ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

"Προηγμένα Θέματα Αρχιτεκτονικής Υπολογιστών"

2η Άσκηση

Ορφέας Μενής-Μαστρομιχαλάκης
ΑΜ: 03114052

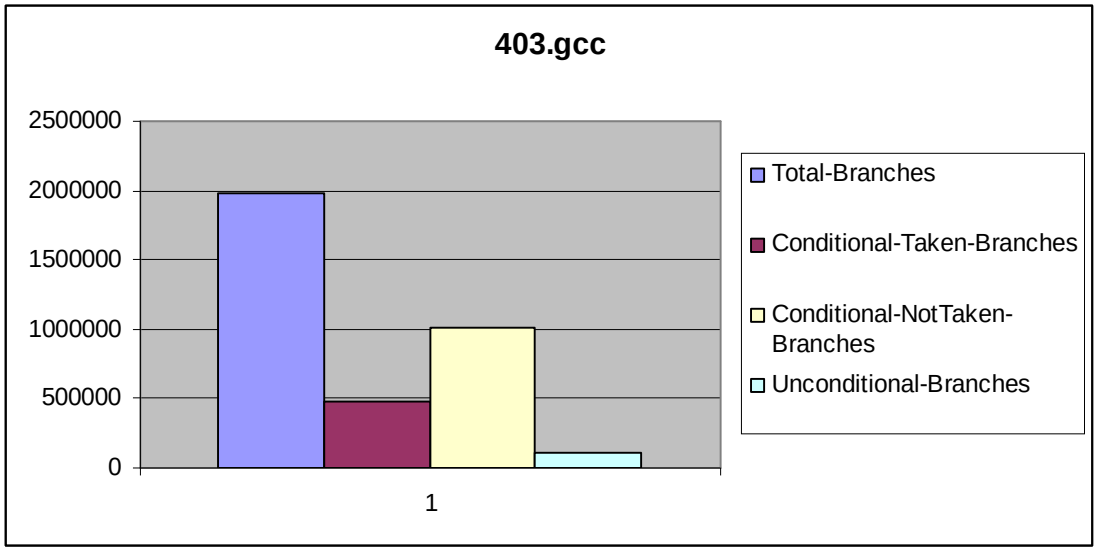
Στα πλαίσια της εργασίας αυτής, θα διερευνηθεί η επίδραση διαφορετικών συστημάτων πρόβλεψης εντολών άλματος καθώς και η αξιολόγηση τους με δεδομένο το διαθέσιμο χώρο πάνω στο τσιπ.

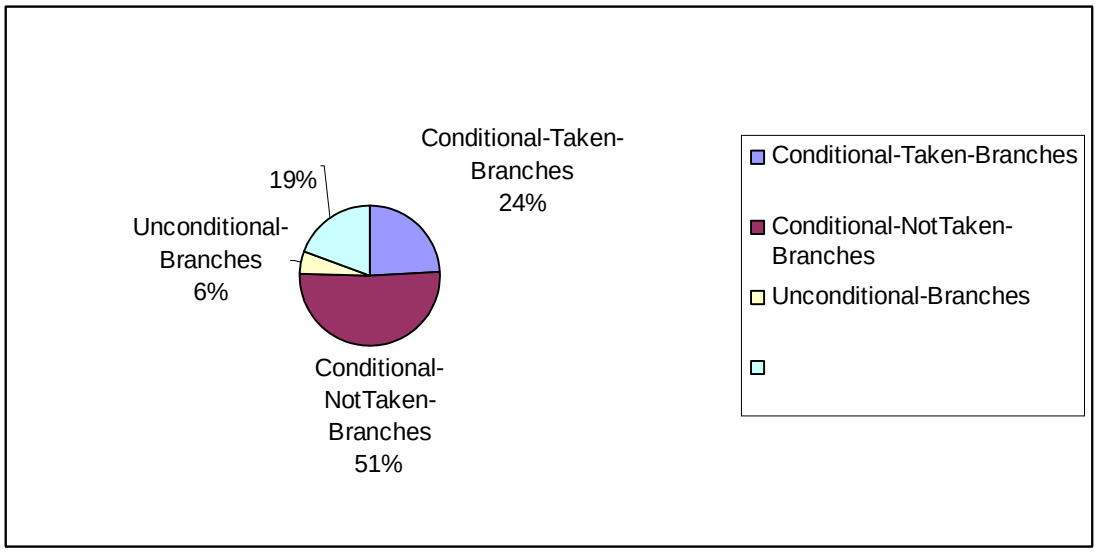
Στα πλαίσια της παρούσας άσκησης χρησιμοποίησα τα SPEC_CPU2006 benchmarks. Πιο συγκεκριμένα χρησιμοποίησα τα παρακάτω 12 benchmarks:

1. 403.gcc
2. 429.mcf
3. 434.zeusmp
4. 436.cactusADM
5. 445.gobmk
6. 450.soplex
7. 456.hmmer
8. 458.sjeng
9. 459.GemsFDTD
10. 471.omnetpp
11. 473.astar
12. 483.xalancbmk

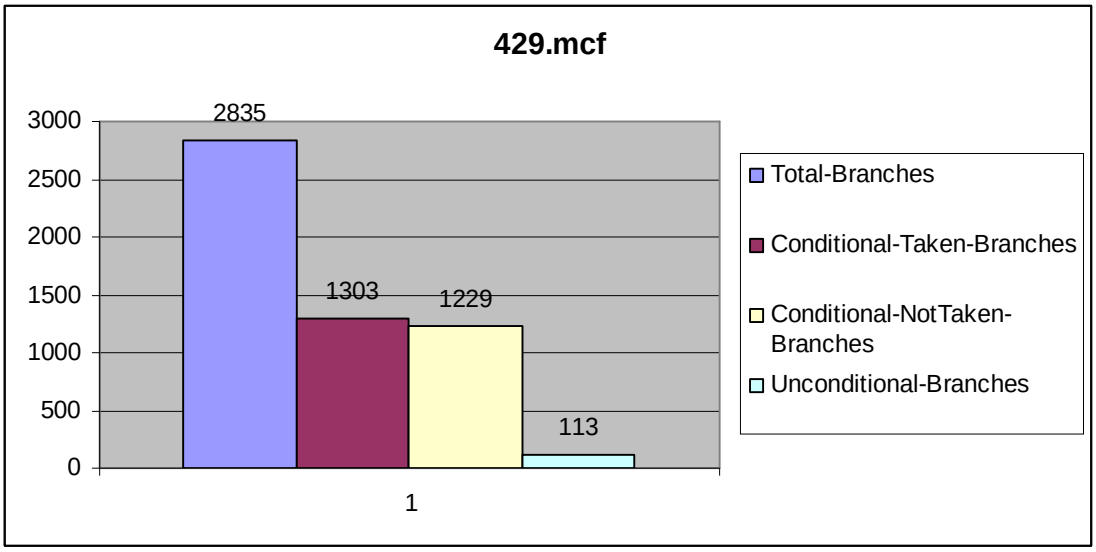
4.1 Μελέτη εντολών άλματος

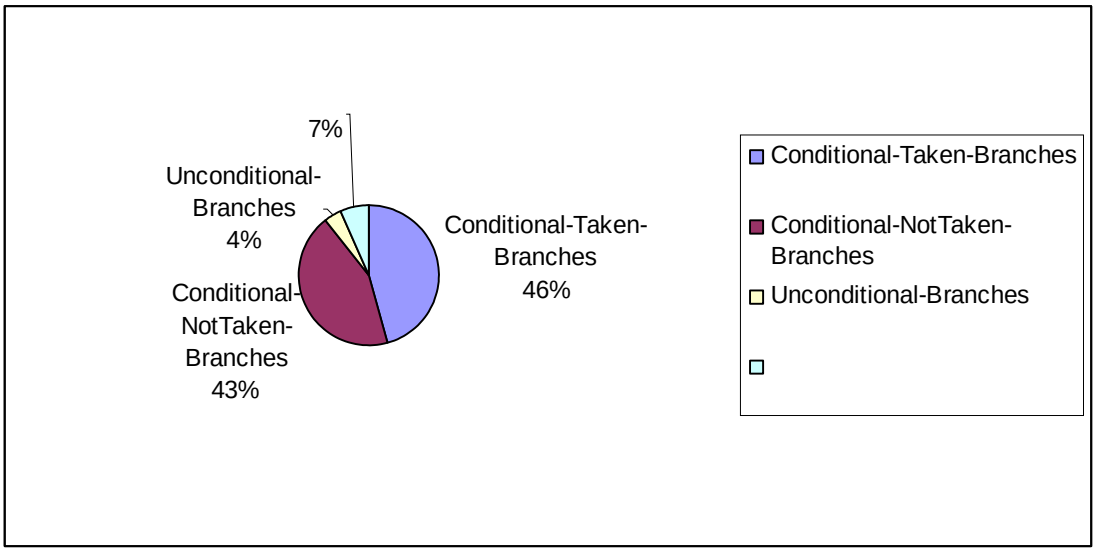
Στο πρώτο κομμάτι της πειραματικής αξιολόγησης ο στόχος είναι η συλλογή στατιστικών για τις εντολές άλματος που εκτελούνται από τα benchmarks. Χρησιμοποίησα το cslab_branch_stats.cpp και ακολουθούν τα διαγράμματα που δημιούργησα για κάθε benchmark που δείχνει τον αριθμό των εντολών άλματος που εκτελέστηκαν και το ποσοστό αυτών που ανήκουν σε κάθε κατηγορία (conditional-taken, conditional-nottaken κλπ.).



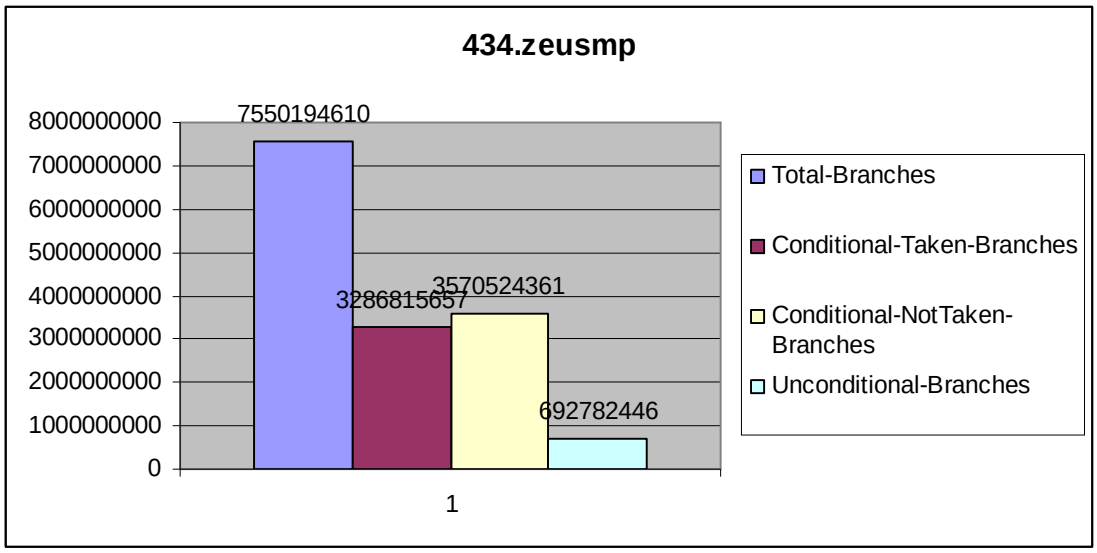


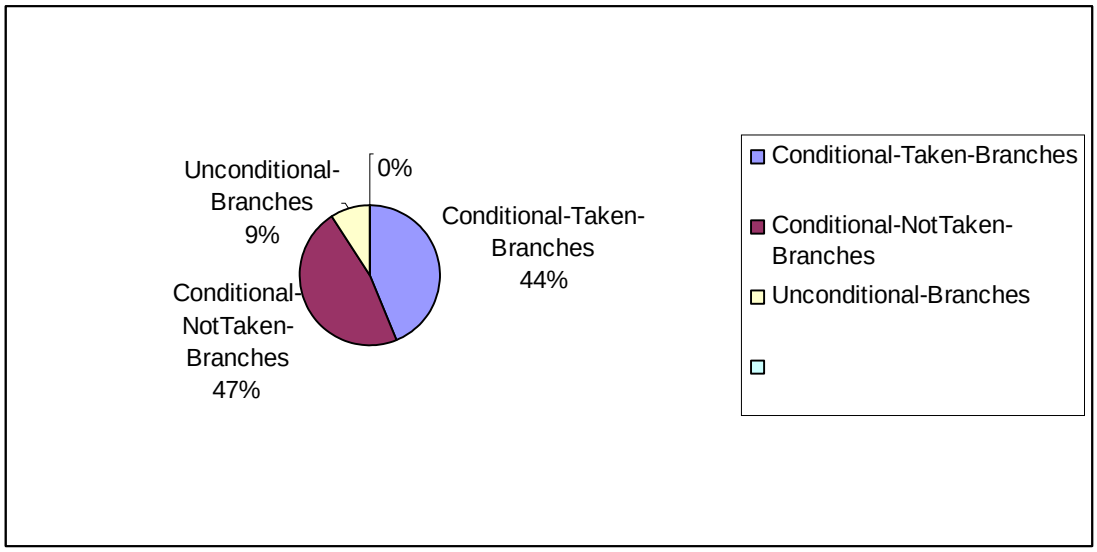
Παρατηρούμε ότι έχουμε διπλάσια Conditional Not Taken άλματα σε σχέση με τα Conditional Taken.



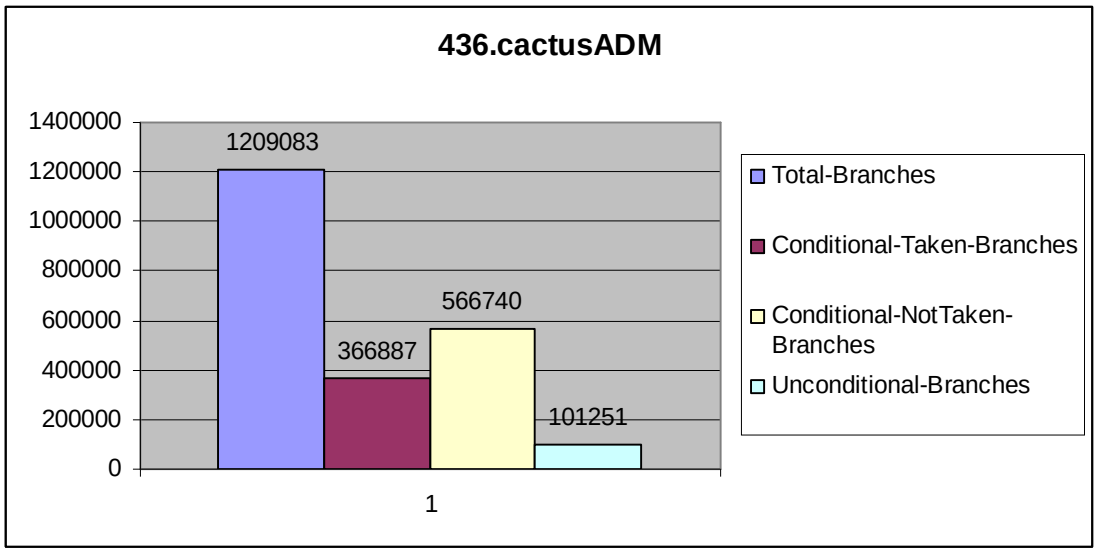


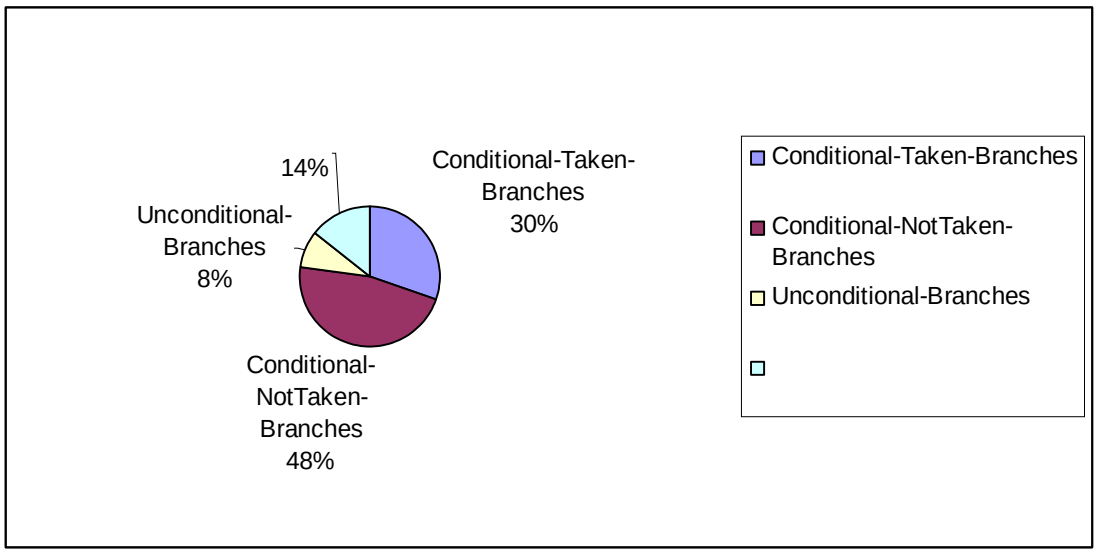
Παρατηρούμε πως έχουμε παραπλήσια ποσοστά Taken και Not Taken Conditional Branches.



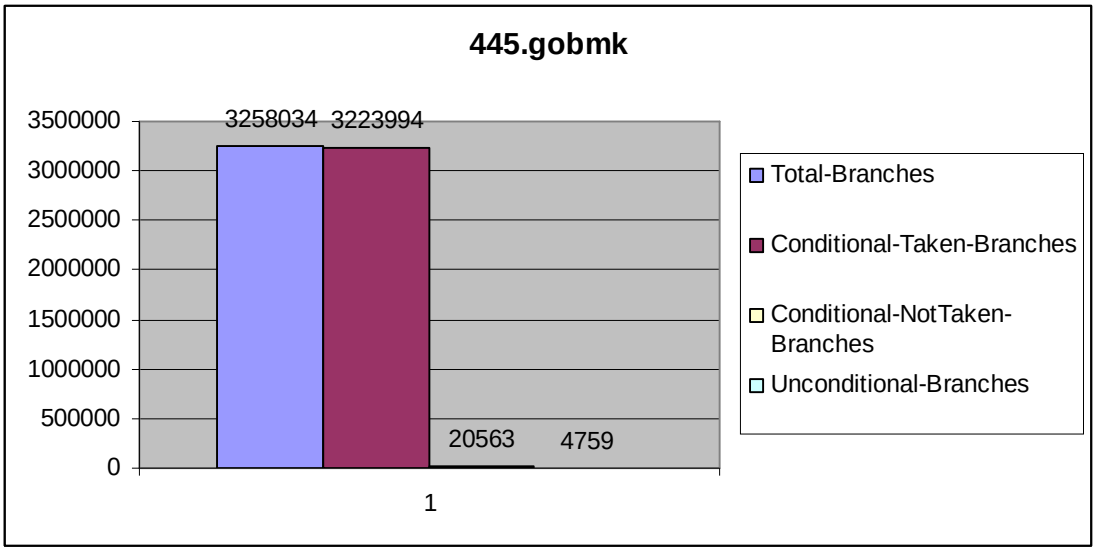


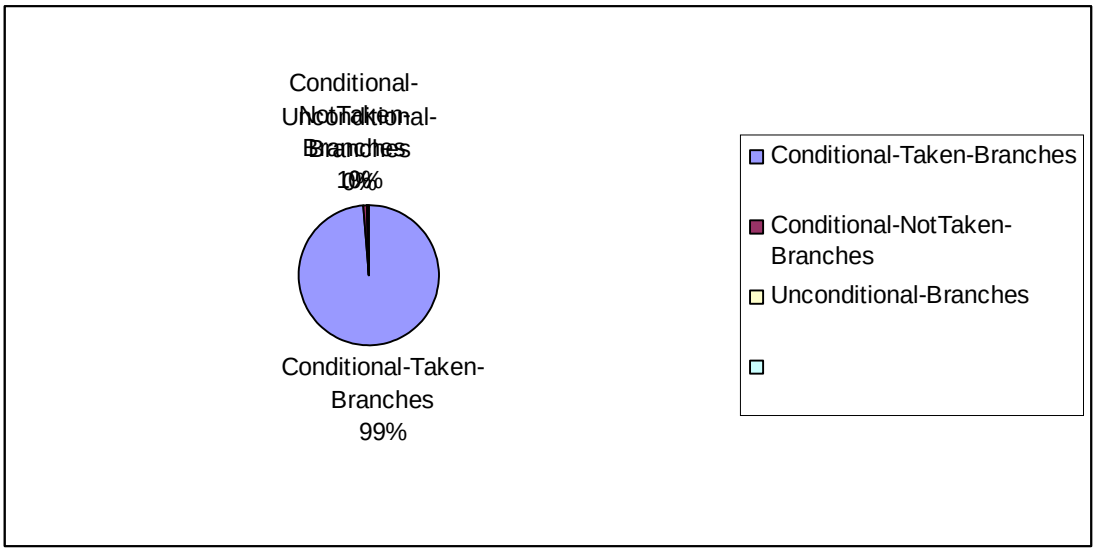
Ομοίως με προηγουμένως παρατηρούμε ότι το ποσοστό των Taken είναι πολύ κοντά με το ποσοστό των Conditional Branches που είναι Not Taken.



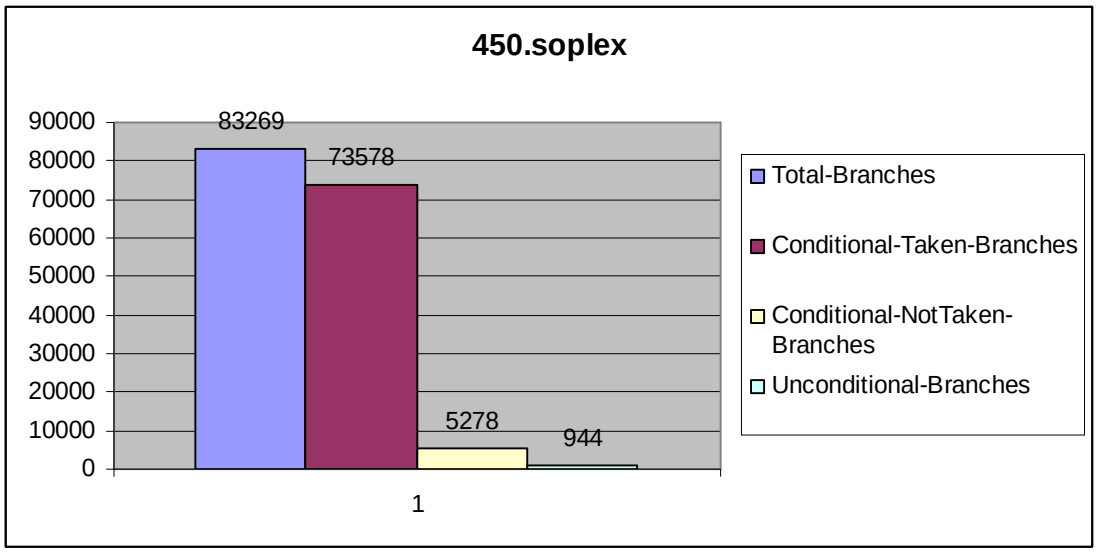


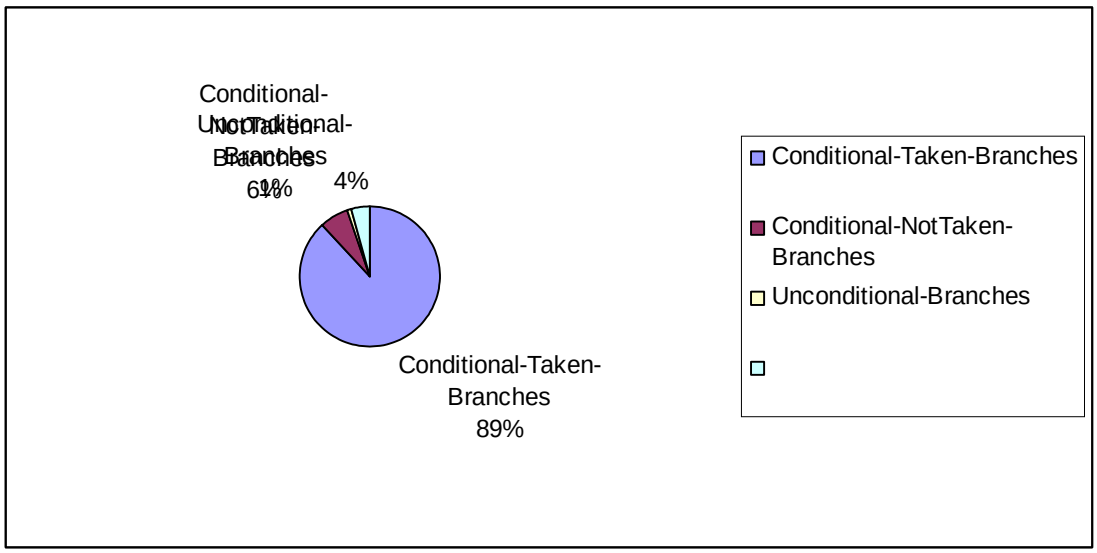
Σε αυτή την περίπτωση παρατηρούμε σημαντικά μεγαλύτερο ποσοστό Conditional Not Taken Branches σε σχέση με τα Conditional Taken (48% > 30%).



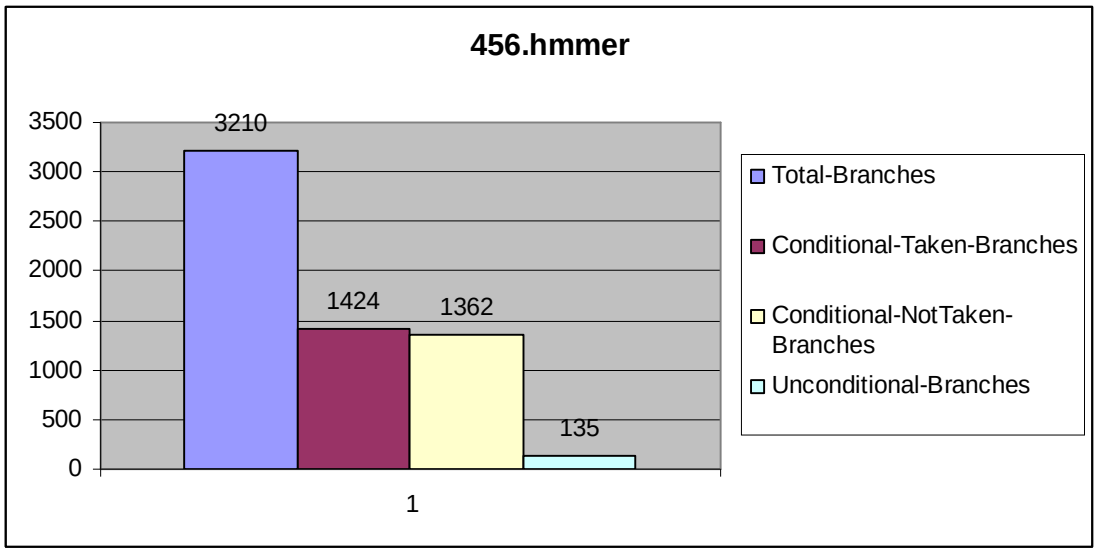


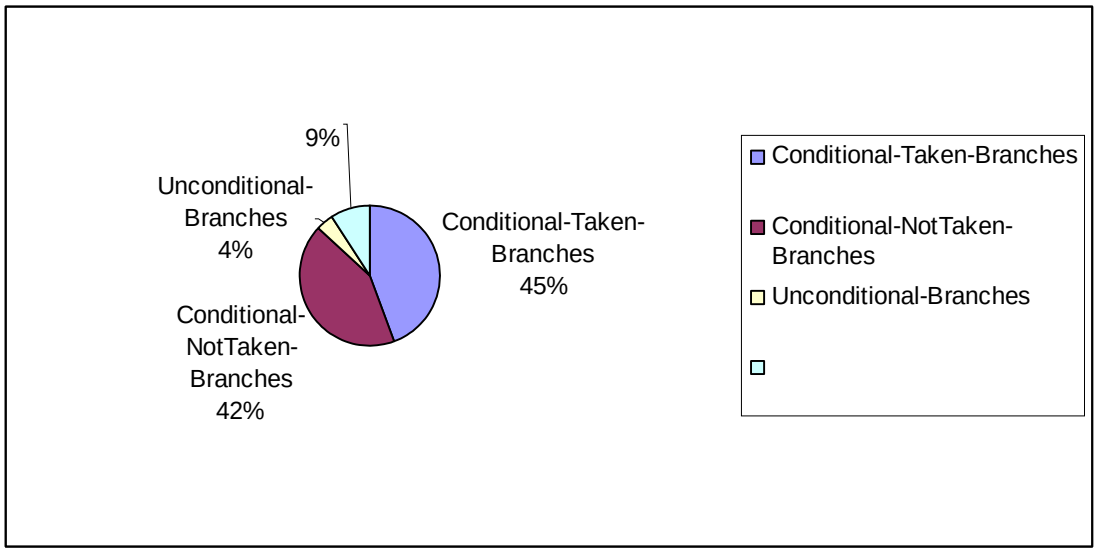
Στο gobmk παρατηρούμε πως σχεδόν όλα τα Branches είναι Conditional Taken. (Θα μπορούσε να πρόκειται για κάποιο κώδικα με loops πολλών επαναλήψεων.)



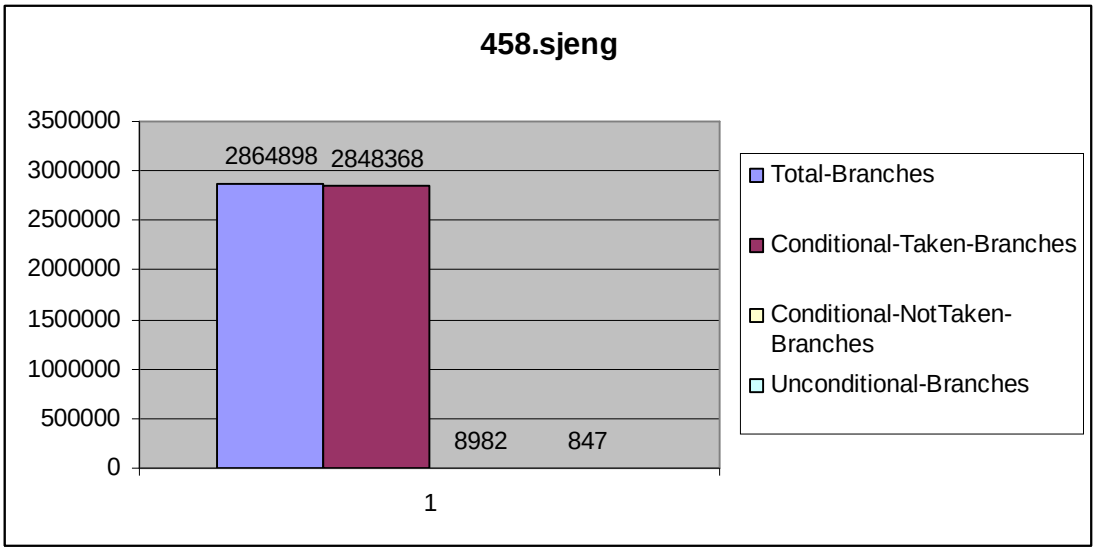


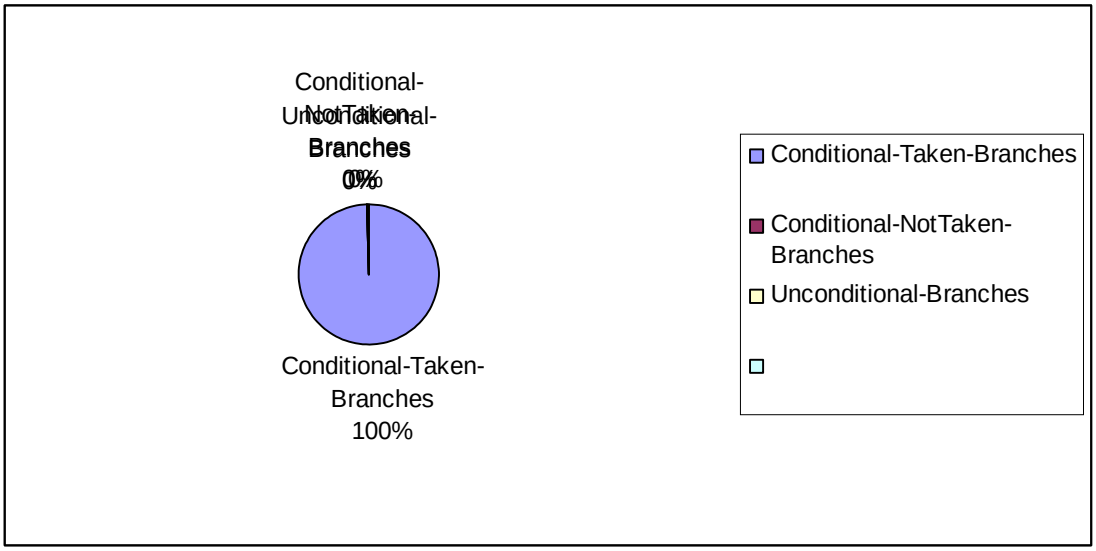
Και πάλι παρατηρούμε ότι ένα πολύ μεγάλο ποσοστό των Branches, είναι Conditional Taken.



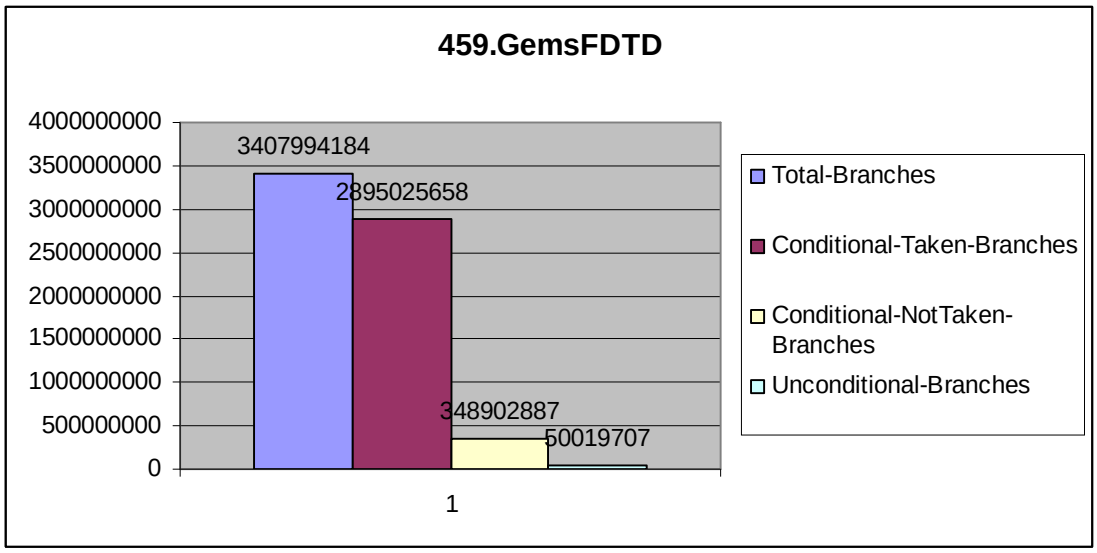


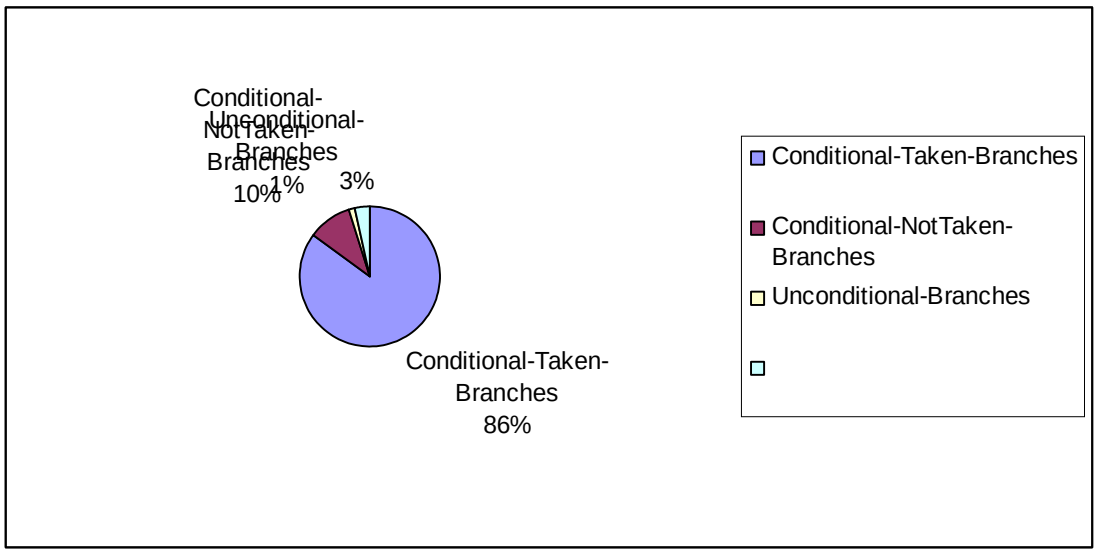
Στο hmmer παρατηρούμε ότι τα Taken και Not Taken Branches είναι πολύ κοντά σε πλήθος.



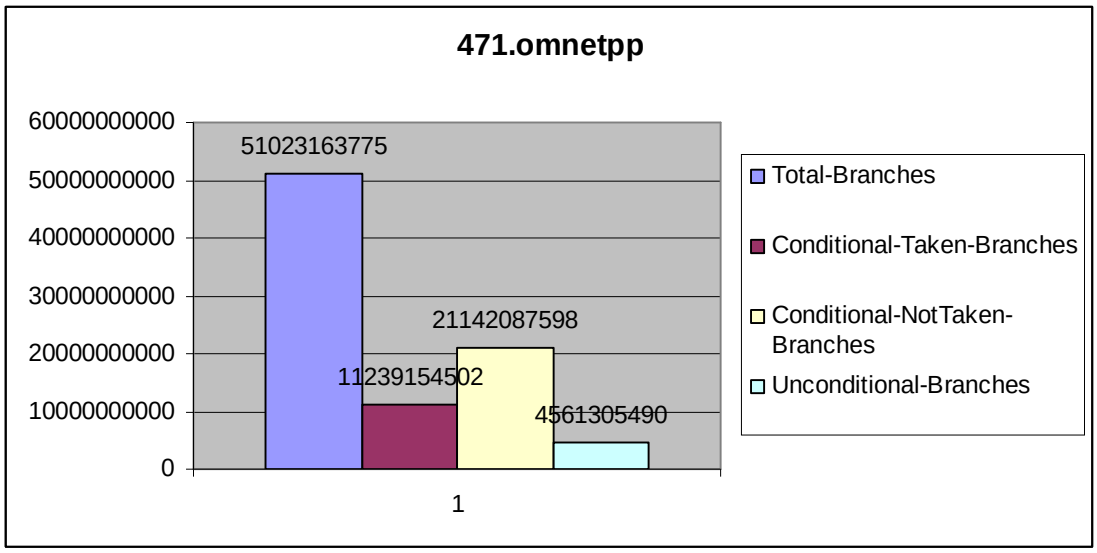


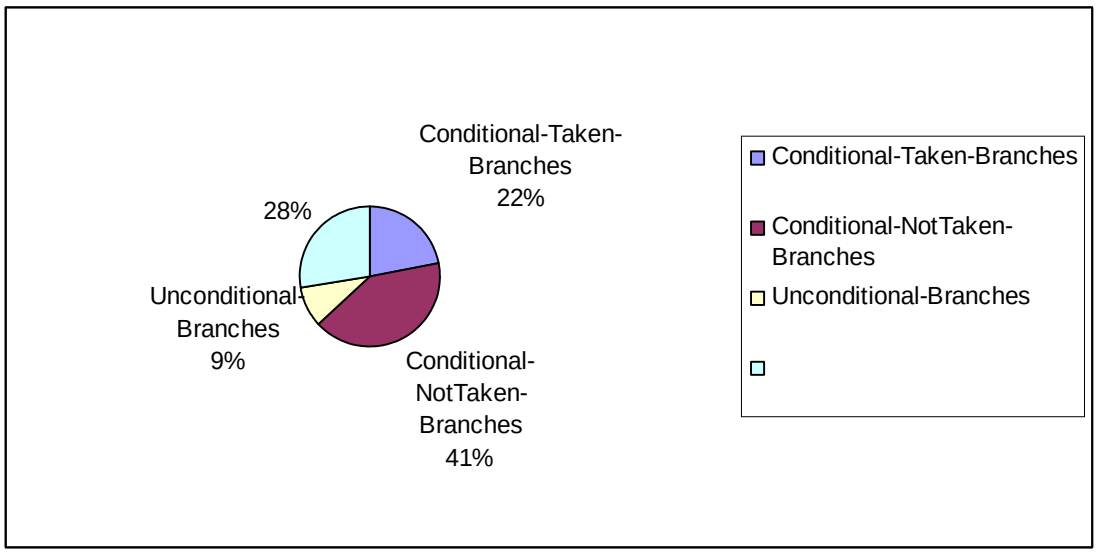
Όπως και στο gobmk, έτσι και στο sjeng παρατηρούμε πως σχεδόν όλα τα Branches είναι Conditional Taken. Σε τέτοιας μορφής κώδικες ένας απλούστατος Branch Predictor που θεωρεί όλα τα branches taken θα είχε πολύ καλά αποτελέσματα.



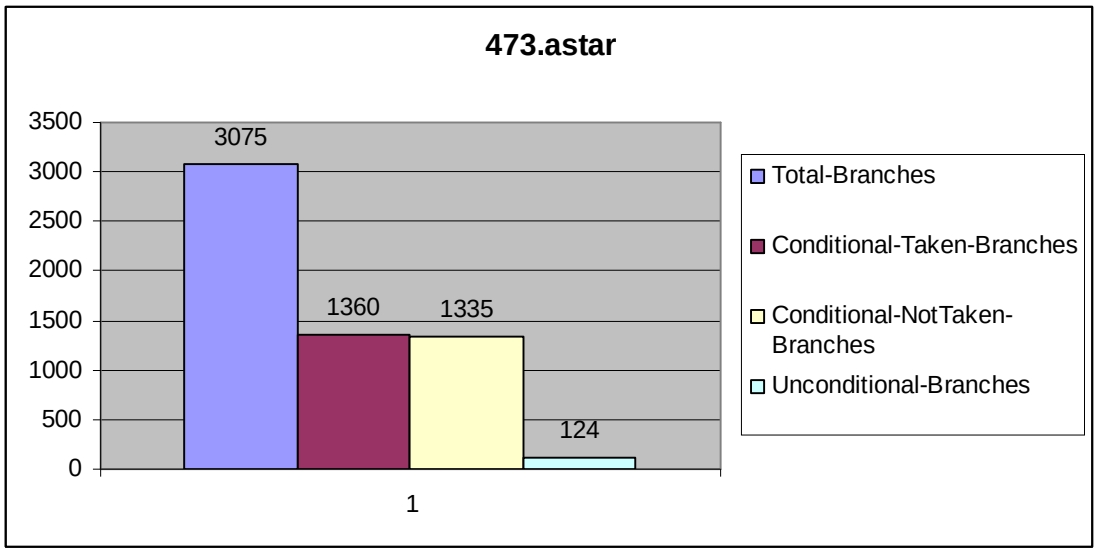


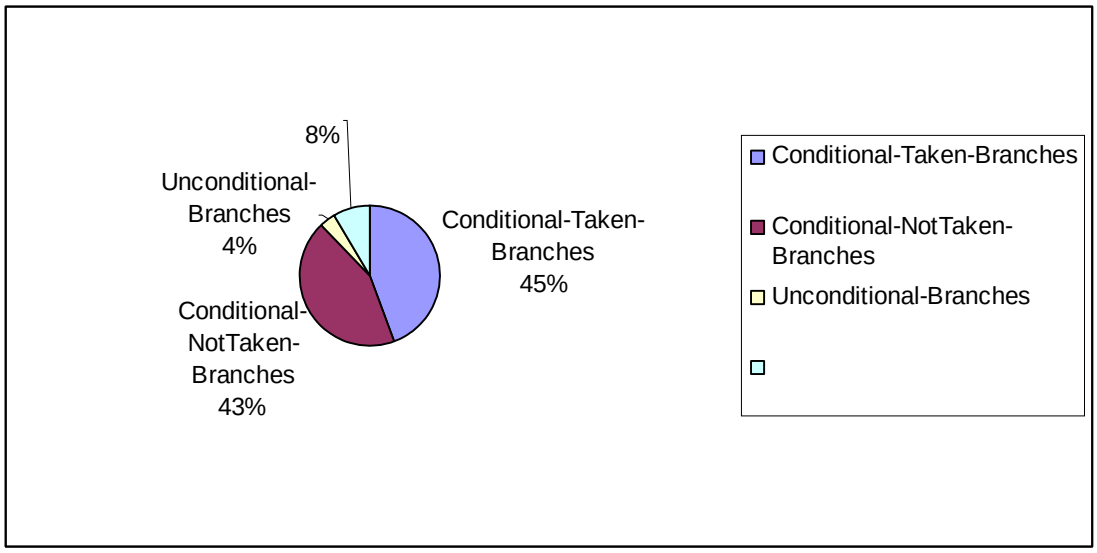
Και σε αυτή την περίπτωση όπως και στο soplex, έχουμε ένα πολύ μεγάλο ποσοστό των Branches (>85%) να είναι Conditional Taken.



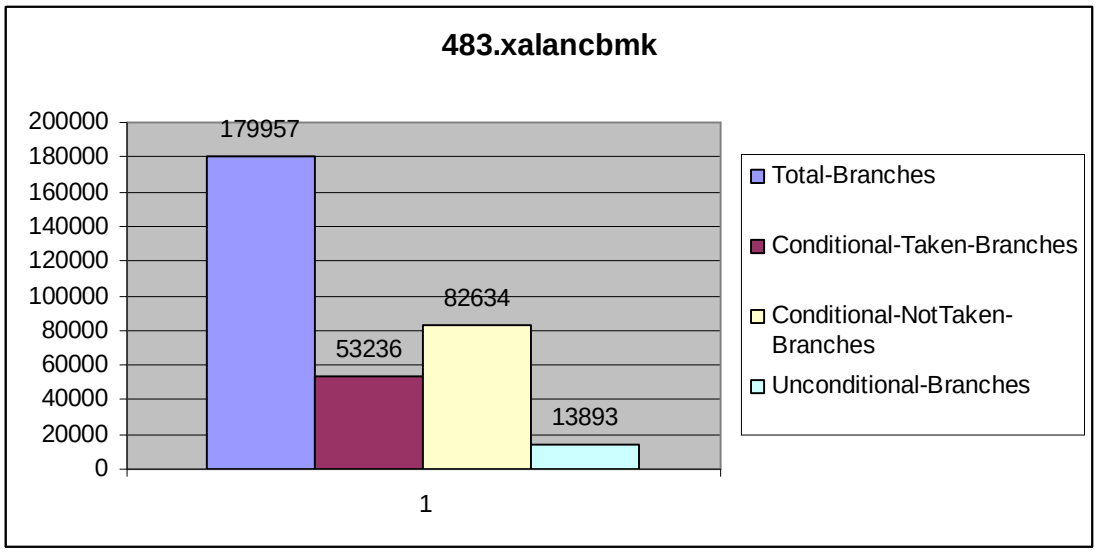


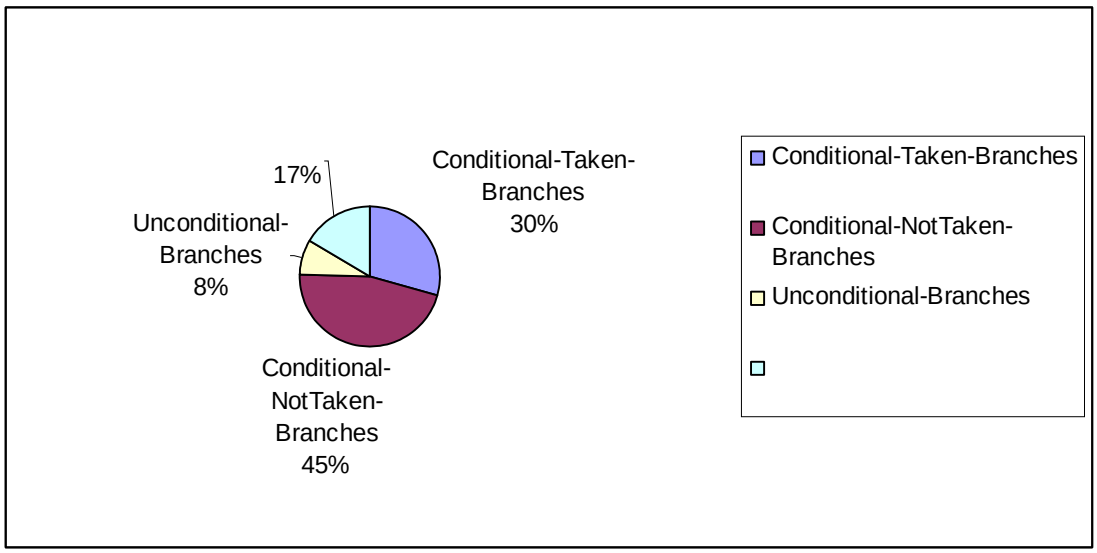
Εδώ παρατηρούμε πως τα Not Taken Conditional Branches είναι περίπου διπλάσια από τα Taken.





Στο astar παρατηρούμε πως τα Conditional Branches είναι σχεδόν διαμοιρασθεί σχεδόν ισόποσα σε Taken και Not Taken.

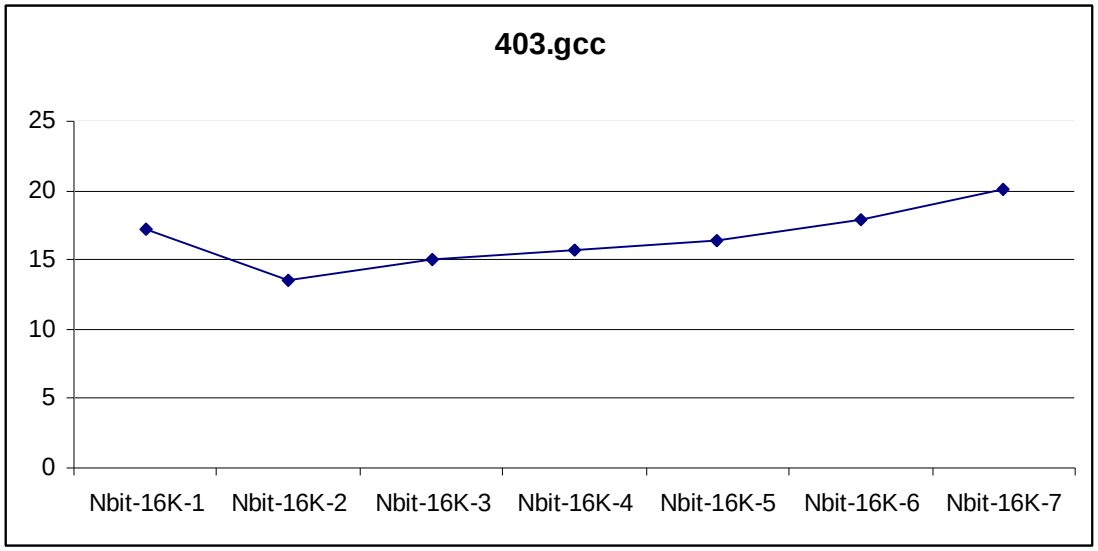




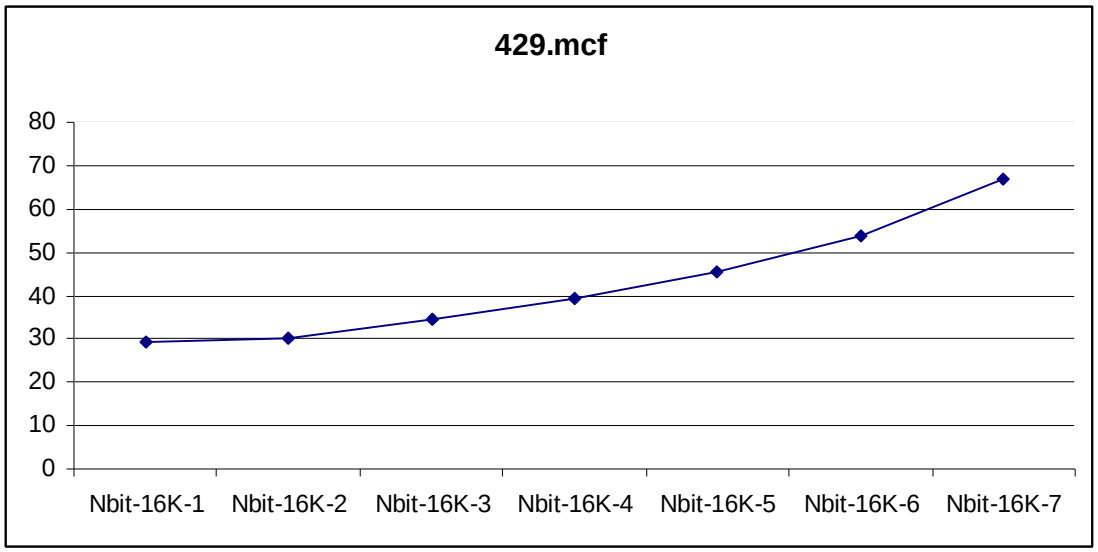
Τέλος στο xalancbmk παρατηρούμε πως τα Not Taken Conditional Branches είναι κατά μισή φορά περισσότερα από τα Taken. (45%-30%)

4.2 Μελέτη των N-bit predictors

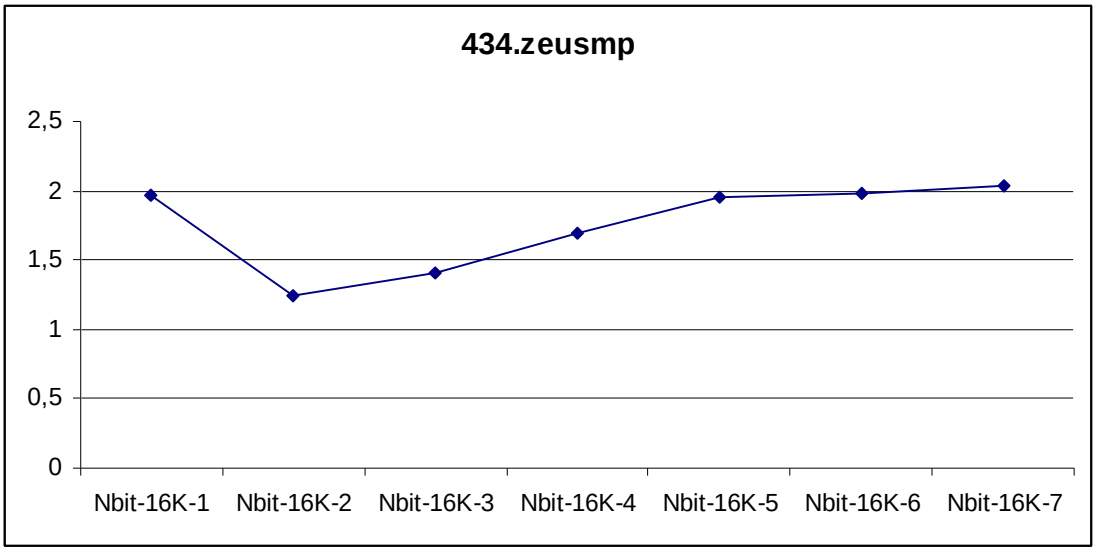
(i) Διατηρώντας σταθερό τον αριθμό των BHT entries και ίσο με 16K, προσομοίωσα τους n-bit predictors, για N=1, 2, …, 7. Στη συνέχεια παρουσιάζονται τα αποτελέσματα σε μορφή γραφημάτων, και η σύγκριση των διαφόρων predictors γίνεται με βάση την τιμή του direction Mispredictions Per Thousand Instructions (direction MPKI).



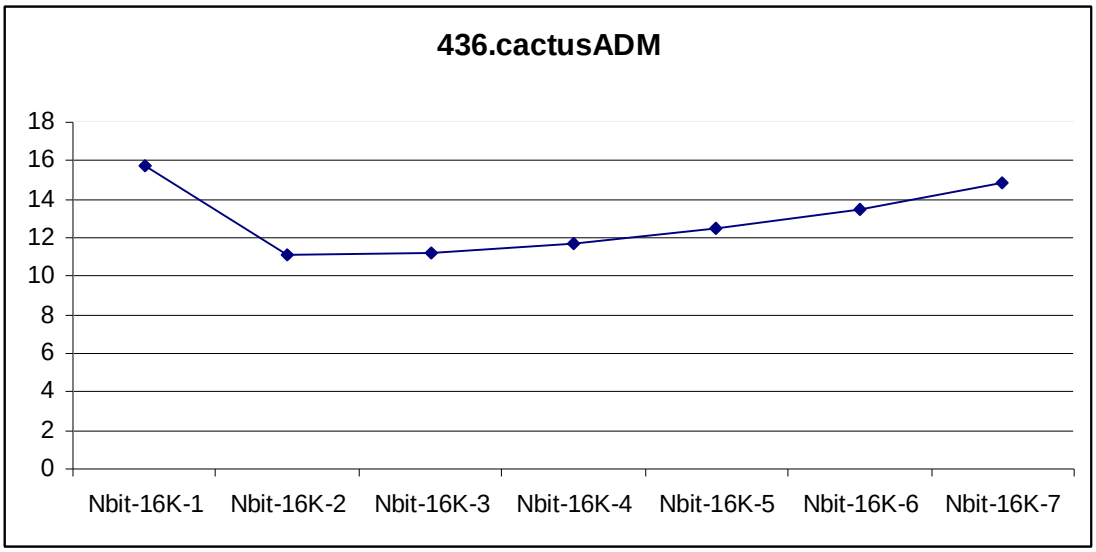
Παρατηρούμε πως η πιο ικανοποιητική τιμή για το MPKI προκύπτει με τον predictor των 2 bits. Από εκεί και πέρα ενώ η αύξηση των bits του predictor από 1 σε δύο μας έδωσε καλύτερο αποτέλεσμα, περεταίρω αύξηση των bits δίνει χειρότερα αποτελέσματα. Για αυτή την περίπτωση λοιπόν βέλτιστος φαίνεται να είναι ο 2-bit predictor μιας και σε αυτόν η γραφική μας παρουσιάζει ελάχιστο.



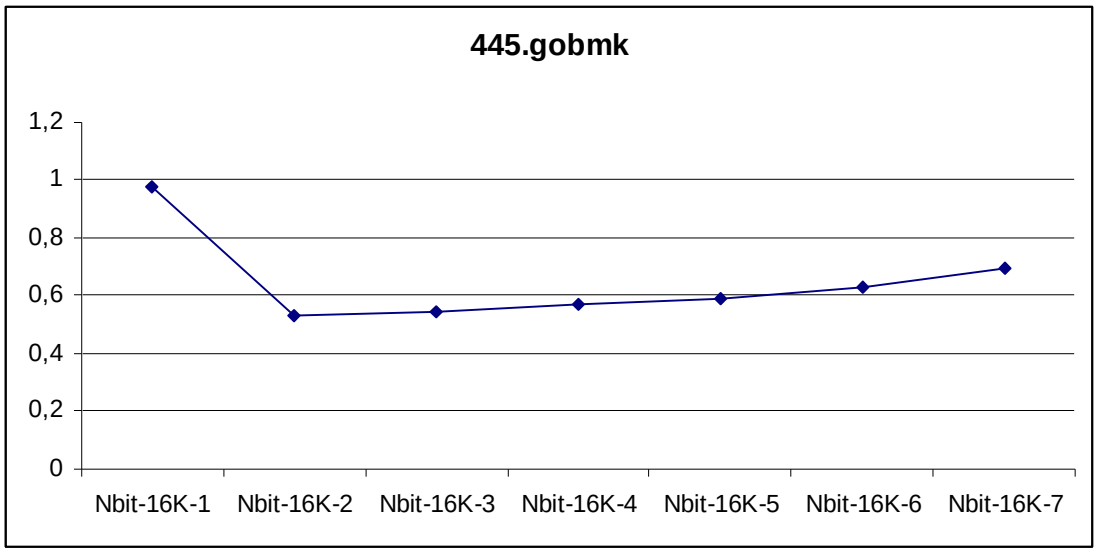
Σε αυτό το benchmark παρατηρούμε πως αύξηση του αριθμού των bits του predictor δίνει πάντα χειρότερο αποτέλεσμα. Βλέπουμε λοιπόν πως δεν ισχύει ότι το να δαπανήσουμε χρόνο και κόστος για να φτιάξουμε έναν πιο πολύπλοκο predictor με περισσότερα bits θα μας αποδώσει απαραίτητα καλύτερα. Εδώ έχουμε βέλτιστο predictor με 1 bit. Επίσης όμως παρατηρούμε ότι ακόμη και η βέλτιστη τιμή του MPKI είναι χειρότερη από τη μέγιστη του προηγούμενου παραδείγματος, άρα γενικότερα οι N-bit Predictors δεν δίνουν και τόσο ικανοποιητικά αποτελέσματα για τη συγκεκριμένη εφαρμογή.



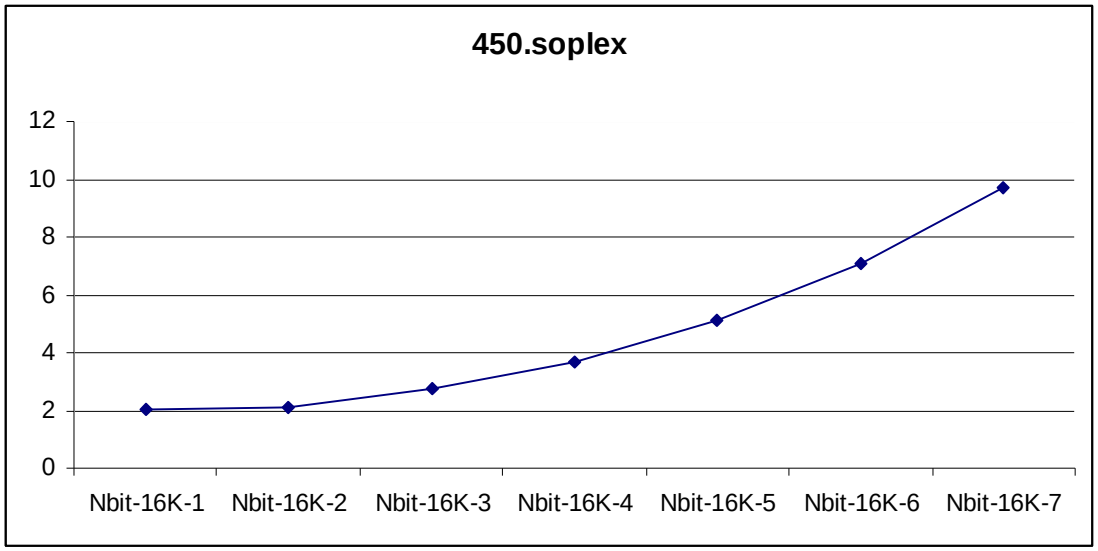
Κι εδώ έχουμε μια ανάλογη εικόνα με το πρώτο benchmark, αρχικά με αύξηση των bits έχουμε βελτίωση, αλλά με αύξηση των bits πέρα των 2 έχουμε χειρότερες τιμές. Όμως σε αυτή την περίπτωση έχουμε γενικότερα πολύ καλύτερες τιμές MPKI σε σύγκριση με τα προηγούμενα benchmarks. Βέλτιστος predictor ο 2-bit Predictor.



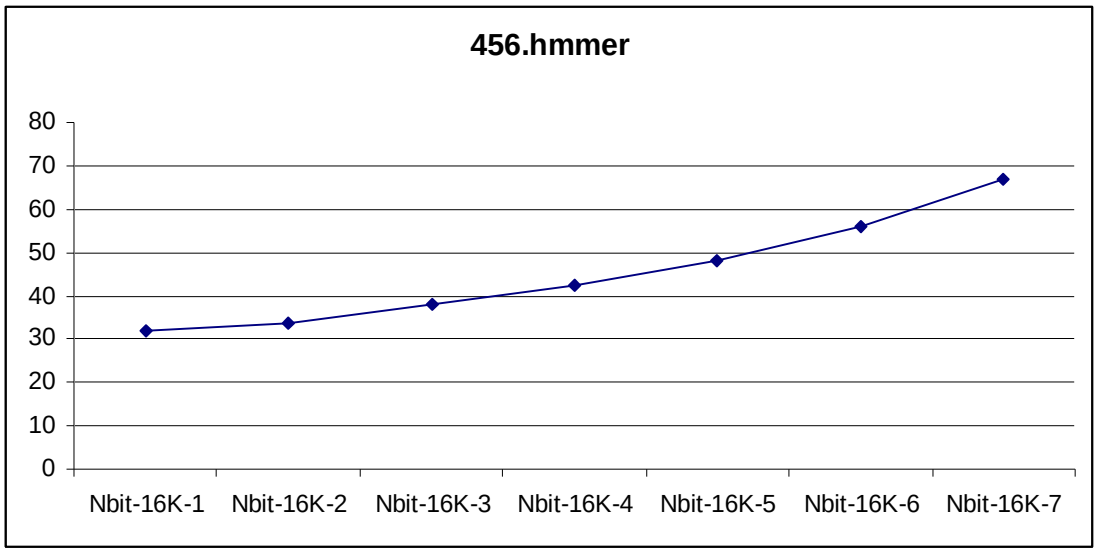
Και στο cactusADM έχουμε ανάλογη συμπεριφορά με προηγουμένως όμως με γενικότερα υψηλότερες τιμές του MPKI. Μόνο που σε αυτή την περίπτωση ο 3-bit predictor είναι πολύ κοντά στον 2-bit. Και πάλι όμως βέλτιστος είναι ο 2-bit.



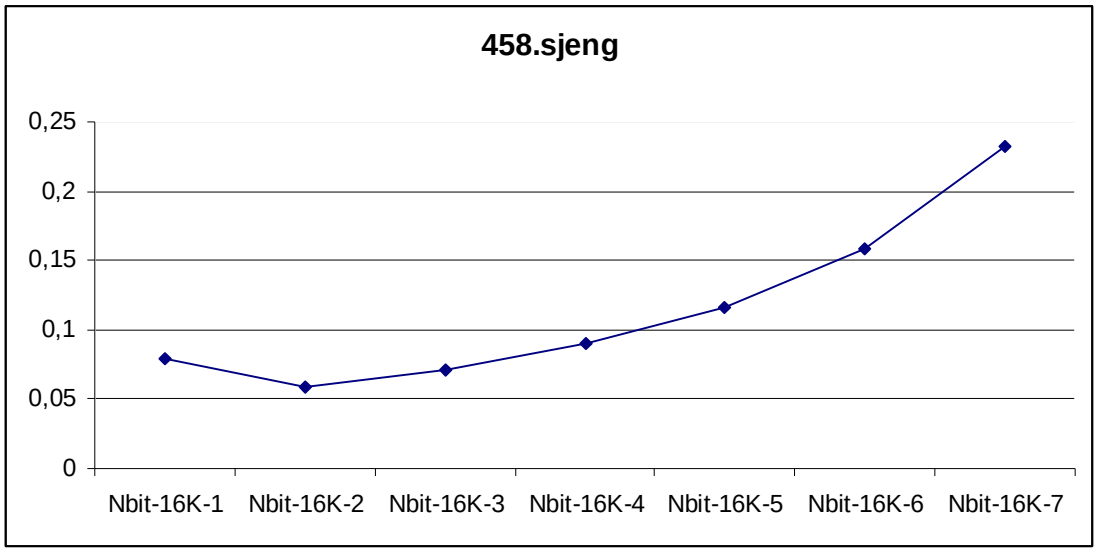
Και πάλι έχουμε ελάχιστο στον 2-bit predictor. Γενικότερα όμως στη συγκεκριμένη εφαρμογή όλοι οι N-bit predictors δίνουν ικανοποιητικά αποτελέσματα μιας και το MPKI για όλους είναι κάτω του 1.



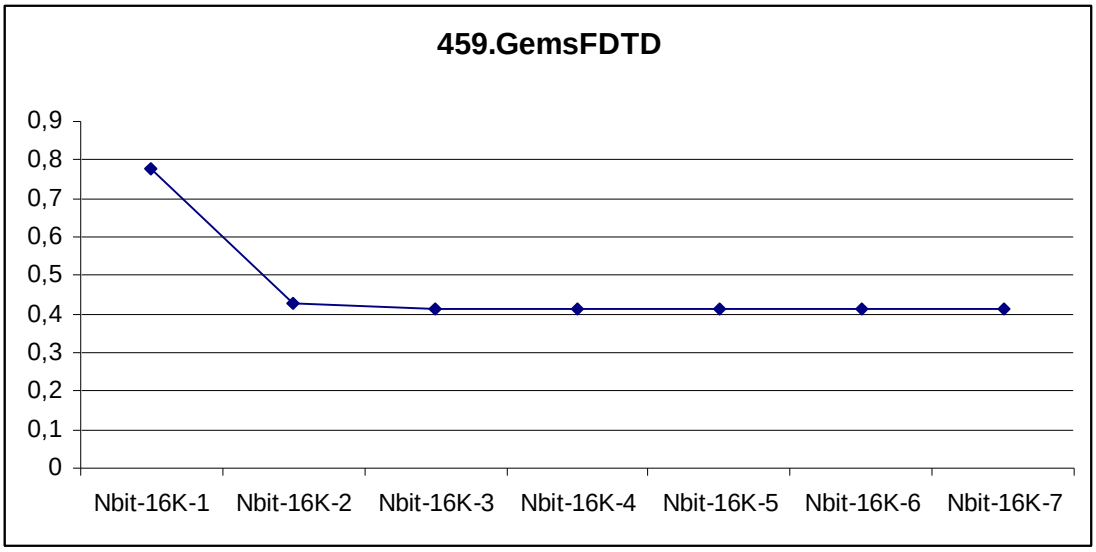
Στο soplex παρατηρούμε μια αύξηση της τιμής του MPKI μαζί με την αύξηση των bits του predictor. Έχουμε λοιπόν βέλτιστα αποτελέσματα με τον 1-bit predictor.



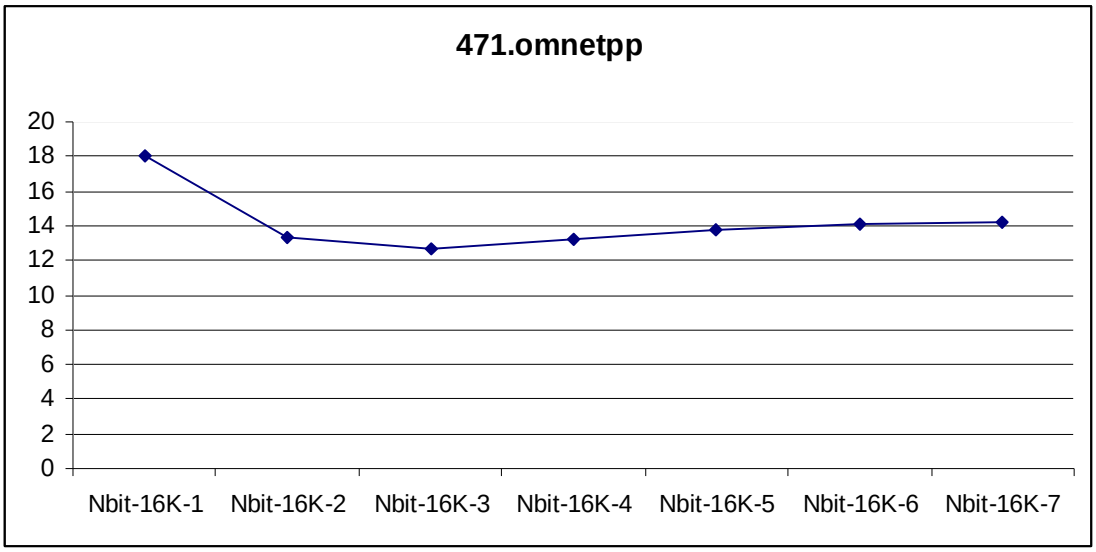
Ομοίως με πριν έχουμε βέλτιστα αποτελέσματα με 1-bit predictor.



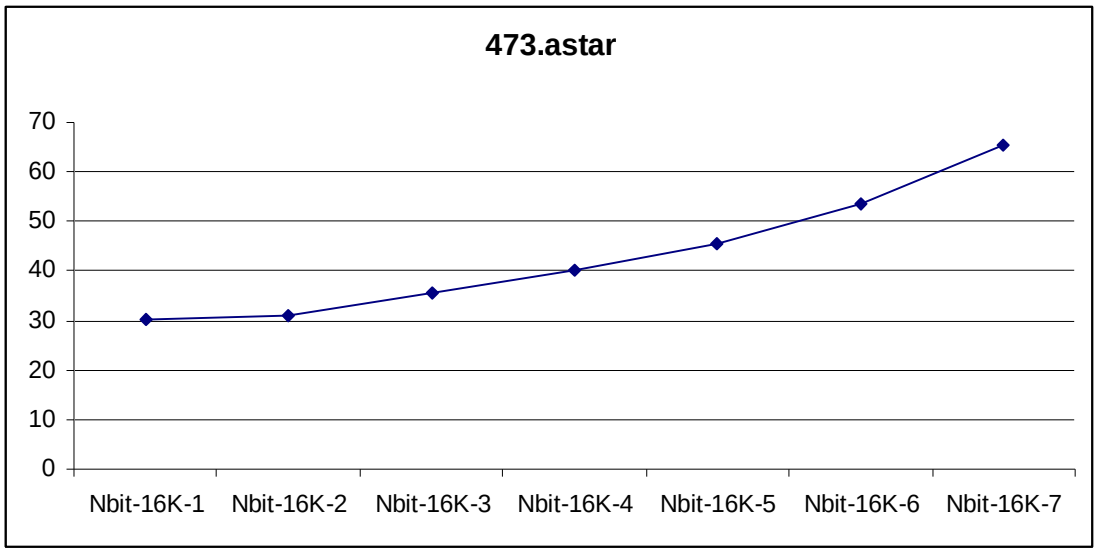
Και στην εφαρμογή sjeng παρατηρούμε βέλτιστα αποτελέσματα με τον 2-bit predictor, όμως γενικότερα έχουμε εξαιρετικά αποτελέσματα μιας και όλοι οι predictors δίνουν MPKI κάτω του 0,25.



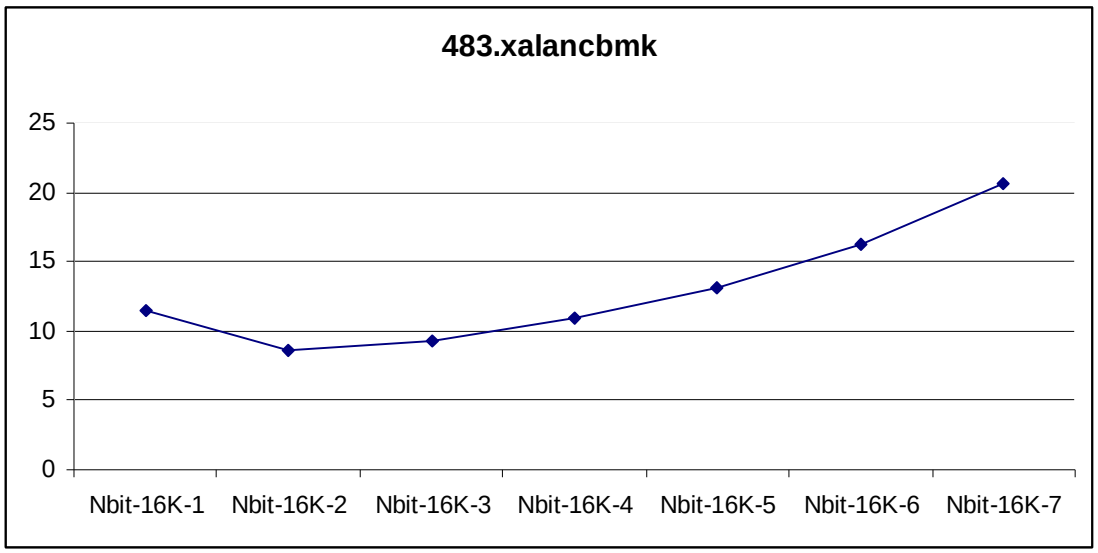
Στο συγκεκριμένο benchmark παρατηρούμε μια βελτίωση του MPKI με την αύξηση των bits του predictor, όμως από τα 2 ή 3 bits και μετά η βελτίωση του MPKI είναι αμελητέα. Βέλτιστα αποτελέσματα δίνει ο 7-bit predictor όμως με τον 3-bit predictor να είναι πρακτικά στο ίδιο επίπεδο, όπως και όλοι οι ενδιάμεσοι. (Και πάλι παρατηρούμε πολύ ικανοποιητικά αποτελέσματα όλων των Bit Predictors μιας και όλοι δίνουν MPKI κάτω του 0,8.)



Σε αυτή την περίπτωση παρατηρούμε ότι η γραφική μας έχει ελάχιστο για τον 3-bit predictor. Αρχικά έχουμε σημαντική βελτίωση με την αύξηση των bits του predictor από 1 σε 2, μετά τα 3 όμως η αύξηση των bits οδηγεί και σε αύξηση των mis-predictions.



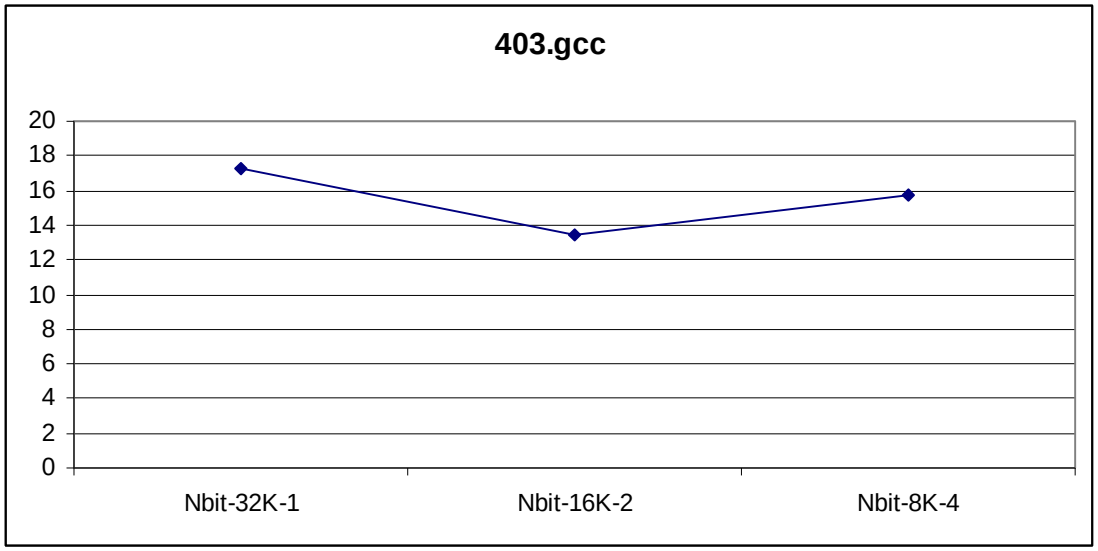
Στον astar έχουμε ακόμη μια φορά μια αύξουσα γραφική παράσταση. Έχουμε λοιπόν βέλτιστα αποτελέσματα για τον 1-bit predictor.



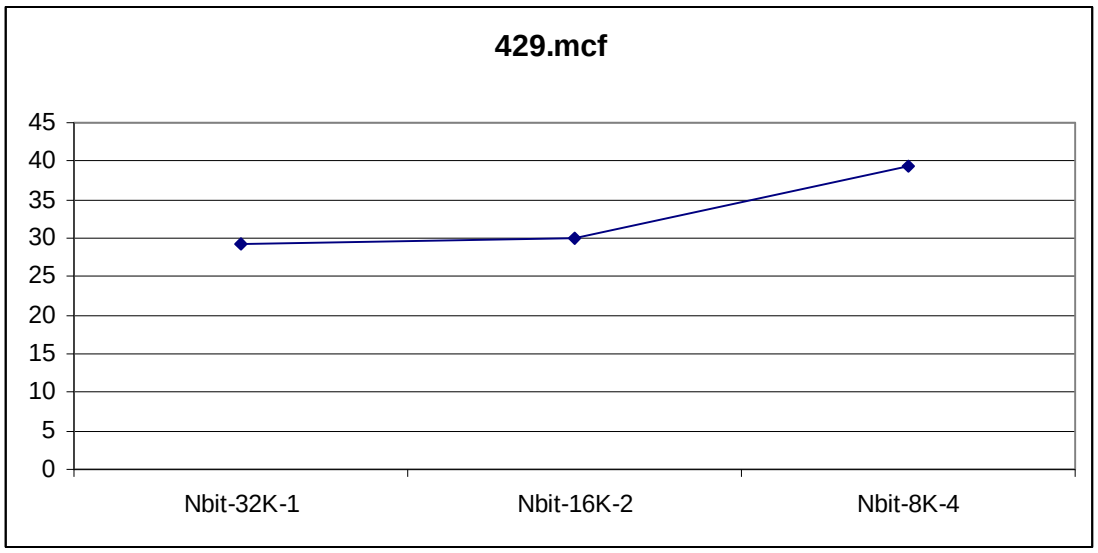
Και στο τελευταίο benchmark παρατηρούμε για πολλοστή φορά ότι η γραφική μας παράσταση παρουσιάζει ελάχιστο MPKI στον 2-bit predictor.

Συνοψίζοντας τα όσα παρατηρήσαμε μέχρι στιγμής η αύξηση των bits των predictors πέρα των 2-3 bits από μόνη της δεν μας δίνει καλύτερα αποτελέσματα σε ότι αφορά τα mis-predictions αντιθέτως που δίνει χειρότερα αποτελέσματα. Οι περισσότερες εφαρμογές είχαν βέλτιστα αποτελέσματα από τον 2-bit predictor, και αυτές που δεν είχαν βέλτιστο αποτέλεσμα σε αυτόν, δεν είχαν μεγάλη απόκλιση τα ελάχιστα mis-predictions από τα mis-predictions με τον 2-bit. Για αυτό το λόγο εάν έπρεπε να επιλέξουμε έναν βέλτιστο predictor από τους N-bit, αυτός θα ήταν ο 2-bit. Το γεγονός ότι καθώς αυξάνουμε τον αριθμό των bits αυξάνεται και το MPKI μπορεί να εξηγηθεί από το ότι τα loops που βρίσκονται μέσα στον κώδικα επειδή το καθένα πιθανώς να εκτελεί αρκετές επαναλήψεις, αυξάνει σημαντικά τον αριθμό σύμφωνα με τον οποίο γίνεται το prediction. Ως αποτέλεσμα έχουμε από ένα σημείο και μετά πρόβλεψη σταθερά taken, αφού ένα loop με μερικές επαναλήψεις εφόσον έχει συνεχόμενα taken branches αυξάνει το μετρητή με αποτέλεσμα να απαιτούνται αρκετά not taken ώστε να τον ξαναμειώσουν και να έχουμε πρόβλεψη not taken.

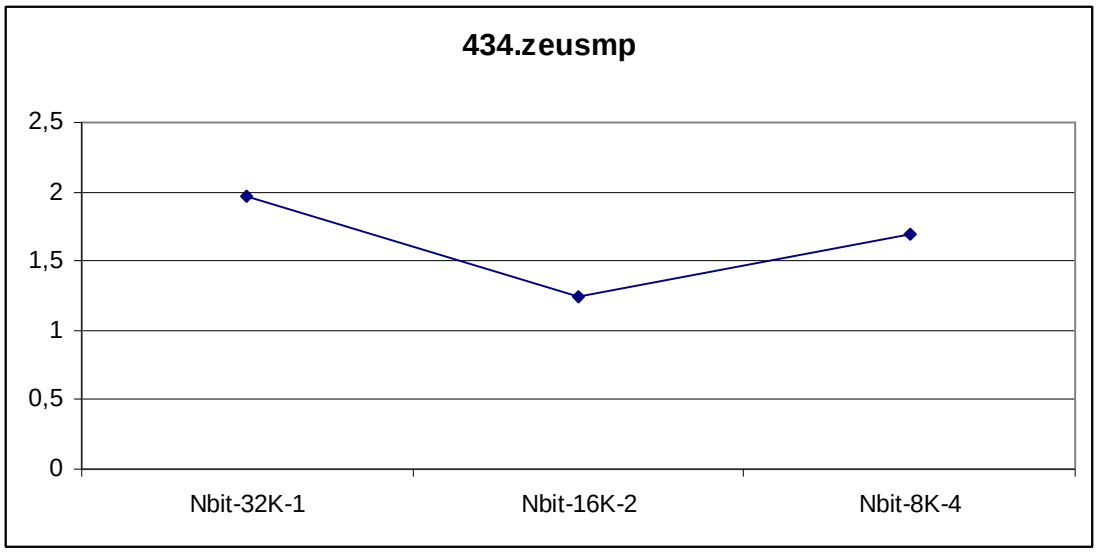
(ii) Στο προηγούμενο ερώτημα η αύξηση του αριθμού των bits ισοδυναμεί με αύξηση του απαιτούμενου hardware, αφού ο αριθμός των entries του BHT παραμένει σταθερός. Διατηρώντας τώρα σταθερό το hardware και ίσο με 32K bits, εκτέλεσα ξανά τις προσομοιώσεις για τα 12 benchmarks, θέτοντας N=1,2,4 και τον κατάλληλο αριθμό entries.



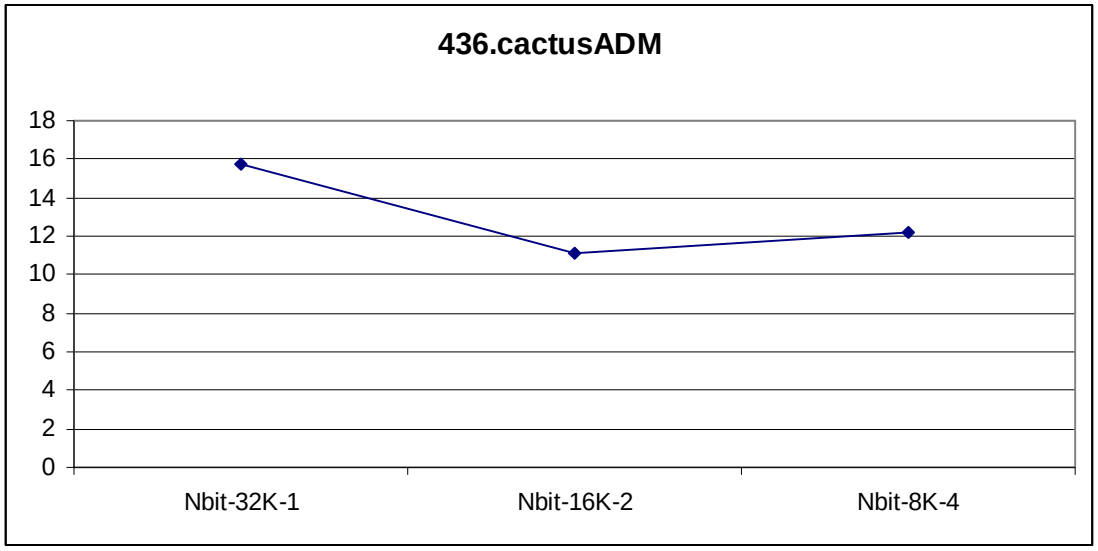
Στο gcc παρατηρούμε πως η τιμή του MPKI ελαχιστοποιείται για 2-bit predictor με 16K BHT entries.



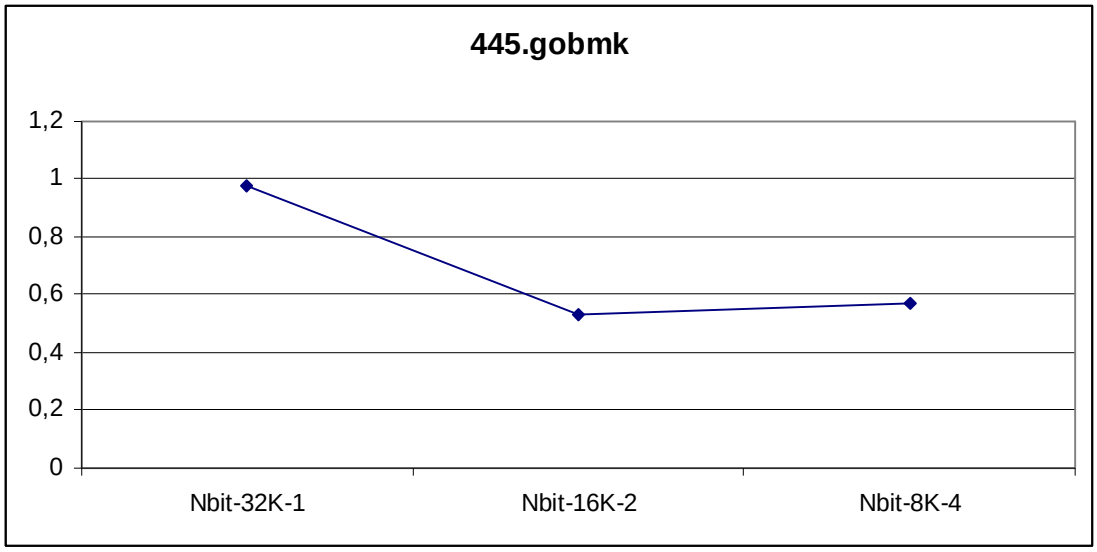
Στο mcf παρατηρούμε βέλτιστο MPKI για 1-bit predictor με 32K BHT entries, με τον 2-bit predictor με 16K BHT entries όμως να είναι επίσης πολύ κοντά.



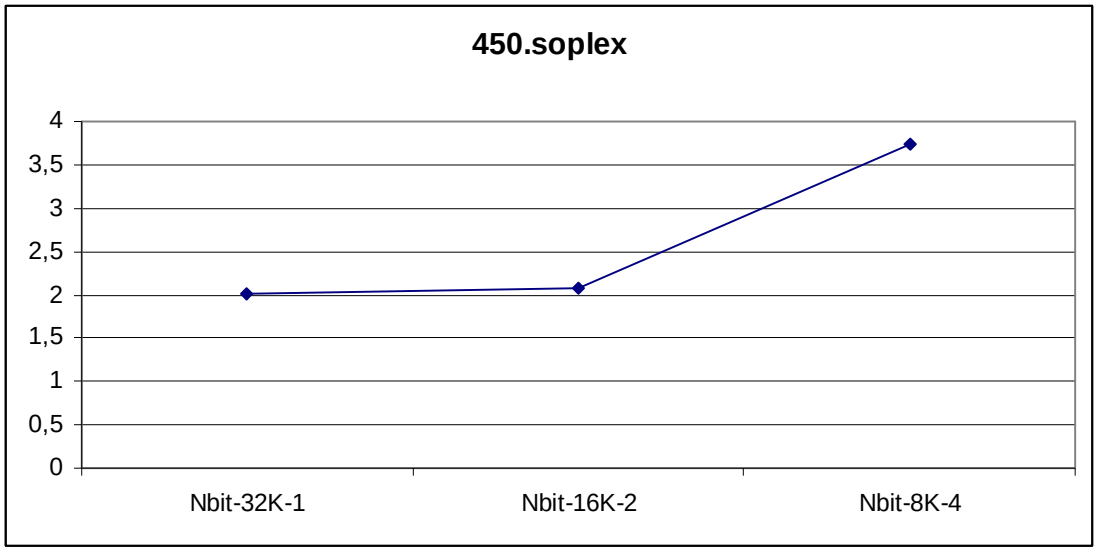
Και σε αυτή την περίπτωση έχουμε βέλτιστο αποτέλεσμα με τον 2-bit predictor με 16K BHT entries.



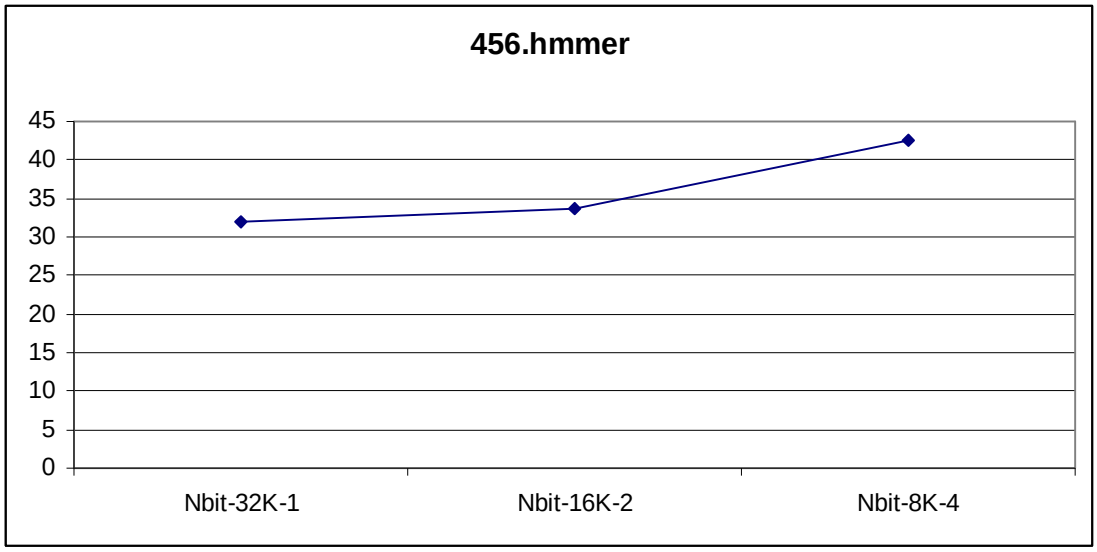
Ομοίως και σε αυτή την εφαρμογή, βέλτιστο αποτέλεσμα δίνει ο 2-bit predictor με 16K BHT entries.



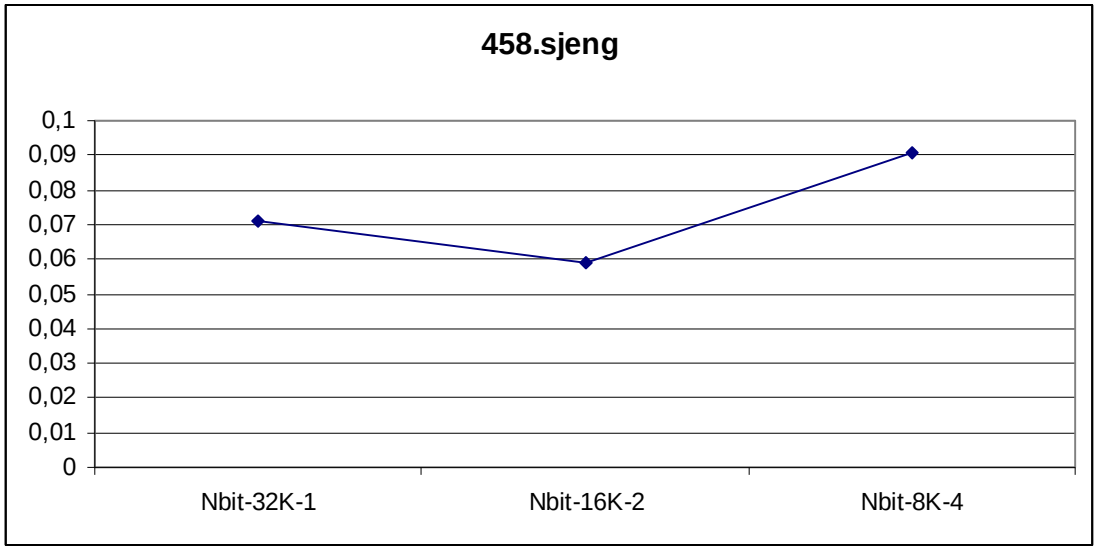
Και πάλι παρατηρούμε ελάχιστο στην τιμή που δίνει ο 2-bit predictor με 16K BHT entries, με την τιμή του MPKI από τον 4-bit predictor όμως να είναι αρκετά κοντά.



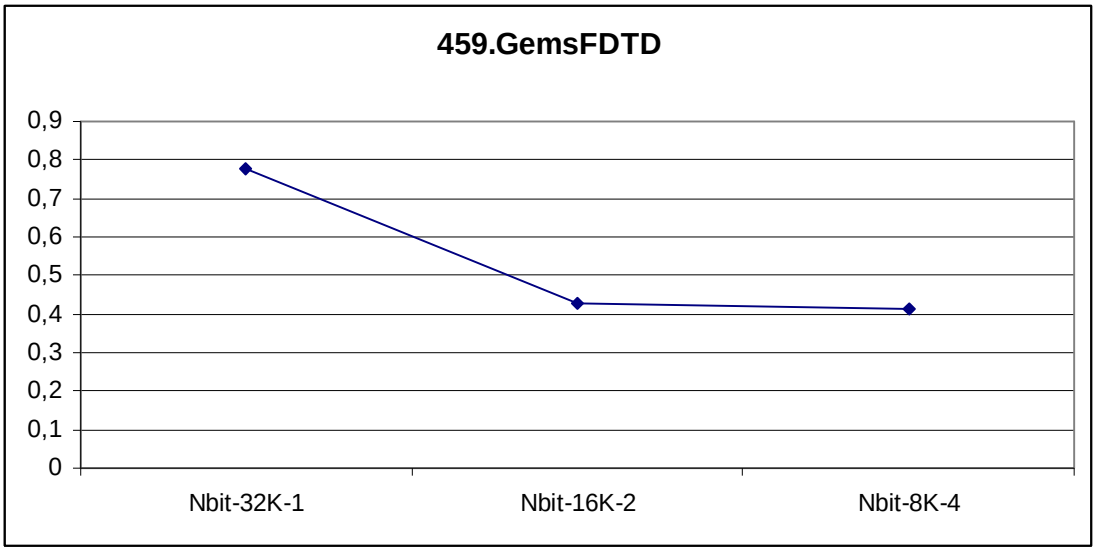
Ομοίως με το mcf παρατηρούμε βέλτιστο MPKI για 1-bit predictor με 32K BHT entries, με τον 2-bit predictor με 16K BHT entries όμως να είναι επίσης πολύ κοντά.



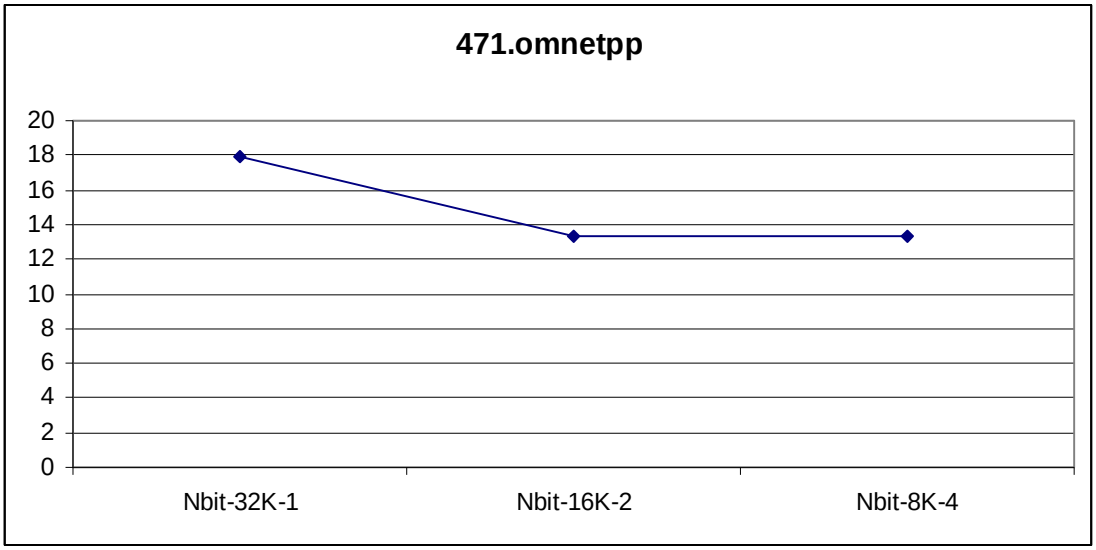
Σε αυτή την περίπτωση ο 1-bit predictor με 32K BHT entries δίνει βέλτιστο MPKI, με τον 2-bit predictor με 16K BHT entries όμως να είναι και πάλι σχετικά κοντά.



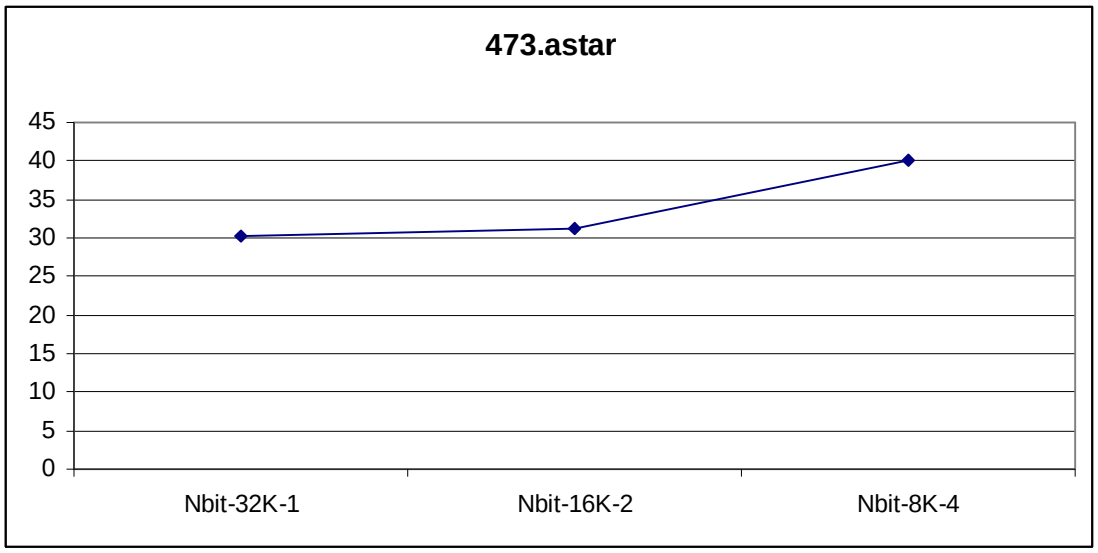
Για μια ακόμη φορά βέλτιστο αποτέλεσμα έχουμε με τον 2-bit predictor.



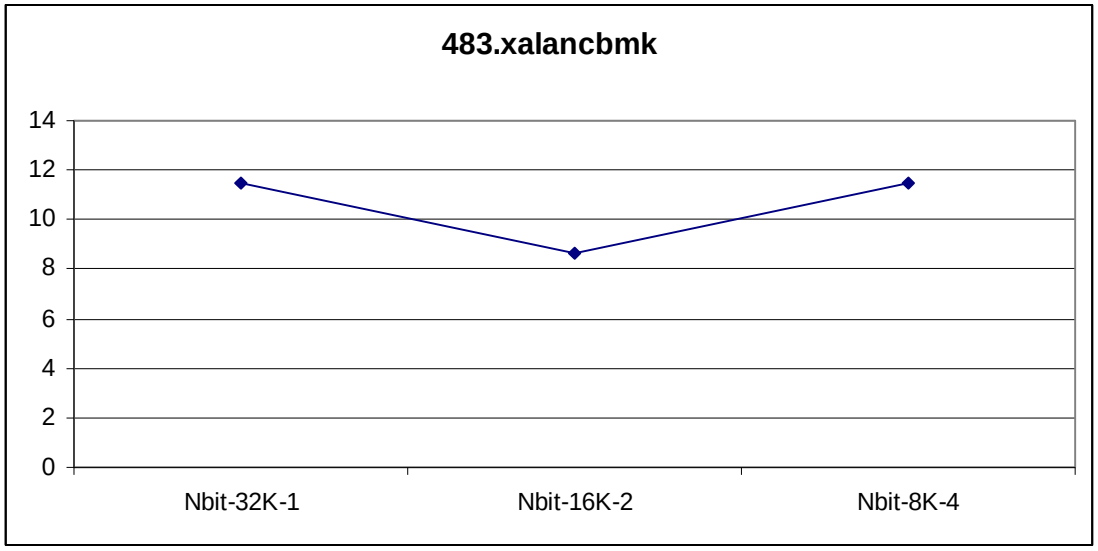
Σε αυτή την εφαρμογή βέλτιστο αποτέλεσμα φαίνεται να δίνει ο 4-bit predictor , με τον 2-bit όμως να δίνει και πάλι πολύ παραπλήσιες τιμές.



Στο omnetpp οι τιμές τους MPKI από τους 4-bit και 2-bit είναι βέλτιστες και σχεδόν ίσες.



Ο astar δίνει βέλτιστο αποτέλεσμα με τον 1-bit predictor, με τον 2-bit να βρίσκεται όπως πάντα πολύ κοντά.



Και στην τελευταία εφαρμογή έχουμε βέλτιστα αποτελέσματα με τον 2-bit predictor.

Σύμφωνα με όσα παρατηρήσαμε μέχρι στιγμής γίνεται ξεκάθαρο πως για τις εφαρμογές μας, αν και σε ορισμένες περιπτώσεις δεν δίνει το βέλτιστο αποτέλεσμα, στη γενική περίπτωση ο **2-bit predictor με 16K BHT entries** όχι μόνο δίνει βέλτιστο αποτέλεσμα στην πλειονότητα των εφαρμογών αλλά και σε αυτές που δεν δίνει βέλτιστο, δίνει ένα σχεδόν βέλτιστο, κατά πολύ λίγο χειρότερο του βέλτιστου. Παρατηρούμε λοιπόν πως δεν χρειάζεται να δαπανήσουμε πολλά σε hardware όπως είδαμε στο (i) αλλά να δώσουμε τόσο ένα περιθώριο προσαρμογής στον predictor μέσω των bits ώστε το αποτέλεσμα ενός άλματος να μην μπορεί να αλλάξει τον τρόπο αντιμετώπισης των επόμενων branches, όσο και μέσω των BHT entries να του δώσουμε επαρκή πληροφορία σχετικά με την πορεία των προηγούμενων branches, χωρίς βέβαια να θυσιάσουμε το 2^o bit για περισσότερα entries αφού όπως είδαμε από το ερώτημα (ii) δεν μας δίνει καλύτερα αποτελέσματα.

4.3 Μελέτη του BTB

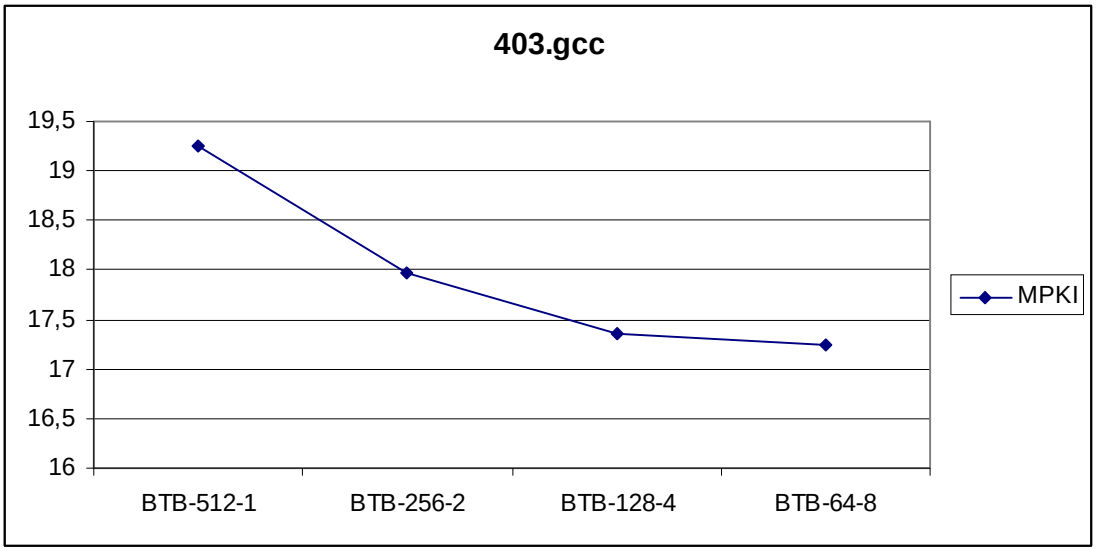
Υλοποίησα έναν BTB και να μελέτησα την ακρίβεια πρόβλεψής του μέσω των τιμών του MPKI και του target MPKI για τις ακόλουθες περιπτώσεις:

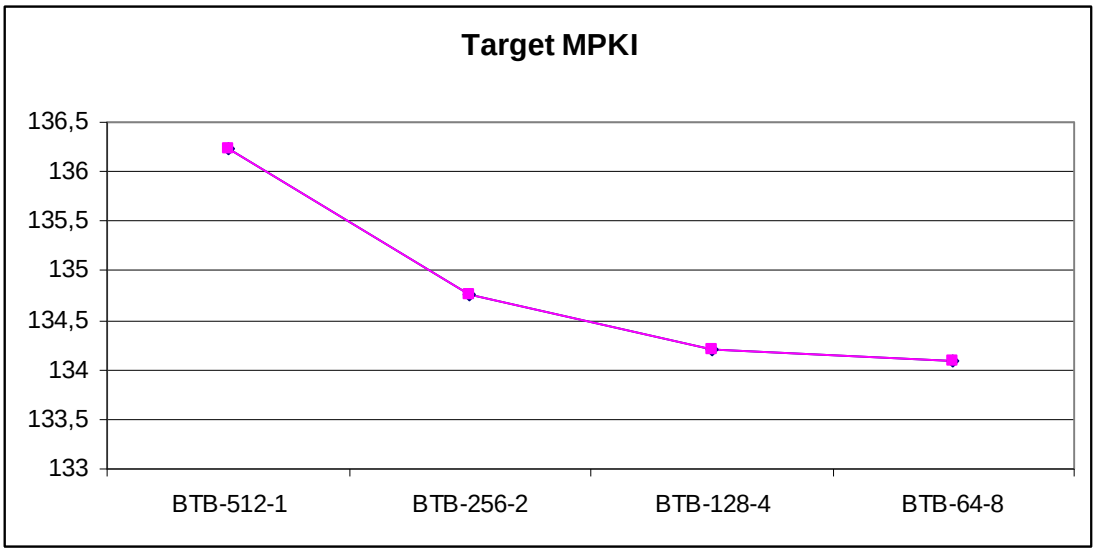
btb entries	btb associativity
512	1
256	2
128	4
64	8

Σημειώνεται ότι ο υπολογισμός του target MPKI που έκανα δεν είναι ακριβώς ορθός μιας και υπολογίζω το target MPKI σύμφωνα με τα Target correct, αφαιρώντας τα από τα συνολικά branches (άθροισμα των direction correct και incorrect) όμως έχουμε target mispredictions μετά από direction hit, άρα η τιμή των target mispredictions θα ήταν ορθότερο να υπολογιστεί αφαιρώντας τα target correct από τα direction correct. Παρά ταύτα η γραφική παράσταση που προκύπτει είναι ένα αντιπροσωπευτικό δείγμα για να μελετήσουμε την απόδοση κάθε περίπτωσης του BTB κι ας είναι πλασματικές οι τιμές στους άξονες των γραφικών του Target MPKI, μιας κι αυτό που μελετάμε εμείς δεν είναι οι ακριβείς τιμές των MPKI αλλά η σύγκρισή τους ανάλογα με το συνδυασμό BTB entries και associativity.

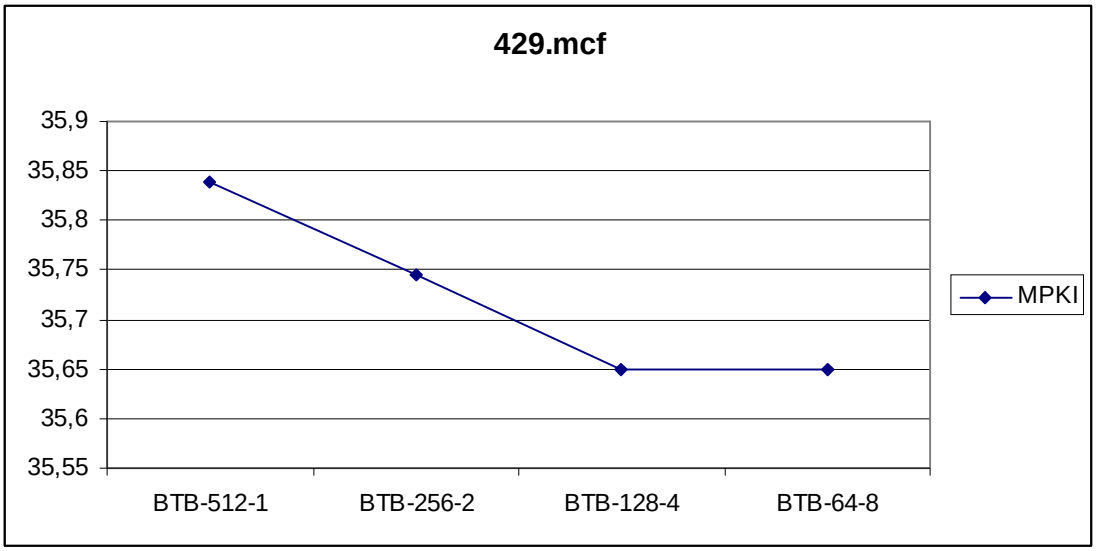
Επέλεξα να σχεδιάσω γραφική παράσταση τόσο για το direction MPKI όσο και για το target MPKI για να υπάρχει μια πιο πλήρης εικόνα για την απόδοση των BTB, αφού έτσι συγκρίνουμε και τα direction mispredictions και τα target mispredictions.

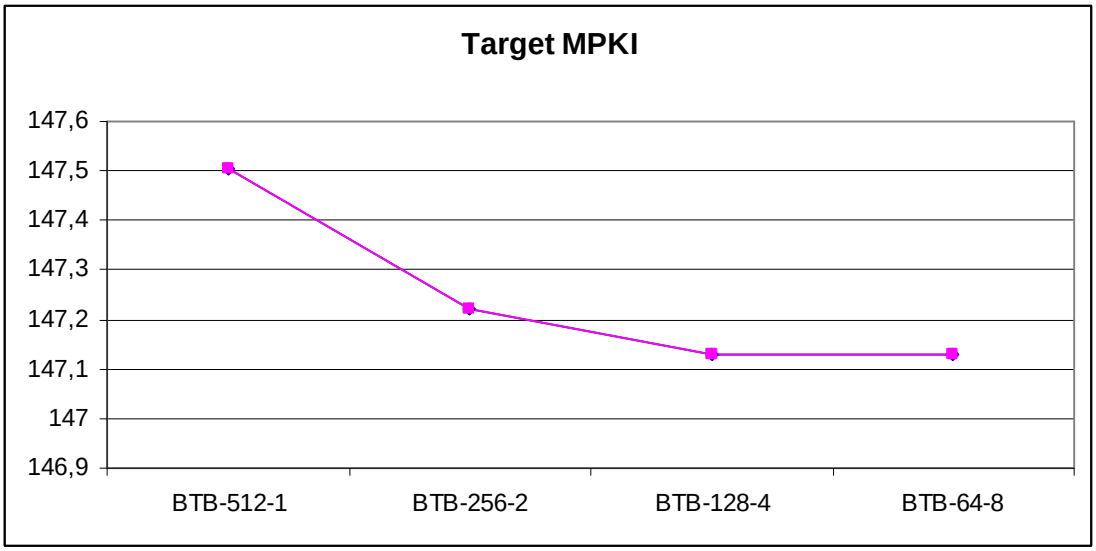
Ακολουθούν τα διαγράμματα που σχεδιάστηκαν για τις 4 διαφορετικές περιπτώσεις BTB όταν τρέξαμε τις προσομοιώσεις στα δοθέντα benchmarks.



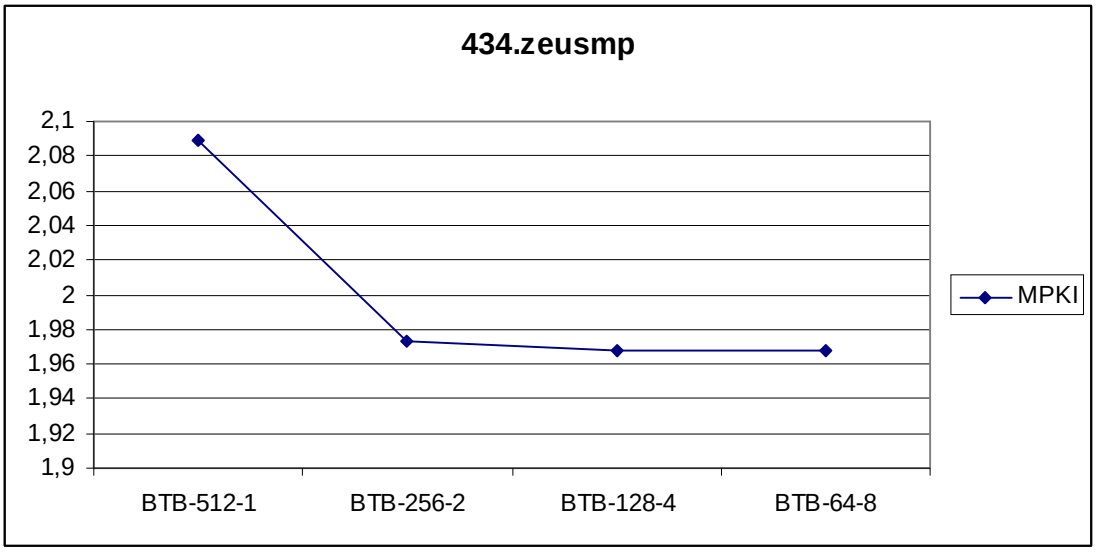


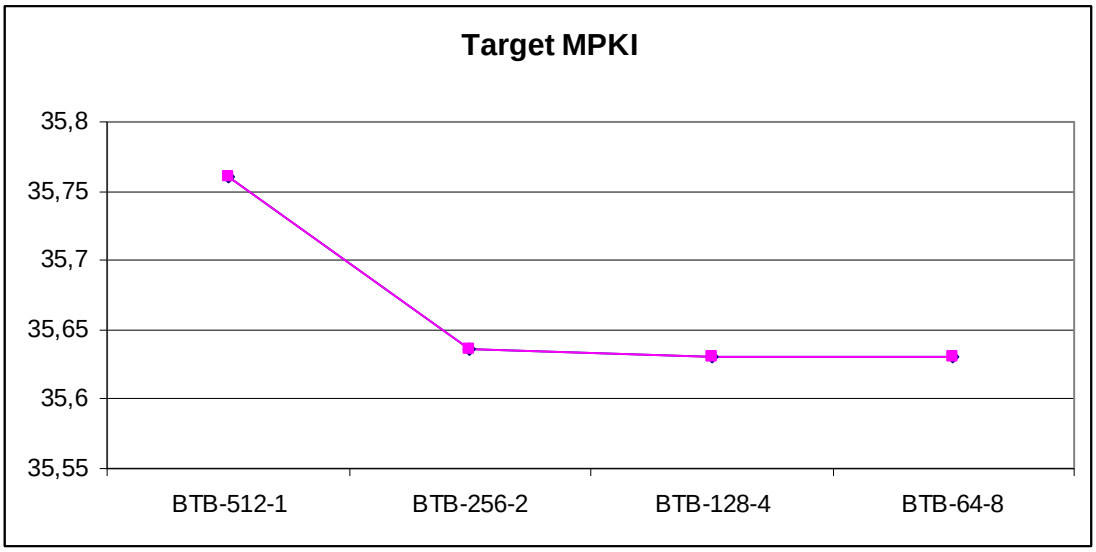
Στο πρώτο benchmark παρατηρούμε πως και στα 2 διαγράμματα (Direction MPKI και Target MPKI) έχουμε μια φθίνουσα πορεία καθώς μειώνονται τα entries και αυξάνεται το associativity. Ένα συμπέρασμα που θα μπορούσαμε να βγάλουμε είναι ότι πιθανότατα υπάρχουν collisions στις εκχωρήσεις του BTB σύμφωνα με την τιμή του PC και γι'αυτό, μικρό associativity δίνει χειρότερα αποτελέσματα διότι νέες εκχωρήσεις διώχνουν τις παλαιότερες που όμως ακόμη είναι χρήσιμες. Βέλτιστο αποτέλεσμα λοιπόν δίνει το BTB με 64K entries και associativity 8.



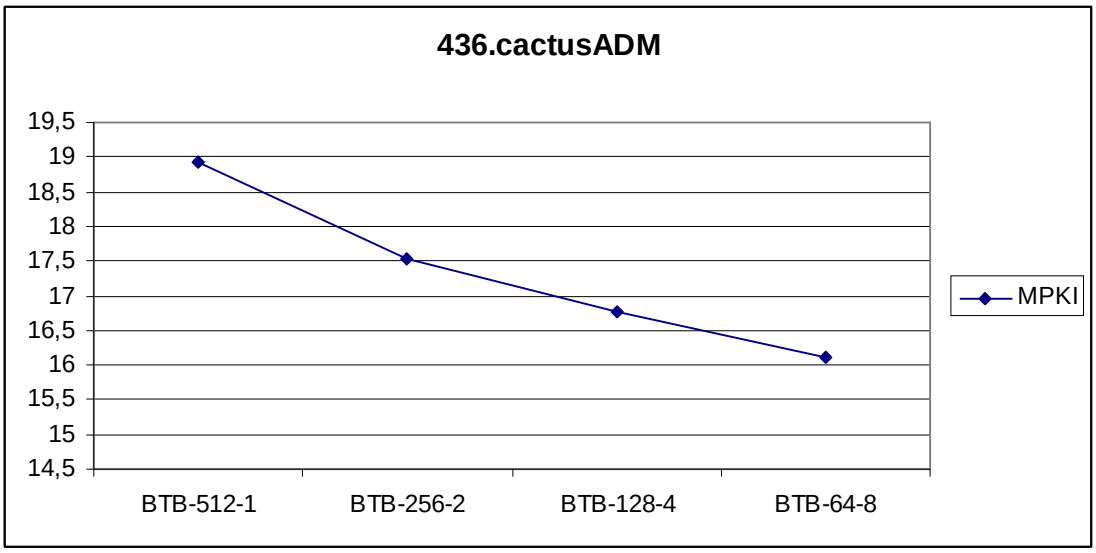


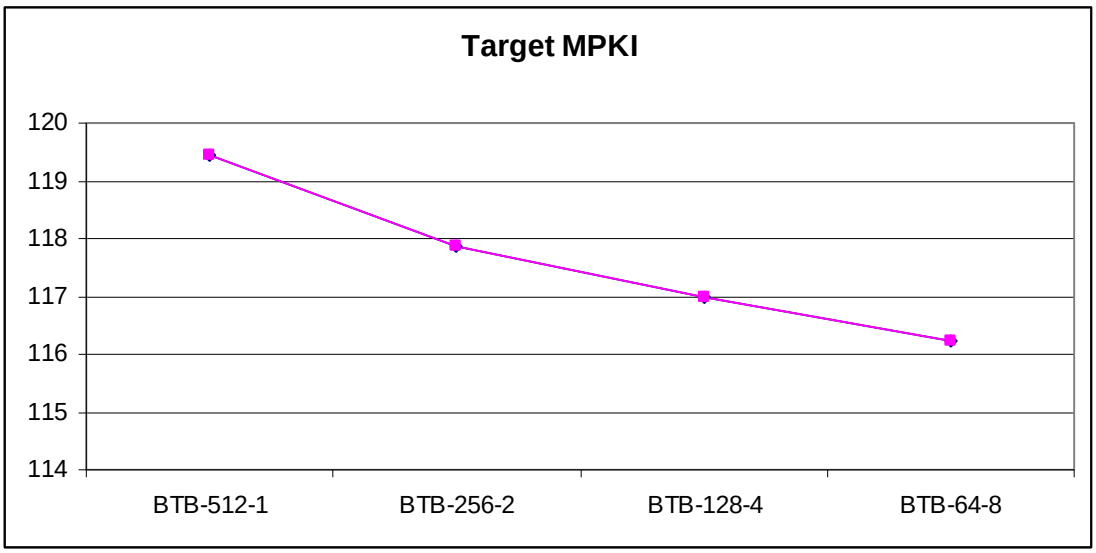
Παρότι βλέπουμε τις δύο συναρτήσεις για το mcf φθίνουσες, οι τιμές μεταξύ τους σε κάθε γραφική είναι σχεδόν ίδιες μιας και η διαφορά από περίπτωση σε περίπτωση είναι στα 3-4 mispredictions, οπότε πρόκειται για σχεδόν αμελητέα διαφοροποίηση. Όπως παρατηρούμε βέλτιστο αποτέλεσμα δίνει το BTB με 128K entries και associativity 4 και το BTB με 64K entries και associativity 8.



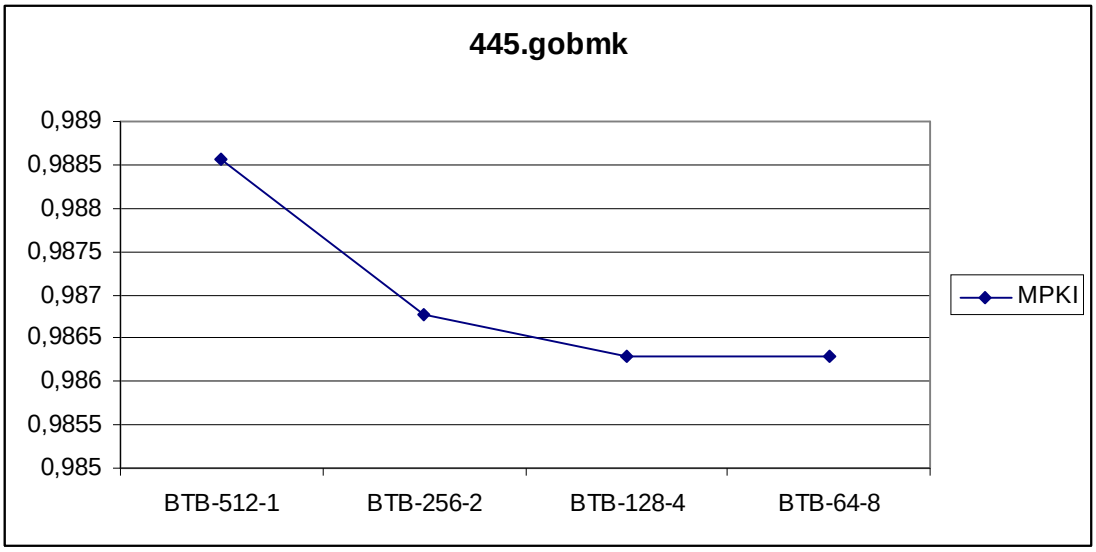


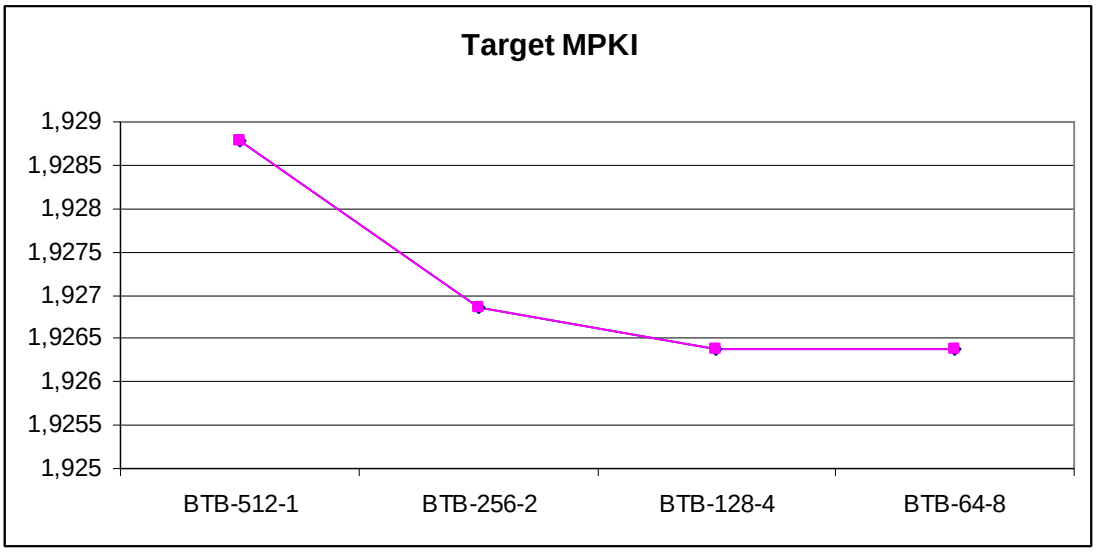
Και σε αυτή την περίπτωση παρατηρούμε πως οι γραφικές μας παραστάσεις είναι φθίνουσες, όμως εκτός της πρώτης, οι υπόλοιπες 3 τιμές είναι πρακτικά ίσες, άρα οποιοσδήποτε συνδυασμός από αυτούς που δοκιμάσαμε, εκτός του BTB με 512 entries και associativity 1, δίνει πρακτικά βέλτιστα αποτελέσματα.



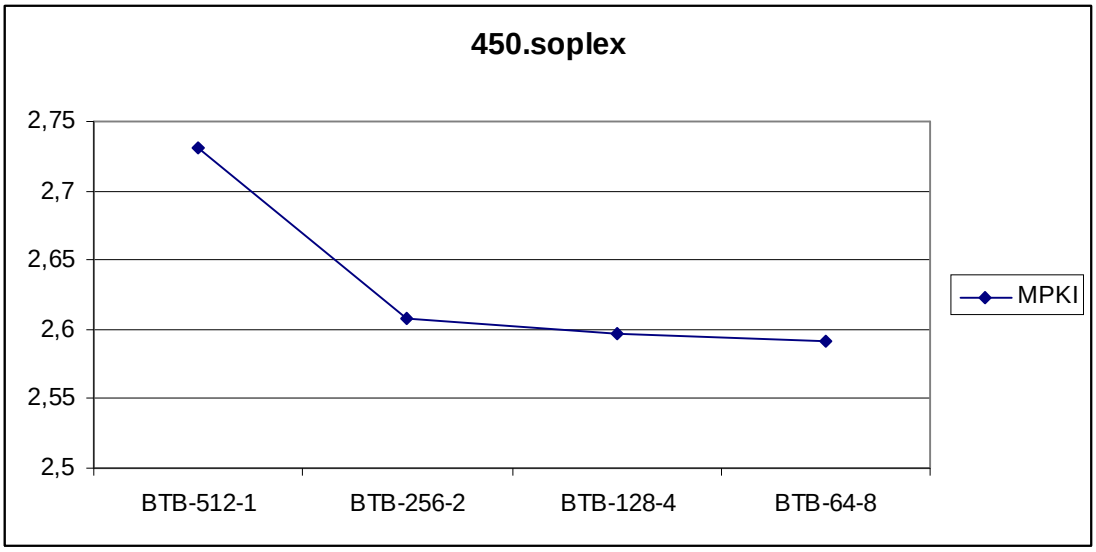


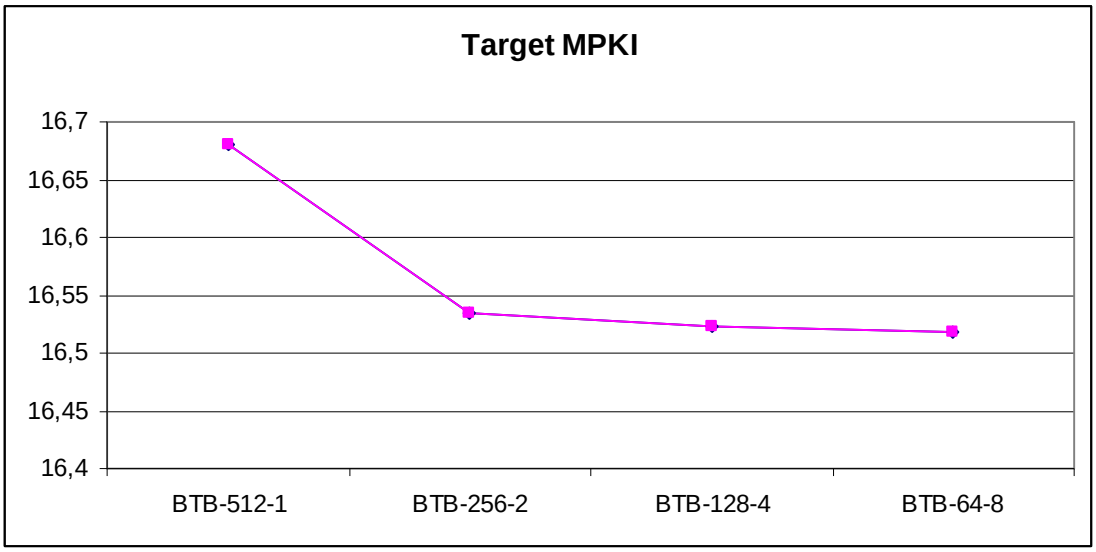
Και στο cactusADM έχουμε δύο φθίνουσες γραφικές οι οποίες μάλιστα παρουσιάζουν και τις μεγαλύτερες βελτιώσεις του MPKI με τη μείωση των entries και την αύξηση του associativity. Και σε αυτή την περίπτωση, βέλτιστα αποτελέσματα δίνει το BTB με 64K entries και associativity 8.



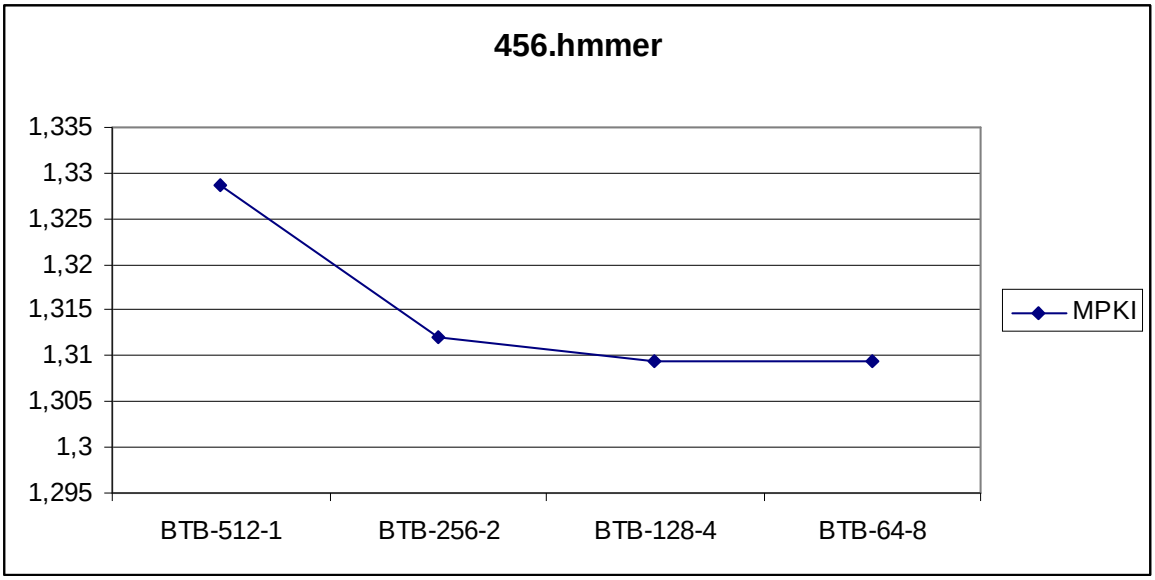


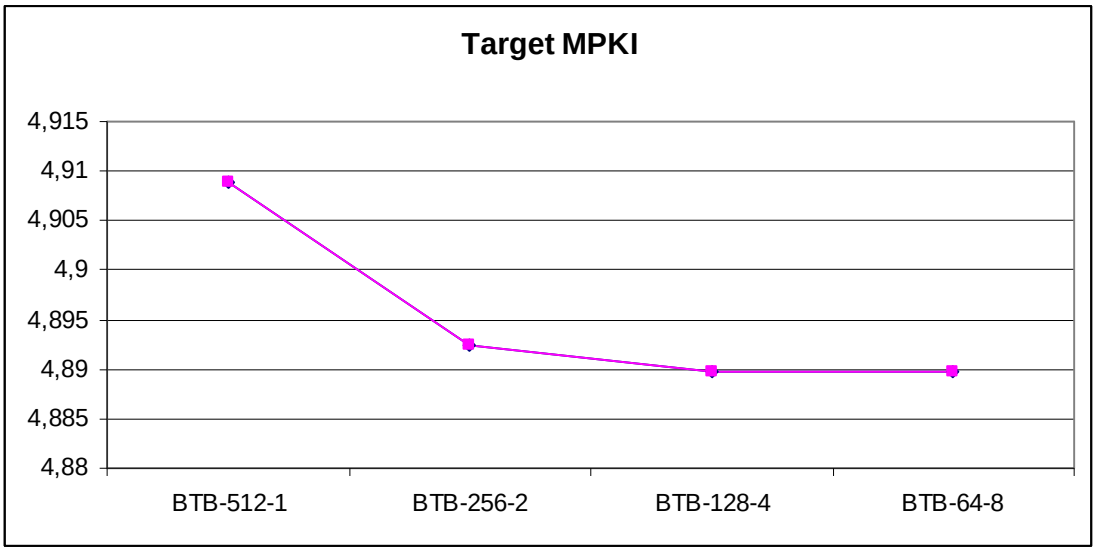
Και σε αυτή την περίπτωση και στα 2 διαγράμματα παρατηρούμε φθίνουσα πορεία με τα βέλτιστα αποτελέσματα να δίνονται από το BTB με 64K entries και associativity 8. Σημειώνεται όμως ότι στο σε αυτή την περίπτωση οι διαφορετικές τιμές των δύο γραφικών έχουν αμελητέα διαφορά μεταξύ τους μιας και η απόστασή τους έγκειται στο 3^{ο} και 4^{ο} δεκαδικό ψηφίο του MPKI.



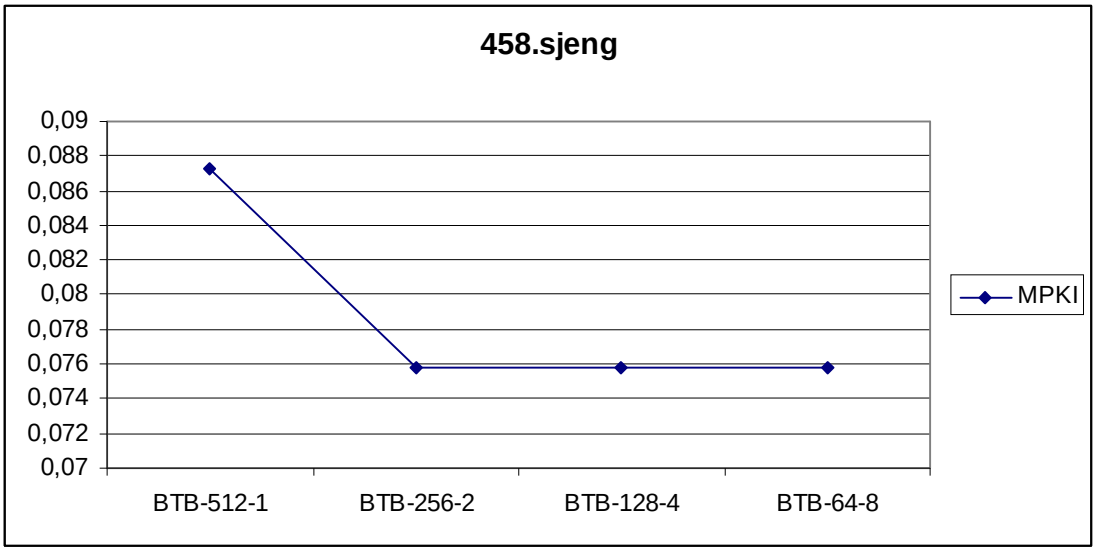


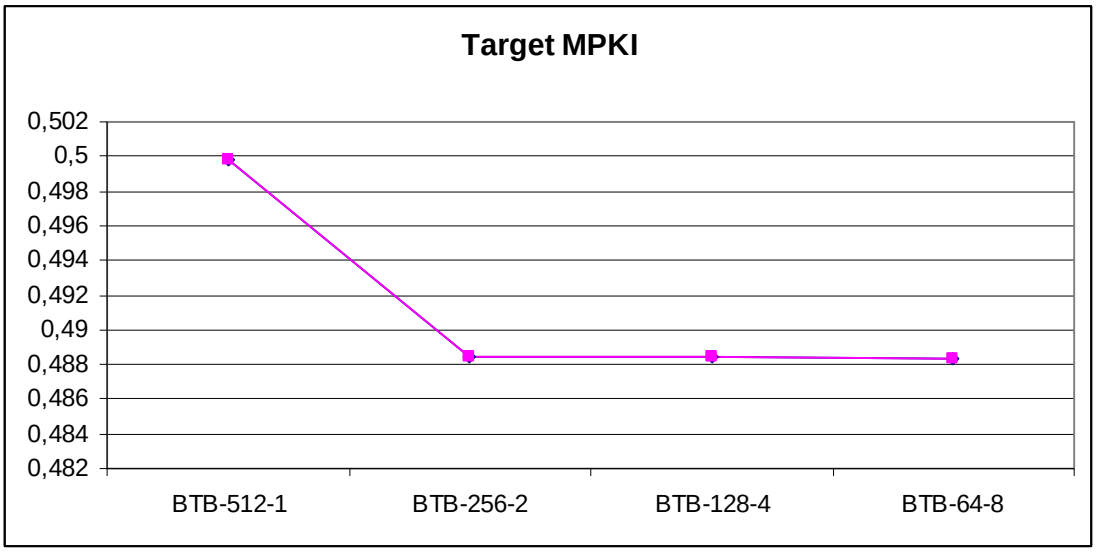
Στα διαγράμματα που προκύψανε από το soplex παρατηρούμε όπως και σε όλα τα προηγούμενα φθίνουσες γραφικές παραστάσεις, όμως όπως και στο zeusmp η μόνη αξιοσημείωτη βελτίωση του MPKI και στις 2 γραφικές είναι η μετάβαση από BTB με 512K entries και associativity 1 σε BTB με 256K entries και associativity 2. OI επόμενες περιπτώσεις δείχνουν μια ελαφριά βελτίωση του MPKI χωρίς όμως να είναι τόσο σημαντική.



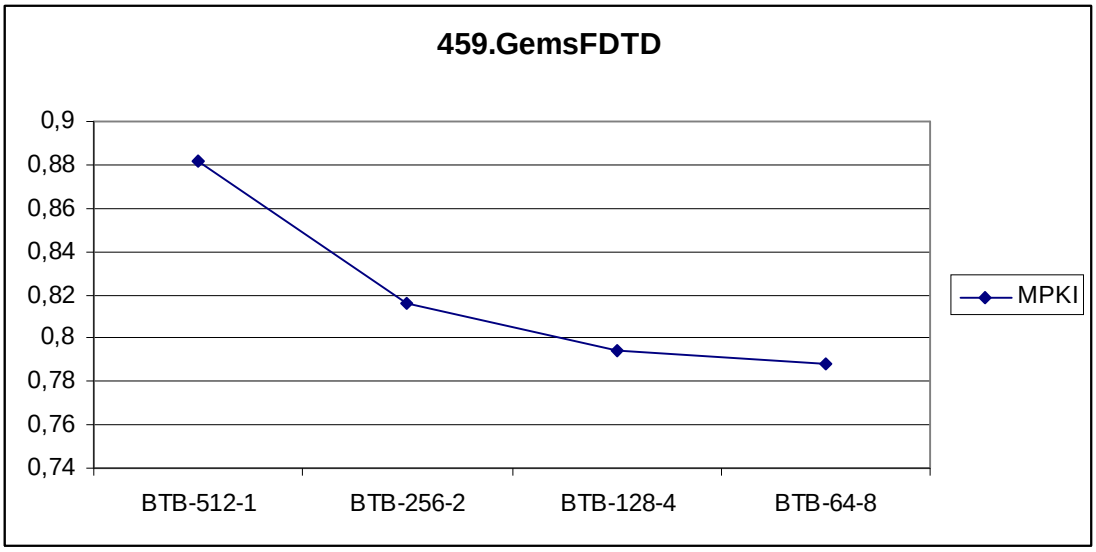


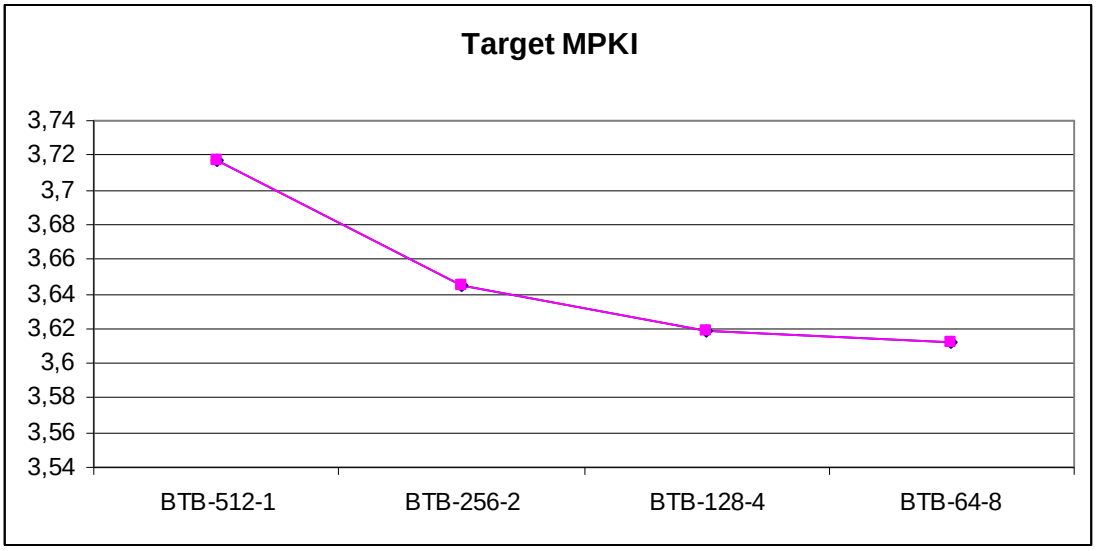
Και σε αυτή την περίπτωση έχουμε εμφανή βελτίωση των MPKI από την πρώτη στη δεύτερη περίπτωση, οι υπόλοιπες 3 ($2^{η}$-$3^{η}$-$4^{η}$) δίνουν παραπλήσια αποτελέσματα μεταξύ τους και συγκεκριμένα η $3^{η}$ και η $4^{η}$ περίπτωση δίνουν ακριβώς τα ίδια MPKI.



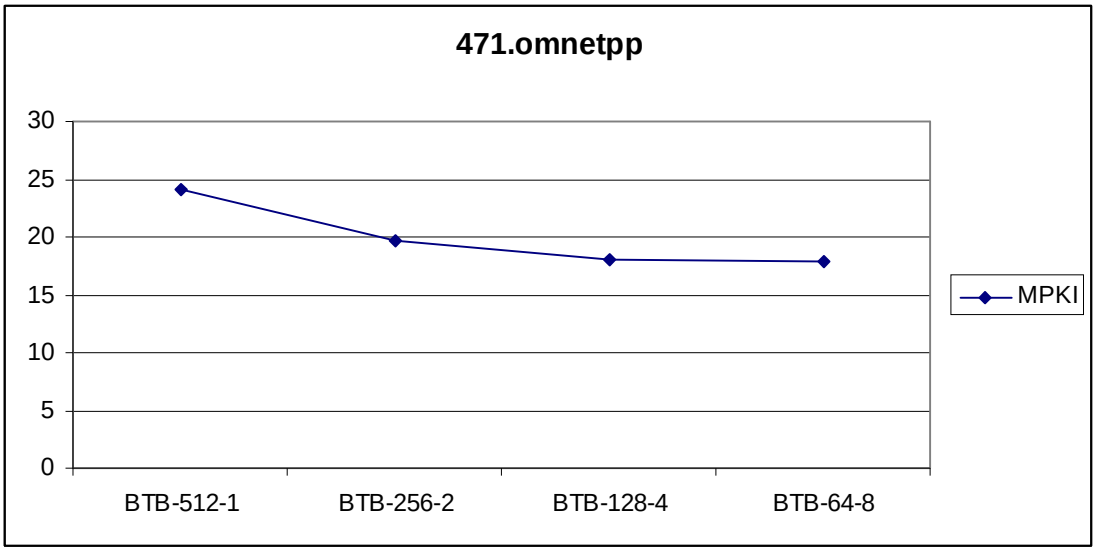


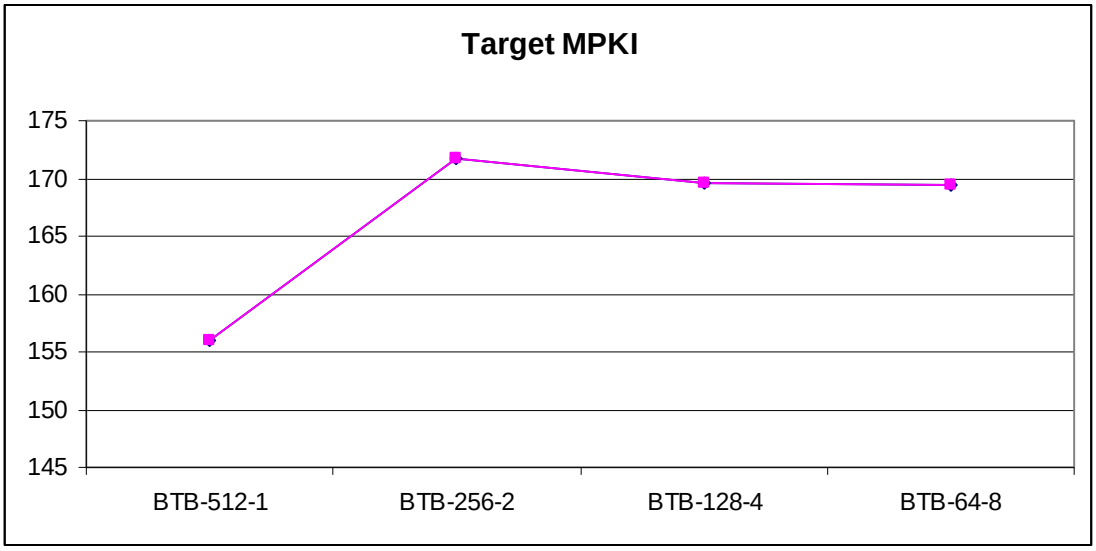
Στο sjeng η μόνη διαφοροποίηση που παρατηρούμε στα αποτελέσματα και των 2 γραφικών είναι ότι το BTB με 512K entries και associativity 1 δίνει χειρότερα αποτελέσματα από τις άλλες 3 περιπτώσεις που μας δίνουν μεταξύ τους ίσες τιμές MPKI.



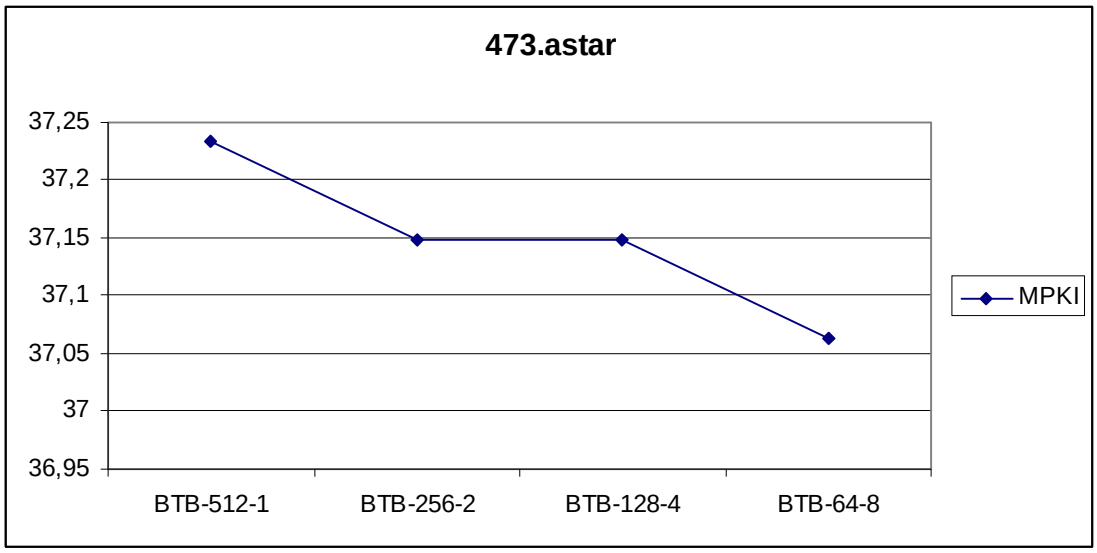


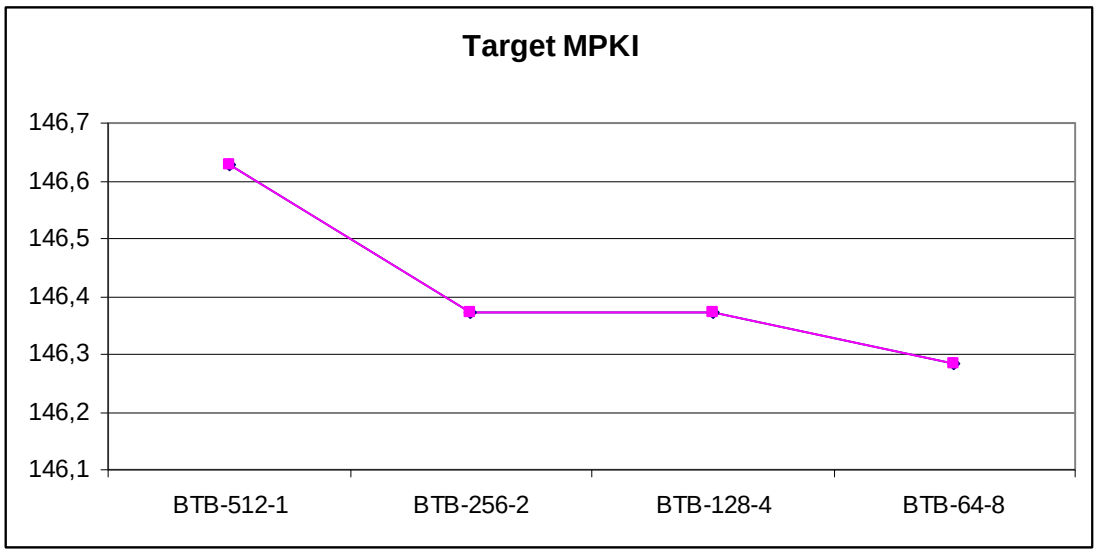
Το GemsFDTD μας έδωσε 2 γνησίως φθίνουσες γραφικές παραστάσεις των MPKI, άρα όπως και στα αρχικά benchmarks βέλτιστο αποτέλεσμα δίνει το BTB με 64K entries και associativity 8.



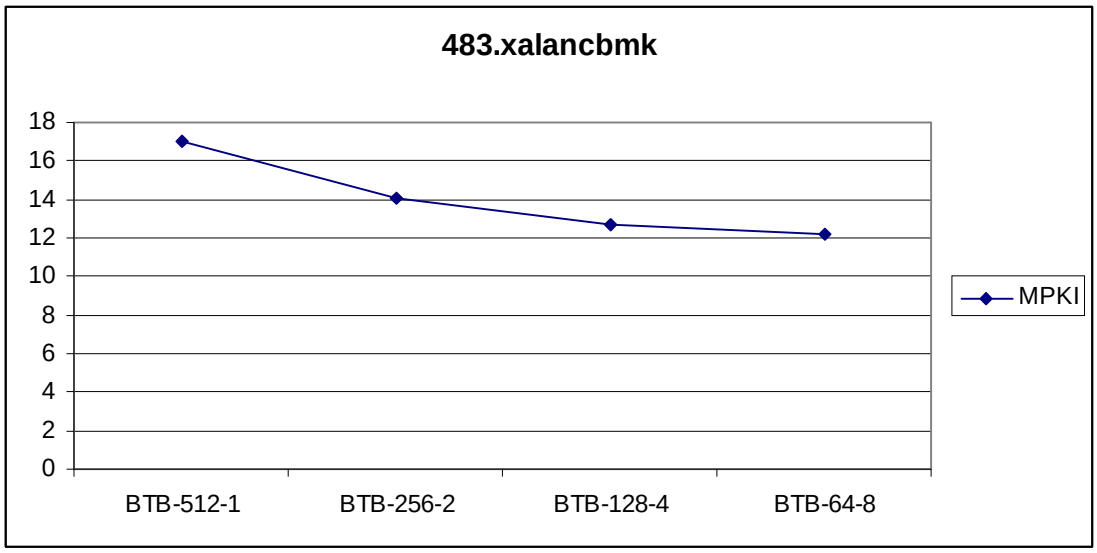


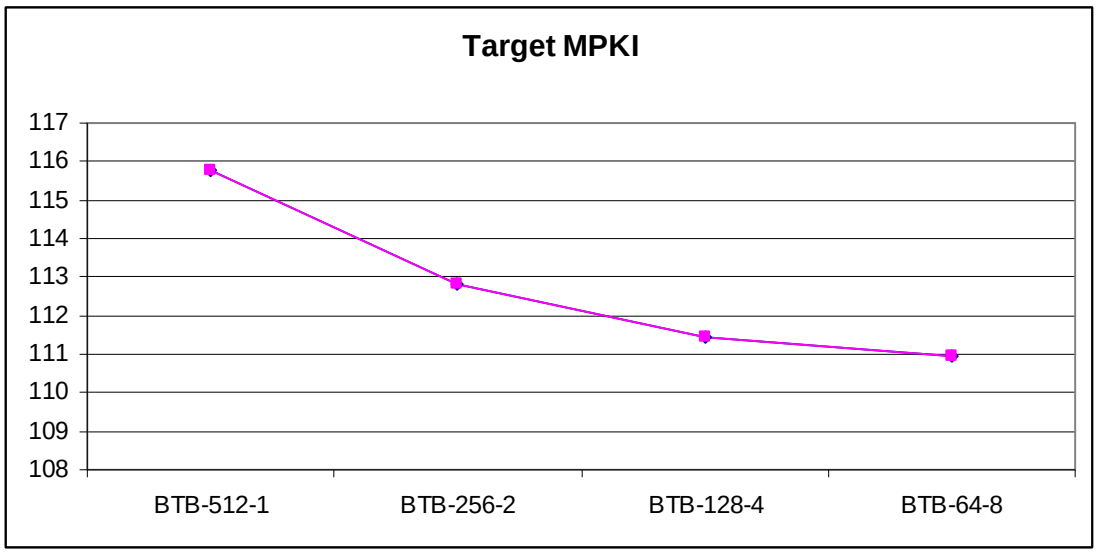
Το omnetpp είναι το μόνο που μας δίνει διαφορετική μορφή μεταξύ των 2 γραφικών παραστάσεων. Ενώ το direction MPKI είναι όμοιο με τα περισσότερα προηγούμενα, δηλαδή μια φθίνουσα γραφική παράσταση, το target MPKI παρουσιάζει μια σημαντικά καλύτερη τιμή για BTB με 512K entries και associativity 1 σε σύγκριση με τις υπόλοιπες περιπτώσεις. Πιθανότατα στη συγκεκριμένη εφαρμογή είχαμε capacity misses από το Branch Target Buffer κι όχι collision misses όπως φαίνεται να έχουμε σε όλες τις προηγούμενες, για αυτό και μεγάλο πλήθος entries με το ελάχιστο associativity δίνει βέλτιστο αποτέλεσμα.





Στο astar παρατηρούμε και πάλι μια φθίνουσα γραφική και για τα δύο MPKI, συγκεκριμένα, παρατηρούμε βελτίωση από την πρώτη στη 2η περίπτωση και από την 3η στην 4η. Δεύτερη και τρίτη περίπτωση μας δίνουν ταυτόσημα αποτελέσματα. Βέλτιστα MPKI λοιπό επιτυγχάνονται με BTB με 64K entries και associativity 8.





Και στο xalancbmk έχουμε όπως και σχεδόν σε όλα τα προηγούμενα 2 φθίνουσες γραφικές, άρα ισχύουν όσα έχουν αναφερθεί προηγουμένως.

Συνοψίζοντας τις παρατηρήσεις μας από τα διαγράμματα που παρουσιάστηκαν παραπάνω, φαίνεται βέλτιστα αποτελέσματα να δίνει το **BTB με 64K entries και associativity 8**, μιας και μόνο σε ένα target MPKI δεν έχουμε βέλτιστη τιμή για τη συγκεκριμένη περίπτωση. Γενικότερα φαίνεται πως για τις εφαρμογές μας είναι σημαντικότερο το associativity από τον αριθμό των BTB entries μιας και σχεδόν σε όλες, αύξηση του associativity μαζί με μείωση των entries οδηγούσε σε μείωση των MPKI, άρα δεν υπήρχε πρόβλημα στο πλήθος των εγγραφών του Buffer αλλά στις συγκρούσεις που συνέβαιναν και έδιωχναν τις ήδη υπάρχουσες εγγραφές. Αυτό ίσως να μπορούσε να αντιμετωπισθεί και με διαφορετική πολιτική σε ότι αφορά το hashing στο Buffer ανάλογα με την τιμή του PC. Στις περισσότερες εφαρμογές η γραφικές του target MPKI και του direction MPKI είχαν παρόμοια συμπεριφορά.

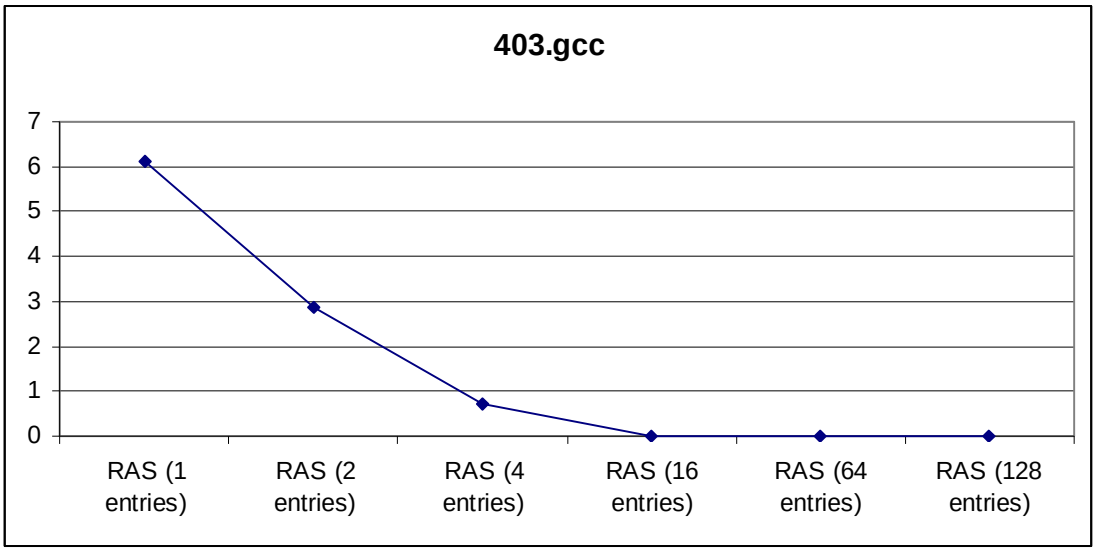
4.4 Μελέτη του RAS

Χρησιμοποιώντας την υλοποίηση της RAS (ras.h) που μου δόθηκε, έτρεξα τα δοθέντα benchmarks για τις ακόλουθες περιπτώσεις.

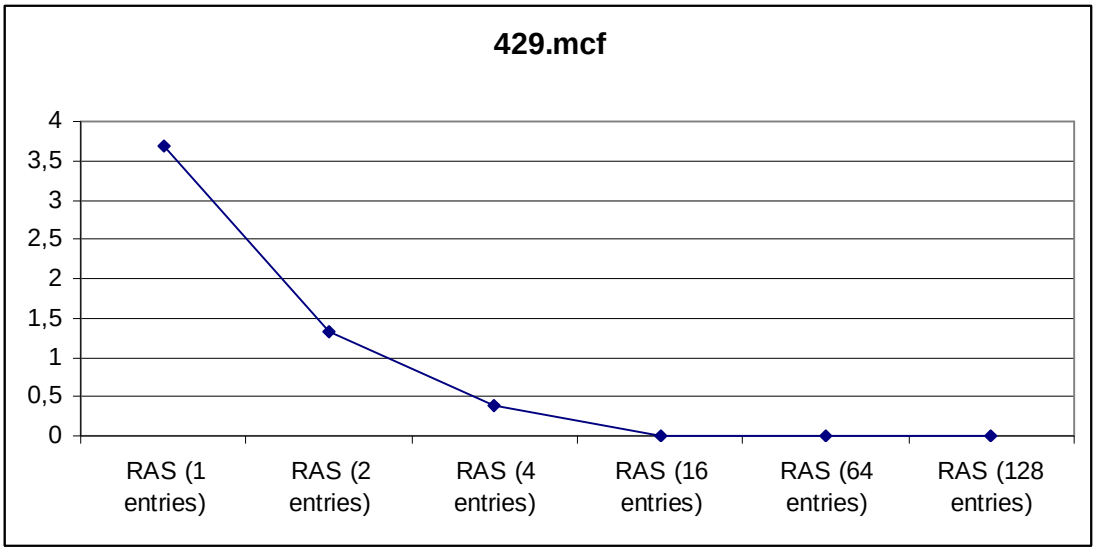
Αριθμός εγγραφών στη RAS
1
2
4

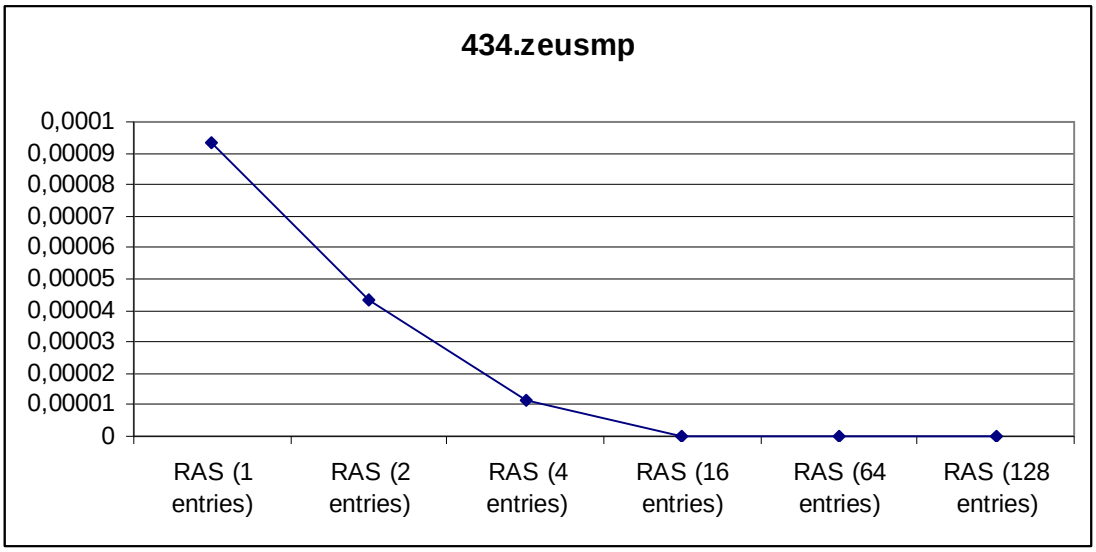
Αριθμός εγγραφών στη RAS
16
64
128

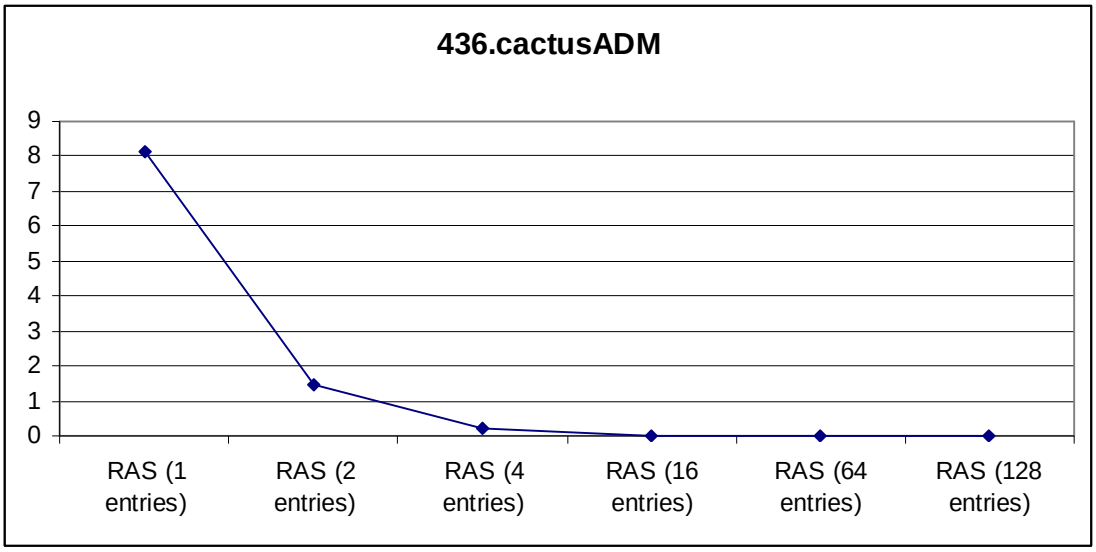
Ακολουθούν τα διαγράμματα που προέκυψαν από την προσομοίωση των benchmarks με τις διάφορες παραμέτρους της RAS συναρτήσει του MPKI.

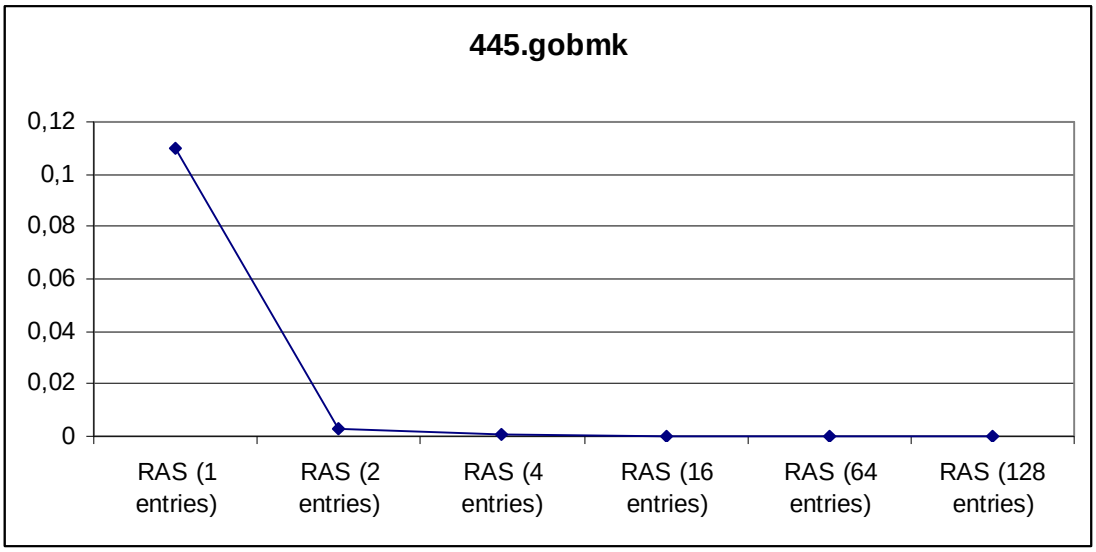


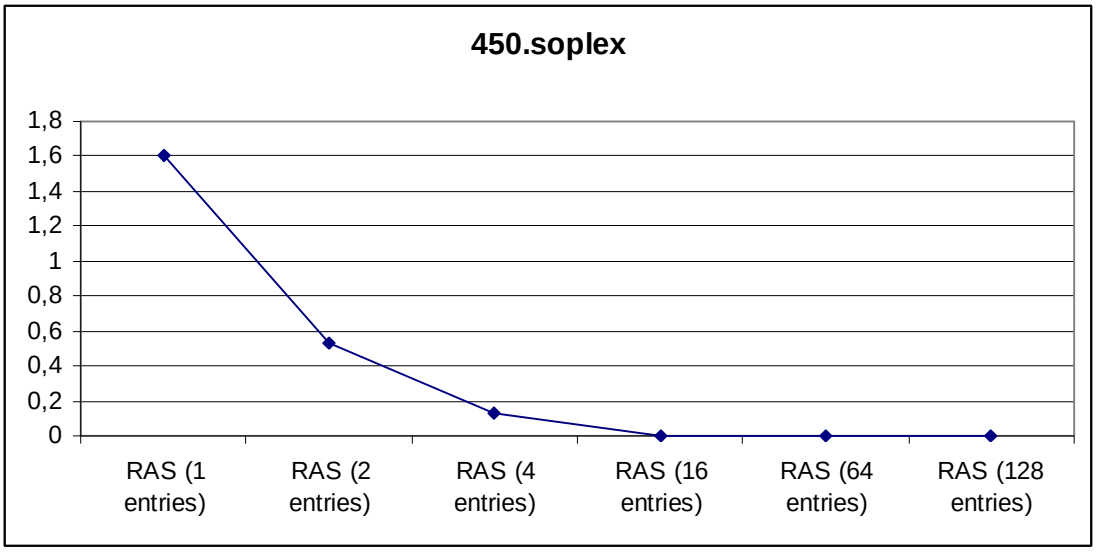
Όπως ήταν αναμενόμενο παρατηρούμε ότι καθώς αυξάνεται ο αριθμός των εγγραφών στη RAS, μειώνεται το MPKI. Μάλιστα από 16 εγγραφές και πάνω βλέπουμε πως έχουμε μηδενικό MPKI. Όταν έχουμε λιγότερες εγγραφές στη RAS τότε εμφωλευμένες κλήσεις συναρτήσεων και γενικότερα εάν έχουμε ταυτόχρονα αρκετές κλήσεις ανοιχτές οδηγούν σε stack overflow με αποτέλεσμα να χάσουμε ορισμένα return addresses και να οδηγηθούμε σε mispredictions. Σε περίπτωση όμως που έχουμε επαρκή χώρο στη στοίβα δεν υπάρχει λόγος να υπάρξει misprediction μιας και κρατούνται σωστά όλες οι διευθύνσεις επιστροφής από τις κλήσεις των συναρτήσεων, και για αυτό το λόγο βλέπουμε και τις μηδενικές τιμές του MPKI για περισσότερες των τεσσάρων εγγραφές στη RAS. Γενικότερα όλα τα διαγράμματα έχουν σχεδόν την ίδια μορφή, φθίνουσες συναρτήσεις με μηδενικές ή σχεδόν μηδενικές τιμές του MPKI για εγγραφές RAS από 16 και πάνω, γι’αυτό και δεν έχει νόημα ο επιμέρους σχολιασμός κάθε γραφικής .

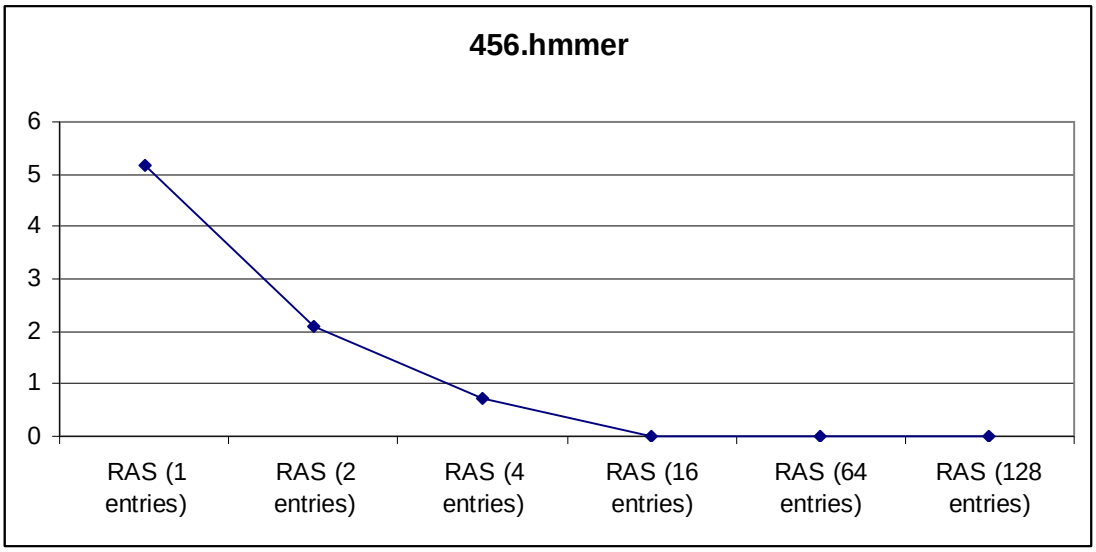

429.mcf
4
3,5
3
2,5
2
1,5
1
0,5
0
RAS (1 entries)
RAS (2 entries)
RAS (4 entries)
RAS (16 entries)
RAS (64 entries)
RAS (128 entries)

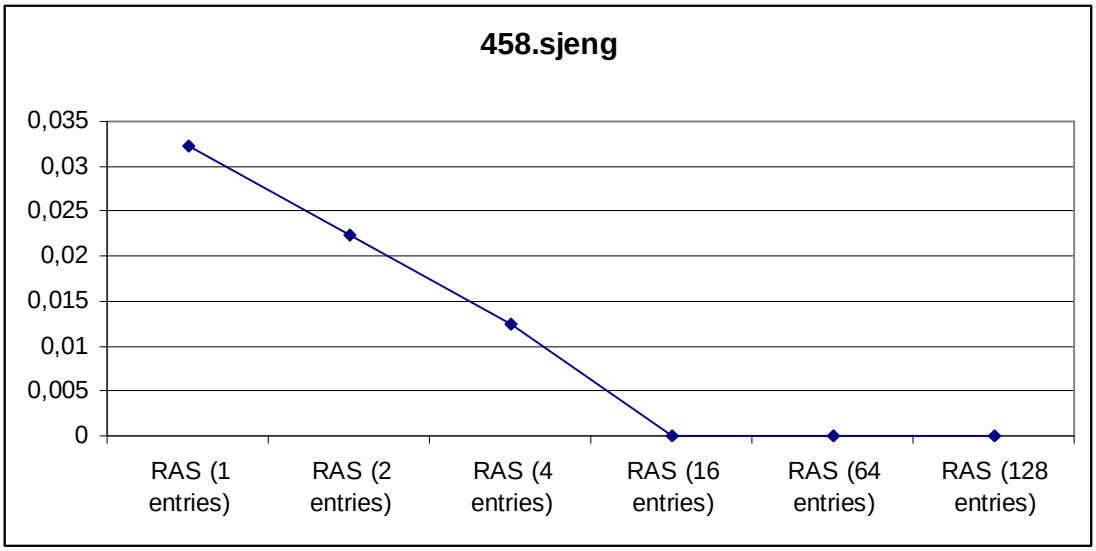

434.zeusmp
0,0001
0,00009
0,00008
0,00007
0,00006
0,00005
0,00004
0,00003
0,00002
0,00001
0
RAS (1 entries)
RAS (2 entries)
RAS (4 entries)
RAS (16 entries)
RAS (64 entries)
RAS (128 entries)

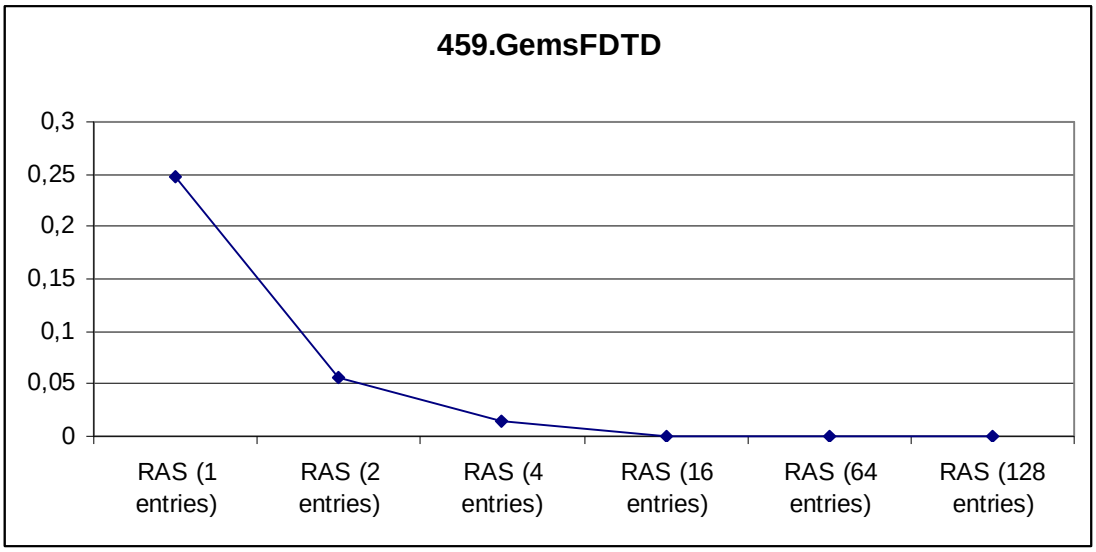

436.cactusADM
9
8
7
6
5
4
3
2
1
0
RAS (1 entries)
RAS (2 entries)
RAS (4 entries)
RAS (16 entries)
RAS (64 entries)
RAS (128 entries)

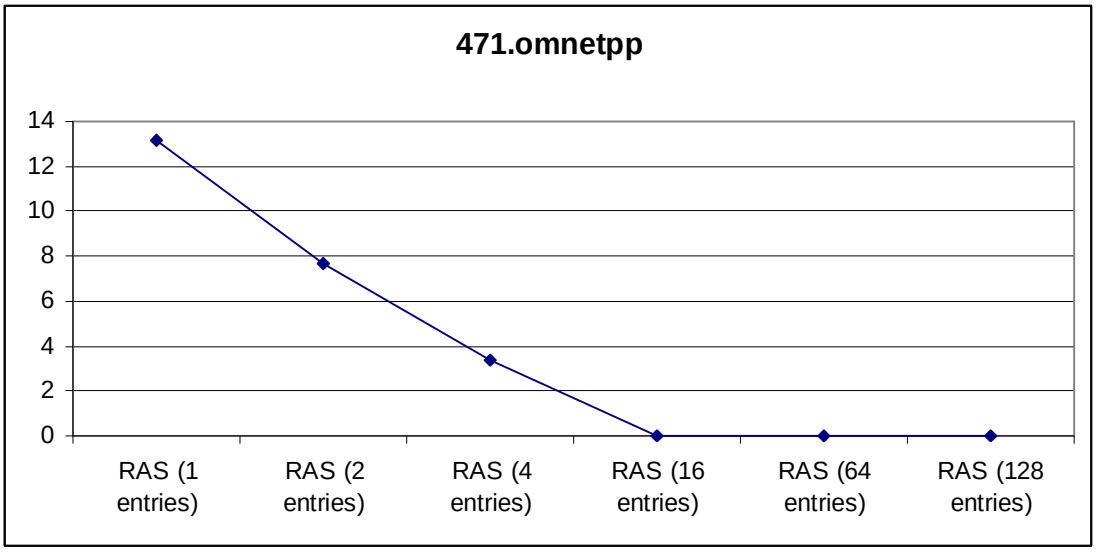

445.gobmk
0,12
0,1
0,08
0,06
0,04
0,02
0
RAS (1 entries)
RAS (2 entries)
RAS (4 entries)
RAS (16 entries)
RAS (64 entries)
RAS (128 entries)

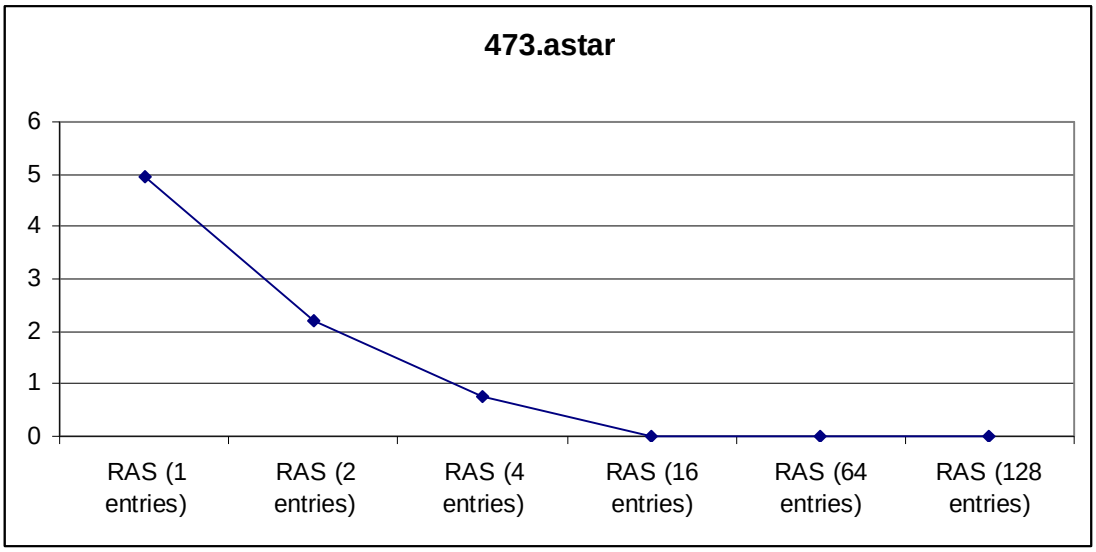

450.soplex
1,8
1,6
1,4
1,2
1
0,8
0,6
0,4
0,2
0
RAS (1 entries)
RAS (2 entries)
RAS (4 entries)
RAS (16 entries)
RAS (64 entries)
RAS (128 entries)

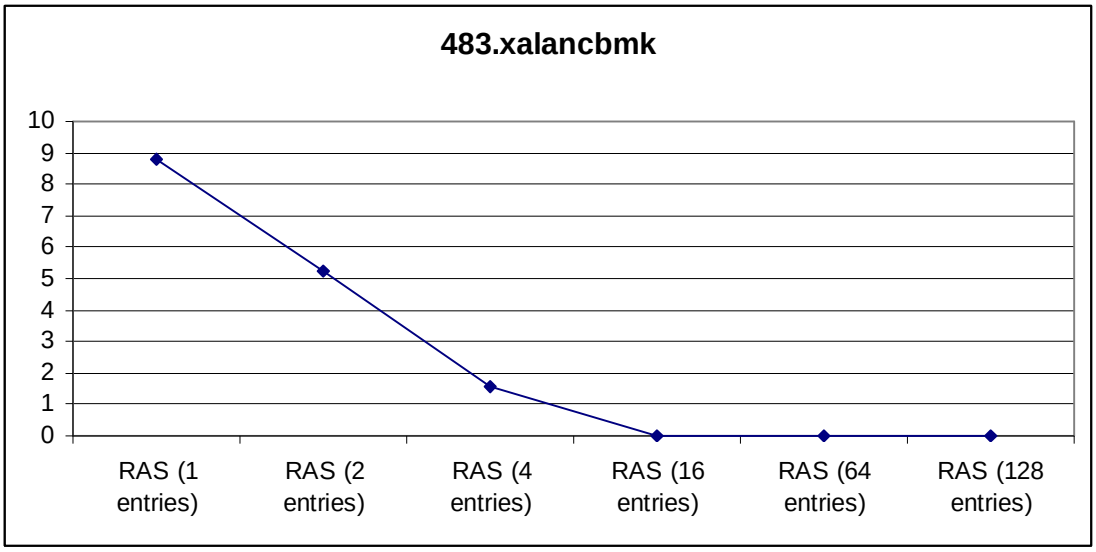

456.hmmer
6
5
4
3
2
1
0
RAS (1 entries)
RAS (2 entries)
RAS (4 entries)
RAS (16 entries)
RAS (64 entries)
RAS (128 entries)


458.sjeng
0,035
0,03
0,025
0,02
0,015
0,01
0,005
0
RAS (1 entries)
RAS (2 entries)
RAS (4 entries)
RAS (16 entries)
RAS (64 entries)
RAS (128 entries)


459.GemsFDTD
0,3
0,25
0,2
0,15
0,1
0,05
0
RAS (1 entries)
RAS (2 entries)
RAS (4 entries)
RAS (16 entries)
RAS (64 entries)
RAS (128 entries)


471.omnetpp
14
12
10
8
6
4
2
0
RAS (1 entries)
RAS (2 entries)
RAS (4 entries)
RAS (16 entries)
RAS (64 entries)
RAS (128 entries)


473.astar
6
5
4
3
2
1
0
RAS (1 entries)
RAS (2 entries)
RAS (4 entries)
RAS (16 entries)
RAS (64 entries)
RAS (128 entries)



Συνοψίζοντας όπως αναφέρθηκε και στην αρχή του ερωτήματος σχεδόν σε όλες τις προσομοιώσεις έχουμε μηδενικό MPKI όταν οι εγγραφές της RAS είναι από 16 και πάνω. Σε μερικές μάλιστα η γραφική πλησιάζει το μηδέν και από τις 2 εγγραφές (π.χ. gobmk) ή και τις 4 εγγραφές (π.χ. cactusADM, GemsFDTD κλπ.). Βλέπουμε λοιπόν ότι σχεδόν σε κανένα από τα benchmarks που μας έχουν δοθεί δεν έχουμε πολλές κλήσεις συναρτήσεων να είναι ανοιχτές ταυτόχρονα μιας και οι 16 θέσεις φτάνουν για να έχουμε αποθηκευμένες τι διευθύνσεις επιστροφής.
Λαμβάνοντας υπόψη ότι 16, 64 και 128 εγγραφές στη RAS δίνουν σχεδόν το ίδιο αποτέλεσμα, θα επιλέξουμε ως βέλτιστο μέγεθος RAS τις 16 εγγραφές μιας και η επιλογή λιγότερων οδηγεί σε αύξηση του MPKI και η επιλογή περισσότερων οδηγεί σε αύξηση του κόστους χωρίς να αποδίδει αυτό το κόστος σε καλύτερες επιδόσεις.