Analyzing Branch Prediction Impact on the Effectiveness of Instruction Prefetch

Tatiana G. S. dos Santos, Maurício L. Pilla, Guilherme Dal Pizzol, Sergio Bampi and Philippe O. A. Navaux

[1] Universidade Federal do Rio Grande do Sul
CP 15064, Porto Alegre - RS - Brazil, 91501-970
E-mail: {tatiana, pilla, gpizzol, bampi, navaux}@inf.ufrgs.br

Abstract—
In order to provide a high instruction throughput for advanced superscalar models, L1 caches must achieve a low miss rate with no increase or minimum change in L1 capacity.

Prefetch mechanisms anticipate data from memory to L1 caches ahead of its request by the CPU instruction execution flow. These kind of schemes reduce the miss occurrence as well as help to hide the memory access latency.

The wrong path prefetch anticipates the next block and also the taken target of a branch. This work analyzes how the branch prediction may influence in the performance of this prefetch scheme. The number of wasted cycles and the effects of cache pollution and prefetch overlap are also presented. [1]

Keywords— **Prefetch mechanisms, branch prediction, superscalar architectures, cache memory.**

I. Introduction

One of the main challenges in superscalar architectures [JOH91] is to provide a sufficient number of decoded instructions to feed the execution engine. The L1 I-cache used in these computers must reach a high hit rate. To meet such requirement, each of the L1 caches have had their sizes increased in the last processor generations from 8 KB to 64 KB. However, if L1 caches demand further increase in capacity, in a few years it will not be possible to access them in just one processor cycle.Large L1 caches will make it difficult to fetch one block in a sub-nanosecond clock period.

The alternative for this problem is the development of schemes which decrease the miss rate as well as hide the access latency without resorting to capacity increase. One class of these schemes, the prefetch mechanisms, performs the fetch from L2 cache or memory to L1 cache before the conventional fetch unit requests. The prefetch schemes cannot totally eliminate the problem, but they can produce a great decrease in cache miss ratio [HSU93].

Although producing lower miss rates, the use of prefetch with some applications does not show a large increase in the overall performance, as measured in effective number of instructions completed per cycle (IPC). Every so often, the prefetch mechanisms may fetch instructions that the CPU will not request anytime later. So, the so-called pollution generated by these schemes and the increase in the bus traffic due to the extra fetches are problems usually brought by the hardware enforcement of prefetch schemes.

Moreover, aspects relative to the branch predictor accuracy are also a concern, specifically, relevant in the wrong path prefetch. In this case, the next line is always anticipated, and, when a branch occurs, the target is prefetched just after the decode stage if the branch was predicted as not taken.

In a previous study [SAN00], the wrong path prefetch did not show a better performance than the next line prefetch, which anticipate only the next block. However, the branch predictor performance may have influenced these results. Thus, the branch predictor has to be good enough to guarantee that the prefetch mechanism anticipates the correct path.

This work studies the wrong path prefetch effectiveness associated with different branch predictors, each one with a different level of accuracy. In this way, it is possible to verify how significant is the influence of the branch predictor on the wrong path prefetch mechanism. The next section presents a brief discussion regarding the mechanism studied as well as the simulation methodology used in this analysis.

II. Previous Works

The wrong path prefetch was studied in detail by [PIE95]. However, it was simulated in a single issue architecture. Machines of this type gave away to the superscalar ones, which represent the state-of-art architecture. Issues relative to cache size, pollution, and bus traffic were widely discussed in [PIE95], but just for scalar architectures models. In that case, the wrong path prefetch presented better performance than other prefetch schemes.

In [SAN00] the wrong path prefetch mechanism was implemented using a superscalar simulator. However, this mechanism did not reach the same improvement presented in the previous work [PIE95]. In superscalar architectures, the branch prediction becomes a more relevant problem, because the instruction throughput is potentially higher. Now, this problem requires more specific concerns relative to the

[1] Research supported by CNPq and CAPES Brazilian agencies.

branch predictor itself and also how it affects the prefetch.

Figure 1 shows the simulated performance of the wrong path prefetch in a 4-issue out-of-order superscalar architecture with an 8 KB 2-way associative I-cache. The horizontal axis represents the benchmarks used, while the vertical axis depicts the IPC (Instruction Per Cycle). The black bar shows the IPC reached with no prefetching. The gray bar presents the IPC achieved with another kind of prefetch mechanism, called next line prefetch. The hatching bar shows the IPC reached with the wrong path prefetch. The low cache capacity chosen will be discussed in the next sections.

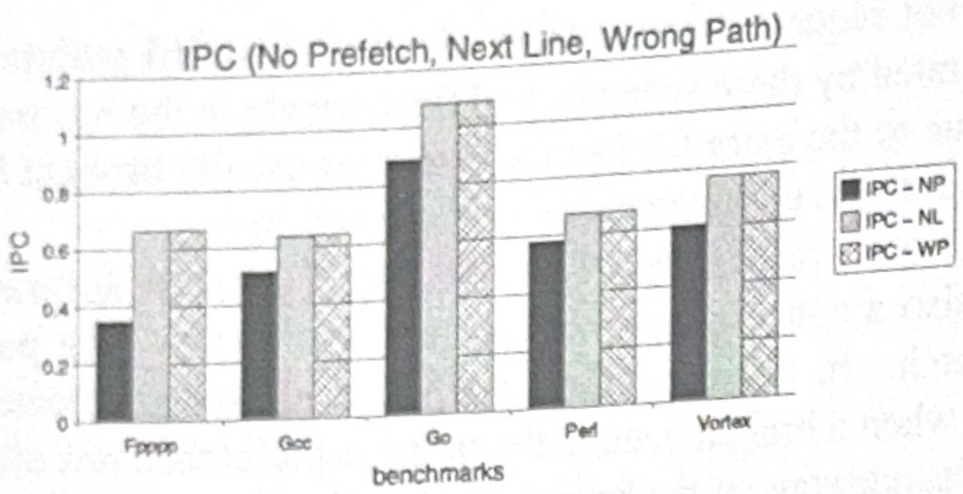


Fig. 1. IPC reached by the use of no prefetch, next line and wrong path

Despite the great potential, it is possible to see that the wrong path prefetch does not have better performance than the next line prefetch. However, in a simple single issue architecture the performance achieved by the wrong path prefetch was different [PIE95]. In that case, the wrong path prefetch reached a satisfactory improvement over the next line prefetch.

Given the results of Figure 1, this work has the objective of verifying if the cause of that result is the accuracy of the branch predictor, and also to establish the real impact that the variation of the branch predictor accuracy causes on the wrong path prefetch performance.

III. The Prefetch Mechanism and Simulation

The **Wrong Path Prefetch Scheme** anticipates the next L1 cache block and also the taken target of a branch. The target is prefetched after the decode stage, when the correct address is known. The target is prefetched only when branches are predicted to be not-taken. In this work the wrong path prefetch is analyzed in a superscalar architecture with state-of-the-art features. The branch predictor accuracy was varied in order to verify the impact caused by this mechanism in the prefetch scheme.

The simulator which implements the prefetch scheme was developed in the *sim-outorder* processor timing simulator, from the SimpleScalar Tool Set [BUR97]. Initially, four benchmarks of the SPEC95 suite were used (*fpppp, gcc, go,* and *vortex*), and for each simulation run 100 million instructions were simulated. This workload is enough to produce significant results for I-cache miss simulations [STO93].

The reference model for the experiments is an out-of-order architecture which has the main features described in Table I.

TABLE I
THE BASIC ARCHITECTURE

Argument	Value configurated
Fetch width	8 instructions
Decode width	8 instructions
Issue width	8 instructions
Number of integer FUs	8
Number of floating-point FUs	4
Misprediction penalty	7 cycles

In [SAN00], the best improvement achieved by the wrong path prefetch was obtained using 8 KB L1 caches, 2 way-associative. In that work, it was concluded that prefetch schemes should be used especially in caches with a high level of activity. Hence the L1 caches capacity was less than provided by current generation microprocessors as [HOR99, KES99, INT00]. This is done because the SPEC95 benchmarks do not achieve a sizable miss activity in caches with large capacity and the evaluation may not be fair in extremely large caches with very low miss-ratio [FER 97]. However, experiments were made with both 8 KB and 64 KB L1 cache sizes, in order to predict the prefetch impact in extremely large caches.

For the simulation model, this paper assumed the L1 caches sizes of 8 KB, distributed as follows: 128 lines, 32 bytes per line and 2 way-associative; and 64 KB caches, distributed as follows: 1024 lines, 32 bytes per line and 2 way-associative.

The unified L2 cache was fixed at 512 KB capacity. Table II summarizes the cache configurations used in the experiments.

TABLE II
THE MEMORY SYSTEM CONFIGURATIONS

Argument	Value configurated
Cache L1 size	8 KB
Cache L1 associativity	2 way-associative
Cache L1 line size	32 bytes
Cache L2 size	512 KB
Cache L2 associativity	4 way-associative
L1 hit latency	1 processor cycle
L2 hit latency	12 processor cycles
Memory latency	70 processor cycles

After the definition of the base architecture, the branch predictors were varied to use the following mechanisms: bimodal predictor, 2-level adaptive branch predictor and combined predictor (combining 2-level adaptive and bimodal). Thus, using three different kinds of predictors and varying the tables capacity associated with each one, it was possible to reach several branch prediction accuracies. Note that the variation in the accuracy was achieved using different predictors and this may cause different behavior among the experiments. This may happen because each mechanism reaches different accuracy for each kind of branch. For example: the bimodal predictor achieves its best accuracy in loop prediction, while the 2-level adaptative mechanism is better to predict conditional branches.

The next section summarizes the main results obtained by the execution-driven simulations.

IV. Results

As discussed before, several aspects regarding the impact of branch prediction accuracy on the performance of the wrong path prefetch were studied. In this way, the graphs show, in the x axis, the prediction accuracy rate. Moreover, despite the simulation of all configurations with each benchmark, just the most relevant results are presented in this paper.

The *fpppp* was the benchmark which presented the most peculiar behavior. Some of its results are presented in the paper.

Figure 2 shows the performance reached by the *fpppp* benchmark using and not using the wrong path prefetch with different branch accuracies. The horizontal axis means the accuracy reached by the branch predictors, while the vertical axis depicts the IPC achieved. The effect of no wrong path prefetch is shown. The upside of the figure presents the results for simulations using an 8 KB cache, while the downside shows the experiments performed with a 64 KB cache. These notations will be also used in the next figures of this paper.

As it is possible to see, there is a performance peak, using an 8 KB cache with 86% of prediction accuracy. In this point, the accuracy was reached using the bimodal predictor with a 64-entry table. The bimodal mechanism is well-known to predict loops correctly and *fpppp* benchmarks has, basically, a single loop with a very large basic block [FER 97]. Probably, reaching 86% of accuracy in the branch predictor is such as it causes the right anticipation, at the right time, in a cache with high level of activity. This means that in one specific point the prefetch mechanism may anticipate the right path very often, reaching the peak showed in the graph. This is not observed for a large cache. In this case, the branch predictor produces a lower impact in the prefetch scheme.

This section was divided in subsections and, in each one,

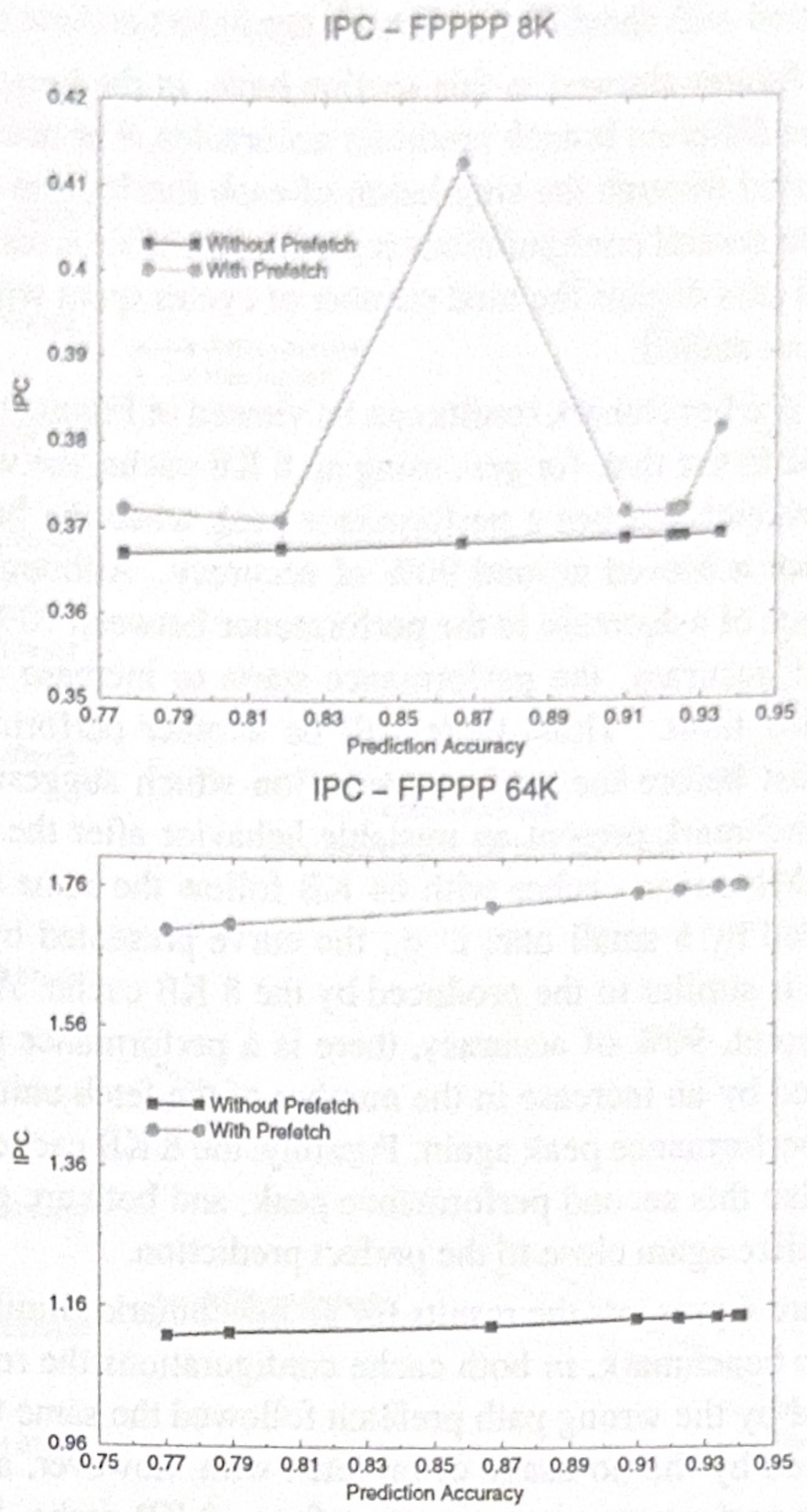


Fig. 2. IPC with and without prefetch in *fpppp*

a feature is focalized. The subsections are:

- Cycles wasted with the fetch unit blocked,
- Prefetch performance, and
- IPC.

A. Cycles wasted with the fetch unit blocked

Some key features must be verified in order to measure the performance achieved by the prefetch schemes. One of the most important characteristics is relative to the total number of cycles spent with the fetch unit blocked. These cycles, which are wasted during the stalls of the fetch unit, directly influence in the architecture performance. Then, if the miss rate is reduced but there is no gain in the block access that are being anticipated, the prefetch performance is restricted. The miss rate decrease is not enough to measure the prefetch performance and the hidden latency is a relevant feature of these schemes. Moreover, all the gains produced by the miss rate decrease are reflected in the number of cycles in which the fetch unit was blocked. This section analyzes the behavior of the fetch unit in the presence of wrong path prefetch

associated with the different branch predictors accuracies.

The figures showed in this section have, in the horizontal axis, the different branch predictor accuracies. The accuracy was varied through the simulation of each mechanism associated to several configurations regarding the table sizes. The vertical axis depicts the total number of cycles spent with the fetch unit stalled.

The *gcc* benchmark results can be viewed in Figure 3. It is possible to see that, for *gcc*, using an 8 KB cache, the wrong path prefetch reaches a performance peak when the branch predictor achieved around 90% of accuracy. Although the existence of a decrease in the performance between 90% and 92% of accuracy, the performance starts to increase again after this limit. Thus, there will be another performance peak just before the perfect prediction which suggest that *gcc* benchmark present an unstable behavior after the 90% limit. Moreover, caches with 64 KB follow the same trend produced by a small one, i. e., the curve presented by the 64 KB is similar to the produced by the 8 KB cache. At the same point, 90% of accuracy, there is a performance peak, followed by an increase in the number of the fetch unit stall and a performance peak again. Possibly, the 8 KB cache will have also this second performance peak, and both are going to stabilize again close to the perfect prediction.

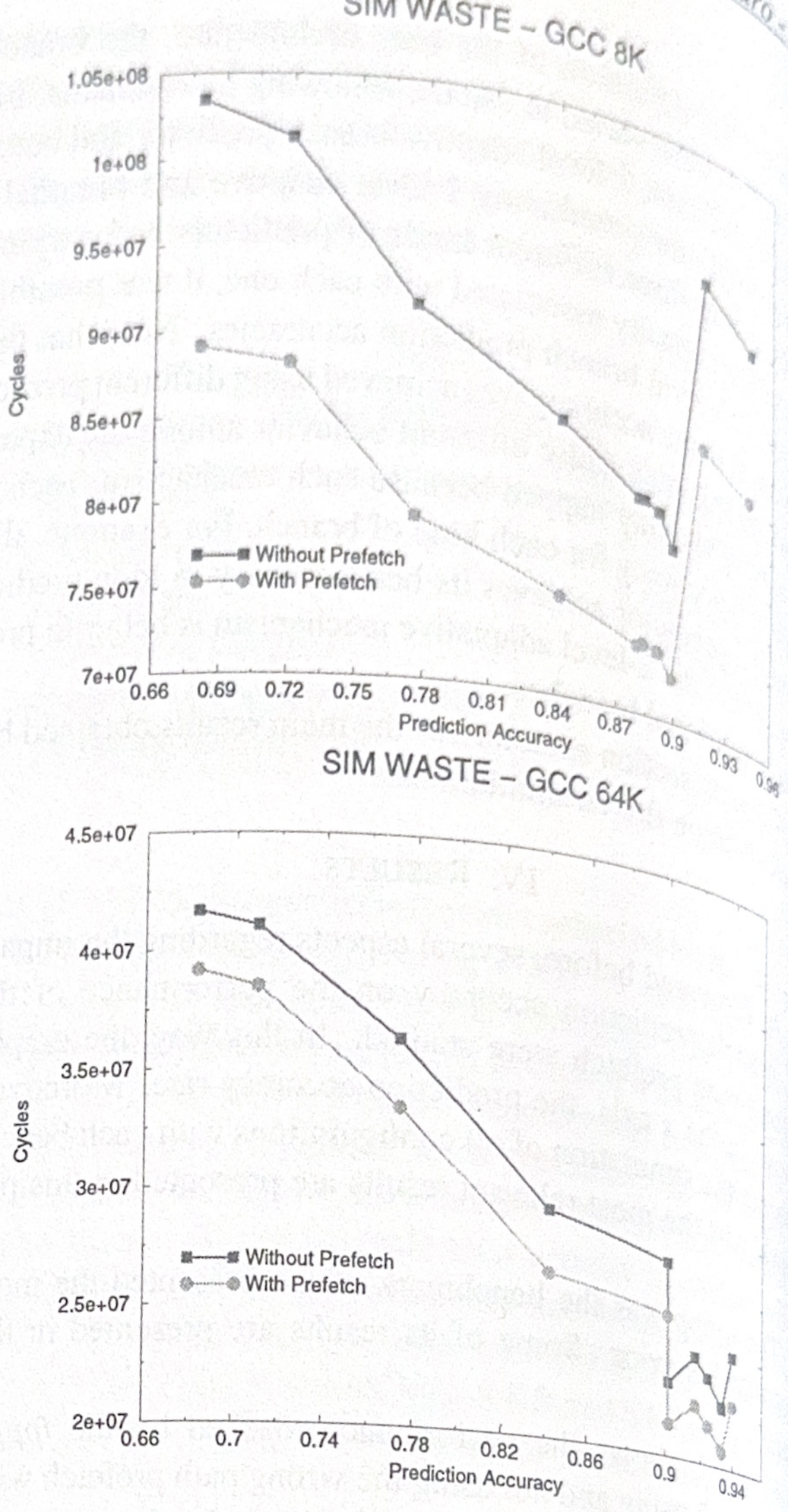


Fig. 3. Cycles wasted with and without prefetch in *gcc*

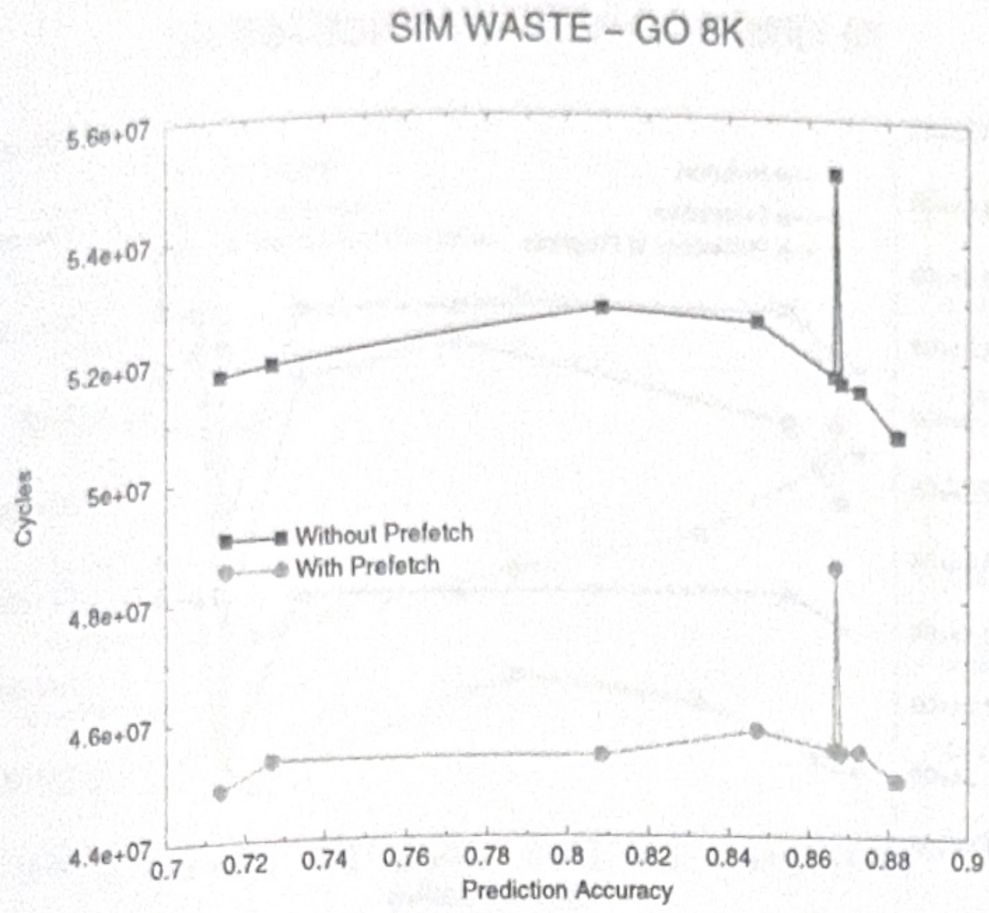
Figure 4 presents the results for *go* benchmark. Similar to the *gcc* benchmark, in both cache configurations the results reached by the wrong path prefetch followed the same trend produced by the no usage of prefetch with, however, a significant performance improvement, for an 8 KB cache. In *go* benchmark, the prediction accuracy does not present a large influence in the fetch unit stalls, in case of an 8 KB cache. In this benchmark that reaches high activity in the I-cache, prediction is not the critical problem. The opposite conclusion may be taken from the 64 KB cache. In this case, the stalls in the fetch unit decrease significantly, with the improvement in the branch prediction. Also, for 8 KB cache there is a peak in the fetch stalls when the branch prediction accuracy achieve 87%. Possibly, this peak is due to the behavior of the own benchmark, because this is happening in the no usage of prefetch as well.

The *vortex* benchmark, presented in Figure 5, is the most regular benchmark. It is possible to see that the number of stalls in the fetch unit decreases linearly for both cache configurations, through the variation of the branch prediction. It means the wrong path prefetch used associated to this benchmark, mainly, depends on the branch prediction accuracy.

It is possible to verify that in *go* and *vortex* benchmarks wrong path prefetch has less accented curves than in the architecture without prefetch. Also, all benchmarks presented at least one peak of fetch unit stalls during the increase of the branch prediction accuracy. The peaks, normally, happen after a decrease in the number of stalls. This behavior is quite contradictory, because with the improvement of the branch predictor accuracy the number of cycles in which the fetch unit is stalled should be smaller and those peaks should not exist. However, the fetch unit stalls not just because of mispredicted branches, but it may stalls due to I-caches misses, branch occurrence and full pipeline as well. Thus, the peaks may be provoked by one of those another reasons.

B. *Prefetch performance*

In this subsection, the number of prefetches executed, the number of instructions fetched while its prefetch was being performed and the pollution generated in the L1 I-Cache are analyzed. The goal of this section is to understand the relation among branch prediction accuracy, number of prefetches performed and the cache pollution. As pollution values are smaller than number of prefetches and number of prefetches in progress, the graphs in this subsection were plotted using two *y* axis, at left showing values for the number of



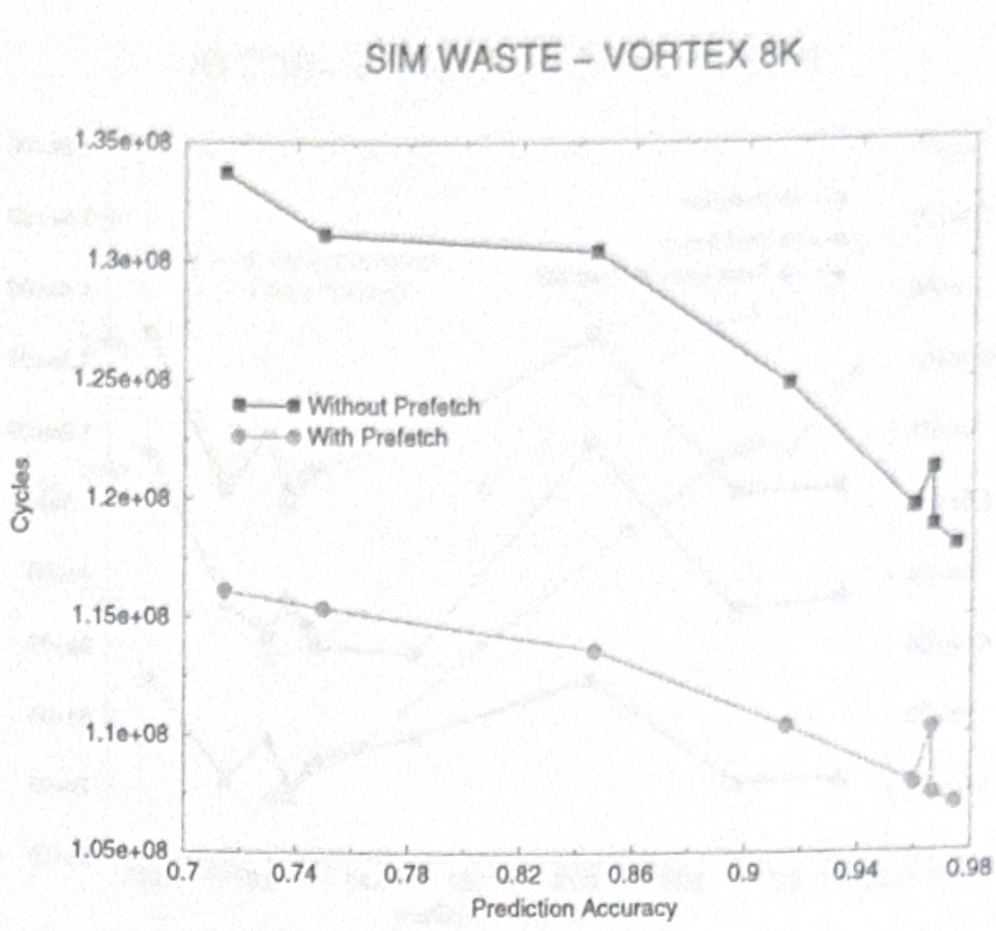


Fig. 4. Cycles wasted with and without prefetch in *go*

Fig. 5. Cycles wasted with and without prefetch in *vortex*

prefetches and prefetches in progress, while in the right side the values are for the pollution.

When the prediction accuracy increases, in the general case, the number of prefetches executed increases. This occurs because the fetch unit spend more time busy due to the decrease in the mispredicted branches and, as the fetch unit is fetching more frequently, the number of instructions anticipated increases as well.

The behavior of the *gcc* and *go* benchmarks can be observed in the Figures 6 and 6, respectively. It is possible to see that for the two benchmarks there are some peaks in the number of instructions prefetched, the number of prefetch in progress and also in the pollution. In general, the three prefetch features vary together, i. e., when there is a peak in the number of prefetches, there is also a peak in the pollution and in the number of instructions fetched before the end of their prefetch. The results show that the branch predictors, as well as the accuracy reached by them, really affect the prefetch scheme. However, the affect is given globally, not just for a specific feature. Thus, a new study should focus each mechanism in each accuracy. This study can support the choice for use or not use of the wrong path prefetch with some class of programs and branch predictors.

The *vortex* benchmark is shown in Figure 8. The results reached by *vortex* are the most regular in this set of benchmarks. However, different from the other benchmarks, the pollution does not follow the same trend of the other features. The pollution in *vortex* benchmark is always increasing according the improvement in the prediction accuracy, for the both cache sizes. Hence, in this particular benchmark, the performance of the branch predictor exerts a bad influence in the wrong path prefetch mechanism due to this increase in the pollution.

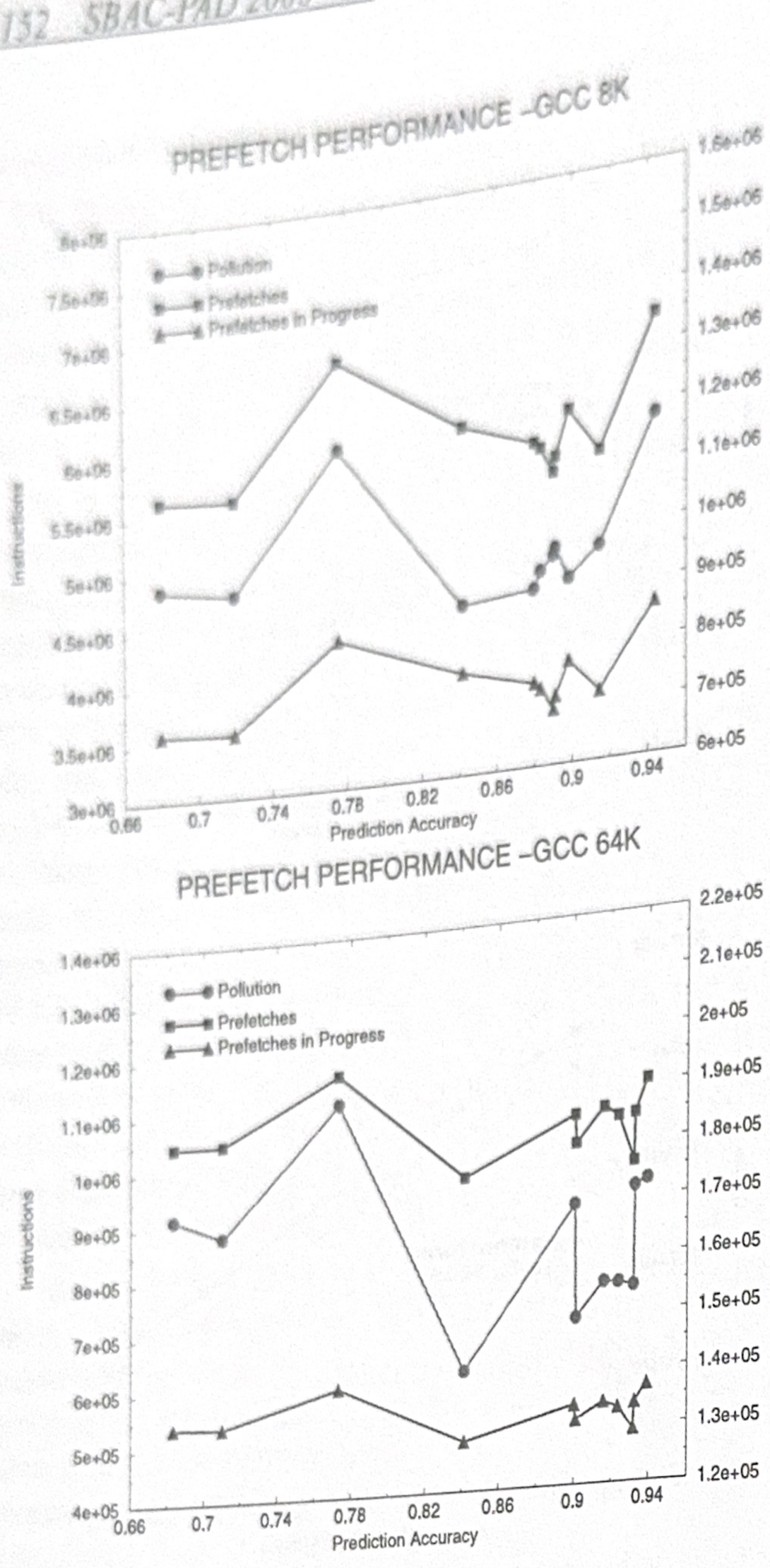


Fig. 6. Prefetch performance in *gcc*

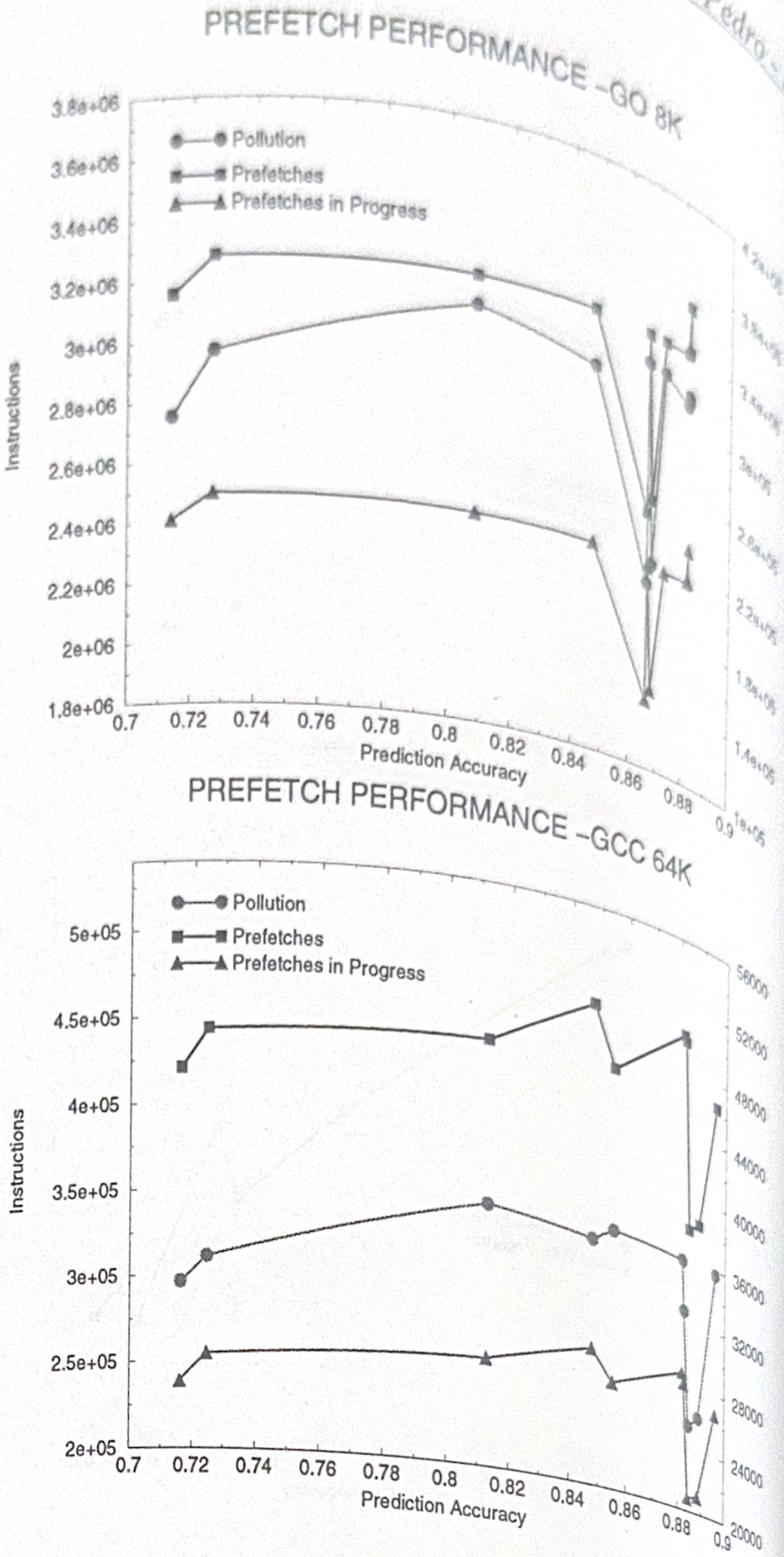


Fig. 7. Prefetch performance in *go*

C. Instructions per cycle

The most important parameter to measure performance in superscalar microprocessors is the number of Instructions Per Cycle (IPC) achieved. This section shows the results obtained with the utilization of the wrong path prefetch against the several branch prediction configurations. However, IPC does not depend exclusively on the I-cache, but it depends on the well balanced performance of the overall architecture. With the I-cache misses reduction, another problems which affect the IPC may arise, such as resource conflicts and D-cache miss rate increase. Then, some of the results presented in this section may be well affected by the one of these effects.

The results regarding to the *gcc* benchmark are being shown in Figure 9. The *gcc* benchmark depicts a regular behavior for IPC. It is possible to see that the highest IPC reached is in the highest branch predictor accuracy as well. Using the 8 KB caches, the performance increases almost linearly until the branch predictors accuracy reach 90%. Between 90% and 92% the performance is stabilized and it is followed by another significant performance increase between 92% and 94% of accuracy. This means that, in *gcc* case, is still necessary increase the performance of the branch prediction. In just two percent interval of accuracy, the IPC achieved an accented increase. In simulations with 64 KB caches, the behavior is regular as well.

Figures 10 and 11 present the IPC reached by *go* and *vortex*, respectively. These two benchmarks achieved similar results. After a stabilized behavior in the very beginning of the experiments, the performance have grown until the best accuracy reached by the branch prediction mechanisms. However, in *vortex* benchmark, it is possible to observe an accented behavior between 74 and 84% of prediction accuracy. However, like *go* benchmark the performance is still increasing even achieving 97% of branch predictor accuracy. In the two cases, the behavior achieved by the IPC using 64 KB

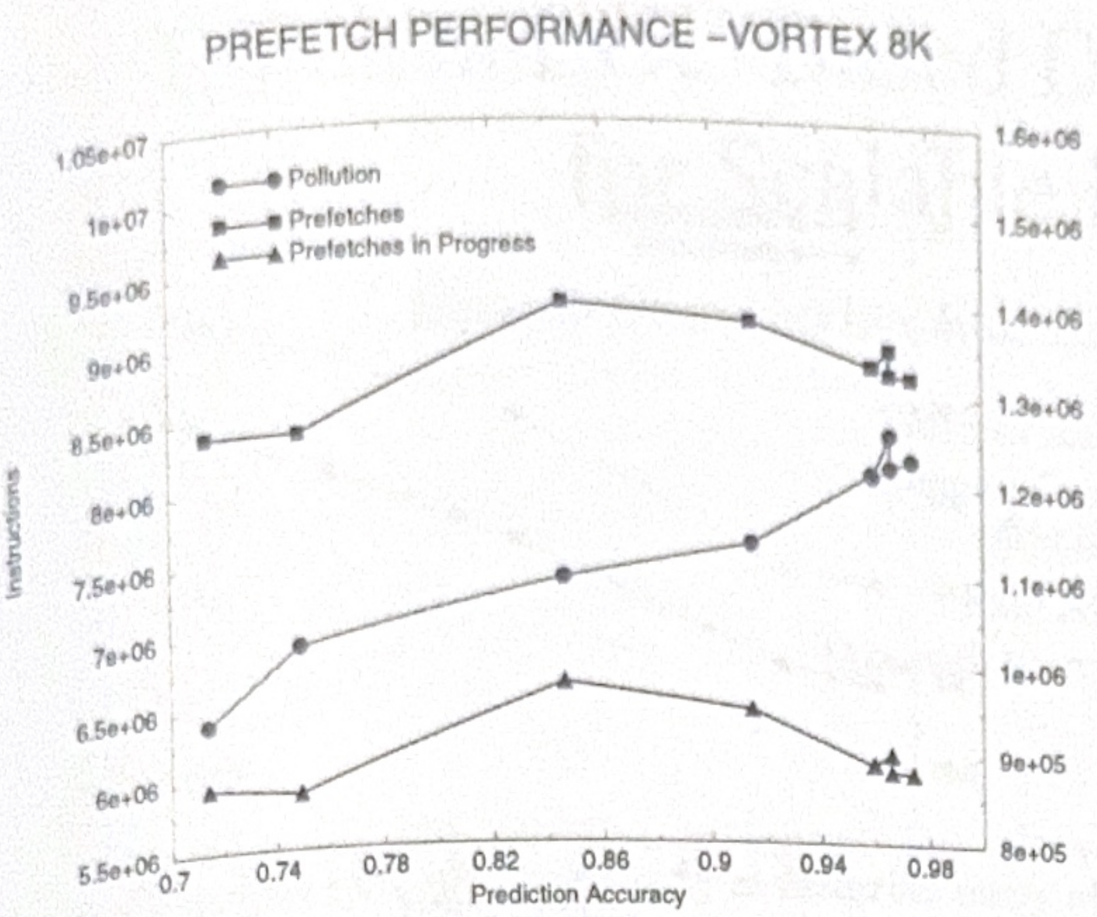


Fig. 8. Prefetch performance in *vortex*

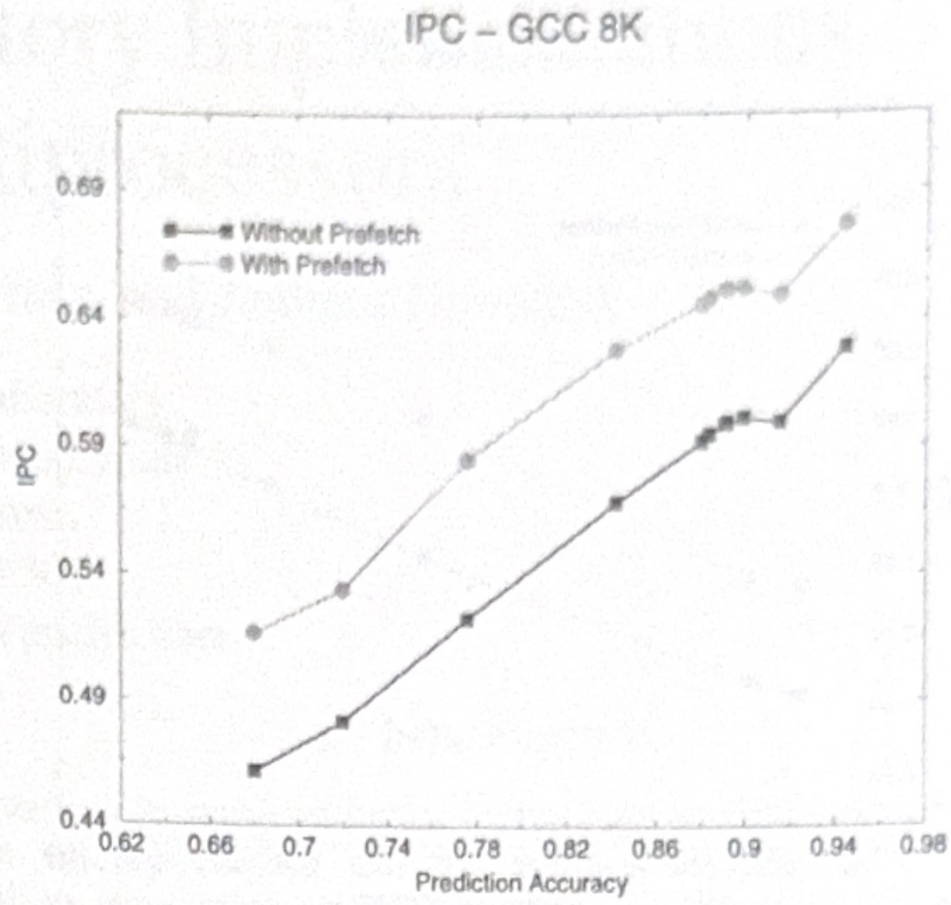


Fig. 9. IPC with and without prefetch in *gcc*

caches are very regular.

V. Conclusions

In this paper, the role of prediction accuracy in the performance of the wrong path prefetch mechanism was studied.

The results have shown that, besides the importance of a high prediction accuracy for the superscalar architectures performance, using the wrong path prefetch may achieve the best performance in a different point from the best prediction accuracy.

Some peculiar results were observed, mainly, in the *fpppp* benchmark. This benchmark presented its best performance with a very simple branch predictor (64-entry bimodal mechanism) and even using more sophisticated predictors, the performance decreased. This may happen for some reasons that acted together and caused the strange result. These reasons are relative to the structure of the benchmark, which is basically a single loop; the structure of the branch prediction used, which reaches its best performances predicting loops and, finally, the prefetch itself, which may be anticipating the right block at the right time.

The *vortex* benchmark had also an interesting behavior. For this benchmark, the pollution generated by the wrong path prefetch increases with the improvement of the prediction accuracy. This means that for this specific benchmark, it is not necessary to invest in expensive branch predictors, if the wrong path prefetch is going to be used.

The *gcc* and *go* benchmarks presented some peaks and drops in their statistics, such as the wasted cycles by the fetch unit and also in the overall performance of the wrong path prefetch. Nevertheless, the global performance was not affected.

Excepting *fpppp*, all benchmarks presented improvement in the IPC, increasing the branch predictor accuracy. In a few points in *gcc* there are some stabilizations followed by other peaks. Even *vortex* benchmark, which had increase in

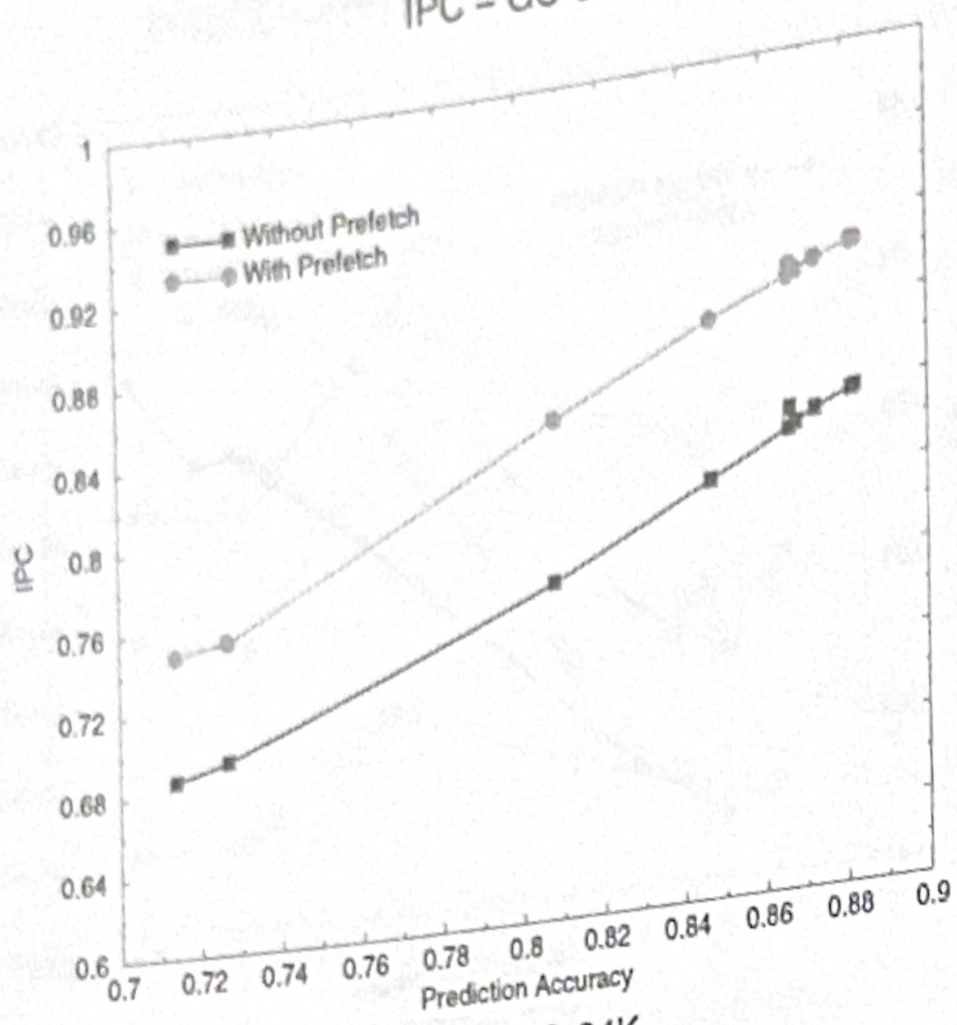


Fig. 10. IPC with and without prefetch in *go*

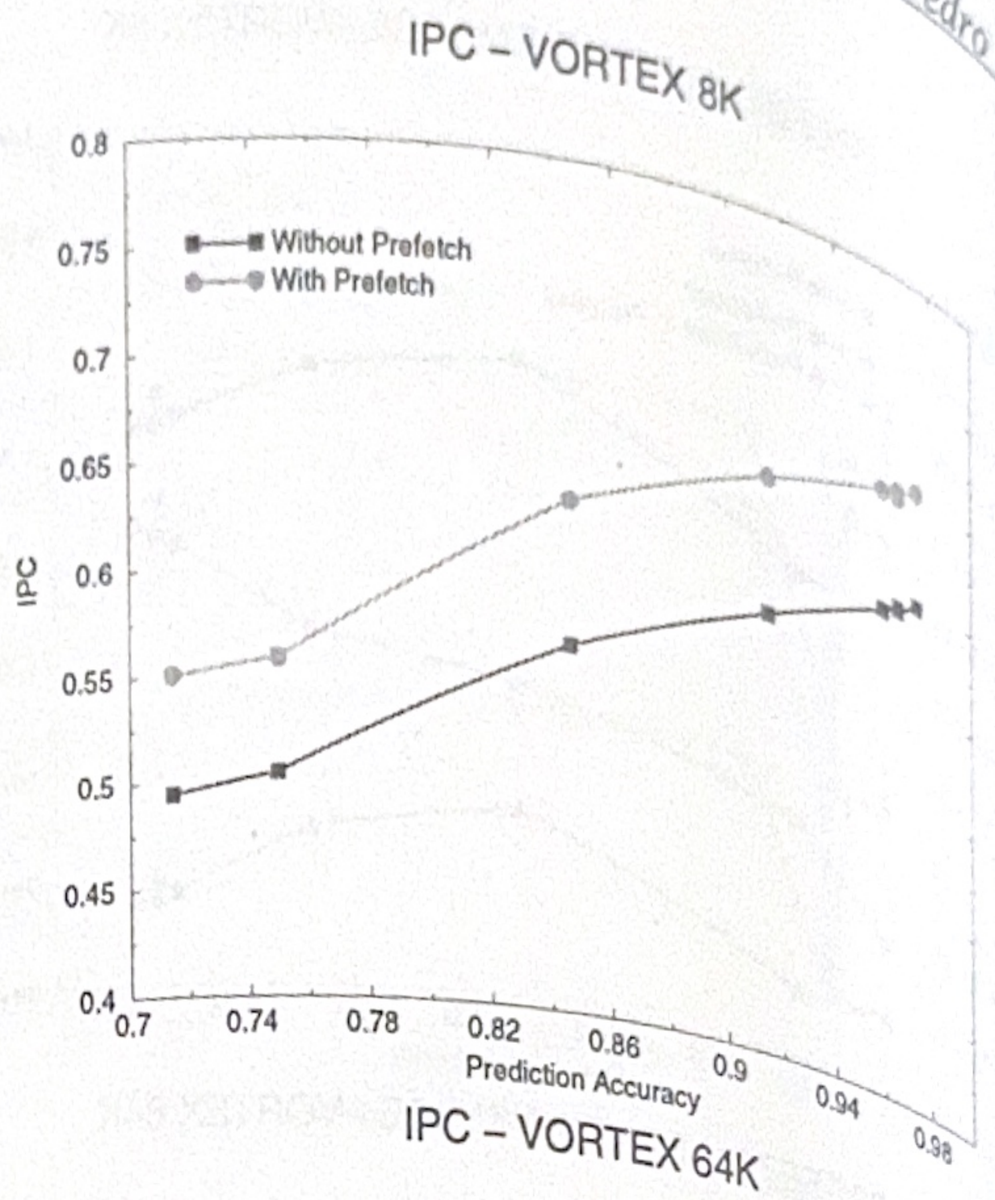


Fig. 11. IPC with and without prefetch in *vortex*

the pollution, has shown improvement in the global performance. However, for this case, the result would be better if the pollution has not increased.

As future works, a new study should be done to verify each branch mechanism in separate. In this way, will be possible to isolate the anomalous points and identify clearly the reasons for that.

REFERENCES

[BUR97] BURGER, Doug; AUSTIN, Todd. **The SimpleScalar Tool Set, version 2.0**. Madison: Computer Sciences Department/University of Wisconsin-Madison. (Technical Report).

[FER 97] FERNÁNDEZ, Augustin. **Un análisis cuantitativo del Spec95**. Barcelona: Universitat Politècnica de Catlunya. (Technical Report).

[HOR99] HOREL, Tim; LAUTHERBACH, Gary. UltraSparc-III: designing third-generation 64-bit performance. **IEEE Micro**, v.19, n.13, p.73–85, May/June 1999.

[HSU93] HSU, Peter. Design of the TFP microprocessor. **IEEE Micro**, Los Alamitos, v.14, n.2, p. 23–33, March/April 1994.

[INT00] INTEL CORPORATION. **Pentium III processor for the SC242 at 450 MHz to 866 MHz**. Avaiable at http://developer.intel.com/design/pentiumiii/datashts/244452.htm (March 2000).

[JOH91] JOHNSON, Mike. **Superscalar microprocessor design**. Englewood Cliffs: Prentice Hall, 1991.

[KES99] KESSLER, Richard E. The Alpha 21264 microprocessor. **IEEE Micro**, Los Alamitos, v.19, n.2, March/April 1999.

[PIE95] PIERCE, James E. **Cache behaviour in the presence of speculative execution – The Benefits of Misprediction**. Ann Arbor: Department of Computer Science and Engineering/University of Michigan, 1995. (Ph.D. Thesis).

[SAN00] SANTOS, Tatiana G. S. dos. **Analyzing the prefetch mechanisms on the RISC superscalar microprocessors memory hierarchy**. Porto Alegre: Institute of Computer Science/Federal University of Rio Grande do Sul, 2000. (Ms.c. Thesis). (In Portuguese).

[STO93] STONE, Harold. **High performance computing**. Englewood Cliffs: Prentince Hall, 1993.