Cameron Calv and Nicholas Sica
ECEC 621 High-Performance Computer Architecture
Project Four – Implementing Cache Replacement Policy and evaluating its performance using AI Workloads Using PC-Signature-based Hit Predictor

How PC-Signature-based Hit Prediction Works

This particular algorithm works to use some aspect of the cache block to determine whether a block will be re-referenced again. In this case, we use a certain number of the most significant bits (MSBs) of the program counter (PC) to determine whether certain program counter values have a higher chance of being re-referenced. Each cache block has two new parameters, the signature_m which is an index into a new hardware component called the Signature History Counter Table (SHCT) and a single bit indicator, the outcome. Each cache block has a signature that is determined by taking some number of the MSBs of the PC to index into the SHCT which has a counter that is incremented or decremented based on two situations. When a hit is encountered in the cache, the corresponding counter of the block that is a hit is incremented and its outcome bit is set to 1. Upon eviction from the cache, if the victim block's outcome bit is 0, meaning it was not referenced more than once, then its corresponding counter in the SHCT is decremented. When it comes to place a new block in the cache, its value if it is distant, if its corresponding entry in the SHCT is zero, then its is placed with a value that is more likely to be evicted. In the case of LFU, this is a frequency of zero and for LRU this is a previous access time.

Evaluating Prediction for LRU and LFU with Various Mask Magnitudes

The number of bits used to determine the up-to-32-bit signature_m of the SHCT hardware component was changed from the 1 MSBs up to the 64 MSBs. Note that results did not change until an 18-bit mask was used. This was because even though PC is stored as a 64-bit integer, its values never go above that of a 46-bit number therefore the 18 MSB of the PC are all zeros. For any number of MSBs greater than 49, a window of 32-bits is still used to index into SHCT, but that window would slide to the right leaving the MSBs beyond the window out of the signature calculation. The change in hit rate of the policies are reflected below in the tables testing them against parameters chosen for an LRU policy and an LFU policy. Note that the parameters chosen for LRU and LFU represented the ones with the lowest hit rates so that the change in hit rate caused by the signature-based prediction would be more obvious.

Table 1: Various hit rates by changing the number of mask bits determining signature_m. An LRU policy was used for these values with parameters set to N-Associatives = 4 and Cache Size = 128 kBytes.

Signature Mask Bit #	Sample	Deepsjeng	Leela	Exchange2
1-17	42.580000%	76.939192%	42.554451%	99.926982%
18	42.590000%	76.939192%	42.554453%	99.926982%
20 - 40	42.580000%	76.939188%	42.554451%	99.927012%
42	42.580000%	76.939188%	42.554451%	99.926997%
44 - 46	42.580000%	76.939184%	42.554455%	99.927012%
48	42.590000%	76.939186%	42.554461%	99.927012%
49	42.610000%	76.939186%	42.554461%	99.927041%
50	42.620000%	76.939188%	42.554466%	99.927041%
51	42.620000%	76.939184%	42.554468%	99.927056%
52	42.600000%	76.939202%	42.554464%	99.927056%
53	42.660000%	76.939182%	42.554500%	99.927041%

54	42.720000%	76.939186%	42.554523%	99.927070%
55	42.740000%	76.939210%	42.554527%	99.927070%
56	42.740000%	76.939196%	42.554521%	99.927070%
57	42.730000%	76.939196%	42.554482%	99.927070%
58	42.670000%	76.939174%	42.554445%	99.927070%
59	42.560000%	76.939101%	42.554394%	99.927070%
60	42.550000%	76.939083%	42.554374%	99.927070%
61	42.540000%	76.939045%	42.554356%	99.927070%
62	42.540000%	76.939011%	42.554339%	99.927070%
63	42.540000%	76.939013%	42.554335%	99.927070%
64	42.540000%	76.939005%	42.554335%	99.927070%

Table 2: Various hit rates by changing the number of mask bits determining signature_m. An LFU policy was used for these values with parameters set to N-Associatives = 4 and Cache Size = 128 kBytes.

Signature Mask Bit #	Sample	Deepsjeng	Leela	Exchange2
1-17	47.520000%	75.302526%	45.554564%	86.318316%
18	47.520000%	76.770015%	49.222116%	86.318316%
20 - 35	47.520000%	76.768995%	49.222116%	86.318316%
36 - 42	47.520000%	76.768995%	49.218510%	86.318316%
44	47.520000%	76.767120%	49.218824%	86.318316%
46	47.520000%	76.760702%	49.212382%	86.318316%
48	47.520000%	76.751613%	49.222116%	86.318316%
49	47.520000%	76.760497%	49.199272%	86.318316%
50	47.520000%	76.757622%	49.199012%	86.318316%
51	47.520000%	76.756714%	49.193357%	86.318316%
52	47.520000%	76.749485%	49.177947%	86.318316%
53	47.520000%	76.755219%	49.170410%	86.318316%
54	47.520000%	76.751826%	49.157456%	86.318316%
55	47.520000%	76.752946%	49.117310%	86.318316%
56	47.520000%	76.690455%	49.075223%	86.318316%
57	47.520000%	76.688750%	49.007673%	86.318316%
58	47.520000%	76.692675%	48.994607%	86.318316%
59	47.520000%	76.629675%	48.994607%	86.318316%
60	47.520000%	76.738588%	48.914410%	86.318316%
61	47.520000%	76.682054%	48.884678%	86.318316%
62	47.520000%	76.699903%	48.843889%	86.318316%
63	47.520000%	76.702234%	48.843889%	86.318316%
64	47.520000%	76.687639%	48.843889%	86.318316%

It seemed that the signature-based prediction policy gave a slightly increased hit rate across all workloads. This is because some set of instructions that occur sequentially may be more likely to access information again and again. This might be the case for a set of instructions that may check a flag multiple times in the same memory location awaiting an event to happen such as in a while loop condition. The benefit of this seems to only be for PC values are within rather close proximity to each other. It also seemed that this predictor worked best to improve the results of an LRU replacement policy and not much for an LFU policy.