CS422 Assignment1

Name: Manish Roll no. 190477

Question1:

Task1:

Number of warmup and simulation instructions are given as command line argument.
Total Instructions: 356860 Warmup Instruction: 343107 Simulation Instruction: 13753

IPC: 0.251274

Task2:

Subtask1

Number of warmup and simulation instructions are given as global variables in champsim_tracer.cpp which we pass through the trace file to the Champsim.
Total Instructions: 356862 Warmup Instruction: 343107 Simulation Instruction: 13755

IPC: 0.251201

Subtask2

Whether an instruction is warmup or simulation is encoded into the instruction using an extra bit.
Total Instructions: 356878 Warmup Instruction: 343107 Simulation Instruction: 13771

IPC: 0.25547

Task3:

Warmup and simulation instructions are given by the user program. Result given below is when we put warmup_start() just after main function and warmup pause() where we put exit(0) just after it simulation_start() and at end simulation_pause() in demo.c.

IPC: 0.20854
For Capturing the function name from the demo, I used the routine function given in one of the examples provided in intel pin documentation.
Link: Example Link

Here we can see Task1, Task2.1 and Task2.2 have almost similar IPC which we expect. But In Task3 our IPC differ significantly which I think is because when I printed how many instructions are in different sections, I got:
Before Warmup: 147984
Warmup total: 194924
Simulation total:12832
After Simulation: 1236
I put warmup_start() at start of main. But We can see there are 147984 instructions executing before that. It's okay as these are warmups. I put simulation pause at the end of

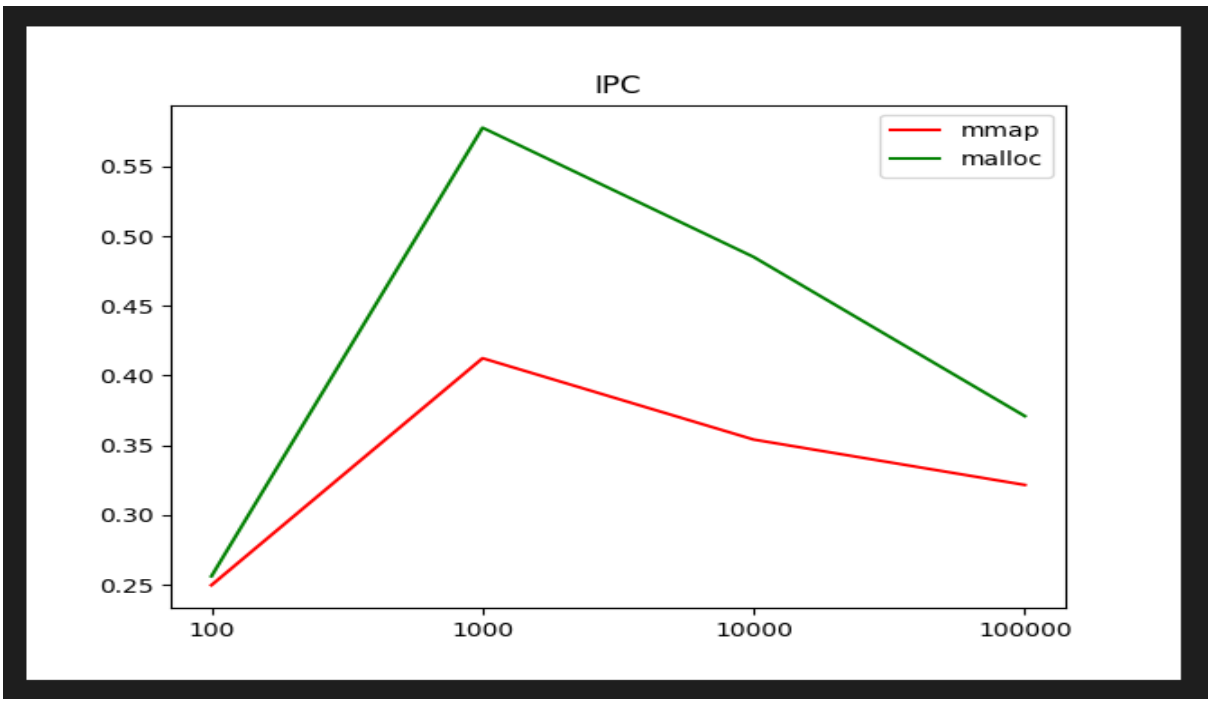
main but still, 1236 instructions are executing after that which can make a diff.

Question2:

12 PLOTS:

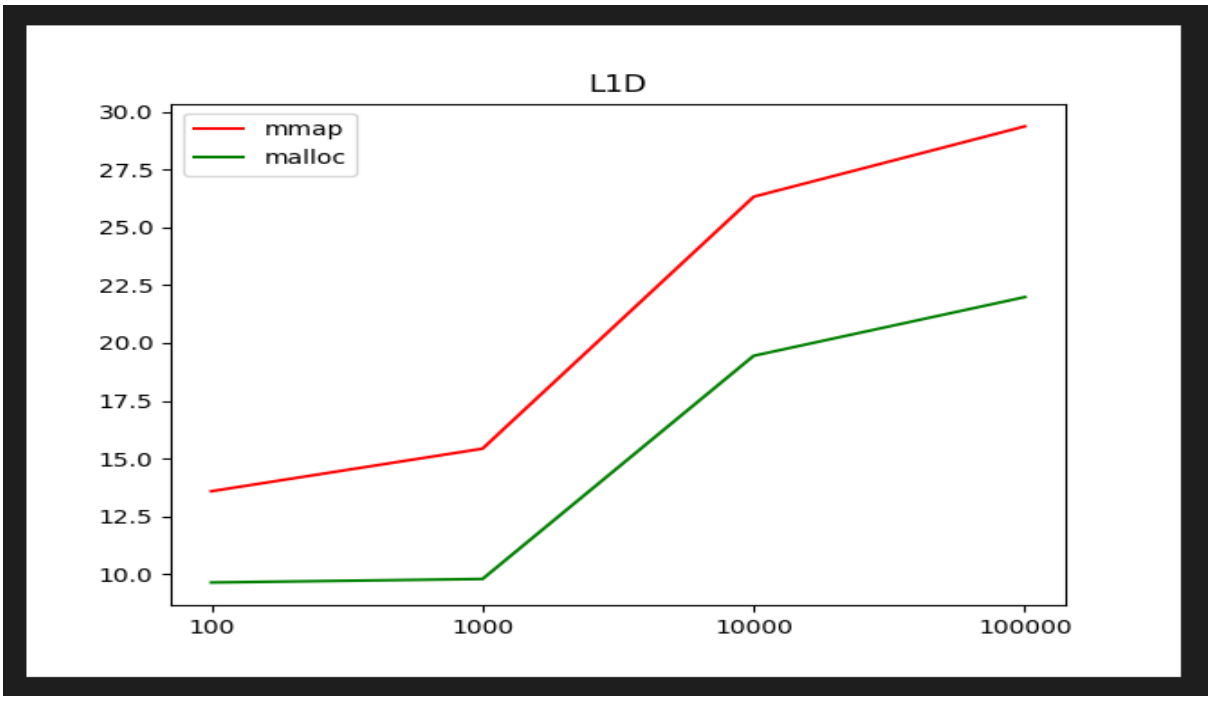
1. Default Configuration:

a) IPC



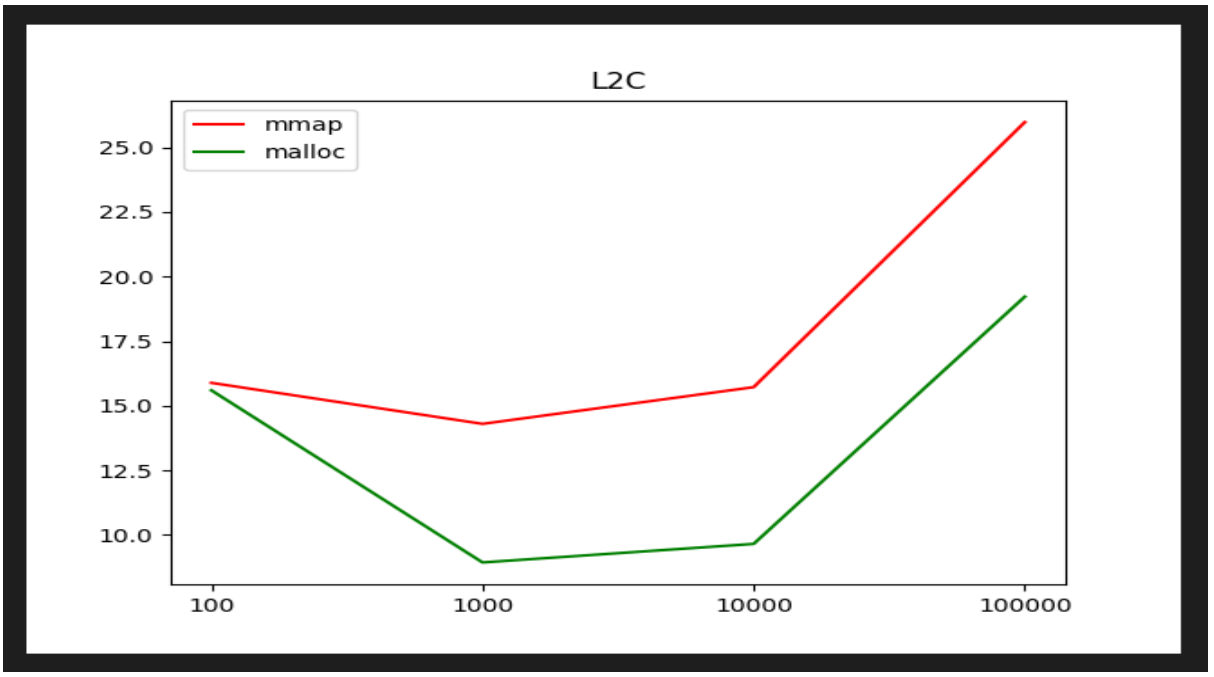
Graph No. 1

b) L1D



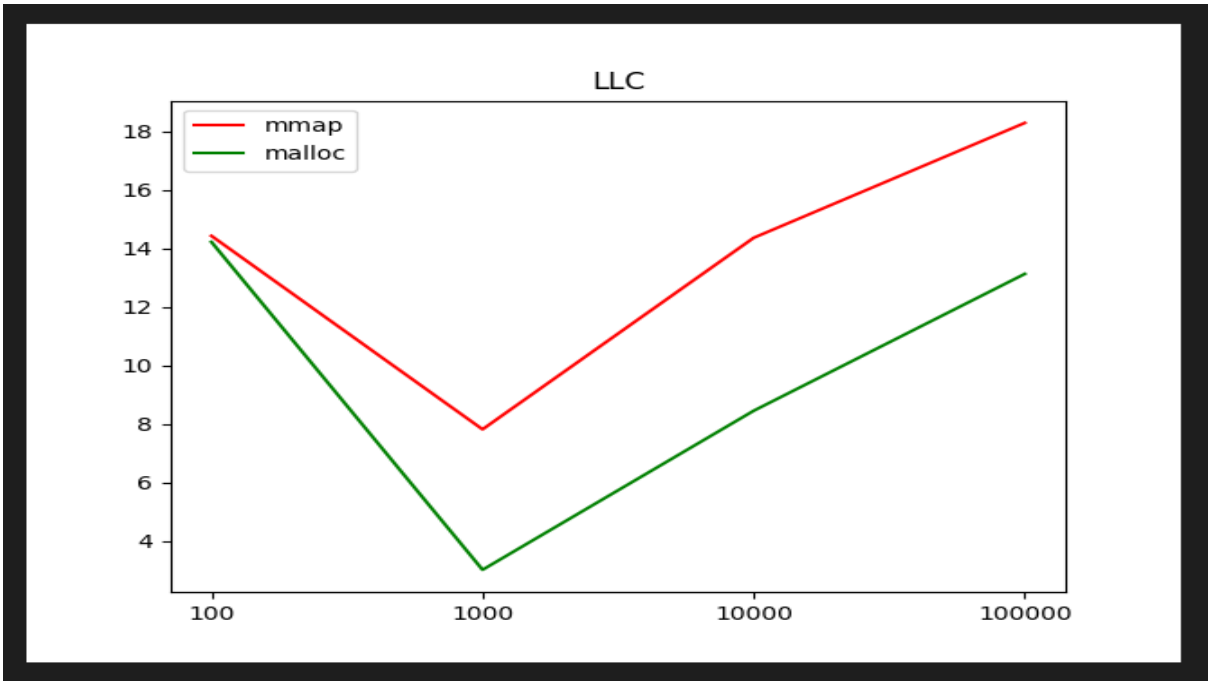
Graph No. 2

c) L2C



Graph No. 3

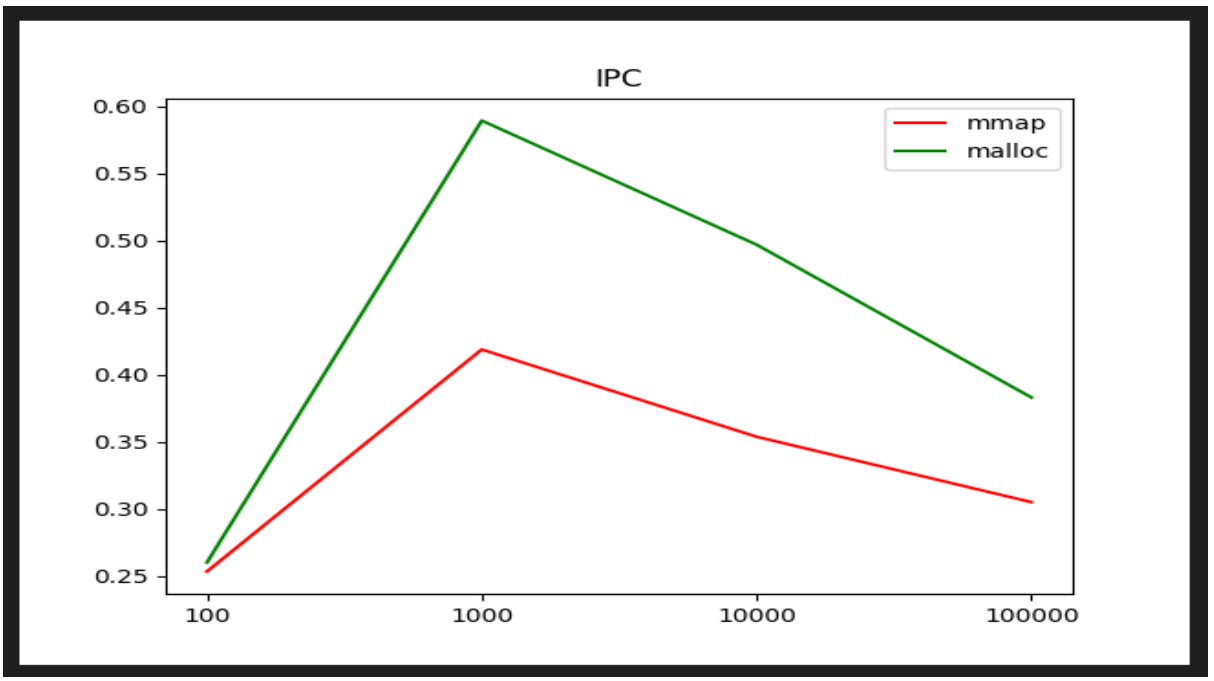
d) LLC



Graph No. 4

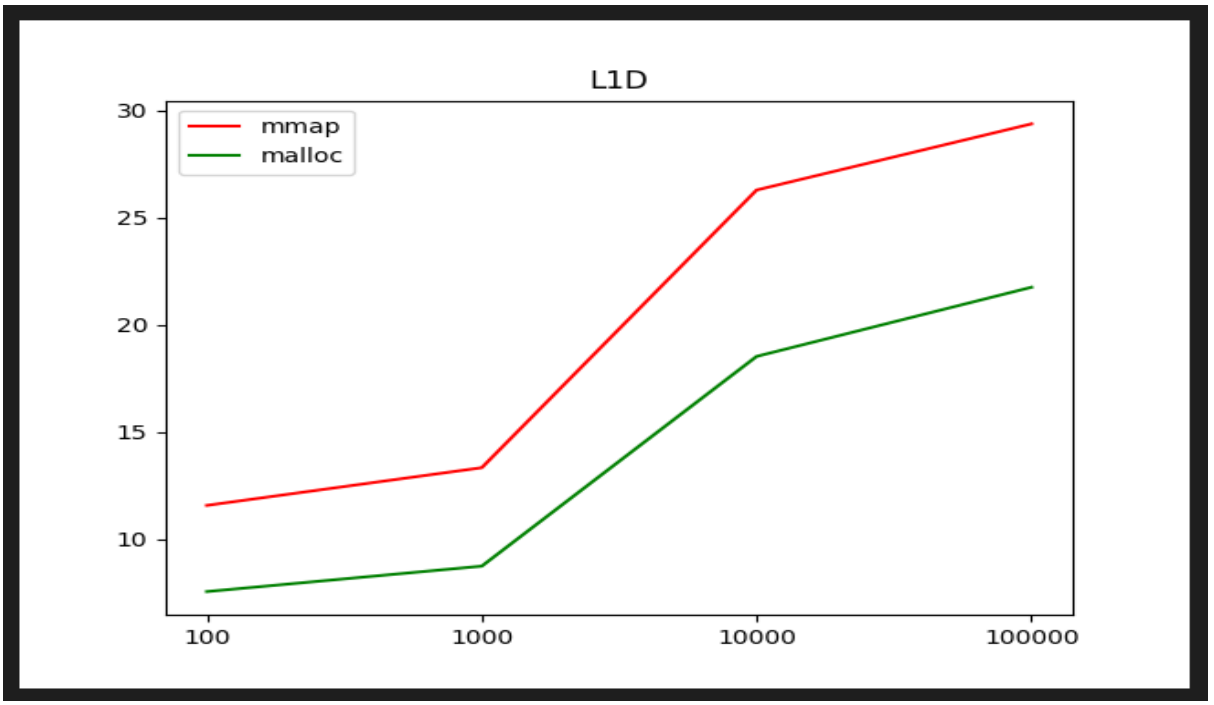
2. SWITCH ON NEXT LINE PREFETCHER AT L1D:

a) IPC



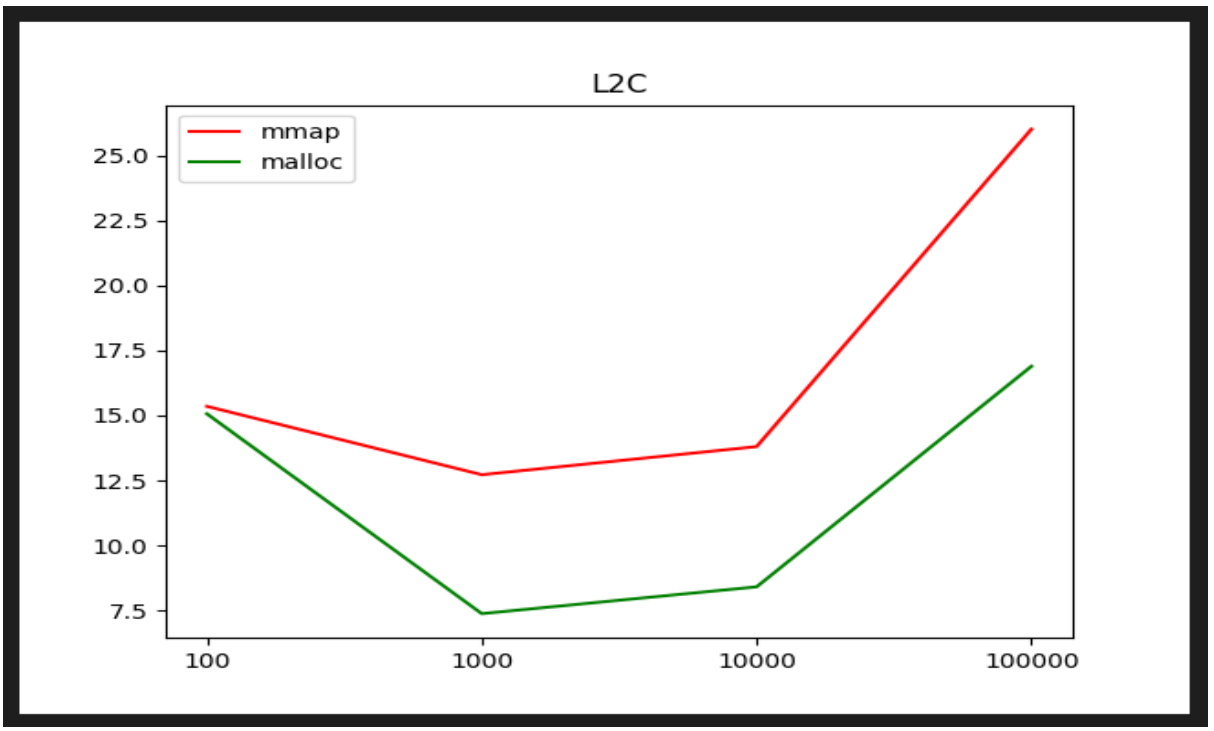
Graph No. 5

b) L1D



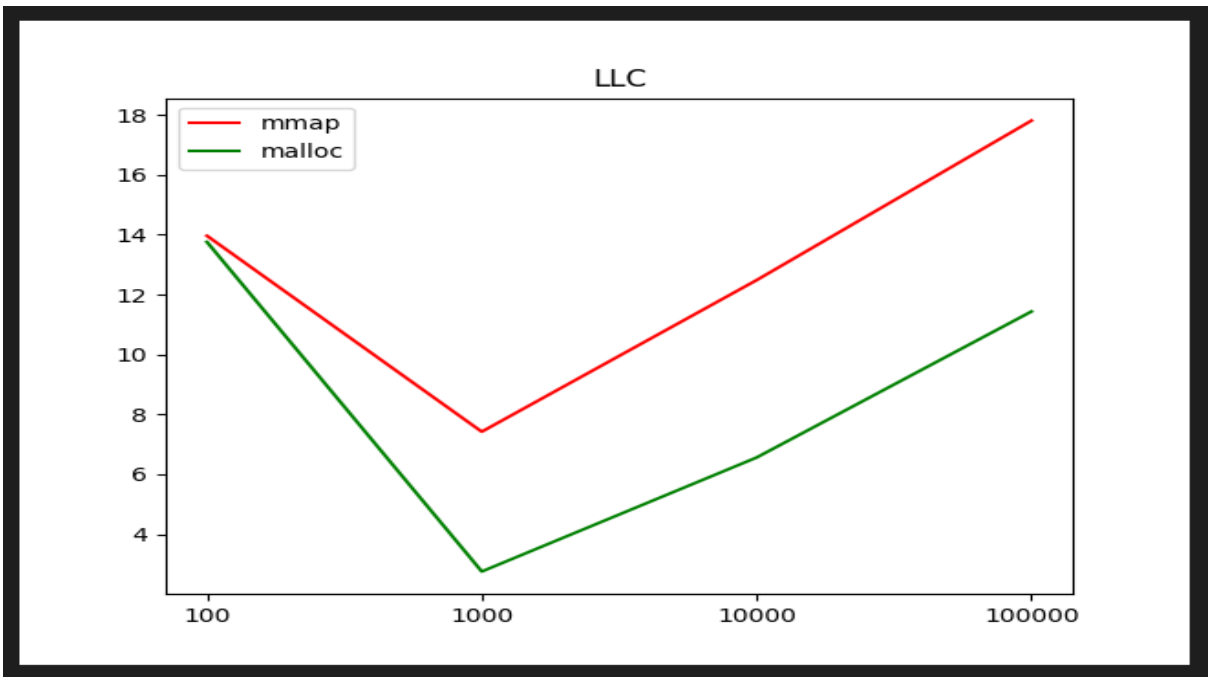
Graph No. 6

c) L2C



Graph No. 7

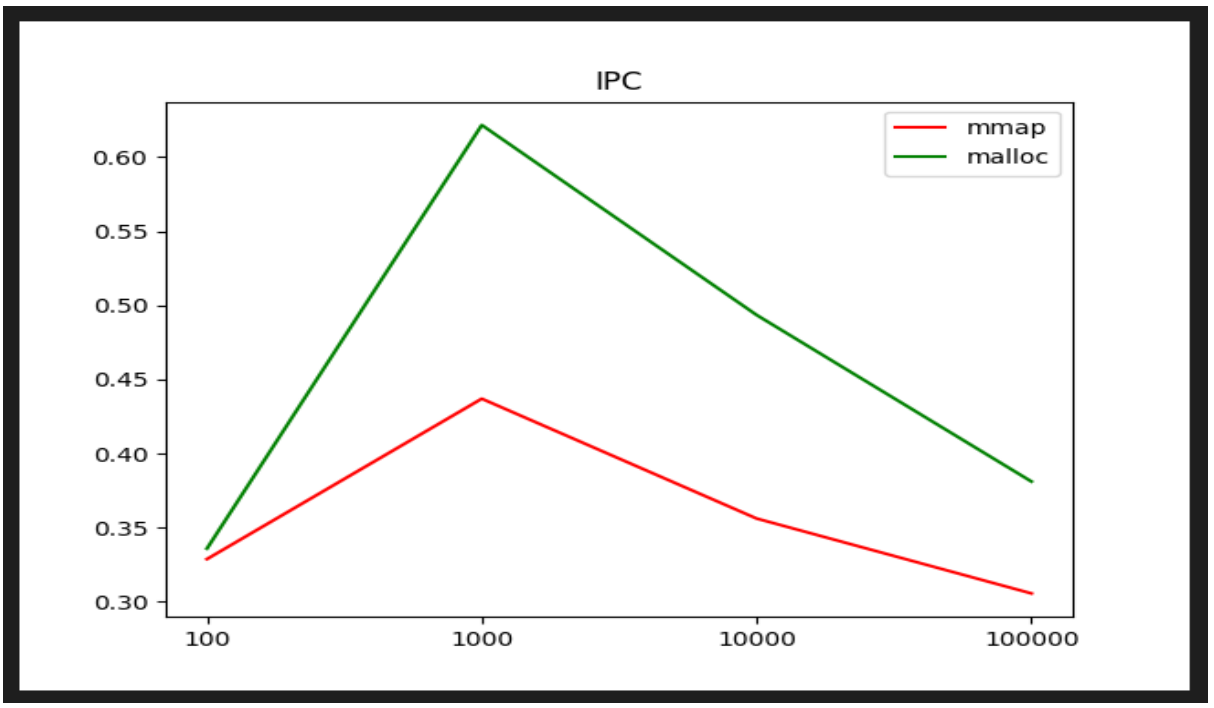
d) LLC



Graph No. 8

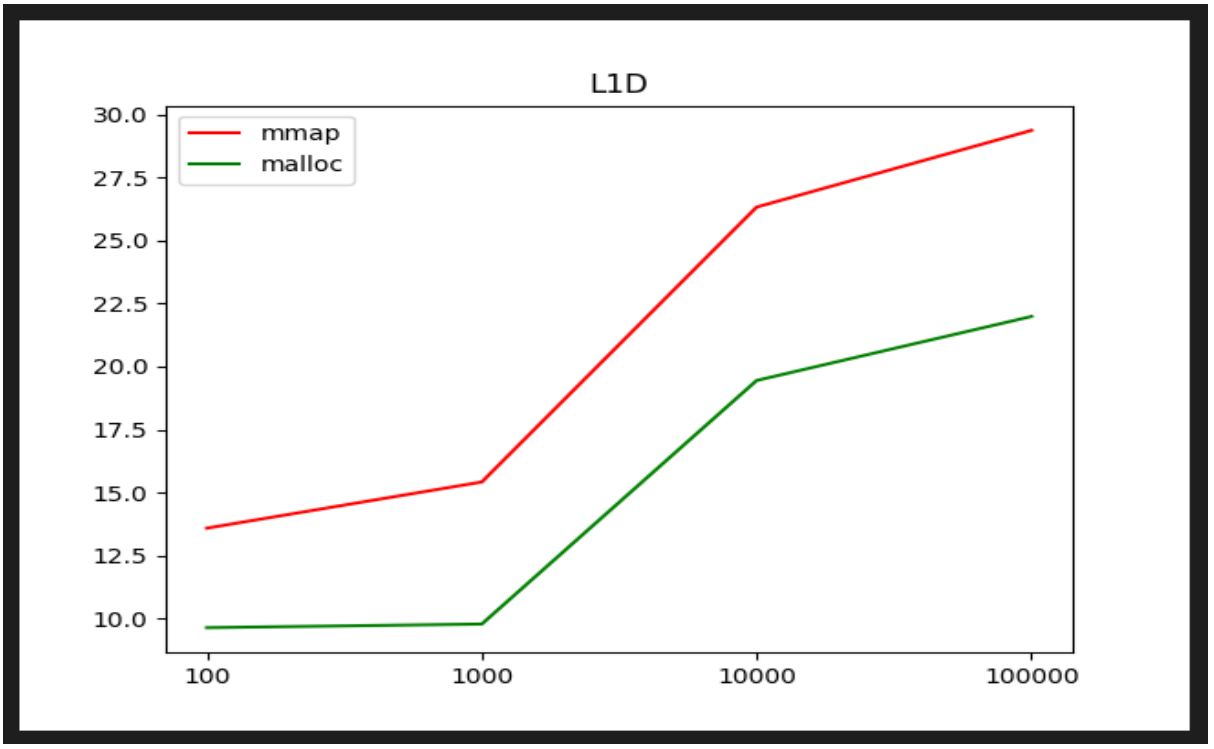
3. SWITCH ON NEXT LINE PREFETCHER AT L2C:

a) IPC



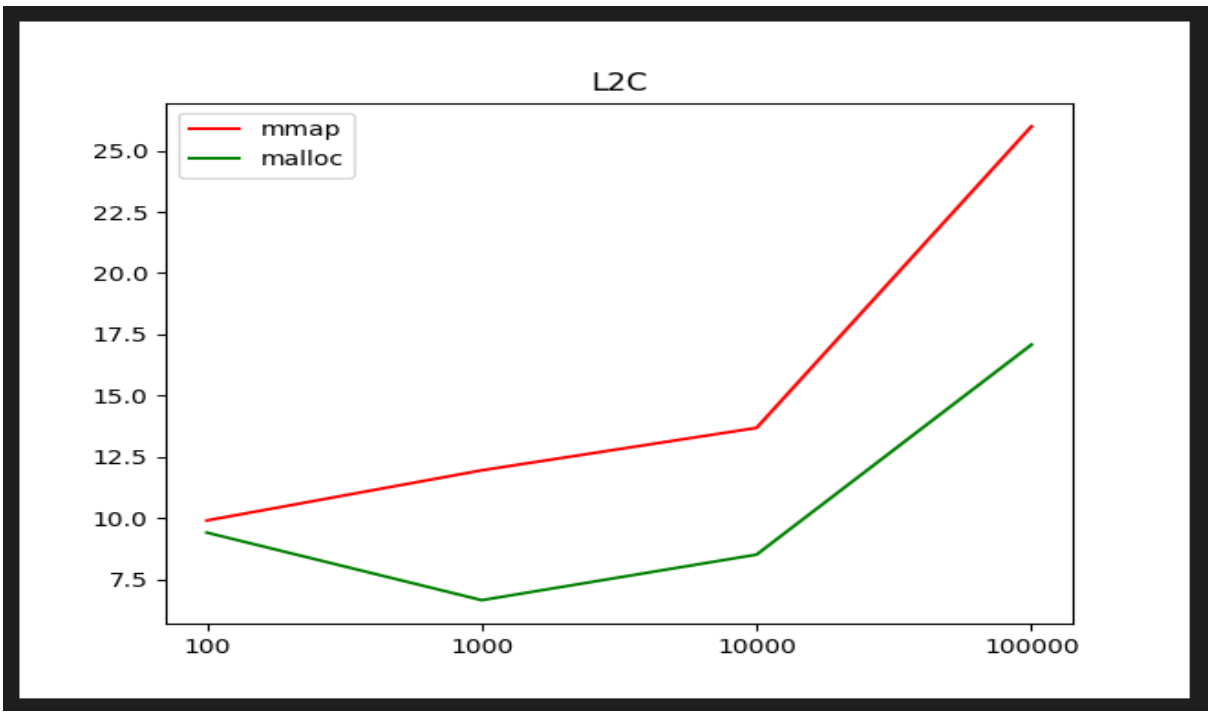
Graph No. 9

b) L1D



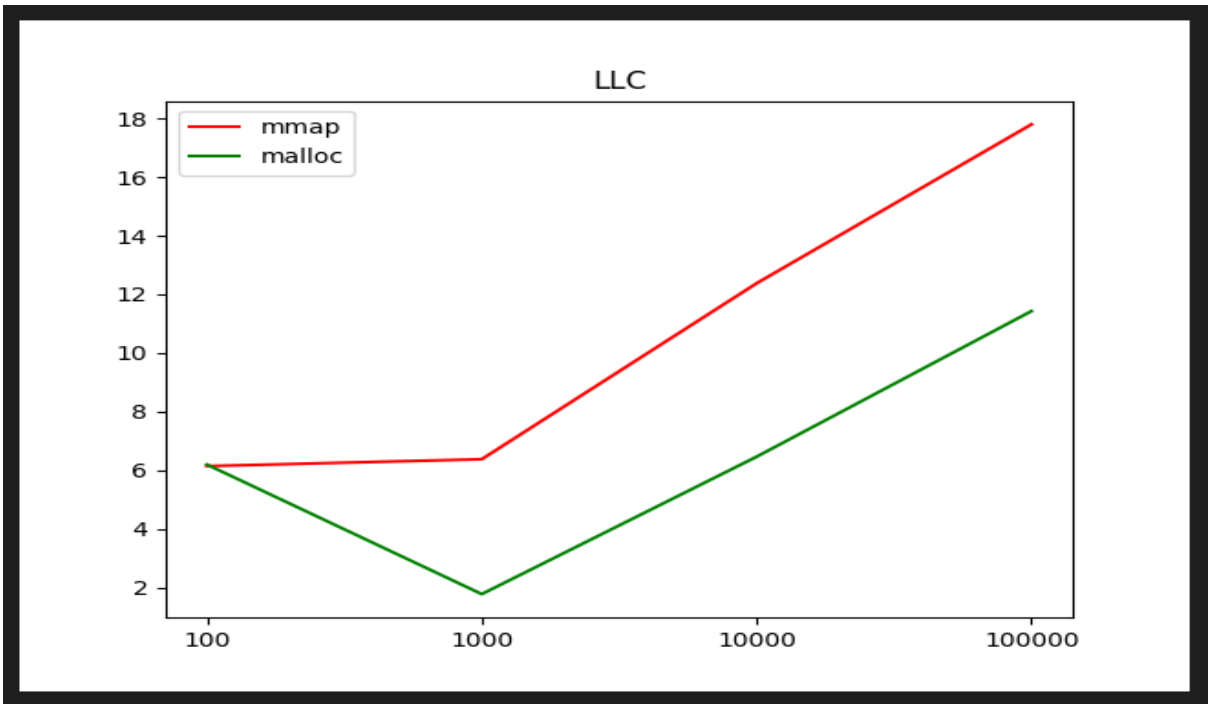
Graph No. 10

c) L2C



Graph No. 11

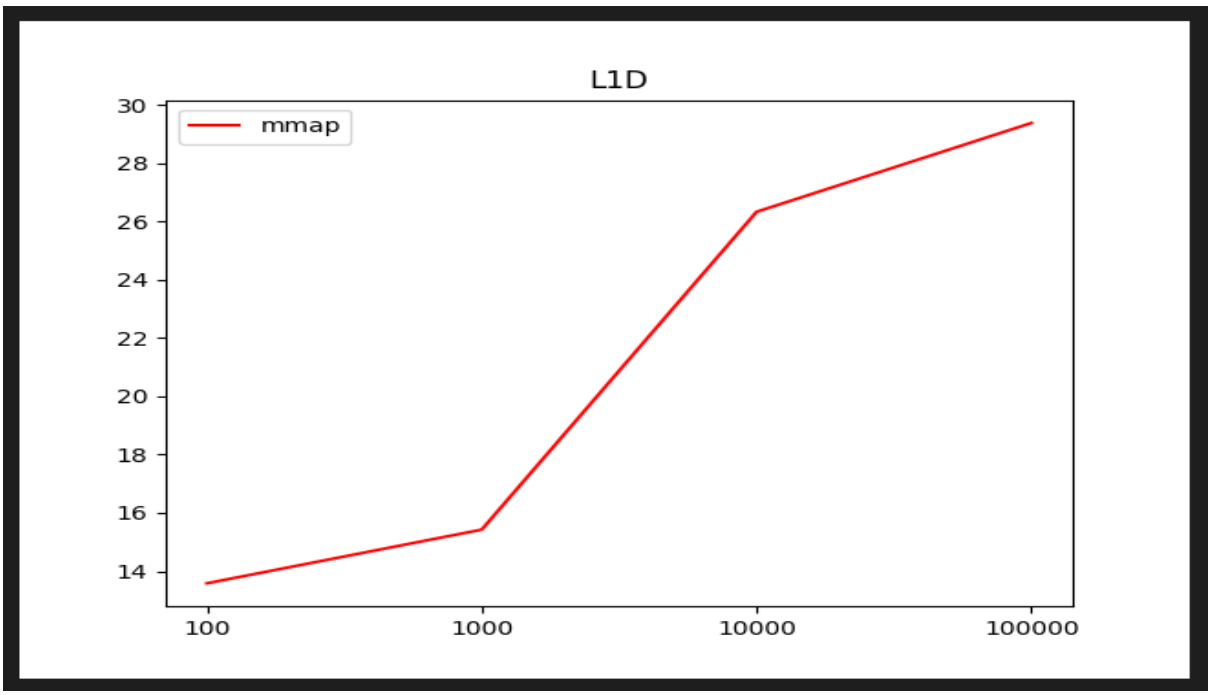
d) LLC



Graph No. 12

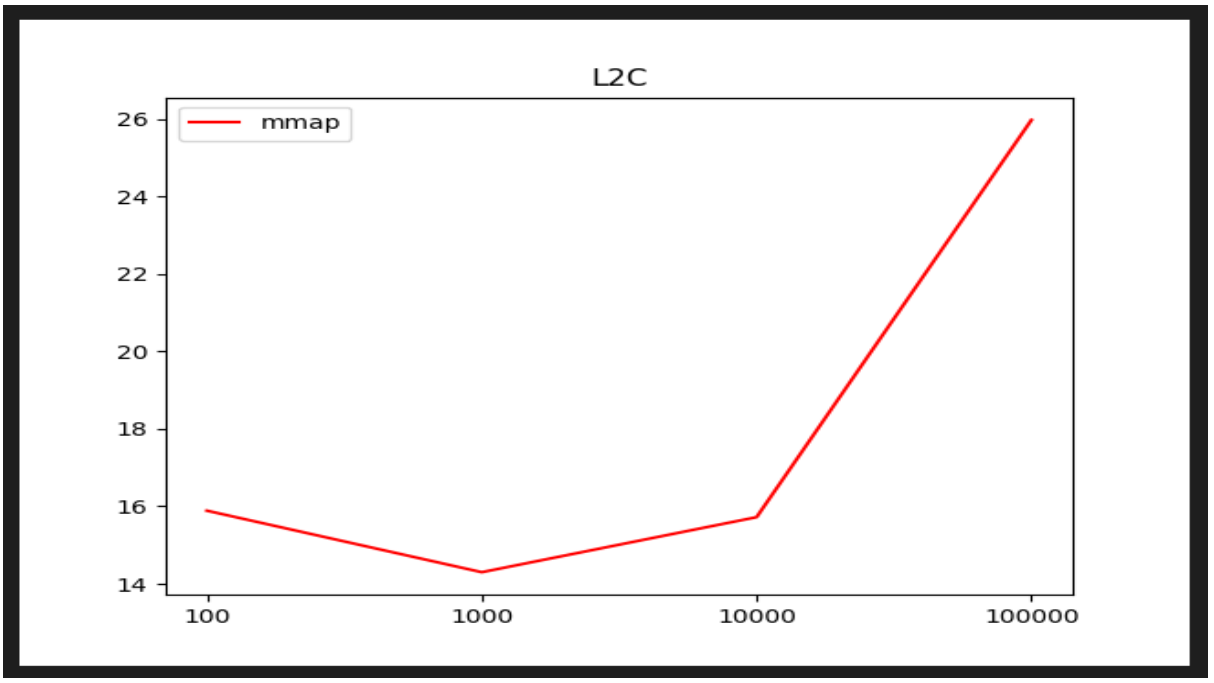
Task 4.1

1. MMAP Traces
   a) L1D



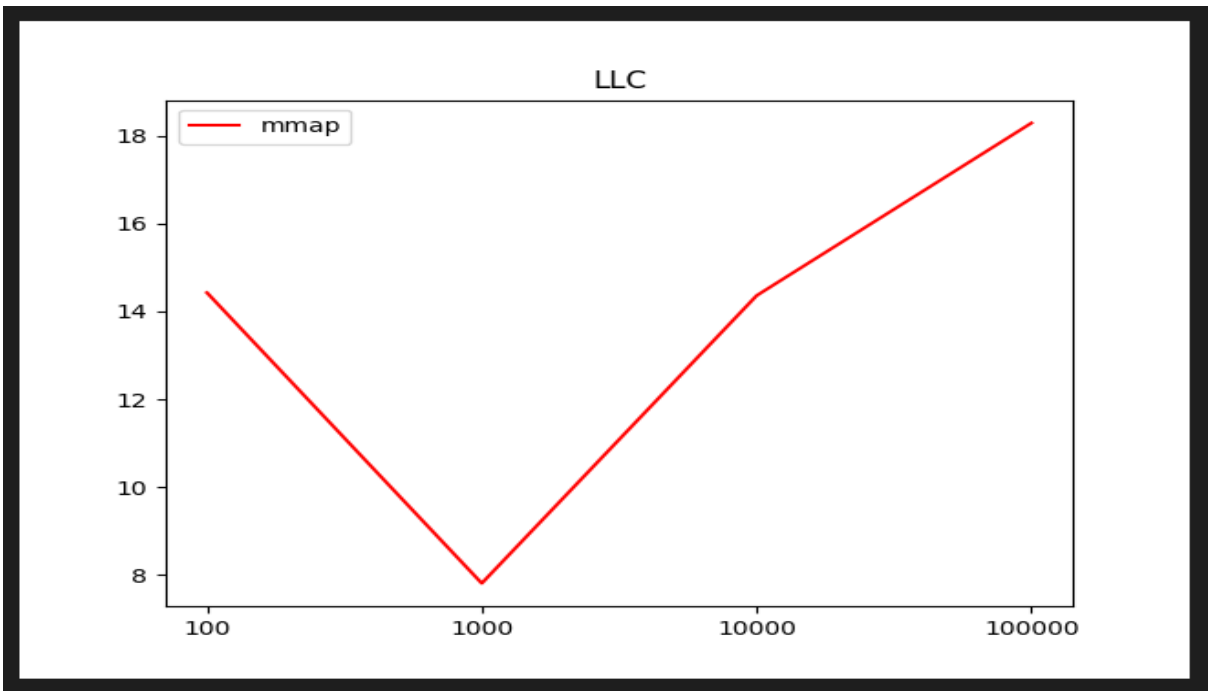
Graph No. 13

b) L2C



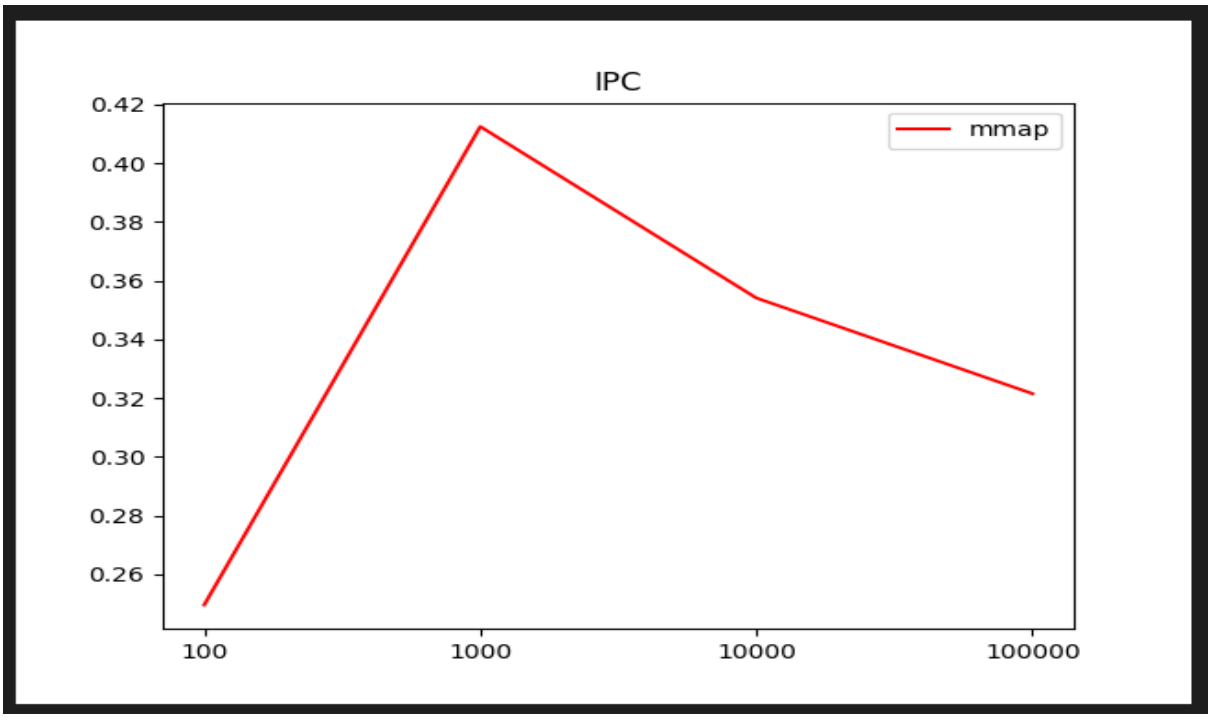
Graph No. 14

c) LLC



Graph No. 15

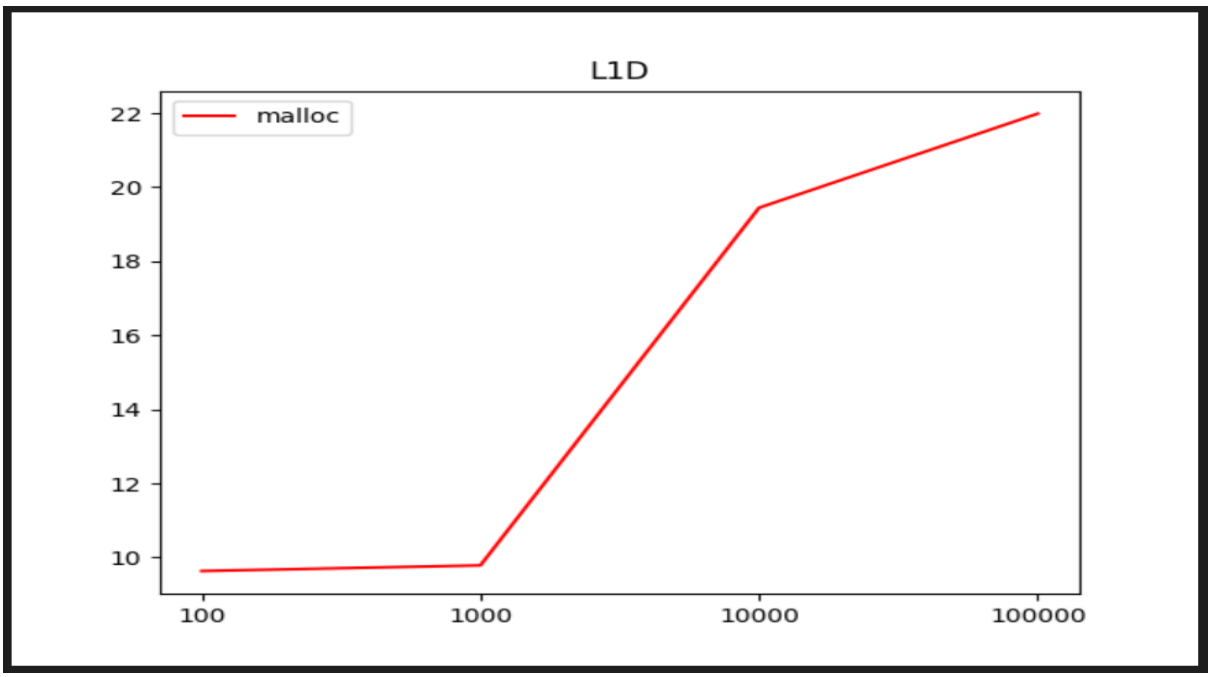
d) IPC



Graph No. 16

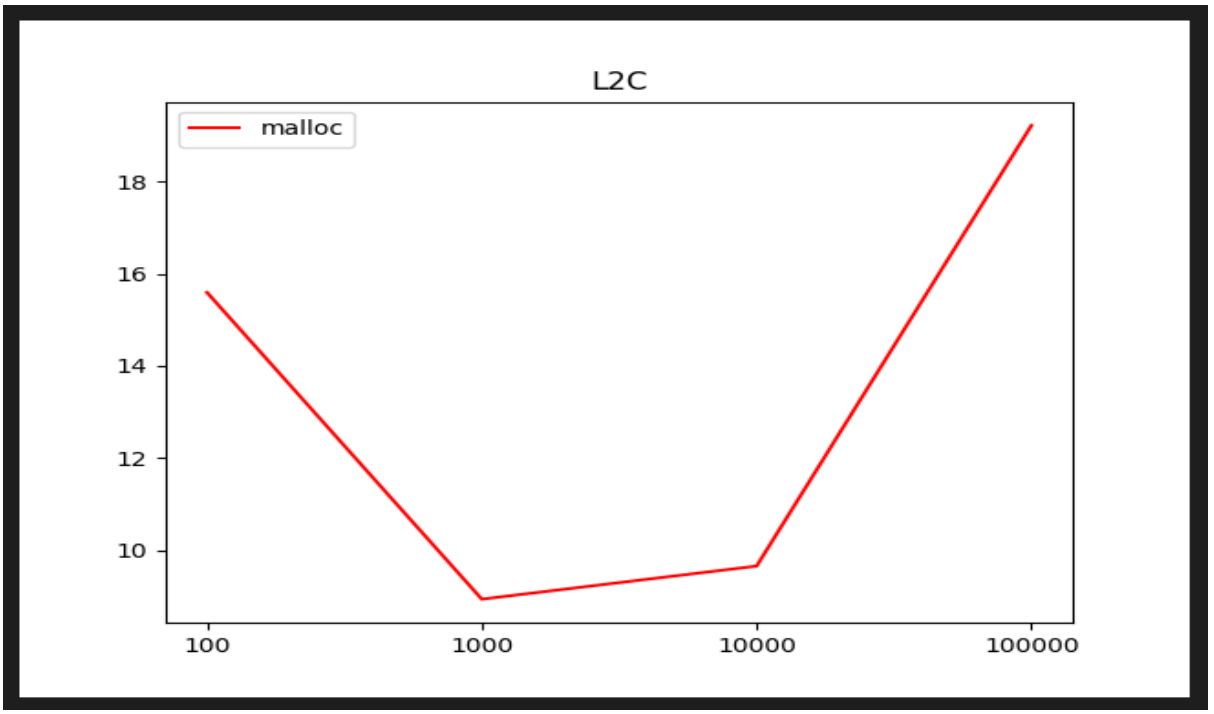
2. MALLOC Traces

a)L1D



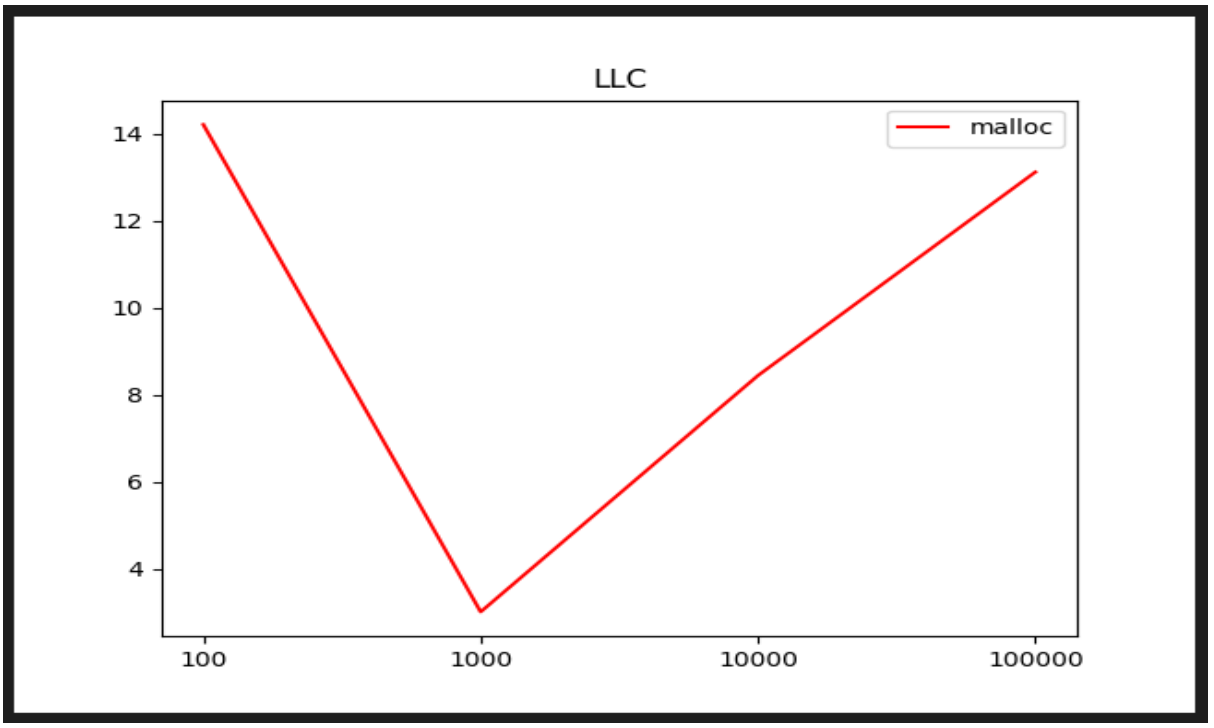
Graph No. 17

b)L2C



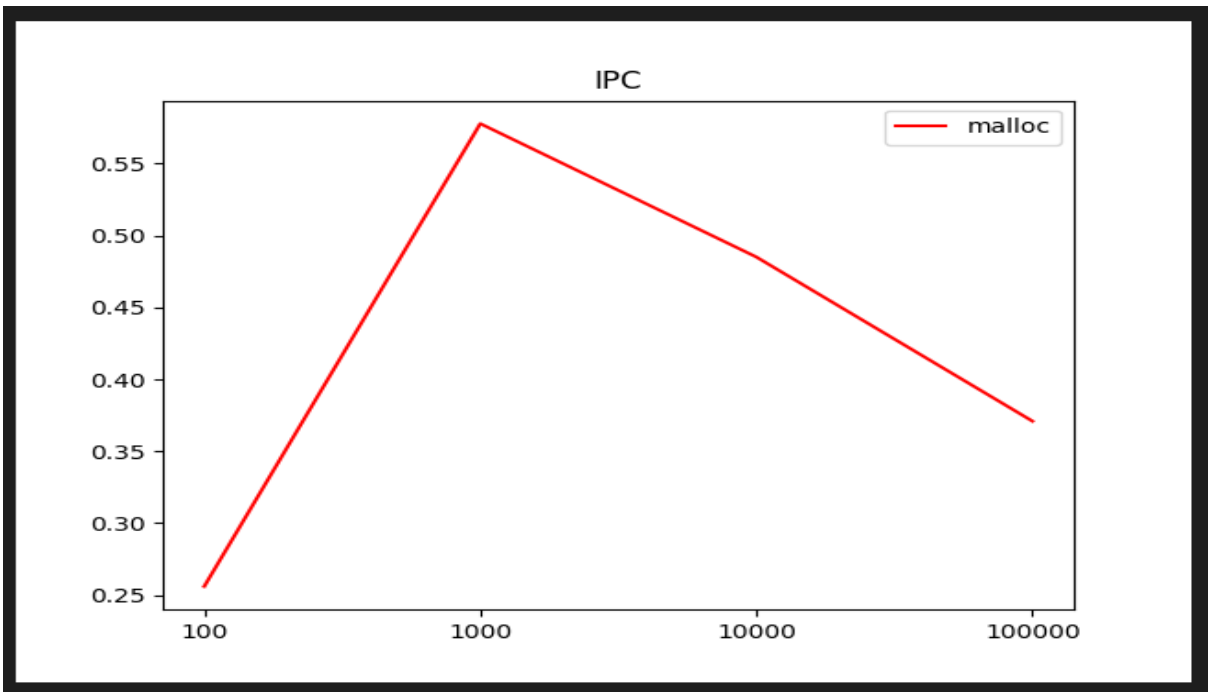
Graph No. 18

c)LLC



Graph No. 19

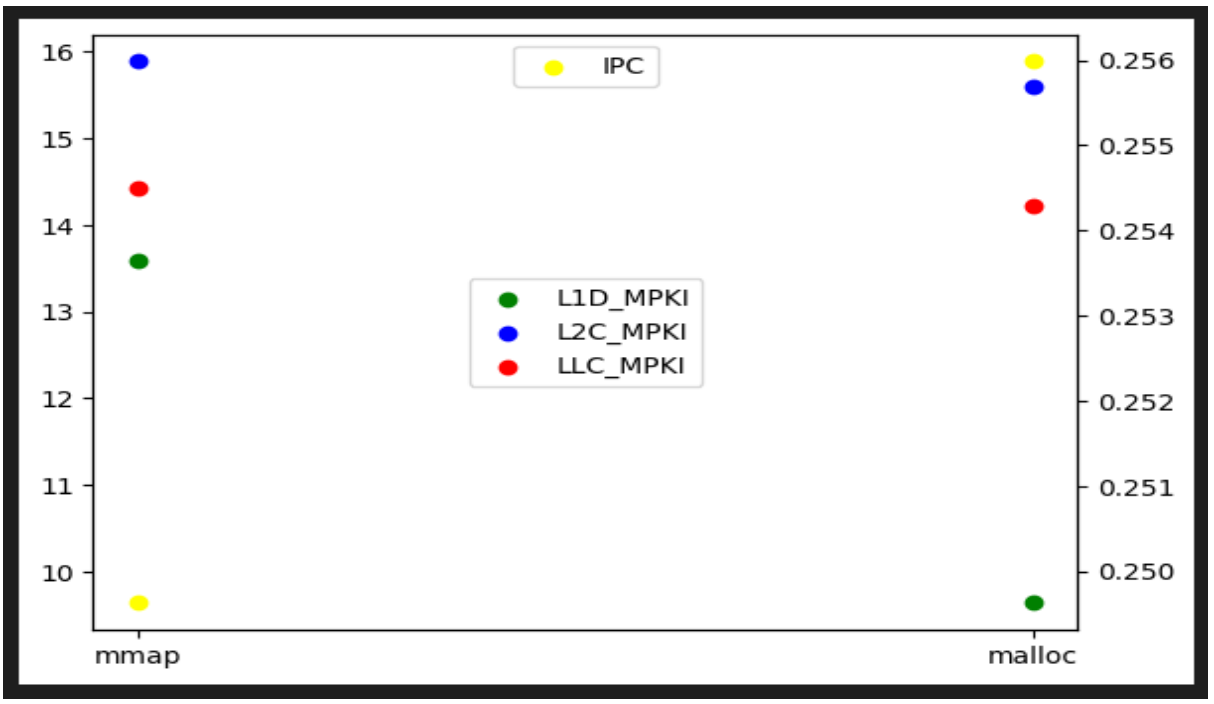
d) IPC



Graph No. 20

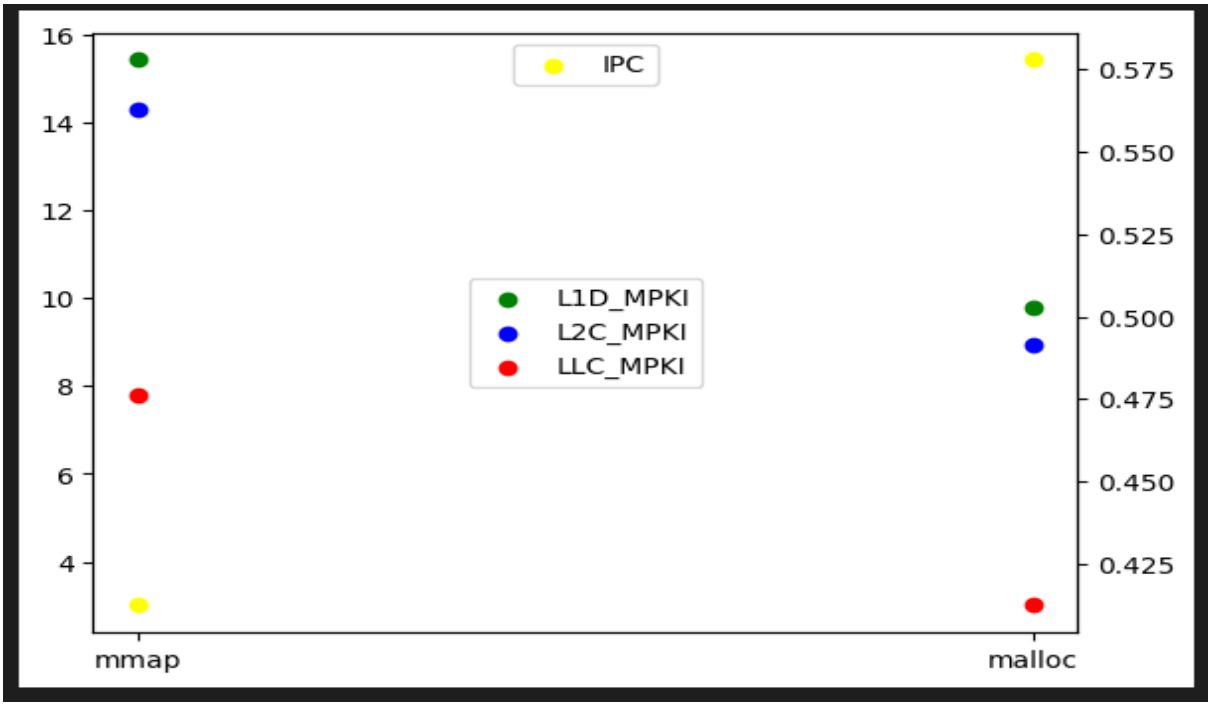
3.

a) Malloc100 vs Mmap100 (On right side of graph, that scale is for IPC)



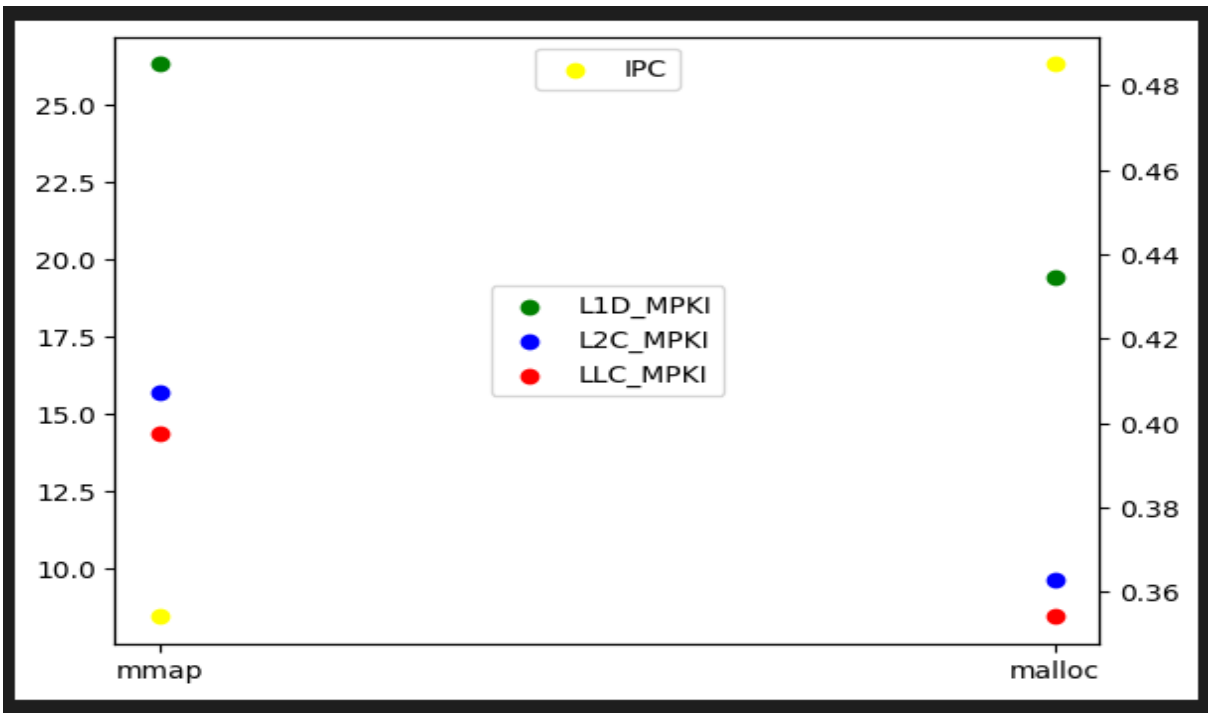
Graph No. 21

b) Malloc1000 vs Mmap1000



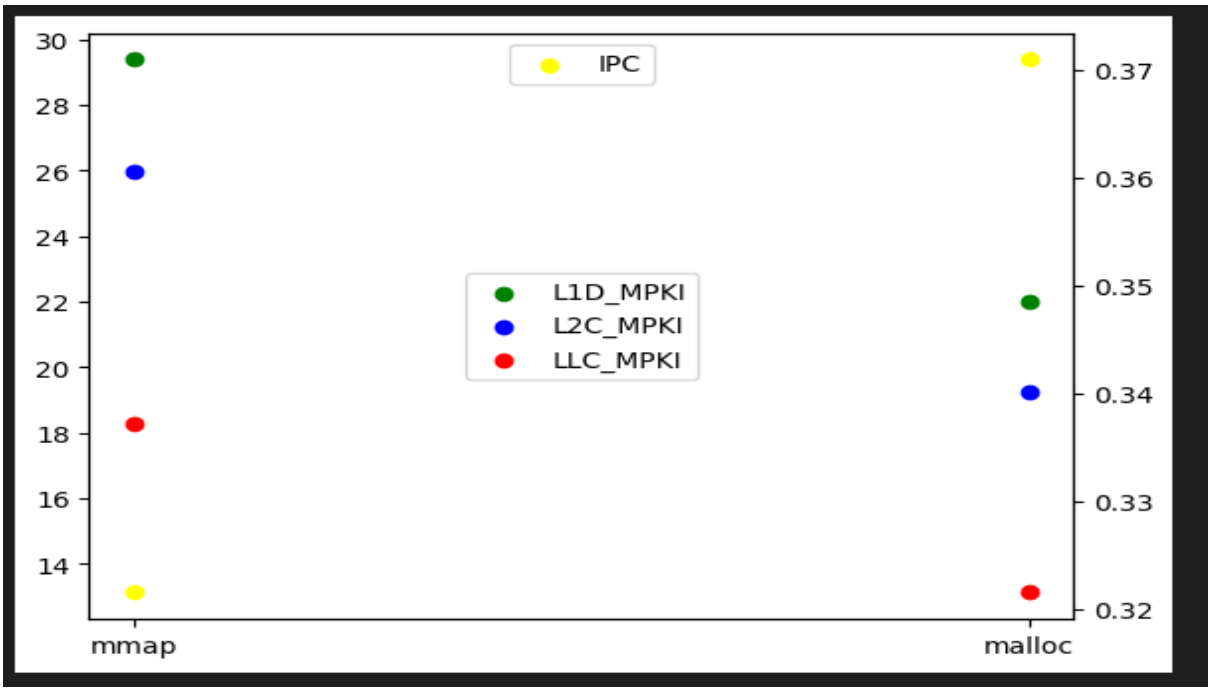
Graph No. 22

c)Malloc10000 vs Mmap10000



Graph No. 23

d) Malloc100000 vs Mmap100000



Graph No. 24

4. Yes, In Graph No. 19-24 we can see in for mmap vs malloc, In every case, IPC for malloc is greater and L1D, L2C, LLC MPKI is less for malloc. When we vary the number of instructions keeping malloc or mmap the same, we can see the IPC graphs(Graph no. 16, Graph no. 20), IPC is changing with L1D(Graph No. 13,17), L2C(Graph No. 14, 18), LLC(Graph No. 15, 19). In these varying graphs of IPC(Graph no. 16, Graph no. 20) with L1D, L2C and LLC MPKI, We can see

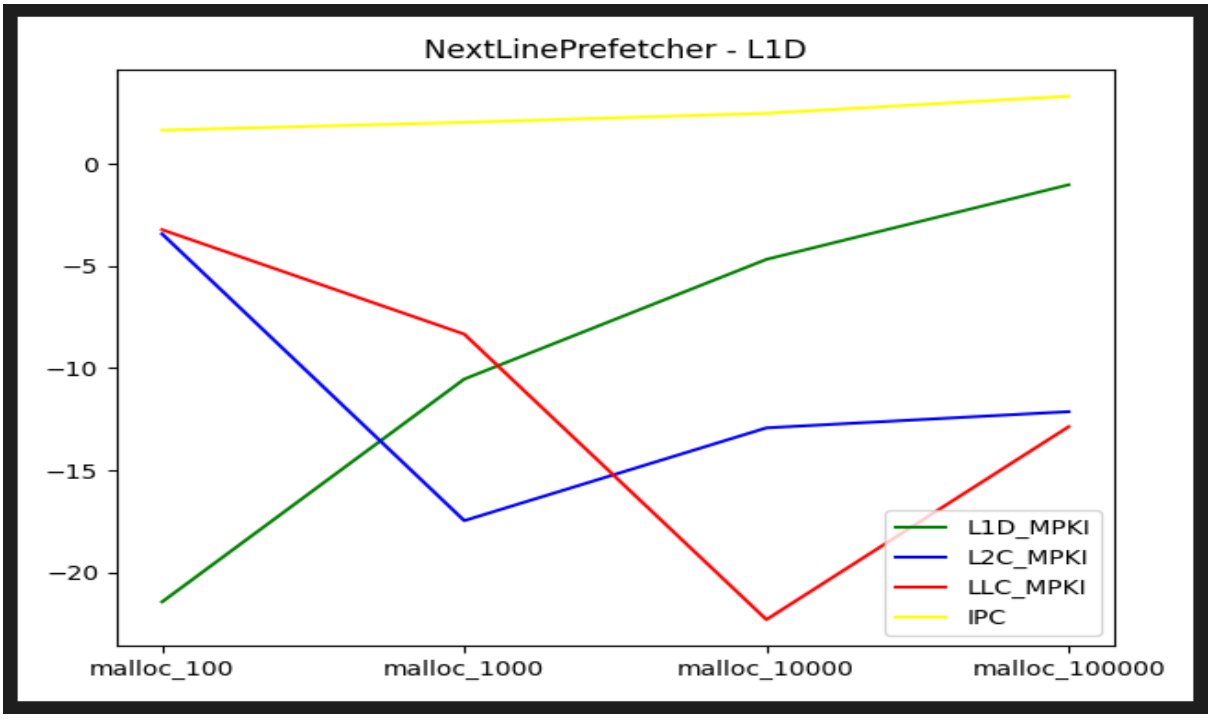
IPC(Graph no. 16, Graph no. 20) and LLC graphs(Graph No. 15, 19) are almost inverted to each other means when LLC MPKI drops IPC increases and vice versa. So we can say that **LLC MPKI** is closely reflected in IPC.

Task 4.2

1. Percentage Change(Next line prefetcher L1D vs Default Configuration)
   (Red :Decreased and green: Increased)

Malloc:

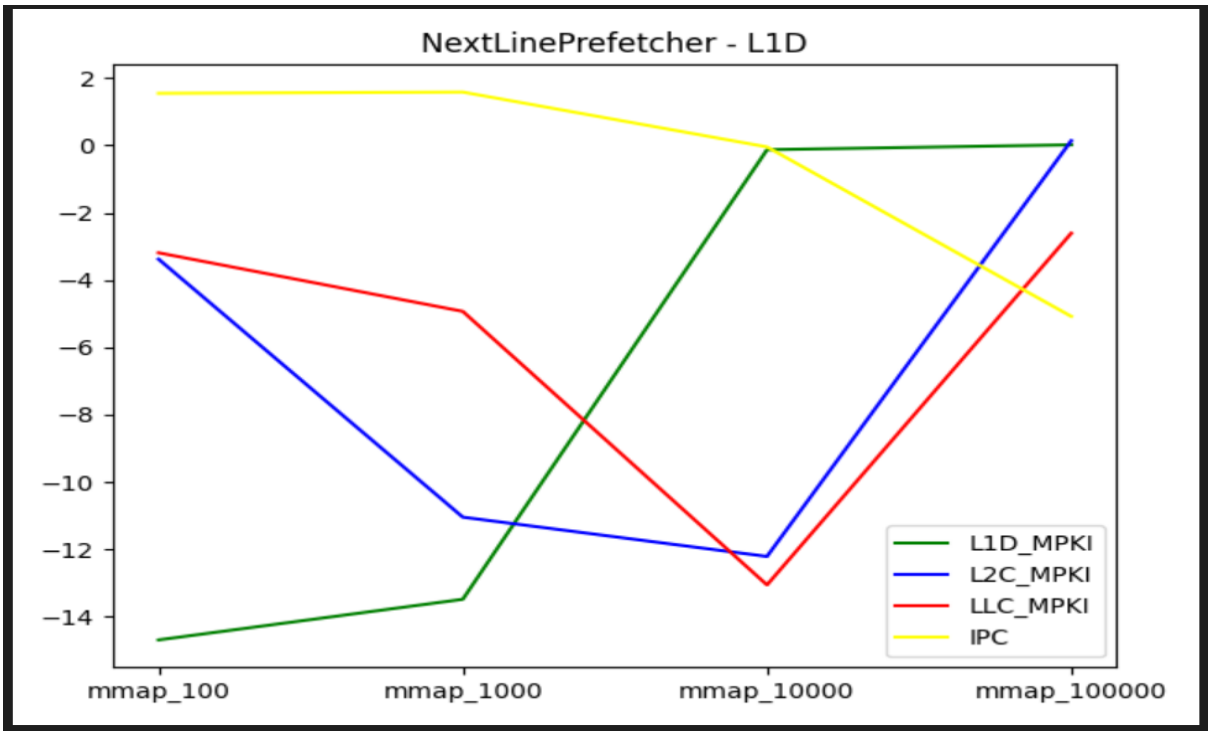
Change(%)	malloc_100	malloc_1000	malloc_10000	malloc_100000
L1D MKPI	21.4	10.5	4.7	1.0
L2C MKPI	21.4	10.5	4.7	1.0
LLC MKPI	3.2	8.3	22.3	12.9
IPC	1.6	2.0	2.5	3.3



Graph No. 25

Mmap:

Change(%)	mmap_100	mmap_1000	mmap_10000	mmap_100000
L1D MKPI	14.7	13.5	0.13	0.01
L2C MKPI	3.4	11.0	12.2	0.13
LLC MKPI	3.2	4.9	13.0	2.6
IPC	1.5	1.6	0.1	5.1

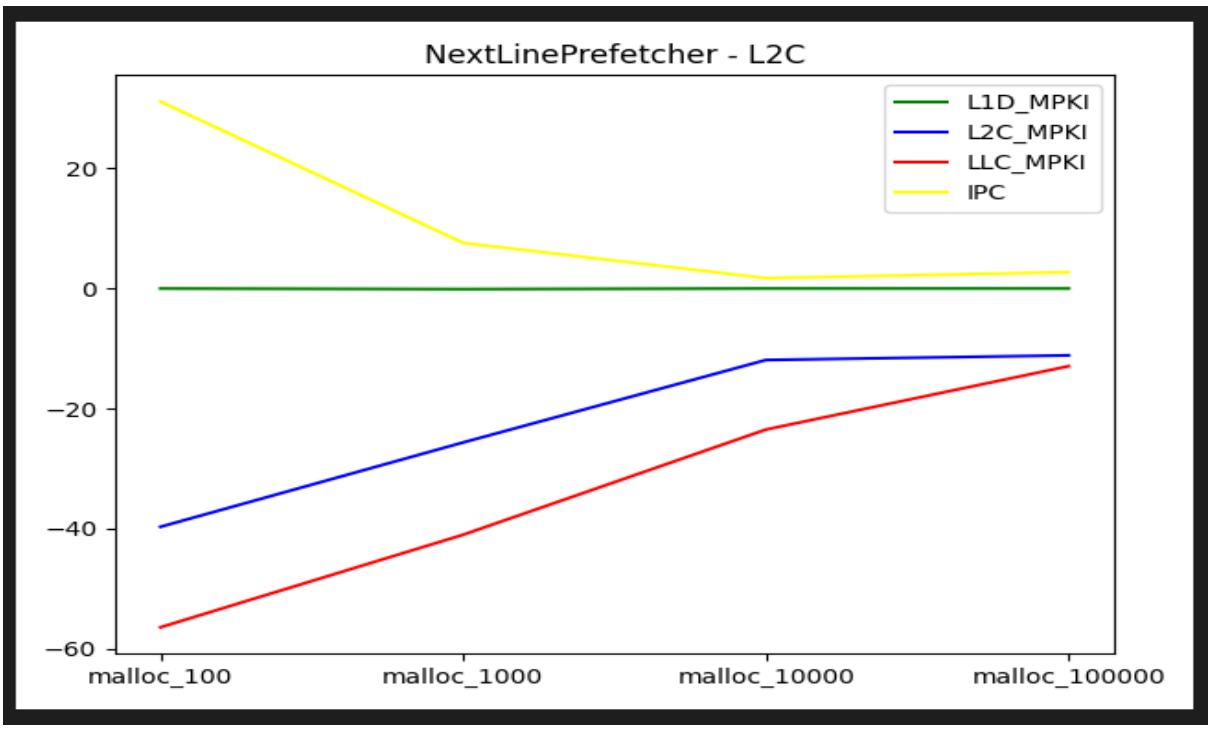


Graph No. 26

2. Percentage Change(Next line prefetcher L2C vs Default Configuration)
(Red :Decreased and green: Increased)

Malloc:

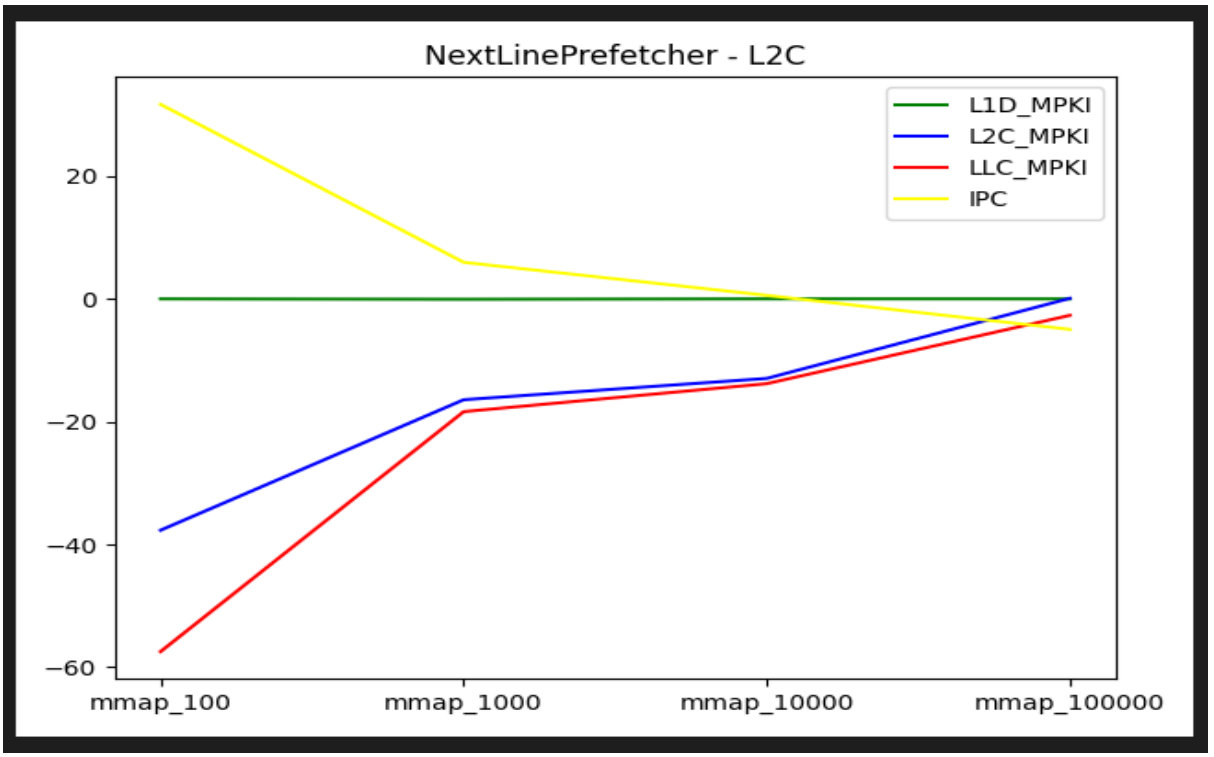
Change(%)	malloc_100	malloc_1000	malloc_10000	malloc_100000
L1D MKPI	0	0.1	0.01	0.002
L2C MKPI	39.7	25.6	11.9	11.1
LLC MKPI	56.5	41.0	23.5	13.0
IPC	31.6	7.6	1.7	2.7



Graph No. 27

Mmap:

Change(%)	mmap_100	mmap_1000	mmap_10000	mmap_100000
L1D MKPI	0	0.06	0.004	0
L2C MKPI	37.7	16.4	13.0	0.09
LLC MKPI	57.4	18.4	13.8	2.7
IPC	31.7	6.0	0.6	5.0



Graph No. 28

3. No, it doesn't increase MKPI instead MKPI is reduced for that and lower levels. We can see in Graphs (25-28) and 4 tables shown above.

4. From the % change shown in the tables we can clearly see that we got more benefit from enabling the next line prefetcher for L2C.

5. Malloc gets more benefit from both the prefetcher as compared to mmap. This is because malloc allocates a contiguous big chunk of memory as compared to mmap which allocates many chunks of small size. As the data is contiguous malloc gets more benefit from the prefetcher because the data prefetched is used in upcoming instructions.