Kristopher Lopez
Dr. Anup Das
ECEC 412: Modern Processor Design
29 November 2021

Project 3: Implementing gShare branch prediction and evaluating its performance using ML workloads

Objective

The objective of this project is to evaluate the performance of a gShare predictor and compare various configurations of it to that of a 2-bit local predictor and a tournament predictor with modified parameters

Implementation

In order to implement the gShare predictor, portions of the tournament code were copied over as the tournament predictor is a combination of a global and local predictor, meaning it has sections of both incorporated into it. Since the gShare predictor is only a global predictor, only the values shown in figure 1 were necessary in the Branch_Predictor struct for gShare.

```
unsigned global_predictor_size;
unsigned global_history_mask;

Sat_Counter *global_counters;

uint64_t global_history;
```

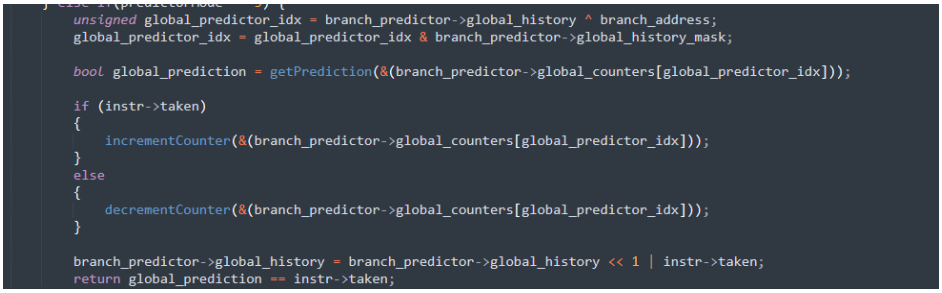
Figure 1

In the initialization of the branch predictor, segments of code were taken from the tournament predictor and copied over as shown in figure 2. Since gShare only needs to worry about its global table, all we needed was the initializer for it and the proper setting of the variables shown in figure 1.

```
assert(checkPowerofTwo(globalPredictorSize));
branch_predictor->global_predictor_size = globalPredictorSize;

branch_predictor->global_counters =
    (Sat_Counter *)malloc(globalPredictorSize * sizeof(Sat_Counter));
for (int i = 0; i < globalPredictorSize; i++)
{
    initSatCounter(&(branch_predictor->global_counters[i]), globalCounterBits);
}

branch_predictor->global_history_mask = globalPredictorSize - 1;
branch_predictor->global_history = 0;
```

Figure 2

The same methodology goes for the prediction itself, with the code in figure 3 showing what was copied over from the tournament predictor into the gShare predictor.

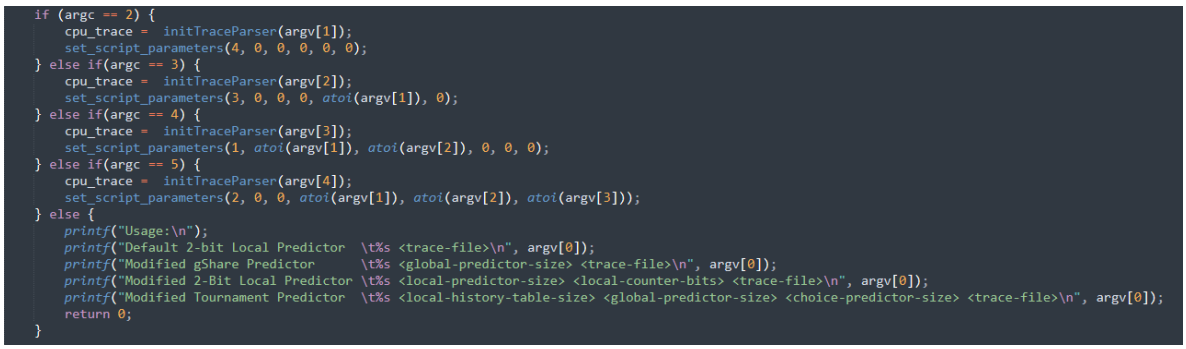


```
unsigned global_predictor_idx = branch_predictor->global_history ^ branch_address;
global_predictor_idx = global_predictor_idx & branch_predictor->global_history_mask;

bool global_prediction = getPrediction(&(branch_predictor->global_counters[global_predictor_idx]));

if (instr->taken)
{
    incrementCounter(&(branch_predictor->global_counters[global_predictor_idx]));
}
else
{
    decrementCounter(&(branch_predictor->global_counters[global_predictor_idx]));
}

branch_predictor->global_history = branch_predictor->global_history << 1 | instr->taken;
return global_prediction == instr->taken;
```

Figure 3

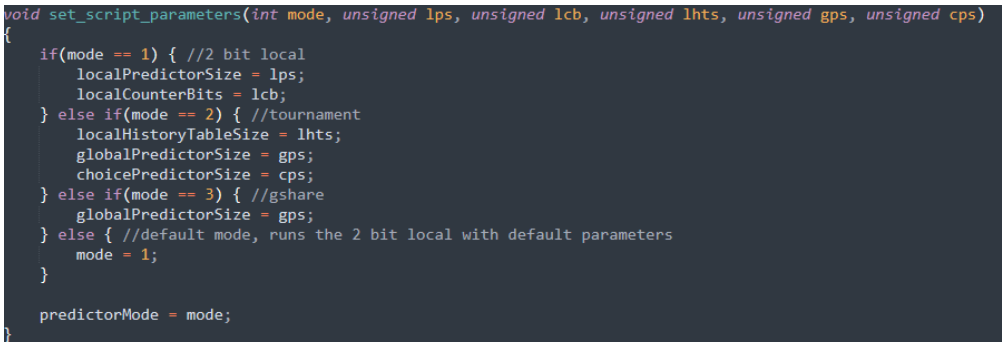
The main difference between the global portion of the tournament predictor and the gShare predictor is how the global predictor index is calculated, with the gShare predictor taking the xor of both the branch address and the global history in order to locate the index in the counters table.

Aside from the code written directly for the gShare implementation, extra was added for easy testing of the 3 different predictors with different parameters. The use of defining the predictor was removed, and was replaced with a mode value that is passed at the execution time of the code to determine which predictor to run. Along with this, other values specific to that mode are to be passed as well to allow for modification of the table sizes. As shown in figure 4 and 5, the values are passed in to the main function as script arguments and then passed to the branch predictor so that the table size constants can be set.



```
if (argc == 2) {
    cpu_trace =  initTraceParser(argv[1]);
    set_script_parameters(4, 0, 0, 0, 0, 0);
} else if(argc == 3) {
    cpu_trace =  initTraceParser(argv[2]);
    set_script_parameters(3, 0, 0, 0, atoi(argv[1]), 0);
} else if(argc == 4) {
    cpu_trace =  initTraceParser(argv[3]);
    set_script_parameters(1, atoi(argv[1]), atoi(argv[2]), 0, 0, 0);
} else if(argc == 5) {
    cpu_trace =  initTraceParser(argv[4]);
    set_script_parameters(2, 0, 0, atoi(argv[1]), atoi(argv[2]), atoi(argv[3]));
} else {
    printf("Usage:\n");
    printf("Default 2-bit Local Predictor  \t%s <trace-file>\n", argv[0]);
    printf("Modified gShare Predictor      \t%s <global-predictor-size> <trace-file>\n", argv[0]);
    printf("Modified 2-Bit Local Predictor \t%s <local-predictor-size> <local-counter-bits> <trace-file>\n", argv[0]);
    printf("Modified Tournament Predictor  \t%s <local-history-table-size> <global-predictor-size> <choice-predictor-size> <trace-file>\n", argv[0]);
    return 0;
}
```

Figure 4



```
void set_script_parameters(int mode, unsigned lps, unsigned lcb, unsigned lhts, unsigned gps, unsigned cps)
{
    if(mode == 1) { //2 bit local
        localPredictorSize = lps;
        localCounterBits = lcb;
    } else if(mode == 2) { //tournament
        localHistoryTableSize = lhts;
        globalPredictorSize = gps;
        choicePredictorSize = cps;
    } else if(mode == 3) { //gshare
        globalPredictorSize = gps;
    } else { //default mode, runs the 2 bit local with default parameters
        mode = 1;
    }

    predictorMode = mode;
}
```

Figure 5

The original method of running the script still works, however new ways of passing different parameters have also been added and outlined in the usage statement printed out by the script, as also shown in figure 4.

Results and Comparison

Below are the charts showcasing the results of the 3 predictors with the parameters provided in the assignment outline. Each of the tables displays the predictor method at the top, with the cpu_trace follow displayed directly below it. On the left side of the table is the partition displaying the modified parameters of the predictor, as well as the values of those modified parameters. On the right side of the table are the output values of the simulation runs, with the number of correct/incorrect predictions as well as the correctness percentage being shown.

Two Bit Local Predictor				
sample.cpu_trace				
Parameters		Output Values		
Local Predictor Size	Local Counter Bits	# of Correct Predictions	# of Incorrect Predictions	Correctness
2048	1	8217	1783	82.17
2048	2	8467	1533	84.67
4096	2	8468	1532	84.68
8192	2	8476	1524	84.76
16384	2	8474	1526	84.74
32768	2	8473	1527	84.73
65536	2	8473	1527	84.73

Two Bit Local Predictor				
531.deepsjeng_r_branches.cpu_trace				
Parameters		Output Values		
Local Predictor Size	Local Counter Bits	# of Correct Predictions	# of Incorrect Predictions	Correctness
2048	1	168274768	36445198	82.19753
2048	2	175794247	28925719	85.87059
4096	2	177385518	27334448	86.64789
8192	2	178085223	26634743	86.98966
16384	2	178223608	26496358	87.05727
32768	2	178223992	26495974	87.05746
65536	2	178224001	26495965	87.05746

Two Bit Local Predictor				
541.leela_r_branches.cpu_trace				
Parameters		Output Values		
Local Predictor Size	Local Counter Bits	# of Correct Predictions	# of Incorrect Predictions	Correctness
2048	1	168602556	47530217	78.00879
2048	2	178713549	37419224	82.68693
4096	2	178989861	37142912	82.81477
8192	2	179391515	36741258	83.00061
16384	2	179415029	36717744	83.01148
32768	2	179417105	36715668	83.01245
65536	2	179417270	36715503	83.01252

Two Bit Local Predictor				
548.exchange2_r_branches.cpu_trace				
Parameters		Output Values		
Local Predictor Size	Local Counter Bits	# of Correct Predictions	# of Incorrect Predictions	Correctness
2048	1	255602221	98248979	72.23438
2048	2	292073706	61777494	82.54139
4096	2	292146797	61704403	82.56204
8192	2	292178159	61673041	82.57091
16384	2	292195854	61655346	82.57591
32768	2	292198756	61652444	82.57673
65536	2	292200308	61650892	82.57717

Tournament Predictor					
sample.cpu_trace					
Parameters			Output Values		
Local History Table Size	Global Predictor Size	Choice Predictor Size	# of Correct Predictions	# of Incorrect Predictions	Correctness
2048	8192	8192	8197	1803	81.97
4096	8192	8192	8207	1793	82.07
4096	16384	16384	8203	1797	82.03
16384	32768	32768	8219	1781	82.19
32768	65536	65536	8212	1788	82.12

Tournament Predictor					
531.deepsjeng_r_branches.cpu_trace					
Parameters			Output Values		
Local History Table Size	Global Predictor Size	Choice Predictor Size	# of Correct Predictions	# of Incorrect Predictions	Correctness
2048	8192	8192	186109401	18610565	90.90926
4096	8192	8192	186687194	18032772	91.1915
4096	16384	16384	188341148	16378818	91.99941
16384	32768	32768	190059081	14660885	92.83857
32768	65536	65536	191183580	13536386	93.38786

Tournament Predictor					
541.leela_r_branches.cpu_trace					
Parameters			Output Values		
Local History Table Size	Global Predictor Size	Choice Predictor Size	# of Correct Predictions	# of Incorrect Predictions	Correctness
2048	8192	8192	181642358	34490415	84.04202
4096	8192	8192	181934854	34197919	84.17736
4096	16384	16384	183491957	32640816	84.89779
16384	32768	32768	185250784	30881989	85.71157
32768	65536	65536	186794868	29337905	86.42598

Tournament Predictor					
548.exchange2_r_branches.cpu_trace					
Parameters			Output Values		
Local History Table Size	Global Predictor Size	Choice Predictor Size	# of Correct Predictions	# of Incorrect Predictions	Correctness
2048	8192	8192	336251743	17599457	95.02631
4096	8192	8192	336333680	17517520	95.04947
4096	16384	16384	337201241	16649959	95.29465
16384	32768	32768	338211245	15639955	95.58007
32768	65536	65536	338994373	14856827	95.80139

gShare Predictor			
sample.cpu_trace			
Parameters	Output Values		
Global Predictor Size	# of Correct Predictions	# of Incorrect Predictions	Correctness
8192	7485	2515	74.85
16384	7422	2578	74.22
32768	7327	2673	73.27
65536	7275	2725	72.75
131072	7219	2781	72.19

gShare Predictor			
531.deepsjeng_r_branches.cpu_trace			
Parameters	Output Values		
Global Predictor Size	# of Correct Predictions	# of Incorrect Predictions	Correctness
8192	179529273	25190693	87.69505
16384	183701357	21018609	89.733
32768	186896906	17823060	91.29393
65536	189182509	15537457	92.41039
131072	190893186	13826780	93.246

gShare Predictor			
541.leela_r_branches.cpu_trace			
Parameters	Output Values		
Global Predictor Size	# of Correct Predictions	# of Incorrect Predictions	Correctness
8192	172317632	43815141	79.72768
16384	176292704	39840069	81.56686
32768	179837616	36295157	83.20701
65536	183053012	33079761	84.6947
131072	185592336	30540437	85.8696

gShare Predictor			
548.exchange2_r_branches.cpu_trace			
Parameters	Output Values		
Global Predictor Size	# of Correct Predictions	# of Incorrect Predictions	Correctness
8192	328448818	25402382	92.82118
16384	331533004	22318196	93.69276
32768	333952566	19898634	94.37656
65536	336179077	17672123	95.00578
131072	338074790	15776410	95.54151

Below are the combinations that gave the best outcome for each AI workload across the 3 different predictors.

Two Bit Local Predictor			
Input Parameters			Output
Trace Filename	Local Predictor Size	Local Counter Size	Correctness
sample.cpu_trace	8192	2	84.76
531.deepsjeng_r_branches.cpu_trace	65536	2	87.05746
541.leela_r_branches.cpu_trace	65536	2	83.01252
548.exchange2_r_branches.cpu_trace	65536	2	82.57717

Tournament Predictor				
Input Parameters				Output
Trace Filename	Local History Table Size	Global Predictor Size	Choice Predictor Size	Correctness
sample.cpu_trace	16384	32768	32768	82.19
531.deepsjeng_r_branches.cpu_trace	32768	65536	65536	93.38786
541.leela_r_branches.cpu_trace	32768	65536	65536	86.42598
548.exchange2_r_branches.cpu_trace	32768	65536	65536	95.80139

gShare Predictor		
Input Parameters		Output
Trace Filename	Global Predictor Size	Correctness
sample.cpu_trace	8192	74.85
531.deepsjeng_r_branches.cpu_trace	131072	93.246
541.leela_r_branches.cpu_trace	131072	85.8696
548.exchange2_r_branches.cpu_trace	131072	95.54151

Based on the values in the 3 tables above, the two bit local predictor best performed with a local predictor size of 65536 and a local counter bit size of 2 when simulated under the AI workloads. The tournament predictor also performed best with its max table values, with a local history table size of 32768 and a global/choice predictor size of 65536. The gShare predictor had overall better performance when compared to the two bit local predictor with a global predictor size of 131072, and almost matched the performance of the tournament predictor in each of the workloads.

Conclusion

In conclusion, based on the implementation of gShare above and the final results shown in the previous section, the gShare predictor is able to match the tournament predictor and completely out-perform the two bit local predictor when given a large enough global predictor size.

Based on the results of the gShare runs, it seems that the correctness of the predictor increases linearly by a small percentage with each power of 2 increase to the global predictor size.