Summary
Continual Flow Pipelines
Venkatesan Packirisamy
April 20, 2005

1 To be completed before class

What are the problems solved by this paper? (50 words)
Conventional superscalar processors relied on large instruction windows to extract instruction level parallelism. But these large instruction windows are not scalable leading to many design complexities and power wastage. We need such large windows to hold large number of instructions, so if any earlier instructions gets held up due to long latency operations, we can get other independent instructions to execute. But as the memory and CPU performance gap increasing, solution of using large windows in not efficient. This paper uses smaller windows but can tolerate long latency cache misses.

What are the approaches attempted by this paper? (50 words)
Continual flow pipelines allows processors to execute instructions after the long latency cache miss that are independent of the miss instruction. All the instructions that are dependent on the load is put into a Slice data buffer. The register mappings are properly handled in such a way that the following instructions will be able to execute as usual. When the load instruction completes, the instructions in the buffer are also committed. Thus we can still extract ILP beyond cache misses.

What are the main conclusions of this paper? (50 words)
With Continual flow pipelines we are able to tolerate the cache misses with a much smaller instruction windows than the conventional ROB based architectures. The CFP technique is found to be more effective than the CPR technique aimed at tolerating low confidence branches. After applying CFP, the branch mis-predictions have become the major bottleneck.

2 To be completed after class

Did this paper address an important issue? Explain. (100 words)
To increase the performance of the current superscalar processors is to hide the long latency operations like cache misses. To do this the current approach is to increase the instruction window. But as we keep increasing the instruction window, we get diminishing performance. Also design complexity and other factors are more favourable for multi-core processors. So it is desirable to hide the long latency operations with smaller windows. If we have smaller windows then the pipeline depth will be reduced due to faster access of instruction windows. If we are able to hide cache misses, then cache size can also be reduced. So it is desirable to have a efficient technique which can hide cache misses and also reduces design complexity.

Are the proposed approaches valid? Describe its strength and weakness. (100 words)
The approach is to delay the execution of the instructions that are dependent on the long latency operations, by putting them into a slice buffer. The instructions which come after the cache miss, which are not dependent on the cache miss are allowed to execute as usual. So we are able to hide the long latency with the instructions that follow the cache miss. Also the instructions in the slice buffer are already renamed and decoded, so executing them after the cache miss is serviced is also faster.
Weakness: The solution looks more efficient than increasing window size. But this holds for only cache misses. If the main bottleneck is data dependencies, than increasing the window size is the only solution. The CFP is basically a special technique to hide data cache misses.

Do the results support the conclusions? Explain. (100 words)
The result compares the CFP with CPR, ideal CPR and the basic ROB technique. It is showed that CFP can almost reach the performance of the ideal CPR, which has infinite ROB. This shows that CPR can have the same effect of having very large number of ROB entries. So it proves their conclusion that a less number of slice buffer entries with very less complexity can replace the huge complex ROB. It is clear from their results that most of the instructions that follow the L2 cache miss are independent of the load instruction. So it proves the effectiveness of putting dependent instructions in a small buffer. Also they prove that they could have a smaller caches and achieve similar performance. So the technique can potentially make the cores smaller.

Describe the potential future works? (100 words)
Though the slice buffer can hide the L2 cache misses, the data shows that L1 misses could be a bottleneck. So we could use a similar technique of having a distributed instruction window, by delaying each window and using the other window to take the following instructions and execute. The data also show that branch prediction could be a bottleneck, so we need similar techniques to take branch mis-predictions. We could use the slice buffer to improve the power efficiency. So if we could reduce the unnecessary occupancy of the instruction window, we could reduce the power consumption of the instruction window.