Khyati Mardia – ASU ID -1215346587

CSE520 Computer Architecture II – Spring 2019
Programming Assignment -3

1. 2-bit local branch predictor :

A local branch predictor has a table with n- bit saturating counters which are indexed by the branch address. 2-bit local branch predictor has a 2 bit counter where MSB bit gives the branch prediction result. Each conditional jump has the history buffer for each one branch and the pattern history table may be separate as well or it may be shared between all conditional jumps. This 2 bit predictor takes its input size as the total number of bits ie number of entries in the counter bits. Number of local predictor sets are equal to local predictor size divided by local control bits. The saturation bit counter in case of 2 bit predictor has 4 states, strongly taken branch, weakly taken branch, weakly not taken branch and strongly not taken branch. Hence, LSB bit gives the additional information about result prediction made by MSB bits. In case of next branch, based on whether the branch is taken or not taken , counter value is incremented(if branch taken) or decremented(if branch not taken).

2. Tournament branch predictor :

Tournament branch predictor combines the functionalities of local and global predictors. It has a global history with choice counters. The choice counters selects the prediction outcome from one of them. The outcome is based on the prediction correctness of two predictors. Both the predictors and the choice counters to be updated after each branch prediction. Here, global predictor is indexed by history. register and local predictor is indexed by LSBs of the program counter. Tournament branch predictor is accurate which predicts the correct branch, but it requires extra hardware.

3. gDAC branch predictor :

Global history divide and conquer branch predictor neural-class based method. It divides the long global branch history register into non overlapping segments. Each segment has branch history input from branch history register. There are three main functions in gDAC branch predictor.
1. Lookup : Lookup function predicts whether the branch is taken or not taken and what is the target address.
2. Update : Update function reserves the branch instructions and also updates the history tables based on the result.
3. Squash : Squash function comes into the picture when there is a mis prediction. Say, during the out of order execution, branch 2 is mis predicted but we have already predicted branch 3, 4 and 5, then we need to delete the prediction of branch 5, 4 and 3 (in this order) to recover the history. After that, we have to update the history and delete the history. GDAC is bimodal based implementation with 2 bit saturating counter.
4. uncondBranch : Here, branch is always taken and hence nothing to be done. Only global history to be updated as unconditional branch is always taken.

Implementation of the Branch Predictors in Gem5 :

Local 2-bit branch predictor, Tournament branch predictor and gDAC branch for 8Kb and 32 Kb size for each is implemented.

Local branch predictor for 8 Kb and 32 Kb is inherit-ant from the base class - LocalBP provided in the BranchPredictor.py file. Local control bit is given as 2 as it is 2 bit BP and local predictor size is the size of the predictor (For 8 Kb = 8*1024 bits and for 32 Kb = 32*1024 bits). (The values are mentioned in bits instead of entries because of the structure of its source code)

Tournament branch predictor for 8Kb and 32 Kb is inherit-ant from the base class – Tournament BP provided in the BranchPredictor.py file. Local control bits are given as 2 as it is 2 bit BP, and local history table, local predictor size , global history register, global predictor size and choice predictor size entries are given from the table in the assignment.

Implementation of gDAC branch predictor:

1. gDAC branch predictor base class is created in the BranchPredictor.py file which has parameters like global predictor sizes, global control bits, choice predictor size, choice control bits, global history bits, segment sizes , fusion predictor sizes and function control bits.

2. Two classes with 8Kb and 32 Kb are inherit-ant from the above class and sizes are given from the table from the IEEE paper.

3. gDACBP has two segments (from branch history register) with each bimode implementation (32 bits total), program counter (branch address register – 52 bits).

4. The bits are considered from taken and non taken branches or history table which is hashed with choice history table which is further given to fusion table for the prediction.

5. After every branch prediction, all values like counters, history etc are updated.

6. Hash function is created to calculate the program counter bits with utilization of all 52 bits dynamically. Say program counter is 52 bits, which is divided by total number of bits from branch address which is to be segmented , which gives the number of bits PC to be divided into. Instead of considering on LSBs, I am considering all 52 bits by folding. Each branch address is folded number of times as bits and is hashed or xored to get desired number of bits. This is valid for any number of segmentation.

7. Lookup function first creates number of segments required from global history register or branch history register. For segment – n we have to consider from LSB , so accordingly bits are shifted and hashed and segments are obtained.

8. Choice history table index is obtained using hash function defined. Global history index is defined for each segment and is hashed.

9. Assert functions check for certain true conditions and throws an error if false.

10. BP history has elements like whether branch is taken or not.

11. Fusion table is defined in lookup , where it have 4 bit counters and of size of 4096 bits. For the final prediction, to input to the fusion, the bits from program counter , and output from segment 1 and segment 2 is given.

12. Number of bits are calculated based on PHT entries and segment 1 and segment 2. Instead of ignoring MSB bits, I have considered all the bits by folding it to get desired bit which gives correlation instead of losing information.

13. MSB bit of 4 bit counter from the fusion table is the final prediction for branch is taken or not taken.

14. Update function updates the branch history , choice history table index , global history index etc after each branch prediction.

15. There are shared hysteresis bit shared by segments which says 1.5 bits implementation. I have implemented hysteresis bit function in the program. For the same I have created gDAC counter which counts the value and increment and decrement based on the branch prediction.

16. The sequence of the counter is 01-00-10-11. When there are two branch assigned to the same address, and where one branch is taken where as one branch is not taken , the prediction causes the interference. To avoid this, hysteresis bit is introduced which is shared by branch taken and branch not taken.

17. There is LSB bit which is hysteresis bit which is shared by branch taken and not taken, the MSB bit is for the prediction of each branch. Instead on 2 bit saturating counters in bimode , we are splitting it in single bit counters for each branch taken and not taken.

18. Hence, when there is branch is taken and say counter value is 01, the next branch taken with counter value of 00, the LSB bit is extracted for hysteresis bit, if the hysteresis bit of last prediction and current prediction is different, it is should be updated. For each increment and decrement , hysteresis bit is to be compared and updated. It is implemented in the program.

19. Fusion index and fusion tables are updated and fusion counters are increment and decremented as per branch prediction.

20. All header files are defined and linked. Script is created to run all commands together.

21. Squash deletes the history in case of mis prediction, previous history to be recovered.

22. Unconditional branch make all values as true as branch is always taken and global history is updated.

Branch prediction implementation is done for various inputs, various size and compared its accuracy and Instructions per cycle.

Observations from graphs :

1. BFS with input R10K.graph

		BFS with input R10K.graph		
	Instructions per cycle	Incorrect branch predicted	Total branch Predicted	Accuracy
Local – 8 KBP	0.9357	835911	16968378	95.07
Local – 32 KBP	0.9354	836101	16969550	95.46
Tournament– 8 KBP	0.9554	775028	17034599	96.22
Tournament– 32 KBP	0.9671	743180	16870093	96.11
GDAC – 8 KBP	0.7693	1640163	21716449	92.47
GDACl – 32 KBP	0.7219	1969706	23472929	92.37

- Instructions per cycle is highest in case of Tournament predictor with high predictor size. Local 2-bit predictor is second best which show same results for 8 Kb and 32 Kb sizes. gDAC gives poor instructions per cycle , the reason can be as we have not implemented ahead pipelining, which calculates predictions from 3 cycles ahead.
-Accuracy for BFS input is highest for Tournament BP and second highest is for Local BP. GDAC shows poor performance as compared to other two.
-There are no significance difference with respect to change in the predictor size , the accuracy obtained is same.

2. MST with input rand-weighted-small.graph

		MST with input rand-weighted-small.graph		
	Instructions per cycle	Incorrect branch predicted	Total branch Predicted	Accuracy
Local – 8 KBP	1.0885	390495	7915447	95.07
Local – 32 KBP	1.1017	351669	7748777	95.46
Tournament– 8 KBP	1.1676	270876	7166222	96.22
Tournament– 32 KBP	1.1598	279814	7201749	96.11
GDAC – 8 KBP	0.8306	780104	10355407	92.47
GDACl – 32 KBP	0.8091	813574	10660414	92.37

-For MST inputs, IPC are similar in case of local BP and tournament BP for 8kB and 32 kB, gDAC shows less IPC as compared to other two.
- Accuracy for the tournament is highest and same for predictor sizes, whereas Local predictor shows little less accuracy as compared to tournament BP. GDAC shows less accuracy for MST inputs and almost same values for predictor size of 8 Kb and 32 Kb. Accuracy is less for GDAC by 3-5%

3. queens with option " -c 11 "

		queens with option "-c 11"		
	Instructions per cycle	Incorrect branch predicted	Total branch Predicted	Accuracy
Local – 8 KBP	0.7764	1388653	15811540	95.07
Local – 32 KBP	0.7764	1388753	15811651	95.46
Tournament– 8 KBP	0.7277	1481531	15864850	96.22
Tournament– 32 KBP	0.7333	1467454	15793656	96.11
GDAC – 8 KBP	0.7253	1504694	15800898	92.47
GDACl – 32 KBP	0.7289	1490947	15680503	92.37

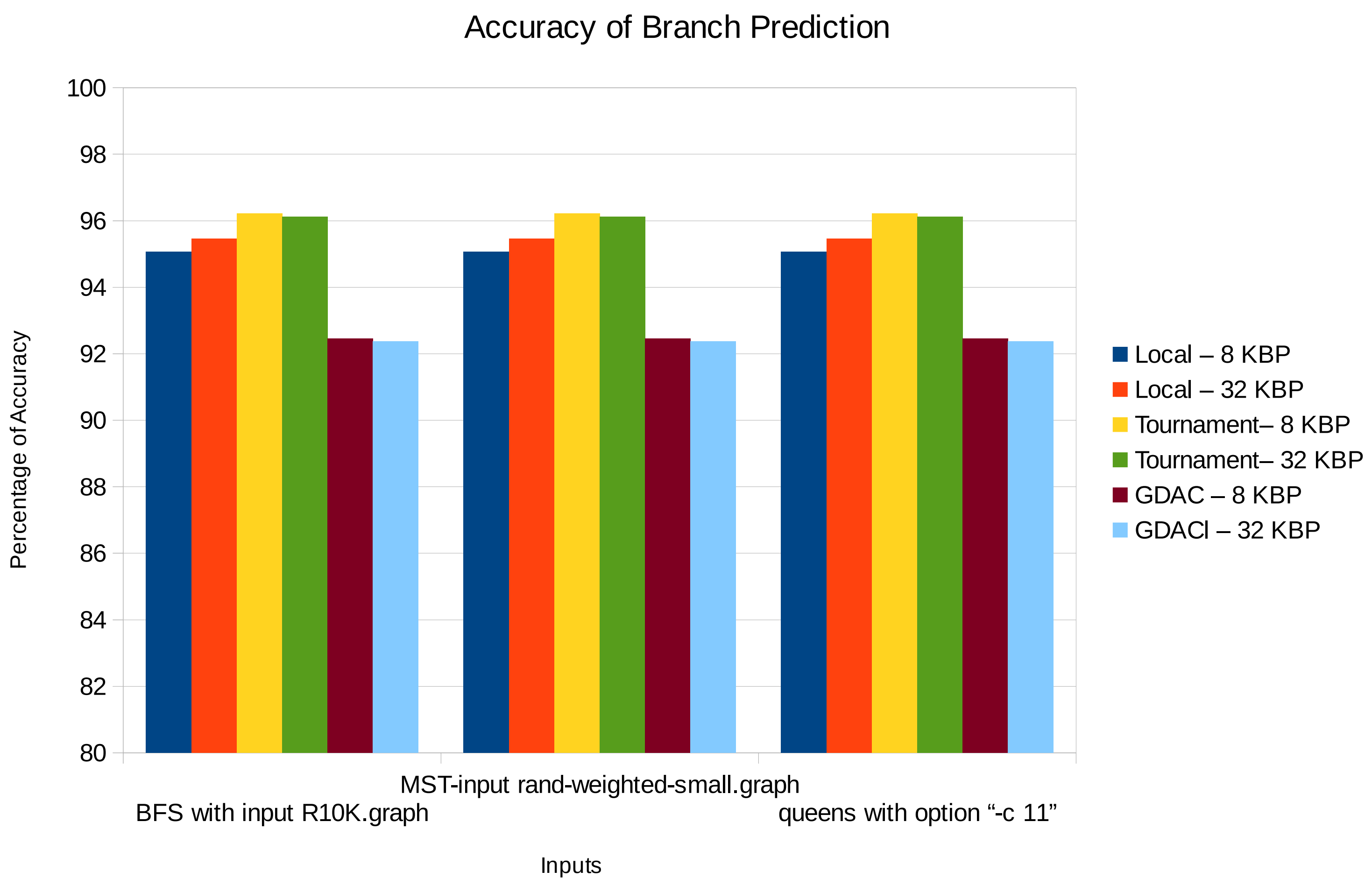
-IPC for all 3 BPs are observed less as compared to other 2 inputs as the computational time for

“queens” input can be more and which is causing less number of instruction execution for all BP.
- Accuracy for Tournament BP is again highest for input and Local is slightly less than Tournament BP. GDAC shows approximately 3-5% less accuracy for “queens” input for predictor size of 8 Kb and 32 Kb.

4. Accuracy for all inputs and all predictors

	Accuracy		
	BFS with input R10K.graph	MST-input rand-weighted-small.graph	queens with option “-c 11”
Local – 8 KBP	95.07	95.07	95.07
Local – 32 KBP	95.46	95.46	95.46
Tournament– 8 KBP	96.22	96.22	96.22
Tournament– 32 KBP	96.11	96.11	96.11
GDAC – 8 KBP	92.47	92.47	92.47
GDACl – 32 KBP	92.37	92.37	92.37

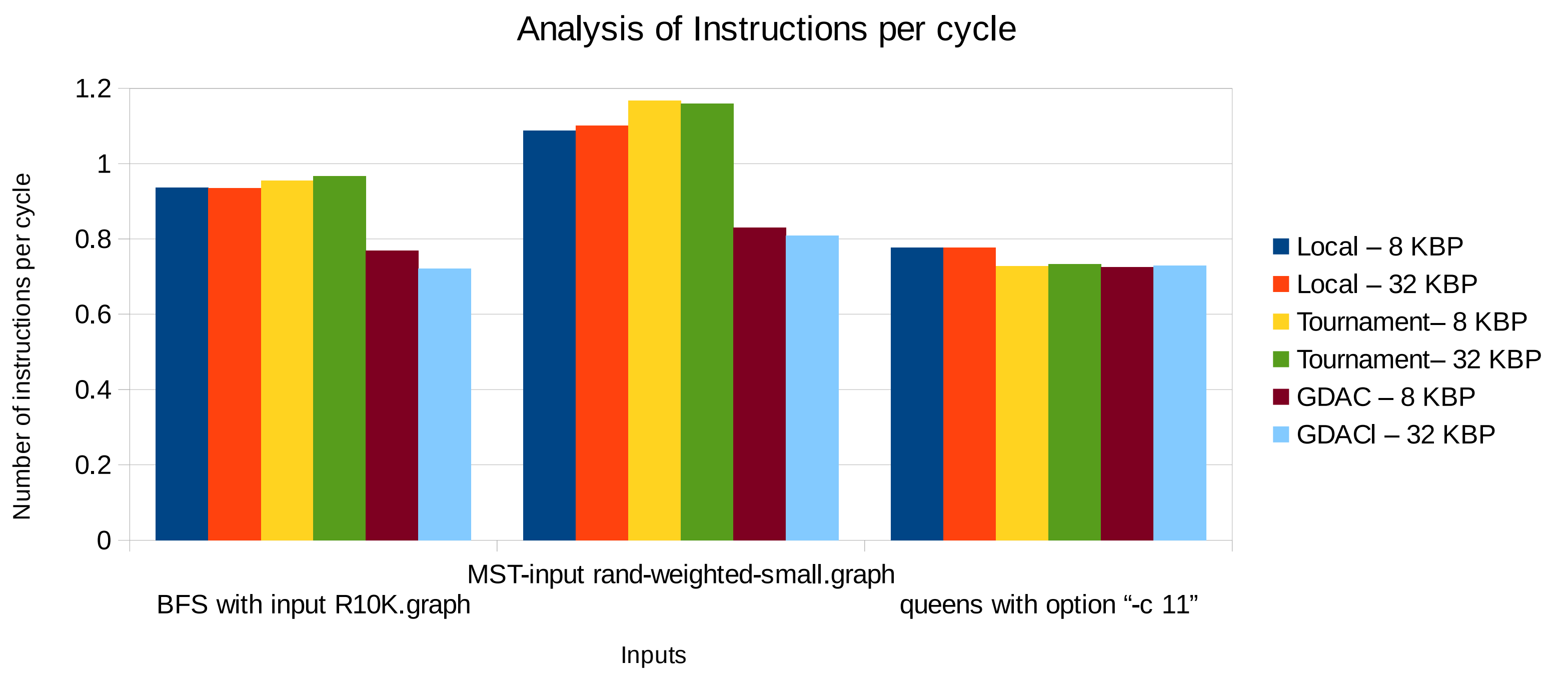
This is the summary of table 1, 2 and 3 and graphs are shown below.



5. Instructions per cycle for all inputs and all predictors

Instructions per cycle			
	BFS with input R10K.graph	MST-input rand-weighted-small.graph	queens with option "-c 11"
Local – 8 KBP	0.9357	1.0885	0.7764
Local – 32 KBP	0.9354	1.1017	0.7764
Tournament– 8 KBP	0.9554	1.1676	0.7277
Tournament– 32 KBP	0.9671	1.1598	0.7333
GDAC – 8 KBP	0.7693	0.8306	0.7253
GDACl – 32 KBP	0.7219	0.8091	0.7289

This is the summary of tables 1, 2 and 3 and graphs are shown below.



General Observations:

1. We have not implemented ahead pipelining because of which the accuracy of gDAC is not that good. Ahead pipelining calculate parameters 3 cycles ahead which has more accuracy towards branch prediction.

2. As the predictor size is increased, the accuracy should increase as it will have less hash collision and more space to accommodate branch history and same can be seen in the Tournament predictor results.

3. We can reduce mis prediction by saving next branch addresses which results in better correlation.

4. gDAC accuracy is overall poor because of overheads and it is because it is a complex branch

predictor as compared to local and tournament predictor. Also, hysteresis slightly improves the performance of the gDAC but not as much as the local and tournament predictors.

5. IPC increases with Accuracy as branch penalty is reduced and that can be seen in each observations.

6. Tournament BP shows the best performance for all cases. Second best performance is shown by Local predictor. GDAC shows less accuracy and less IPC as compared to other two.

7. Output also depend on the pattern of inputs and how many conditional and unconditional branch it has.

8. Output for different predictor sizes is seen almost same/similar accuracy and IPC for all three predictors.