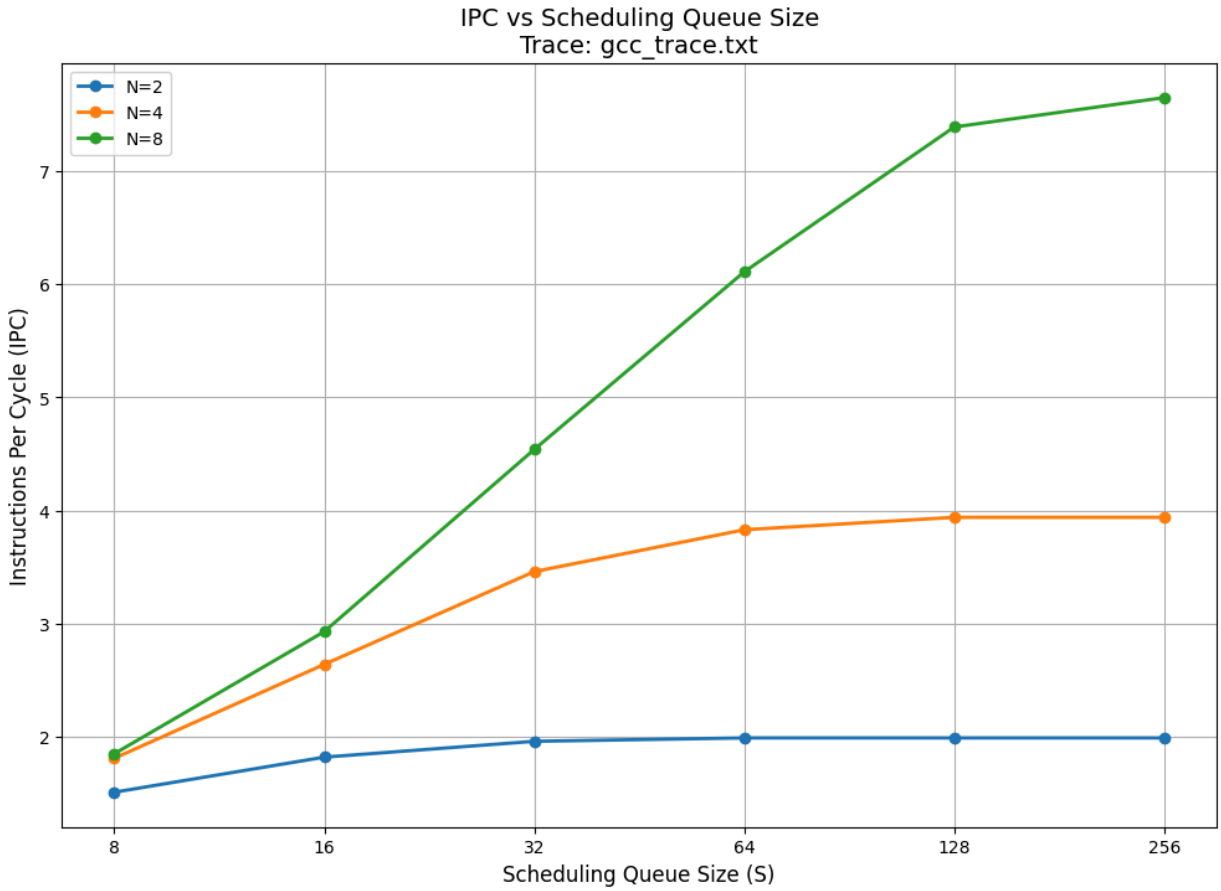
**ASSIGNMENT 3: Dynamic Scheduling Report**
**NAME: VARUN M**
**ROLL NO: EE20B149**

1) **gcc_trace.txt**




**Trends**

1. Increasing N (Number of Execution Units):
- N=2 shows modest IPC (1.5-2.0) with minimal improvement across queue sizes
- N=4 demonstrates better IPC (up to 4.0) with diminishing returns after queue size 64
- N=8 exhibits the highest IPC (reaching 7.5) and continues to scale with larger queue sizes

2. Queue Size Impact:
- Small queues (8-16): Limited IPC across all configurations
- Medium queues (32-64): Sharp improvement for N=4 and N=8
- Large queues (128-256): Only N=8 continues to show significant improvements

Reasons for these trends:

1. Limited Scaling with N=2:
- With only 2 execution units, the hardware cannot exploit available instruction-level parallelism (ILP)

- Even with larger scheduling queues, the bottleneck remains the limited execution resources
- The flattening curve suggests that available ILP cannot be utilized due to hardware constraints

2. Moderate Scaling with N=4:

- More execution units allow better exploitation of ILP
- Plateauing after queue size 64 indicates that the gcc trace has moderate ILP that can be captured with 4 units
- Additional queue space provides diminishing returns as execution resources become the limiting factor

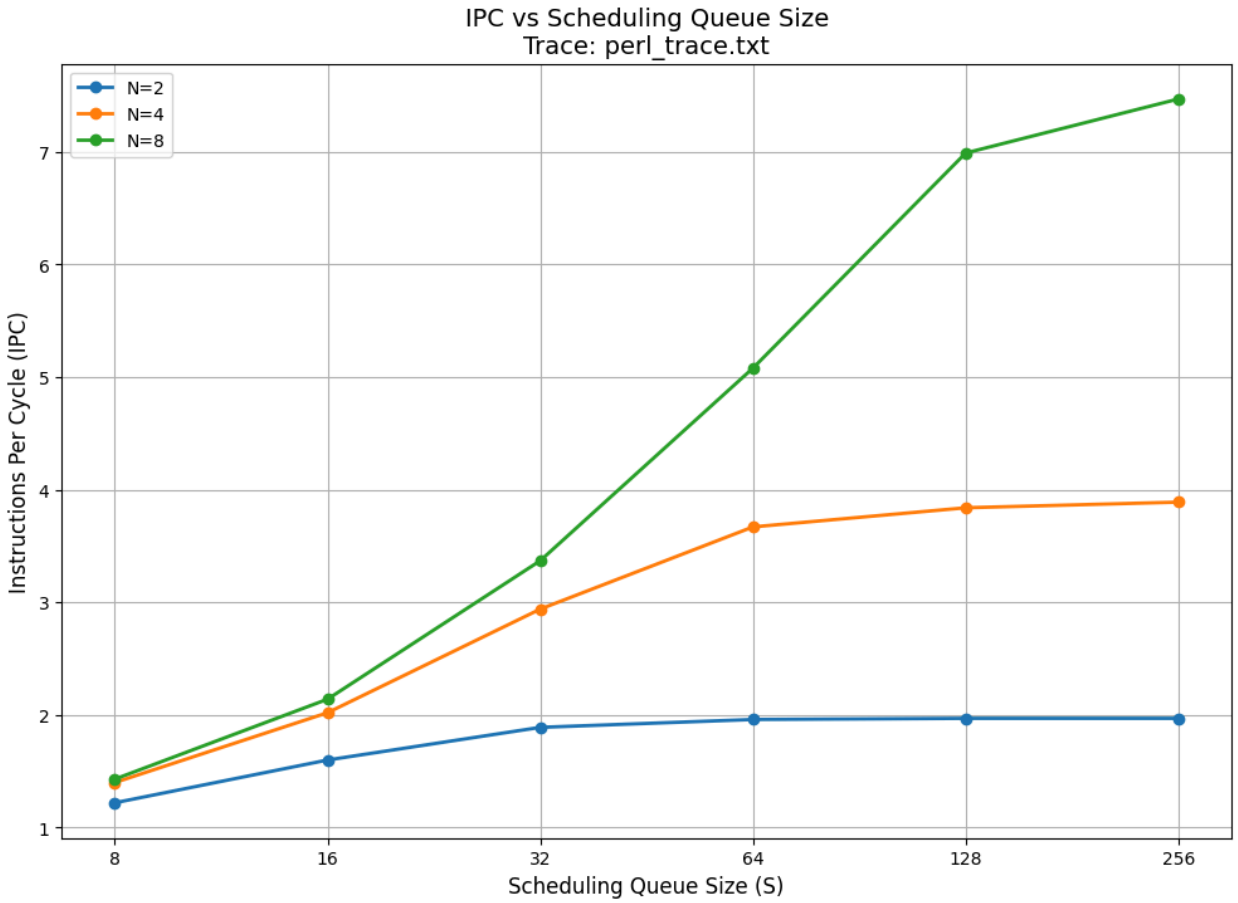
3. Superior Scaling with N=8:

- Continuous improvement with larger queues suggests gcc has substantial ILP
- Eight execution units provide enough hardware parallelism to utilize available ILP
- Larger scheduling windows allow the hardware to find and schedule more independent instructions simultaneously

4. Queue Size Impact:

- Small queues limit visibility of independent instructions, restricting ILP exploitation
- Larger queues allow more instructions to be examined for parallel execution
- The relationship between queue size and IPC is non-linear, showing diminishing returns at different points based on N

**2) perl_trace.txt**




**Trends:**

1. Impact of Execution Units (N):

- N=2 shows limited IPC (1.2-2.0) with minimal improvement across queue sizes
- N=4 achieves moderate IPC (up to 4.0) and plateaus after queue size 64
- N=8 demonstrates highest IPC (reaching 7.5) with continued scaling at larger queue sizes

2. Queue Size Influence:

- Small queues (8-16): All configurations show similar, limited performance
- Medium queues (32-64): Significant divergence between N=4 and N=8
- Large queues (128-256): Only N=8 continues to show substantial improvements

Reasons for these trends:

1. Conservative Scaling with N=2:

- Two execution units create a hardware bottleneck
- Additional scheduling queue space provides minimal benefit
- Available instruction-level parallelism (ILP) cannot be exploited due to limited execution resources
- IPC plateaus around 2.0, indicating resource saturation

2. Moderate Performance with N=4:

- Four execution units allow better ILP exploitation
- Saturation at queue size 64 suggests perl's inherent ILP limitations with 4 units
- The flattening curve indicates reaching maximum extractable parallelism for this configuration

3. Aggressive Scaling with N=8:

- Continued improvement with larger queues indicates perl contains significant extractable ILP
- Eight execution units provide sufficient hardware parallelism
- Larger scheduling windows enable finding and executing more independent instructions
- Near-linear scaling up to size 128 suggests effective ILP utilization

4. Queue Size Effects:

- Limited instruction visibility in small queues restricts ILP exploitation
- Larger queues enable better instruction reordering and parallel execution
- Performance benefits scale with execution units, showing highest impact for N=8

The perl trace exhibits behavior similar to gcc, indicating it's also a workload with substantial instruction-level parallelism that benefits from both increased execution resources and larger instruction windows.

Both traces demonstrate that modern applications can effectively utilize wider superscalar processors when provided with sufficient scheduling queue sizes to expose available parallelism.

**Comparative analysis:**

**For gcc_trace.txt:**

**S value | N=2 N=4 N=8**

**----------**

**8 | 1.220 1.400 1.430**

**16 | 1.600 2.020 2.140**

**32 | 1.890 2.940 3.370**

**64 | 1.960 3.670 5.080**

**128 | 1.970 3.840 6.990**

**256 | 1.970 3.890 7.470**

**For perl_trace.txt:**

**S value | N=2  N=4  N=8**

**----------**

**8 | 1.510 1.810 1.850**

**16 | 1.820 2.640 2.930**

**32 | 1.960 3.460 4.540**

**64 | 1.990 3.830 6.110**

**128 | 1.990 3.940 7.390**

**256 | 1.990 3.940 7.650**

1. Initial Performance (S=8):
- gcc shows higher initial IPC: 1.51/1.81/1.85 (N=2/4/8)
- perl starts lower: 1.22/1.40/1.43 (N=2/4/8)
- gcc demonstrates ~24% better initial performance
2. Mid-range Scaling (S=32):
- gcc: 1.96/3.46/4.54
- perl: 1.89/2.94/3.37
- gcc maintains higher IPC, with ~35% better performance for N=8
3. Maximum Performance (S=256):
- gcc: 1.99/3.94/7.65
- perl: 1.97/3.89/7.47
- Performance gap narrows at larger queue sizes, with only ~2% difference
4. Scaling Characteristics: For N=2:
- Both traces saturate quickly (gcc at 1.99, perl at 1.97)
- Minimal improvement beyond S=32

For N=4:

- Both show similar saturation points (S=64)
- gcc achieves slightly higher peak IPC (3.94 vs 3.89)

For N=8:

- Both demonstrate continued scaling up to S=256
- gcc maintains a slight performance edge throughout

Conclusion:

1. Common Trends:

- Both traces show similar scaling patterns
- Performance benefits diminish with smaller N values
- Larger queue sizes benefit most from higher N values
- N=8 configuration shows the best scaling for both traces

**2.** Key Differences:

- gcc shows better initial performance across all configurations
- gcc maintains higher IPC throughout, though the gap narrows at larger scales
- perl shows more dramatic improvement from small to large queue sizes for N=8

**3.** Architectural Implications:

- Both workloads benefit significantly from increased execution resources
- Scheduling queue size has a substantial impact on performance
- Optimal queue size depends on the number of execution units
- N=8 with large queue sizes (≥128) provides the best performance for both workloads

The data suggests that while gcc and perl have different absolute performance characteristics, they exhibit similar scaling patterns and resource utilization behaviors. This indicates that modern processor designs with multiple execution units and large scheduling queues can effectively exploit instruction-level parallelism across different types of applications.