Code used (As provided in the problem bank 15):

The following code written in STL Language, implements Sorting of elements in an array using Bubble Sort technique.

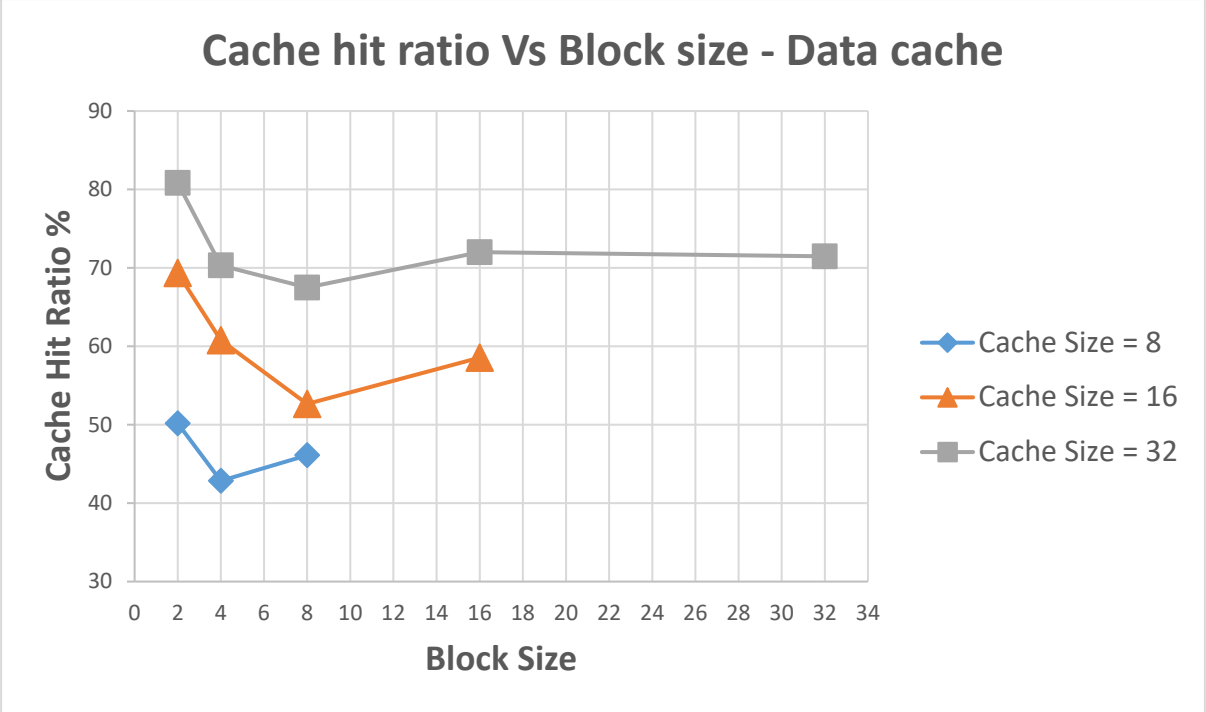
```
program BubbleSort
var a array(7) byte
a(0) = 4
a(1) = 5
a(2) = 2
a(3) = 7
a(4) = 6
a(5) = 3
a(6) = 1
var len byte
var temp byte
var l1 byte
var l2 byte
var x1 byte
var x2 byte
var j byte
var j1 byte
var k byte
var i byte
len = 7
l1 = len - 1
for k =0 to len
        write(a(k),"  ")
next
writeln("")
writeln("Bubble Sort Starts")
for i = 0 to l1
        l2 = len - i - 1
        for j=0 to l2
                j1 = j + 1
                x1 = a(j)
                x2 = a(j1)
                if x1 > x2 then
                temp = x2
                a(j1) = x1
                a(j) = temp
                end if
        next
next
for k =0 to len
        write(a(k),"  ")
next
writeln("")
writeln("Bubble Sort Ends")
end
```

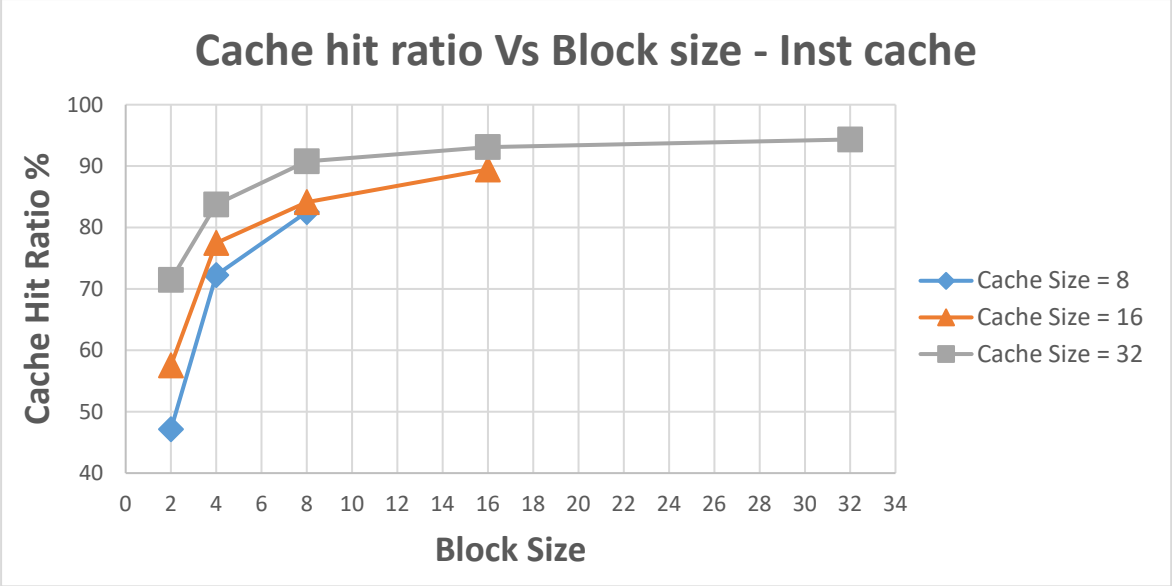
Part I: Direct Mapped Cache

a) *Execute the above program by setting block size to 2, 4, 8, 16 and 32 for cache size = 8, 16 and 32. Record the observation in the following table.*

		Data cache				Instruction cache			
Block Size	**Cache size**	**# Hits**	**# Misses**	**% Miss Ratio**	**%Hit Ratio**	**# Hits**	**# Misses**	**% Miss Ratio**	**%Hit Ratio**
2	**8**	568	564	49.82	50.18	675	757	52.86	47.14
4		485	647	57.16	42.84	1033	397	27.76	72.24
8		522	610	53.89	46.11	1181	250	17.47	82.53
2	**16**	785	347	30.65	69.35	823	607	42.45	57.55
4		688	444	39.22	60.78	1108	322	22.52	77.48
8		596	536	47.35	52.65	1203	227	15.87	84.13
16		647	458	41.45	58.55	1279	151	10.56	89.44
2	**32**	915	217	19.17	80.83	1022	408	28.53	71.47
4		796	336	29.68	70.32	1196	232	16.25	83.75
8		764	368	32.51	67.49	1298	132	9.23	90.77
16		815	317	28.00	72.00	1322	98	6.90	93.10
32		809	323	28.53	71.47	1349	81	5.66	94.34

b) *Plot a single graph of Cache hit ratio Vs Block size with respect to cache size = 8, 16 and 32. Comment on the graph that is obtained.*





For the data cache, the hit ratio generally increases as the cache size increases but for a given cache size, as the block size increases, there is a dip in the hit ratio and as it increased further, hit ratio improved and stabilized.

For the instruction cache, the trend is similar, i.e., the hit ratio generally increases as the cache size increases but the hit ratios are generally lower compared to the data cache

Overall, these results suggest that larger cache sizes can improve the hit ratio for both the data cache and instruction cache and instruction cache has lower hit ratios compared to the data cache

c) *Fill the below table and write a small note on your observation from **data cache**.*
- *Block Size =8*
- *Cache Size = 8*
- *Cache Type = Direct Mapped*

Addresses	Data	Miss (%)
0112	74	0
0113	20	50
0114	45	67
0115	6E	75
0116	64	80
0117	73	83
0118	00	86
0119	00	88

As the data is loaded into the cache, the miss rate starts at 100% because the cache is empty initially. When the data at address 0112 is loaded into the cache, the miss rate becomes 0% because the data was found in the cache. When the data at address 0113 is requested, it results

in a miss because it is not in the cache yet. The miss rate then becomes 50% because there have been 2 requests, and 1 of them resulted in a miss.

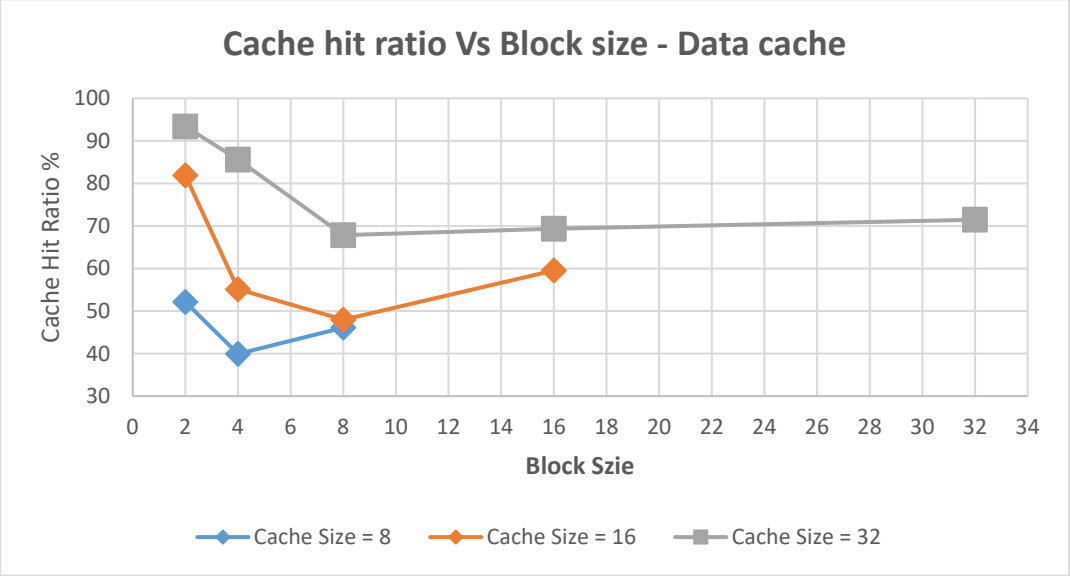
The miss rate continues to fluctuate as more data is loaded into the cache, but it should eventually stabilize as the cache becomes full and the probability of a miss decreases. In this case, the miss rate eventually stabilizes at around 88% because the cache size is too small to hold all of the data being requested.

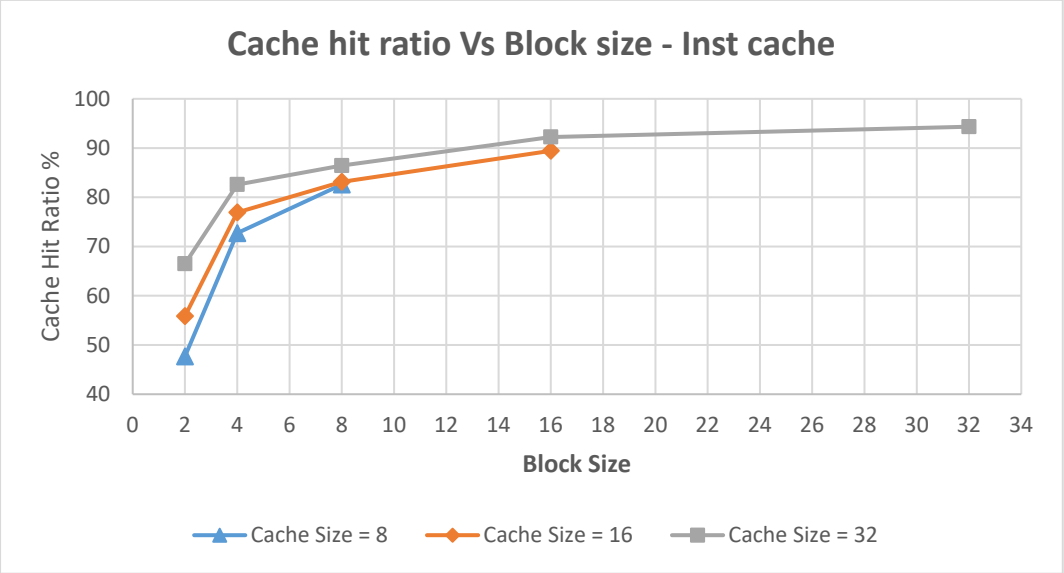
Part II: Associative Mapped Cache

a) Execute the above program by setting block size to 2, 4, 8, 16 and 32 for cache size = 8, 16 and 32. Record the observation in the following table.

LRU Replacement Algorithm									
Data Cache						Instruction Cache			
Block Size	**Cache size**	**# Hits**	**# Misses**	**% Miss Ratio**	**%Hit Ratio**	**# Hits**	**# Misses**	**% Miss Ratio**	**%Hit Ratio**
2	**8**	590	542	47.88	52.12	682	748	52.31	47.69
4		452	680	60.07	39.93	1040	390	27.27	72.73
8		522	610	53.89	46.11	1181	249	17.41	82.59
2	**16**	927	205	18.11	81.89	799	631	44.13	55.87
4		624	508	44.88	55.12	1100	330	23.08	76.92
8		543	589	52.03	47.97	1189	241	16.85	83.15
16		674	458	40.46	59.54	1279	151	10.56	89.44
2	**32**	1057	75	6.63	93.37	951	479	33.50	66.50
4		969	163	14.40	85.60	1181	249	17.41	82.59
8		768	364	32.16	67.85	1236	194	13.57	86.43
16		785	347	30.65	69.35	1319	111	7.76	92.24
32		809	323	28.54	71.47	1349	81	5.66	94.34

b) Plot a single graph of Cache hit ratio Vs Block size with respect to cache size = 8, 16 and 32. Comment on the graph that is obtained.





For the data cache, the hit ratio generally increases as the cache size increases for associative mapping but for a given cache size, as the block size increases, there is a dip in the hit ratio and as it increased further, hit ratio improved and stabilized.

For the instruction cache, the trend is similar but the hit ratios are generally lower compared to the data cache for associative mapping

Overall, these results suggest that larger cache sizes can improve the hit ratio, for both the data cache and instruction cache when using associative mapping and the instruction cache still has lower hit compared to the data cache, indicating that it may be less effective at finding the instructions it needs in its cache.

c) *Fill up the following table for three different replacement algorithms and state which replacement algorithm is better and why?*

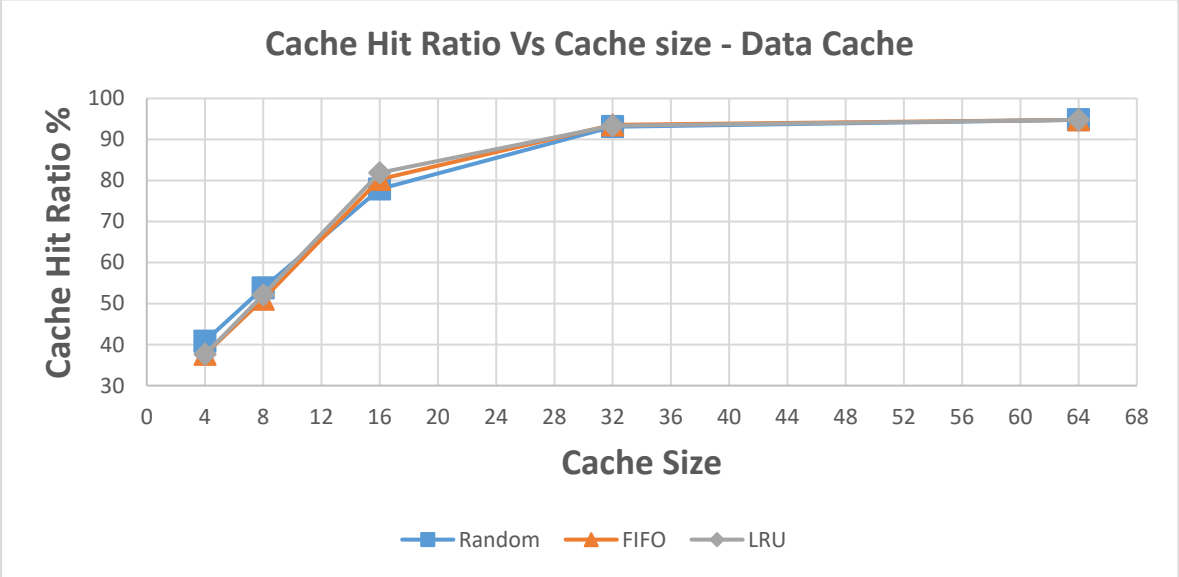
Data Cache					Instruction Cache				
Replacement Algorithm: Random					**Replacement Algorithm: Random**				
Block Size	**Cache size**	**Miss**	**Hit**	**% Hit ratio**	**Block Size**	**Cache size**	**Miss**	**Hit**	**% Hit ratio**
2	**4**	670	462	40.81	**2**	**4**	757	673	47.06
2	**8**	523	609	53.80	**2**	**8**	737	693	48.46
2	**16**	250	882	77.91	**2**	**16**	621	809	56.57
2	**32**	78	1054	93.11	**2**	**32**	397	1033	72.24
2	**64**	59	1073	94.79	**2**	**64**	112	1318	92.17
Replacement Algorithm: FIFO					**Replacement Algorithm: FIFO**				
Block Size	**Cache size**	**Miss**	**Hit**	**Hit ratio**	**Block Size**	**Cache size**	**Miss**	**Hit**	**Hit ratio**
2	**4**	706	426	37.63	**2**	**4**	757	673	47.06
2	**8**	554	578	51.06	**2**	**8**	748	682	47.69
2	**16**	223	909	80.30	**2**	**16**	631	799	55.87
2	**32**	73	1059	93.55	**2**	**32**	482	948	66.29
2	**64**	59	1073	94.79	**2**	**64**	77	1353	94.61
Replacement Algorithm: LRU					**Replacement Algorithm: LRU**				
Block Size	**Cache size**	**Miss**	**Hit**	**Hit ratio**	**Block Size**	**Cache size**	**Miss**	**Hit**	**Hit ratio**
2	**4**	706	426	37.63	**2**	**4**	757	673	47.06
2	**8**	542	590	52.12	**2**	**8**	748	682	47.69
2	**16**	205	927	81.89	**2**	**16**	631	799	55.87
2	**32**	75	1057	93.37	**2**	**32**	479	951	66.50
2	**64**	59	1073	94.79	**2**	**64**	77	1353	94.61

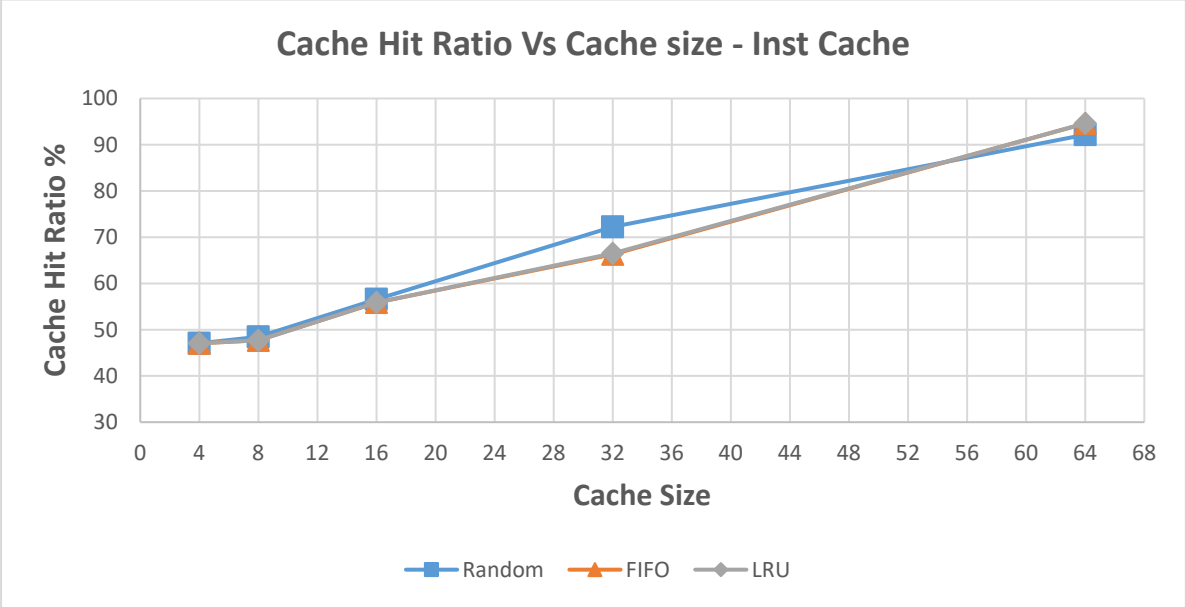
From the record above for data cache, we can see that, at a cache size of 8 and a block size of 2, the hit ratio is 53.80% with a random replacement algorithm, 51.06% with a FIFO replacement algorithm, and 52.12% with an LRU replacement algorithm. Similarly, at a cache size of 32 and a block size of 2, the hit ratio is 93.11% with a random replacement algorithm, 93.55% with a FIFO replacement algorithm, and 93.37% with an LRU replacement algorithm.

Also, for instruction cache, at a cache size of 8 and a block size of 2, the hit ratio is 47.06% with a random replacement algorithm, 47.69% with a FIFO replacement algorithm, and 47.69% with an LRU replacement algorithm. Similarly, at a cache size of 32 and a block size of 2, the hit ratio is 72.24% with a random replacement algorithm, 66.29% with a FIFO replacement algorithm, and 66.50% with an LRU replacement algorithm.

These observations suggest that larger cache sizes and the use of the LRU replacement algorithm can improve the hit for both the data cache and instruction cache.

d) *Plot the graph of Cache Hit Ratio Vs Cache size with respect to different replacement algorithms. Comment on the graph that is obtained.*





For the data cache, the hit ratio generally increases as the cache size increases and the replacement algorithm changes from random to FIFO to LRU

For the instruction cache, the trend is similar but the hit ratios are generally lower compared to the data cache. The hit ratio also tends to be slightly higher with the LRU replacement algorithm compared to the FIFO and random algorithms.

Part III: Set Associative Mapped Cache

Execute the above program by setting the following Parameters:

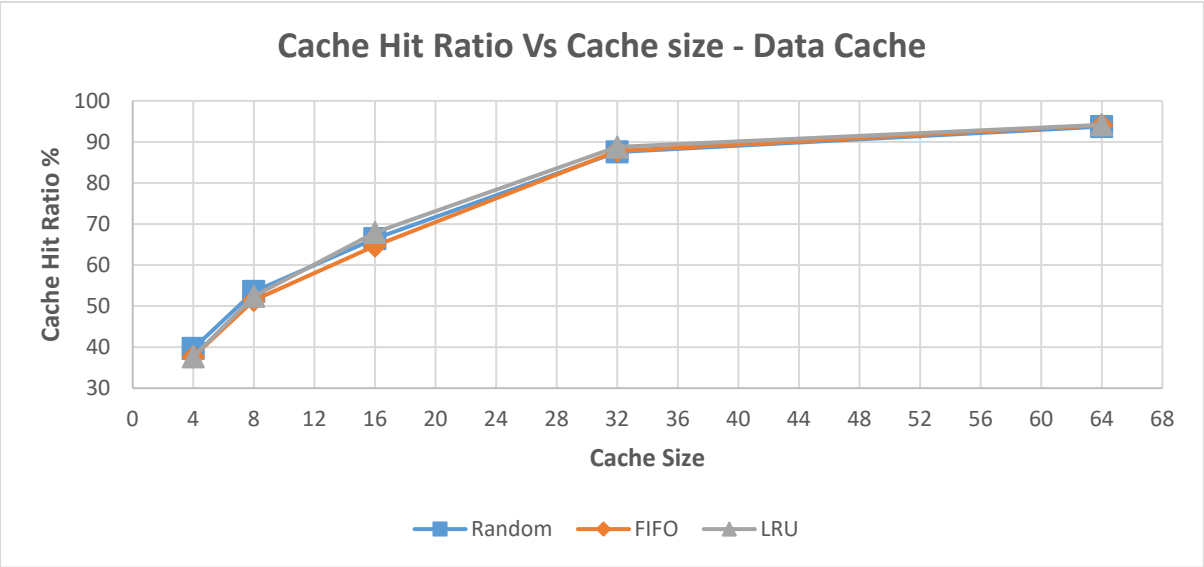
- *Number of sets (Set Blocks): 2 way*
- *Cache Type: Set Associative*
- *Replacement: LRU/FIFO/Random*

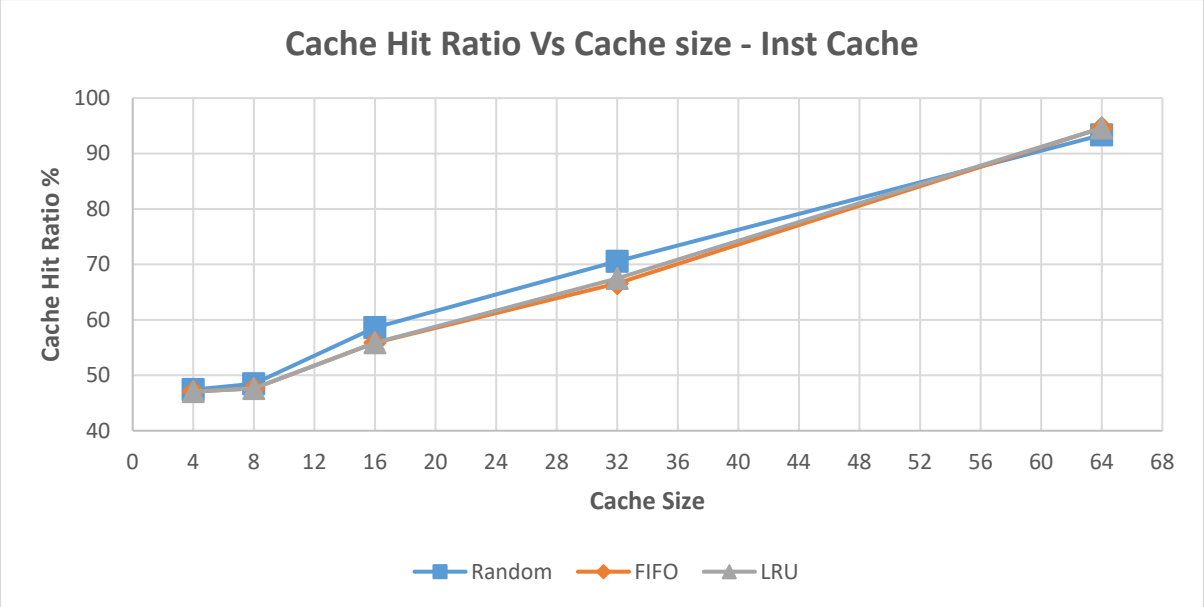
a) *Fill up the following table for three different replacement algorithms and state which replacement algorithm is better and why?*

DATA CACHE					INSTRUCTION CACHE				
Replacement Algorithm: Random					**Replacement Algorithm: Random**				
Block Size	**Cache size**	**Miss**	**Hit**	**Hit ratio**	**Block Size**	**Cache size**	**Miss**	**Hit**	**Hit ratio**
2	**4**	683	449	39.66	**2**	**4**	752	678	47.41
2	**8**	526	606	53.53	**2**	**8**	737	693	48.46
2	**16**	380	752	66.43	**2**	**16**	592	838	58.60
2	**32**	141	991	87.54	**2**	**32**	421	1009	70.566
2	**64**	71	1061	93.72	**2**	**64**	95	1335	93.36
Replacement Algorithm: FIFO					**Replacement Algorithm: FIFO**				
Block Size	**Cache size**	**Miss**	**Hit**	**Hit ratio**	**Block Size**	**Cache size**	**Miss**	**Hit**	**Hit ratio**
2	**4**	706	426	37.63	**2**	**4**	757	673	47.06
2	**8**	550	582	51.41	**2**	**8**	749	681	47.62
2	**16**	400	732	64.66	**2**	**16**	631	799	55.87
2	**32**	138	994	87.81	**2**	**32**	478	952	66.57
2	**64**	67	1065	94.08	**2**	**64**	77	1353	94.62
Replacement Algorithm: LRU					**Replacement Algorithm: LRU**				
Block Size	**Cache size**	**Miss**	**Hit**	**Hit ratio**	**Block Size**	**Cache size**	**Miss**	**Hit**	**Hit ratio**
2	**4**	706	426	37.63	**2**	**4**	757	673	47.06
2	**8**	539	593	52.38	**2**	**8**	749	681	47.62
2	**16**	363	769	67.93	**2**	**16**	631	799	55.87
2	**32**	127	1005	88.78	**2**	**32**	465	965	67.48
2	**64**	66	1066	94.17	**2**	**64**	77	1353	94.62

Based on the Hit ratio for the three algorithms, we can observe that as the cache size increases, the hit ratio also increases for the given parameters. Also, at the higher cache sizes, the hit ratio for FIFO and LRU are almost same (e.g., at cache size 64 and block size 2, hit ratio is ~ 94%), but at lower cache sizes, (cache size 8 and 16 and block size 2), hit ratios are fairly better for random and LRU than FIFO. Hence, for all cache sizes, best replacement algorithm would be LRU

b) Plot the graph of Cache Hit Ratio Vs Cache size with respect to different replacement algorithms. Comment on the graph that is obtained.





For the data cache, the hit ratio generally increases as the cache size increases.

For the instruction cache, the trend is similar but the hit ratios are generally lower

Overall, these results suggest that larger cache sizes and the use of the LRU replacement algorithm can improve the hit ratio (percentage of successful cache accesses) for both the data cache and instruction cache. However, the instruction cache still has lower hit ratios.

c) Fill in the following table and analyse the behaviour of Set Associate Cache. Which one is better and why?

DATA CACHE					INSTRUCTION CACHE				
Replacement Algorithm: LRU									
Block Size, Cache size	Set Blocks	Miss	Hit	Hit ratio	Block Size, Cache size	Set Blocks	Miss	Hit	Hit ratio
2, 64	2 – Way	66	1066	94.17	2, 64	2 – Way	77	1353	94.61
2, 64	4 – Way	67	1065	94.08	2, 64	4 – Way	77	1353	94.61
2, 64	8 – Way	67	1065	94.08	2, 64	8 – Way	77	1353	94.61

For the data cache, the hit ratio is relatively consistent regardless of the set block size, with values of 94.17%, 94.08%, and 94.08% for 2-way, 4-way, and 8-way set blocks, respectively.

For the instruction cache, the hit ratio is also relatively consistent regardless of the set block size, with values of 94.61%, 94.61%, and 94.61% for 2-way, 4-way, and 8-way set blocks, respectively.

Based on these observations, all the set blocks performed same and hit ratio is independent of the set blocks for the block size 2, cache size 64