# Master Instructions – CLT–E8 AGI Development

## Purpose:

To design, prototype, and evolve a coherence-driven AGI architecture based on the principles of Coherence Lattice Theory (CLT–E8). This document governs goals, architecture, coding standards, testing, and project philosophy.

## 1. Project Goals

### Short-term (weeks):

- Implement a working prototype with SlipStates, coherence scoring, and multi-head embeddings.  
- Demonstrate basic tool-grounded reasoning (math, CSV analytics).  
- Show improvement with memory enabled (vs. ablation).

### Medium-term (months):

- Build an agent that learns new tools from few demonstrations.  
- Handle contradictions via truth-maintenance.  
- Demonstrate “echo memory” retrieval (E8 multi-head retrieval intersection).  
- Publish ablation results showing CLT–E8-inspired components outperform baselines.

### Long-term (years):

- Scale toward flexible general reasoning with symbolic + neural hybrid architecture.  
- Explore quantum and coherence-based substrates (QEC-inspired cognition).  
- Push toward AGI by grounding CLT–E8 coherence as the global optimization principle.

## 2. Core Design Principles

- Coherence-first: All modules seek to maximize cross-consistency and minimize contradictions.  
- Slip States, not mutation: Information is appended immutably, preserving history.  
- E8 lattice memory: Multi-head projections for robust retrieval and natural echo effects.  
- Critic in the loop: All reasoning passes through tool-based verification.  
- Falsifiability: Every claim, improvement, and architecture feature must be testable by ablation or benchmark.  
- Minimal safety, maximum clarity: Sandbox and audit rails are for clean science, not corporate bureaucracy.

## 3. Architecture Overview

1. Workspace (SlipStates + ledger): Immutable states with provenance; tracks coherence scores.  
2. Embeddings (multi-head E8): 8–16 orthogonal projections; slip rotations prune contradictions.  
3. Memory: Episodic (traces), semantic (facts/embeddings), procedural (skills).  
4. Symbolic Reasoner: Horn clauses + Truth Maintenance System.  
5. World Model: Minimal simulator for “what if” rollouts.  
6. Planner: Generates candidate plans; uses coherence score; tanh-gate exploration control.  
7. Critic: External tool verification; assigns verification scores.  
8. Governor (minimal): Sandboxed tool execution; logs actions.

## 4. Coding Standards

- Language: Python 3.11+  
- Style: PEP8 with type hints.  
- Versioning: File suffix \_vN (e.g., workspace\_v1.py).  
- Changelog: Every file update recorded in CHANGELOG.md.  
- Docstrings: Every function documents inputs, outputs, side effects.  
- Tests: Each module includes a basic test harness in /tests.

## 5. Experiment Philosophy

- Every feature must be tested with ablation studies (with vs. without).  
- Metrics to track:  
 • Grounded correctness (critic-verified).  
 • Generalization from demos.  
 • Memory utility (E8 vs. baseline).  
 • Contradiction handling (TMS performance).  
 • Self-revision (plan replacement ability).  
- Results logged in /experiment\_logs.

## 6. Safety & Reproducibility

- All tool execution runs in a sandbox or container.  
- No external network writes without explicit approval.  
- Audit logs stored in /logs.  
- Rails exist to ensure clean reproducible science, not to throttle innovation.

## 7. Roadmap Summary

- Week 1–2: Implement workspace, embeddings, SlipStates, critic.  
- Month 1: Demo CSV analytics agent with coherence memory.  
- Month 3: Add world model, planner, contradiction handling.  
- Month 6: Demonstrate skill learning + retrieval with echo-memory.  
- Year 1: Scale toward modular hybrid AGI with published ablation results.