BRANCH PREDICTION: A CRITICISM AND A NOVEL SCHEME

Lucian N. VINTAN, **Adrian FLOREA**
"L. Blaga" University of Sibiu, Department of Computer Science, Str. E. Cioran, No. 4, Sibiu-2400, ROMANIA,
E-mail: vintan@vectra.sibiu.ro, aflorea@vectra.sibiu.ro

Abstract: The main aim of this work is to propose a new Two Level Adaptive Branch Prediction scheme, based on a new additional correlation information. We prove that branch's history is insufficient for a good branch correlation and as a consequence, for high prediction accuracy. Also, we investigate comparatively, through a trace driven simulation method, a classical branch prediction scheme called GAp, firstly proposed by Pan et al in 1992 and the proposed new scheme, both of them integrated into a MII (Multiple Instruction Issue) environment. We point out that our new proposed branch prediction scheme performs better than a classical GAp scheme, at the same level of hardware complexity.

Key Words: MII Architectures, Branch Prediction, Two Level Adaptive Branch Prediction, Trace Driven Simulation.

1. INTRODUCTION

As the average instruction issue rate and depth of the pipeline in multiple instruction issue (MII) processors increase, the necessity of an efficient hardware branch predictor becomes more and more essential. Very high prediction accuracies are necessary, because taking into account the MII processors characteristics as pipeline depth or issue rates, even a prediction miss rate of a few percent involves a substantial performance loss.

The main aim of this work is to propose a new Two Level Adaptive Branch Prediction scheme, based on a new additional prediction information, available during the instruction fetch stage. We prove that branch's history is insufficient for a good correlation and therefore for a high prediction accuracy. Also, we investigate comparatively, through a trace driven simulation method, a classical branch prediction scheme called GAp, firstly proposed by Pan et al [Pan92] and the proposed new scheme, both of them integrated into a MII environment. We used the traces obtained based on the eight C Stanford integer benchmarks. These benchmarks were compiled through the HSA (Hatfield Superscalar Architecture) compiler, developed at the University of Hertfordshire, Research Group of Computer Architecture, UK. Further, the traces were obtained using the HSA simulator, developed at the same university [Ste96]. Based on these tools, we have developed an original simulator to investigate some branch prediction schemes.

The first efficient approach in hardware branch prediction consists in Branch Target Buffer (BTB) structures [Per93]. BTB is a small (associative) memory, integrated on chip, that retains the addresses of recently executed branches, their targets and optionally other information (e.g. target opcode). Due to some intrinsic limitations, BTB's accuracies are limited on some benchmarks having unpropitious characteristics (e.g. correlated branches).

In order to improve BTB's efficiency, Yeh and Patt (1992) and independently Pan et al (1992), generalised it through a new approach called Two Level Adaptive Branch Prediction. According to [Yeh92], the Two Level Adaptive Branch Prediction uses two distinct levels of branch history information to make predictions. The first level consists in the History Register (HR), that contains the last k branches encountered (taken/ not taken) or the last k occurrences of the same branch instruction. The second level consists in the branch behaviour of the last occurrences of the specific pattern of these branches. A Pattern History Table (PHT) that contains essentially the branch prediction automaton (usually 2 - bit saturating counters) implements it.

HR shifts left with a binary position when updated according to the actual branch behaviour (taken=1/ not taken=0). There is a corresponding entry in the PHT for each of the 2^k HR's patterns.

The prediction of the branch (P) is a function (f) of the actual prediction automaton state S_t.

$$\mathbf{P = f(S_t)} \qquad \mathbf{(1)}$$

After the branch is resolved, HR is shifted left and the prediction automaton state becomes S_{t+1}.

$$\mathbf{S_{t+1} = g(\ S_t\ , B_t)} \qquad \mathbf{(2)}$$

where g( ) represents the automaton's transition function and B_t represents the behaviour of the last branch encountered (taken/ not taken).

These Two Level Adaptive Branch Prediction schemes are very effective in predicting correlated branches with high accuracy. It's well known that the average prediction rate for these schemes, neglecting the bad target addresses, measured on nine of the ten Spec benchmarks, is about 97%, while BTB schemes achieved at most 94% on the same benchmarks [Yeh92]. An excellent generalisation of these Two Level Adaptive Branch Prediction schemes, based on the universal compression/prediction algorithm called "prediction by partial matching" (PPM), is given in [Mud96].

2. AN IMPROVED BRANCH PREDICTION PRINCIPLE

In our opinion, a common criticism for all the present Two Level Adaptive Branch Prediction schemes consists in the fact that they used an **insufficient global correlation information** (HRg). So, as our statistics clearly point out, for the same static branch and in the same HRg (containing the last k branches encountered as taken or not taken) and HRl (containing the last l occurrences of the same branch) context pattern, it's possible to find different ("alternative") branch's behaviours (for example about 50% taken and respectively 50% not taken), making that branch difficult to predict even through adaptive schemes. Otherwise, as it can be seen in [Sec95], "the role of adaptivity at the second level of two level branch prediction schemes is more limited than has been thought". In other words, it's difficult correctly to predict a branch that has randomly behaviour in the same prediction context (HRg, HRl). If each bit belonging to HRg (on k bits) will be associated during the prediction process with its corresponding PC, the correlation information will be more complete and therefore the prediction accuracy would be better. In this way it will be not only known if the previous k encountered branches were taken or not (through HRg content), but it will be exactly known which branches they were, through their labels (PC_1 PC_2 ... PC_k). Therefore, instead of using only HRg, it could be used a more complex and complete prediction context, consisting of HRg together with its corresponding labels of branches with better performances. For each different pattern of this context, we'll have a corresponding prediction automaton. Based on this principle, we propose a new prediction scheme like that presented in figure 2.

As it can be observed in figure 2, we implement the Prediction Table (PT) as fully associative. Also we implemented a MPP (Minimum Performance Potential) replacing algorithm, similar with that presented in [Per93]. This algorithm replaces that entry having the minimum product of the probability of reference (LRU bits) and the probability of branch taken (derived from prediction automata's two bits - HRl). It's well known that discarding a branch that is not likely to be taken has little penalty. The prediction automaton implemented is the optimal known: a two bit saturating counter [Nai95]. As we already mentioned, PC_i represents the PC associated with the i-th branch belonging to HRg register.

Figure 1 presents a classical full associative GAp branch prediction scheme [Pan92], in order to be compared with our new modified GAp scheme (MGAp) presented in figure 2 and having the same characteristics. In GAp scheme the PC and the HRg are concatenated before being used to index into the Prediction Table. Therefore, GAp used a separate PT for each branch. Certainly, the comparisons between GAp and MGAp will be made, considering equivalent schemes from the complexity/cost point of view.

3. SIMULATION WORK

3.1. BENCHMARKS PROGRAMS

The simulation work has been centred on the Stanford integer benchmark suite, a collection of eight C programs designed by Professor John Hennessy (Stanford University), to be representative of non - numeric code while at the same time being compact. The benchmarks are computationally intensive with higher dynamic instruction counts. All these benchmarks were compiled by the HSA gnu C compiler, which targets the HSA instruction set. A dedicated HSA simulator [Ste96] that generates the corresponding traces simulated the resulted HSA object code.

The average instruction number is about 273.000 and the average percentage of total instructions that are branches is about 18%, with about 76% of them being taken. Derived from HSA traces, special traces were obtained, containing exclusively all the processed branches. Each branch belonging to these modified HSA traces is stored in the following format: branch's type the PC of the branch and it's target address. Some of these benchmarks are well known as very difficult to be predicted. For example, as Mudge et al proved very clearly [Mud96], 75% accuracy could be an ultimate limit on *"quick-sort"* benchmark.

Following our aims, we developed a dedicated trace driven simulator that uses the above mentioned traces. The most important input parameters for this simulator are the number of HRg bits (k) and HRl bits (l). As outputs, the simulator generates prediction accuracy, number of bad target addresses and other useful statistics (see table 1).

Taking into account Stanford benchmark's characteristics together with the present technological on-chip integration level, during the simulation PHT tables up to 256 entries were considered.

3.2. RESULTS

For both schemes we considered PT capacity of 100 entries. Table 1 presents for a MGAp scheme, some branch prediction statistics, considering different lengths of HRg (on k bits) concatenated with its corresponding branches (PC_1 PC_2 ... PC_k). As it can be observed in table 1, these lengths are considered successively of 9, 18, 27, 36 and 45 bits, corresponding respectively to HRg on 1, 2, 3, 4 and 5 bits. Generally speaking, considering HRg register on k bits and PC's length on 8 bits (sufficient for HSA Stanford integer benchmarks), the corresponding PT's tag length is n=9k (see figure 2). Last column in table 1 represents the number of replacing (NR), countered after PT's filling. From one point of view, NR indicator represents a good metric of branch interferences to PT. Branches with

associated "rich" contexts (great k value), involve rapidly filling of the PT table and thus, a large number of replacing with a bad influence on prediction's accuracy.

Table 2 is coming from table 1 and points out for each of the eight benchmarks the best MGAp scheme (the optimal n, see figure 2). Finally, table 3 shows the same statistics as the previous tables for a GAp prediction scheme, considering HRg's length of 1 bit respectively 9 bits.

Based on table 2, the average prediction accuracy (Ap) for the best MGAp scheme is 87.12% (neglecting bad targets it growths to about 90%). Interesting, with three exceptions, the best schemes are obtained for a one bit HRg register (k=1). As an exception, for *"permute"* benchmark the best MGAp scheme involves a HRg on 5 bits. The explanation of this behaviour could involve two antagonist aspects. Firstly, a "rich" branch context (great k value) could involve better performances because essentially each branch context has its own prediction automata stored in PT. Secondly, as we already mentioned, "rich" contexts could determine NR's growth and therefore poor performances due to interferences. The best trade-off between these two aspects - great k values and thus "rich" contexts but few different context patterns (from 2^{9k} possible) and therefore few replacing process - offers the optimal performance.

According to this, from table 1 (MGAp scheme) results at average NR(k=1)=1.62 evacuations and NR(k=5)=3646 evacuations. Analogously, from table 3 (GAp scheme) results at average NR(k=1)=0 and NR(k=9)=3575. From table 1 (MGAp scheme) results at average Ap(k=1)=85.19%, better than the corresponding GAp scheme involving at average Ap(k=9)=81.43% (see table 3). Analogously, further simulations show us that for a MGAp scheme, Ap(k=2)=86.03% and Ap(k=3)=85.39%, while for a simple equivalent GAp scheme we obtained at average Ap(k=18)=74.43% and Ap(k=27)=66.15%. These last comparisons show obviously that at the same structural complexity, a MGAp scheme performs better than a classical GAp scheme. (Surprisingly, a GAp scheme having k=1 obtains at average an Ap=83.74%, better than a GAp having k=9!)

Figures 3 to 9 present comparatively, prediction accuracies for a MGAp scheme respectively a GAp scheme, only for those branches belonging to HSA Stanford benchmarks, considered by us being "difficult to be predicted". We considered a branch difficult predictable, if it is taken for example up to about 70%, in the same (HRg, HRl) associated context, indifferent of HRg's length. As an example, in *"bubble-sort"* trace, from 41.200 processed branches, 39.800 are difficult to be predicted from this point of view. *"Matrix"* benchmark doesn't contain "difficult" branches. Prediction accuracies are presented in figures 3 to 9 comparatively, for both MGAp respectively GAp scheme, for different k values. For example if k=3, that means a MGAp scheme having 3 bits of global history concatenated with the associated PCs, each PC on 8 bit length, and respectively a GAp scheme having 24 bits global history; therefore equivalent schemes from the hardware complexity point of view. The average prediction accuracy (Ap) for all the difficult to predict N branches belonging to a trace is given by the following formula:

$$Ap = \frac{\sum_{i=1}^{N} Ki * Ap(Bi)}{\sum_{i=1}^{N} Ki} \qquad \textbf{(3)}$$

Ap(Bi) represents the prediction accuracy for the branch Bi and Ki represents the number of instances during the trace, for a certain branch (Bi). As it can be observed, MGAp scheme performs significantly better than the corresponding GAp for these special branches. That means, our new MGAp scheme proposed in paragraph 2, involves better results especially for those branches difficult to be predicted. In other words, all these results point out the fact that the associated PC that each HRg bit, could be an efficient prediction information.

As another example, table 4 presents only a branch (labelled 35) belonging to *"Permute"* HSA benchmark, having a strange behaviour for HRg=xxxxx101 and HRl=10, indifferent if HRg is on 3 bits or on 8 bits length. Correspondingly, figure 10 presents comparatively prediction accuracies only for this branch, considering different k length, for both prediction schemes.

4. CONCLUSIONS AND FURTHER WORK

We proposed a new branch prediction approach based on an additional prediction information available to be used during the instruction fetch stage in the pipeline. This information consists of history register together with its corresponding PCs of branches. Using this new information together with the global history register, the current branch's context becomes more precisely and therefore its prediction accuracy could be better. Our first simulation results are encouraging, we show that a scheme based on this principle performs better than a classical GAp scheme, at the same level of complexity.

As it can be observed, the obtained prediction accuracies are smaller than those reported by other researchers that used in simulation the Spec benchmarks. Because of the additional warm up time our models are likely to perform less successfully with relatively small benchmarks like Stanford. We would expect to show some substantial improvement in prediction accuracy with longer benchmarks like Spec 95 for example. Anyway, the obtained prediction accuracies are perfect comparable with those obtained by other researchers that used Stanford benchmarks [Ega98].

As a next step, it will be interesting and useful to analyse a MGAp scheme that compress through hashing

some of the used prediction information, in order to minimise scheme's costs and complexity.

ACKNOWLEDGMENTS

This work was supported in part by the Romanian National Agency for Science, Technology and Innovation (ANSTI) grants No. 4086/1998, 1999 and respectively by the Romanian National Council of Academic Research grants CNCSU No. 391/1998 and CNCSU 489/1999. Also our gratitude to Professor Gordon B. Steven from the University of Hertfordshire, UK, for providing HSA Stanford traces and for his encouragement related to our Instruction Level Parallelism research.

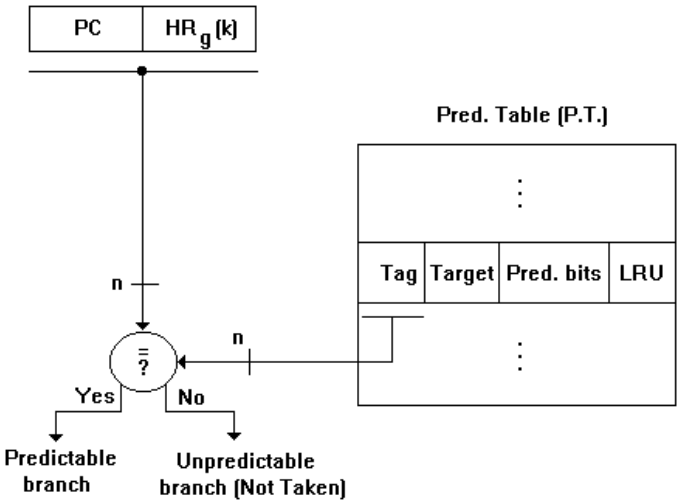

Figure 1. A full associative GAp scheme

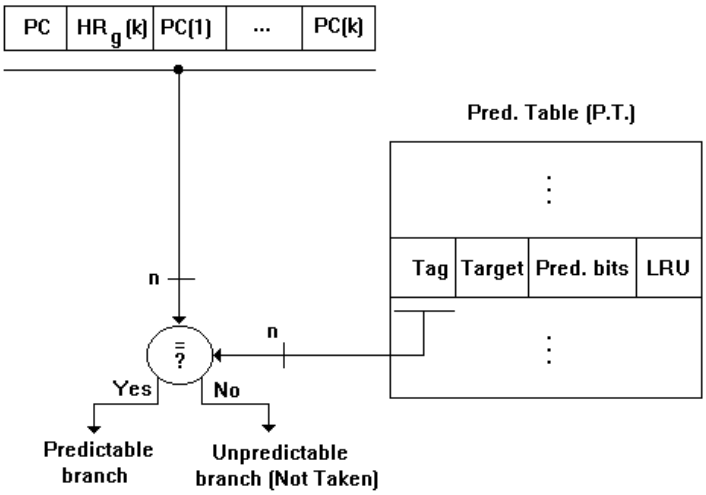

Figure 2. A full associative modified GAp (MGAp) scheme

TABLE 1. History with associated PCs (MGAp scheme - 100 entries)

Bench	HRg	Br.no.	Pr. Accuracy	Incorrect pr.	Bad target	NT branches	No repl.
fsort.tra	9	12601	9441(74.92%)	2929(23.24%)	231(1.83%)	4414(35.03%)	0
fsort.tra	18	12601	9031(71.67%)	3325(26.39%)	245(1.94%)	4414(35.03%)	862
fsort.tra	27	12601	8590(68.17%)	3849(30.55%)	162(1.29%)	4414(35.03%)	2493
fsort.tra	36	12601	8234(65.34%)	4277(33.94%)	90(0.71%)	4414(35.03%)	3380
fsort.tra	45	12601	7935(62.97%)	4619(36.66%)	47(0.37%)	4414(35.03%)	3989
fbubble.tra	9	41216	35174(85.34%)	6042(14.66%)	0(0.00%)	10140(24.60%)	0
fbubble.tra	18	41216	35109(85.18%)	6107(14.82%)	0(0.00%)	10140(24.60%)	0
fbubble.tra	27	41216	35107(85.16%)	6116(14.84%)	0(0.00%)	10140(24.60%)	0
fbubble.tra	36	41216	34520(83.75%)	6696(16.25%)	0(0.00%)	10140(24.60%)	0
fbubble.tra	45	41216	34478(83.65%)	6738(16.35%)	0(0.00%)	10140(24.60%)	44

fmatrix.tra	9	21341	20607(96.56%)	733(3.43%)	1(0.00%)	703(3.29%)	0
fmatrix.tra	18	21341	20601(96.53%)	739(3.46%)	1(0.00%)	703(3.29%)	0
fmatrix.tra	27	21341	20595(96.50%)	745(3.49%)	1(0.00%)	703(3.29%)	0
fmatrix.tra	36	21341	20589(96.48%)	751(3.52%)	1(0.00%)	703(3.29%)	0
fmatrix.tra	45	21341	20583(96.45%)	757(3.55%)	1(0.00%)	703(3.29%)	0
fperm.tra	9	54819	42828(78.13%)	5272(9.62%)	6719(2.26%)	10862(9.81%)	0
fperm.tra	8	54819	47857(87.30%)	5282(9.64%)	1680(3.06%)	10862(19.81%)	0
fperm.tra	27	54819	48010(87.58%)	5129(9.36%)	1680(3.06%)	10862(19.81%)	0
fperm.tra	36	54819	49303(89.94%)	3417(6.23%)	2099(3.83%)	10862(19.81%)	0
fperm.tra	45	54819	50321(91.79%)	2316(4.22%)	2182(3.98%)	10862(9.81%)	0
ftower.tra	9	37930	33043(87.12%)	1305(3.44%)	3582(9.44%)	9153(24.13%)	0
ftower.tra	18	37930	32778(86.42%)	1315(3.47%)	3837(10.12%)	9153(24.13%)	0
ftower.tra	27	37930	32701(86.21%)	1265(3.34%)	3964(10.45%)	9153(24.13%)	0
ftower.tra	36	37930	32622(86.01%)	1280(3.37%)	4028(10.62%)	9153(24.13%)	0
ftower.tra	45	37930	32694(86.20%)	1302(3.43%)	3934(10.37%)	9153(24.13%)	44
fqueens.tra	9	38462	30511(79.33%)	7932(20.62%)	19(0.05%)	19181(49.87%)	0
fqueens.tra	18	38462	31074(80.79%)	7369(19.16%)	19(0.05%)	19181(49.87%)	0
fqueens.tra	27	38462	31075(80.79%)	7368(19.16%)	19 (0.05%)	19181(49.87%)	158
fqueens.tra	36	38462	29061(75.56%)	9391(24.42%)	10(0.03%)	19181(49.87%)	4612
fqueens.tra	45	38462	26521(68.95%)	11931(31.02%)	10(0.03%)	19181(49.87%)	8709
ftree.tra	9	32887	28122(85.51%)	3510(10.67%)	1255(3.82%)	8721(26.52%)	0
ftree.tra	18	32887	28188(85.71%)	3477(10.57%)	1222(3.72%)	8721(26.52%)	0
ftree.tra	27	32887	28216(85.80%)	3547(10.79%)	1124(3.42%)	8721(26.52%)	37
ftree.tra	36	32887	27922(84.90%)	3876(11.79%)	1089(3.31%)	8721(26.52%)	548
ftree.tra	45	32887	26016(79.11%)	5848(17.78%)	1023(3.11%)	8721(26.52%)	3309
fpuzzle.tra	9	204527	193579(94.65%)	10946(5.35%)	2(0.00%)	18576(9.08%)	13
fpuzzle.tra	18	204527	193183(94.45%)	11342(5.55%)	2(0.00%)	18576(9.08%)	3322
fpuzzle.tra	27	204527	190037(92.92%)	14488(7.08%)	2(0.00%)	18576(9.08%)	7349
fpuzzle.tra	36	204527	187417(91.63%)	17109(8.37%)	1(0.00%)	18576(9.08%)	10576
fpuzzle.tra	45	204527	185285(90.59%)	19241(9.41%)	1(0.00%)	18576(9.08%)	13074

TABLE 2. History with associated PCs - best predictions (MGAp scheme - 100 entries)

Bench	HRg	Br.no.	Pr. Accuracy	Incorrect pr.	Bad target	NT branches	No of ev.
fsort.tra	9	12601	9441(74.92%)	2929(23.24%)	231(1.83%)	4414(35.03%)	0
fbubble.tra	9	41216	35174(85.34%)	6042(14.66%)	0(0.00%)	10140(24.60%)	0
fmatrix.tra	9	21341	20607(96.56%)	733(3.43%)	1(0.00%)	703(3.29%)	0
fperm.tra	45	54819	50321(91.79%)	2316(4.22%)	2182(3.98%)	10862(9.81%)	0
ftower.tra	9	37930	33043(87.12%)	1305(3.44%)	3582(9.44%)	9153(24.13%)	0
fqueens.tra	18	38462	31074(80.79%)	7369(19.16%)	19(0.05%)	19181(49.87%)	0
ftree.tra	27	32887	28216(85.80%)	3547(10.79%)	1124(3.42%)	8721(26.52%)	37
fpuzzle.tra	9	204527	193579(94.65%)	10946(5.35%)	2(0.00%)	18576(9.08%)	13

TABLE 3. History without associated PCs (GAp scheme) - 100 entries

Bench	HRg	Br.no.	Pr. Accuracy	Incorrect pr.	Bad target	NT branches	No of ev.
fsort.tra	1	12601	9354(74.23%)	3027(24.02%)	220(1.75%)	4414(35.03%)	0
fsort.tra	9	12601	7924(62.88%)	4569(36.26%)	108(0.86%)	4414(35.03%)	3390
fbubble.tra	1	41216	35166(85.32%)	6047(14.67%)	3(0.01%)	10140(24.60%)	0
fbubble.tra	9	41216	33499(81.28%)	7717(18.72%)	0(0.00%)	10140(24.60%)	2640
fmatrix.tra	1	21341	20613(96.59%)	725(3.40%)	3(0.01%)	703(3.29%)	0
fmatrix.tra	9	21341	20577(96.42%)	763(3.58%)	1(0.00%)	703(3.29%)	0
fperm.tra	1	54819	36998(67.49%)	6060(11.05%)	11761(21.45%)	10862(19.81%)	0
fperm.tra	9	54819	48188(87.90%)	2331(4.25%)	4300(7.84%)	10862(19.81%)	0
ftower.tra	1	37930	33432(88.14%)	1419(3.74%)	3079 (8.12%)	9153(24.13%)	0

ftower.tra	9	37930	32923(86.80%)	1348(3.55%)	3659 (9.65%)	9153(24.13%)	0
fqueens.tra	1	38462	3033(78.88%)	8101(21.06%)	22(0.06%)	19181(49.87%)	0
fqueens.tra	9	38462	26903(69.95%)	11546(30.02%)	13(0.03%)	19181(49.87%)	7508
ftree.tra	1	32887	28027(85.22%)	3617(11.00%)	1243(3.78%)	8721(26.52%)	0
ftree.tra	9	32887	25308(76.95%)	6400(19.46%)	1179(3.59%)	8721(26.52%)	3890
fpuzzle.tra	1	204527	192455(94.10%)	12070(5.90%)	2(0.00%)	18576(9.08%)	0
fpuzzle.tra	9	204527	182695(89.33%)	21832(10.67%)	0(0.00%)	18576(9.08%)	15059

TABLE 4. The behaviour of branch 35 belonging to "perm" benchmark

HRg	HRl	Taken (%)	Not taken (%)
101	10	1680 (67%)	839 (33%)
11101	10	1680 (67%)	839 (33%)
11111101	10	840 (59%)	579 (41%)
01011101	10	840 (76%)	260 (24%)

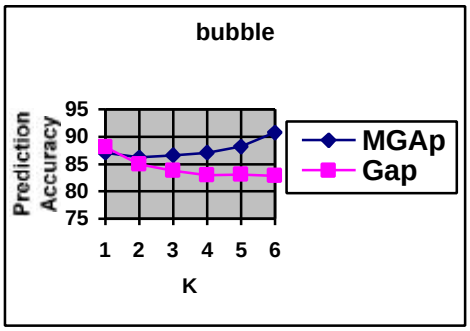

Figure 3. MGAp vs. GAp

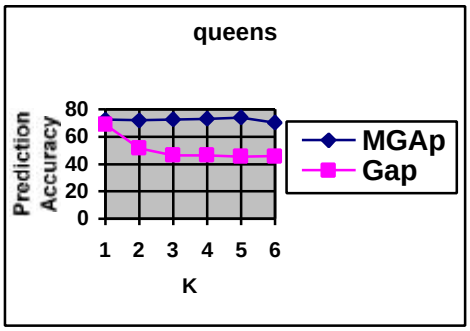

Figure 6. MGAp vs. GAp

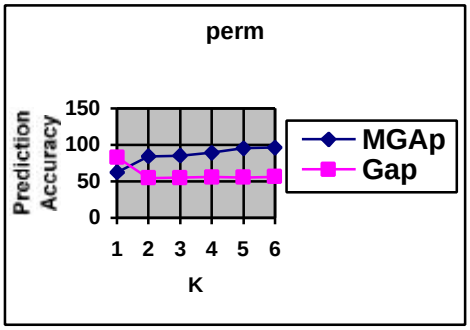

Figure 4. MGAp vs. GAp

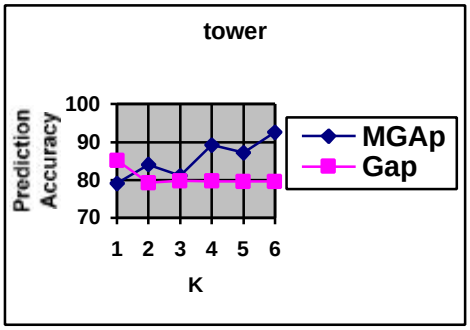

Figure 7. MGAp vs. Gap

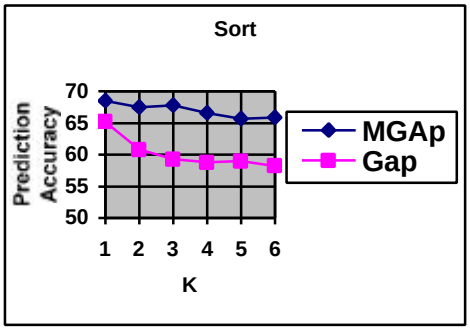

Figure 5. MGAp vs. GAp

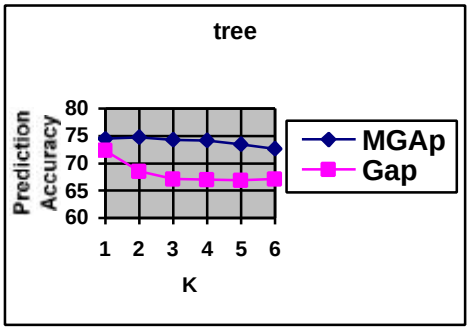

Figure 8. MGAp vs. GAp

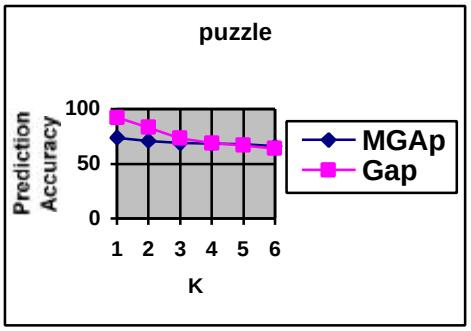

Figure 9. MGAp vs. Gap

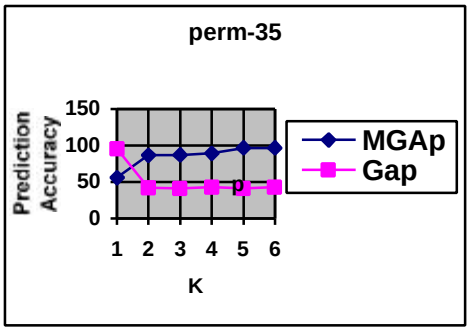

Figure 10. MGAp vs. GAp

REFERENCES

[Cha97] **Chang P.Y., Hao E., Patt Y.N**. - *Target Prediction for Indirect Jumps*, ISCA '97 - Ann. Int.'L Symp. Computer Architecture (http://www.eecs.umich.edu/HPS)

[Ega98] **Egan C.** - *Branch Predictor Report*, University of Hertfordshire, Department of Computer Science, UK, November, 1998

[Eve96] **Evers M., Chang P.Y., Patt Y.N**. - *Using Hybrid Branch Predictors to Improve Branch Prediction Accuracy in the Presence of Context Switches*, ISCA '96 (Ann. Int.'L Symp. Computer Architecture)

[Kim98] **Kim S.P., Tyson G.S.** – *Analyzing the Working Set Characteristics of Branch Execution*, 31st Annual ACM/IEEE Int'l Symp. On Microarchitecture, Dallas, USA, 30 Nov.- 2 Dec., 1998

[Mud96] **Mudge T.N., et al** - *Limits of Branch prediction*, Technical Report, Electrical Engineering and Computer Science Department, The University of Michigan, Ann Arbor, Michigan, USA, 1996

[Pan92] **Pan S.T., So K., Rahmeh J.T.** - *Improving the Accuracy of Dynamic Branch Prediction Using Branch Correlation*, ASPLOS V Conference, Boston, October 1992

[Per93] **Perleberg C., Smith A. J**. - *Branch Target Buffer Design and Optimisation*, IEEE Transactions on Computers, No. 4, 1993

[Rec98] **Reches S., Weiss S.** – *Implementation and Analysis of path History in Dynamic Branch Prediction Schemes*, IEEE Transactions on Computers, No. 8, 1998

[Sec95] **Sechrest S., Lee C., Mudge T.** - *The Role of Adaptivity in Two-Level Adaptive Branch Prediction*, Proceedings of MICRO-28, 1995

[Ste96] **Steven G. B. et al** - *A Superscalar Architecture to Exploit Instruction Level Parallelism*, Proceedings of the Euromicro Conference, 2-5 September, Prague, 1996.

[Vin99] **Vintan L., Armat C., Steven G.** - *The Impact of Cache Organisation on the Instruction Issue Rate of a Superscalar Processor*, Proceedings of Euromicro 7th Workshop on Parallel and Distributed Systems (http://www.elet.polimi.it/pdp99/), pg. 58-65, ISBN 0-7695-0059-5, Funchal, Portugal, 3rd -5th February, 1999

[Vin99a] **Egan C., Steven G., Vintan L.,** - *A Cost Effective Cached Correlated Two Level Adaptive Branch Predictor*, Eighteenth IASTED International Conference, AI '2000, February 14-17, Innsbruck, Austria, 2000

[Vin99b] **Steven G., Egan C., Quick P., Vintan L.** – *Reducing Cold Start Mispredictions in Two Level Adaptive Branch Predictors*, Proceedings of The 12th International Conference on Control Systems and Computer Science (CSCS 12), vol.2, ISBN 973-96609-5-9, Bucharest, Romania, May 26-29, 1999

[Yeh92] **Yeh T., Patt Y.** - *Alternative Implementations of Two Level Adaptive Branch Prediction*, 19 th Ann. Int.'L Symp. on Computer Architecture, 1992.