Instructions vs. Data Streams: You're on the right track with the idea of having a continuous signal dedicated to running something like an interactive shell. It does make sense to have one part of the system always ready to process incoming data or user input while the ALU focuses on task execution. You could think of the signal as a sort of low-level “heartbeat” of the system, ensuring the shell is responsive. In an analog world, this signal could also carry different forms of modulation—frequency for instructions, amplitude for data, or vice versa. This could allow you to "weave" multiple streams within a single channel, which is like multiplexing but with an analog twist.

Analog Memory vs. Binary Memory: In a binary system, as you noted, the ALU retrieves values, operates on them via logic gates, and sends them back to memory. The challenge in an analog system is different because you're capturing continuous ranges of values. Capacitors can indeed be used for short-term memory since they can store a voltage for a certain amount of time. The “state” in analog terms would likely be the voltage (or current) level at each storage point. The tricky part is keeping that value stable without degradation.

You might store data in an analog system by measuring voltages across a network of capacitors, where each voltage level corresponds to a different data value. Long-term storage would require constantly refreshing or “holding” those values, so you'd need a method of stabilization. This is why, traditionally, analog systems have struggled with long-term memory compared to digital ones.

Universal Gates: The idea of using purpose-built or universal gates is powerful. If each gate can perform multiple types of operations, you're essentially introducing flexibility at the hardware level. This could be particularly useful in an analog environment because instead of binary logic (where each gate is rigidly tied to a specific operation), analog gates could be “programmed” by altering the energy signature or component configuration to perform different tasks. In a way, you're moving towards reconfigurable hardware, like Field Programmable Gate Arrays (FPGAs) but in the analog domain.

Parallel Processing & Machine Learning: The analogy to neural networks is insightful. Yes, in machine learning, especially in deep learning, nodes (neurons) are adjusted or "tuned" by adjusting weights. In your system, each universal gate could act like a node, and the interconnections between them could allow for flexible, dynamic problem-solving. The question becomes: how do you "train" these analog gates? Instead of adjusting weights, you might be adjusting energy parameters (voltage, current, etc.) at each gate, essentially tuning the system to recognize patterns or solve specific types of problems, much like a neural network does with data.

Parallelism: With universal gates acting independently, parallel processing could become a powerful advantage. Each gate could solve a part of the problem simultaneously, potentially making this system very efficient for tasks like differential equations (as you mentioned). In fact, certain types of mathematical problems, especially ones involving continuous data or real-time adjustments, could be more naturally solved in this setup than in traditional digital computers.
Analog Memory Beyond Capacitors: While capacitors are great for short-term analog memory, longer-term solutions might involve feedback mechanisms that "lock in" a value using a combination of inductors, resistors, and other analog components. I imagine this could be a way to “refresh” and stabilize memory values without too much degradation. You might also explore hybrid storage mechanisms where digital elements help maintain analog values over

time.

Food for Thought While Driving:
Imagine how you'd design a system where multiple independent signals (whether analog or hybrid) flow through the same circuit, each carrying different types of data (instructions, raw data, shell state). Could these signals overlap, or would they need to remain entirely distinct?

How might feedback loops (a common component of analog circuits) be used in your memory system? Could feedback stabilize values in a way that mimics refreshing memory in DRAM?

What about error correction? In digital systems, error correction codes (ECC) are used to keep data reliable. What's the analog equivalent? Would you embed correction mechanisms in the gates or memory components to account for noise or signal degradation?

If the gates are universal, would they "learn" or "adapt" in some way based on the problem they're solving? Could tuning these gates for different operations offer insight into pattern recognition, similar to ML?

Is there a pathway from this analog framework to quantum computing? Quantum systems deal with superpositions and entanglement, while your analog system deals with continuous values. Could there be a bridge where they intersect, either conceptually or functionally?