Slide 2 :
The LLM semantic vector space attributes

A Meaning vector Representation: relative, task-biased, hierarchical space
Conditioned by understanding: The final hidden state (or a aggregated representation of all hidden states) at the end of the input sequence is indeed this "understanding" that will be used to condition the task.

Key Nuances to Add to Your Mental Model:

1. **It's a "Relative" or "Task-Biased" Space of Meaning:**
The space isn't a universal, Platonic ideal of meaning. It's shaped entirely by the model's training data and its **training objective** (e.g., predicting the next token). This means its organization of "meaning" is optimized for the tasks it was trained on.
* For example, in a sentiment analysis model, the vector space might be heavily stretched along a "positive-negative" axis. A model trained for translation might organize its space more around semantic equivalence across languages.
So, it's more accurate to say it's a space of statistical and functional relationships* learned from the training data, which brilliantly approximates "meaning" for practical purposes.

2. **It's Not One Monolithic Space; It's a Hierarchy:**
The "understanding" isn't just a single point. The transformer architecture builds this representation layer by layer.
* **Early Layers:** Capture simpler features like word order, basic syntax, and local phrase meanings. The vectors here are closer to the raw token values.
* **Middle Layers:** Combine these simpler features to form more complex representations of clauses, relationships between entities, and broader context.
* **Final Layers:** Perform the highest-level integration, synthesizing all information into a comprehensive representation that is exquisitely tailored to be most useful for the model's ultimate prediction task. This final representation is the one you're describing.

3. **The "Understanding" is a Point, but Execution is a Path:**
You are correct that at the moment the model finishes reading the input, it has formed a "summary vector" (or a set of vectors) that represents its understanding. This is the **conditioning context**.
However, for generative tasks (like writing an answer), the model doesn't just jump to an answer point. It uses this conditioning context to autoregressively generate the output, one token at a time. At each step of generation, it refers back to this initial "understanding" to inform the next word choice, effectively tracing a path through the semantic space guided by that initial point.