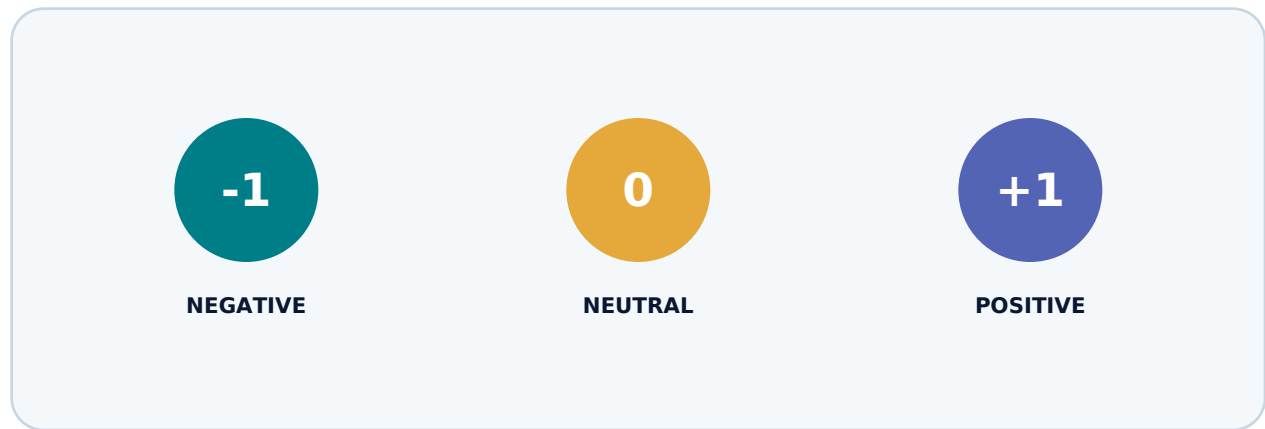


RADIX

BEYOND BINARY COMPUTING

A speculative architecture for computation using richer numerical states, relational execution, and adaptive precision.



What if the two-state computer is not the destination - just one successful starting point?

A research concept and invitation to experiment | October 2026

QUESTION TO EXPLORE

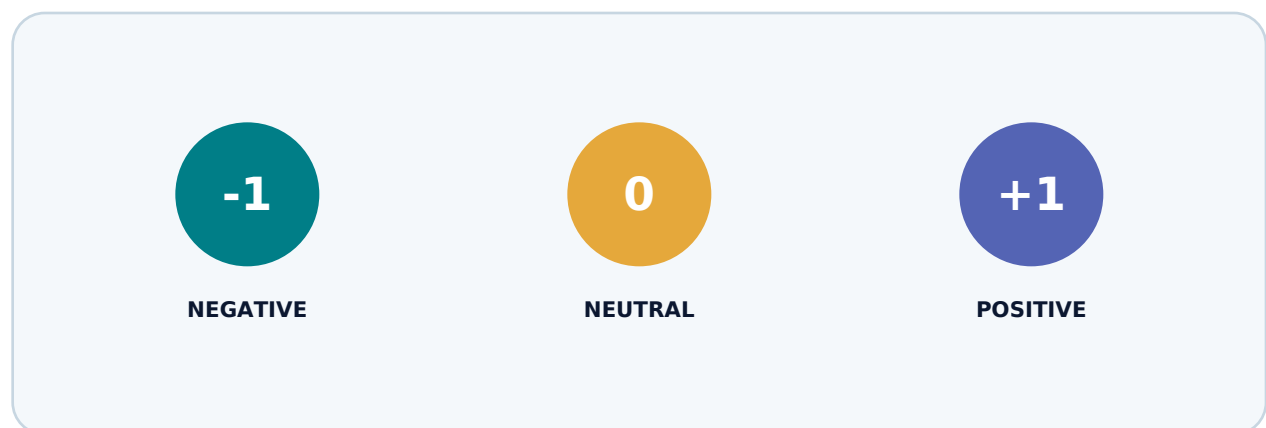
If we could redesign computing without inheriting today's hardware conventions, what would we keep - and what would we abandon?

01 / The assumption worth questioning

Binary is practical because physical switches can distinguish two states reliably. That is a powerful engineering choice, not a proof that binary is universally optimal. Any alternative must account for physical error rates, energy, circuit complexity, memory movement, and the work being computed.

The RADIX hypothesis

RADIX (Relational Adaptive Dynamics and Information eXecution) combines balanced-ternary arithmetic, incremental relational computation, explicit uncertainty, and mixed precision. Its ambition is to optimize useful computation rather than insist on one representation everywhere.



A trit stores $\log_2(3) =$ approximately 1.585 bits of information. Twenty trits encode 3^{20} possible states, more than 31 bits but fewer than 32 bits. This measures representational capacity, not speed, efficiency, or storage density in a real device.

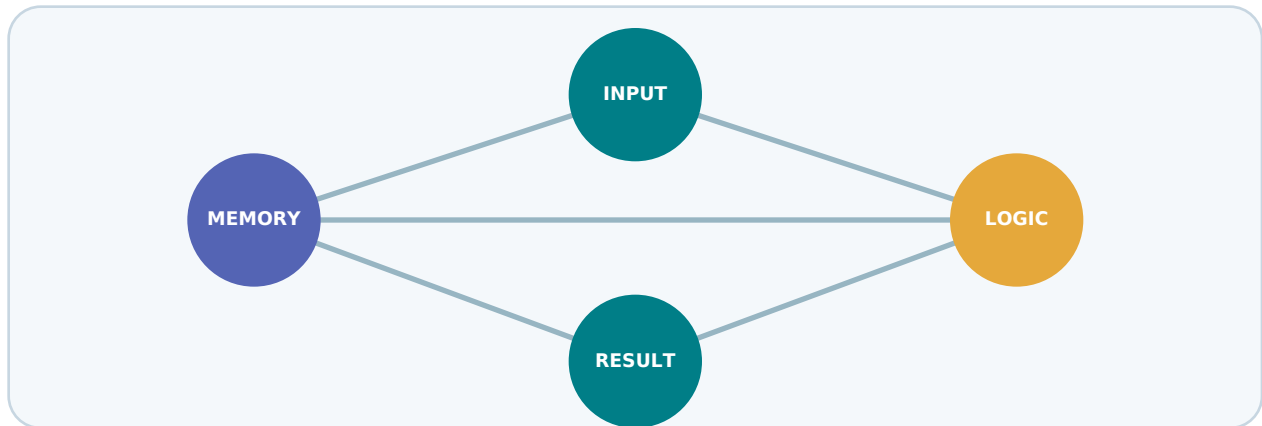
QUESTION TO EXPLORE

When would the benefit of a third state outweigh the additional difficulty of reading it reliably?

Design guardrail: uncertainty is not arithmetic zero. A separate logical type should represent true, false, and unknown.

02 / Move the computation to the relationships

Conventional workloads often spend substantial energy transporting values between memory and processors. RADIX would represent dependent calculations as a graph, updating only affected regions as inputs change. This builds on established dataflow, incremental computing, and processing-in-memory ideas.



Node: value, type, precision, and dependencies. Edge: a relationship or data dependency. Execution: propagate changes through affected nodes, with scheduling informed by locality and cost.

QUESTION TO EXPLORE

Could we reduce memory traffic more effectively by changing where calculations happen rather than making arithmetic gates faster?

The hard problem

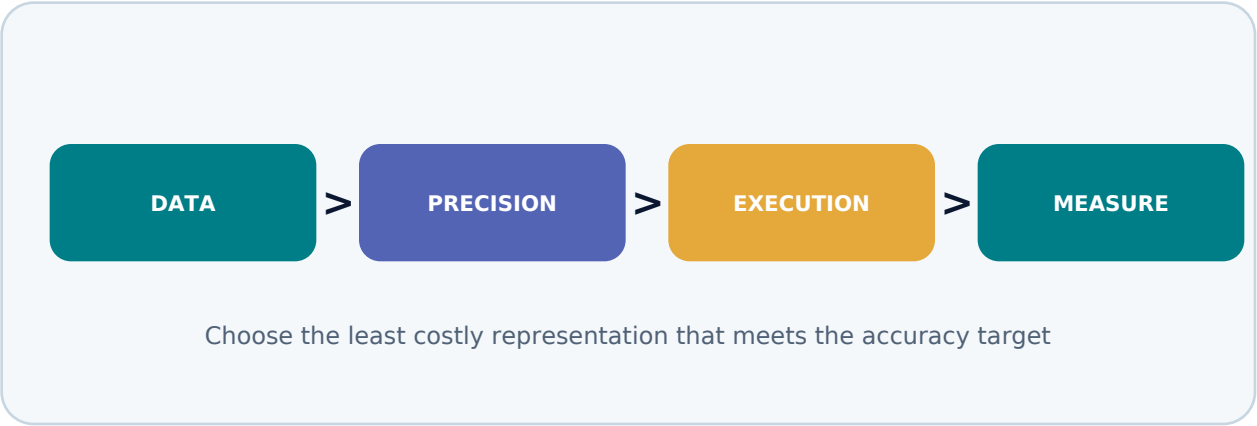
Graph scheduling, synchronization, and communication can erase the gains. A benchmark must include graph-management overhead and compare against highly optimized conventional implementations.

QUESTION TO EXPLORE

Which workloads change a small fraction of their inputs but repeatedly recompute nearly everything?

03 / Let precision follow the problem

Different tasks require different guarantees. A financial ledger needs exact values. An AI model may tolerate quantization. A scientific simulation may need dynamically adjusted precision. RADIX would ask the compiler to select an execution strategy subject to an explicit error budget.



WORKLOAD	POSSIBLE STRATEGY
Ledger / accounting	Exact integer or decimal arithmetic
AI inference	Ternary / low-precision weights where validated
Scientific simulation	Adaptive precision with bounded error
Streaming sensors	Incremental graph updates
Incomplete evidence	Explicit uncertainty propagation

QUESTION TO EXPLORE

Should a compiler optimize for speed, energy, memory, reliability - or a user-defined balance of all four?

04 / What programming could look like

An experimental language, RDX, could describe relationships and accuracy requirements rather than forcing programmers to specify every intermediate calculation. This is illustrative syntax, not a running implementation.

```
sensor temperature: real
sensor humidity: real
```

```
derive comfort:
  from temperature, humidity
  precision adaptive
  uncertainty propagate
```

```
when comfort > 0.8:
  emit "Comfortable"
```

```
when comfort is unknown:
  request more_data
```

QUESTION TO EXPLORE

What should the programmer declare explicitly, and what should the runtime be trusted to choose automatically?

What must be proven

A useful implementation would need clear semantics, reproducible results, error handling, profiling, and a compiler that makes better decisions than an expert choosing precision manually.

05 / Three experiments to falsify the idea

Experiment A: Ternary arithmetic. Implement balanced-ternary encoding and operations. Compare correctness, memory footprint, throughput, and conversion overhead against optimized binary arithmetic.

Experiment B: Incremental graph execution. Build a dependency engine and test workloads with sparse input changes. Compare total runtime and memory traffic against recomputing from scratch.

Experiment C: Adaptive precision. Compare ternary-quantized, binary, and standard low-precision AI models at matched quality. Measure latency, energy where hardware instrumentation permits, and end-to-end cost.

QUESTION TO EXPLORE

What negative result would convince us to abandon or radically redesign RADIX?

A rigorous success criterion

Claim an advantage only when a reproducible benchmark shows improvement in at least one measured dimension without hiding unacceptable regressions in accuracy, reliability, or other costs. Simulation results must not be presented as hardware energy savings.



ELECTRONIC



OPTICAL



OSCILLATOR

Beyond digits: optical interference, coupled oscillators, and physical energy minimization might someday provide better primitives for selected tasks. Each introduces its own constraints.

06 / The open frontier

The most radical possibility is that the best computational primitive is not a digit at all. Instead of encoding a problem and repeatedly manipulating symbols, a physical system could evolve toward a solution. The key challenge is whether encoding, control, readout, and error correction cost more than the computation saved.

QUESTION TO EXPLORE

What if the fundamental unit of computing were a relationship, a wave, or a physical constraint rather than a stored number?

QUESTION TO EXPLORE

Could one machine combine conventional digital logic for exact work, multistate arithmetic for dense computation, and analog physics for optimization?

QUESTION TO EXPLORE

If energy were the scarce resource and time were abundant, how would we redesign the entire computer?

QUESTION TO EXPLORE

Can a system discover its own most efficient representation for a new class of problems?

A practical first milestone

Build RADIX 0.1 as a software prototype in Python or Rust: a ternary arithmetic library, a typed dependency graph, an adaptive precision policy, and a benchmark suite. Publish both wins and losses. Only then consider specialized hardware.

Closing thought

The objective is not to prove that humans were wrong to choose binary. It is to test whether a more flexible foundation can outperform binary-first designs for specific real-world problems.

Research concept - not a validated new computing architecture. Existing research includes ternary computing, dataflow architectures, quantization, and analog/optical computation.