Meeting 1: Initial Research Meeting Discussion 11/1

1. Project

1.1 Project Goals

Develop performance models that predict per-instruction execution time or overall IPC using **TACIT's detailed microarchitectural traces**.

1.2 Potential Approaches

1. Hybrid Analytical + ML (Inspired by Concorde)

Build analytical performance models for core microarchitectural structures (ROB, LSQ, bandwidth limits) and use ML to fuse them or model residuals.

2. Pure Transformer (Fully ML)

Autoregressive or long-context transformers trained on TACIT traces to predict performance directly, without analytical structure.

1.3 Motivation

- Hybrid Analytical + ML (Choosing this one)
    - Pros:
        - Analytical models encode known architectural constraints (ROB fullness, LSQ pressure, issue bandwidth, cache miss penalties).
        - Provides interpretability ("This window is ROB-bound", "LSQ stall dominates").
        - Improves generalization beyond training workloads.
        - Concorde shows hybrid systems outperform pure ML when modeling engineered systems like CPUs.
    - Cons:
- Pure Transformers
    - Pros:
        - TACIT provides thorough microarchitectural data (per-instruction events, deps, misses).
        - Large models may implicitly learn architectural phenomena.
        - Simpler pipeline to implement
        - Good for exploring scaling laws and limits of model capacity.

- Cons:
  - Minimal Interpretability
  - May memorize rather than generalize
  - Needs large compute + data

2. Implementation Plan / Potential Next Steps

2.1 Hybrid Analytical + ML

Stages for implementation:

1. Analytical Modeling (Creating the Performance Models) [Try to implement before next meeting]
    a. Implement ROB model
    b. Implement LSQ (load/store) models
    c. Build bandwidth/width constraints
    d. Add dynamic constraint models (misses, mispredicts, TLB)
2. Feature Extraction
    a. Write TACIT -> feature pipeline
    b. Extract dependency graphs, event markers
3. ML Integration
    a. Train ML to:
        i. Fuse analytical models
        ii. Predict Residuals
        iii. Classify window-level bottlenecks
    b. Validate on unseen programs/new traces
    c. Produce interpretability visualizations