NC State University

Department of Electrical and Computer Engineering

ECE 563: Fall 2019

Project #3: Dynamic Instruction Scheduling

by

VARUN VARADARAJAN (vvarada2)

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

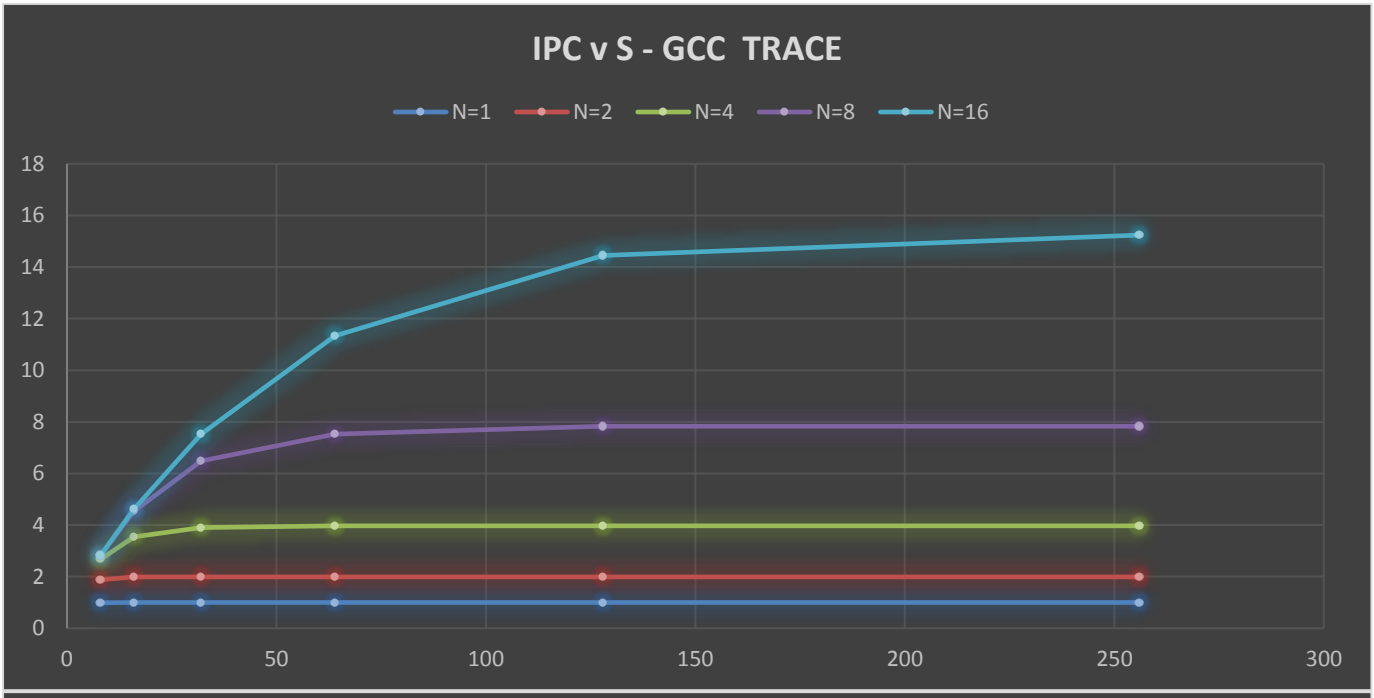
Student's electronic signature: _____
(Varun Varadarajan)

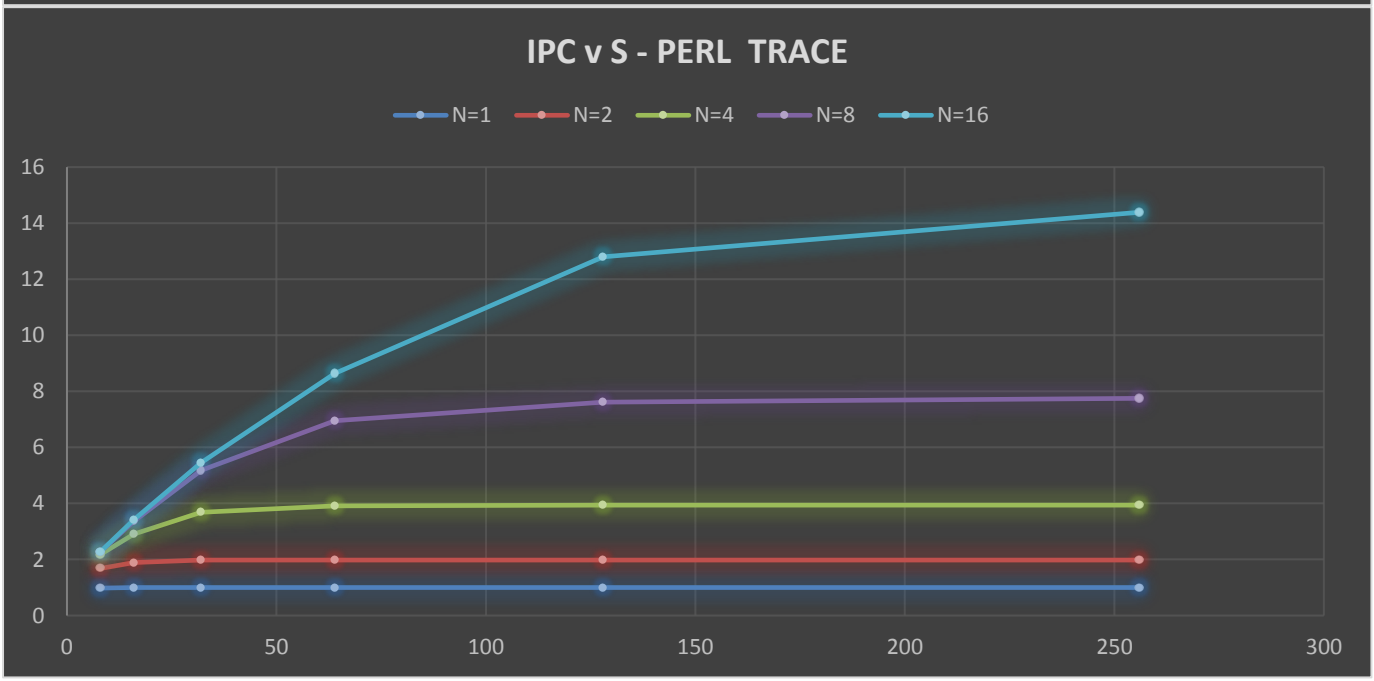
Course number: _____
(563)

GRAPHS AND OBSERVATIONS

Following configurations were evaluated (Row – N & Column – S)

GCC - N/S	8	16	32	64	128	256
1	0.99	1	1	1	1	1
2	1.88	1.99	1.99	1.99	1.99	1.99
4	2.67	3.54	3.9	3.97	3.97	3.97
8	2.82	4.54	6.48	7.52	7.83	7.83
16	2.82	4.62	7.52	11.34	14.45	15.24
PERL - N/S	**8**	**16**	**32**	**64**	**128**	**256**
1	0.98	1	1	1	1	1
2	1.68	1.89	1.98	1.98	1.98	1.98
4	2.18	2.91	3.68	3.91	3.94	3.94
8	2.28	3.37	5.18	6.95	7.61	7.75
16	2.28	3.42	5.46	8.64	12.8	14.39





When superscalar bandwidth is low, IPC is also very low. Because only few instructions are processed in a cycle, the performance will not change although the scheduling and dispatch queue sizes are high.

When superscalar bandwidth gets higher, IPC is improved because more instructions can be processed in a single cycle. Combined with higher queue sizes will maximize our IPC value.

I also investigated some cases where N > S **(extra 5th curve on graph)**. Here, the IPC won't change because although more instructions can be fetched in a cycle due to higher bandwidth, the queues size limit will come into play… and hence execute the same number of instructions in the later stages (for example, 'S')

When N (queue sizes) are small, performance of IPC is lower because even if we have higher fetch bandwidth, only those many number of instructions in these queues can be processed. Others have to wait and be stalled, like developed in the code.

When N=1, it means only one instruction can be fetched per cycle. So, IPC can never go past 1 because 'k' instructions take at least 'k' full cycles to finish. If stalled due to true dependencies, then IPC goes lower than 1 very easily.

Perl shows slightly lower IPC than Gcc trace. We can tell from this observation that Perl's program order of instructions is expected to have less register dependencies per subsequent instructions as compared to Gcc, although both have same instruction count. However, when we involve caches in this design, we can't really support this expectation because different traces can have different numbers of TYPE 2 OPERATION. So although IPC of Perl may be lesser, its total access time etc. can also be higher than Gcc… if in case it has more frequent cache misses.