NC State University

Department of Electrical and Computer Engineering

ECE 463/563: Fall 2022 (Rotenberg)

Project #3: Dynamic Instruction Scheduling

by

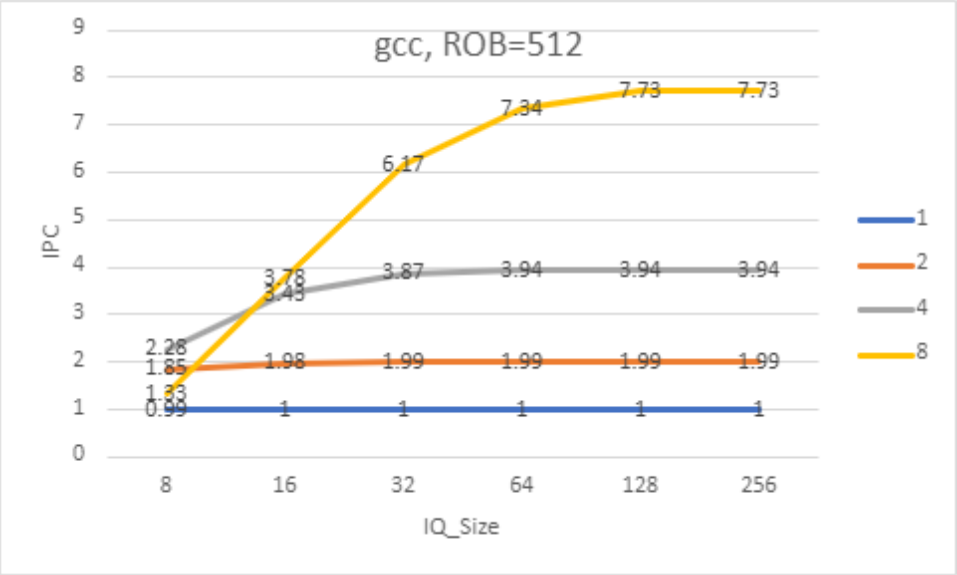
<< Darsh Kiran Asher >>

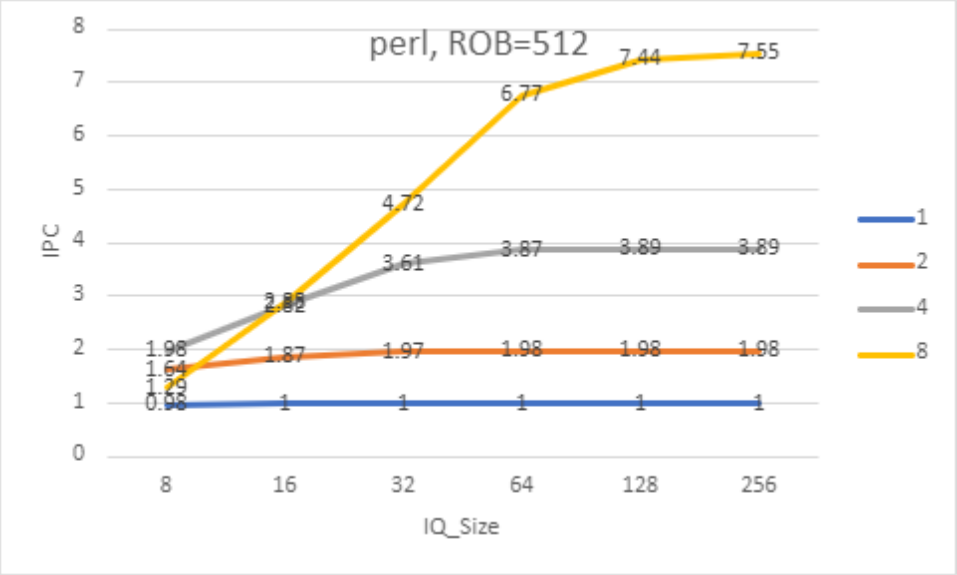
NCSU Honor Pledge: "I have neither given nor received unauthorized aid on this project."

Student's electronic signature: Darsh Kiran Asher
(sign by typing your name)

Course number: 563
(463 or 563 ?)

1. **Graphs [10 points]:** Keep ROB_SIZE fixed at 512 entries so that it is not a resource bottleneck. For each benchmark (gcc and perl), make a graph with IPC on the y-axis and IQ_SIZE on the x-axis. Use IQ_SIZE = 8, 16, 32, 64, 128, and 256. Plot 4 different curves (lines) on the graph: one curve for each of WIDTH = 1, 2, 4, and 8. Title the two graphs "gcc, ROB=512" and "perl, ROB=512", respectively.





2. **Graph Analysis [2 points]:**
Using the data in the graphs above, for each WIDTH (1, 2, 4, and 8), find the minimum IQ_SIZE that still achieves within 5% of the IPC of the largest IQ_SIZE (256). This exercise should give four optimized IQ_SIZE's per benchmark, one optimized for each of WIDTH = 1, 2, 4, and 8. Tabulate the results of this exercise as follows:

	"Optimized IQ_SIZE per WIDTH" Minimum IQ_SIZE that still achieves within 5% of the IPC of the largest IQ_SIZE	
	gcc	perl
WIDTH = 1	8	8

WIDTH = 2	8	13
WIDTH = 4	24	35
WIDTH = 8	64	81

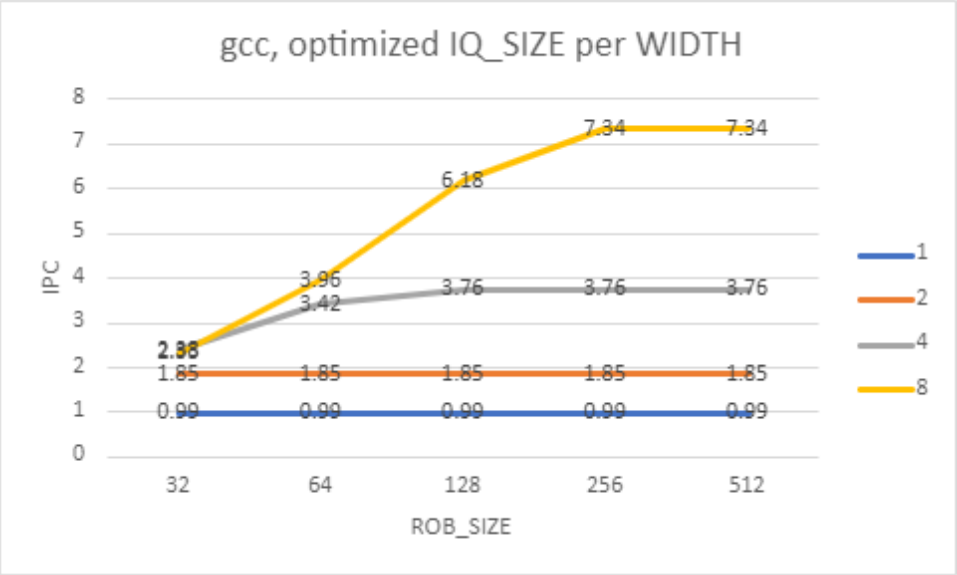
Grading rubric: Each cell in the table is worth 0.25 points.

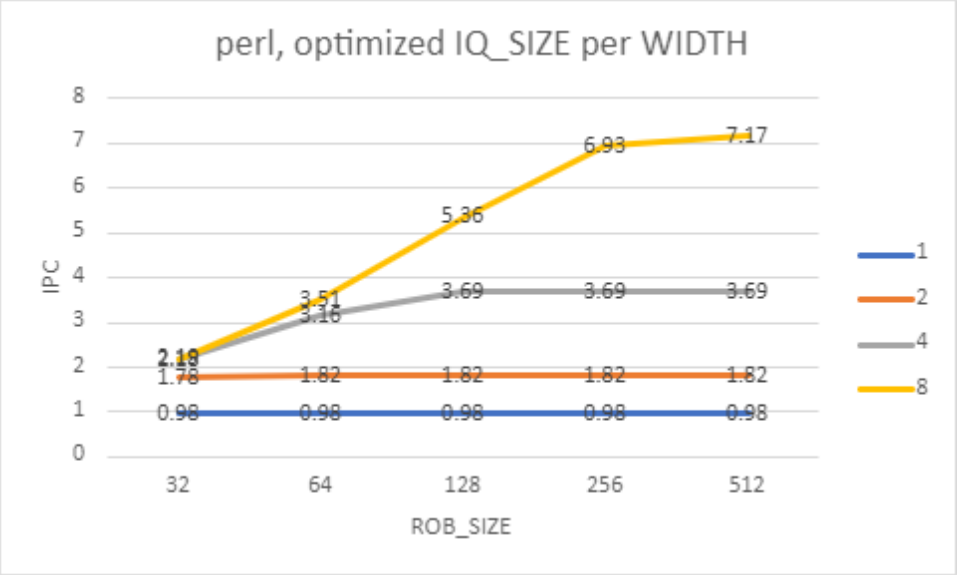
3. **Discussion [2 points]:**

   - The goal of a superscalar processor is to achieve an IPC that is close to WIDTH, which is the peak theoretical IPC of the processor. As we increase WIDTH, we observe that a <larger> IQ is needed to achieve this goal. This is because, with greater WIDTH, the IQ needs to look <farther> in the dynamic instruction stream to find <more> independent instructions that can issue in parallel to WIDTH execution lanes, each cycle.
   - For WIDTH=8, perl's "optimized IQ_SIZE" is <less than> gcc's "optimized IQ_SIZE" (reference your table above).
     Why might this be the case?
     a. Perhaps perl has <more> data-dependent instructions within a fixed window of instructions, such that it <may look closer> in the dynamic instruction stream to get the same number of independent instructions as gcc.
     b. Perhaps perl has <fewer> long-latency instructions within a fixed window of instructions as compared to gcc.
     c. All of the above: both a and b are plausible explanations.
     Answer: <c>

7.3.1. Effect of ROB_SIZE

4. **Graphs [6 points]:** For each benchmark (gcc and perl), make a graph with IPC on the y-axis and ROB_SIZE on the x-axis. Use ROB_SIZE = 32, 64, 128, 256, and 512. Plot 4 different curves (lines) on the graph: one curve for each of WIDTH = 1, 2, 4, and 8. For a given WIDTH, use the optimized IQ_SIZE for that WIDTH, as obtained from the table in Section 7.3.1. Title the two graphs "gcc, optimized IQ_SIZE per WIDTH" and "perl, optimized IQ_SIZE per WIDTH", respectively.


gcc, optimized IQ_SIZE per WIDTH
IPC
ROB_SIZE
32
64
128
256
512
0
1
2
3
4
5
6
7
8
1.85
1.85
1.85
1.85
1.85
0.99
0.99
0.99
0.99
0.99
2.38
3.95
6.18
7.34
7.34
3.42
3.76
3.76
3.76
1
2
4
8


perl, optimized IQ_SIZE per WIDTH
IPC
ROB_SIZE
32
64
128
256
512
0
1
2
3
4
5
6
7
8
1.78
1.82
1.82
1.82
1.82
0.98
0.98
0.98
0.98
0.98
2.19
3.51
5.36
6.93
7.17
3.16
3.69
3.69
3.69
1
2
4
8