Assignment 3

Group Members-

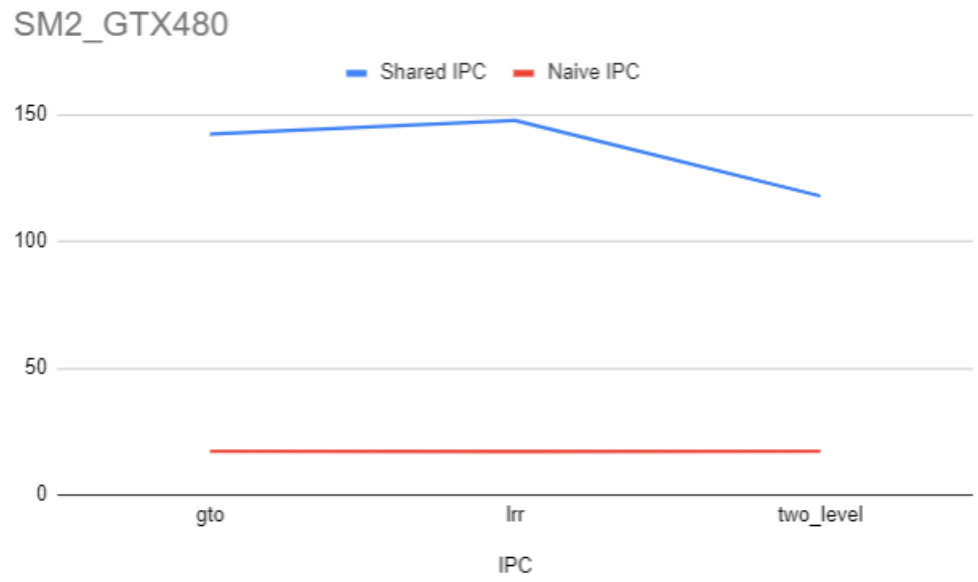
1.Manan Khanna (21CS01028)

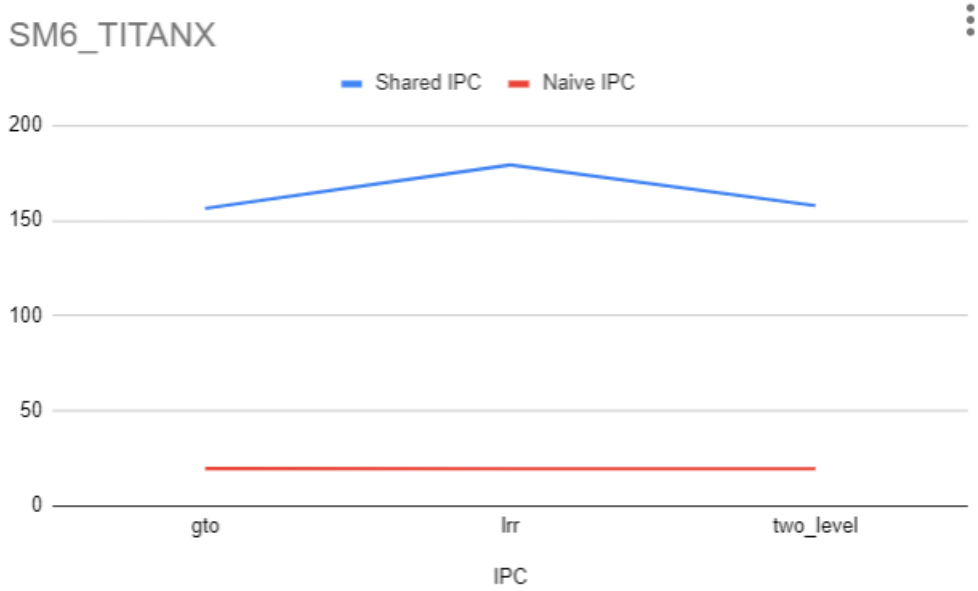
2.Aman Dangi (21CS01027)

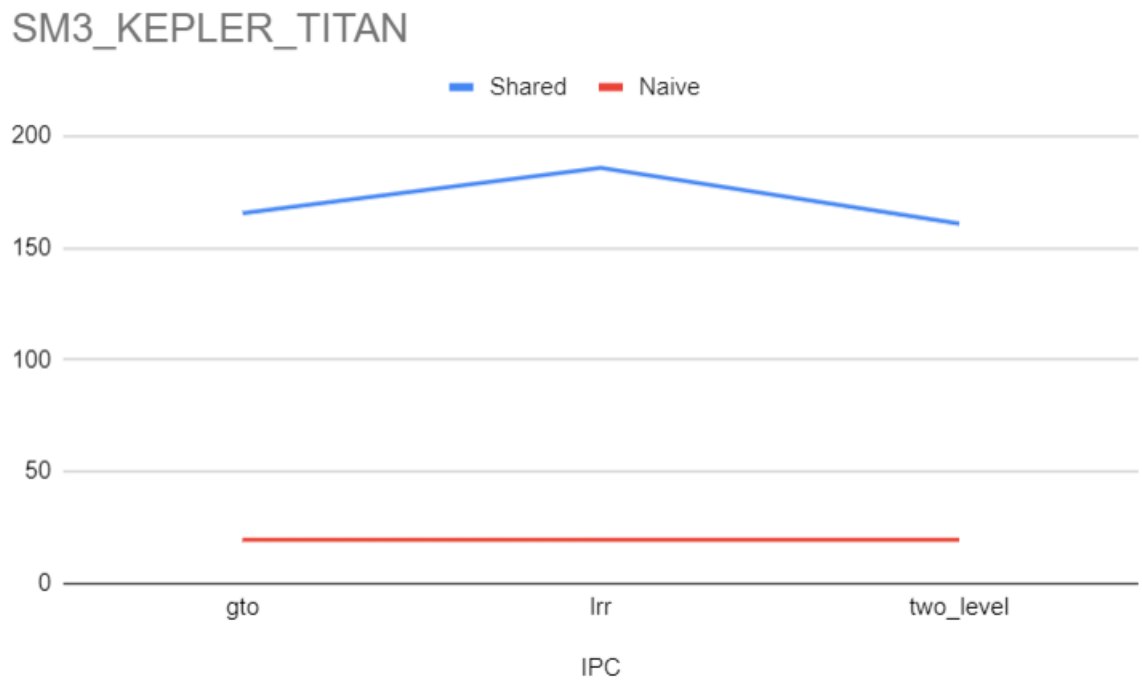
3.Soumyabrata Chaudhuri (21CS1032)

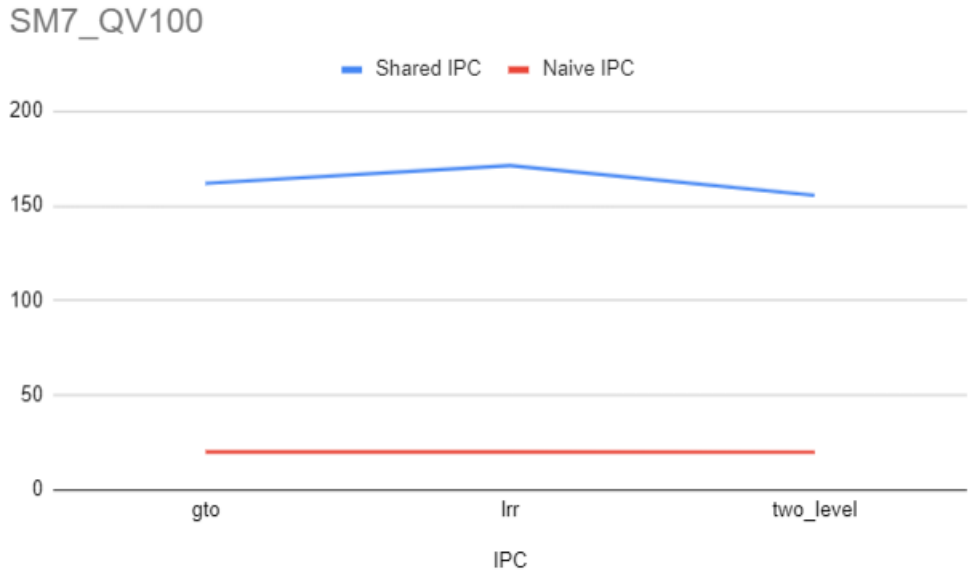
4.Shubham Kaushik (21CS01030)

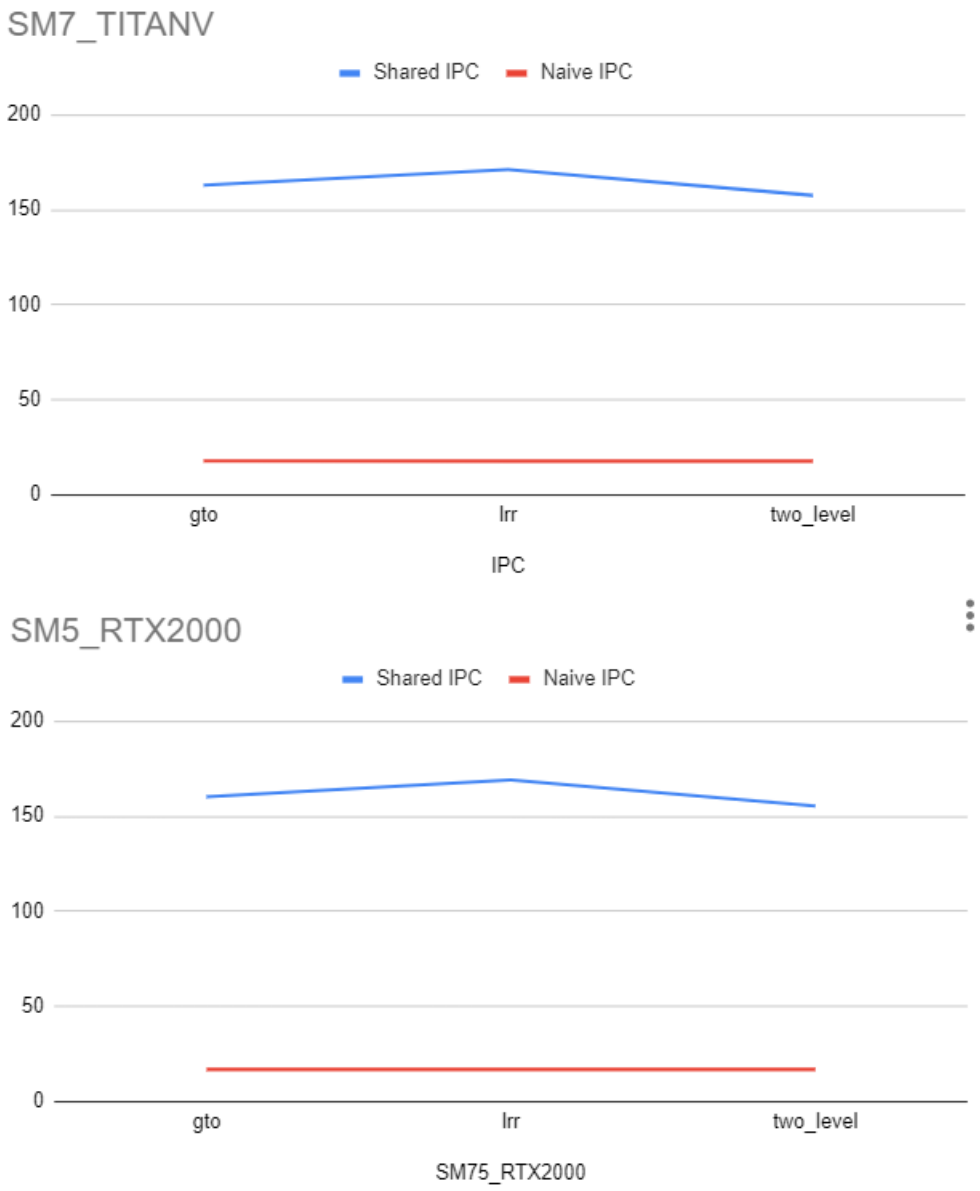
	A	B	C	D	E
1	SM75_RTX2000	Shared IPC	Naive IPC	Shared Time	Naive Time
2	gto	160.24	16.97	21	45
3	lrr	169.14	16.96	21	46
4	two_level	155.42	16.94	22	45
5					
6	SM7_TITANV	Shared IPC	Naive IPC	Shared Time	Naive Time
7	gto	163.01	17.86	39	93
8	lrr	171.2	17.84	37	93
9	two_level	157.63	17.82	39	94
10					
11	SM7_QV100	Shared IPC	Naive IPC	Shared Time	Naive Time
12	gto	161.89	20.11	43	96
13	lrr	171.3	20.1	44	96
14	two_level	155.58	20.04	43	100
15					
16	SM6_TITANX	Shared IPC	Naive IPC	Shared Time	Naive Time
17	gto	156.47	19.65	22	39
18	lrr	179.31	19.64	20	40
19	two_level	157.9	19.61	21	41
20					
21	SM2_GTX480	Shared IPC	Naive IPC	Shared Time	Naive Time
22	gto	142.44	17.26	17	26
23	lrr	147.75	17.18	16	25
24	two_level	118.01	17.22	18	25
25					
26	SM3_KEPLER_TITAN	Shared IPC	Naive IPC	Shared Time	Naive Time
27	gto	165.42	19.63	18	31
28	lrr	185.72	19.62	16	32
29	two_level	160.9	19.5	18	31
30					


SM2_GTX480
Shared IPC
Naive IPC
150
100
50
0
gto
lrr
two_level
IPC


SM6_TITANX
Shared IPC
Naive IPC
200
150
100
50
0
gto
lrr
two_level
IPC


SM3_KEPLER_TITAN
Shared
Naive
200
150
100
50
0
gto
lrr
two_level
IPC


SM7_QV100
Shared IPC
Naive IPC
200
150
100
50
0
gto
lrr
two_level
IPC



The impact of these schedulers on Instructions Per Cycle (IPC) depends on how effectively they manage the available resources and thread scheduling.

GTO (Greedy Throttle Out): GTO is a warp scheduler that tries to maximize the throughput

by selecting the warps that have the highest number of ready threads to execute. It doesn't prioritize fairness, so if there's a warp with many ready threads, it can starve other warps, potentially reducing overall GPU efficiency. This scheduler often results in high IPC for workloads that have large thread-level parallelism and relatively uniform resource usage.

LRR (Least Recently Returned): LRR is a more balanced scheduler that tries to ensure fairness among warps. It gives preference to the warp that has been waiting the longest since its last execution. This approach helps prevent warp starvation but might not always maximize throughput in cases where some warps have more ready threads than others.

Two-Level Active Scheduler: This type of scheduler combines characteristics of both GTO and LRR. It has a global level where it selects warps based on their readiness, similar to GTO. It also has a local level where it enforces fairness by prioritizing the least recently active warp, similar

to LRR. This combination aims to provide good throughput while preventing warp starvation.

The impact of these schedulers on IPC can vary depending on the specific workload being executed on the GPU. Here are some general trends:

GTO: GTO can achieve high IPC when the workload has a lot of thread-level parallelism and uniform resource usage. However, it might result in poor fairness and potential warp starvation for some warps, leading to lower overall GPU utilization.

LRR: LRR is better at preventing warp starvation and ensuring fairness among warps. This can lead to more consistent performance across different parts of the workload, but it might sacrifice some throughput compared to GTO, especially in cases with highly parallel workloads.

Two-Level Active: This scheduler aims to strike a balance between maximizing throughput and

ensuring fairness. It can provide good IPC for workloads with varying levels of parallelism and resource usage. However, its effectiveness depends on how well it manages the trade-off between these two factors.

In summary, the choice of warp scheduler can significantly impact IPC on a GPU. The most suitable scheduler depends on the characteristics of the workload being executed and the desired balance between throughput and fairness. Modern GPUs often use dynamic scheduling techniques that combine aspects of different schedulers to adapt to the workload's requirements dynamically.

Question 2

GTO

Shared

L1D_total_cache_accesses = 32512

L1D_total_cache_misses = 32512

L1D_total_cache_miss_rate = 1.0000

L1D_total_cache_pending_hits = 0

L1D_total_cache_reservation_fails = 0

L1D_cache_data_port_util = 0.000

0.077L2_total_cache_accesses = 32512

L2_total_cache_misses = 16512

L2_total_cache_miss_rate = 0.5079

L2_total_cache_pending_hits = 14574

L2_total_cache_reservation_fails = 0

Naïve

L1D_total_cache_accesses = 128128

L1D_total_cache_misses = 72128

L1D_total_cache_miss_rate = 0.5629

L1D_total_cache_pending_hits = 0

L1D_total_cache_reservation_fails = 0

L1D_cache_data_port_util = 0.041

L1D_cache_fill_port_util = 0.012

L2_total_cache_accesses = 16128

L2_total_cache_misses = 8128
L2_total_cache_miss_rate = 0.5040
L2_total_cache_pending_hits = 8000
L2_total_cache_reservation_fails = 0

LRR
Shared
L1D_total_cache_accesses = 32512
    L1D_total_cache_misses = 32512
    L1D_total_cache_miss_rate = 1.0000
    L1D_total_cache_pending_hits = 0
    L1D_total_cache_reservation_fails = 0
    L1D_cache_data_port_util = 0.000

L2_total_cache_accesses = 32512
L2_total_cache_misses = 16512
L2_total_cache_miss_rate = 0.5079
L2_total_cache_pending_hits = 15956
L2_total_cache_reservation_fails = 0

Naïve
L1D_total_cache_accesses = 128128
    L1D_total_cache_misses = 72128
    L1D_total_cache_miss_rate = 0.5629

L1D_total_cache_pending_hits = 0

L1D_total_cache_reservation_fails = 0

L1D_cache_data_port_util = 0.041

L1D_cache_fill_port_util = 0.012

L2_total_cache_accesses = 16128

L2_total_cache_misses = 8128

L2_total_cache_miss_rate = 0.5040

L2_total_cache_pending_hits = 8000

L2_total_cache_reservation_fails = 0

Two:level:active

L1D_total_cache_accesses = 32512

L1D_total_cache_misses = 32512

L1D_total_cache_miss_rate = 1.0000

L1D_total_cache_pending_hits = 0

L1D_total_cache_reservation_fails = 0

L1D_cache_data_port_util = 0.000

L1D_cache_fill_port_util =

0.074L2_total_cache_accesses = 32512

L2_total_cache_misses = 16512

L2_total_cache_miss_rate = 0.5079

L2_total_cache_pending_hits = 8859

L2_total_cache_reservation_fails = 0

Conclusion:

The time that passes between the start of a request or action and the matching answer or result is referred to as latency. In an operating system, latency is the amount of time that passes between an interrupt occurring and the processor starting to execute in order to handle the interrupt. This measurement is expressed in milliseconds and is specifically defined as the total amount of time that passes between an input or command and the desired result.

When it comes to networking, latency is the amount of time that passes between when a user submits a request for network access and when the user receives a response. The term "latency" describes the amount of time that passes between two separate events.The time a data packet takes to travel from its point of origin to its intended destination is referred to as its latency.

The network latency can be measured using one of two techniques. The first type of latency is called one-way latency, and it is the amount of time that passes between a packet leaving the source and being received at the destination. The individual one-way latency from node A to node B is combined with the individual one-way latency from node B to node A in the alternate category, sometimes known as a round trip.
A DRAM module's MF (maximum memory fetch) delay depends on a number of variables, including the kind of DRAM used, how frequently it operates, and how the memory controller is built. "DRAM latency" is the amount of time that a memory module needs to complete a memory request.

Different DRAM generations, such as DDR3, DDR4, and DDR5, have different latencies. Furthermore, different speed grades may show varying latencies throughout each generation. Commonly, latency is measured in nanoseconds or clock cycles.

The following latency numbers are noted for the three different warp scheduler types in both global and shared implementations:global_gto:350

shared_gto:210

global_lrr:345

shared_lrr:200

global_two_level:375

shared_two_level:204
From the facts provided, it can be concluded that the global implementation has more delay than the shared version, which suggests that the shared memory code performs better.

The two-level architecture has the maximum delay, followed by GTO and LRR. This gives the

rate at which the quality of warp schedulers has improved effectively.