NC State University

Department of Electrical and Computer Engineering

ECE 463/563: Fall 2017

Project #3: Dynamic Instruction Scheduling

by

<< AMARESH SUBBURAJ>>

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

Student's electronic signature: _____Amaresh Subburaj________________________

Course number: _________563__________

Intro into Hardware Based Dynamic Scheduling:

In in-order execution machine the instructions are fetched in order, decoded, executed written back to the register file in order. If RAW , WAR ,WAW dependency happens then the processor stalls. This leads to decrease in instructions per cycle and resources are not fully utilized. A static dynamic scheduling involves compiler to rearrange the instructions in a program so the result produced will be same as the in-order execution.

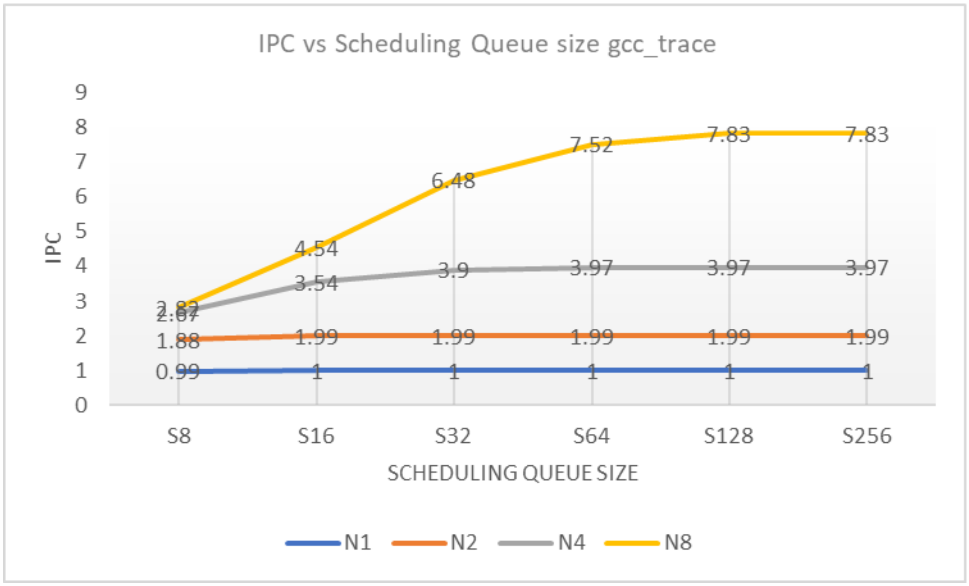
A dynamic scheduling involves having extra stages in pipeline namely Rename stage and has renaming table and reservation stations.
The renaming station removes the false dependency arising out of the use same register name by renaming it.
The reservation station has renamed Instruction waiting for its source register values to be made available ready and once its available is sent to FU based on priority.
Normally oldest instruction is given priority. The below results are simulated for the given trace using the tamasulo algorithm with imprecise interrupts

The below table has **IPC** listed for different scheduling size vs super scalar bandwidth for **gcc_trace** provided.

Scheduling queue size	N1	N2	N4	N8
S8	0.99	1.88	2.67	2.82
S16	1	1.99	3.54	4.54
S32	1	1.99	3.9	6.48
S64	1	1.99	3.97	7.52
S128	1	1.99	3.97	7.83
S256	1	1.99	3.97	7.83

The below table has **IPC** listed for different scheduling size vs super scalar bandwidth for **perl_trace** provided.

Scheduling queue size	N1	N2	N4	N8
S8	0.99	1.68	2.18	2.28
S16	1	1.89	2.91	3.37
S32	1	1.98	3.68	5.18
S64	1	1.98	3.91	6.95
S128	1	1.98	3.94	7.61
S256	1	1.98	3.94	7.75



From the above diagrammatic representation, it can be inferred that, IPC (Instruction count per cycle) is depends on,

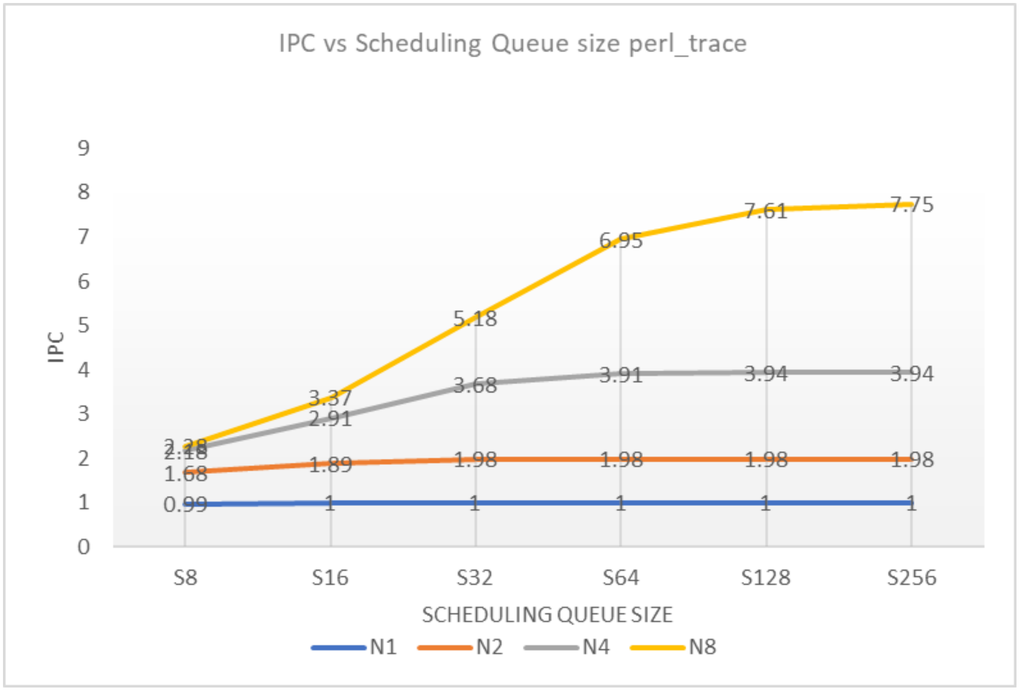
1) Scheduling Queue size
2) Super scalar bandwidth.

Suppose we consider a S=8 then an increase in N cause an increase in IPC. Hence it can be inferred that IPC is directly proportional to Super scalar bandwidth.

As scheduling queue size increases for a particular N value it responds linear to some extent and then becomes flat. From this behavior we can infer that the increase in scheduling queue can create extra space to store reordered instructions and hence IF and ID stages may not be stalled if scheduling queue becomes full.

But if the scheduling queue size is increased beyond a certain extant, its waste of resource as the extra space may never be used. This behavior is characterized by the flat portion of the curve.

The above inference holds good for both gcc trace and perl trace.



So, for a given N an increase in S leads to linear increase in IPC and beyond a certain point S value the increase in S does not lead to improvement in IPC and IPC becomes almost constant.

We can see that gcc trace has a slightly better performance in terms of IPC compared to perl trace. This can be ascribed to number of factors.

1)The perl trace may have a lot of type 2 instructions which may lead to an increase in execution time since memory has to accessed to load value.

2)The gcc trace could be having less real dependent instructions compared to perl trace.