RESULTS:

PART I :

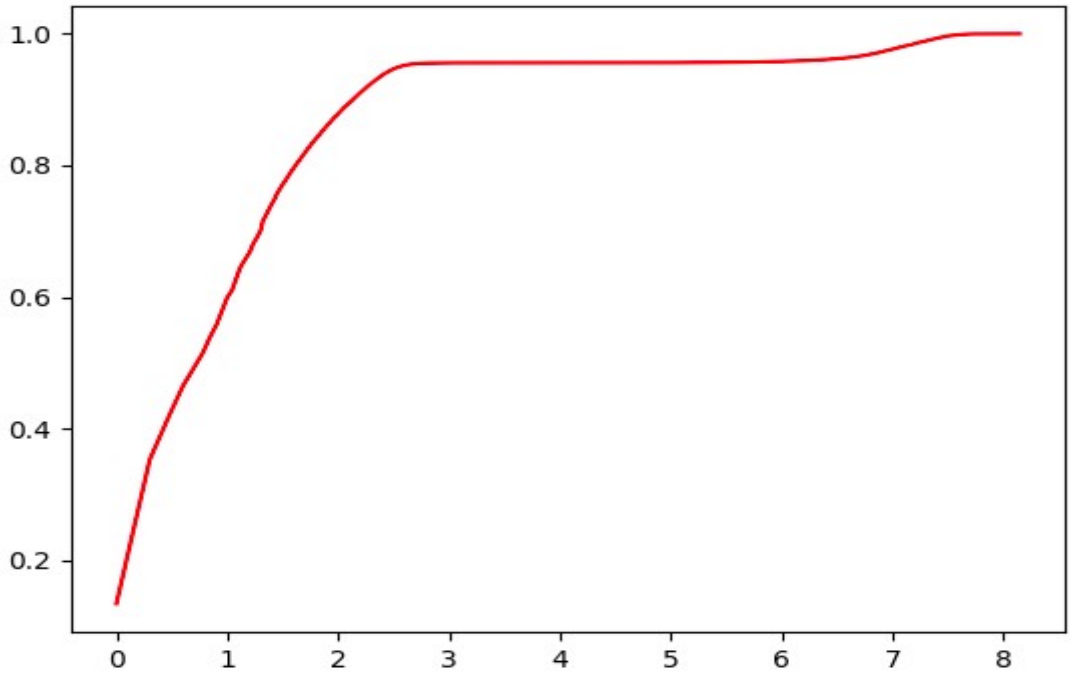
Following table summarises the machine accesses obtained from the binaries:

	Machine accesses
prog1	140526611
prog2	2514862
prog3	9478276
prog4	1065963

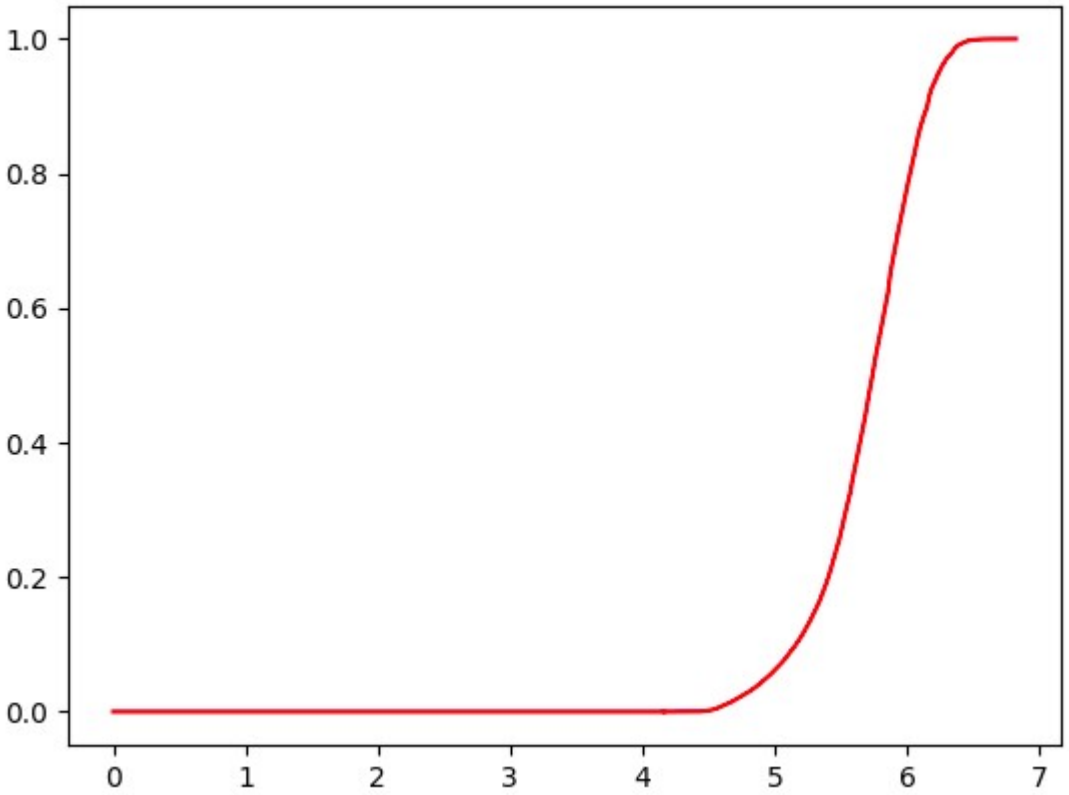
PART II and PART III :

	Number of accesses	Number of hits	Number of misses	Hit rate
prog1	140526611	133832799	6693812	0.95
prog2	2514862	2283862	231000	0.91
prog3	9478276	8838982	639294	0.93
prog4	1065963	940811	125152	0.88

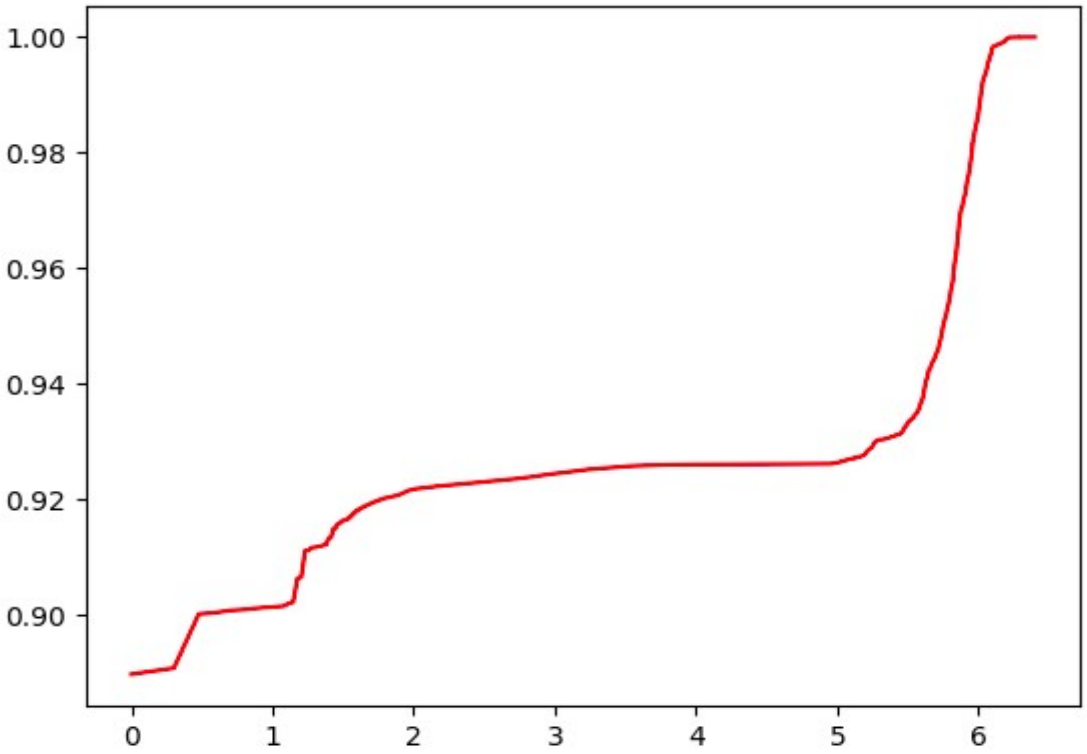
Cumulative density function plot for prog1- Plot1.1



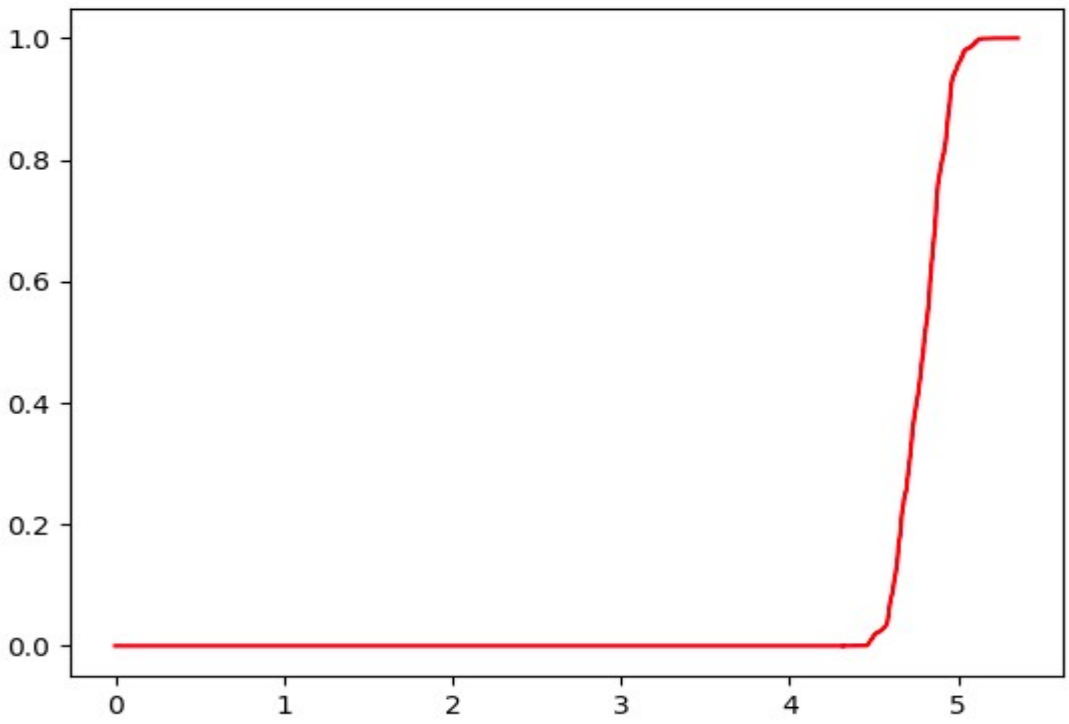
Cumulative density function plot for prog1 with LRU cache-Plot 1.2



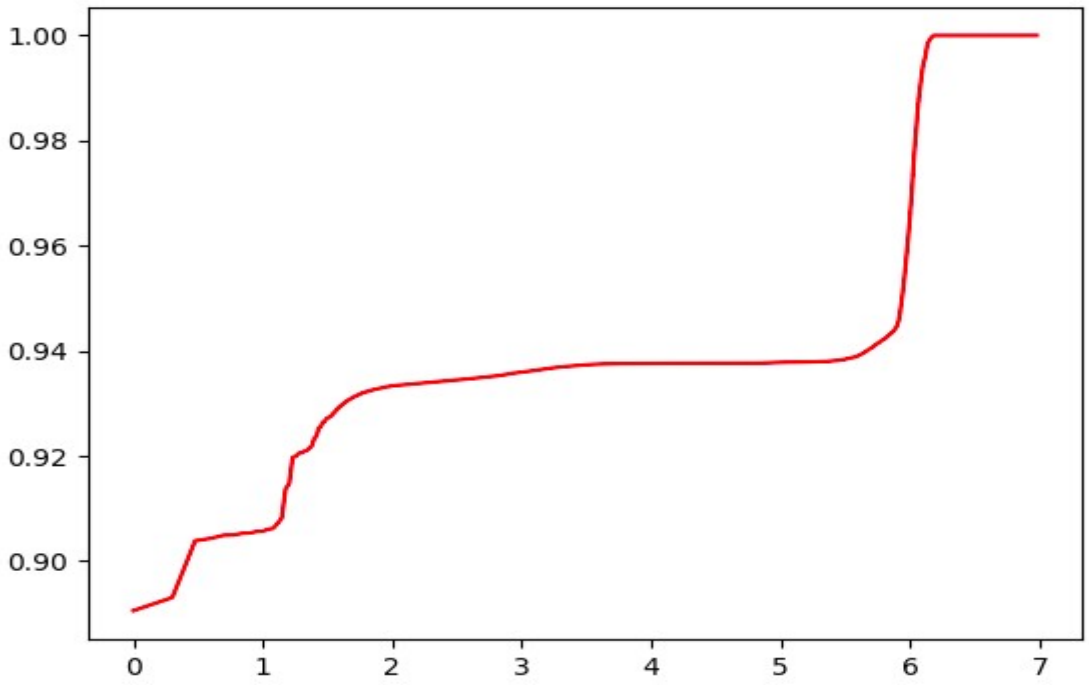
Cumulative density function plot for prog2 – Plot 2.1



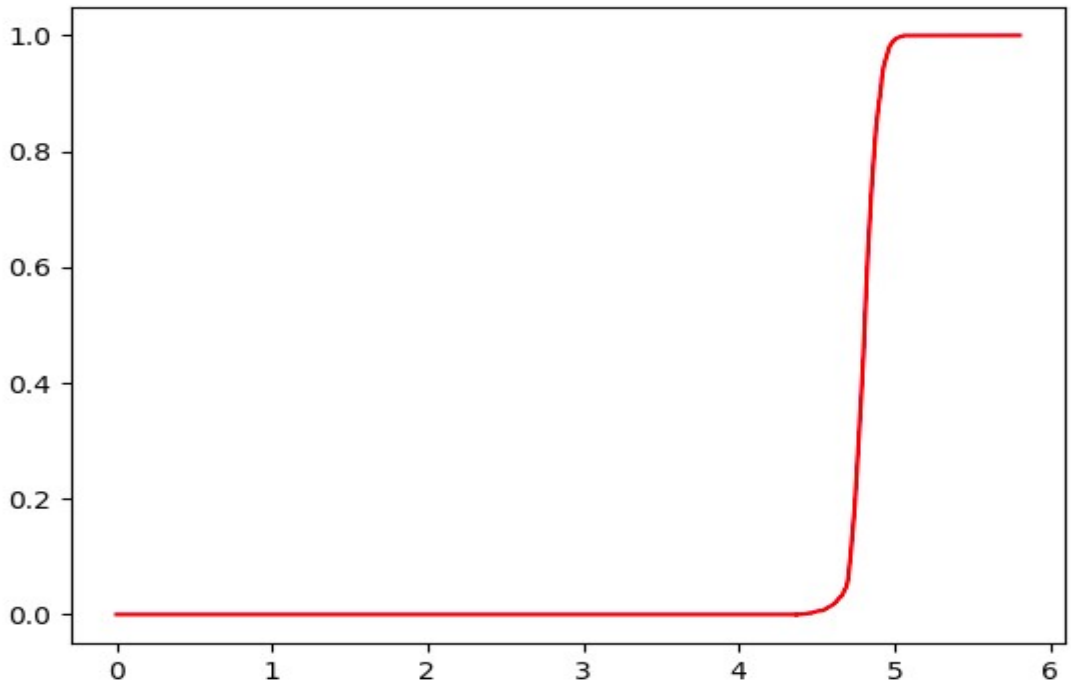
Cumulative density function plot for prog2 with LRU cache – Plot 2.2



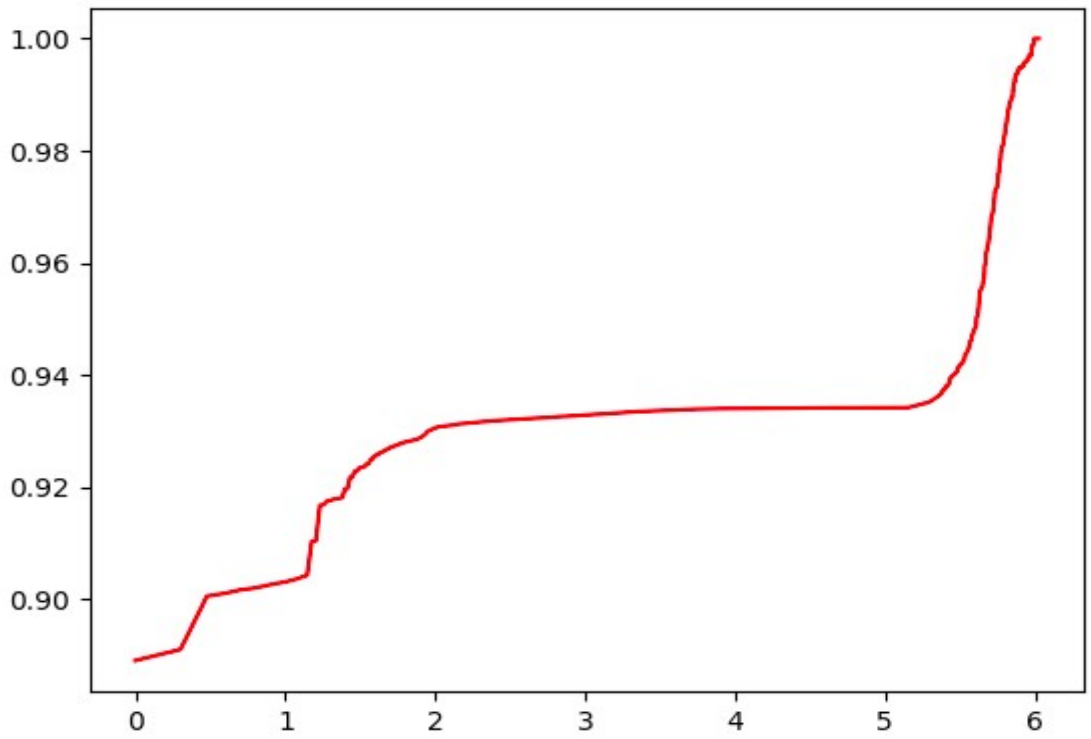
Cumulative density function plot for prog3 – Plot 3.1



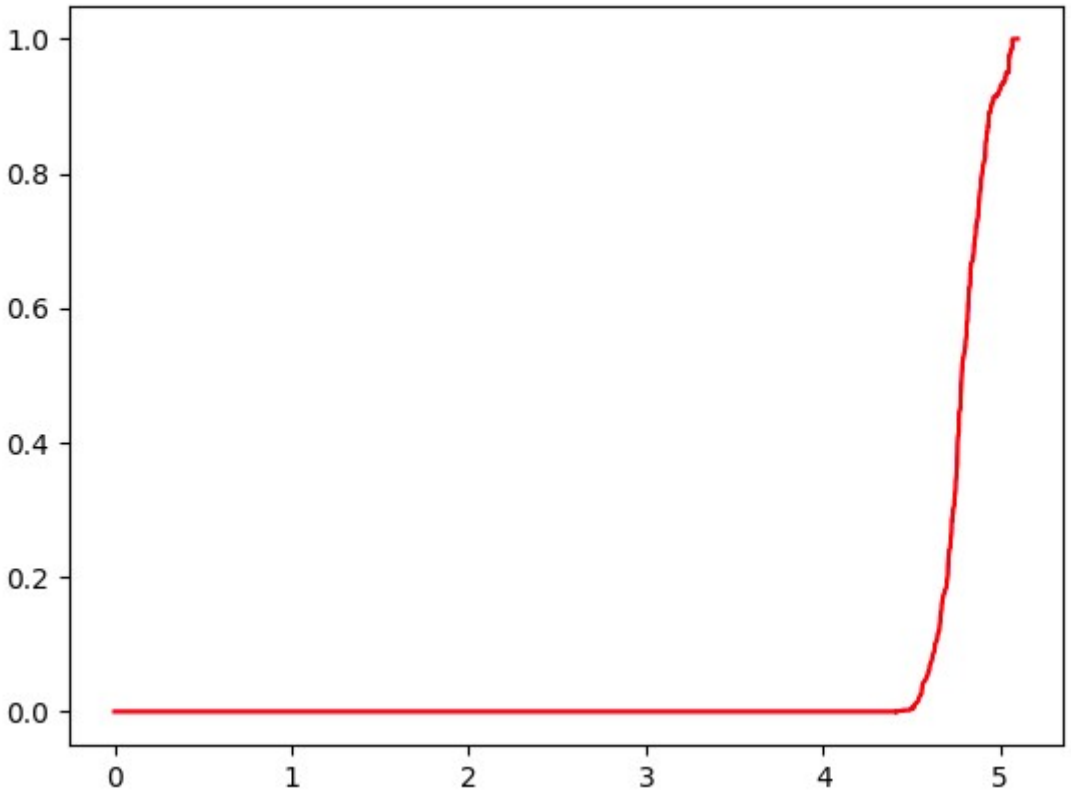
Cumulative density function plot for prog3 with LRU cache – Plot 3.2



Cumulative density function plot for prog4 – Plot 4.1



Cumulative density function plot for prog4 with LRU cache – Plot 4.2



OBSERVATIONS:

For prog1, acc dist 10^2.5 ------> F(D) = 0.9

For prog2, acc dist 10^1 ------> F(D) = 0.9

For prog3, acc dist 10^1 -------> F(D) = 0.91

For prog4, acc_dist 10^1 ------> F(D) = 0.9

If we observe the plot 2.1, plot 3.1, plot 4.1, we can infer that the prog2, prog3 and prog4 an access distance of 10 (that is, on log scale x=1) has a very high frequency (≈ 90%) indicating that these programs might have a very high degree of locality in access and hence they can get high number of cache hits, provided we use a cache.

For prog 1,access distance of 10 has a much lower frequency (≈ 50%) and shows a gradual increase in the cumulative distribution till log(x) ≈ 2.5. Thus prog1 will have relatively lesser degree of locality in access.

Plot 2.1, plot 3.1, plot 4.1 have a knee at around say,log(x) and M% in the CDF curve. This corresponds to a cache with approximately (10 ^ log(x)) blocks. Therefore, a crude upper bound of the capacity of a fully associative cache is equal to (10 ^ log(x))*64B, should be able to capture M% reuses in the prog.

The curves before and after the LRU cache implementation vary significantly. If we compare the plots 1.2, 2.2, 3.2, 4.2 ,which represent the CDF for access distances after the LRU caching , we can see that CDF is approximately 0 until log(x) ≈ 4.5. Whereas, the CDF for the access distances before the cache filter, depicted by plots 1.1, 2.1, 3.1, 4.1, have a large number of accesses with short access distance. The variation in the shapes of the curves is because accesses to blocks that have a short access distance constitute an access to a recently used block and are highly likely to be a cache hit. The CDF plot after applying the cache filter is approximately 0 until log(x) ≈ 4.5. This corresponds to an access distance of 10 ^ 4.5 ≈ 31, 622 which is approximately equal to the number of cache blocks (32,768) in the cache. We can thus conclude that most of the

misses in the cache (excluding cold misses) are likely to be capacity misses.

This can also be verified by looking at the table of hit rate, which depicts high percentage of hit.

PART IV :

Following table summarises the no. of blocks shared by the threads:

No. of blocks	prog1	prog2	prog3	prog4
Private	443	441	449	8628
Shared by 2 threads	70	8262	63	57409
Shared by 3 threads	1872	16384	0	6
Shared by 4 threads	32455	40957	0	1
Shared by 5 threads	143250	4	0	1
Shared by 6 threads	244970	0	0	0
Shared by 7 threads	173832	1	1	1
Shared by 8 threads	124529	12	65547	12
Total no. of memory blocks	721421	66061	66060	66058

OBSERVATIONS:

From the table contents given in part IV, prog1, prog2 and prog3 have approximately the same number of private memory blocks. Most of the blocks are shared among multiple threads and only a small number of variables/memory blocks are local to the threads. In contrast, a relatively larger number of memory blocks are private in prog4 indicating that a lot of accesses involves private memory blocks. Almost all of the other blocks are being shared by a maximum of 2 threads in prog4. Prog1 one has high number of shared data, which essentially means that it have large blocks which are accessed by multiple threads.

If we compare the blocks accessed privately in prog4 with the rest of the blocks accessed in prog1,prog2 and prog3, we can conclude that there is high degree of parallelism in prog4 as the large amount of blocks are getting accessed in prog4 independently and there is less dependency between the threads as they do not need to wait for completion of other threads for starting their own execution.