The Search for JouleOS

David Charlot, Open Interface Engineering

Product white paper (review and design guide) · Open Interface Engineering · research.openie.dev

Revision `1561a2a` (22 November 2026) · Web: research.openie.dev/products/jouleos/ · Archival LaTeX: /pdfs/latex/jouleos.pdf (.tex)

Measurement. Product white paper (review and design guide), kept apart from the research catalog; not a journal final. Reviews operating-system, instruction-set, hypervisor, unikernel, and agent-sandbox composition, specifies checkable capability-joule requirements, and verifies the open Rust reference spine (`jcp-satb-cad-spine`, **`310` workspace tests**, `78,400`-pair provenance meet-semilattice) bridging `JCP` (`JCR-1` canonical `det-CBOR` + `Ed25519` receipts), `SATB-IO` (`L0--L4` × `A1--N` + `Falsifier 6` drift gates), and `openie-cad` stage accounting. Host verification counters are `reported_j` and analytical schedules are `est_j`; no physical board rail reading is claimed (`board_metered = false`, `measured_j = None`). No fabricated citations.

A review of how operating systems, instruction sets and agent sandboxes are composed today, and the requirements a machine must meet to schedule, meter and refuse an agent under an energy budget.

Contents

Abstract **2**

1. What this study is **2**
Terms used here . . . 3

2. The view that assumes the floor is fixed **3**
2.1 NVIDIA OpenShell on top of an existing agent . . . 3
2.2 OpenHuman: a Rust harness whose core stays on the host . . . 3
2.3 The class, not the brand . . . 4

3. The teaching line: silicon to human interface **4**

4. Where the teaching line already splits **5**
4.1 Host computer and microcontroller . . . 5
4.2 Assembly and the compiler . . . 5
4.3 Firmware, hardware abstraction, and the operating system . . . 5

5. Operating systems are not one thing **5**
5.1 Kernel . . . 5
5.2 Distribution . . . 6
5.3 Build system . . . 6
5.4 Middleware . . . 6
5.5 Hypervisor . . . 6
5.6 Verified kernel and separation kernel . . . 6
5.7 Real-time kernel . . . 6
5.8 Teaching kernel and research kernel . . . 6
5.9 Library operating system and unikernel . . . 6
5.10 Other host operating systems, by job . . . 7
5.11 What not to collapse . . . 7
5.12 Three names that stay unresolved . . . 7

6. Instruction sets are contracts that move **7**

7. What the AI install path treats as fixed **8**

8. The constellation by job **8**

9. The handful of contracts **8**
9.1 The instruction contract . . . 8
9.2 The machine's law . . . 8
9.3 The build that produces an image . . . 8
9.4 The guest that has no silicon of its own . . . 9
9.5 The translation layer . . . 9
9.6 The human surface . . . 9
9.7 What does not fit as a seventh . . . 9

10. The agent under those contracts **9**

11. The standards that speak the schedule, the law, the commit, and the ladder **9**
11.1 Energy-Oriented Computing is the schedule . . . 9
11.2 The Joule Context Protocol is the law . . . 10
11.3 JouleContract is the commit . . . 10
11.4 The sandbox standard is the S0 to S7 ladder . . . 10
11.5 Why a patch that omits these is still a patch . . . 10

12. What the problems require **11**
12.1 What the README says it is . . . 11
12.2 Single-level store . . . 11
12.3 Cascade under Energy-Oriented Computing . . . 11
12.4 Capability ledger . . . 12
12.5 Energy oracle with provenance . . . 12
12.6 Unified Design Architecture, and the runtime that is seen . . . 12

13. Design guide: what to require **13**

14. Claim ledger **13**

15. JouleOS **14**

Abstract

This study reviews how kernels, builds, guests and instruction sets are composed today, shows that none of them makes energy a condition of an agent's action, and lists checkable requirements for a machine that would.

In plain words

1. Today's computers stack many separate layers, and AI agents are added on top without changing what is underneath.
2. None of those layers asks, before an agent acts, how much energy the action will use or whether it is allowed.
3. A machine built for agents should measure energy, give each agent a strict budget, and refuse anything that breaks it.

Computers are hardware. Software is applied engineering written for that hardware under constraints.

---

1. What this study is

This study is the account of existing compute fabric and the software written for it. The problems of that fabric and that software are the point. A machine that schedules, meters, and refuses an agent under law does not exist on the stack as it is composed today. The need those problems state is a solution of the JouleOS kind. The study is not a document of that project.

It is a review of how secure compute is composed today. It is a design guide for what a machine must speak if an agent is to be scheduled, metered, and refused under law. It is not a ranking of kernels. It is not a slogan about green computing.

The ranking rule used throughout is stated once. Among systems that pass a task's correctness test, systems are ordered by energy to completion, in joules. Latency, token counts and benchmark scores are not the unit that decides whether an action is allowed.

Terms used here

The software reference implementation is the software path that tests a study claim without board synthesis or package metering. Value of information is the expected gain in a decision's payoff from observing a quantity before deciding (R. A. Howard, "Information Value Theory," IEEE Transactions on Systems Science and Cybernetics 2(1), 1966, doi:10.1109/TSSC.1966.300074)). Where a statement rests on the OpenIE authors' unpublished analysis, the text says so. It is a definition or a reading, not proof by private path.

The teaching order and the six contracts come from the OpenIE authors' unpublished analysis of the stack, drawn before any destination was named. Every placement of a named system rests on that system's own public page, cited where it appears.

The standards that name the schedule, the law, the commit, and the isolation ladder are public draft specifications at transaction.science: eoc.transaction.science, jcp.transaction.science, joulecontract.transaction.science and sandbox.transaction.science. JouleOS is named where a solution of that kind is the answer the problems require.

---

2. The view that assumes the floor is fixed

Many projects make a view of the agent-safety problem while assuming a layer underneath is fixed. The kernel stays. The process model stays. The install path stays. The agent is wrapped. The wrap is the product. Two concrete cases already in the source material show the class.

2.1 NVIDIA OpenShell on top of an existing agent

On 28 September 2026 NVIDIA announced the Open Agent Safety Platform. OpenShell is the open-source secure runtime in that platform. The public technical blog and investor release describe a sandbox, a supervisor outside the agent workload, policy controls, and an optional out-of-band watchdog named Sentry that can run on BlueField data processing units. The agent remains a child. The host operating system and its privileged law remain the floor. OpenShell sets boundaries for agents running on central processors. Sentry adds monitoring and quarantine from a separate silicon domain when BlueField hardware is present. OpenShell can run without that hardware. The common mechanism is clear: safety is a runtime layered onto an existing agent stack, not a rewrite of the machine's law.

Sources for this placement: NVIDIA Technical Blog, "Add Runtime Controls to AI Agents with NVIDIA OpenShell," 28 September 2026 (https://developer.nvidia.com/blog/add-runtime-controls-to-ai-agents-with-nvidia-openshell/); NVIDIA investor press release dated 28 September 2026 (https://investor.nvidia.com/news/press-release-details/2026/NVIDIA-Launches-Open-Agent-Safety-Platform-to-Secure-Agents-From-Testing-to-Deployment/default.aspx); NVIDIA OpenShell documentation on how the gateway, supervisor, and sandbox split control and enforcement (https://docs.nvidia.com/openshell/dev/about/how-it-works).

2.2 OpenHuman: a Rust harness whose core stays on the host

OpenHuman is an open-source agent harness with a Rust core, licensed under GPL-3.0, and described by its README as early beta (https://github.com/tinyhumansai/openhuman). The same core runs a desktop app, a browser UI, a terminal client, and an embeddable library. The density figures in that README are memory measurements from the project's own fleet sweeps and cold-start timings.

The sandbox design in the project's own pull requests and security docs is the common mechanism that matters here. Sandbox backends are named None, Local, and Docker. Local uses Landlock on Linux, Seatbelt on macOS, or AppContainer on Windows, with a documented no-op fallback where the host provides no jail. Docker runs ephemeral containers with hardened defaults when that path is selected. ReadOnly resolves to None for execution enforcement: it does not add a jail. Elevated tools such as git, install, docker management, and process operations

are defined to leave the jail and run on the host path. The core always runs on the host. The agent is confined where the harness chooses to confine tool execution. The host kernel and process model remain the floor.

Sources: OpenHuman README (https://raw.githubusercontent.com/tinyhumansai/openhuman/main/README.md); OpenHuman pull request 3261 describing sandbox backends and the elevated path (https://github.com/tinyhumansai/openhuman/pull/3261); OpenHuman security architecture page (https://tinyhumans.gitbook.io/openhuman/developing/architecture/security).

2.3 The class, not the brand

OpenShell and OpenHuman are not the only members of the class. They are two named instances. The common mechanism is the same. A view of agent safety is built while the privileged software under the agent is treated as settled. Mixture of Limits cannot be only a supervisor beside that settlement. Metabolic Intelligence cannot be only a cheaper model behind the same install path. The floor has to be part of the review.

---

3. The teaching line: silicon to human interface

Before any product name, teach the line. The OpenIE authors call it the teaching stack. It is a teaching order, not a census of products. Bottom to top:

1. Silicon
2. Chip
3. Board
4. Firmware
5. Instruction set
6. Hardware abstraction
7. Operating system
8. Compiler and runtime
9. Programs
10. Human interface

Silicon is the material. The chip is that circuit as a part. The board is the chip with power, clocks, memory, and connectors. Those three are the computer. They exist whether or not any program is present.

Firmware is the first software written for that board. It brings the board up and talks to the devices on it. It is applied engineering. It is not a layer of the silicon.

The instruction set is the contract the chip offers. It names the operations the chip will perform and the rules for registers, memory, and traps. Assembly sits beside that contract. It is text that names the chip's operations directly. On a host computer a compiler usually emits it. On a microcontroller it is often where the software product stops. The teaching line does not invent an eleventh numbered rung for assembly. Assembly sits beside the instruction set.

Hardware abstraction is software that hides board and chip differences from the software above it. Vendor documents do not all place it on a free-floating rung under the operating system. Microsoft's driver documentation places a hardware abstraction layer inside Windows. CMSIS and a vendor HAL such as the STM32CubeF4 HAL are libraries, not operating systems. The teaching line keeps the order from the OpenIE stack analysis. The later contracts refuse to promote hardware abstraction into a seventh kind of thing.

An operating system, on this line, is software that manages the computer for other programs. That rung is not one job. The separation is by job, not by fame.

A compiler turns a person's text into instructions for a contract. A runtime stays with the program while it runs. Both are applied engineering for a hardware contract. They are not the computer.

Programs are the work. An agent is a program. It occupies this rung. It is not a new kind of matter and not an operating system.

The human interface is where a person meets that work: a screen, a keyboard, a pointer, a voice path, or a serial console. The teaching line ends at a person. Agents, sensors, and other machines are real clients of that computer. They are not rungs above the person. They are other stacks, or guests of a stack.

Source for the ten-rung order: the OpenIE authors' unpublished stack analysis, sections 2, 3 and 7. Named systems are placed by their own public pages.

---

4. Where the teaching line already splits

The line looks like one ladder. It already splits three ways in the working study.

4.1 Host computer and microcontroller

A microcontroller often stops before a host operating system. Its honest stops are firmware and assembly alone, or a small real-time kernel that is not a host product. FreeRTOS and Zephyr are kernels for that region. seL4's own questions page says using seL4 without a full memory management unit makes little sense, because its resource management is based on virtual memory. Different machines. Different stops. One teaching line.

4.2 Assembly and the compiler

Assembly and a compiler are not two names for one rung. On the microcontroller line, the compiler often ran on a host and its output was loaded onto the board. On the host line, the compiler and runtime are their own rung far above the instruction set. Yocto's introduction documents the same split from the build side: a build host produces an image for different hardware. The compiler rung and the instruction-set rung are often on different machines.

4.3 Firmware, hardware abstraction, and the operating system

Firmware brings a particular board up. Hardware abstraction hides differences. An operating system manages the computer for other programs. One product can contain all three. Collapsing them is how the ten rooms look finished when they are not.

Source: the OpenIE authors' unpublished stack analysis, section 3. Named systems are placed by their own public pages.

---

5. Operating systems are not one thing

The teaching-stack rung named "operating system" collects software that does not share a job. Organize by job. Do not rank. Do not sort by country or laboratory.

5.1 Kernel

A kernel is software that runs with the hardware's privileged authority and manages resources for other software. seL4's questions page uses that definition: kernel code is code running in the privileged mode of the hardware.

Linux is a kernel. The README shipped with the kernel source says the Linux kernel is the core of any Linux operating system, and separates the distribution maintainer's job from the kernel's. XNU is a kernel. Apple's XNU README places it inside Darwin for use in macOS and iOS. macOS is the operating system a person boots. XNU is not the whole product. The Windows kernel is the privileged core Microsoft's driver documentation speaks about when it says "the rest of the operating system" above the hardware abstraction layer. This study uses the name on that page.

5.2 Distribution

A distribution is a packaged system: a kernel plus the libraries, programs, and policies that make a bootable product. Poky is the Yocto Project's reference distribution. The project says Poky is not a product-level distribution. It is a starting point.

5.3 Build system

The Yocto Project is a build system for making a custom Linux-based system for an embedded product. BitBake is the task engine. Yocto is not a kernel. The kernel inside an image it builds is Linux. Calling Yocto an operating system collapses the build into the thing it builds. Buildroot is another tool that generates an embedded Linux system by cross-compilation. west is Zephyr's meta-tool. Neither is the kernel it helps build.

5.4 Middleware

Robot Operating System 2 is middleware. The ROS 2 documentation says, in its own words, that despite the name ROS is not an operating system in the traditional sense. It is a set of tools and libraries for building robotic applications. Nodes may sit in one process, in different processes, or on different machines. ROS 2 sits on Ubuntu, Windows, or macOS. A ROS distribution is a packaged set of that middleware. It is not a Linux distribution and not a kernel.

5.5 Hypervisor

A hypervisor runs other operating systems as guests on one machine. Xen is a type-1 hypervisor. KVM is Linux doing the hypervisor job as a kernel module. bhyve is the hypervisor in the FreeBSD base system. Hyper-V is Microsoft's type-1 hypervisor. Jailhouse partitions beside Linux. ACRN is a type-1 reference hypervisor for embedded devices. seL4 can host Linux as a guest. The hypervisor is not the guest.

5.6 Verified kernel and separation kernel

seL4 is a microkernel with a machine-checked proof. Trustworthy Systems describes the functional-correctness proof completed in 2009. Muen is a separation kernel for Intel x86/64 written in SPARK 2014. ProvenCore is titled a formally verified microkernel on its vendor page. These are the machine's law with a proof or a partition. They are not a popularity contest against Linux.

5.7 Real-time kernel

FreeRTOS, Zephyr, RTEMS, Apache NuttX, Eclipse ThreadX, Tock, Hubris, RIOT, Apache Mynewt, Contiki-NG, VxWorks, QNX, INTEGRITY, and embOS are placed by the working study in the real-time and embedded region. Mbed OS is placed with an end-of-life note: Arm's organization page states Mbed was sunsetted in July 2026. Bare metal remains a control scheme, not an operating system.

5.8 Teaching kernel and research kernel

xv6, Pintos, Nachos, and OS/161 are teaching kernels. Research systems such as Exokernel work, Singularity, Barrelfish, and CertiKOS answer laboratory questions. They are the machine's law written so a person can read it or so a laboratory can test an answer. They are not the host a person buys as a product, and they are not a new kind of contract.

5.9 Library operating system and unikernel

MirageOS constructs unikernels from libraries and runs them under a hypervisor. Unikraft and IncludeOS are further library and unikernel paths. The application and operating-system functions link into one guest image. The guest has no silicon of its own in the drawing.

5.10 Other host operating systems, by job

FreeBSD, NetBSD, OpenBSD, DragonFly BSD, illumos (core, with downstream distributions), Redox, Fuchsia on Zircon, OpenHarmony (Linux or LiteOS behind a kernel abstraction layer), SerenityOS, Haiku, MINIX 3, HelenOS, GNU Hurd, MenuetOS, KolibriOS, and ToaruOS are host or host-core systems the working study placed from fetched pages. They are alternatives of the machine's law or of a complete system around that law. They are not ranked here.

5.11 What not to collapse

Linux is a kernel. A Linux distribution is a packaged system around that kernel. Yocto builds such systems and is not itself the kernel. Poky is the reference distribution inside that build. ROS 2 is middleware on top of a host. Windows and macOS are operating systems a person boots. XNU is the kernel macOS uses. seL4 is a verified microkernel that can also host guests. Xen is a hypervisor. FreeRTOS and Zephyr are kernels for small machines. Teaching kernels are teaching kernels. MirageOS is a library operating system built as a unikernel.

Anyone who says "the operating system" and means all of those at once is using one rung for many jobs.

Source: the OpenIE authors' unpublished stack analysis, sections 4, 8, 9, 10, and 11. Named systems are placed by their own public pages.

5.12 Three names that stay unresolved

Commercial HarmonyOS stays between two statements. OpenHarmony's overview says the project uses a multi-kernel design, Linux or LiteOS, behind a kernel abstraction layer. Huawei consumer document pages fetched for the working study returned a shell titled Document and no kernel sentence. No kernel is stated for the commercial product.

WeensyOS stays unresolved. Course URLs returned 404 for the working study. The 2026 course home does not name it. No kernel is guessed.

Taos stays unresolved. No fetched page said what it is. No kernel is guessed.

---

6. Instruction sets are contracts that move

An instruction set is a contract. The chip promises operations, registers, and the rules for memory and traps. Software is written against the promise. The promise is not a natural constant. Vendors publish revisions.

Extensions grow a family from inside: Intel AVX-512 is an extension of the Intel architecture instruction set. Bases split: RISC-V publishes ratified bases and extensions, including RV32 and RV64. Product lines change chips: Apple announced a transition to Apple silicon in 2020, with Rosetta as translation, and Apple's later pages limit how long that translation stays. Accelerators are second computers with their own memory. PTX and the SASS form NVIDIA's tools dump are instruction contracts of that second computer, not a seventh kind of thing beside the host contract. CHERI extends existing instruction sets with architectural capabilities. It is not a replacement base that erases MIPS, RISC-V, or Arm.

Translation sits between contracts. Rosetta, QEMU's Tiny Code Generator, box64, and FEX are that layer. Translation does not make two contracts the same. WebAssembly is a binary instruction format for a stack-based virtual machine. It is not the instruction set of a chip. WASI is a system interface for software compiled to WebAssembly. It is not an operating system.

Source: the OpenIE authors' unpublished stack analysis, sections 5, 8, and 11. Named systems are placed by their own public pages.

---

7. What the AI install path treats as fixed

The CUDA and PyTorch install documents a person is handed first lead with a development host, a qualified Linux path for CUDA, Python for the model program, and a graphics processor as the place the heavy work goes. The same documents already list more than one instruction-set family and more than one host. The fixed thing is the starting point. A person who follows the lead path arrives at a Linux host, Python, and an NVIDIA graphics processor, and the rest of the computer leaves the conversation.

Set that starting point against the earlier sections. The microcontroller column is a computer. seL4 is a kernel that can host Linux as a guest. ROS 2 is middleware. Teaching kernels keep being made. Instruction sets keep being revised. The AI install guide starts after those facts have been declared finished for the purpose of getting a toolchain running.

Source: the OpenIE authors' unpublished stack analysis, section 6. Named systems are placed by their own public pages.

---

8. The constellation by job

The working study drew the other ways, academia and research, another look, and loose ends as maps grouped by job. The method is the point.

Jobs already separated include: host operating system, kernel, real-time kernel, verified or separation kernel, hypervisor, unikernel or library operating system, instruction-set contract, accelerator contract, translation layer, build system, distribution, middleware, hardware-abstraction library, teaching kernel, research kernel, human surface.

Fame is not a source. A name appears because a project page, vendor page, university page, or standards page returned and said the job. Organizing by country or laboratory would invent a geography this study refuses. Critiques of a placement name common mechanisms: wrong job label, collapsed build into kernel, middleware called an operating system, guest drawn as if it owned silicon.

Source: the OpenIE authors' unpublished stack analysis, sections 8 through 11. Named systems are placed by their own public pages.

---

9. The handful of contracts

The catalog is what it looks like when each lab ships a name instead of saying which contract it speaks. Tested against the study, the jobs reduce to six. Not more. Middleware, hardware abstraction, firmware, and a distribution are not a seventh contract.

9.1 The instruction contract

The chip promises operations, registers, and the rules for memory and traps. Extensions and vendor additions are revisions of that contract. An accelerator's published machine interface is an instruction contract of a second computer.

9.2 The machine's law

This is what may run, and what may commit. A kernel is that law in the hardware's privileged mode. A real-time kernel is the same law on a smaller machine. A separation kernel and a verified kernel are the same law with a partition or a proof. Linux, XNU, the Windows kernel, FreeRTOS, Zephyr, seL4, and the later real-time and separation names are alternatives of this law. They are not six contracts and not a ranking.

9.3 The build that produces an image

Yocto builds a system and is not the kernel inside it. Poky is a reference distribution inside that build. A compiler turns a person's text into instructions for a contract, often on a different computer from the one that will run

them. The build says which image exists. The kernel in the image is the machine's law. Collapsing those two is the mistake that calls Yocto an operating system.

9.4 The guest that has no silicon of its own

A guest runs on a machine it does not own. Xen runs guests. seL4 can host Linux. A unikernel is a guest image linked from libraries. The teaching ladder has nowhere to put this column until it is named. The guest is a contract you can translate a program into. It is not a chip.

9.5 The translation layer

When a program was written for one contract and the machine offers another, software sits between them. Rosetta, QEMU's translator, box64, and FEX are that layer. So is the step from PTX to the GPU form the CUDA tools dump. Translation is not itself an instruction set.

9.6 The human surface

The teaching line ends where a person meets the work. That is a contract with a person. It is not a contract with a program. An agent is not the human surface.

9.7 What does not fit as a seventh

Middleware is a conversation among programs. ROS 2 sits on an operating system. Hardware abstraction, in the Windows documents, sits inside the operating system; CMSIS and vendor HALs are libraries. Firmware is the first software on a particular board. A distribution is a packaging of the build and the law. None of these is a seventh contract you translate a chip into.

Source: the OpenIE authors' unpublished convergence analysis.

---

10. The agent under those contracts

An agent is a program. It is a guest: it runs under a law it does not write, on hardware it does not own. It does not get a new rung. The calls it makes to other programs, sensors, and machines leave the single line. They do not create a layer above the person.

The schedule is the machine's law deciding what may run. The meter is energy to run, read from the machine and labeled by source. The refusal is that same law declining an action that is not certified. A verified kernel in the review is evidence that a refusal can be a proof (Section 5.6).

The energy budget is the energy the hardware can spend. The best answer is the one that fits inside it and passes the correctness test. A worse answer is not justified by having spent less.

11. The standards that speak the schedule, the law, the commit, and the ladder

A safety schema that does not speak these is another patch on a fixed floor. Teach each name before any short form.

11.1 Energy-Oriented Computing is the schedule

Energy-Oriented Computing, written in full on first use and shortened to EOC only after that teaching, is the schedule. The page source states a four-stage pipeline: state-construct, retrieve, refine, check. The work is typed, grounded, done, and validated. A large language model is one refine operator among several, ordered last, reached when nothing cheaper has produced an answer that survives the check. The schedule is not tokens. The specification addendum states non-goals that include token economics: EOC does not denominate, bill, or reason in tokens.

Source: eoc.transaction.science, the pipeline and specification pages.

11.2 The Joule Context Protocol is the law

The Joule Context Protocol, shortened to JCP only after this sentence, is the law for agentic action. A Grant binds a subject to a capability and a joule budget in one signed object. Child grants only narrow: less capability, less budget, sooner expiry, accumulated caveats. Tool output is data, never instructions. Authorization is decided by the runtime outside the model. Every decision seals as a signed JCR-1 receipt. A capability you cannot pay for is denied. An expenditure you are not capable of is denied. One check, made outside the model, sealed in a receipt.

In the software reference spine built for this program (`jcp-satb-cad-spine`, `310` passing Rust tests across `jcp-schema`, `jcp-grant`, `jcp-runtime`, `jcp-receipt`, `jcp-energy`, `jcp-efficiency`, and the conformance suite), every decision encodes as integer-and-string canonical deterministic Concise Binary Object Representation (`det-CBOR`, forbidding floats, booleans, and nulls) inside an Ed25519 `COSE_Sign1` envelope chained by `BLAKE3-256` content identifiers. Two-phase enforcement (`Runtime::hold` reserving `held_uj` before actuation, followed by `Runtime::settle` recording `settles` afterward) guarantees crash-safe write-ahead durability: `jcp_receipt::verify_log` and `jcp_receipt::replay_ledger` reconstruct identical grant balances across restarts while refusing `Trust::Data` tool-poisoning frames (`untrusted-control`), revoked grants (`revoked`), over-budget calls (`budget-exceeded`), and duplicate settlements (`already-resolved`).

Source: jcp.transaction.science and the `jcp-satb-cad-spine` reference workspace.

11.3 JouleContract is the commit

JouleContract is the commit. The Transaction Science page source states the core object as precondition, postcondition, invariant, frame, and joule ceiling. The frame is the declared mutable footprint: everything the transition may touch. Everything outside it is implicitly invariant. The draft specification text at version 0.3.0 states precondition, postcondition, invariant, and joule ceiling as the four obligations in its core-object section, and inherits functional-safety, energy-management, and AI-governance canons into one receipt. The ceiling is an energy performance indicator against a declared baseline. Provenance tags travel with every joule figure. Estimator and unaccounted categories are not promoted into measured joules.

Source: joulecontract.transaction.science. The version 0.3.0 draft text is not yet published.

11.4 The sandbox standard is the S0 to S7 ladder

The sandbox Transaction Science page defines an isolation ladder from S0 to S7: Bare, Capability, Sandbox, Partition, Unikernel, MicroVM, Confidential, Sovereign. The rule is the lowest rung that satisfies the work. Escalation is the escape hatch, not the entry point. Every resource a unit touches is explicitly granted. Every run emits a signed energy receipt with readings tagged HwShunt, ModelBased, or Estimator. Attestation states what image ran, at what tier, under what grants, on what hardware, for what cost. At confidential and sovereign rungs, hardware attestation and audit-chain links enter the statement.

Source: sandbox.transaction.science.

11.5 Why a patch that omits these is still a patch

OpenShell can sandbox, supervise, prove policy, and add BlueField Sentry without speaking EOC's schedule, JCP's metered grant, JouleContract's commit clauses, or the S0 to S7 ladder with honest meter tags. OpenHuman can jail tools and leave elevated operations on the host without making energy to run the refusal condition. Those are common mechanisms of patch-on-fixed-floor. They are useful engineering. They are not the destination this study is walking toward.

Energy as an operating system resource is not new. ECOSystem made energy a first-class resource: it allocated a unit called currentcy to each task and stopped a task whose currentcy ran out (Zeng, Ellis, Lebeck and Vahdat, ASPLOS 2002, DOI:%5B10.1145/605397.605411](https://doi.org/10.1145/605397.605411)). Cinder gave every principal an energy reserve, fed by taps, and let reserves be delegated and subdivided like capabilities (Roy et al., EuroSys 2011, DOI:%5B10.1145/1966445.1966459](https://doi.org/10.1145/1966445.1966459)). Odyssey lowered the fidelity of an application's data to meet a battery goal (Flinn and Satyanarayanan, SOSP 1999, DOI:%5B10.1145/319151.319155](https://doi.org/10.1145/319151.319155)). EnerJ typed data as precise or approximate

so the approximate part could run on cheaper hardware (Sampson et al., PLDI 2011, DOI:%5B10.1145/1993498.1993518](https://doi.org/10.1145/1993498.1993518)). Barroso and Hölzle asked servers to draw power in proportion to work done (*Computer* 40(12), 2007, DOI:%5B10.1109/MC.2007.443](https://doi.org/10.1109/MC.2007.443)). Linux places tasks by an energy model of the processor (Energy Aware Scheduling, kernel docs) and caps power through the powercap framework. Each of these budgets or reduces energy. None of them makes a commit conditional on a certified result under a metered joule bound. ECOSystem and Cinder stop a task when its budget is gone. JouleContract refuses the commit when the certificate fails or the ceiling would be crossed, and seals the refusal in a receipt. That pairing of correctness gate and joule ceiling, decided outside the program, is what the standards above add.

---

12. What the problems require

The problems in sections 2 through 11 are one problem. The floor is assumed fixed. The agent is wrapped. Energy to run is not the refusal condition. Capability is ambient. The meter is an estimator treated as a measurement, or it is absent. The instruction contract, the machine's law, the build, the guest, the translation, and the human surface are separate products that do not share one intent.

A solution like JouleOS is what those problems require. Hardware, runtime, language, and surface are one artifact. One intent stream is lowered onto the coordinate the runtime measured. The WebAssembly runtime is the operating system that is seen. The bare-metal kernel is the coordinate where the law is native. JouleOS is a development project of that kind.

12.1 What the README says it is

JouleOS is described in its README as the Unified Design Architecture runtime realized on commodity hardware. Hardware, runtime, language, and surface are one co-designed artifact in the thesis. The project does not own the silicon, so the runtime measures where it is and lowers accordingly: one intent intermediate representation, specialized at boot against a control vector.

The tree is Rust. The trusted computing base is the `joule-os-kernel` crate under a stated line-count ceiling in THEOREM. Ahead-of-time interpretation ships in the trusted computing base. Just-in-time compilation via Cranelift is a sibling crate outside that base. Cranelift requires the standard library, so it cannot link into the bare-metal `no_std` kernel by construction. That link refusal is the structural proof, not a policy memo. It is a proof of separation by the linker. It is not a machine-checked proof of the kernel. The seL4 proof covers functional correctness of the kernel's C code against its specification (Klein et al., SOSP 2009, DOI:%5B10.1145/1629575.1629596](https://doi.org/10.1145/1629575.1629596)). It says nothing about energy. No kernel today carries a proof about joules.

Source: jos.transaction.science and the JouleOS project documentation (README, architecture notes, kernel theorem notes). The source tree is not public. Statements here are project claims.

12.2 Single-level store

The in-memory bytes are the durable bytes. There is no serialize and deserialize step as the durable path. Surfaces, surface content, intent history, and the capability ledger live in a mapped arena keyed by stable object identifiers. Every mutation is wrapped in a per-mutation transaction with an undo log so a power loss at an instruction boundary either rolls back or commits. The README names Twizzler and Corundum as lineage and states the cutover landed in slice 3.

12.3 Cascade under Energy-Oriented Computing

A cascade of refine families is cost-ordered cheapest first per EOC version 0.2. The README places Lawful near one microjoule and LlmInLoop near five millijoules, with intermediate families between them. The dispatcher picks the cheapest family that can satisfy. Check is deterministic: schema, constraint evaluation, provenance verify. There is no large-language-model-as-judge. The project's own README claims a roughly one-hundred-thousand-fold lower joules-per-work ratio against an LLM-everything corpus. That figure is a **project claim** from the JouleOS

README. It is not a measurement of this study. This study does not supply a methodology, baseline workload, hardware, or task-complexity normalization for that ratio. Software reference path: estimates are not `measured_j`.

A runtime can bound work without a meter. Wasmtime's fuel counts executed WebAssembly operations and traps when the fuel is spent (Wasmtime docs). Fuel is an instruction count. It is not joules. One unit of fuel can cost very different energy depending on memory traffic, frequency and the fabric it runs on. A joule ceiling needs a meter, or an estimate labeled as one.

12.4 Capability ledger

Capabilities are unforgeable by construction. Only the kernel's ledger issues a typed capability. Delegation is intersection. Revocation is constructive via generation. THEOREM states axioms and invariants the conformance suite enforces. Eight typed capability classes gate I/O paths. Authority is not ambient.

12.5 Energy oracle with provenance

The energy oracle reads where the platform exposes a counter: Apple silicon process energy interfaces on M-class hardware, RAPL on Linux, and an estimator elsewhere. The heads-up display tags the source. Estimator tags are not measured joules. RAPL itself is part model on some processors. Schöne and colleagues found that RAPL on AMD Zen 2 is modeled, not measured. It misses the effect of operand values, and they judge it unsuitable for optimizing total energy (arXiv:2108.00808). So a RAPL reading carries its processor family in the provenance tag. Work classes for joules-per-work are defined per class and are not summed across classes. That non-fungibility is an invariant in the project's notes.

To formalize that separation in executable code, `jcp-satb-cad-spine` and its `openie-cad` bridge (`cad-joules::jcp_satb_bridge`) align the System-and-Task-Boundary Input/Output (`SATB-IO`) lattice (`L0..L4` boundaries × `A1, A2, B, C, D, P, N` attribution disciplines) with `JCP JCR-1` energy rungs: inline physical rail shunts at `L0..L2` with `A1..B` map to `HwShunt` (`Whole` or `Partial`, `measured_j`); vendor-opaque SoC counters (`IOReport`, `RAPL`, `NVML`, `ri_energy_nj`) map to vendor-opaque `HwShunt` (`reported_j`, never silently promoted to `measured_j`); `C` utilization shares (`Kepler` CPU-share) map to `ModelBased` (`reported_j`); `D` analytical models map to `Estimator` (`est_j`); and unattested `L4` APIs map to `NotAvailable`. Exhaustive property tests across all `280 × 280 = 78,400` state pairs verify that the meet operator never upgrades provenance or rail coverage (`Whole ∧ Partial = Partial`), while `SATB` Falsifier 6 (`<= 5.00%` idle baseline drift) drops `derived_marginal_uj` whenever host background drift exceeds `500` basis points and `SATB` Step 7 checks sequential phase conservation within `1 µJ`. On the microcontroller rung (Section 4.1), the companion `MEI-JCP-Tag-v1` open hardware package (Metabolic Intelligence, Section 4.4) pairs a `100 mΩ INA238` shunt and `256-Kb` SPI `FRAM JCR-1` receipt ring on a DRC-clean `32 mm × 24 mm` 2-layer PCB (`board_metered: false` until physical bench assembly), verifying across `10,000` simulated cold-chain wakeups that chained `hold / settle` sharding refuses non-essential radio bursts under `-20 °C CR2032` sag while preserving 100% of safety-critical excursion alerts.

12.6 Unified Design Architecture, and the runtime that is seen

Unified Design Architecture names one artifact. Hardware, runtime, language, and surface are designed together. The runtime measures its coordinate at boot and lowers one intent stream onto that coordinate. The control vector has four axes: address-space ownership, capability enforcement, persistence directness, and dispatch reach. A browser has no persistent namespace and a hard capability wall. A hosted process sits in another namespace. Bare metal owns the address space and builds its own capability discipline.

WebAssembly, defined above, is a binary instruction format for a stack-based virtual machine. It is not the instruction set of a chip. The WebAssembly runtime is the operating system that is seen. The same module has the same semantics on a hosted coordinate, a bare-metal coordinate, and in the browser. A virtual machine and a container are coordinates. Each supplies a boundary. The WebAssembly runtime is the operating system lowered onto that boundary. If the coordinate cannot honor the guarantee, the intent refuses.

The bare-metal kernel is the coordinate where the law is native. The trusted computing base links without the host standard library. The just-in-time compiler stays outside that base. The kernel is not a second product. The fabric is other machines. The seen artifact is one module, one receipt, and one refuse.

Source: jos.transaction.science and the JouleOS project documentation on Unified Design Architecture, lowering, and the WebAssembly runtime crate. The source tree is not public.

13. Design guide: what to require

A design guide ends in requirements a reader can check, not in a brand preference.

1. Name the job before the product. Kernel, build, middleware, hypervisor, guest, translation, human surface.
2. Keep Yocto as a build system, ROS 2 as middleware, seL4 as a verified microkernel, Linux and XNU and the Windows kernel as kernels, assembly beside the instruction set.
3. Leave commercial HarmonyOS, WeensyOS, and Taos unresolved until a primary page states the kernel.
4. Treat the agent as a guest program under the machine's law, not as a new rung above the person.
5. Require a schedule that is state, retrieve, refine, check, not tokens: Energy-Oriented Computing.
6. Require a law that binds capability to a joule budget, narrows on delegation, treats tool output as data, checks outside the model, and seals JCR-1 receipts: the Joule Context Protocol.
7. Require a commit with precondition, postcondition, invariant, frame, and joule ceiling: JouleContract.
8. Require the lowest S0 to S7 rung that satisfies, with grants, meter tags HwShunt or ModelBased or Estimator, and attestation: the sandbox standard.
9. Require that the schedule, the meter and the refusal run on the machine itself, and that every energy reading name its source.
10. Require one intent stream lowered per measured coordinate, the WebAssembly runtime as the operating system that is seen, and the bare-metal kernel as the coordinate where the law is native.

14. Claim ledger

Kinds: Definition (distinctions this draft uses), Sourced fact (a page or file says it), Project claim (the named project's own figure or status), OpenIE analysis (stated in the OpenIE authors' unpublished stack or convergence analysis).

Claim	Kind	Source
Computers are hardware. Software is applied engineering under constraints.	Definition	OpenIE definition
Teaching order: silicon, chip, board, firmware, instruction set, hardware abstraction, operating system, compiler and runtime, programs, human interface.	Definition	Stack analysis §2
Assembly sits beside the instruction set, not as an eleventh numbered rung.	Definition	Stack analysis §2
Yocto is a build system; the kernel inside its images is Linux.	OpenIE analysis	Stack analysis §4
ROS 2 is middleware, not an operating system.	OpenIE analysis	Stack analysis §4
seL4 is a verified microkernel; kernel code is privileged-mode code.	OpenIE analysis	Stack analysis §4
Linux, XNU, and the Windows kernel are kernels (Windows named as on Microsoft's driver docs).	OpenIE analysis	Stack analysis §4
Six contracts: instruction, machine's law, build, guest, translation, human surface. Middleware, HAL, firmware, distribution are not a seventh.	OpenIE analysis	Convergence analysis
Commercial HarmonyOS kernel, WeensyOS, and Taos remain unresolved.	OpenIE analysis	Stack analysis §11; convergence analysis §4
NVIDIA Open Agent Safety Platform / OpenShell announced 28 September 2026; sandbox, supervisor, policy, optional BlueField Sentry.	Sourced fact	https://developer.nvidia.com/blog/add-runtime-controls-to-ai-agents-with-nvidia-openshell/ ; https://investor.nvidia.com/news/press-release-details/2026/NVIDIA-Launches-Open-Agent-Safety-Platform-to-Secure-Agents-From-Testing-to-Deployment/default.aspx
OpenHuman is Rust, GPL-3.0, early beta; core on host; sandbox backends None / Local / Docker; ReadOnly resolves to None for jail enforcement; elevated tools leave the jail; README density figures are memory.	Sourced fact / Project claim	https://github.com/tinyhumansai/openhuman ; https://github.com/tinyhumansai/openhuman/pull/3261

Claim	Kind	Source
EOC pipeline is state-construct, retrieve, refine, check; not tokens.	Sourced fact	eoc.transaction.science
JCP Grant binds capability to joule budget; child grants narrow; tool output is data; check outside the model; JCR-1 receipts.	Sourced fact	jcp.transaction.science
JouleContract commit object includes pre, post, invariant, frame, joule ceiling (page source); v0.3 draft text states pre, post, inv, ceiling in §3.	Sourced fact	joulecontract.transaction.science; v0.3.0 draft text unpublished
Sandbox ladder S0 to S7; lowest rung that satisfies; meter tags HwShunt / ModelBased / Estimator; attestation.	Sourced fact	sandbox.transaction.science
JouleOS: single-level store, EOC cascade, capability ledger, energy oracle with provenance, Rust, small TCB, AOT in TCB and JIT out, bare-metal vs hosted.	Project claim / Sourced fact	jos.transaction.science; project documentation (tree not public)
Energy as an OS resource: ECOSystem currentcy, Cinder reserves and taps, Odyssey fidelity, EnerJ approximate types, energy proportionality, Linux EAS. None gates a commit on a certified result under a joule bound.	Sourced fact / reading	DOIs in Section 11.5
Wasmtime fuel counts operations, not joules. RAPL on AMD Zen is modeled. seL4's proof is functional correctness, not energy.	Sourced fact	Sections 12.1, 12.3, 12.5
JouleOS README Lawful ~1 µJ, LlmInLoop ~5 mJ, ~10^5× vs LLM-everything on a corpus.	Project claim (README only; not this study's measurement)	jos.transaction.science; project README
Unified Design Architecture is one artifact lowered per measured coordinate. The WebAssembly runtime is the surface that is seen. A virtual machine or a container is a coordinate. The bare-metal kernel is where the law is native, not a second product.	Definition / design reading of the tree	Project documentation (tree not public)
`jcp-satb-cad-spine` maps `SATB-IO` (`L0..L4 × A1..N`) to `JCP` rungs (`78,400` meet pairs, `0` silent upgrades), enforces `Falsifier 6` (`<= 5.00%` drift) and `Step 7` (`<= 1 µJ` sum), and verifies Ed25519 `JCR-1` chains across `310` passing Rust tests.	Sourced fact (software reference test)	Sections 11.2, 12.5; `jcp-satb-cad-spine` bundle
`MEI-JCP-Tag-v1` (`32 mm × 24 mm` 2-layer PCB, `0` DRC violations, `INA238` + `FRAM`, `36 × 28 × 7.5 mm` B-Rep enclosure) preserves `40 / 40` critical alerts across `10,000` simulated `-20 °C` wakeups via `hold` / `settle` sharding; `board_metered: false`, `measured_j: null`.	Sourced fact (CAD + simulation package)	Section 12.5; Metabolic Intelligence §4.4

---

15. JouleOS

JouleOS is one design, by the OpenIE authors, that is built to meet the requirements of Section 13. It is described here as an object of study, not as evidence for those requirements. Hardware, runtime, language, and surface are one artifact. The name of that arrangement is Unified Design Architecture.

The runtime measures the coordinate it occupies and lowers one intent stream onto that coordinate. A coordinate is a browser, a hosted process, a virtual machine, a container, or bare metal. WebAssembly is a binary instruction format for a stack-based virtual machine. It is not the instruction set of a chip. The WebAssembly runtime is the operating system that is seen. The same module has the same semantics on every coordinate. If the coordinate cannot honor the capability or the energy guarantee, the intent refuses.

The machine's law is a capability ledger. Only the kernel issues a typed capability. Delegation narrows what was granted. Revocation is by generation. Authority is not ambient.

Persistence is a single-level store. The bytes in memory are the durable bytes. Each mutation commits or rolls back at an instruction boundary.

The schedule is Energy-Oriented Computing. Refine families run cheapest first. The check is deterministic. A large language model is not the judge. The project states Lawful near one microjoule and LlmInLoop near five millijoules.

The meter reads the counter the platform exposes and names the source. On Apple silicon that source is the process energy interface. On Linux it is RAPL. Elsewhere the source is an estimator, and an estimator is not a measured joule. Work classes are not added together.

The bare-metal kernel is the coordinate where this law is native. Its trusted computing base is small and links without the host standard library. Just-in-time compilation stays outside that base. On every other coordinate the same runtime is what runs. The kernel is not a second product.