ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ & ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Άσκηση 3 - Προηγμένα Θέματα Αρχιτεκτονικής Υπολογιστών

Γρηγόριος Θανάσουλας

gregthanasoulas@gmail.com

A.M: 03114131

4 Ιουνίου 2020

Περιεχόμενα

1 Σκοπός

Η άσκηση αυτή αποσκοπεί στη μελέτη των χαρακτηριστικών των σύγχρονων superscalar, out-of-order επεξεργαστών και του τρόπου με τον οποίο αυτά επηρεάζουν την απόδοση του συστήματος, την κατανάλωση ενέργειας καθώς και το μέγεθος του chip του επεξεργαστή. Για την αξιολόγηση τους γίνεται χρήση του εργαλείου Sniper Multicore Simulator με τα παρακάτω μετροπρογράμματα (SPEC CPU2006 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

2 Πειραματική Αξιολόγηση

2.1 Ερώτημα i

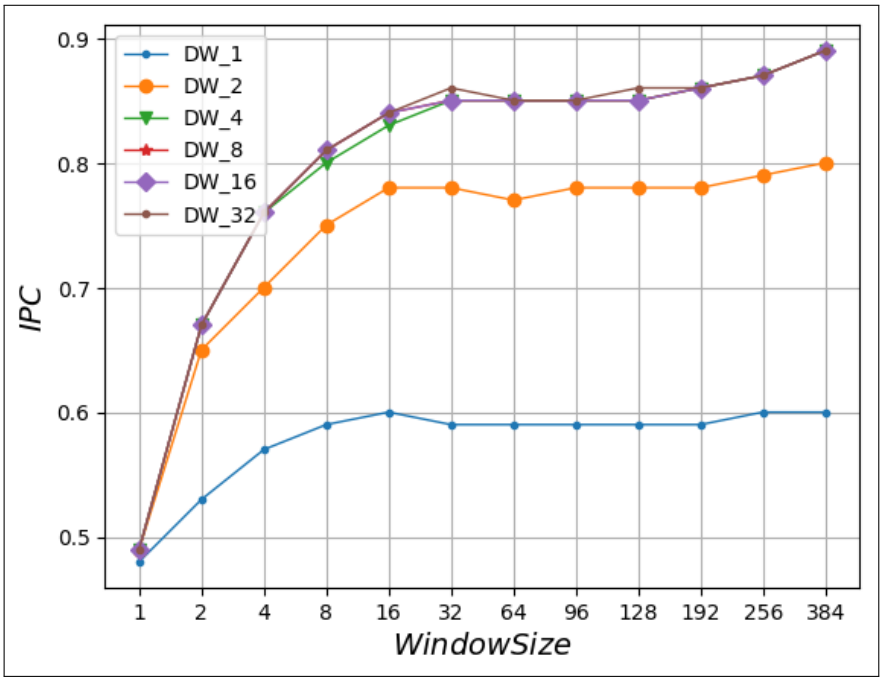
Ζητείται να εκτελέσουμε όλα τα benchmarks για κάθε διαφορετικό επεξεργαστή που προκύπτει από το συνδυασμό των παρακάτω τιμών για τις παραμέτρους dispatch_width και window_size:

dispatch_width	1	2	4	8	16	32						
window_size	1	2	4	8	16	32	64	96	128	192	256	384

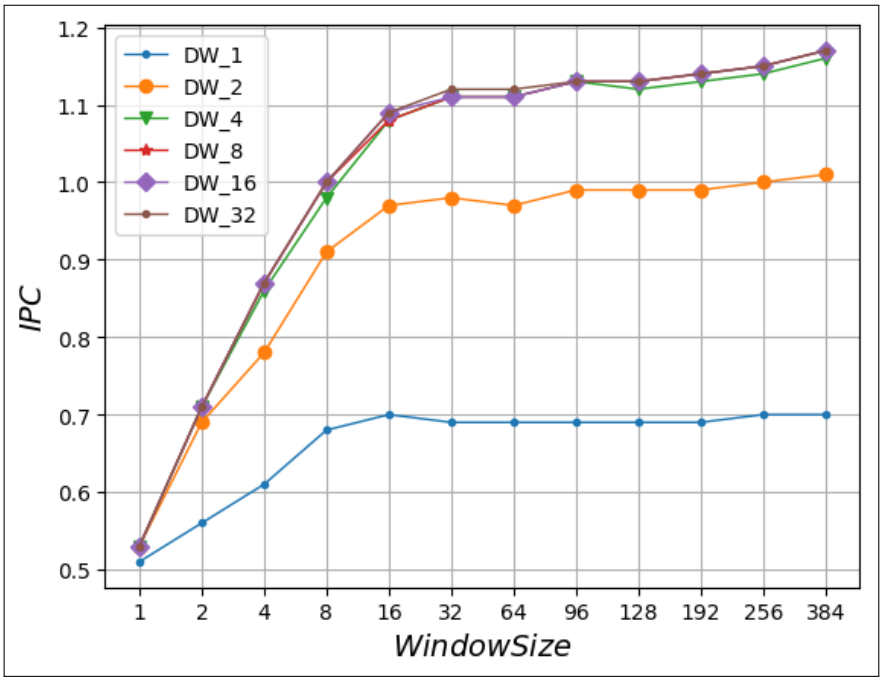
Από τους παραπάνω $6 \times 12 = 72$ δυνατούς συνδυασμούς νόημα έχουν μόνο εκείνοι για τους οποίους ισχύει $window\_size \geq dispatch\_width$. Αυτό μπορεί να γίνει αντιληπτό θεωρητικά αν μελετήσουμε τον τρόπο με τον οποίο γίνονται dispatch για issue οι εντολές και το ρόλο του Reorder Buffer (ROB) Όπως γνωρίζουμε για να γίνει μία εντολή issue πρέπει να υπάρχει διαθέσιμη θέση στον ROB. Επομένως, είναι χωρίς νόημα να κάνουμε dispatch παραπάνω εντολές από όσες μπορούν να χωρέσουν στον Reorder Buffer, γιατί απλά αυτές θα περιμένουν μέχρι να υπάρξει ελεύθερη θέση στον ROB και άρα η επίδοση δε θα βελτιωθεί καθόλου. Με βάση αυτό αγνοούμε για οικονομία χρόνου στις προσομοιώσεις τους συνδυασμούς για τους οποίους $window\_size < dispatch\_width$ και εκτελούμε τους εναπομείναντες 57 διαφορετικούς συνδυασμούς.

Την παρατήρηση αυτή μπορούμε να επιβεβαιώσουμε και πειραματικά. Στα παρακάτω διαγράμματα απεικονίζεται η μετρική Instructions Per Cycle για τα μετροπρογράμματα gcc και sjeng για όλους τους δυνατούς συνδυασμούς. Παρατηρούμε πως για κάθεμία από τις καμπύλες των διαγραμμάτων (που αντιστοιχεί σε ορισμένο dispatch width), η απόδοση (IPC) για τις τιμές windows_size που είναι μικρότερες από το dispatch width είναι αρκετά χαμηλή.

403-gcc



458-sjeng



2.2 Ερώτημα ii

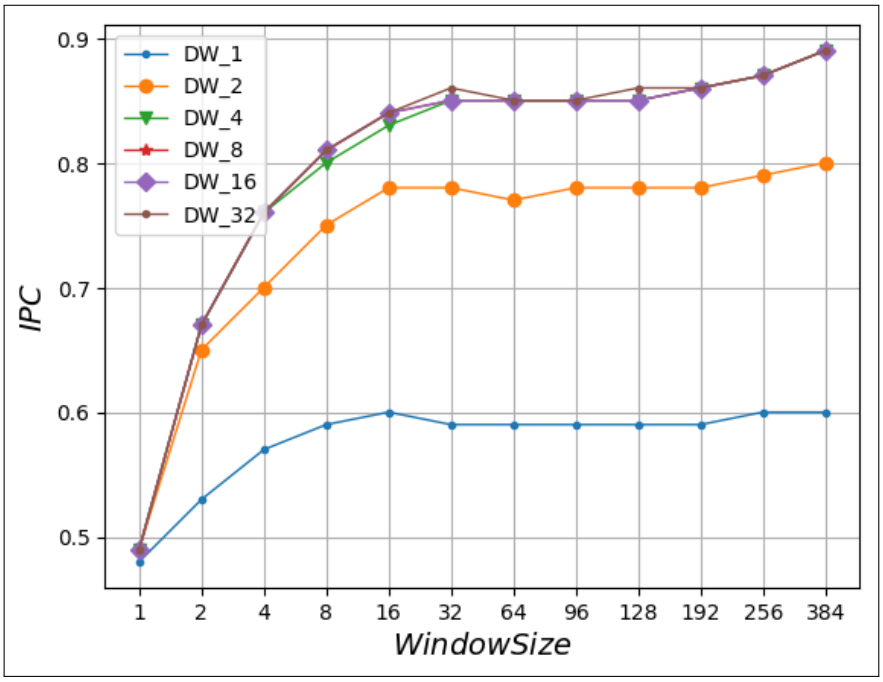
Για την μελέτη των χαρακτηριστικών εκτελέσαμε τα 12 παραπάνω benchmarks για τους συνδυασμούς χαρακτηριστικών επεξεργαστή που αναφέρθηκαν στο προηγούμενο ερώτημα. Ωστόσο, πρέπει σε αυτό το σημείο να σημειώσουμε ότι ορισμένα από τα benchmarks εμφάνιζαν errors κατά την εκτέλεση με αποτέλεσμα η προσομοίωση να εκτελείται για μικρό αριθμό εντολών, γεγονός που σημαίνει ότι δεν είναι αξιόπιστη η "εικόνα" της προσομοίωσης αυτής. Λαμβάνοντας υπ' όψιν βάσει της εκφώνησης της εργαστηριακής άσκησης ότι το κάθε pinball περιέχει περίπου 1 billion εντολές, και βάσει των αποτελεσμάτων στα αρχεία sim.out βρέθηκε για το κάθε benchmark ότι εκτελούνται τα παρακάτω ποσοστά εντολών:

```
✘ gregth@Dellis  ex3/scripts  master ●  ./get_instructions_count.sh
Outputs to be processed located in: /home/gregth/workspace/advcomparch/ex3/outputs
*Benchmark*  *Instructions Run*  *Percentage %*
astar        3932266             0.39
cactusADM    1000003023          100.00
gcc          166605946           16.66
GemsFDTD     890240332           89.02
gobmk        140748144           14.07
hmmer        12770222            1.27
mcf          1000003004          100.00
omnetpp      14099               0.00
sjeng        234366580           23.43
soplex       859741              0.08
xalancbmk    90394               0.00
zeusmp       1000002961          100.00
```

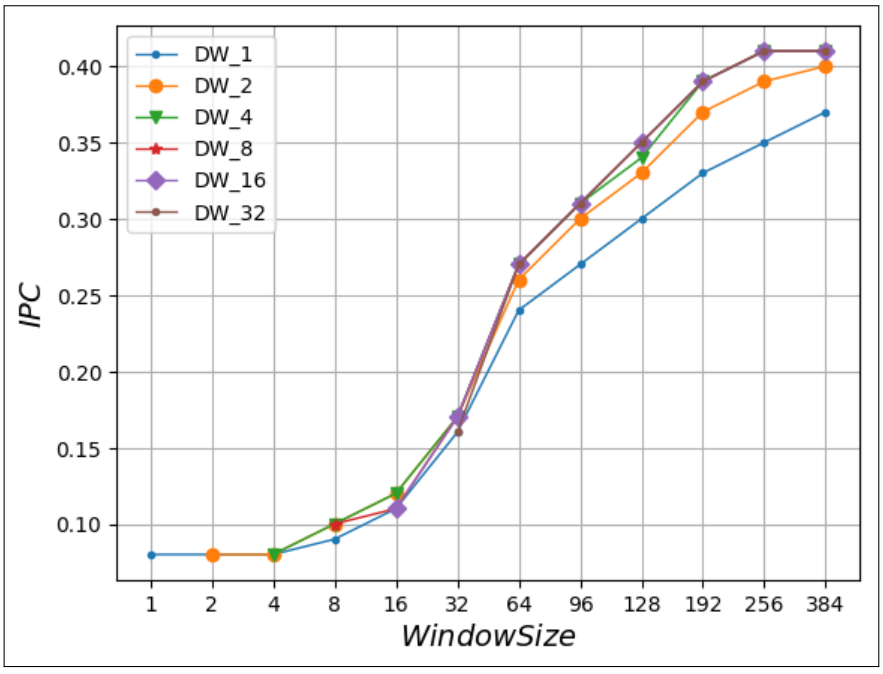
Από τα benchmarks αυτά, και βάσει τον διευκρινήσεων που δόθηκαν θα κρατήσουμε τις προσμοποιώσεις οπου έχει εκτελεστεί παραπάνω από το 10 % του pinball. Συνεπώς, δεν έχει νόημα να μελετήσουμε 5 από τα 12 benchmarks, και συγκεκριμένα τα astar, hmmer, soplex, xalancbmk, omnetpp.

Ακολουθούν τα διαγράμματα και ο σχολιασμός τους:

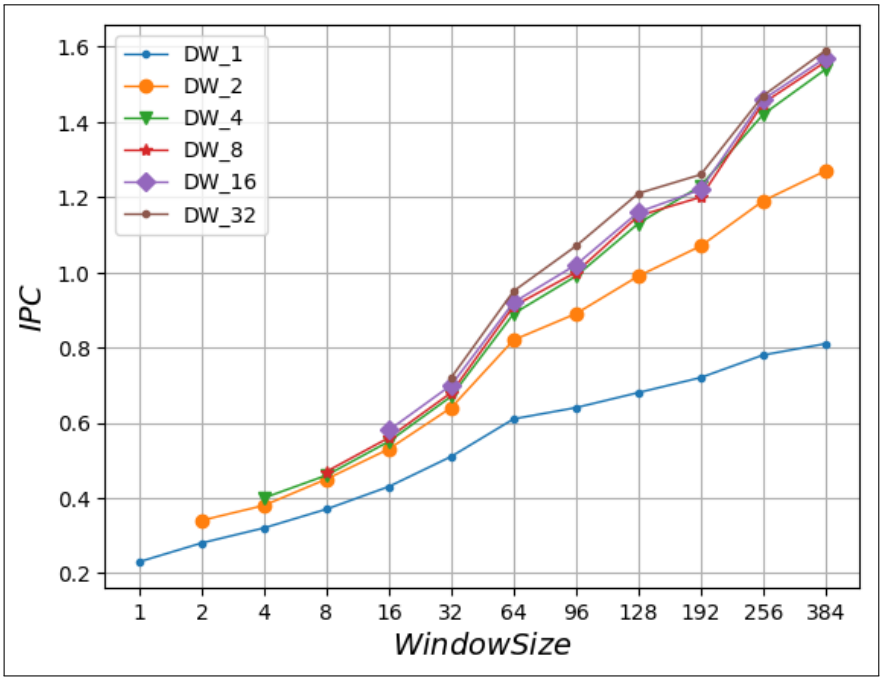
403-gcc



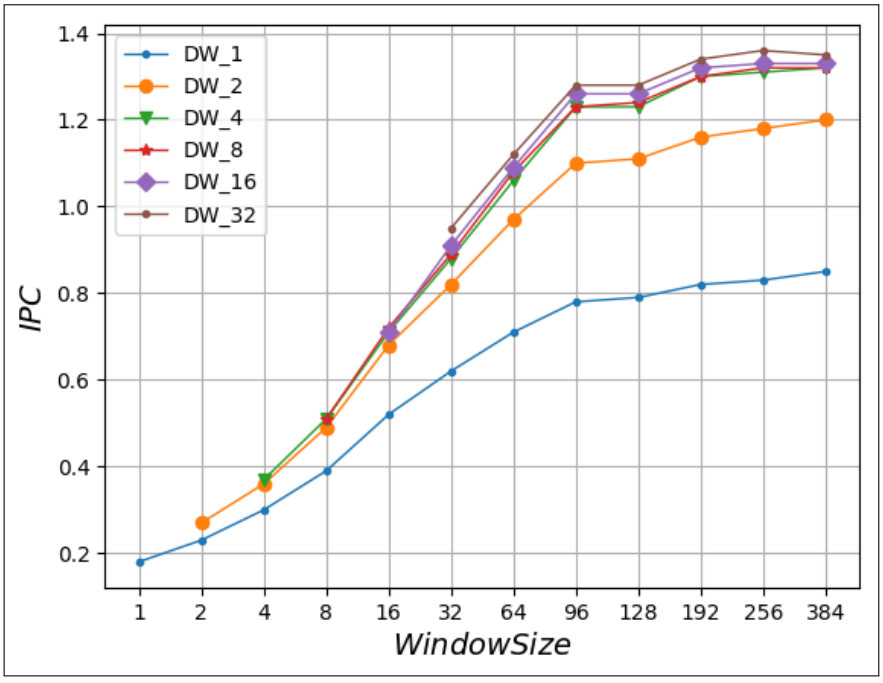
429-mcf



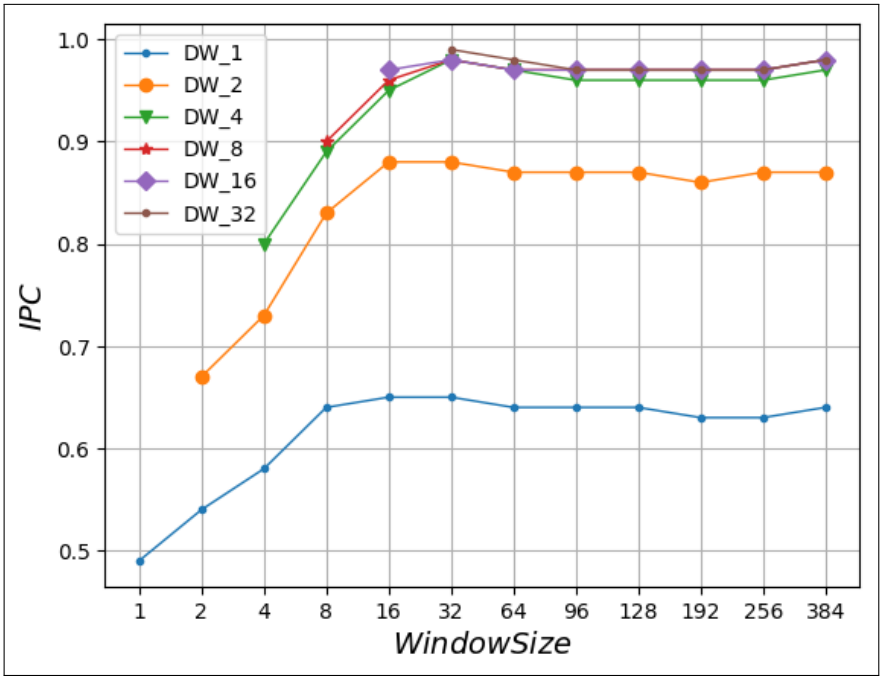
434-zeusmp



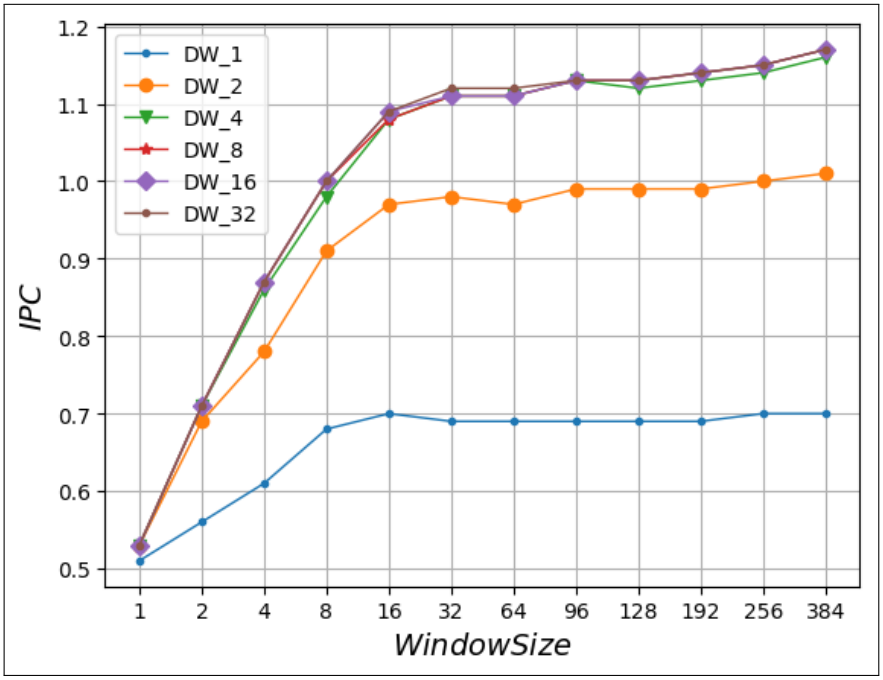
436-cactusADM



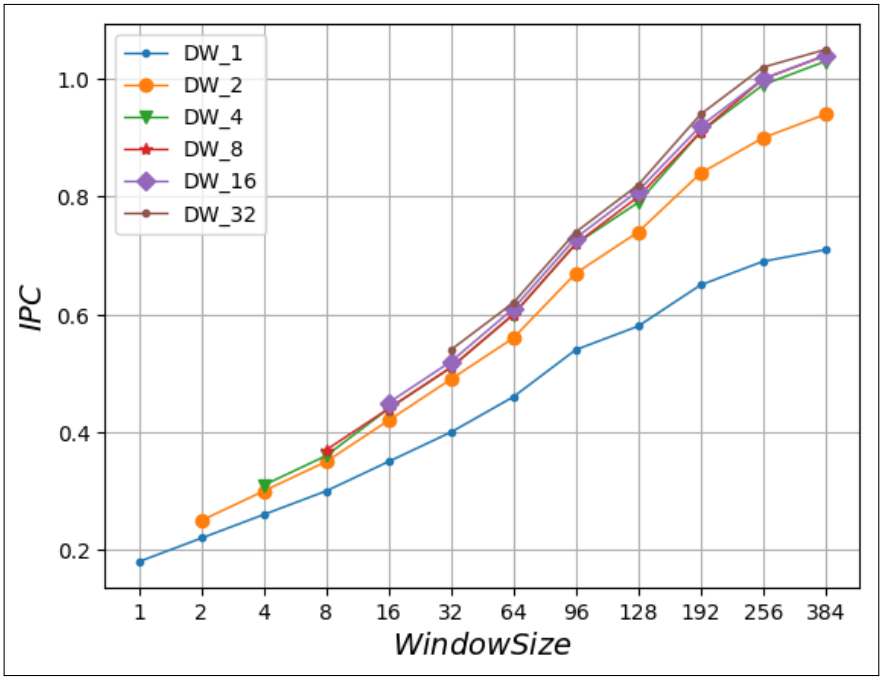
445-gobmk



458-sjeng



459-GemsFDTD



Συμπεράσματα - Σχόλια Από τις παραπάνω γραφικές του IPC συναρτήσει των dispatch width και window size μπορούμε εύκολα να συμπεράνουμε ότι η αύξηση του dispatch width από 1 σε 2 και από 2 σε 4 εντολές επιφέρει σημαντική βελτίωση της επίδοσης. Ωστόσο, περαιτέρω αύξηση σε του dispatch width σε 8, 16 ή και 32 εντολές δεν επιφέρει σημαντική αλλαγή στην επίδοση (οι γραφικές για τις τιμές αυτές είναι ως επί το πλείστον επικαλυπτόμενες) και άρα δεν έχει νόημα.

Αυτό μπορεί να ερμηνευθεί λόγω των περιορισμών του ILP (Instruction Level Parallelism) του κώδικα που εκτελείται. Δηλαδή, είναι δύσκολο να υπαρξουν και να γίνουν issue μεγάλες πλειάδες εντολών (κάθε πλειάδα πάνω από 4 εντολές) που να είναι ανεξάρτητες μεταξύ τους ώστε να μπορούν να γίνουν process παράλληλα και να επιτύχουμε με ικανοποιητικό ipc.

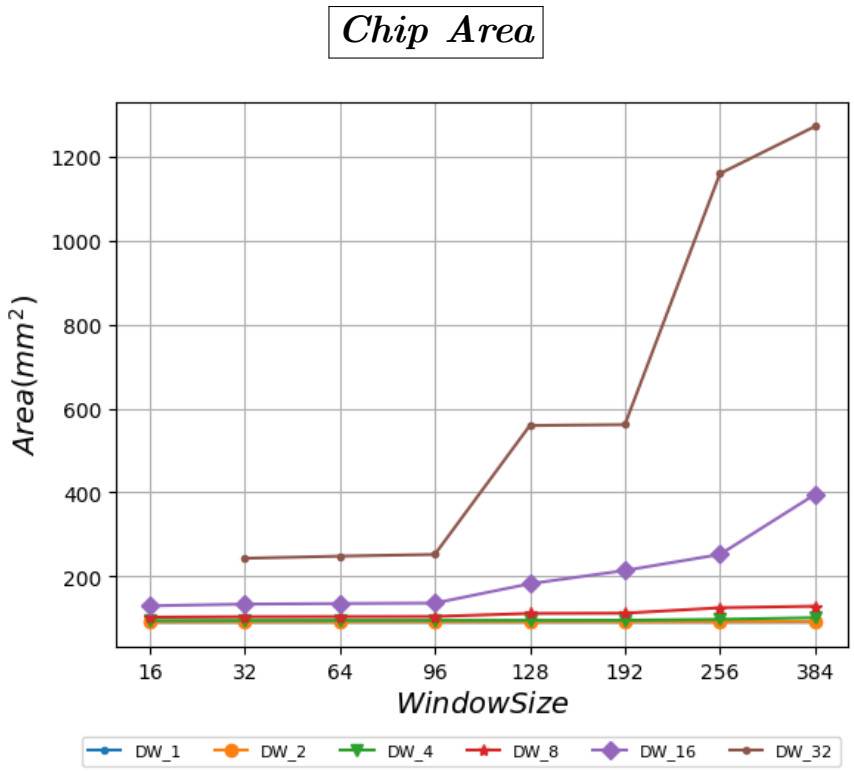
Όσον αφορά το window size, δηλαδή το μέγεθος του Reorder Buffer, βλέπουμε πως καθώς αυξάνεται, αυξάνεται συνήθως και το ipc. Αναλυτικότερα, υπάρχουν benchmarks (gemsFDTD, cactusADM, zeusmp, mcf) που αυτή η αύξηση συνεχίζεται διαρκώς καθώς αυξάνεται το μέγεθος του ROB, λαμβάνοντας μέγιστη τιμή για το μεγαλύτερο window size = 384 και άλλα benchmarks

(sjeng, gobmk, gcc) για τα οποία η αύξηση είναι σημαντική μέχρι ενός ορίου windows size = 32 περίπου, και από το σημείο αυτό και πέρα η αύξηση του ipc δεν είναι τόσο σημαντική. Αξίζει επίσης να σημειώσουμε ότι στα benchmarks zeusmp, cactusADM, sjeng, GemsFDTD το IPC καταφέρνει να ξεπεράσει τη μονάδα για dispatch width = 4 και αρκούντως μεγάλο window size.

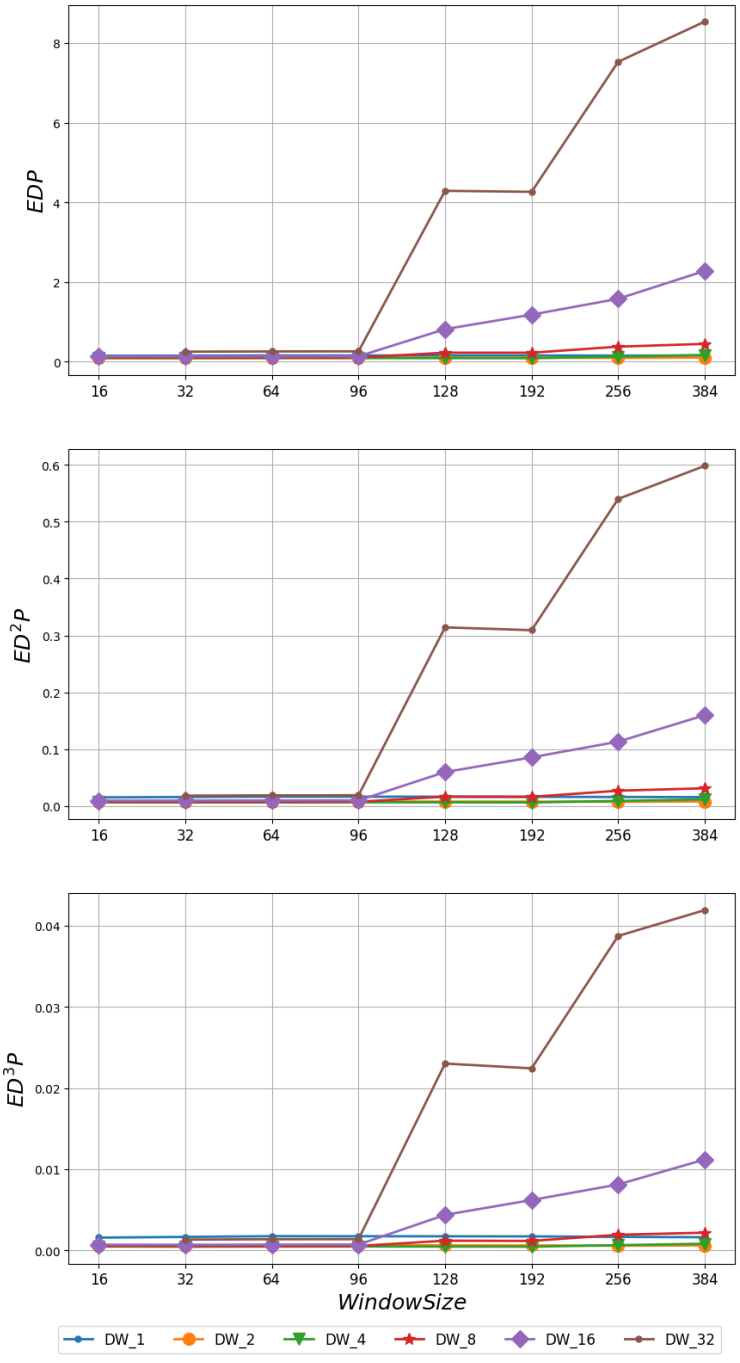
Βάσει των παραπάνω, για την κατασκευή θα επιλέγαμε πιθανόττα dispatch width = 4 και ένα αρκετά μεγάλο window size, το οποίο θα μας υπαγόρευαν άλλοι περιορισμοί, όπως η ενέργεια και το κόστος.

2.3 Ερώτημα iii

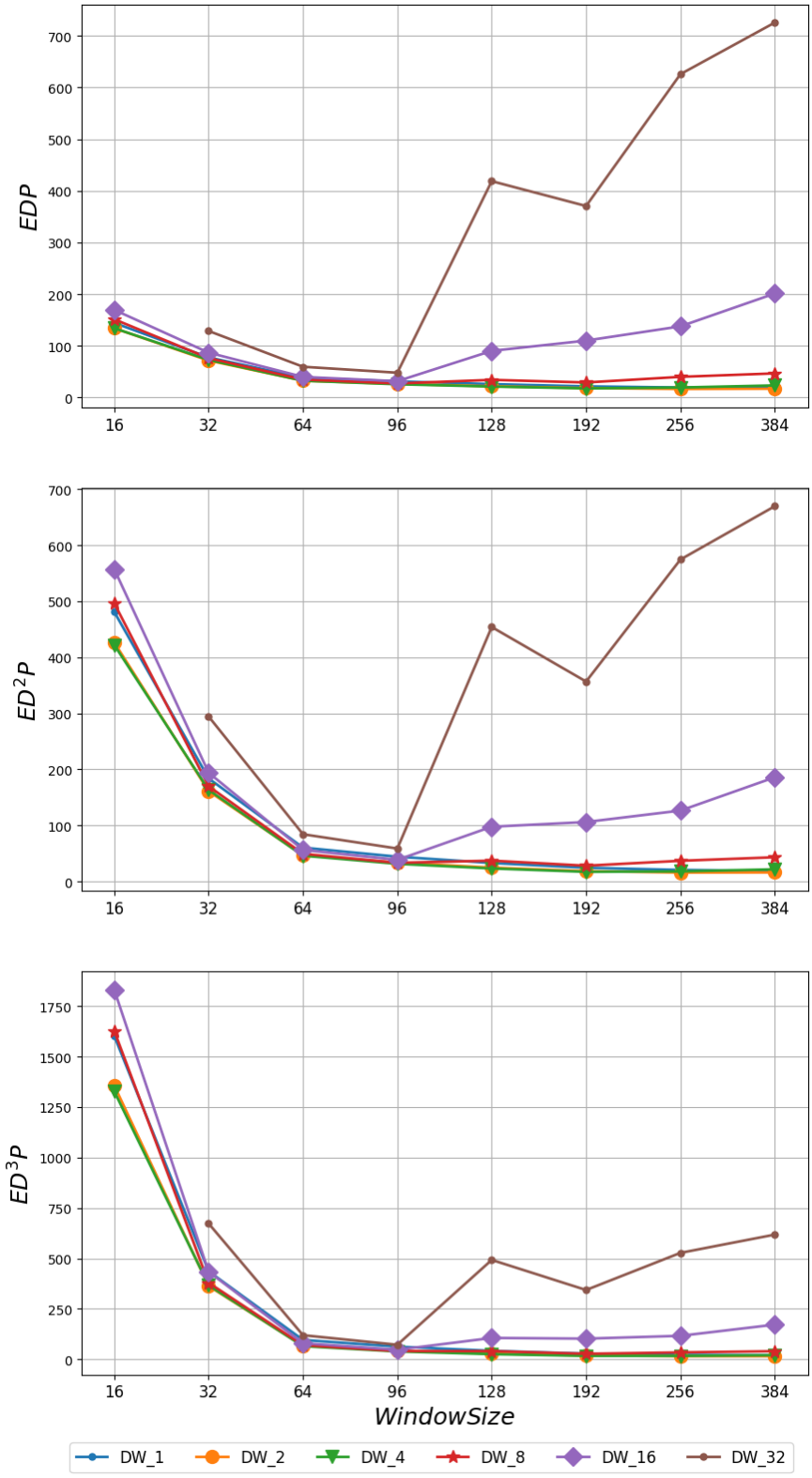
Ακολουθούν διαγράμματα για το μέγεθος του επεξεργαστή και κατανάλωση ενέργειας.



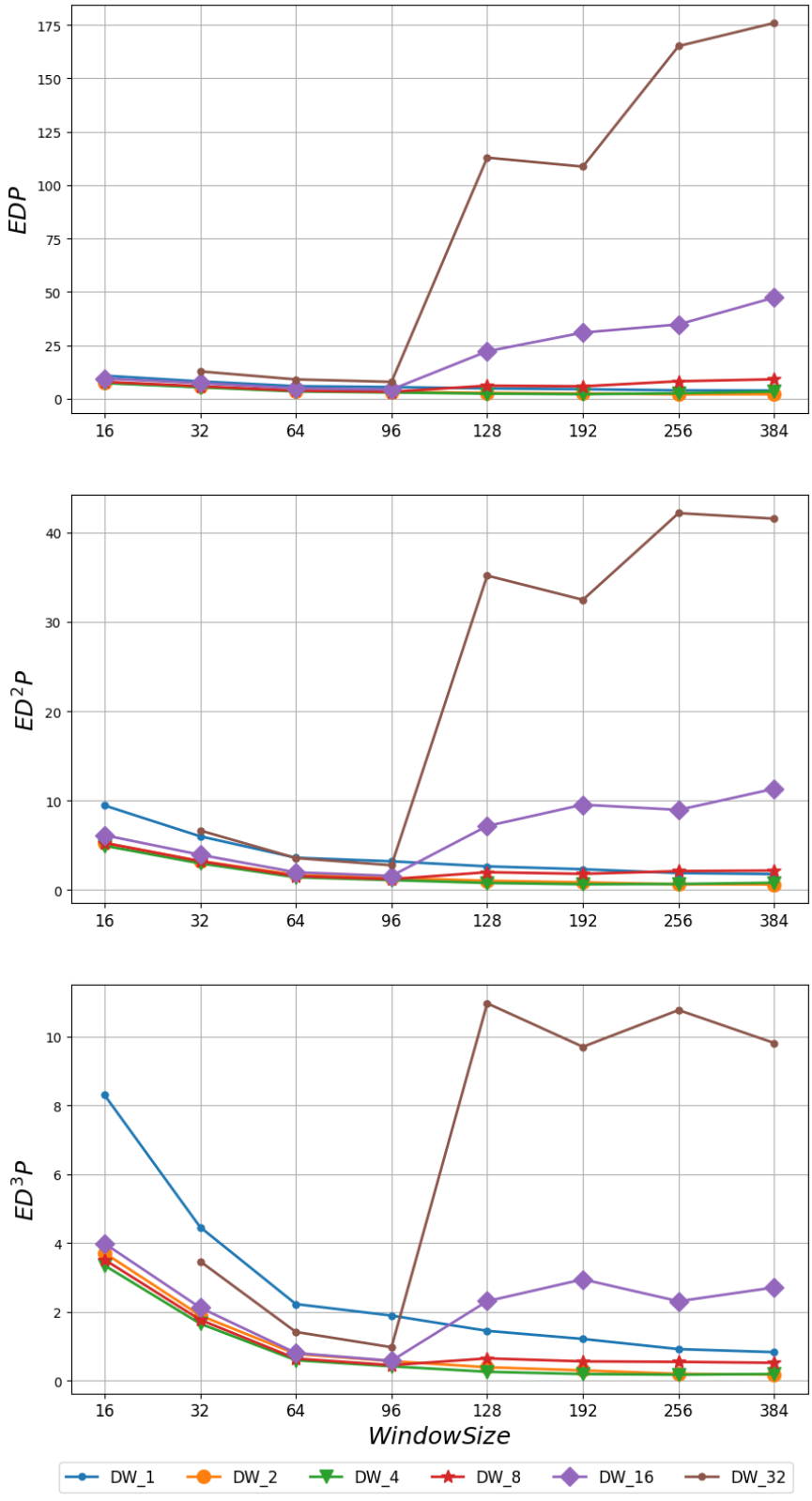
403-gcc


EDP
ED²P
ED³P
WindowSize
16
32
64
96
128
192
256
384
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

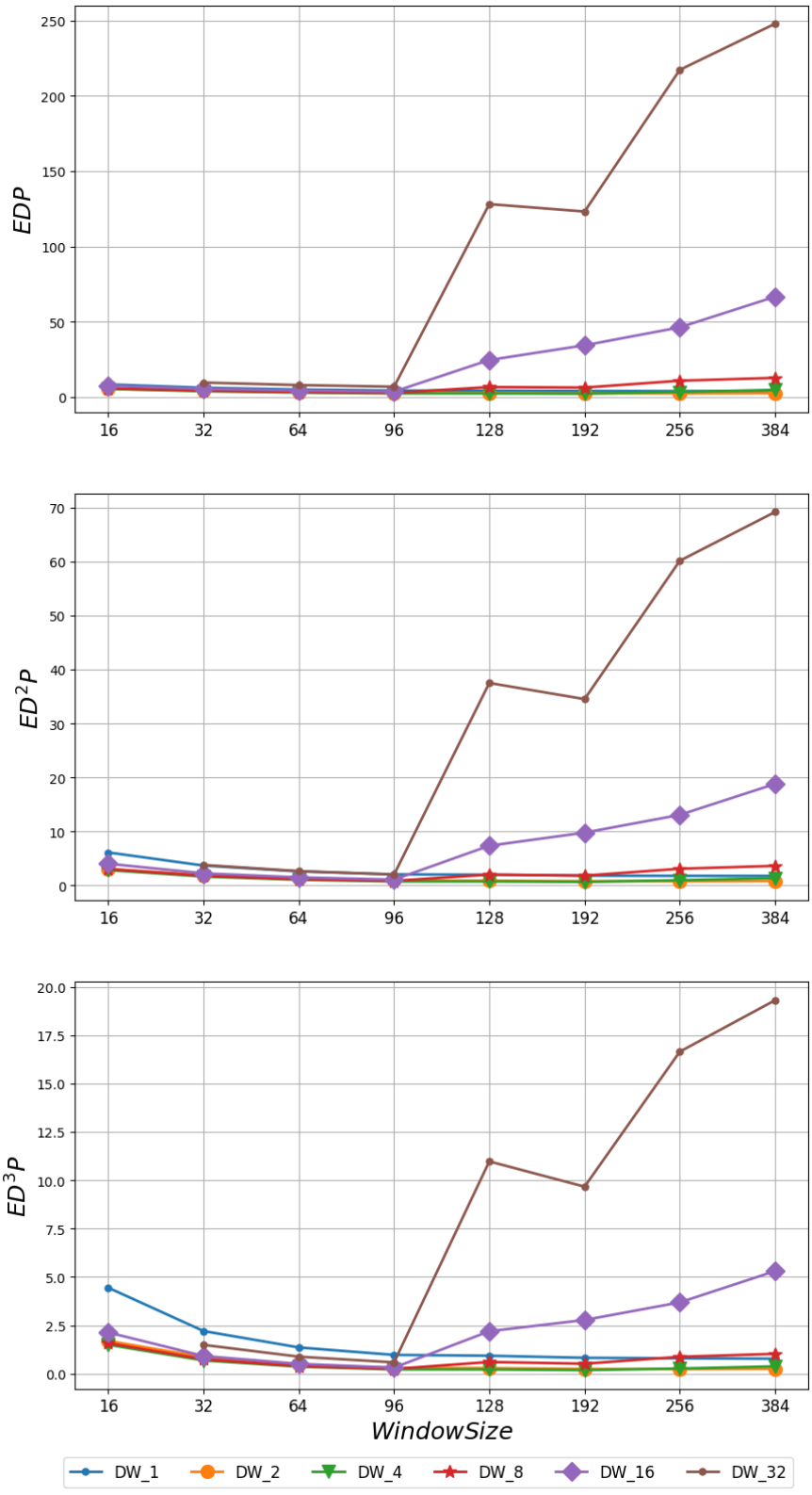
429-mcf



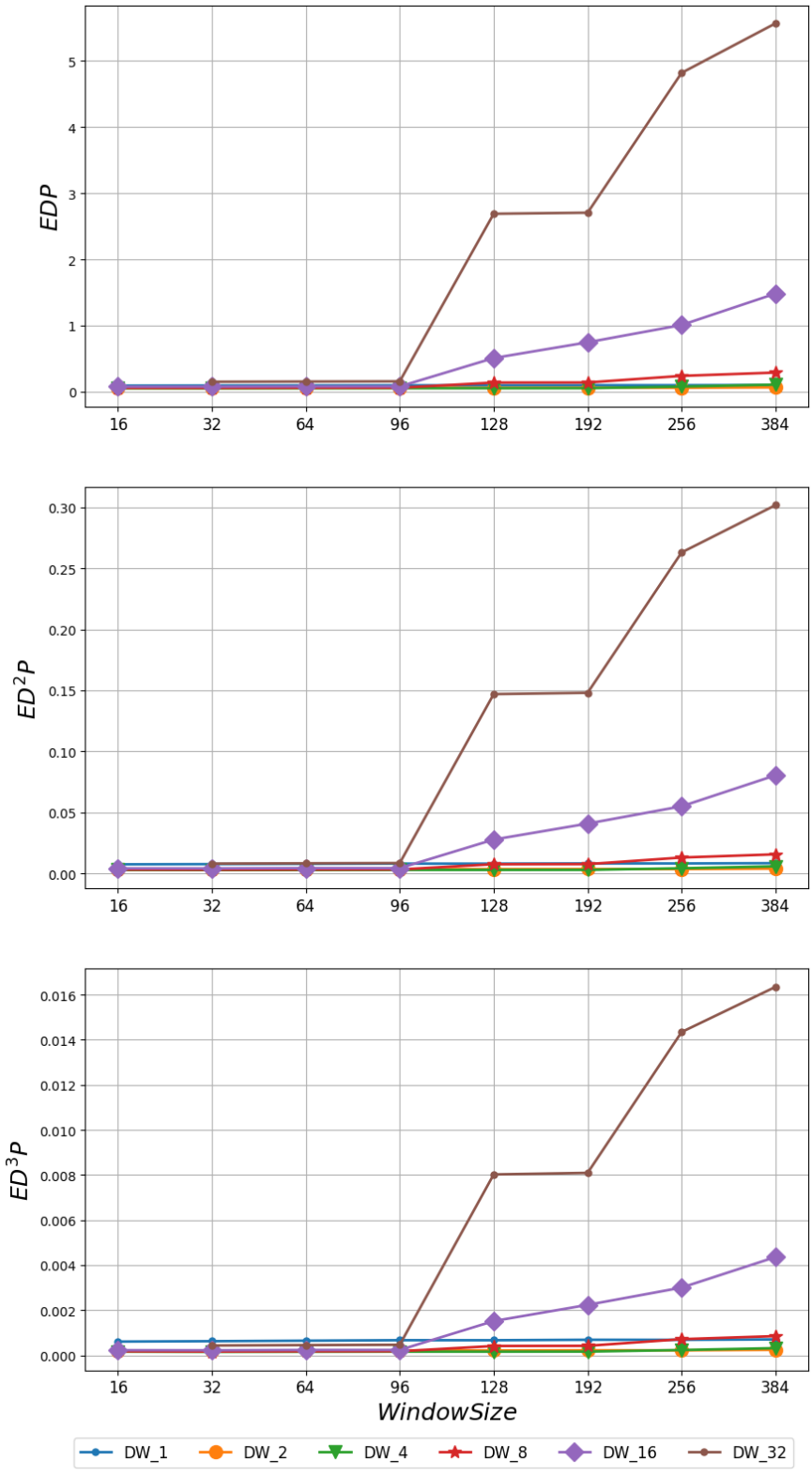
434-zeusmp


EDP
ED²P
ED³P
WindowSize
16
32
64
96
128
192
256
384
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

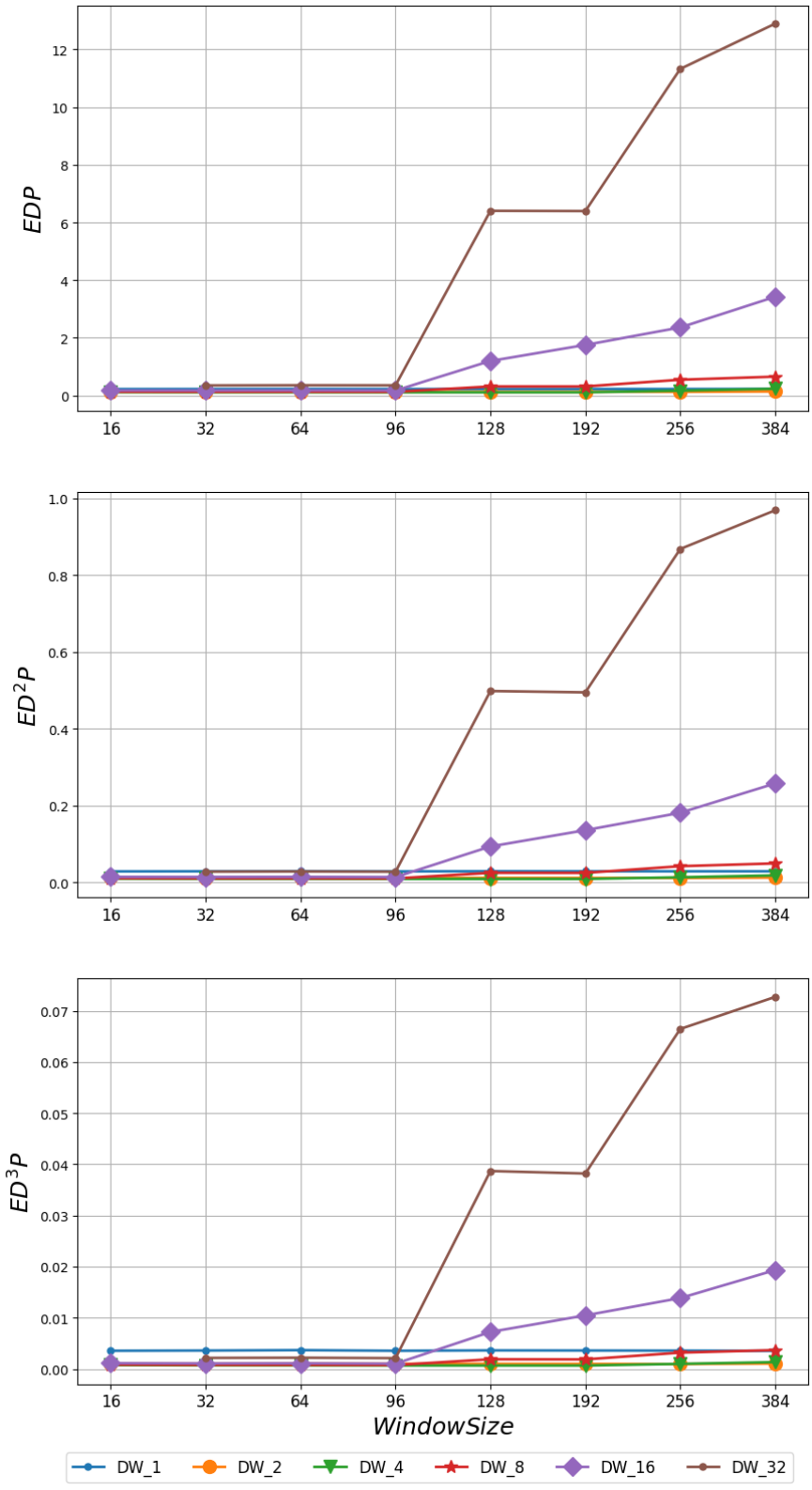
436-cactusADM



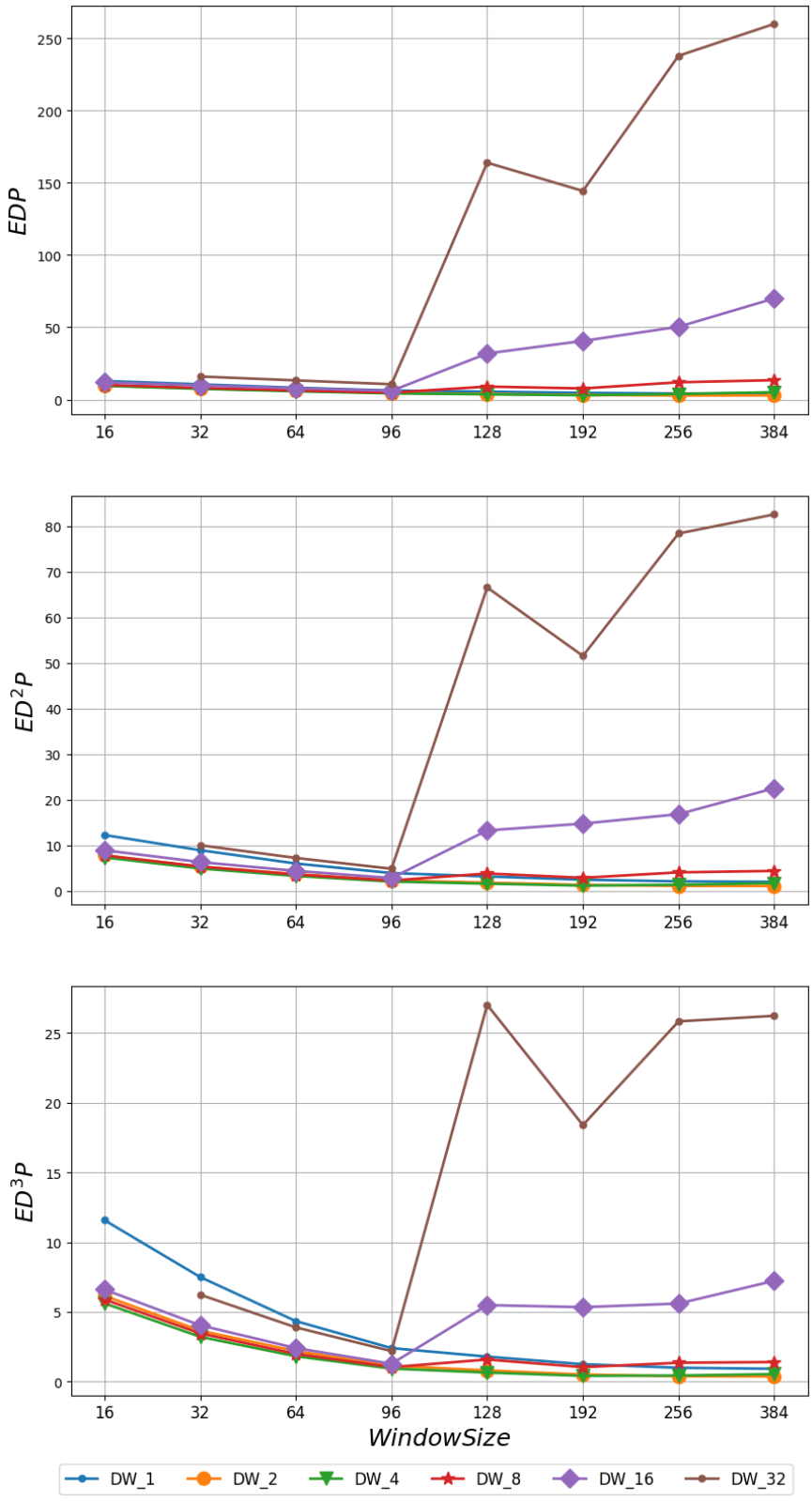
445-gobmk


EDP
ED²P
ED³P
WindowSize
16
32
64
96
128
192
256
384
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

458-sjeng


EDP
ED²P
ED³P
WindowSize
16
32
64
96
128
192
256
384
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

459-GemsFDTD



Συμπεράσματα - Σχόλια Αναφορικά με το μέγεθος του chip, παρατηρούμε πως δεν υπάρχει σημαντική διαφοροποίηση για επεξεργαστές με dispatch width 1, 2, 4 ή 8 εντολών. Μάλιστα, για τις τιμές αυτές του dispatch width το μέγεθος είναι περί τα 100 με 150 mm^2 και δεν μεταβάλλεται σημαντικά καθώς το window size αυξάνει. Ωστόσο, για dispatch width 16 και 32 το μέγεθος αυξάνει δραματικά mm^2 και επηρεάζεται από το window size. Μάλιστα, για dispatch width 32 και window size 384 το chip αποκτά το αρκετά μεγάλο μέγεθος $1300mm^2$.

Ως προς την ενέργεια που καταναλώνεται, παρατηρούμε ότι σε όλα τα benchmarks οι επιμέρους γραφικές για dispatch width 1, 2, 4 και 8 είναι παραπλήσιες, και άρα και η ενέργεια που καταναλώνεται είναι περίπου ίδια για ίδιο window size και dispatch width 1, 2, 4 ή 8. Παρατηρούμε επίσης πως σε ορισμένα benchmarks (mcf, zeus, cactusADM, GemsFDTD) για τις παραπάνω τιμές dispatch width και μικρές τιμές window size = 16 ή 32 η ενέργεια είναι πιο μεγάλη σε σχέση με λίγο μεγαλύτερο window size. Η ελάχιστη ενέργεια στις περιπτώσεις αυτές δείχνει να είναι για window size 128 ή 256. Ωστόσο και για μεγαλύτερα window size δεν υπάρχει σοβαρή επίπτωση.

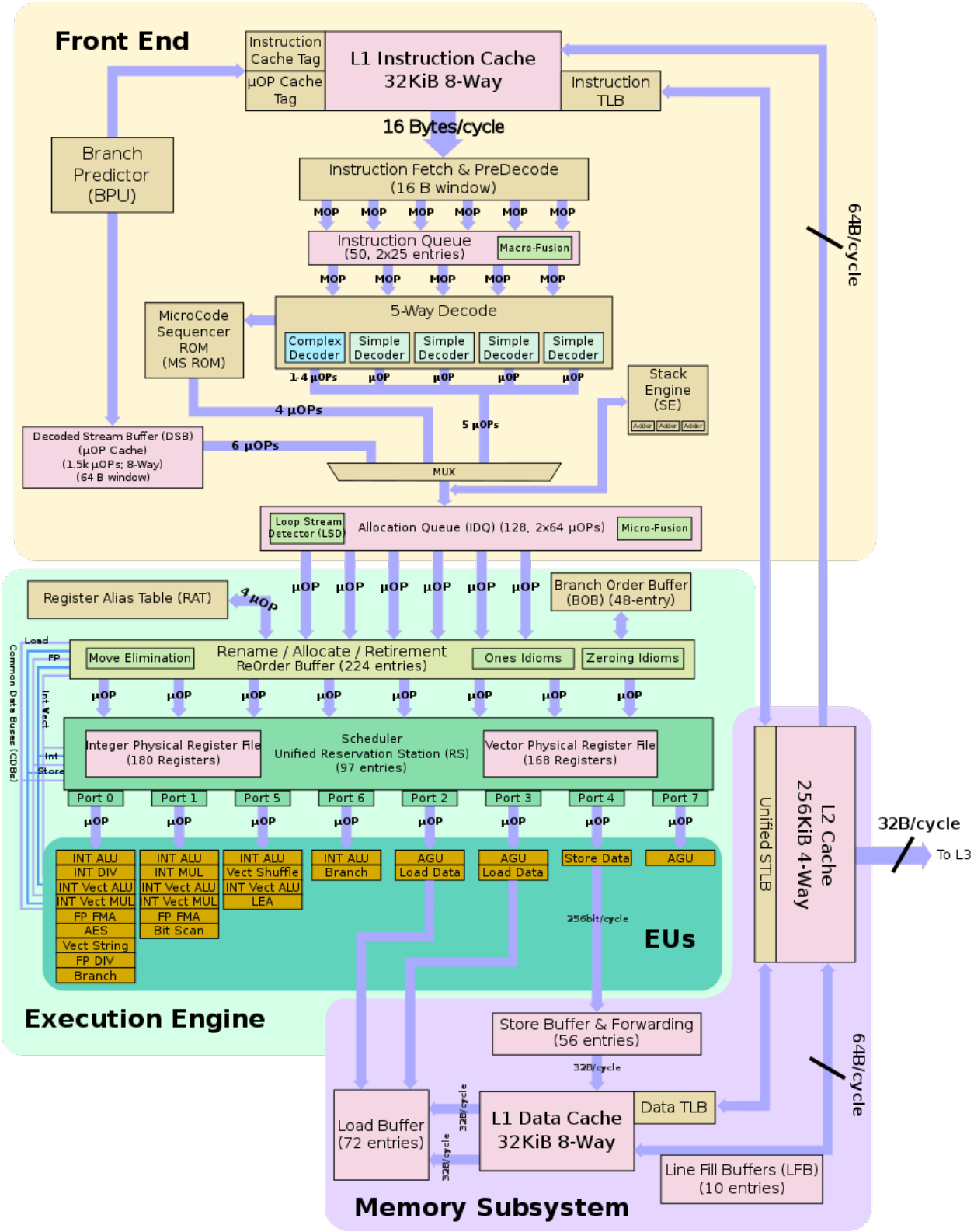
Η επιλογή dispatch width 16 και 32 αυξάνει δραστικά την ενέργεια που καταναλώνεται και μάλιστα στην περίπτωση αυτή αυξάνει σημαντικά καθώς αυξάνεται το window size.

Με βάση την ανάλυση αυτή αλλά και λεμβάνοντας υπόψιν την ανάλυση για την επίδοση στα προηγούμενα ερωτήματα, θα επέλεγα dispatch width = 4 και dispatch width (δεδομένου ότι η πράσινη καμπύλη είναι φθίνουσα ή σταθερή καθώς αυξάνει το window size) τουλάχιστον 256.

2.4 Ερώτημα iv

Για τον προσωπικό μου υπολογιστή / latpop, ο επεξεργαστής του Intel Core **i7-8550U** χρησιμοποιεί την αρχιτεκτονική Kaby Lake. (https://en.wikichip.org/wiki/intel/microarchitectures/kaby_lake). Ακολουθεί διάγραμμα της εν λόγω αρχιτεκτονικής:

Kaby Lake Microarchitecture



Όπως φαίνεται στο διάγραμμα, η αρχιτεκτονική Kaby Lake χρησιμοποιεί ReOrder Buffer με 224 entries (window size) και dispatcher width = 6 εντολές. Σύμφωνα με την ανάλυση που προηγήθηκε, η επιλογή των χαρακτηριστικών αυτών είναι απολύτως λογική και δικαιολογητέα, αφού επιτυγχάνει αρκετά καλή απόδοση λαμβάνοντας υπ' όψιν το περιορισμένο μέγεθος chip και την χαμηλή κατανάλωση ενέργειας, αφού πρόκειται για επεξεργαστής προορισμένος για χρήση σε laptop.

Συμπεράσματα - Σχόλια