NC State University

Department of Electrical and Computer Engineering

ECE 521: Fall 2014

Project #3: Dynamic Instruction Scheduling

by

SIDDHARTH SINGH

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

Student's electronic signature: SIDDHARTH SINGH
(sign by typing your name)

Course number: 521

INTRODUCTION

This project involves building a simulator to illustrate dynamic scheduling using Tomasulo's Algorithm. This simulator implements Tomasulo's algorithm with given Fetch/ Issue rate (N) and scheduling queue size (S). Advanced functional units like ROB (Read only Buffer) are not implemented in this simulator. Thus this simulator considers imprecise interrupt state.

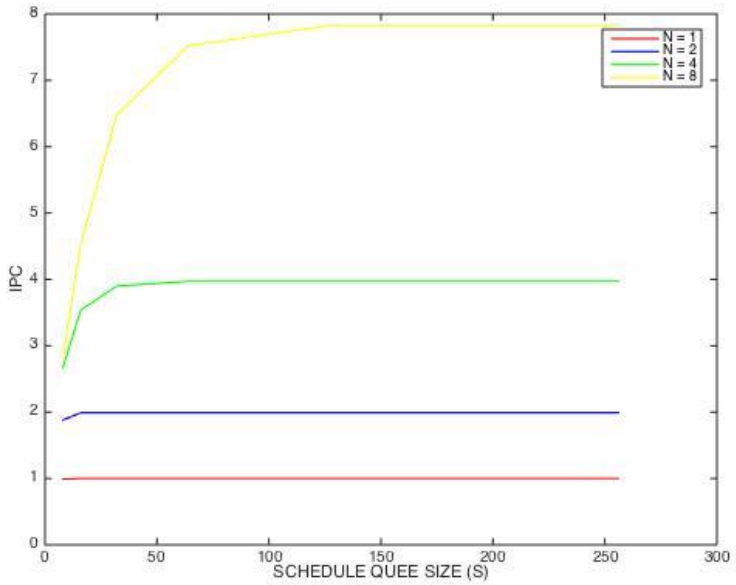
The prime target of this simulator is to observe performance gain with superscalar and out of order (OOO) execution with the given parameters. The execution speed of the system can greatly improved using superscalar as well as out of order execution. However due to data dependencies and RAW (Read after Write), WAW (Write after Write) Hazards the performance is greatly hampered.

This simulator can be configured to be run for different benchmarks as well as has the additional added feature of running it for a multiple level of memory hierarchy of L1 and L2 Data caches or it can simply be run for an ideal system of caches too.

Simulator Benchmarks

The simulator performance results were calculated with the two trace files viz. gcc_trace.txt, perl_trace.txt. The predictor parameters S, N are varied in the simulation. The following values are considered for the simulation for which the graphs were plotted the scheduling quee size was varied from S = 8 , 16 , 32 , 64 , 128 , 256, values of N = Degree of superscalar processor was varied from N = 1 , 2 , 4 ,8.

Gcc Benchmark



The above graph shows the experiment suggested above for GCC Trace benchmark.

Experimental Data for GCC Benchmark

S	N = 1	N = 2	N = 3	N = 4	N = 5	N = 6	N = 7	N = 8
	IPC	IPC	IPC	IPC	IPC	IPC	IPC	IPC
8	0.99	1.88	2.4	2.67	2.78	2.81	2.82	2.82
16	1	1.99	2.85	3.54	4	4.3	4.46	4.54
32	1	1.99	2.97	3.9	4.75	5.44	6.07	6.48
64	1	1.99	2.98	3.97	4.94	5.86	6.74	7.52
128	1	1.99	2.98	3.97	4.95	5.9	6.87	7.83
256	1	1.99	2.98	3.97	4.95	5.9	6.87	7.83

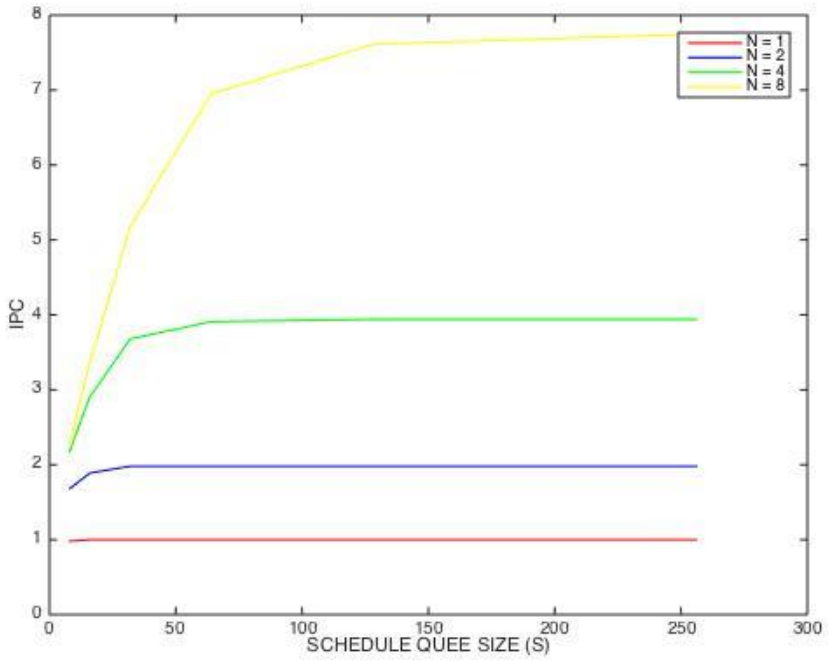
DISCUSSION

The trend suggested from above experimental data is that for GCC Benchmark for a given S(Scheduling size) and increasing N there is always a greater number of Instructions executed per cycle thus this trend results in highest values of IPC for N=8 for S = 256 and lowest or worst performance is obtained from S = 8 scheduling quee size.

Similarly for variation across N (Bandwidth of Superscalar processor) the N=8 reaps the maximum benefit out in giving us the maximum IPC while N = 1 or simple scalar processor does give us any benefit.

For Higher value of N > 1 we can clearly say that IPC is directly proportional to S and N values.
Clearly, the relationship between S and N is that higher the values of both the parameters higher the IPC that can be obtained for this gcc benchmark. This benchmark at optimised S and N values offers best performance of IPC = 7.83 at S = 256 and N = 8.

Perl Benchmark



The above graph shows the experiment suggested above for PERL Trace benchmark.

Experimental Data for Perl Benchmark

S	N = 1	N = 2	N = 3	N = 4	N = 5	N = 6	N = 7	N = 8
	IPC	IPC	IPC	IPC	IPC	IPC	IPC	IPC
8	0.98	1.68	2.03	2.18	2.23	2.26	2.27	2.28
16	1	1.89	2.51	2.91	3.16	3.28	3.33	3.37
32	1	1.98	2.9	3.68	4.28	4.7	4.97	5.18
64	1	1.98	2.96	3.91	4.79	5.62	6.34	6.95
128	1	1.98	2.97	3.94	4.88	5.83	6.75	7.61
256	1	1.98	2.97	3.94	4.98	5.86	6.8	7.75

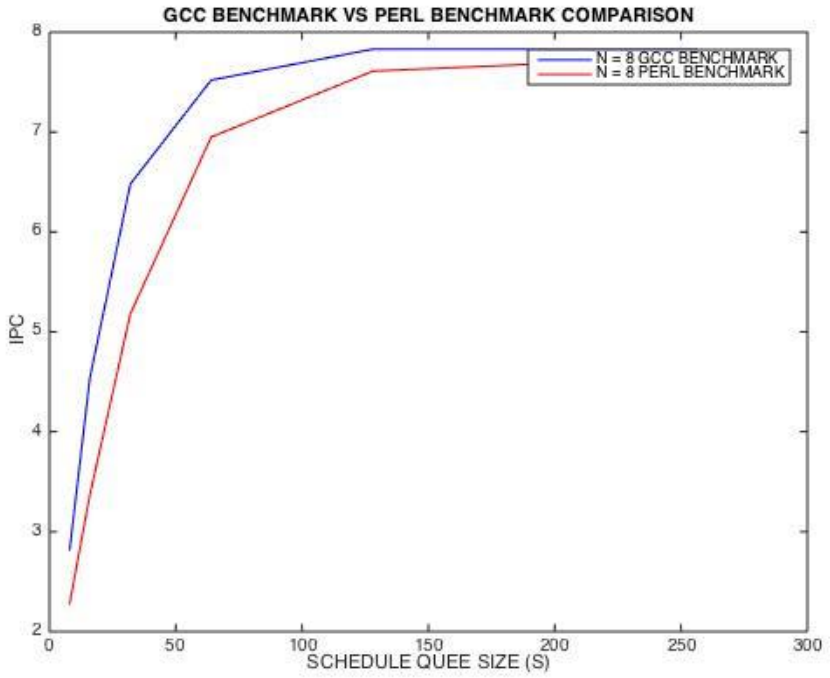
DISCUSSION

The trend suggested from above experimental data is that for PERL Benchmark for a given S(Scheduling size) and increasing N there is always a greater number of Instructions executed per cycle thus this trend results in highest values of IPC for N=8 for S = 256 and lowest or worst performance is obtained from S = 8 scheduling queue size.

Similarly for variation across N (Bandwidth of Superscalar processor) the N=8 reaps the maximum benefit out in giving us the maximum IPC while N = 1 or simple scalar processor does give us any benefit.

For Higher value of N > 1 we can clearly say that IPC is directly proportional to S and N values.
Clearly, the relationship between S and N is that higher the values of both the parameters higher the IPC that can be obtained for this gcc benchmark. This benchmark at optimised S and N values offers best performance of IPC = 7.75 at S = 256 and N = 8.

COMPARISON OF GCC BENCHMARK VS PERL BENCHMARK



We can clearly see from the above graph that GCC benchmark simulations always offer higher IPC irrespective of any value of S or N chosen. We can clearly infer from this kind of behavior that there are higher level of Data dependencies in GCC benchmark than in Perl benchmark thus this is the reason as we choose higher values of S and N that we reap better benefits with it.