

**NC State University**  
**Department of Electrical and Computer Engineering**  
**ECE 463/563: Fall 2022 (Rotenberg)**  
**Project #3: Dynamic Instruction Scheduling**

**by**

**<< Nirmal Kumar Sedhumadhavan >>**

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

Student's electronic signature: **Nirmal Kumar Sedhumadhavan**  
(sign by typing your name)

Course number: **563**  
(463 or 563 ?)

## 7. Tasks, Grading Breakdown

### 7.1. [0 – 50 points] Simulator development effort

You may earn up to 50 points for handing in significant commented code for your simulator, even if the simulator does not compile, run, and validate. To receive the maximum of 50 points, there needs to be a good-faith attempt at coding the functionality of all pipeline stages described in Section 5 as well as the supporting code (parsing arguments, reading trace file, producing outputs, the `Advance_Cycle()` function, the `do-while()` simulator loop, *etc.*). Partial credit will be assessed at a coarse granularity based on a qualitative assessment by the TAs of the amount of functionality coded.

### 7.2. [30 points] Validation

Gradescope will evaluate your simulator on the eight validation runs, “val1.txt” through “val8.txt”, posted on the website. Gradescope will also evaluate your simulator on two mystery runs. Each validation run and mystery run is worth 3 points. Gradescope must say that you match all eight validation runs to get credit for the experiments with the simulator (report), however.

### 7.3. [20 points] Report: graphs, graph analysis, and discussion

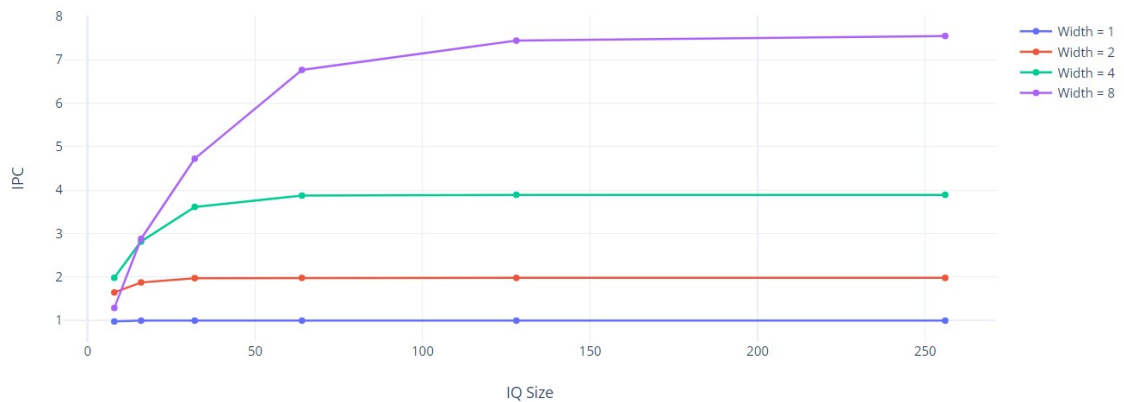
*Note: you can only get credit for the report if Gradescope says that you match all eight validation runs.*

#### 7.3.1. Large ROB, effect of IQ\_SIZE

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.



Perl Trace



**Grading rubric:** (1) 5 points will be deducted for each missing graph. (2) For each graph that exists in the report, the TAs will check for missing or blatantly incorrect datapoints. As there are 24 datapoints (6 iq sizes x 4 widths), the TAs will deduct 0.2 points for each missing or blatantly incorrect datapoint that is spotted.

## 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	IQ size = 8	IQ size = 8
WIDTH = 2	IQ size = 16	IQ size = 16
WIDTH = 4	IQ size = 32	IQ size = 64
WIDTH = 8	IQ size = 64	IQ size = 128

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

## 3. Discussion [2 points]:

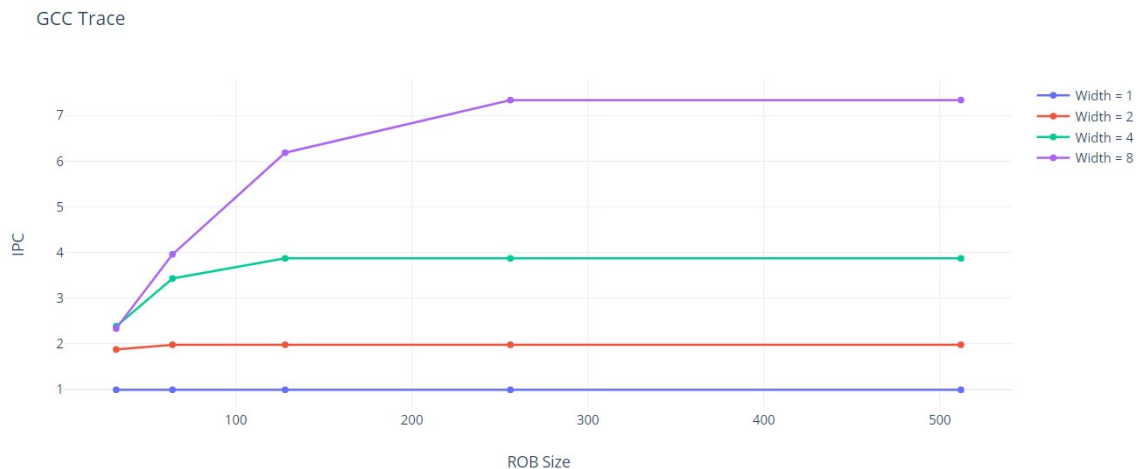
- The goal of a superscalar processor is to achieve an IPC that is close to WIDTH, which is the peak theoretical IPC of the processor. As we increase WIDTH, we observe that a **<larger>** IQ is needed to achieve this goal. This is because, with greater WIDTH, the IQ needs to look **<farther>** in the 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 **<greater 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 **<must look farther>** in the dynamic instruction stream to get the same number of independent instructions as gcc.
    - b. Perhaps perl has **<more>** 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>**

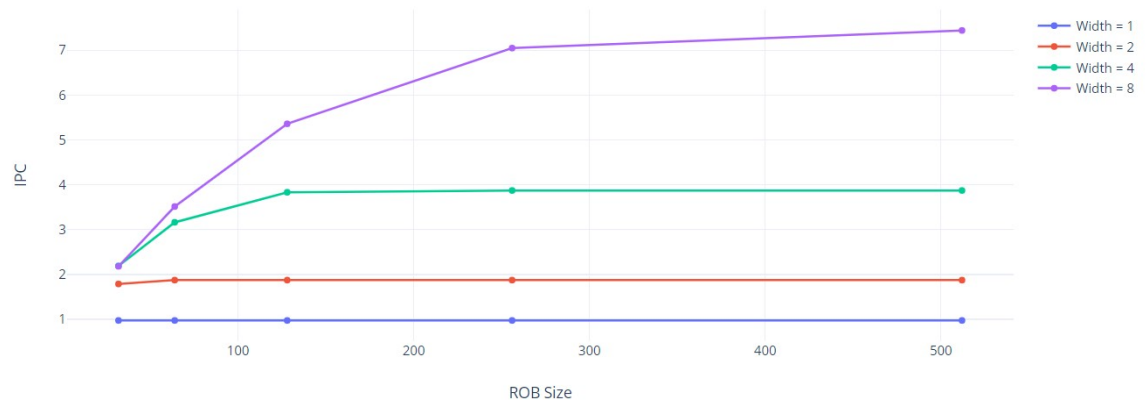
Grading rubric: Each yellow blank is worth 0.25 points. Note that there are eight yellow blanks.

### 7.3.2. 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.



Perl Trace



Grading rubric: (1) 3 points will be deducted for each missing graph. (2) For each graph that exists in the report, the TAs will check for missing or blatantly incorrect datapoints. As there are 20 datapoints (5 rob sizes x 4 widths), the TAs will deduct 0.15 points for each missing or blatantly incorrect datapoint that is spotted.