ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΠΡΟΗΓΜΕΝΑ ΘΕΜΑΤΑ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

(2020-2021)

2η Σειρά Ασκήσεων

Ονοματεπώνυμο:

- Χρήστος Τσούφης

Αριθμός Μητρώου:

- 03117176

Στοιχεία Επικοινωνίας:

- el17176@mail.ntua.gr

1. Εισαγωγή

Στα πλαίσια της παρούσας άσκησης χρησιμοποιήθηκε το εργαλείο "PIN" για την μελέτη της επίδραση διαφορετικών συστημάτων πρόβλεψης εντολών άλματος καθώς και την αξιολόγηση τους με δεδομένο το διαθέσιμο χώρο πάνω στο τσιπ.

2. PINTOOL

Στον βοηθητικό κώδικα της άσκησης βρίσκονται τα pintools cslab_branch_stats.cpp και cslab_branch.cpp. Αφού τροποποιηθεί το PIN_ROOT path στο αρχείο Makefile, να γίνει η μεταγλώττιση. Το cslab_branch_stats.cpp χρησιμοποιείται για την εξαγωγή στατιστικών σχετικά με τις εντολές αλμάτων που εκτελούνται στην εφαρμογή. Το cslab_branch.cpp χρησιμοποιείται για την αξιολόγηση τεχνικών πρόβλεψης άλματος ενώ για τις εντολές επιστροφής από διαδικασίες προσομοιώνει διαφορετικά μεγέθη στοίβας διεύθυνσης επιστροφής (RAS). Στο αρχείο branch_predictor.h ορίζονται οι διαφορετικοί branch predictors. Για την προσθήκη ενός branch predictor απαιτείται η δημιουργία μίας νέας υπο-κλάσης της κλάσης BranchPredictor και ο ορισμός τριών μεθόδων predict(), update() και getName(). Η πρώτη συνάρτηση δέχεται ως ορίσματα το PC της εντολής και τη διεύθυνση προορισμού και καλείται να προβλέψει αν το άλμα θα εκτελεστεί ή όχι (Taken / Not Taken). Η δεύτερη μέθοδος καλείται να αποθηκεύσει τις πληροφορίες εκείνες που απαιτούνται για τις μελλοντικές προβλέψεις. Τα ορίσματα της είναι η πρόβλεψη που έκανε ο predictor, το πραγματικό αποτέλεσμα της εντολής διακλάδωσης, το PC της εντολής και η διεύθυνση προορισμού. Τέλος, η μέθοδος getName() χρησιμοποιείται για την εκτύπωση των αποτελεσμάτων του branch predictor στο αρχείο εξόδου του pintool.

3. Μετροπρογράμματα

Το PIN μπορεί να χρησιμοποιηθεί για την εκτέλεση οποιασδήποτε εφαρμογής. Στα πλαίσια της παρούσας άσκησης χρησιμοποιήθηκαν τα SPEC_CPU2006 benchmarks. Πιο συγκεκριμένα χρησιμοποιήθηκαν τα παρακάτω 14 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. *462.libquantum*
11. *470.lbm*
12. *471.omnetpp*
13. *473.astar*
14. *483.xalancbmk*

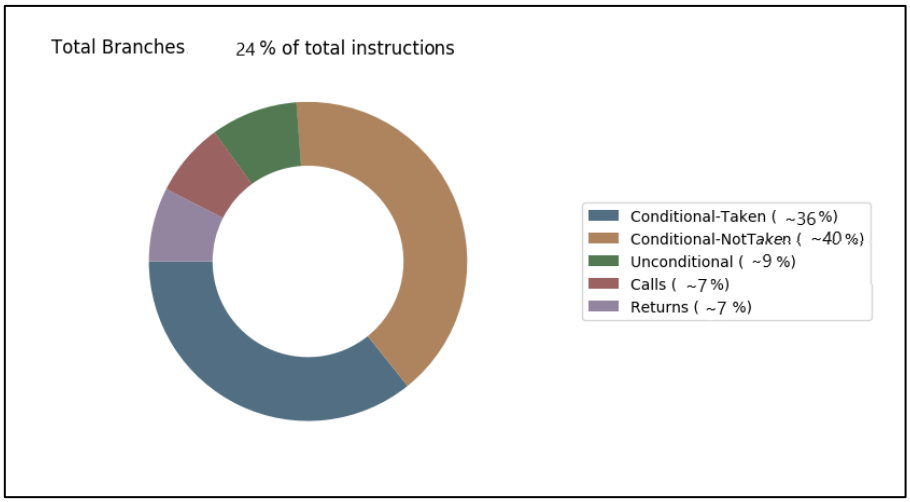
Στον βοηθητικό κώδικα της άσκησης δίνεται ο φάκελος spec_execs_train_inputs ο οποίος περιέχει τα εκτελέσιμα και τα απαραίτητα αρχεία εισόδου για τα παραπάνω benchmarks.

4. Πειραματική Αξιολόγηση

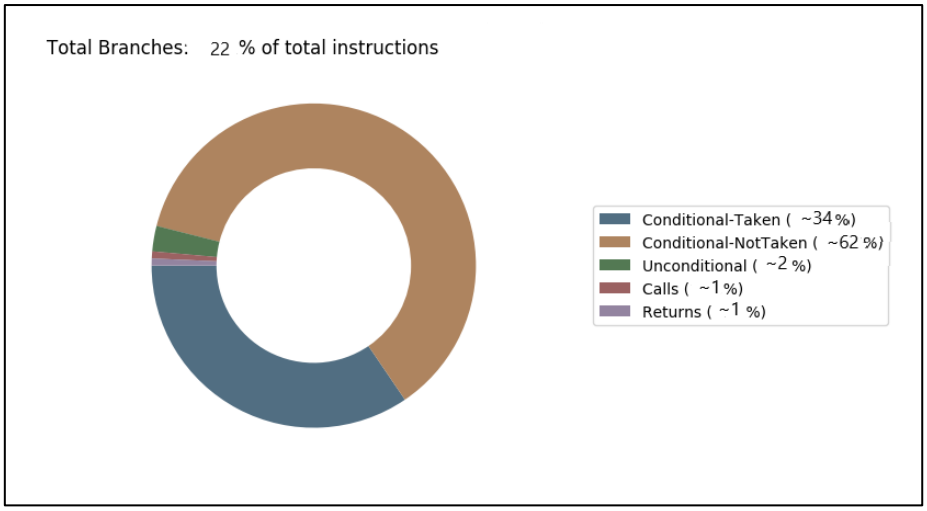
4.1 Μελέτη εντολών άλματος

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

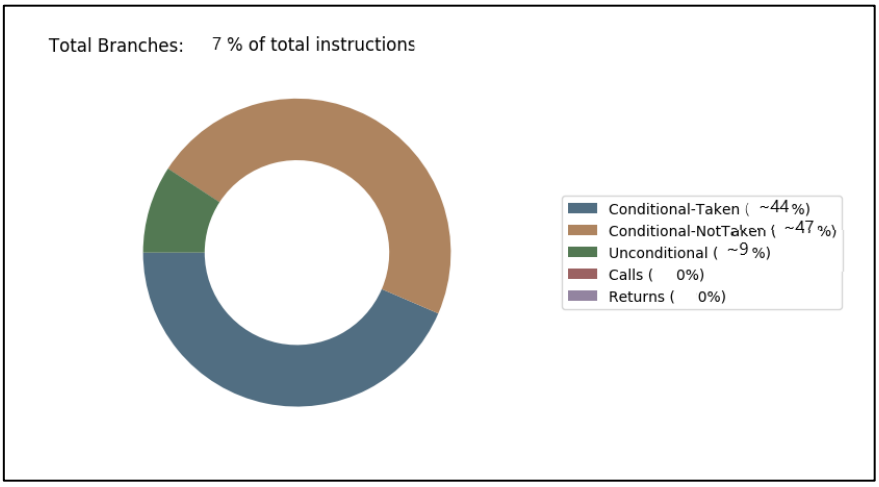
403.gcc



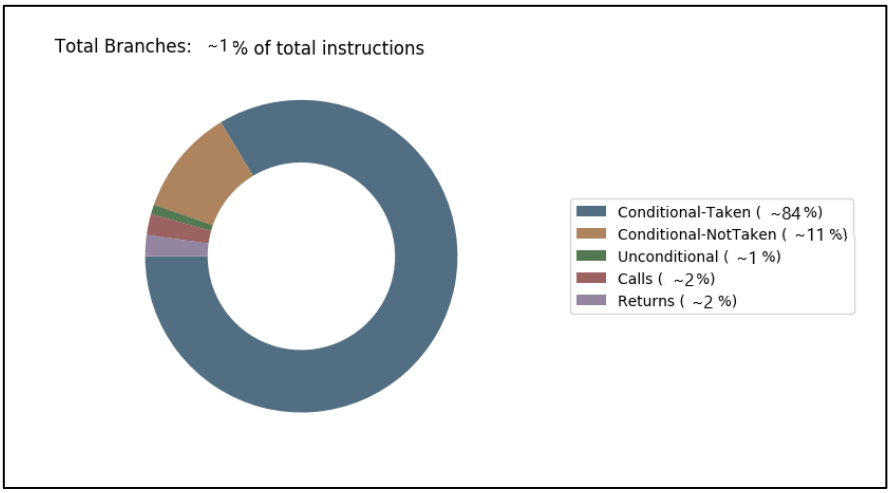
429.mcf



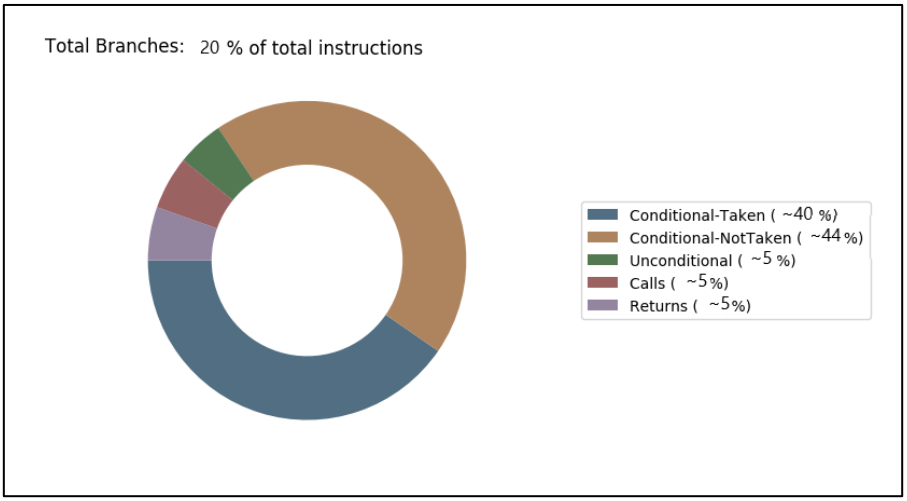
434.zeusmp



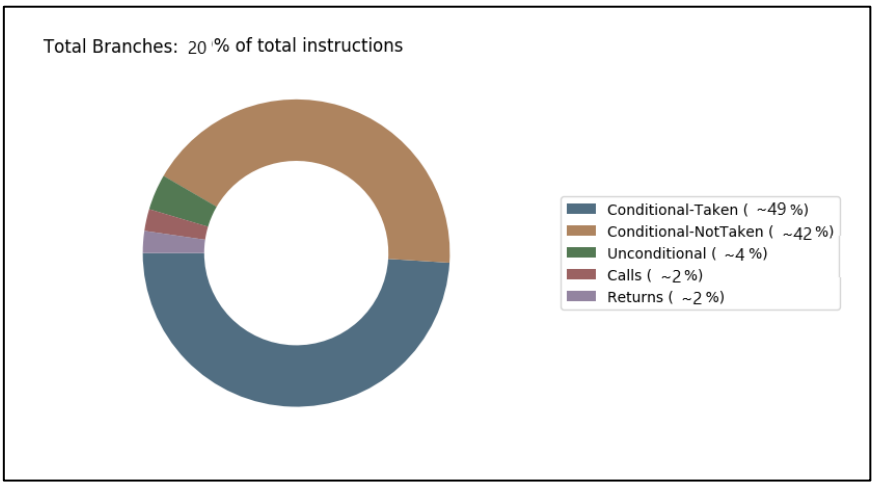
436.cactusADM



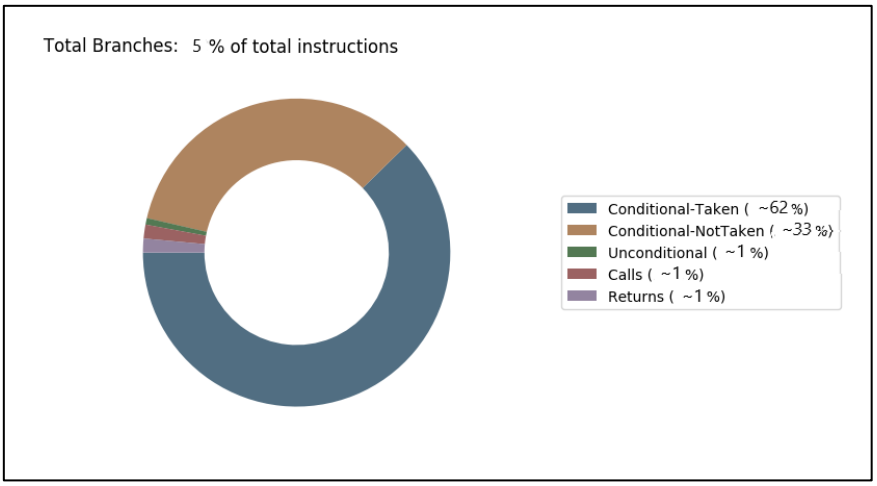
445.gobmk



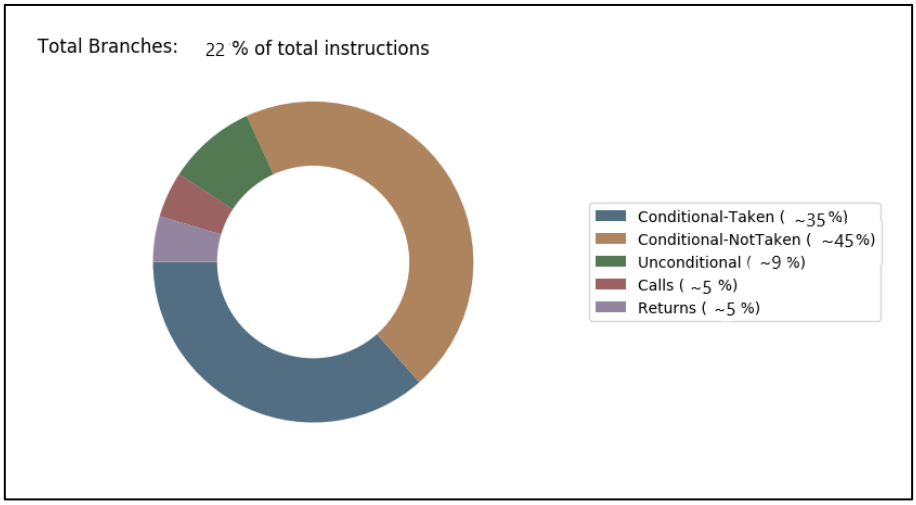
450.soplex



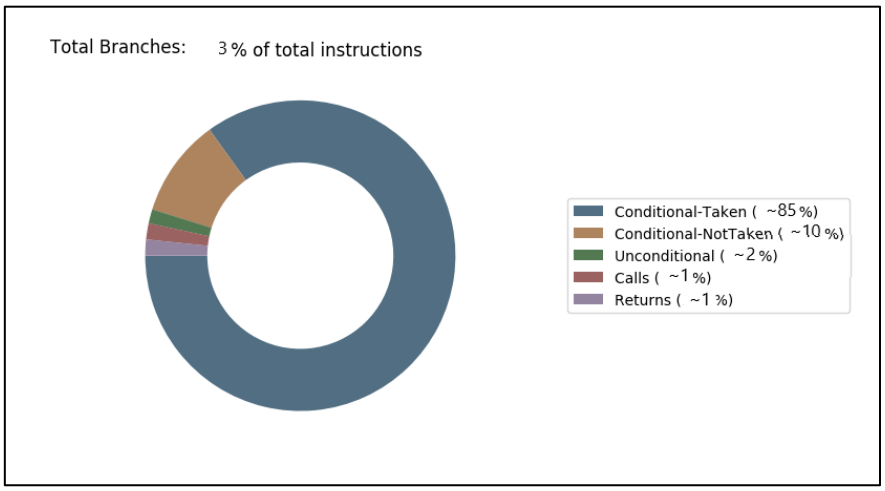
456.hmmer



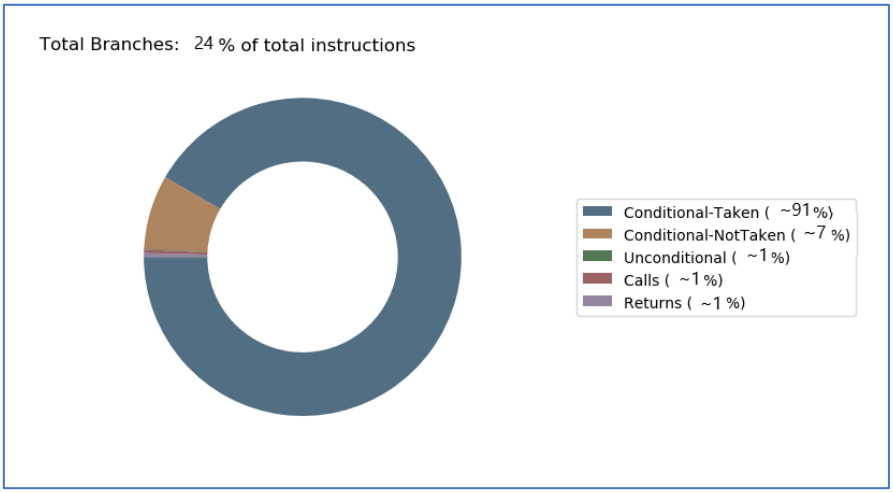
458.sjeng



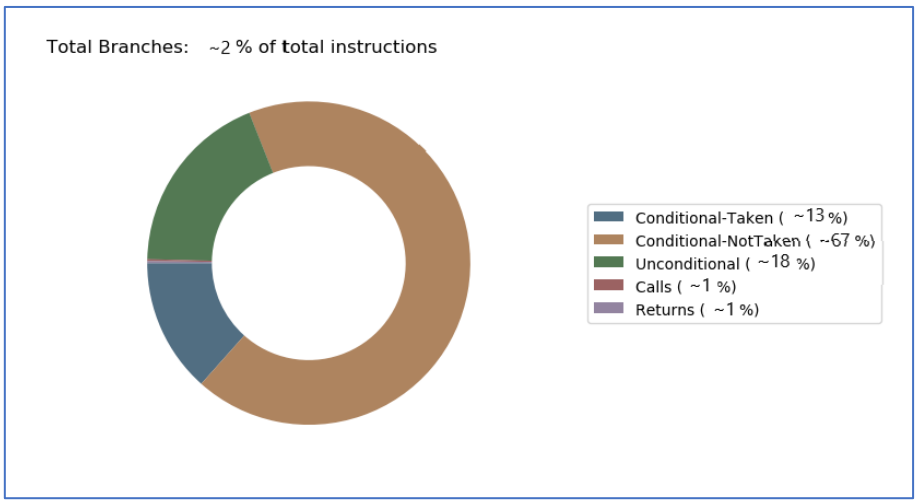
459.GemsFDTD



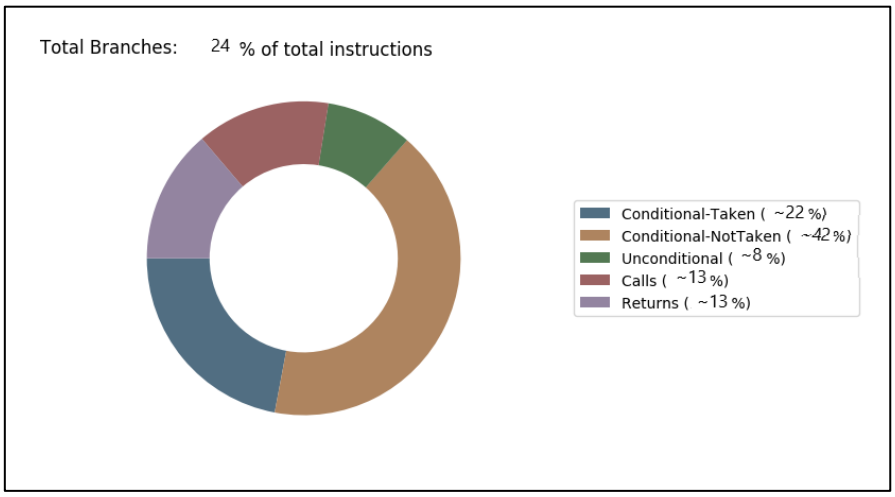
462.libquantum



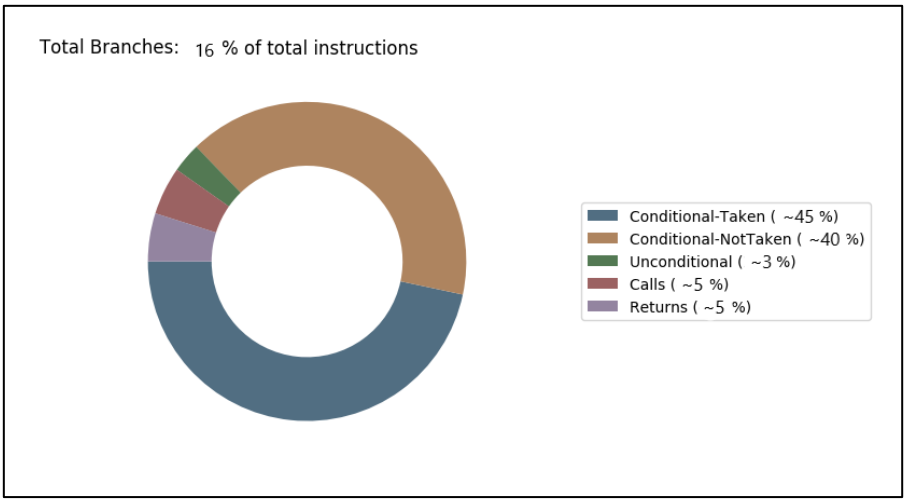
470.lbm



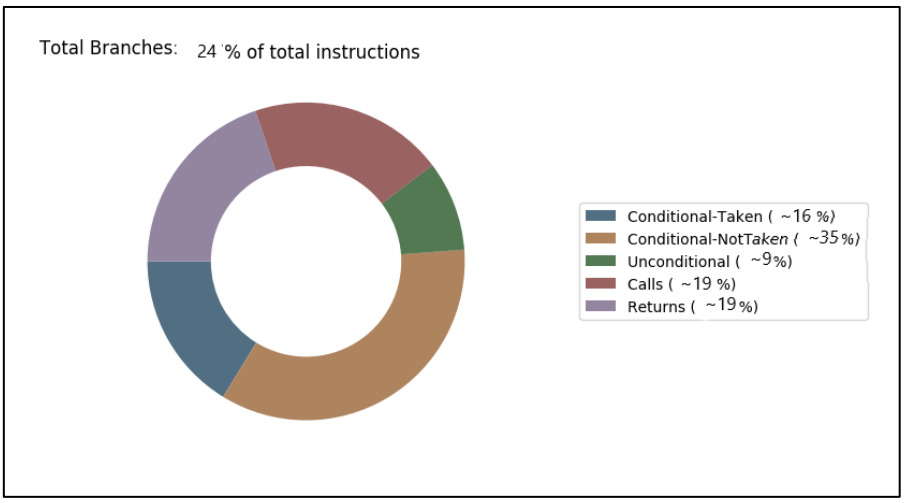
471.omnetpp



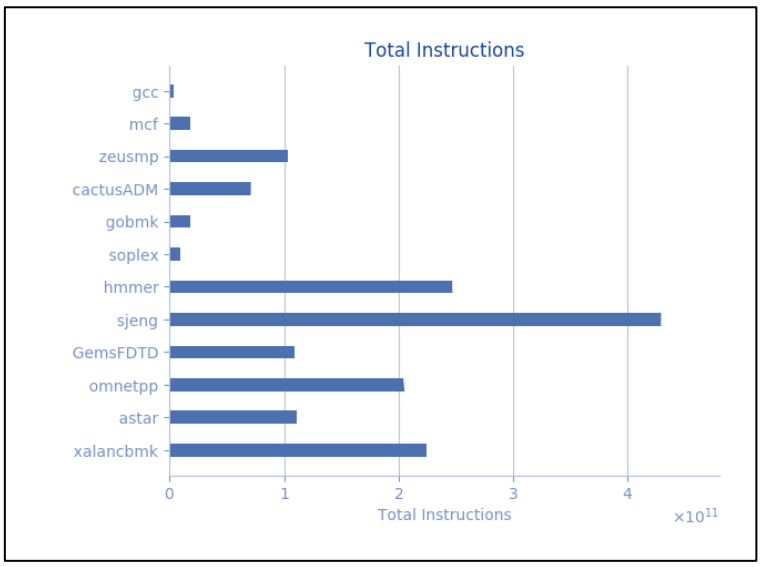
473.astar



483.xalancbmk



Total



Παρατηρήσεις & Συμπεράσματα:

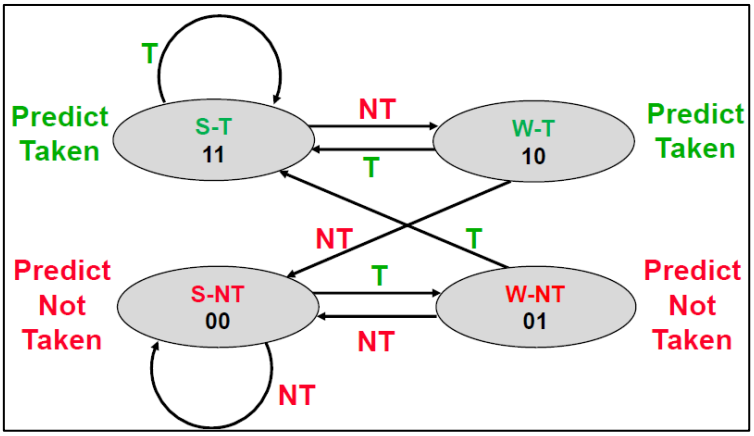
Παρατηρείται ότι στα μετροπρογράμματα, οι εντολές άλματος αποτελούν ένα σημαντικό ποσοστό των συνολικών εντολών που εκτελούνται. Υπάρχουν benchmarks όπου οι εντολές άλματος είναι περίπου το 20%-30% των συνολικών εντολών (403.gcc, 429.mcf, 445.gobmk, 450.soplex, 458.sjeng, 483.xalancbmk, 473.astar) και benchmarks όπου οι εντολές άλματος είναι σημαντικά λιγότερες κάτω του 5% των συνολικών (459.GemsFDTD και 436.cactusADM).

Σχετικά με την κατηγορία των αλμάτων, τα περισσότερα είναι είτε Conditional Taken, είτε Conditional NotTaken. Αρκετά λιγότερα είναι τα Unconditional Branches, Calls, Returns. Τέλος, για το σύνολο των εντολών, όπως προκύπτει στο σχετικό ραβδόγραμμα, υπάρχουν benchmarks με μικρό πλήθος εντολών (gcc, mcf, zeusmp, soplex, gobmk) και άλλα με αρκετά μεγάλο πλήθος εντολών τα οποία απαιτούν και μεγαλύτερο χρόνο εκτέλεσης (hmmer, sjeng, omnetop).

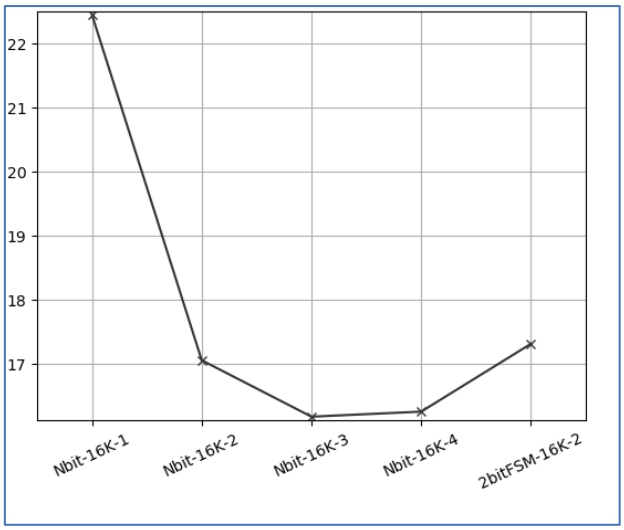
4.2 Μελέτη των N-bit predictors

Γίνεται μελέτη της απόδοσης των n-bits predictors χρησιμοποιώντας την υλοποίηση τους στο cslab_branch.cpp.

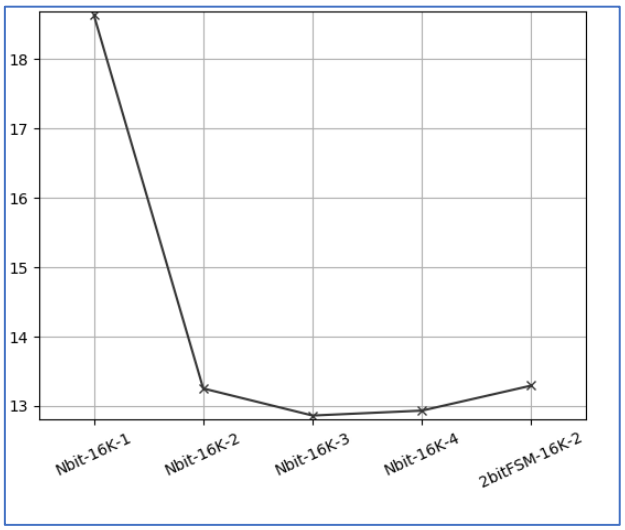
(i) *Διατηρώντας σταθερό τον αριθμό των BHT entries και ίσο με 16K, γίνεται προσομοίωση των n-bit predictors, για N = 1, 2, 3, 4. Τα n-bits υλοποιούν ένα saturating up-down counter. Για N = 2 υλοποιείται επιπλέον και το παρακάτω εναλλακτικό FSM (ως 2β). Τέλος, συγκρίνονται οι 5 predictors χρησιμοποιώντας τη μετρική direction Mispredictions Per Thousand Instructions (direction MPKI).*



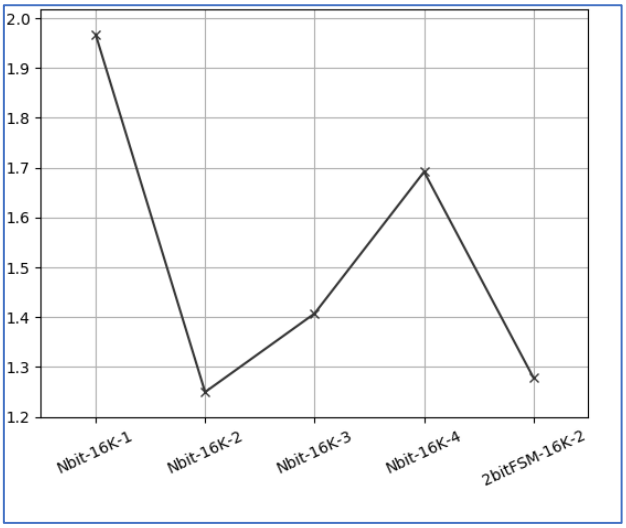
403.gcc



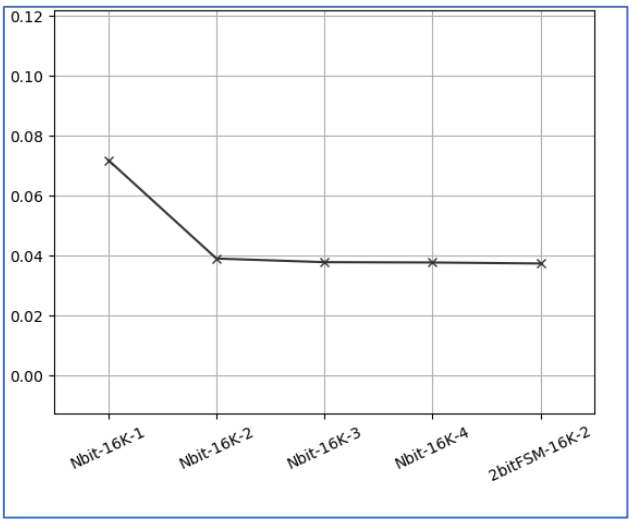
429.mcf



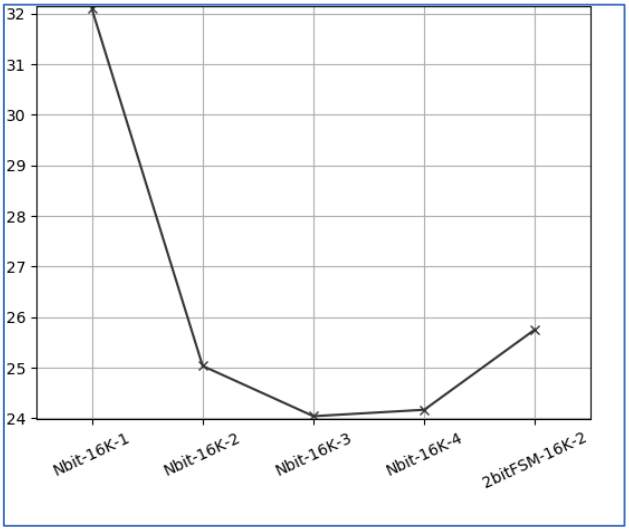
434.zeusmp



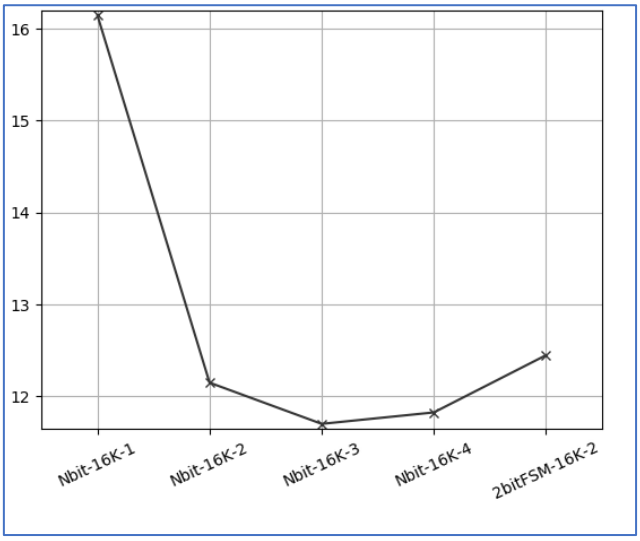
436.cactusADM



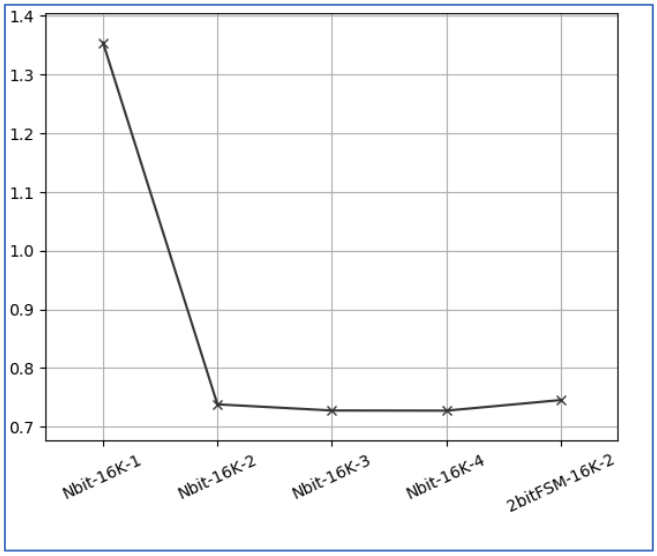
445.gobmk



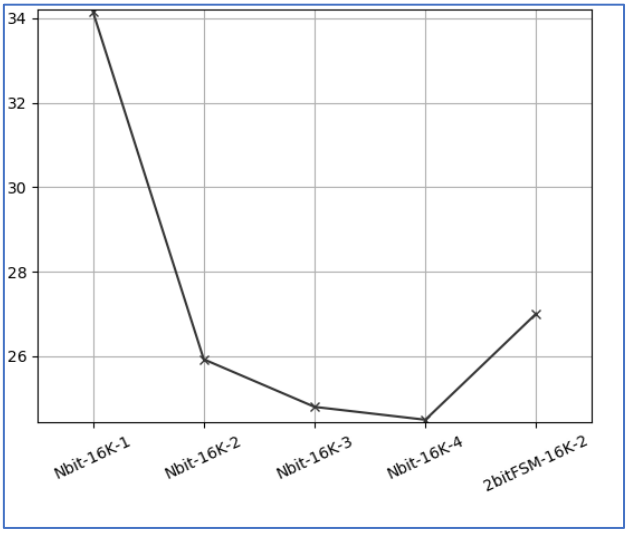
450.soplex



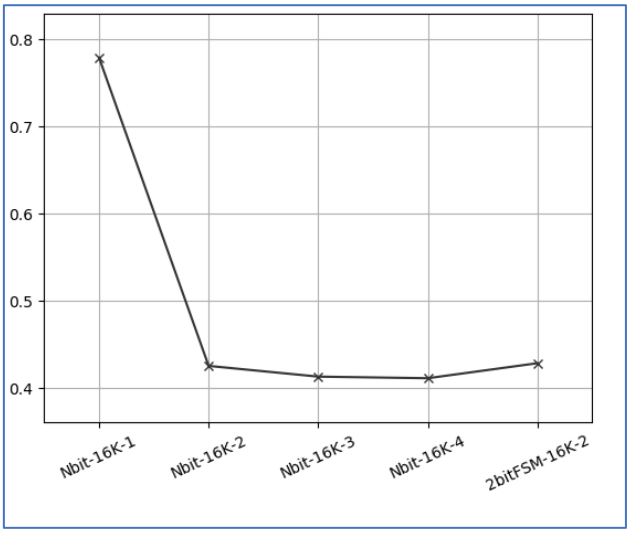
456.hmmer



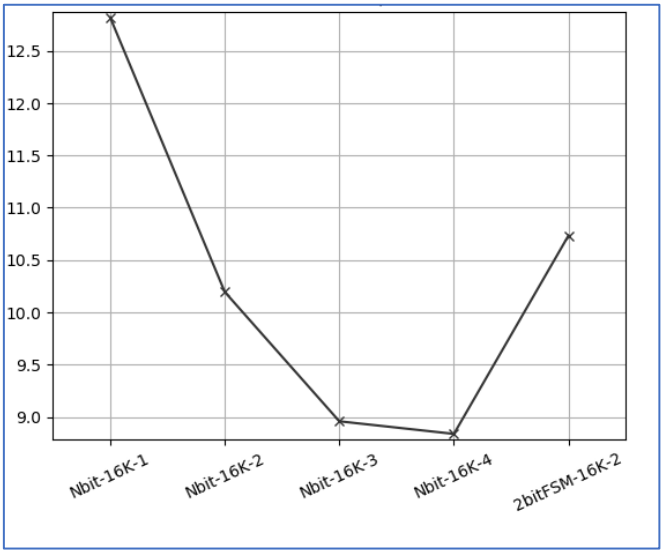
458.sjeng



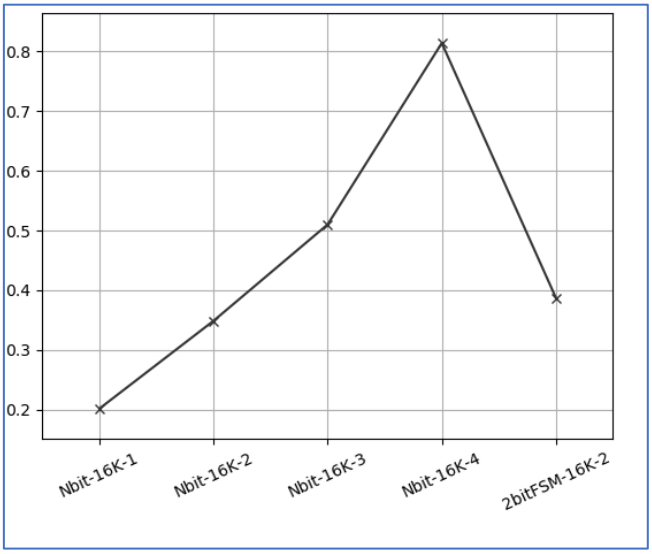
459.GemsFDTD



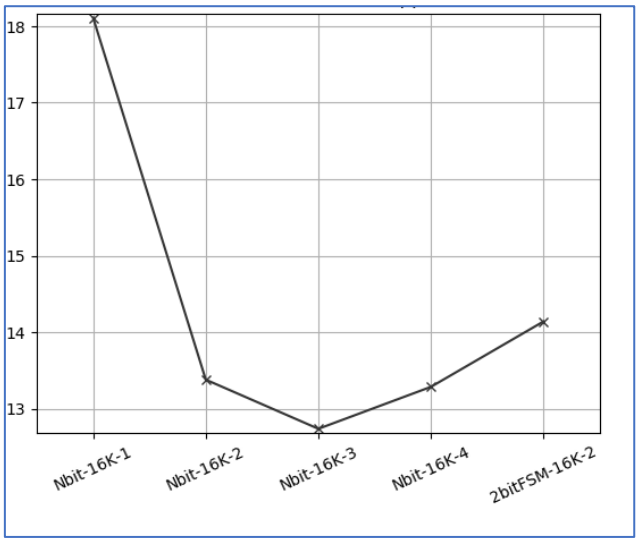
462.libquantum



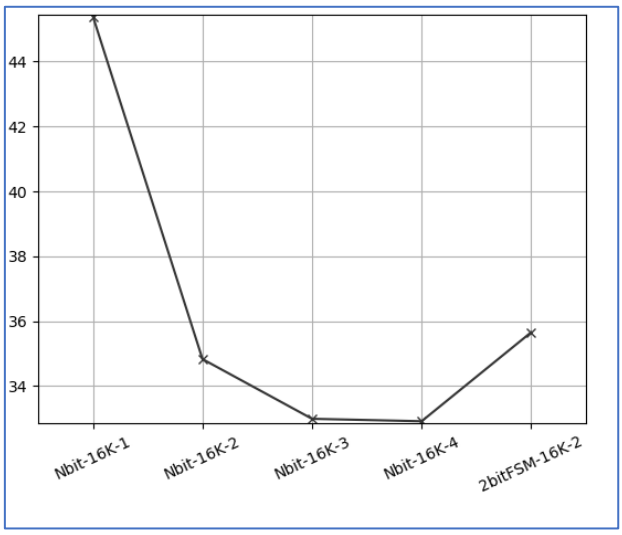
470.lbm



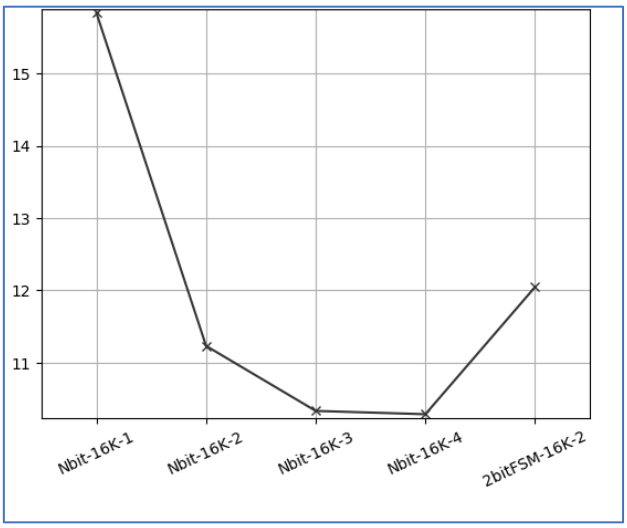
471.omnetpp



473.astar



483.xalancbmk



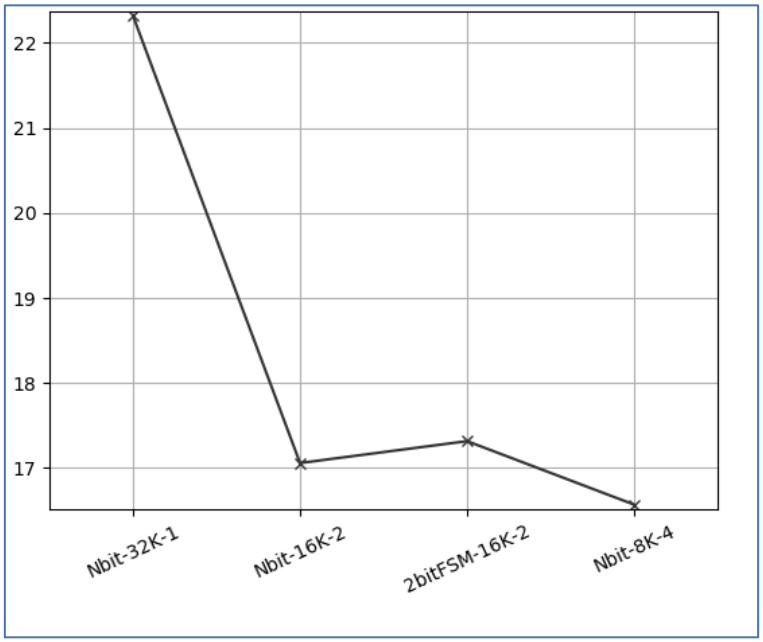
Παρατηρήσεις & Συμπεράσματα:

Παρατηρείται ότι, από της μορφές των καμπυλών στα παραπάνω διαγράμματα, τα περισσότερα benchmarks παρουσιάζουν βελτίωση καθώς το πλήθος των bits του predictor αυξάνει, δηλαδή η μετρική dMPKI φθίνει καθώς τα bits αυξάνονται από 1 σε 2 και από 2 σε 3. Η μετάβαση από 3bit σε 4bit δεν επιφέρει πάντα βελτίωση. Η μόνη διαφορετική ως προς την μορφή καμπύλη αντιστοιχεί στο μετροπρόγραμμα 434.zeusmp για το οποίο το μικρότερο MPKI αντιστοιχεί σε 2-bit predictor. Ωστόσο πρέπει να σημειωθεί πως για το εν λόγω μετροπρόγραμμα η τιμή του MPKI είναι ήδη αρκετά χαμηλή και η διαφοροποίηση της επίδοσης που επέρχεται με τη χρήση διαφορετικών Nbit-Predictors είναι αρκετά μικρή (εύρος 1.2 έως 2.0 Misses Per KILOInstructions), άρα δεν επηρεάζει και πολύ.

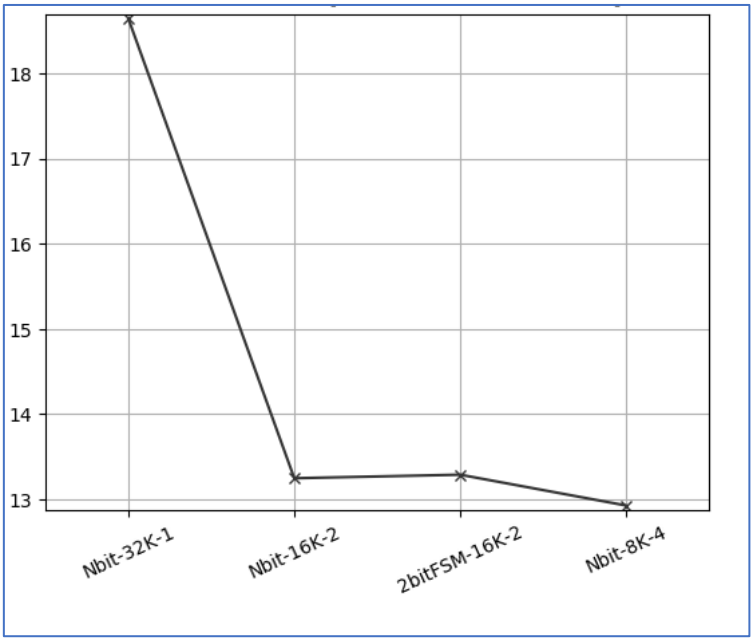
Αναφορικά με το FSM που υλοποιήθηκε, αποδίδει καλύτερα μονάχα σε σχέση με τον 1bit Predictor. Ο αντίστοιχος 2bit Predictor που υλοποιείται ως saturating up-down counter αποδίδει πάντα καλύτερα σε σχέση με τον 2-bit FSM Predictor.

(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, εκτελούνται ξανά οι προσομοιώσεις για τα 14 benchmarks, θέτοντας N=1, 2, 2β, 4 και τον κατάλληλο αριθμό entries. Δίνεται το κατάλληλο διάγραμμα και εξηγούνται οι μεταβολές που παρατηρούνται. Ποιος predictor θα επιλεγόταν ως η βέλτιστη επιλογή;*

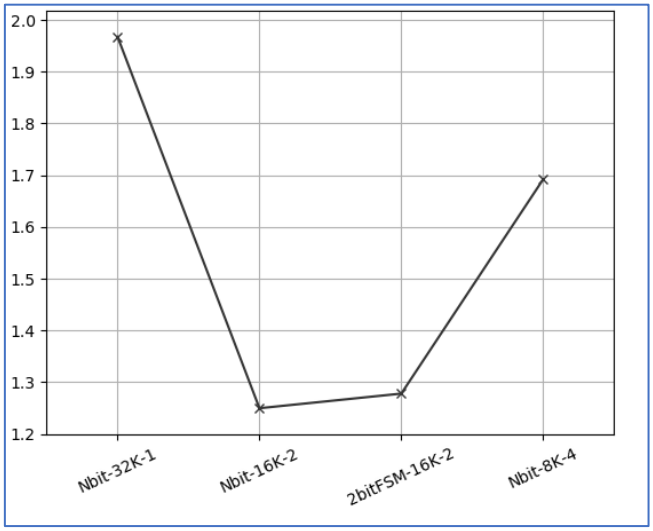
403.gcc



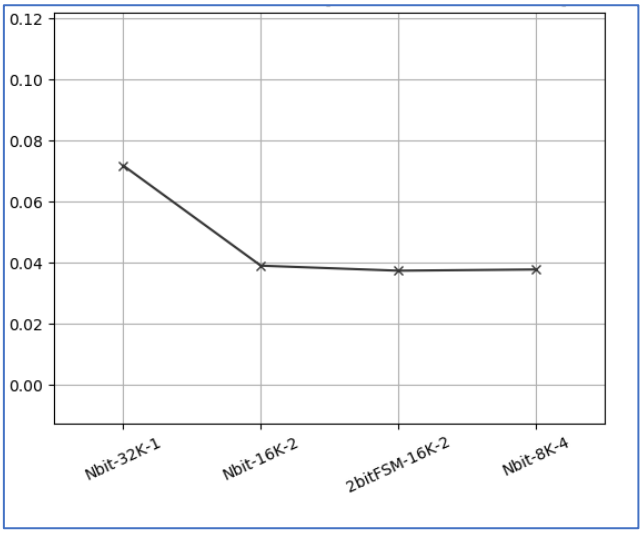
429.mcf



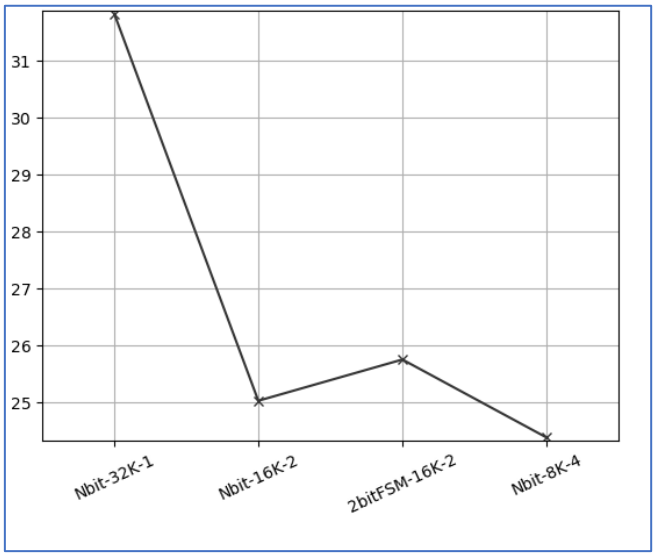
434.zeusmp



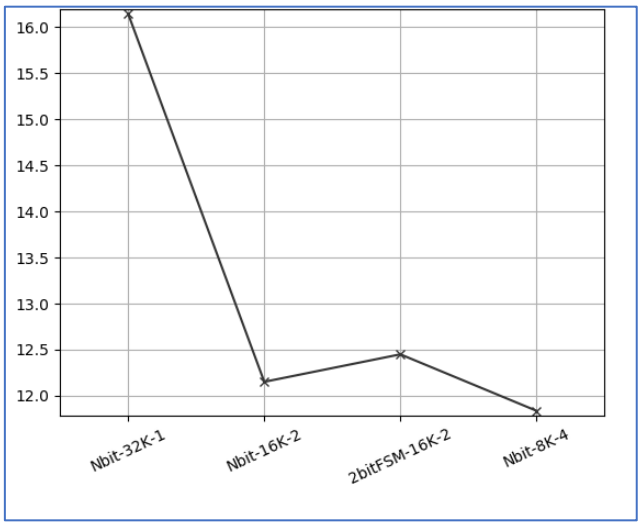
436.cactusADM



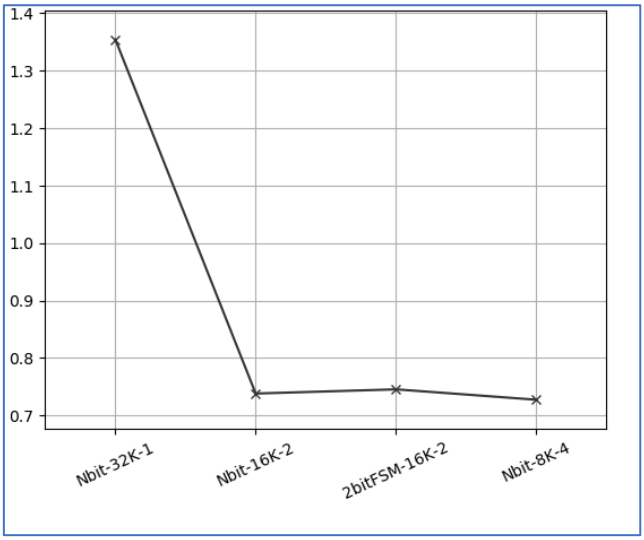
445.gobmk



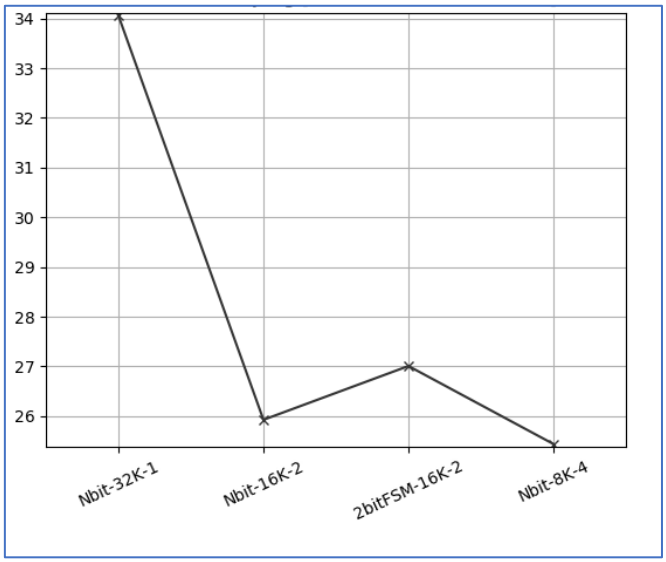
450.soplex



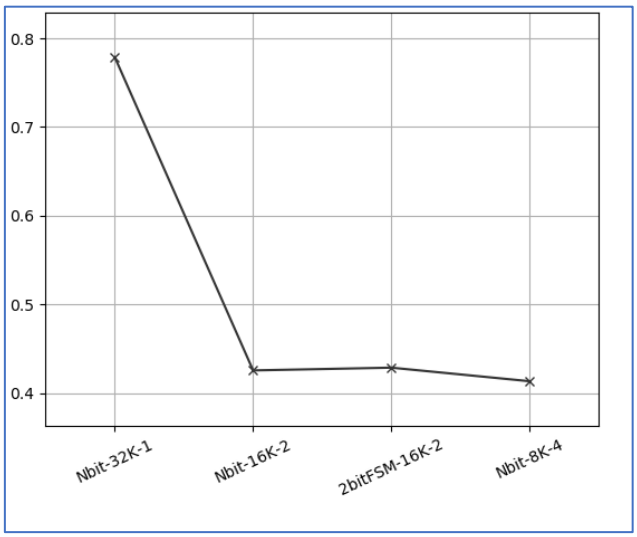
456.hmmer



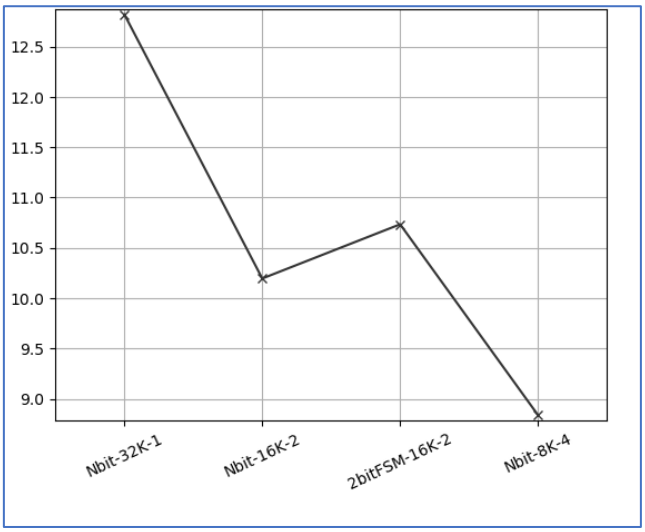
458.sjeng



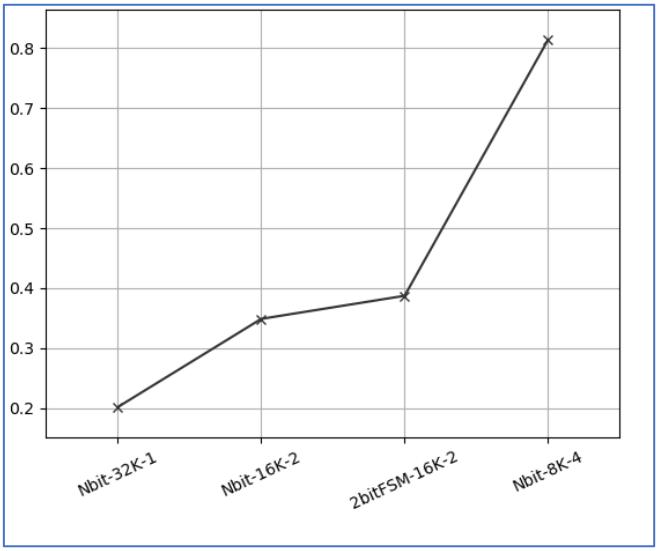
459.GemsFDTD



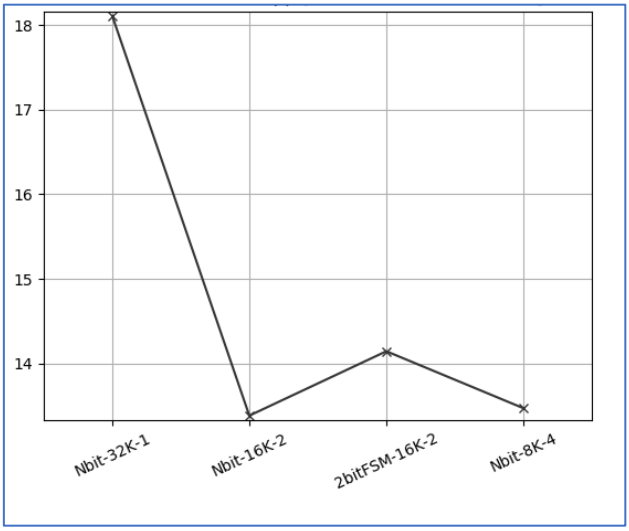
462. libquantum



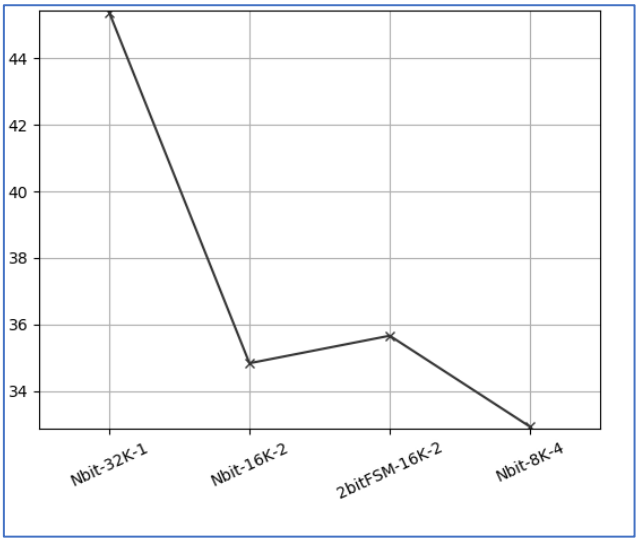
470.lbm



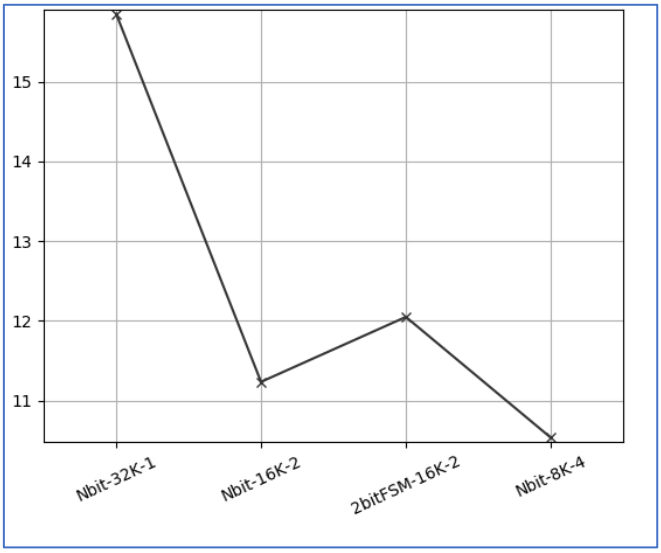
471.omnetpp



473.astar



483.xalancbmk



Παρατηρήσεις & Συμπεράσματα:

Παρατηρείται ότι και στους νέους συνδυασμούς για σταθερό υλικό, η καμπύλες μοιάζουν με του προηγούμενου ερωτήματος και τα συμπεράσματα είναι ανάλογα. Από της μορφές των καμπυλών στα παραπάνω διαγράμματα προκύπτει πως τα περισσότερα benchmarks παρουσιάζουν βελτίωση καθώς το πλήθος των bits του predictor αυξάνει, δηλαδή η μετρική dMPKI φθίνει καθώς τα bits αυξάνονται, παρά την μείωση του πλήθος των predictors. Η μόνη διαφορετική ως προς την μορφή καμπύλη αντιστοιχεί στο μετροπρόγραμμα 434.zeusmp για το οποίο το μικρότερο MPKI αντιστοιχεί σε 2-bit predictor. Λαμβάνοντας υπόψιν το διάγραμμα των γεωμετρικών μέσων, αλλά και την επίδοση με βάση τα επιμέρους διαγράμματα, προκύπτει πως καλύτερη επιλογή είναι ο 4-bit Predictor με 8K entries. Εναλλακτικά θα μπορούσε να επιλεχθεί και ο 2-bit Predictor με 16K entries, αφού αποδίδει επίσης σχετικά καλά.

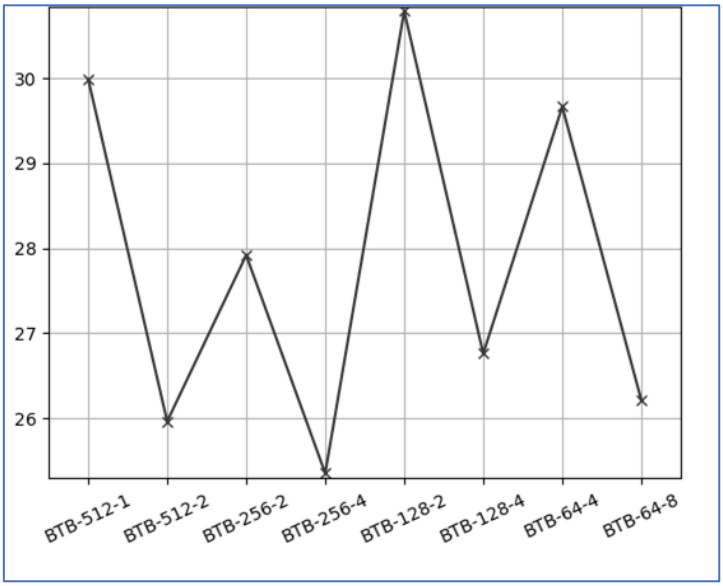
4.3 Μελέτη του BTB

Ζητείται η υλοποίηση ενός BTB και η μελέτη της ακρίβειας πρόβλεψής του για τις ακόλουθες περιπτώσεις:

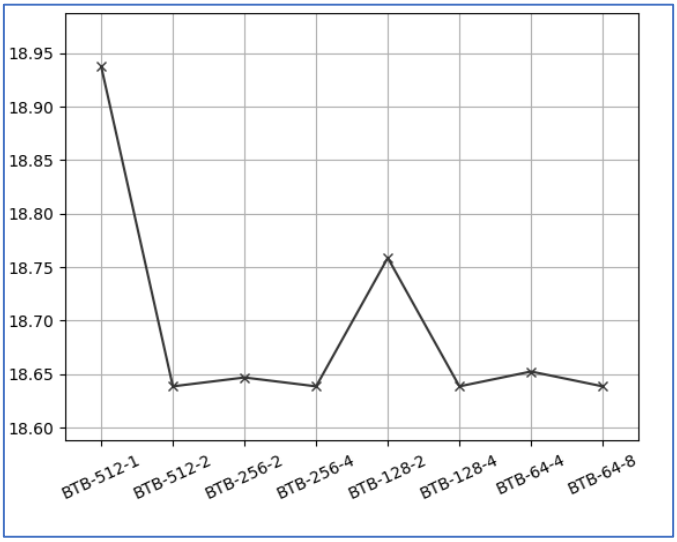
btb entries	***btb associativity***
512	*1, 2*
256	*2, 4*
128	*2, 4*
64	*4, 8*

Ζητείται η προσομοίωση για τα benchmarks που παρέχονται και να δοθούν τα κατάλληλα διαγράμματα. Υπενθυμίζεται ότι για τον BTB υπάρχουν 2 περιπτώσεις misses. Η πρώτη είναι direction misprediction και η δεύτερη target misprediction στην περίπτωση ενός direction hit. Πώς εξηγείται η διαφορά επίδοσης ανάμεσα στις διαφορετικές περιπτώσεις; Να επιλεχθεί η καλύτερη οργάνωση για το BTB.

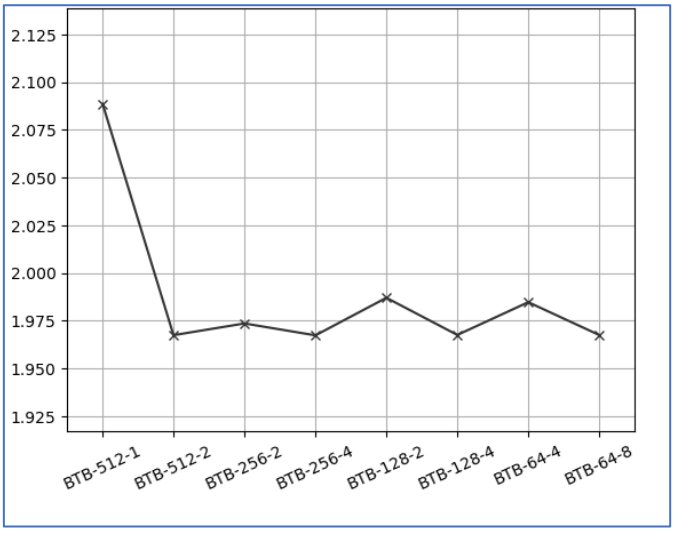
403.gcc



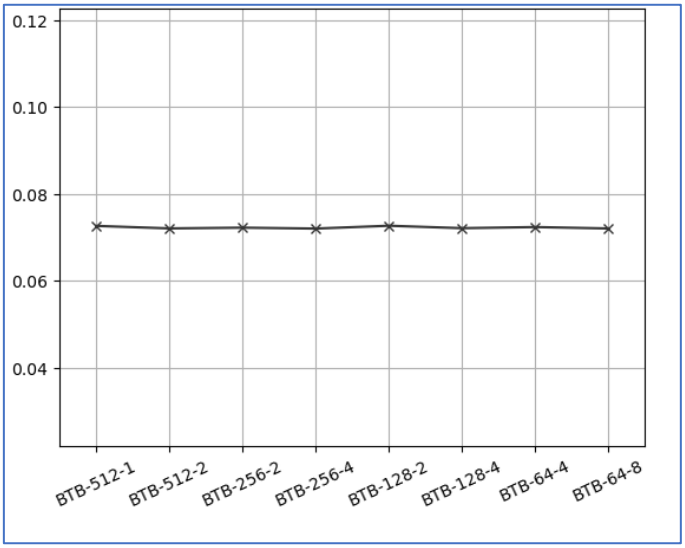
429.mcf



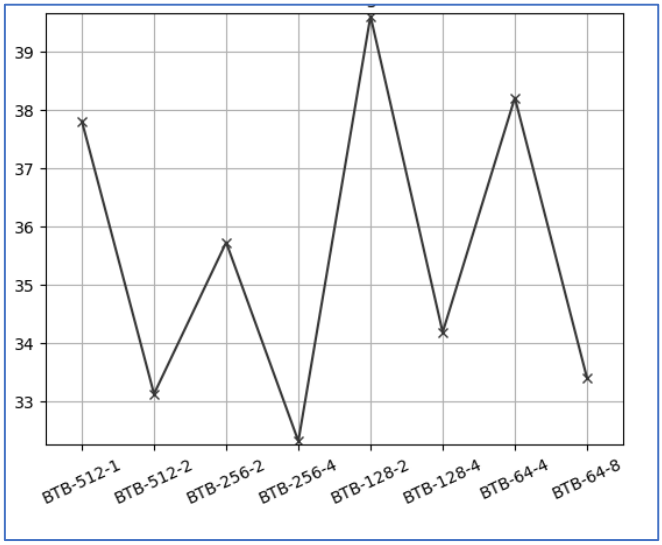
434.zeusmp



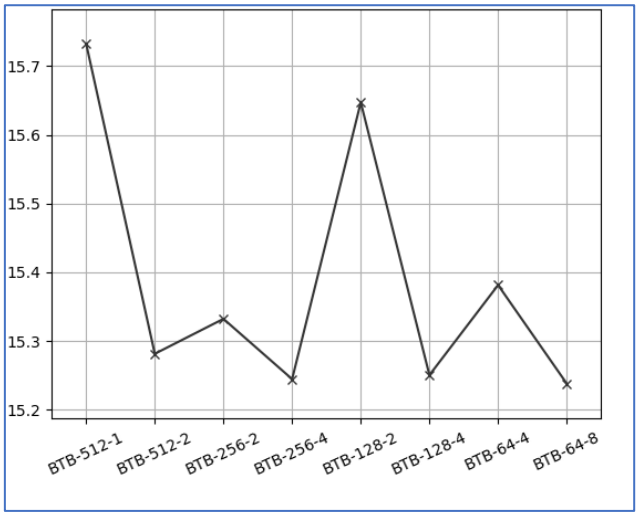
436.cactusADM



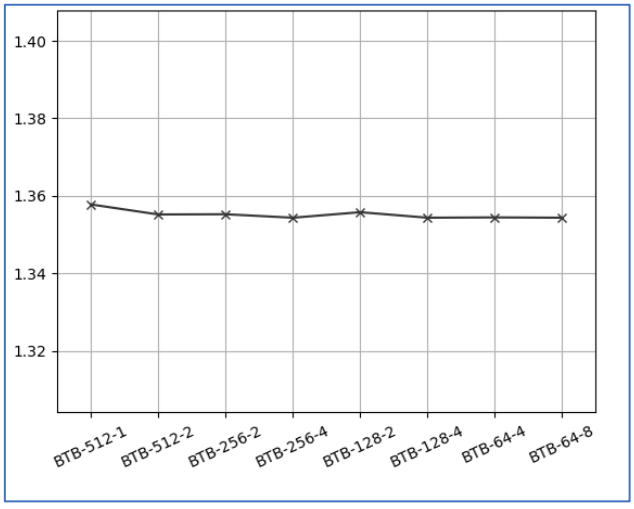
445.gobmk



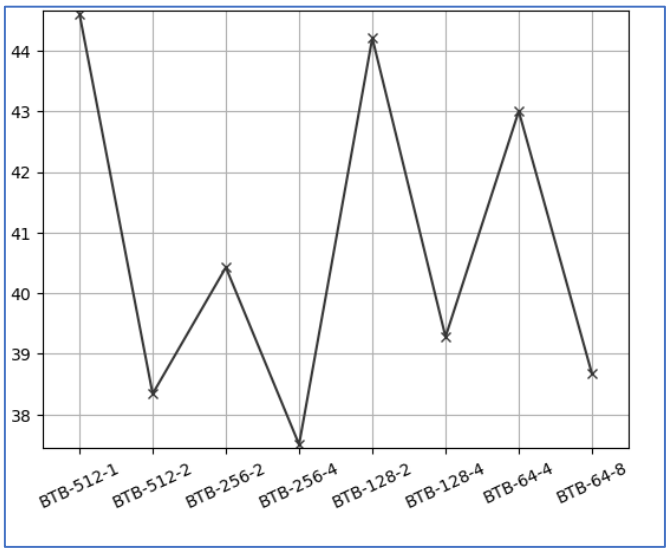
450.soplex



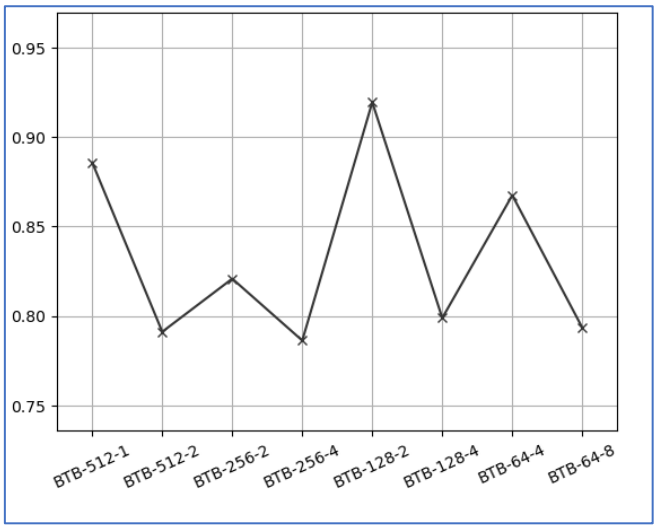
456.hmmer



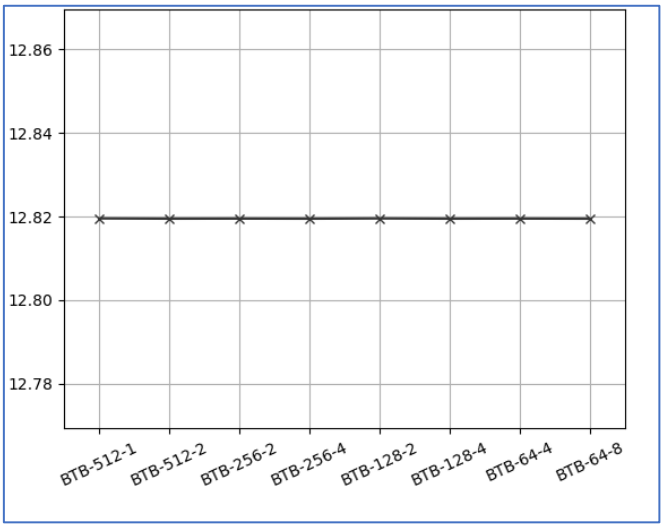
458.sjeng



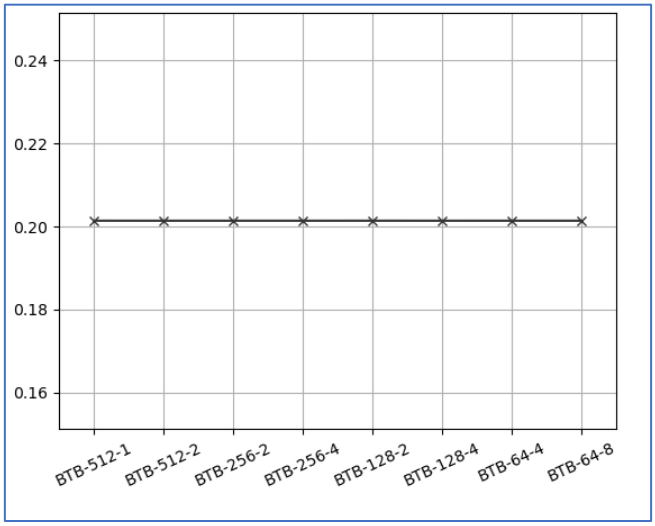
459.GemsFDTD



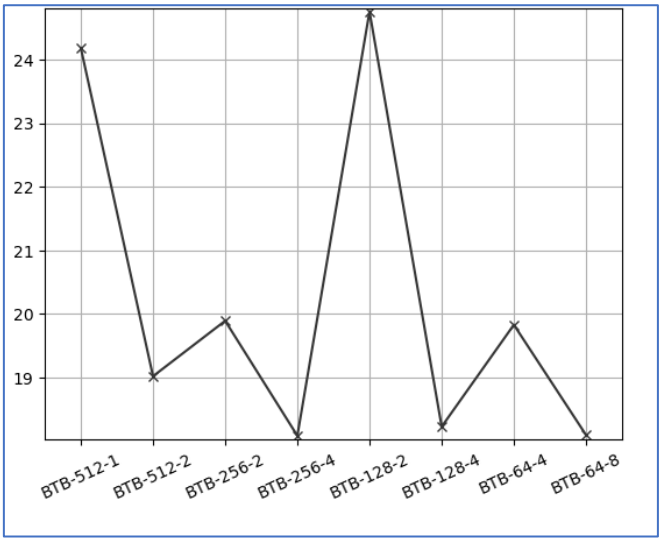
462.libquantum



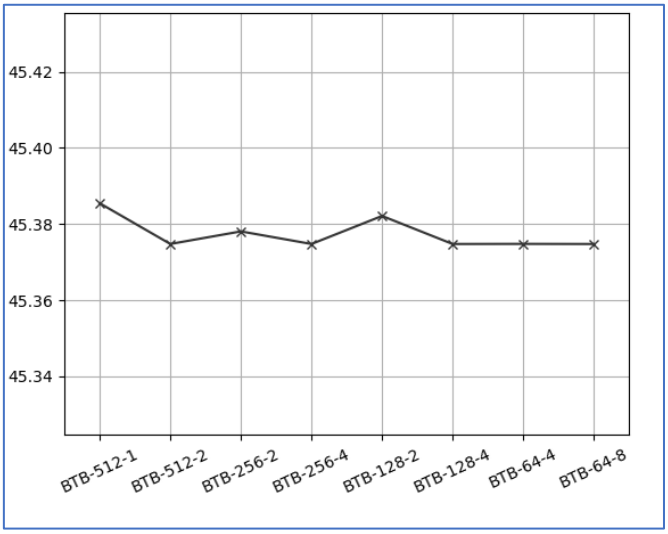
470.lbm



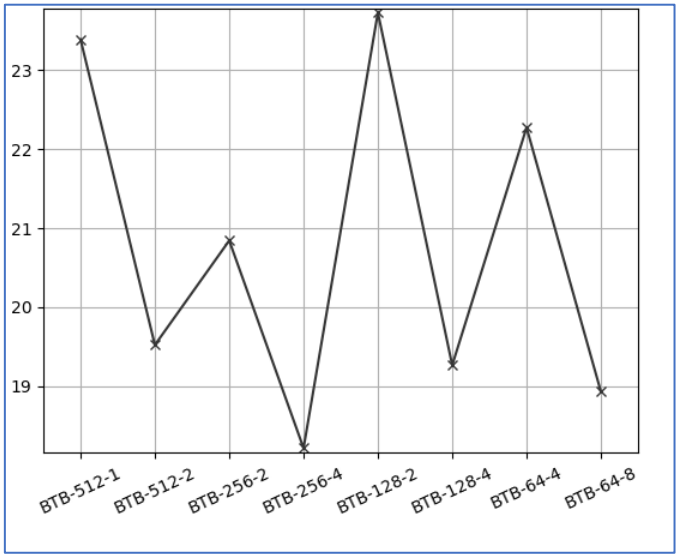
471.omnetpp



473.astar



483.xalancbmk



Παρατηρήσεις & Συμπεράσματα:

Παρατηρείται ότι στα παραπάνω διαγράμματα χρησιμοποιείται ως μετρική τα επιμέρους prediction misses + target misspredictions per KILOInstructions.

Από τα επιμέρους διαγράμματα διαπιστώνεται ότι για σταθερό πλήθος entries (table lines x associativity) η αύξηση του associativity επιφέρει βελτίωση. Την μικρότερη τιμή misses επιτυγχάνουν οι συνδυασμοί BTB-256-2 και BTB-64-4. Μάλιστα ο BTB-256-2 φαίνεται να έχει καθολικό προβάδισμα στην απόδοση όπως φαίνεται και στο διάγραμμα των γεωμετρικών μέσων. Αξίζει να σημειωθεί πως ο συνδυασμός BTB-8-8 έχει τη χειρότερη απόδοση, και άρα η υπερβολική αύξηση του associativity μειώνοντας τα table lines μετά από ένα όριο δε δρα βελτιωτικά.

Συνοπτικά, καλύτερη επιλογή είναι ο BTB Predictor με 512 entries, οργανωμένα σε 256 lines και associativity 2.

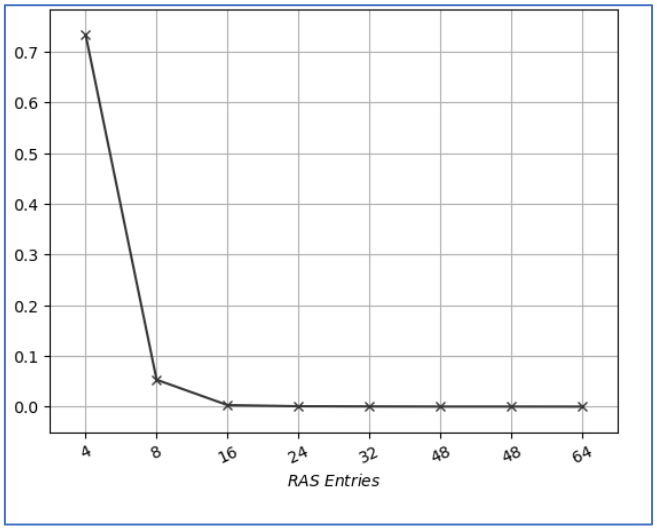
4.4 Μελέτη του RAS

Χρησιμοποιώντας την υλοποίηση της RAS (ras.h) που δίνεται, μελετάται το ποσοστό αστοχίας για τις ακόλουθες περιπτώσεις:

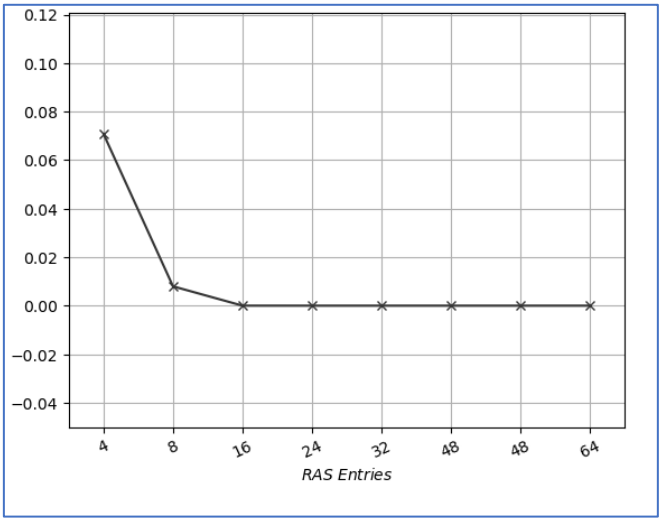
***Αριθμός εγγραφών στη RAS**: 4, 8, 16, 24, 32, 48, 64*

Ζητείται η προσομοίωση για τα benchmarks που παρέχονται και να δοθούν τα κατάλληλα διαγράμματα εξηγώντας τις μεταβολές που παρατηρείτε. Επίσης, γίνεται επιλογή του κατάλληλου μεγέθους για το RAS.

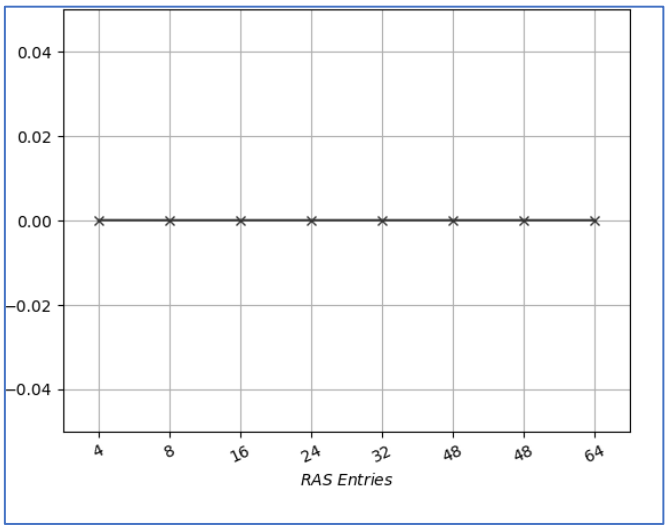
403.gcc



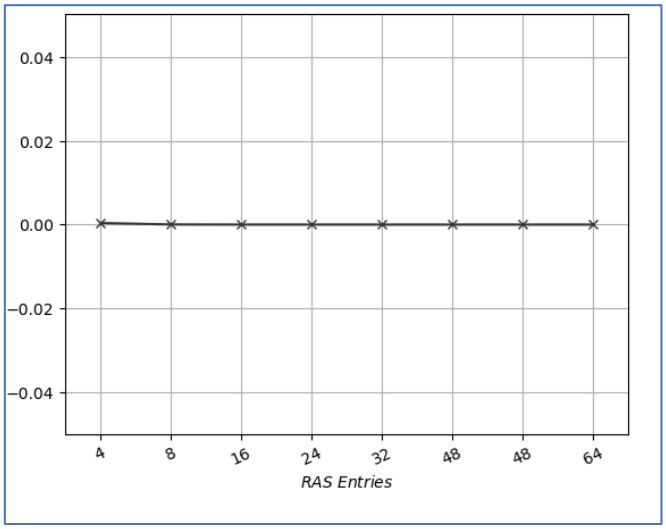
429.mcf



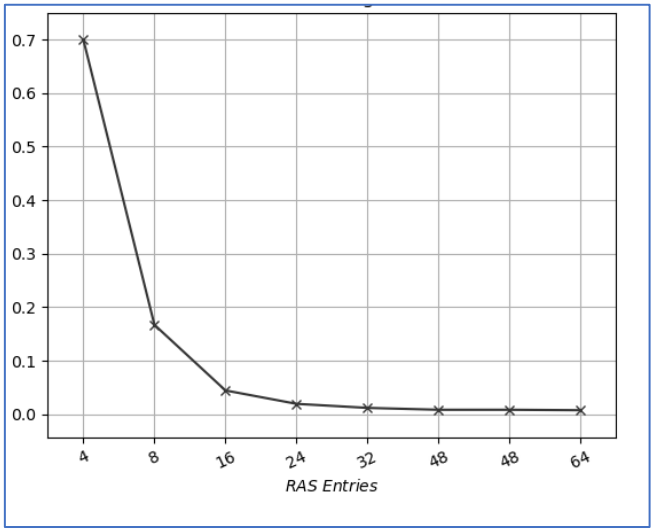
434.zeusmp



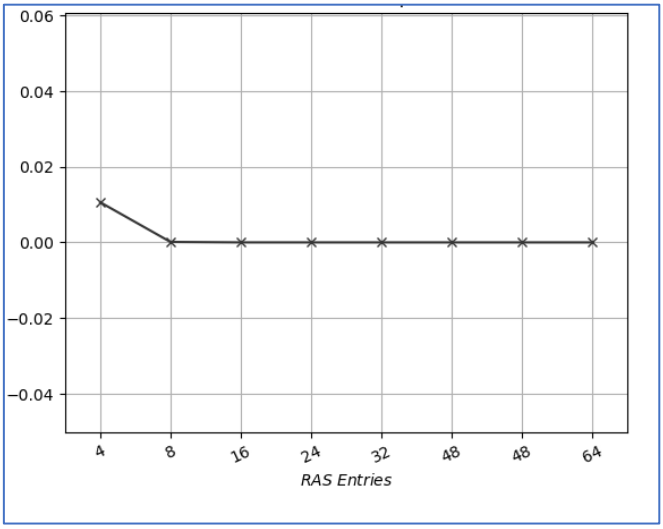
436.cactusADM



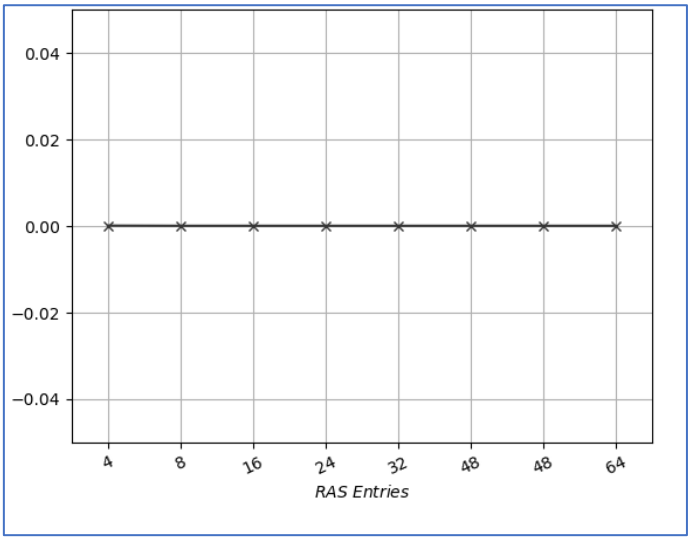
445.gobmk



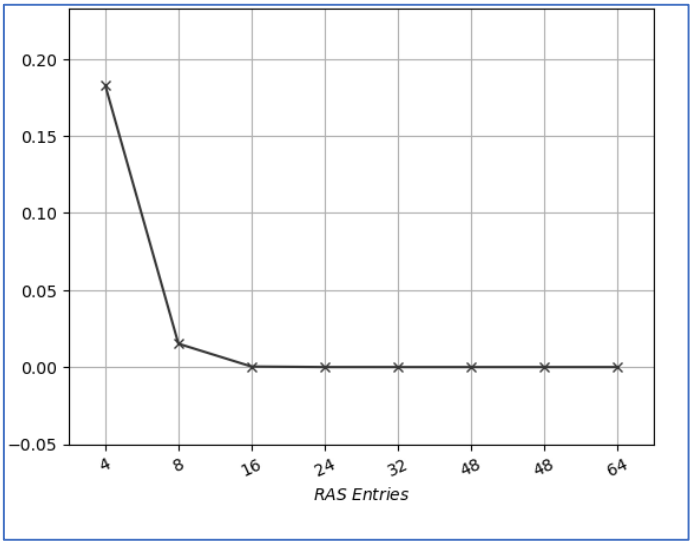
450.soplex



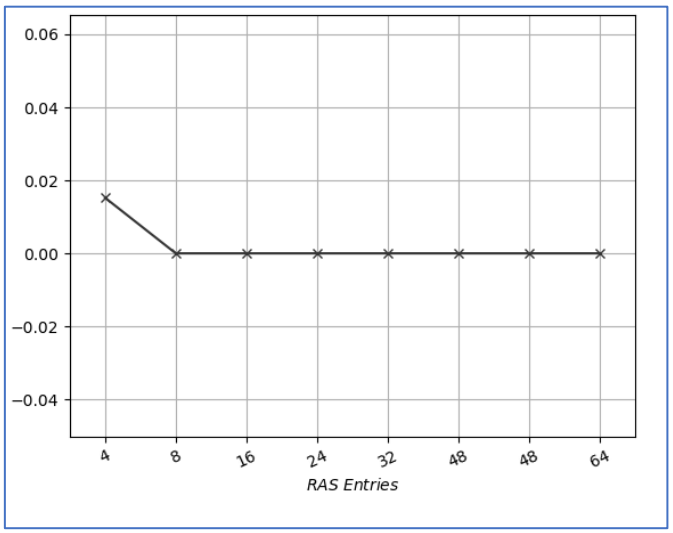
456.hmmer



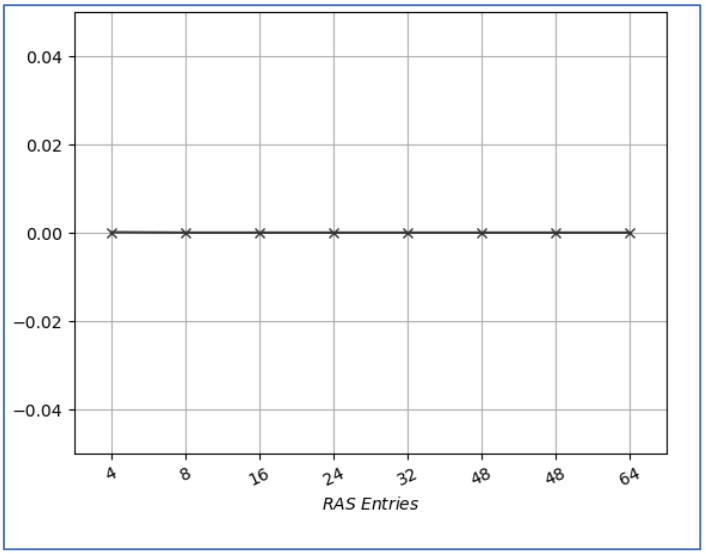
458.sjeng



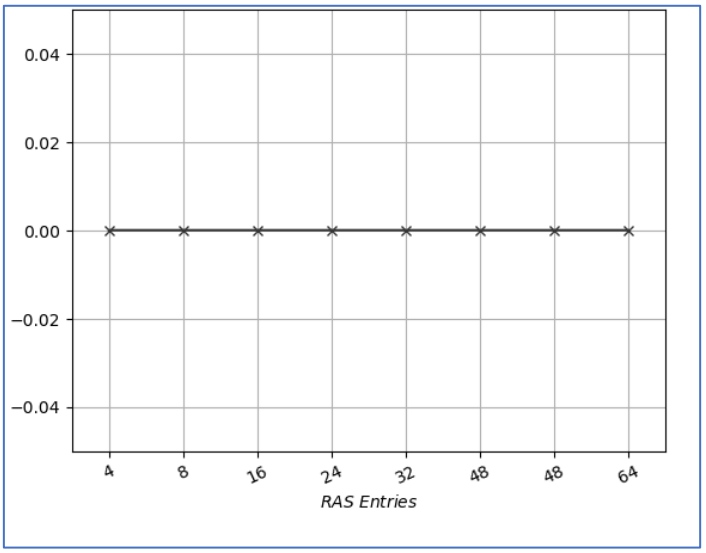
459.GemsFDTD



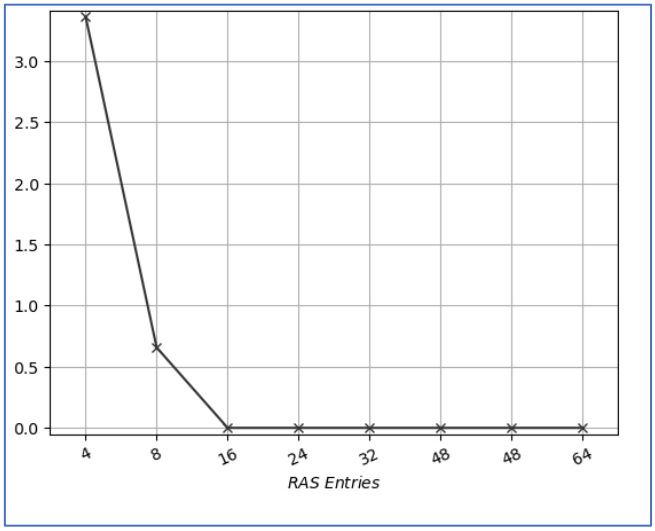
462.libquantum



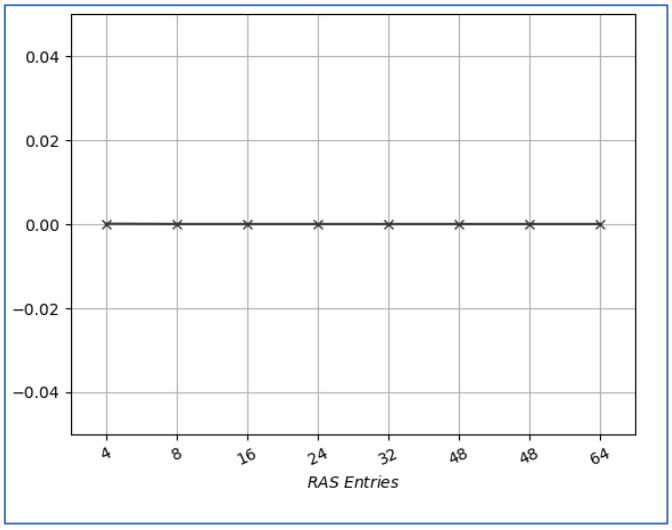
470.lbm



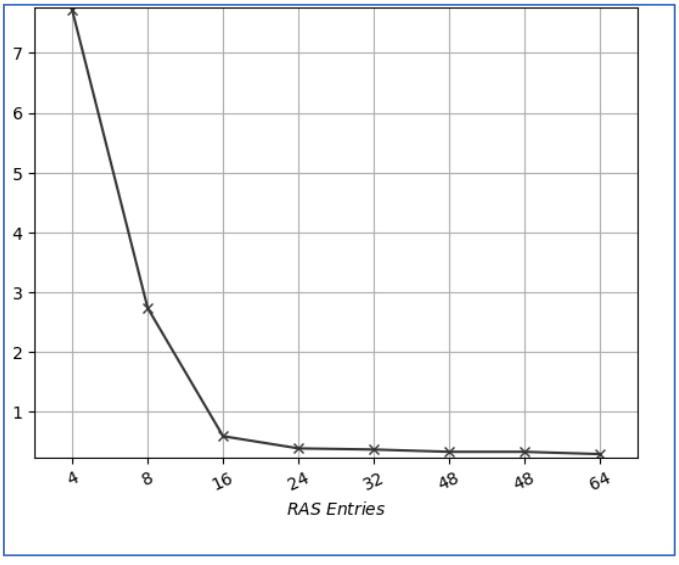
471.omnetpp



473.astar



483.xalancbmk



Παρατηρήσεις & Συμπεράσματα:

Παρατηρείται ότι για αριθμό εγγραφών στη RAS ίσο με 1, σε σχεδόν όλα τα benchmarks υπάρχει πολύ υψηλό MPKI. Μόλις το RAS κάνει χρήση 2 εγγραφών, είναι εμφανής η σημαντική μείωση του MPKI. Για μεταβολή από 2 σε 4 εγγραφές, και πάλι σχεδόν σε όλα τα μετροπρογράμματα υπάρχει μείωση του MPKI, όπως είναι επιθυμητό. Για παραπάνω entries, τα MPKI είτε μειώνονται με αρκετά μικρότερο ρυθμό, είτε παραμένουν σχεδόν σταθερά. Σε κάθε περίπτωση με την αύξηση των εγγραφών βελτιώνεται η επίδοση, ωστόσο από ένα σημείο και πέρα η περεταίρω αύξηση δεν έχει νόημα καθώς η επίδοση δεν βελτιώνονται σημαντικά αναλογικά με την αύξηση του υλικού. Με βάση τα παραπάνω, οι επιλογή 16 ή 32 ή 64 εγγραφών έχει τη βέλτιστη επίδοση με κριτήριο αποκλειστικά το MPKI. Ωστόσο, δεν είναι σοφή η σπατάλη extra υλικού (που μεταφράζεται σε κόστος), όταν αυτή η δαπάνη υλικού δεν μεταφράζεται σε ανάλογη βελτίωση της απόδοσης. Συνεπώς θα επιλεχθεί το ελάχιστο υλικό που επιφέρει την καλύτερη απόδοση. Με βάση το κριτήριο αυτό, θα επιλεχθούν οι 16 εγγραφές RAS, καθώς η επίδοση είναι η βέλτιστη χωρίς την σπατάλη hardware. Σε διαφορετική περίπτωση θα μπορούσαν να επιλεχθούν και οι 8 εγγραφές RAS, με σχετικά μικρή επίπτωση στην απόδοση. Συνοπτικά, καλύτερη επιλογή είναι 16 εγγραφές στην RAS.

4.5 Σύγκριση διαφορετικών predictors

Στο κομμάτι αυτό συγκρίνονται οι παρακάτω predictors (οι predictors σε bold δε δίνονται και πρέπει να υλοποιηθούν):

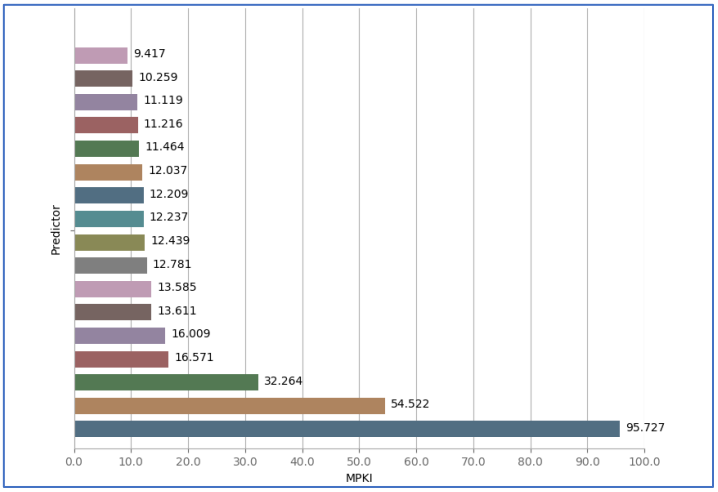
- ***Static AlwaysTaken***
- ***Static BTFNT (BackwardTaken-ForwardNotTaken)***
- *O n-bit predictor που επιλέχθηκε στο 4.2 (ii)*
- *Pentium-M predictor (δίνεται ότι το hardware overhead είναι περίπου 30K)*
- ***Local-History two-level predictors** (βλ. διαφάνειες μαθήματος) με τα εξής χαρακτηριστικά :*
  - *PHT entries = 8192*
  - *PHT n-bit counter length = 2*
  - *BHT entries = X*
  - *BHT entry length = Z*

  Να υπολογιστεί το Z ώστε το απαιτούμενο hardware να είναι σταθερό και ίσο με 32K, όταν X=2048 και X=4096.
- ***Global History two-level predictors** με τα εξής χαρακτηριστικά:*
  - *PHT entries = Z*
  - *o PHT n-bit counter length = X*
  - *o BHR length = 5, 10*

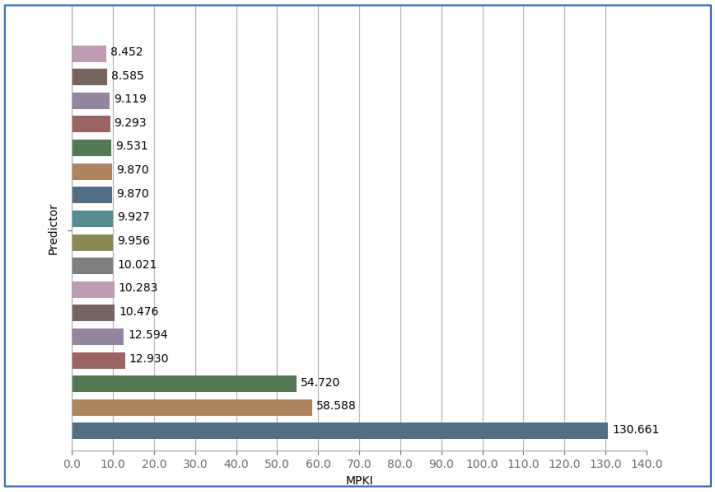
  Να υπολογιστεί το Z ώστε το απαιτούμενο hardware να είναι σταθερό και ίσο με 32K όταν X=2 και X = 4. Το κόστος του Branch History Register (5 και 10 bits) θεωρείται αμελητέο.
- ***Alpha 21264 predictor** (βλ. διαφάνειες μαθήματος – hardware overhead 29K)*
- ***Tournament Hybrid predictors** (βλ. διαφάνειες μαθήματος) με τα εξής χαρακτηριστικά:*
  - *O meta-predictor M είναι ένας 2-bit predictor με 1024 ή 2048 entries (το overhead του μπορεί να το αγνοηθεί κατά την ανάλυση)*
  - *Οι P_0, P_1 μπορούν να είναι n-bit, local-history ή global-history predictors*
  - *Οι P_0, P_1 έχουν overhead 16K ο καθένας.*
  - *Να υλοποιηθούν τουλάχιστον 4 διαφορετικούς tournament predictors*

Ζητείται προσομοίωση για τα benchmarks που παρέχονται και η σύγκρισή των παραπάνω (τουλάχιστον 15) predictors. Να δοθούν τα κατάλληλα διαγράμματα. Ποιος predictor θα μπορούσε να επιλεχθεί για την υλοποίηση στο σύστημα;

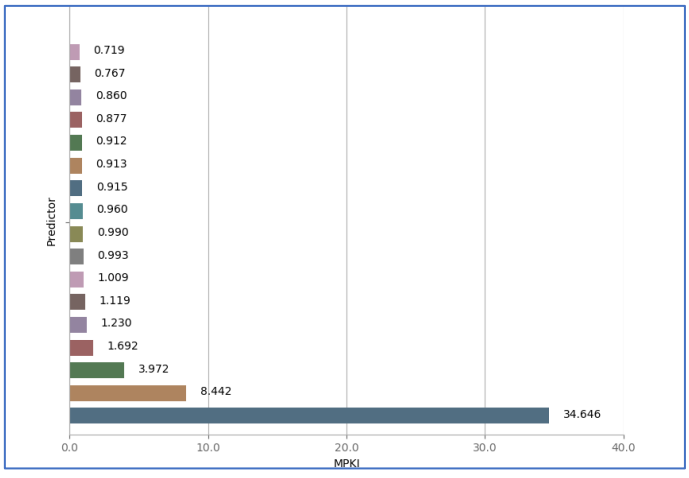
403.gcc



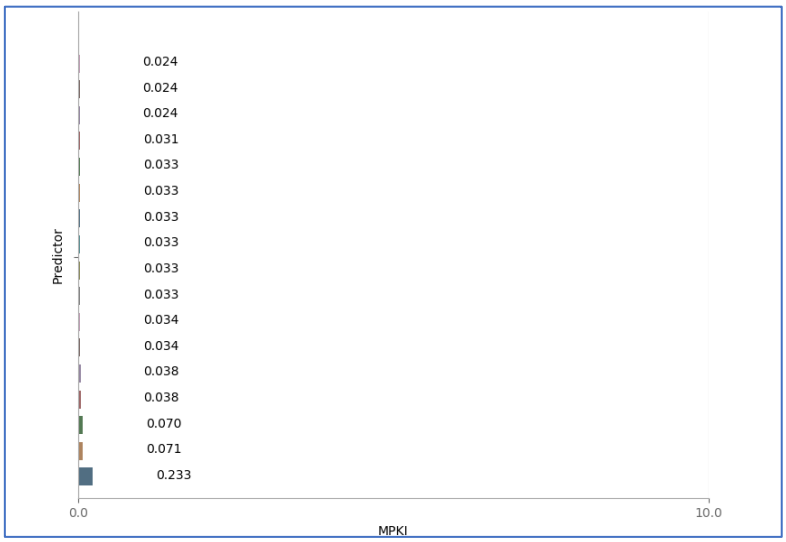
429.mcf



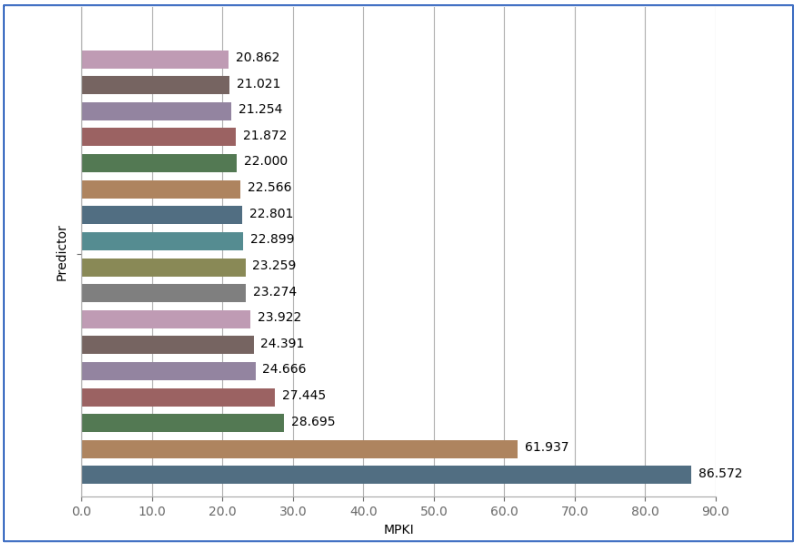
434.zeusmp



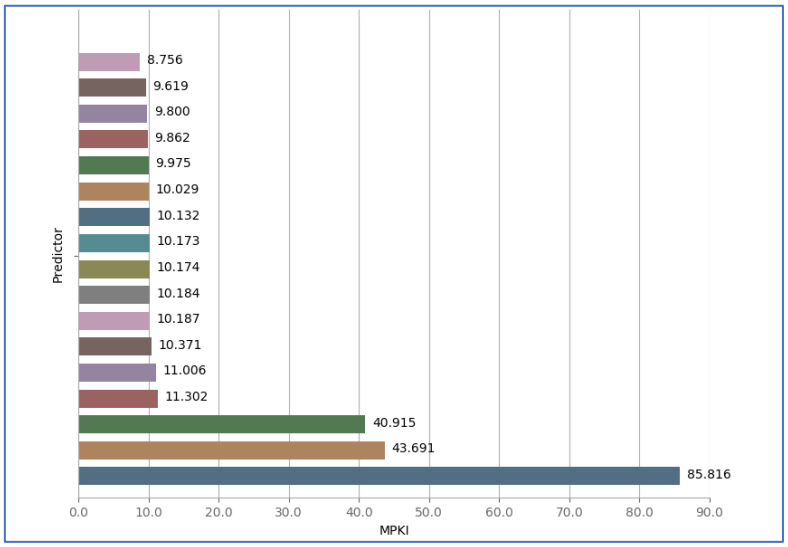
436.cactusADM



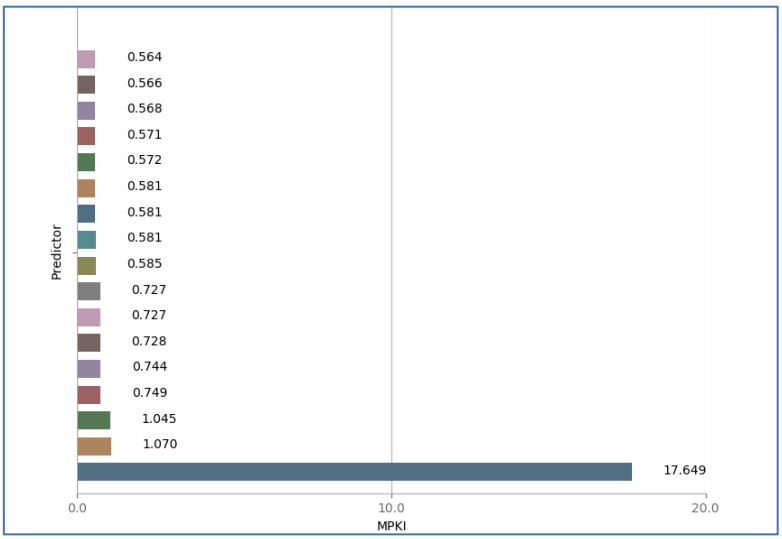
445.gobmk



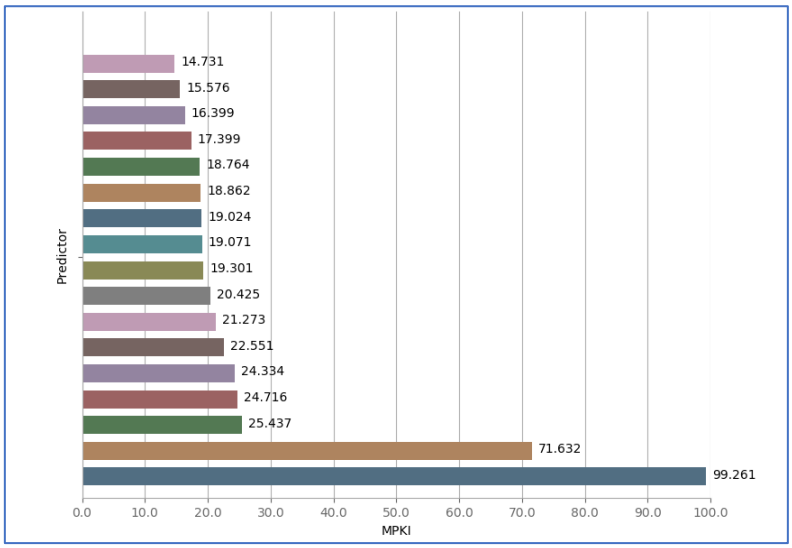
450.soplex



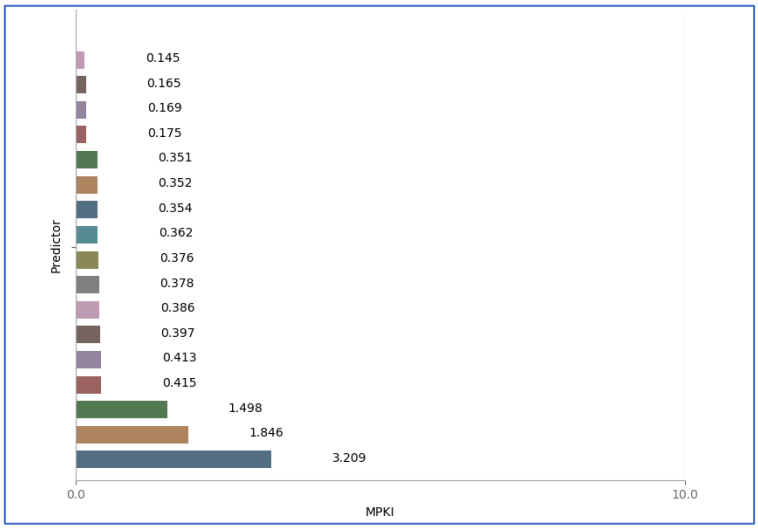
456.hmmer



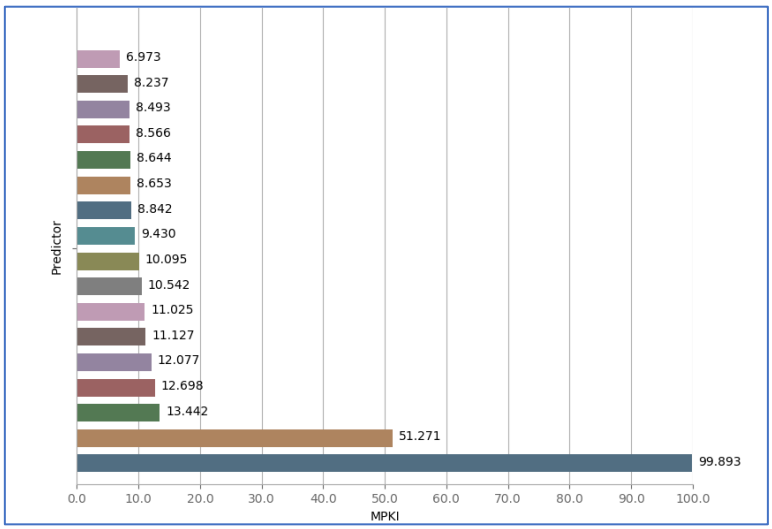
458.sjeng



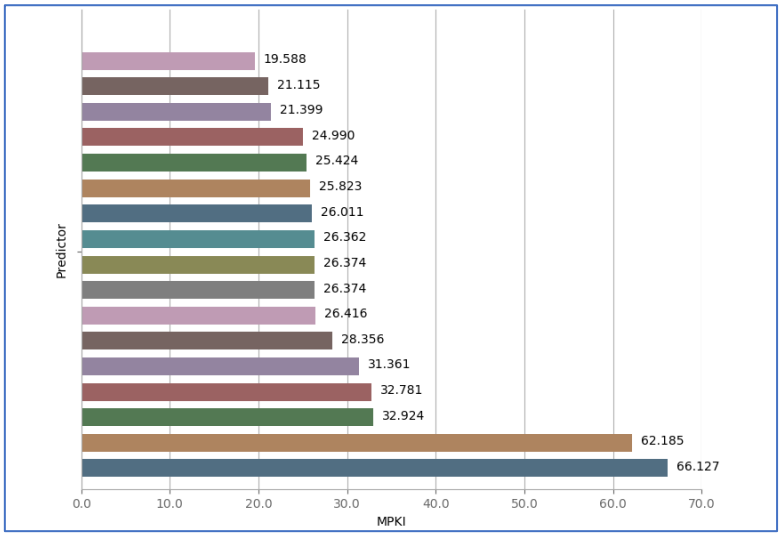
459.GemsFDTD



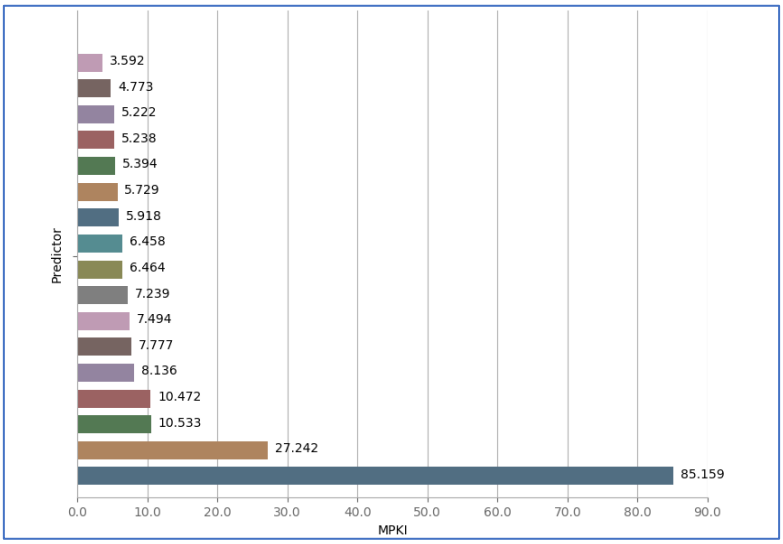
471.omnetpp



473.astar



483.xalancbmk



Παρατηρήσεις & Συμπεράσματα:

Παρατηρείται ότι ο Static Always Taken Predictor έχει τη χειρότερη επίδοση σε όλα τα μετροπρογράμματα. Επόμενος χειρότερος predictor είναι ο BTFNT, εξίσου αναμενόμενο αφού πρόκειται για static predictor, με ελαφρά ωστόσο καλύτερη επίδοση από τον Always Taken. Ακολουθεί ο Pentium Predictor. Όσον αφορά την επιλογή ενός συγκεκριμένου predictor, θα πρέπει να ληφθεί υπόψιν η χρήση που θα κάνει το σύστημα. Αν το προφίλ της χρήσης aligns με ένα ή περισσότερα benchmarks τότε θα επιλεχθεί predictor που τα πηγαίνει καλύτερα σε αυτήν την χρήση. Όμως στη μελέτη αυτή δε δίνεται έμφαση σε ένα συγκεκριμένο benchmark, έτσι το συμπέρασμα προκύπτει με βάση τη συνολική επίδοση των 12 προηγούμενων benchmarks κατά μέσο όρο. Συμπεραίνεται, πως ο Alpha Predictor έχει κατά μέσο όρο την καλύτερη επίδοση (χαμηλό MPKI) συνολικά, αλλά και την καλύτερη επίδοση σε κάθε ένα από τα benchmarks. Εξίσου καλά αποδίδουν και οι GlobalHistory-PHT(16K, 2bit)-BHR(10bit) LocalHistory-PHT(8K, 2bit)-BHT(2K, 8bit).

Υπάρχουν δύο ειδών predictors, οι static και οι dynamic. Οι πρώτοι λειτουργούν ανεξάρτητα από την ροή του προγράμματος, ενώ οι δεύτεροι μπορούν να αλλάξουν τον τρόπο πρόβλεψης κατά την εκτέλεση ενός προγράμματος. Οι predictors αυτοί αναφέρονται παρακάτω :

Static predictors: 1. Static AlwaysTaken & 2. BTFNT

Οι στατικοί προβλεπτές συνήθως δεν έχουν καλή επίδοση καθώς δεν μπορούν να προσαρμοστούν στο εκάστοτε πρόγραμμα. Ωστόσο είναι αρκετά εύκολοι στην υλοποίηση και το κόστος τους ελάχιστο.

Dynamic predictors: 1. N-bit predictor, 2. Global History Predictor & 3. Local History Predictor.

Είναι εμφανές πως οι παραπάνω τεχνικές έχουν βασικές αδυναμίες για συγκεκριμένα branch, γεγονός που μπορεί να ρίξει πολύ την απόδοση του συστήματος. Είναι θεμιτό κάτι καλύτερο από όλα τα παραπάνω. Για τον σκοπό αυτό, υπάρχουν οι λεγόμενοι tournament predictors. Πρακτικά αποτελείται από δύο κυρίως βαθμίδες. Έναν meta predictor στο πρώτο σκέλος, και έναν οποιαδήποτε συνδυασμό δύο από τους παραπάνω προβλεπτές. Επιλέχθηκε η προσομοίωση διάφορων συνδυασμών από predictors για τον tournament για μια πιο ολοκληρωμένη εικόνα.

Βάσει των παραπάνω διαγραμμάτων, είναι ξεκάθαρη η υπεροχή των dynamic predictors έναντι των static, καθώς οι dynamic εξασφαλίζουν ικανοποιητικά ποσοστά απόδοσης, που στην πλειοψηφία των benchmarks ξεπερνούν το 90%. Στο σύνολό τους, όλοι οι dynamic predictors παρουσιάζουν παρόμοια ποσοστά απόδοσης, με ορισμένους να εμφανίζουν μεγαλύτερη σταθερότητα ανά τα benchmarks. Συγκεκριμένα, φαίνεται να έχουν καλύτερες αποδόσεις οι LocalTwoLevel 8192-2048-8 και 8192-4096-4. Όσον αφορά τους tournament predictors, είναι εμφανές πως όλοι οι διαφορετικοί συνδυασμοί έχουν ικανοποιητικά αποτελέσματα, με τον καθένα από αυτούς να υπερτερεί ανάλογα την εφαρμογή. Σίγουρα 2 από τους κορυφαίους συνδυασμούς είναι οι δύο local-history-predictors(είτε 8192-4096-2 είτε 8192-2048-4) καθώς και ένας local με έναν global history predictor.