2.

a) <u>Direct Mapping</u>

Mapping algorithm K%N is used as K is the cache line number and N is the number of cache slots. The number of cache slots can be calculated as N = cache memory/1 cache slot size = (1024*1024)/512 = 2048. Therefore, there is a total of 2048 cache slots.

First Loop - 1st Execution

Initially, the first 550 cache lines of main RAM memory is contained in the cache. These cache lines were mapped using the mapping algorithm K%N. Therefore, cache lines 0 to 549 are mapped to cache slots 0 to 549.

- Cache Slot Number = 0 % 2048 = 0
- Cache Slot Number = 549 % 2048 = 549

However, the needed code and data for the first loop is stored in cache lines 400 to 950. Since cache lines up to 549 have already been stored in the cache memory, cache lines 400 to 549 will have a hit for every access. It is stated that there are 5 accesses per cache line so the cache hits can be computed as followed:

- Cache Hits = (549 - 400 + 1) * 2 = 750

Initially, cache lines 550 to 950 are located in the cache memory so the first access to each of these cache lines will be one miss.

- Cache Misses = (950 - 550 + 1) = 401

Then, cache lines 550 to 950 will be loaded into cache slots 550 to 950 in numerical order since these cache slots are empty as well using the direct mapping algorithm. Since the needed code and data for lines 550 to 950 are now located in the cache memory, there will be cache hits for the remainder of accesses per line, which is 4.

- Cache Hits = (950 - 550 + 1) * 4 = 1604

First Loop - 2nd to 8th Execution

Now that cache lines 0 to 950 have been mapped to cache slots 0 to 950 in consecutive order, the code and data in cache lines 400 to 950 is now held in the cache memory. Therefore, since each cache line has 5 accesses there will be a hit for each access. After the first execution, this loop is executed 7 more times so the cache hits will be multiplied by 7.

- Cache Hits = (950 - 400 + 1) * 5 * 7 = 19285

Second Loop - 1st Execution

For the second loop, the data and code are stored in cache lines 4666 to 5028 and 16968 to 17314 with 3 accesses per line. For the first execution of the second loop, these cache lines will be mapped into cache slots. Recall, the mapping algorithm is K%N with **N = 2084**.

For cache lines 4666 to 5028 the mapped cache slots will be cache slots 570 to 932.

- 4666 % 2084 = 570
- 5028 % 2048 = 932

For cache lines 16968 to 17314 the mapped cache slots are slots 584 to 930.

- 16968 % 2048 = 584
- 17314 % 2048 = 930

However, in the first loop, cache slots 570 to 932 have already been mapped with cache lines 570 to 932. Therefore, for the first execution of the second loop, there will be misses for the cache slots as cache lines 4666 to 5028 have not been mapped yet.

- Cache Misses = (932 - 570 + 1) = 363

Similarly, cache lines 16968 to 17314 have also not been mapped to their corresponding slots of 584 to 930 so the first access to these cache lines will be misses as well.

- Cache Misses = (930 - 584 + 1) = 347

Since, there is 3 accesses per cache line, there are 2 accesses left for these cache lines and therefore 2 cache hits.

- Cache Hits = (363 * 2) + (347 * 2) = 1420

Second Loop - 2nd to 43rd Execution

Since cache lines 4666 to 4679, 5027, and 5028 have been mapped to cache slots 570 to 583, 931, and 932, these cache slots are never replaced. Therefore, there will be cache hits for all 3 accesses.

- Cache Hits = ((583 -570 + 1) + 1 + 1) * 3 = 48

However, for cache slots 584 to 930, which currently contain cache lines 16968 to 17314, cache lines 4680 to 5026 code and data are needed as the second execution starts. Therefore, the first access will be a miss, then lines 4680 to 5026 will be mapped to these slots and the remaining 2 accesses will be hits. Then cache lines 16968 to 17314 will replace cache lines 4680 to 5026, experience 1 miss and two hits as well. Cache lines 16968 to 17314 and 4680 to 5026 will continuously replace each other once during each execution. This results in 2 cache misses and 4 cache hits per cache line.

- Cache Misses = (930 - 584 + 1) * 2 = 694
- Cache Hits = (930 - 584 + 1) * 2 * 2 = 1388

After the first execution, there will be 42 executions.

TOTALS:

- Cache Hits = 750 + 1604 + 19285 + 1420 + (48 + 1388) * 42 = 83371
- Cache Misses = 401 + 363 + 347 + 694 * 42 = 30259
- Hit Ratio = (Total Hits)/(Total Hits and Misses) = 83371 / (83371 + 30259) = 0.7337

b) 4 Associative Mapping

Loop 1 - 1st Execution

Since the algorithm is K%P=Q and P*M=N with M=4, N=2048, P is 512 cache slot groups. Then, the cache slot number is equal to Q*4±0,1,2,3. Cache line K will be mapped to cache group K%512. Initially, cache lines 0 to 549 will be mapped to the cache memory using 4 way associative mapping. Therefore, cache lines 0 to 511 will be placed in cache slot groups 0 to 511 as well.

- 0 % 512 = 0
- 511 % 512 = 511

Since these cache groups are originally empty, these cache lines will be placed in the first cache slot of their group. Therefore, the cache slot number is their cache group multiplied by 4. This means cache line 0 is in cache slot 0, 1 is cache slot 4, 2 is in cache slot 8 and so on to 511 in cache slot 2044. Then, cache lines 512 to 549 will be mapped in cache slot groups 0 to 37 in the second slot.

- 512 % 512 = 0
- 549 % 512 = 37

Cache lines 512 to 549 are mapped in the second slot of their cache group. Hence, cache line 512 is in cache slot 0*4+1=1 and cache line 549 which is in group 37, cache slot 37*4+1=149.
Since cache lines 0 to 549 are initially in the cache slots, there will be hits for all 5 accesses in the first loop.

- Cache Hits = (549 - 400 + 1) * 5 = 750

Then for cache lines 550 to 950, there will be misses for the first access of the cache line

- Cache Misses = (950 - 550 + 1) = 401

These cache lines 550 to 950 will be mapped using the 4 way associative mapping.

- Cache Group = 550 % 512 = 38
- Cache Group = 950 % 512 = 438

However, since cache groups 38 to 438 have cache lines 38 to 438 mapped in these groups, cache lines 550 to 950 will be placed in the smallest unoccupied cache slot in their group which is the second slot. The equation to get the number of the cache slot when the cache line is placed in the second slot is

- Cache Slot Number = (Group Q * 4) + 1

Therefore, cache line 550 is placed in cache slot 153, (38 * 4) + 1 = 153, cache line 551 is placed in cache slot 157, (39 * 4)+ 1 = 157, and this will continue to cache line 950 as cache slot 1753, (438 * 4) + 1 = 1753.
Since the initial access was a miss, there are 4 remaining hits for cache lines 550 to 950.

- Cache Hits = 401 * 4 = 1604

First Loop - 2nd to 8th Execution
Since cache lines 400 to 950 are now in cache, all 5 accesses will be hits and will be executed 7 times.

- Cache Hits = (950 - 400 + 1) * 5 * 7 = 19285

Second Loop - 1st Execution
The first accesses for cache lines 4666 to 5028 and lines 16968 to 17314 will be misses. Then, these cache lines will be mapped to the cache memory.

- 4666 % 512 = 58
- 5028 % 512 = 420
- 16968 % 512 = 72
- 17314 % 512 = 418

Since two slots in group 58 to 420 have already been taken, these cache lines will be placed in the third slot of groups 58 to 420. The equation for the cache slot when the cache line is placed in the third slot of the group is

- Cache Slot Number = (Group Q*4) + 2

For the fourth slot of the group:

- Cache Slot Number = (Group Q*4) + 3

Therefore, cache line 4666 will be placed in cache slot (58*4)+2=534 which is the third slot of group 58. Group 58 initially held cache lines 58 in cache slot 232 and 570 in cache slot 233. This will numerically continue up to cache line 5028 which will be placed in cache slot 420*4+2=1682. Likewise, cache lines 16968 to 17314 will be placed in cache groups 72 to 418 which will be located in the 4th slot of these cache groups since the first 3 are taken. Therefore, cache line 16968 is in cache line 72*4+3=291. Group 72 has already mapped cache lines 72 to cache slot 288, 584 to cache slot 289, and 4680 to cache slot 290 in sequential order. This will continue up to cache line 17314 where it will be placed in cache slot 1675.

- Cache Misses = (5028 - 4666 + 1) + (17314 - 16968 + 1) = 363 + 347 = 710
- Cache Hits = 710 * 2 = 1420

Second Loop – 2nd to 43rd Execution

Since all the necessary code and data have been mapped to the correct cache slots, there is no cache slots that need to be overwritten when the second loop is being executed. Therefore, there will be hits for all 3 accesses of cache lines 4666 to 5028 and lines 16968 to 17314 and it will be executed 42 times.

- Cache Hits = ((420 - 58 + 1) + (418 - 72 + 1)) * 3 * 42 = 89460

TOTALS:

- Cache Hits = 21639 + 90880 = 112519
- Cache Misses = 410 + 710 = 1111
- Hit Ratio = 112519 / (112519 + 1111) = 0.9902

Part A

Current Time (ms)	Current Job				Ready Queue (Front - Back)	Notes
	A	B	C	D		
2-8			36ms - 30ms		D	D arrives at 7ms
8-11				9ms - 6ms	C	D leaves CPU at 11ms
11-17			30ms - 24ms		D A	D prints for 2ms, arrives back in front of ready queue at 13ms and takes presedence over A which arrived at 12ms with e.ii (Unfinished Quanta)
17-20				6ms - 3ms	A C B	B arrives at 19ms
20-26	28ms - 22ms				C B D	D prints until 22ms and then sent to the end of ready queue
26-32			24ms - 18ms		B D A	
32-38		34ms - 28ms			D A C	
38-41				**3ms - 0ms**	A C B	D prints until 43ms and then finishes
41-47	22ms - 16ms				C B	
47-53			18ms - 12ms		B A	
53-59		28ms - 22ms			A C	
59-65	16ms - 10ms				C B	
65-71			12ms - 6ms		B A	
71-77		22ms - 16ms			A C	
77-83	10ms - 4ms				C B	
83-89			**6ms - 0ms**		B A	C finishes at 89ms
89-95		16ms - 10ms			A	
95-99	**4ms - 0ms**				B	A finishes at 99ms
99-105		10ms - 4ms			-	
105-109		**4ms - 0ms**			-	B finishes at 109ms

Job	Arrival (ms)	Finish (ms)	Turn Around Time (Finish - Arrival)
A	12	99	99 - 12 = 87ms
B	19	109	109 - 19 = 90ms
C	2	89	89 - 2 = 87ms
D	7	43	43 - 7 = 36ms

Average Turn Around Time For All Processes = (87 + 90 + 87 + 36) / 4 = 75ms

Part B

Current Time (ms)	Current Job				High Priority Queue (Front - Back)	Low Priority Queue (Front - Back)	Notes
	A	B	C	D			
2-7			36ms - 31ms		D	-	D arrives at high priority queue at 7ms
7-10				9ms - 6ms	-	C	D kicks out C and sends C back to front of low priority queue so D can start using it's time in CPU at 7ms
10-12			31ms - 29ms		D	A	D prints for 2ms, arrives back in front of high priority queue at 12ms, A arrives at low priority queue at 12ms
12-15				6ms - 3ms	-	C A	D kicks out C and put back to the front of the low priority queue to finish it's remaing 3ms in CPU
15-17			29ms - 27ms		D	A	D prints for 2ms, arrives back in front of high priority queue at 17ms
17-20				**3ms - 0ms**	B	C A	D kicks out C and sends C back to front of low priority queue so D can start using it's time in CPU at 17ms
20-26		34ms - 28ms			-	C A	D prints for 2ms and finishes at 22ms
26-32		28ms - 22ms			-	C A	
32-38		22ms - 16ms			-	C A	
38-44		16ms - 10ms			-	C A	
44-50		10ms - 4ms			-	C A	
50-54		**4ms - 0ms**			-	C A	B finishes at 54ms
54-60			27ms - 21ms		-	A	
60-66	28ms - 22ms				-	C	
66-72			21ms - 15ms		-	A	
72-78	22ms - 16ms				-	C	
78-84			15ms - 9ms		-	A	
84-90	16ms - 10ms				-	C	
90-96			9ms - 3ms		-	A	
96-102	10ms - 4ms				-	C	
102-105			**3ms - 0ms**		-	A	C finishes at 105ms
105-109	**4ms - 0ms**				-	-	A finishes at 109ms

Job	Arrival (ms)	Finish (ms)	Turn Around Time (Finish - Arrival)
A	12	109	109 - 12 = 97ms
B	19	54	54 - 19 = 35ms
C	2	105	105 - 2 = 103ms
D	7	22	22 - 7 = 15ms

Average Turn Around Time For All Processes = (97 + 35 + 103 + 15) / 4 = 62.5ms

Part C

Current Time (ms)	Current Job				High Priority Queue (Front - Back)	Low Priority Queue (Front - Back)	Notes
	A	B	C	D			
2-38			**36ms - 0ms**		D B	A	D arrives at 7ms, A arrives at 12ms, B arrives at 19ms but B goes in front of A because of higher priority and before D since they have same priority and came after D, C finishes at 38ms
38-41				9ms - 6ms	B	A	D prints for 2ms, arrives back in front of ready queue at 43ms and takes presedence over A with e.ii (Unfinished Quanta)
41-75		**34ms - 0ms**			D	A	B finishes at 75ms
75-78				6ms - 3ms	-	A	D prints for 2ms and sent back to the front of the ready queue
78-106	**28ms - 0ms**				D	-	A finishes at 106ms
106-109				**3ms - 0ms**	-	-	D prints for 2ms and finishes at 111ms

Job	Arrival (ms)	Finish (ms)	Turn Around Time (Finish - Arrival)
A	12	106	106 - 12 = 94ms
B	19	75	75 - 19 = 56ms
C	2	38	38 - 2 = 36ms
D	7	111	111 - 7 = 104ms

Average Turn Around Time For All Processes = (94 + 56 + 36 + 104) / 4 = 72.5ms