5. Prefetching [40 points]

A processor is observed to have the following access pattern to cache blocks. Note that the addresses are **cache block addresses**, not byte addresses. This pattern is repeated for a large number of iterations.

Access Pattern P: A, A + 3, A + 6, A, A + 5

Each cache block is 8KB. The hardware has a fully associative cache with LRU replacement policy and a total size of 24KB.

None of the prefetchers mentioned in this problem employ confidence bits, but they all start out with empty tables at the beginning of the access stream shown above. Unless otherwise stated, assume that 1) each access is separated long enough in time such that all prefetches issued can complete before the next access happens, and 2) the prefetchers have large enough resources to detect and store access patterns.

(a) You have a stream prefetcher (i.e., a next-N-block prefetcher), but you don't know the prefetch degree (N) of it. However, you have a magical tool that displays the coverage and accuracy of the prefetcher. When you run a large number of repetitions of access pattern P, you get 40% coverage and 10% accuracy. What is the degree of this prefetcher (how many next blocks does it prefetch)?

Next 4 blocks.
40% coverage with a stream prefetcher for this pattern means blocks A+3 and A+6 are prefetched. Possible N at this point are 3 and 4. Accuracy 10% = 2/(N*5), so N is 4.

(b) You didn't like the performance of the stream prefetcher, so you switched to a PC-based stride prefetcher that issues prefetch requests based on the stride detected for each memory instruction. Assume all memory accesses are incurred by the *same* load instruction (i.e., the same PC value) and the initial stride value for the prefetcher is set to 0.

Circle which of the cache block addresses are prefetched by this prefetcher:

A, A + 3, $\boxed{A+6}$, A, A + 5

A, A + 3, $\boxed{A+6}$, A, A + 5

A, A + 3, $\boxed{A+6}$, A, A + 5

A, A + 3, $\boxed{A+6}$, A, A + 5

Explain:

This prefetcher remembers the last stride and applies that to prefetch the next block from the current access.

(c) Stride prefetcher couldn't satisfy you either. You changed to a Markov prefetcher with a correlation table of 12 entries (assume each entry can store a single address to prefetch, and remembers the most recent correlation). When all the entries are filled, the prefetcher replaces the entry that is least-recently accessed.

Circle which of the cache block addresses are prefetched by this prefetcher:

A, A + 3, A + 6, A, A + 5

A, A + 3, $\boxed{A+6}$, $\boxed{A}$, A + 5

$\boxed{A}$, A + 3, $\boxed{A+6}$, $\boxed{A}$, A + 5

$\boxed{A}$, A + 3, $\boxed{A+6}$, $\boxed{A}$, A + 5

Explain:

All entries are filled after the first repetition, except the entry for A+5. Accesses to A+3 and A+5 thrash each other for block A's next-block entry, so they cannot be correctly prefetched.

(d) Just in terms of coverage, after how many repetitions of access pattern P does the Markov prefetcher from part (c) start to outperform the stream prefetcher from part (a), if it can at all? Show your work.

5 repetitions.
Coverage for Markov prefetcher from part (c) is (0+2+3*(N-2))/(5N). This is equal to 40% when N=4, so it outperforms 40% at the 5th repetition. We gave full credit if you wrote either 4 or 5, depending on the work shown.

(e) You think having a correlation table of 12 entries makes the hardware too costly, and want to reduce the number of correlation table entries for the Markov prefetcher. What is the minimum number of entries that gives the same prefetcher performance as 12 entries? Similar to the last part, assume each entry can store a single next address to prefetch, and remembers the most recent correlation. Show your work.

4 entries.
With 4 different accesses, we need at least 4 entries.

(f) Your friend is running the same program on a different machine that has a Markov prefetcher with 2 entries. The same assumptions from part (e) apply.

Circle which of the cache block addresses are prefetched by the prefetcher:

A, A + 3, A + 6, A, A + 5

A, A + 3, A + 6, A, A + 5

A, A + 3, A + 6, A, A + 5

A, A + 3, A + 6, A, A + 5

Explain:

NONE. Not enough entries. Entries will be evicted before they are used again.

(g) As an avid computer architect, you decide to update the processor by increasing the cache size to 32KB with the same cache block size. Assume you will be only running a program with the same access pattern P for a large number of iterations (i.e., one trillion), describe a prefetcher that provides smaller memory bandwidth consumption than the baseline without a prefetcher. Explain:

No prefetcher. Since all lines will sit in the cache after the first iteration, the prefetcher needs to achieve 100% accuracy on the first iteration in order to consume the same amount of memory bandwidth as the baseline, which is not possible without any training on the prefetcher beforehand.
Note: We leave it up to you to think whether or not you can ever design a prefetcher that leads to less memory consumption than the baseline without a prefetcher.