));
        cacheSizeDed = Integer.parseInt(Configuration.SizeKMDedomenon(intent));
        blockSIzeDed = Integer.parseInt(Configuration.SizeBlockDedomenon(intent));
        if(Integer.parseInt(Configuration.BlocksSinolouCacheDed(intent)) != 0)
        {
            blocksSinolouCacheDed = Integer.parseInt(Configuration.BlocksSinolouCacheDed(intent));
        }
        cacheMappingDed = Configuration.ArxKMDedomenon(intent);
        writingPolicyDed = Configuration.EgrafiKMDedomenon(intent);

        setupDeiktisCache();
        clear();

        if(cacheMappingEnt.equals("Associative"))
        {
            instructionNumTextView.setText("Sets");
        }
        if(cacheMappingDed.equals("Associative"))
        {
            dataNumTextView.setText("Sets");
        }
    }
  }
}
```

Κλάση Configuration.java

```
package com.example.myapplication;

import android.content.Context;
import android.content.Intent;
```

```
import android.os.Bundle;
import android.text.Editable;
import android.text.TextWatcher;
import android.widget.EditText;
import android.widget.RadioButton;
import android.widget.RadioGroup;

import androidx.appcompat.app.AppCompatActivity;

public class Configuration extends AppCompatActivity
{
   private static final String INSTRUCTION_STARTING_ADDRESS = "AA";
   private static final String SIZE_KM_ENTOLON = "BB";
   private static final String SIZE_BLOCK_ENTOLON = "CC";
   private static final String BLOCKS_SINOLOY_KM_ENTOLON = "DD";
   private static final String ARXIT_KM_ENTOLON = "EE";
   private static final String EGRAFI_BLOCK_ENTOLON = "FF";
   private static final String DATA_STARTING_ADDRESS = "A";
   private static final String SIZE_KM_DEDOMENON = "B";
   private static final String SIZE_BLOCK_DEDOMENON = "C";
   private static final String BLOCKS_SINOLOY_KM_DEDOMENON = "D";
   private static final String ARXIT_KM_DEDOMENON = "E";
   private static final String EGRAFI_BLOCK_DEDOMENON = "F";
   private static String cacheMappingEnt;
   private static String cacheMappingDed;
   private static String writingPolicyEnt;
   private static String writingPolicyDed;

   private EditText instructionStartingAddress;
   private EditText dataStartingAddress;
   private EditText sizeKMEntolon;
   private EditText sizeKMDedomenon;
   private EditText sizeBlockEntolon;
   private EditText sizeBlockDedomenon;
   private EditText blocksSinolouKMEntolon;
   private EditText blocksSinolouKMDedomenon;
   private RadioGroup arxitKMEntolon;
   private RadioGroup egrafiBlockEntolon;
   private RadioGroup arxitKMDedomenon;
   private RadioGroup egrafiBlockDedomenon;
   private RadioButton randomEnt;
   private RadioButton LRUEnt;
   private RadioButton FIFOEnt;
   private RadioButton randomDed;
   private RadioButton LRUDed;
   private RadioButton FIFODed;

   public static Intent newIntent (Context packageContext,String cacheMappingent,String
cacheMappingded,String writingPolicyent,String writingPolicyded)
   {
      Intent intent = new Intent(packageContext,Configuration.class);
      cacheMappingEnt = cacheMappingent;
      cacheMappingDed = cacheMappingded;
      writingPolicyEnt = writingPolicyent;
      writingPolicyDed = writingPolicyded;
      return intent;
```

```
    }

    @Override
    protected void onCreate(Bundle savedInstance)
    {
        super.onCreate(savedInstance);
        setContentView(R.layout.configuration);

        Intent intent = new Intent();
        Bundle bundle = new Bundle();

        randomEnt = findViewById(R.id.random_entolon);
        LRUEnt = findViewById(R.id.lru_entolon);
        FIFOEnt = findViewById(R.id.fifo_entolon);
        randomDed = findViewById(R.id.random_dedomenon);
        LRUDed = findViewById(R.id.lru_dedomenon);
        FIFODed = findViewById(R.id.fifo_dedomenon);

//=====================================================================
======================//

        instructionStartingAddress = (EditText) findViewById(R.id.instruction_address_editText);
        instructionStartingAddress.addTextChangedListener(new TextWatcher()
        {
            @Override
            public void beforeTextChanged(CharSequence s, int start, int count, int after)
            {           }
            @Override
            public void onTextChanged(CharSequence s, int start, int before, int count)
            {
                bundle.putString(INSTRUCTION_STARTING_ADDRESS, s.toString());
                intent.putExtras(bundle);
                setResult(RESULT_OK,intent);
            }
            @Override
            public void afterTextChanged(Editable s)
            {           }
        });
        sizeKMEntolon = (EditText) findViewById(R.id.size_km_entolon);
        sizeKMEntolon.addTextChangedListener(new TextWatcher()
        {
            @Override
            public void beforeTextChanged(CharSequence s, int start, int count, int after)
            {           }
            @Override
            public void onTextChanged(CharSequence s, int start, int before, int count)
            {
                bundle.putString(SIZE_KM_ENTOLON, s.toString());
                intent.putExtras(bundle);
                setResult(RESULT_OK,intent);
            }
            @Override
            public void afterTextChanged(Editable s)
            {           }
        });
        sizeBlockEntolon = (EditText) findViewById(R.id.size_block_entolon);
```

```
sizeBlockEntolon.addTextChangedListener(new TextWatcher()
{
   @Override
   public void beforeTextChanged(CharSequence s, int start, int count, int after)
   {          }
   @Override
   public void onTextChanged(CharSequence s, int start, int before, int count)
   {
      bundle.putString(SIZE_BLOCK_ENTOLON, s.toString());
      intent.putExtras(bundle);
      setResult(RESULT_OK,intent);
   }
   @Override
   public void afterTextChanged(Editable s)
   {          }
});
blocksSinolouKMEntolon = (EditText) findViewById(R.id.sinola_km_entolon);
blocksSinolouKMEntolon.addTextChangedListener(new TextWatcher()
{
   @Override
   public void beforeTextChanged(CharSequence s, int start, int count, int after)
   {

   }
   @Override
   public void onTextChanged(CharSequence s, int start, int before, int count)
   {
      arxitKMEntolon.check(R.id.sisxetiki_entolon);

      bundle.putString(BLOCKS_SINOLOY_KM_ENTOLON, s.toString());
      intent.putExtras(bundle);
      setResult(RESULT_OK,intent);
   }
   @Override
   public void afterTextChanged(Editable s)
   {          }
});
dataStartingAddress = (EditText) findViewById(R.id.data_address_editText);
dataStartingAddress.addTextChangedListener(new TextWatcher()
{
   @Override
   public void beforeTextChanged(CharSequence s, int start, int count, int after)
   {          }
   @Override
   public void onTextChanged(CharSequence s, int start, int before, int count)
   {
      bundle.putString(DATA_STARTING_ADDRESS, s.toString());
      intent.putExtras(bundle);
      setResult(RESULT_OK,intent);
   }
   @Override
   public void afterTextChanged(Editable s)
   {          }
});
sizeKMDedomenon = (EditText) findViewById(R.id.size_km_dedomenon);
sizeKMDedomenon.addTextChangedListener(new TextWatcher()
```

```
{
   @Override
   public void beforeTextChanged(CharSequence s, int start, int count, int after)
   {          }
   @Override
   public void onTextChanged(CharSequence s, int start, int before, int count)
   {
      bundle.putString(SIZE_KM_DEDOMENON, s.toString());
      intent.putExtras(bundle);
      setResult(RESULT_OK,intent);
   }
   @Override
   public void afterTextChanged(Editable s)
   {          }
});
sizeBlockDedomenon = (EditText) findViewById(R.id.size_block_dedomenon);
sizeBlockDedomenon.addTextChangedListener(new TextWatcher()
{
   @Override
   public void beforeTextChanged(CharSequence s, int start, int count, int after)
   {          }
   @Override
   public void onTextChanged(CharSequence s, int start, int before, int count)
   {
      bundle.putString(SIZE_BLOCK_DEDOMENON, s.toString());
      intent.putExtras(bundle);
      setResult(RESULT_OK,intent);
   }
   @Override
   public void afterTextChanged(Editable s)
   {          }
});
blocksSinolouKMDedomenon = (EditText) findViewById(R.id.sinola_km_dedomenon);
blocksSinolouKMDedomenon.addTextChangedListener(new TextWatcher()
{
   @Override
   public void beforeTextChanged(CharSequence s, int start, int count, int after)
   {

   }
   @Override
   public void onTextChanged(CharSequence s, int start, int before, int count)
   {
      arxitKMDedomenon.check(R.id.sisxetiki_dedomenon);

      bundle.putString(BLOCKS_SINOLOY_KM_DEDOMENON, s.toString());
      intent.putExtras(bundle);
      setResult(RESULT_OK,intent);
   }
   @Override
   public void afterTextChanged(Editable s)
   {

   }
});
```

```
//=====================================================================
====================//
        arxitKMEntolon = findViewById(R.id.arxit_km_entolon);
        arxitKMEntolon.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener()
        {
            @Override
            public void onCheckedChanged(RadioGroup group, int checkedId)
            {
                RadioButton radioButton = findViewById(checkedId);
                bundle.putString(ARXIT_KM_ENTOLON,radioButton.getText().toString());
                intent.putExtras(bundle);

                cacheMappingEnt = radioButton.getText().toString();
                if(cacheMappingEnt.equals("Direct Mapping"))
                {
                    randomEnt.setEnabled(false);
                    LRUEnt.setEnabled(false);
                    FIFOEnt.setEnabled(false);
                }
                else
                {
                    randomEnt.setEnabled(true);
                    LRUEnt.setEnabled(true);
                    FIFOEnt.setEnabled(true);
                }

                setResult(RESULT_OK,intent);
            }
        });
        egrafiBlockEntolon = findViewById(R.id.egrafi_block_entolon);
        egrafiBlockEntolon.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener()
        {
            @Override
            public void onCheckedChanged(RadioGroup group, int checkedId)
            {
                RadioButton radioButton = findViewById(checkedId);
                bundle.putString(EGRAFI_BLOCK_ENTOLON,radioButton.getText().toString());
                intent.putExtras(bundle);
                setResult(RESULT_OK,intent);
            }
        });
        arxitKMDedomenon = findViewById(R.id.arxit_km_dedomenon);
        arxitKMDedomenon.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener()
        {
            @Override
            public void onCheckedChanged(RadioGroup group, int checkedId)
            {
                RadioButton radioButton = findViewById(checkedId);
                bundle.putString(ARXIT_KM_DEDOMENON,radioButton.getText().toString());
                intent.putExtras(bundle);

                cacheMappingDed = radioButton.getText().toString();
                if(cacheMappingDed.equals("Direct Mapping"))
                {
                    randomDed.setEnabled(false);
                    LRUDed.setEnabled(false);
```

```
            FIFODed.setEnabled(false);
        }
        else
        {
            randomDed.setEnabled(true);
            LRUDed.setEnabled(true);
            FIFODed.setEnabled(true);
        }

        setResult(RESULT_OK,intent);
    }
});
egrafiBlockDedomenon = findViewById(R.id.egrafi_block_dedomenon);
egrafiBlockDedomenon.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener()
{
    @Override
    public void onCheckedChanged(RadioGroup group, int checkedId)
    {
        RadioButton radioButton = findViewById(checkedId);
        bundle.putString(EGRAFI_BLOCK_DEDOMENON,radioButton.getText().toString());
        intent.putExtras(bundle);
        setResult(RESULT_OK,intent);
    }
});


//=============================================================================
======================//

if(cacheMappingEnt.equals("Direct Mapping"))
{
    randomEnt.setEnabled(false);
    LRUEnt.setEnabled(false);
    FIFOEnt.setEnabled(false);
}
else if(cacheMappingEnt.equals("Associative"))
{
    randomEnt.setEnabled(true);
    LRUEnt.setEnabled(true);
    FIFOEnt.setEnabled(true);
    arxitKMEntolon.check(R.id.sisxetiki_entolon);
    if(writingPolicyEnt.equals("random"))
    {
        egrafiBlockEntolon.check(R.id.random_entolon);
    }
    if(writingPolicyEnt.equals("LRU"))
    {
        egrafiBlockEntolon.check(R.id.lru_entolon);
    }
    if(writingPolicyEnt.equals("FIFO"))
    {
        egrafiBlockEntolon.check(R.id.fifo_entolon);
    }
}
else if(cacheMappingEnt.equals("Full Associative"))
{
```

```
        randomEnt.setEnabled(true);
        LRUEnt.setEnabled(true);
        FIFOEnt.setEnabled(true);
        arxitKMEntolon.check(R.id.pliros_sisxetiki_entolon);
        if(writingPolicyEnt.equals("random"))
        {
            egrafiBlockEntolon.check(R.id.random_entolon);
        }
        if(writingPolicyEnt.equals("LRU"))
        {
            egrafiBlockEntolon.check(R.id.lru_entolon);
        }
        if(writingPolicyEnt.equals("FIFO"))
        {
            egrafiBlockEntolon.check(R.id.fifo_entolon);
        }
    }
    if(cacheMappingDed.equals("Direct Mapping"))
    {
        randomDed.setEnabled(false);
        LRUDed.setEnabled(false);
        FIFODed.setEnabled(false);
    }
    else if(cacheMappingDed.equals("Associative"))
    {
        randomDed.setEnabled(true);
        LRUDed.setEnabled(true);
        FIFODed.setEnabled(true);
        arxitKMDedomenon.check(R.id.sisxetiki_dedomenon);
        if(writingPolicyDed.equals("random"))
        {
            egrafiBlockDedomenon.check(R.id.random_dedomenon);
        }
        if(writingPolicyDed.equals("LRU"))
        {
            egrafiBlockDedomenon.check(R.id.lru_dedomenon);
        }
        if(writingPolicyDed.equals("FIFO"))
        {
            egrafiBlockDedomenon.check(R.id.fifo_dedomenon);
        }
    }
    else if(cacheMappingDed.equals("Full Associative"))
    {
        randomDed.setEnabled(true);
        LRUDed.setEnabled(true);
        FIFODed.setEnabled(true);
        arxitKMDedomenon.check(R.id.pliros_sisxetiki_dedomenon);
        if(writingPolicyDed.equals("random"))
        {
            egrafiBlockDedomenon.check(R.id.random_dedomenon);
        }
        if(writingPolicyDed.equals("LRU"))
        {
            egrafiBlockDedomenon.check(R.id.lru_dedomenon);
        }
```

```
        if(writingPolicyDed.equals("FIFO"))
        {
           egrafiBlockDedomenon.check(R.id.fifo_dedomenon);
        }
      }
   }

//=====================================================================
======================//
   public static String InstructionStartingAddress (Intent i)
   {
      return i.getExtras().getString(INSTRUCTION_STARTING_ADDRESS,"1000");
   }
   public static String SizeKMEntolon (Intent i)
   {
      return i.getExtras().getString(SIZE_KM_ENTOLON,"128");
   }
   public static String SizeBlockEntolon (Intent i)
   {
      return i.getExtras().getString(SIZE_BLOCK_ENTOLON,"4");
   }
   public static String BlocksSinolouCacheEnt (Intent i)
   {
      return i.getExtras().getString(BLOCKS_SINOLOY_KM_ENTOLON,"8");
   }
   public static String ArxKMEntolon (Intent i)
   {
      return i.getExtras().getString(ARXIT_KM_ENTOLON,"Direct Mapping");
   }
   public static String EgrafiKMEntolon (Intent i)
   {
      return i.getExtras().getString(EGRAFI_BLOCK_ENTOLON,"LRU");
   }
   public static String DataStartingAddress (Intent i)
   {
      return i.getExtras().getString(DATA_STARTING_ADDRESS,"10000");
   }
   public static String SizeKMDedomenon (Intent i)
   {
      return i.getExtras().getString(SIZE_KM_DEDOMENON,"128");
   }
   public static String SizeBlockDedomenon (Intent i)
   {
      return i.getExtras().getString(SIZE_BLOCK_DEDOMENON,"4");
   }
   public static String BlocksSinolouCacheDed (Intent i)
   {
      return i.getExtras().getString(BLOCKS_SINOLOY_KM_DEDOMENON,"8");
   }
   public static String ArxKMDedomenon (Intent i)
   {
      return i.getExtras().getString(ARXIT_KM_DEDOMENON,"Direct Mapping");
   }
   public static String EgrafiKMDedomenon (Intent i)
   {
      return i.getExtras().getString(EGRAFI_BLOCK_DEDOMENON,"LRU");
```

```
    }
}
```

Κλάση WriteText.java

```
package com.example.myapplication;

import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.text.Editable;
import android.text.TextWatcher;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.util.Scanner;

public class WriteText extends AppCompatActivity
{
    private EditText mEditText;
    private Button mButton;

    public static Intent newIntent (Context packageContext)
    {
        return new Intent(packageContext, WriteText.class);
    }

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.text_write);

        mEditText = findViewById(R.id.text);
        mEditText.setText(ReadText());

        mButton = findViewById(R.id.button);
        mButton.setOnClickListener(new View.OnClickListener()
```

```
        {
            @Override
            public void onClick(View v)
            {
                if(!mEditText.getText().toString().isEmpty())
                {
                    FileOutputStream fileOutputStream = null;
                    try
                    {
                        fileOutputStream = openFileOutput("codefile.txt", MODE_PRIVATE);
                    }
                    catch (FileNotFoundException e)
                    {
                        e.printStackTrace();
                    }
                    OutputStreamWriter outputWriter = new OutputStreamWriter(fileOutputStream);
                    try
                    {
                        outputWriter.write(mEditText.getText().toString());
                        outputWriter.close();
                    }
                    catch (IOException e)
                    {
                        e.printStackTrace();
                    }
                    Toast.makeText(getApplicationContext(),"File saved
successfully!",Toast.LENGTH_SHORT).show();
                }
                else
                {
                    Toast.makeText(getApplicationContext(), "No input?", Toast.LENGTH_SHORT).show();
                }
            }
        });
    }
    private String ReadText()
    {
        String str = "";
        Scanner myReader = null;
        try {
            FileInputStream fileInputStream = openFileInput("codefile.txt");
            myReader = new Scanner(fileInputStream);

            while (myReader.hasNextLine())
            {
                str = str+myReader.nextLine()+"\n";
            }
            myReader.close();
        }
        catch (FileNotFoundException e)
        {
            e.printStackTrace();
        }
        return str;
    }
```

}

Κλάση ReadInstructions.java

```
package com.example.myapplication;

import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.text.method.ScrollingMovementMethod;
import android.widget.TextView;

import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;

public class ReadInstructions extends AppCompatActivity
{
   private TextView instructionTextView;

   public static Intent newIntent (Context packageContext)
   {
      return new Intent(packageContext,ReadInstructions.class);
   }

   protected void onCreate(@Nullable Bundle savedInstanceState)
   {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.instructions_read);

      instructionTextView = (TextView) findViewById(R.id.instructions_textView);
      instructionTextView.setMovementMethod(new ScrollingMovementMethod());

      ReadText();
   }
   private void ReadText()
   {
      String str = "data: (anathesi timon)\n" +
            "R = TIMI \n"+
            "main: (ekkinisi kodika)\n" +

            "LOAD R,TIMI(RD) \n" +
            "LOAD R,TIMI\n" +
            "LOADI R,TIMI\n" +
            "STORE R,TIMI(RD)\n" +
            "STORE R,TIMI\n" +
            "STOREI TIMI2,TIMI(RD)\n" +
            "STOREI TIMI2,TIMI\n" +
            "ADDI R1,R2,TIMI\n" +
            "SUBI R1,R2,TIMI\n" +
            "MULI R1,R2,TIMI\n" +
```

```
            "DIVI R1,R2,TIMI \n" +
            "ANDI R1,R2,TIMI\n" +
            "ORI R1,R2,TIMI\n" +
            "XORI R1,R2,TIMI\n" +

            "BEQ R1,R2,LABEL \n" +
            "BEQZ R,LABEL\n" +
            "BNE R1,R2,LABEL\n" +
            "BNEZ R,LABEL\n" +

            "\n" +

            "MOVE R1,R2\n" +
            "MOVEN R1,R2,R3\n" +
            "MOVEZ R1,R2,R3\n" +
            "ADD R1,R2,R3\n" +
            "SUB R1,R2,R3\n" +
            "MUL R1,R2,R3\n" +
            "DIV R1,R2,R3 \n" +
            "AND R1,R2,R3\n" +
            "OR R1,R2,R3\n" +
            "XOR R1,R2,R3\n" +
            "SLIDEL R1,R2,TIMI\n" +
            "SLIDER R1,R2,TIMI\n" +
            "SMALLER R1,R2,R3\n" +
            "BIGGER R1,R2,R3\n" +
            "\n" +

            "J LABEL\n" +
            "JAL LABEL\n" +
            "\n" +
            "ENDF\n" +
            "END";

        instructionTextView.setText(str);
    }
}
```

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Δημήτρης Κεχαγιάς, Διαφάνειες μαθήματος.

[2] D. Patterson & J. Hennessy: Computer Organization and Design, 5th Edition.

[3] Intel.

[4] Bruce Jacob, David Wang, Spencer Ng, '*Memory system: Cache, DRAM, DISK, 1η έκδοση*', Morgan Kaufmann, 2007

[5] M. Morris Mano, Michael D. Ciletti, '*Ψηφιακή σχεδίαση, 4η έκδοση*', Εκδόσεις Παπασωτηρίου, 2010

[6] Sedra/Smith, '*Μικροηλεκτρονικά κυκλώματα, 5η έκδοση*', Εκδόσεις Παπασωτηρίου, 2011

[7] Bill Phillips, Chris Stewart, Kristin Marsicano, '*Android Programming: the big nerd ranch guide, 3η έκδοση*', 2017

[8] https://www.w3schools.com/java/default.asp , προσπέλαση 2023

[9] https://www.geeksforgeeks.org/java/?ref=shm_outind , προσπέλαση 2023

[10] https://www.geeksforgeeks.org/android-tutorial , προσπέλαση 2023

[11] https://www.javatpoint.com/java-tutorial , προσπέλαση 2023

[12] https://www.javatpoint.com/android-tutorial , προσπέλαση 2023

[13] https://www.tutorialspoint.com/java/index.htm , προσπέλαση 2023

[14] https://www.tutorialspoint.com/android/index.htm , προσπέλαση 2023