3η εργαστηριακή άσκηση προηγμένα θέματα αρχιτεκτονικής υπολογιστών

03117103 ΑΘΑΝΑΣΙΟΣ ΔΕΛΗΣ

A. Definitions(Wikipedia sourse)

Out of order superscalar processor: A superscalar processor is a CPU that implements a form of parallelism called instruction-level parallelism within a single processor. In contrast to a scalar processor that can execute at most one single instruction per clock cycle, a superscalar processor can execute more than one instruction during a clock cycle by simultaneously dispatching multiple instructions to different execution units on the processor.

In computer engineering, out-of-order execution (or more formally dynamic execution) is a paradigm used in most high-performance central processing units to make use of instruction cycles that would otherwise be wasted. In this paradigm, a processor executes instructions in an order governed by the availability of input data and execution units,[1] rather than by their original order in a program.[2]

Issue/Dispatch width: issue width = the number of issue slots, 1 slot/instruction, consserns processors with the ability to execute Multiple instructions issued each cycle

Window size and ROB: An instruction window in computer architecture refers to the set of instructions which can execute out-of-order in a speculative processor.

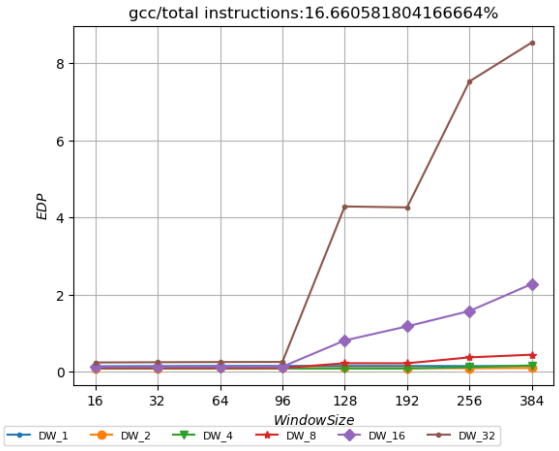
In particular, in a conventional design, the instruction window consists of all instructions which are in the re-order buffer (ROB).[1] In such a processor, any instruction within the instruction window can be executed when its operands are ready. Out-of-order processors derive their name because this may occur out-of-order (if operands to a younger instruction are ready before those of an older instruction).

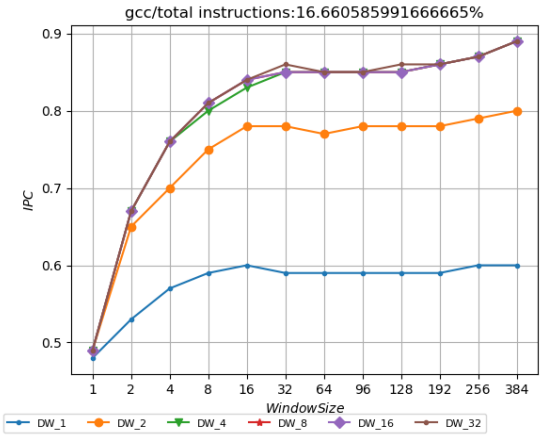
The instruction window has a finite size, and new instructions can enter the window (usually called dispatch or allocate) only when other instructions leave the window (usually called retire or commit). Instructions enter and leave the instruction window in program order, and an instruction can only leave the window when it is the oldest instruction in the window and it has been completed. Hence, the instruction window can be seen as a sliding window in which the instructions can become out-of-order. All execution within the window is speculative (i.e., side-effects are not applied outside the CPU) until it is committed in order to support asynchronous exception handling like interrupts.

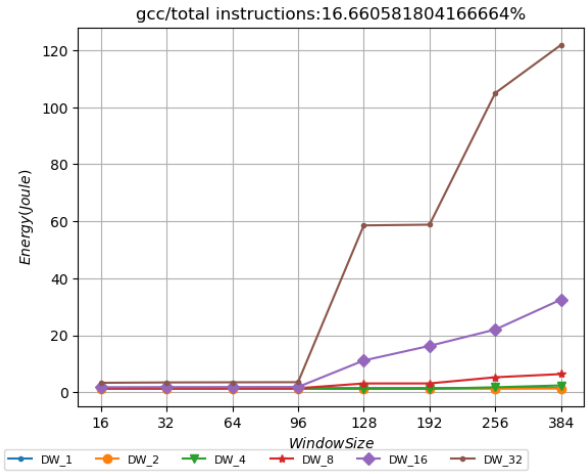
EDP: Metric runtime*enrgy

B.Plots

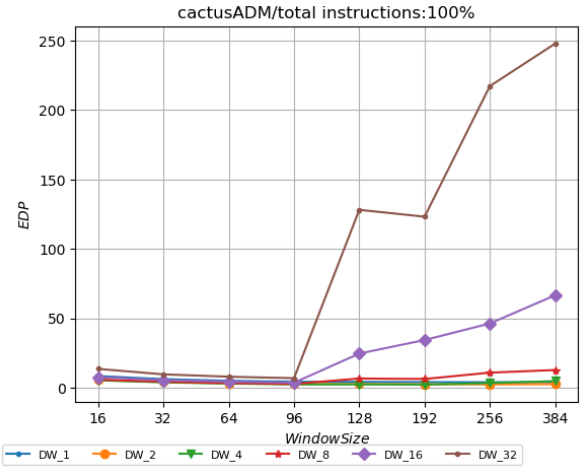
Gcc

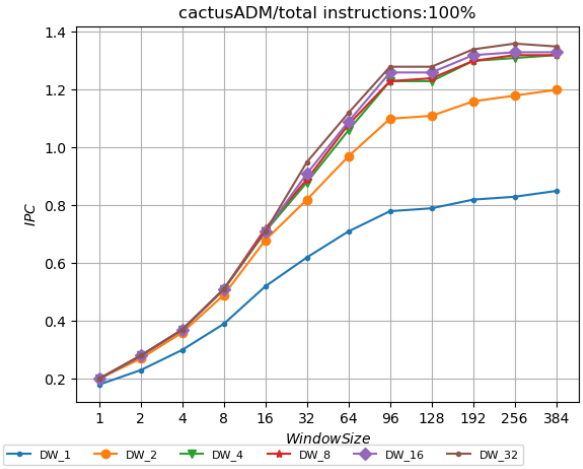

gcc/total instructions:16.660581804166664%
EDP
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

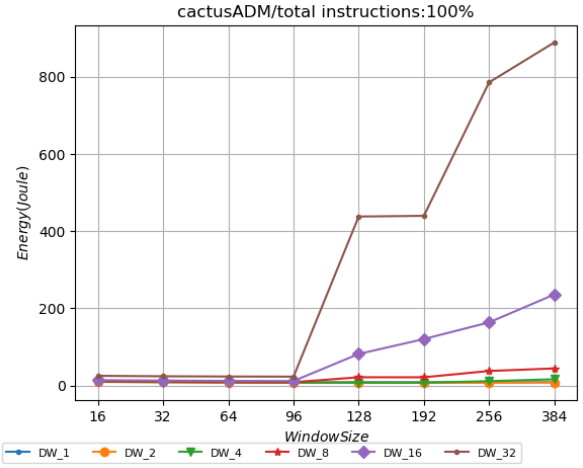

gcc/total instructions:16.660585991666665%
IPC
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32


gcc/total instructions:16.660581804166664%
Energy(Joule)
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

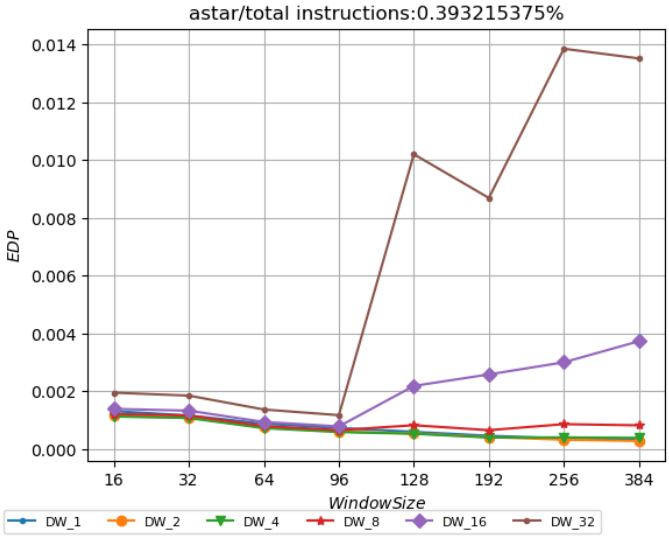
CactusADM

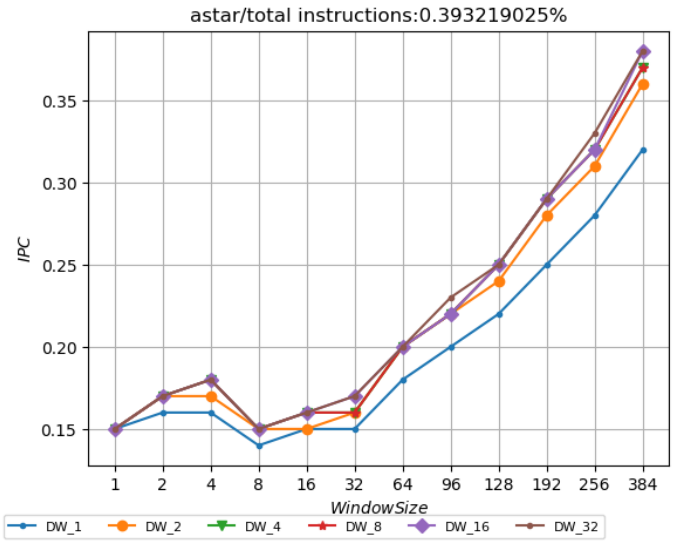

cactusADM/total instructions:100%
EDP
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

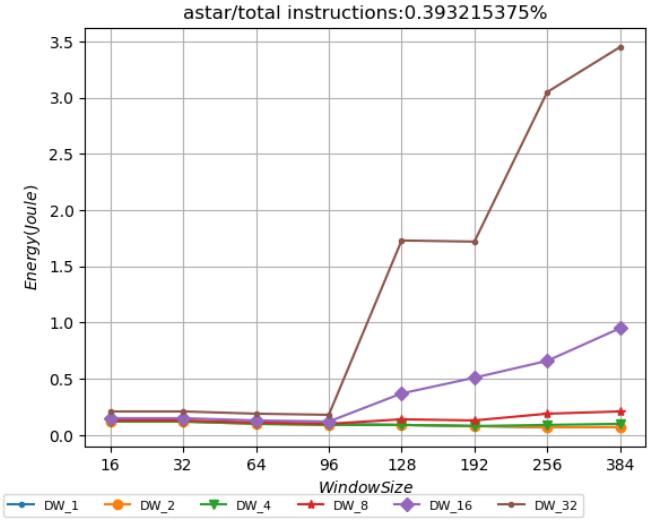

cactusADM/total instructions:100%
IPC
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32


cactusADM/total instructions:100%
Energy(Joule)
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

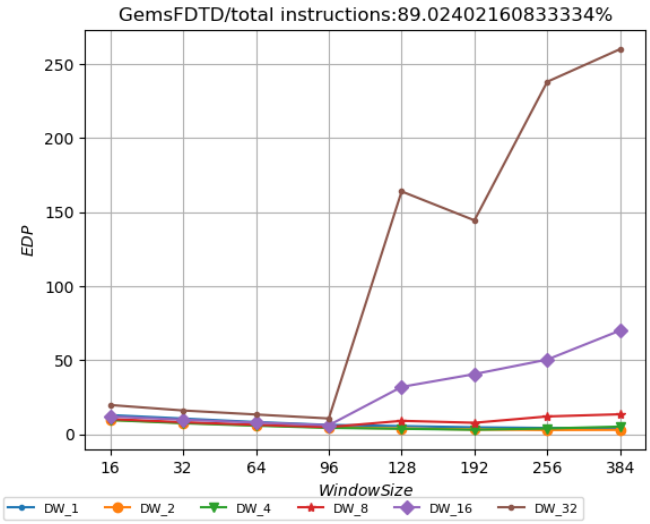
Astar

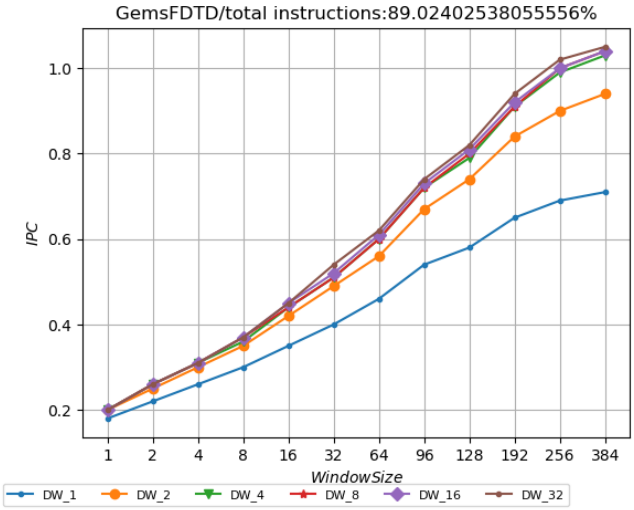

astar/total instructions:0.393215375%
EDP
0.014
0.012
0.010
0.008
0.006
0.004
0.002
0.000
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

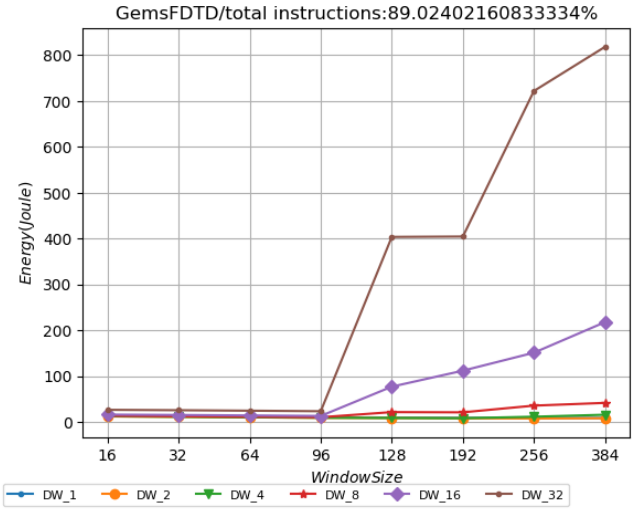

astar/total instructions:0.393219025%
IPC
0.35
0.30
0.25
0.20
0.15
1
2
4
8
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32


astar/total instructions:0.393215375%
Energy(Joule)
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

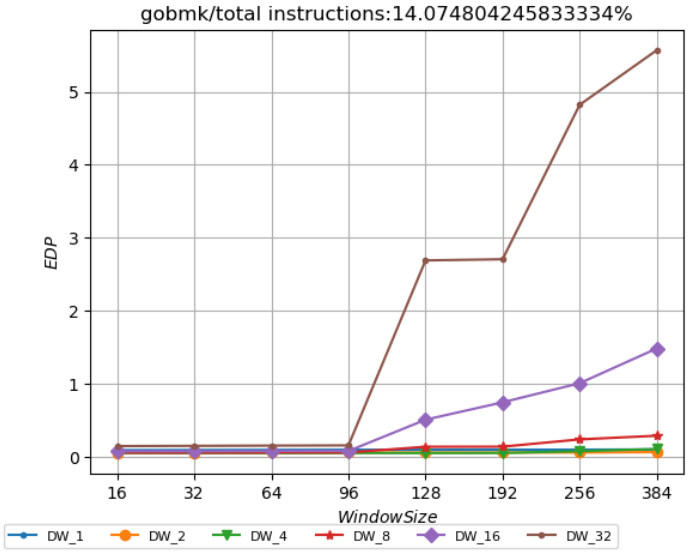
GemsFDTD

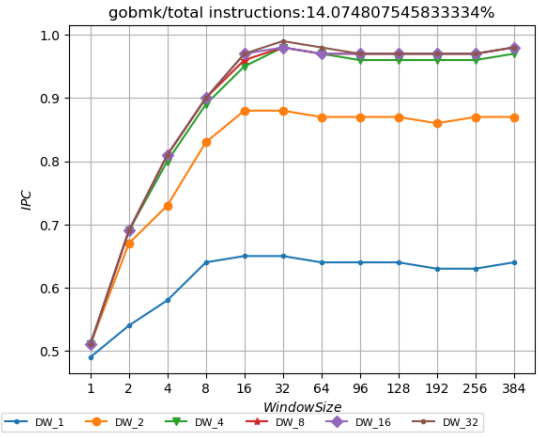

GemsFDTD/total instructions:89.02402160833334%
EDP
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

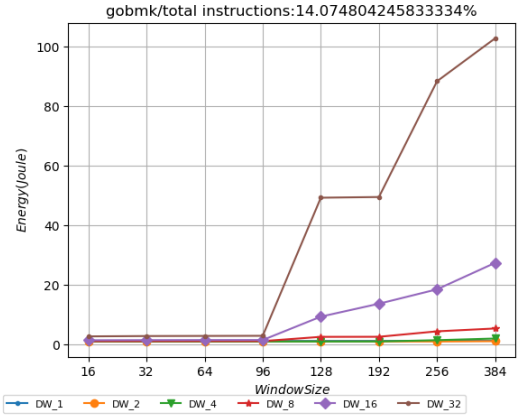

GemsFDTD/total instructions:89.02402538055556%
IPC
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32


GemsFDTD/total instructions:89.02402160833334%
Energy(Joule)
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

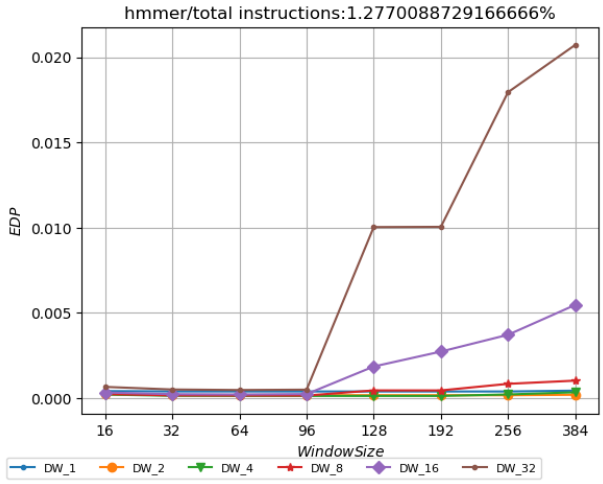
Gobmk

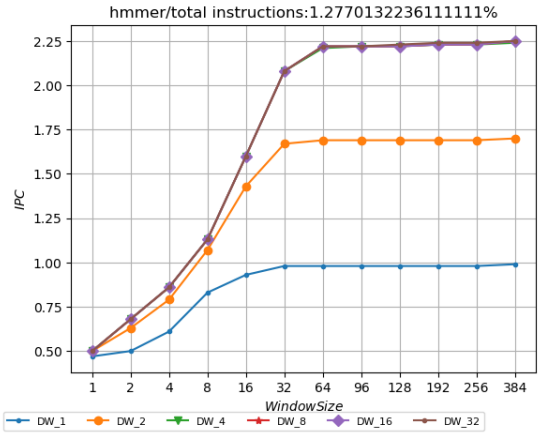

gobmk/total instructions:14.074804245833334%
EDP
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

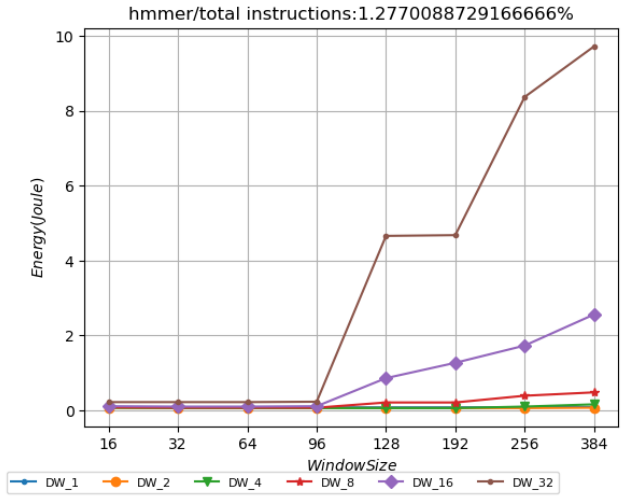

gobmk/total instructions:14.074807545833334%
IPC
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32


gobmk/total instructions:14.074804245833334%
Energy(Joule)
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

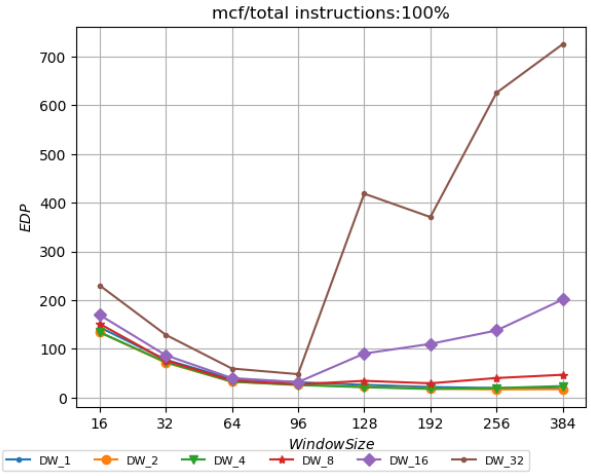
Hmmer

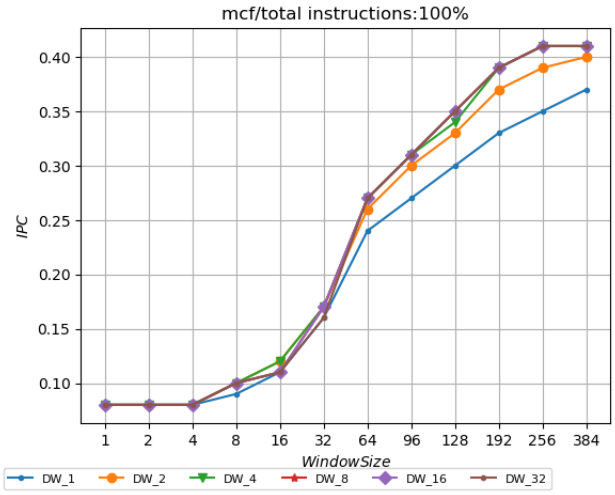

hmmer/total instructions:1.2770088729166666%
EDP
0.020
0.015
0.010
0.005
0.000
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

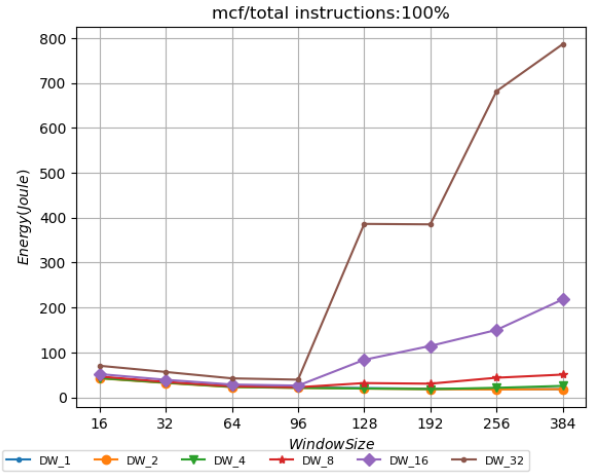

hmmer/total instructions:1.2770132236111111%
IPC
2.25
2.00
1.75
1.50
1.25
1.00
0.75
0.50
1
2
4
8
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32


hmmer/total instructions:1.2770088729166666%
Energy(Joule)
10
8
6
4
2
0
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

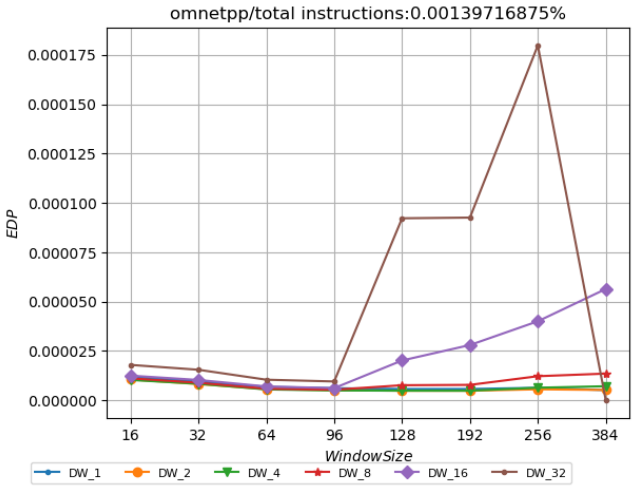
Mcf

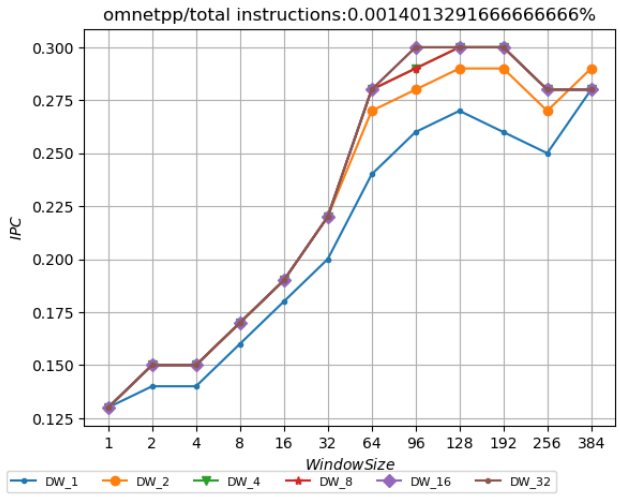

mcf/total instructions:100%
EDP
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

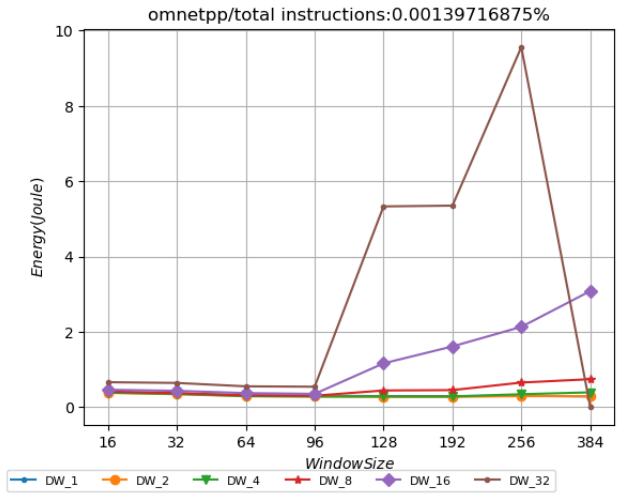

mcf/total instructions:100%
IPC
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32


mcf/total instructions:100%
Energy(Joule)
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

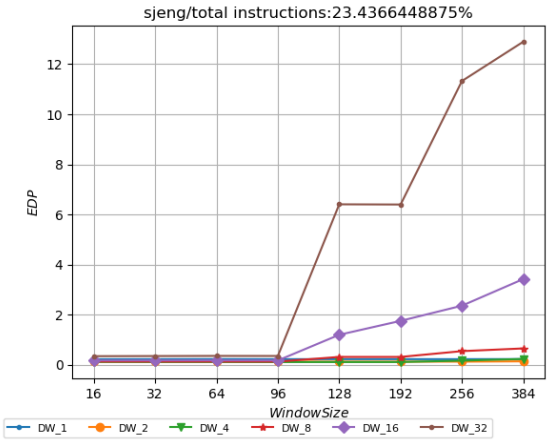
Omnetpp

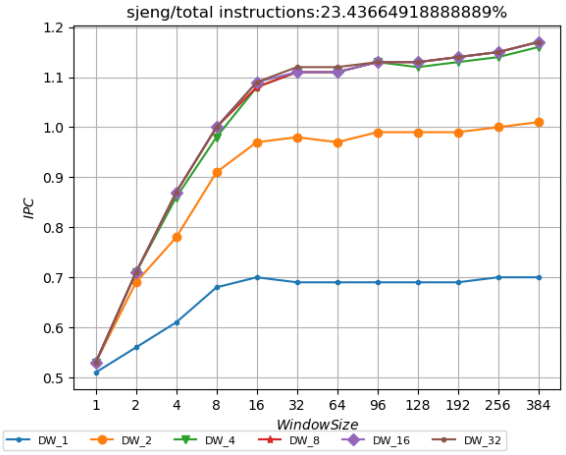

omnetpp/total instructions:0.00139716875%
EDP
0.000175
0.000150
0.000125
0.000100
0.000075
0.000050
0.000025
0.000000
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

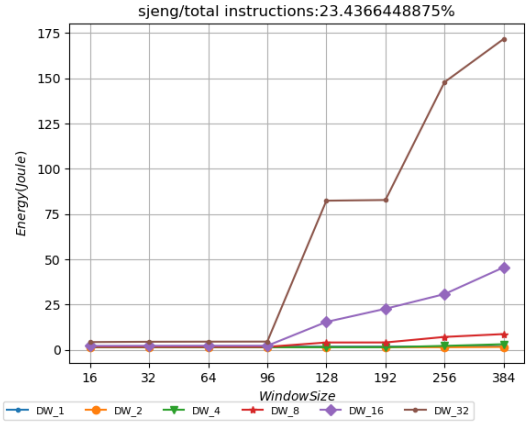

omnetpp/total instructions:0.0014013291666666666%
IPC
0.300
0.275
0.250
0.225
0.200
0.175
0.150
0.125
1
2
4
8
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32


omnetpp/total instructions:0.00139716875%
Energy(Joule)
10
8
6
4
2
0
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

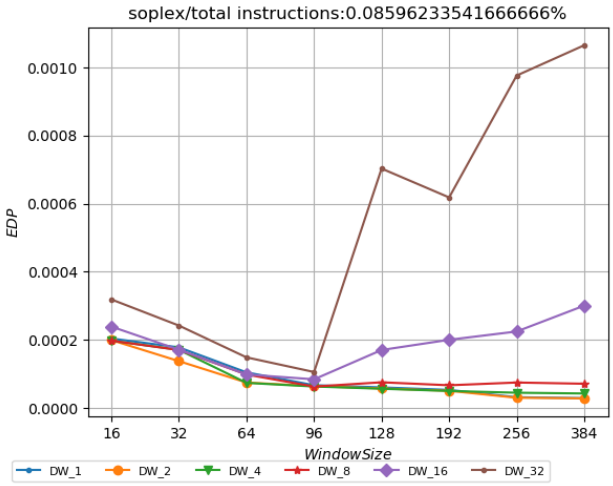
Sjeng

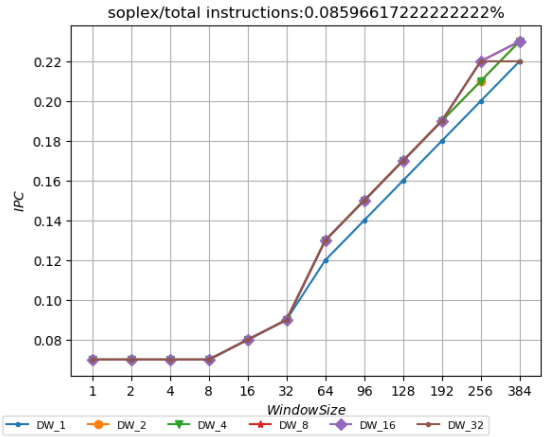

sjeng/total instructions:23.4366448875%
EDP
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

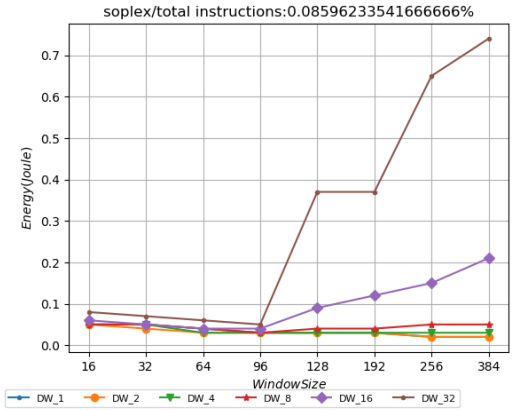

sjeng/total instructions:23.43664918888889%
IPC
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32


sjeng/total instructions:23.4366448875%
Energy(Joule)
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

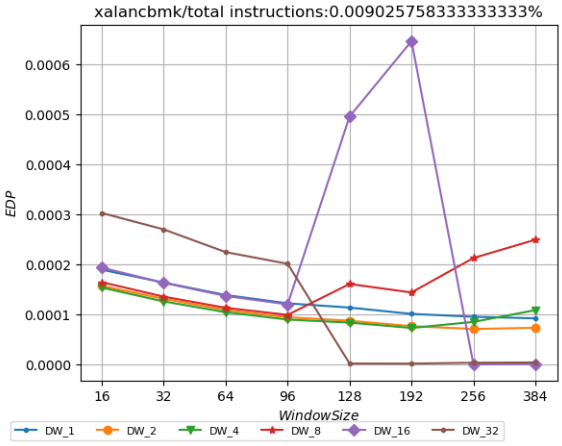
Soplex

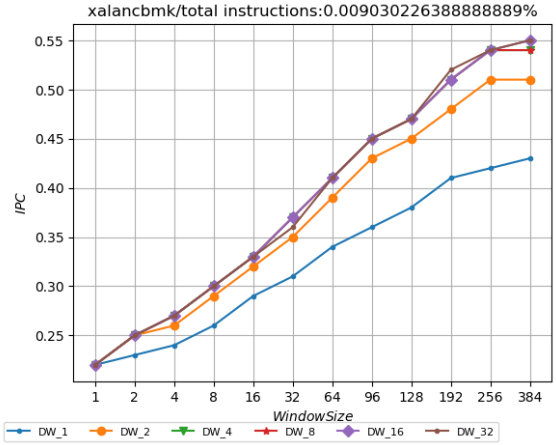

soplex/total instructions:0.08596233541666666%
EDP
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

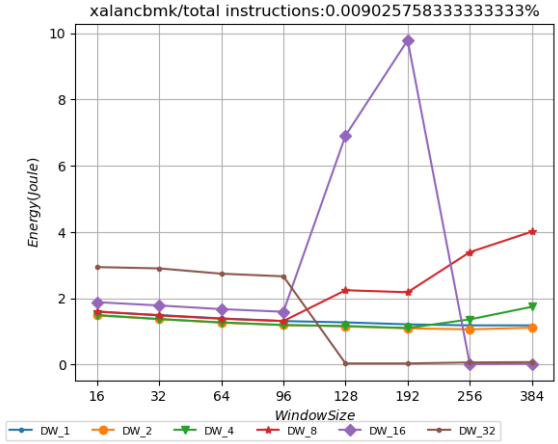

soplex/total instructions:0.08596617222222222%
IPC
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32


soplex/total instructions:0.08596233541666666%
Energy(Joule)
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

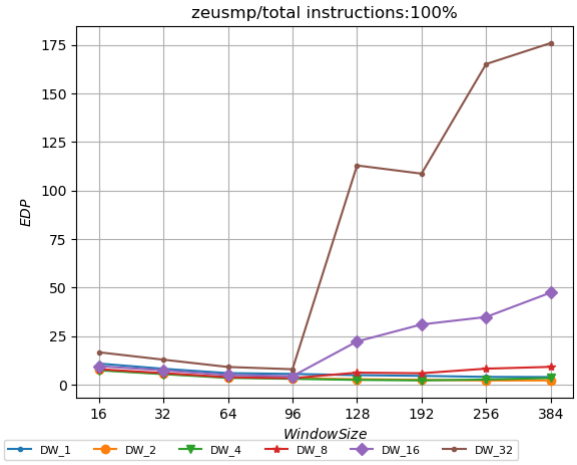
Xalanbcmk

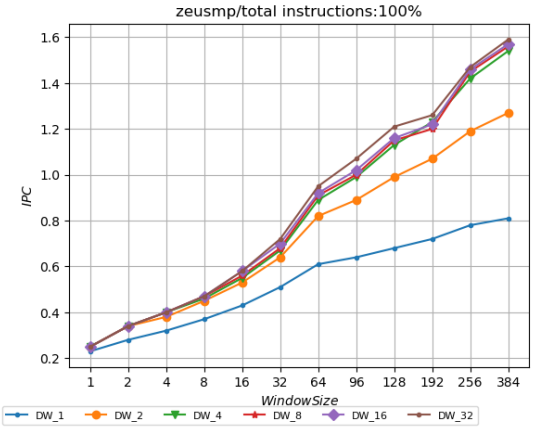

xalancbmk/total instructions:0.009025758333333333%
EDP
0.0006
0.0005
0.0004
0.0003
0.0002
0.0001
0.0000
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

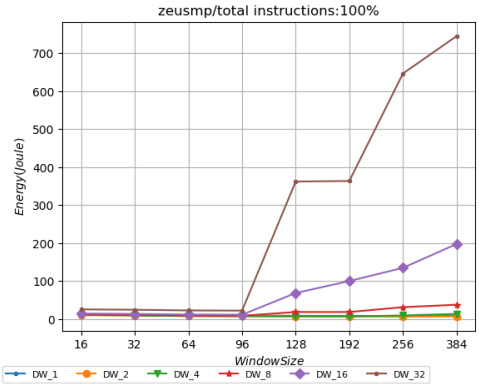

xalancbmk/total instructions:0.009030226388888889%
IPC
0.55
0.50
0.45
0.40
0.35
0.30
0.25
1
2
4
8
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32


xalancbmk/total instructions:0.009025758333333333%
Energy(Joule)
10
8
6
4
2
0
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

Zeusmp







C.Questions-Conclusions

I If the inequality window size<dispatch width is true, any further growth of dispatch width cannot alterate the performance of our system. This is a case because in every clock cycle the processor will handle the workload that exists in the ROB, so anything more cannot be done even thought we have mfurther increase of the dispatch width. Thid would have allowed us to make less simulations in our study, except maybe some with window size<dispatch to be true in order to prove this observations. This approach was not followed thought due to the circumstance that some of the benchmarks cannot be completed, due to internal failure, so our system resources could be used to simulate everything

ii) If we consider the function IPC with independent variable the window size we observe an increasing function , for different dispatch width values. So the increase of window size will give a better performance. In a narrow window of windows' values, a little different for its benchmark, the increase is exponential.

Another observation is that for the same window size value the increase of dispatch width results in better performance for our core(single core processors where simulated)

iii)As for the energy consumption of the processors we calculated a function of energy in joule with independent variable window size, for different dispatch width values, and again the function is increasing. We exclude some dispatch width values due to inefficient design of the McPAT program

The energy consumption is near zero(comparably to the rest of the function values really small) for window size values less than 96(values of order 0 or 1,base of 10, no more than 100(J))

We calculated EDP,(in expiramentation there was not a significant qualitative difference between EDP, ED^2P or other orders of ED^kP). In many, almost all ,cases there was a decrease in EDP value till the value of window size 96 were exponential increase was observed.

IPC started to flatten in each benchmark, starting from values ranging from 8 to 64 with the mean value in between, no exceptions observed in the accepted benchmarks.

AS a rule of thumb, we propose in a chip design window size between 16,32,64 to be implemented to achieve both performance and energy efficiency.

Dispatch width should be kept as low as possible to avoid extra unessecary use of space, with no significant increase in performance, so we recommend between 2 to 8(Possibly plots could be drown thought the lack of time could not allow that, this conclusion in drive from observation of some values) and theory)

iv) Our processor: Intel core 2 cpu cpu 4400 2.00GHz but I couldnot find information for this processor

For SKYLEKE and SANDY BRIDGE I found that window size is 224 and 168 and dispatch width is 6 and 4 rescpectivelly. So for skylake we know that if ROB has 224 microperations scheduler that has 8 ports can issue up to 8 instructions.Window size gives sufficient performance and we coclude that it is not wider due to energy restrictions

D.The code was written for energy information extraction and plotting of the various data, both .py and .sh files