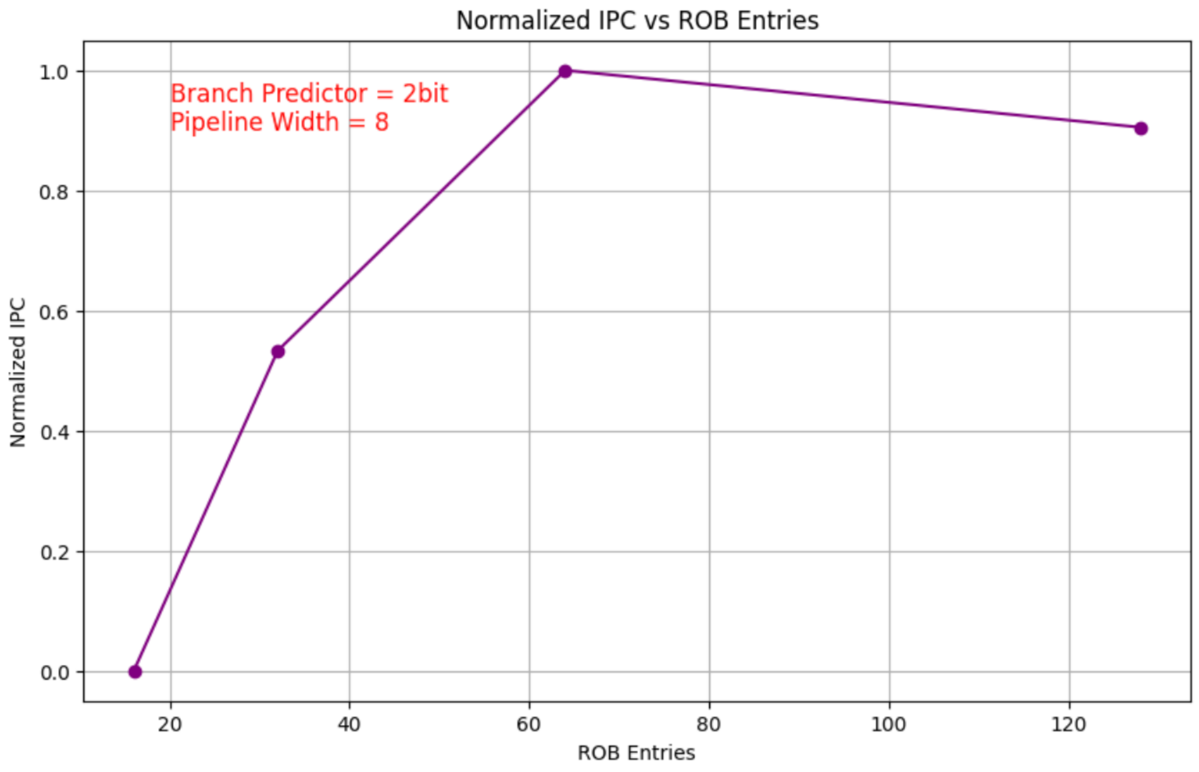
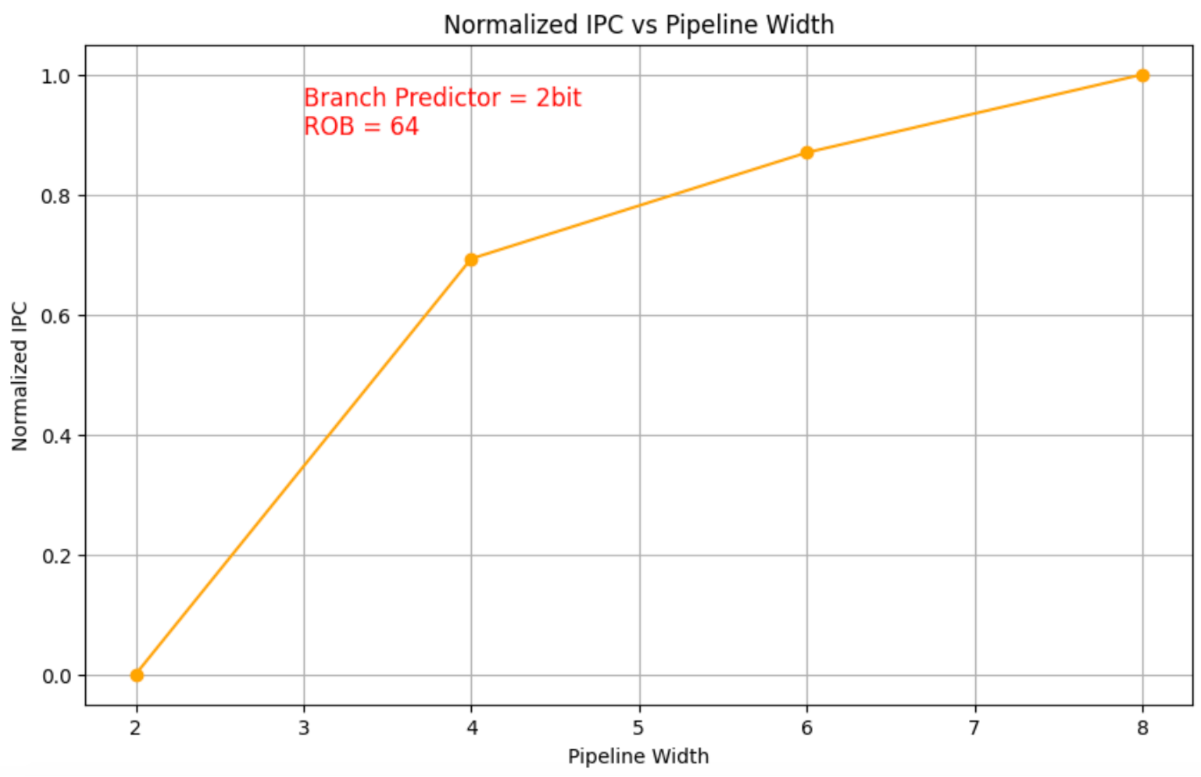
Assignment #4: Core Pipeline Width, ROB size and Branch Prediction

The Gem5 tool is widely used in computer architecture simulations as it provides significant insights which helps us to understand sensitivity of various parameters by varying and adjusting different parameters like CPU, Memory Configurations, frequency etc..
Gem5 allows us to compare the simulation results by providing use various performance metrics.
The focus of this assignment lies in how parameters like pipeline width, branch predictor and ROB affect the performance of benchmarks. Goal is to arrive at the Good Core architecture.

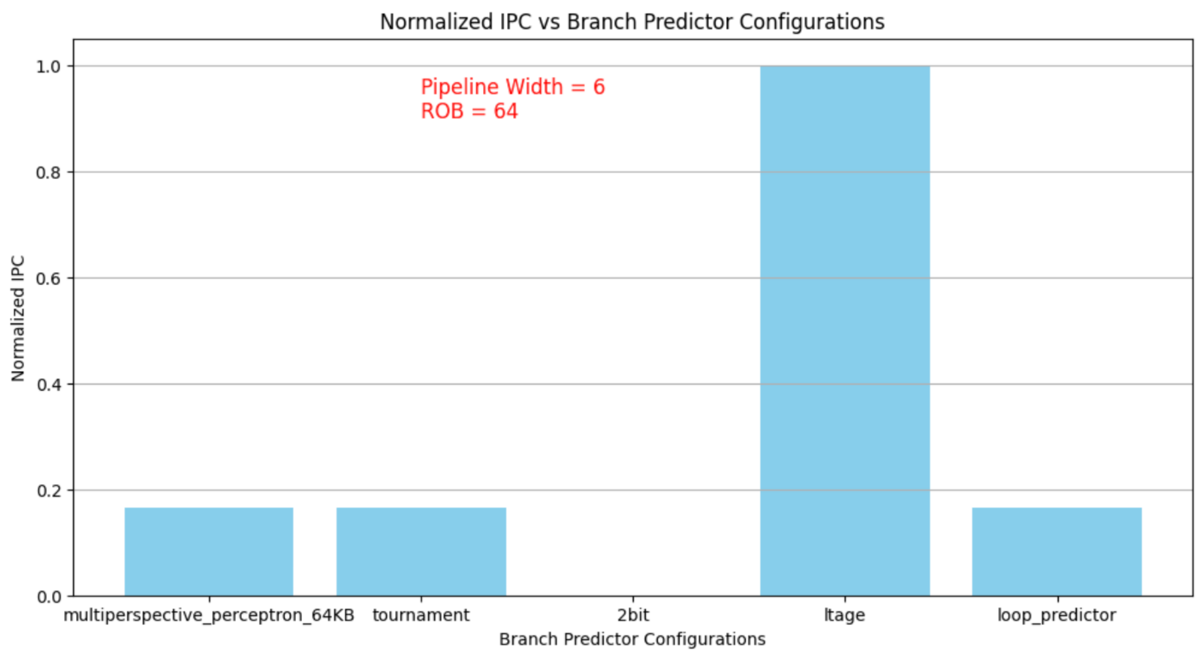
Experiments:



The above graph show behaviour of values of IPC with respect to varying ROB. I kept branch predictor as 2bit and pipeline width as 8. As you can see max performance behaviour is given by ROB at 64 so i took ROB as 64 for Good Core configuration.



The above graph show behaviour of values of IPC with respect to varying Pipeline Width. I kept branch predictor as 2bit and ROB as 64. As you can see max performance behaviour is given by width at 8 but difference between 6 and 8 is not big so it is not worth the resources to use 8 as pipeline width. So i took width as 6 for Good Core configuration.

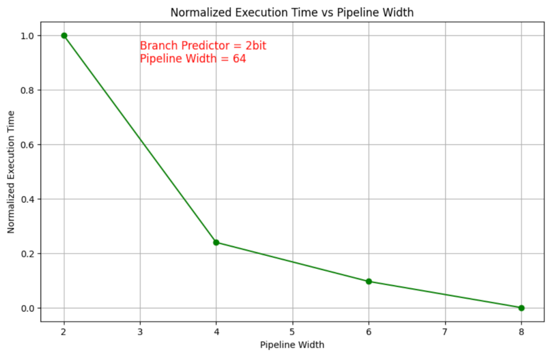


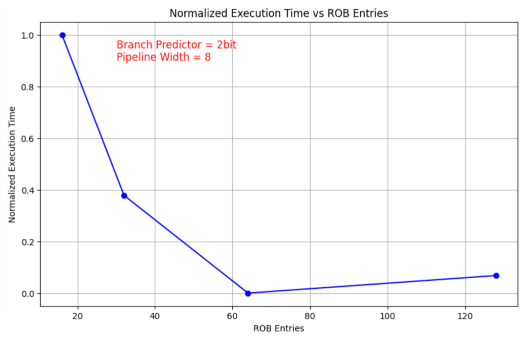
The above graph show behaviour of values of IPC with respect to different branch predictors. I kept pipeline width as 6 and ROB as 64. As you can see max behaviour is given by branch predictor ltage. As difference is huge I took branch predictor as ltage for Good Core configuration. Therefore i get GoodCore asPipeline Width = 6, ROB Entries = 64, Branch Predictor = Ltage.

Questions:

1. **What is the benchmark you picked? Also paste the link to the zip folder for relevant simulation outputs & scripts**. (1 line)
   Benchmark : Stanford[5]: quicksort, Link: Drive Link

2. **Which core parameter(s) have the most impact on performance? Are there any dependencies between the parameters?** (Max 150 words)





   The core parameters that have the highest impact on performance are the ROB and pipeline width. If the ROB size increases it allows the CPU to handle instructions parallelly. Similarly, pipeline width plays the same way.
   There are dependencies between these parameters. Regarding the dependencies between the parameters, increasing the value of pipeline width will get best results if and only if the ROB size is large enough to support the instructions. There can be a situation where potential is not fully reached. If we go with a smaller ROB, performance gains from a wider pipeline may not reach its full potential because of the size of it. Therefore, we need to fine tune these parameters in order to achieve balanced and efficient performance improvements.

3. **What are the parameters you chose? Let us call this GoodCore.** (3 lines)
   pipeline_width = 6, branch_preditctor = ltage, num_robs = 64
   This configuration strikes a balance between the performance benefits of the LargeCore and the resource efficiency of the SmallCore

4. **What are the IPC improvements and execution time speedups of LargeCore and GoodCore with respect to the SmallCore?** (3 lines / 1 table)
   SmallCore: Pipeline Width = 2, ROB Entries = 16, Branch Predictor = 2bit
   GoodCore: Pipeline Width = 6, ROB Entries = 64, Branch Predictor = Ltage
   LargeCore: Pipeline Width = 8, ROB Entries = 192, Branch Predictor = Ltage

Core	IPC	IPC Improvei	SimSeconds	Execution Time Speedup
SmallCore	0.703426	-	0.120297	-
GoodCore	1.277605	1.82x	0.06233	1.93x
LargeCore	1.360978	1.94x	0.062176	1.94x

5. **How did the benchmark characteristics affect your GoodCore design decisions? Look at the benchmark code (both the .c and .s files may be useful). (Max 150 words)**
   Quicksort is a recursive, comparison based sorting algorithm. It has multiple branching decision and will benefit from a better branch predictor. Higher pipeline width and branch predictor like ltage will ensure efficient out of order execution and

minimizing pipe line stalls. 64 ROB is balanced value and it is not too high but sufficient to maintain number of instructions to improve performance without increasing the usage of the resources. I analysed the benchmark and noticed that recursion will require parallel instruction handling and good branch predictor to achieve optimum performance.

6. **How did you reduce the search space? (Max 50 words)**
   As report says "Fix two …..”
   First I maintained a constant value for the branch-predictor and pipeline width while varying the ROB entries to get best ROB. Then I used ROB and initial branch-predictor while varying the pipeline width to determine the best width. Finally, while varying the branch-predictor to achieve the GoodCore.