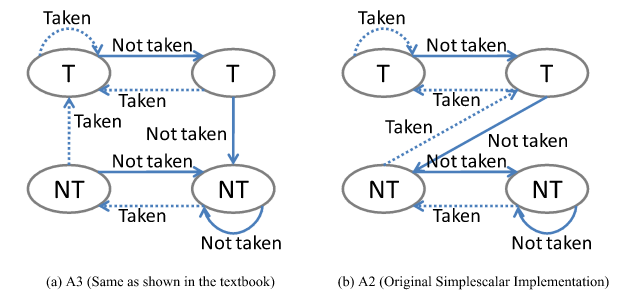
Computer Architecture

Submitted By- Ranjita Das (UIN – 524006365)

# **Homework 2 (Branch Predictor)**

**Part A**

The two different state diagrams in two-level branch prediction are:



A2 is a saturation up-down counter which is incremented when the branch is taken and is decremented when the branch is not taken The branch path of the next execution of the branch will be predicted as taken when the counter value is greater than or equal to two otherwise the branch will be predicted as not taken In Two Level Adaptive Branch Prediction the bit saturating up down counter keeps track of the history of a certain history pattern The counter is incremented when the result of a branch whose history register content is the same as the pattern history table entry index is taken otherwise the counter is decremented. The next time the branch has the same history register content which accesses the same pattern history table entry the branch is predicted taken if the counter value is greater or equal to two otherwise the branch is predicted not taken. A3 is simply a variation of A2.

The instructions per cycle executed by each benchmark on using the two predictors (shown above) are plotted in the graphs below:

As can be seen from the graph above , both A2 and A3 give execute almost equal number of instructions per cycle for all the benchmarks, except ammp. Incase of benchmark ammp, the the IPC achieved using A2 has been found to be quite higher than the rate achieved using A3.

The branch direction prediction rate obtained for each benchmark on using the two predictors (shown above) are plotted in the graphs below:

As can be seen from the graph above , both A2 and A3 give almost identical branch prediction rates for all the benchmarks, except ammp. Incase of benchmark ammp, the the prediction rate achieved using A3 has been found to be quite higher than the rate achieved using A2.

Overall, both being two – level dynamic branch predictors, these two state diagrams have almost same performance.

**Part B**

We have compared the performance of the following four branch predictors:

(1) GAg: 1 global history register and 1 global prediction table

(2) GAp: 1 global history register and 8 per-address prediction tables

(3) PAg: 8 per-address history registers and 1 global prediction table

(4) PAp: 8 per-address history registers and 8 per-address prediction tables

GAg

In GAg there is only a single global history register GHR and a single global pattern history table GPHT used by the Two Level Adaptive Branch Prediction. All branch predictions are based on the same global history register and global pattern history table which are updated after each branch is resolved This variation therefore is called Global TwoLevel Adaptive Branch Prediction using a global pattern history table Gag. Since the outcomes of different branches update the same history register and the same pattern history table the information of both branch history and pattern history is influenced by results of different branches The prediction for a conditional branch in this scheme is actually dependent on the outcomes of other branches.

PAg

In order the reduce the interference in the first level branch history information, one history register is associated with each distinct static conditional branch to collect branch history information individually The history registers are contained in a per address branch history table PBHT in which each entry is accessible by one specific static branch instruction and is accessed by branch instruction addresses Since the branch history is kept for each distinct static conditional branch individually and all history registers access the same global pattern history table this variation is called Per address Two Level Adaptive Branch Prediction using a global pattern history table PAg. The execution results of a static conditional branch update the branches own history register and the global pattern history table The prediction for a conditional branch is based on the branches own history and the pattern history bits in the global pattern history table entry indexed by the content of the branches history register Since all branches update the same pattern history table the pattern history interference still exists.

Pap

In order to completely remove the interference in both levels each static branch has its own pattern history table a set of which is called a per address pattern history table PPHT. Therefore a per address history register and a per address pattern history table are associated with each static conditional branch All history registers are grouped in a per address branch history table Since this variation of Two Level Adaptive Branch Prediction keeps separate history and pattern information for each distinct static conditional branch it is called Per address Two Level Adaptive Branch Prediction using Per address pattern history tables PAp

The instructions per cycle executed by each benchmark on using the four predictors (shown above) are plotted in the graphs below:

As seen in the graphs above, the instructions executed per cycle is highest when the Gap predictor is used. And this has been found to be true for all the benchmarks except benchmark bzip where it is just slightly less than the IPC achieved using PAg. For all the other benchmarks IPC achieved using PAg is found to be lower than IPC achieved using Gap as well as Pap. IPC achieved using Gag is found to be almost same as IPC achieved using PAp (except benchmark parser, where it is much lower).

Hence overall, predictor Gap has been found to give the best performance.

As seen in the graphs above, among the benchmarks that use floating point values too, the instructions executed per cycle is highest when the Gap predictor is used. And this has been found to be invariably true for all the benchmarks except benchmark art where it is just slightly less than the IPC achieved using PAg. For all the other benchmarks IPC achieved using PAg is found to be lower than IPC achieved using Pap. IPC achieved using Gag is found to be either same or less than IPC achieved using the other predictors

Therefore we can conclude that for benchmarks using floating point values too, predictor Gap has been found to give the best performance.

The branch direction prediction rate obtained for each benchmark on using the four predictors are plotted in the graphs below:

As seen in the graphs above, the instructions executed per cycle is highest when the GAp predictor for most of the benchmarks. It is second highest in case of benchmarks applu, galgel, mgrid. The performance of predictor PAg relative to the other predictors is found to be better in case of benchmarks using floating point values than its relative performance when used in benchmarks dealing with integer values.

Considering all the benchmarks, we can consider Gap to be the best branch direction predictor for floating point values.

As seen in the graphs above, the instructions executed per cycle is highest when the GAp predictor is used for all benchmarks without any exception. The performance of predictor GAg relative to the other predictors is found to be better in case of benchmarks using integers than its relative performance when used in benchmarks dealing with floating point values. The prediction rate achieved using predictor PAp is less than GAp but is more than the other two predictors and hence is the second best choice.

Hence overall, predictor GAp has been found to give the best branch direction prediction rate for integer values.